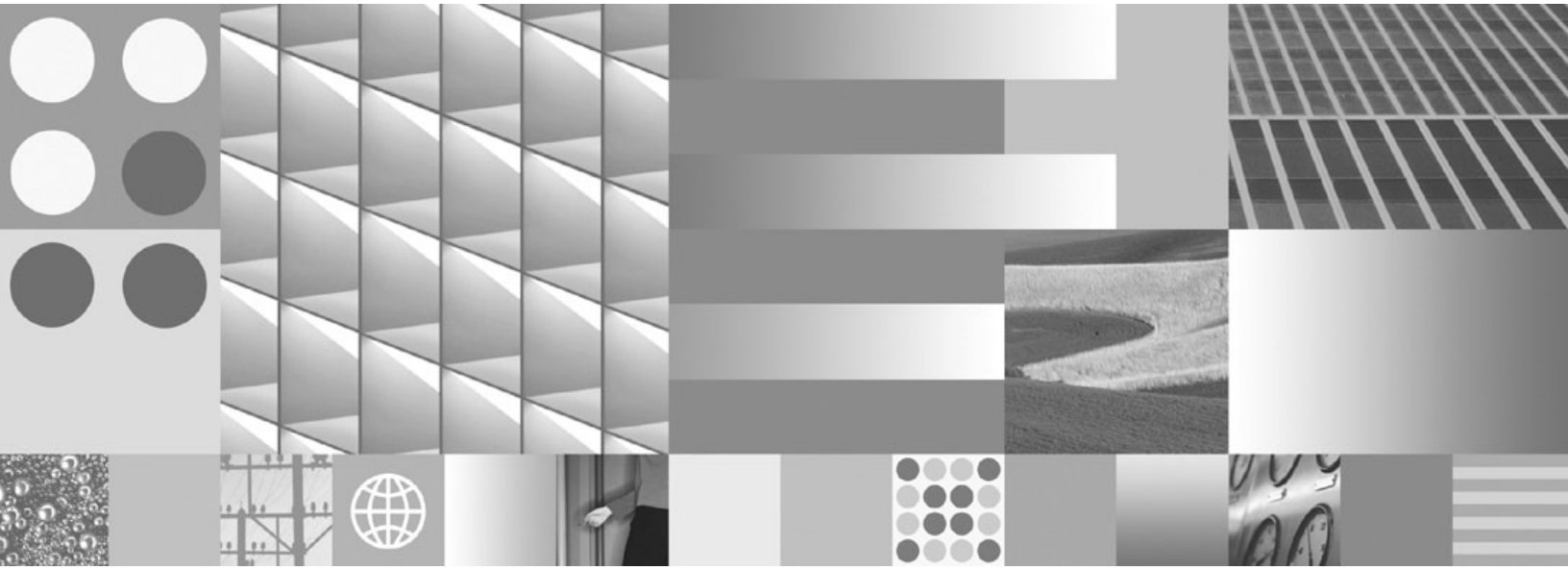
DB2 Version 9.5
für Linux, UNIX und Windows

Version 9 Release 5



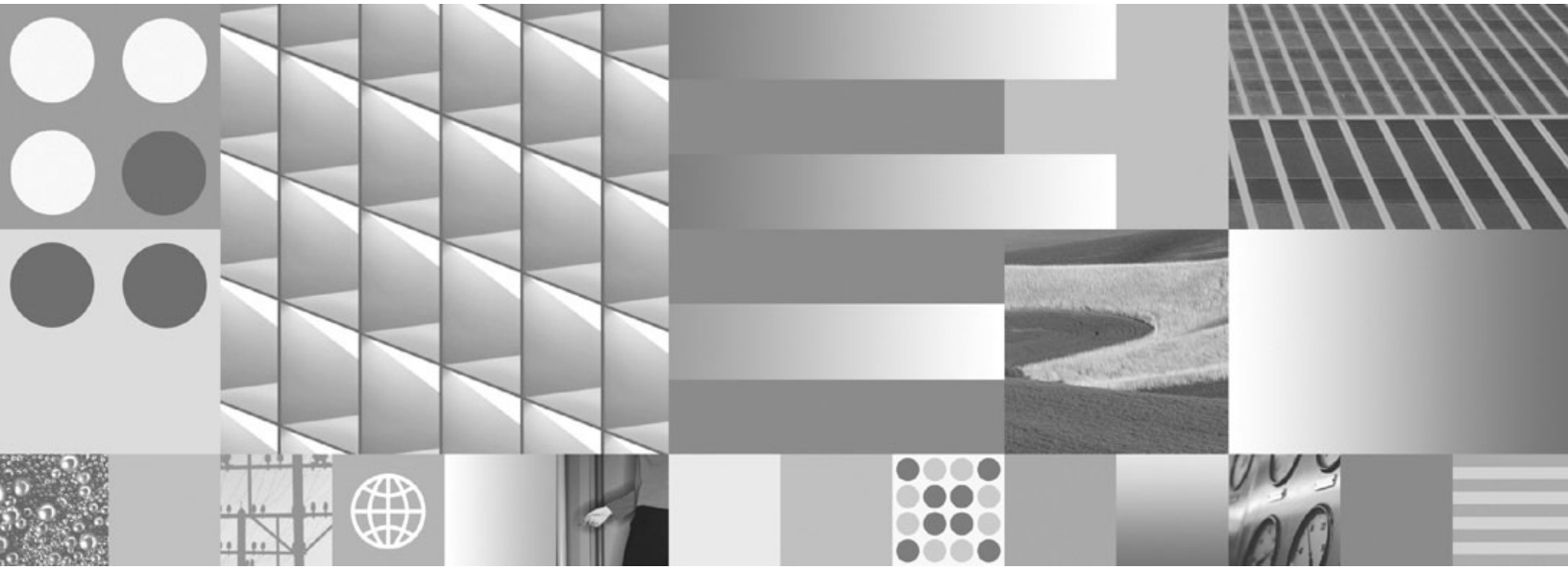
Partitionierung und Clustering

Aktualisierung: Dezember 2010

SC12-3915-03

DB2 Version 9.5
für Linux, UNIX und Windows

Version 9 Release 5



Partitionierung und Clustering

Aktualisierung: Dezember 2010

SC12-3915-03

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen in Anhang E, „Bemerkungen", auf Seite 483 gelesen werden.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM DB2 9.5 for Linux, UNIX, and Windows Version 9 Release 5, Partitioning and Clustering Guide,
IBM Form SC23-5860-03,
herausgegeben von International Business Machines Corporation, USA

© Copyright International Business Machines Corporation 1993, 2010
© Copyright IBM Deutschland GmbH 2010

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
Dezember 2010

Inhaltsverzeichnis

Zu diesem Handbuch ix
Zielgruppe ix
Aufbau des Handbuchs ix
Hervorhebungskonventionen xiv

Teil 1. Planung und Entwurf 1

Kapitel 1. Partitionierte Datenbanken und Tabellen 3
Einrichten von Umgebungen mit partitionierten Datenbanken 3
Datenbankpartitionierung in mehrere Datenbankpartitionen 4
Authentifizierungsaspekte bei partitionierten Datenbanken 5
Datenbankpartitionsgruppen 5
Aufbau von Datenbankpartitionsgruppen 7
Verteilungszuordnungen 8
Verteilungsschlüssel 9
Tabellenkollokation 11
Partitionskompatibilität 12
Partitionierte Tabellen 13
Tabellenpartitionierung 13
Datenpartitionen und Datenbereiche 15
Datenorganisationsschemata 16
Datenorganisationsschemata in DB2- und Informix-Datenbanken 21
Tabellenpartitionierungsschlüssel 27
Ladeaspekte für partitionierte Tabellen 29
Replizierte MQTs 32
Tabellenbereiche in Datenbankpartitionsgruppen 33
Tabellenpartitionierung und MDC-Tabellen . . . 33

Kapitel 2. Bereichsclustertabellen . . . 39
Vorteile einer Struktur mit Bereichsclustertabellen 39
Inkompatibilitäten von Bereichsclustertabellen . . . 40
Bereichsclustertabellen und Satzschlüsselwerte außerhalb des Bereichs 41
Sperren für Bereichsclustertabellen. 41

Kapitel 3. Tabellen mit mehrdimensionalem Clustering (MDC) 43
Tabellen mit mehrdimensionalem Clustering . . . 43
Vergleich zwischen regulären Tabellen und MDC-Tabellen 43
Auswählen von MDC-Tabellendimensionen . . . 45
Aspekte der Erstellung von MDC-Tabellen 56
Ladeaspekte für MDC-Tabellen 62
Protokollierungsaspekte für MDC-Tabellen . . . 63
Blockindexaspekte für MDC-Tabellen 63
Blockindizes für MDC-Tabellen 64
Szenario: MDC-Tabellen 67
Blockindizes und Abfrageleistung für MDC-Tabellen 70
Automatisches Beibehalten des Clustering bei INSERT-Operationen 74
Blockzuordnungen für MDC-Tabellen 76
Löschen aus MDC-Tabellen 78
Aktualisierungen an MDC-Tabellen 78
Tabellenpartitionierung und MDC-Tabellen 79

Kapitel 4. Parallele Datenbanksysteme 85
Parallelität 85
Umgebungen mit partitionierten Datenbanken. . . 89
Datenbankpartitions- und Prozessorumgebungen . . 90

Teil 2. Installationshinweise 99

Kapitel 5. Installationsvoraussetzungen 101
Installieren von DB2-Servern (Windows) 101
Vorbereiten der Umgebung für partitionierte DB2-Server (Windows) 103
Fast Communication Manager (Windows) . . . 105
Übersicht über die Installation Ihres DB2-Serverprodukts (Linux und UNIX) 106
DB2-Installationsmethoden 107
Installieren von DB2-Servern mit dem DB2-Installationsassistenten (Linux und UNIX) 110

Kapitel 6. Installationsvorbereitungen 115
Zusätzliche Tasks zur Installationsvorbereitung für eine Umgebung mit partitionierten Datenbanken (Linux und UNIX). 115
Aktualisieren der Umgebungseinstellungen für eine partitionierte DB2-Installation (AIX) . . . 115
Einrichten eines Arbeitsverbunds zum Verteilen von Befehlen an ESE-Workstations (AIX) . . . 117
Prüfen, ob NFS aktiv ist (Linux und UNIX) . . 118
Prüfen der Verfügbarkeit des Portbereichs auf zugehörigen Computern (Linux und UNIX) . . 119
Erstellen eines Dateisystems für ein partitioniertes Datenbanksystem (Linux) 120
Erstellen eines DB2-Ausgangsdateisystems für ein partitioniertes Datenbanksystem (AIX). . . 122
Erstellen erforderlicher Benutzer für die Installation eines DB2-Servers in einer Umgebung mit partitionierten Datenbanken (Linux). 124
Erstellen erforderlicher Benutzer für die Installation eines DB2-Servers in einer Umgebung mit partitionierten Datenbanken (AIX) 125

Kapitel 7. Installieren des DB2-Serverprodukts 129
Einrichten einer Umgebung mit partitionierten Datenbanken 129

Installieren von Datenbankpartitionsservern auf zugehörigen Computern mithilfe einer Antwortdatei (Windows) 131
Installieren von Datenbankpartitionsservern auf zugehörigen Computern mittels einer Antwortdatei (Linux und UNIX). 132

Kapitel 8. Installationsnachbereitung 135
Prüfen der Installation 135
Prüfen der Installation einer Umgebung mit partitionierten Datenbanken (Windows). 135
Prüfen der Installation eines partitionierten Datenbankservers (Linux und UNIX) 136

Teil 3. Implementierung und Pflege 137

Kapitel 9. Vorbereitung zum Erstellen einer Datenbank 139
Einrichten von Umgebungen mit partitionierten Datenbanken 139
Erstellen von Knotenkonfigurationsdateien . . . 140
Format der DB2-Knotenkonfigurationsdatei . . 142
Angeben der Liste von Systemen in einer Umgebung mit partitionierten Datenbanken . . . 149
Eliminieren doppelter Einträge aus einer Liste von Systemen in einer Umgebung mit partitionierten Datenbanken 150
Aktualisieren der Knotenkonfigurationsdatei (Linux und UNIX). 150
Einrichten mehrerer logischer Knoten 152
Konfigurieren mehrerer logischer Partitionen 152
Aktivieren der partitionsübergreifenden Abfrageparallelität 153
Aktivieren der partitionsinternen Parallelität für Abfragen 155
Verwaltung der Datenserverkapazität 156
Fast Communications Manager 157
Fast Communication Manager (Windows) . . . 157
Fast Communication Manager (Linux und UNIX) 157
Aktivieren der Kommunikation zwischen Datenbankpartitionen durch FCM-Kommunikation . . 158
Aktivieren der Kommunikation zwischen Datenbankpartitionsservern (Linux und UNIX) . . . 159

Kapitel 10. Erstellen und Verwalten von Umgebungen mit partitionierten Datenbanken 163
Erste Datenbankpartitionsgruppen 163
Erstellen von Datenbankpartitionsgruppen . . . 163
Tabellenbereiche in Datenbankpartitionsgruppen 164
Verwalten von Datenbankpartitionen 164
Hinzufügen von Datenbankpartitionen in Umgebungen mit partitionierten Datenbanken . . 166
Hinzufügen einer Datenbankpartition zu einem aktiven Datenbanksystem 167
Hinzufügen einer Datenbankpartition zu einem gestoppten Datenbanksystem (Windows) . . . 168
Hinzufügen einer Datenbankpartition zu einem gestoppten Datenbanksystem (UNIX) 169
Fehlerbehebung beim Hinzufügen von Datenbankpartitionen 171
Löschen von Datenbankpartitionen 172
Auflisten der Datenbankpartitionsserver einer Instanz 173
Hinzufügen von Datenbankpartitionsservern zu einer Instanz (Windows). 173
Hinzufügen von Datenbankpartitionen zu einer Instanz mithilfe des Assistenten 'Partitionen hinzufügen' 175
Ändern von Datenbankpartitionen (Windows) 175
Hinzufügen von Containern zu SMS-Tabellenbereichen in Datenbankpartitionen 177
Löschen einer Datenbankpartition aus einer Instanz (Windows) 177
Löschen von Datenbankpartitionen aus der Instanz mit dem Launchpad 'Partitionen löschen' . 178
Szenario: Partitionieren von Daten in einer Datenbank 179
Absetzen von Befehlen in Umgebungen mit partitionierten Datenbanken 181
Übersicht über die Befehle rah und db2_all . . . 182
Angeben der Befehle 'rah' und 'db2_all' . . . 182
Paralleles Ausführen von Befehlen (Linux, UNIX). 183
Erweiterung des Befehls 'rah' zur Verwendung von Baumstrukturlogik (AIX und Solaris) . . . 184
Befehle 'rah' und 'db2_all' 184
Präfixsequenzen für die Befehle 'rah' und 'db2_all' 185
Steuern des Befehls 'rah'. 187
Angeben der .-Dateien, die mit 'rah' ausgeführt werden (Linux und UNIX) 189
Beheben von Fehlern mit 'rah' (Linux, UNIX) 190
Überwachen von rah-Prozessen (Linux, UNIX) 191
Einstellen des Standardumgebungsprofils für 'rah' unter Windows 192

Kapitel 11. Erstellen von Tabellen und anderen zugehörigen Objekten 193
Tabellen in Umgebungen mit partitionierten Datenbanken 193
Verhalten großer Objekte in partitionierten Tabellen 194
Erstellen partitionierter Tabellen 195
Definieren von Bereichen für partitionierte Tabellen 196
Migrieren vorhandener Tabellen und Sichten in partitionierte Tabellen 200
Verhalten von partitionierten MQTs 202
Erstellen von Bereichsclustertabellen (RCT) . . . 206
Algorithmen für Bereichsclustertabellen . . . 206
Indizes für Bereichsclustertabellen 207
Unterschiede zu regulären Tabellen 207
Richtlinien zur Verwendung von Bereichsclustertabellen 208
Funktionsweise des SQL-Compilers bei Bereichsclustertabellen 208
Szenarios: Bereichsclustertabellen. 209
Aspekte der Erstellung von MDC-Tabellen . . . 211

Kapitel 12. Ändern einer Datenbank 217
Ändern einer Instanz 217
Ändern der Datenbankkonfiguration über mehrere Datenbankpartitionen 217
Ändern einer Datenbank 217
Ändern einer Datenbankpartitionsgruppe . . . 217
Verwalten von Datenbankpartitionen über die Steuerzentrale 217

Kapitel 13. Ändern von Tabellen und anderen zugehörigen Objekten 219
Ändern partitionierter Tabellen 219
Richtlinien und Einschränkungen für das Ändern partitionierter Tabellen 220
Zuordnen von Datenpartitionen 221
Richtlinien zum Hinzufügen von Datenpartitionen zu partitionierten Tabellen 223
Aufheben der Zuordnung von Datenpartitionen 226
Attribute von Datenpartitionen mit aufgehobener Zuordnung 228
Hinzufügen von Datenpartitionen zu partitionierten Tabellen 230
Löschen von Datenpartitionen 232
Szenario: Rotieren von Daten in einer partitionierten Tabelle 233
Szenarios: Rollin und Rollout von Daten partitionierter Tabellen 235

Kapitel 14. Laden von Daten 241
Parallelität und Laden 241
Aspekte des mehrdimensionalen Clustering . . . 241
Ladeaspekte für partitionierte Tabellen 242

Kapitel 15. Laden von Daten in einer Umgebung mit partitionierten Datenbanken 247
Übersicht zum Laden - Umgebungen mit partitionierten Datenbanken 247
Hinweise und Tipps für das Laden von Daten in einer Umgebung mit partitionierten Datenbanken . 249
Laden von Daten in einer Umgebung mit partitionierten Datenbanken 251
Überwachen einer Ladeoperation in einer Umgebung mit partitionierten Datenbanken mit LOAD QUERY 257
Fortsetzen, erneutes Starten oder Beenden von Ladeoperationen in einer Umgebung mit partitionierten Datenbanken 258
LOAD-Konfigurationsoptionen für Umgebungen mit partitionierten Datenbanken 261
LOAD-Sitzungen in einer Umgebung mit partitionierten Datenbanken - CLP-Beispiele 266
Migration und Versionskompatibilität 269

Kapitel 16. Migration von Umgebungen mit partitionierten Datenbanken . 271
Migrieren von partitionierten Datenbanken . . . 271

Kapitel 17. Verwenden von Momentaufnahme- und Ereignismonitoren . . 273
Überwachen der Reorganisation einer partitionierten Tabelle mit Snapshot Monitor-Daten 273
Globale Momentaufnahmen auf partitionierten Datenbanksystemen 280
Erstellen eines Ereignismonitors für partitionierte Datenbanken 281

Kapitel 18. Entwickeln einer geeigneten Backup- und Recovery-Strategie . 285
Recovery nach Systemabsturz 285
Recovery nach Transaktionsfehlern in einer Umgebung mit partitionierten Datenbanken 286
Recovery nach dem Ausfall eines Datenbankpartitionsservers 290
Erneutes Erstellen von partitionierten Datenbanken 290
Datenrecovery mit 'db2adutl' 291
Synchronisieren der Systemzeit in einer Umgebung mit partitionierten Datenbanken 302

Kapitel 19. Fehlerbehebung 305
Fehlerbehebung in DB2 305
Fehlerbehebung in Umgebungen mit partitionierten Datenbanken 305
Absetzen von Befehlen in Umgebungen mit partitionierten Datenbanken 305

Teil 4. Leistungsaspekte 307

Kapitel 20. Leistungsaspekte beim Datenbankentwurf 309
Leistungsverbessernde Funktionalität 309
Tabellenpartitionierung und MDC-Tabellen . . 309
Optimierungsstrategien für partitionierte Tabellen 314
Optimierungsstrategien für MDC-Tabellen. . . 319

Kapitel 21. Indizes 323
Indizes in partitionierten Tabellen 323
Indexverhalten bei partitionierten Tabellen . . 323
Clusterindexverhalten bei partitionierten Tabellen 327

Kapitel 22. Designadvisor 331
Verwenden des Designadvisors zur Migration von einer Datenbank mit einer einzelnen Partition auf eine Datenbank mit mehreren Partitionen 331

Kapitel 23. Verwalten des gemeinsamen Zugriffs 333
Sperrmodi für Tabellen- und Satz-ID-Indexsuchen für MDC-Tabellen 333
Sperren für Blockindexsuchen für MDC-Tabellen 337
Sperrverhalten für partitionierte Tabellen 340

Kapitel 24. Agentenverwaltung 345
Agenten in einer partitionierten Datenbank . . . 345

Kapitel 25. Optimieren von Zugriffsplänen 347
Indexzugriff und Clusterverhältnisse 347
Tabellen- und Indexverwaltung für MDC-Tabellen 348
Clustering 350
Optimierungsstrategien für partitionsinterne Parallelität 350
Beispiele für die Ausgabe von db2expln und dynexpln 352
Beispiel 1: keine Parallelität. 352
Beispiel 2: Zugriffsplan für Einzelpartition mit partitionsinterner Parallelität 354
Beispiel 3: Zugriffsplan für mehrere Partitionen mit partitionsübergreifender Parallelität . . . 356
Beispiel 4: Zugriffsplan für mehrere Partitionen mit partitionsübergreifender und partitionsinterner Parallelität 358
Joins 360
Auswirkung von Datenbankpartitionsgruppen auf die Abfrageoptimierung 361
Joinstrategien in partitionierten Datenbanken 362
Joinmethoden in Umgebungen mit partitionierten Datenbanken 363
Replizierte MQTs in Umgebungen mit partitionierten Datenbanken 369
Lerneinheit 4. Verbessern eines Zugriffsplans in einer Umgebung mit partitionierten Datenbanken . . 372
Verwenden von Zugriffsplandiagrammen . . . 372
Ausführen einer Abfrage ohne Indizes und ohne Statistikdaten in einer Umgebung mit partitionierten Datenbanken 372
Erfassen aktueller Statistikdaten für die Tabellen und Indizes mithilfe des Befehls 'runstats' in einer Umgebung mit partitionierten Datenbanken . 375
Erstellen von Indizes für Spalten, die verwendet werden, um Tabellen in einer Abfrage zu verknüpfen (Umgebung mit partitionierten Datenbanken) 378
Erstellen zusätzlicher Indizes für Tabellenspalten in einer Umgebung mit partitionierten Datenbanken 383

Kapitel 26. Umverteilung von Daten 387
Einschränkungen bei der Datenumverteilung. . . . 387
Feststellen der aktuellen Datenverteilung innerhalb einer Knotengruppe und für eine Tabelle 389
Umverteilen von Daten auf Datenbankpartitionen unter Verwendung des Befehls REDISTRIBUTE DATABASE PARTITION GROUP. 390
Umverteilen von Daten in einer Datenbankpartitionsgruppe 392
Protokollspeicherbedarf für die Datenumverteilung 392
Ereignisprotokolldatei für die Umverteilung . . . 394
Umverteilen von Datenbankpartitionsgruppen mithilfe der Prozedur STEPWISE_REDISTRIBUTE_DBPG 395

Kapitel 27. Konfigurieren des Speichers mit automatischer Leistungsoptimierung 399
Speicher mit automatischer Leistungsoptimierung in Umgebungen mit partitionierten Datenbanken . 399
Verwenden von Speicher mit automatischer Leistungsoptimierung in Umgebungen mit partitionierten Datenbanken 401

Kapitel 28. DB2-Konfigurationsparameter und -Variablen 403
Konfigurieren von Datenbanken über mehrere Partitionen 403
Variablen für Umgebungen mit partitionierten Datenbanken 404
Konfigurationsparameter für Umgebungen mit partitionierten Datenbanken 406
Kommunikation 406
Parallelverarbeitung 411

Teil 5. APIs, Befehle und SQL-Anweisungen zur Verwaltung 413

Kapitel 29. APIs zur Verwaltung . . . 415
sqleaddn - Datenbankpartitionsserver der Umgebung mit partitionierten Datenbanken hinzufügen . 415
sqlecran - Datenbank auf einem Datenbankpartitionsserver erstellen. 417
sqledpan - Datenbank auf einem Datenbankpartitionsserver löschen 418
sqledrpn - Überprüfen, ob ein Datenbankpartitionsserver gelöscht werden kann 419
sqlugrpn - Nummer des Datenbankpartitionsservers für eine Zeile abrufen 421

Kapitel 30. Befehle 425
REDISTRIBUTE DATABASE PARTITION GROUP 425
db2nchg - Konfiguration des Datenbankpartitionsservers ändern 436
db2ncrt - Datenbankpartitionsserver einer Instanz hinzufügen 437
db2ndrop - Datenbankpartitionsserver aus einer Instanz löschen 439

Kapitel 31. SQL-Sprachelemente . . . 441
Datentypen 441
Datenbankpartitionskompatible Datentypen . . 441
Sonderregister 443
CURRENT DBPARTITIONNUM 443

Kapitel 32. SQL-Funktionen 445
DATAPARTITIONNUM 445
DBPARTITIONNUM 446

Kapitel 33. SQL-Anweisungen 449
ALTER DATABASE PARTITION GROUP 449
CREATE DATABASE PARTITION GROUP . . . 452

Kapitel 34. Unterstützte SQL-Verwaltungsroutinen und Verwaltungssichten 455
Gespeicherte Prozedur ADMIN_CMD und zugeordnete SQL-Verwaltungsroutinen 455
Befehl GET STMM TUNING DBPARTITIONNUM mit Verwendung der Prozedur ADMIN_CMD 455
Befehl UPDATE STMM TUNING DBPARTITIONNUM mit Verwendung der Prozedur ADMIN_CMD 456
SQL-Verwaltungsroutinen und Verwaltungssichten für die Konfiguration 457
DB_PARTITIONS 457
SQL-Verwaltungsroutinen zur schrittweisen Datenumverteilung 458
STEPWISE_REDISTRIBUTE_DBPG (Prozedur) - Teil der Datenbankpartitionsgruppe umverteilen 458

Teil 6. Anhänge und Schlussteil 461

Anhang A. Installieren als Benutzer ohne Rootberechtigung 463
Installieren eines DB2-Produkts als Benutzer ohne Rootberechtigung 463

Anhang B. Verwenden von Backup 465
Verwenden von Backup 465

Anhang C. Katalogsichten für Umgebungen mit partitionierten Datenbanken 469
SYSCAT.BUFFERPOOLDBPARTITIONS 469
SYSCAT.DATAPARTITIONEXPRESSION 469
SYSCAT.DATAPARTITIONS 469
SYSCAT.DBPARTITIONGROUPDEF 471
SYSCAT.DBPARTITIONGROUPS 471
SYSCAT.PARTITIONMAPS 472

Anhang D. Übersicht über die technischen Informationen zu DB2 473
Bibliothek mit technischen Informationen zu DB2 im Hardcopy- oder PDF-Format 474
Bestellen gedruckter DB2-Bücher 476
Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor 477
Zugriff auf verschiedene Versionen der DB2-Informationszentrale 478
Anzeigen von Themen in der gewünschten Sprache in der DB2-Informationszentrale 478
Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten DB2-Informationszentrale . 479
DB2-Lernprogramme 481
Informationen zur Fehlerbehebung in DB2 . . . 481
Bedingungen 482

Anhang E. Bemerkungen 483

Index 487

Zu diesem Handbuch

Willkommen zum Handbuch 'Partitionierung und Clustering'!

Für die Funktionalität des DB2-Verwaltungssystems für relationale Datenbanken spielen die Partitionierungs- und Clusteringfunktionen eine bedeutende Rolle. Sie bieten Administratoren und Systembedienern die Möglichkeit, die Leistung der Datenbank effektiv zu verbessern und viele der Datenbankobjekte auf Hardwareressourcen zu verteilen. Ein schnellerer Datenabruf sowie die Möglichkeit, Objekte über ständig wachsende Hardwareressourcen verteilen und so von Parallelität und Speicherkapazität profitieren zu können, führen letztendlich zu einer höheren Produktivität. Dieses Handbuch stellt durch die spezielle Auswahl von Themen aus der DB2-Bibliothek eine zentrale Quelle für umfassende Informationen bereit, die ausschließlich den Bereich Planung, Entwurf, Implementierung, Verwendung und Pflege von Datenbankpartitionen, Tabellenpartitionen, Tabellenclustern, Tabellenbereichsclustern, MDC-Tabellen und Parallelität behandeln.

Zielgruppe

Dieses Handbuch ist in erster Linie für Datenbankadministratoren, Systemadministratoren, Sicherheitsadministratoren und Systembediener vorgesehen, die eine partitionierte Datenbank bzw. eine Datenbank mit Clustering für den Zugriff durch lokale und ferne Clients entwerfen, implementieren und/oder verwalten müssen. Es bietet sich auch für Anwendungsentwickler und andere Benutzer an, die eine umfassende Informationsquelle benötigen, um sich Kenntnisse über die Verwaltung und den Betrieb des DB2-Verwaltungssystems für relationale Datenbanken in Bezug auf die Partitionierungs-, Clustering- und Parallelitätsfunktionalität anzueignen. Für alle, die eine zukünftige Implementierung einer beliebigen oder auch aller erläuterten Hauptfunktionen in Betracht ziehen, stellt dieses Handbuch eine hervorragende Informationsressource dar.

Aufbau des Handbuchs

Diese Auswahl von Themen aus der DB2-Bibliothek stellt eine zentrale Quelle für umfassende Informationen bereit, die ausschließlich den Bereich der Partitionierungs-, Clustering- und Parallelitätsfunktionen von DB2 behandeln. Dieses Handbuch ist aus Gründen des Nutzungskomforts und der Effizienz in sechs Hauptteile untergliedert, deren erste fünf die primären Verwaltungsthemen darlegen, die für Administratoren, Systembediener und Anwendungsentwickler von Interesse sind. Ein Thema innerhalb eines Hauptteiles dieses Handbuchs kann Themen zugeordnet werden, die den Inhalt anderer Handbücher in der DB2-Bibliothek darstellen. Dies erleichtert die Arbeit mit Querverweisen auf allgemeinere Informationen, die eine Vielzahl anderer DB2-Funktionen und -Objekte behandeln. Wenn Sie zum Beispiel ein Thema in Teil 4, Kapitel 20, zu Optimierungsstrategien für MDC-Tabellen und den durch sie erzielbaren Leistungsverbesserungen gelesen haben, möchten Sie sich vielleicht über weitere allgemeine Leistungsverbesserungen informieren, die für reguläre Tabellen konfiguriert werden können, indem Sie die Informationen des Handbuchs *Optimieren der Datenbankleistung* lesen, die diesem bestimmten Beispielthema zugeordnet sind. In der nachfolgenden Tabelle 1 finden Sie die Hauptteile dieses Handbuchs und ihre Querverweise auf andere Handbücher, die zusätzliche Informationen zu anderen DB2-Objekten und -Funktionen im Hinblick auf ein ähnliches Thema enthalten.

Tabelle 1. Zuordnung der Teile dieses Handbuchs zu anderen Handbüchern in der DB2-Bibliothek

Teile im Handbuch 'Partitionierung und Clustering'	**Zuordnung zu Handbüchern in der DB2-Bibliothek**
Teil 1. Planung und Entwurf	*Datenserver, Datenbanken und Datenbankobjekte* *Datenbanksicherheit*
Teil 2. Installationshinweise	*Datenserver, Datenbanken und Datenbankobjekte* *DB2-Server - Einstieg*
Teil 3. Implementierung und Pflege	*Dienstprogramme für das Versetzen von Daten - Handbuch und Referenz* *Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz* *Datenserver, Datenbanken und Datenbankobjekte* *Migration* *Systemmonitor - Handbuch und Referenz* *Fehlerbehebung* *Lernprogramm für Visual Explain* *XQuery - Referenz*
Teil 4. Leistungsaspekte	*Datenserver, Datenbanken und Datenbankobjekte* *Optimieren der Datenbankleistung* *Lernprogramm für Visual Explain*
Teil 5. APIs, Befehle und SQL-Anweisungen zur Verwaltung	*Administrative API Reference* *Administrative Routines and Views* *Command Reference* *Developing ADO.NET and OLE DB Applications* *Developing Embedded SQL Applications* *Developing Java Applications* *Developing Perl and PHP Applications* *Developing User-defined Routines (SQL and External)* *Getting Started with Database Application Development* *SQL Reference, Volume 1* *SQL Reference, Volume 2*
Teil 6. Anhänge	*Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz* *DB2-Server - Einstieg* *SQL Reference, Volume 1*

Die folgenden übergeordneten Themenbereiche werden in den Kapiteln dieses Handbuchs behandelt:

Teil 1. Planung und Entwurf

Alle folgenden Kapitel enthalten Informationen zu Konzepten, die für die Planung und den Entwurf von Datenbanken bzw. Tabellen relevant sind, die entweder partitioniert, mit Clustering genutzt oder in parallelen Datenbanksystemen verwendet werden sollen.

- Kapitel 1, „Partitionierte Datenbanken und Tabellen", enthält eine Einführung in relevante Konzepte in Bezug auf die Funktionsmerkmale und Vorteile der Partitionierung von Datenbanken und Tabellen.
- Kapitel 2, „Bereichsclustertabellen", enthält allgemeine Informationen zu den Funktionsmerkmalen und Vorteilen der Verwendung von Bereichsclustertabellen.
- Kapitel 3, „Tabellen mit mehrdimensionalem Clustering (MDC)", enthält eine Beschreibung der Verwendung des mehrdimensionalen Clusterings als elegante Methode für das Clustering von Daten in Tabellen.
- Kapitel 4, „Parallele Datenbanksysteme", erläutert, wie sich Parallelität nutzen lässt, um eine beträchtliche Leistungsverbesserung zu erzielen.

Teil 2. Installationshinweise

Die folgenden Kapitel enthalten Informationen zu den Tasks der Installationsvorbereitung sowie der eigentlichen Installation, die zur Vorbereitung der Datenbankpartitionierung erforderlich sind.

- Kapitel 5, „Installationsvoraussetzungen", beschreibt die Voraussetzungen und Einschränkungen im Hinblick auf die Vorbereitung eines DB2-Servers, der in eine Umgebung mit partitionierten Datenbanken integriert werden soll.
- Kapitel 6, „Installationsvorbereitung", behandelt zusätzliche Tasks zur Installationsvorbereitung bei Verwendung von UNIX®- und Linux®-Systemen.
- Kapitel 7, „Installieren des DB2-Serverprodukts", beschreibt die Installation von Datenbankpartitionsservern und die Konfiguration einer Umgebung mit partitionierten Datenbanken.
- Kapitel 8, „Installationsnachbereitung", beschreibt, wie die Installation auf Windows®-, UNIX- und Linux-Systemen geprüft wird.

Teil 3. Implementierung und Pflege

Nach Abschluss der Schritte zur Planung, zum Entwurf und zur Installation wird in den folgenden Kapiteln erläutert, wie die Funktionen bzw. Objekte, für die zuvor Vorbereitungen getroffen wurden, implementiert und gepflegt werden.

- Kapitel 9, „Vorbereitung zum Erstellen einer Datenbank", beschreibt die Gesichtspunkte, die vor der Erstellung einer Datenbank zu beachten sind, wie zum Beispiel die Aktivierung der Parallelität, die Erstellung von Umgebungen mit partitionierten Datenbanken, die Erstellung und Konfiguration von Knoten bzw. Partitionen sowie die Einrichtung der Kommunikation zwischen Datenbankpartitionen.
- Kapitel 10, „Erstellen und Verwalten von Umgebungen mit partitionierten Datenbanken", beschreibt die Erstellung und Verwaltung von Datenbankpartitionen und Partitionsgruppen.
- Kapitel 11, „Erstellen von Tabellen und anderen zugehörigen Tabellenobjekten", enthält Informationen zur Erstellung und Einrichtung von partitionierten Tabellen, Bereichsclustertabellen (RCT) und MDC-Tabellen.

- Kapitel 12, „Ändern einer Datenbank", beschreibt, wie eine Instanz bzw. eine Datenbank geändert wird.
- Kapitel 13, „Ändern von Tabellen und anderen zugehörigen Tabellenobjekten", enthält Informationen zur Änderung von partitionierten Tabellen.
- Kapitel 14, „Laden von Daten", behandelt Aspekte für das Laden von Daten im Hinblick auf Parallelität, mehrdimensionales Clustering und partitionierte Tabellen.
- Kapitel 15, „Laden von Daten in einer Umgebung mit partitionierten Datenbanken", beschreibt, wie Datenladeoperationen in Umgebungen mit partitionierten Datenbanken gestartet, fortgesetzt, erneut gestartet oder beendet werden.
- Kapitel 16, „Migration von Umgebungen mit partitionierten Datenbanken", enthält eine kurze Übersicht über die Migration partitionierter Datenbanken und Querverweise auf detailliertere Informationen.
- Kapitel 17, „Verwenden von Momentaufnahme- und Ereignismonitoren", enthält relevante Informationen zur Verwendung von Snapshot Monitor-Ergebnissen zur Überwachung einer Tabellenreorganisation und zur Bewertung des globalen Status eines partitionierten Datenbanksystems. Darüber hinaus wird die Verwendung der Anweisung CREATE EVENT MONITOR erläutert.
- Kapitel 18, „Entwickeln einer geeigneten Backup- und Recovery-Strategie", beschreibt die Konzepte, die hinter einer Recovery nach Systemabsturz in einer Umgebung mit partitionierten Datenbanken stehen. Diese Informationen sind bei der Entwicklung von Backup- und Recovery-Strategien hilfreich, bevor ein Fehler auftritt.
- Kapitel 19, „Fehlerbehebung", enthält eine kurze Übersicht über die Fehlerbehebung sowie nützliche Informationen dazu, wie Befehle, die bei der Fehlerbehebung hilfreich sind, wie zum Beispiel db2trc, auf allen Computern in der Instanz bzw. auf allen Datenbankpartitionsservern ausgeführt werden.

Teil 4. Leistungsaspekte

Die folgenden Kapitel enthalten relevante Informationen, mit deren Hilfe Sie die Leistung Ihrer Umgebung mit partitionierten Datenbanken bzw. Ihrer Clusterumgebung verbessern können.

- Kapitel 20, „Leistungsaspekte beim Datenbankentwurf", beschreibt die Funktionsmerkmale der Tabellenpartitionierung und des mehrdimensionalen Clustering (MDC) hinsichtlich der Leistungsverbesserung und erläutert jeweils entsprechende Optimierungsstrategien.
- Kapitel 21, „Indizes", enthält Informationen zu Konzepten, die für das Verständnis von Indizes für partitionierte Tabellen hilfreich sind.
- Kapitel 22, „Designadvisor", beschreibt die Verwendung des Designadvisors zum Abrufen von Informationen zur Migration von einer Einzelpartitionsdatenbank auf eine Mehrpartitionsdatenbank sowie zum Abrufen von Empfehlungen zur Verteilung der Daten und zur Erstellung neuer Indizes, MQTs und MDC-Tabellen.
- Kapitel 23, „Verwalten des gemeinsamen Zugriffs", enthält Informationen zu Sperrmodi.
- Kapitel 24, „Agentenverwaltung", beschreibt die Optimierung von Datenbankagenten, die zur Verarbeitung von Anwendungsanforderungen verwendet werden.

- Kapitel 25, „Optimieren von Zugriffsplänen", beschreibt, wie ein Zugriffsplan verbessert wird, erläutert, wie das Optimierungsprogramm Informationen aus verschiedenen Suchen zur Optimierung von Datenzugriffsstrategien nutzt, und enthält Informationen zu Joinstrategien. Alle diese Optimierungsmethoden dienen der Verbesserung der Leistung für Umgebungen mit partitionierten Datenbanken, für Clustertabellen sowie für Systeme, auf denen Parallelität genutzt wird.
- Kapitel 26, „Datenumverteilung", enthält Informationen, die Ihnen bei der Feststellung helfen, ob eine Datenumverteilung durchgeführt werden sollte, und beschreibt, wie Daten auf Datenbankpartitionen umverteilt werden, wenn dies erforderlich ist.
- Kapitel 27, „Konfigurieren des Speichers mit automatischer Leistungsoptimierung", behandelt die Verwendung der Funktion für die automatische Speicherleistungsoptimierung in einer Umgebung mit partitionierten Datenbanken und gibt Empfehlungen zur Konfiguration.
- Kapitel 28, „DB2-Konfigurationsparameter und -Variablen", enthält Informationen dazu, wie Datenbankkonfigurationsparameter und Umgebungsvariablen über mehrere Partitionen hinweg definiert werden, und listet die Parameter und Variablen auf, die für die Funktionalität von Umgebungen mit partitionierten Datenbanken und der Parallelität relevant sind.

Teil 5. APIs, Befehle und SQL-Anweisungen zur Verwaltung

In den folgenden Kapiteln werden Informationen zu APIs, Befehlen und SQL-Elementen für die Verwaltung zusammengefasst, die für Umgebungen mit partitionierten Datenbanken relevant sind.

- Kapitel 29, „APIs zur Verwaltung", enthält Informationen zu den APIs, die nur für Umgebungen mit partitionierten Datenbanken relevant sind.
- Kapitel 30, „Befehle", enthält Informationen zu den Befehlen, die nur für Umgebungen mit partitionierten Datenbanken relevant sind.
- Kapitel 31, „SQL-Sprachelemente", beschreibt datenbankpartitionskompatible Datentypen und Sonderregister.
- Kapitel 32, „SQL-Funktionen", beschreibt SQL-Funktionen, die nur für Umgebungen mit partitionierten Datenbanken relevant sind.
- Kapitel 33, „SQL-Anweisungen", beschreibt SQL-Anweisungen, die nur für Umgebungen mit partitionierten Datenbanken relevant sind.
- Kapitel 34, „Unterstützte SQL-Verwaltungsroutinen und Verwaltungssichten", beschreibt SQL-Routinen und Sichten, die nur für Umgebungen mit partitionierten Datenbanken relevant sind.

Teil 6. Anhänge

- Anhang A, „Installieren als Benutzer ohne Rootberechtigung", beschreibt die Installation des DB2-Produkts durch einen Benutzer, der keine Rootberechtigung hat, auf UNIX- und Linux-Systemen.
- Anhang B, „Verwenden von Backup", beschreibt die Verwendungsweise des Befehls BACKUP DATABASE.
- Anhang C, „Katalogsichten für Umgebungen mit partitionierten Datenbanken", listet die Katalogsichten auf, die speziell für eine Umgebung mit partitionierten Datenbanken relevant sind.

Hervorhebungskonventionen

In diesem Handbuch werden die folgenden Hervorhebungskonventionen verwendet.

Fett	Kennzeichnet Befehle, Schlüsselwörter und andere Elemente, deren Namen vom System vordefiniert werden.
Kursiv	Kennzeichnet folgende Elemente: • Namen oder Werte (Variablen), die vom Benutzer angegeben werden müssen • Begriffe, die allgemein hervorgehoben werden sollen • Neue Begriffe, die eingeführt werden • Verweise auf andere Informationsquellen
`Monospace`	Kennzeichnet folgende Elemente: • Dateien und Verzeichnisse • Informationen, die Sie laut Anweisung in eine Eingabeaufforderung oder ein Fenster eingeben sollen • Beispiele für bestimmte Datenwerte • Beispiele für Text, der einer Ausgabe des Systems ähnlich ist • Beispiele für Systemnachrichten • Muster für Programmcode

Teil 1. Planung und Entwurf

Kapitel 1. Partitionierte Datenbanken und Tabellen

Einrichten von Umgebungen mit partitionierten Datenbanken

Die Entscheidung, eine Mehrpartitionsdatenbank zu erstellen, muss vor der Erstellung der betreffenden Datenbank getroffen werden. Im Rahmen Ihrer Entscheidungen beim Datenbankentwurf müssen Sie festlegen, ob die Leistungsverbesserungen genutzt werden sollen, die eine Datenbankpartitionierung zu bieten hat.

Auch in einer Umgebung mit partitionierten Datenbanken verwenden Sie den Befehl CREATE DATABASE oder die Funktion 'sqlecrea()' zum Erstellen einer Datenbank. Unabhängig von der verwendeten Methode kann die Anforderung über eine beliebige der Partitionen erfolgen, die in der Datei `db2nodes.cfg` aufgelistet sind.

Vor der Erstellung einer Mehrpartitionsdatenbank müssen Sie die Datenbankpartition auswählen, die als Katalogknoten für die Datenbank fungieren soll. Anschließend können Sie die Datenbank direkt von dieser Datenbankpartition oder von einem fernen Client aus erstellen, der mit dieser Datenbankpartition verbunden ist. Die Datenbankpartition, zu der Sie die Verbindung (mit ATTACH) herstellen, um den Befehl CREATE DATABASE auszuführen, wird zur *Katalogpartition* für diese spezielle Datenbank.

Die Katalogpartition ist die Datenbankpartition, in der alle Systemkatalogtabellen gespeichert werden. Jeglicher Zugriff auf die Systemtabellen muss über diese Datenbankpartition erfolgen. Alle Objekte föderierter Datenbanken (z. B. Wrapper, Server und Kurznamen) werden in den Systemkatalogtabellen in dieser Datenbankpartition gespeichert.

Wenn möglich, sollten Sie jede Datenbank in einer getrennten Instanz erstellen. Falls dies nicht möglich ist (d. h., Sie müssen mehr als eine Datenbank pro Instanz erstellen), sollten Sie die Katalogpartitionen auf die verfügbaren Datenbankpartitionen verteilen. Dadurch verringern sich Konkurrenzsituationen beim Abrufen von Katalogdaten in einer einzelnen Datenbankpartition.

Anmerkung: Sie sollten regelmäßig ein Backup der Katalogpartition durchführen und nach Möglichkeit vermeiden, Benutzerdaten in ihr zu speichern, da diese Daten die für das Backup benötigte Zeit verlängern.

Wenn Sie eine Datenbank erstellen, wird sie automatisch in allen Datenbankpartitionen erstellt, die in der Datei `db2nodes.cfg` definiert sind.

Wenn die erste Datenbank im System erstellt wird, wird ein Systemdatenbankverzeichnis gebildet. An dieses werden Informationen zu anderen Datenbanken, die Sie erstellen, angehängt. Wenn Sie mit UNIX arbeiten, heißt das Systemdatenbankverzeichnis `sqldbdir` und befindet sich im Verzeichnis `sqllib` unter Ihrem Ausgangsverzeichnis bzw. unter dem Verzeichnis, in dem die DB2-Datenbank installiert wurde. Dieses Verzeichnis muss sich unter UNIX in einem gemeinsam genutzten Dateisystem (z. B. NFS auf UNIX-Plattformen) befinden, weil es nur ein Systemdatenbankverzeichnis für alle Datenbankpartitionen gibt, die zu einer Umgebung mit partitionierten Datenbanken gehören. Unter Windows befindet sich das Systemdatenbankverzeichnis im Instanzverzeichnis.

Ebenfalls im Verzeichnis sqldbdir befindet sich die Systemintentionsdatei. Sie hat den Namen sqldbins und stellt sicher, dass die Datenbankpartitionen synchronisiert bleiben. Diese Datei muss ebenfalls in einem gemeinsam genutzten Dateisystem gespeichert sein, da es innerhalb aller Datenbankpartitionen nur ein Verzeichnis gibt. Die Datei wird von allen Datenbankpartitionen, die die Datenbank bilden, gemeinsam genutzt.

Zur Nutzung der Datenbankpartitionierung müssen Konfigurationsparameter geändert werden. Mithilfe der Befehle GET DATABASE CONFIGURATION und GET DATABASE MANAGER CONFIGURATION können Sie die Werte für einzelne Einträge in einer bestimmten Datenbank oder in der Konfigurationsdatei des Datenbankmanagers ermitteln. Zur Änderung einzelner Einträge in einer bestimmten Datenbank oder in der Konfigurationsdatei des Datenbankmanagers werden die Befehle UPDATE DATABASE CONFIGURATION bzw. UPDATE DATABASE MANAGER CONFIGURATION verwendet.

Zu den Konfigurationsparametern des Datenbankmanagers, die sich auf eine Umgebung mit partitionierten Datenbanken auswirken, gehören **conn_elapse**, **fcm_num_buffers**, **fcm_num_channels**, **max_connretries**, **max_coordagents**, **max_time_diff**, **num_poolagents** und **stop_start_time**.

Datenbankpartitionierung in mehrere Datenbankpartitionen

Der Datenbankmanager ermöglicht große Flexibilität bei der Verteilung von Daten über mehrere Datenbankpartitionen (Knoten) einer partitionierten Datenbank. Benutzer können wählen, wie die Daten zu verteilen sind, indem sie Verteilungsschlüssel deklarieren, und sie können bestimmen, auf welche und auf wie viele Datenbankpartitionen ihre Daten verteilt werden können, indem sie die Datenbankpartitionsgruppe und den Tabellenbereich auswählen, in denen die Daten gespeichert werden sollen.

Darüber hinaus gibt eine Verteilungszuordnung (die aktualisierbar ist) die Zuordnung zwischen Verteilungsschlüsselwerten und Datenbankpartitionen an. Dies ermöglicht eine flexible Parallelverarbeitung von Auslastungen in einer partitionierten Datenbank für große Tabellen, während gleichzeitig kleinere Tabellen in einer oder nur einer kleinen Anzahl von Datenbankpartitionen gespeichert werden können, wenn der Anwendungsentwickler dies so vorsieht. Jede lokale Datenbankpartition kann lokale Indizes für die in ihr gespeicherten Daten haben, um einen hochleistungsfähigen lokalen Datenzugriff bereitzustellen.

In einer partitionierten Datenbank dient der Verteilungsschlüssel zur Verteilung der Tabellendaten auf eine Gruppe von Datenbankpartitionen. Indexdaten werden ebenfalls zusammen mit den zugehörigen Tabellen partitioniert und lokal in den einzelnen Datenbankpartitionen gespeichert.

Bevor Datenbankpartitionen zum Speichern von Daten genutzt werden können, müssen Sie für den Datenbankmanager definiert werden. Die Datenbankpartitionen werden in einer Datei mit dem Namen db2nodes.cfg definiert.

Der Verteilungsschlüssel für eine Tabelle in einem Tabellenbereich in einer Datenbankpartitionsgruppe einer partitionierten Datenbank wird in der Anweisung CREATE TABLE oder der Anweisung ALTER TABLE angegeben. Wenn kein Verteilungsschlüssel angegeben wird, wird standardmäßig ein Verteilungsschlüssel für die Tabelle aus der ersten Spalte des Primärschlüssels erstellt. Wenn kein Primärschlüssel definiert ist, wird als Standardverteilungsschlüssel die erste in der Tabelle definierte Spalte verwendet, die einen anderen Datentyp als einen LONG- oder

LOB-Datentyp besitzt. Tabellen in partitionierten Datenbanken müssen mindestens eine Spalte enthalten, die weder ein LONG-Datentyp noch ein LOB-Datentyp ist. Eine Tabelle in einem Tabellenbereich, der sich in einer Datenbankpartitionsgruppe einer Einzelpartitionsdatenbank befindet, erhält nur dann einen Verteilungsschlüssel, wenn dieser explizit angegeben wird.

Zeilen werden in eine Datenbankpartition nach folgendem Verfahren eingefügt:

1. Ein Hashalgorithmus (d. h. eine Datenbankpartitionierungsfunktion) wird auf alle Spalten des Verteilungsschlüssels angewendet und generiert einen Indexwert für die Verteilungszuordnung.
2. Die Datenbankpartitionsnummer an diesem Indexwert in der Verteilungszuordnung gibt die Datenbankpartition an, in der die Zeile zu speichern ist.

Der Datenbankmanager unterstützt eine *Teilentclusterung*. Dies bedeutet, dass eine Tabelle auf eine Untermenge der Datenbankpartitionen im System (d. h. in einer Datenbankpartitionsgruppe) verteilt werden kann. Tabellen brauchen nicht auf alle Datenbankpartitionen im System verteilt zu werden.

Der Datenbankmanager besitzt die Fähigkeit zu erkennen, wenn sich Daten, auf die für einen Join oder eine Unterabfrage zugegriffen wird, in derselben Datenbankpartition in derselben Datenbankpartitionsgruppe befinden. Diese Situation wird als *Tabellenkollokation* bezeichnet. Zeilen in kollokierten (zusammengefassten) Tabellen mit denselben Verteilungsschlüsselwerten befinden sich in derselben Datenbankpartition. Der Datenbankmanager kann entscheiden, die Join- oder Unterabfrageverarbeitung in der Datenbankpartition auszuführen, in der die Daten gespeichert sind. Dies kann erhebliche Leistungsvorteile mit sich bringen.

Für kollokierte Tabellen gelten folgende Voraussetzungen:

- Sie müssen sich in derselben Datenbankpartitionsgruppe befinden, und zwar in einer, die nicht umverteilt wird. (Bei der Umverteilung können Tabellen in der Datenbankpartitionsgruppe möglicherweise unterschiedliche Verteilungszuordnungen verwenden, sodass sie nicht kollokiert sind.)
- Sie müssen Verteilungsschlüssel mit derselben Anzahl von Spalten besitzen.
- Ihre entsprechenden Spalten des Verteilungsschlüssels müssen datenbankpartitionskompatibel sein.
- Sie müssen sich in einer Datenbankpartitionsgruppe einer Einzelpartitionsdatenbank in derselben Datenbankpartition befinden.

Authentifizierungsaspekte bei partitionierten Datenbanken

In einer partitionierten Datenbank muss jede Partition der Datenbank über dieselbe Menge definierter Benutzer und Gruppen verfügen. Wenn die Definitionen nicht übereinstimmen, kann der Benutzer in verschiedenen Partitionen zu verschiedenen Operationen berechtigt sein. Konsistenz über alle Partitionen hinweg ist zu empfehlen.

Datenbankpartitionsgruppen

Eine Datenbankpartitionsgruppe ist eine Gruppe aus einer oder mehreren Datenbankpartitionen, die als einer Datenbank zugehörig definiert sind. 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 zur selben Instanz gehören. Eine Datenbankpartitionsgruppe kann nur eine einzige Datenbankpartition enthalten oder auch sämtliche Datenbankpartitionen der Datenbank umfassen.

Abb. 1 zeigt ein Beispiel einer Datenbank mit fünf Datenbankpartitionen in folgenden Datenbankpartitionsgruppen:

- 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.

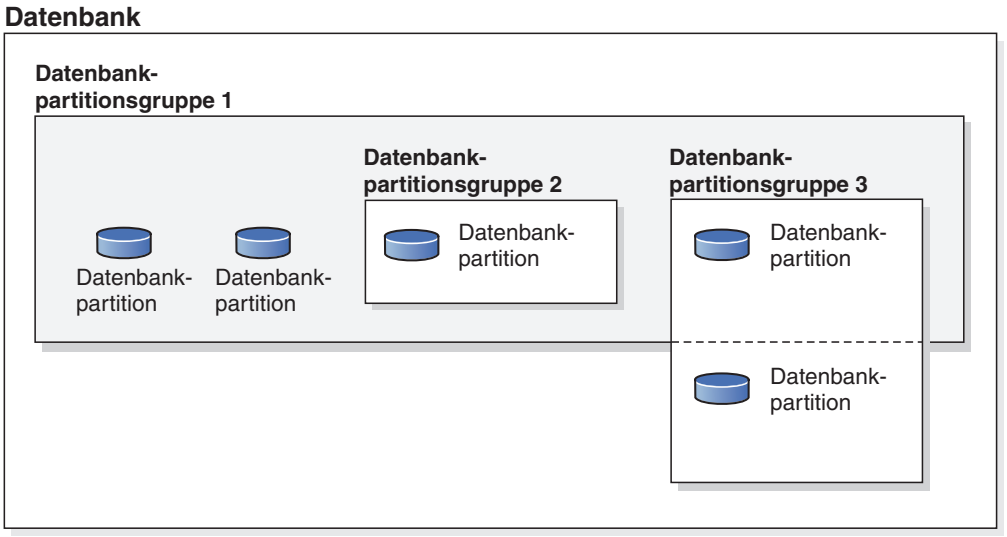


Abbildung 1. Datenbankpartitionsgruppen in einer Datenbank

Sie können eine neue Datenbankpartitionsgruppe mithilfe der Anweisung CREATE DATABASE PARTITION GROUP erstellen. Mithilfe der Anweisung ALTER DATABASE PARTITION GROUP lässt sie sich ändern. Die Daten werden auf alle Datenbankpartitionen in einer Datenbankpartitionsgruppe verteilt, und Sie können eine oder mehrere Datenbankpartitionen einer Datenbankpartitionsgruppe hinzufügen oder aus ihr löschen. Wenn Sie eine Datenbankpartitionsgruppe mit mehreren Datenbankpartitionen 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 *Konfigurationsdatei für Datenbankpartitionen* namens `db2nodes.cfg` defi-

niert sein. Eine Datenbankpartitionsgruppe kann eine Datenbankpartition, mehrere Datenbankpartitionen und auch sämtliche für das Datenbanksystem definierten Datenbankpartitionen enthalten.

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

In einer nicht partitionierten Datenbank ist kein Verteilungsschlüssel und keine Verteilungszuordnung 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.

Bei der Arbeit mit Datenbankpartitionsgruppen haben Sie folgende Möglichkeiten:

- Sie können eine Datenbankpartitionsgruppe erstellen.
- Sie können den Kommentar ändern, der einer Datenbankpartitionsgruppe zugeordnet ist.
- Sie können einer Datenbankpartitionsgruppe Datenbankpartitionen hinzufügen.
- Sie können Datenbankpartitionen aus einer Datenbankpartitionsgruppe löschen.
- Sie können Tabellendaten innerhalb einer Datenbankpartitionsgruppe umverteilen.

Aufbau von Datenbankpartitionsgruppen

Wenn Sie eine Einzelpartitionsdatenbank verwenden, spielen Überlegungen zum Aufbau von Datenbankpartitionsgruppen keine Rolle. Der DB2-Designadvisor ist ein Tool, das zur Erstellung von Empfehlungen zu Datenbankpartitionsgruppen genutzt werden kann. Der Zugriff auf den DB2-Designadvisor ist über die Steuerzentrale sowie mithilfe 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 Verteilungsschlüssels 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 Nummer zugeordnet sein. Dieselbe Datenbankpartition kann in einer oder mehreren Datenbankpartitionsgruppen enthalten sein.

- Zur Gewährleistung einer schnellen Recovery 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 Database für Linux, UNIX und Windows den Einsatz effizienterer Joinstrategien. 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 Verteilungsschlüssel haben und die Datentypen der sich entsprechenden Spalten kompatibel sind. Zeilen in zusammengefassten Tabellen mit dem gleichen Wert im Verteilungsschlüssel 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.

Verteilungszuordnungen

In einer Umgebung mit partitionierten Datenbanken muss der Datenbankmanager wissen, wo sich die benötigten Daten befinden. Zum Auffinden der Daten verwendet der Datenbankmanager eine so genannte *Verteilungszuordnung*.

Eine Verteilungszuordnung ist eine intern generierte Tabelle von entweder `4.096` Einträgen für Datenbankpartitionsgruppen mit mehreren Partitionen oder einem einzigen Eintrag für Datenbankpartitionsgruppen mit Einzelpartitionen. Für eine Datenbankpartitionsgruppe mit nur einer Partition enthält die Verteilungszuordnung nur einen Eintrag mit der Nummer der Datenbankpartition, in der alle Zeilen einer Datenbanktabelle gespeichert werden. Für Datenbankpartitionsgruppen mit mehreren Partitionen werden die Nummern der Datenbankpartitionsgruppe so angegeben, dass die Datenbankpartitionen eine nach der anderen verwendet werden, um eine gleichmäßige Verteilung über die gesamte Zuordnung sicherzustellen. Vergleichbar mit der Verwendung einer Stadtkarte, die durch Gitterlinien in Sektoren unterteilt ist, verwendet der Datenbankmanager einen *Verteilungsschlüssel*, 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 Verteilungszuordnung für die Datenbankpartitionsgruppe IBMDEFAULTGROUP dieser Datenbank sähe wie folgt aus:

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

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

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

Wenn der Verteilungsschlüssel für eine in die Datenbank zu ladende Tabelle eine ganze Zahl (Integer) mit möglichen Werten zwischen 1 und `500.000` ist, wird der Verteilungsschlüssel mit einem Hashverfahren auf eine Nummer zwischen 0 und `4.095` abgebildet. Diese Nummer wird als Index auf die Verteilungszuordnung verwendet, um die Datenbankpartition für die betreffende Zeile auszuwählen.

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

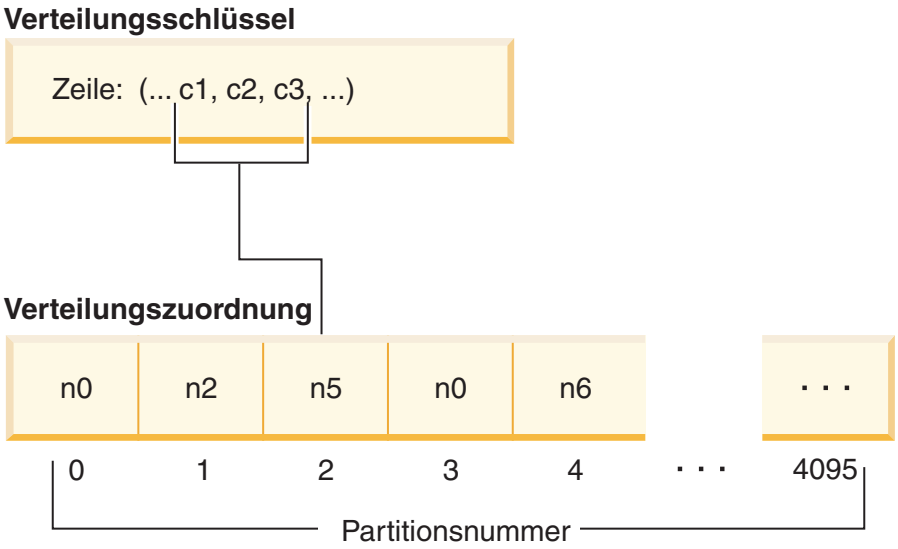


Abbildung 2. Datenverteilung mit einer Verteilungszuordnung

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

Mithilfe der API sqlugtpi können Sie eine Kopie der Verteilungszuordnung abrufen, die Sie prüfen können.

Verteilungsschlüssel

Ein *Verteilungsschlüssel* ist eine Spalte (oder Spaltengruppe), die dazu verwendet wird, die Datenbankpartition zu bestimmen, in der eine bestimmte Datenzeile gespeichert wird. Ein Verteilungsschlüssel wird für eine Tabelle mithilfe der Anweisung CREATE TABLE definiert. Wenn kein Verteilungsschlüssel für eine Tabelle in

einem Tabellenbereich, der über mehr als eine Datenbankpartition in einer Datenbankpartitionsgruppe verteilt ist, definiert ist, wird standardmäßig ein Verteilungsschlüssel aus der ersten Spalte des Primärschlüssels erstellt.

Wenn kein Primärschlüssel angegeben ist, wird standardmäßig die erste für die Tabelle definierte Spalte, die keine langen Daten enthält, als Verteilungsschlüssel angenommen. (*Lang* bezieht sich hier auf 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 Verteilungsschlüssel erstellen wollen, müssen Sie den Verteilungsschlüssel explizit definieren. In diesem Fall wird kein Standardschlüssel erstellt.

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

Die Auswahl eines guten Verteilungsschlüssels 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 Joinstrategien

Wenn Kollokation kein Hauptgesichtspunkt ist, dann zeichnet sich ein guter Verteilungsschlüssel für eine Tabelle dadurch aus, dass er die Daten gleichmäßig über alle Datenbankpartitionen in der Datenbankpartitionsgruppe verteilt. Der Verteilungsschlüssel 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 Verteilungsschlüssel 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 Verteilungsschlüssel in derselben Datenbankpartition gespeichert werden.

Ein ungeeigneter Verteilungsschlüssel kann zu einer ungleichmäßigen Datenverteilung führen. Spalten mit ungleichmäßig verteilten Daten und Spalten mit einer kleinen Anzahl unterschiedlicher Werte sollten nicht als Verteilungsschlüssel 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 Verteilung ist proportional zur Größe des Verteilungsschlüssels. Der Verteilungsschlüssel kann nicht mehr als 16 Spalten enthalten. Jedoch wird bei weniger Spalten eine bessere Leistung erzielt. Nicht benötigte Spalten sollten daher nicht in den Verteilungsschlüssel aufgenommen werden.

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

- Die Erstellung einer Mehrpartitionstabelle, die ausschließlich lange Datentypen (LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB oder DBCLOB) enthält, wird nicht unterstützt.
- Die Definition des Verteilungsschlüssels kann nicht geändert werden.
- Der Verteilungsschlüssel sollte die am häufigsten an Joins beteiligten Spalten enthalten.
- Der Verteilungsschlüssel sollte außerdem aus Spalten bestehen, die häufig von Klauseln GROUP BY betroffen sind.
- Jeder eindeutige Schlüssel oder Primärschlüssel muss alle Spalten des Verteilungsschlüssels enthalten.
- In einer OLTP-Umgebung (Online-Transaktionsverarbeitung) sollten alle Spalten des Verteilungsschlüssels an der Transaktion durch 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 Verteilungsschlüssel für die Tabelle EMP_TABLE.

Als *Datenbankpartitionierung* wird die Methode bezeichnet, mit der die Speicherposition jeder Zeile in der Tabelle bestimmt wird. Diese Methode funktioniert folgendermaßen:

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

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 Verteilungsschlüssel kompatibel sind. Durch Speichern beider Tabellen in derselben Datenbankpartitionsgruppe wird sichergestellt, dass sie eine gemeinsame Verteilungszuordnung 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 Verteilungsschlüsseln müssen *partitionskompatibel* sein.

DB2 Database für Linux, UNIX und Windows kann beim Zugriff auf mehr als eine Tabelle bei einem Join oder einer Unterabfrage erkennen, dass sich die zu verknüpfenden Daten in derselben Datenbankpartition befinden. Wenn dies geschieht, kann DB2 den Join oder die Unterabfrage in der Datenbankpartition ausführen, in der die Daten gespeichert sind, anstatt die Daten zwischen Datenbankpartitionen austauschen zu müssen. Diese Möglichkeit bietet beträchtliche Leistungsvorteile.

Partitionskompatibilität

Die Basisdatentypen entsprechender Spalten von Verteilungsschlüsseln 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 Nummer zuordnet, wenn sie denselben Wert haben.

Eine Partitionskompatibilität liegt vor, wenn die folgenden Merkmale gelten:

- 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 Zeichen- oder Grafikdatentypen kompatibel.
- Die Partitionskompatibilität wird nicht von der Optionalität der Dateneingabe für eine Spalte beeinflusst.
- Die Partitionskompatibilität wird von der Sortierfolge beeinflusst. Die localeabhängigen, UCA-basierten Sortierfolgen benötigen eine exakte Übereinstimmung in der Sortierfolge, das Attribut 'Strength (S)' der Sortierfolge wird hierbei ignoriert. Alle anderen Sortierfolgen werden im Hinblick auf das Feststellen der Partitionskompatibilität als identisch betrachtet.
- Zeichenspalten, die mit FOR BIT DATA definiert wurden, sind nur dann mit Zeichenspalten ohne FOR BIT DATA kompatibel, wenn eine andere Sortierfolge als eine localeabhängige, UCA-basierte Sortierfolge verwendet wird.
- 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 Verteilungsschlüssel werden gleich 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.
- Wird eine localeabhängige, UCA-basierte Sortierfolge verwendet, sind CHAR, VARCHAR, GRAPHIC und VARGRAPHIC kompatible Datentypen. Wird eine andere Sortierfolge verwendet, sind CHAR und VARCHAR mit unterschiedlichen Längen kompatible Typen und GRAPHIC und VARGRAPHIC sind kompatible Typen, aber CHAR und VARCHAR sind keine kompatiblen Typen für GRAPHIC und VARGRAPHIC.
- Die Partitionskompatibilität gilt nicht für die Datentypen LONG VARCHAR, LONG VARGRAPHIC, CLOB, DBCLOB und BLOB, da sie in Verteilungsschlüsseln nicht unterstützt werden.

Partitionierte Tabellen

Die Funktionalität der Tabellenpartitionierung wird mit DB2 Version 9.1 Enterprise Server Edition für Linux, UNIX und Windows (und späteren Versionen) zur Verfügung gestellt. Partitionierte Tabellen arbeiten mit einem Datenorganisationsschema, bei dem Tabellendaten auf mehrere Speicherobjekte, die als *Datenpartitionen* oder *Datenbereiche* bezeichnet werden, entsprechend den Werten einer oder mehrerer Spalten der Tabelle, die den Tabellenpartitionierungsschlüssel bilden, verteilt werden.

Eine Datenpartition oder Datenbereich (Range) ist ein Teil einer Tabelle, der eine Teilmenge der Zeilen einer Tabelle enthält und separat von anderen Gruppen von Zeilen gespeichert wird. Daten aus einer gegebenen Tabelle werden auf der Basis der in der Klausel PARTITION BY der Anweisung CREATE TABLE angegebenen Spezifikationen in mehrere Datenpartitionen oder Bereiche partitioniert. Diese Datenpartitionen oder Datenbereiche können sich in verschiedenen Tabellenbereichen, in denselben Tabellenbereichen oder in einer Kombination beider Arten von Tabellenbereichen befinden. Wenn eine Tabelle unter Verwendung der Klausel PARTITION BY erstellt wird, wird sie partitioniert.

Alle angegebenen Tabellenbereiche müssen dieselbe Seitengröße (PAGESIZE), Bereichsgröße (EXTENTSIZE), denselben Speichermechanismus (DMS, SMS) und -typ (REGULAR oder LARGE) haben. Darüber hinaus müssen sich alle Tabellenbereiche in derselben *Datenbankpartitionsgruppe* befinden.

Eine partitionierte Tabelle vereinfacht das Rollin und Rollout von Tabellendaten (d. h. die Ein- und Auslagerung ganzer Datenbereiche). Zudem kann eine partitionierte Tabelle wesentlich mehr Daten enthalten als eine gewöhnliche Tabelle. Sie können eine partitionierte Tabelle mit maximal 32767 Datenpartitionen erstellen. Datenpartitionen können einer partitionierten Tabelle hinzugefügt (ADD PARTITION) und zugeordnet (ATTACH PARTITION) werden und die Zuordnung von Datenpartitionen kann aufgehoben werden (DETACH PARTITION). Darüber hinaus können mehrere Datenpartitionsbereiche einer Tabelle in einem einzigen Tabellenbereich gespeichert werden.

Einschränkungen: Partitionierte hierarchische oder temporäre Tabellen, Bereichsclustertabellen und partitionierte Sichten werden zur Verwendung in partitionierten Tabellen nicht unterstützt. XML-Spaltentypen werden zur Verwendung in partitionierten Tabellen ebenfalls nicht unterstützt.

Tabellenpartitionierung

Die Tabellenpartitionierung ist ein Datenorganisationsschema, bei dem Tabellendaten auf mehrere Speicherobjekte, die als *Datenpartitionen* oder *Datenbereiche* (RANGE) bezeichnet werden, entsprechend den Werten einer oder mehrerer Spalten der Tabelle verteilt werden. Jede Datenpartition wird separat gespeichert. Diese Speicherobjekte können sich in verschiedenen Tabellenbereichen, in denselben Tabellenbereichen oder in einer Kombination beider Arten von Tabellenbereichen befinden.

Speicherobjekte verhalten sich ähnlich wie einzelne Tabellen. Daher lassen sich schnelle Dateneinlagerungen (Rollins) problemlos ausführen, indem eine vorhandene Tabelle mithilfe der Anweisung ALTER TABLE ...ATTACH in eine partitionierte Tabelle integriert wird. Analog lässt sich eine Datenauslagerung (Rollout) problemlos mithilfe der Anweisung ALTER TABLE ...DETACH durchführen. Auch die Abfrageverarbeitung kann von der Trennung von Daten profitieren, da sie dadurch in

der Lage ist, ein Durchsuchen irrelevanter Daten zu vermeiden. Dies führt bei vielen Abfragen, wie sie für Data-Warehouses verwendet werden, zu einer besseren Abfrageleistung.

Tabellendaten werden gemäß den Angaben in der Klausel PARTITION BY der Anweisung CREATE TABLE partitioniert. Die Spalten, die in dieser Definition verwendet werden, bilden den so genannten Tabellenpartitionierungsschlüssel.

Dieses Organisationsschema kann isoliert oder in Kombination mit anderen Organisationsschemata verwendet werden. Durch die Kombination der Klauseln DISTRIBUTE BY und PARTITION BY der Anweisung CREATE TABLE können Daten über Datenbankpartitionen verteilt werden, die sich über mehrere Tabellenbereiche erstrecken. Die folgenden Organisationsschemata sind verfügbar:

- DISTRIBUTE BY HASH
- PARTITION BY RANGE
- ORGANIZE BY DIMENSIONS

Die Funktionalität der Tabellenpartitionierung wird mit DB2 Version 9.1 Enterprise Server Edition für Linux, UNIX und Windows (und späteren Versionen) zur Verfügung gestellt.

Vorteile der Tabellenpartitionierung

Wenn eine der folgenden Bedingungen auf Sie und Ihr Unternehmen zutrifft, sollten Sie die zahlreichen Vorteile der Tabellenpartitionierung in Betracht ziehen:

- Sie haben ein Data-Warehouse, das von einfacheren Rollin- und Rolloutoperationen für Tabellendaten profitieren würde.
- Sie haben ein Data-Warehouse, das sehr große Tabellen enthält.
- Sie erwägen eine Migration auf eine Datenbank der Version 9.1 von einem vorherigen Release oder einem konkurrierenden Datenbankprodukt.
- Sie benötigen eine effektivere Nutzung von HSM-Lösungen (Hierarchical Storage Management).

Die Tabellenpartitionierung ermöglicht eine einfachere Ein- und Auslagerung (Rollin und Rollout) von Tabellendaten, eine einfachere Verwaltung, eine flexible Positionierung von Indizes sowie eine bessere Abfrageverarbeitung.

Effizientes Rollin und Rollout von Daten
Die Partitionierung von Tabellen ermöglicht eine effiziente Ein- und Auslagerung von Tabellendaten (engl. Rollin und Rollout). Diese können Sie durch die Klauseln ATTACH PARTITION und DETACH PARTITION in der Anweisung ALTER TABLE ausführen. Durch ein Rollin partitionierter Tabellendaten lässt sich ein neuer Datenbereich auf einfache Weise in eine partitionierte Tabelle in Form einer zusätzlichen Datenpartition integrieren. Durch ein Rollout partitionierter Tabellendaten lassen sich Datenbereiche auf einfache Weise von einer partitionierten Tabelle zur nachfolgenden Löschung oder Archivierung abtrennen.

Einfachere Verwaltung großer Tabellen
Die Verwaltung auf Tabellenebene ist flexibler, weil Sie Verwaltungstasks an einzelnen Datenpartitionen ausführen können. Solche Verwaltungstasks sind zum Beispiel das Aufheben der Zuordnung und das erneute Zuordnen einer Datenpartition, das Durchführen von Backup- und Restoreoperationen für einzelne Datenpartitionen sowie das Reorganisieren einzelner Indizes. Zeitaufwändige Verwaltungsoperationen können abgekürzt wer-

den, indem sie in eine Reihe kleinerer Operationen zerlegt werden. Zum Beispiel können Backupoperationen eine Datenpartition nach der anderen abarbeiten, wenn sich die Datenpartitionen in separaten Tabellenbereichen befinden. Auf diese Weise ist es möglich, das Backup für jeweils nur eine Datenpartition einer partitionierten Tabelle durchzuführen.

Flexible Platzierung von Indizes

Indizes können jetzt in anderen Tabellenbereichen gespeichert werden, sodass eine differenziertere Steuerung der Indexplatzierung möglich ist. Dies bietet zum Beispiel folgende Vorteile:

- Beim Löschen und bei der Online-Erstellung von Indizes wird eine bessere Leistung erzielt.
- Es besteht die Möglichkeit, verschiedene Werte für die Tabellenbereichsmerkmale für verschiedene Indizes der Tabelle zu verwenden (z. B. können verschiedene Seitengrößen für die einzelnen Indizes zu einer besseren Speicherplatznutzung beitragen).
- Weniger E/A-Konkurrenzsituationen lassen einen effizienteren gemeinsamen Zugriff auf die Indexdaten für die Tabelle zu.
- Wenn einzelne Indizes gelöscht werden, wird der Speicherplatz sofort für das System verfügbar, ohne dass eine Indexreorganisation erforderlich ist.
- Wenn eine Indexreorganisation ausgeführt werden soll, kann ein einzelner Index reorganisiert werden.

Sowohl DMS- als auch SMS-Tabellenbereiche unterstützen die Verwendung von Indizes, die sich an einer anderen Position befinden als die Tabelle.

Verbesserte Leistung für Abfragen, wie sie für Business-Intelligence-Anwendungen verwendet werden

Die Abfrageverarbeitung wurde erweitert, sodass Datenpartitionen auf der Basis von Vergleichselementen der Abfrage automatisch ausgeschlossen werden. Von dieser Funktionalität zum *Ausschluss von Datenpartitionen* (Data Partition Elimination) können viele Entscheidungsunterstützungsabfragen profitieren.

Im folgenden Beispiel wird eine Tabelle *customer* erstellt, bei der Zeilen mit l_shipdate >= '01/01/2006' und l_shipdate <= '03/31/2006' im Tabellenbereich ts1, Zeilen mit l_shipdate >= '04/01/2006' und l_shipdate <= '06/30/2006' in Tabellenbereich ts2 usw. gespeichert werden.

```
CREATE TABLE customer (l_shipdate DATE, l_name CHAR(30))
IN ts1, ts2, ts3, ts4, ts5
PARTITION BY RANGE(l_shipdate) (STARTING FROM ('01/01/2006')
ENDING AT ('12/31/2006') EVERY (3 MONTHS))
```

Datenpartitionen und Datenbereiche

Partitionierte Tabellen arbeiten mit einem Datenorganisationsschema, bei dem Tabellendaten auf mehrere Speicherobjekte, die als *Datenpartitionen* oder *Datenbereiche* bezeichnet werden, entsprechend den Werten einer oder mehrerer Spalten der Tabelle, die den Tabellenpartitionierungsschlüssel bilden, verteilt werden. Die für die einzelnen Datenpartitionen angegebenen Bereiche können bei der Erstellung einer Tabelle automatisch generiert oder manuell angegeben werden.

Auf Datenpartitionen wird innerhalb der DB2-Bibliothek auf verschiedene Weise Bezug genommen. Die folgende Liste gibt die am häufigsten verwendeten Verweise an:

- DATAPARTITIONNAME ist der permanente Name, der einer Datenpartition für eine bestimmte Tabelle bei der Erstellung zugeordnet wird. Dieser Spaltenwert wird in der Katalogsicht SYSCAT.DATAPARTITIONS gespeichert. Dieser Name bleibt bei einer ATTACH- oder DETACH-Operation nicht erhalten.
- DATAPARTITIONID ist die permanente ID, die einer Datenpartition für eine bestimmte Tabelle bei der Erstellung zugeordnet wird. Sie dient zur eindeutigen Kennzeichnung einer bestimmten Datenpartition in einer gegebenen Tabelle. Diese ID bleibt bei einer ATTACH- oder DETACH-Operation nicht erhalten. Dieser Wert wird vom System generiert und kann in den Ausgaben verschiedener Dienstprogramme angezeigt werden.
- SEQNO gibt die Reihenfolge eines bestimmten Datenpartitionsbereichs in Bezug auf andere Datenpartitionsbereiche in der Tabelle an, wobei Datenpartitionen, deren Zuordnung aufgehoben wurde, bei einer Sortierung hinter allen sichtbaren und zugeordneten Datenpartitionen angeordnet werden.

Datenorganisationsschemata

Mit der Einführung der Tabellenpartitionierung bietet eine DB2-Datenbank ein drei Ebenen umfassendes Datenorganisationsschema an. Es gibt drei Klauseln der Anweisung CREATE TABLE, die einen Algorithmus zur Angabe enthalten, wie die Daten zu organisieren sind.

Die folgenden drei Klauseln zeigen die Ebenen der Datenorganisation, die in beliebiger Kombination verwendet werden können:

- Die Klausel DISTRIBUTE BY dient zur gleichmäßigen Verteilung der Daten auf Datenbankpartitionen (zur Einrichtung der abfrageinternen Parallelität und zum Lastausgleich zwischen den Datenbankpartitionen: Datenbankpartitionierung).
- Die Klausel PARTITION BY dient zur Gruppierung von Zeilen mit ähnlichen Werten einer einzigen Dimension in der gleichen Datenpartition (Tabellenpartitionierung).
- Die Klausel ORGANIZE BY dient zur Gruppierung von Zeilen mit ähnlichen Werten in mehreren Dimensionen im gleichen Tabellenspeicherbereich (Mehrdimensionales Clustering).

Diese Syntax ermöglicht Konsistenz zwischen den Klauseln und lässt zukünftige Algorithmen der Datenorganisation zu. Jede dieser Klauseln kann isoliert oder in Kombination mit den anderen verwendet werden. Durch die Kombination der Klauseln DISTRIBUTE BY und PARTITION BY der Anweisung CREATE TABLE können Daten über Datenbankpartitionen verteilt werden, die sich über mehrere Tabellenbereiche erstrecken. Dieses Lösung ermöglicht ein ähnliches Verhalten wie die Hybridfunktionalität von Informix Dynamic Server und Informix Extended Parallel Server.

Sie können die Klauseln in einer Tabelle kombinieren, um durch die verschiedenen Datenorganisationsschemata ausgereiftere Partitionierungsschemata zu erstellen. Zum Beispiel ist DB2 Database Partitioning Feature (DPF) nicht nur kompatibel mit der Tabellenpartitionierung, sondern auch eine geeignete Ergänzung.

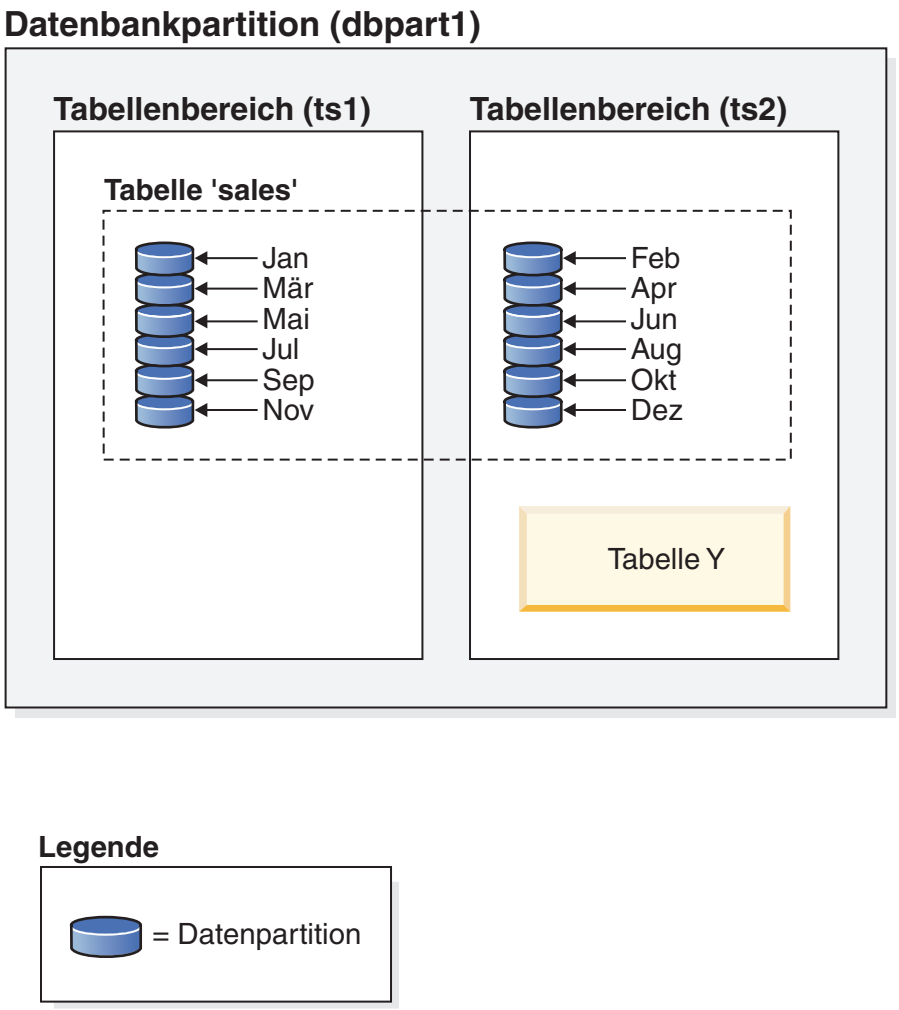


Abbildung 3. Veranschaulichung des Organisationsschemas der Tabellenpartitionierung, bei dem eine Tabelle mit monatlichen Vertriebsdaten in mehrere Datenpartitionen unterteilt ist. Die Tabelle erstreckt sich außerdem über zwei Tabellenbereiche ('ts1' und 'ts2').

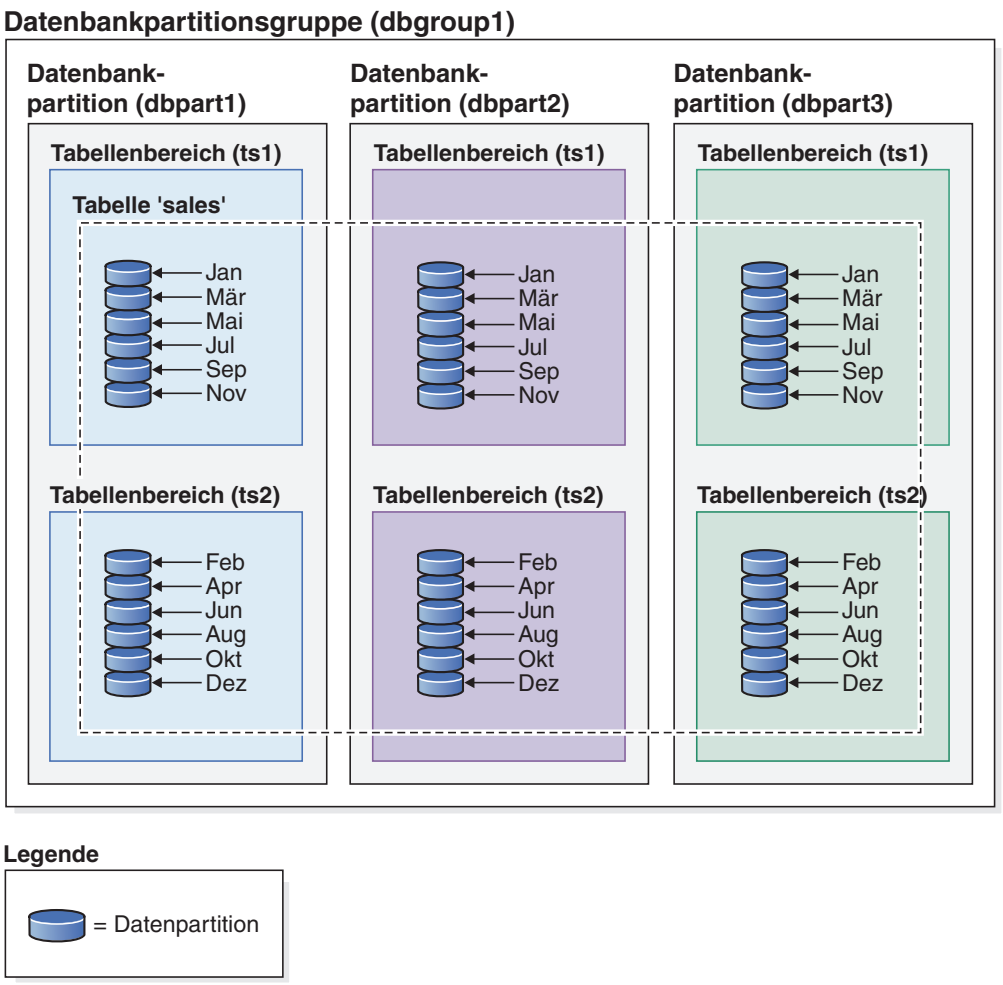


Abbildung 4. Veranschaulichung der ergänzenden Organisationsschemata der Datenbankpartitionierung und Tabellenpartitionierung. Eine Tabelle mit monatlichen Vertriebsdaten ist in mehrere Datenpartitionen partitioniert, die sich über zwei Tabellenbereiche ('ts1' und 'ts2') erstrecken, die wiederum über mehrere Datenbankpartitionen ('dbpart1', 'dbpart2', 'dbpart3') einer Datenbankpartitionsgruppe ('dbgroup1') verteilt sind.

Der auffälligste Unterschied zwischen dem Mehrdimensionalen Clustering (MDC) und der Tabellenpartitionierung besteht im Vorhandensein mehrerer Dimensionen im Gegensatz zu einer einzigen Dimension. MDC eignet sich für Datenwürfel (d. h. Tabellen mit mehreren Dimensionen), während sich die Tabellenpartitionierung anbietet, wenn eine einzige Dimension vorhanden ist, die für den Aufbau der Datenbank von zentraler Bedeutung ist, wie zum Beispiel eine Datumsspalte. MDC und Tabellenpartitionierung können sich ergänzen, wenn beide dieser Bedingungen erfüllt sind. Dies wird in Abb. 5 auf Seite 19 veranschaulicht.

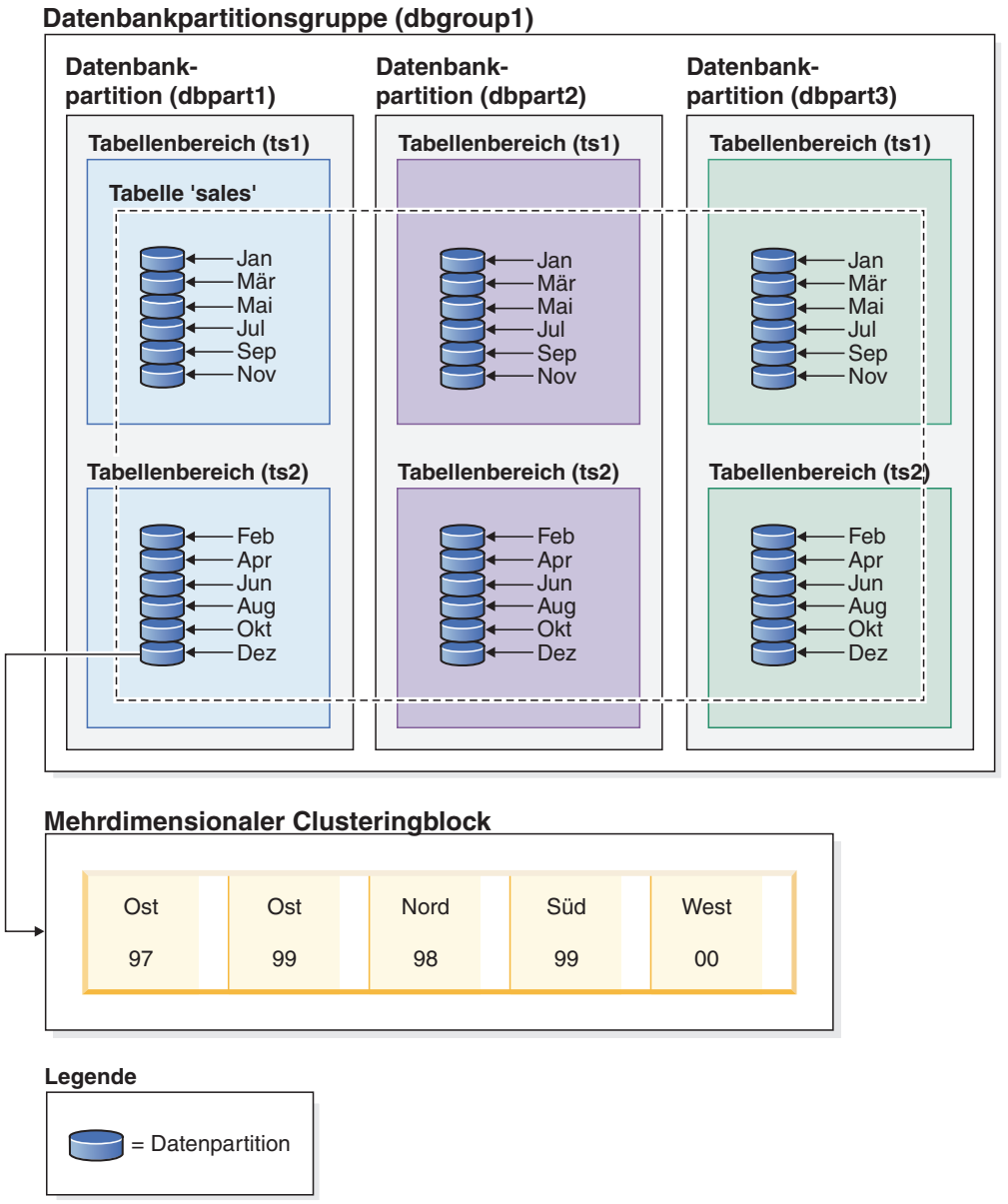


Abbildung 5. Eine Darstellung der Organisationsschemata der Datenbankpartitionierung, der Tabellenpartitionierung und des Mehrdimensionalen Clustering, in denen Daten aus der Tabelle 'sales' nicht nur über mehrere Datenbankpartitionen verteilt und über die Tabellenbereiche 'ts1' und 'ts2' partitioniert sind, sondern auch Zeilen mit ähnlichen Werten in den Dimensionen für Datum und Region gruppiert werden.

Es gibt ein weiteres Datenorganisationsschema, das nicht in Verbindung mit einem der anderen oben gezeigten Schemata verwendet werden kann. Dieses Schema wird durch die Klausel ORGANIZE BY KEY SEQUENCE angeben. Es dient zur Einfügung der einzelnen Datensätze jeweils in eine Zeile, die bei der Erstellung der Tabelle für diesen Datensatz reserviert wurde (Bereichsclustertabelle).

Terminologie der Datenorganisation

Datenbankpartitionierung

Ein Datenorganisationsschema, bei dem Tabellendaten über mehrere Datenbankpartitionen auf der Basis der Hashwerte in einer oder mehreren Verteilungsschlüsselspalten der Tabelle sowie mithilfe einer Verteilungszuordnung der Datenbankpartitionierung verteilt werden. Daten aus einer

gegebenen Tabelle werden gemäß den Spezifikationen verteilt, die in der Klausel DISTRIBUTE BY HASH der Anweisung CREATE TABLE angegeben wurden.

Datenbankpartition
Ein Teil einer Datenbank auf einem Datenbankpartitionsserver, der eigene Benutzerdaten und Indizes, eine Konfigurationsdatei und Transaktionsprotokolle enthält. Datenbankpartitionen können logisch oder physisch sein.

Tabellenpartitionierung
Ein Datenorganisationsschema, bei dem Tabellendaten auf mehrere Datenpartitionen entsprechend den Werten einer oder mehrerer Partitionierungsspalten der Tabelle verteilt werden. Daten aus einer gegebenen Tabelle werden in mehrere Speicherobjekte auf der Basis der in der Klausel PARTITION BY der Anweisung CREATE TABLE angegebenen Spezifikationen partitioniert. Diese Speicherobjekte können sich in verschiedenen Tabellenbereichen befinden.

Datenpartition
Eine Gruppe von Tabellenzeilen, die separat von anderen Gruppen von Zeilen gespeichert und durch die Spezifikationen zu Gruppen zusammengefasst werden, die in der Klausel PARTITION BY RANGE der Anweisung CREATE TABLE angegeben wurden.

MDC-Tabelle (MDC - Mehrdimensionales Clustering)
Eine Tabelle, deren Daten physisch in Blöcken in einer oder mehreren Dimensionen bzw. Clusteringschlüsseln organisiert werden, die in der Klausel ORGANIZE BY DIMENSIONS angegeben werden.

Vorteile der einzelnen Datenorganisationsschemata

Die Kenntnis der Vorteile der einzelnen Datenorganisationsschemata kann Ihnen helfen, bei der Planung, beim Entwurf oder bei der erneuten Beurteilung der Anforderungen Ihres Datenbanksystems die beste Strategie zu wählen. Tabelle 2 bietet eine Übersicht über häufige Anforderungen von Kunden und zeigt, wie die verschiedenen Datenorganisationsschemata bei der Erfüllung dieser Anforderungen behilflich sein können.

Tabelle 2. Verwendung der Tabellenpartitionierung mit dem Database Partitioning Feature (DPF)

Problemstellung	Empfohlenes Schema	Erläuterung
Datenrollout	Tabellenpartitionierung	Nutzt die Aufhebung der Zuordnung (DETACH PARTITION) zur Durchführung eines Rollouts großer Datenmengen unter minimaler Beeinträchtigung.
Parallele Abfrageausführung (Abfrageleistung)	Database Partitioning Feature	Ermöglicht Abfrageparallelität zur Realisierung einer besseren Abfrageleistung.
Ausschluss von Datenpartitionen (Abfrageleistung)	Tabellenpartitionierung	Ermöglicht den Ausschluss von Datenpartitionen zur Realisierung einer besseren Abfrageleistung.

Tabelle 2. Verwendung der Tabellenpartitionierung mit dem Database Partitioning Feature (DPF) (Forts.)

Problemstellung	Empfohlenes Schema	Erläuterung
Maximierung der Abfrageleistung	Beide	Maximale Abfrageleistung bei kombinierter Verwendung: Abfrageparallelität und Ausschluss von Datenpartitionen ergänzen sich.
Hohe Belastung von Administratoren	Database Partitioning Feature	Ausführung zahlreicher Tasks für die einzelnen Datenbankpartitionen.

Tabelle 3. Verwendung der Tabellenpartitionierung mit MDC-Tabellen

Problemstellung	Empfohlenes Schema	Erläuterung
Datenverfügbarkeit während des Rollouts	Tabellenpartitionierung	Sie können die Klausel DETACH PARTITION verwenden, um ein Rollout großer Datenmengen unter minimaler Beeinträchtigung durchzuführen.
Abfrageleistung	Beide	MDC eignet sich am besten für die Abfrage mehrerer Dimensionen. Die Tabellenpartitionierung assistiert durch den Ausschluss von Datenpartitionen.
Minimale Reorganisation	MDC	MDC-Tabellen behalten das Clustering bei, sodass sich die Notwendigkeit von Reorganisationen verringert.

Anmerkung: Die Tabellenpartitionierung wird jetzt vor UNION ALL-Sichten empfohlen.

Datenorganisationsschemata in DB2- und Informix-Datenbanken

Die Tabellenpartitionierung ist ein Datenorganisationsschema, bei dem Tabellendaten auf mehrere Speicherobjekte, die als Datenpartitionen oder Datenbereiche (RANGE) bezeichnet werden, entsprechend den Werten einer oder mehrerer Spalten der Tabelle verteilt werden. Jede Datenpartition wird separat gespeichert. Diese Speicherobjekte können sich in verschiedenen Tabellenbereichen, in denselben Tabellenbereichen oder in einer Kombination beider Arten von Tabellenbereichen befinden.

Tabellendaten werden gemäß den Angaben in der Klausel PARTITION BY der Anweisung CREATE TABLE partitioniert. Die Spalten, die in dieser Definition verwendet werden, bilden den so genannten Tabellenpartitionierungsschlüssel. Die DB2-Tabellenpartitionierung lässt sich mit der Datenfragmentierungsstrategie für die Datenorganisation vergleichen, die von Informix Dynamic Server und Informix Extended Parallel Server angeboten wird.

Das Informix-Konzept

Informix unterstützt eine Reihe von Datenorganisationsschemata, die in den Informix-Produkten als *Fragmentierung* bezeichnet werden. Einer der häufiger genutzten Typen von Fragmentierung ist FRAGMENT BY EXPRESSION. Dieser Fragmentierungstyp funktioniert ähnlich einer CASE-Anweisung, wobei jedem Fragment der Tabelle ein Ausdruck zugeordnet ist. Diese Ausdrücke werden ausgewertet, um zu bestimmen, wo die Zeile abzulegen ist.

Vergleich zwischen dem Informix- und dem DB2-Datenbanksystem

Das DB2-Datenbanksystem stellt ein reich ausgestattetes Sortiment an ergänzenden Funktionen bereit, die sich direkt den Datenorganisationsschemata von Informix zuordnen lassen und es Kunden relativ leicht machen, sich von der Informix-Syntax auf die DB2-Syntax umzustellen. Der DB2-Datenbankmanager realisiert komplizierte Informix-Schemata durch eine Kombination aus generierten Spalten und der Klausel PARTITION BY RANGE der Anweisung CREATE TABLE. Tabelle 4 stellt die von Informix- und DB2-Datenbankprodukten verwendeten Datenorganisationsschemata einander gegenüber.

Tabelle 4. Übersicht über die Informix-Datenorganisationsschemata und ihre DB2-Entsprechungen

Datenorganisationsschema	Informix-Syntax	Syntax in DB2 Version 9.1
• Informix: ausdrucksbasiert • DB2: Tabellenpartitionierung	FRAGMENT BY EXPRESSION	PARTITION BY RANGE
• Informix: Umlaufmodus • DB2: Standard	FRAGMENT BY ROUND ROBIN	Keine Syntax: Der DB2-Datenbankmanager verteilt Daten automatisch auf Container.
• Informix: Bereichsverteilung • DB2: Tabellenpartitionierung	FRAGMENT BY RANGE	PARTITION BY RANGE
• Informix: systemdefiniertes Hashverfahren • DB2: Datenbankpartitionierung	FRAGMENT BY HASH	DISTRIBUTE BY HASH
• Informix: HYBRID • DB2: Datenbankpartitionierung mit Tabellenpartitionierung	FRAGMENT BY HYBRID	DISTRIBUTE BY HASH, PARTITION BY RANGE
• Informix: n/v • DB2: Mehrdimensionales Clustering (MDC)	n/v	ORGANIZE BY DIMENSION

Beispiele

Die folgenden Beispiele zeigen Details dazu, wie äquivalente DB2-Ergebnisse für die einzelnen FRAGMENT BY EXPRESSION-Schemata von Informix realisiert werden.

Beispiel 1: Die folgende einfache Anweisung CREATE TABLE zeigt die Informix-Fragmentierung und die äquivalente Tabellenpartitionierungssyntax für ein DB2-Datenbanksystem:

Informix-Syntax:

```
CREATE TABLE demo(a INT) FRAGMENT BY EXPRESSION
 a = 1 IN db1,
 a = 2 IN db2,
 a = 3 IN db3;
```

DB2-Syntax:

```
CREATE TABLE demo(a INT) PARTITION BY RANGE(a)
 (STARTING(1) IN db1,
 STARTING(2) IN db2,
 STARTING(3) ENDING(3) IN db3);
```

Informix XPS unterstützt ein zweistufiges Fragmentierungsschema, das als Hybridschema bezeichnet wird. Bei diesem Schema werden Daten über Co-Server mithilfe eines Ausdrucks und innerhalb des Co-Servers mithilfe eines zweiten Ausdrucks verteilt. Dadurch können alle Co-Server an einer Abfrage aktiv arbeiten (d. h. auf allen Co-Servern befinden sich Daten), und die Abfrage kann den Ausschluss von Datenpartitionen vorteilhaft nutzen.

Das DB2-Datenbanksystem realisiert ein zum Informix-Hybridschema äquivalentes Organisationsschema durch eine Kombination aus den Klauseln DISTRIBUTE BY und PARTITION BY der Anweisung CREATE TABLE.

Beispiel 2: Das folgende Beispiel zeigt die Syntax für die kombinierten Klauseln:

Informix-Syntax

```
CREATE TABLE demo(a INT, b INT) FRAGMENT BY HYBRID HASH(a)
 EXPRESSION b = 1 IN dbsl1,
   b = 2 IN dbsl2;
```

DB2-Syntax

```
CREATE TABLE demo(a INT, b INT) IN dbsl1, dbsl2
 DISTRIBUTE BY HASH(a),
 PARTITION BY RANGE(b) (STARTING 1 ENDING 2 EVERY 1);
```

Darüber hinaus können Sie MDC-Tabellen verwenden, um eine zusätzliche Ebene der Datenorganisation zu gewinnen:

```
CREATE TABLE demo(a INT, b INT, c INT) IN dbsl1, dbsl2
 DISTRIBUTE BY HASH(a),
 PARTITION BY RANGE(b) (STARTING 1 ENDING 2 EVERY 1)
 ORGANIZE BY DIMENSIONS(c);
```

In diesem Beispiel werden alle Zeilen mit dem gleichen Wert in der Spalte **a** in derselben Datenbankpartition abgelegt. Alle Zeilen mit dem gleichen Wert in Spalte **b** werden im selben Tabellenbereich gespeichert. Für einen gegebenen Wert in Spalte **a** und Spalte **b** werden alle Zeilen mit gleichen Wert in Spalte **c** im selben Clusteringbereich auf der Platte zusammengefasst. Diese Lösung eignet sich ideal für OLAP-Operationen, die detailliertere Informationen abrufen, weil nur ein oder mehrere Speicherbereiche (Extents) bzw. Blöcke in einem einzigen Tabellenbereich in einer einzigen Datenbankpartition durchsucht werden müssen, um eine Abfrage dieses Typs zu erfüllen.

Anwendung der Tabellenpartitionierung auf häufig anzutreffende Anwendungsprobleme

In den folgenden Abschnitten wird die Anwendung der verschiedenen Funktionsmerkmale der DB2-Tabellenpartitionierung auf Anwendungsprobleme erläutert, die häufiger vorkommen. In jedem Abschnitt werden dabei bewährte Verfahrensweisen zur Umsetzung verschiedener Fragmentierungsschemata von Informix in äquivalente DB2-Tabellenpartitionierungsschemata behandelt.

Zu beachtende Aspekte für die Erstellung einfacher Datenpartitionsbereiche

Eine der gängigsten Anwendungen der Tabellenpartitionierung ist die Partitionierung einer großen Fakttabelle nach einem Datumsschlüssel. Wenn Sie gleichmäßig große Bereiche von Daten erzeugen müssen, ziehen Sie die Verwendung einer Form der CREATE TABLE-Syntax zur automatischen Generierung in Betracht.

Beispiele

Beispiel 1: Das folgende Beispiel zeigt die Form der Syntax zur automatischen Generierung:

```
CREATE TABLE orders
(
 l_orderkey  DECIMAL(10,0) NOT NULL,
 l_partkey  INTEGER,
 l_suppkey  INTEGER,
 l_linenumber  INTEGER,
 l_quantity  DECIMAL(12,2),
 l_extendedprice  DECIMAL(12,2),
 l_discount  DECIMAL(12,2),
 l_tax   DECIMAL(12,2),
 l_returnflag  CHAR(1),
 l_linestatus  CHAR(1),
 l_shipdate  DATE,
 l_commitdate  DATE,
 l_receiptdate  DATE,
 l_shipinstruct  CHAR(25),
 l_shipmode  CHAR(10),
 l_comment  VARCHAR(44))
  PARTITION BY RANGE(l_shipdate)
  (STARTING '1/1/1992' ENDING '12/31/1993' EVERY 1 MONTH);
```

Durch diese Anweisung werden 24 Bereiche, d. h. einer für jeden Monat im Zeitraum 1992-1993, erstellt. Ein Versuch, eine Zeile mit einem Wert für l_shipdate einzufügen, der außerhalb dieses Bereichs liegt, führt zu einem Fehler.

Beispiel 2: Vergleichen Sie das vorausgehende Beispiel mit der folgenden Informix-Syntax:

```
create table orders
(
 l_orderkey  decimal(10,0) not null,
 l_partkey  integer,
 l_suppkey  integer,
 l_linenumber integer,
 l_quantity  decimal(12,2),
 l_extendedprice decimal(12,2),
 l_discount  decimal(12,2),
 l_tax   decimal(12,2),
 l_returnflag char(1),
 l_linestatus char(1),
 l_shipdate  date,
```

```
 l_commitdate date,
 l_receiptdate date,
 l_shipinstruct char(25),
 l_shipmode  char(10),
 l_comment  varchar(44)
) fragment by expression
l_shipdate < '1992-02-01' in ldbs1,
l_shipdate >= '1992-02-01' and l_shipdate < '1992-03-01' in ldbs2,
l_shipdate >= '1992-03-01' and l_shipdate < '1992-04-01' in ldbs3,
l_shipdate >= '1992-04-01' and l_shipdate < '1992-05-01' in ldbs4,
l_shipdate >= '1992-05-01' and l_shipdate < '1992-06-01' in ldbs5,
l_shipdate >= '1992-06-01' and l_shipdate < '1992-07-01' in ldbs6,
l_shipdate >= '1992-07-01' and l_shipdate < '1992-08-01' in ldbs7,
l_shipdate >= '1992-08-01' and l_shipdate < '1992-09-01' in ldbs8,
l_shipdate >= '1992-09-01' and l_shipdate < '1992-10-01' in ldbs9,
l_shipdate >= '1992-10-01' and l_shipdate < '1992-11-01' in ldbs10,
l_shipdate >= '1992-11-01' and l_shipdate < '1992-12-01' in ldbs11,
l_shipdate >= '1992-12-01' and l_shipdate < '1993-01-01' in ldbs12,
l_shipdate >= '1993-01-01' and l_shipdate < '1993-02-01' in ldbs13,
l_shipdate >= '1993-02-01' and l_shipdate < '1993-03-01' in ldbs14,
l_shipdate >= '1993-03-01' and l_shipdate < '1993-04-01' in ldbs15,
l_shipdate >= '1993-04-01' and l_shipdate < '1993-05-01' in ldbs16,
l_shipdate >= '1993-05-01' and l_shipdate < '1993-06-01' in ldbs17,
l_shipdate >= '1993-06-01' and l_shipdate < '1993-07-01' in ldbs18,
l_shipdate >= '1993-07-01' and l_shipdate < '1993-08-01' in ldbs19,
l_shipdate >= '1993-08-01' and l_shipdate < '1993-09-01' in ldbs20,
l_shipdate >= '1993-09-01' and l_shipdate < '1993-10-01' in ldbs21,
l_shipdate >= '1993-10-01' and l_shipdate < '1993-11-01' in ldbs22,
l_shipdate >= '1993-11-01' and l_shipdate < '1993-12-01' in ldbs23,
l_shipdate >= '1993-12-01' and l_shipdate < '1994-01-01' in ldbs24,
l_shipdate >= '1994-01-01' in ldbs25;
```

Beachten Sie, dass die Informix-Syntax einen Bereich mit offenem Anfang und offenem Ende vorsieht, um Datumseingaben aufzufangen, die nicht innerhalb des erwarteten Bereichs liegen. Durch das Hinzufügen von Bereichen mit MINVALUE und MAXVALUE lässt sich die DB2-Syntax der Informix-Syntax entsprechend anpassen.

Beispiel 3: Im folgenden Beispiel wird Beispiel 1 zur Angleichung an die Informix-Syntax abgewandelt:

```
CREATE TABLE orders
(
 l_orderkey  DECIMAL(10,0) NOT NULL,
 l_partkey  INTEGER,
 l_suppkey  INTEGER,
 l_linenumber INTEGER,
 l_quantity  DECIMAL(12,2),
 l_extendedprice DECIMAL(12,2),
 l_discount  DECIMAL(12,2),
 l_tax   DECIMAL(12,2),
 l_returnflag CHAR(1),
 l_linestatus CHAR(1),
 l_shipdate  DATE,
 l_commitdate DATE,
 l_receiptdate DATE,
 l_shipinstruct CHAR(25),
 l_shipmode  CHAR(10),
 l_comment  VARCHAR(44)
) PARTITION BY RANGE(l_shipdate)
 (STARTING MINVALUE,
  STARTING '1/1/1992' ENDING '12/31/1993' EVERY 1 MONTH,
  ENDING MAXVALUE);
```

Durch dieses Verfahren kann nun jedes beliebige Datum in die Tabelle eingefügt werden.

Partitionierung durch Ausdruck mit generierten Spalten

Obwohl das DB2-Datenbanksystem keine direkte Unterstützung für eine Partitionierung durch einen Ausdruck bietet, wird die Partitionierung über eine generierte Spalte unterstützt, sodass sich dasselbe Ergebnis erzielen lässt.

Beachten Sie die folgenden Verwendungshinweise, bevor Sie entscheiden, ob diese Lösung in Betracht kommt:

- Die generierte Spalte ist eine reale Spalte, die physischen Plattenspeicher belegt. Tabellen, die eine generierte Spalte enthalten, können daher etwas größer sein.
- Eine Änderung des Ausdrucks für die generierte Spalte, über die eine partitionierte Tabelle partitioniert wird, wird nicht unterstützt. Jeder Versuch, dies zu tun, führt zu einer Nachricht SQL0190. Das Hinzufügen einer neuen Datenpartition zu einer Tabelle, die generierte Spalten in der im folgenden Abschnitt beschriebenen Weise verwendet, erfordert in der Regel eine Änderung des Ausdrucks, der die generierte Spalte definiert. Das Ändern des Ausdrucks, der eine generierte Spalte definiert, wird gegenwärtig nicht unterstützt.
- Die Möglichkeiten zur Anwendung des Ausschlusses von Datenpartitionen unterliegen Einschränkungen, wenn eine Tabelle generierte Spalten verwendet.

Beispiele

Beispiel 1: Im folgenden Beispiel wird die Informix-Syntax anstellen verwendet, an denen sich die Verwendung generierter Spalten anbietet. In diesem Beispiel enthält die Spalte, über die die Partitionierung erfolgt, kanadische Provinzen und Territorien. Da es unwahrscheinlich ist, dass sich die Liste der Provinzen ändert, ist es auch unwahrscheinlich, dass sich der Ausdruck für die generierte Spalte ändert.

```
CREATE TABLE customer (
 cust_id  INT,
 cust_prov CHAR(2))
 FRAGMENT BY EXPRESSION
  cust_prov = "AB" IN dbspace_ab
  cust_prov = "BC" IN dbspace_bc
  cust_prov = "MB" IN dbspace_mb
  ...
  cust_prov = "YT" IN dbspace_yt
  REMAINDER IN dbspace_remainder;
```

Beispiel 2: In diesem Beispiel wird die DB2-Tabelle über eine generierte Spalte partitioniert:

```
CREATE TABLE customer (
 cust_id     INT,
 cust_prov   CHAR(2),
 cust_prov_gen GENERATED ALWAYS AS (CASE
  WHEN cust_prov = 'AB' THEN 1
  WHEN cust_prov = 'BC' THEN 2
  WHEN cust_prov = 'MB' THEN 3
  ...
  WHEN cust_prov = 'YT' THEN 13
  ELSE 14 END))
 IN tbspace_ab, tbspace_bc, tbspace_mb, .... tbspace_remainder
 PARTITION BY RANGE (cust_prov_gen)
  (STARTING 1 ENDING 14 EVERY 1);
```

Hier entsprechen die Ausdrücke in der CASE-Anweisung den zuvor gezeigten Ausdrücken in der Klausel FRAGMENT BY EXPRESSION. Die CASE-Anweisung ordnet jedem ursprünglichen Ausdruck eine Nummer zu, die in der generierten Spalte (hier: cust_prov_gen) gespeichert wird. Diese Spalte ist eine reale Spalte, die auf der Platte gespeichert wird. Daher kann die Tabelle in dieser Form geringfügig mehr Speicher beanspruchen als wenn DB2 die Partitionierung durch einen Ausdruck direkt unterstützen würde. In diesem Beispiel wird die Kurzform der Syntax verwendet. Daher müssen die Tabellenbereiche, in denen die Datenpartitionen gespeichert werden sollen, in der IN-Klausel der Anweisung CREATE TABLE aufgelistet werden. Bei der Langform der Syntax ist für jede Datenpartition eine separate IN-Klausel erforderlich.

Anmerkung: Dieses Verfahren lässt sich auf jede beliebige Klausel FRAGMENT BY EXPRESSION anwenden.

Tabellenpartitionierungsschlüssel

Ein Tabellenpartitionierungsschlüssel ist eine geordnete Gruppe aus mindestens einer Spalte in einer Tabelle. Mithilfe der Werte in den Spalten des Tabellenpartitionierungsschlüssels wird bestimmt, zu welcher Datenpartition die einzelnen Tabellenzeilen gehören.

Zur Definition des Tabellenpartitionierungsschlüssels für eine Tabelle verwenden Sie die Anweisung CREATE TABLE mit der Klausel PARTITION BY.

Die Auswahl einer geeigneten Spalte für den Tabellenpartitionierungsschlüssel spielt eine wesentliche Rolle, wenn Sie die Vorteile der Tabellenpartitionierung in vollem Umfang ausschöpfen möchten. Die folgenden Richtlinien können Ihnen bei der Auswahl der am besten geeigneten Spalten für den Tabellenpartitionierungsschlüssel Ihrer partitionierten Tabelle helfen:

- Definieren Sie Bereiche, die der Rollin-Größe von Daten entsprechen. Daten werden zum Beispiel häufig über eine Datums- oder Zeitspalte partitioniert.
- Definieren Sie die Bereichsunterteilung, die dem Datenrollout entsprechen soll. Dazu werden sehr häufig Monate oder Quartale verwendet.
- Partitionieren Sie über eine Spalte, die Vorteile beim Ausschluss von Partitionen bietet.

Unterstützte Datentypen

In Tabelle 5 werden die Datentypen (einschließlich Synonymen) aufgeführt, die zur Verwendung in einer Spalte eines Tabellenpartitionierungsschlüssels unterstützt werden:

Tabelle 5. Unterstützte Datentypen

Datentypspalte 1	Datentypspalte 2
SMALLINT	INTEGER
INT	BIGINT
FLOAT	REAL
DOUBLE	DECIMAL
DEC	NUMERIC
NUM	CHARACTER
CHAR	VARCHAR
DATE	TIME

Tabelle 5. Unterstützte Datentypen (Forts.)

Datentypspalte 1	Datentypspalte 2
GRAPHIC	VARGRAPHIC
CHARACTER VARYING	TIMESTAMP
CHAR VARYING	CHARACTER FOR BIT DATA
CHAR FOR BIT DATA	VARCHAR FOR BIT DATA
CHARACTER VARYING FOR BIT DATA	CHAR VARYING FOR BIT DATA
Benutzerdefinierte Datentypen (einzigartige Datentypen)	

Nicht unterstützte Datentypen

Die folgenden Datentypen können in einer partitionierten Tabelle enthalten sein, werden jedoch nicht zur Verwendung in einer Spalte des Tabellenpartitionierungsschlüssels unterstützt:

- Benutzerdefinierte Datentypen (strukturierte Datentypen)
- LONG VARCHAR
- LONG VARCHAR FOR BIT DATA
- BLOB
- BINARY LARGE OBJECT
- CLOB
- CHARACTER LARGE OBJECT
- DBCLOB
- LONG VARGRAPHIC
- REF
- Zeichenfolge mit variabler Länge für C
- Zeichenfolge mit variabler Länge für Pascal

Der Datentyp XML wird in einer partitionierten Tabelle nicht unterstützt.

Wenn Sie eine automatische Generierung von Datenpartitionen mithilfe der Klausel EVERY der Anweisung CREATE TABLE anfordern, kann nur eine Spalte als Tabellenpartitionierungsschlüssel verwendet werden. Wenn Sie Datenpartitionen manuell generieren, indem Sie die einzelnen Bereiche in der Klausel PARTITION BY der Anweisung CREATE TABLE angeben, können mehrere Spalten als Tabellenpartitionierungsschlüssel verwendet werden, wie im folgenden Beispiel gezeigt:

```
CREATE TABLE sales (year INT, month INT)
   PARTITION BY RANGE(year, month)
   (STARTING FROM (2001, 1) ENDING (2001,3) IN tbsp1,
   ENDING (2001,6) IN tbsp2, ENDING (2001,9)
   IN tbsp3, ENDING (2001,12) IN tbsp4,
   ENDING (2002,3) IN tbsp5, ENDING (2002,6)
   IN tbsp6, ENDING (2002,9) IN tbsp7,
   ENDING (2002,12) IN tbsp8)
```

Mit dieser Anweisung werden acht Datenpartitionen erstellt, d. h. eine für jedes Quartal der Jahre 2001 und 2002.

Anmerkung:

1. Wenn mehrere Spalten als Tabellenpartitionierungsschlüssel verwendet werden, werden sie insofern wie ein zusammengesetzter Schlüssel behandelt (ähnlich

den zusammengesetzten Schlüsseln in einem Index), als dass nachfolgende Spalten von den führenden Spalten abhängig sind. Jeder Anfangs- oder Endwert (aller Spalten zusammen) muss in höchstens 512 Zeichen angegeben werden. Diese Begrenzung entspricht der Größe der Spalten LOWVALUE und HIGHVALUE der Katalogsicht SYSCAT.DATAPARTITIONS. Ein Anfangs- oder Endwert, der mit mehr als 512 Zeichen angegeben wird, führt zu einem Fehler SQL0636N mit Ursachencode 9.

2. Die Tabellenpartitionierung erstreckt sich über mehrere Spalten, nicht Dimensionen. Bei der Tabellenpartitionierung gehören alle Spalten nur zu einer Dimension.

Generierte Spalten

Generierte Spalten können als Tabellenpartitionierungsschlüssel verwendet werden. Im folgenden Beispiel wird eine Tabelle mit zwölf Datenpartitionen, d. h. eine für jeden Monat, erstellt. Alle Zeilen für Januar eines beliebigen Jahres werden in der ersten Datenpartition, die Zeilen für Februar in der zweiten usw. abgelegt.

Beispiel 1

```
CREATE TABLE monthly_sales (sales_date date,
 sales_month int GENERATED ALWAYS AS (month(sales_date)))
     PARTITION BY RANGE (sales_month)
     (STARTING FROM 1 ENDING AT 12 EVERY 1);
```

Anmerkung:

1. Sie können den Ausdruck einer generierten Spalte, die im Tabellenpartitionierungsschlüssel verwendet wird, nicht ändern oder löschen. Auch das Hinzufügen eines Ausdrucks zur Generierung einer Spalte in einer Spalte, die im Tabellenpartitionierungsschlüssel verwendet wird, ist nicht zulässig. Der Versuch, einen Ausdruck zur Generierung einer Spalte in einer Spalte, die im Tabellenpartitionierungsschlüssel verwendet wird, hinzuzufügen, zu löschen oder zu ändern führt zu einem Fehler (SQL0270N RC=52).
2. Der Ausschluss von Datenpartitionen wird für Bereichsvergleichselemente nicht verwendet, wenn die generierte Spalte nicht monoton ist oder das Optimierungsprogramm nicht erkennen kann, dass sie monoton ist. Wenn nicht monotone Ausdrücke vorhanden sind, kann der Ausschluss von Datenpartitionen nur für Gleichheitsvergleichselemente oder IN-Vergleichselemente ausgeführt werden. Eine detaillierte Beschreibung und Beispiele zur Monotonie finden Sie in „Aspekte der Erstellung von MDC-Tabellen" auf Seite 56.

Ladeaspekte für partitionierte Tabellen

Alle vorhandenen Ladefunktionen werden unterstützt, wenn die Zieltabelle partitioniert ist. Hierbei gelten jedoch die folgenden allgemeinen Einschränkungen:

- Konsistenzpunkte werden nicht unterstützt, wenn die Anzahl der Partitionierungsagenten größer als 1 ist.
- Das Laden von Daten in eine Untergruppe von Datenpartitionen, während die verbleibenden Datenpartitionen vollständig online bleiben, wird nicht unterstützt.
- Die von einer Ladeoperation verwendete Ausnahmetabelle kann nicht partitioniert sein.
- Ein eindeutiger Index kann nicht erneut erstellt werden, wenn das Dienstprogramm LOAD im Einfügemodus (INSERT) oder Neustartmodus (RESTART) ausgeführt wird und freigegebene Objekte von der Zieltabelle der Ladeoperation abhängen.

- Ähnlich wie beim Laden von MDC-Tabellen bleibt die genaue Reihenfolge der Eingabedatensätze beim Laden von partitionierten Tabellen nicht erhalten. Sortierungen werden lediglich innerhalb der Zelle oder Datenpartition beibehalten.
- Ladeoperationen, die mehrere Formatierungsprogramme auf jeder Datenbankpartition verwenden, erhalten lediglich eine ungefähre Reihenfolge der Eingabedatensätze. Beim Ausführen eines einzelnen Formatierungsprogramms auf jeder Datenbankpartition werden die Eingabedatensätze nach Zellen- oder Tabellenpartitionierungsschlüssel gruppiert. Um auf jeder Datenbankpartition ein einziges Formatierungsprogramm auszuführen, muss für CPU_PARALLELISM explizit der Wert 1 angefordert werden.

Allgemeines Verhalten des Dienstprogramms LOAD
Das Dienstprogramm LOAD fügt Datensätze in die richtige Datenpartition ein. Es ist nicht erforderlich, ein externes Dienstprogramm (wie beispielsweise einen Verteilerprozess) zu verwenden, um die Eingabedaten vor dem Laden zu partitionieren.

Das Dienstprogramm LOAD greift nicht auf freigegebene oder zugeordnete Datenpartitionen zu. Daten werden lediglich in sichtbare Datenpartitionen eingefügt. Sichtbare Datenpartitionen sind weder zugeordnet noch freigegeben. Darüber hinaus werden freigegebene oder zugeordnete Datenpartitionen von LOAD REPLACE-Operationen nicht abgeschnitten. Da das Dienstprogramm LOAD Sperren für die Systemkatalogtabellen anfordert, wartet das Dienstprogramm LOAD auf alle ALTER TABLE-Transaktionen, für die noch kein Commit durchgeführt wurde. Solche Transaktionen fordern eine exklusive Sperre für die relevanten Zeilen in den Katalogtabellen an, und die exklusive Sperre muss beendet werden, bevor die Ladeoperation fortgesetzt werden kann. Dies bedeutet, dass nicht festgeschriebene Transaktionen ALTER TABLE ... ADD PARTITION, ALTER TABLE ... ATTACH PARTITION bzw. ALTER TABLE ... DETACH PARTITION nicht zulässig sind, während die Ladeoperation ausgeführt wird. Alle Eingabequellensätze, die für eine zugeordnete (ATTACHED) oder freigegebene (DETACHED) Datenpartition bestimmt sind, werden zurückgewiesen und können aus der Ausnahmetabelle abgerufen werden, sofern eine angegeben wurde. Eine Informationsnachricht wird in die Nachrichtendatei geschrieben, um anzugeben, dass einige Datenpartitionen der Zieltabelle im Status zugeordnet (ATTACHED) oder freigegeben (DETACHED) waren. Sperren für die relevanten Zeilen der Katalogtabelle, die der Zieltabelle entsprechen, verhindern, dass Benutzer die Partitionierung der Zieltabelle durch Absetzen einer Operation ALTER TABLE ... ADD PARTITION, ALTER TABLE ... ATTACH PARTITION bzw. ALTER TABLE ... DETACH PARTITION während der Ausführung des Dienstprogramms LOAD ändern.

Verarbeitung ungültiger Zeilen
Wenn das Dienstprogramm LOAD auf einen Datensatz trifft, der zu keiner der sichtbaren Datenpartitionen gehört, wird der betreffende Datensatz zurückgewiesen, und das Dienstprogramm LOAD setzt die Verarbeitung fort. Die Anzahl der Datensätze, die aufgrund einer Verletzung der Bereichsvorgabe zurückgewiesen wurden, wird nicht explizit angegeben, ist aber in der Gesamtanzahl der zurückgewiesenen Datensätze enthalten. Durch das Zurückweisen eines Datensatzes aufgrund einer Verletzung der Bereichsvorgabe wird die Anzahl der Warnungen für Zeilen nicht erhöht. Es wird eine einzige Nachricht (SQL0327N) in die Nachrichtendatei des Dienstprogramms LOAD geschrieben, die angibt, dass Bereichsverletzungen festgestellt wurden. Es wird jedoch nicht für jeden Datensatz eine eigene Nachricht protokolliert. Zusätzlich zu allen Spalten der Zieltabelle enthält die Ausnahmetabelle auch Spalten, die den Typ des Verstoßes beschreibt, der für eine be-

stimmte Zeile aufgetreten ist. Zeilen, die ungültige Daten enthalten (einschließlich Daten, die nicht partitioniert werden können), werden in die Speicherauszugsdatei geschrieben.

Da Einfügungen (INSERT-Operationen) in die Ausnahmetabelle ressourcenintensiv sind, können Sie steuern, welche Verstöße gegen Integritätsbedingungen in die Ausnahmetabelle eingefügt werden sollen. Das Standardverhalten des Dienstprogramms LOAD ist beispielsweise, Zeilen in die Ausnahmetabelle einzufügen, die aufgrund einer Verletzung der Bereichsvorgabe oder der eindeutigen Integritätsbedingung zurückgewiesen wurden, ansonsten jedoch gültig sind. Sie können dieses Verhalten inaktivieren, indem Sie jeweils NORANGEEXC oder NOUNIQUEEXC mit der Ausnahmebedingung FOR EXCEPTION angeben. Wenn Sie angeben, dass diese Verletzungen der Bereichsvorgabe nicht in die Ausnahmetabelle eingefügt werden sollen, oder wenn Sie keine Ausnahmetabelle angeben, gehen alle Informationen zu Zeilen, die eine Bereichsvorgabe oder eine eindeutige Integritätsbedingung verletzen, verloren.

Protokolldatei
Bei partitionierten Zieltabellen enthält der entsprechende Eintrag in der Protokolldatei keine Liste der Tabellenbereiche, die die Zieltabelle umfasst. Eine andere Kennung für die Granularität von Operationen ('R' anstelle von 'T') gibt an, dass eine Ladeoperation für eine partitionierte Tabelle ausgeführt wurde.

Beenden einer Ladeoperation
Beim Beenden einer LOAD REPLACE-Operation werden alle sichtbaren Datenpartitionen vollständig abgeschnitten; beim Beenden einer LOAD INSERT-Operation werden alle sichtbaren Datenpartitionen auf ihre jeweilige Länge vor der Ladeoperation abgeschnitten. Beim Beenden einer ALLOW READ ACCESS-Ladeoperation, die in der LOAD COPY-Phase fehlschlug, werden die entsprechenden Indizes ungültig gemacht. Ein Index wird ebenfalls ungültig gemacht, wenn eine Ladeoperation mit der Option ALLOW NO ACCESS beendet wird, die den Index geändert hat. (Der Index wird ungültig gemacht, weil der Indexierungsmodus REBUILD (erneut erstellen) ist oder weil ein Schlüssel während der inkrementellen Verwaltung eingefügt wurde und den Index in einem inkonsistenten Status belassen hat). Das Laden von Daten in mehrere Ziele hat keinen Einfluss auf Recoveryoperationen, mit der Ausnahme, dass die Ladeoperation von einem Konsistenzpunkt, der während der LOAD-Phase erstellt wurde, nicht erneut gestartet werden kann. In diesem Fall wird die LOAD-Option SAVECOUNT ignoriert, wenn die Zieltabelle partitioniert ist. Dieses Verhalten entspricht dem Verhalten beim Laden von Daten in einer MDC-Zieltabelle.

Generierte Spalten
Befindet sich eine generierte Spalte in einem der Partitionierungs-, Dimensions- oder Verteilungsschlüssel, wird der Änderungswert `generatedoverride` für den Dateityp ignoriert, und das Dienstprogramm LOAD generiert Werte, als wäre der Änderungswert `generatedignore` für den Dateityp angegeben. Das Laden eines falschen Wertes einer generierten Spalte kann in diesem Fall dazu führen, dass der Datensatz in die falsche physische Position (wie beispielsweise in die falsche Datenpartition, den falschen MDC-Block oder die falsche Datenbankpartition) gestellt wird. Beispiel: Befindet sich ein Datensatz in einer falschen Datenpartition, muss die Operation SET INTEGRITY zum Festlegen der Integrität den Satz in eine andere physische Position versetzen, was im Verlauf von SET INTEGRITY-Operationen, die online ausgeführt werden, nicht bewerkstelligt werden kann.

Datenverfügbarkeit
Der aktuelle Algorithmus für ALLOW READ ACCESS-Ladeoperationen erstreckt

sich auf partitionierte Tabellen. Eine ALLOW READ ACCESS-Ladeoperation ermöglicht den gleichzeitigen Lesezugriff auf die gesamte Tabelle. Dies schließt sowohl Ladedatenpartitionen als auch Nicht-Ladedatenpartitionen ein.

Datenpartitionsstatus
Nach einer erfolgreichen Ladeoperation können sichtbare Datenpartitionen entweder in den Tabellenstatus Set Integrity Pending oder Read Access Only oder beide wechseln. Dies hängt von bestimmten Bedingungen ab. So können Datenpartitionen in diese Status versetzt werden, wenn für die Tabelle Integritätsbedingungen bestehen, die von der Ladeoperation nicht verwaltet werden können. Dabei kann es sich z. B. um Integritätsbedingungen handeln, die Prüfungen auf Integritätsbedingungen und freigegebene MQTs (Materialized Query Table) beinhalten. Schlägt eine Ladeoperation fehl, verbleiben alle sichtbaren Datenpartitionen im Tabellenstatus Load Pending ('Laden anstehend').

Fehlerisolation
Eine Fehlerisolation auf Datenpartitionsebene wird nicht unterstützt. Das Isolieren von Fehlern bedeutet, Ladeoperationen für Datenpartitionen ohne Fehler fortzusetzen und Ladeoperationen für Datenpartitionen mit Fehlern zu stoppen. Fehler können in verschiedenen Datenbankpartitionen übergreifend isoliert werden. Das Dienstprogramm LOAD kann jedoch keine Transaktionen in einer Untergruppe von sichtbaren Datenpartitionen festschreiben und in den verbleibenden sichtbaren Datenpartitionen rückgängig machen.

Sonstige Aspekte

- Eine inkrementelle Indexierung wird nicht unterstützt, wenn einer der Indizes als ungültig markiert ist. Ein Index gilt als ungültig, wenn er erneut erstellt werden muss oder wenn freigegebene abhängige Objekte mit der Anweisung SET INTEGRITY überprüft werden müssen.
- Das Laden von Daten in Tabellen, deren Partitionierung anhand einer beliebigen Kombination aus Algorithmen für das Partitionieren nach Bereich, das Verteilen nach Hash oder das Organisieren nach Dimension erfolgte, wird ebenfalls unterstützt.
- Die Größe von Protokollsätzen, die eine Liste der von der Ladeoperation betroffenen Objekt- und Tabellenbereichs-IDs umfassen (Protokollsätze LOAD START und COMMIT (PENDING LIST)) kann stark anwachsen und so den für andere Anwendungen zur Verfügung stehenden aktiven Protokollspeicher reduzieren.
- Wenn eine Tabelle sowohl partitioniert als auch verteilt ist, kann es sein, dass sich eine Ladeoperation für partitionierte Datenbanken nicht auf alle Datenbankpartitionen auswirkt. Nur die Objekte in den Ausgabedatenbankpartitionen werden geändert.
- Während einer Ladeoperation nimmt die Speicherbelegung für partitionierte Tabellen mit der Anzahl der Tabellen zu. Bitte beachten Sie, dass der Gesamtanstieg nicht linear ist, da nur ein geringer Prozentsatz des Gesamtspeicherbedarfs proportional zur Anzahl der Datenpartitionen ist.

Replizierte MQTs

Eine *MQT (Materialized Query Table, gespeicherte Abfragetabelle)* ist eine Tabelle, die durch eine Abfrage definiert ist, mit der auch die Daten für die Tabelle festgelegt werden. Durch MQTs kann die Leistungsfähigkeit von Abfragen erhöht werden. Wenn DB2 Database für Linux, UNIX und Windows erkennt, dass ein Teil einer Abfrage durch eine MQT aufgelöst werden könnte, kann der Datenbankmanager die Abfrage so abwandeln, dass die entsprechende MQT verwendet wird.

In einer Umgebung mit partitionierten Datenbanken können Sie MQTs replizieren und verwenden, um die Abfrageleistung zu steigern. Eine *replizierte MQT* 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 MQT wird durch Ausführen der Anweisung CREATE TABLE mit dem Schlüsselwort REPLICATED erstellt.

Mithilfe von replizierten MQTs können Sie Kollokationen zwischen Tabellen herstellen, die normalerweise nicht kollokiert sind. Replizierte MQTs sind besonders nützlich für Joins 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 mithilfe der Kollokation erzielten Leistungsvorteile wettgemacht.

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

DELETE- oder UPDATE-Anweisungen mit nicht deterministischen Operationen werden bei replizierten MQTs nicht unterstützt.

Tabellenbereiche in Datenbankpartitionsgruppen

Wenn ein Tabellenbereich in einer Datenbankpartitionsgruppe mit mehreren Datenbankpartitionen erstellt wird, werden alle Tabellen innerhalb des Tabellenbereichs auf alle Datenbankpartitionen in der Datenbankpartitionsgruppe verteilt oder partitioniert.

Der Tabellenbereich wird in einer Datenbankpartitionsgruppe erstellt. Wenn der Tabellenbereich in einer Datenbankpartitionsgruppe erstellt wurde, muss er dort verbleiben und kann nicht in eine andere Datenbankpartitionsgruppe versetzt werden. Für die Zuordnung eines Tabellenbereichs zu einer Datenbankpartitionsgruppe wird die Anweisung CREATE TABLESPACE verwendet.

Tabellenpartitionierung und MDC-Tabellen

Bei einer Tabelle kann es sich sowohl um eine MDC-Tabelle (Tabelle mit mehrdimensionalem Clustering) als auch um eine partitionierte Tabelle handeln. In einer Tabelle, die beide Merkmale aufweist, können Spalten sowohl in der Bereichspartitionsspezifikation (Range-Partition-Spec) als auch im MDC-Schlüssel für die Tabellenpartitionierung verwendet werden. Diese Möglichkeit ermöglicht eine exaktere Differenzierung des Datenpartitions- und Blockausschlusses, als dies bei Verwendung nur einer dieser Funktionalitäten der Fall wäre. Es gibt auch zahlreiche Anwendungen, in denen es sinnvoll ist, für den MDC-Schlüssel andere Spalten als die für die Tabellenpartitionierung genutzten Spalten anzugeben. Hierbei ist zu beachten, dass die Tabellenpartitionierung mit mehreren Spalten arbeitet, während beim MDC mit mehreren Dimensionen gearbeitet wird.

Merkmale eines normalen DB2 Data Warehouse der Version 9.1

Die folgenden Empfehlungen beziehen sich auf typische, normale Data Warehouses, die in DB2 Version 9.1 neu implementiert wurden. Die folgenden Merkmale werden angenommen:

- Die Datenbank arbeitet auf mehreren Systemen oder in mehreren logischen AIX-Partitionen.
- Die Datenbankpartitionierungsfunktion (DPF) wird verwendet (Tabellen werden mithilfe der Klausel DISTRIBUTE BY HASH erstellt).
- Es wurden zwischen vier und 50 Datenpartitionen definiert.
- Die Tabelle, für die die Möglichkeit zur Verwendung des MDC und der Tabellenpartitionierung geprüft wird, ist eine der zentralen Fakttabellen.
- Die Tabelle umfasst zwischen 100 Millionen und 100 Milliarden Zeilen.
- Neue Daten werden in unterschiedlichen Zeitrahmen geladen: Jeweils über Nacht, wöchentlich, monatlich.
- Das tägliche Aufnahmevolumen liegt zwischen 10.000 und 10 Millionen Datensätzen.
- Die Datenvolumen schwanken: Hierbei liegt das Volumen des Spitzenmonats um das Fünffache höher als das des Monats mit dem geringsten Datenaufkommen. Entsprechend beläuft sich auch der Umfang der größten Dimensionen (Produktlinie, Bereich) auf das Fünffache des Umfangs der kleinsten Dimensionen.
- Die Datenbank enthält detaillierte Datenbestände der letzten 1 - 5 Jahre.
- Abgelaufene Daten werden in monatlichen oder vierteljährlichen Abständen mit einer Rollout-Operation ausgelagert.
- Tabellen verwenden eine Vielzahl von Abfragetypen. Der Workload besteht jedoch größtenteils aus analytischen Abfragen, die in Bezug auf OLTP-Workloads die folgenden Merkmale aufweisen:
  - Es gibt umfangreichere Ergebnismengen mit bis zu 2 Millionen Zeilen.
  - Die Mehrzahl oder alle Abfragen beziehen sich auf Sichten und nicht auf Basistabellen.
- SQL-Klauseln zur Auswahl von Daten nach Bereichen (Klausel BETWEEN), Elementen in Listen etc.

Merkmale einer Fakttabelle eines normalen DB2 Data Warehouse der Version 9.1

Eine normale Data Warehouse-Fakttabelle kann z. B. das folgende Design verwenden:

- Erstellung von Datenpartitionen über die Spalte 'Month'.
- Definition einer Datenpartition für jeden Rollout-Zeitraum, z. B. 1 Monat, 3 Monate.
- Erstellung von MDC-Dimensionen für 'Day' und für 1 - 4 weitere Dimensionen. Typische Dimensionen sind: Produktlinie und Bereich.
- Alle Datenpartitionen und MDC-Cluster sind über alle Datenpartitionen verteilt.

Das MDC und die Tabellenpartitionierung bieten teilweise dieselben Vorteile. Die folgende Tabelle enthält mögliche Anforderungen innerhalb Ihres Unternehmens und Empfehlungen zu einem Organisationsschema, die auf der Basis der zuvor festgestellten Merkmale gegeben werden.

Tabelle 6. Verwendung der Tabellenpartitionierung mit MDC-Tabellen

Problemstellung	Empfohlenes Schema	Empfehlung
Datenverfügbarkeit während des Rollouts	Tabellenpartitionierung	Sie können die Klausel DETACH PARTITION verwenden, um ein Rollout großer Datenmengen unter minimaler Beeinträchtigung durchzuführen.
Abfrageleistung	Tabellenpartitionierung und MDC	MDC eignet sich am besten für die Abfrage mehrerer Dimensionen. Die Tabellenpartitionierung assistiert durch den Ausschluss von Datenpartitionen.
Minimale Reorganisation	MDC	MDC-Tabellen behalten das Clustering bei, sodass sich die Notwendigkeit von Reorganisationen verringert.
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums während eines traditionellen Offlinezeitfensters	Tabellenpartitionierung	Die Datenpartitionierung kann diese Anforderung voll erfüllen. Durch das MDC würden sich keine zusätzlichen Vorteile ergeben und dieses Verfahren wäre weniger geeignet.
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums während eines Mikro-Offlinezeitfensters (weniger als 1 Minute)	Tabellenpartitionierung	Die Datenpartitionierung kann diese Anforderung voll erfüllen. Durch das MDC würden sich keine zusätzlichen Vorteile ergeben und dieses Verfahren wäre weniger geeignet.
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums bei gleichzeitiger Erhaltung der Tabellenverfügbarkeit für Geschäftsbenutzer, die Abfragen ohne Serviceverluste weiterhin übergeben können	MDC	MDC kann nicht alle in diesem Zusammenhang geltenden Anforderungen voll erfüllen. Die Tabellenpartitionierung eignet sich hier nicht, da die Tabelle nur für kurze Zeit in den Offlinemodus versetzt wird.
Tägliches Laden von Daten (entweder ALLOW READ ACCESS oder ALLOW NO ACCESS)	Tabellenpartitionierung und MDC	MDC bietet hier die meisten Vorteile. Die Tabellenpartitionierung bietet Vorteile in Teilbereichen.
Kontinuierliches Laden von Daten (ALLOW READ ACCESS)	Tabellenpartitionierung und MDC	MDC bietet hier die meisten Vorteile. Die Tabellenpartitionierung bietet Vorteile in Teilbereichen.
Leistung bei der Ausführung von Abfragen für traditionelle BI-Abfragen	Tabellenpartitionierung und MDC	MDC eignet sich besonders gut für Abfragen in Datenkuben und mehreren Dimensionen. Die Tabellenpartitionierung bietet Unterstützung über den Partitionsausschluss.

Tabelle 6. Verwendung der Tabellenpartitionierung mit MDC-Tabellen (Forts.)

Problemstellung	Empfohlenes Schema	Empfehlung
Minimierung des Reorganisationsaufwandes durch Vermeidung des Reorganisationsbedarfs oder Reduzierung des Aufwands für die Ausführung der Task	MDC	Beim MDC wird das Clustering beibehalten, sodass sich die Notwendigkeit von Reorganisationen verringert. Wenn das MDC verwendet wird, dann bietet die Datenpartitionierung auch in Teilbereichen keine Vorteile. Allerdings ermöglicht die Bereichspartitionierung bei Nichtverwendung des MDC die Reduzierung des Reorganisationsbedarfs, indem auf Partitionsebene ein etwas groberes Clustering beibehalten wird.

Beispiel 1:

Sie arbeiten mit einer Tabelle mit den Schlüsselspalten 'YearAndMonth' und 'Province'. Bei der Planung dieser Tabelle wäre eine Partitionierung nach Datum mit einer Zeitspanne von zwei Monaten pro Datenpartition sinnvoll. Darüber hinaus können Sie die Daten auch nach der Spalte 'Province' organisieren, sodass alle Zeilen für ein bestimmtes Gebiet innerhalb eines Datumsbereichs von zwei Monaten in einer Gruppe zusammengefasst werden. Diese Vorgehensweise ist in Abb. 6 auf Seite 37 dargestellt.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (Province);
```

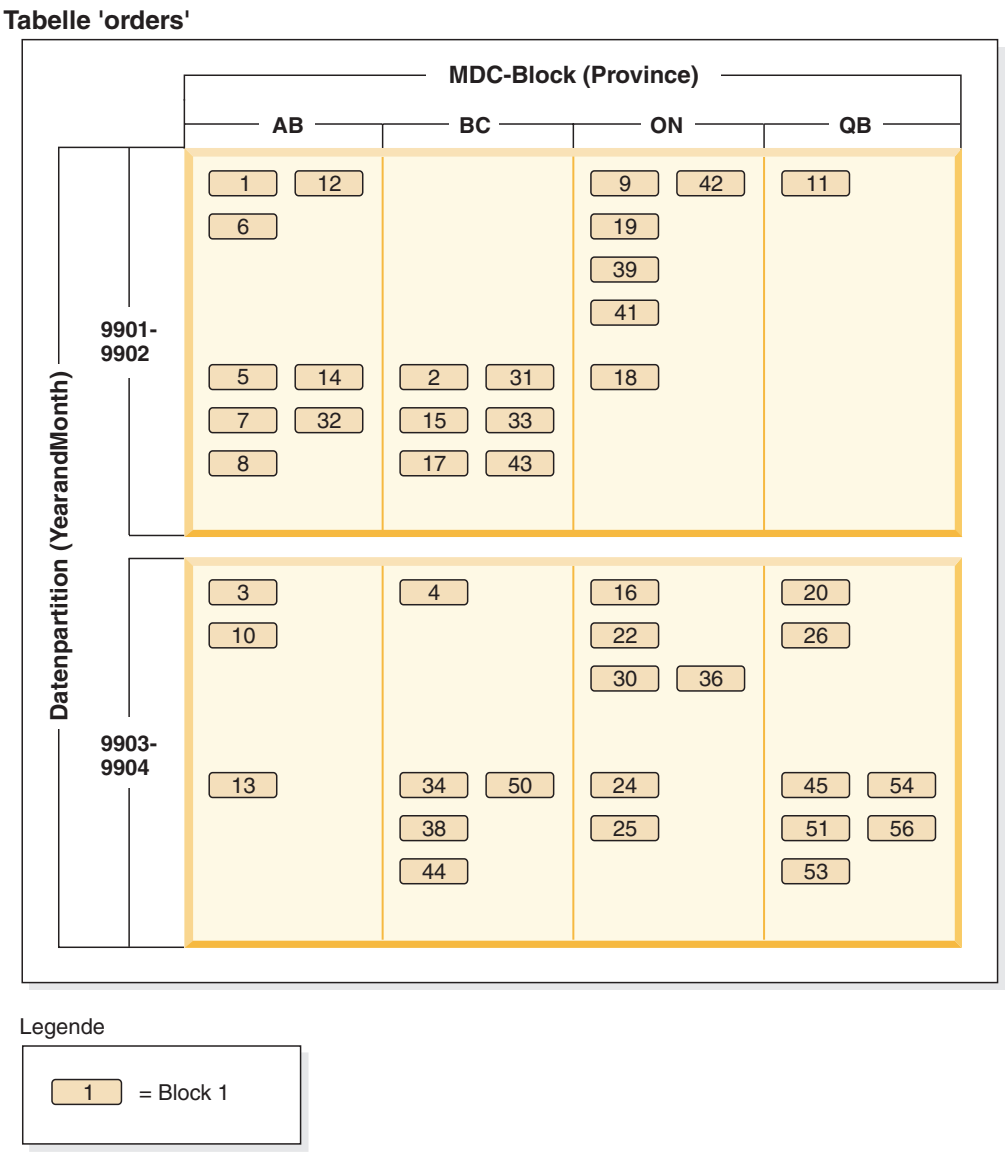


Abbildung 6. Eine nach 'YearAndMonth' partitionierte und nach 'Province' organisierte Tabelle

Beispiel 2:

Eine exaktere Differenzierung kann durch Hinzufügen von 'YearAndMonth' zur Klausel ORGANIZE BY (siehe hierzu Abb. 7 auf Seite 38) erzielt werden.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (YearAndMonth, Province);
```

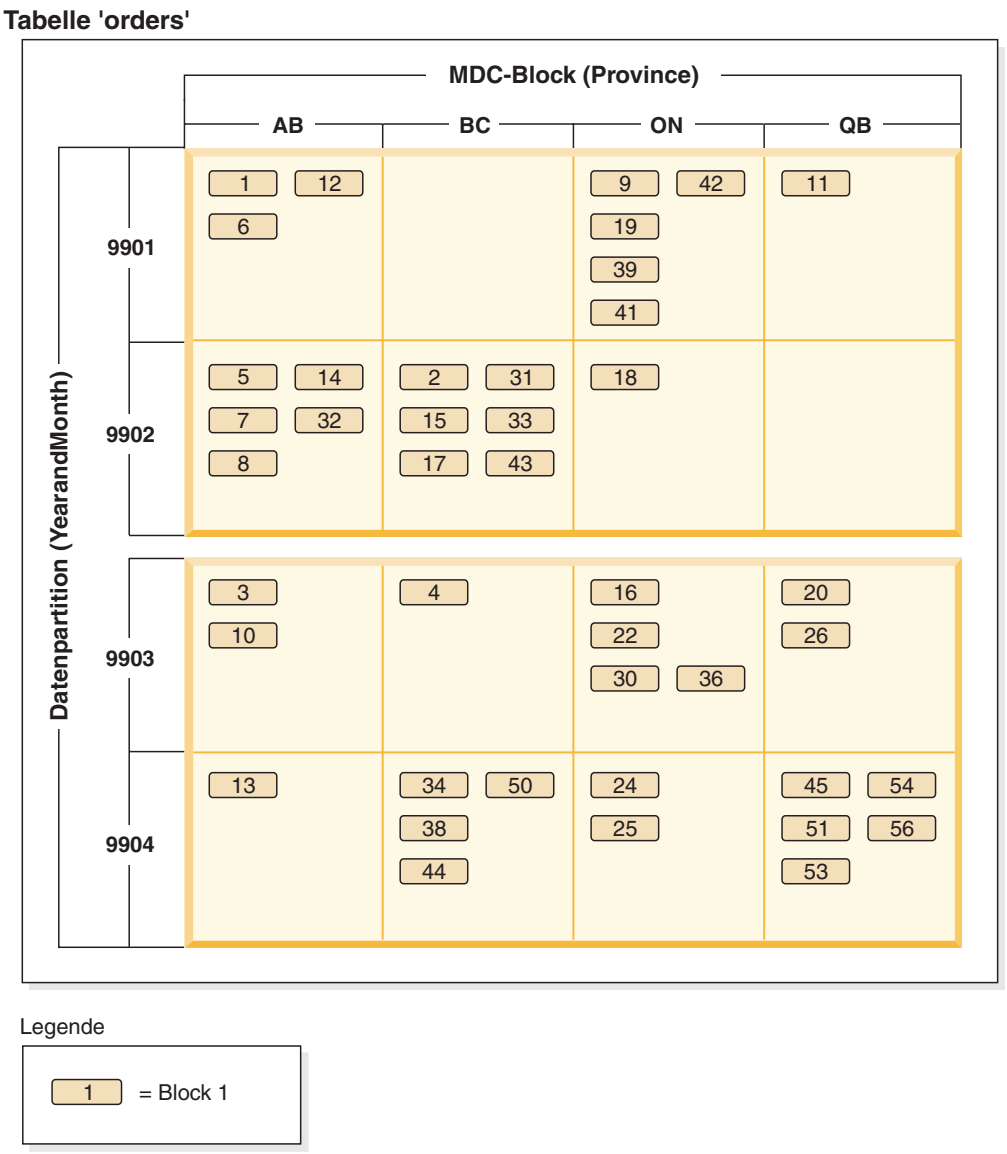


Abbildung 7. Eine nach 'YearAndMonth' partitionierte und nach 'Province' und 'YearAndMonth' organisierte Tabelle

Wenn die Partitionierung so festgelegt wurde, dass jeder Bereich nur einen einzigen Wert enthält, ergeben sich keine Vorteile, wenn die Tabellenpartitionierungsspalte in den MDC-Schlüssel aufgenommen wird.

Wichtige Hinweise

- Im Vergleich mit einer Basistabelle benötigen sowohl MDC-Tabellen als auch partitionierte Tabellen mehr Speicherplatz. Diese Speicherplatzanforderungen gelten zusätzlich zu den sonstigen Anforderungen, sind jedoch unter Berücksichtigung der sich daraus ergebenden Vorteile sinnvoll.
- Wenn Sie die Tabellenpartitionierung und die MDC-Funktionalität in Ihrer partitionierten Datenbankumgebung nicht zusammen einsetzen wollen, dann sollten Sie die Tabellenpartitionierung in den Fällen einsetzen, in denen die Datenverteilung verlässlich vorausgesagt werden kann. Dies ist normalerweise bei den hier erläuterten Systemtypen der Fall. Andernfalls sollten Sie die Verwendung von MDC in Betracht ziehen.

Kapitel 2. 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.

Vorteile einer Struktur mit Bereichsclustertabellen

Die Verwendung von Bereichsclustertabellen bietet eine Reihe von Vorteilen.

- Direktzugriff

  Der Zugriff erfolgt über eine Funktion, die Bereichsclustertabellenschlüssel Satz-IDs (RIDs) zuordnet.
- 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 werden, und die Daten müssen eine dichte Clusterbildung aufweisen.
- Weniger Indizes

  Durch die Abbildung jedes Schlüssels auf eine Plattenposition kann die Tabelle mit einem Index weniger erstellt werden, als ansonsten erforderlich wären. Bei Bereichsclustertabellen ist aufgrund der Anwendungserfordernisse für den Zugriff auf die Daten in der Tabelle möglicherweise kein zweiter, separater Index mehr erforderlich. Sie können jedoch weiterhin reguläre Indizes erstellen, insbesondere wenn die Anwendung dies erforderlich macht.

Inkompatibilitäten von Bereichsclustertabellen

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:

- Bereichsclustertabellen für partitionierte Tabellen werden nicht unterstützt.

  Wenn Sie versuchen, eine partitionierte Tabelle mit Bereichsclustering zu erstellen, wird die Fehlernachricht SQL0270 mit Rückkehrcode (RC) 87 zurückgegeben.
- 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 Datenbank befinden, die mehr als eine Datenbankpartition enthält.
- 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.

Bereichsclustertabellen und Satzschlüsselwerte außerhalb des Bereichs

Die Funktionsweise einer Bereichsclustertabelle (RCT = Range-Clustered Table), in der Überlaufdatensätze erlaubt sind, kann mithilfe 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.

Sperren für Bereichsclustertabellen

Im Rahmen der normalen Verarbeitung werden Datensätze gesperrt, um sicherzustellen, dass jeweils nur eine Anwendung bzw. ein Benutzer Zugriff auf einen Datensatz bzw. eine Gruppe von Datensätzen hat. Bei Bereichsclustertabellen wird anstelle des Sperrens von Schlüsseln und nächsten Schlüsseln das „diskrete Sperren" 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 aktiviert werden, hängt von der gewählten Isolationsstufe ab.

Den Suchbedingungen entsprechende Zeilen in Bereichsclustertabellen, die momentan leer sind, jedoch vorab zugeordnet wurden, werden gesperrt. Dadurch wird das Sperren des nächsten Schlüssels überflüssig. Infolgedessen werden zur Erstellung einer Bereichsclustertabelle mit dichtem Clustering weniger Sperren benötigt.

Kapitel 3. Tabellen mit mehrdimensionalem Clustering (MDC)

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. MDC trägt bedeutend zur Verbesserung des Abfrageleistung bei.

Außerdem kann MDC den Aufwand der Datenpflege, wie beispielsweise die Neuorganisation und Indexpflegeoperationen bei INSERT-, UPDATE- und DELETE-Operationen, bedeutend verringern. MDC ist in erster Linie für Data-Warehouse-Umgebungen 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 vom Datenbankmanager lediglich ein eindimensionales Clustering von Daten über Clusterindizes unterstützt. Der Datenbankmanager versucht mithilfe 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 Clusterschlüsselwerte 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.

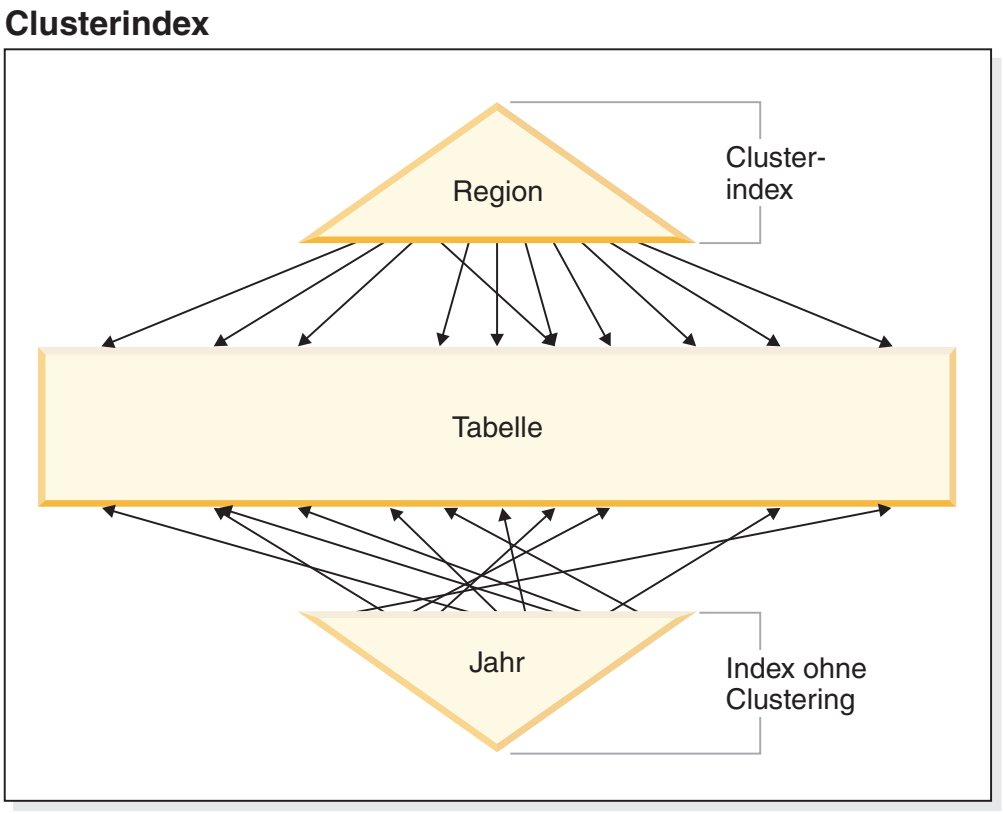


Abbildung 8. Eine reguläre Tabelle mit einem Clusterindex

Für die Tabelle in Abb. 8 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 Clusterschlüssel 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, in den meisten Fällen das Clustering über alle Dimensionen automatisch und kontinuierlich beibehalten und garantieren. Dadurch brauchen MDC-Tabellen zur Wiederherstellung der physischen Anordnung der Daten nicht häufig reorganisiert zu werden. Während die Reihenfolge der Datensätze innerhalb der Blöcke immer beibehalten wird, bleibt die physische Reihenfolge von Blöcken (d. h. von einem Block zum anderen bei einer Blockindexsuche) bei Einfügungen (oder manchmal nicht einmal beim anfänglichen Laden) erhalten.

Dritten sind bei MDC die Clusterindizes blockbasiert. Diese Indizes sind beträchtlich kleiner als die regulären datensatzbasierten Indizes, sodass sie wesentlich weniger Plattenspeicherplatz belegen und sich schneller durchsuchen lassen.

Auswählen von MDC-Tabellendimensionen

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. Informationen zu verschiedenen Aspekten des Entwurfs von MDC-Tabellen sowie Hilfestellung im Hinblick auf die Auswahl der geeigneten Dimensionen und Blockgrößen erhalten Sie über die Links unter **Zugehörige Konzepte**.

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, bieten 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 Sternjoins 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, sodass 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'
```

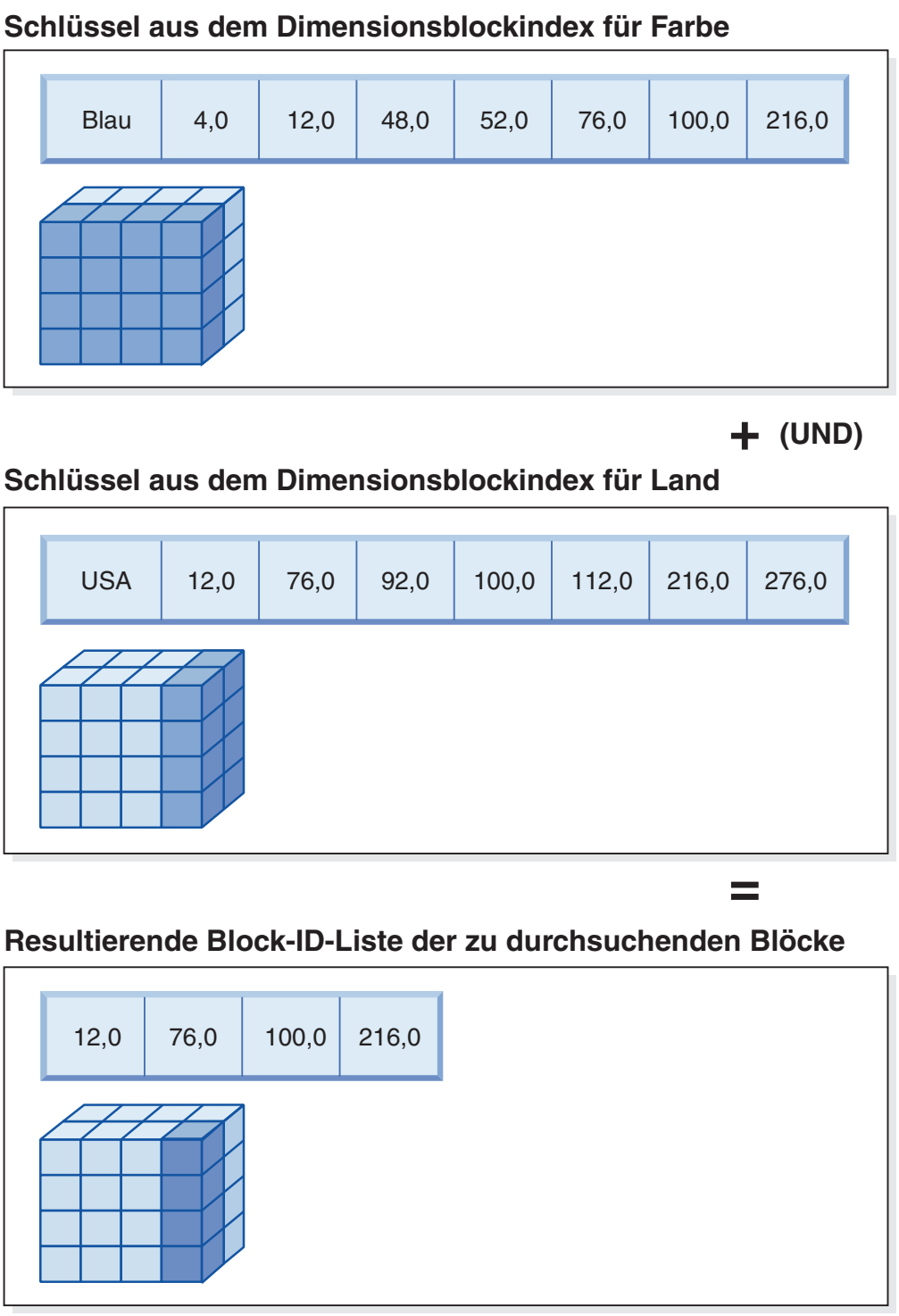


Abbildung 9. Eine Abfrageanforderung mit einer logischen UND-Operation und zwei Blockindizes

Diese Abfrageanforderung wird wie folgt ausgeführt (siehe Abb. 9 auf Seite 47):

- 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:

- Es werden Suchen in den Dimensionsblockindizes ausgeführt: eine für jeden Sektor.
- Eine logische ODER-Operation wird angewendet, um die Vereinigungsmenge (Union) 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 möglich:

```
SELECT * FROM MDCTABLE WHERE FARBE='BLAU' AND TEILENUMMER < 1000
```

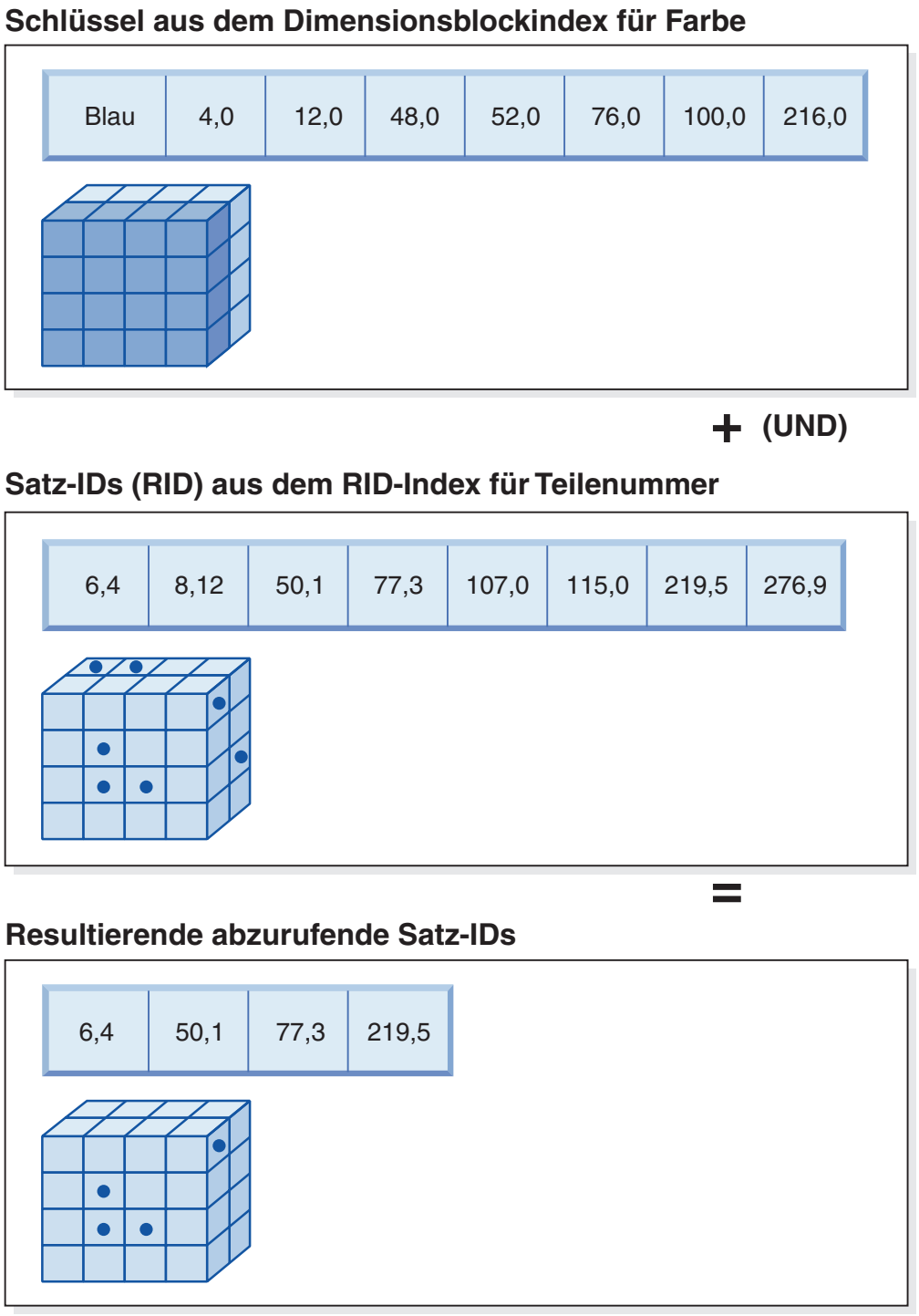


Abbildung 10. Eine Abfrageanforderung mit einer logischen UND-Operation sowie einem Blockindex und einem Satz-ID-Index

Diese Abfrageanforderung wird wie folgt ausgeführt (siehe Abb. 10 auf Seite 50):

- 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
```

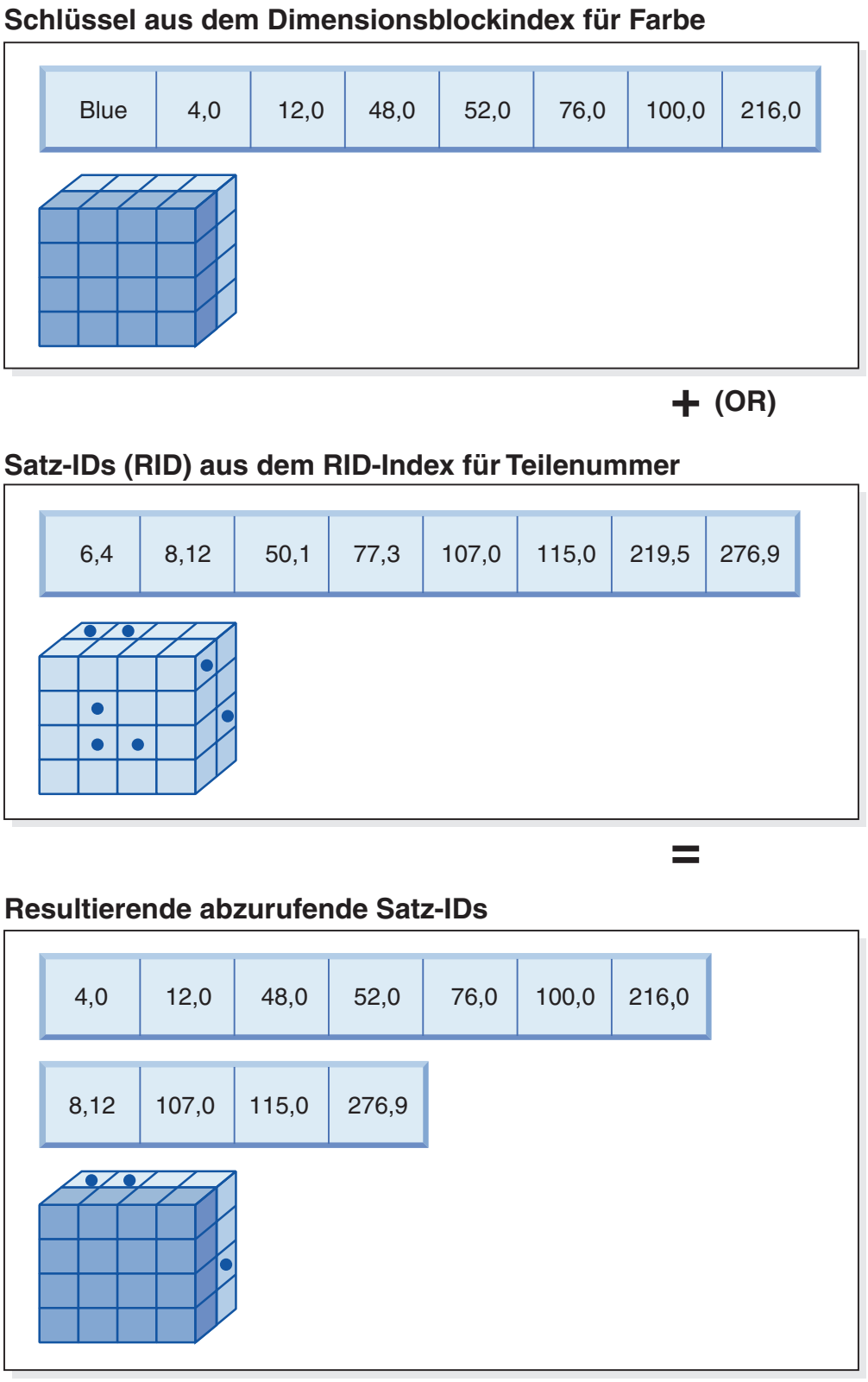


Abbildung 11. Funktionsweise eines Blockindex und eines Satz-ID-Index mit einer logischen ODER-Operation

Diese Abfrageanforderung wird wie folgt ausgeführt (siehe Abb. 11):

- 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 (Union) 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 einen Sternjoin. 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 mithilfe 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.

Zelldichte

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, sodass 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
   - Joinklauseln (insbesondere in Sternschema-Umgebungen)

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 Verteilungsschlüssel 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 Datenbankpartitionen 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 Datenbankpartition vorhanden sind, da Blöcke für Daten auf Datenbankpartitionsebene zugeordnet werden. Bei der Schätzung der Speicherplatzbelegung und der Dichte in einer Umgebung mit partitionierten Datenbanken müssen Sie die durchschnittliche Anzahl von Datensätzen pro Zelle in jeder Datenbankpartition, 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, sodass 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, sodass 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 Database für Linux, UNIX und Windows mithilfe 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 be-

stimmen. 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, mithilfe einer benutzerdefinierten Funktion auf die Regionen 'West', 'Nord', 'Süd' und 'Ost' hochgestuft werden.

Aspekte der Erstellung von MDC-Tabellen

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 einigen dieser Aspekte behandelt.

Versetzen von Daten aus vorhandenen Tabellen in MDC-Tabellen

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 (mithilfe 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, ist eine Verwendung der mehrseitigen Dateizuordnung erforderlich. (Die mehrseitige Dateizuordnung ist ab Version 8.2 die Standardeinstellung für neu erstellte Daten-

banken.) Dies 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.

Eine Voraussetzung für die Verwendung der MDC-Funktion des DB2-Designadvisors ist das Vorhandensein mindestens mehrerer EXTENTSIZE großer Speicherbereiche mit Daten innerhalb der Datenbank. Der DB2-Designadvisor verwendet die Daten zur Modellierung der Datendichte und Kardinalität.

Wenn die Datenbank keine Daten in den Tabellen enthält, empfiehlt der DB2-Designadvisor kein MDC, selbst wenn die Datenbank zwar leere Tabellen enthält, jedoch über nachgebildete Statistikdaten verfügt, die eine mit Daten gefüllte Datenbank simulieren.

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, sodass 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 be-

steht darin, das Anwachsen des Speicherbedarfs der Datenbank auf einer moderaten Ebene zu halten. Das maximale Speicherwachstum jeder Tabelle wird mithilfe 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 MQTs (Materialized Query Tables, gespeicherte Abfragetabellen), „C" für MDC und „P" für eine Umgebung mit partitionierten Datenbanken. 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 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 Umgebungen mit partitionierten Datenbanken

Das mehrdimensionale Clustering (MDC) kann in Verbindung mit einer Umgebung mit partitionierten Datenbanken verwendet werden. Tatsächlich kann MDC eine Umgebung mit partitionierten Datenbanken ergänzen. Eine Umgebung mit partitionierten Datenbanken 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 Datenbankpartition hinaus
- Verbesserung der Skalierbarkeit der Datenbank

Der Grund zur Verteilung 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 Verteilungsschlüssel nicht. Der Verteilungsschlüssel 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 Verteilungsschlüssel 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 Datenbankpar-

titionen verteilt wäre, würde die Spalte 'Farbe' zur Unterteilung der Daten verwendet. Infolgedessen können zum Beispiel die Sektoren 'Rot' und 'Blau' in einer Datenbankpartition und der Sektor 'Gelb' in einer anderen Datenbankpartition abgelegt werden. Wenn der Verteilungsschlüssel 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 Datenbankpartitionen bestimmen lässt.

MDC-Tabellen mit mehreren Dimensionen

Wenn Sie wissen, dass bestimmte Vergleichselemente in Abfragen sehr häufig verwendet werden, können Sie die Tabelle mithilfe 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, sodass 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, sodass 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, sodass Abfragen, die nur die Spalte c4 betreffen, mehr Verarbeitungsaufwand erfordern. Allerdings ist in Beispiel 2 ein 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 geographische 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 Dimensionen auf generierten Spalten

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 die 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 die Dimension (monat) können Blockindexsuchen ausgeführt werden. Für Abfragen auf die 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 DATE > '1999-03-03' AND DATE < '2000-01-15'
```

Der Compiler generiert die folgenden zusätzlichen Vergleichselemente: „month >= 199903" und „month <= 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('01/05/1999')) = 1
MONTH(date('02/08/1999')) = 2
MONTH(date('03/24/1999')) = 3
MONTH(date('04/30/1999')) = 4
...
MONTH(date('12/09/1999')) = 12
MONTH(date('01/18/2000')) = 1
MONTH(date('02/24/2000')) = 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. Version 9.1 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).

Der Datenbankmanager 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 Monotonie bewahrend erkannt werden, wie zum Beispiel DATENUM( ), DAYS( ), YEAR( ). Außerdem sind verschiedene mathematische Ausdrücke wie Division, Multiplikation oder Addition einer Spalte und einer Kon-

stante Monotonie bewahrend. In Fällen, in denen DB2 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.

Ladeaspekte für 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. In Version 9.5 wurde die MDC-Rolloutfunktion (Auslagerung) mit verzögerter Bereinigung hinzugefügt. Wenn der Rollout, bei dem es sich auch um ein Löschen handelt, festgeschrieben ist, sind die Blöcke nicht frei und können noch nicht wiederverwendet werden. Ein Hintergrundprozess wird aufgerufen, um die Satz-ID-Indizes (RID-Indizes) zu pflegen. Wenn die Pflegeoperation abgeschlossen ist, werden die Blöcke freigegeben und können wiederverwendet werden.

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_sz*.

  Um eine bessere Leistung des Dienstprogramms LOAD beim Laden von MDC-Tabellen zu erzielen, sollte der Wert des Datenbankkonfigurationsparameters *util_heap_sz* erhöht werden. Die Leistung des Algorithmus 'mdc-load' ist deutlich besser, wenn dem Dienstprogramm mehr Speicher zur Verfügung steht. Dies verringert die Platten-E/A während des Datenclusterings, das in der LOAD-Phase stattfindet. Falls der Befehl LOAD eingesetzt wird, um mehrere MDC-Tabellen gleichzeitig zu laden, sollte der Wert von *util_heap_sz* entsprechend erhöht werden.
- 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. Ab Version 9.5 kann der Wert der Option DATA BUFFER des Befehls LOAD die Zwischenspeichergröße *util_heap_sz* vorübergehend überschreiten, wenn mehr Systemspeicher verfügbar ist.
- Stellen Sie sicher, dass die für den Pufferpool verwendete Seitengröße mit der größten Seitengröße für den temporären Tabellenbereich übereinstimmt.

  Während der LOAD-Phase werden zusätzliche Protokollierungen zur Verwaltung der Blockzuordnung ausgeführt. Pro zugeordnetem Speicherbereich werden ca. zwei zusätzliche Protokollsätze erstellt. Um eine zufriedenstellende Leistung zu gewährleisten, sollte der Datenbankkonfigurationsparameter *logbufsz* auf einen Wert gesetzt werden, der diesen Umstand berücksichtigt.

Für das Laden von Daten in MDC-Tabellen (Multi-Dimensional Clustering - mehrdimensionales Clustering) gelten die folgenden Einschränkungen:

- Die Option SAVECOUNT des Befehls LOAD wird nicht unterstützt.
- Der Änderungswert `totalfreespace` für den Dateityp wird nicht unterstützt, da diese Tabellen ihren freien Speicherbereich selbst verwalten.

- Der Änderungswert `anyorder` für den Dateityp ist für MDC-Tabellen erforderlich. Werden Daten ohne den Änderungswert `anyorder` in eine MDC-Tabelle geladen, wird dieser Änderungswert vom Dienstprogramm explizit aktiviert.

Bei Verwendung des Befehls LOAD mit einer MDC-Tabelle werden Verstöße gegen eindeutige Integritätsbedingungen wie folgt behandelt:

- Wenn die Tabelle vor der Ladeoperation einen eindeutigen Schlüssel enthielt und doppelte Datensätze in die Tabelle geladen werden, verbleibt der ursprüngliche Datensatz in der Tabelle, und die neuen Datensätze werden während der DELETE-Phase gelöscht.
- Enthielt die Tabelle vor der Ladeoperation keinen eindeutigen Schlüssel und werden sowohl ein eindeutiger Schlüssel als auch doppelte Datensätze in die Tabelle geladen, wird nur einer der Datensätze mit dem eindeutigen Schlüssel geladen, und die anderen Datensätze werden während der DELETE-Phase gelöscht.

  Anmerkung: Es gibt keine explizite Methode, mit der bestimmt werden kann, welche Datensätze geladen und welche gelöscht werden.

Das Laden beginnt an einer Blockgrenze, sodass sich Ladeoperationen am besten für Daten, die zu neuen Zellen gehören, oder zum ersten Füllen einer Tabelle mit Daten eignen.

MDC-Ladeoperationen enthalten immer eine BUILD-Phase, da alle MDC-Tabellen mit Blockindizes ausgestattet sind.

Protokollierungsaspekte für MDC-Tabellen

In Fällen, in denen zuvor bzw. auf andere Weise durch Satz-ID-Indizes indizierte Spalten nun Dimensionen sind und daher durch Blockindizes indiziert werden, verringert sich der Aufwand für Pflege und Protokollierung der Indizes erheblich.

Nur wenn der letzte Datensatz in einem gesamten Block gelöscht wird, muss der Datenbankmanager die BID aus den Blockindizes entfernen und diese Indexoperation protokollieren. Analog braucht der Datenbankmanager 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. Bei Rolloutlöschungen werden die gelöschten Datensätze nicht protokolliert. Stattdessen werden die Seiten mit den Sätzen durch eine Neuformatierung von Teilen der Seiten optisch geleert. Die Änderungen an den neu formatierten Teilen werden protokolliert, die Sätze selbst werden allerdings nicht protokolliert.

Blockindexaspekte 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 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 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 (c4,c3,c1,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)
```

Blockindizes für MDC-Tabellen

Dieser Abschnitt erläutert, wie Datensätze in MDC-Tabellen mithilfe von Blockindizes organisiert werden.

Die MDC-Tabelle in Abb. 12 auf Seite 65 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, sodass 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, sodass 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 die-

se Datensätze zusammengefasst auf sequenziell angeordneten Seiten innerhalb der einzelnen Blöcke.

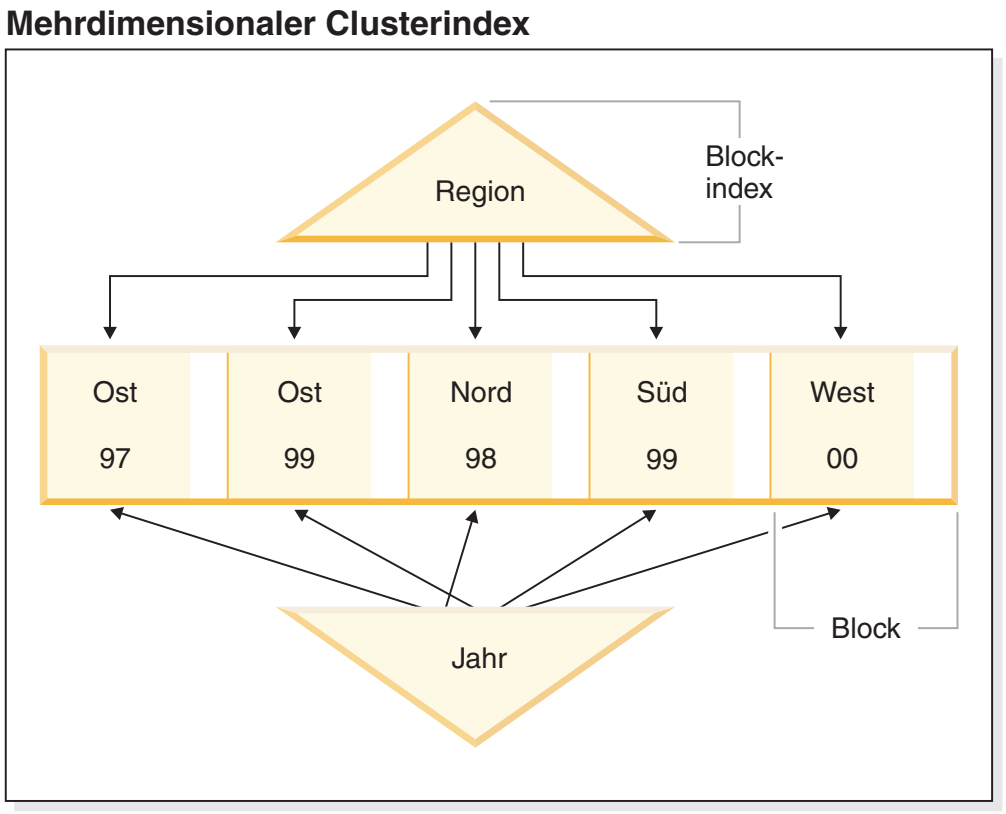


Abbildung 12. Eine Tabelle mit mehrdimensionalem Clustering (MDC)

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 Datenbankpartitionen" 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. Weitere Informationen finden Sie in „Auswählen von MDC-Tabellendimensionen" auf Seite 45 und *Designadvisor*.

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 in „Blockindexaspekte für MDC-Tabellen" auf Seite 63.

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.

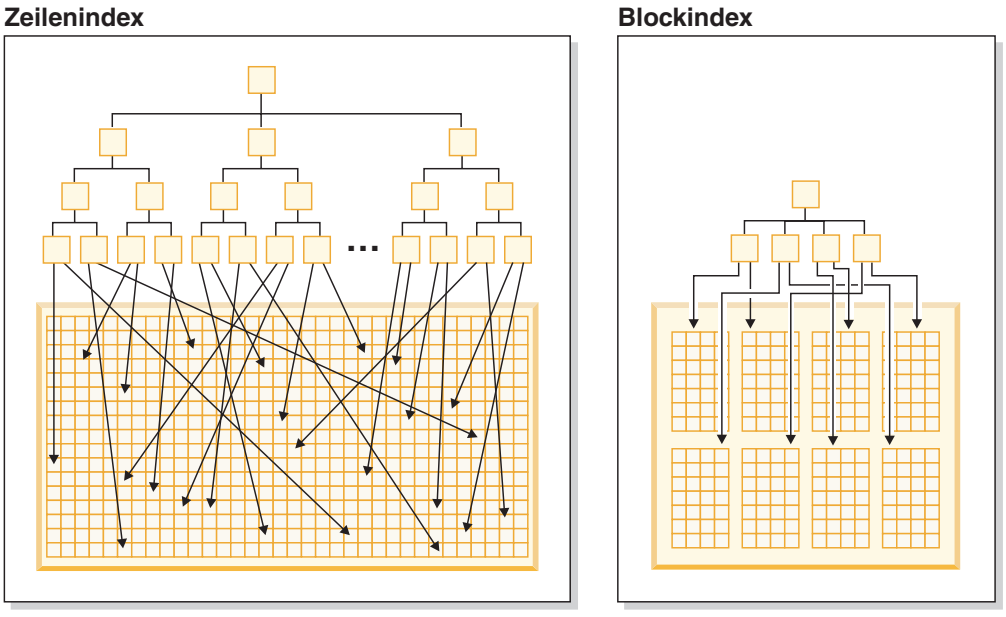


Abbildung 13. Unterschiede zwischen Zeilenindizes und Blockindizes

Wie in Abb. 13 auf Seite 66 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.

Szenario: MDC-Tabellen

Als Szenario für die Arbeit mit einer MDC-Tabelle stellen Sie sich 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. 14 auf Seite 68 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.

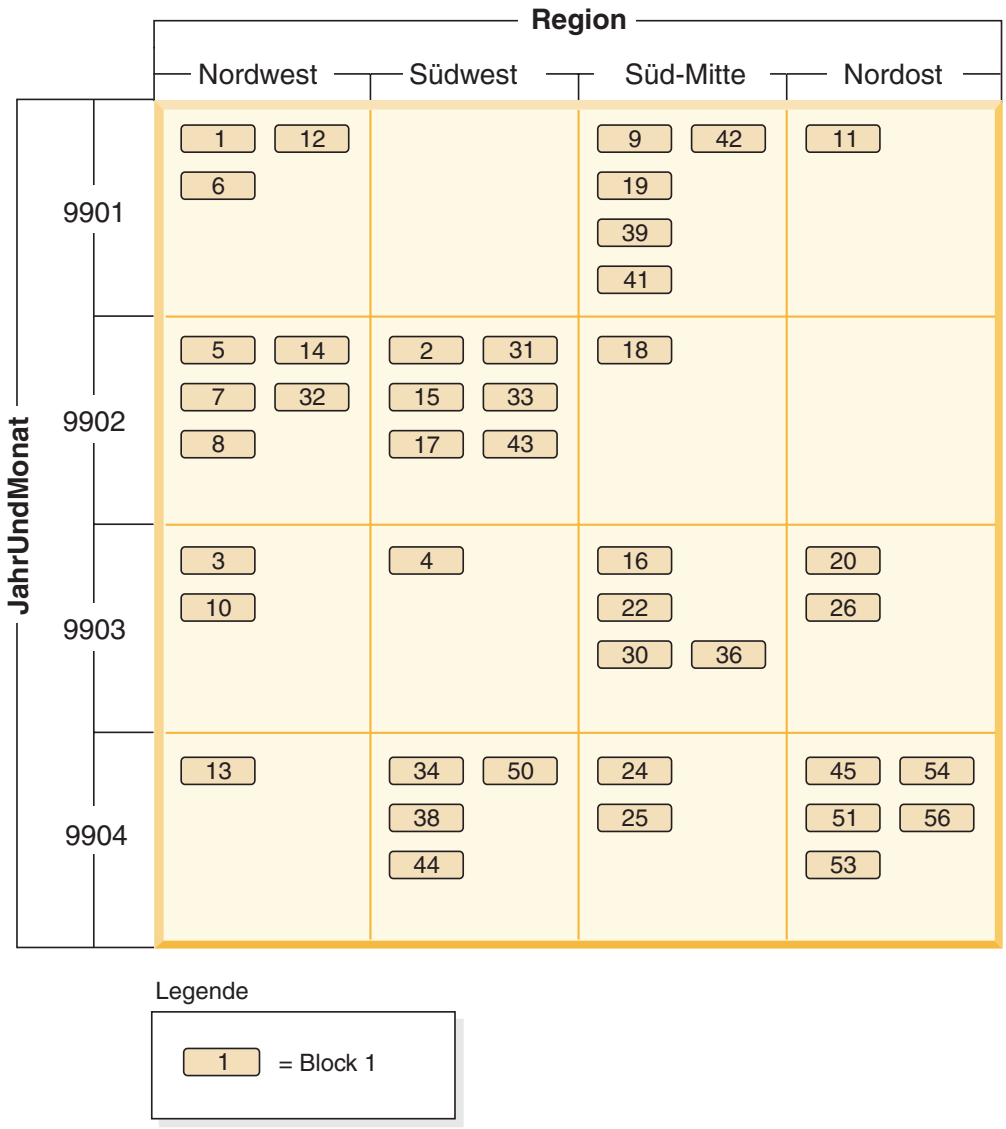


Abbildung 14. Mehrdimensionale Tabelle mit den Dimensionen 'Region' und 'JahrUndMonat' mit dem Namen 'Umsatz'

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. 15 auf Seite 69 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 mithilfe 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.

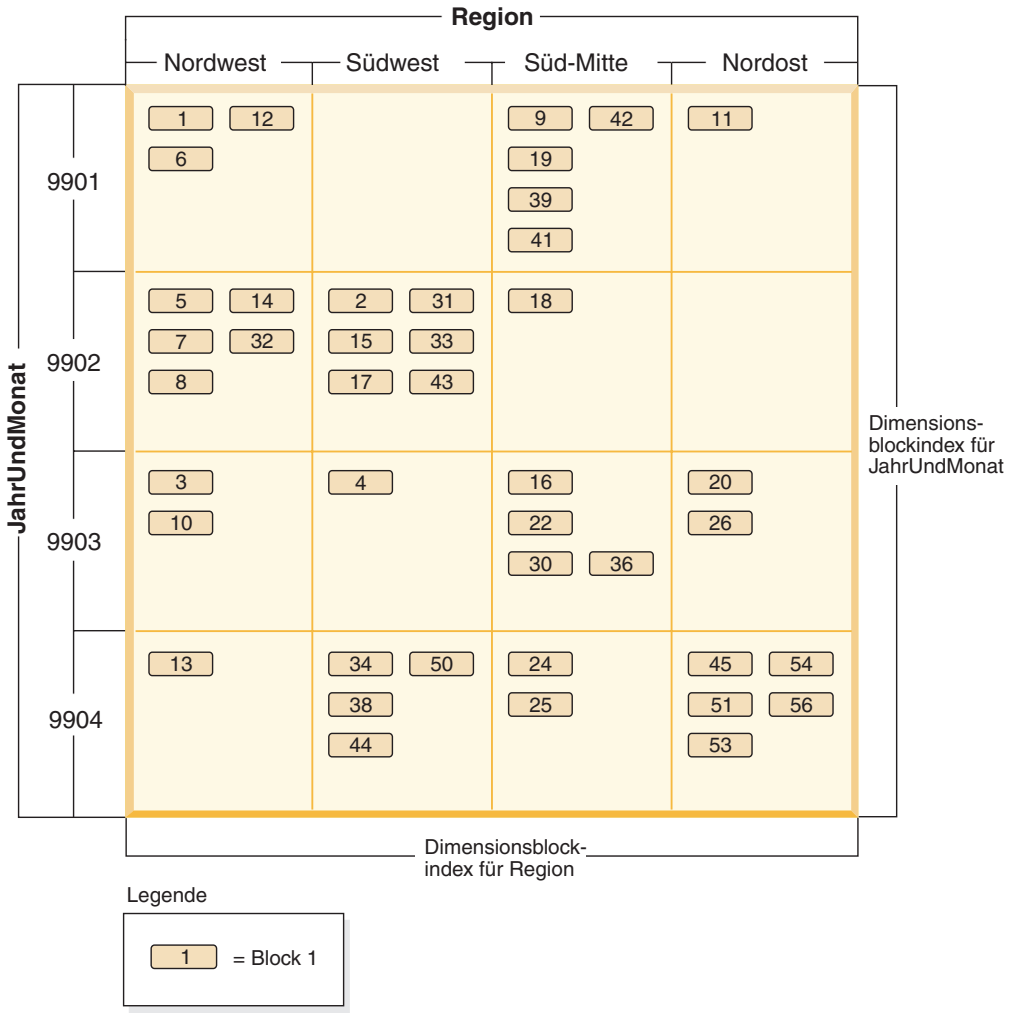


Abbildung 15. Tabelle 'Umsatz' mit den Dimensionen 'Region' und 'JahrUndMonat' mit gezeigten Dimensionsblockindizes

Abb. 16 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. 16 stimmen die aufgeführten Blocknummern mit denen überein, die sich im Sektor 'Süd-Mitte' im Gitter der Umsatztabelle befinden (siehe Abb. 14 auf Seite 68).

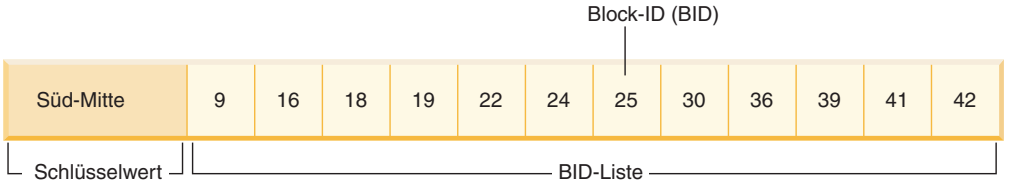


Abbildung 16. Schlüssel aus dem Dimensionsblockindex für 'Region'

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. 17 gezeigt wird.

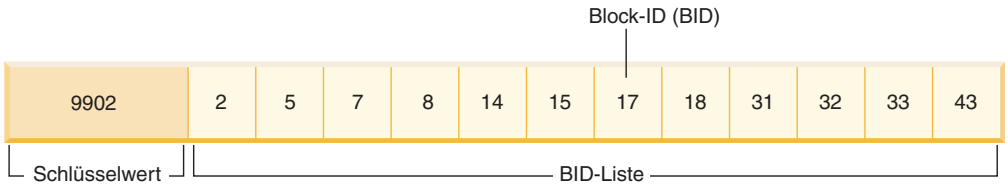


Abbildung 17. Schlüssel aus dem Dimensionsblockindex für 'JahrUndMonat'

Blockindizes und Abfrageleistung für MDC-Tabellen

Suchen in einem der Blockindizes einer MDC-Tabelle ermöglichen einen Clusterdatenzugriff, da jede Block-ID (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 durchsucht den Index vorausschauend (Look-Ahead) und liest die Datenseiten von Blöcken in den Speicher durch E/A-Operationen in großen Blöcken ein, sodass 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 mithilfe 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 „Datenbankpartitionsausschlusses", 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. 18 veranschaulicht.

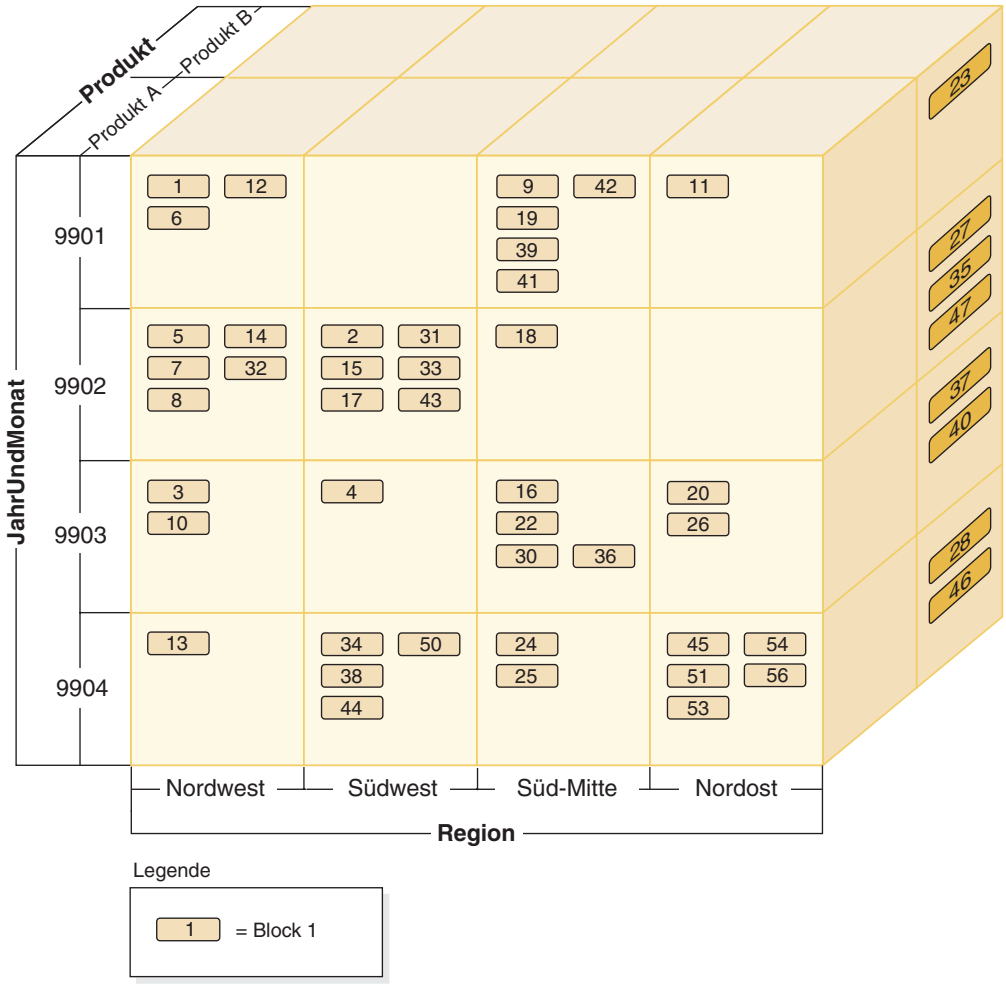


Abbildung 18. Mehrdimensionale Tabelle mit den Dimensionen 'Region', 'JahrUndMonat' und 'Produkt'

Für die MDC-Tabelle in Abb. 18 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 der Datenbankmanager zunächst nach dem Schlüssel für Produkt A im Dimensionsblockindex für „Produkt" suchen. (Siehe Abb. 19.) Anschließend bestimmt der Datenbankmanager durch Suchen des Schlüssels „Nordost" im Dimensionsblockindex „Region" die Blöcke, die alle Datensätze mit „Region" gleich „Nordost" enthalten. (Siehe Abb. 20 auf Seite 72.)

Produkt A	1	2	3	...	11	...	20	22	24	25	26	30	...	56

Abbildung 19. Schlüssel aus dem Dimensionsblockindex für 'Produkt'

Nordost	11	20	23	26	27	28	35	37	40	45	46	47	51	53	54	56

Abbildung 20. 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 mithilfe der logischen ODER-Operation für Blockindizes eine Abfrage mit Vergleichselementen erfüllt wird, die sich auf zwei Dimensionen beziehen. In Abb. 21 auf Seite 73 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.

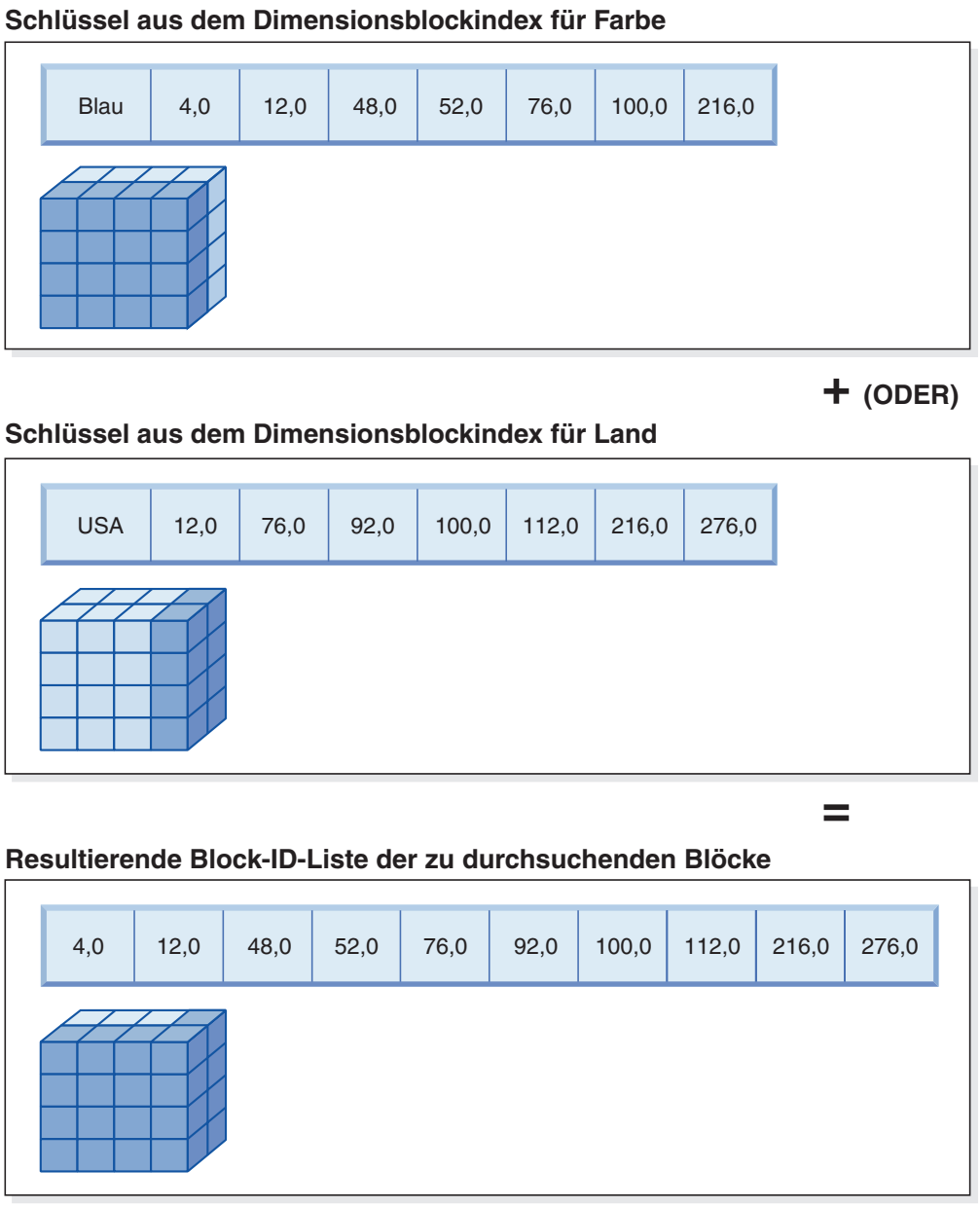


Abbildung 21. Verwendung der logischen ODER-Operation für Blockindizes

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. (Die Zahlen geben Satz-IDs (RIDs), Bereichsfelder an.)

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 (Union) 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 der Datenbankmanager eine Liste von Blöcken zum Durchsuchen ermittelt hat, kann er 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 der Datenbankmanager 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, Joins 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. 22 auf Seite 75.) 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.

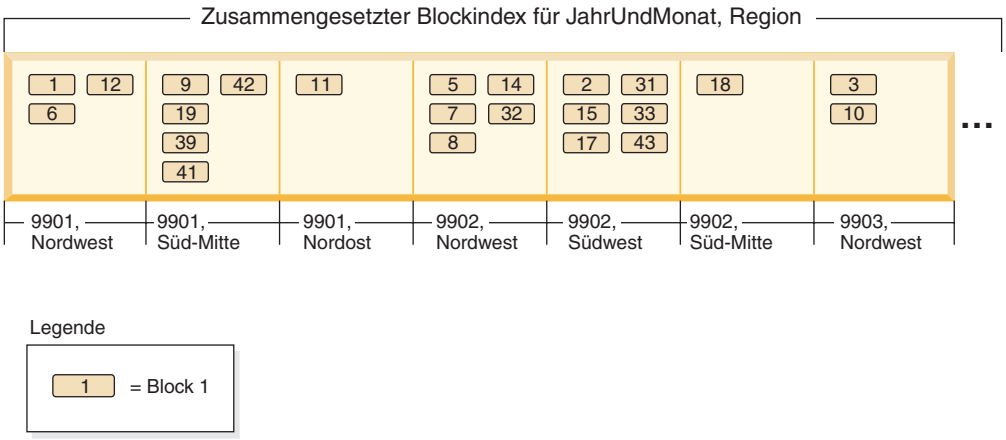


Abbildung 22. Zusammengesetzter Blockindex für 'JahrUndMonat', 'Region'

Datensätze mit bestimmten Dimensionswerten werden 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, sodass 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 in Abb. 22 abrufen wollen, die den Wert 'Nordwest' für „Region” und den Wert '9903' für „JahrUndMonat” haben, würde der Datenbankmanager den Schlüsselwert '9903, Nordwest' im zusammengesetzten Blockindex aufsuchen, wie in Abb. 23 auf Seite 76 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.

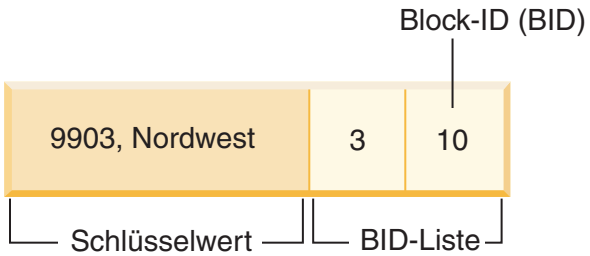


Abbildung 23. Schlüssel aus dem zusammengesetzten Blockindex für 'JahrUndMonat', 'Region'

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. Der Datenbankmanager würde diesen Schlüsselwert im zusammengesetzten Blockindex suchen und die BIDs für die Blöcke 3 und 10 finden. Diese Blöcke enthalten alle Datensätze und die einzigen Datensätze, die diese Dimensionsschlüsselwerte besitzen. Wenn Speicherplatz verfügbar ist, fügt der Datenbankmanager den neuen Datensatz in einen dieser Blöcke ein. Wenn kein Platz in den Seiten dieser Blöcke mehr vorhanden ist, ordnet der Datenbankmanager 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. Der Datenbankmanager fügt den Datensatz in den Block ein und ordnet diesen Block der aktuellen logischen Zelle zu, indem er die Block-ID (BID) des Blocks dem zusammengesetzten Blockindex sowie den einzelnen Dimensionsblockindizes hinzufügt. Abb. 24 stellt die Schlüssel der Dimensionsblockindizes nach dem Hinzufügen von Block 48 dar.

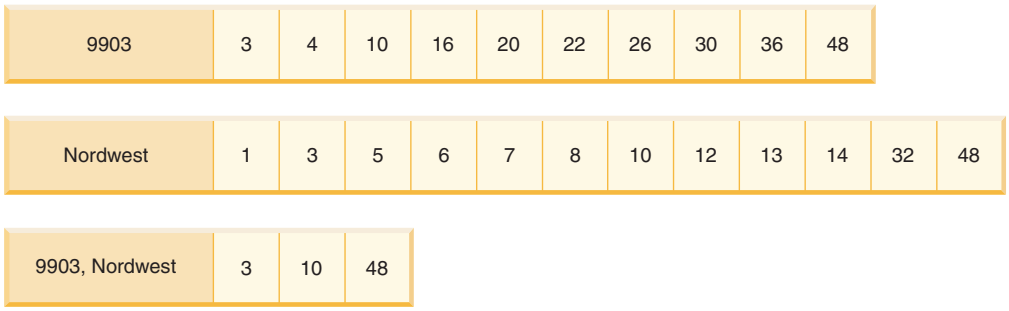


Abbildung 24. Schlüssel aus den Dimensionsblockindizes nach Hinzufügen von Block 48

Blockzuordnungen für MDC-Tabellen

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 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.

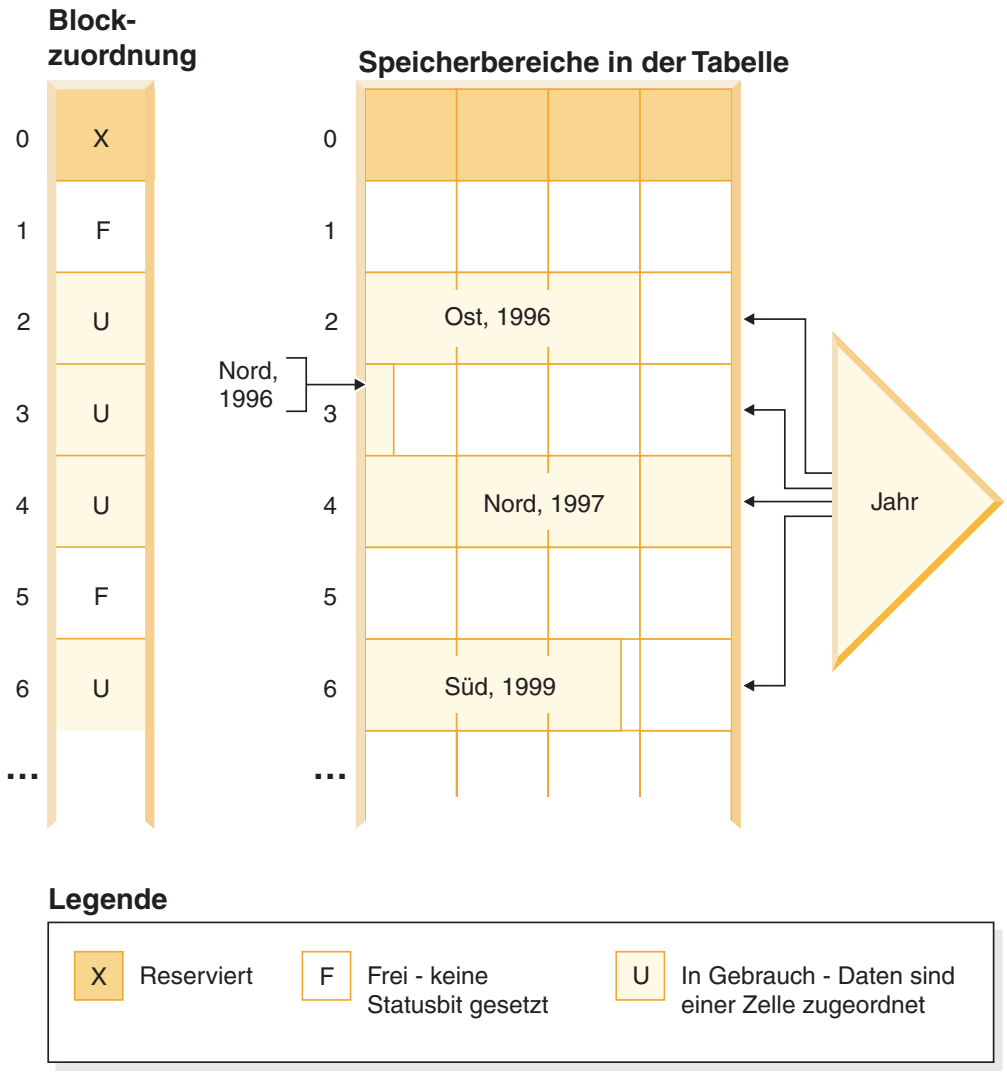


Abbildung 25. Funktionsweise einer Blockzuordnung

Die linke Seite in Abb. 25 zeigt die Blockzuordnungsfeldgruppe mit verschiedenen Einträgen für die Blöcke in der Tabelle. Die rechte Seite 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 in Gebrauch markiert sind. Aus Gründen der Übersichtlichkeit zeigt die Abbildung nur einen der beiden Dimensionsblockindizes.

Anmerkung:

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. 25 auf Seite 77) 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.

Löschen aus MDC-Tabellen

Wenn ein Datensatz in einer MDC-Tabelle gelöscht wird, löscht der Datenbankmanager 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 der Datenbankmanager den Block frei, indem er das Statusbit IN_USE ('In Gebrauch') ändert und die Block-ID (BID) aus allen Blockindizes entfernt. 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 an 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 Lö-

schung 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 Trigger, referenzielle Integritätsbedingungen, Sichten und MQTs (MQT = Materialized Query Table, gespeicherte Abfragetabellen) definiert werden können.

Tabellenpartitionierung und MDC-Tabellen

Bei einer Tabelle kann es sich sowohl um eine MDC-Tabelle (Tabelle mit mehrdimensionalem Clustering) als auch um eine partitionierte Tabelle handeln. In einer Tabelle, die beide Merkmale aufweist, können Spalten sowohl in der Bereichspartitionsspezifikation (Range-Partition-Spec) als auch im MDC-Schlüssel für die Tabellenpartitionierung verwendet werden. Diese Möglichkeit ermöglicht eine exaktere Differenzierung des Datenpartitions- und Blockausschlusses, als dies bei Verwendung nur einer dieser Funktionalitäten der Fall wäre. Es gibt auch zahlreiche Anwendungen, in denen es sinnvoll ist, für den MDC-Schlüssel andere Spalten als die für die Tabellenpartitionierung genutzten Spalten anzugeben. Hierbei ist zu beachten, dass die Tabellenpartitionierung mit mehreren Spalten arbeitet, während beim MDC mit mehreren Dimensionen gearbeitet wird.

Merkmale eines normalen DB2 Data Warehouse der Version 9.1

Die folgenden Empfehlungen beziehen sich auf typische, normale Data Warehouses, die in DB2 Version 9.1 neu implementiert wurden. Die folgenden Merkmale werden angenommen:

- Die Datenbank arbeitet auf mehreren Systemen oder in mehreren logischen AIX-Partitionen.
- Die Datenbankpartitionierungsfunktion (DPF) wird verwendet (Tabellen werden mithilfe der Klausel DISTRIBUTE BY HASH erstellt).
- Es wurden zwischen vier und 50 Datenpartitionen definiert.
- Die Tabelle, für die die Möglichkeit zur Verwendung des MDC und der Tabellenpartitionierung geprüft wird, ist eine der zentralen Fakttabellen.
- Die Tabelle umfasst zwischen 100 Millionen und 100 Milliarden Zeilen.
- Neue Daten werden in unterschiedlichen Zeitrahmen geladen: Jeweils über Nacht, wöchentlich, monatlich.
- Das tägliche Aufnahmevolumen liegt zwischen 10.000 und 10 Millionen Datensätzen.
- Die Datenvolumen schwanken: Hierbei liegt das Volumen des Spitzenmonats um das Fünffache höher als das des Monats mit dem geringsten Datenaufkommen. Entsprechend beläuft sich auch der Umfang der größten Dimensionen (Produktlinie, Bereich) auf das Fünffache des Umfangs der kleinsten Dimensionen.
- Die Datenbank enthält detaillierte Datenbestände der letzten 1 - 5 Jahre.

- Abgelaufene Daten werden in monatlichen oder vierteljährlichen Abständen mit einer Rollout-Operation ausgelagert.
- Tabellen verwenden eine Vielzahl von Abfragetypen. Der Workload besteht jedoch größtenteils aus analytischen Abfragen, die in Bezug auf OLTP-Workloads die folgenden Merkmale aufweisen:
  - Es gibt umfangreichere Ergebnismengen mit bis zu 2 Millionen Zeilen.
  - Die Mehrzahl oder alle Abfragen beziehen sich auf Sichten und nicht auf Basistabellen.
- SQL-Klauseln zur Auswahl von Daten nach Bereichen (Klausel BETWEEN), Elementen in Listen etc.

Merkmale einer Fakttabelle eines normalen DB2 Data Warehouse der Version 9.1

Eine normale Data Warehouse-Fakttabelle kann z. B. das folgende Design verwenden:

- Erstellung von Datenpartitionen über die Spalte 'Month'.
- Definition einer Datenpartition für jeden Rollout-Zeitraum, z. B. 1 Monat, 3 Monate.
- Erstellung von MDC-Dimensionen für 'Day' und für 1 - 4 weitere Dimensionen. Typische Dimensionen sind: Produktlinie und Bereich.
- Alle Datenpartitionen und MDC-Cluster sind über alle Datenpartitionen verteilt.

Das MDC und die Tabellenpartitionierung bieten teilweise dieselben Vorteile. Die folgende Tabelle enthält mögliche Anforderungen innerhalb Ihres Unternehmens und Empfehlungen zu einem Organisationsschema, die auf der Basis der zuvor festgestellten Merkmale gegeben werden.

Tabelle 7. Verwendung der Tabellenpartitionierung mit MDC-Tabellen

Problemstellung	Empfohlenes Schema	Empfehlung
Datenverfügbarkeit während des Rollouts	Tabellenpartitionierung	Sie können die Klausel DETACH PARTITION verwenden, um ein Rollout großer Datenmengen unter minimaler Beeinträchtigung durchzuführen.
Abfrageleistung	Tabellenpartitionierung und MDC	MDC eignet sich am besten für die Abfrage mehrerer Dimensionen. Die Tabellenpartitionierung assistiert durch den Ausschluss von Datenpartitionen.
Minimale Reorganisation	MDC	MDC-Tabellen behalten das Clustering bei, sodass sich die Notwendigkeit von Reorganisationen verringert.
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums während eines traditionellen Offlinezeitfensters	Tabellenpartitionierung	Die Datenpartitionierung kann diese Anforderung voll erfüllen. Durch das MDC würden sich keine zusätzlichen Vorteile ergeben und dieses Verfahren wäre weniger geeignet.

Tabelle 7. Verwendung der Tabellenpartitionierung mit MDC-Tabellen (Forts.)

Problemstellung	Empfohlenes Schema	Empfehlung
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums während eines Mikro-Offlinezeitfensters (weniger als 1 Minute)	Tabellenpartitionierung	Die Datenpartitionierung kann diese Anforderung voll erfüllen. Durch das MDC würden sich keine zusätzlichen Vorteile ergeben und dieses Verfahren wäre weniger geeignet.
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums bei gleichzeitiger Erhaltung der Tabellenverfügbarkeit für Geschäftsbenutzer, die Abfragen ohne Serviceverluste weiterhin übergeben können	MDC	MDC kann nicht alle in diesem Zusammenhang geltenden Anforderungen voll erfüllen. Die Tabellenpartitionierung eignet sich hier nicht, da die Tabelle nur für kurze Zeit in den Offlinemodus versetzt wird.
Tägliches Laden von Daten (entweder ALLOW READ ACCESS oder ALLOW NO ACCESS)	Tabellenpartitionierung und MDC	MDC bietet hier die meisten Vorteile. Die Tabellenpartitionierung bietet Vorteile in Teilbereichen.
Kontinuierliches Laden von Daten (ALLOW READ ACCESS)	Tabellenpartitionierung und MDC	MDC bietet hier die meisten Vorteile. Die Tabellenpartitionierung bietet Vorteile in Teilbereichen.
Leistung bei der Ausführung von Abfragen für traditionelle BI-Abfragen	Tabellenpartitionierung und MDC	MDC eignet sich besonders gut für Abfragen in Datenkuben und mehreren Dimensionen. Die Tabellenpartitionierung bietet Unterstützung über den Partitionsausschluss.
Minimierung des Reorganisationsaufwandes durch Vermeidung des Reorganisationsbedarfs oder Reduzierung des Aufwands für die Ausführung der Task	MDC	Beim MDC wird das Clustering beibehalten, sodass sich die Notwendigkeit von Reorganisationen verringert. Wenn das MDC verwendet wird, dann bietet die Datenpartitionierung auch in Teilbereichen keine Vorteile. Allerdings ermöglicht die Bereichspartitionierung bei Nichtverwendung des MDC die Reduzierung des Reorganisationsbedarfs, indem auf Partitionsebene ein etwas groberes Clustering beibehalten wird.

Beispiel 1:

Sie arbeiten mit einer Tabelle mit den Schlüsselspalten 'YearAndMonth' und 'Province'. Bei der Planung dieser Tabelle wäre eine Partitionierung nach Datum mit einer Zeitspanne von zwei Monaten pro Datenpartition sinnvoll. Darüber hinaus können Sie die Daten auch nach der Spalte 'Province' organisieren, sodass alle

Zeilen für ein bestimmtes Gebiet innerhalb eines Datumsbereichs von zwei Monaten in einer Gruppe zusammengefasst werden. Diese Vorgehensweise ist in Abb. 6 auf Seite 37 dargestellt.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (Province);
```

Tabelle 'orders'

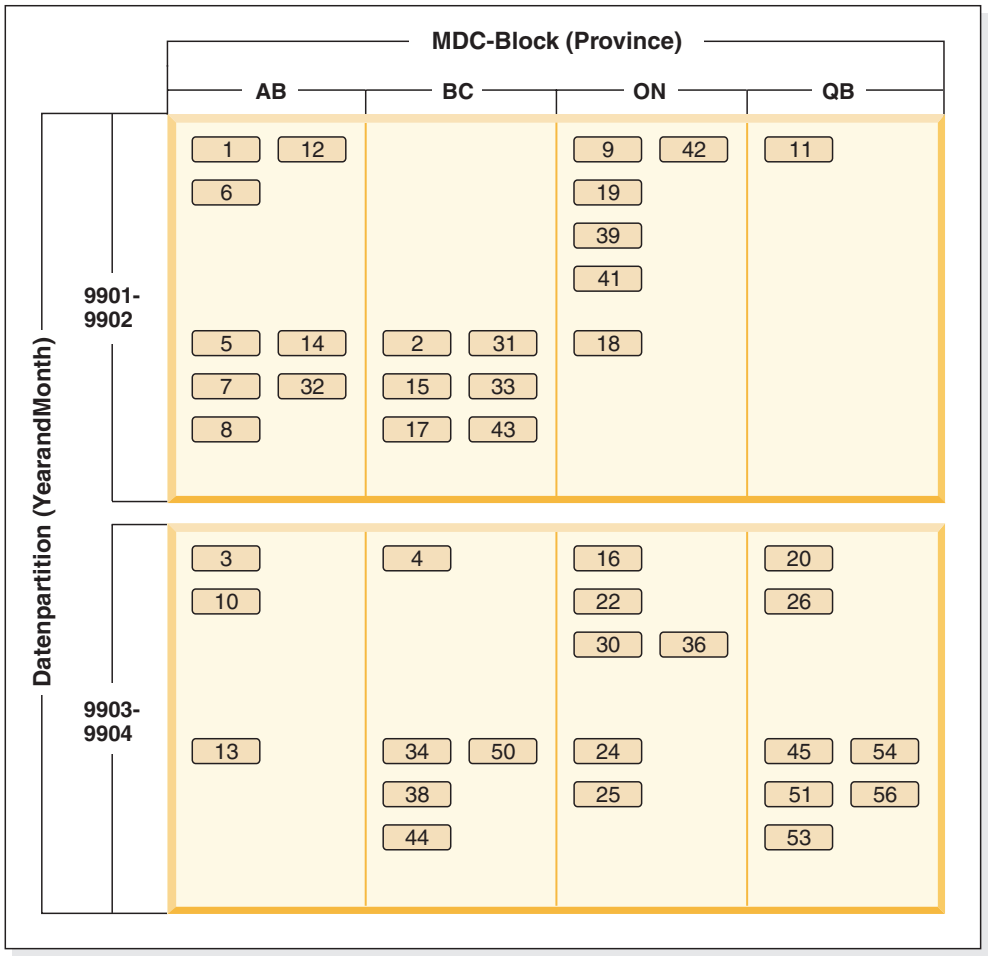


Abbildung 26. Eine nach 'YearAndMonth' partitionierte und nach 'Province' organisierte Tabelle

Beispiel 2:

Eine exaktere Differenzierung kann durch Hinzufügen von 'YearAndMonth' zur Klausel ORGANIZE BY (siehe hierzu Abb. 7 auf Seite 38) erzielt werden.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (YearAndMonth, Province);
```

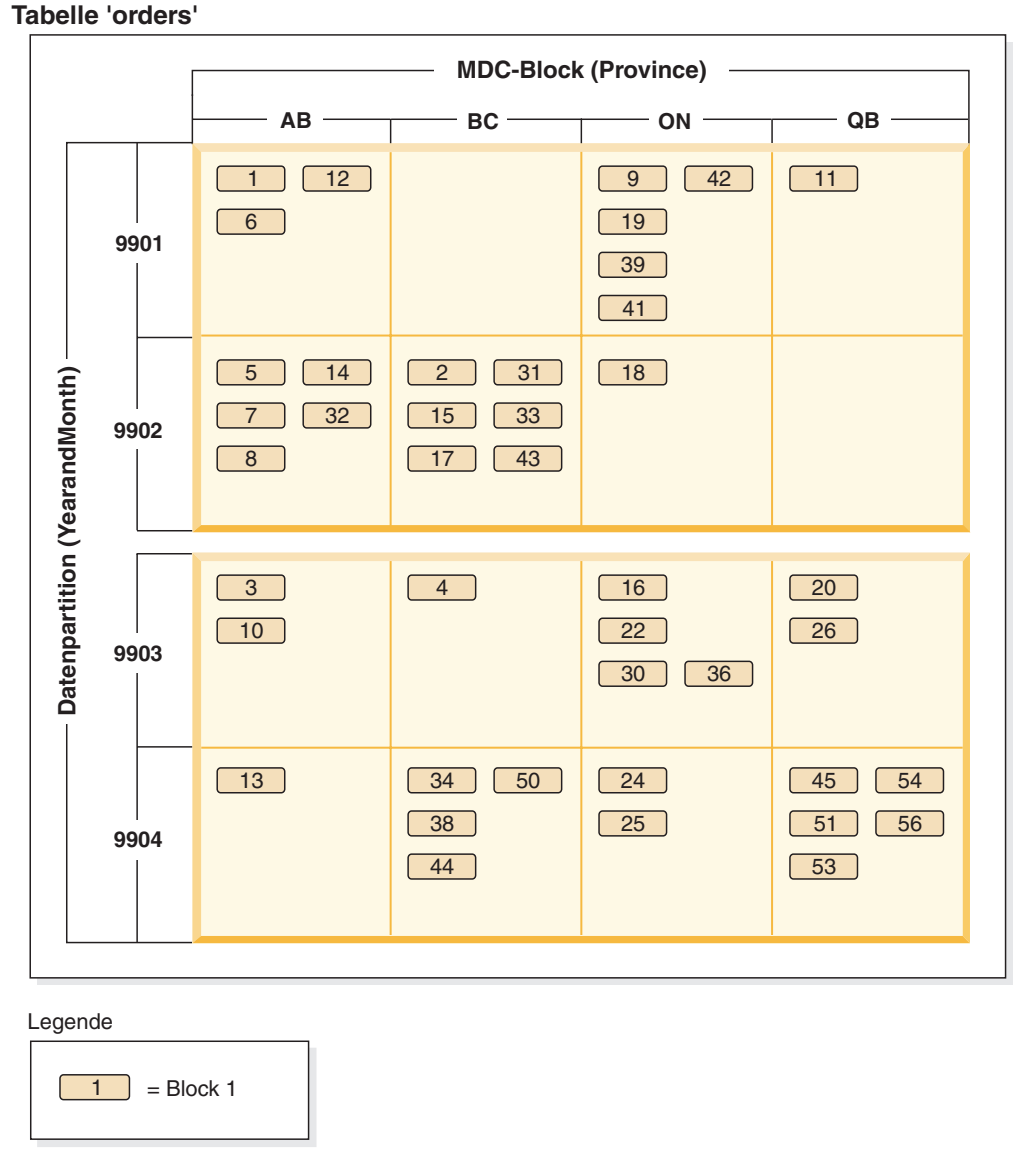


Abbildung 27. Eine nach 'YearAndMonth' partitionierte und nach 'Province' und 'YearAndMonth' organisierte Tabelle

Wenn die Partitionierung so festgelegt wurde, dass jeder Bereich nur einen einzigen Wert enthält, ergeben sich keine Vorteile, wenn die Tabellenpartitionierungsspalte in den MDC-Schlüssel aufgenommen wird.

Wichtige Hinweise

- Im Vergleich mit einer Basistabelle benötigen sowohl MDC-Tabellen als auch partitionierte Tabellen mehr Speicherplatz. Diese Speicherplatzanforderungen gelten zusätzlich zu den sonstigen Anforderungen, sind jedoch unter Berücksichtigung der sich daraus ergebenden Vorteile sinnvoll.
- Wenn Sie die Tabellenpartitionierung und die MDC-Funktionalität in Ihrer partitionierten Datenbankumgebung nicht zusammen einsetzen wollen, dann sollten Sie die Tabellenpartitionierung in den Fällen einsetzen, in denen die Datenverteilung verlässlich vorausgesagt werden kann. Dies ist normalerweise bei den hier erläuterten Systemtypen der Fall. Andernfalls sollten Sie die Verwendung von MDC in Betracht ziehen.

Kapitel 4. Parallele Datenbanksysteme

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 das DB2-Datenbankprodukt 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. Das DB2-Datenbanksystem unterstützt die folgenden Parallelitätstypen:

- Ein-/Ausgabeparallelität
- Abfrageparallelität
- Dienstprogrammparallelität

Ein-/Ausgabeparallelität

Wenn mehrere Container 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, der Datenbankmanager führt jedoch alle Abfragen gleichzeitig aus. DB2-Datenbankprodukte haben diese Art der Parallelität schon immer unterstützt.

Abfrageinterne Parallelität bezeichnet die gleichzeitige Verarbeitung von Teilen einer einzelnen Abfrage mithilfe 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-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. 28 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.

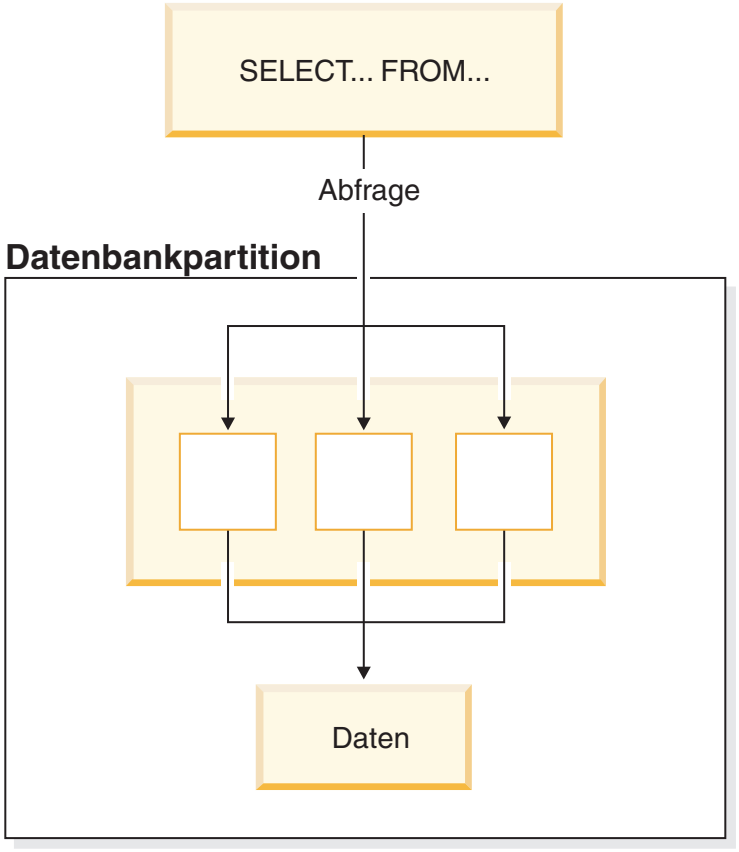


Abbildung 28. Partitionsinterne Parallelität

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-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. 29 auf Seite 87 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 einzigen Datenbankpartition.

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

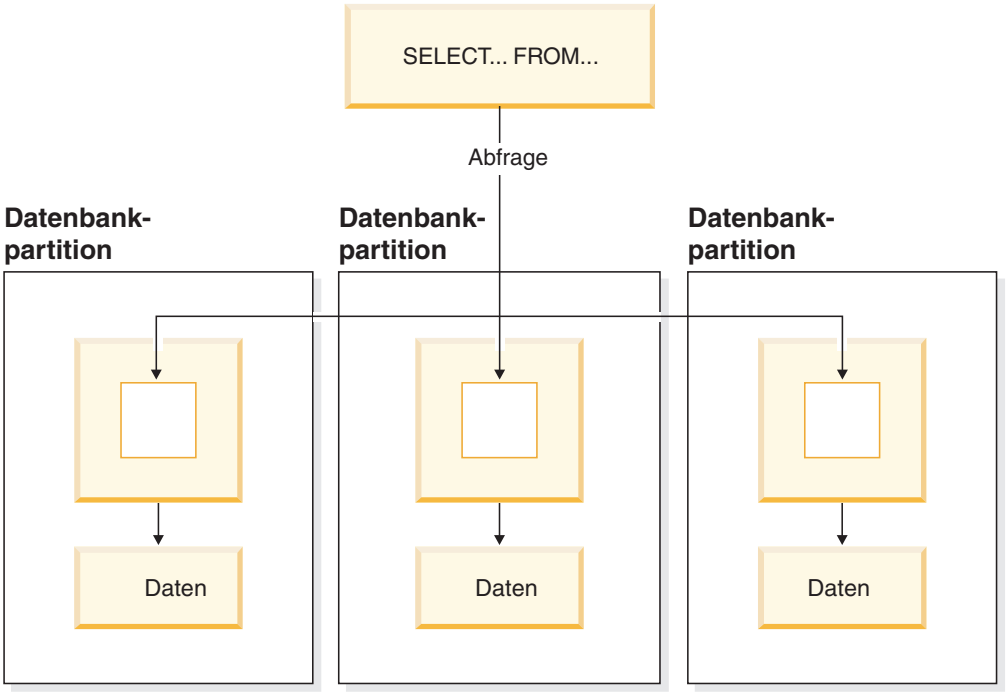


Abbildung 29. Partitionsübergreifende Parallelität

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.

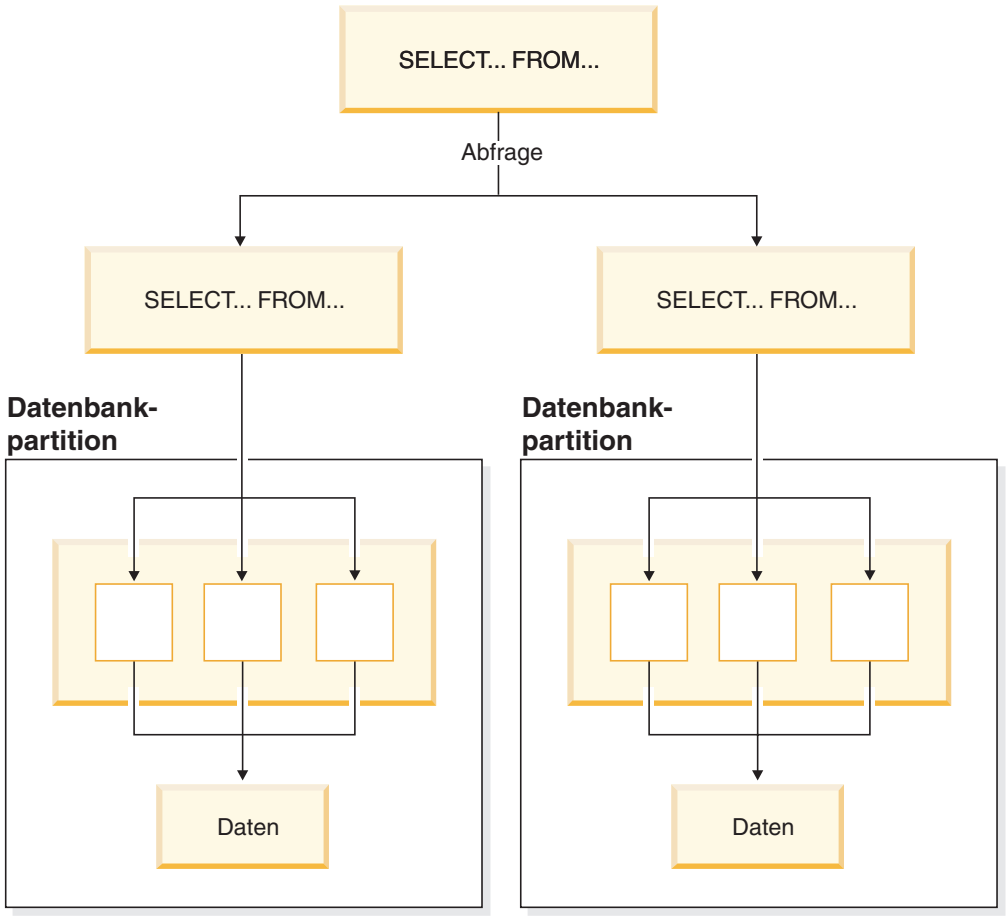


Abbildung 30. Simultane partitionsinterne und partitionsübergreifende Parallelität

Dienstprogrammparallelität

DB2-Dienstprogramme können die Vorteile der partitionsinternen Parallelität nutzen. Sie können außerdem die Vorteile der partitionsübergreifenden Parallelität nutzen. Wenn mehrere Datenbankpartitionen vorhanden sind, werden die Dienstprogramme in den einzelnen Datenbankpartitionen 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 Container 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. Das DB2-System 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.

Das Backup und der Restore von Daten sind Operationen, die wesentlich von der E/A-Leistung abhängig sind. Das DB2-System 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 Tabellenbereichscontainern parallel liest und asynchron auf mehrere Backup-Medien parallel schreibt.

Umgebungen mit partitionierten Datenbanken

Die Datenbankpartitionierungsfunktion (Database Partitioning Feature, DPF) erweitert den Datenbankmanager auf die parallele Umgebung mit mehreren Partitionen.

- 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 Umgebung mit partitionierten Datenbanken ist eine Datenbankinstallation, die die Verteilung von Daten für Datenbankpartitionen unterstützt.
- Eine *Einzelpartitionsdatenbank* ist eine Datenbank, die nur über eine Datenbankpartition verfügt. Sämtliche Daten in der Datenbank werden in dieser einen Datenbankpartition gespeichert. In diesem Fall bieten Datenbankpartitionsgruppen, wenn sie vorhanden sind, keine zusätzlichen Leistungsvorteile.
- Eine *Mehrpartitionsdatenbank* 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 Datenbankpartitionen gespeichert ist, heißt dies, dass einige ihrer Zeilen in einer Datenbankpartition gespeichert werden, während sich andere Zeilen in anderen Datenbankpartitionen befinden.

In der Regel befindet sich auf jedem physischen System nur eine Datenbankpartition. Die Prozessoren des Systems werden vom Datenbankmanager dazu verwendet, den in der jeweiligen Datenbankpartition gespeicherten Teil der Gesamtdaten der Datenbank zu verwalten.

Da die Daten über Partitionen verteilt sind, können Sie das Potenzial mehrerer Prozessoren auf mehreren physischen Systemen 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 eine einzige Datenbankpartition, die als *Koordinatorpartition* für den jeweiligen Benutzer bezeichnet wird. Die Koordinatorpartition ist in derselben Datenbankpartition aktiv wie die Anwendung bzw., im Fall einer fernen Anwendung, in der Datenbankpartition, mit der die Anwendung verbunden ist. Alle Datenbankpartitionen können als Koordinatorpartition verwendet werden.

Der Datenbankmanager ermöglicht es, Daten über verschiedene Datenbankpartitionen in der Datenbank verteilt 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 Mehrpartitionsdatenbank zugreifen, müssen die physische Speicherposition der Daten nicht kennen.

Die Daten werden trotz der physischen Trennung als logische Einheit verwendet und verwaltet. Benutzer können wählen, wie ihre Daten verteilt werden, indem Sie Verteilungsschlüssel deklarieren. Benutzer können darüber hinaus bestimmen, auf welche und auf wie viele Datenbankpartitionen ihre Daten verteilt werden, indem sie den Tabellenbereich und die zugeordnete Datenbankpartitionsgruppe auswählen, in der die Daten gespeichert werden sollen. Empfehlungen zur Verteilung und Replikation können mithilfe des DB2-Designadvisor ermittelt werden. Weiterhin wird eine aktualisierbare Verteilungszuordnung zusammen mit einem Hashalgorithmus verwendet, um die Zuordnung von Werten der Verteilungsschlüssel zu Datenbankpartitionen anzugeben, wodurch das Platzieren und Abrufen der einzelnen Datenzeilen festgelegt wird. Dadurch können Sie die Auslastung für große Tabellen über eine Mehrpartitionsdatenbank 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, sodass für den lokalen Datenzugriff eine erhöhte Leistung erzielt wird.

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

Eine alternative Methode, die Sie in Erwägung ziehen können, wenn Sie möchten, dass Tabellen in den einzelnen Datenbankpartitionen angelegt werden, ist die Verwendung und Replikation von MQTs (MQT = Materialized Query Table, gespeicherte Abfragetabelle). Sie können eine MQT mit den von Ihnen benötigten Informationen erstellen und diese dann auf jede Datenbankpartition replizieren.

Eine Nichtrootinstallation eines DB2-Datenbankprodukts unterstützt die Datenbankpartitionierung nicht. Infolgedessen kann eine Operation zum Hinzufügen eines Knotens nicht ausgeführt werden. Sie dürfen die Datei `db2nodes.cfg` nicht manuell aktualisieren. Eine manuelle Aktualisierung führt zur Anzeige des Fehlers SQL6031N.

Datenbankpartitions- und Prozessorumgebungen

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.

Dieser Abschnitt enthält eine Übersicht über folgende Hardwareumgebungen:

- Einzeldatenbankpartition auf Einzelprozessormaschine
- Einzeldatenbankpartition mit Multiprozessormaschine (SMP)
- Konfigurationen mit mehreren Datenbankpartitionen
  - Datenbankpartitionen mit einem Prozessor (MPP)
  - Datenbankpartitionen mit mehreren Prozessoren (Cluster aus SMP-Systemen)
  - Logische Datenbankpartitionen

Für jede Umgebung werden die Kapazität und Skalierbarkeit behandelt.

Einzeldatenbankpartition auf Einzelprozessormaschine

Diese Umgebung verfügt über Hauptspeicher und Platte, enthält aber nur eine CPU (siehe Abb. 31). Für diese Umgebung werden viele verschiedene Bezeichnungen verwendet, wie zum Beispiel eigenständige Datenbank, Client-/Server-datenbank, serielle Datenbank, Einzelprozessorsystem 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.

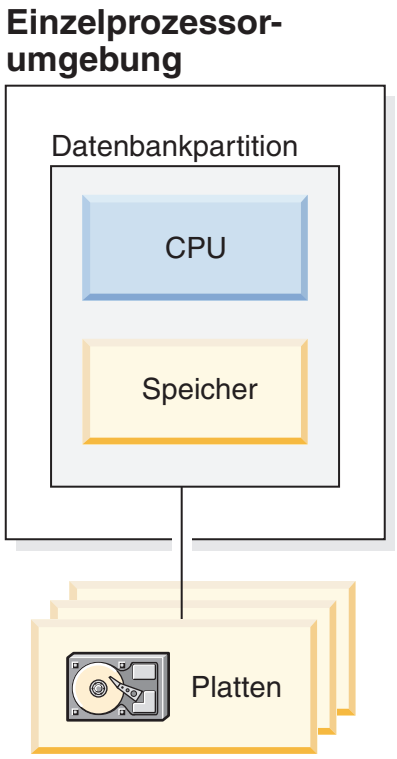


Abbildung 31. Einzeldatenbankpartition auf Einzelprozessormaschine

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 System mit einer Einzeldatenbankpartition und mehreren Prozessoren in Erwägung ziehen.

Einzeldatenpartition auf Multiprozessormaschine

Diese Umgebung besteht gewöhnlich aus mehreren gleich starken Prozessoren innerhalb derselben Maschine (siehe Abb. 32) und wird als *symmetrisches Multiprozessorsystem (SMP-System)* bezeichnet. (SMP steht für Symmetric Multiprocessor). Ressourcen wie Plattenspeicherplatz und Hauptspeicher werden *gemeinsam* genutzt.

Wenn mehrere Prozessoren verfügbar sind, können verschiedene Datenbankoperationen schneller durchgeführt werden. DB2-Datenbanksysteme können auch die Arbeit einer einzigen Abfrage auf die verfügbaren Prozessoren verteilen, um die Verarbeitungsgeschwindigkeit zu erhöhen. Andere Datenbankoperationen wie das Laden von Daten, das Backup und der Restore von Tabellenbereichen sowie die Erstellung von Indizes für vorhandene Daten können die Verfügbarkeit mehrerer Prozessoren ausnutzen.

Symmetrische Multiprozessorumgebung (SMP)

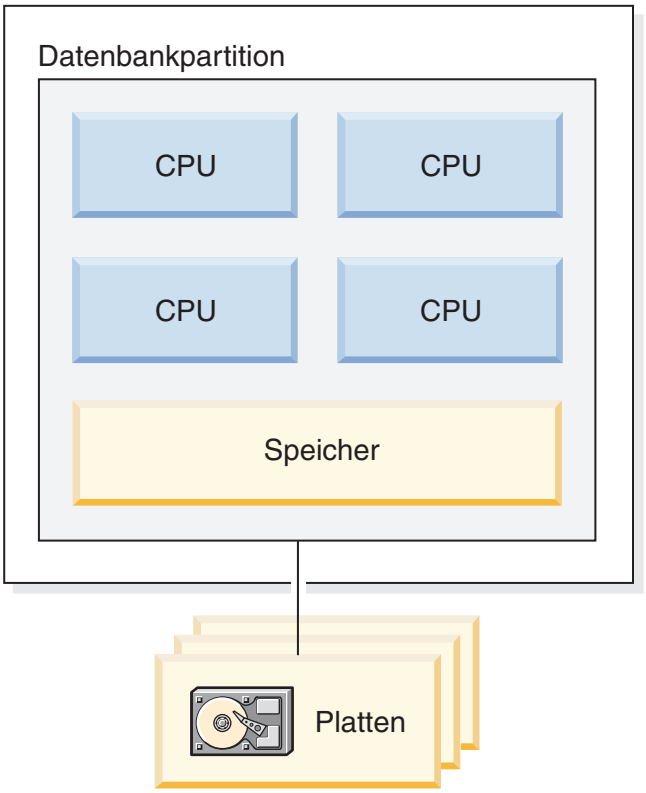


Abbildung 32. Einzelpartitionsdatenbank in einer symmetrischen Multiprozessorumgebung

Kapazität und Skalierbarkeit

In dieser Umgebung können Sie weitere Prozessoren hinzufügen. Da jedoch 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 genutzt.

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 Datenbankpartitionen in Erwägung ziehen.

Konfigurationen mit mehreren Datenbankpartitionen

Sie können eine Datenbank in mehrere Datenbankpartitionen 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 Konfigurationen von Datenbankpartitionen beschrieben:

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

Datenbankpartitionen mit einem Prozessor

In dieser Umgebung gibt es viele Datenbankpartitionen. Jede Datenbankpartition befindet sich auf einer eigenen Maschine mit eigenem Prozessor, eigenem Hauptspeicher und eigenen Platten (Abb. 33 auf Seite 94). 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 Datenbankpartitionen eine logische Gesamtheit bilden. Die Tatsache, dass die Daten verteilt sind, bleibt für die Mehrheit der Benutzer transparent. Die Arbeit kann unter den Datenbankmanagern aufgeteilt werden. Die Datenbankmanager in den einzelnen Datenbankpartitionen verrichten die Arbeit jeweils am eigenen Teil der Datenbank.

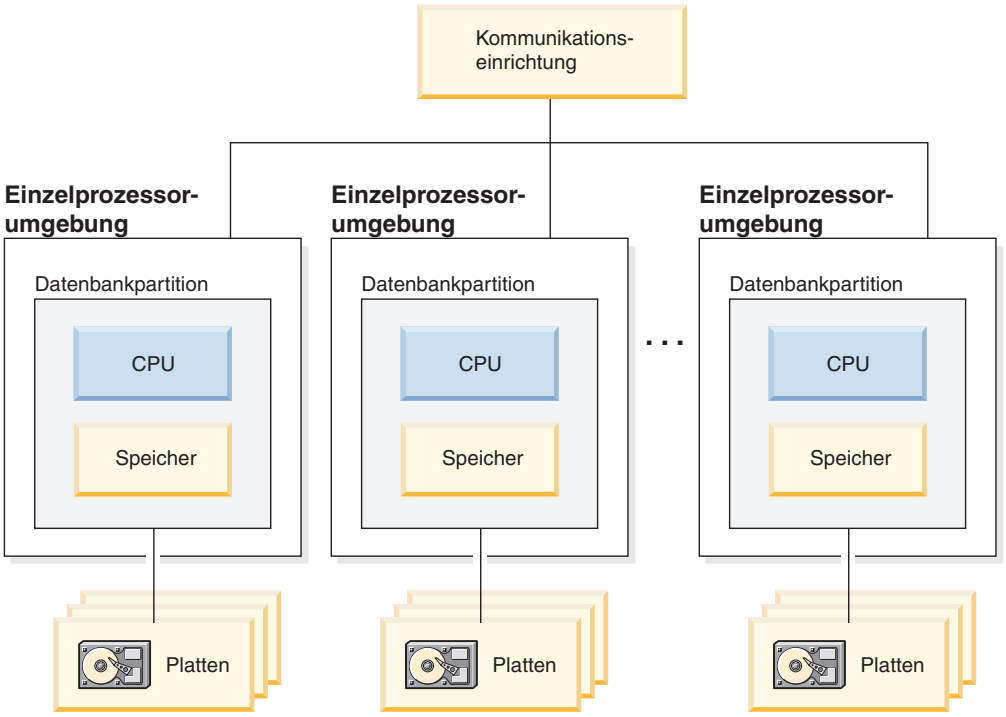


Abbildung 33. Umgebung zur exklusiven Parallelverarbeitung (MPP-Umgebung)

Kapazität und Skalierbarkeit

In dieser Umgebung können Sie Ihrer Konfiguration weitere Datenbankpartitionen (Knoten) hinzufügen. Auf einigen Plattformen, zum Beispiel RS/6000 SP, beträgt die höchstmögliche Anzahl 512 Knoten. Jedoch kann es praktische Grenzen für die Verwaltung einer höheren Anzahl von Maschinen und Instanzen geben.

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

Datenbankpartitionen mit mehreren Prozessoren

Eine Alternative zu einer Konfiguration, in der jede Datenbankpartition nur einen Prozessor hat, ist eine Konfiguration, in der jede Datenbankpartition über mehrere Prozessoren verfügt. Eine solche Konfiguration wird als *SMP-Cluster* bezeichnet (Abb. 34 auf Seite 95).

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

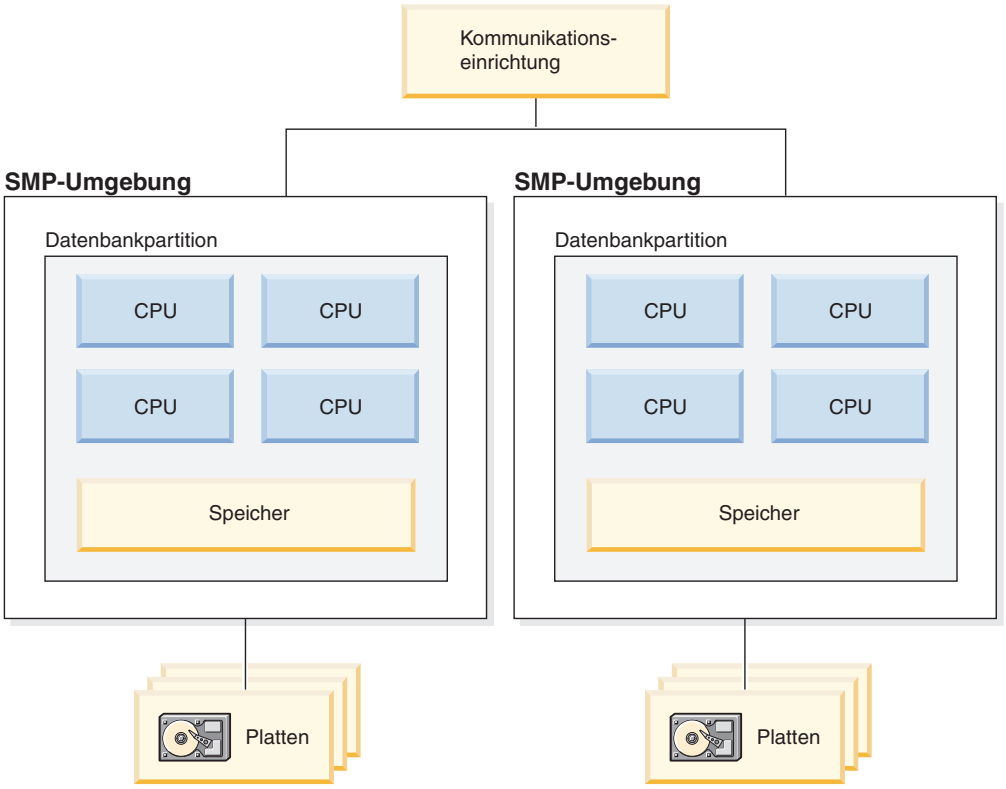


Abbildung 34. Mehrere symmetrische Multiprozessorumgebungen (SMP) in einem Cluster

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 genutzte Ressourcen verfügt, nutzen Datenbankpartitionen die Ressourcen nicht gemeinsam. Prozessoren werden gemeinsam genutzt, 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. 35 auf Seite 96 veranschaulicht die Möglichkeit, mehr Skalierbarkeit auf einer SMP-Maschine durch Hinzufügen weiterer Datenbankpartitionen zu gewinnen. Dies gilt insbesondere für Maschinen mit vielen Prozessoren. Durch die Verteilung der Datenbank können Sie jede Datenbankpartition getrennt verwalten und wiederherstellen.

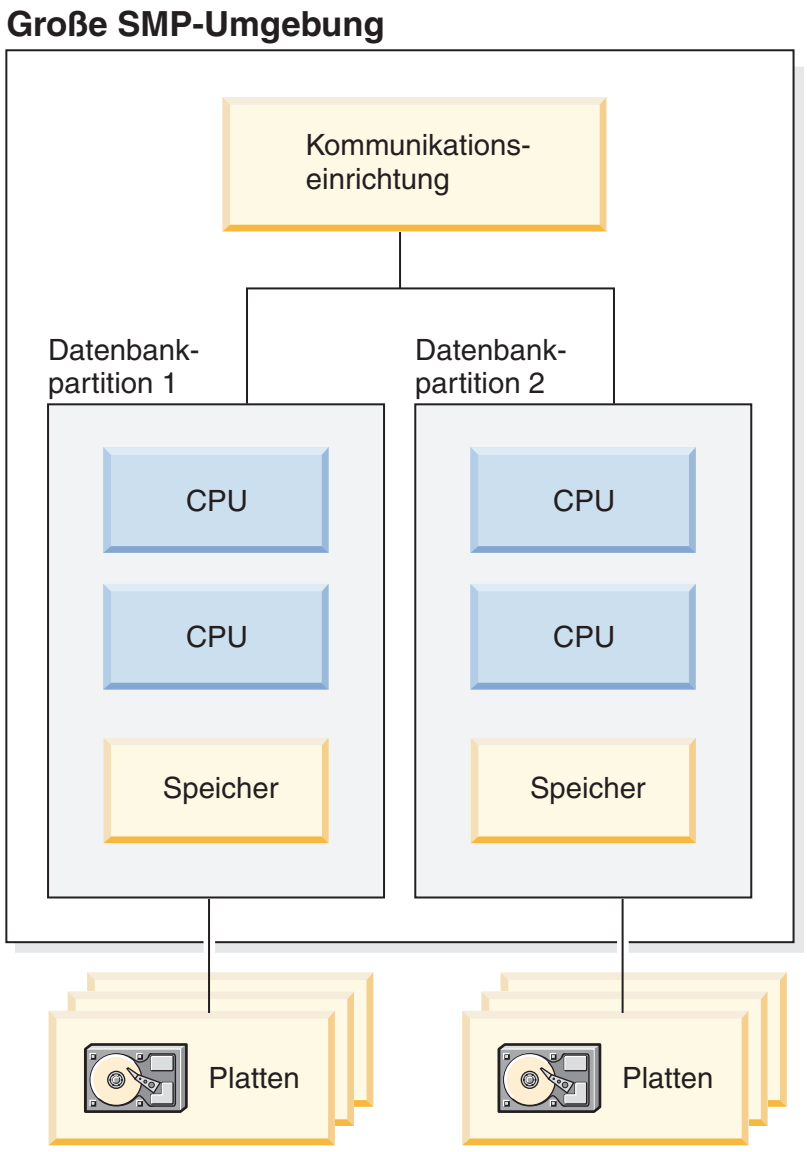


Abbildung 35. Partitionierte Datenbank mit symmetrischer Multiprozessorumgebung

Abb. 36 auf Seite 97 veranschaulicht die Möglichkeit, die in Abb. 35 gezeigte Konfiguration zu vervielfachen, um die Verarbeitungsleistung zu erhöhen.

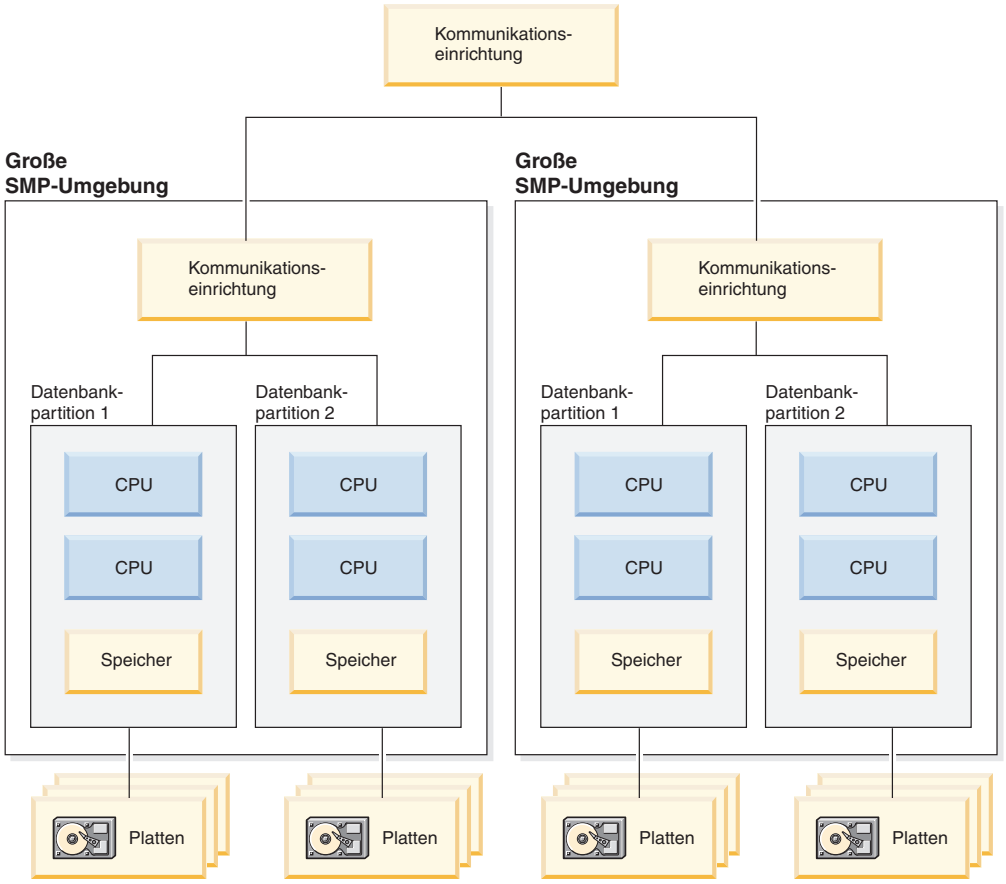


Abbildung 36. Partitionierte Datenbank mit SMP-Umgebungen in einem Cluster

Anmerkung: Die Möglichkeit, zwei oder mehr Datenbankpartitionen 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 Datenbankpartition derselben Datenbank enthält, übernommen und wieder gestartet werden.

Zusammenfassung der in den einzelnen Hardwareumgebungen realisierbaren Parallelität

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

Tabelle 8. Mögliche Arten der Parallelität in den verschiedenen Hardwareumgebungen

Hardwareumgebung	**E/A-Parallelität**	**Abfrageinterne Parallelität**	
		Partitionsinterne Parallelität	**Partitions-übergreifende Parallelität**
Einzeldatenbankpartition, Einzelprozessor	Ja	Nein (1)	Nein
Einzeldatenbankpartition, mehrere Prozessoren (SMP)	Ja	Ja	Nein
Mehrere Datenbankpartitionen, ein Prozessor (MPP)	Ja	Nein (1)	Ja
Mehrere Datenbankpartitionen, mehrere Prozessoren (Cluster von SMP-Systemen)	Ja	Ja	Ja
Logische Datenbankpartitionen	Ja	Ja	Ja
Anmerkung: (1) Es kann vorteilhaft sein, den Grad der Parallelität (mithilfe 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).			

Teil 2. Installationshinweise

Kapitel 5. Installationsvoraussetzungen

Installieren von DB2-Servern (Windows)

Diese Task beschreibt das Starten des **DB2-Installationsassistenten** unter Windows. Der **DB2-Installationsassistent** wird verwendet, um die gewünschte Installation anzugeben und das DB2-Datenbankprodukt auf dem System zu installieren.

Gehen Sie wie folgt vor, bevor Sie den **DB2-Installationsassistenten** starten:

- Wenn Sie eine Umgebung mit partitionierten Datenbanken einrichten wollen, lesen Sie den Abschnitt "Einrichten einer Umgebung mit partitionierten Datenbanken".
- Stellen Sie sicher, dass Ihr System die Installations-, Speicher- und Plattenvoraussetzungen erfüllt.
- Wenn Sie LDAP unter Windows verwenden möchten, um den DB2-Server in Active Directory zu registrieren, müssen Sie das Verzeichnisschema vor der Installation erweitern. Andernfalls müssen Sie die Registrierung der Datenbank und die Katalogisierung der Datenbank manuell durchführen. Weitere Informationen hierzu finden Sie im Abschnitt „Erweitern des Active Directory-Schemas für LDAP-Verzeichnisservices (Windows)".
- Sie müssen über ein Benutzerkonto eines lokalen Administrators verfügen, das die empfohlenen Benutzerrechte aufweist, um die Installation ausführen zu können. Bei DB2-Datenbankservern, bei denen LocalSystem als DAS und DB2-Instanzbenutzer verwendet werden kann und Sie nicht das Database Partitioning Feature verwenden, kann ein Benutzer, der kein Administrator ist, aber über umfangreichere Zugriffsrechte verfügt, die Installation ausführen.

  Anmerkung: Wenn die Produktinstallation über ein Benutzerkonto ohne Administratorrechte ausgeführt werden soll, muss die VS2005-Laufzeitbibliothek installiert werden, bevor das DB2-Datenbankprodukt installiert wird. Die VS2005-Laufzeitbibliothek ist auf dem Betriebssystem erforderlich, bevor das DB2-Datenbankprodukt installiert werden kann. Die VS2005-Laufzeitbibliothek ist auf der Download-Website für Microsoft®-Laufzeitbibliotheken verfügbar. Sie haben zwei Wahlmöglichkeiten: Wählen Sie für 32-Bit-Systeme die Datei `vcredist_x86.exe` und für 64-Bit-Systeme die Datei `vcredist_x64.exe` aus.
- Es ist zwar nicht obligatorisch, es empfiehlt sich allerdings, alle Programme zu schließen, damit das Installationsprogramm alle Dateien auf dem Computer ohne Neustart aktualisieren kann.

Einschränkungen

- Der Name der DB2-Kopie und der Instanzname dürfen nicht mit einem numerischen Wert beginnen.
- Der Name der DB2-Kopie und der Instanzname müssen für alle DB2-Kopien eindeutig sein.
- Die Verwendung von XML-Funktionen ist auf Datenbanken beschränkt, die nur über eine Datenbankpartition verfügen.
- Wenn eines der folgenden Produkte bereits installiert ist, können in demselben Pfad keine anderen DB2-Datenbankprodukte installiert werden:
  - IBM® Data Server Runtime Client
  - IBM Data Server Driver Package

– *DB2-Informationszentrale*

- In die Felder des **DB2-Installationsassistenten** können keine nicht englischen Zeichen eingegeben werden.
- Wenn Sie unter Windows Vista die erweiterte Sicherheit aktivieren, müssen Benutzer zur Gruppe DB2ADMNS oder DB2USERS gehören, um lokale DB2-Befehle und -Anwendungen auszuführen, da eine zusätzliche Sicherheitsfunktion (User Access Control) die Berechtigungen einschränkt, die lokalen Administratoren standardmäßig erteilt werden. Wenn Benutzer keiner dieser Gruppen angehören, haben sie keinen Lesezugriff auf lokale DB2Konfigurations- oder -Anwendungsdaten.

Vorgehensweise

Gehen Sie wie folgt vor, um den **DB2-Installationsassistenten** zu starten:

1. Melden Sie sich beim System mit dem lokalen Administratorkonto an, das Sie für die DB2-Installation definiert haben.
2. Wenn Sie über die DVD des DB2-Datenbankprodukts verfügen, legen Sie sie in das DVD-Laufwerk ein. Das **DB2 Setup-Launchpad** wird von der Funktion für automatische Ausführung automatisch gestartet, sofern diese Funktion aktiviert ist. Wenn die Funktion für automatische Ausführung nicht funktionieren sollte, durchsuchen Sie mithilfe des Windows-Explorers die DVD des DB2-Datenbankprodukts und klicken Sie das Installationssymbol (setup) doppelt an, um das **DB2 Setup-Launchpad** zu starten.
3. Wenn Sie das DB2-Datenbankprodukt von Passport Advantage heruntergeladen haben, führen Sie die ausführbare Datei aus, um die Installationsdateien des DB2-Datenbankprodukts zu extrahieren. Verwenden Sie Windows Explorer, um die DB2-Installationsdateien anzuzeigen, und klicken Sie das Symbol setup doppelt an, um das **DB2-Setup-Launchpad** zu starten.
4. Im **DB2-Setup-Launchpad** können Sie Installationsvoraussetzungen und die Release-Informationen anzeigen oder direkt mit der Installation fortfahren. Es empfiehlt sich, die Installationsvoraussetzungen und die Release-Informationen zu lesen, um die neuesten Informationen zu erhalten.
5. Klicken Sie **Produkt installieren** an. Im Fenster **Produkt installieren** werden die Produkte angezeigt, die zur Installation zur Verfügung stehen.

   Wenn auf dem Computer keine vorhandenen DB2-Datenbankprodukte installiert sind, müssen Sie die Installation durch Anklicken von **Neue installieren** starten. Setzen Sie die Installation fort, indem Sie die Eingabeaufforderungen des **DB2-Installationsassistenten** bearbeiten.

   Wenn mindestens ein vorhandenes DB2-Datenbankprodukt auf dem Computer installiert ist, können Sie folgende Aktionen ausführen:

   - Klicken Sie **Neue installieren** an, um eine neue DB2-Kopie zu erstellen.
   - Klicken Sie **Mit Vorhandenen arbeiten** an, um ein Upgrade für eine vorhandene DB2-Kopie durchzuführen, Funktionen zu einer vorhandenen DB2-Kopie hinzuzufügen, eine vorhandene Kopie von DB2 Version 8 oder Version 9.1 zu migrieren oder ein Add-on-Produkt zu installieren.
6. Der **DB2-Installationsassistent** ermittelt die Systemsprache und startet das Installationsprogramm für diese Sprache. Es steht eine Onlinehilfefunktion zur Verfügung, die Sie durch die verbleibenden Schritte leitet. Klicken Sie zum Aufrufen der Onlinehilfe die Schaltfläche **Hilfe** an oder drücken Sie die Taste **F1**. Sie können die Installation jederzeit durch Anklicken von **Abbrechen** beenden.

Das DB2-Datenbankprodukt wird standardmäßig im Verzeichnis *Programme*\IBM\sqllib installiert. Dabei ist *Programme* die Position des Verzeichnisses Programme.

Wenn Sie die Installation auf einem System ausführen, auf dem dieses Verzeichnis bereits verwendet wird, wird dem Installationspfad für das DB2-Datenbankprodukt die Kennung *_xx* hinzugefügt, wobei *_xx* für Ziffern steht, die mit 01 beginnen und je nach Anzahl der installierten DB2-Kopien ansteigen.

Sie können auch einen eigenen Pfad für die Installation des DB2-Datenbankprodukts angeben.

- Überprüfen Sie die Installation.
- Führen Sie die erforderlichen Tasks nach der Installation aus.

Informationen zu Fehlern, die während der Installation auftreten, können Sie der Installationsprotokolldatei im Verzeichnis Eigene Dateien\DB2LOG\ entnehmen. Die Protokolldatei verwendet das folgende Format: DB2-*Produktabkürzung*-*DatumZeit*.log, zum Beispiel DB2-ESE-Tue Apr 04 17_04_45 2006.log.

Wenn Sie möchten, dass Ihr DB2-Datenbankprodukt auf die DB2-Dokumentation auf dem lokalen Computer oder auf einem anderen Computer im Netz zugreifen kann, müssen Sie die *DB2-Informationszentrale* installieren. Die *DB2-Informationszentrale* enthält die Dokumentation für das DB2-Datenbanksystem und die zugehörigen DB2-Produkte. Standardmäßig erfolgt der Zugriff auf die DB2-Informationen über das World Wide Web, wenn die *DB2-Informationszentrale* nicht lokal installiert ist.

Speicherbegrenzungen für DB2 Express Edition und DB2 Workgroup Server Edition Wenn Sie DB2 Express Edition installieren, beträgt der maximal zulässige Speicherbereich für die Instanz 4 GB.

Wenn Sie DB2 Workgroup Server Edition installieren, beträgt der maximal zulässige Speicherbereich für die Instanz 16 GB.

Wie groß der für die Instanz zugeordnete Speicherbereich ist, hängt vom Konfigurationsparameter **INSTANCE_MEMORY** des Datenbankmanagers ab.

Wichtige Hinweise für die Migration von Version 9.1:

- Wenn die Speicherkonfiguration für Ihr DB2-Datenbankprodukt der Version 9.1 den zulässigen Grenzwert überschreitet, kann das DB2-Datenbankprodukt nach der Migration auf die aktuelle Version möglicherweise nicht gestartet werden.
- Der Manager für den Speicher mit automatischer Leistungsoptimierung vergrößert den Gesamtspeicher für die Instanz nicht über die Lizenzgrenzwerte hinaus.

Vorbereiten der Umgebung für partitionierte DB2-Server (Windows)

Dieser Abschnitt beschreibt die erforderlichen Schritte zum Vorbereiten Ihrer Windows-Umgebung für eine partitionierte Installation des DB2-Produkts.

Alle zugehörigen Computer müssen jeweils unter dem gleichen Betriebssystem ausgeführt werden.

Gehen Sie wie folgt vor, um die Windows-Umgebung für die Installation vorzubereiten:

1. Stellen Sie sicher, dass der Primärcomputer und die zugehörigen Computer zu derselben Windows-Domäne gehören. Überprüfen Sie die Domäne, zu der der Computer gehört, mithilfe des Dialogs 'System' in der Systemsteuerung.
2. Stellen Sie sicher, dass die Einstellungen für Uhrzeit und Datum auf dem Primärcomputer und den zugehörigen Computern konsistent sind. Damit die Zeit als gleich betrachtet wird, dürfen die Unterschiede in der Zeit (bezogen auf GMT - Westeuropäische Zeit) auf allen Computern nicht größer als eine Stunde sein.

   Systemdatum und -uhrzeit können im Dialog 'Datum/Uhrzeit' über die Systemsteuerung geändert werden. Sie können den Konfigurationsparameter `max_time_diff` verwenden, um diese Einschränkung zu ändern. Die Standardeinstellung ist `max_time_diff = 60`. Sie ermöglicht einen Unterschied von weniger als 60 Minuten.
3. Stellen Sie sicher, dass jedes Computerobjekt, das an einer Umgebung mit partitionierten Datenbanken beteiligt ist, das Zugriffsrecht zum Akzeptieren des Computers für die Delegierung ("Trust computer for delegation") markiert hat. Sie können überprüfen, ob das Markierungsfeld "Trust computer for delegation" auf der Registerkarte für allgemeine Optionen des Fensters für Kontoeigenschaften auf jedem Computer in der Konsole für Active Directory-Benutzer und -Computer markiert ist.
4. Stellen Sie sicher, dass alle zugehörigen Computer über TCP/IP miteinander kommunizieren können:
   a. Geben Sie auf einem der zugehörigen Computer den Befehl hostname ein, der den Hostnamen des Computers zurückgibt.
   b. Geben Sie auf einem anderen zugehörigen Computer den folgenden Befehl ein:

      ```
      ping hostname
      ```

      Hierbei ist *hostname* der Hostname des Primärcomputers. Ist der Test erfolgreich, erhalten Sie eine Ausgabe ähnlich der folgenden:

      ```
      Pinging ServerA.ibm.com [9.21.27.230] with 32 bytes of data:

      Reply from 9.21.27.230: bytes=32 time<10ms TTL=128
      Reply from 9.21.27.230: bytes=32 time<10ms TTL=128
      Reply from 9.21.27.230: bytes=32 time<10ms TTL=128
      ```

      Wiederholen Sie diese Schritte, bis Sie überprüft haben, ob alle zugehörigen Computer über TCP/IP miteinander kommunizieren können. Jeder Computer muss eine statische IP-Adresse haben.

      Wenn Sie planen, mehrere Netzwerkadapter zu verwenden, können Sie angeben, welcher Adapter für die Kommunikation zwischen den Datenbankpartitionsservern verwendet werden soll. Verwenden Sie den Befehl db2nchg, um nach Abschluss der Installation das Feld 'netname' in der Datei `db2nodes.cfg` anzugeben.
5. Während der Installation werden Sie aufgefordert, ein Benutzerkonto für den DB2-Verwaltungsserver (DAS) anzugeben. Hierbei handelt es sich um ein lokales Benutzerkonto oder ein Domänenbenutzerkonto, das für den DB2-Verwaltungsserver (DAS) verwendet wird. Der DAS ist ein Verwaltungsservice, der verwendet wird, um die GUI-Tools zu unterstützen und bei Verwaltungstasks zu helfen. Sie können an dieser Stelle einen Benutzer definieren oder den DB2-Installationsassistenten einen Benutzer erstellen lassen. Wenn der DB2-Installationsassistent einen neuen Domänenbenutzer erstellen soll, muss das für die Installation verwendete Konto über eine Berechtigung zum Erstellen von Domänenbenutzern verfügen.

6. Auf dem Primärcomputer, auf dem die Partition, der die Instanz gehört, installiert werden soll, muss ein Domänenbenutzerkonto vorhanden sein, das zur lokalen Gruppe der *Administratoren* gehört. Wenn Sie DB2 installieren, melden Sie sich als dieser Benutzer an. Sie müssen der lokalen Gruppe der *Administratoren* auf jedem der zugehörigen Computer jeweils das gleiche Benutzerkonto hinzufügen. Dieser Benutzer muss über die Benutzerberechtigung *Als Teil des Betriebssystems handeln* verfügen.
7. Stellen Sie sicher, dass der lokale Laufwerkbuchstabe vor dem Datenbankverzeichnis auf allen Computern in der Instanz identisch ist. Sie können dies überprüfen, indem Sie den Befehl GET DATABASE CONFIGURATION ausführen, und den Wert des Konfigurationsparameters DFTDBPATH DBM prüfen.
8. Während der Installation werden Sie aufgefordert, ein Domänenbenutzerkonto anzugeben, das der DB2-Instanz zugeordnet werden soll. Jeder DB2-Instanz ist ein Benutzer zugeordnet. Beim Starten der Instanz wird DB2 über diesen Benutzernamen angemeldet. Sie können an dieser Stelle einen Benutzer definieren oder den DB2-Installationsassistenten einen neuen Domänenbenutzer erstellen lassen.

   Beim Hinzufügen eines neuen Knotens zu einer partitionierten Umgebung muss der Name der DB2-Kopie auf allen Computern identisch sein.

   Wenn der DB2-Installationsassistent einen neuen Domänenbenutzer erstellen soll, muss das für die Installation verwendete Konto über eine Berechtigung zum Erstellen von Domänenbenutzern verfügen. Das Domänenkonto des Instanzbenutzers muss auf allen zugehörigen Computern zur lokalen Gruppe *Administratoren* gehören und erhält die folgenden Benutzerberechtigungen:
   - Als Teil des Betriebssystems handeln
   - Erstellen von Tokenobjekten
   - Sperren von Seiten im Speicher
   - Anmelden als Service
   - Anheben von Quoten
   - Ersetzen eines Tokens auf Prozessebene

   Wenn die erweiterte Sicherheit ausgewählt wurde, muss das Konto auch Mitglied der Gruppe DB2ADMNS sein. Da die Gruppe DB2ADMNS bereits über diese Berechtigungen verfügt, bräuchten Sie sie nicht explizit zum Konto hinzuzufügen.

Fast Communication Manager (Windows)

FCM (Fast Communication Manager) stellt die Kommunikationsunterstützung für DB2-Serverprodukte zur Verfügung, die zu derselben Instanz gehören. Jeder Datenbankpartitionsserver verfügt über einen FCM-Senderdämon und einen FCM-Empfängerdämon, der die Kommunikation zwischen Datenbankpartitionsservern zur Verfügung stellt, um Anforderungen von Agenten zu verarbeiten und Nachrichtenpuffer zu senden. Der FCM-Dämon wird beim Start der Instanz gestartet.

Schlägt die Kommunikation zwischen Datenbankpartitionsservern fehl oder wird die Kommunikation wiederhergestellt, aktualisiert der FCM-Thread Informationen (diese Informationen können mit dem Datenbanksystemmonitor abgefragt werden) und veranlasst, dass die entsprechende Aktion (z. B. ein Rollback der betroffenen Transaktion) ausgeführt wird. Sie können den Datenbanksystemmonitor verwenden, um Unterstützung beim Einstellen der FCM-Konfigurationsparameter zu erhalten.

Sie können die Anzahl der FCM-Nachrichtenpuffer mit dem Konfigurationsparameter *fcm_num_buffers* des Datenbankmanagers und die Anzahl der FCM-Kanäle mit dem Konfigurationsparameter *fcm_num_channels* des Datenbankmanagers festlegen. Die Konfigurationsparameter *fcm_num_buffers* und *fcm_num_channels* des Datenbankmanagers sind standardmäßig auf den Wert AUTOMATIC eingestellt. Der FCM überwacht die Ressourcennutzung, wenn diese Parameter auf AUTOMATIC gesetzt sind, und gibt Ressourcen teilweise frei. Es wird empfohlen, die Einstellung für diese Parameter auf AUTOMATIC zu belassen.

Übersicht über die Installation Ihres DB2-Serverprodukts (Linux und UNIX)

Dieser Abschnitt beschreibt die Schritte zum Installieren Ihres DB2-Serverprodukts unter AIX, HP-UX, Linux und Solaris.

Führen Sie die folgenden Schritte aus, um das DB2-Serverprodukt zu installieren:

1. Prüfen Sie die Voraussetzungen für das DB2-Produkt.
2. Lesen Sie die Informationen zur DB2-Migration (sofern anwendbar).
3. Ändern Sie die Kernelparameter unter HP-UX, Linux und Solaris. Auf allen Plattformen außer Linux auf x86_32 müssen Sie einen 64-Bit-Kernel installieren, bevor Sie mit der Installation fortfahren können; andernfalls schlägt die Installation fehl.
4. Bereiten Sie die Installationsmedien vor:

   Produkt-DVD
   Wenn die DB2-Produkt-DVD nicht automatisch angehängt wird, hängen Sie die DB2-Produkt-DVD selbst an.

   Installationsimage
   Wenn Sie ein Installationsimage heruntergeladen haben, müssen Sie die Datei entpacken.
5. Installieren Sie das DB2-Produkt mit einer der zur Verfügung stehenden Methoden:
   - DB2-Installationsassistent
   - Befehl db2_install
   - Unbeaufsichtigte Installation mithilfe einer Antwortdatei
   - Implementierung mit Nutzdatendateien

   Für DB2-Server können Sie den DB2-Installationsassistenten verwenden, um die folgenden Installations- und Konfigurationstasks auszuführen:
   - Auswählen des DB2-Installationstyps (typisch, komprimiert oder angepasst).
   - Auswählen der Position für die DB2-Produktinstallation.
   - Installieren der Sprachen, von denen zu einem späteren Zeitpunkt eine als voreingestellte Sprache für die Produktschnittstelle und die Nachrichten angegeben werden kann.
   - Installieren oder Durchführen eines Upgrades von IBM Tivoli System Automation for Multiplatforms Base Component (Linux und AIX).
   - Konfigurieren einer DB2-Instanz.
   - Installieren des DB2-Verwaltungsservers (DB2 Administration Server, DAS), einschließlich der Einrichtung von DAS-Benutzern.
   - Konfigurieren des DB2 Text Search-Servers.
   - Einrichten der Verwaltungsansprechpartner und der Benachrichtigungen des Diagnosemonitors.

- Installieren und Konfigurieren von Instanzen (einschließlich der Einrichtung von Instanzbenutzern).
- Einrichten der Informix-Datenquellenunterstützung.
- Vorbereiten des Katalogs der vorhandenen DB2-Tools.
- Angeben des Ports der DB2-Informationszentrale.
- Erstellen von Antwortdateien.

6. Wenn Sie einen DB2-Server mit einer anderen Methode als der Verwendung des DB2-Installationsassistenten installiert haben, sind Konfigurationsschritte nach der Installation erforderlich.

DB2-Installationsmethoden

Dieser Abschnitt enthält Informationen zu den DB2-Installationsmethoden. Die nachstehende Tabelle zeigt die für die einzelnen Betriebssysteme jeweils verfügbaren Installationsmethoden.

Tabelle 9. Installationsmethode nach Betriebssystem

Installationsmethode	Windows	Linux oder UNIX
DB2-Installationsassistent	Ja	Ja
Installation mithilfe einer Antwortdatei	Ja	Ja
Befehl db2_install	Nein	Ja
Implementierung mit Nutzdatendateien	Nein	Ja

Die nachstehende Liste beschreibt die DB2-Installationsmethoden.

DB2-Installationsassistent

Der DB2-Installationsassistent ist ein GUI-Installationsprogramm, das für die Betriebssysteme Linux, UNIX und Windows zur Verfügung steht. Der DB2-Installationsassistent bietet eine benutzerfreundliche Schnittstelle für die Installation von DB2-Produkten und die Durchführung der anfänglichen Installations- und Konfigurationstasks.

Der DB2-Installationsassistent kann auch zum Erstellen von DB2-Instanzen und Antwortdateien verwendet werden, mit denen diese Installation auf andere Maschinen dupliziert werden kann.

Anmerkung: Bei Installationen ohne Rootberechtigung auf Linux- und UNIX-Plattformen kann lediglich eine einzige DB2-Instanz vorhanden sein. Der DB2-Installationsassistent erstellt die nicht als Root ausgeführte Instanz automatisch.

Auf Linux- und UNIX-Plattformen ist ein X-Server erforderlich, um den DB2-Installationsassistenten anzuzeigen.

Installation mithilfe einer Antwortdatei

Eine Antwortdatei ist eine Textdatei, die Werte für die Installation und Konfiguration enthält. Die Datei wird vom Programm DB2 Setup gelesen, und die Installation wird auf der Grundlage der in dieser Datei angegebenen Werte ausgeführt.

Eine Installation mithilfe einer Antwortdatei kann auch als unbeaufsichtigte Installation bezeichnet werden.

Darüber hinaus ermöglichen Antwortdateien den Zugriff auf Parameter, die nicht über den DB2-Installationsassistenten definiert werden können.

Auf den Betriebssystemen Linux und UNIX kann Ihre eigene Anwendung Informationen zum Installationsfortschritt und Eingabeaufforderungen des Installationsprogramms in computerlesbarer Form empfangen, wenn Sie das DB2-Installationsimage in diese integrieren. Dieses Verhalten wird durch das Schlüsselwort INTERACTIVE der Antwortdatei gesteuert.

Antwortdateien können auf unterschiedliche Arten erstellt werden:

Durch Verwendung des Antwortdateigenerators (Windows-Plattformen)
Unter Windows können Sie mithilfe des Antwortdateigenerators eine Antwortdatei erstellen, die eine bestehende Installation repliziert. Wenn Sie beispielsweise einen IBM Data Server-Client installieren, würden Sie den Client zunächst vollständig konfigurieren und dann eine Antwortdatei generieren, um die Installation und Konfiguration des Clients auf anderen Computern zu replizieren.

Durch Verwendung des DB2-Installationsassistenten
Mithilfe des DB2-Installationsassistenten können Sie eine Antwortdatei auf der Basis der Optionen erstellen, die Sie mit dem DB2-Installationsassistenten auswählen. Die von Ihnen gewählten Optionen werden in einer Antwortdatei aufgezeichnet, die Sie in einem Verzeichnis auf Ihrem System speichern können. Wenn Sie die Installation einer partitionierten Datenbank auswählen, werden zwei Antwortdateien generiert: eine für den Computer, der Instanzeigner ist, und eine für die zugehörigen Computer.

Ein Vorteil dieser Installationsmethode besteht darin, dass Sie eine Antwortdatei erstellen können, ohne eine Installation durchzuführen. Diese Funktion kann hilfreich sein, um die erforderlichen Optionen zum Installieren des DB2-Produkts zu erfassen. Die Antwortdatei kann später verwendet werden, um das DB2-Produkt genau mit den angegebenen Optionen zu installieren.

Sie können ein Client- oder Serverprofil mit dem Befehl db2cfexp exportieren, um die Client- oder Serverkonfiguration zu speichern. Anschließend können Sie das Profil ohne großen Aufwand mit dem Befehl db2cfimp importieren. Mit dem Befehl db2cfexp exportierte Client- und Serverprofile können auch bei einer Installation mit Antwortdatei mithilfe des Schlüsselworts CLIENT_IMPORT-_PROFILE importiert werden.

Exportieren Sie das Client- oder Serverprofil erst nach Durchführung der Installation und Katalogisierung der Datenquellen.

Durch Anpassung der Musterantwortdateien, die mit allen DB2-Produkten geliefert werden
Wenn Sie für die Erstellung einer Antwortdatei nicht den Antwortdateigenerator oder den DB2-Installationsassistenten verwenden wollen, haben Sie auch die Möglichkeit, eine Musterantwortdatei manuell zu ändern. Musterantwortdateien werden auf der DB2-Produkt-DVD bereitgestellt. Die Musterantwortdateien enthalten detaillierte Angaben zu den gültigen Schlüsselwörtern für die einzelnen Produkte.

Befehl db2_install (Linux- und UNIX-Plattformen)
Der Befehl db2_install installiert *alle* Komponenten für das von Ihnen angegebene DB2-Produkt für die englische Schnittstelle. Weitere Sprachenunter-

stützungen können Sie mithilfe des Parameters -L auswählen. Sie können keine Komponenten auswählen oder abwählen.

Der Befehl db2_install installiert zwar alle Komponenten für das von Ihnen angegebene DB2-Produkt, aber er erstellt keine Benutzer, Gruppen und Instanzen und führt keine Konfiguration durch. Diese Installationsmethode ist unter Umständen in Fällen vorzuziehen, in denen das Konfigurieren nach der Installation erforderlich ist. Wenn Sie es vorziehen, das DB2-Produkt während der Installation zu konfigurieren, sollten Sie in Betracht ziehen, den DB2-Installationsassistenten zu verwenden.

Auf den Betriebssystemen Linux und UNIX kann Ihre eigene Anwendung Informationen zum Installationsfortschritt und Eingabeaufforderungen des Installationsprogramms in computerlesbarer Form empfangen, wenn Sie das DB2-Installationsimage in diese integrieren.

Bei diesen Installationsmethoden sind nach dem Implementieren der Produktdateien manuelle Konfigurationsschritte erforderlich.

Implementierung mit Nutzdatendateien (nur Linux und UNIX)
Bei dieser Methode handelt es sich um eine fortgeschrittene Installationsmethode, die für die meisten Benutzer nicht zu empfehlen ist. Der Benutzer muss in ihrem Verlauf manuell Dateien mit Nutzdaten installieren. Eine Datei mit Nutzdaten ist eine komprimierte TAR-Datei, in der alle Dateien und Metadaten für eine installierbare Komponente enthalten sind.

Bei diesen Installationsmethoden sind nach dem Implementieren der Produktdateien manuelle Konfigurationsschritte erforderlich.

Implementierung von DB2-Produkten
Sie können alle oben beschriebenen Installationsmethoden mit allen verfügbaren Optionen verwenden, um eine große Zahl Clients und DB2-Server zu implementieren. Version 9.5 weist Funktionen und Tools auf, mit denen Sie sowohl Server als auch Clients rasch in Ihrem gesamten Unternehmen implementieren können.

Im Artikel Best Practices: Deploying IBM DB2 Products finden Sie Details zu empfohlenen Methoden zur Implementierung der DB2-Datenbankprodukte von Version 9.5 auf Linux-, UNIX- und Windows-Plattformen auf mehreren Computern. Es gibt viele DB2-Datenbankprodukte, es kann also eine Herausforderung sein, das richtige Produkt auszuwählen, wenn Sie mit der DB2-Produktlinie nicht vertraut sind.

Das Dokument DB2 Deployment Guide enthält Details zur Vorgehensweise bei der Implementierung von DB2 Version 9.5, zur Implementierung von DB2-Servern und -Clients auf einzelnen und mehreren Systemen und zur Implementierung von DB2 Version 9.5 mit Microsoft System Management Server (SMS) und verschiedenen Anwendungen, wie Java™-, C/C++-, PHP-, Python-, Ruby-, Perl- und .Net-Anwendungen. Diese IBM Redbooks-Veröffentlichung enthält allgemeine Anleitungen und dient als Referenzressource für die Implementierung von Lösungen auf Basis von DB2.

Anmerkung: DB2-Produktinstallationen sind auf Linux- und UNIX-Plattformen keine Betriebssystempakete mehr. Sie können für die Installation daher keine Betriebssystembefehle mehr verwenden. Alle vorhandenen Scripts, die Sie für die Schnittstelle mit und zum Abfragen von DB2-Installationen verwenden, müssen geändert werden.

Installieren von DB2-Servern mit dem DB2-Installationsassistenten (Linux und UNIX)

In diesem Abschnitt wird beschrieben, wie der **DB2-Installationsassistent** unter Linux- und UNIX-Betriebssystemen gestartet wird. Der **DB2-Installationsassistent** wird verwendet, um die gewünschten Installationsvorgaben festzulegen und das DB2-Datenbankprodukt auf dem System zu installieren.

Vorbereitung

Vor dem Starten des **DB2-Installationsassistenten**:

- Wenn Sie beabsichtigen, eine Umgebung mit partitionierten Datenbanken zu konfigurieren, finden Sie entsprechende Informationen im Abschnitt „Einrichten einer Umgebung mit partitionierten Datenbanken" in *DB2-Server - Einstieg*.
- Stellen Sie sicher, dass Ihr System die Anforderungen im Hinblick auf die Installation, den Hauptspeicher und die Plattenspeicherkapazität erfüllt.
- Sie können einen DB2-Datenbankserver als Benutzer mit Rootberechtigung und als Benutzer ohne Rootberechtigung installieren. Weitere Informationen zur Installation als Benutzer ohne Rootberechtigung finden Sie im Abschnitt „Installation als Benutzer ohne Rootberechtigung - Übersicht (Linux und UNIX)" in *DB2-Server - Einstieg*.
- Das DB2-Datenbankproduktimage muss verfügbar sein. DB2-Installationsimages sind entweder durch den Erwerb einer physischen DVD des DB2-Datenbankprodukts oder durch Herunterladen eines Installationsimages von Passport Advantage erhältlich.
- Wenn Sie landessprachliche Versionen eines DB2-Datenbankprodukts installieren, benötigen Sie die entsprechenden Landessprachenpakete.
- Der **DB2-Installationsassistent** ist ein grafisch orientiertes Installationsprogramm. Um den **DB2-Installationsassistenten** auf Ihrem System ausführen zu können, benötigen Sie die X Windows-Software zur Wiedergabe einer grafischen Benutzerschnittstelle (GUI). Stellen Sie sicher, dass der X Windows-Server aktiv ist. Stellen Sie sicher, dass Sie Ihre Anzeige (DISPLAY) ordnungsgemäß exportiert haben. Beispiel: `export DISPLAY=9.26.163.144:0.`
- Wird in Ihrer Umgebung Sicherheitssoftware verwendet, müssen Sie die erforderlichen DB2-Benutzer manuell erstellen, bevor Sie den **DB2-Installationsassistenten** starten.

Einschränkungen

- Die Verwendung von XML-Funktionen ist auf Datenbanken beschränkt, die mit dem codierten Zeichensatz UTF-8 definiert sind und nur über eine Datenbankpartition verfügen.
- In den Feldern des **DB2-Installationsassistenten** werden keine Sonderzeichen der jeweiligen Landessprache akzeptiert.

Vorgehensweise

Gehen Sie wie folgt vor, um den **DB2-Installationsassistenten** zu starten:

1. Wenn Sie über eine physische DVD des DB2-Datenbankprodukts verfügen, wechseln Sie in das Verzeichnis, in dem die DVD des DB2-Datenbankprodukts angehängt ist. Geben Sie dazu den folgenden Befehl ein:

   ```
   cd /dvdrom
   ```

Dabei steht */dvdrom* für den Mountpunkt der DVD des DB2-Datenbankprodukts.

2. Wenn Sie das DB2-Datenbankproduktimage heruntergeladen haben, müssen Sie die Produktdatei dekomprimieren und entpacken.
   a. Dekomprimieren Sie die Produktdatei:
      ```
      gzip -d produkt.tar.gz
      ```
      Dabei steht *produkt* für den Namen des Produkts, das Sie heruntergeladen haben.
   b. Entpacken Sie die Produktdatei:

      Unter Linux-Betriebssystemen
      ```
      tar -xvf produkt.tar
      ```

      Unter AIX-, HP-UX- und Solaris-Betriebssystemen
      ```
      gnutar -xvf produkt.tar
      ```

      Dabei steht *produkt* für den Namen des Produkts, das Sie heruntergeladen haben.
   c. Wechseln Sie das Verzeichnis:
      ```
      cd ./produkt
      ```
      Dabei steht *produkt* für den Namen des Produkts, das Sie heruntergeladen haben.

   Anmerkung: Wenn Sie das Landessprachenpaket heruntergeladen haben, entpacken Sie es in demselben Verzeichnis. So werden die Unterverzeichnisse (z. B. `./nlpack/disk1`) in demselben Verzeichnis erstellt und das Installationsprogramm kann die Installationsimages automatisch und ohne Benutzereingabeaufforderungen finden.
3. Geben Sie den Befehl ./db2setup von dem Verzeichnis aus ein, in dem sich das Produktimage befindet, um den **DB2-Installationsassistenten** zu starten.
4. Das **IBM DB2-Setup-Launchpad** wird geöffnet. In diesem Fenster können Sie die Installationsvoraussetzungen und die Release-Informationen anzeigen oder direkt mit der Installation fortfahren. Es empfiehlt sich, die Vorbedingungen für die Installation und die Release-Informationen zu lesen, um die neuesten Informationen zu erhalten.
5. Klicken Sie **Produkt installieren** an. Im Fenster **Produkt installieren** werden die Produkte angezeigt, die zur Installation zur Verfügung stehen.

   Starten Sie die Installation, indem Sie **Neue installieren** anklicken. Führen Sie die Installation aus, indem Sie den Eingabeaufforderungen des **DB2-Installationsassistenten** folgen.

   Arbeiten Sie nach dem Start der Installation die einzelnen Installationsanzeigen des **DB2-Installationsassistenten** nacheinander durch und wählen Sie die gewünschten Optionen aus. Informationen zur Ausführung der restlichen Schritte finden Sie in der Installationshilfe. Klicken Sie zum Aufrufen der Installationshilfe **Hilfe** an, oder drücken Sie die Taste F1. Sie können die Installation jederzeit durch Anklicken von **Abbrechen** beenden.

Ergebnisse

Nicht unter der Rootberechtigung ausgeführte Installationen von DB2-Datenbankprodukten werden immer im Verzeichnis $*HOME*/sqllib installiert. Dabei ist $*HOME* das Ausgangsverzeichnis des Benutzers ohne Rootberechtigung.

Bei Rootinstallationen werden DB2-Datenbankprodukte standardmäßig in folgendem Verzeichnis bzw. einem der folgenden Verzeichnisse installiert:

AIX, HP-UX oder Solaris
`/opt/IBM/db2/V9.5`

Linux `/opt/ibm/db2/V9.5`

Wenn Sie die Installation auf einem System vornehmen, auf dem dieses Verzeichnis bereits verwendet wird, wird dem Installationspfad für das DB2-Datenbankprodukt die Kennung *_xx* hinzugefügt, wobei *_xx* für Ziffern steht, die mit 01 beginnen und je nach Anzahl der installierten DB2-Kopien ansteigen.

Sie können auch einen eigenen Pfad für die Installation des DB2-Datenbankprodukts angeben.

Für DB2-Installationspfade gelten die folgenden Regeln:

- Sie dürfen Kleinbuchstaben (a–z), Großbuchstaben (A–Z) und das Unterstreichungszeichen (_) enthalten.
- Sie dürfen nicht länger als 128 Zeichen sein.
- Sie dürfen keine Leerzeichen enthalten.
- Sie dürfen keine Sonderzeichen der jeweiligen Landessprache enthalten.

Die folgenden Installationsprotokolldateien werden verwendet:

- Die DB2-Installationsprotokolldatei. Diese Datei erfasst alle DB2-Installationsinformationen einschließlich Fehlern.
  - Bei Rootinstallationen lautet der Name der DB2-Installationsprotokolldatei `db2setup.log`.
  - Bei nicht als Root ausgeführten Installationen lautet der Name der DB2-Installationsprotokolldatei `db2setup_`*`benutzername`*`.log`. Dabei ist *benutzername* die Benutzer-ID ohne Rootberechtigung, unter der die Installation durchgeführt wurde.
- Die DB2-Fehlerprotokolldatei. Diese Datei erfasst alle Fehlernachrichten, die von Java™ zurückgegeben werden (z. B. Nachrichten zu Ausnahmebedingungen und Traps).
  - Bei Rootinstallationen lautet der Name der DB2-Fehlerprotokolldatei `db2setup.err`.
  - Bei nicht als Root ausgeführten Installationen lautet der Name der DB2-Fehlerprotokolldatei `db2setup_`*`benutzername`*`.err`. Dabei ist *benutzername* die Benutzer-ID ohne Rootberechtigung, unter der die Installation durchgeführt wurde.

Diese Protokolldateien befinden sich standardmäßig im Verzeichnis `/tmp`. Die Speicherposition der Protokolldateien kann angegeben werden.

Die Datei `db2setup.his` wird nicht mehr verwendet. Stattdessen speichert das DB2-Installationsprogramm eine Kopie der DB2-Installationsprotokolldatei im Verzeichnis `DB2_DIR/install/logs/` und benennt sie in `db2install.history` um. Wenn der Name bereits vorhanden ist, benennt das DB2-Installationsprogramm die Datei in `db2install.history.`*`xxxx`* um. Dabei ist *xxxx* eine Zahl von 0000 bis 9999, die davon abhängt, wie viele Installationen sich auf der Maschine befinden.

Jede Installationskopie verfügt über eine separate Liste von Protokolldateien. Wenn eine Installationskopie entfernt wird, werden auch die Protokolldateien in diesem

Installationspfad entfernt. Dieser Kopiervorgang wird gegen Ende der Installation ausgeführt. Wenn das Programm vor dem Abschluss der Installation gestoppt oder abgebrochen wird, wird die Protokolldatei nicht erstellt.

Weitere Schritte

- Überprüfen Sie die Installation.
- Führen Sie die erforderlichen Tasks nach der Installation aus.

Landessprachenpakete können auch durch Ausführen des Befehls ./db2setup in dem Verzeichnis, in dem sich das Landessprachenpaket befindet, installiert werden, nachdem das DB2-Datenbankprodukt installiert wurde.

Wenn Sie unter Linux x86 möchten, dass Ihr DB2-Datenbankprodukt auf die DB2-Dokumentation auf dem lokalen Computer oder auf einem anderen Computer im Netzwerk zugreifen kann, müssen Sie die *DB2-Informationszentrale* installieren. Die *DB2-Informationszentrale* enthält die Dokumentation für das DB2-Datenbanksystem und die zugehörigen Produkte.

Speicherbegrenzungen für DB2 Express Edition und DB2 Workgroup Server Edition Wenn Sie DB2 Express Edition installieren, beträgt der maximal zulässige Speicherbereich für die Instanz 4 GB.

Wenn Sie DB2 Workgroup Server Edition installieren, beträgt der maximal zulässige Speicherbereich für die Instanz 16 GB.

Wie groß der für die Instanz zugeordnete Speicherbereich ist, hängt vom Konfigurationsparameter **INSTANCE_MEMORY** des Datenbankmanagers ab.

Wichtige Hinweise für die Migration von Version 9.1:

- Wenn die Speicherkonfiguration für Ihr DB2-Datenbankprodukt der Version 9.1 den zulässigen Grenzwert überschreitet, kann das DB2-Datenbankprodukt nach der Migration auf die aktuelle Version möglicherweise nicht gestartet werden.
- Der Manager für den Speicher mit automatischer Leistungsoptimierung vergrößert den Gesamtspeicher für die Instanz nicht über die Lizenzgrenzwerte hinaus.

Fast Communication Manager (Linux und UNIX)

Fast Communication Manager (FCM) stellt die Kommunikationsunterstützung für DB2-Serverprodukte zur Verfügung, die Database Partitioning Feature (DPF) verwenden.

Bei Instanzen mit mehreren Partitionen verfügt jeder Datenbankpartitionsserver über einen FCM-Senderdämon und einen FCM-Empfängerdämon, der die Kommunikation zwischen Datenbankpartitionsservern zur Verfügung stellt, um Anforderungen von Agenten zu verarbeiten und Nachrichtenpuffer zu senden. Der FCM-Dämon wird beim Start der Instanz mit mehreren Partitionen gestartet.

Schlägt die Kommunikation zwischen Datenbankpartitionsservern fehl oder wird die Kommunikation wiederhergestellt, aktualisieren die FCM-Dämonen Informationen. Sie können diese Informationen mit dem Datenbanksystemmonitor abrufen. Die FCM-Dämonen lösen außerdem die entsprechende Aktion aus. Ein Beispiel für eine entsprechende Aktion ist die Rollback-Operation für eine betroffene Transaktion. Sie können den Datenbanksystemmonitor verwenden, um Unterstützung beim Einstellen der FCM-Konfigurationsparameter zu erhalten.

Sie können die Anzahl der FCM-Nachrichtenpuffer mit dem Konfigurationsparameter *fcm_num_buffers* des Datenbankmanagers festlegen. Außerdem können Sie die Anzahl der FCM-Kanäle mit dem Konfigurationsparameter *fcm_num_channels* des Datenbankmanagers festlegen. Die Konfigurationsparameter *fcm_num_buffers* und *fcm_num_channels* des Datenbankmanagers werden standardmäßig auf den Wert AUTOMATIC gesetzt. Der FCM überwacht die Ressourcennutzung, wenn diese Parameter auf AUTOMATIC gesetzt sind, und gibt Ressourcen teilweise frei. Es wird empfohlen, die Einstellung für diese Parameter auf AUTOMATIC zu belassen.

Kapitel 6. Installationsvorbereitungen

Zusätzliche Tasks zur Installationsvorbereitung für eine Umgebung mit partitionierten Datenbanken (Linux und UNIX)

Aktualisieren der Umgebungseinstellungen für eine partitionierte DB2-Installation (AIX)

Diese Task beschreibt die Umgebungseinstellungen, die Sie auf allen Computern, die Ihrem partitionierten Datenbanksystem angehören sollen, aktualisieren müssen.

Um die AIX-Umgebungseinstellungen zu aktualisieren, gehen Sie wie folgt vor:

1. Melden Sie sich am Computer als Benutzer mit der Berechtigung `Root` an.
2. Setzen Sie das AIX-Einheitenattribut `maxuproc` (maximale Anzahl der Prozesse pro Benutzer) auf *4096*, indem Sie den folgenden Befehl eingeben:

   ```
   chdev -l sys0 -a maxuproc='4096'
   ```

   Anmerkung: Für den Wechsel zu einem 64-Bit-Kernel kann der Befehl `bosboot/reboot` erforderlich sein, wenn ein anderes Image ausgeführt wird.
3. Setzen Sie auf allen Workstations, die dem partitionierten Datenbanksystem angehören, die Parameter für das TCP/IP-Netzwerk auf die nachfolgend aufgelisteten Werte. Diese Werte stellen die Mindestwerte für diese Parameter dar. Ist ein Netzwerkparameter bereits auf einen höheren Wert eingestellt, lassen Sie den Wert unverändert.

   ```
   thewall       = 65536
      sb_max         = 1310720
      rfc1323        = 1
      tcp_sendspace = 221184
      tcp_recvspace = 221184
      udp_sendspace = 65536
      udp_recvspace = 65536
      ipqmaxlen     = 250
      somaxconn     = 1024
   ```

   Um die aktuellen Einstellungen aller Netzwerkparameter aufzulisten, geben Sie den folgenden Befehl ein:

   ```
   no -a | more
   ```

   Geben Sie den folgenden Befehl ein, um einen Parameter einzustellen:

   ```
   no -o parametername=wert
   ```

   Dabei gilt Folgendes:

   - *parametername* steht für den Parameter, der eingestellt werden soll
   - *wert* steht für den Wert, der für diesen Parameter verwendet werden soll

   Geben Sie beispielsweise den folgenden Befehl ein, um den Parameter `tcp_sendspace` auf `221184` einzustellen:

   ```
   no -o tcp_sendspace=221184
   ```
4. Wenn Sie eine Hochgeschwindigkeitsverbindung verwenden, müssen Sie die Parameter *spoolsize* und *rpoolsize* für *css0* auf folgende Werte einstellen:

   ```
   spoolsize     16777216
   rpoolsize     16777216
   ```

Um die aktuellen Einstellungen dieser Parameter aufzulisten, geben Sie den folgenden Befehl ein:

```
lsattr -l css0 -E
```

Geben Sie die folgenden Befehle ein, um diese Parameter einzustellen:

```
/usr/lpp/ssp/css/chgcss -l css0 -a spoolsize=16777216
  /usr/lpp/ssp/css/chgcss -l css0 -a rpoolsize=16777216
```

Wenn Sie die Datei /tftpboot/tuning.cst zum Optimieren Ihres Systems nicht verwenden, können Sie die Beispielscriptdatei DB2DIR/misc/rc.local.sample verwenden, um die Netzwerkparameter nach der Installation zu aktualisieren. Dabei steht DB2DIR für den Pfad, in dem das DB2-Produkt installiert wurde. Um die Netzwerkparameter mithilfe der Beispielscriptdatei nach der Installation zu aktualisieren, gehen Sie wie folgt vor:

a. Kopieren Sie die Scriptdatei in das Verzeichnis /etc, und berechtigen Sie den Benutzer mit Root-Berechtigung zum Ausführen der Datei. Geben Sie hierfür die folgenden Befehle ein:

   ```
   cp /usr/opt/db2_09_01/misc/rc.local.sample /etc/rc.local
   chown root:sys /etc/rc.local
   chmod 744 /etc/rc.local
   ```

b. Zeigen Sie die Datei /etc/rc.local an und aktualisieren Sie sie falls erforderlich.

c. Fügen Sie einen Eintrag zur Datei /etc/inittab hinzu, sodass das Script /etc/rc.local bei jedem Neustart der Maschine ausgeführt wird. Mit dem Befehl mkitab können Sie einen Eintrag zur Datei /etc/inittab hinzufügen. Geben Sie zum Hinzufügen des Eintrags den folgenden Befehl ein:

   ```
   mkitab "rclocal:2:wait:/etc/rc.local > /dev/console 2>&1"
   ```

d. Überprüfen Sie, ob der Eintrag /etc/rc.nfs in der Datei /etc/inittab vorhanden ist, indem Sie den folgenden Befehl eingeben:

   ```
   lsitab rcnfs
   ```

e. Aktualisieren Sie die Netzwerkparameter, ohne Ihr System neu zu starten, indem Sie den folgenden Befehl eingeben:

   ```
   /etc/rc.local
   ```

5. Stellen Sie sicher, dass der Paging-Bereich für die Ausführung einer partitionierten Installation von DB2 ESE groß genug ist. Falls der Paging-Bereich nicht ausreicht, bricht das Betriebssystem den Prozess, der am meisten virtuellen Speicher verwendet, mit dem Befehl 'kill' ab. Dies wäre wahrscheinlich einer der DB2-Prozesse. Um die Größe des verfügbaren Paging-Bereichs zu überprüfen, geben Sie den folgenden Befehl ein:

   ```
   lsps -a
   ```

   Die von diesem Befehl zurückgegebene Ausgabe sieht etwa wie folgt aus:

   ```
   Page Space  Physical Volume  Volume Group  Size  %Used  Active  Auto  Type
   paging00    hdisk1           rootvg        60MB     19     yes   yes    lv
   hd6         hdisk0           rootvg        60MB     21     yes   yes    lv
   hd6         hdisk2           rootvg        64MB     21     yes   yes    lv
   ```

   Der verfügbare Paging-Bereich sollte doppelt so groß sein wie der auf dem Computer installierte physische Speicher.

6. Wenn Sie ein kleines bis mittelgroßes partitioniertes Datenbanksystem erstellen, sollte die Anzahl der Network File System-Dämonen (NFSDs) auf dem Computer, dem die Instanz gehört, etwa dem folgenden Wert entsprechen:

   ```
   Anzahl der biod-Prozesse auf einem Computer * Anzahl der Computer in einer Instanz
   ```

Idealerweise sollten auf jedem Computer 10 biod-Prozesse ausgeführt werden. Nach der oben angegebenen Formel würden auf einem System mit vier Computern mit jeweils 10 biod-Prozessen daher 40 NFSDs verwendet.

Wenn Sie ein größeres System installieren, können Sie bis zu 120 NFSDs auf dem Computer verwenden.

Weitere Informationen zu NFS finden Sie in der NFS-Dokumentation.

Einrichten eines Arbeitsverbunds zum Verteilen von Befehlen an ESE-Workstations (AIX)

In einer Umgebung mit partitionierten Datenbanken unter AIX können Sie einen Arbeitsverbund zum Verteilen von Befehlen an die Gruppe der RS/6000 SP-Workstations einrichten, die dem partitionierten Datenbanksystem angehören. Befehle können mithilfe des Befehls dsh an die Workstations verteilt werden.

Vorbereitung

Diese Funktion kann beim Installieren und Verwalten eines partitionierten Datenbanksystems unter AIX nützlich sein, da auf diese Weise dieselben Befehle schnell auf allen Computern ausgeführt werden können und gleichzeitig weniger Möglichkeiten bestehen, hierbei einen Fehler zu machen.

Sie müssen den Hostnamen jedes Computers kennen, den Sie in den Arbeitsverbund aufnehmen wollen.

Sie müssen als Benutzer mit Rootberechtigung an der Steuerworkstation angemeldet sein.

Vorgehensweise

Sie benötigen eine Datei, die die Hostnamen aller RS/6000 SP-Workstations enthält, die dem partitionierten Datenbanksystem angehören. Um den Arbeitsverbund so zu definieren, dass Befehle an diese Liste der Workstations verteilt werden, gehen Sie wie folgt vor:

1. Erstellen Sie eine Datei mit dem Namen `eeelist.txt`; in dieser Datei werden die Hostnamen aller Workstations aufgelistet, die dem Arbeitsverbund angehören sollen.

   Nehmen Sie beispielsweise an, Sie wollen einen Arbeitsverbund mit den beiden SP-Knoten `workstation1` und `workstation2` erstellen. Der Inhalt der Datei `eeelist.txt` wäre in diesem Fall:

   ```
   workstation1
   workstation2
   ```

2. Aktualisieren Sie die Umgebungsvariable des Arbeitsverbunds. Geben Sie den folgenden Befehl ein, um diese Liste zu aktualisieren:

   ```
   export WCOLL=pfad/eeelist.txt
   ```

   Dabei ist *pfad* die Position, an der die Datei `eeelist.txt` erstellt wurde, und `eeelist.txt` ist der Name der von Ihnen erstellten Datei, die die RS/6000 SP-Workstations im Arbeitsverbund enthält.

3. Stellen Sie sicher, dass die Namen im Arbeitsverbund tatsächlich die Namen der gewünschten Workstations sind. Geben Sie dazu den folgenden Befehl ein:

   ```
   dsh -q
   ```

   Es wird eine Ausgabe angezeigt, die der folgenden ähnelt:

```
Working collective file /eeelist.txt:
   workstation1
   workstation2
   Fanout: 64
```

Prüfen, ob NFS aktiv ist (Linux und UNIX)

Vor dem Einrichten einer Umgebung mit partitionierten Datenbanken müssen Sie prüfen, ob Network File System (NFS) auf allen Computern aktiv ist, die dem partitionierten Datenbanksystem angehören sollen.

NFS muss auf jedem Computer aktiv sein.

Zum Überprüfen, ob NFS auf jedem Computer ausgeführt wird, gehen Sie wie folgt vor:

AIX-Betriebssysteme

Geben Sie auf jedem Computer den folgenden Befehl ein:

```
lssrc -g nfs
```

Das Feld `Status` für die NFS-Prozesse sollte `active` anzeigen.

Nachdem Sie geprüft haben, ob NFS auf jedem System aktiv ist, müssen Sie nach den spezifischen NFS-Prozessen suchen, die für DB2-Produkte erforderlich sind. Hierbei handelt es sich um folgende Prozesse:

```
rpc.lockd
     rpc.statd
```

HP-UX- und Solaris-Betriebssysteme

Geben Sie auf jedem Computer den folgenden Befehl ein:

```
showmount -e hostname
```

Geben Sie den Befehl showmount ohne den Parameter *hostname* ein, um das lokale System zu prüfen.

Ist NFS nicht aktiv, wird eine Nachricht ähnlich der folgenden zurückgegeben:

```
showmount: ServerA: RPC: Program not registered (Programm ist nicht registriert)
```

Nachdem Sie geprüft haben, ob NFS auf jedem System aktiv ist, müssen Sie nach den spezifischen NFS-Prozessen suchen, die für DB2-Produkte erforderlich sind:

```
rpc.lockd
     rpc.statd
```

Sie können diese Prozesse mithilfe der folgenden Befehle suchen:

```
ps -ef | grep rpc.lockd
     ps -ef | grep rpc.statd
```

Linux-Betriebssysteme

Geben Sie auf jedem Computer den folgenden Befehl ein:

```
showmount -e hostname
```

Geben Sie den Befehl showmount ohne den Parameter *hostname* ein, um das lokale System zu prüfen.

Ist NFS nicht aktiv, wird eine Nachricht ähnlich der folgenden zurückgegeben:

```
showmount: ServerA: RPC: Program not registered (Programm ist nicht registriert)
```

Nachdem Sie geprüft haben, ob NFS auf jedem System aktiv ist, müssen Sie nach den spezifischen NFS-Prozessen suchen, die für DB2-Produkte erforderlich sind. Der Prozess `rpc.statd` ist erforderlich.

Mithilfe der Befehle ps -ef | grep rpc.statd kann nach diesem Prozess gesucht werden.

Sind diese Prozesse nicht aktiv, entnehmen Sie bitte weitere Informationen der Dokumentation des Betriebssystems.

Prüfen der Verfügbarkeit des Portbereichs auf zugehörigen Computern (Linux und UNIX)

Diese Task beschreibt die erforderlichen Schritte zum Prüfen der Verfügbarkeit des Anschlussbereichs auf zugehörigen Computern. Der Anschlussbereich wird von Fast Communication Manager (FCM) verwendet. FCM ist eine Funktion von DB2, die die Kommunikation zwischen Datenbankpartitionsservern steuert.

Das Prüfen der Verfügbarkeit des Portbereichs auf zugehörigen Computern sollte erfolgen, nachdem Sie den Datenbankpartitionsserver installiert haben, dem die Instanz gehört, und bevor Sie zugehörige Datenbankpartitionsserver installieren.

Wenn Sie den Datenbankpartitionsserver, dem die Instanz gehört, auf der primären Maschine installieren, reserviert DB2 einen Portbereich entsprechend der angegebenen Anzahl logischer Datenbankpartitionsserver, die der Umgebung mit partitionierten Datenbanken angehören. Der Standardbereich besteht aus vier Anschlüssen. Für jeden Server, der der Umgebung mit partitionierten Datenbanken angehört, müssen Sie die Datei `/etc/services` für die FCM-Ports manuell konfigurieren. Der Bereich der FCM-Ports ist davon abhängig, wie viele logische Partitionen auf dem zugehörigen Computer verwendet werden sollen. Mindestens die beiden Einträge **DB2_<instance>** und **DB2_<instance>_END** sind erforderlich. Weitere Anforderungen für die auf den zugehörigen Computern angegebenen FCM-Ports:

- Die Anfangsportnummer muss mit der Anfangsportnummer des primären Computers übereinstimmen.
- Weitere Ports müssen fortlaufend nummeriert werden.
- Die angegebenen Portnummern müssen frei sein.
-

Um Änderungen an der Datei `services` vornehmen zu können, benötigen Sie die Berechtigung `root`.

Um die Verfügbarkeit des Portbereichs auf zugehörigen Computern zu prüfen, gehen Sie wie folgt vor:

1. Öffnen Sie die Datei `services` im Verzeichnis `/etc/services`.
2. Suchen Sie die Anschlüsse, die für DB2 Fast Communications Manager (FCM) reserviert sind. Die Einträge sollten ungefähr wie folgt aussehen:

```
DB2_db2inst1        60000/tcp
DB2_db2inst1_1       60001/tcp
DB2_db2inst1_2       60002/tcp
DB2_db2inst1_END    60003/tcp
```

DB2 reserviert die ersten vier verfügbaren Anschlüsse nach 60000.

3. Öffnen Sie auf jedem zugehörigen Computer die Datei `services` und prüfen Sie, ob die für DB2 FCM reservierten Anschlüsse in der Datei 'services' des Primärcomputers tatsächlich nicht verwendet werden.
4. Falls die erforderlichen Ports auf einem zugehörigen Computer bereits verwendet werden, ermitteln Sie einen verfügbaren Portbereich für alle Computer und aktualisieren Sie jeweils alle Servicedateien, einschließlich der entsprechenden Datei auf dem Primärcomputer.

Nachdem Sie den Datenbankpartitionsserver, dem die Instanz gehört, auf der primären Maschine installiert haben, müssen Sie Ihr DB2-Produkt auf den zugehörigen Datenbankpartitionsservern installieren. Sie können die für die Partitionierungsserver generierte Antwortdatei verwenden (Standardname ist db2ese_addpart.rsp), müssen jedoch die Dateien in /etc/services manuell für die FCM-Ports konfigurieren. Der Bereich der FCM-Ports ist davon abhängig, wie viele logische Partitionen auf dem aktuellen Computer verwendet werden sollen. Mindestens die beiden Einträge für DB2_ und DB2__END mit fortlaufenden freien Portnummern sind erforderlich. Die auf den einzelnen zugehörigen Maschinen verwendeten FCM-Portnummern müssen über dieselben Anfangsportnummern verfügen und die weiteren Ports müssen fortlaufend nummeriert werden.

Erstellen eines Dateisystems für ein partitioniertes Datenbanksystem (Linux)

Im Rahmen der Konfiguration Ihres partitionierten Datenbanksystems unter Linux-Betriebssystemen müssen Sie ein DB2-Ausgangsdateisystem erstellen. Dann müssen Sie das Ausgangsdateisystem über NFS exportieren und für jeden Computer, der am partitionierten Datenbanksystem beteiligt ist, anhängen.

Informationen zu dieser Task

Sie benötigen ein Dateisystem, auf das alle Maschinen, die dem partitionierten Datenbanksystem angehören, zugreifen können. Dieses Dateisystem wird als Ausgangsverzeichnis der Instanz verwendet.

Für Konfigurationen, die mehr als eine Maschine für eine einzelne Datenbankinstanz verwenden, wird NFS (Network File System) verwendet, um dieses Dateisystem gemeinsam benutzbar zu machen. Normalerweise wird eine Maschine im Cluster verwendet, um das Dateisystem mit NFS zu exportieren, und die übrigen Maschinen im Cluster hängen das NFS-Dateisystem von dieser Maschine aus an. Auf der Maschine, die das Dateisystem exportiert, wird das Dateisystem lokal angehängt.

Die Dokumentation der entsprechenden Linux-Variante enthält weitere Informationen zu Befehlen.

Vorgehensweise

Um das DB2-Ausgangsdateisystem zu erstellen sowie in NFS zu exportieren und anzuhängen, gehen Sie wie folgt vor:

1. Wählen Sie auf einer Maschine eine Plattenpartition aus oder erstellen Sie eine Partition mithilfe des Befehls fdisk.
2. Verwenden Sie ein Dienstprogramm wie mkfs, und erstellen Sie auf dieser Partition ein Dateisystem. Das Dateisystem sollte groß genug sein, um die erforderlichen DB2-Programmdateien aufnehmen zu können, und über ausreichend Speicherkapazität für Ihre Datenbankanforderungen verfügen.

3. Hängen Sie das soeben erstellte Dateisystem lokal an und fügen Sie einen Eintrag zur Datei /etc/fstab hinzu, damit dieses Dateisystem bei jedem Neustart des Systems angehängt wird. Beispiel:

   ```
   /dev/hda1    /db2home    ext3   defaults   1 2
   ```

4. Fügen Sie einen Eintrag zur Datei /etc/exports hinzu, damit ein NFS-Dateisystem unter Linux bei jedem Neustart automatisch exportiert wird. In diesem Eintrag müssen alle Hostnamen, die dem Cluster angehören, sowie alle Namen, unter denen die Maschine bekannt sein könnte, enthalten sein. Stellen Sie außerdem sicher, dass jede Maschine im Cluster über Rootberechtigung für das exportierte Dateisystem verfügt, indem Sie die Option root" verwenden.

   Die Datei /etc/exports liegt im ASCII-Format vor und enthält folgende Informationen:

   ```
   /db2home  name_maschine1(rw)  name_maschine2(rw)
   ```

   Führen Sie zum Exportieren des NFS-Verzeichnisses folgenden Befehl aus:

   ```
   /usr/sbin/exportfs -r
   ```

5. Fügen Sie auf allen übrigen Maschinen im Cluster einen Eintrag zur Datei /etc/fstab hinzu, damit das Dateisystem beim Systemstart automatisch über NFS angehängt wird. Stellen Sie bei der Angabe der Optionen für den Mountpunkt sicher, dass das Dateisystem beim Systemstart angehängt wird, dass es über den Zugriff rw (Lesen und Schreiben) verfügt, dass ein absoluter Mount (Hard Mount) ausgeführt wird, dass die Option bg (Hintergrund) verwendet wird und dass setuid-Programme korrekt ausgeführt werden können. Diese Optionen werden im folgenden Beispiel dargestellt.

   ```
   fusion-en:/db2home /db2home nfs rw,timeo=7,
         hard,intr,bg,suid,lock
   ```

   Hierbei ist *fusion-en* der Name der Maschine.

6. Hängen Sie das exportierte Dateisystem über NFS auf allen verbleibenden Maschinen im Cluster an. Geben Sie den folgenden Befehl ein:

   ```
   mount /db2home
   ```

   Schlägt der Befehl mount fehl, können Sie den Befehl showmount verwenden, um den Status des NFS-Servers zu überprüfen. Beispiel:

   ```
   showmount -e fusion-en
   ```

   Hierbei ist *fusion-en* der Name der Maschine.

   Diese Version des Befehls showmount sollte die Dateisysteme auflisten, die aus der Maschine mit dem Namen fusion-en exportiert wurden. Wenn dieser Befehl fehlschlägt, wurde möglicherweise der NFS-Server nicht gestartet. Um den Server manuell zu starten, führen Sie den folgenden Befehl als Benutzer mit Rootberechtigung auf dem NFS-Server aus:

   ```
   /etc/rc.d/init.d/nfs restart
   ```

   Vorausgesetzt, die derzeitige Ausführungsebene ist 3, können Sie diesen Befehl beim Systemstart automatisch ausführen lassen, indem Sie im nachfolgenden Verzeichnis K20nfs in S20nfs umbenennen: /etc/rc.d/rc3.d.

Ergebnisse

Mit diesen Schritten haben Sie die folgenden Tasks ausgeführt:

1. Sie haben auf einem einzelnen Computer in der partitionierten Datenbankumgebung ein Dateisystem erstellt, das als Instanz- und Ausgangsverzeichnis verwendet wird.

2. Im Falle einer Konfiguration, die mehr als eine Maschine für eine einzelne Datenbankinstanz verwendet, haben Sie dieses Dateisystem über NFS exportiert.
3. Sie haben das exportierte Dateisystem auf jedem zugehörigen Computer im System angehängt.

Erstellen eines DB2-Ausgangsdateisystems für ein partitioniertes Datenbanksystem (AIX)

Im Rahmen der Konfiguration Ihres partitionierten Datenbanksystems müssen Sie ein DB2-Ausgangsdateisystem erstellen. Dann müssen Sie das Ausgangsdateisystem über NFS exportieren und für jeden Computer, der am partitionierten Datenbanksystem beteiligt ist, anhängen.

Vorbereitung

Es wird empfohlen, ein Ausgangsdateisystem zu erstellen, das mindestens den Inhalt der DB2-Datenbankprodukt-DVD aufnehmen kann. Sie können die Größe (angegeben in KB) mit dem folgenden Befehl ermitteln:

```
du -sk DVD-Mountpunkt
```

Eine DB2-Instanz benötigt einen Speicherbereich von mindestens 50 MB. Falls der freie Speicherbereich nicht ausreicht, können Sie die DB2-Datenbankprodukt-DVD auch über jeden Computer im System anhängen, anstatt ihren Inhalt auf die Festplatte zu kopieren.

Sie müssen:

- über die Berechtigung `Root` verfügen, um ein Dateisystem erstellen zu können.
- eine Datenträgergruppe erstellt haben, in der Ihr Dateisystem physisch vorhanden sein soll.

Vorgehensweise

Um das DB2-Ausgangsdateisystem zu erstellen sowie in NFS zu exportieren und anzuhängen, gehen Sie wie folgt vor:

1. Erstellen Sie das DB2-Ausgangsdateisystem.

   Melden Sie sich als Benutzer mit der Berechtigung `Root` am Primärcomputer (ServerA) Ihres partitionierten Datenbanksystems an, und erstellen Sie ein Ausgangsdateisystem mit dem Namen `/db2home` für dieses partitionierte Datenbanksystem.

   a. Geben Sie den Befehl smit jfs ein.
   b. Klicken Sie das Symbol **Add a Journaled File System** an.
   c. Klicken Sie das Symbol **Add a Standard Journaled File System** an.
   d. Wählen Sie in der Liste **Volume Group Name** eine vorhandene Datenträgergruppe aus, in der sich das Dateisystem physisch befinden soll.
   e. Definieren Sie das Feld für die Größe des Dateisystems **SIZE of file system (in 512–byte blocks) (Num.)**. Diese Größe ist nach 512-Byte-Blöcken nummeriert. Wenn Sie lediglich ein Dateisystem für das Ausgangsverzeichnis der Instanz erstellen müssen, empfiehlt es sich, einen Wert von 180 000 (etwa 90 MB) zu verwenden. Wenn Sie das Image der Produkt-CD kopieren müssen, um die Installation auszuführen, können Sie es mit dem Wert 2 000 000 (etwa 1 GB) erstellen.
   f. Geben Sie im Feld **MOUNT POINT** einen Mountpunkt für dieses Dateisystem ein. Im vorliegenden Beispiel ist der Mountpunkt `/db2home`.

g. Setzen Sie das Feld **Mount AUTOMATICALLY at system restart** auf `yes`.

   Für die übrigen Felder können die Standardeinstellungen belassen werden.

h. Klicken Sie **OK** an.

2. Exportieren Sie das DB2-Ausgangsdateisystem.

   Exportieren Sie das Dateisystem `/db2home` über NFS, damit alle Computer, die dem partitionierten Datenbanksystem angehören sollen, darauf zugreifen können.

   a. Geben Sie den Befehl smit nfs ein.
   b. Klicken Sie das Symbol **Network File System (NFS)** an.
   c. Klicken Sie das Symbol **Add a Directory to Exports List** an.
   d. Geben Sie den Pfadnamen und das zu exportierende Verzeichnis (beispielsweise `/db2home`) in das Feld **PATHNAME of directory to export** ein.
   e. Geben Sie in das Feld **HOSTS allowed root access** den Namen aller Workstations ein, die dem partitionierten Datenbanksystem angehören sollen. Verwenden Sie ein Komma (,) als Begrenzer zwischen den einzelnen Namen. Beispiel: `ServerA, ServerB, ServerC`. Falls Sie eine Hochgeschwindigkeitsverbindung verwenden, wird empfohlen, dass Sie in diesem Feld auch die Namen der einzelnen Workstations in der Hochgeschwindigkeitsverbindung angeben. Für die verbleibenden Felder können die Standardeinstellungen beibehalten werden.
   f. Klicken Sie **OK** an.

3. Melden Sie sich ab.

4. Hängen Sie das DB2-Ausgangsdateisystem für jeden beteiligten Computer an.

   Melden Sie sich an *jedem* Computer (ServerB, ServerC, ServerD), der dem partitionierten Datenbanksystem angehören soll, an und hängen Sie das soeben exportierte Dateisystem über NFS an. Führen Sie hierzu die folgenden Schritte aus:

   a. Geben Sie den Befehl smit nfs ein.
   b. Klicken Sie das Symbol **Network File System (NFS)** an.
   c. Klicken Sie das Symbol **Add a File System for Mounting** an.
   d. Geben Sie im Feld **PATHNAME of the mount point (Path)** den Pfadnamen des Mountpunkts ein.

      Der Pfadname des Mountpunkts ist die Speicherposition, an der Sie das DB2-Ausgangsverzeichnis erstellen sollten. Verwenden Sie in diesem Beispiel `/db2home`.

   e. Geben Sie im Feld **PATHNAME of the remote directory** den Pfadnamen des fernen Verzeichnisses ein.

      Im vorliegenden Beispiel sollten Sie den gleichen Wert eingeben wie im Feld **PATHNAME of the mount point (Path)**.

   f. Geben Sie im Feld **HOST where the remote directory resides** den Hostnamen (*hostname*) der Maschine ein, auf die Sie das Dateisystem exportiert haben.

      Dies ist der Hostname der Maschine, auf der das Dateisystem, das Sie anhängen, erstellt wurde.

      Zum Verbessern der Leistung ist es möglicherweise empfehlenswert, das soeben erstellte Dateisystem über eine Hochgeschwindigkeitsverbindung über NFS anzuhängen. Wenn Sie das Dateisystem über eine Hochgeschwindigkeitsverbindung anhängen wollen, müssen Sie seinen Namen im Feld **HOST where remote directory resides** eingeben.

Bitte beachten Sie, dass eine eventuelle Nichtverfügbarkeit der Hochgeschwindigkeitsverbindung dazu führt, dass jede Workstation, die dem partitionierten Datenbanksystem angehört, den Zugriff auf das DB2-Ausgangsverzeichnis verliert.

g. Setzen Sie den Wert im Feld **MOUNT now, add entry to /etc/filesystems or both?** auf `both`.
h. Setzen Sie den Wert im Feld **/etc/filesystems entry will mount the directory on system RESTART** auf `yes`.
i. Setzen Sie den Wert im Feld **MODE for this NFS file system** auf `read-write`.
j. Setzen Sie den Wert im Feld **Mount file system soft or hard** auf `hard`.

   Bei einem bedingten Mount (Soft Mount) versucht der Computer *nicht* auf unbegrenzte Zeit, das Verzeichnis fern anzuhängen. Bei einem absoluten Mount (Hard Mount) versucht die Maschine auf unbegrenzte Zeit, das Verzeichnis anzuhängen. Dies könnte bei einem Systemabsturz zu Problemen führen. Es wird empfohlen, dieses Feld auf `hard` (absoluter Mount) zu setzen.

   Für die verbleibenden Felder können die Standardeinstellungen beibehalten werden.
k. Stellen Sie sicher, dass das Dateisystem mit dem Wert `Yes` im Feld **Allow execution of SUID and sgid programs in this file system?** angehängt wird. Dies ist die Standardeinstellung.
l. Klicken Sie **OK** an.
m. Melden Sie sich ab.

Erstellen erforderlicher Benutzer für die Installation eines DB2-Servers in einer Umgebung mit partitionierten Datenbanken (Linux)

Für den Betrieb einer DB2-Datenbank sind jeweils drei Benutzer und Gruppen erforderlich. Die folgende Tabelle enthält die Namen der Benutzer und Gruppen, die in den nachstehenden Anweisungen verwendet werden. Sie können Ihre eigenen Benutzer- und Gruppennamen angeben, sofern diese den Namensregeln für das System und für DB2 entsprechen.

Wenn Sie beabsichtigen, den DB2-Installationsassistenten zum Installieren des DB2-Produkts zu verwenden, werden diese Benutzer vom DB2-Installationsassistenten erstellt.

Tabelle 10. Erforderliche Benutzer und Gruppen

Erforderlicher Benutzer	Benutzername	Gruppenname
Instanzeigner	db2inst1	db2iadm1
Abgeschirmter Benutzer	db2fenc1	db2fadm1
Benutzer des DB2-Verwaltungsservers	dasusr1	dasadm1

Falls der Benutzer des DB2-Verwaltungsservers bereits vorhanden ist, muss er vor der Installation auf allen zugehörigen Computern ebenfalls vorhanden sein. Wenn Sie den DB2-Installationsassistenten verwenden, um einen neuen Benutzer für den DB2-Verwaltungsserver auf dem Computer zu erstellen, der als Instanzeigner fungiert, wird dieser Benutzer (falls erforderlich) bei Installationen mit einer Antwort-

datei auf den zugehörigen Computern ebenfalls erstellt. Ist der Benutzer auf den zugehörigen Computern bereits vorhanden, muss er über dieselbe Primärgruppe verfügen.

Voraussetzungen

- Um Benutzer und Gruppen erstellen zu können, müssen Sie über die Berechtigung Root verfügen.
- Wenn Sie Benutzer und Gruppen mit Sicherheitssoftware verwalten, sind beim Definieren von DB2-Benutzern und -Gruppen möglicherweise zusätzliche Schritte erforderlich.

Einschränkung

Die von Ihnen erstellten Benutzernamen müssen sowohl den Namensregeln Ihres Betriebssystems als auch den DB2-Namensregeln entsprechen.

Führen Sie die folgenden Schritte aus, um alle drei genannten Benutzer zu erstellen:

1. Melden Sie sich am Primärcomputer an.
2. Erstellen Sie je eine Gruppe für den Instanzeigner (z. B. db2iadm1), die Gruppe, die abgeschirmte UDFs oder gespeicherte Prozeduren ausführt (z. B. db2fadm1), und die Gruppe, der der DB2-Verwaltungsserver gehören soll (z. B. dasadm1). Geben Sie hierzu die folgenden Befehle ein:

   ```
   groupadd -g 999 db2iadm1
   groupadd -g 998 db2fadm1
      groupadd -g 997 dasadm1
   ```

   Stellen Sie sicher, dass die spezifischen Nummern, die Sie verwenden, derzeit auf keiner der Maschinen vorhanden sind.
3. Erstellen Sie einen Benutzer für jede Gruppe, die Sie im vorigen Schritt erstellt haben. Verwenden Sie hierzu die nachstehenden Befehle. Das Ausgangsverzeichnis für alle Benutzer ist das DB2-Ausgangsverzeichnis, das Sie zuvor erstellt und zur gemeinsamen Benutzung verfügbar gemacht haben (db2home).

   ```
   useradd -u 1004 -g db2iadm1 -m -d /db2home/db2inst1 db2inst1
   useradd -u 1003 -g db2fadm1 -m -d /db2home/db2fenc1 db2fenc1
   useradd -u 1002 -g dasadm1 -m -d /home/dasusr1 dasusr1
   ```
4. Definieren Sie ein Anfangskennwort für jeden Benutzer, den Sie erstellt haben, indem Sie die folgenden Befehle eingeben:

   ```
   passwd db2inst1    passwd db2fenc1    passwd dasusr1
   ```
5. Melden Sie sich ab.
6. Melden Sie sich am Primärcomputer als jeder der von Ihnen erstellten Benutzer an (db2inst1, db2fenc1 und dasusr1). Sie erhalten für jeden Benutzer möglicherweise die Aufforderung, das Kennwort zu ändern, da sich diese Benutzer zum ersten Mal am System angemeldet haben.
7. Melden Sie sich ab.
8. Erstellen Sie auf jedem Computer, der Ihrer Umgebung mit partitionierten Datenbanken angehören soll, genau dieselben Benutzer- und Gruppenkonten.

Erstellen erforderlicher Benutzer für die Installation eines DB2-Servers in einer Umgebung mit partitionierten Datenbanken (AIX)

Für den Betrieb einer DB2-Datenbank sind jeweils drei Benutzer und Gruppen erforderlich. Die folgende Tabelle enthält die Namen der Benutzer und Gruppen, die

in den nachstehenden Anweisungen verwendet werden. Sie können Ihre eigenen Benutzer- und Gruppennamen angeben, sofern diese den Namensregeln für das System und für DB2 entsprechen.

Wenn Sie beabsichtigen, den DB2-Installationsassistenten zum Installieren des DB2-Produkts zu verwenden, werden diese Benutzer vom DB2-Installationsassistenten erstellt.

Tabelle 11. Erforderliche Benutzer und Gruppen

Erforderlicher Benutzer	Benutzername	Gruppenname
Instanzeigner	db2inst1	db2iadm1
Abgeschirmter Benutzer	db2fenc1	db2fadm1
Benutzer des DB2-Verwaltungsservers	dasusr1	dasadm1

Falls der Benutzer des DB2-Verwaltungsservers bereits vorhanden ist, muss er vor der Installation auf allen zugehörigen Computern ebenfalls vorhanden sein. Wenn Sie den DB2-Installationsassistenten verwenden, um einen neuen Benutzer für den DB2-Verwaltungsserver auf dem Computer zu erstellen, der als Instanzeigner fungiert, wird dieser Benutzer (falls erforderlich) bei Installationen mit einer Antwortdatei auf den zugehörigen Computern ebenfalls erstellt. Ist der Benutzer auf den zugehörigen Computern bereits vorhanden, muss er über dieselbe Primärgruppe verfügen.

Voraussetzungen

- Um Benutzer und Gruppen erstellen zu können, müssen Sie über die Berechtigung `Root` verfügen.
- Wenn Sie Benutzer und Gruppen mit Sicherheitssoftware verwalten, sind beim Definieren von DB2-Benutzern und -Gruppen möglicherweise zusätzliche Schritte erforderlich.

Einschränkung

Die von Ihnen erstellten Benutzernamen müssen sowohl den Namensregeln Ihres Betriebssystems als auch den DB2-Namensregeln entsprechen.

Führen Sie die folgenden Schritte aus, um alle drei genannten Benutzer zu erstellen:

1. Melden Sie sich am Primärcomputer an.
2. Erstellen Sie je eine Gruppe für den Instanzeigner (z. B. `db2iadm1`), die Gruppe, die abgeschirmte UDFs oder gespeicherte Prozeduren ausführt (z. B. `db2fadm1`), und die Gruppe, der der DB2-Verwaltungsserver gehören soll (z. B. `dasadm1`). Geben Sie hierzu die folgenden Befehle ein:

   ```
   mkgroup id=999 db2iadm1
      mkgroup id=998 db2fadm1
      mkgroup id=997 dasadm1
   ```

3. Erstellen Sie einen Benutzer für jede Gruppe, die Sie im vorigen Schritt erstellt haben. Verwenden Sie hierzu die nachstehenden Befehle. Das Ausgangsverzeichnis für alle Benutzer ist das DB2-Ausgangsverzeichnis, das Sie zuvor erstellt und zur gemeinsamen Benutzung verfügbar gemacht haben (`db2home`).

   ```
   mkuser id=1004 pgrp=db2iadm1 groups=db2iadm1 home=/db2home/db2inst1
         core=-1 data=491519 stack=32767 rss=-1 fsize=-1 db2inst1
   mkuser id=1003 pgrp=db2fadm1 groups=db2fadm1 home=/db2home/db2fenc1
      db2fenc1
   mkuser id=1002 pgrp=dasadm1 groups=dasadm1 home=/home/dasusr1
      dasusr1
   ```

4. Definieren Sie ein Anfangskennwort für jeden Benutzer, den Sie erstellt haben, indem Sie die folgenden Befehle eingeben:

   ```
   passwd db2inst1
      passwd db2fenc1
      passwd dasusr1
   ```

5. Melden Sie sich ab.
6. Melden Sie sich am Primärcomputer als jeder der von Ihnen erstellten Benutzer an (`db2inst1`, `db2fenc1` und `dasusr1`). Sie erhalten für jeden Benutzer möglicherweise die Aufforderung, das Kennwort zu ändern, da sich diese Benutzer zum ersten Mal am System angemeldet haben.
7. Melden Sie sich ab.
8. Erstellen Sie auf jedem Computer, der Ihrer Umgebung mit partitionierten Datenbanken angehören soll, genau dieselben Benutzer- und Gruppenkonten.

Kapitel 7. Installieren des DB2-Serverprodukts

Einrichten einer Umgebung mit partitionierten Datenbanken

In diesem Abschnitt wird beschrieben, wie Sie eine Umgebung mit partitionierten Datenbanken einrichten. Sie verwenden den **DB2-Installationsassistenten**, um den Datenbankserver zu installieren, der Eigner der Instanz ist, und um die Antwortdateien zu erstellen, die zum Erstellen der zugehörigen Datenbankserver verwendet werden.

Vorbereitung

Anmerkung: Eine Umgebung mit partitionierten Datenbanken wird in nicht als Root ausgeführten Installationen nicht unterstützt.

- Stellen Sie sicher, dass Sie über den Lizenzschlüssel der Aktivierungs-CD für InfoSphere Warehouse verfügen, der auf alle beteiligten Computer kopiert werden muss.
- Auf jedem Computer, der der Umgebung mit partitionierten Datenbanken angehören soll, muss die gleiche Anzahl aufeinanderfolgender Ports frei sein. Wenn die Umgebung mit partitionierten Datenbanken beispielsweise vier Computer umfassen soll, müssen auf jedem der vier Computer die gleichen vier aufeinanderfolgenden Ports frei sein. Bei der Instanzerstellung wird eine bestimmte Anzahl von Ports, die der Anzahl der logischen Partitionen auf dem aktuellen Server entspricht, im Verzeichnis `/etc/services` unter Linux und UNIX bzw. im Verzeichnis `%SystemRoot%\system32\drivers\etc\services` unter Windows reserviert. Diese Ports werden von Fast Communication Manager (FCM) verwendet. Die reservierten Ports besitzen das folgende Format:

  ```
  DB2_InstanceName
  DB2_InstanceName_1
  DB2_InstanceName_2
  DB2_InstanceName_END
  ```

  Lediglich die Einträge des Ports am Anfang (DB2_InstanceName) und am Ende (DB2_InstanceName_END) sind verbindlich. Die anderen Einträge werden in der Servicedatei reserviert, damit die betreffenden Ports nicht von anderen Anwendungen genutzt werden.
- Zur Unterstützung mehrerer zugehöriger DB2-Datenbankserver muss der Computer, auf dem DB2 installiert werden soll, zu einer Domäne gehören, auf die Zugriff besteht. Sie können diesem Computer jedoch auch dann lokale Partitionen hinzufügen, wenn er keiner Domäne angehört.
- Unter Linux- und UNIX-Systemen ist für partitionierte Datenbanksysteme ein Dienstprogramm für eine ferne Shell erforderlich. DB2-Datenbanksysteme unterstützen die folgenden Dienstprogramme für ferne Shells:
  - rsh
  - ssh

  DB2Datenbanksysteme verwenden standardmäßig 'rsh' für die Ausführung von Befehlen auf fernen DB2-Knoten, zum Beispiel zum Starten einer fernen DB2-Datenbankpartition. Zum Verwenden des DB2-Standards muss das Paket 'rsh-server' installiert sein. Weitere Informationen finden Sie im Abschnitt „Sicherheitsaspekte bei Installation und Verwendung des DB2-Datenbankmanagers” im Handbuch *Datenbanksicherheit*.

Wenn Sie das Dienstprogramm 'rsh' für die ferne Shell ausführen möchten, muss auch 'inetd' (oder 'xinetd') installiert und aktiv sein. Wenn Sie das Dienstprogramm 'ssh' für die ferne Shell ausführen möchten, müssen Sie die Registrierdatenbankvariable **DB2RSHCMD** sofort nach dem Abschluss der Installation von DB2 festlegen. Wenn diese Registrierdatenbankvariable nicht eingestellt wird, wird 'rsh' verwendet.

- Stellen Sie unter Linux- und UNIX-Betriebssystemen sicher, dass die Datei `hosts` im Verzeichnis `etc` keinen Eintrag für „127.0.0.2" enthält, wenn die IP-Adresse dem vollständig qualifizierten Hostnamen der Maschine zugeordnet ist.

Anmerkung: Die Verwendung von XML-Funktionen schließt die spätere Verwendung einer Umgebung mit partitionierten Datenbanken aus.

Informationen zu dieser Task

Eine Datenbankpartition ist Bestandteil einer Datenbank, die aus eigenen Daten, Indizes, Konfigurationsdateien und Transaktionsprotokollen besteht. Eine partitionierte Datenbank ist eine Datenbank, die aus mindestens zwei Partitionen besteht.

Vorgehensweise

Gehen Sie wie folgt vor, um eine Umgebung mit partitionierten Datenbanken einzurichten:

1. Installieren Sie den Datenbankserver, der Eigner der Datenbank ist, mithilfe des **DB2-Installationsassistenten**. Detaillierte Anweisungen finden Sie im entsprechenden Abschnitt zur Installation von DB2-Servern für Ihre Plattform.
   - Stellen Sie im Fenster **Installation und/oder Antwortdateierstellung auswählen** sicher, dass die Option **Installationseinstellungen in einer Antwortdatei speichern** ausgewählt ist. Nach Abschluss der Installation werden zwei Dateien in das Verzeichnis kopiert, das im **DB2-Installationsassistenten** angegeben wurde: `PROD_ESE.rsp` und `PROD_ESE_addpart.rsp`. Die Datei `PROD_ESE.rsp` ist die Antwortdatei der Datenbankserver, die Eigner einer Instanz sind. Die Datei `PROD_ESE_addpart.rsp` ist die Antwortdatei für zugehörige Datenbankserver.
   - Stellen Sie im Fenster **Partitionierungsoptionen für die DB2-Instanz konfigurieren** sicher, dass die Option **Mehrfachpartitionsinstanz** ausgewählt ist, und geben Sie die maximale Anzahl für die logischen Partitionen ein.
2. Machen Sie das DB2-Installationsimage für alle zugehörigen Computer in der Umgebung mit partitionierten Datenbanken verfügbar.
3. Verteilen Sie die Antwortdatei der zugehörigen Datenbankserver (`PROD_ESE_addpart.rsp`).
4. Installieren Sie einen DB2-Datenbankserver auf jedem beteiligten Computer mit dem Befehl db2setup unter Linux und UNIX bzw. mit dem Befehl setup unter Windows:

   Linux und UNIX

   Wechseln Sie in das Verzeichnis, in dem der DB2-Datenbankproduktcode verfügbar ist, und führen Sie folgenden Befehl aus:

   ```
   ./db2setup -r /verzeichnis_der_antwortdatei/name_der_antwortdatei
   ```

   Windows

   ```
   setup -u x:\verzeichnis_der_antwortdatei\name_der_antwortdatei
   ```

   Der Befehl, der die Antwortdatei `PROD_ESE_addpart.rsp` verwendet, würde dann wie folgt lauten:

Linux und UNIX

Wechseln Sie in das Verzeichnis, in dem der DB2-Datenbankproduktcode verfügbar ist, und führen Sie folgenden Befehl aus:

```
./db2setup -r /db2home/PROD_ESE_addpart.rsp
```

Dabei steht `/db2home` für das Verzeichnis, in das Sie die Antwortdatei kopiert haben.

Windows

```
setup -u c:\resp_files\PROD_ESE_addpart.rsp
```

Dabei steht `c:\resp_files\` für das Verzeichnis, in das Sie die Antwortdatei kopiert haben.

5. (Nur Linux und UNIX) Konfigurieren Sie die Datei `db2nodes.cfg`. Bei der DB2-Installation wird nur die maximale Anzahl logischer Partitionen reserviert, die Sie für den aktuellen Computer verwenden möchten. Die Datei `db2nodes.cfg` wird nicht konfiguriert. Wenn Sie die Datei `db2nodes.cfg` nicht konfigurieren, bleibt die Instanz weiterhin eine Instanz mit einer einzigen Partition.
6. Aktualisieren Sie die Datei `services` auf den beteiligten Servern, um den entsprechenden FCM-Port für die DB2-Instanz zu definieren. Die Servicedatei befindet sich an der folgenden Position:
   - `/etc/services` unter Linux und UNIX
   - `%SystemRoot%\system32\drivers\etc\services` unter Windows

Installieren von Datenbankpartitionsservern auf zugehörigen Computern mithilfe einer Antwortdatei (Windows)

In dieser Task wird die Antwortdatei verwendet, die Sie mithilfe des DB2-Installationsassistenten erstellt haben, um Datenbankpartitionsserver auf zugehörigen Computern zu installieren.

Voraussetzungen

- Sie haben eine DB2-Kopie mithilfe des DB2-Installationsassistenten auf dem primären Computer installiert.
- Sie haben eine Antwortdatei für die Installation auf zugehörigen Computern erstellt und auf den zugehörigen Computer kopiert.
- Sie benötigen Administratorberechtigung für die zugehörigen Computer.
- Der Inhalt der DB2-Produkt-DVD wurde auf den zugehörigen Computer kopiert.

Um zusätzliche Datenbankpartitionsserver mithilfe einer Antwortdatei zu installieren, gehen Sie wie folgt vor:

1. Melden Sie sich an dem Computer, der der Umgebung mit partitionierten Datenbanken angehören soll, mit dem für die Installation von DB2 definierten Benutzerkonto des lokalen Administrators an.
2. Wechseln Sie in das Verzeichnis, in das Sie den Inhalt der DB2-Produkt-DVD kopiert haben. Beispiel:

   ```
   cd c:\db2dvd
   ```

   Dabei ist `db2dvd` der Name des Verzeichnisses, in das Sie den Inhalt der DB2-Produkt-DVD kopiert haben.
3. Geben Sie an einer Eingabeaufforderung den Befehl setup wie folgt ein:

   ```
   setup -u verzeichnis_der_antwortdatei\name_der_antwortdatei
   ```

Im folgenden Beispiel befindet sich die Antwortdatei `Addpart.file` im Verzeichnis `c:\antwortdatei`. Der entsprechende Befehl lautet wie folgt:

```
setup -u c:\antwortdatei\Addpart.file
```

4. Überprüfen Sie nach Abschluss der Installation die Nachrichten in der Protokolldatei. Die Protokolldatei befindet sich im Verzeichnis `Eigene Dateien\DB2LOG\`. Am Ende der Protokolldatei müssten Ausgabedaten ähnlich den folgenden stehen:

   ```
   === Logging stopped: 5/9/2007  10:41:32 ===
   MSI (c) (C0:A8) [10:41:32:984]: Product: DB2
   Enterprise Server Edition - DB2COPY1 -- Installation
   operation completed successfully.
   ```

5. Wenn Sie den Datenbankpartitionsserver, dem die Instanz gehört, auf der primären Maschine installieren, reserviert das DB2-Produkt einen Anschlussbereich entsprechend der angegebenen Anzahl logischer Datenbankpartitionsserver, die der Umgebung mit partitionierten Datenbanken angehören. Der Standardbereich besteht aus vier Anschlüssen. Für jeden Server, der der Umgebung mit partitionierten Datenbanken angehört, müssen Sie die Datei `/etc/services` für die FCM-Ports manuell konfigurieren. Der Bereich der FCM-Ports ist davon abhängig, wie viele logische Partitionen auf dem zugehörigen Computer verwendet werden sollen. Mindestens die beiden Einträge **DB2_<instance>** und **DB2_<instance>_END** sind erforderlich. Weitere Anforderungen für die auf den zugehörigen Computern angegebenen FCM-Ports:
   - Die Anfangsportnummer muss mit der Anfangsportnummer des primären Computers übereinstimmen.
   - Weitere Ports müssen fortlaufend nummeriert werden.
   - Die angegebenen Portnummern müssen frei sein.

Sie müssen sich an jedem einzelnen zugehörigen Computer anmelden und dort diese Schritte wiederholen.

Wenn Sie möchten, dass Ihr DB2-Produkt auf die DB2-Dokumentation auf dem lokalen Computer oder auf einem anderen Computer im Netzwerk zugreifen kann, müssen Sie die DB2-Informationszentrale installieren. Die DB2-Informationszentrale enthält die Dokumentation für das DB2-Datenbanksystem und die zugehörigen DB2-Produkte.

Installieren von Datenbankpartitionsservern auf zugehörigen Computern mittels einer Antwortdatei (Linux und UNIX)

In dieser Task wird die Antwortdatei verwendet, die Sie mithilfe des DB2-Installationsassistenten erstellt haben, um Datenbankpartitionsserver auf zugehörigen Computern zu installieren.

Vorbereitung

- Sie haben mit dem **DB2-Installationsassistenten** ein DB2-Datenbankprodukt auf dem primären Computer installiert und eine Antwortdatei für die Installation auf zugehörigen Computern erstellt.
- Sie benötigen Rootberechtigung für die zugehörigen Computer.

Vorgehensweise

Um zusätzliche Datenbankpartitionsserver mithilfe einer Antwortdatei zu installieren, gehen Sie wie folgt vor:

1. Melden Sie sich als 'Root' an einem Computer an, der der Umgebung mit partitionierten Datenbanken angehören soll.
2. Wechseln Sie in das Verzeichnis, in das Sie den Inhalt der DB2-Datenbankprodukt-DVD kopiert haben. Beispiel:

   ```
   cd /db2home/db2dvd
   ```

3. Geben Sie den Befehl db2setup wie folgt ein:

   ```
   ./db2setup -r /verzeichnis_der_antwortdatei/name_der_antwortdatei
   ```

   Im vorliegenden Beispiel wurde die Antwortdatei `'AddPartitionResponse.file'` im Verzeichnis `'/db2home'` gespeichert. In diesem Fall lautet der Befehl wie folgt:

   ```
   ./db2setup -r /db2home/AddPartitionResponse.file
   ```

4. Überprüfen Sie nach Abschluss der Installation die Nachrichten in der Protokolldatei.

Ergebnisse

Sie müssen sich an jedem einzelnen zugehörigen Computer anmelden und dort jeweils die Installation mithilfe der Antwortdatei vornehmen.

Weitere Schritte

Wenn Sie möchten, dass Ihr DB2-Datenbankprodukt auf die DB2-Datenbankdokumentation auf dem lokalen Computer oder auf einem anderen Computer im Netz zugreifen kann, müssen Sie die *DB2-Informationszentrale* installieren. Die *DB2-Informationszentrale* enthält die Dokumentation für das DB2-Datenbanksystem und die zugehörigen DB2-Datenbankprodukte.

Kapitel 8. Installationsnachbereitung

Prüfen der Installation

Prüfen der Installation einer Umgebung mit partitionierten Datenbanken (Windows)

Um zu prüfen, ob Ihr DB2-Server erfolgreich installiert wurde, erstellen Sie eine Beispieldatenbank und führen SQL-Befehle aus, um Beispieldaten abzurufen und um zu prüfen, ob die Daten an alle zugehörigen Datenbankpartitionsserver der Installation verteilt wurden.

Sie haben alle Installationsschritte vollständig ausgeführt.

Um eine Beispieldatenbank (SAMPLE) zu erstellen, gehen Sie wie folgt vor:

1. Melden Sie sich am Primärcomputer (ServerA) als Benutzer mit der Berechtigung SYSADM an.
2. Geben Sie den Befehl db2sampl ein, um die Beispieldatenbank (SAMPLE) zu erstellen.

   Die Verarbeitung dieses Befehls kann einige Minuten in Anspruch nehmen. Wenn die Eingabeaufforderung des Befehls wieder angezeigt wird, ist die Verarbeitung abgeschlossen.

   Die Datenbank SAMPLE wird beim Erstellen automatisch mit dem Aliasnamen SAMPLE katalogisiert.
3. Starten Sie den Datenbankmanager durch Eingabe des Befehls db2start.
4. Geben Sie in einem DB2-Befehlsfenster die nachstehenden DB2-Befehle ein, um eine Verbindung zur Datenbank SAMPLE herzustellen und eine Liste aller Mitarbeiter ('staff') in Abteilung ('dept') 20 abzurufen:

   ```
   db2 connect to sample
      db2 "select * from staff where dept = 20"
   ```
5. Um zu prüfen, ob diese Daten an alle Datenbankpartitionsserver verteilt wurden, geben Sie die folgenden Befehle über ein DB2-Befehlsfenster ein:

   ```
   db2 "select distinct dbpartitionnum(empno) from employee"
   ```

   In der Ausgabe dieses Befehls werden die von der Tabelle employee verwendeten Datenbankpartitionen aufgelistet. Die spezifische Ausgabe hängt von der Anzahl der Datenbankpartitionen in der Datenbank ab sowie von der Anzahl der Datenbankpartitionen in der Datenbankpartitionsgruppe, die von dem Tabellenbereich verwendet wird, in dem die Tabelle employee erstellt wurde.

Nachdem Sie die Installation geprüft haben, können Sie die Datenbank SAMPLE löschen, um Plattenspeicherplatz freizugeben. Es ist jedoch sinnvoll, die Beispieldatenbank beizubehalten, falls Sie beabsichtigen, Beispielanwendungen zu verwenden.

Geben Sie den Befehl db2 drop database sample ein, um die Datenbank SAMPLE zu löschen.

Prüfen der Installation eines partitionierten Datenbankservers (Linux und UNIX)

Um zu prüfen, ob Ihr DB2-Server erfolgreich installiert wurde, erstellen Sie eine Beispieldatenbank und führen SQL-Befehle aus, um Beispieldaten abzurufen und um zu prüfen, ob die Daten an alle zugehörigen Datenbankpartitionsserver der Installation verteilt wurden.

Vergewissern Sie sich vor dem Ausführen dieser Schritte, dass alle Installationsschritte vollständig ausgeführt wurden.

Um eine Beispieldatenbank (SAMPLE) zu erstellen, gehen Sie wie folgt vor:

1. Melden Sie sich am Primärcomputer (ServerA) als Instanzeigner an. Im vorliegenden Beispiel ist der Benutzer 'db2inst1' der Instanzeigner.
2. Geben Sie den Befehl db2sampl ein, um die Beispieldatenbank (SAMPLE) zu erstellen. Standardmäßig wird die Beispieldatenbank im Ausgangsverzeichnis des Instanzeigners erstellt. Im vorliegenden Beispiel ist /db2home/db2inst1/ das Ausgangsverzeichnis des Instanzeigners. Dieses Ausgangsverzeichnis ist der Standarddatenbankpfad.

   Die Verarbeitung dieses Befehls kann einige Minuten in Anspruch nehmen. Es gibt keine Abschlussnachricht. Wenn die Eingabeaufforderung wieder angezeigt wird, ist die Verarbeitung abgeschlossen.

   Die Datenbank SAMPLE wird beim Erstellen automatisch mit dem Aliasnamen SAMPLE katalogisiert.
3. Starten Sie den Datenbankmanager durch Eingabe des Befehls db2start.
4. Geben Sie in einem DB2-Befehlsfenster die nachstehenden DB2-Befehle ein, um eine Verbindung zur Datenbank SAMPLE herzustellen und eine Liste aller Mitarbeiter ('staff') in Abteilung ('dept') 20 abzurufen:

   ```
   db2 connect to sample
      db2 "select * from staff where dept = 20"
   ```
5. Um zu prüfen, ob diese Daten an alle Datenbankpartitionsserver verteilt wurden, geben Sie die folgenden Befehle über ein DB2-Befehlsfenster ein:

   ```
   db2 "select distinct dbpartitionnum(empno) from employee"
   ```

   In der Ausgabe dieses Befehls werden die von der Tabelle employee verwendeten Datenbankpartitionen aufgelistet. Die resultierende Ausgabe hängt von den folgenden Faktoren ab:

   - Anzahl der Partitionen in der Datenbank
   - Anzahl der Datenbankpartitionen in der Datenbankpartitionsgruppe, die vom Tabellenbereich verwendet wird, in dem die Tabelle employee erstellt wurde

Nachdem Sie die Installation geprüft haben, können Sie die Datenbank SAMPLE löschen, um Plattenspeicherplatz freizugeben. Geben Sie den Befehl db2 drop database sample ein, um die Datenbank SAMPLE zu löschen.

Teil 3. Implementierung und Pflege

Kapitel 9. Vorbereitung zum Erstellen einer Datenbank

Einrichten von Umgebungen mit partitionierten Datenbanken

Die Entscheidung, eine Mehrpartitionsdatenbank zu erstellen, muss vor der Erstellung der betreffenden Datenbank getroffen werden. Im Rahmen Ihrer Entscheidungen beim Datenbankentwurf müssen Sie festlegen, ob die Leistungsverbesserungen genutzt werden sollen, die eine Datenbankpartitionierung zu bieten hat.

Auch in einer Umgebung mit partitionierten Datenbanken verwenden Sie den Befehl CREATE DATABASE oder die Funktion 'sqlecrea()' zum Erstellen einer Datenbank. Unabhängig von der verwendeten Methode kann die Anforderung über eine beliebige der Partitionen erfolgen, die in der Datei `db2nodes.cfg` aufgelistet sind.

Vor der Erstellung einer Mehrpartitionsdatenbank müssen Sie die Datenbankpartition auswählen, die als Katalogknoten für die Datenbank fungieren soll. Anschließend können Sie die Datenbank direkt von dieser Datenbankpartition oder von einem fernen Client aus erstellen, der mit dieser Datenbankpartition verbunden ist. Die Datenbankpartition, zu der Sie die Verbindung (mit ATTACH) herstellen, um den Befehl CREATE DATABASE auszuführen, wird zur *Katalogpartition* für diese spezielle Datenbank.

Die Katalogpartition ist die Datenbankpartition, in der alle Systemkatalogtabellen gespeichert werden. Jeglicher Zugriff auf die Systemtabellen muss über diese Datenbankpartition erfolgen. Alle Objekte föderierter Datenbanken (z. B. Wrapper, Server und Kurznamen) werden in den Systemkatalogtabellen in dieser Datenbankpartition gespeichert.

Wenn möglich, sollten Sie jede Datenbank in einer getrennten Instanz erstellen. Falls dies nicht möglich ist (d. h., Sie müssen mehr als eine Datenbank pro Instanz erstellen), sollten Sie die Katalogpartitionen auf die verfügbaren Datenbankpartitionen verteilen. Dadurch verringern sich Konkurrenzsituationen beim Abrufen von Katalogdaten in einer einzelnen Datenbankpartition.

Anmerkung: Sie sollten regelmäßig ein Backup der Katalogpartition durchführen und nach Möglichkeit vermeiden, Benutzerdaten in ihr zu speichern, da diese Daten die für das Backup benötigte Zeit verlängern.

Wenn Sie eine Datenbank erstellen, wird sie automatisch in allen Datenbankpartitionen erstellt, die in der Datei `db2nodes.cfg` definiert sind.

Wenn die erste Datenbank im System erstellt wird, wird ein Systemdatenbankverzeichnis gebildet. An dieses werden Informationen zu anderen Datenbanken, die Sie erstellen, angehängt. Wenn Sie mit UNIX arbeiten, heißt das Systemdatenbankverzeichnis `sqldbdir` und befindet sich im Verzeichnis `sqllib` unter Ihrem Ausgangsverzeichnis bzw. unter dem Verzeichnis, in dem die DB2-Datenbank installiert wurde. Dieses Verzeichnis muss sich unter UNIX in einem gemeinsam genutzten Dateisystem (z. B. NFS auf UNIX-Plattformen) befinden, weil es nur ein Systemdatenbankverzeichnis für alle Datenbankpartitionen gibt, die zu einer Umgebung mit partitionierten Datenbanken gehören. Unter Windows befindet sich das Systemdatenbankverzeichnis im Instanzverzeichnis.

Ebenfalls im Verzeichnis sqldbdir befindet sich die Systemintentionsdatei. Sie hat den Namen sqldbins und stellt sicher, dass die Datenbankpartitionen synchronisiert bleiben. Diese Datei muss ebenfalls in einem gemeinsam genutzten Dateisystem gespeichert sein, da es innerhalb aller Datenbankpartitionen nur ein Verzeichnis gibt. Die Datei wird von allen Datenbankpartitionen, die die Datenbank bilden, gemeinsam genutzt.

Zur Nutzung der Datenbankpartitionierung müssen Konfigurationsparameter geändert werden. Mithilfe der Befehle GET DATABASE CONFIGURATION und GET DATABASE MANAGER CONFIGURATION können Sie die Werte für einzelne Einträge in einer bestimmten Datenbank oder in der Konfigurationsdatei des Datenbankmanagers ermitteln. Zur Änderung einzelner Einträge in einer bestimmten Datenbank oder in der Konfigurationsdatei des Datenbankmanagers werden die Befehle UPDATE DATABASE CONFIGURATION bzw. UPDATE DATABASE MANAGER CONFIGURATION verwendet.

Zu den Konfigurationsparametern des Datenbankmanagers, die sich auf eine Umgebung mit partitionierten Datenbanken auswirken, gehören **conn_elapse**, **fcm_num_buffers**, **fcm_num_channels**, **max_connretries**, **max_coordagents**, **max_time_diff**, **num_poolagents** und **stop_start_time**.

Erstellen von Knotenkonfigurationsdateien

Wenn Ihre Datenbank in einer partitionierten Datenbankumgebung arbeiten soll, müssen Sie eine Knotenkonfigurationsdatei mit dem Namen db2nodes.cfg erstellen.

Um eine Datenbankpartitionierung zu ermöglichen, muss sich die Datei db2nodes.cfg im Unterverzeichnis sqllib des Ausgangsverzeichnisses (Home) für die Instanz befinden, bevor Sie den Datenbankmanager starten. Diese Datei enthält Konfigurationsdaten für alle Datenbankpartitionen einer Instanz und wird von allen Datenbankpartitionen für diese Instanz gemeinsam genutzt.

Hinweise für Windows

Wenn Sie DB2 Enterprise Server Edition unter Windows verwenden, wird die Knotenkonfigurationsdatei beim Erstellen der Instanz erstellt. Sie sollten nicht versuchen, eine Knotenkonfigurationsdatei manuell zu erstellen oder manuell zu ändern. Mithilfe des Befehls db2ncrt können Sie einer Instanz einen Datenbankpartitionsserver hinzufügen. Mithilfe des Befehls db2ndrop können Sie einen Datenbankpartitionsserver aus einer Instanz löschen. Mithilfe des Befehls db2nchg können Sie die Konfiguration der Datenbankpartitionsserver ändern. Dies umfasst das Versetzen des Datenbankpartitionsservers von einem Computer auf einen anderen, das Ändern des TCP/IP-Hostnamens oder das Auswählen eines anderen logischen Portnamens oder Netzwerknamens.

Anmerkung: Erstellen Sie keine anderen als die vom Datenbankmanager erstellten Dateien oder Verzeichnisse unter dem Unterverzeichnis sqllib, um Datenverluste zu vermeiden, wenn eine Instanz gelöscht wird. Es gibt jedoch zwei Ausnahmen. Wenn Ihr System gespeicherte Prozeduren unterstützt, stellen Sie die gespeicherten Prozeduranwendungen in das Unterverzeichnis 'function' im Unterverzeichnis sqllib. Die andere Ausnahme betrifft eventuell erstellte benutzerdefinierte Funktionen (UDFs). Benutzerdefinierte Funktionen können im selben Verzeichnis gespeichert werden.

Die Datei enthält eine Zeile für jede Datenbankpartition, die zu einer Instanz gehört. Jede Zeile hat folgendes Format:

dbpartitionnum hostname [*logischer-port* [*netzname*]]

Die Token einer Zeile sind durch Leerzeichen voneinander getrennt. Die Variablen sind:

dbpartitionsnummer
Die Datenbankpartitionsnummer (mögliche Werte: 0 - 999) definiert eine Datenbankpartition eindeutig. Datenbankpartitionsnummern müssen in aufsteigender Reihenfolge angegeben sein. Es dürfen Sprünge in der Folge der Nummern auftreten.

Wenn eine Datenbankpartitionsnummer einmal zugeordnet ist, kann sie nicht mehr geändert werden. (Ansonsten könnten die Informationen in der Verteilungszuordnung, die bestimmt, wie Daten verteilt werden, inkonsistent werden.)

Wenn Sie eine Datenbankpartition löschen, kann ihre Datenbankpartitionsnummer für jede neue Datenbankpartition, die Sie hinzufügen, wieder verwendet werden.

Die Datenbankpartitionsnummer wird zur Generierung eines Datenbankpartitionsnamens im Datenbankverzeichnis verwendet. Er hat folgendes Format:

```
NODE nnnn
```

Die Ziffernfolge *nnnn* ist die Datenbankpartitionsnummer, die links mit Nullen aufgefüllt wird. Diese Datenbankpartitionsnummer wird außerdem durch die Befehle CREATE DATABASE und DROP DATABASE verwendet.

hostname
Der Hostname der IP-Adresse für die partitionsübergreifende Kommunikation. Verwenden Sie den vollständig qualifizierten Namen für den Hostnamen. In der Datei `/etc/hosts` sollte ebenfalls der vollständig qualifizierte Name verwendet werden. Wenn der vollständig qualifizierte Name in der Datei `db2nodes.cfg` und in der Datei `/etc/hosts` nicht verwendet wird, empfangen Sie eventuell die Fehlernachricht `SQL30082N RC=3`.

(Es gibt eine Ausnahme, wenn der Netzname angegeben wird. In diesem Fall wird der Netzname für den Großteil der Kommunikation verwendet, während Hostname nur für die Befehle db2start, db2stop und db2_all verwendet wird.)

logischer-port
Dieser Parameter ist optional und gibt die logische Portnummer für die Datenbankpartition an. Diese Nummer wird mit dem Instanznamen des Datenbankmanagers verwendet, um einen TCP/IP-Servicenamenseintrag in der Datei `etc/services` anzugeben.

Die Kombination aus IP-Adresse und logischem Port wird als allgemein bekannte Adresse verwendet und muss für alle Anwendungen eindeutig sein, um die Verbindungen zur Kommunikation zwischen Datenbankpartitionen zu unterstützen.

Für jeden Hostnamen muss ein *logischer-port* entweder 0 (null) oder leer sein (was standardmäßig dem Wert 0 entspricht). Die Datenbankpartition, der dieser *logische-port* zugeordnet ist, ist der Standardknoten auf dem Host, zum dem Clients die Verbindung herstellen. Diese Einstellung kann

mithilfe der Umgebungsvariablen **DB2NODE** im Script db2profile oder mit der API sqlesetc() überschrieben werden.

netzname
Dieser Parameter ist optional und wird zur Unterstützung eines Hosts verwendet, der über mehr als eine aktive TCP/IP-Schnittstelle verfügt, von denen jede ihren eigenen Hostnamen hat.

Das folgende Beispiel zeigt eine mögliche Knotenkonfigurationsdatei für ein System RS/6000 SP, in dem SP2EN1 mehrere TCP/IP-Schnittstellen und zwei logische Datenbankpartitionen hat und SP2SW1 als DB2-Datenbankschnittstelle verwendet. Es zeigt außerdem die Datenbankpartitionsnummern, beginnend bei 1 (und nicht bei 0), sowie einen Sprung in der Folge der *dbpartitionsnummern*:

Tabelle 12. Eine Tabelle mit einem Beispiel für Datenbankpartitionsnummern

dbpartitionsnummer	*hostname*	*logischer-port*	*netzname*
1	SP2EN1.mach1.xxx.com	0	SP2SW1
2	SP2EN1.mach1.xxx.com	1	SP2SW1
4	SP2EN2.mach1.xxx.com	0	
5	SP2EN3.mach1.xxx.com		

Die Datei `db2nodes.cfg` kann mit einem beliebigen Editor aktualisiert werden. (Ausnahme: Unter Windows sollte kein Editor verwendet werden.) Sie müssen jedoch sorgfältig auf den Schutz der Integrität der Daten in der Datei achten, da die Datenbankpartitionierung erfordert, dass die Knotenkonfigurationsdatei gesperrt wird, wenn der Befehl db2start ausgeführt wird, und entsperrt wird, wenn der Befehl db2stop den Datenbankmanager beendet hat. Der Befehl db2start kann die Datei bei Bedarf aktualisieren, während sie gesperrt ist. Sie können beispielsweise den Befehl db2start mit der Option **RESTART** oder **ADD DBPARTITIONNUM** ausführen.

Anmerkung: Wenn der Befehl db2stop nicht erfolgreich ausgeführt und die Knotenkonfigurationsdatei nicht entsperrt wird, führen Sie den Befehl db2stop **FORCE** aus, um sie zu entsperren.

Format der DB2-Knotenkonfigurationsdatei

Mithilfe der Datei `db2nodes.cfg` werden die Datenbankpartitionsserver definiert, die einer DB2-Instanz angehören. Darüber hinaus dient die Datei `db2nodes.cfg` zur Angabe der IP-Adresse oder des Hostnamens einer Hochgeschwindigkeitsverbindung, wenn Sie eine Hochgeschwindigkeitsverbindung zur Kommunikation für die Datenbankpartitionsserver verwenden möchten.

Für eine Umgebung mit einer Einzeldatenbankpartition muss die Datei `db2nodes.cfg` definiert und mit dem Namen des Hosts eingerichtet werden, auf dem sich die Instanz befindet. In einer Failover-Konfiguration mit Bereitschaftsmodus (Hot Standby) unterscheiden sich der Hostname des sekundären Servers (des Übernahmeservers) von dem des primären Servers. Bei einem Fehlerereignis, das eine Funktionsübernahme (Failover) durch den sekundären Server auslöst, kann die Instanz nicht gestartet werden, wenn sie noch immer den Hostnamen des primären Servers enthält.

Die Datei `db2nodes.cfg` unter Linux- und UNIX-Betriebssystemen hat das folgende Format:

```
nodenumber hostname logicalport netname resourcesetname
```

nodenumber, *hostname*, *logicalport*, *netname* und *resourcesetname* werden im folgenden Abschnitt definiert.

Die Datei `db2nodes.cfg` unter Windows-Betriebssystemen hat das folgende Format:

```
nodenumber hostname computername logicalport netname resourcesetname
```

Unter Windows-Betriebssystemen werden diese Einträge mit dem Befehl db2ncrt oder db2 add db partition der Datei `db2nodes.cfg` hinzugefügt. Sie sollten diese Zeilen weder direkt hinzufügen noch diese Datei bearbeiten.

nodenumber
: Eine eindeutige Nummer zwischen 0 und 999, die einen Datenbankpartitionsserver in einem partitionierten Datenbanksystem identifiziert.

  Wenn Sie das partitionierte Datenbanksystem skalieren möchten, fügen Sie der Datei `db2nodes.cfg` für jeden Datenbankpartitionsserver jeweils einen Eintrag hinzu. Die Werte für *nodenumber*, die Sie für weitere Datenbankpartitionsserver auswählen, müssen aufsteigend, aber nicht direkt aufeinanderfolgend sein. Sie können zwischen den Werten von *nodenumber* beispielsweise eine Lücke lassen, wenn Sie später logische Partitionsserver hinzufügen und für die Knoten eine logische Gruppierung in dieser Datei beibehalten möchten.

  Dieser Eintrag ist erforderlich.

hostname
: Der TCP/IP-Hostname des Datenbankpartitionsservers zur Verwendung durch FCM (Fast Communications Manager). Dieser Eintrag ist erforderlich.

  Wenn in der Datei `db2nodes.cfg` anstelle von IP-Adressen Hostnamen angegeben werden, versucht der Datenbankmanager, die Hostnamen dynamisch aufzulösen. Die Auflösung kann entweder lokal oder über einen Suchvorgang bei registrierten Domänennamensservern (DNS) erfolgen, die in den Betriebssystemeinstellungen des betreffenden Systems festgelegt sind.

  Wenn auf dem System mehr als eine Netzschnittstellenkarte installiert ist und der Hostname in der Datei `db2nodes.cfg` nicht in den Standardhost des Systems aufgelöst werden kann, kann es sein, dass er als ferner Host behandelt wird. Diese Einrichtung bringt die Einschränkung mit sich, dass die Datenbankmigration nicht erfolgreich ausgeführt werden kann, weil das lokale Datenbankverzeichnis nicht gefunden werden kann, falls die Instanz nicht gestartet ist. Daher muss für HADR, HACMP und andere Hochverfügbarkeitsumgebungen möglicherweise der Eintrag *hostname* mit dem Namen übereinstimmen, der vom Betriebssystem für die Angabe des Hosts verwendet wird, damit die Migration durchgeführt werden kann.

  Ab DB2 Version 9.1 werden sowohl das Protokoll TCP/IPv4 als auch das Protokoll TCP/IPv6 unterstützt. Die Methode zum Auflösen von Hostnamen wurde geändert.

  Während in Releases vor Version 9.1 die Zeichenfolge entsprechend der Definition in Datei `db2nodes.cfg` aufgelöst wird, wird ab Version 9.1 versucht, die vollständig qualifizierten Domänennamen aufzulösen, wenn in der Datei `db2nodes.cfg` Kurznamen enthalten sind. Werden anstelle der vollständig qualifizierten Hostnamen die in der Konfigurationsdatei definierten Kurznamen verwendet, kann dies zu unnötigen Verzögerungen bei Prozessen führen, bei denen Hostnamen aufgelöst werden.

Verwenden Sie eine der folgenden Ausweichlösungen, um Verzögerungen bei DB2-Befehlen zu vermeiden, bei denen die Auflösung von Hostnamen erforderlich ist:

1. Wenn in der Datei db2nodes.cfg und in der Datei 'hosts' des Betriebssystems Kurznamen angegeben sind, geben Sie den Kurznamen und den vollständig qualifizierten Domänennamen für den Hostnamen in der Datei 'hosts' des Betriebssystems an.
2. Setzen Sie den folgenden Befehl ab, um lediglich IPv4-Adressen zu verwenden, wenn Sie wissen, dass der DB2-Server an einem IPv4-Port empfangsbereit ist:

   ```
   db2 catalog tcpip4 node db2tcp2 remote 192.0.32.67 server db2inst1
      with "IPv4-Adresssuche - 192.0.32.67"
   ```

3. Wenn Sie nur IPv6-Adressen verwenden wollen, weil Sie wissen, dass der DB2-Server an einem IPv6-Port empfangsbereit ist, geben Sie folgenden Befehl ein:

   ```
   db2 catalog tcpip6 node db2tcp3 1080:0:0:0:8:800:200C:417A server 50000
      with "Look up IPv6 address from 1080:0:0:0:8:800:200C:417A"
   ```

logicalport
Gibt die logische Portnummer für den Datenbankpartitionsserver an. Mit diesem Feld wird ein bestimmter Datenbankpartitionsserver auf einer Workstation angegeben, die logische Datenbankpartitionsserver ausführt.

DB2 reserviert während der Installation in der Datei /etc/services einen Portbereich (z. B. 60000 - 60003) für die partitionsübergreifende Kommunikation. Mit diesem Feld (*logicalport*) in der Datei db2nodes.cfg wird angegeben, welchen Port in diesem Bereich Sie einem bestimmten logischen Partitionsserver zuweisen wollen.

Wenn dieses Feld keinen Eintrag enthält, ist die Standardeinstellung 0. Wenn Sie jedoch einen Eintrag für das Feld *netname* hinzufügen, müssen Sie eine Nummer für das Feld *logicalport* angeben.

Wenn Sie logische Datenbankpartitionen verwenden, muss der Wert von *logicalport zwingend* bei 0 beginnen und in in aufsteigender Reihenfolge fortgesetzt werden (z. B. 0,1,2).

Außerdem müssen Sie bei Angabe des Eintrags *logicalport* für einzigen einen Datenbankpartitionsserver in der Datei db2nodes.cfg für jeden Datenbankpartitionsserver einen Wert für *logicalport* angeben.

Dieses Feld ist nur dann optional, wenn Sie *weder* logische Datenbankpartitionen *noch* eine Hochgeschwindigkeitsverbindung nutzen.

netname
Gibt den Hostnamen oder die IP-Adresse der Hochgeschwindigkeitsverbindung für die FCM-Kommunikation an.

Wenn für dieses Feld ein Eintrag angegeben wird, wird die gesamte Kommunikation zwischen Datenbankpartitionsservern (mit Ausnahme der Kommunikation, die aus den Befehlen db2start, db2stop und db2_all resultiert) über die Hochgeschwindigkeitsverbindung geleitet.

Dieser Parameter ist nur dann erforderlich, wenn Sie für die Datenbankpartitionskommunikation eine Hochgeschwindigkeitsverbindung verwenden.

resourcesetname
Mit dem Eintrag *resourcesetname* wird die Betriebssystemressource definiert, in der der Knoten gestartet werden soll. *resourcesetname* dient zur Unter-

stützung der Prozessaffinität, die für mehrere logische Knoten (MLNs, Multiple Logical Nodes) verwendet wird. Diese Unterstützung wird mit einem Feld vom Typ 'string' angegeben, das bisher als 'quadname' bezeichnet wurde.

Dieser Parameter wird nur unter den Betriebssystemen AIX, HP-UX und Solaris unterstützt.

Unter AIX wird dieses Konzept als "Ressourcengruppen" und unter dem Betriebssystem Solaris als "Projekte" bezeichnet. Weitere Informationen zum Ressourcenmanagement finden Sie in Ihrer Betriebssystemdokumentation.

Unter HP-UX ist der Parameter *resourcesetname* der Name einer PRM-Gruppe. Weitere Informationen finden Sie in der Dokumentation 'HP-UX Process Resource Manager. User Guide. (B8733-90007)' von HP.

Unter dem Betriebssystem Windows kann die Prozessaffinität für einen logischen Knoten über die Registrierdatenbankvariable **DB2PROCESSORS** definiert werden.

Unter dem Betriebssystem Linux wird mit der Spalte *resourcesetname* eine Zahl definiert, die einem NUMA-Knoten (NUMA - Non-Uniform Memory Access, nicht gleichmäßiger Speicherzugriff) auf dem System entspricht. Dazu müssen das Systemdienstprogramm numactl sowie ein 2.6-Kernel mit NUMA-Richtlinienunterstützung verfügbar sein.

Der Parameter *netname* muss angegeben werden, wenn der Parameter *resourcesetname* verwendet wird.

Beispielkonfigurationen

Anhand der folgenden Beispielkonfigurationen können Sie die geeignete Konfiguration für Ihre Umgebung ermitteln.

Ein Computer, vier Datenbankpartitionsserver

Wenn Sie keine Clusterumgebung verwenden und vier Datenbankpartitionsserver auf einer einzigen physischen Workstation mit dem Namen `ServerA` nutzen wollen, müssen Sie die Datei `db2nodes.cfg` wie folgt aktualisieren:

```
0          ServerA        0
1          ServerA        1
2          ServerA        2
3          ServerA        3
```

Zwei Computer, ein Datenbankpartitionsserver pro Computer

Wenn Ihr partitioniertes Datenbanksystem zwei physische Workstations mit den Namen `ServerA` und `ServerB` enthalten soll, müssen Sie die Datei `db2nodes.cfg` wie folgt aktualisieren:

```
0          ServerA        0
1          ServerB        0
```

Zwei Computer, drei Datenbankpartitionsserver auf einem Computer

Wenn Ihr partitioniertes Datenbanksystem zwei physische Workstations mit den Namen `ServerA` und `ServerB` enthalten soll und auf `ServerA` drei Datenbankpartitionsserver laufen, müssen Sie die Datei `db2nodes.cfg` wie folgt aktualisieren:

```
4          ServerA        0
6          ServerA        1
8          ServerA        2
9          ServerB        0
```

Zwei Computer, drei Datenbankpartitionsserver mit Hochgeschwindigkeitsswitches Wenn Ihr partitioniertes Datenbanksystem zwei Computer mit den Namen ServerA und ServerB enthalten soll (auf ServerB laufen dabei zwei Datenbankpartitionsserver) und eine Hochgeschwindigkeitsverbindung mit den Namen switch1 und switch2 verwendet wird, müssen Sie die Datei db2nodes.cfg wie folgt aktualisieren:

```
0          ServerA        0              switch1
1          ServerB        0              switch2
2          ServerB        1              switch2
```

Beispiele mit 'resourcesetname'

Für die folgenden Beispiele gelten folgende Einschränkungen:

- Dieses Beispiel veranschaulicht die Verwendung von *resourcesetname* ohne Hochgeschwindigkeitsverbindung in der Konfiguration.
- Der Eintrag *netname* entspricht der vierten Spalte, in der auch der Eintrag *hostname* angegeben werden kann, wenn kein Switchname vorhanden ist und Sie den Eintrag *resourcesetname* verwenden wollen. Der fünfte Parameter ist *resourcesetname* (falls definiert). Die Ressourcengruppenspezifikation kann lediglich als fünfte Spalte der Datei db2nodes.cfg angezeigt werden. Dies bedeutet, dass Sie auch eine vierte Spalte eingeben müssen, damit Sie eine Ressourcengruppe angeben können. Die vierte Spalte ist für einen Hochgeschwindigkeitsswitch vorgesehen.
- Wenn Sie keinen Hochgeschwindigkeitsswitch besitzen oder diesen nicht verwenden wollen, müssen Sie den Eintrag *hostname* (wie in der zweiten Spalte) angeben. Mit anderen Worten: Das DB2-Datenbankmanagementsystem unterstützt in der Datei db2nodes.cfg keine Spaltenlücken (bzw. den Austausch von Spalten). Diese Einschränkung galt in der Vergangenheit bereits für die ersten drei Spalten und gilt nun für alle fünf Spalten.

Beispiel für AIX

Nachfolgend finden Sie ein Beispiel für die Einrichtung der Ressourcengruppe für das Betriebssystem AIX.

In diesem Beispiel gibt es einen (1) physischen Knoten mit 32 Prozessoren und 8 logischen Datenbankpartitionen (MLNs). Dieses Beispiel veranschaulicht die Bereitstellung von Prozessaffinität für die einzelnen MLNs.

1. Definieren Sie Ressourcengruppen in der Datei /etc/rset:

```
DB2/MLN1:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00000,sys/cpu.00001,sys/cpu.00002,sys/cpu.00003

DB2/MLN2:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00004,sys/cpu.00005,sys/cpu.00006,sys/cpu.00007

DB2/MLN3:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00008,sys/cpu.00009,sys/cpu.00010,sys/cpu.00011

DB2/MLN4:
    owner     = db2inst1
```

```
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00012,sys/cpu.00013,sys/cpu.00014,sys/cpu.00015

DB2/MLN5:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00016,sys/cpu.00017,sys/cpu.00018,sys/cpu.00019

DB2/MLN6:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00020,sys/cpu.00021,sys/cpu.00022,sys/cpu.00023

DB2/MLN7:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00024,sys/cpu.00025,sys/cpu.00026,sys/cpu.00027

DB2/MLN8:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00028,sys/cpu.00029,sys/cpu.00030,sys/cpu.00031
```

2. Aktivieren Sie die Speicheraffinität, indem Sie den folgenden Befehl eingeben:

   ```
   vmo -p -o memory_affinity=1
   ```

3. Erteilen Sie Instanzberechtigungen zur Verwendung von Ressourcengruppen:

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

4. Fügen Sie den Namen der Ressourcengruppe als fünfte Spalte der Datei `db2nodes.cfg` hinzu:

   ```
   1 regatta 0 regatta DB2/MLN1
   2 regatta 1 regatta DB2/MLN2
   3 regatta 2 regatta DB2/MLN3
   4 regatta 3 regatta DB2/MLN4
   5 regatta 4 regatta DB2/MLN5
   6 regatta 5 regatta DB2/MLN6
   7 regatta 6 regatta DB2/MLN7
   8 regatta 7 regatta DB2/MLN8
   ```

Beispiel für HP-UX

Dieses Beispiel veranschaulicht die Verwendung von PRM-Gruppen für CPU-Anteile auf einem System mit 4 CPUs und 4 MLNs und 24 % CPU-Anteil pro MLN, wodurch 4 % für andere Anwendungen verbleiben. Der Name der DB2-Instanz ist db2inst1.

1. Bearbeiten Sie den Abschnitt GROUP der Datei `/etc/prmconf`:

   ```
   OTHERS:1:4::
   db2prm1:50:24::
   db2prm2:51:24::
   db2prm3:52:24::
   db2prm4:53:24::
   ```

2. Fügen Sie der Datei `/etc/prmconf` einen Eintrag für den Instanzeigner hinzu:

   ```
   db2inst1::::OTHERS,db2prm1,db2prm2,db2prm3,db2prm4
   ```

3. Initialisieren Sie die Gruppen und aktivieren Sie den CPU-Manager durch Eingabe des folgenden Befehls:

```
prmconfig -i
prmconfig -e CPU
```

4. Fügen Sie der Datei db2nodes.cfg als fünfte Spalte PRM-Gruppennamen hinzu:

```
1 voyager 0 voyager db2prm1
2 voyager 1 voyager db2prm2
3 voyager 2 voyager db2prm3
4 voyager 3 voyager db2prm4
```

Die PRM-Konfiguration (Schritte 1-3) kann mit dem interaktiven GUI-Tool xprm ausgeführt werden.

Beispiel für Linux

Unter dem Betriebssystem Linux wird mit der Spalte *resourcesetname* eine Zahl definiert, die einem NUMA-Knoten (NUMA - Non-Uniform Memory Access, nicht gleichmäßiger Speicherzugriff) auf dem System entspricht. Dazu müssen das Systemdienstprogramm numactl sowie ein 2.6-Kernel mit NUMA-Richtlinienunterstützung verfügbar sein. Weitere Informationen zur NUMA-Unterstützung unter dem Betriebssystem Linux finden Sie auf der Man-Page für numact1.

Dieses Beispiel veranschaulicht die Einrichtung eines NUMA-Computers mit vier Knoten, bei dem jeder logische Knoten einem NUMA-Knoten zugeordnet ist.

1. Stellen Sie sicher, dass NUMA-Funktionen auf dem System vorhanden sind.
2. Setzen Sie den folgenden Befehl ab:

```
$ numactl --hardware
```

   Die Ausgabe sieht etwa wie folgt aus:

```
available: 4 nodes (0-3)
node 0 size: 1901 MB
node 0 free: 1457 MB
node 1 size: 1910 MB
node 1 free: 1841 MB
node 2 size: 1910 MB
node 2 free: 1851 MB
node 3 size: 1905 MB
node 3 free: 1796 MB
```

3. In diesem Beispiel gibt es vier NUMA-Knoten im System. Bearbeiten Sie die Datei db2nodes.cfg wie folgt, um jeden MLN einem NUMA-Knoten im System zuzuordnen:

```
0 hostname 0 hostname 0
1 hostname 1 hostname 1
2 hostname 2 hostname 2
3 hostname 3 hostname 3
```

Beispiel für Solaris

Das folgende Beispiel zeigt, wie das Projekt für Solaris Version 9 eingerichtet wird.

In diesem Beispiel gibt es einen (1) physischen Knoten mit acht (8) Prozessoren: Eine CPU wird für das Standardprojekt verwendet, drei (3) CPUs werden vom Anwendungsserver verwendet und vier (4) CPUs werden für DB2 verwendet. Der Name der Instanz ist db2inst1.

1. Erstellen Sie mithilfe eines Editors eine Konfigurationsdatei für den Ressourcenpool. In diesem Beispiel hat die Datei den Namen pool.db2. Sie hat folgenden Inhalt:

```
create system hostname
create pset pset_default (uint pset.min = 1)
create pset db0_pset (uint pset.min = 1; uint pset.max = 1)
create pset db1_pset (uint pset.min = 1; uint pset.max = 1)
create pset db2_pset (uint pset.min = 1; uint pset.max = 1)
create pset db3_pset (uint pset.min = 1; uint pset.max = 1)
create pset appsrv_pset (uint pset.min = 3; uint pset.max = 3)
create pool pool_default (string pool.scheduler="TS";
     boolean pool.default = true)
create pool db0_pool (string pool.scheduler="TS")
create pool db1_pool (string pool.scheduler="TS")
create pool db2_pool (string pool.scheduler="TS")
create pool db3_pool (string pool.scheduler="TS")
create pool appsrv_pool (string pool.scheduler="TS")
associate pool pool_default (pset pset_default)
associate pool db0_pool (pset db0_pset)
associate pool db1_pool (pset db1_pset)
associate pool db2_pool (pset db2_pset)
associate pool db3_pool (pset db3_pset)
associate pool appsrv_pool (pset appsrv_pset)
```

2. Bearbeiten Sie die Datei `/etc/project` wie folgt, um die DB2-Projekte und das Projekt 'appsrv' hinzuzufügen:

```
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
appsrv:4000:App Serv project:root::project.pool=appsrv_pool
db2proj0:5000:DB2 Node 0 project:db2inst1,root::project.pool=db0_pool
db2proj1:5001:DB2 Node 1 project:db2inst1,root::project.pool=db1_pool
db2proj2:5002:DB2 Node 2 project:db2inst1,root::project.pool=db2_pool
db2proj3:5003:DB2 Node 3 project:db2inst1,root::project.pool=db3_pool
```

3. Erstellen Sie den Ressourcenpool: `# poolcfg -f pool.db2`
4. Aktivieren Sie den Ressourcenpool: `# pooladm -c`
5. Fügen Sie der Datei `db2nodes.cfg` den Projektnamen als fünfte Spalte hinzu:

```
0 hostname 0 hostname db2proj0
1 hostname 1 hostname db2proj1
2 hostname 2 hostname db2proj2
3 hostname 3 hostname db2proj3
```

Angeben der Liste von Systemen in einer Umgebung mit partitionierten Datenbanken

Standardmäßig wird die Liste der Computer der Knotenkonfigurationsdatei `db2nodes.cfg` entnommen.

Dies kann auf folgende Weise geändert werden:

- Angeben eines Pfadnamens zu der Datei, die die Liste der Maschinen enthält, indem die Umgebungsvariable RAHOSTFILE exportiert (auf Linux- und UNIX-Plattformen) oder definiert wird (unter Windows).
- Explizites Angeben der Liste als Zeichenfolge durch Leerzeichen getrennter Namen, indem die Umgebungsvariable RAHOSTLIST exportiert (auf Linux- und UNIX-Plattformen) oder definiert wird (unter Windows).

  Anmerkung: Wenn beide dieser Umgebungsvariablen angegeben werden, erhält RAHOSTLIST den Vorrang.

Anmerkung: Unter Windows sollten Sie die Knotenkonfigurationsdatei auf *keinen* Fall manuell editieren, um keine Inkonsistenzen zu verursachen. Zum Abrufen der Liste der Computer in der Instanz verwenden Sie den Befehl db2nlist.

Eliminieren doppelter Einträge aus einer Liste von Systemen in einer Umgebung mit partitionierten Datenbanken

Wenn Sie mit DB2 Enterprise Server Edition und mehreren logischen Datenbankpartitionsservern auf einem Computer arbeiten, enthält Ihre Datei db2nodes.cfg mehrere Einträge für diesen Computer.

In diesem Fall muss dem Befehl rah mitgeteilt werden, ob der Befehl nur einmal auf jedem Computer oder einmal für jede in der Datei db2nodes.cfg aufgeführte logische Datenbankpartition ausgeführt werden soll. Verwenden Sie den Befehl rah, wenn Sie Computer angeben möchten. Verwenden Sie den Befehl db2_all, wenn Sie logische Datenbankpartitionen angeben möchten.

Anmerkung: Wenn Sie auf Linux- und UNIX-Plattformen Computer angeben, entfernt rah normalerweise doppelte Einträge aus der Computerliste, jedoch mit folgender Ausnahme: Wenn Sie logische Datenbankpartitionen angeben, setzt der Befehl db2_all Ihrem Befehl die folgende Zuweisung voran:

```
export DB2NODE=nnn  (für Korn-Shellsyntax)
```

Dabei ist 'nnn' die Datenbankpartitionsnummer, die der entsprechenden Zeile in der Datei 'db2nodes.cfg' entnommen ist, sodass der Befehl an den gewünschten Datenbankpartitionsserver weitergeleitet wird.

Bei der Angabe logischer Datenbankpartitionen können Sie die Liste mithilfe der Präfixsequenzen <<-nnn< und <<+nnn< auf alle logischen Datenbankpartitionen außer einer beschränken bzw. nur eine angeben. Diese Möglichkeit kann nützlich sein, wenn Sie einen Befehl ausführen möchten, um zuerst die Datenbankpartition zu katalogisieren, und nach Abschluss dieses Befehls denselben Befehl in allen anderen Datenbankpartitionsservern eventuell parallel ausführen möchten. Normalerweise ist diese Methode für die Ausführung des Befehls db2 restart database erforderlich. Für diesen Fall müssen Sie die Datenbankpartitionsnummer der Katalogpartition kennen.

Wenn Sie den Befehl db2 restart database mithilfe des Befehls rah ausführen, werden doppelte Einträge aus der Liste der Computer eliminiert. Wenn Sie jedoch das Präfix " angeben, werden doppelte Einträge nicht eliminiert, da angenommen wird, dass die Verwendung des Präfixes " das Senden der Befehle an alle Datenbankpartitionsserver und nicht an alle Computer impliziert.

Aktualisieren der Knotenkonfigurationsdatei (Linux und UNIX)

Die vorliegende Task beschreibt die erforderlichen Schritte zum Aktualisieren der Datei db2nodes.cfg, um Einträge für die zugehörigen Computer hinzuzufügen.

Die Knotenkonfigurationsdatei (db2nodes.cfg) im Ausgangsverzeichnis des Instanzeigners enthält Konfigurationsdaten, mit deren Hilfe DB2 ermittelt, welche Server einer Instanz der Umgebung mit partitionierten Datenbanken angehören. Für jede Instanz in einer Umgebung mit partitionierten Datenbanken ist eine Datei db2nodes.cfg vorhanden.

Die Datei db2nodes.cfg muss für jeden Server, der der Instanz angehört, jeweils einen Eintrag enthalten. Bei der Erstellung einer Instanz wird die Datei db2nodes.cfg automatisch erstellt, und für den Server, der Instanzeigner ist, wird ein Eintrag hinzugefügt.

Wenn Sie beispielsweise die DB2-Instanz mithilfe des DB2-Installationsassistenten auf dem Server ServerA (Instanzeigner) erstellt haben, wird die Datei db2nodes.cfg wie folgt aktualisiert:

```
0     ServerA     0
```

Voraussetzungen

- Die DB2-Anwendung muss auf allen zugehörigen Computern installiert sein.
- Auf dem Primärcomputer muss eine DB2-Instanz vorhanden sein.
- Sie müssen als Benutzer über die Berechtigung SYSADM verfügen.
- Prüfen Sie die Konfigurationsbeispiele und Informationen zum Dateiformat im Abschnitt über das Format der DB2-Knotenkonfigurationsdatei, wenn eine der folgenden Bedingungen zutrifft:
  - Sie beabsichtigen, einen Hochgeschwindigkeitsswitch für die Kommunikation zwischen Datenbankpartitionsservern zu verwenden
  - Ihre partitionierte Konfiguration soll mehrere logische Partitionen enthalten

Einschränkung

Bei den Hostnamen, die in den Schritten im Abschnitt 'Vorgehensweise' verwendet werden, muss es sich um vollständig qualifizierte Hostnamen handeln.

Um die Datei db2nodes.cfg zu aktualisieren, gehen Sie wie folgt vor:

1. Melden Sie sich als Instanzeigner an. Im vorliegenden Beispiel ist 'db2inst1' der Instanzeigner.
2. Stoppen Sie die DB2-Instanz, indem Sie folgenden Befehl eingeben:

   INSTANZAUSGANGSVERZEICHNIS/sqllib/adm/db2stop

   Hierbei ist *INSTANZAUSGANGSVERZEICHNIS* das Ausgangsverzeichnis des Instanzeigners. (Ist die Instanz aktiv, wird die Datei db2nodes.cfg gesperrt. Sie kann erst nach dem Stoppen der Instanz editiert werden.)

   Lautet das Ausgangsverzeichnis der Instanz beispielsweise /db2home/db2inst1, geben Sie folgenden Befehl ein:

   ```
   /db2home/db2inst1/sqllib/adm/db2stop
   ```

3. Fügen Sie der Datei .rhosts für jeder DB2-Instanz einen Eintrag hinzu. Aktualisieren Sie die Datei, indem Sie Folgendes hinzufügen:

   ```
   <hostname> <db2-instanz>
   ```

   Dabei ist <hostname> der TCP/IP-Hostname des Datenbankservers und <db2-instanz> der Name der Instanz, die Sie für den Zugriff auf den Datenbankserver verwenden.
4. Fügen Sie der Datei db2nodes.cfg jedes zugehörigen Servers einen Eintrag hinzu. Wenn Sie die Datei db2nodes.cfg zum ersten Mal anzeigen, sollte sie einen ähnlichen Eintrag wie den folgenden enthalten:

   ```
   0     ServerA     0
   ```

   Dieser Eintrag enthält die Nummer des Datenbankpartitionsservers (Knotennummer), den TCP/IP-Hostnamen des Servers, auf dem sich der Datenbankpartitionsserver befindet, und die logische Portnummer des Datenbankpartitionsservers.

   Wenn Sie beispielsweise eine partitionierte Konfiguration mit vier Computern und einem Datenbankpartitionsserver auf jedem Computer installieren, sollte die aktualisierte Datei db2nodes.cfg etwa wie folgt aussehen:

```
0     ServerA     0
1     ServerB     0
2     ServerC     0
3     ServerD     0
```

5. Geben Sie nach abgeschlossener Aktualisierung der Datei db2nodes.cfg den Befehl INSTANZAUSGANGSVERZEICHNIS/sqllib/adm/db2start ein, wobei *INSTANZAUSGANGSVERZEICHNIS* das Ausgangsverzeichnis des Instanzeigners ist. Lautet das Ausgangsverzeichnis der Instanz beispielsweise /db2home/db2inst1, geben Sie folgenden Befehl ein:

   ```
   /db2home/db2inst1/sqllib/adm/db2start
   ```

6. Melden Sie sich ab.

Einrichten mehrerer logischer Knoten

Normalerweise wird DB2 Enterprise Server Edition so konfiguriert, dass jedem Computer ein Datenbankpartitionsserver zugeordnet ist. Es gibt jedoch verschiedene Fälle, in denen es günstiger ist, wenn auf einem Computer mehrere Datenbankpartitionsserver ausgeführt werden.

Dies bedeutet, dass die Konfiguration eine höhere Anzahl von Datenbankpartitionen als Computer umfassen kann. In diesen Fällen werden auf dem betreffenden Computer *mehrere logische Partitionen* oder *mehrere logische Knoten* ausgeführt, wenn diese *dieselbe* Instanz verwenden. Andernfalls werden auf dem Computer *nicht* mehrere logische Partitionen ausgeführt.

Durch die Unterstützung mehrerer logischer Partitionen können drei verschiedene Konfigurationsarten ausgewählt werden:

- Eine Standardkonfiguration, bei der jeder Computer über nur einen Datenbankpartitionsserver verfügt.
- Eine Konfiguration mit mehreren logischen Partitionen, bei der ein Computer über mehrere Datenbankpartitionsserver verfügt.
- Eine Konfiguration, bei der mehrere logische Partitionen auf mehreren Computern ausgeführt werden.

Konfigurationen mit mehreren logischen Partitionen eignen sich gut, wenn mit dem System Abfragen auf einem Computer mit SMP-Architektur (SMP = Symmetric Multiprocessor) ausgeführt werden. Die Möglichkeit, mehrere logische Partitionen auf einem Computer zu konfigurieren, ist auch im Falle eines Computerausfalls von Vorteil. Wenn ein Computer (und damit auch der bzw. die auf diesem Computer ausgeführte(n) Datenbankpartitionsserver) ausfällt, können Sie den/die Datenbankpartitionsserver mit dem Befehl DB2START NODENUM auf einem anderen Computer neu starten. Dies stellt die kontinuierliche Verfügbarkeit der Benutzerdaten sicher.

Ein weiterer Vorteil bei mehreren logischen Partitionen besteht darin, dass diese die SMP-Hardwarekonfigurationen nutzen können. Darüber hinaus lassen sich durch die kleineren Datenbankpartitionen bessere Leistungswerte beim Backup und Restore von Datenbankpartitionen und Tabellenbereichen sowie bei der Indexerstellung erzielen.

Konfigurieren mehrerer logischer Partitionen

Es gibt zwei Methoden zur Konfiguration mehrerer logischer Partitionen.

- Konfigurieren der logischen Partitionen (Datenbankpartitionen) in der Datei db2nodes.cfg. Anschließend können Sie alle logischen und fernen Partitionen mit dem Befehl db2start bzw. der entsprechenden API starten.

Anmerkung: Für Windows müssen Sie zum Hinzufügen einer Datenbankpartition den Befehl db2ncrt verwenden, wenn auf dem System keine Datenbank vorhanden ist. Wenn bereits eine oder mehrere Datenbanken vorhanden sind, muss der Befehl db2start addnode verwendet werden. Unter Windows sollte die Datei db2nodes.cfg nie manuell bearbeitet werden.

- Neustarten einer logischen Partition auf einem anderen Prozessor, auf dem bereits andere logische Partitionen (Knoten) ausgeführt werden. Mit dieser Methode können Sie den Hostnamen und die Portnummer, die in der Datei db2nodes.cfg für die logische Partition angegeben sind, außer Kraft setzen.

Zur Konfiguration einer logischen Partition (Knoten) in der Datei db2nodes.cfg müssen Sie einen Eintrag in der Datei vornehmen, um der Datenbankpartition eine logische Portnummer zuzuordnen. Dabei gilt die folgende Syntax:

```
knotenummer hostname logischer-port netzwerkname
```

Anmerkung: Für Windows müssen Sie zum Hinzufügen einer Datenbankpartition den Befehl db2ncrt verwenden, wenn auf dem System keine Datenbank vorhanden ist. Wenn bereits eine oder mehrere Datenbanken vorhanden sind, muss der Befehl db2start addnode verwendet werden. Unter Windows sollte die Datei db2nodes.cfg nie manuell bearbeitet werden.

Das Format der Datei db2nodes.cfg unter Windows unterscheidet sich vom Format dieser Datei unter UNIX. Unter Windows gilt folgendes Spaltenformat:

```
knotennummer hostname computername logischer_port netzwerkname
```

Verwenden Sie den vollständig qualifizierten Namen für den Hostnamen. In der Datei /etc/hosts sollte ebenfalls der vollständig qualifizierte Name verwendet werden. Wenn der vollständig qualifizierte Name in der Datei db2nodes.cfg und in der Datei /etc/hosts nicht verwendet wird, empfangen Sie eventuell die Fehlernachricht SQL30082N RC=3.

Sie müssen sicherstellen, dass in der Datei services des Verzeichnisses etc genügend Ports für die FCM-Kommunikation definiert sind.

Aktivieren der partitionsübergreifenden Abfrageparallelität

Partitionsübergreifende Parallelität tritt entsprechend der Anzahl von Datenbankpartitionen und der Verteilung von Daten auf diese Datenbankpartitionen automatisch auf.

Anmerkung: Zur Nutzung der Parallelität innerhalb einer Datenbankpartition oder einer nicht partitionierten Datenbank müssen Sie die Konfigurationsparameter ändern. Durch die partitionsinterne Parallelität können Sie zum Beispiel mehrere Prozessoren einer SMP-Maschine (SMP = Symmetric Multiprocessor) nutzen.

Aktivieren der Parallelität zum Laden von Daten

Das Dienstprogramm LOAD nutzt die Parallelität automatisch, und Sie können die folgenden Parameter im Befehl LOAD verwenden:

- CPU_PARALLELISM
- DISK_PARALLELISM

In einer Umgebung mit partitionierten Datenbanken erfolgt die partitionsübergreifende Parallelität zum Laden von Daten automatisch, wenn die Zieltabelle für mehrere Datenbankpartitionen definiert ist. Die partitionsübergreifende Parallelität zum Laden von Daten kann durch die Angabe OUTPUT_DBPARTNUMS überschrieben werden. Das Dienstprogramm

LOAD macht außerdem intelligenten Gebrauch von der Parallelität durch die Datenbankpartitionierung in Abhängigkeit von der Größe der Zieldatenbankpartitionen. Der Parameter MAX_NUM_PART_AGENTS kann zur Steuerung des maximalen Grads der Parallelität verwendet werden, der von dem Dienstprogramm LOAD ausgewählt wird. Die Parallelität durch Datenbankpartitionierung kann durch Angeben von PARTITIONING-_DBPARTNUMS überschrieben werden, wenn außerdem ANYORDER angegeben wird.

Aktivieren der Parallelität beim Erstellen von Indizes

Für die Aktivierung der Parallelität bei der Erstellung eines Index gilt:

- Die Tabelle muss ausreichend groß sein, um die Parallelität vorteilhaft nutzen zu können.
- Auf einem SMP-Computer müssen mehrere Prozessoren aktiviert sein.

Aktivieren der E/A-Parallelität beim Backup einer Datenbank oder eines Tabellenbereichs

Gehen Sie wie folgt vor, um die E/A-Parallelität für das Backup einer Datenbank oder eines Tabellenbereichs zu aktivieren:

1. Verwenden Sie mehr als einen Zieldatenträger.
2. Konfigurieren Sie Tabellenbereiche für parallele E/A, indem Sie mehrere Container definieren, oder verwenden Sie einen einzigen Container mit mehreren Platten, und verwenden Sie die Registrierdatenbankvariable DB2_PARALLEL_IO entsprechend. Wenn Sie eine parallele E/A nutzen wollen, müssen Sie bereits vor dem Definieren von Containern die erforderlichen Vorbedingungen berücksichtigen und entsprechend vorgehen. Diese Vorbedingungen können nicht erst bei Erkennung eines Bedarfs erfüllt werden, sondern müssen eingeplant werden, bevor Sie den Punkt erreichen, an dem Sie Ihre Datenbank oder Ihren Tabellenbereich sichern müssen.
3. Geben Sie mit dem Parameter PARALLELISM im Befehl BACKUP den Grad der Parallelität an.
4. Verwenden Sie den Parameter WITH `anzahl-puffer` BUFFERS im Befehl BACKUP, um sicherzustellen, dass genügend Puffer für den Grad der Parallelität verfügbar sind. Die Anzahl der Puffer sollte der Summe aus der Anzahl Ihrer Zieldatenträger und dem ausgewählten Grad der Parallelität plus einigen zusätzlichen Puffern entsprechen.

   Verwenden Sie ferner einen Backup-Puffer, für dessen Größe Folgendes gilt:

   - So groß wie möglich. 4 MB oder 8 MB (1024 oder 2048 Seiten) ist eine zweckmäßige Faustregel.
   - Mindestens so groß wie das größte Produkt (EXTENTSIZE * Anzahl der Container) der zu sichernden Tabellenbereiche.

Aktivieren der E/A-Parallelität beim Restore einer Datenbank oder eines Tabellenbereichs

Gehen Sie wie folgt vor, um die E/A-Parallelität für den Restore einer Datenbank oder eines Tabellenbereichs zu aktivieren:

- Verwenden Sie mehr als einen Quellendatenträger.
- Konfigurieren Sie Tabellenbereiche für eine parallele E/A. Sie müssen die Entscheidung für die Verwendung dieser Option treffen, bevor Sie Ihre Container definieren. Die entsprechenden Vorbedingungen können nicht erst bei Erkennung eines Bedarfs erfüllt werden, sondern müssen eingeplant werden, bevor Sie den Punkt erreichen, an dem Sie Ihre Datenbank oder Ihren Tabellenbereich wiederherstellen müssen.

- Geben Sie mit dem Parameter PARALLELISM im Befehl RESTORE den Grad der Parallelität an.
- Verwenden Sie den Parameter WITH `anzahl-puffer` BUFFERS im Befehl RESTORE, um sicherzustellen, dass genügend Puffer für den Grad der Parallelität verfügbar sind. Die Anzahl der Puffer sollte der Summe aus der Anzahl Ihrer Zieldatenträger und dem ausgewählten Grad der Parallelität plus einigen zusätzlichen Puffern entsprechen.

  Verwenden Sie ferner einen Restorepuffer, für dessen Größe Folgendes gilt:
  - So groß wie möglich. 4 MB oder 8 MB (1024 oder 2048 Seiten) ist eine zweckmäßige Faustregel.
  - Mindestens so groß wie das größte Produkt (EXTENTSIZE * Anzahl der Container) der wiederherzustellenden Tabellenbereiche.
  - Dieselbe Größe oder ein gerades Vielfaches der Größe des Backup-Puffers.

Aktivieren der partitionsinternen Parallelität für Abfragen

Zum Ermitteln der Werte für individuelle Einträge in einer bestimmten Datenbank oder der Konfigurationsdatei des Datenbankmanagers können Sie auch die Befehle GET DATABASE CONFIGURATION und GET DATABASE MANAGER CONFIGURATION verwenden. Zur Änderung einzelner Einträge für eine bestimmte Datenbank oder in der Konfigurationsdatei des Datenbankmanagers werden die Befehle UPDATE DATABASE CONFIGURATION und UPDATE DATABASE MANAGER CONFIGURATION verwendet.

Konfigurationsparameter, die sich auf die partitionsinterne Parallelität auswirken, sind die Datenbankmanagerparameter *max_querydegree* und *intra_parallel* sowie der Datenbankparameter *dft_degree*.

Zur Einrichtung der partitionsinternen Abfrageparallelität müssen Sie einen oder mehrere Konfigurationsparameter der Datenbank und des Datenbankmanagers, Vorkompilierungs- und Bindeoptionen oder ein Sonderregister ändern.

intra_parallel
: Dieser Konfigurationsparameter des Datenbankmanagers gibt an, ob der Datenbankmanager partitionsinterne Parallelität verwenden kann. Der Standardwert gibt an, dass die partitionsinterne Parallelität nicht verwendet werden kann.

max_querydegree
: Dieser Konfigurationsparameter des Datenbankmanagers gibt den maximalen Grad partitionsinterner Parallelität an, der für SQL-Anweisungen verwendet wird, die auf dieser Instanz ausgeführt werden. Eine SQL-Anweisung führt nicht mehr als diese Anzahl paralleler Operationen innerhalb einer Datenbankpartition aus. Der Konfigurationsparameter *intra_parallel* muss zudem auf den Wert „YES" gesetzt sein, damit der Wert des Parameters *max_querydegree* verwendet wird. Der Standardwert für diesen Konfigurationsparameter ist -1. Dieser Wert bedeutet, dass das System den vom Optimierungsprogramm festgelegten Parallelitätsgrad verwendet. Andernfalls wird der vom Benutzer angegebene Wert verwendet.

dft_degree
: Ein Datenbankkonfigurationsparameter, der den Standardwert für die Bindeoption DEGREE und das Sonderregister CURRENT DEGREE angibt. Der

Standardwert ist 1. Der Wert ANY bedeutet, dass das System den vom Optimierungsprogramm festgelegten Parallelitätsgrad verwendet.

DEGREE
Option für den Vorcompiler bzw. das Binden von statischem SQL.

CURRENT DEGREE
Sonderregister für dynamisches SQL.

Verwaltung der Datenserverkapazität

Wenn die Kapazität des Datenservers nicht Ihren gegenwärtigen oder zukünftigen Anforderungen genügt, können Sie die Kapazität des Datenservers erweitern, indem Sie Plattenspeicherplatz hinzufügen und weitere Container erstellen oder Speicher hinzufügen. Wenn diese einfachen Strategien nicht die erforderliche Kapazität erbringen, können Sie auch das Hinzufügen von Prozessoren oder physischen Partitionen in Betracht ziehen. Wenn Sie Ihr System durch Änderung der Umgebung skalieren, sollten Sie sich über die Auswirkungen dieser Änderung auf die Prozeduren in Ihrer Datenbank wie zum Beispiel auf das Laden von Daten oder das Backup oder den Restore von Datenbanken im Klaren sein.

Hinzufügen von Prozessoren

Wenn eine Konfiguration mit Einzelpartitionsdatenbanken und einem Einzelprozessor bis zur maximalen Kapazitätsgrenze ausgelastet wird, können Sie entweder Prozessoren oder Datenbankpartitionen hinzufügen. Der Vorteil hinzugefügter Prozessoren liegt in der größeren Verarbeitungskapazität. In einem SMP-System benutzen Prozessoren Hauptspeicher und Speichersystemressourcen gemeinsam. Alle Prozessoren befinden sich in einem System, sodass keine Aufwände zu beachten sind, wie zum Beispiel die Kommunikation oder die Koordination von Tasks zwischen Systemen. Die Dienstprogramme beispielsweise zum Laden, Backup und Restore können die zusätzlichen Prozessoren vorteilhaft nutzen.

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

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

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

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

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

Hinzufügen physischer Partitionen

Wenn sich Ihr Datenbankmanager momentan in einer Umgebung mit partitionierten Datenbanken befindet, können Sie sowohl die Datenspeicherkapazitäten als auch die Verarbeitungskapazitäten erhöhen, indem Sie separate physische Einzelprozessor- oder Multiprozessorpartitionen hinzufügen.

Die Hauptspeicher- und Speichersystemressourcen jeder Datenbankpartition werden nicht mit den anderen Datenbankpartitionen gemeinsam genutzt. Zwar kann das Hinzufügen von Datenbankpartitionen mit Problemen hinsichtlich der Kommunikation und der Taskkoordination verbunden sein, aber diese Wahl hat den Vorteil, dass Daten und Benutzerzugriffe auf mehr als ein System verteilt werden können. Der Datenbankmanager unterstützt diese Umgebung.

Sie können Datenbankpartitionen hinzufügen, während das Datenbankmanagersystem aktiv bzw. während es gestoppt ist. Wenn Sie Datenbankpartitionen hinzufügen, während das System aktiv ist, müssen Sie das System jedoch stoppen und neu starten, bevor Datenbanken auf die neue Datenbankpartition migriert werden.

Wenn Sie eine neue Datenbankpartition hinzufügen, können Sie eine Datenbank, die die neue Datenbankpartition nutzt, erst dann löschen oder erstellen, wenn die Prozedur abgeschlossen und der neue Server erfolgreich in das System integriert ist.

Fast Communications Manager

Fast Communication Manager (Windows)

FCM (Fast Communication Manager) stellt die Kommunikationsunterstützung für DB2-Serverprodukte zur Verfügung, die zu derselben Instanz gehören. Jeder Datenbankpartitionsserver verfügt über einen FCM-Senderdämon und einen FCM-Empfängerdämon, der die Kommunikation zwischen Datenbankpartitionsservern zur Verfügung stellt, um Anforderungen von Agenten zu verarbeiten und Nachrichtenpuffer zu senden. Der FCM-Dämon wird beim Start der Instanz gestartet.

Schlägt die Kommunikation zwischen Datenbankpartitionsservern fehl oder wird die Kommunikation wiederhergestellt, aktualisiert der FCM-Thread Informationen (diese Informationen können mit dem Datenbanksystemmonitor abgefragt werden) und veranlasst, dass die entsprechende Aktion (z. B. ein Rollback der betroffenen Transaktion) ausgeführt wird. Sie können den Datenbanksystemmonitor verwenden, um Unterstützung beim Einstellen der FCM-Konfigurationsparameter zu erhalten.

Sie können die Anzahl der FCM-Nachrichtenpuffer mit dem Konfigurationsparameter *fcm_num_buffers* des Datenbankmanagers und die Anzahl der FCM-Kanäle mit dem Konfigurationsparameter *fcm_num_channels* des Datenbankmanagers festlegen. Die Konfigurationsparameter *fcm_num_buffers* und *fcm_num_channels* des Datenbankmanagers sind standardmäßig auf den Wert AUTOMATIC eingestellt. Der FCM überwacht die Ressourcennutzung, wenn diese Parameter auf AUTOMATIC gesetzt sind, und gibt Ressourcen teilweise frei. Es wird empfohlen, die Einstellung für diese Parameter auf AUTOMATIC zu belassen.

Fast Communication Manager (Linux und UNIX)

Fast Communication Manager (FCM) stellt die Kommunikationsunterstützung für DB2-Serverprodukte zur Verfügung, die Database Partitioning Feature (DPF) verwenden.

Bei Instanzen mit mehreren Partitionen verfügt jeder Datenbankpartitionsserver über einen FCM-Senderdämon und einen FCM-Empfängerdämon, der die Kommunikation zwischen Datenbankpartitionsservern zur Verfügung stellt, um Anforde-

rungen von Agenten zu verarbeiten und Nachrichtenpuffer zu senden. Der FCM-Dämon wird beim Start der Instanz mit mehreren Partitionen gestartet.

Schlägt die Kommunikation zwischen Datenbankpartitionsservern fehl oder wird die Kommunikation wiederhergestellt, aktualisieren die FCM-Dämonen Informationen. Sie können diese Informationen mit dem Datenbanksystemmonitor abrufen. Die FCM-Dämonen lösen außerdem die entsprechende Aktion aus. Ein Beispiel für eine entsprechende Aktion ist die Rollback-Operation für eine betroffene Transaktion. Sie können den Datenbanksystemmonitor verwenden, um Unterstützung beim Einstellen der FCM-Konfigurationsparameter zu erhalten.

Sie können die Anzahl der FCM-Nachrichtenpuffer mit dem Konfigurationsparameter *fcm_num_buffers* des Datenbankmanagers festlegen. Außerdem können Sie die Anzahl der FCM-Kanäle mit dem Konfigurationsparameter *fcm_num_channels* des Datenbankmanagers festlegen. Die Konfigurationsparameter *fcm_num_buffers* und *fcm_num_channels* des Datenbankmanagers werden standardmäßig auf den Wert AUTOMATIC gesetzt. Der FCM überwacht die Ressourcennutzung, wenn diese Parameter auf AUTOMATIC gesetzt sind, und gibt Ressourcen teilweise frei. Es wird empfohlen, die Einstellung für diese Parameter auf AUTOMATIC zu belassen.

Aktivieren der Kommunikation zwischen Datenbankpartitionen durch FCM-Kommunikation

In einer partitionierten Datenbankumgebung wird ein Großteil der Kommunikation zwischen Datenbankpartitionen von FCM (Fast Communication Manager) verarbeitet.

Zur Aktivierung von FCM in einer Datenbankpartition und zur Ermöglichung der Kommunikation mit anderen Datenbankpartitionen müssen Sie einen Serviceeintrag in der Datei `services` des Verzeichnisses `etc` der Datenbankpartition erstellen. Die Vorgehensweise wird im Folgenden erläutert. FCM verwendet den angegebenen Port für die Kommunikation. Wenn mehrere Datenbankpartitionen auf demselben Host definiert sind, müssen Sie einen Bereich von Ports, wie unten gezeigt, definieren.

Bevor Sie versuchen, Speicher für FCM manuell zu konfigurieren, wird empfohlen, mit der Einstellung AUTOMATIC für die Anzahl von FCM-Puffern (*fcm_num_buffers*) und die Anzahl von FCM-Kanälen (*fcm_num_channels*) zu beginnen, die gleichzeitig die Standardeinstellung ist. Stellen Sie mithilfe der Systemmonitordaten für die FCM-Aktivität fest, ob diese Einstellung geeignet ist.

Windows-Aspekte

Wenn Sie DB2 Enterprise Server Edition in der Windows-Umgebung verwenden, wird der TCP/IP-Portbereich durch folgende Programme automatisch der Datei 'services' hinzugefügt:

- Das Installationsprogramm, wenn es die Instanz erstellt oder eine neue Datenbankpartition hinzufügt.
- Das Dienstprogramm db2icrt, wenn es eine neue Instanz erstellt.
- Das Dienstprogramm db2ncrt, wenn es die erste Datenbankpartition auf dem Computer hinzufügt.

Der Serviceeintrag hat folgende Syntax:

```
DB2_instanz port/tcp #kommentar
```

DB2_*instanz*
Der Wert für *instanz* ist der Name der Datenbankmanagerinstanz. Alle Zeichen des Namens müssen in Kleinschreibung angegeben sein. Wenn der Instanzname `db2puser` wäre, müssten Sie `DB2_db2puser` angeben.

port/**tcp**
Der TCP/IP-Port, den Sie für die Datenbankpartition reservieren wollen.

#*kommentar*
Ein beliebiger Kommentar, den Sie dem Eintrag hinzufügen wollen. Dem Kommentar muss ein Nummernzeichen (#) vorangestellt werden.

Wenn die Datei `services` des Verzeichnisses `etc` gemeinsam genutzt wird, müssen Sie sicherstellen, dass die Anzahl der in der Datei zugeordneten Ports größer oder gleich der größten Anzahl mehrerer Datenbankpartitionen in dieser Instanz ist. Stellen Sie bei der Zuordnung von Ports außerdem sicher, dass Sie jeden möglichen Prozessor, der als Ersatz verwendet werden kann, mit berücksichtigen.

Wenn die Datei `services` im Verzeichnis `etc` nicht gemeinsam genutzt wird, gelten dieselben Überlegungen, jedoch mit einem zusätzlichen Gesichtspunkt: Sie müssen sicherstellen, dass die für die DB2-Datenbankinstanz definierten Einträge in allen Dateien `services` des Verzeichnisses `etc` die gleichen sind (obwohl andere Einträge, die sich nicht auf Ihre partitionierte Datenbankumgebung beziehen, nicht gleich sein müssen).

Wenn Sie mehrere Datenbankpartitionen auf demselben Host in einer Instanz haben, müssen Sie mehr als einen Port für die Verwendung durch FCM definieren. Dazu müssen zwei Zeilen in die Datei `services` des Verzeichnisses `etc` eingefügt werden, die den Bereich von Ports angeben, den Sie zuordnen. Die erste Zeile gibt den ersten Port an, während die zweite Zeile das Ende des Blocks von Ports angibt. Im folgenden Beispiel werden fünf Ports für die Instanz `sales` zugeordnet. Dies bedeutet, dass kein Prozessor in der Instanz mehr als fünf Datenbankpartitionen hat. Beispiel:

```
DB2_sales         9000/tcp
DB2_sales_END     9004/tcp
```

Anmerkung: Die Zeichenfolge `END` muss unbedingt durchgehend in Großschreibung angegeben werden. Außerdem müssen unbedingt beide Unterstreichungszeichen (_) vorhanden sein.

Aktivieren der Kommunikation zwischen Datenbankpartitionsservern (Linux und UNIX)

Diese Task beschreibt, wie die Kommunikation zwischen den Datenbankpartitionsservern aktiviert wird, die Ihrem partitionierten Datenbanksystem angehören. Die Kommunikation zwischen Datenbankpartitionsservern wird von Fast Communication Manager (FCM) gesteuert. Zum Aktivieren von FCM muss auf jedem Computer im partitionierten Datenbanksystem ein Port oder Portbereich in der Datei `/etc/services` reserviert sein.

Sie müssen über eine Benutzer-ID mit Rootberechtigung verfügen.

Sie müssen diese Task auf allen Computern ausführen, die an der Instanz beteiligt sind.

Die Anzahl der für FCM zu reservierenden Ports entspricht der maximalen Anzahl von Datenbankpartitionen, die von einem der Computer in der Instanz per Hosting bereitgestellt wird oder potenziell bereitgestellt werden kann.

Die Datei db2nodes.cfg im folgenden Beispiel enthält diese Einträge:

```
0 server1 0
1 server1 1
2 server2 0
3 server2 1
4 server2 2
5 server3 0
6 server3 1
7 server3 2
8 server3 3
```

Im Beispiel wird davon ausgegangen, dass die FCM-Ports beginnend mit 60000 nummeriert sind. In dieser Situation gilt Folgendes:

- server1 verwendet zwei Ports (60000, 60001) für seine zwei Datenbankpartitionen
- server2 verwendet drei Ports (60000, 60001, 60002) für seine drei Datenbankpartitionen
- server3 verwendet vier Ports (60000, 60001, 60002, 60003) für seine vier Datenbankpartitionen

Alle Computer müssen die Ports 60000, 60001, 60002 und 60003 reservieren, da diese den größten Portbereich bilden, der für einen der Computer in der Instanz erforderlich ist.

Wenn Sie eine Hochverfügbarkeitslösung wie High Availability Cluster Multi-Processing (HACMP) oder Tivoli System Automation (TSA) verwenden, um Datenbankpartitionen von einem Computer an einen anderen übergeben zu können, müssen Sie die möglichen Portanforderungen berücksichtigen. Wenn ein Computer zum Beispiel normalerweise vier Datenbankpartitionen per Hosting bereitstellt, er jedoch potenziell zwei Datenbankpartitionen eines anderen Computers übernehmen kann, müssen für diesen Computer sechs Ports geplant werden.

Wenn Sie eine Instanz erstellen, wird auf dem Primärcomputer ein Portbereich reserviert. Der Primärcomputer wird auch als Instanzeigner bezeichnet. Wenn der Portbereich, der der `/etc/services`-Datei ursprünglich hinzugefügt wurde, für Ihren Bedarf jedoch nicht ausreichend ist, müssen Sie den Bereich der reservierten Ports durch manuelles Hinzufügen weiterer Einträge erweitern.

Gehen Sie wie folgt vor, um die Kommunikation zwischen Servern in einer Umgebung mit partitionierten Datenbanken unter Verwendung von `/etc/services` zu aktivieren:

1. Melden Sie sich am Primärcomputer (dem Computer, dem die Instanz gehört) als Benutzer mit Rootberechtigung an.
2. Erstellen Sie eine Instanz.
3. Zeigen Sie den Standardportbereich an, der in der Datei `/etc/services` reserviert wurde. Zusätzlich zur Basiskonfiguration sollten die FCM-Ports etwa wie folgt aussehen:

   ```
   db2c_db2inst1         50000/tcp
   #Add FCM port information
   DB2_db2inst1        60000/tcp
   DB2_db2inst1_1        60001/tcp
   DB2_db2inst1_2        60002/tcp
   DB2_db2inst1_END      60003/tcp
   ```

Standardmäßig ist der erste Port (50000) für Verbindungsanforderungen reserviert, und die ersten vier verfügbaren Ports über 60000 sind standardmäßig für die FCM-Kommunikation reserviert. Ein Port ist für den Datenbankpartitionsserver reserviert, der als Instanzeigner fungiert. Drei Ports sind für Server logischer Datenbankpartitionen reserviert, die Sie nach Abschluss der Installation möglicherweise dem Computer hinzufügen möchten.

Der Portbereich muss einen Start- und einen Endeintrag aufweisen. Zwischeneinträge sind optional. Das explizite Aufnehmen von Zwischenwerten kann nützlich sein, um die Verwendung dieser Ports durch andere Anwendungen zu verhindern. Diese Einträge werden vom Datenbankmanager jedoch nicht geprüft.

Die DB2-Porteinträge verwenden das folgende Format:

```
DB2_instanzname_suffix  portnummer/tcp  # kommentar
```

Dabei gilt Folgendes:

- *instanzname* ist der Name der partitionierten Instanz.
- *suffix* wird für den ersten FCM-Port nicht verwendet. Zwischeneinträge sind die Einträge zwischen dem niedrigsten und dem höchsten Port. Wenn Sie die Zwischeneinträge zwischen dem ersten und dem letzten FCM-Port einfügen, besteht das *suffix* aus einer ganzen Zahl, die für jeden weiteren Port jeweils um eins erhöht wird. Auf diese Weise wird der zweite Port beispielsweise mit der Zahl 1 nummeriert, der dritte Port mit der Zahl 2 und so weiter, um die Eindeutigkeit zu gewährleisten. Für den letzten Eintrag muss das Wort END als *suffix* verwendet werden.
- *portnummer* ist die portnummer, die Sie für die Kommunikation zwischen Datenbankpartitionsservern reservieren.
- *kommentar* ist ein optionaler Kommentar, der einen Eintrag beschreibt.

4. Stellen Sie sicher, dass für die FCM-Kommunikation ausreichend Ports reserviert sind. Wenn der Bereich der reservierten Ports nicht ausreichend ist, fügen Sie der Datei neue Einträge hinzu.
5. Melden Sie sich nacheinander jeweils an allen zur Instanz zugehörigen Computern als Benutzer mit Rootberechtigung an, und fügen Sie der Datei `/etc/services` identische Einträge hinzu.

Kapitel 10. Erstellen und Verwalten von Umgebungen mit partitionierten Datenbanken

Erste Datenbankpartitionsgruppen

Wenn eine Datenbank anfangs erstellt wird, werden Datenbankpartitionen für alle in der Datei 'db2nodes.cfg' angegebenen Datenbankpartitionen erstellt. Andere Datenbankpartitionen können mit den Befehlen ADD DBPARTITIONNUM und DROP DBPARTITIONNUM VERIFY hinzugefügt bzw. gelöscht werden.

Es werden drei Datenbankpartitionsgruppen definiert:

- IBMCATGROUP ist die Datenbankpartitionsgruppe für den Tabellenbereich SYSCATSPACE, in dem die Systemkatalogtabellen gespeichert werden.
- IBMTEMPGROUP ist die Datenbankpartitionsgruppe für den Tabellenbereich TEMPSPACE1, in dem temporäre Tabellen gespeichert werden, die bei Datenbankoperationen erstellt werden.
- IBMDEFAULTGROUP ist die Datenbankpartitionsgruppe für den Tabellenbereich USERSPACE1, der standardmäßig die Benutzertabellen und Indizes enthält.

Erstellen von Datenbankpartitionsgruppen

Eine Datenbankpartitionsgruppe wird mit der Anweisung CREATE DATABASE PARTITION GROUP erstellt. Diese Anweisung gibt eine Gruppe von Datenbankpartitionen an, in der sich die Container eines Tabellenbereichs und die Tabellendaten befinden sollen.

Die Computer und Systeme müssen verfügbar sein und eine Umgebung mit partitionierten Datenbanken verarbeiten können. Sie haben DB2 Enterprise Server Edition erworben und installiert. Die Datenbank muss vorhanden sein.

Außerdem führt diese Anweisung folgende Operationen aus:

- Sie erstellt eine Verteilungszuordnung für die Datenbankpartitionsgruppe.
- Sie generiert eine ID für die Verteilungszuordnung.
- Sie fügt Datensätze in die folgenden Katalogtabellen ein:
  - SYSCAT.DBPARTITIONGROUPS
  - SYSCAT.PARTITIONMAPS
  - SYSCAT.DBPARTITIONGROUPDEF

Gehen Sie wie folgt vor, um eine Datenbankpartitionsgruppe über die Steuerzentrale zu erstellen:

1. Erweitern Sie die Sicht der Objektbaumstruktur so lange, bis der Ordner **Datenbankpartitionsgruppen** angezeigt wird.
2. Klicken Sie mit der rechten Maustaste auf den Ordner **Datenbankpartitionsgruppen**, und wählen Sie im Kontextmenü **Erstellen** aus.
3. Geben Sie im Fenster **Datenbankpartitionsgruppen erstellen** die erforderlichen Informationen ein, verschieben Sie mithilfe der Pfeile Knoten aus dem Feld **Verfügbare Partitionen** in das Feld **Ausgewählte Datenbankpartitionen**, und klicken Sie **OK** an.

Geben Sie in der Befehlszeile Folgendes ein, um eine Datenbankpartitionsgruppe zu erstellen:

```
CREATE DATABASE PARTITION GROUP db-partitionsgruppenname
   ON DBPARTITIONNUM (db-partitionsnummer1,db-partitionsnummer2)
```

oder

```
CREATE DATABASE PARTITION GROUP db-partitionsgruppenname
   ON DBPARTITIONNUMS (db-partitionsnummer1 TO db-partitionsnummer2)
```

Nehmen Sie zum Beispiel an, Sie möchten einige Tabellen in eine Untermenge der Datenbankpartitionen in Ihrer Datenbank laden. Sie würden den folgenden Befehl verwenden, um eine Datenbankpartitionsgruppe mit zwei Datenbankpartitionen (1 und 2) in einer Datenbank zu erstellen, die aus mindestens drei Datenbankpartitionen (0 bis 2) besteht:

```
CREATE DATABASE PARTITION GROUP mixng12 ON DBPARTITIONNUM (1,2)
```

oder

```
CREATE DATABASE PARTITION GROUP mixng12 ON DBPARTITIONNUMS (1 TO 2)
```

Der Befehl CREATE DATABASE bzw. die API sqlecrea() erstellen auch die Standarddatenbankpartitionsgruppen des Systems IBMDEFAULTGROUP, IBMCATGROUP und IBMTEMPGROUP.

Tabellenbereiche in Datenbankpartitionsgruppen

Wenn ein Tabellenbereich in einer Datenbankpartitionsgruppe mit mehreren Datenbankpartitionen erstellt wird, werden alle Tabellen innerhalb des Tabellenbereichs auf alle Datenbankpartitionen in der Datenbankpartitionsgruppe verteilt oder partitioniert.

Der Tabellenbereich wird in einer Datenbankpartitionsgruppe erstellt. Wenn der Tabellenbereich in einer Datenbankpartitionsgruppe erstellt wurde, muss er dort verbleiben und kann nicht in eine andere Datenbankpartitionsgruppe versetzt werden. Für die Zuordnung eines Tabellenbereichs zu einer Datenbankpartitionsgruppe wird die Anweisung CREATE TABLESPACE verwendet.

Verwalten von Datenbankpartitionen

Über die Sicht 'Partitionen' der Steuerzentrale können Sie Partitionen starten und stoppen, Partitionen löschen, Traceoperationen für Partitionen durchführen und das Diagnoseprotokoll anzeigen.

Zur Arbeit mit Datenbankpartitionen bzw. zum Anzeigen der DB2-Protokolle müssen Sie über die Berechtigung zum Herstellen der Verbindung (ATTACH) zu einer Instanz verfügen. Jeder Benutzer mit der Berechtigung SYSADM oder DBADM kann Ihnen die Berechtigung zum Zugriff auf eine bestimmte Instanz erteilen.

Wenn Sie dazu von der IBM Unterstützungsfunktion aufgefordert werden, führen Sie das Tracedienstprogramm mit den von der Unterstützungsfunktion angegebenen Optionen aus. Das Tracedienstprogramm zeichnet Informationen zu DB2-Operationen auf und formatiert diese Informationen in lesbarer Form. Weitere Informationen finden Sie in der Beschreibung des DB2-Befehls db2trc (Trace).

Achtung: Verwenden Sie das Tracedienstprogramm nur, wenn Sie vom DB2-Kundendienst bzw. von einem technischen Unterstützungsbeauftragten dazu aufgefordert werden.

Im Fenster **'Diagnoseprotokoll'** können Sie die vom DB2-Tracedienstprogramm protokollierten Informationen in Form von Textdaten prüfen.

Die Sicht 'Partitionen' zeigt die folgenden Informationen an:

Knotennummer
Diese Spalte enthält Symbole und Knotennummern. Die Knotennummern sind eindeutige Nummern aus dem Bereich von 0 bis 999. Die Nummern sind in der Datei `db2nodes.cfg` gespeichert. Die Knotennummern werden in aufsteigender Reihenfolge angezeigt, wobei Lücken in der Reihenfolge vorhanden sein können.

Einmal zugeordnete Knotennummern können nicht geändert werden. Dies stellt sicher, dass die Integrität der Daten in der Verteilungszuordnung (die Details zur Partitionierung der Daten enthält) nicht beeinträchtigt wird.

Hostname
Der Hostname ist die IP-Adresse, die von FCM (Fast Communication Manager) für die interne Kommunikation verwendet wird. (Wird jedoch ein Schaltername angegeben, verwendet FCM den Schalternamen. In diesem Fall wird der Hostname nur für DB2START, DB2STOP und db2_all verwendet.) Der Hostname ist in der Datei `db2nodes.cfg` gespeichert.

Portnummer
Die Portnummer ist die logische Portnummer für den Knoten. Diese Nummer wird mit dem Instanznamen des Datenbankmanagers verwendet, um einen TCP/IP-Servicenamenseintrag in der Datei `/etc/services` anzugeben. Diese Nummer ist in der Datei `db2nodes.cfg` gespeichert.

Die Kombination aus IP-Adresse (Hostname) und logischem Port wird als herkömmliche Adresse verwendet. Sie muss für alle Anwendungen eindeutig sein, damit Kommunikationsverbindungen zwischen den Knoten unterstützt werden.

Für jeden angezeigten Hostnamen ist eine Portnummer die Nummer 0. Die Portnummer 0 gibt den Standardknoten auf dem Host an, zu dem Clients eine Verbindung herstellen. (Dies kann mit der Umgebungsvariablen **DB2NODE** im Script db2profile überschrieben werden.)

Schaltername
Der Schaltername (bzw. Netzwerkname) wird zur Unterstützung eines Hosts verwendet, der über mehrere aktive TCP/IP-Schnittstellen mit jeweils eigenen Hostnamen verfügt. Er wird außerdem für die schnelle Kommunikation (auch als FCM bezeichnet) zwischen Knoten verwendet, die einen Hochgeschwindigkeitsschalter (Switch) gemeinsam nutzen. Der Schaltername wird in der Datei `db2nodes.cfg` gespeichert. Wenn der Schaltername in der Datei `db2nodes.cfg` nicht angegeben wird, entspricht er dem Hostnamen.

Ressourcengruppenname
Der Ressourcengruppenname (Resourcesetname) definiert die Betriebssystemressource, in der der Knoten gestartet werden soll. Der Ressourcengruppenname dient zur Unterstützung der Prozessaffinität, die für mehrere logische Knoten (MLNs, Multiple Logical Nodes) verwendet wird und mit einem Feld vom Typ 'string' angegeben wird, das bisher als 'quadname' bezeichnet wurde.

1. Öffnen Sie die Sicht 'Partitionen': Erweitern Sie in der Steuerzentrale die Objektbaumstruktur, bis die Instanz angezeigt wird, für die Sie die Partitionen anzeigen möchten. Klicken Sie die gewünschte Instanz mit der rechten Maustaste an und wählen Sie **Öffnen → Partitionen** im Kontextmenü aus. Die Sicht Partitionen wird geöffnet.
2. Zum Starten von Partitionen: Heben Sie eine oder mehrere Partitionen hervor und wählen Sie **Partitionen → Starten** aus. Die ausgewählten Partitionen werden gestartet.
3. Zum Stoppen von Partitionen: Heben Sie eine oder mehrere Partitionen hervor und wählen Sie **Partitionen → Stoppen** aus. Die ausgewählten Partitionen werden gestoppt.
4. Gehen Sie wie folgt vor, um das Tracedienstprogramm für eine Partition auszuführen:
   a. Öffnen Sie das Fenster 'DB2-Trace': Heben Sie eine Partition hervor und wählen Sie **Partitionen → Service → Trace** aus. Das Fenster **'DB2-Trace'** wird geöffnet.
   b. Geben Sie die Traceoptionen an.
   c. Klicken Sie **Starten** an, um mit dem Aufzeichnen von Informationen zu beginnen. Klicken Sie **Stoppen** an, um das Aufzeichnen von Informationen zu beenden. Klicken Sie **Speichern unter** an, um die Informationen in einer Datei zu speichern.
   d. Optional: Prüfen Sie die Protokolle.
   e. Senden Sie die Traceausgabe an die IBM Unterstützungsfunktion, wenn Sie dazu aufgefordert werden.

Hinzufügen von Datenbankpartitionen in Umgebungen mit partitionierten Datenbanken

Sie können einem System mit einer partitionierten Datenbank Datenbankpartitionen hinzufügen, wenn es aktiv ist oder wenn es gestoppt ist. Da das Hinzufügen eines neuen Servers sehr zeitaufwendig sein kann, ist es vielleicht wünschenswert, dies zu tun, wenn der Datenbankmanager bereits aktiv ist.

Verwenden Sie den Befehl ADD DBPARTITIONNUM, um einem System eine Datenbankpartition hinzuzufügen. Dieser Befehl kann folgendermaßen aufgerufen werden:

- Als Option im Befehl `db2start`
- Mit dem Befehl ADD DBPARTITIONNUM über den Befehlszeilenprozessor
- Mit der API-Funktion `sqleaddn`
- Mit der API-Funktion `sqlepstart`

Wenn Ihr System gestoppt ist, verwenden Sie den Befehl `db2start`. Wenn es aktiv ist, können Sie eine der anderen Möglichkeiten verwenden.

Wenn Sie dem System mit dem Befehl ADD DBPARTITIONNUM eine neue Datenbankpartition hinzufügen, werden alle in dieser Instanz vorhandenen Datenbanken auf die neue Datenbankpartition erweitert. Sie können auch angeben, welche Container für temporäre Tabellenbereiche für die Datenbanken verwendet werden sollen. Für die Container gibt es folgende Möglichkeiten:

- Es sind dieselben Container wie die, die für die Katalogpartition der jeweiligen Datenbanken definiert sind (Standardeinstellung).
- Es sind dieselben Container wie die, die für eine andere Datenbankpartition definiert sind.

- Sie wurden noch nicht erstellt. Sie müssen jeder Datenbank mithilfe der Anweisung ALTER TABLESPACE Tabellenbereichscontainer für temporäre Tabellen hinzufügen, bevor die Datenbank verwendet werden kann.

Sie können eine Datenbank in der neuen Datenbankpartition nicht zur Speicherung von Daten verwenden, bevor nicht mindestens eine Datenbankpartitionsgruppe zur Aufnahme der neuen Datenbankpartition geändert wurde.

Sie können nicht von einer Einzelpartitionsdatenbank zu einer Mehrpartitionsdatenbank wechseln, indem Sie Ihrem System einfach eine Partition hinzufügen, da für die Umverteilung von Daten zwischen Datenbankpartitionen für jede betroffene Tabelle ein entsprechender Verteilungsschlüssel erforderlich ist. Beim Erstellen einer Tabelle in einer Mehrpartitionsdatenbank werden die Verteilungsschlüssel automatisch generiert. In einer Einzelpartitionsdatenbank können Verteilungsschlüssel mithilfe der SQL-Anweisungen CREATE TABLE oder ALTER TABLE explizit erstellt werden.

Anmerkung: Wenn keine Datenbanken im System definiert sind und Sie Enterprise Server Edition unter einem UNIX-Betriebssystem ausführen, bearbeiten Sie die Datei `db2nodes.cfg`, um eine neue Definition für eine Datenbankpartition hinzuzufügen. Verwenden Sie keine der beschrieben Prozeduren, da sie nur gelten, wenn eine Datenbank vorhanden ist.

Für Windows ist zu beachten: Wenn Sie Enterprise Server Edition unter Windows verwenden und keine Datenbanken in der Instanz definiert haben, verwenden Sie den Befehl DB2NCRT, um das Datenbanksystem zu skalieren. Wenn Sie jedoch bereits Datenbanken haben, verwenden Sie den Befehl DB2START ADDNODE, um sicherzustellen, dass beim Skalieren des Systems für jede vorhandene Datenbank eine Datenbankpartition erstellt wird. Unter Windows sollten Sie die Knotenkonfigurationsdatei (`db2nodes.cfg`) nie manuell editieren, da dies zu Inkonsistenzen in der Datei führen kann.

Hinzufügen einer Datenbankpartition zu einem aktiven Datenbanksystem

Sie können einer Umgebung mit partitionierten Datenbanken neue Datenbankpartitionen hinzufügen, während sie aktiv ist und Anwendungen mit Datenbanken verbunden sind. Allerdings wird der hinzugefügte Server erst dann für alle Datenbanken verfügbar, wenn der Datenbankmanager gestoppt und erneut gestartet wird.

Gehen Sie wie folgt vor, um unter Verwendung der Befehlszeile einem aktiven Datenbankmanager eine Datenbankpartition hinzuzufügen:

1. Führen Sie den Befehl DB2START in jeder vorhandenen Datenbankpartition aus.

   Geben Sie auf allen Plattformen die neuen Datenbankpartitionswerte für die Parameter DBPARTITIONNUM, ADD DBPARTITIONNUM, HOSTNAME, PORT und NETNAME an. Auf der Windows-Plattform geben Sie außerdem die Parameter COMPUTER, USER und PASSWORD an.

   Sie können auch die Quelle für die Definitionen von Tabellenbereichscontainern für temporäre Tabellen angeben, die mit den Datenbanken erstellt werden müssen. Wenn Sie keine Tabellenbereichsinformationen angeben, werden die Definitionen für Tabellenbereichscontainer für temporäre Tabellen von der Katalogpartition der einzelnen Datenbanken abgerufen.

   Wenn der Befehl DB2START ausgeführt wurde, wird der neue Server gestoppt.

2. Stoppen Sie den Datenbankmanager in allen Datenbankpartitionen durch Ausführen des Befehls DB2STOP.

   Wenn Sie alle Datenbankpartitionen in einem System stoppen, wird die Knotenkonfigurationsdatei aktualisiert und enthält nun die neue Datenbankpartition. Die Knotenkonfigurationsdatei wird erst dann mit den neuen Serverinformationen aktualisiert, wenn DB2STOP ausgeführt wird. Dadurch wird sichergestellt, dass der Befehl ADD DBPARTITIONNUM, der aufgerufen wird, wenn Sie den Parameter ADDNODE im Befehl DB2START angeben, in der richtigen Datenbankpartition ausgeführt wird. Wenn das Dienstprogramm beendet ist, wird die neue Serverpartition gestoppt.
3. Starten Sie den Datenbankmanager durch Ausführen des Befehls DB2START.

   Nun wird die hinzugefügte Datenbankpartition gemeinsam mit dem restlichen System gestartet.

   Wenn alle Datenbankpartitionen des Systems aktiv sind, können Sie Aktionen durchführen, die das gesamte System betreffen, wie zum Beispiel das Erstellen oder Löschen einer Datenbank.

   Anmerkung: Möglicherweise müssen Sie den Befehl DB2START zweimal ausführen, damit alle Datenbankpartitionsserver auf die neue Datei `db2nodes.cfg` zugreifen.
4. Optional: Führen Sie ein Backup aller Datenbanken in der neuen Datenbankpartition durch.
5. Optional: Verteilen Sie Daten auf die neue Datenbankpartition um.

Hinzufügen einer Datenbankpartition zu einem gestoppten Datenbanksystem (Windows)

Sie können einem partitionierten Datenbanksystem neue Datenbankpartitionen hinzufügen, während es gestoppt ist. Die neu hinzugefügte Datenbankpartition wird erst nach einem Neustart des Datenbankmanagers für alle Datenbanken verfügbar.

Sie müssen den neuen Server installieren, bevor Sie eine Datenbankpartition auf diesem Server erstellen können.

Gehen Sie wie folgt vor, um unter Verwendung der Befehlszeile einem gestoppten partitionierten Datenbankserver eine Datenbankpartition hinzuzufügen:

1. Setzen Sie den Befehl DB2STOP ab, um alle Datenbankpartitionen zu stoppen.
2. Führen Sie den Befehl ADD DBPARTITIONNUM auf dem neuen Server aus.

   Eine Datenbankpartition wird lokal für jede Datenbank erstellt, die bereits im System vorhanden ist. Die Datenbankparameter für die neuen Datenbankpartitionen werden auf die Standardwerte gesetzt, und jede Datenbankpartition bleibt leer, bis Sie Daten dorthin versetzen. Aktualisieren Sie die Datenbankkonfigurationsparameter, sodass sie mit denen in den anderen Datenbankpartitionen übereinstimmen.
3. Führen Sie den Befehl DB2START aus, um das Datenbanksystem zu starten. Beachten Sie, dass die Knotenkonfigurationsdatei ('db2nodes.cfg') bereits bei der Installation des neuen Servers aktualisiert wurde, um den neuen Server aufzunehmen.
4. Aktualisieren Sie die Konfigurationsdatei in der neuen Datenbankpartition wie folgt:
   a. Führen Sie den Befehl DB2START in jeder vorhandenen Datenbankpartition aus.

Geben Sie die neuen Datenbankpartitionswerte für die Parameter DBPARTITIONNUM, ADDDBPARTITIONNUM, HOSTNAME, PORT und NETNAME sowie für die Parameter COMPUTER, USER und PASSWORD an.

Sie können auch die Quelle für die Definitionen von Tabellenbereichscontainern für temporäre Tabellen angeben, die mit den Datenbanken erstellt werden müssen. Wenn Sie keine Tabellenbereichsinformationen angeben, werden die Definitionen für Tabellenbereichscontainer für temporäre Tabellen von der Katalogpartition der einzelnen Datenbanken abgerufen.

Wenn der Befehl DB2START ausgeführt wurde, wird der neue Server gestoppt.

b. Stoppen Sie den gesamten Datenbankmanager durch Ausführen des Befehls DB2STOP.

   Wenn Sie alle Datenbankpartitionen in einem System stoppen, wird die Knotenkonfigurationsdatei aktualisiert und enthält nun die neue Datenbankpartition. Die Knotenkonfigurationsdatei wird erst dann mit den neuen Serverinformationen aktualisiert, wenn DB2STOP ausgeführt wird. Dadurch wird sichergestellt, dass der Befehl ADD DBPARTITIONNUM, der aufgerufen wird, wenn Sie den Parameter ADDDBPARTITIONNUM im Befehl DB2START angeben, in der richtigen Datenbankpartition ausgeführt wird. Wenn das Dienstprogramm beendet ist, wird die neue Serverpartition gestoppt.

5. Starten Sie den Datenbankmanager durch Ausführen des Befehls DB2START.

   Nun wird die hinzugefügte Datenbankpartition mit dem restlichen System gestartet.

   Wenn alle Datenbankpartitionen des Systems aktiv sind, können Sie Aktionen durchführen, die das gesamte System betreffen, wie zum Beispiel das Erstellen oder Löschen einer Datenbank.

   Anmerkung: Möglicherweise müssen Sie den Befehl DB2START zweimal ausführen, damit alle Datenbankpartitionsserver auf die neue Datei `db2nodes.cfg` zugreifen.

6. Optional: Führen Sie ein Backup aller Datenbanken in der neuen Datenbankpartition durch.
7. Optional: Verteilen Sie Daten auf die neue Datenbankpartition um.

Hinzufügen einer Datenbankpartition zu einem gestoppten Datenbanksystem (UNIX)

Sie können einem partitionierten Datenbanksystem neue Datenbankpartitionen hinzufügen, während es gestoppt ist. Die neu hinzugefügte Datenbankpartition wird erst nach einem Neustart des Datenbankmanagers für alle Datenbanken verfügbar.

Sie müssen den neuen Server installieren. falls er nicht vorhanden ist. Dies umfasst folgende Aufgaben:

- Einrichten des Zugriffs auf ausführbare Dateien (durch Anhängen gemeinsamer Dateisysteme oder lokale Kopien)
- Synchronisieren von Betriebssystemdateien mit denen auf vorhandenen Prozessoren
- Sicherstellen des Zugriffs auf das Verzeichnis `sqllib` als gemeinsames Dateisystem
- Sicherstellen der richtigen Werte für die relevanten Betriebssystemparameter (z. B. die maximale Anzahl von Prozessen)

Sie müssen auch den Hostnamen auf dem Namensserver oder in der Datei 'hosts' im Verzeichnis 'etc' in allen Datenbankpartitionen registrieren.

Gehen Sie wie folgt vor, um unter Verwendung der Befehlszeile einem gestoppten partitionierten Datenbankserver eine Datenbankpartition hinzuzufügen:

1. Setzen Sie den Befehl DB2STOP ab, um alle Datenbankpartitionen zu stoppen.
2. Führen Sie den Befehl ADD DBPARTITIONNUM auf dem neuen Server aus.

   Eine Datenbankpartition wird lokal für jede Datenbank erstellt, die bereits im System vorhanden ist. Die Datenbankparameter für die neuen Datenbankpartitionen werden auf die Standardwerte gesetzt, und jede Datenbankpartition bleibt leer, bis Sie Daten dorthin versetzen. Aktualisieren Sie die Datenbankkonfigurationsparameter, sodass sie mit denen in den anderen Datenbankpartitionen übereinstimmen.
3. Führen Sie den Befehl DB2START aus, um das Datenbanksystem zu starten. Beachten Sie, dass die Knotenkonfigurationsdatei ('db2nodes.cfg') bereits bei der Installation des neuen Servers aktualisiert wurde, um den neuen Server aufzunehmen.
4. Aktualisieren Sie die Konfigurationsdatei in der neuen Datenbankpartition wie folgt:
   a. Führen Sie den Befehl DB2START in jeder vorhandenen Datenbankpartition aus.

      Geben Sie die neuen Datenbankpartitionswerte für die Parameter DBPARTITIONNUM, ADDDBPARTITIONNUM, HOSTNAME, PORT und NETNAME sowie für die Parameter COMPUTER, USER und PASSWORD an.

      Sie können auch die Quelle für die Definitionen von Tabellenbereichscontainern für temporäre Tabellen angeben, die mit den Datenbanken erstellt werden müssen. Wenn Sie keine Tabellenbereichsinformationen angeben, werden die Definitionen für Tabellenbereichscontainer für temporäre Tabellen aus der Katalogpartition der einzelnen Datenbanken abgerufen.

      Wenn der Befehl DB2START ausgeführt wurde, wird der neue Server gestoppt.
   b. Stoppen Sie den gesamten Datenbankmanager durch Ausführen des Befehls DB2STOP.

      Wenn Sie alle Datenbankpartitionen in einem System stoppen, wird die Knotenkonfigurationsdatei aktualisiert und enthält nun die neue Datenbankpartition. Die Knotenkonfigurationsdatei wird erst dann mit den neuen Serverinformationen aktualisiert, wenn DB2STOP ausgeführt wird. Dadurch wird sichergestellt, dass der Befehl ADD DBPARTITIONNUM, der aufgerufen wird, wenn Sie den Parameter ADDDBPARTITIONNUM im Befehl DB2START angeben, in der richtigen Datenbankpartition ausgeführt wird. Wenn das Dienstprogramm beendet ist, wird die neue Serverpartition gestoppt.
5. Starten Sie den Datenbankmanager durch Ausführen des Befehls DB2START.

   Nun wird die hinzugefügte Datenbankpartition mit dem restlichen System gestartet.

   Wenn alle Datenbankpartitionen des Systems aktiv sind, können Sie Aktionen durchführen, die das gesamte System betreffen, wie zum Beispiel das Erstellen oder Löschen einer Datenbank.

   Anmerkung: Möglicherweise müssen Sie den Befehl DB2START zweimal ausführen, damit alle Datenbankpartitionsserver auf die neue Datei `db2nodes.cfg` zugreifen.

6. Optional: Führen Sie ein Backup aller Datenbanken in der neuen Datenbankpartition durch.
7. Optional: Verteilen Sie Daten auf die neue Datenbankpartition um.

Sie können die Konfigurationsdatei auch wie folgt manuell aktualisieren:

1. Editieren Sie die Datei 'db2nodes.cfg', und fügen Sie die neue Datenbankpartition ein.
2. Setzen Sie den folgenden Befehl ab, um die neue Datenbankpartition zu starten:
   `DB2START DBPARTITIONNUM partitionsnummer`

   Geben Sie als Wert für *partitionsnummer* die Nummer an, die Sie der neuen Datenbankpartition zuordnen.
3. Wenn der neue Server eine logische Partition sein soll (d. h. nicht die Datenbankpartition 0), verwenden Sie den Befehl DB2SET, um die Registrierdatenbankvariable DBPARTITIONNUM zu aktualisieren. Geben Sie die Nummer der Datenbankpartition an, die Sie hinzufügen.
4. Führen Sie den Befehl ADD NODE in der neuen Datenbankpartition aus.

   Dieser Befehl erstellt lokal eine Datenbankpartition für jede Datenbank, die bereits im System vorhanden ist. Die Datenbankparameter für die neuen Datenbankpartitionen werden auf die Standardwerte gesetzt, und jede Datenbankpartition bleibt leer, bis Sie Daten dorthin versetzen. Aktualisieren Sie die Datenbankkonfigurationsparameter, sodass sie mit denen in den anderen Datenbankpartitionen übereinstimmen.
5. Wenn der Befehl ADD DBPARTITIONNUM ausgeführt wurde, setzen Sie den Befehl DB2START ab, um die anderen Datenbankpartitionen des Systems zu starten.

   Führen Sie keine Aktionen aus, die das gesamte System betreffen, wie zum Beispiel das Erstellen oder Löschen einer Datenbank, bis alle Datenbankpartitionen erfolgreich gestartet wurden.

Fehlerbehebung beim Hinzufügen von Datenbankpartitionen

Das Hinzufügen von Datenbankpartitionen schlägt nicht aufgrund nicht vorhandener Pufferpools fehl, weil der Datenbankmanager Systempufferpools erstellt, um eine automatische Standardunterstützung für alle Pufferpoolseitengrößen bereitzustellen. Wenn allerdings einer dieser Systempufferpools verwendet wird, kann die Leistung erheblich beeinträchtigt werden, da die Systempufferpools sehr klein sind. Wenn ein Systempufferpool verwendet wird, wird eine Nachricht in das Benachrichtigungsprotokoll für die Systemverwaltung geschrieben.

Systempufferpools werden in Szenarios, in denen Datenbankpartitionen hinzugefügt werden, unter den folgenden Umständen verwendet:

- Sie fügen Datenbankpartitionen einer Umgebung mit partitionierten Datenbanken hinzu, die mindestens einen Tabellenbereich für temporäre Systemtabellen mit einer anderen Seitengröße als die Standardgröße von 4 KB besitzt. Wenn eine Datenbankpartition erstellt wird, ist nur der Pufferpool IBMDEFAULTDP vorhanden und dieser Pufferpool hat eine Seitengröße von 4 KB.

  Betrachten Sie die folgenden Beispiele:

  1. Sie verwenden den Befehl db2start, um der aktuellen Mehrpartitionsdatenbank eine Datenbankpartition hinzuzufügen:

     ```
     DB2START DBPARTITIONNUM 2 ADD DBPARTITIONNUM HOSTNAME newhost PORT 2
     ```
  2. Sie verwenden den Befehl ADD DBPARTITIONNUM, nachdem Sie die Datei `db2nodes.cfg` manuell mit der neuen Datenbankpartitionsbeschreibung aktualisiert haben.

Eine Möglichkeit zur Vermeidung dieser Probleme ist die Angabe der Klausel WITHOUT TABLESPACES im Befehl ADD NODE oder db2start. Anschließend müssen Sie die Pufferpools mit der Anweisung CREATE BUFFERPOOL erstellen und dem Pufferpool die Tabellenbereiche für temporäre Systemtabellen mit der Anweisung ALTER TABLESPACE zuordnen.

- Sie fügen Datenbankpartitionen einer vorhandenen Datenbankpartitionsgruppe hinzu, die mindestens einen Tabellenbereich mit einer anderen Seitengröße als der Standardseitengröße von 4 KB hat. Dies geschieht, weil die in der neuen Datenbankpartition erstellten Pufferpools mit einer vom Standard abweichenden Seitengröße für die Tabellenbereiche nicht aktiviert wurden.

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

  Betrachten Sie das folgende Beispiel:

  – Sie verwenden die Anweisung ALTER DATABASE PARTITION GROUP, um einer Datenbankpartitionsgruppe wie folgt eine Datenbankpartition hinzuzufügen:

    ```
    DB2START
       CONNECT TO mpp1
    ALTER DATABASE PARTITION GROUP ng1 ADD NODE (2)
    ```

    Eine Möglichkeit, dieses Problems zu vermeiden, besteht darin, Pufferpools für jede Seitengröße zu erstellen und die Verbindung zur Datenbank wieder herzustellen, bevor Sie die folgende Anweisung ALTER DATABASE PARTITION GROUP absetzen:

    ```
    DB2START
       CONNECT TO mpp1
    CREATE BUFFERPOOL bp1 SIZE 1000 PAGESIZE 8192
    CONNECT RESET
    CONNECT TO mpp1
    ALTER DATABASE PARTITION GROUP ng1 ADD NODE (2)
    ```

Anmerkung: Wenn die Datenbankpartitionsgruppe Tabellenbereiche mit der Standardseitengröße hat, wird die Nachricht SQL1759W zurückgegeben.

Löschen von Datenbankpartitionen

Sie können eine Datenbankpartition löschen, die von keiner Datenbank genutzt wird, und den Computer für andere Zwecke freigeben.

Vergewissern Sie sich, dass die Datenbankpartition nicht in Gebrauch ist, indem Sie den Befehl DROP DBPARTITIONNUM VERIFY oder die API sqledrpn ausführen.

- Wenn Sie die Nachricht SQL6034W (Die Datenbankpartition wird von keiner Datenbank verwendet) empfangen, können Sie die Datenbankpartition löschen.
- Wenn Sie die Nachricht SQL6035W (Datenbankpartition wird von Datenbank verwendet) empfangen, verwenden Sie den Befehl REDISTRIBUTE NODEGROUP, um die Daten von der Datenbankpartition, die Sie löschen wollen, auf andere Datenbankpartitionen vom Aliasnamen der Datenbank umzuverteilen.

Stellen Sie außerdem sicher, dass alle Transaktionen, die von der betreffenden Datenbankpartition koordiniert wurden, erfolgreich mit COMMIT festgeschrieben oder mit ROLLBACK rückgängig gemacht wurden. Dazu ist möglicherweise eine Recovery nach einem Systemabsturz auf anderen Servern erforderlich. Wenn Sie zum Beispiel die Koordinatorpartition löschen und vor dem Löschen der Koordinatorpartition eine andere an einer Transaktion beteiligte Datenbankpartition abge-

stürzt war, ist die abgestürzte Datenbankpartition nicht in der Lage, die Koordinatorpartition nach dem Ergebnis unbestätigter Transaktionen abzufragen.

Zum Löschen einer Datenbankpartition über die Befehlszeile geben Sie den Befehl DB2STOP mit dem Parameter DROP NODENUM ein. Nach erfolgreicher Ausführung des Befehls wird das System gestoppt. Starten Sie anschließend den Datenbankmanager mit dem Befehl DB2START.

Auflisten der Datenbankpartitionsserver einer Instanz

Unter Windows können Sie mit dem Befehl db2nlist eine Liste der Datenbankpartitionsserver abrufen, die an einer Instanz beteiligt sind.

Der Befehl ist wie folgt anzugeben:

```
db2nlist
```

Wenn Sie den Befehl wie angegeben verwenden, wird die aktuelle Instanz (wie in der Umgebungsvariablen DB2INSTANCE festgelegt) als Standardinstanz verwendet. Sie können in dem Befehl auch eine bestimmte Instanz angeben:

```
db2nlist /i:instName
```

Dabei ist `instName` der Name der gewünschten Instanz.

Mit dem folgenden Befehl können Sie außerdem (optional) den Status jedes Datenbankpartitionsservers abfragen:

```
db2nlist /s
```

Jeder Datenbankpartitionsserver kann einen der folgenden Statuswerte aufweisen: Starten, Ausführen, Stoppen, Gestoppt.

Hinzufügen von Datenbankpartitionsservern zu einer Instanz (Windows)

Unter Windows können Sie mit dem Befehl db2ncrt einer Instanz einen Datenbankpartitionsserver hinzufügen.

Anmerkung: Wenn die Instanz bereits Datenbanken enthält, darf der Befehl db2ncrt nicht verwendet werden. Verwenden Sie in diesem Fall stattdessen den Befehl db2start addnode. Dadurch wird sichergestellt, dass die Datenbank dem neuen Datenbankpartitionsserver korrekt hinzugefügt wird. **Editieren Sie AUF KEINEN FALL** die Datei `db2nodes.cfg`! Dies kann zu Inkonsistenzen in der Umgebung mit partitionierten Datenbanken führen.

Der Befehl hat folgende erforderliche Parameter:

```
db2ncrt /n:knotennummer
        /u:benutzername,kennwort
        /p:logischer_port
```

/n: Die eindeutige Datenbankpartitionsnummer zum Identifizieren des Datenbankpartitionsservers. Die Nummer kann aus den Werten 1 bis 999 (in aufsteigender Reihenfolge) bestehen.

/u: Der Name des Anmeldekontos und das Kennwort des DB2-Service.

/p:logischer_port
Die logische Portnummer, die für den Datenbankpartitionsserver verwendet wird, wenn der logische Port ungleich null (0) ist. Wird hier kein Wert angegeben, wird die logische Portnummer '0' zugeordnet.

Die logische Portnummer ist nur dann optional, wenn die erste Datenbankpartition auf einem Computer erstellt wird. Beim Erstellen einer logischen Datenbankpartition müssen Sie diesen Parameter angeben und eine logische Portnummer auswählen, die noch nicht verwendet wird. Es gelten verschiedene Einschränkungen:

- Auf jedem Computer muss ein Datenbankpartitionsserver mit dem logischen Port '0' definiert sein.
- Die Portnummer darf den für die FCM-Kommunikation reservierten Portbereich in der Servicedatei im Verzeichnis %SystemRoot%\system32\drivers\etc nicht überschreiten. Wenn Sie zum Beispiel einen Bereich von vier Ports für die aktuelle Instanz reservieren, ist die höchste zulässige Portnummer 3 (Ports 1, 2 und 3; Port 0 ist der logischen Standarddatenbankpartition zugeordnet). Der Portbereich wird definiert, wenn der Befehl db2icrt mit dem Parameter '/r:basisport,endport' angegeben wird.

Außerdem gibt es folgende optionale Parameter:

/g:netzwerkname
Gibt den Netzwerknamen für den Datenbankpartitionsserver an. Wird dieser Parameter nicht angegeben, verwendet DB2 die erste IP-Adresse, die auf dem System festgestellt wird.

Verwenden Sie diesen Parameter, wenn auf einem Computer mehrere IP-Adressen definiert sind und wenn Sie eine bestimmte IP-Adresse für den Datenbankpartitionsserver angeben möchten. Sie können für den Parameter 'netzwerkname' den Netzwerknamen oder die IP-Adresse der gewünschten Einheit angeben.

/h:hostname
Gibt den TCP/IP-Hostnamen an, der von FCM für die interne Kommunikation verwendet wird, wenn der Hostname nicht der lokale Hostname ist. Dieser Parameter ist erforderlich, wenn Sie den Datenbankpartitionsserver auf einem fernen Computer hinzufügen möchten.

/i:instanzname
Gibt den Instanznamen an. Der Standardwert ist die aktuelle Instanz.

/m:computername
Der Computername der Windows-Workstation, auf der sich die Datenbankpartition befindet (der Standardname ist der Computername des lokalen Computers).

/o:instanzeignercomputer
Der Name des Computers, der Instanzeigner ist (der Standardname ist der Name des lokalen Computers). Dieser Parameter ist erforderlich, wenn der Befehl db2ncrt auf einem Computer aufgerufen wird, der nicht der Instanzeignercomputer ist.

Wenn Sie z. B. der Instanz TESTMPP einen neuen Datenbankpartitionsserver auf dem Instanzeignercomputer MYMACHIN hinzufügen möchten (um mehrere logische Datenbankpartitionen auszuführen) und diese neue Datenbankpartition als Datenbankpartition 2 mit dem logischen Port 1 definieren möchten, müssen Sie Folgendes eingeben:

```
db2ncrt /n:2 /p:1 /u:meine_id,mein_kwort /i:TESTMPP
   /M:TEST /o:MYMACHIN
```

Hinzufügen von Datenbankpartitionen zu einer Instanz mithilfe des Assistenten 'Partitionen hinzufügen'

Verwenden Sie den Assistenten **Partitionen hinzufügen**, um eine Partition zu erstellen, und diese einer oder mehreren Datenbankpartitionsgruppen hinzuzufügen. Zunächst fügen Sie Ihrer Instanz eine neue Partition hinzu und ordnen die Partition einer oder mehreren Datenbankpartitionsgruppen zu. Anschließend wählen Sie erweiterte Optionen aus.

Zum Arbeiten mit Datenbankpartitionsgruppen müssen Sie über die Berechtigung SYSADM oder DBADM verfügen.

1. Öffnen Sie den Assistenten **Partitionen hinzufügen**:
   a. Erweitern Sie in der Steuerzentrale die Objektbaumstruktur, bis das Instanzobjekt angezeigt wird, mit dem Sie arbeiten wollen. Klicken Sie das Objekt mit der rechten Maustaste an, und klicken Sie **Partitionen hinzufügen** im Kontextmenü an. Das Launchpad **Partitionen hinzufügen** wird geöffnet.
   b. Klicken Sie den Knopf **Partitionen hinzufügen** an. Der Assistent **Partitionen hinzufügen** wird geöffnet.
2. Füllen Sie alle erforderlichen Seiten des Assistenten aus. Klicken Sie die Programmverbindung (Link) zur Assistentenübersicht auf der ersten Seite an, um weitere Informationen zu erhalten. Der Druckknopf **Fertig stellen** ist verfügbar, wenn Sie ausreichend Informationen angegeben haben, sodass der Assistent die Partition hinzufügen kann.

Ändern von Datenbankpartitionen (Windows)

Unter Windows können Datenbankpartitionen mit dem Befehl db2nchg geändert werden.

- Versetzen der Datenbankpartition von einem Computer auf einen anderen
- Ändern des TCP/IP-Hostnamens des Computers

  Wenn Sie mit mehreren Netzwerkadaptern arbeiten wollen, müssen Sie mit diesem Befehl die TCP/IP-Adresse für das Feld 'netzwerkname' in der Datei `db2nodes.cfg` angeben.
- Verwenden einer anderen logischen Portnummer
- Verwenden eines anderen Namens für den Datenbankpartitionsserver

Der Befehl hat den folgenden erforderlichen Parameter:

```
db2nchg /n:knotennummer
```

Der Parameter /n: gibt die Nummer der Konfiguration des Datenbankpartitionsservers an, die geändert werden soll. Dieser Parameter ist obligatorisch.

Der Befehl unterstützt die folgenden optionalen Parameter:

/i:instanzname
Gibt die Instanz an, in der sich der aktuelle Datenbankpartitionsserver befindet. Wird dieser Parameter nicht angegeben, wird standardmäßig die aktuelle Instanz verwendet.

/u:benutzername,kennwort
Ändert den Namen des Anmeldekontos und das Kennwort für den DB2-Datenbankservice. Wird dieser Parameter nicht angegeben, bleiben das Anmeldekonto und das Kennwort unverändert.

/p:logischer_port
Ändert den logischen Port für den Datenbankpartitionsserver. Dieser Para-

meter muss angegeben werden, wenn Sie den Datenbankpartitionsserver auf einen anderen Computer versetzen wollen. Wird dieser Parameter nicht angegeben, bleibt die logische Portnummer unverändert.

/h:hostname
Ändert den TCP/IP-Hostnamen, der von FCM für die interne Kommunikation verwendet wird. Wird dieser Parameter nicht angegeben, bleibt der Hostname unverändert.

/m:computername
Versetzt den Datenbankpartitionsserver auf einen anderen Computer. Der Datenbankpartitionsserver kann nur dann versetzt werden, wenn die Instanz keine bereits definierten Datenbanken enthält.

/g:netzwerkname
Ändert den Netzwerknamen des Datenbankpartitionsservers.

Verwenden Sie diesen Parameter, wenn auf einem Computer mehrere IP-Adressen definiert sind und wenn Sie eine spezifische IP-Adresse für den Datenbankpartitionsserver verwenden wollen. Sie können für den Parameter 'netzwerkname' den Netzwerknamen oder die IP-Adresse der gewünschten Einheit angeben.

Wenn zum Beispiel der logische Port für Datenbankpartition 2 geändert werden soll, der an der Instanz TESTMPP beteiligt ist, um den logischen Port 3 zu verwenden, müssen Sie folgenden Befehl eingeben:

```
db2nchg /n:2 /i:TESTMPP /p:3
```

Der DB2-Datenbankmanager ermöglicht Ihnen den Zugriff auf Registrierdatenbankvariablen des DB2-Datenbanksystems auf der Instanzebene auf einem fernen Computer. Gegenwärtig werden Registrierdatenbankvariablen des DB2-Datenbanksystems auf drei verschiedenen Ebenen gespeichert: Computerebene (globale Ebene), Instanzebene und Datenbankpartitionsebene. Die auf Instanzebene (einschließlich der auf Datenbankpartitionsebene) gespeicherten Registrierdatenbankvariablen können mit DB2REMOTEPREG an einen anderen Computer umgeleitet werden. Wenn DB2REMOTEPREG definiert ist, greift der DB2-Datenbankmanager auf die Registrierdatenbankvariablen des DB2-Datenbanksystems auf dem Computer zu, der in DB2REMOTEPREG angegeben ist. Der Befehl db2set sähe folgendermaßen aus:

```
db2set DB2REMOTEPREG=<ferne_workstation>
```

Dabei ist <ferne_workstation> der Name der fernen Workstation.

Anmerkung:

- Bei der Definition dieser Option ist Vorsicht geboten, da alle DB2-Datenbankinstanzprofile und Instanzlisten unter dem angegebenen fernen Computernamen lokalisiert werden.
- Wenn in Ihrer Umgebung Benutzer aus Domänen vorhanden sind, stellen Sie sicher, dass das Anmeldekonto, das dem DB2-Instanzservice zugeordnet ist, ein Domänenkonto ist. Dadurch wird gewährleistet, dass die DB2-Instanz die richtigen Zugriffsrechte zur Aufzählung von Gruppen auf der Domänenebene besitzt.

Diese Einrichtung kann in Kombination mit der Einstellung von DBINSTPROF dazu verwendet werden, ein fernes LAN-Laufwerk auf demselben Computer anzugeben, der auch die Registrierdatenbank enthält.

Hinzufügen von Containern zu SMS-Tabellenbereichen in Datenbankpartitionen

Sie können einen Container nur einem SMS-Tabellenbereich in einer Datenbankpartition hinzufügen, der zurzeit keine Container besitzt.

Geben Sie in der Befehlszeile Folgendes ein, um einem SMS-Tabellenbereich einen Container hinzuzufügen:

```
ALTER TABLESPACE <name>
   ADD ('<pfad>')
   ON DBPARTITIONNUM (<datenbankpartitionsnummer>)
```

Die durch die Nummer angegebene Datenbankpartition und jede Partition in dem Bereich der Datenbankpartitionen muss in der Datenbankpartitionsgruppe vorhanden sein, in der der Tabellenbereich definiert ist. Eine Datenbankpartitionsnummer kann möglicherweise nur explizit oder innerhalb eines Bereichs in genau einer Klausel ON DBPARTITIONNUM für die Anweisung angegeben werden.

Das folgende Beispiel zeigt, wie ein neuer Container der Datenbankpartitionsnummer 3 der Datenbankpartitionsgruppe hinzugefügt wird, die vom Tabellenbereich „plans" auf einem UNIX-basierten Betriebssystem verwendet wird:

```
ALTER TABLESPACE plans
   ADD ('/dev/rhdisk0')
   ON DBPARTITIONNUM (3)
```

Löschen einer Datenbankpartition aus einer Instanz (Windows)

Unter Windows können Sie mit dem Befehl db2ndrop einen Datenbankpartitionsserver aus einer Instanz löschen, in der keine Datenbanken definiert sind. Wenn Sie einen Datenbankpartitionsserver löschen, kann dessen Datenbankpartitionsnummer für einen neuen Datenbankpartitionsserver verwendet werden.

Gehen Sie beim Löschen von Datenbankpartitionsservern aus einer Instanz mit besonderer Sorgfalt vor. Wenn Sie den Instanzeigner-Datenbankpartitionsserver (0) aus der Instanz entfernen, wird die Instanz dadurch unbrauchbar. Wenn Sie die Instanz löschen wollen, verwenden Sie den Befehl db2idrop.

Anmerkung: Wenn die Instanz Datenbanken enthält, darf der Befehl db2ndrop nicht verwendet werden. Verwenden Sie stattdessen den Befehl db2stop drop *knotennummer*. Dadurch wird sichergestellt, dass die Datenbank in der Datenbankpartition korrekt gelöscht wird. **Editieren Sie AUF KEINEN FALL** die Datei 'db2nodes.cfg'! Dies kann zu Inkonsistenzen in der Umgebung mit partitionierten Datenbanken führen.

Wenn Sie eine Datenbankpartition löschen möchten, der durch einen Computer mit mehreren logischen Datenbankpartitionen der logische Port 0 zugeordnet ist, müssen Sie alle anderen Datenbankpartitionen löschen, die den anderen logischen Ports zugeordnet sind, bevor Sie die Datenbankpartition mit dem logischen Port 0 löschen können. Jeder Datenbankpartitionsserver muss über eine Datenbankpartition verfügen, der der logische Port 0 zugeordnet ist.

Der Befehl hat folgende Parameter:

```
db2ndrop /n:knotennummer /i:instanzname
```

/n: Die eindeutige Datenbankpartitionsnummer zum Identifizieren des Datenbankpartitionsservers. Dieser Parameter ist erforderlich. Die Nummer kann

ein Wert aus dem Bereich von 0 bis 999 (in aufsteigender Reihenfolge) sein. Beachten Sie, dass die Datenbankpartition null (0) der Instanzeignercomputer ist.

/i:instanzname
Der Instanzname. Dies ist ein optionaler Parameter. Wird dieser Parameter nicht angegeben, wird standardmäßig die (mit der Registrierdatenbankvariablen DB2INSTANCE definierte) aktuelle Instanz verwendet.

Löschen von Datenbankpartitionen aus der Instanz mit dem Launchpad 'Partitionen löschen'

Das Launchpad **Partitionen löschen** führt Sie durch die Tasks, die erforderlich sind, um Datenbankpartitionen aus Datenbankpartitionsgruppen zu löschen, Daten in Datenbankpartitionsgruppen umzuverteilen und Partitionen aus einer Instanz zu löschen.

Informationen zu dieser Task

Anmerkung: Wenn Sie Datenbankpartitionen aus Datenbankpartitionsgruppen löschen, werden die Datenbankpartitionen nicht sofort gelöscht. Die zu löschenden Datenbankpartitionen werden vielmehr markiert, sodass die Daten bei der Umverteilung der Daten in den Datenbankpartitionsgruppen aus ihnen an andere Positionen versetzt werden können.

Es ist empfehlenswert, vor und nach einer Umverteilung von Daten in Datenbankpartitionsgruppen für alle Datenbanken der Instanz ein Backup durchzuführen. Wenn Sie kein Datenbankbackup durchführen, können Sie Ihre Datenbanken im Falle einer Beschädigung nicht wiederherstellen.

Vorgehensweise

Gehen Sie wie folgt vor, um Partitionen mithilfe des Launchpads **Partitionen löschen** zu löschen:

1. Optional: Führen Sie ein Backup von Daten mithilfe des **Backup-Assistenten** durch.
2. Öffnen Sie das Launchpad **Partitionen löschen**:
   a. Öffnen Sie das Fenster **Partitionen**: Erweitern Sie in der Steuerzentrale die Objektbaumstruktur, bis die Instanz angezeigt wird, für die Sie die Partitionen anzeigen möchten. Klicken Sie die gewünschte Instanz mit der rechten Maustaste an und wählen Sie **Datenbankpartitionsserver öffnen** im Kontextmenü aus. Das Fenster **Partitionen** wird für die ausgewählte Instanz geöffnet.
   b. Wählen Sie die Partitionen aus, die Sie löschen wollen.
   c. Klicken Sie die ausgewählten Partitionen mit der rechten Maustaste an und klicken Sie anschließend **Löschen** im Kontextmenü an. Das Launchpad **Partitionen löschen** wird geöffnet.
3. Löschen Sie die Datenbankpartitionen aus Datenbankpartitionsgruppen:
   a. Bestätigen Sie die Datenbankpartitionen, die Sie aus den Datenbankpartitionsgruppen löschen wollen.

      Anmerkung:
      - Sie müssen erst die Datenbankpartitionen aus den Datenbankpartitionsgruppen löschen, bevor Sie Partitionen aus einer Instanz löschen können.

- Diese Operation löscht die Datenbankpartitionen nicht sofort. Die zu löschenden Datenbankpartitionen werden zunächst nur markiert, damit Sie die darin enthaltenen Daten beim Umverteilen in der Datenbankpartitionsgruppe an andere Positionen versetzen können.

4. .Führen Sie eine Umverteilung von Daten in einer Datenbankpartitionsgruppe durch.
5. Löschen Sie Partitionen aus der Instanz:
   a. Öffnen Sie das **Bestätigungsfenster zum Löschen von Partitionen** aus der Instanz:
      - Öffnen Sie das Fenster **Partitionen** wie oben beschrieben.
      - Wählen Sie die Partitionen aus, die Sie löschen wollen.
      - Klicken Sie die ausgewählten Partitionen mit der rechten Maustaste an und klicken Sie anschließend **Löschen** im Kontextmenü an. Das Launchpad **Partitionen löschen** wird geöffnet.
      - Klicken Sie den Druckknopf **Partition aus Instanz löschen** an. Das Bestätigungsfenster **Partition aus Instanz löschen** wird geöffnet.
   b. Vergewissern Sie sich in der Spalte **Löschen**, ob die Partitionen aus der ausgewählten Instanz gelöscht werden sollen.
   c. Klicken Sie **OK** an, um ein Fenster zu öffnen, in dem Sie terminieren können, wann die Partition gelöscht werden soll.
6. Optional: Führen Sie ein Backup von Daten mithilfe des **Backup-Assistenten** durch.

Szenario: Partitionieren von Daten in einer Datenbank

Dieses Szenario zeigt, wie neue Datenbankpartitionen einer Datenbank hinzugefügt und die Daten auf die neuen Datenbankpartitionen umverteilt werden. Die Veranschaulichung des Befehls REDISTRIBUTE DATABASE PARTITION GROUP bildet einen Teil der Erklärung, wie Daten auf unterschiedliche Tabellengruppen innerhalb einer Datenbankpartitionsgruppe umverteilt werden.

Szenario:

Eine Datenbankpartitionsgruppe DBPG_1 hat 2 Datenbankpartitionen, die als (0, 1) angegeben werden, und eine Datenbankpartitionsgruppendefinition (0, 1).

Die folgenden Tabellenbereiche sind in der Datenbankpartitionsgruppe DBPG_1 definiert:

- Tabellenbereich TS1: Dieser Tabellenbereich enthält die beiden Tabellen T1 und T2.
- Tabellenbereich TS2: Dieser Tabellenbereich enthält die drei definierten Tabellen T3, T4 und T5.

Verteilung von Daten auf die Datenbankpartitionen in DBPG_1:

Mit den folgenden Befehlen werden der Datenbank drei neue Datenbankpartitionen hinzugefügt:

```
DB2START DBPARTITIONNUM 3 ADD DBPARTITIONNUM HOSTNAME <HOSTNAME3>
 PORT <PORT3>;

DB2START DBPARTITIONNUM 4 ADD DBPARTITIONNUM HOSTNAME <HOSTNAME4>
 PORT <PORT4>;

DB2START DBPARTITIONNUM 5 ADD DBPARTITIONNUM HOSTNAME <HOSTNAME5>
 PORT <PORT5>;

DB2STOP;

DB2START;
```

Der folgende Befehl REDISTRIBUTE ändert die Definition der Datenbankpartitionsgruppe DBPG_1 von (0, 1) in (0, 1, 3, 4, 5) und verteilt die Daten gleichzeitig um:

```
DB2 REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD RECOVERABLE
 UNIFORM ADD DBPARTITIONNUM (3 TO 5) STOP AT 2006-03-10-07.00.00.000000;
```

In diesem Szenario wird angenommen, dass der Befehl für die Tabellen T1, T2 und T3 erfolgreich ausgeführt und dann infolge der Angabe der Option STOP AT gestoppt wurde.

Wenn die Datenumverteilung für die Datenbankpartitionsgruppe abgebrochen und die ausgeführten Änderungen an den Tabellen T1, T2 und T3 rückgängig gemacht werden sollen, kann der folgende Befehl ausgeführt werden:

```
DB2 REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD
 RECOVERABLE ABORT;
```

Sie können die Datenumverteilung abbrechen, wenn während der Datenumverteilung ein Fehler oder eine Unterbrechung aufgetreten ist und Sie die Umverteilungsoperation nicht fortsetzen wollen. Für dieses Szenario soll angenommen werden, dass dieser Befehl erfolgreich ausgeführt und die Tabellen T1 und T2 in ihren ursprünglichen Status zurückversetzt wurden.

Zur Umverteilung der Tabellen T5 und T4 mit nur 5000 4-KB-Seiten als Datenpuffer (DATA BUFFER) kann der folgende Befehl ausgeführt werden:

```
DB2 REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD RECOVERABLE
 UNIFORM ADD DBPARTITIONNUM (3 TO 5) TABLE (T5, T4) ONLY  DATA BUFFER 5000;
```

Wenn der Befehl erfolgreich ausgeführt wurde, wurden die Daten in den Tabellen T4 und T5 erfolgreich umverteilt.

Mit dem folgenden Befehl wird die Umverteilung der Daten in den Tabellen T1, T2 und T3 in einer angegebenen Reihenfolge abgeschlossen:

```
DB2 REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD RECOVERABLE
 CONTINUE TABLE (T1) FIRST;
```

Die Angabe TABLE (T1) FIRST veranlasst den Datenbankmanager, die Tabelle T1 zuerst zu verarbeiten, sodass sie vor den anderen Tabellen wieder online (schreibgeschützt) verfügbar wird. Alle anderen Tabellen werden in einer vom Datenbankmanager festgelegten Reihenfolge verarbeitet.

Anmerkung:

- Die Option ADD DBPARTITIONNUM und die Option DROP DBPARTITIONNUM brauchen nicht angegeben zu werden. Stattdessen kann die Anweisung ALTER DATABASE PARTITION GROUP zum Hinzufügen oder Löschen von Datenbankpartitionen verwendet werden, bevor der Befehl REDISTRIBUTE DATABASE PARTITION GROUP ausgeführt wird. In diesem Fall erfolgt kein Hinzufügen oder Löschen von Partitionen durch den Befehl REDISTRIBUTE DATABASE PARTITION GROUP, sondern lediglich eine Umverteilung der Daten entsprechend den angegebenen Optionen.
- Es wird dringend empfohlen, ein Offline-Datenbank-Backup der Datenbank zu erstellen, bevor der Befehl REDISTRIBUTE DATABASE PARTITION GROUP ausgeführt wird. Diese Aktion wird in den oben aufgeführten Beispielen nicht gezeigt.
- Der Befehl REDISTRIBUTE DATABASE PARTITION GROUP kann nicht durch eine aktualisierende Recovery wiederholt werden. Eine vollständi-

ge Erörterung dieses Themas finden Sie im Abschnitt zum Befehl „REDISTRIBUTE DATABASE PARTITION GROUP".

- Nach Abschluss des Befehls REDISTRIBUTE DATABASE PARTITION GROUP befinden sich alle Tabellenbereiche, auf die zugegriffen wurde, im Status 'Backup anstehend'. Solche Tabellenbereiche müssen durch Backup gesichert werden, bevor auf die Tabellen, die in ihnen enthalten sind, zu Schreibzwecken zugegriffen werden kann.

Die oben beschriebenen Schritte veranschaulichen, wie Sie verschiedene Varianten des Befehls REDISTRIBUTE zur Umverteilung von Daten auf Datenbankpartitionen verwenden können.

Absetzen von Befehlen in Umgebungen mit partitionierten Datenbanken

In einer Umgebung mit partitionierten Datenbanken kann es wünschenswert sein, Befehle absetzen zu können, die auf Computern in der Instanz oder auf Datenbankpartitionsservern (Knoten) ausgeführt werden. Sie haben diese Möglichkeit mithilfe des Befehls rah oder des Befehls db2_all. Der Befehl rah ermöglicht Ihnen das Absetzen von Befehlen, die Sie auf Computern in der Instanz ausführen wollen.

Wenn Sie die Befehle auf Datenbankpartitionsservern in der Instanz ausführen wollen, verwenden Sie den Befehl db2_all. Dieser Abschnitt enthält eine Übersicht über diese Befehle. Die folgenden Informationen beziehen sich ausschließlich auf Umgebungen mit partitionierten Datenbanken.

Unter Windows müssen Sie mit einem Benutzerkonto angemeldet sein, das zur Administratorgruppe gehört, um die Befehle rah oder db2_all ausführen zu können.

Auf Linux- und UNIX-Plattformen kann Ihre Anmeldeshell eine Korn-Shell oder eine beliebige andere Shell sein. Allerdings gibt es Unterschiede in der Art, wie verschiedene Shells Befehle behandeln, die Sonderzeichen enthalten.

Darüber hinaus verwendet der Befehl 'rah' auf Linux- und UNIX-Plattformen das Programm für die ferne Shell, das durch die Registrierdatenbankvariable **DB2RSHCMD** angegeben wird. Sie können zwischen den beiden Programmen für die ferne Shell wählen: 'ssh' (für zusätzliche Sicherheit) oder 'rsh' (bzw. 'remsh' für HP-UX). Wenn **DB2RSHCMD** nicht definiert ist, wird 'rsh' (bzw. 'remsh' für HP-UX) verwendet. Das Programm 'ssh' für die ferne Shell wird verwendet, um die Übertragung von unverschlüsselten Kennwörtern in UNIX-Betriebssystemumgebungen zu verhindern. Eine detaillierte Beschreibung des Verfahrens, wie eine Version von 'ssh' zur Verwendung mit einer DB2-DPF-Instanz zu konfigurieren ist, finden Sie in folgendem Artikel: "Configure DB2 Universal Database for UNIX to use OpenSSH".

Informationen zum Funktionsbereich eines Befehls finden Sie im Handbuch *Command Reference*, das Angaben darüber enthält, ob ein Befehl auf einem einzigen Datenbankpartitionsserver oder auf allen ausgeführt wird. Wenn der Befehl nur auf einem Datenbankpartitionsserver ausgeführt wird und Sie ihn auf allen von ihnen ausführen wollen, verwenden Sie dazu den Befehl db2_all. Eine Ausnahme bildet der Befehl db2trc, der auf allen logischen Knoten (Datenbankpartitionsservern) auf einem Computer ausgeführt wird. Wenn Sie den Befehl db2trc auf allen logischen Knoten aller Computer ausführen möchten, verwenden Sie dazu den Befehl rah.

Übersicht über die Befehle rah und db2_all

Die Befehle können sequenziell auf einem Datenbankpartitionsserver nach dem anderen oder parallel ausgeführt werden. Wenn Sie die Befehle auf Linux- und UNIX-Plattformen parallel ausführen, können Sie wählen, ob die Ausgabe an einen Puffer gesendet werden soll, in dem sie zur späteren Anzeige gesammelt wird (Standardmodus), oder ob die Ausgabe auf dem Computer angezeigt werden soll, auf dem der Befehl abgesetzt wird.

Unter Windows wird die Ausgabe bei paralleler Ausführung der Befehle auf dem Computer angezeigt, auf dem der Befehl abgesetzt wurde.

Zur Verwendung des Befehls rah geben Sie Folgendes ein:

```
rah befehl
```

Zur Verwendung des Befehls db2_all geben Sie Folgendes ein:

```
db2_all befehl
```

Hilfe zur Syntax von rah erhalten Sie durch folgende Eingabe:

```
rah "?"
```

Als Befehl kann fast alles angegeben werden, was in eine interaktive Eingabeaufforderung eingegeben werden könnte. Dazu gehören zum Beispiel auch mehrere Befehle, die nacheinander ausgeführt werden. Auf Linux- und UNIX-Plattformen werden mehrere Befehle durch ein Semikolon (;) voneinander getrennt. Unter Windows werden mehrere Befehle durch ein Et-Zeichen (&) voneinander getrennt. Das Trennzeichen darf jedoch nicht nach dem letzten Befehl verwendet werden.

Das folgende Beispiel zeigt die Verwendung des Befehls db2_all, um die Datenbankkonfiguration in allen Datenbankpartitionen zu ändern, die in der Knotenkonfigurationsdatei angegeben sind. Da das Semikolon (;) in die doppelten Anführungszeichen mit eingeschlossen ist, wird die Anforderung parallel ausgeführt:

```
db2_all ";DB2 GET DB CFG FOR sample USING LOGFILSIZ 100"
```

Angeben der Befehle 'rah' und 'db2_all'

Sie können den Befehl 'rah' über die Befehlszeile als Parameter oder als Antwort auf die Eingabeaufforderung angeben, wenn Sie keine Parameter angeben.

Die Angabe über die Eingabeaufforderung ist zu empfehlen, wenn der Befehl die folgenden Sonderzeichen enthält:

```
| & ; < > ( )  { } [ ] nicht ersetztes $
```

Wenn Sie den Befehl als Parameter in der Befehlszeile angeben, müssen Sie ihn in doppelte Anführungszeichen setzen, wenn er eines oder mehrere der aufgeführten Sonderzeichen enthält.

Anmerkung: Auf Linux- und UNIX-Plattformen wird der Befehl Ihrem Befehlsprotokoll so hinzugefügt, als hätten Sie ihn in die Eingabeaufforderung eingegeben.

Alle Sonderzeichen im Befehl können normal eingegeben werden (d. h. ohne Anführungszeichen, außer dem umgekehrten Schrägstrich \). Wenn Sie einen umgekehrten Schrägstrich (\) in Ihrem Befehl benötigen, müssen Sie zwei Schrägstriche eingeben (\\).

Anmerkung: Auf Linux- und UNIX-Plattformen können alle Sonderzeichen im Befehl normal eingegeben werden (d. h. ohne Anführungszeichen, außer den Zeichen ", \, nicht ersetztes $ und einfaches Anführungszeichen (')), wenn Sie keine Korn-Shell verwenden. Wenn Sie eines dieser Zeichen in Ihrem Befehl benötigen, müssen Sie ihnen drei umgekehrte Schrägstriche (\\\) voranstellen. Wenn Sie zum Beispiel einen umgekehrten Schrägstrich (\) in Ihrem Befehl benötigen, müssen Sie vier umgekehrte Schrägstriche (\\\\) eingeben.

Wenn Sie ein doppeltes Anführungszeichen (") in Ihrem Befehl benötigen, müssen Sie ihm drei umgekehrte Schrägstriche voranstellen. Beispiel: \\\".

Anmerkung:

1. Auf Linux- und UNIX-Plattformen dürfen Sie kein einfaches Anführungszeichen (') im Befehl angeben, sofern Ihre Befehlsshell nicht irgendeine Möglichkeit zur Eingabe eines einfachen Anführungszeichens innerhalb einer in einfache Anführungszeichen gesetzten Zeichenfolge bereitstellt.
2. Unter Windows dürfen Sie kein einfaches Anführungszeichen (') im Befehl angeben, sofern Ihr Befehlsfenster nicht irgendeine Möglichkeit zur Eingabe eines einfachen Anführungszeichens innerhalb einer in einfache Anführungszeichen gesetzten Zeichenfolge bereitstellt.

Wenn Sie ein beliebiges Shell-Script der Korn-Shell ausführen, das Logik enthält, die von 'stdin' im Hintergrund gelesen wird, sollten Sie 'stdin' ausdrücklich zu einer Quelle umleiten, an der der Prozess den Lesevorgang ohne Unterbrechungen am Terminal (Nachricht SIGTTIN) durchführen kann. Zum Umleiten von 'stdin' können Sie ein Script im folgenden Format ausführen:

```
shell_script </dev/null &
```

wenn keine Eingabe bereitgestellt werden muss.

Auf ähnliche Weise sollten Sie stets `</dev/null` angeben, wenn 'db2_all' im Hintergrund ausgeführt wird. Beispiel:

```
db2_all ";diesen_befehl_ausführen" </dev/null &
```

Auf diese Weise können Sie 'stdin' umleiten und Unterbrechungen am Terminal vermeiden.

Eine Alternative zu dieser Methode, die Sie anwenden können, wenn Sie die Ausgabe von einem fernen Befehl nicht stört, ist die Verwendung der Option „daemonize" im Präfix für 'db2_all':

```
db2_all ";daemonize_diesen_befehl" &
```

Paralleles Ausführen von Befehlen (Linux, UNIX)

Standardmäßig wird der Befehl sequenziell auf jedem Computer ausgeführt. Jedoch können Sie mithilfe von rshell-Hintergrundprozessen die Befehle parallel ausführen, indem Sie den Befehl mit bestimmten Präfixsequenzen versehen. Wenn rshell im Hintergrund ausgeführt wird, schreibt jeder Befehl die Ausgabe in eine Pufferdatei auf dem jeweiligen fernen Computer.

Anmerkung: Die Informationen dieses Abschnitts beziehen sich ausschließlich auf Linux- und UNIX-Plattformen.

Dieser Prozess ruft die Ausgabe in zwei Teilen ab:

1. Nach Beendigung des fernen Befehls

2. Nach Beendigung der rshell, was später geschehen kann, wenn einige Prozesse andauern

Der Name der Pufferdatei ist standardmäßig `/tmp/$USER/rahout`. Er kann aber durch die Umgebungsvariablen $RAHBUFDIR/$RAHBUFNAME definiert werden.

Wenn Sie angeben, dass die Befehle gleichzeitig ausgeführt werden sollen, schaltet dieses Script standardmäßig einen weiteren Befehl vor den Befehl, der an alle Hosts gesendet wird, um zu überprüfen, ob die Umgebungsvariablen $RAHBUFDIR und $RAHBUFNAME für die Pufferdatei verwendbar sind. Sie erstellt das in $RAHBUFDIR definierte Verzeichnis. Sie können dies unterdrücken, indem Sie eine Umgebungsvariable `RAHCHECKBUF=no` exportieren. Dies dient der Zeitersparnis, wenn Sie bereits wissen, dass das Verzeichnis existiert und verwendbar ist.

Bevor Sie rah verwenden, um einen Befehl gleichzeitig auf mehreren Computern auszuführen, sollten Sie folgende Schritte ausführen:

- Stellen Sie sicher, dass ein Verzeichnis `/tmp/$USER` für Ihre Benutzer-ID auf jedem Computer vorhanden ist. Zur Erstellung eines Verzeichnisses, falls es noch nicht vorhanden ist, führen Sie folgenden Befehl aus:

  ```
  rah ")mkdir /tmp/$USER"
  ```

- Fügen Sie die folgende Zeile in Ihre Datei `.kshrc` (für Korn-Shellsyntax) oder `.profile` ein, und geben Sie sie auch in Ihre aktuelle Sitzung ein:

  ```
  export RAHCHECKBUF=no
  ```

- Stellen Sie sicher, dass jede Computer-ID, auf der Sie den fernen Befehl ausführen, in der zugehörigen Datei `.rhosts` über einen Eintrag für die ID verfügt, unter der der Befehl rah ausgeführt wird, und dass die ID, unter der der Befehl rah ausgeführt wird, in der zugehörigen Datei `.rhosts` über einen Eintrag für die einzelnen Computer-IDs verfügt, auf denen Sie den fernen Befehl ausführen.

Erweiterung des Befehls 'rah' zur Verwendung von Baumstrukturlogik (AIX und Solaris)

Zur Leistungsverbesserung wurde 'rah' erweitert, um eine Baumstrukturlogik für große Systeme zu verwenden. Das heißt, 'rah' überprüft, wie viele Knoten in der Liste enthalten sind, und wenn diese Anzahl einen Schwellenwert überschreitet, erstellt 'rah' eine Teilmenge der Liste und sendet einen rekursiven Aufruf von sich selbst an diese Knoten.

An diesen Knoten wendet das rekursiv aufgerufene 'rah' dieselbe Logik an, bis die Liste so klein ist, dass mit der Standardlogik (nun die Logik "leaf-of-tree") die Befehle an alle Knoten in der Liste versendet werden können. Der Schwellenwert kann mit der Umgebungsvariable RAHTREETHRESH angegeben werden, oder er nimmt standardmäßig den Wert 15 an.

Im Fall eines Systems, bei dem mehrere logische Knoten auf einen physischen Knoten kommen, sendet 'db2_all' den rekursiven Aufruf bevorzugt an bestimmte physische Knoten, die dann 'rah' an die anderen logischen Knoten auf demselben physischen Knoten senden, sodass auch die Übertragungen zwischen den physischen Knoten reduziert werden. (Dies gilt nur für 'db2_all', nicht für 'rah', da 'rah' immer nur an bestimmte physische Knoten sendet.)

Befehle 'rah' und 'db2_all'

Dieser Abschnitt enthält Beschreibungen der Befehle 'rah' und 'db2_all'.

Befehl Beschreibung

rah Führt den Befehl auf allen Computern aus.

db2_all
Führt den Befehl auf allen Datenbankpartitionsservern aus, die Sie angeben.

db2_kill
Stoppt abrupt alle Prozesse, die auf mehreren Datenbankpartitionsservern ausgeführt werden, und bereinigt alle Ressourcen auf allen Datenbankpartitionsservern. Durch diesen Befehl werden Ihre Datenbanken inkonsistent. Führen Sie diesen Befehl *nicht* ohne entsprechende Anweisung durch den IBM Kundendienst aus.

db2_call_stack
Auf Linux- und UNIX-Plattformen werden alle Prozesse, die auf allen Datenbankpartitionsservern ausgeführt werden, veranlasst, eine Ablaufverfolgungsliste für Aufrufe in das Systemprotokoll zu schreiben.

Auf Linux- und UNIX-Plattformen führen diese Befehle rah mit bestimmten impliziten Einstellungen wie zum Beispiel folgenden aus:
- Parallele Ausführung auf allen Computern
- Puffern der Befehlsausgabe in `/tmp/$USER/db2_kill` bzw. `/tmp/$USER/db2_call_stack`

Der Befehl db2_call_stack steht unter Windows *nicht* zur Verfügung. Verwenden Sie stattdessen den Befehl db2pd -stack.

Präfixsequenzen für die Befehle 'rah' und 'db2_all'

Eine Präfixsequenz besteht aus einem oder mehreren Sonderzeichen.

Eine oder mehrere Präfixsequenzen müssen unmittelbar vor den Zeichen des Befehls ohne zwischengeschaltete Leerzeichen eingegeben werden. Wenn Sie mehr als eine Sequenz angeben wollen, können Sie sie in beliebiger Reihenfolge eingeben. Zeichen innerhalb einer Mehrzeichensequenz müssen jedoch in der richtigen Reihenfolge eingegeben werden. Wenn Sie Präfixsequenzen eingeben, müssen Sie den gesamten Befehl, einschließlich der Präfixsequenzen, in doppelte Anführungszeichen setzen, wie in folgenden Beispielen gezeigt:

- Auf Linux- und UNIX-Plattformen:

```
rah "};ps -F pid,ppid,etime,args -u $USER"  db2_all "};ps -F pid,ppid,etime,args -u $USER"
```

- Unter Windows:

```
rah "||db2 get db cfg for sample"  db2_all "||db2 get db cfg for sample"
```

Folgende Präfixsequenzen sind möglich:

Sequenz
Zweck

`|` Führt die Befehle nacheinander im Hintergrund aus.

`|&` Führt die Befehle nacheinander im Hintergrund aus und beendet den Befehl, wenn alle fernen Befehle ausgeführt sind, auch wenn einige Prozesse noch aktiv sind. Diese Aktion kann später stattfinden, wenn zum Beispiel untergeordnete Prozesse (auf Linux- und UNIX-Plattformen) oder Hintergrundprozesse (unter Windows) noch aktiv sind. In diesem Fall startet der Befehl einen getrennten Hintergrundprozess, um sämtliche fernen Ausgaben abzurufen, die nach Beendigung des Befehls generiert wurden, und schreibt diese wieder in den Ursprungscomputer.

Anmerkung: Auf Linux- und UNIX-Plattformen wird durch die Angabe von & die Leistung beeinträchtigt, da mehr rsh-Befehle erforderlich werden.

|| Führt die Befehle parallel im Hintergrund aus.

||& Führt die Befehle parallel im Hintergrund aus und beendet den Befehl, wenn alle fernen Befehle ausgeführt sind, wie oben für die Sequenz |& beschrieben.

Anmerkung: Auf Linux- und UNIX-Plattformen wird durch die Angabe von & die Leistung beeinträchtigt, da mehr rsh-Befehle erforderlich werden.

; Hat die gleiche Funktion wie ||&. Dies ist eine kürzere alternative Form.

Anmerkung: Auf Linux- und UNIX-Plattformen wird durch die Angabe von ; die Leistung im Vergleich zu || herabgesetzt, da mehr rsh-Befehle erforderlich sind.

] Schaltet die Punktdateiausführung des Profils des Benutzers vor die Ausführung des Befehls.

Anmerkung: Nur für Linux- und UNIX-Plattformen verfügbar.

} Schaltet die Punktdateiausführung der in $RAHENV definierten Datei (wahrscheinlich `.kshrc`) vor die Ausführung des Befehls.

Anmerkung: Nur für Linux- und UNIX-Plattformen verfügbar.

]} Schaltet die Punktdateiausführung des Profils des Benutzers gefolgt von der Ausführung der in $RAHENV definierten Datei (wahrscheinlich `.kshrc`) vor die Ausführung des Befehls.

Anmerkung: Nur für Linux- und UNIX-Plattformen verfügbar.

) Unterdrückt die Ausführung des Benutzerprofils und der in $RAHENV definierten Datei.

Anmerkung: Nur für Linux- und UNIX-Plattformen verfügbar.

' Meldet den Befehlsaufruf zurück an den Computer.

< Sendet an alle Computer außer diesen.

<<-nnn<

Sendet an alle außer Datenbankpartitionsserver *nnn* (d. h. alle Datenbankpartitionsserver in der Datei `db2nodes.cfg` außer dem mit der Knotennummer *nnn*. Siehe den ersten Abschnitt nach der letzten Präfixsequenz in dieser Tabelle).

nnn ist die zugehörige ein-, zwei- oder dreistellige Datenbankpartitionsnummer für den Wert *nodenum* in der Datei `db2nodes.cfg`.

<<-nnn< ist nur auf db2_all anwendbar.

<<+nnn<

Sendet nur an Datenbankpartitionsserver *nnn* (der Datenbankpartitionsserver in der Datei `db2nodes.cfg`, dessen Datenbankpartitionsnummer *nnn* lautet. Siehe erster Abschnitt nach der letzten Präfixsequenz in dieser Tabelle).

nnn ist die zugehörige ein-, zwei- oder dreistellige Datenbankpartitionsnummer für den Wert *nodenum* in der Datei `db2nodes.cfg`.

<<+nnn< ist nur auf db2_all anwendbar.

(Leerzeichen)
Führt den fernen Befehl im Hintergrund bei geschlossenen `stdin` (Standardeingabeeinheit), `stdout` (Standardausgabeeinheit) und `stderr` (Standardfehlereinheit) aus. Diese Option ist nur gültig, wenn der Befehl im Hintergrund ausgeführt wird, d. h. nur in einer Präfixsequenz, die auch \ oder ; enthält. Sie ermöglicht dem Befehl eine wesentlich frühere Beendigung (sobald der ferne Befehl eingeleitet wurde). Wenn Sie dieses Präfixzeichen in der rah-Befehlszeile angeben, setzen Sie den Befehl entweder in einfache Anführungszeichen oder setzen Sie den Befehl in doppelte Anführungszeichen und stellen dem Präfixzeichen einen umgekehrten Schrägstrich (\ ) voran. Beispiel:

```
rah '; mydaemon'
```

oder

```
rah ";\ mydaemon"
```

Bei der Ausführung als Hintergrundprozess wartet der rah-Befehl nie auf die Rückgabe irgendeiner Ausgabe.

> Ersetzt die Vorkommen von > durch den Computernamen.

" Ersetzt die Vorkommen von () durch den Computerindex und die Vorkommen von ## durch die Datenbankpartitionsnummer.

- Der Computerindex ist eine Nummer, die einem Computer innerhalb des Datenbanksystems zugeordnet ist. Wenn Sie nicht mit mehreren logischen Partitionen arbeiten, entspricht der Computerindex für einen Computer der Datenbankpartitionsnummer für diesen Computer in der Knotenkonfigurationsdatei. Der Computerindex für einen Computer in einer Umgebung mit mehreren logischen Datenbankpartitionen wird ermittelt, indem mehrere Einträge für die Computer, die mehrere logische Partitionen ausführen, nur einfach gezählt werden. Wenn zum Beispiel MACH1 zwei logische Partitionen ausführt und MACH2 ebenfalls zwei logische Partitionen ausführt, ist die Datenbankpartitionsnummer für MACH3 in der Knotenkonfigurationsdatei 5. Der Computerindex für MACH3 wäre jedoch 3.
  - Unter Windows sollten Sie die Knotenkonfigurationsdatei unter keinen Umständen editieren. Verwenden Sie den Befehl db2nlist, wenn Sie den Computerindex abrufen möchten.
- Wenn " angegeben wird, werden doppelte Vorkommen aus der Liste der Computer nicht eliminiert.

Bei Verwendung der Präfixsequenzen `<<-nnn<` und `<<+nnn<` ist *nnn* eine ein-, zwei- oder dreistellige Datenbankpartitionsnummer, die mit dem Wert von *nodenum* in der Datei `db2nodes.cfg` übereinstimmen muss.

Hinweise

- Präfixsequenzen werden als Teil des Befehls behandelt. Wenn Sie Präfixsequenzen angeben, müssen Sie den gesamten Befehl, einschließlich der Präfixsequenzen, in doppelte Anführungszeichen setzen.

Steuern des Befehls 'rah'

In diesem Abschnitt werden die Umgebungsvariablen aufgeführt, mit denen der Befehl rah gesteuert werden kann.

Tabelle 13. Umgebungsvariablen zur Steuerung des Befehls rah

Name	Bedeutung	Standardwert
$RAHBUFDIR **Anmerkung:** Nur für Linux- und UNIX-Plattformen verfügbar.	Verzeichnis für den Puffer	/tmp/$USER
$RAHBUFNAME **Anmerkung:** Nur für Linux- und UNIX-Plattformen verfügbar.	Dateiname für den Puffer	rahout
$RAHOSTFILE (auf Linux- und UNIX-Plattformen); RAHOSTFILE (unter Windows)	Datei mit der Liste der Hosts	`db2nodes.cfg`
$RAHOSTLIST (auf Linux- und UNIX-Plattformen); RAHOSTLIST (unter Windows)	Liste der Hosts in Form einer Zeichenfolge	extrahiert aus $RAHOSTFILE
$RAHCHECKBUF **Anmerkung:** Nur für Linux- und UNIX-Plattformen verfügbar.	Wenn auf "no", Prüfungen übergehen	Nicht definiert
$RAHSLEEPTIME (auf Linux- und UNIX-Plattformen); RAHSLEEPTIME (unter Windows)	Zeit in Sekunden, die dieses Script auf die erste Ausgabe parallel ausgeführter Befehle wartet	86400 Sekunden für db2_kill, 200 Sekunden für alle anderen
$RAHWAITTIME (auf Linux- und UNIX-Plattformen); RAHWAITTIME (unter Windows)	Unter Windows das Intervall in Sekunden zwischen aufeinander folgenden Prüfungen darauf, ob ferne Jobs noch aktiv sind. Auf Linux- und UNIX-Plattformen das Intervall in Sekunden zwischen aufeinander folgenden Prüfungen darauf, ob ferne Jobs noch aktiv sind und Nachrichten `rah: waiting for pid> ...` zurückgegeben werden. Bei allen Plattformen ist eine positive ganze Zahl anzugeben. Stellen Sie dem Wert eine führende Null voran, um Nachrichten zu unterdrücken. Beispiel: export RAHWAITTIME=045. Es ist nicht notwendig, einen niedrigen Wert anzugeben, da der Befehl rah bei der Feststellung von Jobbeendigungen nicht auf diese Prüfungen angewiesen ist.	45 Sekunden
$RAHENV **Anmerkung:** Nur für Linux- und UNIX-Plattformen verfügbar.	Gibt den Dateinamen an, der auszuführen ist, wenn $RAHDOTFILES=E oder K oder PE oder B ist.	$ENV

Tabelle 13. Umgebungsvariablen zur Steuerung des Befehls rah (Forts.)

Name	Bedeutung	Standardwert
$RAHUSER (auf Linux- und UNIX-Plattformen); RAHUSER (unter Windows)	Auf Linux- und UNIX-Plattformen die Benutzer-ID, unter der der ferne Befehl auszuführen ist. Unter Windows das Anmeldekonto, das DB2 Remote Command Service zugeordnet ist.	$USER

Anmerkung: Auf Linux- und UNIX-Plattformen wird der Wert von $RAHENV von dort, wo rah ausgeführt wird, verwendet, nicht der Wert (falls vorhanden), der von der fernen Shell definiert wurde.

Angeben der .-Dateien, die mit 'rah' ausgeführt werden (Linux und UNIX)

In diesem Abschnitt werden die Punktdateien (.-Dateien) beschrieben, die ausgeführt werden, wenn keine Präfixsequenz angegeben wird.

Anmerkung: Die Informationen dieses Abschnitts beziehen sich ausschließlich auf Linux- und UNIX-Plattformen.

P .profile

E Datei, die in $RAHENV definiert ist (wahrscheinlich `.kshrc`)

K Wie E

PE .profile gefolgt von der Datei, die in $RAHENV definiert ist (wahrscheinlich `.kshrc`)

B Wie PE

N Keine (weder noch)

Anmerkung: Wenn Ihre Anmeldeshell keine Korn-Shell ist, werden alle Punktdateien, die Sie zur Ausführung angeben, in einem Korn-Shellprozess ausgeführt und müssen daher der Korn-Shell-Syntax entsprechen. Wenn Ihre Anmeldeshell zum Beispiel eine C-Shell ist und Sie Ihre `.cshrc`-Umgebung für Befehle einrichten wollen, die von rah ausgeführt werden, müssen Sie eine Korn-Shell-Entsprechung der Datei `INSTHOME/.profile` für Ihre Datei `.cshrc` erstellen und in Ihrer Datei `INSTHOME/.cshrc` Folgendes angeben:

```
setenv RAHDOTFILES P
```

Oder Sie müssen eine Korn-Shell-Entsprechung der Datei `INSTHOME/.kshrc` für Ihre Datei `.cshrc` erstellen und in Ihrer Datei `INSTHOME/.cshrc` Folgendes angeben:

```
setenv RAHDOTFILES E
setenv RAHENV INSTHOME/.kshrc
```

Außerdem ist es wichtig, dass Ihre Datei `.cshrc` die Ausgabe nicht in die Standardausgabe (stdout) schreibt, wenn (wie beim Aufruf durch rsh) kein TTY vorhanden ist. Dies können Sie sicherstellen, indem Sie alle Zeilen, die an 'stdout' schreiben, zum Beispiel folgendermaßen einschließen:

```
if { tty -s } then echo "executed .cshrc";
endif
```

Beheben von Fehlern mit 'rah' (Linux, UNIX)

Dieser Abschnitt enthält Vorschläge zur Behandlung einiger Fehler, die bei der Ausführung von rah möglicherweise auftreten.

Anmerkung: Die Informationen dieses Abschnitts beziehen sich ausschließlich auf Linux- und UNIX-Plattformen.

1. rah blockiert (oder benötigt sehr lange Zeit)

   Dieses Problem kann folgende Ursache haben:

   - rah hat festgestellt, dass Ausgaben gepuffert werden müssen und dass `RAHCHECKBUF=no` nicht exportiert wurde. Daher sendet rah vor der Ausführung Ihres Befehls einen Befehl an alle Computer, um die Existenz des Pufferverzeichnisses zu überprüfen und um es zu erstellen, falls es noch nicht vorhanden ist.
   - Mindestens ein Computer, an den Sie Ihren Befehl senden, antwortet nicht. Der Befehl rsh wird schließlich sein Zeitlimit überschreiten. Das Zeitlimit ist in der Regel jedoch recht lang (in der Regel ungefähr 60 Sekunden).

2. Sie haben Nachrichten wie die folgenden empfangen:

   - Anmeldung nicht korrekt (Login incorrect)
   - Berechtigung verweigert (Permission denied)

   Entweder ist auf einem der Computer die ID, unter der rah ausgeführt wird, nicht korrekt in der zugehörigen Datei `.hosts` definiert oder für die ID, unter der rah ausgeführt wird, ist einer der Computer nicht korrekt in der zugehörigen Datei `.rhosts` definiert. Wenn die Registrierdatenbankvariable DB2RSHCMD für die Verwendung von ssh konfiguriert wurde, sind möglicherweise die ssh-Clients und -Server auf dem jeweiligen Computer nicht korrekt konfiguriert.

   Anmerkung: In bestimmten Fällen benötigen Sie möglicherweise einen höheren Grad an Sicherheit hinsichtlich der unverschlüsselten Übertragung von Kennwörtern zwischen Datenbankpartitionen. Dies ist von dem Programm für die ferne Shell abhängig, das Sie einsetzen. rah verwendet das Programm für die ferne Shell, das in der Registrierdatenbankvariablen DB2RSHCMD angegeben ist. Sie können zwischen den beiden Programmen für die ferne Shell wählen: ssh (für zusätzliche Sicherheit) oder rsh (bzw. remsh für HP-UX). Wenn diese Registrierdatenbankvariable nicht definiert ist, wird rsh (bzw. remsh für HP-UX) verwendet.

3. Obwohl bei der parallelen Ausführung von Befehlen mit fernen Hintergrundshells die Befehle ausgeführt und in der erwarteten abgelaufenen Zeit auf den Hosts abgeschlossen werden, benötigt rah viel Zeit, um dies zu erkennen und die Shelleingabeaufforderung anzuzeigen.

   Für die ID, die rah ausführt, ist einer der Computer nicht korrekt in der Datei `.rhosts` definiert, oder es wurden, falls die Registrierdatenbankvariable DB2RSHCMD für die Verwendung von 'ssh' konfiguriert wurde, die ssh-Clients und -Server auf den jeweiligen Computern möglicherweise nicht korrekt konfiguriert.

4. rah läuft bei Ausführung über die Shell-Befehlszeile einwandfrei. Aber wenn rah fern mithilfe von rsh, zum Beispiel mit folgendem Befehl ausgeführt wird:

   ```
   rsh somewher -l $USER db2_kill
   ```

   wird rah niemals beendet.

   Dies ist normal. Der Befehl rah startet Überwachungsprozesse im Hintergrund, die auch nach der Beendigung weiterhin aktiv sind. Diese Prozesse bleiben in

der Regel bestehen, bis alle Prozesse, die zu dem Befehl gehören, den Sie ausgeführt haben, ihrerseits beendet sind. Im Fall von db2_kill bedeutet dies, dass sie bis zur Beendigung aller Datenbankmanager bestehen bleiben. Sie können die Überwachungsprozesse beenden, indem Sie den Prozess auffinden, dessen Befehl rahwaitfor ist, und den Befehl kill <prozess-id> ausführen. Geben Sie keine Signalnummer an. Verwenden Sie stattdessen den Standardwert (15).

5. Die Ausgabe von rah wird nicht korrekt angezeigt, oder rah meldet fälschlicherweise, dass $RAHBUFNAME nicht existiert, wenn mehrere Befehle von rah unter derselben Benutzer-ID $RAHUSER abgesetzt wurden.

   Dies kann eintreten, weil mehrere gleichzeitig ablaufende Ausführungen von rah versuchen, dieselbe Pufferdatei (z. B. $RAHBUFDIR/$RAHBUFNAME) zu verwenden. Zur Vermeidung dieses Problems verwenden Sie für jeden gleichzeitig ablaufenden Befehl rah einen unterschiedlichen Namen für $RAHBUFNAME. Im Folgenden ist ein ksh-Beispiel aufgeführt:

   ```
   export RAHBUFNAME=rahout
   rah ";$befehl_1" &
   export RAHBUFNAME=rah2out
   rah ";$befehl_2" &
   ```

   Alternativ hierzu können Sie eine Methode verwenden, mit der die Shell automatisch einen eindeutigen Namen auswählt. Beispiel:

   ```
   RAHBUFNAME=rahout.$$ db2_all "....."
   ```

   Unabhängig von der verwendeten Methode müssen Sie sicherstellen, dass die Pufferdateien zu einem bestimmten Zeitpunkt bereinigt werden, wenn der Plattenspeicherplatz begrenzt ist. Der Befehl rah löscht keine Pufferdatei am Ende der Ausführung. Eine vorhandene Pufferdatei wird jedoch beim nächsten Mal gelöscht und wiederverwendet, wenn Sie dieselbe Pufferdatei erneut angeben.

6. Sie haben Folgendes eingegeben:

   ```
   rah '"print from ()'
   ```

   Daraufhin haben Sie die folgende Nachricht empfangen:

   ```
   ksh: syntax error at line 1 : (' unexpected
   (ksh: Syntaxfehler in Zeile 1 : (' unerwartet)
   ```

   Voraussetzungen für die Ersetzung von () und ## sind:

   - Es ist db2_all und nicht rah zu verwenden.
   - Stellen Sie sicher, dass eine RAHOSTFILE-Datei verwendet wird. Dies geschieht entweder durch Exportieren der Umgebungsvariablen RAHOSTFILE oder durch die sich standardmäßig ergebene Verwendung der Datei `/sqllib/db2nodes.cfg`. Ohne diese Vorbedingungen lässt rah die Zeichenfolgen () und ## unverändert. Sie empfangen einen Fehler, weil der Befehl print from () ungültig ist.

   Zur Verbesserung der Leistung bei der parallelen Ausführung der Befehle sollten Sie `|` anstelle von `|&` und `||` anstelle von `||&` oder ; verwenden, es sei denn, die von & bereitgestellte Funktion wird unbedingt benötigt. Die Angabe von & erfordert eine größere Anzahl ferner Shellbefehle und führt daher zu einer Beeinträchtigung der Leistung.

Überwachen von rah-Prozessen (Linux, UNIX)

Während einige ferne Befehle immer noch aktiv sind oder die gepufferte Ausgabe noch gesammelt wird, überwachen von `rah` gestartete Prozesse die Aktivitäten, um Nachrichten an das Terminal zu schreiben und anzuzeigen, welche Befehle nicht ausgeführt wurden, sowie um die gepufferte Ausgabe abzurufen.

Anmerkung: Die Informationen dieses Abschnitts beziehen sich ausschließlich auf Linux- und UNIX-Plattformen.

Die Informationsnachrichten werden in einem Intervall geschrieben, das durch die Umgebungsvariable RAHWAITTIME gesteuert wird. Im Hilfetext finden Sie Einzelheiten dazu, wie sie definiert wird. Alle Informationsnachrichten können durch Exportieren von RAHWAITTIME=0 vollständig unterdrückt werden.

Der primäre Überwachungsprozess ist ein Befehl, dessen Befehlsname (wie vom Befehl 'ps' angezeigt) 'rahwaitfor' lautet. Die erste Informationsnachricht teilt Ihnen die Prozess-ID (pid) dieses Prozesses mit. Alle anderen Überwachungsprozesse erscheinen als ksh-Befehle, die das rah-Script (bzw. den Namen der symbolischen Verbindung) ausführen. Falls erwünscht, können Sie alle Überwachungsprozesse durch folgenden Befehl stoppen:

```
kill <pid>
```

Dabei ist <pid> die Prozess-ID des primären Überwachungsprozesses. Geben Sie keine Signalnummer an. Belassen Sie den Standardwert auf 15. Dadurch werden die fernen Befehle in keiner Weise beeinflusst, sondern es wird das automatische Anzeigen der gepufferten Ausgabe verhindert. Beachten Sie, dass es während der Dauer einer einzigen rah-Ausführung zwei oder mehr verschiedene Gruppen von Überwachungsprozessen geben kann, die zu verschiedenen Zeiten aktiv sind. Wenn Sie zu einem Zeitpunkt die aktuelle Gruppe von Überwachungsprozessen stoppen, werden keine weiteren mehr gestartet.

Wenn Ihre reguläre Anmeldeshell keine Korn-Shell (z. B. /bin/ksh) ist, können Sie rah, jedoch gibt es eine Reihe leicht abweichender Regeln für die Eingabe von Befehlen, die die folgenden Sonderzeichen enthalten:

```
"  nicht ersetztes $ '
```

Weitere Informationen erhalten Sie durch die Eingabe von rah "?". Außerdem darf in einer Linux- und UNIX-Umgebung die Anmeldeshell unter der ID, die das rah-Script ausführt, keine Korn-Shell sein, wenn die Anmeldeshell unter der ID, die die fernen Befehle ausführt, keine Korn-Shell ist. (rah entscheidet anhand der lokalen ID, ob die Shell der fernen ID eine Korn-Shell ist.) Die Shell darf keinerlei Substitution oder spezielle Verarbeitung an einer Zeichenfolge durchführen, die in einfache Anführungszeichen gesetzt ist. Die Zeichenfolge muss in der Form belassen werden, in der sie vorliegt.

Einstellen des Standardumgebungsprofils für 'rah' unter Windows

Zum Definieren des Standardumgebungsprofils für den Befehl rah wird eine Datei namens db2rah.env verwendet, die im Instanzverzeichnis zu erstellen ist.

Anmerkung: Die Informationen dieses Abschnitts beziehen sich ausschließlich auf Windows.

Die Datei sollte folgendes Format haben:

```
; Dies ist eine Kommentarzeile
DB2INSTANCE=instanzname
DB2DBDFT=datenbank
; Dateiende
```

Sie können alle Umgebungsvariablen angeben, die Sie zur Initialisierung der Umgebung für den Befehl rah benötigen.

Kapitel 11. Erstellen von Tabellen und anderen zugehörigen Objekten

Tabellen in Umgebungen mit partitionierten Datenbanken

Durch die Erstellung einer Tabelle in mehreren Datenbankpartitionen einer Umgebung mit partitionierten Datenbanken lassen sich Leistungsvorteile erzielen. Die Arbeitslast, die mit dem Abruf von Daten verbunden ist, kann auf die Datenbankpartitionen verteilt werden.

Bevor Sie eine Tabelle erstellen, die physisch geteilt oder verteilt wird, müssen Sie folgende Punkte beachten:

- Tabellenbereiche können sich über mehr als eine Datenbankpartition erstrecken. Die Anzahl der Datenbankpartitionen, über die sie sich erstrecken, hängt von der Anzahl der Datenbankpartitionen in der Datenbankpartitionsgruppe ab.
- Tabellen können in einer Kollokation zusammengefasst werden, indem sie im selben Tabellenbereich gespeichert werden oder in einem anderen Tabellenbereich, der zusammen mit dem ersten Tabellenbereich derselben Datenbankpartitionsgruppe zugeordnet ist.

Die gewünschte Zugehörigkeit einer Tabelle zu mehreren Datenbankpartitionen wird bei der Erstellung der Tabelle angegeben. Bei der Erstellung einer Tabelle in einer partitionierten Datenbankumgebung gibt es eine zusätzliche Option: den *Verteilungsschlüssel*. Ein Verteilungsschlüssel ist ein Schlüssel, der Teil der Definition einer Tabelle ist. Durch ihn wird die Datenbankpartition bestimmt, in der die jeweilige Datenzeile gespeichert wird.

Wenn Sie keinen Verteilungsschlüssel explizit angeben, werden die folgenden Standardwerte verwendet. *Stellen Sie sicher, dass der Standardverteilungsschlüssel geeignet ist.*

- Wenn ein Primärschlüssel in der Anweisung CREATE TABLE angegeben wird, wird die erste Spalte des Primärschlüssels als Verteilungsschlüssel verwendet.
- Wenn es für eine Datenbankpartitionsgruppe mit mehreren Partitionen keinen Primärschlüssel gibt, wird die erste Spalte verwendet, die keine Langfelddaten enthält.
- Wenn die vorhandenen Spalten die Anforderungen für einen Standardverteilungsschlüssel nicht erfüllen, wird die Tabelle ohne Verteilungsschlüssel erstellt (d. h., sie ist nur in Datenbankpartitionsgruppen mit einer Einzelpartition zulässig).

Sie müssen bei der Auswahl eines geeigneten Verteilungsschlüssels mit großer Sorgfalt vorgehen, da *der Schlüssel später nicht mehr geändert werden kann*. Darüber hinaus müssen alle eindeutigen Indizes (und infolgedessen eindeutige Schlüssel bzw. Primärschlüssel) als Obermenge des Verteilungsschlüssels definiert werden. Das heißt, wenn ein Verteilungsschlüssel definiert wird, müssen eindeutige Schlüssel und Primärschlüssel alle die Spalten enthalten, die der Verteilungsschlüssel enthält (sie können mehr Spalten haben).

Die Größe einer Datenbankpartition einer Tabelle ist der kleinere von zwei Werten: entweder die bestimmte Obergrenze, die dem Typ und der Seitengröße des verwendeten Tabellenbereichs zugeordnet ist, oder die Kapazität des verfügbaren Plat-

tenspeicherplatzes. Betrachten Sie zum Beispiel einen großen DMS-Tabellenbereich (Typ LARGE) mit einer Seitengröße von 4 KB. In diesem Fall ist die Größe einer Tabelle der kleinere der folgenden Werte: entweder 2 TB multipliziert mit der Anzahl der Datenbankpartitionen oder die Kapazität des verfügbaren Plattenspeicherplatzes. Die vollständige Liste der Begrenzungen für Seitengrößen des Datenbankmanagers finden Sie über die zugehörigen Links.

Geben Sie in die Befehlszeile Folgendes ein, um eine Tabelle in einer Umgebung mit partitionierten Datenbanken zu erstellen:

```
CREATE TABLE name>
  (<spaltenname> <datentyp> <nullattribut>)
  IN <tabellenbereichsname>
  INDEX IN <indextabellenbereichsname>
  LONG IN <lob-tabellenbereichsname>
  DISTRIBUTE BY HASH (<spaltenname>)
```

Betrachten Sie folgendes Beispiel:

```
CREATE TABLE MIXREC (MIX_CNTL INTEGER NOT NULL,
                     MIX_DESC CHAR(20) NOT NULL,
                     MIX_CHR  CHAR(9) NOT NULL,
                     MIX_INT INTEGER NOT NULL,
                     MIX_INTS SMALLINT NOT NULL,
                     MIX_DEC DECIMAL NOT NULL,
                     MIX_FLT FLOAT NOT NULL,
                     MIX_DATE DATE NOT NULL,
                     MIX_TIME TIME NOT NULL,
                     MIX_TMSTMP TIMESTAMP NOT NULL)
                     IN MIXTS12
                     DISTRIBUTE BY HASH (MIX_INT)
```

In diesem Beispiel heißen der Tabellenbereich MIXTS12 und der Verteilungsschlüssel MIX_INT. Wenn der Verteilungsschlüssel nicht explizit angegeben wird, wird die Spalte MIX_CNTL verwendet. (Wenn kein Primärschlüssel angegeben und kein Verteilungsschlüssel definiert wird, wird die erste Spalte in der Liste, die keine Langfelddaten (LONG) enthält, als Verteilungsschlüsse verwendet.)

Eine Zeile einer Tabelle und sämtliche Informationen zu dieser Zeile werden immer in derselben Datenbankpartition gespeichert.

Verhalten großer Objekte in partitionierten Tabellen

Eine partitionierte Tabelle arbeitet mit einem Datenorganisationsschema, bei dem Tabellendaten auf mehrere Speicherobjekte, die als Datenpartitionen oder Datenbereiche (RANGE) bezeichnet werden, entsprechend den Werten einer oder mehrerer Spalten der Tabelle, die den Tabellenpartitionierungsschlüssel bilden, verteilt werden. Daten aus einer gegebenen Tabelle werden in mehrere Speicherobjekte auf der Basis der in der Klausel PARTITION BY der Anweisung CREATE TABLE angegebenen Spezifikationen partitioniert. Diese Speicherobjekte können sich in verschiedenen Tabellenbereichen, in denselben Tabellenbereichen oder in einer Kombination beider Arten von Tabellenbereichen befinden.

Ein großes Objekt (LOB) für eine partitionierte Tabelle wird standardmäßig in demselben Tabellenbereich wie das entsprechende Datenobjekt gespeichert. Dies gilt für partitionierte Tabellen, die nur einen Tabellenbereich oder mehrere Tabellenbereiche verwenden. Wenn die Daten einer partitionierten Tabelle in mehreren Tabellenbereichen gespeichert werden, werden auch die LOB-Daten in mehreren Tabellenbereichen gespeichert.

Zum Überschreiben dieses Standardverhaltens können Sie die Klausel LONG IN der Anweisung CREATE TABLE verwenden. Sie können eine Liste der Tabellenbereiche für die Tabelle angeben, in denen die langen Daten gespeichert werden sollen. Wenn Sie das Standardverhalten überschreiben, muss der in der Klausel LONG IN angegebene Tabellenbereich ein LOB-Tabellenbereich (Typ LARGE) sein. Wenn Sie angeben, dass lange Daten (LONG) in einem separaten Tabellenbereich für eine oder mehrere Datenpartitionen gespeichert werden sollen, müssen Sie dies für alle Datenpartitionen der Tabelle tun. Das heißt, es ist nicht möglich, lange Daten für einige Datenpartitionen fern zu speichern und für andere Datenpartitionen lokal zu speichern. Unabhängig davon, ob Sie das Standardverhalten verwenden oder mit der Klausel LONG IN das Standardverhalten überschreiben, wird jeweils ein langes Objekt als Entsprechung für jede Datenpartition erstellt. Bei SMS-Tabellenbereichen müssen sich die langen Daten in demselben Tabellenbereich wie das Datenobjekt befinden, zu dem sie gehören. Alle Tabellenbereiche, die zum Speichern von Objekten langer Daten, die den einzelnen Datenpartitionen entsprechen, verwendet werden, müssen die gleiche Seitengröße (PAGESIZE), die gleiche Speicherbereichsgröße (EXTENTSIZE), den gleichen Speichermechanismus (DMS oder SMS) und den gleichen Typ (REGULAR oder LARGE) besitzen. Ferne LOB-Tabellenbereiche müssen vom Typ LARGE sein und können keine SMS-Tabellenbereiche sein.

Mit der folgenden Anweisung CREATE TABLE werden zum Beispiel Objekte für die CLOB-Daten für jede Datenpartition in demselben Tabellenbereich wie die Daten erstellt:

```
CREATE TABLE document(id INT, contents CLOB)
PARTITION BY RANGE(id)
(STARTING FROM 1   ENDING AT 100 IN tbsp1,
 STARTING FROM 101 ENDING AT 200 IN tbsp2,
 STARTING FROM 201 ENDING AT 300 IN tbsp3,
 STARTING FROM 301 ENDING AT 400 IN tbsp4);
```

Sie können die Klausel LONG IN verwenden, um die CLOB-Daten in einem oder mehreren LOB-Tabellenbereichen zu speichern, die sich von denjenigen unterscheiden, in denen die Daten enthalten sind.

```
CREATE TABLE document(id INT, contents CLOB)
PARTITION BY RANGE(id)
(STARTING FROM 1   ENDING AT 100 IN tbsp1 LONG IN large1,
STARTING FROM 101 ENDING AT 200 IN tbsp2 LONG IN large1,
STARTING FROM 201 ENDING AT 300 IN tbsp3 LONG IN large2,
STARTING FROM 301 ENDING AT 400 IN tbsp4 LONG IN large2);
```

Anmerkung: Auf Tabellenebene und pro Datenpartition ist nur eine Klausel LONG IN zulässig.

Erstellen partitionierter Tabellen

Partitionierte Tabellen arbeiten mit einem Datenorganisationsschema, bei dem Tabellendaten auf mehrere Speicherobjekte, die als Datenpartitionen oder Datenbereiche bezeichnet werden, entsprechend den Werten einer oder mehrerer Spalten der Tabelle, die den Tabellenpartitionierungsschlüssel bilden, verteilt werden. Daten aus einer gegebenen Tabelle werden in mehrere Speicherobjekte auf der Basis der in der Klausel PARTITION BY der Anweisung CREATE TABLE angegebenen Spezifikationen partitioniert. Diese Speicherobjekte können sich in verschiedenen Tabellenbereichen, in denselben Tabellenbereichen oder in einer Kombination beider Arten von Tabellenbereichen befinden.

Zum Erstellen einer Tabelle müssen die Zugriffsrechte der Berechtigungs-ID der Anweisung mindestens eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte beinhalten:

- Berechtigung CREATETAB für die Datenbank und das Zugriffsrecht USE für alle von der Tabelle verwendeten Tabellenbereiche sowie eine der folgenden Berechtigungen (bzw. Zugriffsrechte):
  - Berechtigung IMPLICIT_SCHEMA für die Datenbank, wenn der implizite oder explizite Schemaname der Tabelle nicht existiert
  - Zugriffsrecht CREATEIN für das Schema, wenn sich der Schemaname der Tabelle auf ein vorhandenes Schema bezieht
- Berechtigung SYSADM oder DBADM

Sie können eine partitionierte Tabelle mit der Anweisung CREATE TABLE erstellen.

Zum Erstellen einer partitionierten Tabelle über die Befehlszeile geben Sie die Anweisung CREATE TABLE ein:

```
CREATE TABLE <NAME> (<spaltenname>  <datentyp>  <null_attribut>) IN
 <tabellenbereichsliste> PARTITION BY RANGE (<spaltenausdruck>)
 STARTING FROM <konstante> ENDING <konstante> EVERY <konstante>
```

Zum Beispiel erstellt die folgende Anweisung eine Tabelle, in der Zeilen mit a ≥ 1 und a ≤ 20 in PART0 (der ersten Datenpartition), Zeilen mit 21 ≤ a ≤ 40 in PART1 (der zweiten Datenpartition) usw. und schließlich Zeilen mit 81 ≤ a ≤ 100 in PART4 (der letzten Datenpartition) gespeichert werden.

```
CREATE TABLE foo(a INT)
 PARTITION BY RANGE (a) (STARTING FROM (1)
 ENDING AT (100) EVERY (20))
```

Definieren von Bereichen für partitionierte Tabellen

Bei der Erstellung einer partitionierten Tabelle können Sie einen Bereich für jede Datenpartition angeben. Eine partitionierte Tabelle arbeitet mit einem Datenorganisationsschema, bei dem Tabellendaten auf mehrere Datenpartitionen entsprechend den Werten der Spalten des Tabellenpartitionierungsschlüssels der Tabelle verteilt werden.

Daten aus einer gegebenen Tabelle werden in mehrere Speicherobjekte auf der Basis der in der Klausel PARTITION BY der Anweisung CREATE TABLE angegebenen Spezifikationen partitioniert. Ein Bereich wird durch Werte mit STARTING FROM und ENDING AT in der Klausel PARTITION BY angegeben.

Zur vollständigen Definition des Bereichs für eine Datenpartition müssen Sie ausreichende Grenzen spezifizieren. Die folgende Liste enthält Richtlinien, die bei der Definition von Bereichen für eine partitionierte Tabelle berücksichtigt werden sollten:

- Die Klausel STARTING gibt eine untere Grenze für den Datenpartitionsbereich an. Diese Klausel ist obligatorisch für den niedrigsten Datenpartitionsbereich (obwohl Sie die Grenze als MINVALUE definieren können). Der niedrigste Datenpartitionsbereich ist die Datenpartition mit der niedrigsten angegebenen Grenze.
- Die Klausel ENDING (oder VALUES) gibt eine obere Grenze für den Datenpartitionsbereich an. Diese Klausel ist obligatorisch für den höchsten Datenpartitions-

bereich (obwohl Sie die Grenze als MAXVALUE definieren können). Der höchste Datenpartitionsbereich ist die Datenpartition mit der höchsten angegebenen Grenze.

- Wenn Sie für eine Datenpartition keine Klausel ENDING angeben, müssen Sie für die nächst höhere Datenpartition eine Klausel STARTING angeben. Wenn Sie für eine Datenpartition keine Klausel STARTING angeben, müssen Sie analog für die vorherige Datenpartition eine Klausel ENDING angeben.
- MINVALUE gibt einen Wert an, der kleiner als jeder mögliche Wert für den Spaltentyp ist, der verwendet wird. MINVALUE und INCLUSIVE bzw. EXCLUSIVE können nicht zusammen angegeben werden.
- MAXVALUE gibt einen Wert an, der größer als jeder mögliche Wert für den Spaltentyp ist, der verwendet wird. MAXVALUE und INCLUSIVE bzw. EXCLUSIVE können nicht zusammen angegeben werden.
- INCLUSIVE gibt an, dass alle Werte gleich dem angegeben Wert in die Datenpartition, die diese Grenze enthält, mit einzuschließen zu sind.
- EXCLUSIVE gibt an, dass alle Werte gleich dem angegeben Wert in die Datenpartition, die diese Grenze enthält, NICHT mit einzuschließen zu sind.
- Die Klausel NULL der Anweisung CREATE TABLE gibt an, ob Nullwerte bei der Platzierung in einer Datenpartition als hoch oder niedrig einzusortieren sind. Standardmäßig werden Nullwerte hoch einsortiert. Nullwerte in Spalten des Tabellenpartitionierungsschlüssels werden als positiv unendlich behandelt und in einen Bereich eingefügt, der mit MAXVALUE endet. Wenn keine solche Datenpartition definiert ist, werden Nullwerte als außerhalb des gültigen Bereichs betrachtet. Verwenden Sie die Integritätsbedingung NOT NULL, wenn Sie Nullwerte aus den Spalten des Tabellenpartitionierungsschlüssels ausschließen möchten. LAST gibt an, dass Nullwerte als letzte in eine sortierte Liste von Werten einzufügen sind. FIRST gibt an, dass Nullwerte als erste in eine sortierte Liste von Werten einzufügen sind.
- Wenn Sie die Langform der Syntax verwenden, muss für jede Datenpartition mindestens eine Grenze angegeben werden.

Tipp: Vor der Definition von Datenpartitionen für eine Tabelle ist es wichtig, zu verstehen, wie Tabellen von der Tabellenpartitionierung profitieren und welche Faktoren sich auf die Spalten auswirken, die Sie als Partitionierungsspalten auswählen.

Die Bereiche, die für die einzelnen Datenpartitionen angegeben werden, können automatisch oder manuell generiert werden.

Automatische Generierung

Die automatische Generierung ist eine einfache und zeitsparende Methode zur Erstellung zahlreicher Datenpartitionen. Diese Methode eignet sich für Bereiche gleicher Größe auf der Basis von Datumswerten oder Nummern.

Die Beispiele 1 und 2 veranschaulichen, wie die Anweisung CREATE TABLE zur Definition und automatischen Generierung der für die einzelnen Datenpartitionen angegebenen Bereiche verwendet wird.

Beispiel 1:

Führen Sie eine Anweisung CREATE TABLE mit den folgenden definierten Bereichen aus:

```
CREATE TABLE lineitem (
  l_orderkey    DECIMAL(10,0) NOT NULL,
  l_quantity    DECIMAL(12,2),
  l_shipdate    DATE,
  l_year_month  INT GENERATED ALWAYS AS (YEAR(l_shipdate)*100 + MONTH(l_shipdate)))
     PARTITION BY RANGE(l_shipdate)
    (STARTING ('1/1/1992') ENDING ('12/31/1992') EVERY 1 MONTH);
```

Diese Anweisung erstellt 12 Datenpartitionen mit jeweils einem Schlüsselwert (l_shipdate) >= ('1/1/1992'), (l_shipdate) < ('3/1/1992'), (l_shipdate) < ('4/1/1992'), (l_shipdate) < ('5/1/1992'), ..., (l_shipdate) < ('12/1/1992'), (l_shipdate) < ('12/31/1992').

Der Anfangswert (STARTING) der ersten Datenpartition ist einschließend, weil die Gesamtanfangsgrenze ('1/1/1992') einschließend ist (standardmäßig). Analog ist die Endgrenze der letzten Datenpartition einschließend, weil die Gesamtendgrenze ('12/31/1992') einschließend ist (standardmäßig). Die übrigen STARTING-Werte sind einschließend, während die übrigen ENDING-Werte alle ausschließend sind. Jede Datenpartition enthält n Schlüsselwerte, wobei n durch die Klausel EVERY gegeben ist. Mithilfe der Formel (STARTING + EVERY) können Sie das Ende des Bereichs jeder einzelnen Datenpartition ermitteln. Die letzte Datenpartition enthält möglicherweise weniger Schlüsselwerte, wenn der EVERY-Wert nicht glatt in den STARTING- und ENDING-Bereich teilbar ist.

Beispiel 2:

Führen Sie eine Anweisung CREATE TABLE mit den folgenden definierten Bereichen aus:

```
CREATE TABLE t(a INT, b INT)
 PARTITION BY RANGE(b) (STARTING FROM (1)
  EXCLUSIVE ENDING AT (1000) EVERY (100))
```

Durch diese Anweisung werden 10 Datenpartitionen jeweils mit 100 Schlüsselwerten (1 < b <= 101, 101 < b <= 201, ..., 901 < b <= 1000) erstellt.

Der Anfangswert der ersten Datenpartition (b > 1 und b <= 101) ist ausschließend, weil der Gesamtanfangswert (1) ausschließend ist. Analog ist die Endgrenze der letzten Datenpartition (b > 901 b <= 1000) einschließend, weil die Gesamtendgrenze (1000) einschließend ist. Die übrigen STARTING-Werte sind alle ausschließend, während die übrigen ENDING-Werte alle einschließend sind. Jede Datenpartition enthält n Schlüsselwerte, wobei n durch die Klausel EVERY gegeben ist. Wenn schließlich beide Grenzen, d. h. STARTING- und ENDING-Grenze, der gesamten Klausel ausschließend sind, ist der STARTING-Wert der ersten Datenpartition ausschließend, weil die Gesamtanfangsgrenze (1) ausschließend ist. Analog ist die Endgrenze der letzten Datenpartition ausschließend, weil die Gesamtendgrenze (1000) ausschließend ist. Die übrigen STARTING-Werte sind alle ausschließend, während die ENDING-Werte alle einschließend sind. Jede Datenpartition (außer der letzten) enthält n Schlüsselwerte, wobei n durch die Klausel EVERY gegeben ist.

Manuelle Generierung

Durch die manuelle Generierung wird eine neue Datenpartition für jeden Bereich erstellt, der in der Klausel PARTITION BY angegeben wird. Diese Form der Syntax bietet eine höhere Flexibilität bei der Definition von Bereichen und dadurch mehr Optionen zur Platzierung von Daten und LOB-Daten. Die Beispiele 3 und 4 veran-

schaulichen, wie die Anweisung CREATE TABLE zur Definition und manuellen Generierung der für die einzelnen Datenpartitionen angegebenen Bereiche verwendet wird.

Beispiel 3:

Durch die folgende Anweisung wird eine Partitionierung in zwei Datumsspalten ausgeführt, die beide generiert werden. Beachten Sie, dass hier die Form der CREATE TABLE-Syntax für automatische Generierung verwendet und nur jeweils ein Ende der Bereiche angegeben wird. Das andere Ende wird durch die benachbarte Datenpartition und die Verwendung der Option INCLUSIVE impliziert:

```
CREATE TABLE sales(invoice_date date, inv_month int NOT NULL
GENERATED ALWAYS AS (month(invoice_date)), inv_year INT NOT
NULL GENERATED ALWAYS AS ( year(invoice_date)), item_id int NOT NULL,
cust_id int NOT NULL) PARTITION BY RANGE (inv_year, inv_month)
(PART Q1_02 STARTING (2002,1) ENDING (2002, 3) INCLUSIVE,
PART Q2_02 ENDING (2002, 6) INCLUSIVE,
PART Q3_02 ENDING (2002, 9) INCLUSIVE,
PART Q4_02 ENDING (2002,12) INCLUSIVE,
PART CURRENT ENDING (MAXVALUE, MAXVALUE));
```

Lücken zwischen den Bereichen sind zulässig. Die CREATE TABLE-Syntax unterstützt Lücken, indem sie die Angabe eines STARTING-Werts für einen Bereich zulässt, der nicht am ENDING-Wert der vorherigen Datenpartition ausgerichtet ist.

Beispiel 4:

Die folgende Anweisung erstellt eine Tabelle mit einer Lücke zwischen den Werten 101 und 200.

```
CREATE TABLE foo(a INT)
 PARTITION BY RANGE(a)
   (STARTING FROM (1) ENDING AT (100),
   STARTING FROM (201) ENDING AT (300))
```

Durch die Verwendung der Anweisung ALTER TABLE, die ein Hinzufügen oder Entfernen von Datenpartitionen ermöglicht, können ebenfalls Lücken zwischen den Bereichen entstehen.

Wenn Sie eine Zeile in eine partitionierte Tabelle einfügen, wird sie automatisch in der ihrem Schlüsselwert entsprechenden Datenpartition und dem zugehörigen Bereich abgelegt. Wenn sie sich außerhalb aller für die Tabelle definierten Bereiche befindet, schlägt die Einfügung fehl und der folgende Fehler wird an die Anwendung zurückgegeben:

```
SQL0327N  Die Zeile kann nicht in Tabelle "<tabellenname>" eingefügt werden,
weil sie sich außerhalb der Grenzen der definierten Datenpartitionsbereiche befindet.
   SQLSTATE=22525
```

Einschränkungen

- Einschränkungen auf Tabellenebene:
  - Tabellen, die mit der Form der Syntax für automatische Generierung (mit der Klausel EVERY) erstellt werden, sind auf die Verwendung eines numerischen oder eines DATE/TIME-Datentyps im Tabellenpartitionierungsschlüssel beschränkt.
- Einschränkungen auf Anweisungsebene:
  - Die Optionen MINVALUE und MAXVALUE werden in der Form der Syntax für automatische Generierung nicht unterstützt.

– Bereiche sind aufsteigend.
– In der Form der Syntax für automatische Generierung kann nur eine Spalte angegeben werden.
– Das Inkrement in der Klausel EVERY muss größer als null sein.
– Der ENDING-Wert muss größer oder gleich dem STARTING-Wert sein.

Migrieren vorhandener Tabellen und Sichten in partitionierte Tabellen

Zur Migration von Daten aus einer Tabelle der Version 9.1 in eine leere partitionierte Tabelle verwenden Sie den Befehl LOAD.

Es gibt drei Methoden, die Sie zur Migration einer vorhandenen Tabelle oder Sicht in eine partitionierte Tabelle verwenden können:

- Bei der Migration regulärer Tabellen erstellen Sie eine neue, leere partitionierte Tabelle und versetzen die Daten mithilfe der Anweisung LOAD FROM CURSOR aus der alten Tabelle ohne weitere Zwischenschritte direkt in die partitionierte Tabelle.
- Bei der Migration regulärer Tabellen entladen Sie Daten aus der Quellentabelle mit dem Dienstprogramm EXPORT (oder einem leistungsfähigen UNLOAD), erstellen eine neue, leere partitionierte Tabelle und laden Daten mit dem Befehl LOAD in die partitionierte Tabelle.
- Bei der Migration von UNION ALL-Sichten erstellen Sie eine partitionierte Tabelle mit einer einzigen Pseudodatenpartition und ordnen alle Tabellen zu.

Beispiel 1: Nehmen Sie an, Sie haben eine reguläre Tabelle 't1':

```
CREATE TABLE t1 (c1 int, c2 int);
```

Erstellen Sie eine neue, leere partitionierte Tabelle:

```
CREATE TABLE sales_dp (c1 int, c2 int)
   PARTITION BY RANGE (c1)
   (STARTING FROM 0 ENDING AT 10 EVERY 2);
```

Füllen Sie Tabelle 't1' mit Daten:

```
INSERT INTO t1 VALUES (0,1), (4, 2), (6, 3);
```

Um das Erstellen einer dritten Kopie der Daten in einer unstrukturierten Datei zu vermeiden, führen Sie den Befehl LOAD aus, um die Daten aus einer SQL-Abfrage direkt in die neue partitionierte Tabelle zu extrahieren.

```
   SELECT * FROM t1;
   DECLARE c1 CURSOR FOR SELECT * FROM t1;
   LOAD FROM c1 of CURSOR INSERT INTO sales_dp;SELECT * FROM sales_dp;
SELECT * FROM sales_dp;
```

Löschen Sie die alte Tabelle:

```
DROP TABLE t1;
```

Konvertieren von UNION ALL-Sichten

Sie können Daten der Version 9.1 in einer UNION ALL-Sicht in eine partitionierte Tabelle konvertieren. UNION ALL-Sichten dienen zur Verwaltung großer Tabellen sowie zur Realisierung einfacher Rollin- und Rollout-Operationen für Tabellendaten bei gleichzeitiger Nutzung der Leistungsvorteile des Verzweigungsausschlusses. Die Tabellenpartitionierung erfüllt alle diese Zwecke und lässt sich leichter ver-

walten. Mit einer Operation ALTER TABLE ...ATTACH können Sie die Konvertierung ausführen, ohne Daten in der Basistabelle zu versetzen. Indizes und abhängige Sichten oder MQTs (Materialized Query Tables) müssen nach der Konvertierung erneut erstellt werden.

Die empfohlene Strategie besteht darin, ein partitionierte Tabelle mit nur einer Pseudodatenpartition zu erstellen und anschließend alle Tabellen der UNION ALL-Sicht zuzuordnen. Stellen Sie sicher, dass Sie die Pseudodatenpartition bei diesem Prozess frühzeitig löschen, um Probleme aufgrund von Bereichsüberschneidungen zu vermeiden.

Beispiel 2:

CREATE TABLE-Syntax zur Erstellung der ersten Tabelle in der UNION-Sicht:

```
CREATE TABLE sales_0198(
 sales_date DATE NOT NULL,
 prod_id INTEGER,
 city_id INTEGER,
 channel_id INTEGER,
 revenue DECIMAL(20,2),
  CONSTRAINT ck_date
  CHECK
  (sales_date BETWEEN '01-01-1998' AND '01-31-1998'));
```

CREATE VIEW-Syntax für eine UNION ALL-Sicht:

```
CREATE VIEW all_sales AS
 (
 SELECT * FROM sales_0198
 WHERE sales_date BETWEEN '01-01-1998' AND '01-31-1998'
 UNION ALL
 SELECT * FROM sales_0298
 WHERE sales_date BETWEEN '02-01-1998' AND '02-28-1998'
 UNION ALL
 ...
 UNION ALL
 SELECT * FROM sales_1200
 WHERE sales_date BETWEEN '12-01-2000' AND '12-31-2000'
 );
```

Erstellen Sie eine partitionierte Tabelle mit einer einzigen Pseudopartition. Der Bereich sollte so gewählt werden, dass er sich nicht mit der ersten zuzuordnenden Datenpartition überschneidet:

```
CREATE TABLE sales_dp (
 sales_date DATE NOT NULL,
 prod_id INTEGER,
 city_id INTEGER,
 channel_id INTEGER,
 revenue DECIMAL(20,2))
 PARTITION BY RANGE (sales_date)
 (PART dummy STARTING FROM '01-01-1900' ENDING AT '01-01-1900');
```

Ordnen Sie die erste Tabelle zu:

```
ALTER TABLE sales_dp ATTACH PARTITION
STARTING FROM '01-01-1998' ENDING AT '01-31-1998'
FROM sales_0198;
```

Löschen Sie die Pseudopartition ('dummy'):

```
ALTER TABLE sales_dp DETACH PARTITION dummy
INTO dummy;
 DROP TABLE dummy;
```

Ordnen Sie die verbleibenden Partitionen zu:

```
ALTER TABLE sales_dp ATTACH PARTITION STARTING
FROM '02-01-1998' ENDING AT '02-28-1998' FROM sales_0298;
...
ALTER TABLE sales_dp ATTACH PARTITION STARTING
FROM '12-01-2000' ENDING AT '12-31-2000' FROM sales_1200;
```

Führen Sie die Anweisung SET INTEGRITY aus, um die zugeordneten Datenpartitionen online verfügbar zu machen.

```
SET INTEGRITY FOR sales_dp IMMEDIATE CHECKED
FOR EXCEPTION IN sales_dp USE sales_ex;
```

Erstellen Sie geeignete Indizes.

Aspekte der Konvertierung

Die Zuordnung einer Datenpartition ist zulässig, es sei denn, der Wert des Felds IMPLICITVALUE von SYSCAT.COLUMNS in einer bestimmten Spalte ist für die Quellenspalte und die Zielspalte ungleich Null und die Werte stimmen nicht überein. In diesem Fall müssen Sie die Quellentabelle löschen und anschließend erneut erstellen.

Eine Spalte kann einen Wert ungleich Null im Feld IMPLICITVALUE von SYSCAT.COLUMNS enthalten, wenn eine der folgenden Bedingungen zutrifft:

- Die Spalte wurde durch eine Anweisung ALTER TABLE ...ADD COLUMN erstellt.
- Das Feld IMPLICITVALUE wurde während der Zuordnung aus der Quellentabelle weitergegeben.
- Das Feld IMPLICITVALUE wurde aus einer Quellentabelle während der Aufhebung der Zuordnung übernommen.
- Das Feld IMPLICITVALUE wurde bei der Migration von Version 8 auf Version 9 festgelegt, wobei die Spalte als hinzugefügte Spalte ermittelt wurde bzw. möglicherweise eine hinzugefügte Spalte ist. Wenn die Datenbank nicht sicher ermitteln kann, ob die Spalte hinzugefügt wurde oder nicht, wird die Spalte wie eine hinzugefügte Spalte behandelt. Eine hinzugefügte Spalte ist eine Spalte, die durch eine Anweisung ALTER TABLE ...ADD COLUMN erstellt wurde.

Zur Vermeidung solcher Inkonsistenzen wird empfohlen, die Quellen- und die Zieltabelle, die an einer Zuordnungsoperation beteiligt sind, stets mit den gleichen definierten Spalten zu erstellen. Verwenden Sie insbesondere nie die Anweisung ALTER TABLE, um einer Zieltabelle einer Zuordnungsoperation eine Spalte hinzuzufügen.

Informationen zu bewährten Verfahren zur Vermeidung einer Abweichung bei der Arbeit mit partitionierten Tabellen finden Sie in „Richtlinien zum Hinzufügen von Datenpartitionen zu partitionierten Tabellen” auf Seite 223.

Verhalten von partitionierten MQTs

Alle Typen von MQTs (Materialized Query Table, gespeicherte Abfragetabelle) werden mit partitionierten Tabellen unterstützt. Wenn Sie mit partitionierten MQTs arbeiten, können Ihnen einige Richtlinien helfen, zugeordnete Datenpartitionen und Datenpartitionen mit aufgehobener Zuordnung möglichst effektiv zu verwalten.

Die folgenden Richtlinien und Einschränkungen gelten für die Arbeit mit partitionierten MQTs bzw. partitionierten Tabellen mit abhängigen Tabellen, deren Zuordnung aufgehoben wurde:

- Wenn Sie eine Operation DETACH PARTITION zur Aufhebung der Zuordnung einer Datenpartition ausführen und abhängige Tabellen vorhanden sind, die im Hinblick auf die Datenpartition, deren Zuordnung aufgehoben wurde, inkrementell gepflegt werden müssen (solche abhängigen Tabellen werden als freigegebene abhängige Tabellen bezeichnet), ist zunächst kein Zugriff auf die neue Tabelle mit aufgehobener Zuordnung möglich. Die Tabelle wird mit dem Wert L in der Spalte TYPE der Katalogsicht SYSCAT.TABLES markiert. Dies kennzeichnet sie als Tabelle mit aufgehobener Zuordnung (bzw. als freigegebene Tabelle). Dadurch wird verhindert, dass die Tabelle gelesen, modifiziert oder gelöscht wird, bevor die Anweisung SET INTEGRITY ausgeführt wird, um die freigegebenen abhängigen Tabellen inkrementell zu warten. Nach der Ausführung der Anweisung SET INTEGRITY für alle freigegebenen abhängigen Tabellen geht die Tabelle mit aufgehobener Zuordnung in den Status einer regulären Tabelle über, sodass ein vollständiger Zugriff auf sie möglich wird.
- Fragen Sie die Katalogsicht SYSCAT.TABDETACHEDDEP ab, um eine Tabelle mit aufgehobener Zuordnung zu ermitteln, auf die noch nicht zugegriffen werden kann. Wenn Tabellen mit aufgehobener Zuordnung erkannt werden, auf die noch nicht zugegriffen werden kann, führen Sie die Anweisung SET INTEGRITY mit der Option IMMEDIATE CHECKED für alle freigegebenen abhängigen Tabellen aus, um die Tabelle mit aufgehobener Zuordnung in eine reguläre Tabelle umzuwandeln, auf die zugegriffen werden kann. Wenn Sie versuchen, auf eine Tabelle mit aufgehobener Zuordnung zuzugreifen, bevor die zugehörigen abhängigen Tabellen gepflegt werden, wird der Fehlercode SQL20285N zurückgegeben.
- Die Funktion DATAPARTITIONNUM kann in der Definition einer MQT nicht verwendet werden. Ein Versuch, eine MQT mit dieser Funktion zu erstellen, führt zu einem Fehler (SQLCODE SQL20058N, SQLSTATE 428EC).
- Wenn ein Index für eine Tabelle mit Datenpartitionen, deren Zuordnung aufgehoben ist, erstellt wird, enthält der Index die Daten einer Datenpartition mit aufgehobener Zuordnung nur dann, wenn die Datenpartition mit aufgehobener Zuordnung eine MQT hat, die im Hinblick auf die Datenpartition inkrementell aktualisiert werden muss. In diesem Fall enthält der Index die Daten für diese Datenpartition mit aufgehobener Zuordnung.
- Das Ändern einer Tabelle mit zugeordneten Datenpartitionen in eine MQT ist nicht zulässig.
- Partitionierte Zwischenspeichertabellen werden nicht unterstützt.
- Das Zuordnung von Datenpartitionen zu einer MQT wird nicht direkt unterstützt. Weitere detaillierte Informationen finden Sie unter Beispiel 1.

Beispiel 1: Konvertieren einer partitionierten MQT in eine normale Tabelle

Obwohl die Operation ATTACH nicht direkt für partitionierte MQTs unterstützt wird, können Sie denselben Effekt erzielen, indem Sie eine partitionierte MQT in eine normale Tabelle umwandeln, die gewünschten Rollin- und Rollout-Operationen für Tabellendaten ausführen und anschließend die Tabelle wieder in eine MQT umwandeln. Die folgenden Beispielanweisungen CREATE TABLE und ALTER TABLE veranschaulichen, wie sich dieser Effekt realisieren lässt:

```
CREATE TABLE lineitem (
  l_orderkey    DECIMAL(10,0) NOT NULL,
  l_quantity    DECIMAL(12,2),
  l_shipdate    DATE,
  l_year_month  INT GENERATED ALWAYS AS (YEAR(l_shipdate)*100 + MONTH(l_shipdate)))
```

```
    PARTITION BY RANGE(l_shipdate)
    (STARTING ('1/1/1992') ENDING ('12/31/1993') EVERY 1 MONTH);
CREATE TABLE lineitem_ex (
  l_orderkey    DECIMAL(10,0) NOT NULL,
  l_quantity    DECIMAL(12,2),
  l_shipdate    DATE,
  l_year_month  INT,
  ts            TIMESTAMP,
  msg           CLOB(32K));

CREATE TABLE quan_by_month (
  q_year_month, q_count) AS
    (SELECT l_year_month AS q_year_month, COUNT(*) AS q_count
     FROM lineitem
     GROUP BY l_year_month)
     DATA INITIALLY DEFERRED REFRESH IMMEDIATE
     PARTITION BY RANGE(q_year_month)
     (STARTING (199201) ENDING (199212) EVERY (1),
      STARTING (199301) ENDING (199312) EVERY (1));
CREATE TABLE quan_by_month_ex(
  q_year_month  INT,
  q_count       INT NOT NULL,
  ts            TIMESTAMP,
  msg           CLOB(32K));

SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED;
CREATE INDEX qbmx ON quan_by_month(q_year_month);

ALTER TABLE quan_by_month DROP MATERIALIZED QUERY;
ALTER TABLE lineitem DETACH PARTITION part0 INTO li_reuse;
ALTER TABLE quan_by_month DETACH PARTITION part0 INTO qm_reuse;

SET INTEGRITY FOR li_reuse OFF;
ALTER TABLE li_reuse ALTER l_year_month SET GENERATED ALWAYS
AS (YEAR(l_shipdate)*100 + MONTH(l_shipdate));

LOAD FROM part_mqt_rotate.del OF DEL MODIFIED BY GENERATEDIGNORE
MESSAGES load.msg REPLACE INTO li_reuse;

DECLARE load_cursor CURSOR FOR
  SELECT l_year_month, COUNT(*)
    FROM li_reuse
    GROUP BY l_year_month;
LOAD FROM load_cursor OF CURSOR MESSAGES load.msg
  REPLACE INTO qm_reuse;

ALTER TABLE lineitem ATTACH PARTITION STARTING '1/1/1994'
  ENDING '1/31/1994' FROM li_reuse;

SET INTEGRITY FOR lineitem ALLOW WRITE ACCESS IMMEDIATE CHECKED
FOR EXCEPTION IN lineitem USE lineitem_ex;

ALTER TABLE quan_by_month ATTACH PARTITION STARTING 199401
  ENDING 199401 FROM qm_reuse;

SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED
  FOR EXCEPTION IN quan_by_month USE quan_by_month_ex;

ALTER TABLE quan_by_month ADD MATERIALIZED QUERY
  (SELECT l_year_month AS q_year_month, COUNT(*) AS q_count
     FROM lineitem
       GROUP BY l_year_month)
         DATA INITIALLY DEFERRED REFRESH IMMEDIATE;

SET INTEGRITY FOR QUAN_BY_MONTH ALL IMMEDIATE UNCHECKED;
```

Verwenden Sie die Anweisung SET INTEGRITY mit der Option IMMEDIATE CHECKED, um die zugeordnete Datenpartition auf Integritätsverletzungen zu überprüfen. Dieser Schritt muss erfolgen, bevor die Tabelle wieder in eine MQT zurückgeändert wird. Die Anweisung SET INTEGRITY mit der Option IMMEDIATE UNCHECKED dient zur Umgehung der erforderlichen vollständigen Aktualisierung der MQT. Der Index für die MQT ist erforderlich, um eine optimale Leistung zu erreichen. Die Verwendung von Ausnahmetabellen mit der Anweisung SET INTEGRITY wird empfohlen, sofern geeignet.

Typischerweise wird eine partitionierte MQT für eine große Fakttabelle erstellt, die ebenfalls partitioniert ist. Wenn Sie ein Rollout oder Rollin von Tabellendaten für eine große Fakttabelle durchführen, müssen Sie die partitionierte MQT manuell anpassen, wie in Beispiel 2 gezeigt.

Beispiel 2: Manuelles Anpassen einer partitionierten MQT

Ändern Sie die MQT (quan_by_month), um sie in eine normale partitionierte Tabelle umzuwandeln:

```
ALTER TABLE quan_by_month DROP MATERIALIZED QUERY;
```

Heben Sie die Zuordnung der Daten auf, für die ein Rollout aus der Fakttabelle (lineitem) und der MQT ausgeführt werden soll, und laden Sie die neuen Daten, für die ein Rollin durchgeführt werden soll, in die Zwischenspeichertabelle (li_reuse):

```
ALTER TABLE lineitem DETACH PARTITION part0 INTO li_reuse;

LOAD FROM part_mqt_rotate.del OF DEL MESSAGES load.msg REPLACE INTO li_reuse;

ALTER TABLE quan_by_month DETACH PARTITION part0 INTO qm_reuse;
```

Bereinigen Sie die Tabelle 'qm_reuse', bevor Sie die Einfügung ausführen. Dadurch werden die Daten mit aufgehobener Zuordnung gelöscht, bevor die Subselect-Daten eingefügt werden. Dies wird durch Laden (LOAD) mit der Option REPLACE in die MQT erreicht, wobei die Datendatei (datafile.del) der Ladeoperation der Inhalt des Subselect ist.

```
db2 load from datafile.del of del replace into qm_reuse
```

Sie können die Tabelle manuell mit der Anweisung INSERT INTO ... (SELECT ...) aktualisieren. Dies ist nur für die neuen Daten erforderlich, sodass die Anweisung vor dem Zuordnen ausgeführt werden sollte:

```
INSERT INTO qm_reuse
  (SELECT COUNT(*) AS q_count, l_year_month AS q_year_month
       FROM li_reuse
         GROUP BY l_year_month);
```

Jetzt können Sie das Rollin der neuen Daten für die Fakttabelle durchführen:

```
ALTER TABLE lineitem ATTACH PARTITION STARTING '1/1/1994'
ENDING '1/31/1994' FROM TABLE li_reuse;
SET INTEGRITY FOR lineitem ALLOW WRITE ACCESS IMMEDIATE CHECKED FOR
EXCEPTION IN li_reuse USE li_reuse_ex;
```

Im nächsten Schritt führen Sie das Rollin der Daten für die MQT durch:

```
ALTER TABLE quan_by_month ATTACH PARTITION STARTING 199401
ENDING 199401 FROM TABLE qm_reuse;
SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED;
```

Nach der Zuordnung der Datenpartition müssen die neuen Daten überprüft werden, um sicherzustellen, dass sie im gültigen Bereich liegen.

```
ALTER TABLE quan_by_month ADD MATERIALIZED QUERY
  (SELECT COUNT(*) AS q_count, l_year_month AS q_year_month
     FROM lineitem
       GROUP BY l_year_month)
         DATA INITIALLY DEFERRED REFRESH IMMEDIATE;
SET INTEGRITY FOR QUAN_BY_MONTH ALL IMMEDIATE UNCHECKED;
```

Der Zugriff auf die Daten ist erst möglich, wenn sie durch die Anweisung SET INTEGRITY geprüft wurden. Obwohl auch die Operation REFRESH TABLE unterstützt wird, zeigt dieses Szenario die manuelle Pflege einer partitionierten MQT durch die Operationen ATTACH PARTITION und DETACH PARTITION. Die Daten werden durch den Benutzer mithilfe der Klausel IMMEDIATE UNCHECKED der Anweisung SET INTEGRITY als geprüft markiert.

Erstellen von Bereichsclustertabellen (RCT)

Algorithmen für Bereichsclustertabellen

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.

Indizes für Bereichsclustertabellen

In Bereichsclustertabellen lokalisieren Indizes einen Datensatz auf der Basis eines Schlüssels aus dem Datensatz, wenden Start- und Stoppsuchen an und verteilen die Daten vertikal. Bei Verwendung einer Bereichsclustertabelle ist das einzige Merkmal eines Index, das nicht berücksichtigt wird, die vertikale Verteilung von Daten.

Unterschiede zu regulären Tabellen

Bei der Entscheidung zur Verwendung von Bereichsclustertabellen (RCTs) sind einige Merkmale zu berücksichtigen, die 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 des Speicherplatzbedarfs 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, sodass 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.
- Es wird kein Bereichsclusterindex 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.

Richtlinien zur Verwendung von Bereichsclustertabellen

Für die Arbeit mit Bereichsclustertabellen (RCT - Range-Clustered Tables) sind einige Richtlinien zu befolgen.

- Beim Definieren des Bereichs der Schlüsselwerte ist der Mindestwert optional. Wird dieser Wert nicht angegeben, wird als Standardwert 1 verwendet. Negative Werte sind als Mindest- und Maximalwerte zulässig. Beim Arbeiten mit negativen Werten muss der Mindestwert explizit angegeben werden. Beispiel: ORGANIZE BY KEY SEQUENCE (F1 STARTING FROM -100 ENDING AT -10)
- Das Erstellen eines regulären Indexes für dieselben Schlüsselwerte, die zum Definieren der Bereichsclustertabelle verwendet wurden, ist nicht zulässig.
- Einige ALTER TABLE-Optionen können bei Bereichsclustertabellen nicht verwendet werden. Wenn die Option die physische Struktur der Tabelle nicht beeinflusst, ist die Option zulässig.
- Da beim Erstellen einer Bereichsclustertabelle der erforderliche Plattenspeicherplatz vorab zugeordnet wird, muss dieser Speicherplatz verfügbar sein, da andernfalls die Tabellenerstellung fehlschlägt.

Funktionsweise des SQL-Compilers bei Bereichsclustertabellen

Der SQL-Compiler verarbeitet Bereichsclustertabellen (RCT = Range-Clustered Table) in ähnlicher Weise wie reguläre Tabellen, die über eine sekundäre B+-Indexstruktur verfügen. Anstatt eine B+-Indexstruktur zu durchlaufen, um die Speicherposition oder die Satzkennung (RID = Record Identifier) eines Datensatzes festzustellen, wird in Bereichsclustertabellen eine funktionale Suchroutine eingesetzt, die auf der Basis der Datensatz-Schlüsselwerte und des Algorithmus aus der Bereichsdefinition arbeitet. Hierbei wird ähnlich vorgegangen wie bei einem Index, da ein Schlüsselwert verwendet wird, um die RID schnell abrufen zu können.

Wenn der beste Zugriffspfad zu den erforderlichen Daten festgestellt werden soll, verwendet der SQL-Compiler Statistikdaten, die zu den Tabellen gespeichert werden. Die Indexstatistikdaten werden während der Tabellensuche erfasst, wenn der Befehl RUNSTATS ausgegeben wird. Bei einer Bereichsclustertabelle wird die Tabelle selbst als reguläre Tabelle und der Index als funktionsbasierter Index aufgebaut.

Die Reihenfolge der Datensätze in der Tabelle kann nicht mit Sicherheit vorausbestimmt werden, wenn beim Erstellen einer Bereichsclustertabelle der Überlauf als zulässig definiert wird.

Szenarios: Bereichsclustertabellen

Diese Szenarios sind einfach und veranschaulichen die Vorgehensweise zum Erstellen einer Bereichsclustertabelle. Sie zeigen, wie Sie entweder eine einzelne Spalte oder mehrere Spalten als Schlüssel für eine Tabelle verwenden können. Darüber hinaus erläutern sie, wie eine Tabelle erstellt werden kann, in der ein Datenüberlauf zulässig bzw. nicht zulässig ist.

Szenario 1: Erstellen einer Bereichsclustertabelle

Dieses Szenario zeigt eine Bereichsclustertabelle, die zum Auffinden eines Kursteilnehmers über seine ID (STUDENT_ID) verwendet wird. Jeder Kursteilnehmerdatensatz umfasst die folgenden Informationen:

- ID der Schule (SCHOOOL_ID)
- ID des Kurses (PROGRAM_ID)
- Kursteilnehmernummer (STUDENT_NUM)
- Kursteilnehmer-ID (STUDENT_ID)
- Vorname des Kursteilnehmers (FIRST_NAME)
- Nachname des Kursteilnehmers (LAST_NAME)
- Durchschnittliche Leistung des Kursteilnehmers (GPA)

In diesem Fall basieren die Kursteilnehmerdatensätze nur auf der Spalte STUDENT_ID. Die Spalte STUDENT_ID wird zum Hinzufügen, Aktualisieren und Löschen von Kursteilnehmerdatensätzen verwendet.

Anmerkung: Weitere Indizes können zu einem späteren Zeitpunkt separat hinzugefügt werden. Im Rahmen des vorliegenden Beispiels wird die Struktur der Tabelle und die Art und Weise des Zugriffs auf die Tabellendaten jedoch während der Tabellenerstellung definiert.

Die Syntax, die für diese Tabelle erforderlich ist, sieht folgendermaßen aus:

```
CREATE TABLE STUDENTS
   (SCHOOL_ID      INT NOT NULL,
    PROGRAM_ID     INT NOT NULL,
    STUDENT_NUM    INT NOT NULL,
    STUDENT_ID     INT NOT NULL,
    FIRST_NAME     CHAR(30),
    LAST_NAME      CHAR(30),
    GPA            FLOAT)
   ORGANIZE BY KEY SEQUENCE
   (STUDENT_ID     STARTING FROM 1 ENDING AT 1000000)
    ALLOW OVERFLOW
;
```

Die Größe der einzelnen Datensätze ergibt sich aus der Summe der zugehörigen Spalten. Im vorliegenden Fall wird ein 10 Byte großer Header + 4 + 4 + 4 + 4 + 30 + 30 + 8 + 3 (für Spalten, die Nullwerte enthalten können) verwendet, wodurch sich eine Größe von 97 Byte ergibt. Mit einer Seitengröße von 4 KB (d. h. 4096 Byte) verbleiben unter Berücksichtigung des Systemaufwands 4038 Byte, also genug Speicherplatz für 42 Datensätze pro Seite. Wird eine Anzahl von 1 Million Kursteilnehmerdatensätzen zugelassen, dann ergibt sich für die Anzahl der erforderlichen Seiten der Wert 1 Million dividiert durch 42 Datensätze pro Seite = 23809,5. Aufgerundet kann ergibt sich demnach ein Wert von 23810 erforderlichen Seiten. Für die Tabellen werden vier und für die Bereichszuordnung werden weitere drei Seiten einkalkuliert. Daraus ergibt sich ein Wert von 23817 Seiten mit jeweils 4 KB Größe, die vorab zugeordnet werden müssen. (Bei der Bereichs-

zuordnung wird davon ausgegangen, dass die Tabelle in einem einzigen Container gespeichert wird. Für jeden Container sollten drei Seiten verfügbar sein.)

Szenario 2: Erstellen einer Bereichsclustertabelle (Überlauf nicht zulässig)

In diesem Szenario, das eine Variante des ersten Szenarios darstellt, wird eine Vereinigung von Schulen betrachtet. Diese Schulvereinigung umfasst 200 Schulen, die jeweils 20 Schulungsräume mit einer Kapazität von 35 Kursteilnehmern umfassen. Diese Schulvereinigung kann maximal 140.000 Kursteilnehmer unterbringen.

Im diesem Fall basieren die Kursteilnehmerdatensätze auf drei Faktoren: Schul-ID (SCHOOL_ID), Kurs-ID (CLASS_ID) und Kursteilnehmernummer (STUDENT_NUM). Jede dieser drei Spalten verfügt über eindeutige Werte und wird zusammen mit den anderen Spalten zum Hinzufügen, Aktualisieren und Löschen von Kursteilnehmerdatensätzen verwendet.

Anmerkung: Wie im vorherigen Beispiel können auch weitere Indizes zu einem späteren Zeitpunkt separat hinzugefügt werden.

Die Syntax, die für diese Tabelle erforderlich ist, sieht folgendermaßen aus:

```
CREATE TABLE STUDENTS
   (SCHOOL_ID      INT NOT NULL,
    CLASS_ID       INT NOT NULL,
    STUDENT_NUM    INT NOT NULL,
    STUDENT_ID     INT NOT NULL,
    FIRST_NAME     CHAR(30),
    LAST_NAME      CHAR(30),
    GPA            FLOAT)
   ORGANIZE BY KEY SEQUENCE
   (SCHOOL_ID      STARTING FROM 1 ENDING AT 200,
    CLASS_ID       STARTING FROM 1 ENDING AT 20,
    STUDENT_NUM    STARTING FROM 1 ENDING AT 35)
    DISALLOW OVERFLOW
;
```

In diesem Fall ist ein Überlauf nicht zulässig. Dies ist sinnvoll, da wahrscheinlich eine Richtlinie der Schulvereinigung existiert, die die Anzahl der Kursteilnehmer innerhalb eines Kurses beschränkt. In diesem Szenario ist die Kursgröße auf maximal 35 Teilnehmer begrenzt. Wenn Sie diesen Faktor mit den jeweiligen örtlichen Beschränkungen durch die Anzahl der Schulungsräume und Schulen kombinieren, wird klar, dass es keinen Grund gibt, aus dem ein Überlauf bei der Anzahl der Kursteilnehmer in der Schulvereinigung zugelassen werden sollte.

Die Schulen verfügen möglicherweise über unterschiedliche Anzahlen von Schulungsräumen. In diesem Fall muss beim Festlegen des Bereichs für die Anzahl der Schulungsräume (mithilfe von CLASS_ID) als Obergrenze die größte Anzahl an Schulungsräumen verwendet werden, wenn alle Schulen berücksichtigt werden. Dies hat zur Folge, dass einige kleinere Schulen (d. h. Schulen mit weniger Schulungsräumen als die größte Schule) Platz für Kursteilnehmerdatensätze haben, der möglicherweise nie verwendet wird (sofern die Kapazität der betreffenden Schule nicht z. B. durch mobile Schulungsräume aufgestockt wird).

Wenn Sie dieselbe Seitengröße von 4 KB und dieselbe Größe für die Kursteilnehmerdatensätze verwenden wie im vorherigen Beispiel, dann können 42 Datensätze pro Seite gespeichert werden. Bei 140.000 Kursteilnehmerdatensätzen werden also 3333,3 Seiten (bzw. aufgerundet 3334 Seiten) benötigt. Unter Berücksichtigung von zwei Seiten für Tabelleninformationen

und drei Seiten für die Bereichszuordnung ergibt sich demnach ein Wert von 3339 Seiten mit jeweils 4 KB Größe, die vorab zugeordnet werden müssen.

Aspekte der Erstellung von MDC-Tabellen

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 einigen dieser Aspekte behandelt.

Versetzen von Daten aus vorhandenen Tabellen in MDC-Tabellen

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 (mithilfe 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, ist eine Verwendung der mehrseitigen Dateizuordnung erforderlich. (Die mehrseitige Dateizuordnung ist ab Version 8.2 die Standardeinstellung für neu erstellte Datenbanken.) Dies 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.

Eine Voraussetzung für die Verwendung der MDC-Funktion des DB2-Designadvisors ist das Vorhandensein mindestens mehrerer EXTENTSIZE großer Speicherbereiche mit Daten innerhalb der Datenbank. Der DB2-Designadvisor verwendet die Daten zur Modellierung der Datendichte und Kardinalität.

Wenn die Datenbank keine Daten in den Tabellen enthält, empfiehlt der DB2-Designadvisor kein MDC, selbst wenn die Datenbank zwar leere Tabellen enthält, jedoch über nachgebildete Statistikdaten verfügt, die eine mit Daten gefüllte Datenbank simulieren.

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, sodass 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 mithilfe 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 Analy-

seoperation 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 MQTs (Materialized Query Tables, gespeicherte Abfragetabellen), „C" für MDC und „P" für eine Umgebung mit partitionierten Datenbanken. 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 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 Umgebungen mit partitionierten Datenbanken

Das mehrdimensionale Clustering (MDC) kann in Verbindung mit einer Umgebung mit partitionierten Datenbanken verwendet werden. Tatsächlich kann MDC eine Umgebung mit partitionierten Datenbanken ergänzen. Eine Umgebung mit partitionierten Datenbanken 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 Datenbankpartition hinaus
- Verbesserung der Skalierbarkeit der Datenbank

Der Grund zur Verteilung 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 Verteilungsschlüssel nicht. Der Verteilungsschlüssel 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 Verteilungsschlüssel 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 Datenbankpartitionen verteilt wäre, würde die Spalte 'Farbe' zur Unterteilung der Daten verwendet. Infolgedessen können zum Beispiel die Sektoren 'Rot' und 'Blau' in einer Datenbankpartition und der Sektor 'Gelb' in einer anderen Datenbankpartition abgelegt werden. Wenn der Verteilungsschlüssel 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 Datenbankpartitionen bestimmen lässt.

MDC-Tabellen mit mehreren Dimensionen

Wenn Sie wissen, dass bestimmte Vergleichselemente in Abfragen sehr häufig verwendet werden, können Sie die Tabelle mithilfe 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, sodass 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, sodass 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, sodass Abfragen, die nur die Spalte c4 betreffen, mehr Verarbeitungsaufwand erfordern. Allerdings ist in Beispiel 2 ein 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 geographische 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 Er-

stellung 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 Dimensionen auf generierten Spalten

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 die 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 die Dimension (monat) können Blockindexsuchen ausgeführt werden. Für Abfragen auf die 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 DATE > '1999-03-03' AND DATE < '2000-01-15'
```

Der Compiler generiert die folgenden zusätzlichen Vergleichselemente: „month >= 199903" und „month <= 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('01/05/1999')) = 1
MONTH(date('02/08/1999')) = 2
MONTH(date('03/24/1999')) = 3
MONTH(date('04/30/1999')) = 4
...
MONTH(date('12/09/1999')) = 12
MONTH(date('01/18/2000')) = 1
MONTH(date('02/24/2000')) = 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. Version 9.1 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).

Der Datenbankmanager 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 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 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.

Kapitel 12. Ändern einer Datenbank

Ändern einer Instanz

Ändern der Datenbankkonfiguration über mehrere Datenbankpartitionen

Wenn Sie eine Datenbank haben, die auf mehrere Datenbankpartitionen verteilt ist, sollte die Konfigurationsdatei für die Datenbank in allen Datenbankpartitionen die gleiche sein.

Diese Konsistenz ist erforderlich, da der SQL-Compiler verteilte SQL-Anweisungen anhand der Informationen in der Knotenkonfigurationsdatei kompiliert und einen Zugriffsplan erstellt, der die Anforderungen der jeweiligen SQL-Anweisung erfüllt. Wenn sich verschiedene Konfigurationsdateien in den Datenbankpartitionen befinden, kann dies je nachdem, in welcher Datenbankpartition die Anweisung vorbereitet wird, zu verschiedenen Zugriffsplänen führen. Verwenden Sie den Befehl db2_all, um die Konfigurationsdateien in allen Datenbankpartitionen zu verwalten.

Ändern einer Datenbank

Ändern einer Datenbankpartitionsgruppe

Zum Ändern einer Datenbankpartitionsgruppe über den Befehlszeilenprozessor verwenden Sie den Befehl REDISTRIBUTE DATABASE PARTITION GROUP. Wenn Sie Datenbankpartitionen hinzufügen oder löschen, müssen Sie die aktuellen Daten auf die neuen Datenbankpartitionen in der Datenbankpartitionsgruppe umverteilen.

Verwalten von Datenbankpartitionen über die Steuerzentrale

Zur Arbeit mit Datenbankpartitionen können Sie die Sicht **Datenbankpartitionen** der Steuerzentrale verwenden.

Zum Arbeiten mit Datenbankpartitionen müssen Sie über die Berechtigung zum Herstellen der Verbindung zu einer Instanz verfügen. Jeder Benutzer mit der Berechtigung SYSADM oder DBADM kann Ihnen die Berechtigung zum Zugriff auf eine bestimmte Instanz erteilen.

Zur Konfiguration einer Datenbankpartition oder zum Herausnehmen einer Datenbankpartition aus dem Status **Aktualisierende Recovery anstehend** müssen Sie über die Berechtigung SYSADM, SYSCTRL oder SYSMAINT verfügen.

Über die Sicht **Datenbankpartitionen** können Sie eine Datenbankpartition erneut starten, eine Datenbankpartition aus dem Status **Aktualisierende Recovery anstehend** herausnehmen, ein Backup oder einen Restore einer Datenbankpartition durchführen und eine Datenbankpartition mithilfe des Konfigurationsadvisors konfigurieren.

Gehen Sie zum Öffnen der Sicht Datenbankpartitionen in der Steuerzentrale wie folgt vor:

1. Erweitern Sie in der Steuerzentrale die Objektbaumstruktur, bis die partitionierte Datenbank angezeigt wird, deren Datenbankpartitionen Sie anzeigen möchten.
2. Klicken Sie die gewünschte partitionierte Datenbank mit der rechten Maustaste an, und wählen Sie **Datenbankpartitionen öffnen** in der Menüliste aus.
3. Die Sicht **Datenbankpartitionen** für die ausgewählte partitionierte Datenbank wird geöffnet.

Gehen Sie wie folgt vor, um eine Datenbankpartition zu konfigurieren:

1. Wählen Sie die gewünschten Datenbankpartitionen in der Sicht **Datenbankpartitionen** aus.
2. Wählen Sie **Datenbankpartitionen** aus, klicken Sie mit der rechten Maustaste, und wählen Sie **Konfigurationsadvisor** in der Liste aus.
3. Der Konfigurationsadvisor wird geöffnet. Geben Sie mithilfe des Konfigurationsadvisors Werte für die Datenbankkonfigurationsparameter an.

Kapitel 13. Ändern von Tabellen und anderen zugehörigen Objekten

Ändern partitionierter Tabellen

Für eine partitionierte Tabelle werden alle relevanten Klauseln der Anweisung ALTER TABLE unterstützt. Darüber hinaus ermöglicht Ihnen die Anweisung ALTER TABLE, neue Datenpartitionen hinzuzufügen (ADD PARTITION), Rollins (ATTACH PARTITION) neuer Datenpartitionen durchzuführen sowie Rollouts (DETACH PARTITION) für vorhandene Datenpartitionen durchzuführen.

Zum Ändern einer partitionierten Tabelle zum Zweck der Aufhebung der Zuordnung einer Datenpartition muss der Benutzer über die folgenden Berechtigungen bzw. Zugriffsrechte verfügen:

- Der Benutzer, der die DETACH-Operation ausführt, muss über die Berechtigung verfügen, die erforderlich ist, um die Quellentabelle zu ändern (ALTER), Daten aus der Quellentabelle auszuwählen (SELECT) und die Quellentabelle zu löschen (DELETE).
- Der Benutzer muss außerdem über die Berechtigung verfügen, die zum Erstellen (CREATE) der Zieltabelle erforderlich ist. Daher müssen zum Ändern einer Tabelle für die Aufhebung der Zuordnung einer Datenpartition die Zugriffsberechtigungen der Berechtigungs-ID der Anweisung mindestens eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte für die Zieltabelle beinhalten:
  - Berechtigung SYSADM oder DBADM
  - Berechtigung CREATETAB für die Datenbank und das Zugriffsrecht USE für die von der Tabelle verwendeten Tabellenbereiche sowie eine der folgenden Berechtigungen (bzw. Zugriffsrechte):
    - Berechtigung IMPLICIT_SCHEMA für die Datenbank, wenn der implizite oder explizite Schemaname der Tabelle nicht existiert
    - Zugriffsrecht CREATEIN für das Schema, wenn sich der Schemaname der Tabelle auf ein vorhandenes Schema bezieht

Zum Ändern einer partitionierten Tabelle zu dem Zweck, eine Datenpartition zuzuordnen, müssen die Zugriffsrechte der Berechtigungs-ID der Anweisung mindestens eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte für die Quellentabelle beinhalten:

- Zugriffsrecht SELECT für die Quellentabelle und Zugriffsrecht DROPIN für das Schema der Quellentabelle
- Zugriffsrecht CONTROL für die Quellentabelle
- Berechtigung SYSADM oder DBADM

Zum Ändern einer partitionierten Tabelle zu dem Zweck, eine Datenpartition hinzuzufügen, müssen die Zugriffsrechte der Berechtigungs-ID der Anweisung Zugriffsrechte zur Verwendung des Tabellenbereichs, in dem die neue Partition hinzugefügt wird, sowie mindestens eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte für die Quellentabelle beinhalten:

- Zugriffsrecht ALTER
- Zugriffsrecht CONTROL
- Berechtigung SYSADM

- Berechtigung DBADM
- Zugriffsrecht ALTERIN für das Tabellenschema

Richtlinien für die Verwendung

- Jede Anweisung ALTER TABLE, die mit der Klausel PARTITION ausgeführt wird, muss sich einer separaten SQL-Anweisung befinden.
- In einer SQL-Anweisung mit einer Operation ALTER TABLE...PARTITION sind keine anderen ALTER-Operationen zulässig. Es ist zum Beispiel nicht möglich, in derselben SQL-Anweisung einer Tabelle eine Datenpartition zuzuordnen und eine Spalte hinzuzufügen.
- Es ist möglich, im Anschluss an die Ausführung mehrerer ALTER-Anweisungen nur eine Anweisung SET INTEGRITY auszuführen.

Zum Ändern einer partitionierten Tabelle über die Befehlszeile geben Sie die Anweisung ALTER TABLE ein.

Richtlinien und Einschränkungen für das Ändern partitionierter Tabellen

In diesem Abschnitt werden die gängigsten ALTER TABLE-Aktionen behandelt und spezielle Aspekte im Hinblick auf zugeordnete Datenpartitionen und Datenpartitionen mit aufgehobener Zuordnung erläutert.

Hinzufügen oder Ändern einer Prüfung auf Integritätsbedingung oder einer Integritätsbedingung über Fremdschlüssel
Das Hinzufügen einer Prüfung auf Integritätsbedingung oder einer Integritätsbedingung über Fremdschlüssel wird mit Datenpartitionen unterstützt, die zugeordnet sind oder deren Zuordnung aufgehoben wurde.

Hinzufügen einer Spalte
Wenn eine Spalte einer Tabelle mit zugeordneten Datenpartitionen hinzugefügt wird, wird die Spalte auch den zugeordneten Datenpartitionen hinzugefügt. Wenn eine Spalte einer Tabelle, die Datenpartitionen mit aufgehobener Zuordnung hat, hinzugefügt wird, wird die Spalte den Datenpartitionen mit aufgehobener Zuordnung nicht hinzugefügt, weil diese Datenpartitionen der Tabelle nicht mehr physisch zugeordnet sind.

Ändern einer Spalte
Wenn eine Spalte in einer Tabelle mit zugeordneten Datenpartitionen geändert wird, wird die Spalte auch in den zugeordneten Datenpartitionen geändert. Wenn eine Spalte in einer Tabelle, die Datenpartitionen mit aufgehobener Zuordnung hat, geändert wird, wird die Spalte in den Datenpartitionen mit aufgehobener Zuordnung nicht geändert, weil diese Datenpartitionen der Tabelle nicht mehr physisch zugeordnet sind.

Hinzufügen einer generierten Spalte
Wenn eine generierte Spalte einer partitionierten Tabelle mit zugeordneten Datenpartitionen oder Datenpartitionen mit aufgehobener Zuordnung hinzugefügt wird, müssen dabei die Regeln zum Hinzufügen aller anderen Typen von Spalten berücksichtigt werden.

Hinzufügen oder Ändern eines Indexes
Wenn ein Index für eine Tabelle mit hinzugefügten Datenpartitionen erstellt, erneut erstellt oder reorganisiert wird, schließt der Index die Daten in den zugeordneten Datenpartitionen nicht mit ein, weil alle Indizes für sämtliche zugeordneten Datenpartitionen durch die Anweisung SET INTEGRITY gewartet werden. Wenn ein Index für eine Tabelle mit Datenpartiti-

onen, deren Zuordnung aufgehoben ist, erstellt, erneut erstellt oder reorganisiert wird, schließt der Index die Datenpartitionen mit aufgehobener Zuordnung nur dann ein, wenn eine Datenpartition mit aufgehobener Zuordnung freigegebene abhängige Tabellen oder Zwischenspeichertabellen hat, die im Hinblick auf die Datenpartition inkrementell aktualisiert werden müssen. In diesem Fall enthält der Index die Daten für diese Datenpartition mit aufgehobener Zuordnung.

WITH EMPTY TABLE
Sie können eine Tabelle mit zugeordneten Datenpartitionen nicht leeren.

ADD MATERIALIZED QUERY AS
Das Ändern einer Tabelle mit zugeordneten Datenpartitionen in eine MQT ist nicht zulässig.

Ändern zusätzlicher Tabellenattribute, die in einer Datenpartition gespeichert sind
Die folgenden Tabellenattribute werden zusätzlich in einer Datenpartition gespeichert. Änderungen an diesen Attributen werden auf die zugeordneten Datenpartitionen, jedoch nicht auf die Datenpartitionen mit aufgehobener Zuordnung übertragen.

- DATA CAPTURE
- VALUE COMPRESSION
- APPEND
- COMPACT/LOGGED FOR LOB COLUMNS
- ACTIVATE NOT LOGGED INITIALLY (WITH EMPTY TABLE)

Zuordnen von Datenpartitionen

Die Partitionierung von Tabellen ermöglicht eine effiziente Ein- und Auslagerung von Tabellendaten (engl. Rollin und Rollout). Diese Effizienz wird mithilfe der Klauseln ATTACH PARTITION und DETACH PARTITION in der Anweisung ALTER TABLE erreicht. Das Rollin partitionierter Tabellendaten ermöglicht eine einfache Aufnahme eines neuen Datenbereichs in eine partitionierte Tabelle in Form einer zusätzlichen Datenpartition.

Zum Ändern einer Tabelle zu dem Zweck, eine Datenpartition zuzuordnen, müssen die Zugriffsrechte der Berechtigungs-ID der Anweisung mindestens eine der folgenden Berechtigungen und eines der folgenden Zugriffsrechte für die Quellentabelle beinhalten:

- Zugriffsrecht SELECT für die Quellentabelle und Zugriffsrecht DROPIN für das Schema der Quellentabelle
- Zugriffsrecht CONTROL für die Quellentabelle
- Berechtigung SYSADM oder DBADM

Die Klausel ATTACH PARTITION nimmt eine vorhandene Tabelle (Quellentabelle) und ordnet sie als neue Datenpartition der Zieltabelle zu. Die neu zugeordnete Datenpartition ist anfangs für Abfragen nicht zugänglich. Der Rest der Tabelle bleibt online. Um die zugeordnete Datenpartition online verfügbar zu machen, muss die Anweisung SET INTEGRITY ausgeführt werden.

Einschränkungen und Verwendungshinweise

Die folgenden Bedingungen müssen erfüllt sein, bevor Sie eine Datenpartition zuordnen können:

- Die Tabelle, der Sie die neue Datenpartition zuordnen wollen, (d. h. die Zieltabelle) muss eine vorhandene partitionierte Tabelle sein.
- Die Quellentabelle muss eine vorhandene, nicht partitionierte Tabelle oder eine partitionierte Tabelle mit nur einer Datenpartition sein, der keine Datenpartitionen mit ATTACH zugeordnet bzw. von der keine Datenpartitionen mit DETACH getrennt wurden. Zur Zuordnung mehrerer Datenpartitionen ist es erforderlich, mehrere ATTACH-Anweisungen abzusetzen.
- Die Quellentabelle darf keine hierarchische (typisierte) Tabelle sein.
- Die Quellentabelle darf keine Bereichsclustertabelle (RCT - Range-Clustered Table) sein.
- Die Tabellendefinition für eine Quellentabelle muss mit der Zieltabelle übereinstimmen.
- Die Anzahl, der Typ und die Anordnung von Spalten muss zwischen Quellen- und Zieltabelle übereinstimmen.
- Für beide Tabellen müssen Spalten in der Hinsicht übereinstimmen, ob sie Standardwerte enthalten. Wenn die Quellenspalte mit ALTER TABLE ADD COLUMN erstellt wurde, d. h. es gilt SYSCOLUMNS.ADD_DEFAULT = 'Y', muss der existDefault-Wert (SYSCOLUMNS.ADDED_DEFAULT) mit dem der Zielspalte übereinstimmen.
- Für beide Tabellen müssen Spalten in der Hinsicht übereinstimmen, ob Sie NULL zulassen oder nicht.
- Die COMPRESSION-Werte, d. h. sowohl für VALUE COMPRESSION als auch für SYSTEM COMPRESSION DEFAULT müssen für die Quellen- und Zieltabelle übereinstimmen.
- Die Verwendung der Klausel APPEND mit den Optionen DATA CAPTURE und NOT LOGGED INITIALLY muss übereinstimmen.
- Die Zuordnung einer Datenpartition ist zulässig, selbst wenn die Zielspalte eine generierte Spalte ist, während die Quellenspalte keine generierte Spalte ist. Die Anweisung SET INTEGRITY FOR T ALLOW WRITE ACCESS IMMEDIATE CHECKED FORCE GENERATED generiert die Werte für die generierte Spalte der zugeordneten Zeilen. Die Spalte, die einer generierten Spalte entspricht, muss in Typ und Optionalität der Dateneingabe übereinstimmen. Es gibt keine erforderlichen Standardwerte für diese Spalte. Das empfohlene Verfahren besteht darin, sicherzustellen, dass die Quellentabelle der ATTACH-Zuordnung den korrekten generierten Wert in der generierten Spalte enthält. In diesem Fall brauchen Sie die Option FORCE GENERATED nicht zu verwenden. Sie können die folgende Anweisung verwenden:

  ```
  SET INTEGRITY FOR T GENERATED COLUMN IMMEDIATE UNCHECKED
  (Weist das System an, die Prüfung der generierten Spalte zu übergehen.)
  SET INTEGRITY FOR T ALLOW WRITE ACCESS IMMEDIATE CHECKED FOR EXCEPTION
  IN T USE T_EX (Führt eine Integritätsprüfung der zugeordneten Partition aus,
  prüft jedoch nicht auf Korrektheit der generierten Spalte)
  ```

- Die Zuordnung einer Datenpartition ist auch zulässig, wenn die Zielspalte eine Identitätsspalte (IDENTITY) und die Quellenspalte keine Identitätsspalte ist. Die Anweisung SET INTEGRITY IMMEDIATE CHECKED generiert keine Identitätswerte für die zugeordneten Zeilen. Die Anweisung SET INTEGRITY FOR T GENERATE IDENTITY ALLOW WRITE ACCESS IMMEDIATE CHECKED fügt die Identitätswerte für die zugeordneten Zeilen ein. Die Spalte, die einer Identitätsspalte entspricht, muss in Typ und Optionalität der Dateneingabe übereinstimmen. Die Standardwerte dieser Spalte unterliegen keiner bestimmten Anforderung. Das empfohlene Verfahren besteht darin, dass die korrekten Identitätswerte in die Zwischenspeichertabelle einzufügen. In diesem Fall brauchen Sie im

Anschluss an die ATTACH-Zuordnung die Option GENERATE IDENTITY nicht zu verwenden, weil die Identitätswerte in der Quellentabelle bereits garantiert sind.

- Bei Tabellen, deren Daten über mehrere Datenbankpartitionen verteilt sind, muss die Quellentabelle ebenfalls in derselben Datenbankpartitionsgruppe mit demselben Verteilungsschlüssel und derselben Verteilungszuordnung verteilt sein.
- Die Quellentabelle muss löschbar sein (d. h., RESTRICT DROP darf für sie nicht angegeben sein).
- Wenn ein Name mit DATAPARTITIONNAME angegeben wird, darf dieser in der Zieltabelle nicht bereits vorhanden sein.
- Wenn die Zieltabelle eine MDC-Tabelle (MDC, Multi-Dimensional Clustering) ist, muss die Quellentabelle ebenfalls eine MDC-Tabelle sein.
- Der Datentabellenbereich für die Quellentabelle muss mit den Datentabellenbereichen für die Zieltabelle in Typ (d. h. DMS oder SMS), Seitengröße (PAGESIZE), Wert für EXTENTSIZE und Datenbankpartitionsgruppe übereinstimmen. Der Benutzer empfängt eine Warnung, wenn die Werte für PREFETCHSIZE nicht übereinstimmen. Der Tabellenbereich für große Objektdaten der Quellentabelle muss mit den Tabellenbereichen für große Objektdaten der Zieltabelle in Typ, Datenbankpartitionsgruppe und Seitengröße übereinstimmen.

Wenn Sie die DB2-Befehlszeile verwenden wollen, um eine partitionierte Tabelle zu ändern und der Tabelle eine Datenpartition zuzuordnen, müssen Sie die Anweisung ALTER TABLE ausführen.

Richtlinien zum Hinzufügen von Datenpartitionen zu partitionierten Tabellen

In diesem Abschnitt finden Sie Richtlinien für die Korrektur verschiedener Typen von Merkmalsabweichungen, die auftreten können, wenn Sie versuchen, eine Datenpartition einer partitionierten Tabelle mithilfe der Anweisung ALTER TABLE ...ATTACH PARTITION zuzuordnen. Sie können eine Übereinstimmung zwischen Tabellen erreichen, indem Sie die Quellentabelle an den Merkmalen der Zieltabelle ausrichten oder indem Sie die Zieltabelle an den Merkmalen der Quellentabelle ausrichten.

Die Quellentabelle ist die vorhandene Tabelle, der Sie die Zieltabelle zuordnen wollen. Die Zieltabelle ist die Tabelle, der Sie die neue Datenpartition zuordnen wollen.

Ein vorgeschlagenes Verfahren für eine erfolgreiche Zuordnung ist die Anwendung genau jener Anweisung CREATE TABLE, die Sie für die Zieltabelle verwendet haben, auf die Quellentabelle, jedoch ohne die Klausel PARTITION BY. In Fällen, in denen es schwierig ist, die Merkmale entweder der Quellentabelle oder der Zieltabelle zu modifizieren und Kompatibilität zu erreichen, können Sie eine neue Quellentabelle erstellen, die mit der Zieltabelle kompatibel ist. Detaillierte Informationen zum Erstellen einer neuen Quelle finden Sie in *Erstellen von Tabellen nach vorhandenen Tabellen*.

Informationen zur Vermeidung abweichender Merkmale finden im Abschnitt 'Einschränkungen und Verwendungshinweise' in „Zuordnen von Datenpartitionen" auf Seite 221. Der Abschnitt enthält einen Überblick über die Bedingungen, die erfüllt sein müssen, bevor Sie eine Datenpartition erfolgreich zuordnen können. Wenn die aufgeführten Bedingungen nicht erfüllt sind, wird der Fehler SQL20408N oder SQL20307N zurückgegeben.

In den folgenden Abschnitten werden die verschiedenen Typen von Abweichungen beschrieben, die auftreten können, und Schritte zur Herstellung übereinstimmender Merkmale zwischen Tabellen vorgeschlagen:

Die Klausel (VALUE) COMPRESSION (Spalte COMPRESSION in SYSCAT.TABLES) stimmt nicht überein. (SQL20307N Ursachencode 2)

Verwenden Sie eine der folgenden Anweisungen, um Übereinstimmung in der Wertkomprimierung herzustellen:

```
ALTER TABLE... ACTIVATE VALUE COMPRESSION
oder
ALTER TABLE... DEACTIVATE VALUE COMPRESSION
```

Verwenden Sie eine der folgenden Anweisungen, um Übereinstimmung in der Zeilenkomprimierung herzustellen:

```
ALTER TABLE... COMPRESS YES
oder
ALTER TABLE... COMPRESS NO
```

Der Modus APPEND der Tabellen stimmt nicht überein. (SQL20307N Ursachencode 3)

Verwenden Sie eine der folgenden Anweisungen, um Übereinstimmung im APPEND-Modus herzustellen:

```
ALTER TABLE ... APPEND ON
oder
ALTER TABLE ... APPEND OFF
```

Die Codepages der Quellen- und Zieltabelle stimmen nicht überein. (SQL20307N Ursachencode 4)

Erstellen Sie eine neue Quellentabelle.

Die Quellentabelle ist eine partitionierte Tabelle mit mehreren Datenpartitionen oder mit Datenpartitionen, die zugeordnet wurden bzw. deren Zuordnung aufgehoben wurde. (SQL20307N Ursachencode 5)

Heben Sie mit der folgenden Anweisung so viele Zuordnungen von Datenpartitionen zur Quellentabelle auf, dass nur eine sichtbare Datenpartition vorhanden ist:

```
ALTER TABLE ... DETACH PARTITION
```

Fügen Sie alle erforderlichen Anweisungen SET INTEGRITY hinzu. Wenn die Quellentabelle Indizes besitzt, kann die Quellentabelle möglicherweise nicht sofort zugeordnet werden. Datenpartitionen, deren Zuordnung aufgehoben wurde, bleiben unzugeordnet, bis alle Indizes in Bezug auf Schlüssel, deren Zuordnung aufgehoben wurde, bereinigt wurden. Wenn Sie die Tabelle unverzüglich zuordnen wollen, löschen Sie den Index für die Quellentabelle. Ansonsten erstellen Sie eine neue Quellentabelle.

Die Quellentabelle ist eine Systemtabelle, eine Sicht, eine typisierte Tabelle, eine mit ORGANIZED BY KEY SEQUENCE definierte Tabelle oder eine deklarierte globale temporäre Tabelle. (SQL20307N Ursachencode 6)

Erstellen Sie eine neue Quellentabelle.

Die Zieltabelle und die Quellentabelle sind identisch. (SQL20307N Ursachencode 7)

Sie können keine Tabelle sich selbst zuordnen. Bestimmen Sie die richtige Tabelle, die als Quellen- oder Zieltabelle zu verwenden ist.

Die Klausel NOT LOGGED INITIALLY wurde für die Quellentabelle oder für die Zieltabelle angegeben, jedoch nicht für beide. (SQL20307N Ursachencode 8)

Machen Sie die Tabelle, die mit der Klausel NOT LOGGED INITIALLY definiert wurde, zu einer protokollierten Tabelle, indem Sie die Anweisung COMMIT ausführen. Oder machen Sie die Tabelle, die ohne die Klausel NOT LOGGED INITIALLY definiert wurde, zu einer Tabelle mit NOT LOGGED INITIALLY, indem Sie die folgende Anweisung eingeben:

```
ALTER TABLE .... ACTIVATE NOT LOGGED INITIALLY
```

Die Klausel DATA CAPTURE CHANGES wurde entweder für die Quellentabelle oder für die Zieltabelle, jedoch nicht für beide angegeben. (SQL20307N Ursachencode 9)

Führen Sie die folgende Anweisung aus, um der Tabelle, die ohne Klausel DATA CAPTURE CHANGES definiert wurde, diese Klausel hinzuzufügen:

```
ALTER TABLE ... DATA CAPTURE CHANGES
```

Führen Sie die folgende Anweisung aus, um für die Tabelle mit der Klausel DATA CAPTURE CHANGES die Datenerfassung zu inaktivieren:

```
ALTER TABLE ... DATA CAPTURE NONE
```

Die Klauseln DISTRIBUTION der Tabellen stimmen nicht überein. Der Verteilungsschlüssel muss für die Quellen- und Zieltabelle gleich sein. (SQL20307N Ursachencode 10)

Es wird empfohlen, eine neue Quellentabelle zu erstellen. Sie können den Verteilungsschlüssel einer Tabelle, die sich über mehrere Datenbankpartitionen erstreckt, nicht ändern. Führen Sie die folgenden Anweisungen aus, um einen Verteilungsschlüssel für Tabellen in einer Einzelpartitionsdatenbank zu ändern:

```
ALTER TABLE ... DROP DISTRIBUTION;
       ALTER TABLE ... ADD DISTRIBUTION(schlüsselspezifikation)
```

Nur für eine der Tabellen wurde eine Klausel ORGANIZE BY DIMENSIONS angegeben oder die Organisationsdimensionen sind verschieden. (SQL20307N Ursachencode 11)

Erstellen Sie eine neue Quellentabelle.

Der Datentyp der Spalten (TYPENAME) stimmt nicht überein. (SQL20408N Ursachencode 1)

Führen Sie die folgende Anweisung aus, um abweichende Datentypen zu korrigieren:

```
ALTER TABLE ... ALTER COLUMN ... SET DATA TYPE...
```

Die Optionalität der Dateneingabe der Spalten (NULL-Werte) stimmt nicht überein. (SQL20408N Ursachencode 2)

Führen Sie eine der folgenden Anweisungen aus, um die Optionalität der Dateneingabe der nicht übereinstimmenden Spalte in einer der Tabellen zu ändern:

```
ALTER TABLE... ALTER COLUMN... DROP NOT NULL
oder
ALTER TABLE... ALTER COLUMN... SET NOT NULL
```

Der implizite Standardwert (SYSCAT.COLUMNS IMPLICITVALUE) der Spalten ist inkompatibel. (SQL20408N Ursachencode 3)

Erstellen Sie eine neue Quellentabelle. Implizite Standardwerte müssen exakt übereinstimmen, wenn sowohl die Zieltabellenspalte als auch die Quellentabellenspalte implizite Standardwerte besitzen (d. h. wenn IMPLICITVALUE nicht gleich NULL ist).

Wenn IMPLICITVALUE für eine Spalte in der Zieltabelle nicht gleich NULL ist und IMPLICITVALUE für die entsprechende Spalte der Quellentabelle nicht gleich NULL ist, wurde jede der Spalten erst nach der Ausführung der ursprünglichen Anweisung CREATE TABLE für die Tabelle hinzugefügt. In diesem Fall müssen die in IMPLICITVALUE gespeicherten Werte für diese Spalte übereinstimmen.

Es ist möglich, dass durch die Migration einer Tabelle von einer Version vor Version 9.1 oder durch das Zuordnen einer Datenpartition aus einer Tabelle vor Version 9.1 der Wert von IMPLICITVALUE nicht NULL ist, weil das System nicht ermitteln konnte, ob die Spalte nach Ausführung der ursprünglichen Anweisung CREATE TABLE hinzugefügt wurde oder nicht. Wenn die Datenbank nicht sicher ermitteln kann, ob die Spalte hinzugefügt wurde oder nicht, wird die Spalte wie eine hinzugefügte Spalte behandelt. Eine hinzugefügte Spalte ist eine Spalte, die durch eine Anweisung ALTER TABLE ...ADD COLUMN erstellt wurde. In diesem Fall ist die Anweisung nicht zulässig, weil der Wert der Spalte ungültig werden könnte, wenn eine Fortsetzung der Zuordnung zugelassen würde. Sie müssen die Daten aus der Quellentabelle in eine neue Tabelle (deren IMPLICITVALUE-Wert für diese Spalte NULL ist) kopieren und die neue Tabelle als Quellentabelle für die Zuordnungsoperation verwenden.

Die Codepage (COMPOSITE_CODEPAGE) der Spalten stimmt nicht überein. (SQL20408N Ursachencode 4)

Erstellen Sie eine neue Quellentabelle.

Die Standardklausel für Systemkomprimierung (COMPRESS) stimmt nicht überein. (SQL20408N Ursachencode 5)

Führen Sie eine der folgenden Anweisungen aus, um die Komprimierung von Systemstandardwerten in der Spalte zu ändern und die Abweichung zu beheben:

```
ALTER TABLE ... ALTER COLUMN ... COMPRESS SYSTEM DEFAULT
oder
ALTER TABLE ... ALTER COLUMN ... COMPRESS OFF
```

Aufheben der Zuordnung von Datenpartitionen

Die Partitionierung von Tabellen ermöglicht eine effiziente Ein- und Auslagerung von Tabellendaten (engl. Rollin und Rollout). Diese Effizienz wird mithilfe der Klauseln ATTACH PARTITION und DETACH PARTITION in der Anweisung ALTER TABLE erreicht.

Zur Aufhebung der Zuordnung einer Datenpartition zu einer partitionierten Tabelle müssen Sie über die folgenden Berechtigungen und Zugriffsrechte verfügen:

- Der Benutzer, der die DETACH-Operation ausführt, muss über die Berechtigung verfügen, die erforderlich ist, um die Quellentabelle zu ändern (ALTER), Daten aus der Quellentabelle auszuwählen (SELECT) und die Quellentabelle zu löschen (DELETE).
- Der Benutzer muss außerdem über die Berechtigung verfügen, die zum Erstellen (CREATE) der Zieltabelle erforderlich ist. Daher müssen zum Ändern einer Tabelle für die Aufhebung der Zuordnung einer Datenpartition die Zugriffsberechtigungen der Berechtigungs-ID der Anweisung mindestens eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte für die Zieltabelle beinhalten:
  - Berechtigung SYSADM oder DBADM
  - Berechtigung CREATETAB für die Datenbank und das Zugriffsrecht USE für die von der Tabelle verwendeten Tabellenbereiche sowie eine der folgenden Berechtigungen (bzw. Zugriffsrechte):
    - Berechtigung IMPLICIT_SCHEMA für die Datenbank, wenn der implizite oder explizite Schemaname der Tabelle nicht existiert
    - Zugriffsrecht CREATEIN für das Schema, wenn sich der Schemaname der Tabelle auf ein vorhandenes Schema bezieht

Anmerkung: Wenn die Zuordnung einer Datenpartition aufgehoben wird, wird die Anweisung CREATE TABLE effektiv von der Berechtigungs-ID der Anweisung ausgeführt. Daher muss diese Berechtigungs-ID über die erforderlichen Zugriffsrechte zur Ausführung der Operation verfügen. Die Berechtigungs-ID der Anweisung ALTER TABLE wird zum definierenden Benutzer der neuen Tabelle mit der Berechtigung CONTROL, so als ob der Benutzer die Anweisung CREATE TABLE abgesetzt hätte. Von der Tabelle, die geändert wird, werden keine Zugriffsrechte auf die neue Tabelle übertragen. Nur die Berechtigungs-ID der Anweisung ALTER TABLE sowie die Berechtigungen DBADM bzw. SYSADM haben unmittelbar nach der Ausführung der Anweisung ALTER TABLE ... DETACH PARTITION Zugriff auf die Daten.

Das Rollout partitionierter Tabellendaten ermöglicht eine einfache Abtrennung von Datenbereichen von einer partitionierten Tabelle. Wenn die Zuordnung einer Datenpartition aufgehoben (DETACH PARTITION) und die Datenpartition in eine separate Tabelle umgewandelt wird, kann die Tabelle auf verschiedene Arten gehandhabt werden. Sie können die separate Tabelle löschen (wodurch die Daten in der Datenpartition vernichtet werden). Sie können die Tabelle archivieren oder auf andere Weise als separate Tabelle nutzen. Sie können sie einer anderen partitionierten Tabelle zuordnen (ATTACH PARTITION), zum Beispiel einer Protokolltabelle. Oder Sie können sie bearbeiten, bereinigen, umwandeln und wieder der ursprünglichen oder einer anderen partitionierten Tabelle zuordnen.

Einschränkungen

Wenn es sich bei der Quellentabelle um eine MDC-Tabelle (MDC - mehrdimensionales Clustering) handelt, wird der Zugriff auf eine soeben durch DETACH getrennte Tabelle nicht in derselben UOW zugelassen, die auch die Operation ALTER TABLE ...DETACH ausführt. Blockindizes werden beim ersten Zugriff auf die Tabelle erstellt, nachdem die Operation ALTER TABLE ...DETACH festgeschrieben wurde. Die Zugriffszeit ist während der Erstellung der Blockindizes reduziert.

Die folgenden Bedingungen müssen erfüllt sein, damit eine DETACH-Operation ausgeführt werden kann:

- Die Tabelle, zu der die Zuordnung einer Datenpartition aufgehoben werden soll (Quellentabelle), muss vorhanden und eine partitionierte Tabelle sein.
- Die Datenpartition, deren Zuordnung aufgehoben werden soll, muss in der Quellentabelle vorhanden sein.
- Die Quellentabelle muss mehr als eine Datenpartition haben. Eine partitionierte Tabelle muss mindestens eine Datenpartition haben. Dieser Kontext bezieht sich nur auf sichtbare und zugeordnete Datenpartitionen. Eine zugeordnete Datenpartition ist eine Datenpartition, die zwar zugeordnet ist, jedoch noch nicht durch die Anweisung SET INTEGRITY geprüft wurde.
- Der Name der Tabelle, die durch die DETACH-Operation erstellt werden soll (Zieltabelle), darf noch nicht vorhanden sein.
- Eine DETACH-Operation ist für eine Tabelle, die eine übergeordnete Tabelle einer erzwungenen referenziellen Integritätsbeziehung (RI) ist, nicht zulässig.
- Wenn abhängige Tabellen vorhanden sind, die im Hinblick auf die Datenpartition mit aufgehobener Zuordnung inkrementell gewartet werden müssen (solche abhängigen Tabellen werden als freigegebene abhängige Tabellen bezeichnet), ist zunächst kein Zugriff auf die neue Tabelle mit aufgehobener Zuordnung möglich. Die Tabelle wird mit dem Wert L in der Spalte TYPE der Katalogsicht SYSCAT.TABLES markiert. Dies kennzeichnet sie als Tabelle mit aufgehobener Zuordnung (bzw. als freigegebene Tabelle). Dadurch wird verhindert, dass die Tabelle gelesen, modifiziert oder gelöscht wird, bevor die Anweisung SET INTEGRITY ausgeführt wird, um die freigegebenen abhängigen Tabellen inkrementell zu warten. Nach der Ausführung der Anweisung SET INTEGRITY für alle freigegebenen abhängigen Tabellen geht die Tabelle mit aufgehobener Zuordnung in den Status einer regulären Tabelle über, sodass ein vollständiger Zugriff auf sie möglich wird.

Zum Ändern einer partitionierten Tabelle und zum Aufheben der Zuordnung einer Datenpartition zu dieser Tabelle über die Befehlszeile geben Sie die Anweisung ALTER TABLE mit der Klausel DETACH PARTITION ein.

Attribute von Datenpartitionen mit aufgehobener Zuordnung

Wenn Sie die Zuordnung einer Datenpartition zu einer partitionierten Tabelle mit der Klausel DETACH PARTITION der Anweisung ALTER TABLE aufheben, wird diese Datenpartition zu einer eigenständigen Zieltabelle. Viele Attribute der neuen Zieltabelle werden aus der Quellentabelle übernommen. Attribute, die nicht aus der Quellentabelle übernommen werden, werden so definiert, als ob der Benutzer, der die DETACH-Operation ausführt, die Zieltabelle erstellen würde.

Anmerkung: Wenn freigegebene abhängige Tabellen vorhanden sind, wird die Datenpartition, deren Zuordnung aufgehoben wurde, durch die Aufhebung der Zuordnung nicht zu einer eigenständigen Tabelle. In diesem Fall muss die Anweisung SET INTEGRITY ausgeführt werden, um die Aufhebung der Zuordnung abzuschließen und den Zugriff auf die Tabelle zu ermöglichen.

Übernommene Attribute der Zieltabelle

Die folgenden Attribute werden durch die Zieltabelle übernommen:

- Die folgenden Spaltendefinitionen:
  - Spaltenname

  - Datentyp (mit Länge und Genauigkeit bei Datentypen, die eine Länge und Genauigkeit besitzen, z. B. CHAR und DECIMAL)
  - Optionalität der Dateneingabe (NULLABLE)
  - Spaltenstandardwerte
  - Codepage (Spalte CODEPAGE der Katalogsicht SYSCAT.COLUMNS)
  - Protokollierung für LOBs (Spalte LOGGED der Katalogsicht SYSCAT.COLUMNS)
  - Kompaktheit für LOBs (Spalte COMPACT der Katalogsicht SYSCAT.COLUMNS)
  - Komprimierung (Spalte COMPRESS der Katalogsicht SYSCAT.COLUMNS)
  - Typ von verdeckter Spalte (Spalte HIDDEN der Katalogsicht SYSCAT.COLUMNS)
  - Spaltenreihenfolge
- Wenn die Quellentabelle eine MDC-Tabelle (Multidimensional Clustering Table - mehrdimensionale Clusteringtabelle) ist, ist die Zieltabelle ebenfalls eine MDC-Tabelle, die mit den gleichen Dimensionsspalten definiert ist. Der Zugriff auf die neue Tabelle mit aufgehobener Zuordnung wird nicht in derselben UOW zugelassen, in der die Aufhebung der Zuordnung erfolgt, wenn die Quellentabelle eine MDC-Tabelle ist.
- Blockindexdefinitionen. Die Indizes werden beim ersten Zugriff auf die neue unabhängige Tabelle mit aufgehobener Zuordnung erneut erstellt, nachdem die DETACH-Operation festgeschrieben wurde.
- Die Tabellenbereichs-ID und die Tabellenobjekt-ID werden aus der Datenpartition, nicht aus der Quellentabelle übernommen. Dies liegt daran, dass während einer DETACH-Operation keine Tabellendaten versetzt werden. In Bezug auf den Katalog wird die Spalte TBSPACEID der Katalogsicht SYSCAT.DATAPARTITIONS aus der Quellendatenpartition zur Spalte TBSPACEID der Katalogsicht SYSCAT.TABLES. Übersetzt in einen Tabellenbereichsnamen ist dies die Spalte TBSPACE der Katalogsicht SYSCAT.TABLES in der Zieltabelle. Die Spalte PARTITIONOBJECTID der Katalogsicht SYSCAT.DATAPARTITIONS aus der Quellendatenpartition wird zur Spalte TABLEID der Katalogsicht SYSCAT.TABLES in der Zieltabelle.
- Die Spalte LONG_TBSPACEID der Katalogsicht SYSCAT.DATAPARTITIONS aus der Quellendatenpartition wird in einen Tabellenbereichsnamen übersetzt und wird zur Spalte LONG_TBSPACE der Katalogsicht SYSCAT.TABLES der Zieltabelle.
- Tabellenbereichsposition
- ID der Verteilungszuordnung für eine Mehrpartitionsdatenbank (Spalte PMAP_ID der Katalogsicht SYSCAT.TABLES)
- Freizuhaltender Speicherbereich (Spalte PCTFREE der Katalogsicht SYSCAT.TABLES)
- Anfügemodus (Spalte APPEND_MODE der Katalogsicht SYSCAT.TABLES)
- Bevorzugte Sperrgranularität (Spalte LOCKSIZE der Katalogsicht SYSCAT.TABLES)
- Datenerfassung (Spalte DATA_CAPTURE der Katalogsicht SYSCAT.TABLES)
- VOLATILE (Spalte VOLATILE der Katalogsicht SYSCAT.TABLES)
- DROPRULE (Spalte DROPRULE der Katalogsicht SYSCAT.TABLES)
- Komprimierung (Spalte COMPRESSION der Katalogsicht SYSCAT.TABLES)
- MAXFREESPACESEARCH (Spalte MAXFREESPACESEARCH der Katalogsicht SYSCAT.TABLES)

Anmerkung: Partitionierte hierarchische oder temporäre Tabellen, Bereichsclustertabellen und partitionierte Sichten werden nicht unterstützt.

Nicht aus der Quellentabelle übernommene Attribute

Die folgenden Attribute werden nicht aus der Quellentabelle übernommen:

- Der Typ der Zieltabelle wird nicht übernommen. Die Zieltabelle ist in jedem Fall eine reguläre Tabelle.
- Zugriffsrechte und Berechtigungen.
- Schema.
- Generierte Spalten, Identitätsspalten, Prüfungen auf Integritätsbedingung, referenzielle Integritätsbedingungen. In dem Fall, dass eine Quellenspalte eine generierte Spalte oder eine Identitätsspalte ist, hat die entsprechende Zielspalte keinen expliziten Standardwert, d. h., der Standardwert ist NULL.
- Indextabellenbereich (Spalte INDEX_TBSPACE der Katalogsicht SYSCAT.TABLES). Der Wert wird auf NULL gesetzt. Indizes für die Tabelle, die aus der DETACH-Operation resultieren, befinden sich im selben Tabellenbereich wie die Tabelle.
- Trigger
- Primärschlüssel
- Statistiken
- Alle anderen Attribute, die nicht explizit in der Liste der Attribute, die aus der Quellentabelle übernommen werden, aufgeführt sind.

Hinzufügen von Datenpartitionen zu partitionierten Tabellen

Mithilfe der Anweisung ALTER TABLE können Sie eine partitionierte Tabelle nach ihrer Erstellung ändern. Insbesondere können Sie einer vorhandenen partitionierten Tabelle mithilfe der Klausel ADD PARTITION eine neue Datenpartition hinzufügen. Das Hinzufügen einer Datenpartition eignet sich in solchen Fällen besser als das Zuordnen einer Datenpartition, in denen Daten der Datenpartition über einen Zeitraum hinweg verteilt hinzugefügt werden, Daten eher in kleinen Mengen als in großen Ladungen (Rollin) aus einer externen Quelle eintreffen oder Daten direkt in eine partitionierte Tabelle eingefügt oder geladen werden. Beispiele für solche Fälle sind tägliche Operationen zum Laden von Daten in eine Datenpartition für Januardaten oder auch fortlaufende Einfügungen einzelner Zeilen.

Einschränkungen und Verwendungshinweise

- Einer nicht partitionierten Tabelle kann keine Datenpartition hinzugefügt werden. Detaillierte Informationen zur Migration einer vorhandenen Tabelle in eine partitionierte Tabelle finden Sie in „Migrieren vorhandener Tabellen und Sichten in partitionierte Tabellen" auf Seite 200.
- Der Bereich von Werten für jede neue Datenpartition wird durch die Klauseln STARTING und ENDING bestimmt.
- Es muss mindestens eine der Klauseln STARTING und ENDING (oder beide) angegeben werden.
- Der neue Bereich darf sich nicht mit dem Bereich einer vorhandenen Datenpartition überschneiden.
- Wenn eine neue Datenpartition vor der ersten vorhandenen Datenpartition hinzugefügt wird, muss die Klausel STARTING angegeben werden. Mithilfe von MINVALUE können Sie diesen Bereich mit offenem Ende definieren.

- Analog muss die Klausel ENDING angegeben werden, wenn eine neue Datenpartition nach der letzten vorhandenen Datenpartition hinzugefügt werden soll. Mithilfe von MAXVALUE können Sie diesen Bereich mit offenem Ende definieren.
- Wenn die Klausel STARTING nicht angegeben wird, generiert die Datenbank eine Startgrenze, die direkt hinter der Endgrenze der vorherigen Datenpartition liegt. Wenn die Klausel ENDING nicht angegeben wird, generiert die Datenbank analog eine Endgrenze, die direkt vor der Startgrenze der nächsten Datenpartition liegt.
- Die Syntax für die Startklausel und die Endklausel ist die gleiche, die auch in der Anweisung CREATE TABLE zu verwenden ist.
- Wenn keine Klausel IN oder LONG IN für ADD PARTITION angegeben wird, wird der Tabellenbereich, in dem die Datenpartition zu speichern ist, mit der gleichen Methode ausgewählt wie bei der Anweisung CREATE TABLE.
- Pakete werden bei der Operation ALTER TABLE ...ADD PARTITION ungültig gemacht.
- Die neu hinzugefügte Datenpartition wird verfügbar, nachdem die Anweisung ALTER TABLE festgeschrieben wurde.

Das Weglassen der STARTING- oder ENDING-Grenzen für eine ADD- bzw. ATTACH-Operation wird auch dazu genutzt, eine Lücke in Bereichswerten zu füllen. Das folgende Beispiel zeigt, wie eine Lücke mit einer ADD-Operation gefüllt wird, in der nur die STARTING-Grenze angegeben wird:

```
CREATE TABLE hole (c1 int) PARTITION BY RANGE (c1)
(STARTING FROM 1 ENDING AT 10, STARTING FROM 20 ENDING AT 30);
DB20000I  Der SQL-Befehl wurde erfolgreich ausgeführt.

ALTER TABLE hole ADD PARTITION STARTING 15;
DB20000I  Der SQL-Befehl wurde erfolgreich ausgeführt.

SELECT SUBSTR(tabname, 1,12) tabname,
SUBSTR(datapartitionname, 1, 12) datapartitionname,
seqno, SUBSTR(lowvalue, 1, 4) lowvalue, SUBSTR(highvalue, 1, 4) highvalue
FROM SYSCAT.DATAPARTITIONS WHERE TABNAME='HOLE' ORDER BY seqno;

TABNAME DATAPARTITIONNAME SEQNO LOWVALUE HIGHVALUE
------------ ---------------- ----------- -------- ---------
HOLE PART0 0 1 10
HOLE PART2 1 15 20
HOLE PART1 2 20 30

3 Satz/Sätze ausgewählt.
```

Wenn Sie eine partitionierte Tabelle ändern und der Tabelle eine neue Datenpartition hinzufügen wollen, geben Sie in die Befehlszeile die Anweisung ALTER TABLE mit der Klausel ADD PARTITION ein.

Beispiel 1: Sie fügen einer vorhandenen partitionierten Tabelle eine Datenpartition hinzu, die einen Bereich mit Werten von 901 bis 1000 einschließlich enthält. Nehmen Sie an, die Tabelle 'sales' enthält neun Bereiche 0-100, 101-200 usw. bis zum Wert 900. In diesem Beispiel wird ein zusätzlicher Bereich am Ende der Tabelle hinzugefügt, wie durch das Weglassen der Klausel STARTING angegeben wird:

```
 ALTER TABLE sales ADD PARTITION dp10
(ENDING AT 1000 INCLUSIVE)
```

Löschen von Datenpartitionen

Sie können eine Datenpartition löschen, indem Sie die Anweisung ALTER TABLE mit der Klausel DETACH PARTITION verwenden. Mit der Anweisung DROP TABLE können Sie die separate Tabelle löshen.

Zur Aufhebung der Zuordnung einer Datenpartition zu einer partitionierten Tabelle muss der Benutzer über die folgenden Berechtigungen bzw. Zugriffsrechte verfügen:

- Der Benutzer, der die DETACH-Operation ausführt, muss über die Berechtigung verfügen, die erforderlich ist, um die Quellentabelle zu ändern (ALTER), Daten aus der Quellentabelle auszuwählen (SELECT) und die Quellentabelle zu löschen (DELETE).
- Der Benutzer muss außerdem über die Berechtigung verfügen, die zum Erstellen (CREATE) der Zieltabelle erforderlich ist. Daher müssen zum Ändern einer Tabelle für die Aufhebung der Zuordnung einer Datenpartition die Zugriffsberechtigungen der Berechtigungs-ID der Anweisung mindestens eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte für die Zieltabelle beinhalten:
  - Berechtigung SYSADM oder DBADM
  - Berechtigung CREATETAB für die Datenbank und das Zugriffsrecht USE für die von der Tabelle verwendeten Tabellenbereiche sowie eine der folgenden Berechtigungen (bzw. Zugriffsrechte):
    - Berechtigung IMPLICIT_SCHEMA für die Datenbank, wenn der implizite oder explizite Schemaname der Tabelle nicht existiert
    - Zugriffsrecht CREATEIN für das Schema, wenn sich der Schemaname der Tabelle auf ein vorhandenes Schema bezieht

Zum Löschen einer Tabelle muss der Benutzer über die folgenden Berechtigungen bzw. Zugriffsrechte verfügen:

- Sie müssen entweder der definierende Benutzer sein, der in der Spalte DEFINER der Katalogsicht SYSCAT.TABLES aufgezeichnet ist, oder über mindestens eines der folgenden Zugriffsrechte verfügen:
  - Berechtigung SYSADM oder DBADM
  - Zugriffsrecht DROPIN für das Schema der Tabelle
  - Zugriffsrecht CONTROL für die Tabelle

Anmerkung: Der Fall der Aufhebung der Zuordnung einer Datenpartition impliziert, dass die Anweisung CREATE TABLE effektiv von der Berechtigungs-ID der Anweisung absetzt wird. Daher muss diese Berechtigungs-ID über die erforderlichen Zugriffsrechte zur Ausführung der Operation verfügen. Der Tabellenbereich ist derjenige, in dem sich die Datenpartition befindet, deren Zuordnung aufgehoben wird. Die Berechtigungs-ID der Anweisung ALTER TABLE wird zum definierenden Benutzer der neuen Tabelle mit der Berechtigung CONTROL, so als ob der Benutzer die Anweisung CREATE TABLE abgesetzt hätte. Von der Tabelle, die geändert wird, werden keine Zugriffsrechte auf die neue Tabelle übertragen. Nur die Berechtigungs-ID der Anweisung ALTER TABLE sowie die Berechtigungen DBADM bzw. SYSADM haben unmittelbar nach der Operation ALTER TABLE ... DETACH PARTITION Zugriff auf die Daten.

Zum Aufheben der Zuordnung einer Datenpartition einer partitionierten Tabelle geben Sie die Anweisung ALTER TABLE mit der Klausel DETACH PARTITION in die Befehlszeile ein.

Sie können eine Tabelle über den DB2-Befehlszeilenprozessor (CLP) löschen.

Zum Löschen einer Tabelle über die Befehlszeile geben Sie die Anweisung DROP TABLE ein. In folgenden Beispiel wird die Zuordnung der Datenpartition 'dec01' zur Tabelle 'stock' aufgehoben, und die Daten werden in die Tabelle 'junk' versetzt. Anschließend können Sie die Tabelle 'junk' löschen und so effektiv die entsprechende Datenpartition löschen.

```
ALTER TABLE stock DETACH PART dec01 INTO junk;
 DROP TABLE junk;
```

Anmerkung: Um die DETACH-Operation möglichst schnell auszuführen, wird die Indexbereinigung für die Quellentabelle automatisch durch einen asynchronen Hintergrundprozess zur Indexbereinigung ausgeführt. Wenn freigegebene abhängige Tabellen vorhanden sind, wird die Datenpartition, deren Zuordnung aufgehoben wurde, durch die Aufhebung der Zuordnung nicht zu einer eigenständigen Tabelle. In diesem Fall muss die Anweisung SET INTEGRITY ausgeführt werden, um die Aufhebung der Zuordnung abzuschließen und den Zugriff auf die Tabelle zu ermöglichen.

Szenario: Rotieren von Daten in einer partitionierten Tabelle

Der Begriff des Rotierens von Daten in DB2-Datenbanken bezeichnet eine Methode zur Wiederverwendung von Speicherplatz in einer Datenpartition, bei der veraltete Daten aus der Tabelle entfernt und neue Daten anschließend hinzugefügt werden. Die Funktionalität der Tabellenpartitionierung bietet Ihnen die Möglichkeit, die Zuordnung der Datenpartition mit den veralteten Daten aufzuheben und anschließend eine neue Datenpartition mit den aktuellsten Daten zuzuordnen.

Zur Aufhebung der Zuordnung einer Datenpartition zu einer partitionierten Tabelle muss der Benutzer über die folgenden Berechtigungen bzw. Zugriffsrechte verfügen:

- Der Benutzer, der die DETACH-Operation ausführt, muss über die Berechtigung verfügen, die erforderlich ist, um die Quellentabelle zu ändern (ALTER), Daten aus der Quellentabelle auszuwählen (SELECT) und die Quellentabelle zu löschen (DELETE).
- Der Benutzer muss außerdem über die Berechtigung verfügen, die zum Erstellen (CREATE) der Zieltabelle erforderlich ist. Daher müssen zum Ändern einer Tabelle für die Aufhebung der Zuordnung einer Datenpartition die Zugriffsberechtigungen der Berechtigungs-ID der Anweisung mindestens eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte für die Zieltabelle beinhalten:
  - Berechtigung SYSADM oder DBADM
  - Berechtigung CREATETAB für die Datenbank und das Zugriffsrecht USE für die von der Tabelle verwendeten Tabellenbereiche sowie eine der folgenden Berechtigungen (bzw. Zugriffsrechte):
    - Berechtigung IMPLICIT_SCHEMA für die Datenbank, wenn der implizite oder explizite Schemaname der Tabelle nicht existiert
    - Zugriffsrecht CREATEIN für das Schema, wenn sich der Schemaname der Tabelle auf ein vorhandenes Schema bezieht

Zum Ändern einer Tabelle zum Zweck der Zuordnung einer Datenpartition muss der Benutzer über die folgenden Berechtigungen bzw. Zugriffsrechte verfügen:

- Der Benutzer, der die ATTACH-Operation ausführt, muss über die Berechtigung verfügen, die erforderlich ist, um die Zieltabelle zu ändern (ALTER) und Daten in sie einzufügen (INSERT).
- Darüber hinaus muss der Benutzer berechtigt sein, Daten aus der Quellentabelle auszuwählen (SELECT) und die Quellentabelle zu löschen (DROP). Daher müssen zum Ändern einer Tabelle für die Zuordnung einer Datenpartition die Zugriffsberechtigungen der Berechtigungs-ID der Anweisung mindestens eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte für die Quellentabelle beinhalten:
  - Zugriffsrecht SELECT für die Quellentabelle und Zugriffsrecht DROPIN für das Schema der Quellentabelle
  - Zugriffsrecht CONTROL für die Quellentabelle
  - Berechtigung SYSADM oder DBADM

Zum Rotieren von Daten in einer partitionierten Tabelle geben Sie die Anweisung ALTER TABLE in die Befehlszeile ein. Das folgende Beispiel veranschaulicht, wie die Tabelle 'stock' aktualisiert wird, indem die Daten von Dezember 2001 entfernt und durch die aktuellsten Daten von Dezember 2003 ersetzt werden.

1. Sie entfernen Sie die alten Daten aus der Tabelle *stock*.

   ```
   ALTER TABLE stock DETACH PARTITION dec01 INTO newtable;
   ```

2. Sie laden Sie die neuen Daten. Durch das Dienstprogramm LOAD und der Option REPLACE werden die vorhandenen Daten überschrieben.

   ```
   LOAD FROM datendatei OF DEL REPLACE INTO newtable
   ```

   Anmerkung: Wenn abhängige Tabellen von der Tabelle mit aufgehobener Zuordnung vorhanden sind, müssen Sie die Anweisung SET INTEGRITY an diesen abhängigen Tabellen ausführen, bevor Sie Daten in die Tabelle mit aufgehobener Zuordnung laden können.

3. Führen Sie eine Datenbereinigung durch, falls erwünscht. Zur Datenbereinigung gehören folgende Aktivitäten:
   - Auffüllen fehlender Werte
   - Löschen inkonsistenter und unvollständiger Daten
   - Entfernen redundanter Daten aus verschiedenen Quellen
   - Umwandlung von Daten
     - Normalisierung (Daten aus verschiedenen Quellen, die denselben Wert auf verschiedene Arten darstellen, müssen im Rahmen eines Rollins der Daten in das Data-Warehouse miteinander abgeglichen werden.)
     - Zusammenfassung (Rohdaten, die zum Speichern im Data-Warehouse zu detailliert sind, müssen beim Rollin vorab zusammengefasst (Aggregation) werden.)

4. Ordnen Sie die neuen Daten als neuen Bereich zu.

   ```
   ALTER TABLE stock ATTACH PARTITION dec03
   STARTING '12/01/2003' ENDING '12/31/2003'
    FROM newtable;
   ```

   Das Zuordnen einer Datenpartition bewirkt, dass Abfragen auslaufen und Pakete ungültig werden.

5. Verwenden Sie die Anweisung SET INTEGRITY, um Indizes und andere abhängige Objekte zu aktualisieren. Der Lese- und Schreibzugriff wird während der Ausführung der Anweisung SET INTEGRITY zugelassen.

   ```
   SET INTEGRITY FOR stock ALLOW WRITE ACCESS
   IMMEDIATE CHECKED FOR EXCEPTION IN stock USE stock_ex;
   ```

Szenarios: Rollin und Rollout von Daten partitionierter Tabellen

Eine häufige Verwaltungsoperation in Data-Warehouses ist der regelmäßige Rollin neuer Daten und Rollout veralteter Daten. Dieser Abschnitt enthält einige Szenarios, die diese Vorgänge veranschaulichen.

Szenario 1: Rollout veralteter Daten durch Aufheben der Zuordnung für eine Datenpartition

Das folgende Beispiel zeigt, wie die Zuordnung für eine nicht mehr benötigte Partition (DEC01) zu einer partitionierten Tabelle mit dem Namen STOCK aufgehoben wird. Die Datenpartition mit aufgehobener Zuordnung wird verwendet, um eine neue Tabelle mit dem Namen STOCK_DROP zu erstellen, ohne dass Daten versetzt werden.

```
ALTER TABLE stock DETACH PART dec01 INTO stock_drop;
COMMIT WORK;
```

Zur Beschleunigung der DETACH-Operation wird für die Quellentabelle im Hintergrund automatisch eine Indexbereinigung durch einen asynchronen Indexbereinigungsprozess ausgeführt. Wenn keine abhängigen Tabellen mit aufgehobener Zuordnung für die Quellentabelle definiert sind, besteht keine Notwendigkeit, eine Anweisung SET INTEGRITY auszuführen, um die DETACH-Operation abzuschließen.

Die neue Tabelle kann gelöscht oder einer anderen Tabelle zugeordnet werden. Es ist ebenfalls möglich, die Tabelle abzuschneiden und mit neuen Daten zu laden und sie dann erneut der Quellentabelle zuzuordnen. Sie können diese Operationen sofort, d. h. noch vor dem Abschluss der asynchronen Indexbereinigung, ausführen, vorausgesetzt, dass für die Quellentabelle keine abhängigen Tabelle mit aufgehobener Zuordnung vorhanden sind.

Um festzustellen, ob auf eine Tabelle mit aufgehobener Zuordnung Zugriff besteht, fragen Sie die Katalogsicht SYSCAT.TABDETACHEDDEP ab. Wenn auf eine Tabelle mit aufgehobener Zuordnung nicht zugegriffen werden kann, geben Sie die Anweisung SET INTEGRITY mit der Option IMMEDIATE CHECKED für alle abhängigen Tabelle mit aufgehobener Zuordnung ein. Wenn Sie versuchen, auf eine Tabelle mit aufgehobener Zuordnung zuzugreifen, bevor die Verwaltungsoperation für alle abhängigen Tabellen mit aufgehobener Zuordnung abgeschlossen ist, wird ein Fehler (SQL20285N) zurückgegeben.

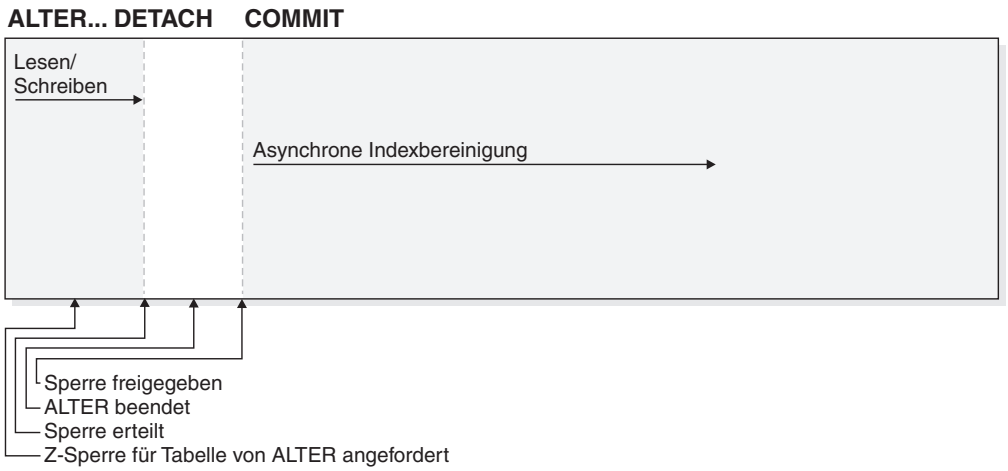


Abbildung 37. Phasen der Datenverfügbarkeit während einer DETACH-Operation. Wenn keine abhängigen Tabellen mit aufgehobener Zuordnung vorhanden sind, beginnt die asynchrone Indexbereinigung unmittelbar nach dem Commit der DETACH-Operation; andernfalls beginnt die asynchrone Indexbereinigung nach dem Commit der Verwaltungsoperation für die abhängigen Tabellen mit aufgehobener Zuordnung.

Szenario 2: Erstellen eines neuen, leeren Bereichs

Das folgende Beispiel zeigt, wie eine leere Datenpartition (DEC02) zu einer partitionierten Tabelle mit dem Namen STOCK hinzugefügt wird. Die Klauseln STARTING FROM und ENDING AT geben den Wertebereich für die neue Datenpartition an.

```
ALTER TABLE stock ADD PARTITION dec02
 STARTING FROM '12/01/2002' ENDING AT '12/31/2002';
```

Diese Anweisung ALTER TABLE...ADD PARTITION führt eine DRAIN-Operation für Abfragen für die Tabelle STOCK durch und inaktiviert Pakete; d. h., bereits vorhandene Abfragen werden normal abgeschlossen, bevor die Hinzufügeoperation fortgesetzt wird. Nach dem Absetzen der Anweisung ALTER TABLE...ADD PARTITION werden alle neuen Abfragen, die auf die Tabelle STOCK zugreifen, in den Wartestatus gesetzt, bis die Datenpartition hinzugefügt ist.

Laden Sie Daten in die Tabelle:

```
LOAD FROM datendatei OF DEL
 INSERT INTO stock
 ALLOW READ ACCESS;
```

Setzen Sie die Anweisung SET INTEGRITY ab, um Integritätsbedingungen zu überprüfen und abhängige MQTs (Materialized Query Tables, gespeicherte Abfragetabellen) zu aktualisieren: Alle Zeilen, die definierte Integritätsbedingungen verletzen, werden in die Ausnahmetabelle STOCK_EX versetzt.

```
SET INTEGRITY FOR stock
 ALLOW READ ACCESS IMMEDIATE CHECKED FOR EXCEPTION IN stock USE stock_ex;

COMMIT WORK;
```

Tipp: Wenn für Ihre Tabelle keine Integritätsbedingungen oder MQTs definiert sind, empfiehlt es sich, eine Ladeoperation nach dem Hinzufügen einer Datenpartition durchzuführen (anstatt vor dem Zuordnen einer Datenpartition), da keine Anweisung SET INTEGRITPY erforderlich ist, um die neuen Daten verfügbar zu machen. Der Hauptnachteil dieser Methode ist jedoch, dass sie Aktualisierungen der Tabelle sowohl während der eigentlichen LOAD-Operation als auch während der

nachfolgenden Ausführung einer Anweisung SET INTEGRITY verhindert. Zwar bewirken sowohl das Hinzufügen als auch das Zuordnen von Partitionen die Inaktivierung von Paketen, eine Ladeoperation gefolgt von einer Datenpartitionszuordnung liefert aber eine bessere Datenverfügbarkeit. Das Hinzufügen einer Datenpartition vor einer Import- oder Einfügeoperation wird jedoch in den Situationen empfohlen, in denen kein Rollin von Daten in großen Blöcken erfolgt. Das Hinzufügen einer Datenpartition ist außerdem in Fällen sinnvoll, in denen die Daten, die durch ein Rollin eingelagert werden, nicht den Bereichsgrenzen der Datenpartitionen entsprechen.

Szenario 3: Rollin neuer Daten durch Zuordnen einer geladenen Datenpartition

Das folgende Beispiel zeigt, wie eine Zuordnungsoperation dazu verwendet werden kann, das Laden neuer Daten in eine vorhandene partitionierte Tabelle (die Zieltabelle STOCK) zu vereinfachen. Die Daten werden in eine neue, leere Tabelle (DEC03) geladen, in der sie bei Bedarf überprüft und bereinigt werden können, ohne dass dies Auswirkungen auf die Zieltabelle hat. Zur Datenbereinigung gehören folgende Aktivitäten:

- Auffüllen fehlender Werte
- Löschen inkonsistenter und unvollständiger Daten
- Entfernen redundanter Daten aus verschiedenen Quellen
- Umsetzen der Daten durch Normalisierung oder Aggregation:
  - *Normalisierung*. Daten aus verschiedenen Quellen, die dieselben Werte auf verschiedene Weise darstellen, müssen im Rahmen des Rollins miteinander abgeglichen werden.
  - *Aggregation*. Rohdaten, die zum Speichern in einem Data-Warehouse zu detailliert sind, müssen beim Rollin aggregiert werden.

Nachdem die Daten auf diese Weise vorbereitet wurden, kann die neu geladene Datenpartition der Zieltabelle zugeordnet werden.

```
CREATE TABLE dec03(...);
LOAD FROM datendatei OF DEL REPLACE INTO dec03;
(Datenbereinigung, falls erforderlich)
ALTER TABLE stock ATTACH PARTITION dec03
 STARTING FROM '12/01/2003' ENDING AT '12/31/2003'
 FROM dec03;
```

Während einer Zuordnungsoperation muss mindestens eine der Klauseln STARTING FROM und ENDING AT angegeben werden; die untere Grenze (Klausel STARTING FROM) muss kleiner-gleich der oberen Grenze (Klausel ENDING AT) sein. Die neu zugeordnete Datenpartition darf sich mit keinem vorhandenen Datenpartitionsbereich in der Zieltabelle überschneiden. Wenn die obere Grenze des höchsten vorhandenen Bereichs als MAXVALUE definiert wurde, schlagen alle Versuche, einen neuen höchsten Bereich zuzuordnen, fehl, da sich der neue Bereich mit dem vorhandenen höchsten Bereich überschneidet. Eine ähnliche Einschränkung gilt für die unteren Bereiche, die bei MINVALUE enden. Darüber hinaus können Sie keine neue Datenpartition in der Mitte hinzufügen oder zuordnen, es sei denn, der neue Bereich liegt in einer Lücke zwischen den vorhandenen Bereichen. Grenzen, die vom Benutzer nicht angegeben werden, werden bei der Erstellung der Tabelle bestimmt.

Die Anweisung ALTER TABLE...ATTACH PARTITION führt eine DRAIN-Operation für Abfragen für die Tabelle STOCK durch und inaktiviert Pakete; d. h., bereits vorhandene Abfragen werden normal abgeschlossen, bevor die Zuordnungsoperation fortgesetzt wird. Nach dem Absetzen der Anweisung ALTER TABLE...ATTACH

PARTITION werden alle neuen Abfragen, die versuchen, auf die Tabelle STOCK zuzugreifen, in den Wartestatus gesetzt, bis die Datenpartition zugeordnet ist. Für die Tabelle 'stock' wird während dieses Übergangs eine Sperre des Modus Z (vollständige Sperrung) aktiviert. Die Daten in der zugeordneten Datenpartition sind noch nicht sichtbar, weil sie noch nicht durch die Anweisung SET INTEGRITY überprüft wurden.

Tipp: Führen Sie eine Anweisung COMMIT unmittelbar nach der Zuordnungsoperation aus, um die Tabelle zur Verwendung verfügbar zu machen.

Die Anweisung SET INTEGRITY ist erforderlich, um zu überprüfen, ob sich die neu zugeordneten Daten innerhalb des definierten Bereichs befinden. Darüber hinaus werden damit alle erforderlichen Verwaltungsaktivitäten für Indizes und andere abhängige Objekte, wie z. B. MQTs, durchgeführt. Neue Daten sind erst nach dem Commit der Anweisung SET INTEGRITY sichtbar; wird die Anweisung SET INTEGRITY jedoch online ausgeführt, besteht auf vorhandene Daten in der Tabelle STOCK vollständiger Zugriff für Lese- und Schreiboperationen.

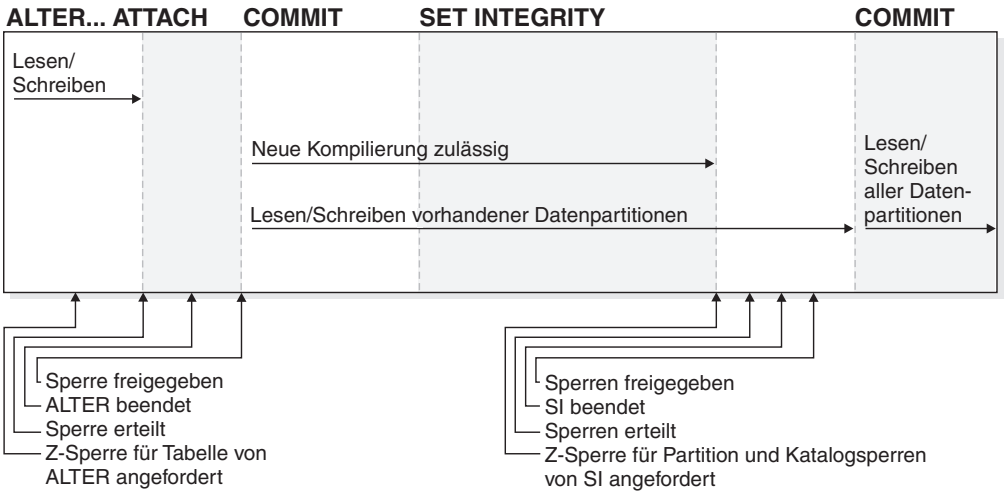


Abbildung 38. Phasen der Datenverfügbarkeit während einer ATTACH-Operation

Anmerkung: Es können keine DDL-Anweisungen oder Dienstprogrammoperationen für die Tabelle ausgeführt werden, während die Ausführung der Anweisung SET INTEGRITY läuft. Zu diesen Operationen gehören unter anderem die nachfolgend aufgeführten Anweisungen und Befehle:

- Befehl LOAD
- Befehl REDISTRIBUTE DATABASE PARTITION GROUP
- Befehl REORG INDEXES/TABLE
- Anweisung ALTER TABLE
  - ADD COLUMN
  - ADD PARTITION
  - ATTACH PARTITION
  - DETACH PARTITION
- Anweisung CREATE INDEX

Die Anweisung SET INTEGRITY prüft die Daten in der neu zugeordneten Datenpartition.

```
SET INTEGRITY FOR stock
 ALLOW WRITE ACCESS IMMEDIATE CHECKED FOR EXCEPTION IN stock USE stock_ex;
```

Durch das Commit der Transaktion wird die Tabelle verfügbar:

```
COMMIT WORK;
```

Alle Zeilen, die außerhalb des gültigen Bereichs liegen oder sonstige Integritätsbedingungen verletzen, werden in die Ausnahmetabelle STOCK_EX versetzt. Sie können Abfragen für diese Tabelle ausführen, die Zeilen korrigieren und diese in die Tabelle STOCK einfügen.

Kapitel 14. Laden von Daten

Parallelität und Laden

Das Dienstprogramm LOAD kann die Vorteile einer Hardwarekonfiguration nutzen, in der mehrere Prozessoren oder mehrere Speichereinheiten verwendet werden, wie z. B. in einer symmetrischen Mehrprozessorumgebung (SMP - Symmetric Multiprocessor).

Es gibt mehrere Möglichkeiten, wie eine parallele Verarbeitung umfangreicher Datenmengen mit dem Dienstprogramm LOAD durchgeführt werden kann. Eine Möglichkeit besteht in der Verwendung mehrerer Speichereinheiten, die eine E/A-Parallelität während der Ladeoperation ermöglicht (siehe Abb. 39). Eine weitere Möglichkeit ist der Einsatz mehrerer Prozessoren in einer SMP-Umgebung, der eine partitionsinterne Parallelität ermöglicht (siehe Abb. 40). Beide Möglichkeiten können kombiniert verwendet werden, um ein noch schnelleres Laden der Daten zu erreichen.

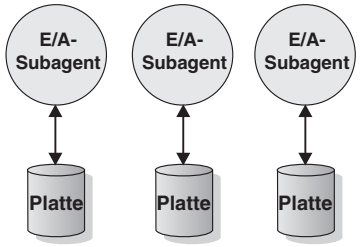


Abbildung 39. E/A-Parallelität beim Laden von Daten

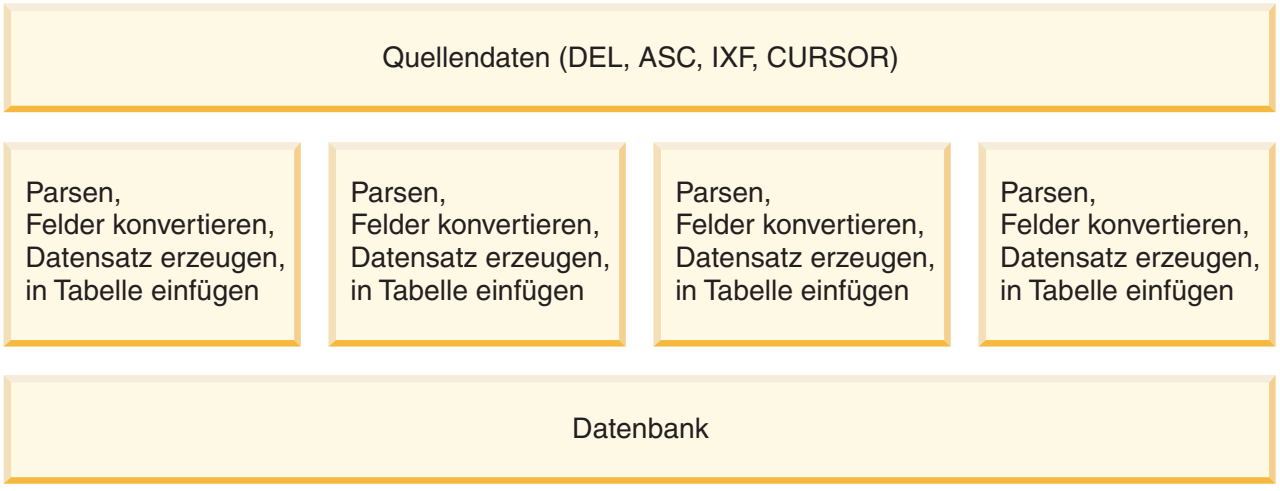


Abbildung 40. Partitionsinterne Parallelität beim Laden von Daten

Aspekte des mehrdimensionalen Clustering

Für das Laden von Daten in MDC-Tabellen (Multi-Dimensional Clustering - mehrdimensionales Clustering) gelten die folgenden Einschränkungen:

- Die Option SAVECOUNT des Befehls LOAD wird nicht unterstützt.

- Der Änderungswert `totalfreespace` für den Dateityp wird nicht unterstützt, da diese Tabellen ihren freien Speicherbereich selbst verwalten.
- Der Änderungswert `anyorder` für den Dateityp ist für MDC-Tabellen erforderlich. Werden Daten ohne den Änderungswert `anyorder` in eine MDC-Tabelle geladen, wird dieser Änderungswert vom Dienstprogramm explizit aktiviert.

Bei Verwendung des Befehls LOAD mit einer MDC-Tabelle werden Verstöße gegen eindeutige Integritätsbedingungen wie folgt behandelt:

- Wenn die Tabelle vor der Ladeoperation einen eindeutigen Schlüssel enthielt und doppelte Datensätze in die Tabelle geladen werden, verbleibt der ursprüngliche Datensatz in der Tabelle, und die neuen Datensätze werden während der DELETE-Phase gelöscht.
- Enthielt die Tabelle vor der Ladeoperation keinen eindeutigen Schlüssel und werden sowohl ein eindeutiger Schlüssel als auch doppelte Datensätze in die Tabelle geladen, wird nur einer der Datensätze mit dem eindeutigen Schlüssel geladen, und die anderen Datensätze werden während der DELETE-Phase gelöscht.

  Anmerkung: Es gibt keine explizite Methode, mit der bestimmt werden kann, welche Datensätze geladen und welche gelöscht werden.

Hinweise zur Leistung

Um eine bessere Leistung des Dienstprogramms LOAD beim Laden von MDC-Tabellen zu erzielen, sollte der Wert des Datenbankkonfigurationsparameters *util_heap_sz* erhöht werden. Die Leistung des Algorithmus 'mdc-load' ist deutlich besser, wenn dem Dienstprogramm mehr Speicher zur Verfügung steht. Dies verringert die Platten-E/A während des Datenclusterings, das in der LOAD-Phase stattfindet. Ab Version 9.5 kann der Wert für die Option DATA BUFFER des Befehls LOAD den Wert von *util_heap_sz* temporär überschreiten, falls im System mehr Speicher zur Verfügung steht.

MDC-Ladeoperationen enthalten immer eine BUILD-Phase, da alle MDC-Tabellen mit Blockindizes ausgestattet sind.

Während der LOAD-Phase werden zusätzliche Protokollierungen zur Verwaltung der Blockzuordnung ausgeführt. Pro zugeordnetem Speicherbereich werden ca. zwei zusätzliche Protokollsätze erstellt. Um eine zufriedenstellende Leistung zu gewährleisten, sollte der Datenbankkonfigurationsparameter *logbufsz* auf einen Wert gesetzt werden, der diesen Umstand berücksichtigt.

Zum Laden von Daten in MDC-Tabellen wird eine temporäre Systemtabelle mit einem Index eingesetzt. Die Größe der Tabelle ist proportional zur Anzahl der unterschiedlichen geladenen Zellen. Die Größe jeder Zeile in der Tabelle ist proportional zur Größe des Schlüssels für die MDC-Dimension. Um die Platten-E/A zu reduzieren, die durch die Bearbeitung dieser Tabelle während einer Ladeoperation entsteht, muss sichergestellt sein, dass der Pufferpool für den Tabellenbereich für temporäre Tabellen groß genug ist.

Ladeaspekte für partitionierte Tabellen

Alle vorhandenen Ladefunktionen werden unterstützt, wenn die Zieltabelle partitioniert ist. Hierbei gelten jedoch die folgenden allgemeinen Einschränkungen:

- Konsistenzpunkte werden nicht unterstützt, wenn die Anzahl der Partitionierungsagenten größer als 1 ist.

- Das Laden von Daten in eine Untergruppe von Datenpartitionen, während die verbleibenden Datenpartitionen vollständig online bleiben, wird nicht unterstützt.
- Die von einer Ladeoperation verwendete Ausnahmetabelle kann nicht partitioniert sein.
- Ein eindeutiger Index kann nicht erneut erstellt werden, wenn das Dienstprogramm LOAD im Einfügemodus (INSERT) oder Neustartmodus (RESTART) ausgeführt wird und freigegebene Objekte von der Zieltabelle der Ladeoperation abhängen.
- Ähnlich wie beim Laden von MDC-Tabellen bleibt die genaue Reihenfolge der Eingabedatensätze beim Laden von partitionierten Tabellen nicht erhalten. Sortierungen werden lediglich innerhalb der Zelle oder Datenpartition beibehalten.
- Ladeoperationen, die mehrere Formatierungsprogramme auf jeder Datenbankpartition verwenden, erhalten lediglich eine ungefähre Reihenfolge der Eingabedatensätze. Beim Ausführen eines einzelnen Formatierungsprogramms auf jeder Datenbankpartition werden die Eingabedatensätze nach Zellen- oder Tabellenpartitionierungsschlüssel gruppiert. Um auf jeder Datenbankpartition ein einziges Formatierungsprogramm auszuführen, muss für CPU_PARALLELISM explizit der Wert 1 angefordert werden.

Allgemeines Verhalten des Dienstprogramms LOAD
Das Dienstprogramm LOAD fügt Datensätze in die richtige Datenpartition ein. Es ist nicht erforderlich, ein externes Dienstprogramm (wie beispielsweise einen Verteilerprozess) zu verwenden, um die Eingabedaten vor dem Laden zu partitionieren.

Das Dienstprogramm LOAD greift nicht auf freigegebene oder zugeordnete Datenpartitionen zu. Daten werden lediglich in sichtbare Datenpartitionen eingefügt. Sichtbare Datenpartitionen sind weder zugeordnet noch freigegeben. Darüber hinaus werden freigegebene oder zugeordnete Datenpartitionen von LOAD REPLACE-Operationen nicht abgeschnitten. Da das Dienstprogramm LOAD Sperren für die Systemkatalogtabellen anfordert, wartet das Dienstprogramm LOAD auf alle ALTER TABLE-Transaktionen, für die noch kein Commit durchgeführt wurde. Solche Transaktionen fordern eine exklusive Sperre für die relevanten Zeilen in den Katalogtabellen an, und die exklusive Sperre muss beendet werden, bevor die Ladeoperation fortgesetzt werden kann. Dies bedeutet, dass nicht festgeschriebene Transaktionen ALTER TABLE ... ADD PARTITION, ALTER TABLE ... ATTACH PARTITION bzw. ALTER TABLE ... DETACH PARTITION nicht zulässig sind, während die Ladeoperation ausgeführt wird. Alle Eingabequellensätze, die für eine zugeordnete (ATTACHED) oder freigegebene (DETACHED) Datenpartition bestimmt sind, werden zurückgewiesen und können aus der Ausnahmetabelle abgerufen werden, sofern eine angegeben wurde. Eine Informationsnachricht wird in die Nachrichtendatei geschrieben, um anzugeben, dass einige Datenpartitionen der Zieltabelle im Status zugeordnet (ATTACHED) oder freigegeben (DETACHED) waren. Sperren für die relevanten Zeilen der Katalogtabelle, die der Zieltabelle entsprechen, verhindern, dass Benutzer die Partitionierung der Zieltabelle durch Absetzen einer Operation ALTER TABLE ... ADD PARTITION, ALTER TABLE ... ATTACH PARTITION bzw. ALTER TABLE ... DETACH PARTITION während der Ausführung des Dienstprogramms LOAD ändern.

Verarbeitung ungültiger Zeilen
Wenn das Dienstprogramm LOAD auf einen Datensatz trifft, der zu keiner der sichtbaren Datenpartitionen gehört, wird der betreffende Datensatz zurückgewiesen, und das Dienstprogramm LOAD setzt die Verarbeitung fort. Die Anzahl der Datensätze, die aufgrund einer Verletzung der Bereichsvorgabe zurückgewiesen

wurden, wird nicht explizit angegeben, ist aber in der Gesamtanzahl der zurückgewiesenen Datensätze enthalten. Durch das Zurückweisen eines Datensatzes aufgrund einer Verletzung der Bereichsvorgabe wird die Anzahl der Warnungen für Zeilen nicht erhöht. Es wird eine einzige Nachricht (SQL0327N) in die Nachrichtendatei des Dienstprogramms LOAD geschrieben, die angibt, dass Bereichsverletzungen festgestellt wurden. Es wird jedoch nicht für jeden Datensatz eine eigene Nachricht protokolliert. Zusätzlich zu allen Spalten der Zieltabelle enthält die Ausnahmetabelle auch Spalten, die den Typ des Verstoßes beschreibt, der für eine bestimmte Zeile aufgetreten ist. Zeilen, die ungültige Daten enthalten (einschließlich Daten, die nicht partitioniert werden können), werden in die Speicherauszugsdatei geschrieben.

Da Einfügungen (INSERT-Operationen) in die Ausnahmetabelle ressourcenintensiv sind, können Sie steuern, welche Verstöße gegen Integritätsbedingungen in die Ausnahmetabelle eingefügt werden sollen. Das Standardverhalten des Dienstprogramms LOAD ist beispielsweise, Zeilen in die Ausnahmetabelle einzufügen, die aufgrund einer Verletzung der Bereichsvorgabe oder der eindeutigen Integritätsbedingung zurückgewiesen wurden, ansonsten jedoch gültig sind. Sie können dieses Verhalten inaktivieren, indem Sie jeweils NORANGEEXC oder NOUNIQUEEXC mit der Ausnahmebedingung FOR EXCEPTION angeben. Wenn Sie angeben, dass diese Verletzungen der Bereichsvorgabe nicht in die Ausnahmetabelle eingefügt werden sollen, oder wenn Sie keine Ausnahmetabelle angeben, gehen alle Informationen zu Zeilen, die eine Bereichsvorgabe oder eine eindeutige Integritätsbedingung verletzen, verloren.

Protokolldatei
Bei partitionierten Zieltabellen enthält der entsprechende Eintrag in der Protokolldatei keine Liste der Tabellenbereiche, die die Zieltabelle umfasst. Eine andere Kennung für die Granularität von Operationen ('R' anstelle von 'T') gibt an, dass eine Ladeoperation für eine partitionierte Tabelle ausgeführt wurde.

Beenden einer Ladeoperation
Beim Beenden einer LOAD REPLACE-Operation werden alle sichtbaren Datenpartitionen vollständig abgeschnitten; beim Beenden einer LOAD INSERT-Operation werden alle sichtbaren Datenpartitionen auf ihre jeweilige Länge vor der Ladeoperation abgeschnitten. Beim Beenden einer ALLOW READ ACCESS-Ladeoperation, die in der LOAD COPY-Phase fehlschlug, werden die entsprechenden Indizes ungültig gemacht. Ein Index wird ebenfalls ungültig gemacht, wenn eine Ladeoperation mit der Option ALLOW NO ACCESS beendet wird, die den Index geändert hat. (Der Index wird ungültig gemacht, weil der Indexierungsmodus REBUILD (erneut erstellen) ist oder weil ein Schlüssel während der inkrementellen Verwaltung eingefügt wurde und den Index in einem inkonsistenten Status belassen hat). Das Laden von Daten in mehrere Ziele hat keinen Einfluss auf Recoveryoperationen, mit der Ausnahme, dass die Ladeoperation von einem Konsistenzpunkt, der während der LOAD-Phase erstellt wurde, nicht erneut gestartet werden kann. In diesem Fall wird die LOAD-Option SAVECOUNT ignoriert, wenn die Zieltabelle partitioniert ist. Dieses Verhalten entspricht dem Verhalten beim Laden von Daten in einer MDC-Zieltabelle.

Generierte Spalten
Befindet sich eine generierte Spalte in einem der Partitionierungs-, Dimensions- oder Verteilungsschlüssel, wird der Änderungswert `generatedoverride` für den Dateityp ignoriert, und das Dienstprogramm LOAD generiert Werte, als wäre der Änderungswert `generatedignore` für den Dateityp angegeben. Das Laden eines falschen Wertes einer generierten Spalte kann in diesem Fall dazu führen, dass der Datensatz in die falsche physische Position (wie beispielsweise in die falsche Da-

tenpartition, den falschen MDC-Block oder die falsche Datenbankpartition) gestellt wird. Beispiel: Befindet sich ein Datensatz in einer falschen Datenpartition, muss die Operation SET INTEGRITY zum Festlegen der Integrität den Satz in eine andere physische Position versetzen, was im Verlauf von SET INTEGRITY-Operationen, die online ausgeführt werden, nicht bewerkstelligt werden kann.

Datenverfügbarkeit
Der aktuelle Algorithmus für ALLOW READ ACCESS-Ladeoperationen erstreckt sich auf partitionierte Tabellen. Eine ALLOW READ ACCESS-Ladeoperation ermöglicht den gleichzeitigen Lesezugriff auf die gesamte Tabelle. Dies schließt sowohl Ladedatenpartitionen als auch Nicht-Ladedatenpartitionen ein.

Datenpartitionsstatus
Nach einer erfolgreichen Ladeoperation können sichtbare Datenpartitionen entweder in den Tabellenstatus Set Integrity Pending oder Read Access Only oder beide wechseln. Dies hängt von bestimmten Bedingungen ab. So können Datenpartitionen in diese Status versetzt werden, wenn für die Tabelle Integritätsbedingungen bestehen, die von der Ladeoperation nicht verwaltet werden können. Dabei kann es sich z. B. um Integritätsbedingungen handeln, die Prüfungen auf Integritätsbedingungen und freigegebene MQTs (Materialized Query Table) beinhalten. Schlägt eine Ladeoperation fehl, verbleiben alle sichtbaren Datenpartitionen im Tabellenstatus Load Pending ('Laden anstehend').

Fehlerisolation
Eine Fehlerisolation auf Datenpartitionsebene wird nicht unterstützt. Das Isolieren von Fehlern bedeutet, Ladeoperationen für Datenpartitionen ohne Fehler fortzusetzen und Ladeoperationen für Datenpartitionen mit Fehlern zu stoppen. Fehler können in verschiedenen Datenbankpartitionen übergreifend isoliert werden. Das Dienstprogramm LOAD kann jedoch keine Transaktionen in einer Untergruppe von sichtbaren Datenpartitionen festschreiben und in den verbleibenden sichtbaren Datenpartitionen rückgängig machen.

Sonstige Aspekte

- Eine inkrementelle Indexierung wird nicht unterstützt, wenn einer der Indizes als ungültig markiert ist. Ein Index gilt als ungültig, wenn er erneut erstellt werden muss oder wenn freigegebene abhängige Objekte mit der Anweisung SET INTEGRITY überprüft werden müssen.
- Das Laden von Daten in Tabellen, deren Partitionierung anhand einer beliebigen Kombination aus Algorithmen für das Partitionieren nach Bereich, das Verteilen nach Hash oder das Organisieren nach Dimension erfolgte, wird ebenfalls unterstützt.
- Die Größe von Protokollsätzen, die eine Liste der von der Ladeoperation betroffenen Objekt- und Tabellenbereichs-IDs umfassen (Protokollsätze LOAD START und COMMIT (PENDING LIST)) kann stark anwachsen und so den für andere Anwendungen zur Verfügung stehenden aktiven Protokollspeicher reduzieren.
- Wenn eine Tabelle sowohl partitioniert als auch verteilt ist, kann es sein, dass sich eine Ladeoperation für partitionierte Datenbanken nicht auf alle Datenbankpartitionen auswirkt. Nur die Objekte in den Ausgabedatenbankpartitionen werden geändert.
- Während einer Ladeoperation nimmt die Speicherbelegung für partitionierte Tabellen mit der Anzahl der Tabellen zu. Bitte beachten Sie, dass der Gesamtanstieg nicht linear ist, da nur ein geringer Prozentsatz des Gesamtspeicherbedarfs proportional zur Anzahl der Datenpartitionen ist.

Kapitel 15. Laden von Daten in einer Umgebung mit partitionierten Datenbanken

Übersicht zum Laden - Umgebungen mit partitionierten Datenbanken

In einer Mehrpartitionsdatenbank befinden sich große Datenmengen auf vielen verschiedenen Datenbankpartitionen. Mit Verteilungsschlüsseln können Sie die Datenbankpartition festlegen, auf der die einzelnen Datenteile jeweils gespeichert werden sollen. Die Daten müssen *verteilt* werden, bevor sie auf die richtige Datenbankpartition geladen werden können.

Beim Laden von Tabellen in Mehrpartitionsdatenbanken stellt das Dienstprogramm LOAD das folgende Funktionsspektrum zur Verfügung:

- Paralleles Verteilen von Eingabedaten
- Gleichzeitiges Laden von Daten auf entsprechende Datenbankpartitionen
- Übertragen von Daten von einem System an ein anderes System

Das Laden von Daten in eine Mehrpartitionsdatenbank erfolgt in zwei Phasen: In einer *SETUP-Phase* werden Datenbankpartitionsressourcen wie beispielsweise Tabellensperren angefordert, und in einer *LOAD-Phase* werden die Daten in die Datenbankpartitionen geladen. Mit der Option ISOLATE_PART_ERRS des Befehls LOAD können Sie auswählen, wie Fehler in diesen beiden Phasen jeweils verarbeitet werden und wie sich Fehler in einer oder mehreren der Datenbankpartitionen auf die Ladeoperation für diejenigen Datenbankpartitionen auswirken, die fehlerfrei sind.

Beim Laden von Daten in eine Mehrpartitionsdatenbank können die folgenden Modi verwendet werden:

PARTITION_AND_LOAD
Die Daten werden verteilt (möglicherweise in Parallelverarbeitung) und gleichzeitig auf die entsprechenden Datenbankpartitionen geladen.

PARTITION_ONLY
Die Daten werden verteilt (möglicherweise in Parallelverarbeitung), und die Ausgabe wird in Dateien an einer angegebenen Speicherposition auf jeder Ladedatenbankpartition geschrieben. Jede Datei enthält Partitionskopfdaten, die angeben, wie die Daten über die Datenbankpartitionen verteilt wurden, und dass die Datei unter Verwendung des Modus LOAD_ONLY in die Datenbank geladen werden kann.

LOAD_ONLY
Es wird davon ausgegangen, dass die Daten bereits über die Datenbankpartitionen verteilt sind. Der Verteilungsprozess wird übersprungen, und die Daten werden gleichzeitig auf die entsprechenden Datenbankpartitionen geladen.

LOAD_ONLY_VERIFY_PART
Es wird davon ausgegangen, dass die Daten bereits über die Datenbankpartitionen verteilt sind, aber die Datendatei enthält keine Partitionskopfdaten. Der Verteilungsprozess wird übersprungen, und die Daten werden gleichzeitig auf die entsprechenden Datenbankpartitionen geladen.

Während der Ladeoperation wird für jede Zeile geprüft, ob sie sich auf der korrekten Datenbankpartition befindet. Zeilen, die Datenbankpartitionsverstöße enthalten, werden in eine Speicherauszugsdatei gestellt, sofern der Änderungswert `dumpfile` für den Dateityp angegeben wurde. Andernfalls werden die Zeilen gelöscht. Wenn für eine bestimmte Ladedatenbankpartition Datenbankpartitionsverstöße vorliegen, wird für die betreffende Datenbankpartition eine einzige Warnung in die LOAD-Nachrichtendatei geschrieben.

ANALYZE
Es wird eine optimale Verteilungszuordnung mit einer gleichmäßigen Verteilung auf alle Datenbankpartitionen generiert.

Konzepte und Terminologie

Im Zusammenhang mit der Funktionsweise und der Ausführung des Dienstprogramms LOAD in einer Umgebung mit partitionierten Datenbanken mit mehreren Datenbankpartitionen wird die folgende Terminologie verwendet:

- Die *Koordinatorpartition* ist die Datenbankpartition, zu der der Benutzer eine Verbindung herstellt, um die Ladeoperation auszuführen. In den Modi PARTITION_AND_LOAD, PARTITION_ONLY und ANALYZE wird davon ausgegangen, dass sich die Datendatei auf dieser Datenbankpartition befindet, sofern nicht die Option CLIENT des Befehls LOAD angegeben ist. Mit der Option CLIENT wird angegeben, dass sich die zu ladenden Daten auf einem Client mit Fernverbindung befinden.
- In den Modi PARTITION_AND_LOAD, PARTITION_ONLY und ANALYZE liest der *Agent für Partitionierungsvorbereitung* die Benutzerdaten und teilt sie reihum den *Partitionierungsagenten* zu, die die Daten verteilen. Dieser Prozess findet immer auf der Koordinatorpartition statt. Bei allen Ladeoperationen ist pro Datenbankpartition maximal ein Partitionierungsagent zulässig.
- In den Modi PARTITION_AND_LOAD, LOAD_ONLY und LOAD_ONLY_VERIFY_PART werden auf jeder Ausgabedatenbankpartition *Ladeagenten* ausgeführt. Sie koordinieren das Laden der Daten auf diese Datenbankpartition.
- *Dateiladeagenten* werden während einer Ladeoperation im Modus PARTITION_ONLY auf jeder Ausgabedatenbankpartition ausgeführt. Sie empfangen Daten von Partitionierungsagenten und schreiben sie in eine Datei auf ihrer Datenbankpartition.
- Die Option SOURCEUSEREXIT stellt eine Funktion bereit, über die das Dienstprogramm LOAD ein angepasstes Script bzw. eine angepasste ausführbare Datei ausführen kann. Dieses Script bzw. diese Datei wird im vorliegenden Handbuch als *Benutzerexit* bezeichnet.

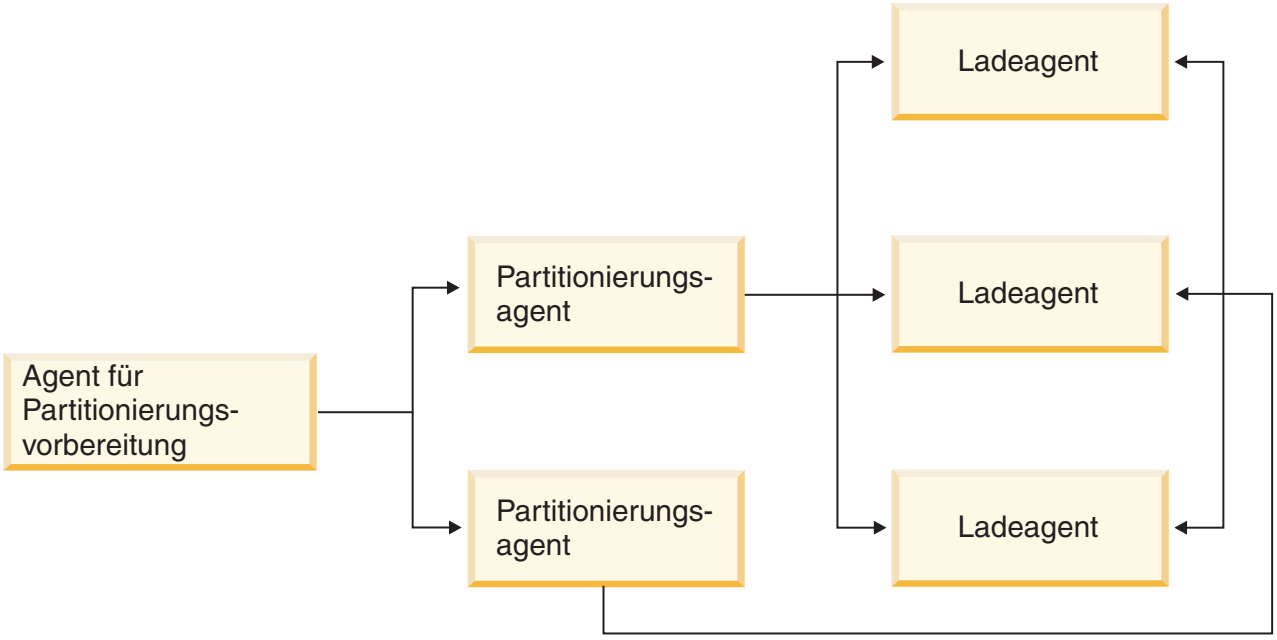


Abbildung 41. Übersicht über das Laden in partitionierte Datenbanken. Die Quellendaten werden durch den Agenten für Partitionierungsvorbereitung gelesen. Ungefähr die Hälfte der Daten wird jeweils an zwei Partitionierungsagenten gesendet, die die Daten verteilen und an eine der drei Datenbankpartitionen senden. Auf den einzelnen Datenbankpartitionen werden die Daten durch den jeweiligen Ladeagenten geladen.

Hinweise und Tipps für das Laden von Daten in einer Umgebung mit partitionierten Datenbanken

Bevor Sie eine Tabelle in eine Mehrpartitionsdatenbank laden, sollten Sie die folgenden Punkte beachten:

- Machen Sie sich mit den Konfigurationsoptionen des Dienstprogramms LOAD vertraut, indem Sie das Dienstprogramm mit kleinen Datenmengen verwenden.
- Wenn die Eingabedaten bereits sortiert sind oder in einer bestimmten Reihenfolge vorliegen und Sie diese Reihenfolge während des Ladeprozesses beibehalten möchten, darf nur eine einzige Datenbankpartition zum Verteilen verwendet werden. Bei paralleler Verteilung kann nicht sichergestellt werden, dass die Daten in derselben Reihenfolge geladen werden, in der sie empfangen wurden. Das Dienstprogramm LOAD wählt standardmäßig einen einzigen Partitionierungsagenten aus, wenn der Änderungswert `anyorder` im Befehl LOAD nicht angegeben wird.
- Wenn große Objekte (LOBs) aus separaten Dateien geladen werden (wenn Sie also den Änderungswert `lobsinfile` über das Dienstprogramm LOAD verwenden), müssen alle Datenbankpartitionen, auf denen Ladevorgänge stattfinden, Lesezugriff auf alle Verzeichnisse mit LOB-Dateien haben. Der LOAD-Parameter *lob-pfad* muss beim Arbeiten mit LOBs vollständig qualifiziert sein.
- Sie können die Fortsetzung eines Jobs, der in einer Mehrpartitionsdatenbank ausgeführt wird, erzwingen, selbst wenn die Ladeoperation (beim Start) feststellt, dass sich einige Ladedatenbankpartitionen oder zugeordnete Tabellenbereiche oder Tabellen im Offlinemodus befinden. Hierzu setzen Sie die Option ISOLATE_PART_ERRS auf SETUP_ERRS_ONLY oder SETUP_AND_LOAD_ERRS.

- Mit der LOAD-Konfigurationsoption STATUS_INTERVAL kann der Status eines Jobs überwacht werden, der in einer Mehrpartitionsdatenbank ausgeführt wird. Die Ladeoperation gibt in den angegebenen Intervallen Nachrichten zurück, aus denen ersichtlich ist, wie viele Megabyte Daten durch den Agenten für die Partitionierungsvorbereitung gelesen wurden. Diese Nachrichten werden in der Nachrichtendatei des Agenten für die Partitionierungsvorbereitung gespeichert. Um den Inhalt dieser Datei während der Ladeoperation anzuzeigen, stellen Sie eine Verbindung zur Koordinatorpartition her und setzen den Befehl LOAD QUERY für die Zieltabelle ab.
- Sie können mit besserer Leistung rechnen, wenn sich die am Verteilungsprozess beteiligten Datenbankpartitionen (durch die Option PARTITIONING_DBPARTNUMS definiert) von den Ladedatenbankpartitionen (durch die Option OUTPUT_DBPARTNUMS definiert) unterscheiden, da dann weniger Konkurrenzsituationen für CPU-Zyklen auftreten. Wenn Daten in eine Mehrpartitionsdatenbank geladen werden, sollte das Dienstprogramm LOAD auf einer Datenbankpartition aufgerufen werden, die weder an der Verteilungs- noch an der Ladeoperation beteiligt ist.
- Bei Angabe des Parameters MESSAGES im Befehl LOAD werden die Nachrichtendateien der Agenten für die Partitionierungsvorbereitung, der Partitionierungs- und der Ladeagenten am Ende der Ladeoperation zu Referenzzwecken gespeichert. Um den Inhalt dieser Dateien während einer Ladeoperation anzuzeigen, stellen Sie eine Verbindung zur gewünschten Datenbankpartition her und setzen einen Befehl LOAD QUERY für die Zieltabelle ab.
- Das Dienstprogramm LOAD wählt nur eine Ausgabedatenbankpartition für die Erfassung von Statistikdaten aus. Mit der Datenbankkonfigurationsoption RUN_STAT_DBPARTNUM können Sie die Datenbankpartition angeben.
- Führen Sie vor dem Laden von Daten in eine Mehrpartitionsdatenbank den Designadvisor aus, um die jeweils beste Partition für die einzelnen Tabellen zu ermitteln. Weitere Informationen hierzu finden Sie in „Designadvisor" in *Optimieren der Datenbankleistung*.

Fehlerbehebung

Wenn das Dienstprogramm LOAD blockiert, haben Sie folgende Möglichkeiten:

- Mit dem Parameter STATUS_INTERVAL kann der Status einer Ladeoperation für Mehrpartitionsdatenbanken überwacht werden. Die Statusintervalldaten werden in der Nachrichtendatei des Agenten für die Partitionierungsvorbereitung auf der Koordinatorpartition gespeichert.
- Überprüfen Sie in der Nachrichtendatei des Partitionierungsagenten den Status der Prozesse des Partitionierungsagenten auf den einzelnen Datenbankpartitionen. Wenn das Laden ohne Fehler fortgesetzt wird und die Option TRACE definiert wurde, sollten diese Nachrichtendateien Tracenachrichten für eine Reihe von Datensätzen enthalten.
- Überprüfen Sie, ob die LOAD-Nachrichtendatei Fehlernachrichten enthält.

  Anmerkung: Damit diese Dateien vorhanden sind, müssen Sie die Option MESSAGES des Befehls LOAD angeben.
- Unterbrechen Sie die aktuelle Ladeoperation, wenn Sie Fehler finden, die darauf schließen lassen, dass bei einem der Ladeprozesse Fehler aufgetreten sind.

Laden von Daten in einer Umgebung mit partitionierten Datenbanken

Dienstprogramm LOAD zum Laden von Daten in eine Umgebung mit partitionierten Datenbanken verwenden

Bevor Sie eine Tabelle in eine Mehrpartitionsdatenbank laden, sollten Sie die folgenden Punkte beachten:

1. Stellen Sie sicher, dass der Konfigurationsparameter *svcename* des Datenbankmanagers und die Profilregistrierdatenbankvariable **DB2COMM** richtig definiert sind. Dies ist wichtig, weil das Dienstprogramm LOAD TCP/IP verwendet, um Daten von den Agenten für Partitionierungsvorbereitung an die Partitionierungsagenten und von den Partitionierungsagenten an die Ladedatenbankpartitionen zu übertragen.
2. Bevor Sie das Dienstprogramm LOAD ausführen, müssen Sie mit der Datenbank, in die die Daten geladen werden, verbunden bzw. in der Lage sein, implizit eine Verbindung zur Datenbank herzustellen. Da das Dienstprogramm LOAD eine Anweisung COMMIT absetzt, sollten Sie vor dem Beginn der Ladeoperation alle Transaktionen beenden und alle Sperren aufheben, indem Sie eine Anweisung COMMIT oder ROLLBACK absetzen. Wird der Modus PARTITION_AND_LOAD, PARTITION_ONLY oder ANALYZE verwendet, muss sich die geladene Datendatei auf dieser Datenbankpartition befinden. Ausgenommen sind die folgenden Fälle:
   a. Die Option CLIENT wurde angegeben. In diesem Fall müssen sich die Daten auf der Clientmaschine befinden.
   b. Der Eingabequellentyp ist CURSOR. Hierbei gibt es keine Eingabedatei.
3. Führen Sie den Designadvisor aus, um die jeweils beste Datenbankpartition für die einzelnen Tabellen zu ermitteln. Weitere Informationen hierzu finden Sie in „Designadvisor" in *Optimieren der Datenbankleistung*.

Bei der Verwendung des Dienstprogramms LOAD zum Laden von Daten in eine Mehrpartitionsdatenbank gelten die folgenden Einschränkungen:

- Die Eingabedateien für die Ladeoperation dürfen nicht auf einer Bandeinheit gespeichert sein.
- Die Option ROWCOUNT wird nur dann unterstützt, wenn der Modus ANALYZE verwendet wird.
- Wenn die Zieltabelle über eine Identitätsspalte verfügt, die zur Verteilung notwendig ist, und der Änderungswert `identityoverride` für den Dateityp nicht angegeben ist, oder wenn Sie mehrere Datenbankpartitionen verwenden, um die Daten zu verteilen und anschließend zu laden, wird ein größerer Wert als 0 für SAVECOUNT im Befehl LOAD nicht unterstützt.
- Wenn eine Identitätsspalte einen Teil des Verteilungsschlüssels darstellt, wird nur der Modus PARTITION_AND_LOAD unterstützt.
- Der Modus LOAD_ONLY und der Modus LOAD_ONLY_VERIFY_PART kann nicht mit der Option CLIENT des Befehls LOAD kombiniert werden.
- Der Modus LOAD_ONLY_VERIFY_PART kann nicht mit dem Eingabequellentyp CURSOR verwendet werden.
- Die Modi LOAD_ERRS_ONLY und SETUP_AND_LOAD_ERRS für die Isolation von Verteilungsfehlern können nicht zusammen mit den Optionen ALLOW READ ACCESS und COPY YES des Befehls LOAD verwendet werden.
- Mehrere Ladeoperationen können gleichzeitig Daten in dieselbe Tabelle laden, wenn sich die durch die Optionen OUTPUT_DBPARTNUMS und PARTITIONING_DBPARTNUMS angegebenen Datenbankpartitionen nicht überlappen. Beispiel: Wenn eine Tabelle auf den Datenbankpartitionen 0 bis 3 definiert ist,

kann eine Ladeoperation Daten auf die Datenbankpartitionen 0 und 1 laden, während eine zweite Ladeoperation Daten auf die Datenbankpartitionen 2 und 3 lädt.

- Nur ASC-Dateien (ASCII-Format mit universellen Zeilenbegrenzern) und DEL-Dateien (ASCII-Format ohne universelle Zeilenbegrenzer) können über Tabellen auf mehreren Datenbankpartitionen verteilt werden. PC/IXF-Dateien können nicht verteilt werden, Sie können eine PC/IXF-Datei jedoch in eine über mehrere Datenbankpartitionen verteilte Tabelle laden, indem Sie die Ladeoperation im Modus LOAD_ONLY_VERIFY_PART verwenden.

Die folgenden Beispiele veranschaulichen, wie Sie mit dem Befehl LOAD unterschiedliche Typen von Ladeoperationen einleiten können. Die in den folgenden Beispielen verwendete Datenbank enthält fünf Datenbankpartitionen: 0, 1, 2, 3 und 4. Jede Datenbankpartition verfügt über das lokale Verzeichnis `/db2/data/`. Die beiden Tabellen TABLE1 und TABLE2 sind auf den Datenbankpartitionen 0, 1, 3 und 4 definiert. Beim Laden von einem Client aus greift der Benutzer auf einen fernen Client zu, bei dem es sich nicht um eine der Datenbankpartitionen handelt.

Laden von einer Serverpartition

Beispiel für Verteilen und Laden

In diesem Szenario besteht eine Verbindung zu einer Datenbankpartition, auf der die Tabelle TABLE1 definiert sein kann oder auch nicht. Die Datendatei `load.del` befindet sich im aktuellen Arbeitsverzeichnis dieser Datenbankpartition. Um die Daten aus der Datei `load.del` auf alle Datenbankpartitionen zu laden, auf denen die Tabelle TABLE1 definiert ist, setzen Sie den folgenden Befehl ab:

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
```

Anmerkung: In diesem Beispiel werden für alle Konfigurationsparameter für Umgebungen mit partitionierten Datenbanken die Standardwerte verwendet:
Der Parameter MODE verwendet den Standardwert PARTITION_AND_LOAD, die Optionen OUTPUT_DBPARTNUMS nehmen alle Datenbankpartitionen als Standardwerte an, auf denen TABLE1 definiert ist, und PARTITIONING_DBPARTNUMS ist standardmäßig die Gruppe der Datenbankpartitionen, die gemäß den Regeln des Befehls LOAD ausgewählt wurde, welche zur Auswahl von Datenbankpartitionen gelten, wenn keine angegeben sind.

Um eine Ladeoperation auszuführen, bei der Daten über die Datenbankpartitionen 3 und 4 verteilt werden, setzen Sie den folgenden Befehl ab:

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
                  PARTITIONED DB CONFIG PARTITIONING_DBPARTNUMS (3,4)
```

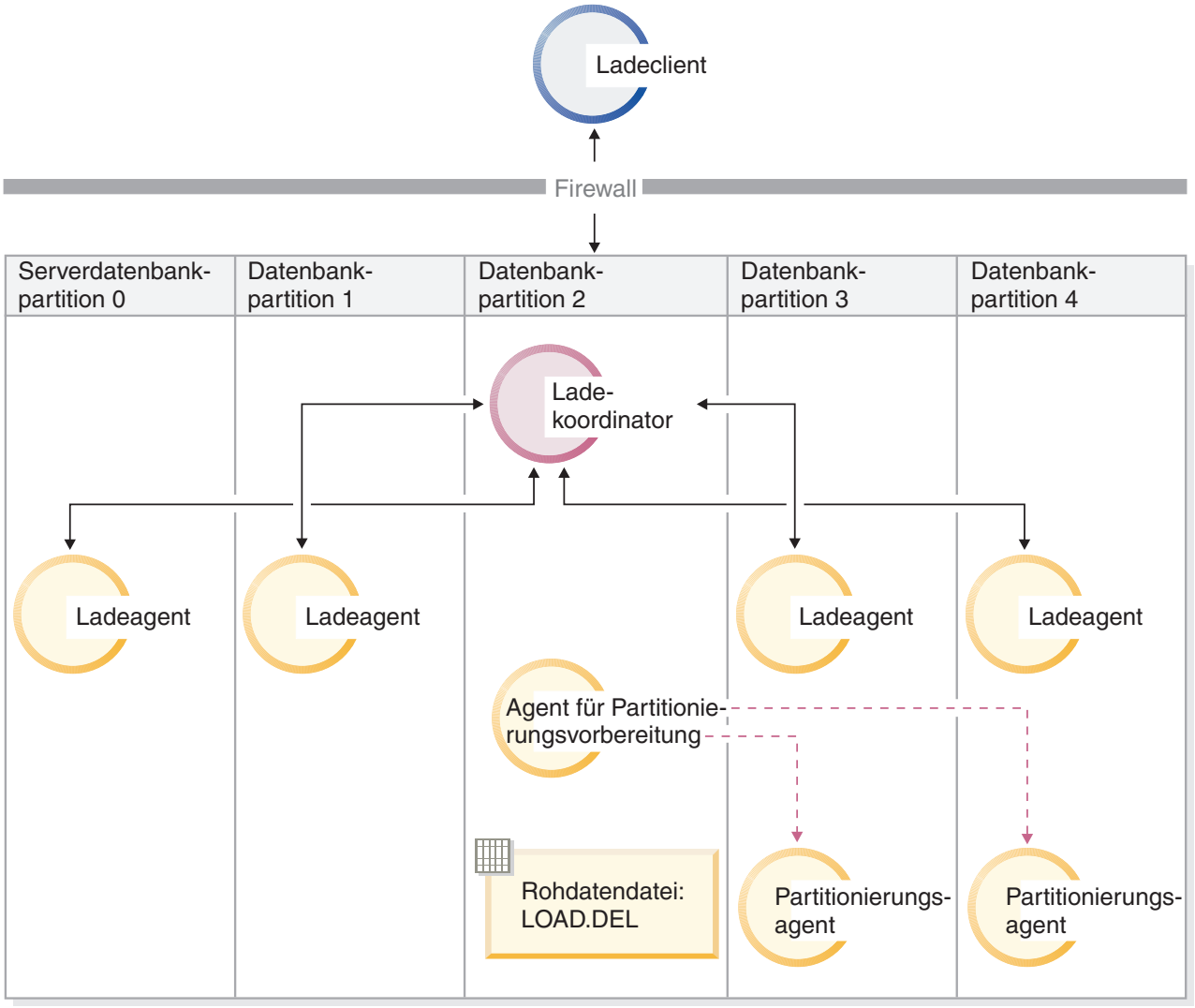


Abbildung 42. Diese Abbildung veranschaulicht das Verhalten, das sich bei Ausführung des vorstehenden Befehls ergibt. Die Daten werden in die Datenbankpartitionen 3 und 4 geladen.

Beispiel für Verteilen ohne Laden

In diesem Szenario besteht eine Verbindung zu einer Datenbankpartition, auf der die Tabelle TABLE1 definiert sein kann oder auch nicht. Die Datendatei `load.del` befindet sich im aktuellen Arbeitsverzeichnis dieser Datenbankpartition. Um die Datei `load.del` auf alle Datenbankpartitionen, auf denen TABLE1 definiert ist, zu verteilen (jedoch nicht zu laden) und dafür die Datenbankpartitionen 3 und 4 zu verwenden, setzen Sie den folgenden Befehl ab:

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE PARTITION_ONLY
                                              PART_FILE_LOCATION /db2/data
                                              PARTITIONING_DBPARTNUMS (3,4)
```

Dieser Befehl bewirkt, dass eine Datei `load.del.xxx` im Verzeichnis `/db2/data` auf jeder Datenbankpartition gespeichert wird. Hierbei steht `xxx` für eine dreistellige Darstellung der Datenbankpartitionsnummer.

Um die Datei `load.del` auf die Datenbankpartitionen 1 und 3 zu verteilen und hierfür nur einen auf der Partition 0 aktiven Partitionierungsagenten zu verwenden (Standardwert für PARTITIONING_DBPARTNUMS), verwenden Sie den folgenden Befehl:

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
                          PARTITIONED DB CONFIG MODE PARTITION_ONLY
                                                PART_FILE_LOCATION /db2/data
                                                OUTPUT_DBPARTNUMS (1,3)
```

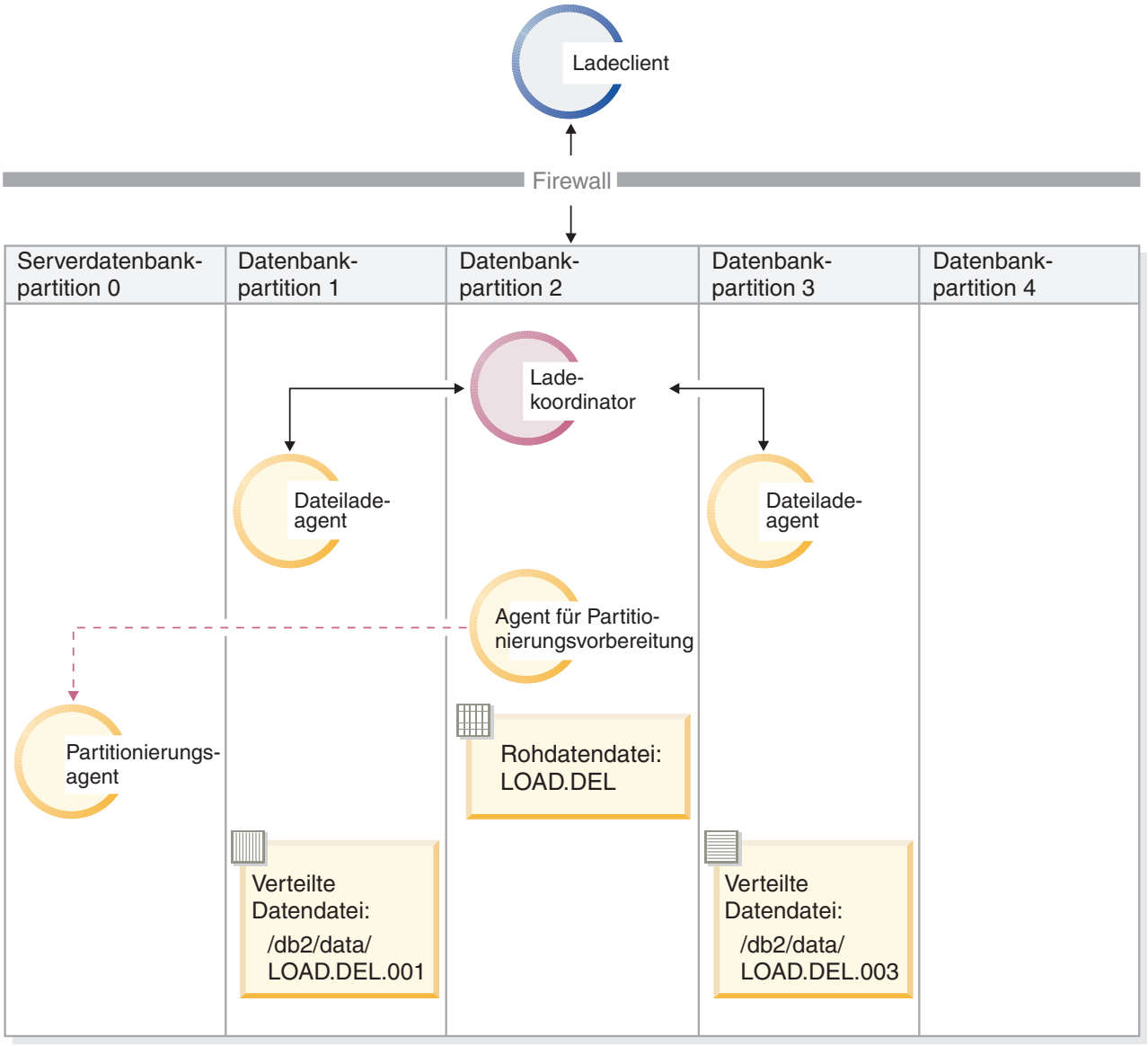


Abbildung 43. Diese Abbildung veranschaulicht das Verhalten, das sich bei Ausführung des vorstehenden Befehls ergibt. Die Daten werden auf die Datenbankpartitionen 1 und 3 geladen, wobei ein Partitionierungsagent verwendet wird, der auf Datenbankpartition 0 aktiv ist.

Beispiel für Laden ohne Verteilen

Wenn Sie bereits eine Ladeoperation im Modus PARTITION_ONLY ausgeführt haben und die partitionierten Dateien im Verzeichnis /db2/data jeder Ladedatenbankpartition auf alle Datenbankpartitionen laden wollen, auf denen TABLE1 definiert ist, setzen Sie den folgenden Befehl ab:

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE LOAD_ONLY
                                              PART_FILE_LOCATION /db2/data
```

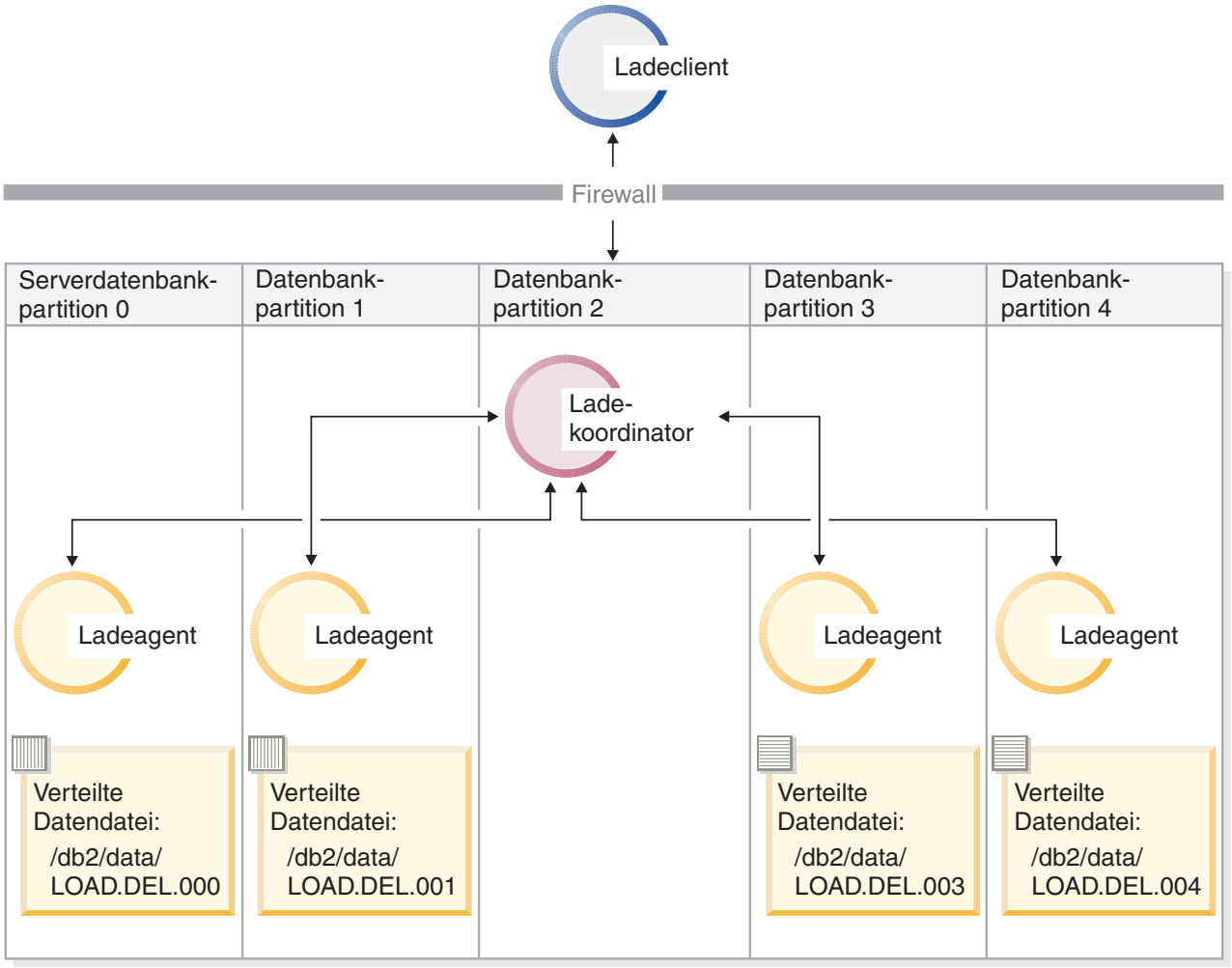


Abbildung 44. Diese Abbildung veranschaulicht das Verhalten, das sich bei Ausführung des vorstehenden Befehls ergibt. Die verteilten Daten werden auf alle Datenbankpartitionen geladen, auf denen TABLE1 definiert ist.

Um Daten nur auf Datenbankpartition 4 zu laden, setzen Sie den folgenden Befehl ab:

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE LOAD_ONLY
                                              PART_FILE_LOCATION /db2/data
                                              OUTPUT_DBPARTNUMS (4)
```

Laden von vorverteilten Dateien ohne Kopfdaten für Verteilungszuordnung

Mit dem Befehl LOAD können Sie Datendateien ohne Verteilungskopfdaten direkt auf mehrere Datenbankpartitionen laden. Wenn die Datendateien im Verzeichnis /db2/data aller Datenbankpartitionen, auf denen TABLE1 definiert ist, bereits vorhanden sind und ihr Name load.del.xxx lautet (hierbei ist xxx die Datenbankpartitionsnummer), können die Dateien mit dem folgenden Befehl geladen werden:

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
                       REPLACE INTO TABLE1
                       PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
                                             PART_FILE_LOCATION /db2/data
```

Um die Daten nur auf Datenbankpartition 1 zu laden, setzen Sie den folgenden Befehl ab:

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
                       REPLACE INTO TABLE1
                       PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
                                             PART_FILE_LOCATION /db2/data
                                             OUTPUT_DBPARTNUMS (1)
```

Anmerkung: Zeilen, die nicht zu der Datenbankpartition gehören, von der aus sie geladen wurden, werden zurückgewiesen und in die Speicherauszugsdatei gestellt, sofern eine Speicherauszugsdatei angegeben wurde.

Laden von einem fernen Client in eine Mehrpartitionsdatenbank

Um Daten in eine Mehrpartitionsdatenbank aus einer Datei zu laden, die sich auf einem fernen Client befindet, müssen Sie mit der Option CLIENT des Befehls LOAD angeben, dass sich die Datendatei nicht auf einer Serverpartition befindet. Beispiel:

```
LOAD CLIENT FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

Anmerkung: Der Modus LOAD_ONLY oder LOAD_ONLY_VERIFY_PART kann nicht zusammen mit der Option CLIENT verwendet werden.

Laden über einen Cursor

Wie bei Einzelpartitionsdatenbanken können Sie Daten über einen Cursor in eine Mehrpartitionsdatenbank laden. In diesem Beispiel muss beim Modus PARTITION_ONLY und LOAD_ONLY die Option PART_FILE_LOCATION einen vollständig qualifizierten Dateinamen angeben. Dieser Name ist der vollständig qualifizierte Basisdateiname der verteilten Dateien, die auf jeder Ausgabedatenbankpartition erstellt oder geladen werden. Wenn die Zieltabelle LOB-Spalten enthält, können mehrere Dateien mit dem angegebenen Basisnamen erstellt werden.

Um alle Zeilen in der Antwortgruppe der Anweisung SELECT * FROM TABLE1 auf eine Datei zu verteilen, die sich jeweils auf allen Datenbankpartitionen namens /db2/data/select.out.xxx befindet (xxx ist die Datenbankpartitionsnummer), damit diese Zeilen später in die Tabelle TABLE2 geladen werden können, setzen Sie die folgenden Befehle ab:

```
DECLARE C1 CURSOR FOR SELECT * FROM TABLE1

LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
                       PARTITIONED DB CONFIG MODE PARTITION_ONLY
                       PART_FILE_LOCATION /db2/data/select.out
```

Die durch die vorstehende Operation erzeugten Datendateien können anschließend durch Absetzen des folgenden Befehls LOAD geladen werden:

```
LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
                       PARTITIONED CB CONFIG MODE LOAD_ONLY
                       PART_FILE_LOCATION /db2/data/select.out
```

Überwachen einer Ladeoperation in einer Umgebung mit partitionierten Datenbanken mit LOAD QUERY

Während einer Ladeoperation in einer Umgebung mit partitionierten Datenbanken werden von einigen Ladeprozessen Nachrichtendateien auf den Datenbankpartitionen erstellt, auf denen sie ausgeführt werden.

In den Nachrichtendateien werden alle Informationen, Warnungen und Fehlernachrichten gespeichert, die während der Ausführung der Ladeoperation erzeugt wurden. Bei den Ladeprozessen, die vom Benutzer anzeigbare Nachrichtendateien erzeugen, handelt es sich um den Ladeagenten, den Agenten für die Partitionierungsvorbereitung und den Partitionierungsagenten. Der Inhalt der Nachrichtendatei steht erst nach Abschluss der Ladeoperation zur Verfügung.

Sie können während einer Ladeoperation eine Verbindung zu einzelnen Datenbankpartitionen herstellen und den Befehl LOAD QUERY für die Zieltabelle absetzen. Wird dieser Befehl über den Befehlszeilenprozessor (CLP) abgesetzt, zeigt er den Inhalt aller Nachrichtendateien an, die sich gegenwärtig für die im Befehl LOAD QUERY angegebene Tabelle auf dieser Datenbankpartition befinden.

Beispiel: Die Tabelle TABLE1 ist auf den Datenbankpartitionen 0 bis 3 in der Datenbank WSDB definiert. Sie sind mit Datenbankpartition 0 verbunden und setzen den folgenden Befehl LOAD ab:

```
load from load.del of del replace into table1 partitioned db config
    partitioning_dbpartnums (1)
```

Dieser Befehl leitet eine Ladeoperation ein, die das Ausführen von Ladeagenten auf den Datenbankpartitionen 0, 1, 2 und 3 umfasst. Außerdem wird ein Partitionierungsagent auf Datenbankpartition 1 und ein Agent für Partitionierungsvorbereitung auf Datenbankpartition 0 ausgeführt.

Datenbankpartition 0 enthält eine Nachrichtendatei für den Agenten für Partitionierungsvorbereitung und eine Nachrichtendatei für den Ladeagenten auf dieser Datenbankpartition. Um den Inhalt dieser Dateien gleichzeitig anzuzeigen, starten Sie eine neue Sitzung, und setzen Sie die folgenden Befehle am Befehlszeilenprozessor ab:

```
set client connect_node 0
connect to wsdb
load query table table1
```

Datenbankpartition 1 enthält eine Datei für den Ladeagenten und eine Datei für den Partitionierungsagenten. Um den Inhalt dieser Dateien anzuzeigen, starten Sie eine neue Sitzung, und setzen Sie die folgenden Befehle am Befehlszeilenprozessor ab:

```
set client connect_node 1
connect to wsdb
load query table table1
```

Anmerkung: Die durch die LOAD-Konfigurationsoption STATUS_INTERVAL generierten Nachrichten werden in der Nachrichtendatei des Agenten für die Partitionierungsvorbereitung angezeigt. Um diese Nachrichten während einer Ladeoperation anzuzeigen, müssen Sie eine Verbindung zur Koordinatorpartition herstellen und den Befehl LOAD QUERY absetzen.

Sichern des Inhalts von Nachrichtendateien

Wenn eine Ladeoperation über die API 'db2Load' eingeleitet wird, muss die Nachrichtenoption (piLocalMsgFileName) angegeben werden. Dann werden die Nachrichtendateien vom Server an den Client übertragen und dort gespeichert, damit Sie die Nachrichten anzeigen können.

Bei Ladeoperationen für Mehrpartitionsdatenbanken, die über den Befehlszeilenprozessor (CLP) eingeleitet werden, werden die Nachrichtendateien nicht auf der Konsole angezeigt oder beibehalten. Um den Inhalt dieser Dateien zu sichern oder anzuzeigen, nachdem das Laden in eine Mehrpartitionsdatenbank abgeschlossen ist, muss die Option MESSAGES des Befehls LOAD angegeben werden. Bei Verwendung dieser Option werden die Nachrichtendateien auf jeder Datenbankpartition nach Abschluss der Ladeoperation an die Clientmaschine übertragen und dort unter dem Basisnamen in Dateien gespeichert, der in der Option MESSAGES angegeben wurde. Die folgende Liste enthält die Dateinamen, die dem Ladeprozess, der die jeweilige Datei erstellt hat, entsprechen (bei Ladeoperationen für Mehrpartitionsdatenbanken):

Prozesstyp	Dateiname
Ladeagent	<message-file-name>.LOAD.<dbpartition-number>
Partitionierungsagent	<message-file-name>.PART.<dbpartition-number>
Agent für Partitionierungsvorbereitung	<message-file-name>.PREP.<dbpartition-number>

Wenn in der Option MESSAGES beispielsweise der Wert `/wsdb/messages/load` angegeben ist, lautet der Name der Nachrichtendatei des Ladeagenten auf Datenbankpartition 2 `/wsdb/messages/load.LOAD.002`.

Anmerkung: Es wird dringend empfohlen, bei Ladeoperationen für Mehrpartitionsdatenbanken, die über den Befehlszeilenprozessor (CLP) eingeleitet werden, die Option MESSAGES zu verwenden.

Fortsetzen, erneutes Starten oder Beenden von Ladeoperationen in einer Umgebung mit partitionierten Datenbanken

Die Arbeitsschritte, die in einer Umgebung mit partitionierten Datenbanken nach fehlgeschlagenen Ladeoperationen ausgeführt werden müssen, hängen davon ab, wann der Fehler aufgetreten ist.

Der Ladeprozess in einer Mehrpartitionsdatenbank besteht aus zwei Phasen:

1. In der *SETUP-Phase* werden Ressourcen auf Datenbankpartitionsebene angefordert. Hierzu gehören z. B. Tabellensperren für Datenbankpartitionen, die für die Ausgabe verwendet werden.

Wenn während der SETUP-Phase ein Fehler auftritt, sind in der Regel keine Neustart- oder Beendigungsoperationen erforderlich. Die erforderlichen Maßnahmen hängen von dem Fehlerisolationsmodus ab, der für die fehlgeschlagene Ladeoperation angegeben wurde.

Wenn für die Ladeoperation angegeben wurde, dass Fehler während der SETUP-Phase nicht isoliert werden sollen, wird die gesamte Ladeoperation abgebrochen und der Status der Tabelle wird auf allen Datenbankpartitionen mittels Rollback auf den Status zurückgesetzt, der vor der Ladeoperation gültig war.

Wenn für die Ladeoperation angegeben wurde, dass Fehler während der SETUP-Phase isoliert werden sollen, wird die Ladeoperation auf den Datenbankpartitionen fortgesetzt, auf denen die SETUP-Phase erfolgreich ausgeführt werden konnte. Die Tabelle auf den Datenbankpartitionen, auf denen ein Fehler auftrat, werden mittels Rollback auf den Status zurückgesetzt, der vor der Ladeoperation gültig war. Dies bedeutet, dass eine einzige Ladeoperation in unterschiedlichen Phasen fehlschlagen kann, wenn einige Partitionen während der SETUP-Phase und andere während der LOAD-Phase fehlschlagen.

2. Die *LOAD-Phase*, während der die Daten formatiert und in Tabellen auf den Datenbankpartitionen geladen werden.

   Wenn eine Ladeoperation für eine Mehrpartitionsdatenbank während der LOAD-Phase auf mindestens einer Datenbankpartition fehlschlägt, muss ein Befehl LOAD RESTART oder ein Befehl LOAD TERMINATE abgesetzt werden. Dies ist notwendig, weil das Laden von Daten in eine Mehrpartitionsdatenbank in einer einzigen Transaktion erfolgt.

   Sie sollten sich für einen Befehl LOAD RESTART entscheiden, wenn Sie die Probleme, die zum Fehlschlagen der Ladeoperation führten, beheben können. Dies spart Zeit. Wenn eine LOAD RESTART-Operation eingeleitet wird, wird die Ladeoperation auf allen Datenbankpartitionen dort fortgesetzt, wo sie abgebrochen wurde.

   Sie sollten sich für eine LOAD TERMINATE-Operation entscheiden, wenn die Tabelle in den Status zurückversetzt werden soll, in der sie sich vor der ursprünglichen Ladeoperation befand.

Vorgehensweise:

Ermitteln des Fehlerzeitpunkts einer Ladeoperation

Wenn Ihre Ladeoperation in einer partitionierten Umgebung fehlschlägt, müssen Sie zuerst ermitteln, auf welchen Partitionen der Fehler aufgetreten ist und in welcher Phase die Verarbeitung fehlschlug. Hierzu müssen Sie die Zusammenfassung für die Partition überprüfen. Wenn der Befehl LOAD über den Befehlszeilenprozessor (CLP) eingegeben wurde, wird die Partitionszusammenfassung am Ende des Ladevorgangs (siehe unten) angezeigt. Wenn der Befehl LOAD über die API db2Load eingegeben wurde, dann ist die Partitionszusammenfassung im Feld poAgentInfoList der Struktur db2PartLoadOut enthalten.

Wenn für "Agent Typ" für eine bestimmte Partition ein Eintrag "LOAD" vorhanden ist, hat diese Partition die LOAD-Phase erreicht. Andernfalls trat der Fehler während der SETUP-Phase auf. Ein negativer SQL-Code gibt an, dass ein Fehler aufgetreten ist. Im folgenden Beispiel schlug die Ladeoperation in Partition 1 während der LOAD-Phase fehl.

```
Agent Typ      Knoten   SQL-Code     Ergebnis
-------------------------------------------------------------
LOAD           000      +00000000    Erfolg.
-------------------------------------------------------------
```

```
   LOAD           001      -00000289    Fehler. Möglicherweise
Neustart (RESTART) erforderlich.
_____________________________________________________________________
   LOAD           002      +00000000    Erfolg.
   ___________________________________________________________________
   LOAD           003      +00000000    Erfolg.
.
.
.
```

Fortsetzen, erneutes Starten oder Beenden von fehlgeschlagenen Ladeoperationen

Nur Ladeoperationen mit der Option ISOLATE_PART_ERRS, für die SETUP_ERRS_ONLY oder SETUP_AND_LOAD_ERRS angegeben wurde, können während der SETUP-Phase fehlschlagen. Bei Ladeoperationen, für die auf mindestens einer Ausgabedatenbankpartition ein Fehler in dieser Phase aufgetreten ist, können Sie den Befehl LOAD REPLACE oder LOAD INSERT eingeben. Verwenden Sie die Option OUTPUT_DBPARTNUMS, um nur die Datenbankpartitionen anzugeben, auf denen ein Fehler aufgetreten ist.

Bei Ladeoperationen, für die auf mindestens einer Ausgabedatenbankpartition ein Fehler in der LOAD-Phase aufgetreten ist, müssen Sie den Befehl LOAD RESTART oder LOAD TERMINATE eingeben.

Bei Ladeoperationen, für die auf mindestens einer Ausgabedatenbankpartition in der SETUP-Phase und in der LOAD-Phase ein Fehler aufgetreten ist, müssen Sie zwei Ladeoperationen ausführen, um die fehlgeschlagene Ladeoperation fortzusetzen. Hierbei muss eine für die Fehler während der SETUP-Phase und eine für die Fehler während der LOAD-Phase verwendet werden. Um eine auf diese Weise fehlgeschlagene Ladeoperation rückgängig zu machen, müssen Sie den Befehl LOAD TERMINATE eingeben. Nach der Eingabe des Befehls müssen Sie allerdings alle Partitionen berücksichtigen, da in der Tabelle für keine der Partitionen, auf denen in der SETUP-Phase ein Fehler aufgetreten ist, Änderungen vorgenommen wurden, und weil alle Änderungen auf Partitionen rückgängig gemacht wurden, die während der LOAD-Phase fehlgeschlagen sind.

Beispiel: Die Tabelle TABLE1 ist auf den Datenbankpartitionen 0 bis 3 in der Datenbank WSDB definiert. Der folgende Befehl wird abgesetzt:

```
load from load.del of del insert into table1 partitioned db config
isolate_part_errs setup_and_load_errs
```

Während der SETUP-Phase tritt auf der Ausgabedatenbankpartition 1 ein Fehler auf. Da Fehler während der SETUP-Phase isoliert werden, wird die Ladeoperation fortgesetzt, während der LOAD-Phase tritt jedoch auf Partition 3 ebenfalls ein Fehler auf. Um die Ladeoperation fortzusetzen, sollten Sie die folgenden Befehle eingeben:

```
load from load.del of del replace into table1 partitioned db config
output_dbpartnums (1)

load from load.del of del restart into table1 partitioned db config
isolate_part_errs setup_and_load_errs
```

Anmerkung: Bei LOAD RESTART-Operationen gelten die im Befehl LOAD RESTART angegebenen Optionen. Es ist daher wichtig, in diesem Befehl dieselben Optionen wie im ursprünglichen Befehl LOAD anzugeben.

LOAD-Konfigurationsoptionen für Umgebungen mit partitionierten Datenbanken

MODE X

Gibt den Modus an, in dem die Ladeoperation beim Laden einer Mehrpartitionsdatenbank stattfindet. PARTITION_AND_LOAD ist der Standardwert. Gültige Werte:

- PARTITION_AND_LOAD. Die Daten werden verteilt (möglicherweise in Parallelverarbeitung) und gleichzeitig auf die entsprechenden Datenbankpartitionen geladen.
- PARTITION_ONLY. Die Daten werden verteilt (möglicherweise in Parallelverarbeitung), und die Ausgabe wird in Dateien an einer angegebenen Speicherposition auf jeder Ladedatenbankpartition geschrieben. Bei anderen Dateitypen als CURSOR lautet das Format des Namens der Ausgabedatei in der betreffenden Datenbankpartition `dateiname.xxx`, wobei `dateiname` der im Befehl LOAD angegebene Name der Eingabedatei und `xxx` die dreistellige Datenbankpartitionsnummer ist. Beim Dateityp CURSOR wird der Name der Ausgabedatei auf jeder Datenbankpartition durch die Option PART_FILE_LOCATION festgelegt. Der Abschnitt zur Option PART_FILE_LOCATION enthält weitere Informationen dazu, wie die Position der Verteilungsdatei für die betreffende Datenbankpartition angegeben wird.

  Anmerkung:

  1. Dieser Modus kann nicht für CLI-Ladeoperationen verwendet werden.
  2. Enthält die Tabelle eine Identitätsspalte, die für die Verteilung erforderlich ist, wird dieser Modus nicht unterstützt, es sei denn, der Änderungswert `identityoverride` für den Dateityp wird angegeben.
  3. Verteilungsdateien, die für den Dateityp CURSOR generiert werden, sind nicht mit verschiedenen DB2-Releases kompatibel. Dies bedeutet, dass Verteilungsdateien vom Dateityp CURSOR, die in einem Vorgängerrelease generiert wurden, nicht mit dem Modus LOAD_ONLY geladen werden können. Ebenso gilt, dass Verteilungsdateien vom Dateityp CURSOR, die im aktuellen Release generiert wurden, in einem zukünftigen Release nicht mit dem Modus LOAD_ONLY geladen werden können.
- LOAD_ONLY. Es wird davon ausgegangen, dass die Daten bereits verteilt sind. Der Verteilungsprozess wird übersprungen, und die Daten werden gleichzeitig auf die entsprechenden Datenbankpartitionen geladen. Bei anderen Dateitypen als CURSOR muss das Format des Namens der Eingabedatei für die betreffende Datenbankpartition `dateiname.xxx` lauten, wobei `dateiname` der im Befehl LOAD angegebene Name der Datei und `xxx` die dreistellige Datenbankpartitionsnummer ist. Beim Dateityp CURSOR wird der Name der Eingabedatei auf jeder Datenbankpartition durch die Option PART_FILE_LOCATION festgelegt. Der Abschnitt zur Option PART_FILE_LOCATION enthält weitere Informationen dazu, wie die Position der Verteilungsdatei für die betreffende Datenbankpartition angegeben wird.

  Anmerkung:

  1. Dieser Modus kann nicht für CLI-Ladeoperationen verwendet werden und nicht, wenn die Option CLIENT des Befehls LOAD angegeben ist.

2. Enthält die Tabelle eine Identitätsspalte, die für die Verteilung erforderlich ist, wird dieser Modus nicht unterstützt, es sei denn, der Änderungswert `identityoverride` für den Dateityp wird angegeben.

- LOAD_ONLY_VERIFY_PART. Es wird davon ausgegangen, dass die Daten bereits verteilt sind, aber die Datendatei enthält keine Partitionskopfdaten. Der Verteilungsprozess wird übersprungen, und die Daten werden gleichzeitig auf die entsprechenden Datenbankpartitionen geladen. Während der Ladeoperation wird für jede Zeile geprüft, ob sie sich auf der korrekten Datenbankpartition befindet. Zeilen, die Datenbankpartitionsverstöße enthalten, werden in eine Speicherauszugsdatei gestellt, sofern der Änderungswert `dumpfile` für den Dateityp angegeben wurde. Andernfalls werden die Zeilen gelöscht. Wenn für eine bestimmte Ladedatenbankpartition Datenbankpartitionsverstöße vorliegen, wird für die betreffende Datenbankpartition eine einzige Warnung in die LOAD-Nachrichtendatei geschrieben. Das Format des Namens der Eingabedatei für die betreffende Datenbankpartition muss `dateiname.xxx` lauten, wobei `dateiname` der im Befehl LOAD angegebene Name der Datei und `xxx` die dreistellige Datenbankpartitionsnummer ist. Der Abschnitt zur Option PART_FILE_LOCATION enthält weitere Informationen dazu, wie die Position der Verteilungsdatei für die betreffende Datenbankpartition angegeben wird.

  Anmerkung:

  1. Dieser Modus kann nicht für CLI-Ladeoperationen verwendet werden und nicht, wenn die Option CLIENT des Befehls LOAD angegeben ist.
  2. Enthält die Tabelle eine Identitätsspalte, die für die Verteilung erforderlich ist, wird dieser Modus nicht unterstützt, es sei denn, der Änderungswert `identityoverride` für den Dateityp wird angegeben.

- ANALYZE. Es wird eine optimale Verteilungszuordnung mit einer gleichmäßigen Verteilung auf alle Datenbankpartitionen generiert.

PART_FILE_LOCATION X

In den Modi PARTITION_ONLY, LOAD_ONLY und LOAD_ONLY_VERIFY_PART kann mit diesem Parameter die Position der verteilten Dateien angegeben werden. Diese Position muss auf jeder Datenbankpartition vorhanden sein, die mit der Option OUTPUT_DBPARTNUMS angegeben wurde. Handelt es sich bei der angegebenen Position um den Namen eines relativen Pfads, wird der Pfad an das aktuelle Verzeichnis angehängt, um die Position für die verteilten Dateien zu erstellen.

Beim Dateityp CURSOR muss diese Option angegeben werden, und die Position muss sich auf einen vollständig qualifizierten Dateinamen beziehen. Dieser Name ist der vollständig qualifizierte Basisdateiname der verteilten Dateien, die auf jeder Ausgabedatenbankpartition erstellt werden (beim Modus PARTITION_ONLY), bzw. die Position der Dateien, aus denen für jede Datenbankpartition gelesen werden soll (beim Modus LOAD_ONLY). Bei Verwendung des Modus PARTITION_ONLY können mehrere Dateien mit dem angegebenen Basisnamen erstellt werden, wenn die Zieltabelle LOB-Spalten enthält.

Bei anderen Dateitypen als CURSOR wird das aktuelle Verzeichnis für die verteilten Dateien verwendet, falls diese Option nicht angegeben ist.

OUTPUT_DBPARTNUMS X

`X` ist eine Liste mit Datenbankpartitionsnummern. Die Datenbankpartitionsnummern geben die Datenbankpartitionen an, auf denen die Ladeoperati-

on ausgeführt werden soll. Die Datenbankpartitionsnummern müssen eine Untermenge der Datenbankpartitionen sein, auf denen die Tabelle definiert ist. Standardmäßig werden alle Datenbankpartitionen ausgewählt. Die Liste muss in runde Klammern gesetzt werden. Die einzelnen Listeneinträge müssen mit einem Komma voneinander abgegrenzt werden. Die Angabe von Bereichen ist zulässig (Beispiel: (0, 2 to 10, 15)).

PARTITIONING_DBPARTNUMS X
X ist eine Liste der Datenbankpartitionsnummern, die im Verteilungsprozess verwendet werden. Die Liste muss in runde Klammern gesetzt werden. Die einzelnen Listeneinträge müssen mit einem Komma voneinander abgegrenzt werden. Die Angabe von Bereichen ist zulässig (Beispiel: (0, 2 to 10, 15)). Die für den Verteilungsprozess angegebenen Datenbankpartitionen können sich von den Datenbankpartitionen, die geladen werden, unterscheiden. Wird die Option PARTITIONING_DBPARTNUMS nicht angegeben, ermittelt das Dienstprogramm LOAD, wie viele Datenbankpartitionen erforderlich sind und welche Datenbankpartitionen im Hinblick auf eine optimale Leistung verwendet werden.

Falls der Änderungswert `anyorder` für den Dateityp im Befehl LOAD nicht angegeben wird, wird in der LOAD-Sitzung nur ein Partitionierungsagent verwendet. Des Weiteren wird - wenn nur eine Datenbankpartition für die Option OUTPUT_DBPARTNUMS angegeben wird oder die Koordinatorpartition der Ladeoperation kein Element von OUTPUT_DBPARTNUMS ist - die Koordinatorpartition der Ladeoperation im Verteilungsprozess verwendet. Andernfalls wird die erste in OUTPUT_DBPARTNUMS angegebene Datenbankpartition (nicht die Koordinatorpartition) im Verteilungsprozess verwendet.

Ist der Änderungswert `anyorder` für den Dateityp angegeben, wird die Anzahl der im Verteilungsprozess verwendeten Datenbankpartitionen wie folgt ermittelt: (Anzahl der Partitionen in OUTPUT_DBPARTNUMS geteilt durch 4) plus 1.

MAX_NUM_PART_AGENTS X
Gibt an, wie viele Partitionierungsagenten in einer LOAD-Sitzung maximal verwendet werden. Der Standardwert ist 25.

ISOLATE_PART_ERRS X
Gibt an, wie die Ladeoperation auf Fehler reagiert, die auf einzelnen Datenbankpartitionen auftreten. Der Standardwert ist LOAD_ERRS_ONLY, sofern nicht sowohl die Option ALLOW READ ACCESS als auch COPY YES des Befehls LOAD angegeben ist. In diesem Fall ist der Standardwert NO_ISOLATION. Gültige Werte:

- SETUP_ERRS_ONLY. Bei Fehlern, die während der SETUP-Phase auf einer Datenbankpartition auftreten (beispielsweise Fehler beim Zugriff auf eine Datenbankpartition oder Probleme beim Zugriff auf einen Tabellenbereich bzw. auf eine Tabelle auf einer Datenbankpartition), wird die Ladeoperation auf der Datenbankpartition mit dem Fehler gestoppt, auf den übrigen Datenbankpartitionen jedoch fortgesetzt. Fehler, die während des Ladens von Daten auf einer Datenbankpartition auftreten, bewirken das Fehlschlagen der gesamten Operation.
- LOAD_ERRS_ONLY. Fehler, die während der SETUP-Phase auf einer Datenbankpartition auftreten, bewirken das Fehlschlagen der gesamten Ladeoperation. Wenn beim Laden der Daten ein Fehler auftritt, stoppt die Ladeoperation für die Datenbankpartition, auf der der Fehler aufgetreten ist.

Die Ladeoperation wird für die verbleibenden Datenbankpartitionen fortgesetzt, bis ein Fehler auftritt oder bis alle Daten geladen wurden. Die neu geladenen Daten sind auf allen Datenbankpartitionen so lange nicht sichtbar, bis eine LOAD RESTART-Operation ausgeführt und erfolgreich beendet wird.

Anmerkung: Dieser Modus kann nicht verwendet werden, wenn sowohl die Option ALLOW READ ACCESS als auch COPY YES des Befehls LOAD angegeben ist.

- SETUP_AND_LOAD_ERRS. In diesem Modus führen Fehler auf Datenbankpartitionsebene während der SETUP-Phase oder beim Laden von Daten dazu, dass die Verarbeitung nur auf den betroffenen Datenbankpartitionen gestoppt wird. Wie im Modus LOAD_ERRS_ONLY sind die neu geladenen Daten - falls beim Laden der Daten Partitionsfehler auftreten - auf allen Datenbankpartitionen so lange nicht sichtbar, bis eine LOAD RESTART-Operation ausgeführt und erfolgreich beendet wird.

  Anmerkung: Dieser Modus kann nicht verwendet werden, wenn sowohl die Option ALLOW READ ACCESS als auch COPY YES des Befehls LOAD angegeben ist.
- NO_ISOLATION. In diesem Modus bewirkt jeder Fehler während der Ladeoperation, dass die Ladeoperation insgesamt fehlschlägt.

STATUS_INTERVAL X
X gibt an, wie häufig der Benutzer eine Benachrichtigung über den Umfang der gelesenen Daten empfängt. Die Maßeinheit ist Megabyte (MB). Der Standardwert 100 MB. Gültige Werte sind ganze Zahlen von 1 bis 4000.

PORT_RANGE X
X gibt den Bereich von TCP-Ports an, die zur Erstellung von Sockets für die interne Kommunikation verwendet werden. Der Standardbereich erstreckt sich von 6000 bis 6063. Wenn die Registrierdatenbankvariable **DB2ATLD_PORTS** zum Zeitpunkt des Aufrufs definiert ist, überschreibt ihr Wert den Wert, der für die LOAD-Konfigurationsoption PORT_RANGE angegeben ist. Der Bereich für die Registrierdatenbank-Variable **DB2ATLD_PORTS** sollte in dem folgenden Format angegeben werden:

```
<niedrige_portnummer:hohe_portnummer>
```

Das Format für den Befehlszeilenprozessor lautet:

```
( niedrige_portnummer, hohe_portnummer )
```

CHECK_TRUNCATION
Gibt an, dass das Programm bei der Ein-/Ausgabe prüfen soll, ob Datensätze abgeschnitten werden. In der Standardeinstellung werden die Daten bei der Ein-/Ausgabe nicht auf ein Abschneiden hin überprüft.

MAP_FILE_INPUT X
X gibt den Namen der Eingabedatei für die Verteilungszuordnung an. Dieser Parameter muss angegeben werden, wenn die Verteilungszuordnung angepasst wurde, da er auf die Datei verweist, in der sich die angepasste Verteilungszuordnung befindet. Eine angepasste Verteilungszuordnung kann erstellt werden, indem die Zuordnung mit dem Programm db2gpmap aus der Tabelle mit dem Datenbanksystemkatalog extrahiert wird oder indem mit dem Modus ANALYZE des Befehls LOAD eine optimale Zuordnung generiert wird. Die mit dem Modus ANALYZE generierte Zuordnung muss auf alle Datenbankpartitionen in der Datenbank versetzt werden, bevor die Ladeoperation fortgesetzt werden kann.

MAP_FILE_OUTPUT X

X ist der Name der Ausgabedatei für die Verteilungszuordnung. Die Ausgabedatei wird auf der Datenbankpartition erstellt, die den Befehl LOAD absetzt, wobei davon ausgegangen wird, dass diese Datenbankpartition zu der Datenbankpartitionsgruppe gehört, in der die Partitionierung stattfindet. Wird der Befehl LOAD auf einer Datenbankpartition aufgerufen, die nicht an der Partitionierung beteiligt ist (wie von PARTITIONING-_DBPARTNUMS definiert), wird die Ausgabedatei auf der ersten Datenbankpartition erstellt, die mit dem Parameter PARTITIONING_DBPARTNUMS definiert ist. Betrachten Sie das folgende Setup einer Umgebung mit partitionierten Datenbanken:

```
1 serv1 0
2 serv1 1
3 serv2 0
4 serv2 1
5 serv3 0
```

Bei Ausführung des folgenden LOAD-Befehls auf Server 3 (serv3) wird die Verteilungszuordnung auf Server 1 (serv1) erstellt.

```
LOAD FROM datei OF ASC METHOD L ( ...) INSERT INTO tabelle CONFIG
MODE ANALYZE PARTITIONING_DBPARTNUMS(1,2,3,4)
MAP_FILE_OUTPUT '/home/db2user/distribution.map'
```

Dieser Parameter sollte verwendet werden, wenn der Modus ANALYZE angegeben wird. Es wird eine optimale Verteilungszuordnung mit einer gleichmäßigen Verteilung auf alle Datenbankpartitionen generiert. Falls dieser Parameter nicht angegeben und der Modus ANALYZE angegeben ist, wird das Programm mit einem Fehler beendet.

TRACE X

Gibt die Anzahl der Datensätze an, für die ein Trace erstellt werden soll, wenn ein Speicherauszug des Datenkonvertierungsprozesses überprüft werden soll und eine Ausgabe der Hash-Werte erforderlich ist. Der Standardwert ist 0.

NEWLINE

Diese Option wird verwendet, wenn es sich bei der Eingabedatendatei um eine ASC-Datei handelt, in der jeder Datensatz durch ein Zeilenvorschubzeichen begrenzt wird, und der Änderungswert `reclen` für den Dateityp im Befehl LOAD angegeben wird. Bei Angabe dieser Option wird jeder Datensatz daraufhin geprüft, ob er ein Zeilenvorschubzeichen enthält. Die Satzlänge, wie im Änderungswert `reclen` für den Dateityp angegeben, wird außerdem geprüft.

DISTFILE X

Wird diese Option angegeben, generiert das Dienstprogramm LOAD eine Datenbankpartitionsverteilungsdatei mit dem angegebenen Namen. Die Datenbankpartitionsverteilungsdatei enthält 4096 Integers: ein Integer für jeden Eintrag in der Verteilungszuordnung der Zieltabelle. Jedes Integer in der Datei stellt die Anzahl der Zeilen in den geladenen Eingabedateien dar, für die zum entsprechenden Eintrag der Verteilungszuordnung ein Hashverfahren ausgeführt wird. Diese Informationen können Ihnen nicht nur dabei helfen, ungleiche Verteilungen in Ihren Daten zu ermitteln, sondern auch zu entscheiden, ob mithilfe des Modus ANALYZE des Dienstprogramms eine neue Verteilungszuordnung für die Tabelle generiert werden soll. Wenn diese Option nicht angegeben wird, ist es das Standardverhalten des Dienstprogramms LOAD, die Verteilungsdatei nicht zu generieren.

Anmerkung: Bei Angabe dieser Option wird maximal ein Partitionierungsagent für die Ladeoperation verwendet. Selbst wenn Sie explizit mehrere Partitionierungsagenten anfordern, wird nur einer verwendet.

OMIT_HEADER
Gibt an, dass die Verteilungsdatei keine Kopfdaten für die Verteilungszuordnung enthalten soll. Wenn diese Option nicht angegeben wird, werden Kopfdaten generiert.

RUN_STAT_DBPARTNUM X
Falls der Parameter STATISTICS YES im Befehl LOAD angegeben ist, werden Statistikdaten nur auf einer Datenbankpartition erfasst. Dieser Parameter gibt an, auf welcher Datenbankpartition Statistikdaten erfasst werden sollen. Wenn dieser Wert -1 lautet oder gar nicht angegeben wird, werden Statistikdaten auf der ersten Datenbankpartition in der Liste der Ausgabedatenbankpartitionen erfasst.

LOAD-Sitzungen in einer Umgebung mit partitionierten Datenbanken - CLP-Beispiele

In den folgenden Beispielen wird das Laden von Daten in eine Mehrpartitionsdatenbank dargestellt.

Die Datenbank hat vier Datenbankpartitionen, die von 0 bis 3 nummeriert sind. Die Datenbank WSDB ist auf allen Datenbankpartitionen definiert. Die Tabelle TABLE1 befindet sich in der Standarddatenbankpartitionsgruppe, die ebenfalls auf allen Datenbankpartitionen definiert ist.

Beispiel 1

Um Daten aus der Benutzerdatendatei `load.del`, die sich auf Datenbankpartition 0 befindet, in die Tabelle TABLE1 zu laden, stellen Sie eine Verbindung zur Datenbankpartition 0 her, und setzen Sie anschließend den folgenden Befehl ab:

```
load from load.del of del replace into table1
```

Wenn die Ladeoperation erfolgreich durchgeführt wird, erhalten Sie die folgende Ausgabe:

```
   Agent Typ      Knoten   SQL-Code     Ergebnis
_____________________________________________________
   LOAD           000      +00000000    Erfolg.
_____________________________________________________
   LOAD           001      +00000000    Erfolg.
_____________________________________________________
   LOAD           002      +00000000    Erfolg.
_____________________________________________________
   LOAD           003      +00000000    Erfolg.
_____________________________________________________
   PARTITION      001      +00000000    Erfolg.
_____________________________________________________
PRE_PARTITION  000      +00000000    Erfolg.
_____________________________________________________
   ERGEBNISSE:    4 von 4 LOAD-Operationen wurden
                erfolgreich beendet.
_____________________________________________________

   Zusammenfassung für Partitionierungsagenten
   Gelesene Zeilen                 = 100000
   Zurückgewiesene Zeilen          = 0
   Partitionierte Zeilen           = 100000
```

```
Zusammenfassung für LOAD-Agenten:
Anzahl gelesener Zeilen         = 100000
Anzahl übersprungener Zeilen     = 0
Anzahl geladener Zeilen         = 100000
Anzahl zurückgewiesener Zeilen   = 0
Anzahl gelöschter Zeilen         = 0
Anzahl festgeschriebener Zeilen = 100000
```

Die Ausgabe gibt an, dass sich auf jeder Datenbankpartition ein Ladeagent befand, der jeweils erfolgreich ausgeführt wurde. Außerdem zeigt die Ausgabe, dass auf der Koordinatorpartition ein Agent für Partitionierungsvorbereitung ausgeführt wurde sowie ein Partitionierungsagent auf Datenbankpartition 1. Diese Prozesse wurden erfolgreich mit einem normalen SQL-Rückkehrcode 0 abgeschlossen. Die Zusammenfassung der Statistikdaten ergibt, dass der Agent für Partitionierungsvorbereitung 100.000 Zeilen gelesen hat, dass der Partitionierungsagent 100.000 Zeilen verteilt hat und dass durch die Ladeagenten insgesamt 100.000 Zeilen geladen wurden.

Beispiel 2

Im folgenden Beispiel werden die Daten in die Tabelle TABLE1 geladen, wobei der Modus PARTITION_ONLY verwendet wird. Die verteilten Ausgabedateien werden auf allen Ausgabedatenbankpartitionen jeweils im Verzeichnis /db/data gespeichert:

```
load from load.del of del replace into table1 partitioned db config mode
partition_only part_file_location /db/data
```

Der Befehl LOAD hat die folgende Ausgabe:

```
   Agent Typ      Knoten   SQL-Code     Ergebnis
_____________________________________________________________
LOAD_TO_FILE   000      +00000000    Erfolg.
_____________________________________________________________
LOAD_TO_FILE   001      +00000000    Erfolg.
_____________________________________________________________
LOAD_TO_FILE   002      +00000000    Erfolg.
_____________________________________________________________
LOAD_TO_FILE   003      +00000000    Erfolg.
_____________________________________________________________
   PARTITION      001      +00000000    Erfolg.
_____________________________________________________________
PRE_PARTITION  000      +00000000    Erfolg.
_____________________________________________________________

   Zusammenfassung für Partitionierungsagenten
   Gelesene Zeilen                 = 100000
   Zurückgewiesene Zeilen          = 0
   Partitionierte Zeilen           = 100000
```

Die Ausgabe macht deutlich, dass auf jeder Ausgabedatenbankpartition jeweils ein Dateiladeagent aktiv war und dass diese Agenten erfolgreich ausgeführt wurden. Auf der Koordinatorpartition wurde ein Agent für Partitionierungsvorbereitung ausgeführt, und auf Datenbankpartition 1 wurde ein Partitionierungsagent ausgeführt. Die Zusammenfassung der Statistikdaten ergibt, dass der Agent für Partitionierungsvorbereitung 100.000 Zeilen erfolgreich gelesen hat und dass 100.000 Zeilen vom Partitionierungsagenten erfolgreich verteilt wurden. Da keine Zeilen in die Tabelle geladen wurden, wird keine Zusammenfassung über die Anzahl der geladenen Zeilen angezeigt.

Beispiel 3

Mit dem folgenden Befehl können Sie die Dateien laden, die mit der vorstehenden Ladeoperation im Modus PARTITION_ONLY generiert wurden:

```
load from load.del of del replace into table1 partitioned db config mode
 load_only part_file_location /db/data
```

Der Befehl LOAD hat die folgende Ausgabe:

```
   Agent Typ      Knoten   SQL-Code     Ergebnis
_____________________________________________________
   LOAD           000      +00000000    Erfolg.
_____________________________________________________
   LOAD           001      +00000000    Erfolg.
_____________________________________________________
   LOAD           002      +00000000    Erfolg.
_____________________________________________________
   LOAD           003      +00000000    Erfolg.
_____________________________________________________
   ERGEBNISSE:    4 von 4 LOAD-Operationen wurden
                 erfolgreich beendet.
_____________________________________________________

   Zusammenfassung für LOAD-Agenten:
   Anzahl gelesener Zeilen         = 100000
   Anzahl übersprungener Zeilen     = 0
   Anzahl geladener Zeilen         = 100000
   Anzahl zurückgewiesener Zeilen   = 0
   Anzahl gelöschter Zeilen         = 0
   Anzahl festgeschriebener Zeilen = 100000
```

Die Ausgabe besagt, dass die Ladeagenten auf allen Ausgabedatenbankpartitionen jeweils erfolgreich ausgeführt wurden und dass insgesamt 100.000 Zeilen durch alle Ladeagenten geladen wurden. Die Zusammenfassung enthält keine Angaben zu verteilten Zeilen, da keine Verteilung erfolgte.

Beispiel 4 - Fehlgeschlagene Ladeoperation

Beim Absetzen des LOAD-Befehls

```
load from load.del of del replace into table1
```

wird eventuell die folgende Ausgabe zurückgegeben, wenn während der Ladeoperation auf einer der Ladedatenbankpartitionen nicht genügend Speicherbereich im Tabellenbereich verfügbar ist:

```
   SQL0289N  Dem Tabellenbereich "DMS4KT" konnten keine neuen Seiten
zugeordnet werden.
   SQLSTATE=57011

   Agent Typ      Knoten   SQL-Code     Ergebnis
_____________________________________________________
   LOAD           000      +00000000    Erfolg.
_____________________________________________________
   LOAD           001      -00000289    Fehler. Möglicherweise
Neustart (RESTART) erforderlich.
_____________________________________________________
   LOAD           002      +00000000    Erfolg.
_____________________________________________________
   LOAD           003      +00000000    Erfolg.
_____________________________________________________
   PARTITION      001      +00000000    Erfolg.
_____________________________________________________
PRE_PARTITION  000      +00000000    Erfolg.
_____________________________________________________
```

```
   ERGEBNISSE:    3 von 4 LOAD-Operationen wurden
erfolgreich beendet.

_____________________________________________________________

   Zusammenfassung für Partitionierungsagenten
   Gelesene Zeilen                 = 0
   Zurückgewiesene Zeilen          = 0
   Partitionierte Zeilen           = 0

   Zusammenfassung für LOAD-Agenten:
   Anzahl gelesener Zeilen         = 0
   Anzahl übersprungener Zeilen     = 0
   Anzahl geladener Zeilen         = 0
   Anzahl zurückgewiesener Zeilen   = 0
   Anzahl gelöschter Zeilen         = 0
   Anzahl festgeschriebener Zeilen = 0
```

Die Ausgabe gibt an, dass die Ladeoperation den Fehler SQL0289 zurückgegeben hat. Die Zusammenfassung für die Datenbankpartitionen besagt, dass auf Datenbankpartition 1 nicht genügend Speicherbereich vorhanden war. Wird zusätzlicher Speicherbereich zu den Containern des Tabellenbereichs auf Datenbankpartition 1 hinzufügt, können Sie die Ladeoperation wie folgt erneut starten:

```
load from load.del of del restart into table1
```

Migration und Versionskompatibilität

Die Registrierdatenbankvariable **DB2_PARTITIONEDLOAD_DEFAULT** kann verwendet werden, um in einer Datenbank mit mehreren Partitionen zum Ladeverhalten von DB2 Universal Database vor Version 8 zurückzukehren.

Anmerkung: Ab Version 9.5 wird die Registrierdatenbankvariable DB2_PARTITIONEDLOAD_DEFAULT nicht weiter unterstützt und in einem höheren Release möglicherweise entfernt.

Durch die Rückkehr zu dem Verhalten des Befehls LOAD von DB2 UDB vor Version 8 in einer Datenbank mit mehreren Partitionen können Sie eine Datei mit gültigen Verteilungskopfdaten in eine einzige Datenbankpartition laden, ohne hierbei zusätzliche Konfigurationsoptionen für partitionierte Datenbanken anzugeben. Hierzu können Sie den Wert von **DB2_PARTITIONEDLOAD_DEFAULT** auf NO setzen. Die Verwendung dieser Option ist beispielsweise sinnvoll, wenn Sie Änderungen an vorhandenen Scripts vermeiden möchten, die den Befehl LOAD für einzelne Datenbankpartitionen absetzen. Um beispielsweise eine Verteilungsdatei in die Datenbankpartition 3 einer Tabelle zu laden, die zu einer Datenbankpartitionsgruppe mit vier Datenbankpartitionen gehört, setzen Sie den folgenden Befehl ab:

```
db2set DB2_PARTITIONEDLOAD_DEFAULT=NO
```

Anschließend setzen Sie die folgenden Befehle über den DB2-Befehlszeilenprozessor (CLP) ab:

```
CONNECT RESET

   SET CLIENT CONNECT_NODE 3

CONNECT TO DB MYDB

   LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

In einer Mehrpartitionsdatenbank findet die Ladeoperation für alle Datenbankpartitionen, in denen die Tabelle definiert ist, statt, wenn keine LOAD-Konfigurationsoptionen für das Laden in Mehrpartitionsdatenbanken angegeben werden. Die Eingabedatei muss keine Verteilungskopfdaten enthalten, und die Option MODE nimmt standardmäßig den Wert PARTITION_AND_LOAD an. Um das Laden für eine einzelne Datenbankpartition auszuführen, muss die Option OUTPUT-_DBPARTNUMS angegeben werden.

Kapitel 16. Migration von Umgebungen mit partitionierten Datenbanken

Migrieren von partitionierten Datenbanken

Die Migration von Umgebungen mit partitionierten Datenbanken setzt voraus, dass Sie DB2 Version 9.5 auf allen Datenbankpartitionsservern installieren. Anschließend migrieren Sie die Instanzen und dann die Datenbanken.

Datenbankpartitionsserver können vom Katalogdatenbankpartitionsserver oder von einem anderen Datenbankpartitionsserver aus migriert werden. Falls der Migrationsprozess fehlschlägt, können Sie die Migration vom Katalogdatenbankpartitionsserver oder von einem anderen Datenbankpartitionsserver aus erneut versuchen.

Da eine Migration dieser Art eine Operation von erheblicher Tragweite ist, würde eine Beschreibung der Migrationsprozedur sowie der Voraussetzungen und Einschränkungen über den Rahmen dieses Handbuchs hinausgehen. Eine detaillierte Beschreibung finden Sie im Thema „Migrieren von Umgebungen mit partitionierten Datenbanken" im Handbuch *Migration*. Dort finden Sie auch Querverweise auf zahlreiche weitere Themen, mit denen Sie sich vertraut machen sollten, bevor Sie eine Migration durchführen.

Kapitel 17. Verwenden von Momentaufnahme- und Ereignismonitoren

Überwachen der Reorganisation einer partitionierten Tabelle mit Snapshot Monitor-Daten

Im Folgenden werden einige der nützlichsten Methoden zur Überwachung des globalen Status einer Tabellenreorganisation erläutert.

Es gibt keine eigene Datengruppe, die den allgemeinen Status der Reorganisation einer partitionierten Tabelle angibt. Eine partitionierte Tabelle arbeitet mit einem Datenorganisationsschema, bei dem Tabellendaten auf mehrere Speicherobjekte, die als Datenpartitionen oder Datenbereiche (*RANGE*) bezeichnet werden, entsprechend den Werten einer oder mehrerer Spalten der Tabelle, die den Tabellenpartitionierungsschlüssel bilden, verteilt werden. Der globale Status einer Tabellenreorganisation lässt sich jedoch aus den Werten der Elemente in den einzelnen reorganisierten Datengruppen der Datenpartitionen ableiten. Im Folgenden werden einige der nützlichsten Methoden zur Überwachung des globalen Status einer Tabellenreorganisation erläutert.

Ermittlung der Anzahl der reorganisierten Datenpartitionen

Die Gesamtanzahl der für eine Tabelle reorganisierten Datenpartitionen lässt sich ermitteln, indem die Anzahl der Überwachungsdatenblöcke für Tabellendaten gezählt werden, die denselben Tabellen- und Schemanamen haben. Dieser Wert gibt die Anzahl der Datenpartitionen an, auf denen die Reorganisation gestartet wurde. Die Beispiele 1 und 2 zeigen, dass drei Partitionen reorganisiert werden.

Ermitteln der reorganisierten Datenpartition

Die Datenpartition, die zum jeweiligen Zeitpunkt gerade reorganisiert wird, lässt sich aus der Startzeit der Reorganisationsphase (reorg_phase_start) ableiten. Während der Phase SORT/BUILD/REPLACE geben die Überwachungsdaten der gerade reorganisierten Datenpartition die neueste Startzeit der betreffenden Phase an. Während der Phase INDEX_RECREATE (Index erneut erstellen) ist die Startzeit der Phase für alle Datenpartitionen gleich. In den Beispielen 1 und 2 wird die Phase INDEX_RECREATE angegeben, sodass die Startzeit für alle Datenpartitionen gleich ist.

Ermitteln des Bedarfs für eine Indexneuerstellung

Sie können ermitteln, ob ein Index erneut erstellt werden muss, indem Sie den Wert des Elements für die maximale Reorganisationsphase (reorg_max_phase) abrufen, das einer der reorganisierten Datenpartitionen entspricht. Hat das Element 'reorg_max_phase' einen Wert von 3 oder 4, ist eine Indexneuerstellung erforderlich. In den Beispielen 1 und 2 hat 'reorg_max_phase' den Wert 3; es ist also eine Indexneuerstellung erforderlich.

Die folgende Beispielausgabe stammt von einem Server mit drei Knoten, der eine Tabelle mit drei Datenpartitionen enthält:

```
CREATE TABLE  sales (c1 INT, c2 INT, c3 INT)
    PARTITION BY RANGE (c1)
        (PART P1 STARTING FROM (1)  ENDING AT (10) IN parttbs,
         PART P2 STARTING FROM (11) ENDING AT (20) IN parttbs,
         PART P3 STARTING FROM (21) ENDING AT (30) IN parttbs)
    DISTRIBUTE BY (c2)
```

Ausgeführte Anweisung:

```
REORG TABLE sales ALLOW NO ACCESS ON ALL DBPARTITIONNUMS
```

Beispiel 1:

```
GET SNAPSHOT FOR TABLES ON DPARTDB GLOBAL
```

Die Ausgabe wurde geändert und enthält lediglich die Informationen für die relevante Tabelle.

```
                        Momentaufnahme einer Tabelle

Zeitmarke für erste Datenbankverbindung = 06/28/2005 13:46:43.061690
Zeitmarke für letzte Zurücksetzung      = 06/28/2005 13:46:47.440046
Zeitmarke für Momentaufnahme            = 06/28/2005 13:46:50.964033
Datenbankname                               = DPARTDB
Datenbankpfad                               = /work/sales/NODE0000/SQL00001/
Aliasname der Eingabedatenbank              = DPARTDB
Anzahl Tabellen im Zugriff              = 5

Tabellenverzeichnis
  Tabellenschema         = NEWTON
  Tabellenname           = SALES
  Tabellenart            = User
  Datenpartitions-ID     = 0
  Datenobjektseiten      = 3
  Gelesene Zeilen        = 12
  Geschriebene Zeilen    = 1
  Überläufe              = 0
  Seitenreorganisationen = 0
  Informationen zur Tabellenreorganisation:
      Knotennummer                          = 0
      Reorganisationstyp                    =
                Wiederherstellen
                Tabellenreorganisation
                Keinen Zugriff zulassen
                Erneute Clustererstellung über Tabellensuche
                Nur Daten reorganisieren
      Reorganisationsindex                  = 0
      Tabellenbereich für Reorg.            = 3
      ID des temporären LOB-Speicherbereichs = 3
      Startzeit                             = 06/28/2005 13:46:49.816883
      Reorganisationsphase                  = 3 - Index erneut erstellen
      Max. Phase                            = 3
      Phasenstartzeit                       = 06/28/2005 13:46:50.362918
      Status                                = Beendet
      Aktueller Zähler                      = 0
      Max. Zähler                           = 0
      Beendigungsstatus                     = 0
      Endzeit                               = 06/28/2005 13:46:50.821244

  Informationen zur Tabellenreorganisation:
      Knotennummer                          = 1
      Reorganisationstyp                    =
                Wiederherstellen
                Tabellenreorganisation
                Keinen Zugriff zulassen
                Erneute Clustererstellung über Tabellensuche
                Nur Daten reorganisieren
```

```
    Reorganisationsindex                  = 0
    Tabellenbereich für Reorg.            = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                             = 06/28/2005 13:46:49.822701
    Reorganisationsphase                  = 3 - Index erneut erstellen
    Max. Phase                            = 3
    Phasenstartzeit                       = 06/28/2005 13:46:50.420741
    Status                                = Beendet
    Aktueller Zähler                      = 0
    Max. Zähler                           = 0
    Beendigungsstatus                     = 0
    Endzeit                               = 06/28/2005 13:46:50.899543


Informationen zur Tabellenreorganisation:
    Knotennummer                          = 2
    Reorganisationstyp                    =
              Wiederherstellen
              Tabellenreorganisation
              Keinen Zugriff zulassen
              Erneute Clustererstellung über Tabellensuche
              Nur Daten reorganisieren
    Reorganisationsindex                  = 0
    Tabellenbereich für Reorg.            = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                             = 06/28/2005 13:46:49.814813
    Reorganisationsphase                  = 3 - Index erneut erstellen
    Max. Phase                            = 3
    Phasenstartzeit                       = 06/28/2005 13:46:50.344277
    Status                                = Beendet
    Aktueller Zähler                      = 0
    Max. Zähler                           = 0
    Beendigungsstatus                     = 0
    Endzeit                               = 06/28/2005 13:46:50.803619


Tabellenschema         = NEWTON
Tabellenname           = SALES
Tabellenart            = User
Datenpartitions-ID     = 1
Datenobjektseiten      = 3
Gelesene Zeilen        = 8
Geschriebene Zeilen    = 1
Überläufe              = 0
Seitenreorganisationen = 0
Informationen zur Tabellenreorganisation:
    Knotennummer                          = 0
    Reorganisationstyp                    =
              Wiederherstellen
              Tabellenreorganisation
              Keinen Zugriff zulassen
              Erneute Clustererstellung über Tabellensuche
              Nur Daten reorganisieren
    Reorganisationsindex                  = 0
    Tabellenbereich für Reorg.            = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                             = 06/28/2005 13:46:50.014617
    Reorganisationsphase                  = 3 - Index erneut erstellen
    Max. Phase                            = 3
    Phasenstartzeit                       = 06/28/2005 13:46:50.362918
    Status                                = Beendet
    Aktueller Zähler                      = 0
    Max. Zähler                           = 0
    Beendigungsstatus                     = 0
    Endzeit                               = 06/28/2005 13:46:50.821244

Informationen zur Tabellenreorganisation:
```

```
    Knotennummer                          = 1
    Reorganisationstyp                    =
              Wiederherstellen
              Tabellenreorganisation
              Keinen Zugriff zulassen
              Erneute Clustererstellung über Tabellensuche
              Nur Daten reorganisieren
    Reorganisationsindex                  = 0
    Tabellenbereich für Reorg.            = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                             = 06/28/2005 13:46:50.026278
    Reorganisationsphase                  = 3 - Index erneut erstellen
    Max. Phase                            = 3
    Phasenstartzeit                       = 06/28/2005 13:46:50.420741
    Status                                = Beendet
    Aktueller Zähler                      = 0
    Max. Zähler                           = 0
    Beendigungsstatus                     = 0
    Endzeit                               = 06/28/2005 13:46:50.899543

Informationen zur Tabellenreorganisation:
    Knotennummer                          = 2
    Reorganisationstyp                    =
              Wiederherstellen
              Tabellenreorganisation
              Keinen Zugriff zulassen
              Erneute Clustererstellung über Tabellensuche
              Nur Daten reorganisieren
    Reorganisationsindex                  = 0
    Tabellenbereich für Reorg.            = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                             = 06/28/2005 13:46:50.006392
    Reorganisationsphase                  = 3 - Index erneut erstellen
    Max. Phase                            = 3
    Phasenstartzeit                       = 06/28/2005 13:46:50.344277
    Status                                = Beendet
    Aktueller Zähler                      = 0
    Max. Zähler                           = 0
    Beendigungsstatus                     = 0
    Endzeit                               = 06/28/2005 13:46:50.803619


Tabellenschema         = NEWTON
Tabellenname           = SALES
Tabellenart            = User
Datenpartitions-ID     = 2
Datenobjektseiten      = 3
Gelesene Zeilen        = 4
Geschriebene Zeilen    = 1
Überläufe              = 0
Seitenreorganisationen = 0
Informationen zur Tabellenreorganisation:
    Knotennummer                          = 0
    Reorganisationstyp                    =
              Wiederherstellen
              Tabellenreorganisation
              Keinen Zugriff zulassen
              Erneute Clustererstellung über Tabellensuche
              Nur Daten reorganisieren
    Reorganisationsindex                  = 0
    Tabellenbereich für Reorg.            = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                             = 06/28/2005 13:46:50.199971
    Reorganisationsphase                  = 3 - Index erneut erstellen
    Max. Phase                            = 3
    Phasenstartzeit                       = 06/28/2005 13:46:50.362918
    Status                                = Beendet
```

```
     Aktueller Zähler                       = 0
     Max. Zähler                            = 0
     Beendigungsstatus                      = 0
     Endzeit                                = 06/28/2005 13:46:50.821244

  Informationen zur Tabellenreorganisation:
     Knotennummer                           = 1
     Reorganisationstyp                     =
               Wiederherstellen
               Tabellenreorganisation
               Keinen Zugriff zulassen
               Erneute Clustererstellung über Tabellensuche
               Nur Daten reorganisieren
     Reorganisationsindex                   = 0
     Tabellenbereich für Reorg.             = 3
     ID des temporären LOB-Speicherbereichs = 3
     Startzeit                              = 06/28/2005 13:46:50.223742
     Reorganisationsphase                   = 3 - Index erneut erstellen
     Max. Phase                             = 3
     Phasenstartzeit                        = 06/28/2005 13:46:50.420741
     Status                                 = Beendet
     Aktueller Zähler                       = 0
     Max. Zähler                            = 0
     Beendigungsstatus                      = 0
     Endzeit                                = 06/28/2005 13:46:50.899543

  Informationen zur Tabellenreorganisation:
     Knotennummer                           = 2
     Reorganisationstyp                     =
               Wiederherstellen
               Tabellenreorganisation
               Keinen Zugriff zulassen
               Erneute Clustererstellung über Tabellensuche
               Nur Daten reorganisieren
     Reorganisationsindex                   = 0
     Tabellenbereich für Reorg.             = 3
     ID des temporären LOB-Speicherbereichs = 3
     Startzeit                              = 06/28/2005 13:46:50.179922
     Reorganisationsphase                   = 3 - Index erneut erstellen
     Max. Phase                             = 3
     Phasenstartzeit                        = 06/28/2005 13:46:50.344277
     Status                                 = Beendet
     Aktueller Zähler                       = 0
     Max. Zähler                            = 0
     Beendigungsstatus                      = 0
     Endzeit                                = 06/28/2005 13:46:50.803619
```

Beispiel 2:

```
GET SNAPSHOT FOR TABLES ON DPARTDB AT DBPARTITIONNUM 2
```

Die Ausgabe wurde geändert und enthält lediglich die Informationen für die relevante Tabelle.

```
                        Momentaufnahme einer Tabelle

Zeitmarke für erste Datenbankverbindung     = 06/28/2005 13:46:43.617833
Zeitmarke für letzte Zurücksetzung          =
Zeitmarke für Momentaufnahme                = 06/28/2005 13:46:51.016787
Datenbankname                               = DPARTDB
Datenbankpfad                               = /work/sales/NODE0000/SQL00001/
Aliasname der Eingabedatenbank              = DPARTDB
Anzahl Tabellen im Zugriff                  = 3

Tabellenverzeichnis
  Tabellenschema         = NEWTON
```

```
Tabellenname           = SALES
Tabellenart            = User
Datenpartitions-ID     = 0
Datenobjektseiten      = 1
Gelesene Zeilen        = 0
Geschriebene Zeilen    = 0
Überläufe              = 0
Seitenreorganisationen = 0
Informationen zur Tabellenreorganisation:
    Knotennummer                          = 2
    Reorganisationstyp                    =
              Wiederherstellen
              Tabellenreorganisation
              Keinen Zugriff zulassen
              Erneute Clustererstellung über Tabellensuche
              Nur Daten reorganisieren
    Reorganisationsindex                  = 0
    Tabellenbereich für Reorg.            = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                             = 06/28/2005 13:46:49.814813
    Reorganisationsphase                  = 3 - Index erneut erstellen
    Max. Phase                            = 3
    Phasenstartzeit                       = 06/28/2005 13:46:50.344277
    Status                                = Beendet
    Aktueller Zähler                      = 0
    Max. Zähler                           = 0
    Beendigungsstatus                     = 0
    Endzeit                               = 06/28/2005 13:46:50.803619


Tabellenschema         = NEWTON
Tabellenname           = SALES
Tabellenart            = User
Datenpartitions-ID     = 1
Datenobjektseiten      = 1
Gelesene Zeilen        = 0
Geschriebene Zeilen    = 0
Überläufe              = 0
Seitenreorganisationen = 0
Informationen zur Tabellenreorganisation:
    Knotennummer                          = 2
    Reorganisationstyp                    =
              Wiederherstellen
              Tabellenreorganisation
              Keinen Zugriff zulassen
              Erneute Clustererstellung über Tabellensuche
              Nur Daten reorganisieren
    Reorganisationsindex                  = 0
    Tabellenbereich für Reorg.            = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                             = 06/28/2005 13:46:50.006392
    Reorganisationsphase                  = 3 - Index erneut erstellen
    Max. Phase                            = 3
    Phasenstartzeit                       = 06/28/2005 13:46:50.344277
    Status                                = Beendet
    Aktueller Zähler                      = 0
    Max. Zähler                           = 0
    Beendigungsstatus                     = 0
    Endzeit                               = 06/28/2005 13:46:50.803619


Tabellenschema         = NEWTON
Tabellenname           = SALES
Tabellenart            = User
Datenpartitions-ID     = 2
Datenobjektseiten      = 1
Gelesene Zeilen        = 4
```

```
Geschriebene Zeilen    = 1
Überläufe              = 0
Seitenreorganisationen = 0
Informationen zur Tabellenreorganisation:
    Knotennummer                          = 2
    Reorganisationstyp                    =
              Wiederherstellen
              Tabellenreorganisation
              Keinen Zugriff zulassen
              Erneute Clustererstellung über Tabellensuche
              Nur Daten reorganisieren
    Reorganisationsindex                  = 0
    Tabellenbereich für Reorg.            = 3
    ID des temporären LOB-Speicherbereichs = 3
    Startzeit                             = 06/28/2005 13:46:50.179922
    Reorganisationsphase                  = 3 - Index erneut erstellen
    Max. Phase                            = 3
    Phasenstartzeit                       = 06/28/2005 13:46:50.344277
    Status                                = Beendet
    Aktueller Zähler                      = 0
    Max. Zähler                           = 0
    Beendigungsstatus                     = 0
    Endzeit                               = 06/28/2005 13:46:50.803619
```

Beispiel 3:

```
SELECT * FROM SYSIBMADM.SNAPLOCK WHERE tabname = 'SALES';
```

Die Ausgabe wurde geändert und enthält lediglich einen Teil der Informationen für die relevante Tabelle.

```
...   TBSP_NAME TABNAME LOCK_OBJECT_TYPE   LOCK_MODE  LOCK_STATUS ...
     --------- ------- ------------------ ---------- -----------
...   PARTTBS   SALES      ROW_LOCK        X          GRNT        ...
...   -         SALES      TABLE_LOCK      IX         GRNT        ...
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT        ...
...   PARTTBS   SALES      ROW_LOCK        X          GRNT        ...
...   -         SALES      TABLE_LOCK      IX         GRNT        ...
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT        ...
...   PARTTBS   SALES      ROW_LOCK        X          GRNT        ...
...   -         SALES      TABLE_LOCK      IX         GRNT        ...
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT        ...

 9 Satz/Sätze ausgewählt.
```

Ausgabe zu der Abfrage (Forts.)

```
... LOCK_ESCALATION LOCK_ATTRIBUTES DATA_PARTITION_ID DBPARTITIONNUM
   --------------- --------------- ------------------ --------------
...               0 INSERT          2                  2
...               0 NONE            -                  2
...               0 NONE            2                  2
...               0 INSERT          0                  0
...               0 NONE            -                  0
...               0 NONE            0                  0
...               0 INSERT          1                  1
...               0 NONE            -                  1
...               0 NONE            1                  1
```

Beispiel 4:

```
SELECT * FROM SYSIBMADM.SNAPTAB WHERE tabname = 'SALES';
```

Die Ausgabe wurde geändert und enthält lediglich einen Teil der Informationen für die relevante Tabelle.

```
... TABSCHEMA TABNAME TAB_FILE_ID TAB_TYPE   DATA_OBJECT_PAGES ROWS_WRITTEN ...
... --------- ------- ----------- ---------- ----------------- ------------ ...
... NEWTON    SALES             2 USER_TABLE                 1            1 ...
... NEWTON    SALES             4 USER_TABLE                 1            1 ...
... NEWTON    SALES             3 USER_TABLE                 1            1 ...

      3 Satz/Sätze ausgewählt.
```

Ausgabe zu der Abfrage (Forts.)

```
... OVERFLOW_ACCESSES PAGE_REORGS DBPARTITIONNUM TBSP_ID DATA_PARTITION_ID
... ----------------- ----------- -------------- ------- -----------------
...                 0           0              0       3                 0
...                 0           0              2       3                 2
...                 0           0              1       3                 1
```

Beispiel 5:

```
SELECT * FROM SYSIBMADM.SNAPTAB_REORG WHERE tabname = 'SALES';;
```

Die Ausgabe wurde geändert und enthält lediglich einen Teil der Informationen für die relevante Tabelle.

```
  REORG_PHASE   REORG_MAX_PHASE REORG_TYPE                                    ...
     ------------ -------------- -------------------------------------------
  INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
  INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
  INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
  INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
  INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
  INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
  INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
  INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
  INDEX_RECREATE              3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...

   9 Satz/Sätze ausgewählt.
```

Ausgabe zu der Abfrage (Forts.)

```
... REORG_STATUS REORG_TBSPC_ID DBPARTITIONNUM DATA_PARTITION_ID
    ------------- -------------- -------------- -----------------
... COMPLETED                3              2                 0
... COMPLETED                3              2                 1
... COMPLETED                3              2                 2
... COMPLETED                3              1                 0
... COMPLETED                3              1                 1
... COMPLETED                3              1                 2
... COMPLETED                3              0                 0
... COMPLETED                3              0                 1
... COMPLETED                3              0                 2
```

Globale Momentaufnahmen auf partitionierten Datenbanksystemen

Auf einem partitionierten Datenbanksystem können Sie eine Momentaufnahme der aktuellen Partition, einer bestimmten Partition oder aller Partitionen erstellen. Beim Erstellen einer globalen Momentaufnahme für alle Partitionen einer partitionierten Datenbank werden die Daten zusammengefasst, bevor die Ergebnisse zurückgegeben werden.

Die Daten werden für die verschiedenen Elementtypen wie folgt zusammengefasst:

- **Zähler, Zeit und Wertangaben**

  Die Daten enthalten die Summe aller gleichen Werte, die in den einzelnen Partitionen der Instanz erfasst wurden. Die Anweisung GET SNAPSHOT FOR DATABASE XYZ ON TEST GLOBAL beispielsweise würde die Anzahl der aus der Datenbank gelesenen Zeilen (rows_read) für alle Partitionen in der partitionierten Datenbankinstanz zurückgeben.

- **Grenzwerte**

  Gibt den höchsten Wert (bei oberer Grenze) bzw. den niedrigsten Wert (bei unterer Grenze) an, der in einer beliebigen Partition im partitionierten Datenbanksystem gefunden wurde. Gibt der zurückgegebene Wert Anlass zu Besorgnis, können Momentaufnahmen für die einzelnen Partitionen erstellt werden, um zu ermitteln, ob eine bestimmte Partition überlastet ist oder ob das Problem für die gesamte Instanz relevant ist.

- **Zeitmarke**

  Die Daten werden auf den Wert der Zeitmarke für die Partition gesetzt, mit der der Agent der Snapshot Monitor-Instanz verbunden ist. Bitte beachten Sie, dass sämtliche Zeitmarkenwerte vom Monitorschalter für `Zeitmarken` (TIMESTAMP) gesteuert werden.

- **Information**

  Gibt die jeweils höchstwertige Information für eine Partition zurück, die unter Umständen die Arbeit behindert. Beispiel für das Element `appl_status`: Ist der Status in einer Partition 'UOW wird ausgeführt' und in einer anderen 'Wartestatus für Sperre', würde 'Wartestatus für Sperre' zurückgegeben werden, da dies der Status ist, der die Ausführung der Anwendung verzögert.

Sie können auch Zähler zurücksetzen, Monitorschalter einstellen und Monitorschalterstellungen abrufen. Dies ist entweder für einzelnen Partitionen oder alle Partitionen in der partitionierten Datenbank möglich.

Anmerkung: Wird eine globale Momentaufnahme erstellt und tritt dabei in einer oder mehreren der Partitionen ein Fehler auf, werden von den Partitionen, in denen die Momentaufnahme erfolgreich war, die Daten erfasst, und gleichzeitig wird eine Warnung (SQLCODE-Wert 1629) zurückgegeben. Schlägt ein globaler Abruf bzw. eine globale Aktualisierung von Monitorschaltern oder das globale Zurücksetzen von Zählern in einer oder mehreren der Partitionen fehl, werden in den betreffenden Partitionen die Schalter nicht gesetzt bzw. die Daten nicht zurückgesetzt.

Erstellen eines Ereignismonitors für partitionierte Datenbanken

Beim Erstellen eines Datei- oder Pipe-Ereignismonitors für Systeme mit partitionierten Datenbanken müssen Sie den Umfang der zu erfassenden Überwachungsdaten festlegen.

Zum Erstellen von Ereignismonitoren für partitionierte Datenbanken benötigen Sie die Datenbankadministratorberechtigung (DBADM).

Ein Ereignismonitor verwendet zum Schreiben der Ereignisdatensätze einen Betriebssystemprozess oder -thread. Die Datenbankpartition, in der dieser Prozess oder Thread ausgeführt wird, wird als Monitorpartition bezeichnet. Datei- und Pipe-Ereignismonitore können Ereignisse bei deren Auftreten entweder lokal in der Monitorpartition überwachen oder global in allen Partitionen, in denen der DB2-Datenbankmanager ausgeführt wird. Ein globaler Ereignismonitor schreibt einen einzigen Trace für die Monitorpartition, der Aktivitäten aus allen Partitionen ent-

hält. Ob es sich bei einem Ereignismonitors um einen lokalen oder einen globalen Monitor handelt, wird durch den Überwachungsumfang (LOCAL bzw. GLOBAL) angegeben.

Sowohl die Monitorpartition als auch der Überwachungsumfang werden mit der Anweisung CREATE EVENT MONITOR angegeben.

Ein Ereignismonitor kann nur dann aktiviert werden, wenn die Monitorpartition aktiv ist. Wird die Anweisung SET EVENT MONITOR verwendet, um einen Ereignismonitor zu aktivieren, bevor die Monitorpartition aktiv ist, erfolgt die Aktivierung des Ereignismonitors beim nächsten Start der Monitorpartition. Darüber hinaus wird der Ereignismonitor automatisch so lange aktiviert, bis er explizit deaktiviert wird oder bis die Instanz explizit gestoppt wird. Beispiel für Datenbankpartition 0:

```
db2 connect to sample
db2 create event monitor foo ... on dbpartitionnum 2
db2 set event monitor foo state 1
```

Nach Ausführung der vorstehenden Befehle wird der Ereignismonitor `foo` automatisch aktiviert, sobald die Datenbank `sample` in Datenbankpartition 2 aktiviert wird. Diese automatische Aktivierung erfolgt so lange, bis der Befehl `db2 set event monitor foo state 0` abgesetzt wird oder Partition 2 gestoppt wird.

Bei Ereignismonitoren mit der Klausel WRITE TO TABLE ('in Tabelle schreiben') gibt es beim Überwachungsumfang keine Unterscheidung zwischen lokal und global. Wenn ein Ereignismonitor mit der Klausel WRITE TO TABLE ('in Tabelle schreiben') aktiviert wird, wird ein Ereignismonitorprozess in allen Partitionen ausgeführt. (Genauer gesagt: Der Ereignismonitorprozess wird in den Partitionen ausgeführt, die zu den Datenbankpartitionsgruppen gehören, in denen sich die Zieltabellen befinden.) Jede Partition, in der der Ereignismonitorprozess ausgeführt wird, verfügt außerdem über dieselbe Gruppe von Zieltabellen. Die Daten in diesen Tabellen sind jedoch unterschiedlich, da sie die Sicht der Überwachungsdaten der jeweiligen Partition darstellen. Es lassen sich zusammenfassende Werte aus allen Partitionen abrufen, indem SQL-Anweisungen abgesetzt werden, die auf die gewünschten Werte in den Zieltabellen des Ereignismonitors in den einzelnen Partitionen zugreifen.

Die erste Spalte heißt in allen Zieltabellen PARTITION_KEY und wird als Partitionierungsschlüssel für die Tabelle verwendet. Der Wert dieser Spalte wird so angegeben, dass jeder Ereignismonitorprozess Daten in die Datenbankpartition einfügt, in der der betreffende Prozess aktiv ist. Dies bedeutet, dass INSERT-Operationen zum Einfügen lokal in der Datenbankpartition ausgeführt werden, in der der Ereignismonitorprozess ausgeführt wird. In allen Datenbankpartitionen enthält das Feld PARTITION_KEY denselben Wert. Dies bedeutet Folgendes: Wenn eine Datenpartition gelöscht und eine Neuverteilung der Daten ausgeführt wird, werden alle Daten in der gelöschten Datenbankpartition in eine einzige andere Datenbankpartition gestellt und nicht gleichmäßig verteilt. Daher sollte vor dem Entfernen einer Datenbankpartition erwogen werden, alle Tabellenzeilen in der betreffenden Partition zu löschen.

Darüber hinaus kann für jede Tabelle eine Spalte namens PARTITION_NUMBER definiert werden. Diese Spalte enthält die Nummer der Partition, in die die Daten eingefügt wurden. Im Gegensatz zur Spalte PARTITION_KEY ist die Spalte PARTITION_NUMBER nicht obligatorisch.

Der Tabellenbereich, in dem Zieltabellen definiert sind, muss in allen Partitionen vorhanden sein, in die Ereignisüberwachungsdaten geschrieben werden. Eine Nichtbeachtung dieser Regel führt dazu, dass Datensätze in Partitionen (mit Ereignismonitoren), in denen der Tabellenbereich nicht vorhanden ist, nicht in das Protokoll geschrieben werden. In Partitionen, in denen der Tabellenbereich vorhanden ist, werden Ereignisdaten weiterhin geschrieben, und es wird kein Fehler zurückgegeben. Dieses Verhalten ermöglicht es Benutzern, eine Untergruppe von Partitionen für die Überwachung auszuwählen, indem ein Tabellenbereich erstellt wird, der nur in bestimmten Partitionen vorhanden ist.

Bei der Aktivierung von Ereignismonitoren mit der Klausel 'In Tabelle schreiben' (WRITE TO TABLE) werden die Zeilen FIRST_CONNECT und EVMON_START der Steuertabelle CONTROL nur in der Partition der Katalogdatenbank eingefügt. Dazu muss der Tabellenbereich für die Tabelle CONTROL in der Katalogdatenbankpartition vorhanden sein. Ist der Tabellenbereich in dieser Partition nicht vorhanden, werden diese Einfügungen nicht ausgeführt.

Ist eine Partition noch nicht aktiv, wenn ein Ereignismonitor mit der Klausel WRITE TO TABLE aktiviert wird, wird der Ereignismonitor aktiviert, sobald die betreffende Partition aktiviert wird.

Anmerkung: Die Sperrenliste bei einer Verbindung mit detaillierten Deadlocks enthält nur diejenigen Sperren, die von der Anwendung in der Partition gehalten werden, in der die Anwendung auf die Sperre wartet. Beispiel: Wartet eine an einem Deadlock beteiligte Anwendung auf eine Sperre in Knoten 20, enthält die Liste nur diejenigen Sperren, die von der Anwendung in Knoten 20 gehalten werden.

1. Geben Sie die Partition an, die überwacht werden soll.

   ```
   CREATE EVENT MONITOR dlmon FOR DEADLOCKS
                       WRITE TO FILE '/tmp/dlevents'
                                             ON PARTITION 3
   ```

   Hierbei gilt: `dlmon` ist der Name des Ereignismonitors;

   `/tmp/dlevents` ist der Name des Verzeichnispfads (unter UNIX), in dem der Ereignismonitor die Ereignisdateien schreiben soll;

   `3` ist die Nummer der zu überwachenden Partition.

2. Geben Sie an, ob die Ereignisüberwachungsdaten auf lokaler oder globaler Ebene (LOCAL bzw. GLOBAL) erfasst werden sollen. Um Ereignisüberwachungsberichte von allen Partitionen zu erfassen, setzen Sie die folgende Anweisung ab:

   ```
   CREATE EVENT MONITOR dlmon FOR DEADLOCKS
                       WRITE TO FILE '/tmp/dlevents'
                                             ON PARTITION 3 GLOBAL
   ```

   Nur Ereignismonitore für Deadlocks und Deadlocks mit Details können als GLOBAL definiert werden. Alle Partitionen melden Ereignisdatensätze im Zusammenhang mit Deadlocks an Partition 3.

3. Um Ereignisüberwachungsberichte nur von der lokalen Partition zu erfassen, setzen Sie die folgende Anweisung ab:

   ```
   CREATE EVENT MONITOR dlmon FOR DEADLOCKS
                       WRITE TO FILE '/tmp/dlevents'
                                             ON PARTITION 3 LOCAL
   ```

   Dies ist das Standardverhalten für Datei- und Pipe-Ereignismonitore in partitionierten Datenbanken. Die Klauseln LOCAL und GLOBAL werden für Ereignismonitore mit der Klausel 'In Tabelle schreiben' (WRITE TO TABLE) ignoriert.

4. Die Werte für die Monitorpartition und den Überwachungsumfang können für vorhandene Ereignismonitore überprüft werden. Hierfür wird die Tabelle SYSCAT.EVENTMONITORS mithilfe der folgenden Anweisung abgefragt:

   ```
   SELECT EVMONNAME, NODENUM, MONSCOPE FROM SYSCAT.EVENTMONITORS
   ```

Nachdem ein Ereignismonitor erstellt und aktiviert wurde, zeichnet er Überwachungsdaten zu den angegebenen Ereignissen auf, sobald diese eintreten.

Kapitel 18. Entwickeln einer geeigneten Backup- und Recovery-Strategie

Recovery nach Systemabsturz

Transaktionen (bzw. UOWs), die für eine Datenbank ausgeführt werden, können auf unerwartete Weise unterbrochen werden. Wenn eine Störung auftritt, bevor alle Änderungen, die Bestandteil der UOW (Unit of Work - Arbeitseinheit) sind, beendet und festgeschrieben wurden, verbleibt die Datenbank in einem inkonsistenten und unbrauchbaren Status. Die *Recovery nach einem Systemabsturz* versetzt die Datenbank wieder in einen konsistenten und verwendbaren Status. Dies geschieht durch Rollback der unvollständigen Transaktionen und Beenden der festgeschriebenen Aktionen, die sich zum Zeitpunkt des Systemabsturzes noch im Hauptspeicher befanden (Abb. 45). Wenn eine Datenbank sich in einem konsistenten und verwendbaren Status befindet, hat sie einen Punkt erreicht, der als „Konsistenzzustand" bezeichnet wird.

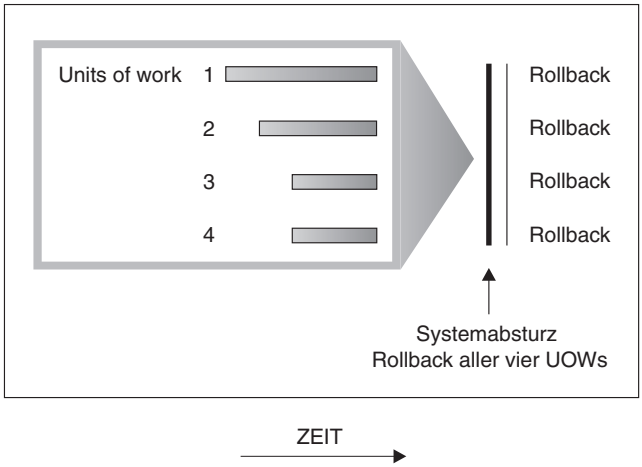


Abbildung 45. Rollback von UOWs (Recovery nach einem Systemabsturz)

Ein *Transaktionsfehler* ergibt sich aus einem schwer wiegenden Fehler bzw. einer Bedingung, die zur abnormalen Beendigung der Datenbank oder des Datenbankmanagers führt. Durch nur teilweise beendete oder zum Zeitpunkt des Fehlers noch nicht auf Platte geschriebene UOWs wird die Datenbank inkonsistent. Daher muss die Datenbank nach einem Transaktionsfehler wiederhergestellt werden. Folgende Bedingungen können zu einem Transaktionsfehler führen:

- Ein Stromausfall auf der Maschine, durch den der Datenbankmanager und die Datenbankpartitionen abnormal beendet werden
- Ein Hardwarefehler, z. B. Datenverlust im Hauptspeicher, oder ein Platten-, CPU- oder Netzwerkfehler
- Ein schwer wiegender Betriebssystemfehler, durch den DB2 abnormal beendet wird
- Eine Anwendung, die abnormal beendet wird

Wenn unvollständige UOWs vom Datenbankmanager automatisch rückgängig gemacht werden sollen, aktivieren Sie den Datenbankkonfigurationsparameter für automatischen Wiederanlauf (*autorestart*), indem Sie ihn auf `ON` setzen. (Dies ist der

Standardwert.) Wenn Sie den automatischen Neustart inaktivieren möchten, setzen Sie den Datenbankkonfigurationsparameter *autorestart* auf OFF. Ist der automatische Neustart inaktiviert, müssen Sie im Fall eines Datenbankfehlers den Befehl RESTART DATABASE absetzen. Wenn die Datenbank-E/A vor dem Auftreten des Absturzes ausgesetzt wurde, müssen Sie mit dem Befehl RESTART DATABASE die Option WRITE RESUME angeben, damit die Recovery nach dem Systemabsturz fortgesetzt wird. Der Neustart der Datenbank wird im Protokoll mit den Benachrichtigungen für die Systemverwaltung aufgezeichnet.

Wenn für eine Datenbank, für die die aktualisierende Recovery aktiviert ist (d. h., der Konfigurationsparameter *logarchmeth1* ist auf einen anderen Wert als OFF gesetzt), eine Recovery nach einem Systemabsturz durchgeführt wird und während dieser Recovery ein Fehler auftritt, der auf einen einzelnen Tabellenbereich zurückzuführen ist, wird dieser Tabellenbereich in den Offlinestatus versetzt. Auf ihn kann erst wieder zugegriffen werden, nachdem er repariert wurde.Die Recovery nach dem Systemabsturz wird fortgesetzt. Nach der Beendigung der Recovery nach Systemabsturz kann auf die anderen Tabellenbereiche in der Datenbank zugegriffen werden, und es können Verbindungen zur Datenbank hergestellt werden. Wenn jedoch der in den Offlinestatus versetzte Tabellenbereich die Systemkataloge enthält, muss er repariert werden, bevor Verbindungen hergestellt werden können.

Recovery nach Transaktionsfehlern in einer Umgebung mit partitionierten Datenbanken

Tritt ein Transaktionsfehler in einer Umgebung mit partitionierten Datenbanken auf, ist in der Regel eine Datenbankrecovery sowohl auf dem ausgefallenen Datenbankpartitionsserver als auch auf allen anderen, an der Transaktion beteiligten Datenbankpartitionsservern erforderlich:

- Eine Recovery nach Systemabsturz wird auf dem ausgefallenen Datenbankpartitionsserver durchgeführt, nachdem die vorausgegangene Bedingung korrigiert wurde.
- Die *Recovery der Datenbankpartitionen nach einem Fehler* erfolgt auf den anderen (weiterhin aktiven) Datenbankpartitionsservern unmittelbar nach Feststellung des Fehlers.

In einer Umgebung mit partitionierten Datenbanken ist der Datenbankpartitionsserver, auf dem die Anwendung übergeben wird, die Koordinatorpartition, und der erste Agent, der für die Anwendung aktiv wird, ist der Koordinatoragent. Der Koordinatoragent ist für die Verteilung der Arbeit auf andere Datenbankpartitionsserver verantwortlich und protokolliert, welche Datenbankpartitionsserver an der Transaktion beteiligt sind. Wenn die Anwendung eine COMMIT-Anweisung für eine Transaktion ausführt, schreibt der Koordinatoragent die Transaktion mithilfe des Protokolls zum zweiphasigen Commit fest. Während der ersten Phase sendet die Koordinatorpartition eine PREPARE-Anforderung an alle anderen an der Transaktion beteiligten Datenbankpartitionsserver. Die Server antworten daraufhin wie folgt:

READ-ONLY
Auf diesem Server gab es keine Datenänderung.

YES Auf diesem Server gab es eine Datenänderung.

NO Aufgrund eines Fehlers wurde der Server nicht für das Commit vorbereitet.

Wenn einer der Server mit NO antwortet, wird die Transaktion rückgängig gemacht. Andernfalls beginnt die Koordinatorpartition mit der zweiten Phase.

Während der zweiten Phase schreibt die Koordinatorpartition einen Commitprotokollsatz und sendet anschließend eine Commitanforderung an alle Server, die mit YES geantwortet haben. Wenn alle anderen Datenbankpartitionsserver das Commit durchgeführt haben, senden sie eine Bestätigung des Commits an die Koordinatorpartition. Die Transaktion ist abgeschlossen, wenn der Koordinatoragent von allen beteiligten Servern die Commitbestätigungen empfangen hat. Wenn dies der Fall ist, schreibt der Koordinatoragent einen FORGET-Protokollsatz.

Recovery auf einem aktiven Datenbankpartitionsserver nach Transaktionsfehler

Wenn ein Datenbankpartitionsserver feststellt, dass ein anderer Server inaktiv ist, werden alle Arbeiten, an denen der ausgefallene Datenbankpartitionsserver beteiligt ist, gestoppt:

- Wenn der noch aktive Datenbankpartitionsserver die Koordinatorpartition für eine Anwendung ist und die Anwendung auf dem ausgefallenen Datenbankpartitionsserver ausgeführt wird (und nicht zum Commit bereit ist), wird der Koordinatoragent unterbrochen, um Recoverymaßnahmen durchzuführen. Wenn der Koordinatoragent in der zweiten Phase der Commitverarbeitung ist, empfängt die Anwendung die Nachricht SQL0279N und verliert die Datenbankverbindung. Ansonsten sendet der Koordinatoragent eine Rollback-Anforderung an alle an der Transaktion beteiligten Server, und die Anwendung empfängt die Nachricht SQL1229N.
- Wenn der ausgefallene Datenbankpartitionsserver die Koordinatorpartition für die Anwendung war, werden Agenten, die noch für die Anwendung auf den aktiven Servern arbeiten, unterbrochen, um Recoverymaßnahmen durchzuführen. Die Transaktion wird lokal auf jeder Datenbankpartition rückgängig gemacht, auf der sich die Transaktion nicht im Status vorbereiteter Transaktionen befindet. Auf den Datenbankpartitionen, auf denen sich die Transaktion im Status vorbereiteter Transaktionen befindet, wird die Transaktion zu einer unbestätigten Transaktion. Die Koordinatordatenbankpartition ist nicht darüber informiert, dass die Transaktion auf einigen Datenbankpartitionen unbestätigt ist, weil sie nicht verfügbar ist.
- Wenn die Anwendung die Verbindung zu dem ausgefallenen Datenbankpartitionsserver (bevor er ausfiel) herstellte, aber weder der lokale Datenbankpartitionsserver noch der ausgefallene Datenbankpartitionsserver die Koordinatorpartition ist, werden Agenten, die für diese Anwendung aktiv sind, unterbrochen. Die Koordinatorpartition sendet eine Nachricht über eine Rollback-Operation oder eine Trennoperation (DISCONNECT) an die anderen Datenbankpartitionsserver. Die Transaktion ist nur auf den Datenbankpartitionsservern unbestätigt, die weiterhin aktiv sind, wenn die Koordinatorpartition die Nachricht SQL0279 zurückgibt.

Jeder Prozess (wie z. B. ein Agent oder ein Deadlock-Detektor), der versucht, eine Anforderung an den ausgefallenen Server zu senden, wird informiert, dass er die Anforderung nicht senden kann.

Recovery nach Transaktionsfehler auf dem ausgefallenen Datenbankpartitionsserver

Wenn der Transaktionsfehler zu einer abnormalen Beendigung des Datenbankmanagers führt, können Sie den Befehl db2start mit der Option RESTART angeben,

um den Datenbankmanager nach dem Neustart der Datenbankpartition erneut zu starten. Falls der Neustart der Datenbankpartition nicht möglich ist, können Sie den Befehl db2start auch in einer anderen Datenbankpartition zum Starten des Datenbankmanagers verwenden.

Die abnormale Beendigung des Datenbankmanagers kann zur Inkonsistenz einiger Datenbankpartitionen auf dem Server führen. Um diese Datenbankpartitionen wieder in einen konsistenten Status zu versetzen, kann auf einem Datenbankpartitionsserver wie folgt eine Recovery nach einem Systemabsturz ausgelöst werden:

- Explizit durch den Befehl RESTART DATABASE
- Implizit durch eine CONNECT-Anforderung, wenn der Datenbankkonfigurationsparameter *autorestart* auf `ON` gesetzt ist

Bei der Recovery nach einem Systemabsturz werden die Protokollsätze in den aktiven Protokolldateien erneut angewendet, um sicherzustellen, dass die Ergebnisse aller vollständigen Transaktionen in der Datenbank vorhanden sind. Nachdem alle Änderungen erneut angewendet wurden, werden alle nicht festgeschriebenen Transaktionen *außer* unbestätigten Transaktionen lokal rückgängig gemacht. In einer Umgebung mit partitionierten Datenbanken gibt es zwei Typen unbestätigter Transaktionen:

- Auf einem Datenbankpartitionsserver, der nicht die Koordinatorpartition ist, gilt eine Transaktion als unbestätigt, wenn sie zwar vorbereitet (PREPARE), aber noch nicht festgeschrieben (COMMIT) wurde.
- In der Koordinatorpartition ist eine Transaktion unbestätigt, wenn sie festgeschrieben (COMMIT), aber noch nicht als abgeschlossen protokolliert wurde (d. h., der Protokollsatz FORGET wurde noch nicht geschrieben). Diese Situation tritt ein, wenn der Koordinatoragent noch nicht alle Commitbestätigungen von allen Servern empfangen hat, die für die Anwendung aktiv waren.

Bei der Recovery nach einem Systemabsturz wird versucht, alle unbestätigten Transaktionen durch eine der folgenden Aktionen aufzulösen. Die durchgeführte Aktion ist davon abhängig, ob der Datenbankpartitionsserver die Koordinatorpartition für eine Anwendung war:

- Wenn der Server, der erneut gestartet wird, nicht die Koordinatorpartition für die Anwendung ist, sendet er eine Abfragenachricht an den Koordinatoragenten, um das Ergebnis der Transaktion festzustellen.
- Wenn der erneut gestartete Server die Koordinatorpartition für die Anwendung *ist*, sendet er eine Nachricht an alle anderen Agenten (untergeordneten Agenten), von denen der Koordinatoragent immer noch Commitbestätigungen erwartet.

Es ist möglich, dass durch eine Recovery nach einem Systemabsturz nicht alle unbestätigten Transaktionen aufgelöst werden können. Einige der Datenbankpartitionsserver sind z. B. möglicherweise nicht verfügbar. Wenn die Koordinatorpartition die Recovery nach einem Systemabsturz vor den anderen, an der Transaktion beteiligen Datenbankpartitonen abschließt, kann die Recovery nach einem Systemabsturz die unbestätigten Transaktionen nicht auflösen. Dies ist die erwartete Funktionsweise, da die Recovery nach einem Systemabsturz von jeder Datenbankpartition unabhängig durchgeführt wird. In diesem Fall wird die SQL-Warnung SQL1061W zurückgegeben. Unbestätigte Transaktionen belegen Ressourcen, z. B. Sperren und Speicherbereich für aktive Protokolle. Daher ist es möglich, dass ein Punkt erreicht wird, an dem keine Änderungen an der Datenbank mehr durchgeführt werden können, weil der Speicherbereich für die aktiven Protokolldateien durch unbestätigte Transaktionen belegt ist. Aus diesem Grund sollten Sie nach einer Recovery nach einem Systemabsturz feststellen, ob unbestätigte Transaktionen

verblieben sind, und alle Datenbankpartitionsserver, die zur Auflösung der unbestätigten Transaktionen erforderlich sind, so schnell wie möglich wieder verfügbar machen. In einigen Szenarios, in denen **LOCKTIMEOUT** -1 ist oder wenn der Parameter SET CURRENT LOCK TIMEOUT WAIT verwendet wird, können Sperren, die von unbestätigten Transaktionen verursacht werden, dazu führen, dass die Knoten-Recovery auf dem Koordinatorknoten nicht beendet werden kann. Diese Situation wird am besten dadurch aufgelöst, dass für sämtliche Datenbankpartitionsserver eine Recovery durchgeführt wird.

Anmerkung: In einer Umgebung mit partitionierten Datenbankservern wird der Befehl RESTART DATABASE für jeden Knoten einzeln ausgeführt. Wenn Sie sicherstellen wollen, dass die Datenbank auf allen Knoten erneut gestartet wird, müssen Sie den folgenden empfohlenen Befehl verwenden:

```
db2_all "db2 restart database <datenbankname>"
```

Wenn mindestens ein Server, der zum Auflösen einer unbestätigten Transaktion erforderlich ist, nicht rechtzeitig wiederhergestellt werden kann und auf Datenbankpartitionen auf anderen Servern Zugriff erforderlich ist, können Sie die unbestätigte Transaktion manuell auflösen, indem Sie eine heuristische Entscheidung treffen. Mithilfe des Befehls LIST INDOUBT TRANSACTIONS können Sie die unbestätigte Transaktion auf dem Server abfragen, festschreiben oder rückgängig machen.

Anmerkung: Der Befehl LIST INDOUBT TRANSACTIONS wird auch in einer verteilten Transaktionsumgebung verwendet. Zur Unterscheidung zwischen den beiden Typen unbestätigter Transaktionen enthält das Feld für die Quelle (*Originator*) in der Ausgabe des Befehls LIST INDOUBT TRANSACTIONS eine der folgenden Angaben:

- DB2 Enterprise Server Edition. Dies zeigt an, dass die Transaktion aus einer Umgebung mit partitionierten Datenbanken stammt.
- XA. Dies zeigt an, dass die Transaktion aus einer verteilten Umgebung stammt.

Identifizieren des ausgefallenen Datenbankpartitionsservers

Wenn ein Datenbankpartitionsserver ausfällt, empfängt die Anwendung normalerweise einen der folgenden SQLCODE-Werte. Die Methode zum Identifizieren des jeweils ausgefallenen Datenbankmanagers hängt vom empfangenen SQLCODE-Wert ab:

SQL0279N
: Dieser SQLCODE-Wert wird empfangen, wenn ein Datenbankpartitionsserver, der an einer Transaktion beteiligt ist, während der Commitverarbeitung beendet wird.

SQL1224N
: Dieser SQLCODE-Wert wird empfangen, wenn der ausgefallene Datenbankpartitionsserver die Koordinatorpartition für die Transaktion ist.

SQL1229N
: Dieser SQLCODE-Wert wird empfangen, wenn ein fehlgeschlagener Datenbankpartitionsserver nicht die Koordinatorpartition für die Transaktion ist.

Das Feststellen, welcher Datenbankpartitionsserver fehlgeschlagen ist, ist ein Prozess mit zwei Schritten. Der SQL-Kommunikationsbereich, der zum SQLCODE-Wert SQL1229N gehört, enthält in der sechsten Feldposition des Felds *sqlerrd* die Knotennummer des Servers, der den Fehler erkannte. (Die Knotennummer, die für den Server geschrieben wird, entspricht der Knotennummer in der Datei `db2nodes.cfg`.) Auf dem Datenbankpartitionsserver, der den Fehler feststellt, wird

eine Nachricht mit der Knotennummer des ausgefallenen Servers in das Protokoll mit den Benachrichtigungen für die Systemverwaltung geschrieben.

Anmerkung: Wenn auf einem Prozessor mehrere logische Knoten verwendet werden, kann der Ausfall eines einzigen logischen Knotens das Ausfallen anderer logischer Knoten auf demselben Prozessor verursachen.

Recovery nach dem Ausfall eines Datenbankpartitionsservers

Gehen Sie wie folgt vor, um eine Recovery nach dem Ausfall eines Datenbankpartitionsservers auszuführen.

1. Beheben Sie den Fehler, der den Ausfall verursachte.
2. Starten Sie den Datenbankmanager erneut, indem Sie auf einem beliebigen Datenbankpartitionsserver den Befehl db2start absetzen.
3. Starten Sie die Datenbank erneut, indem Sie auf dem bzw. den ausgefallenen Datenbankpartitionsserver(n) den Befehl RESTART DATABASE absetzen.

Erneutes Erstellen von partitionierten Datenbanken

Zur erneuten Erstellung (Rebuild) einer partitionierten Datenbank erstellen Sie jede Datenbankpartition separat erneut. Für jede Datenbankpartition, angefangen bei der Katalogpartition, stellen Sie zunächst alle Tabellenbereiche, die Sie benötigen, mit RESTORE wieder her. Alle Tabellenbereiche, die nicht wiederhergestellt werden, werden in den Status *Restore anstehend* versetzt. Wenn alle Datenbankpartitionen wiederhergestellt sind, setzen Sie den Befehl ROLLFORWARD DATABASE in der Katalogpartition ab, um eine aktualisierende Recovery aller Datenbankpartitionen auszuführen.

Anmerkung: Wenn Sie zu einem späteren Zeitpunkt Tabellenbereiche wiederherstellen müssen, die ursprünglich nicht in die Rebuildphase mit eingeschlossen waren, müssen Sie sicherstellen, dass bei der nachfolgenden aktualisierenden Recovery des Tabellenbereichs das Dienstprogramm für aktualisierende Recovery (Rollforward) alle Daten über die Datenbankpartitionen hinweg synchron hält. Wenn ein Tabellenbereich bei der ursprünglichen Restore- und aktualisierenden Recoveryoperation versehentlich unberücksichtigt bleibt, wird sein Fehlen unter Umständen erst erkannt, wenn bei einem Versuch, auf die Daten zuzugreifen, ein Datenzugriffsfehler auftritt. Sie müssen den fehlenden Tabellenbereich mit Restore wiederherstellen und eine aktualisierende Recovery für ihn ausführen, um ihn mit den übrigen Partitionen zu resynchronisieren.

Betrachten Sie das folgende Beispiel im Hinblick auf die erneute Erstellung einer partitionierten Datenbank mithilfe von Backup-Images, die auf Tabellenbereichsebene erstellt wurden.

Für dieses Beispiel wird angenommen, dass eine wiederherstellbare Datenbank mit dem Namen SAMPLE vorhanden ist und drei Datenbankpartitionen enthält:

- Datenbankpartition 1 enthält die Tabellenbereiche SYSCATSPACE, USERSP1 und USERSP2 und ist die Katalogpartition.
- Datenbankpartition 2 enthält die Tabellenbereiche USERSP1 und USERSP3.
- Datenbankpartition 3 enthält die Tabellenbereiche USERSP1, USERSP2 und USERSP3.

Die folgenden Backups wurden erstellt, wobei BKxy die Backupnummer x auf Partition y bezeichnet:

- BK11 ist ein Backup der Tabellenbereiche SYSCATSPACE, USERSP1 und USERSP2.
- BK12 ist ein Backup der Tabellenbereiche USERSP2 und USERSP3.
- BK13 ist ein Backup der Tabellenbereiche USERSP1, USERSP2 und USERSP3.
- BK21 ist ein Backup des Tabellenbereichs USERSP1.
- BK22 ist ein Backup des Tabellenbereichs USERSP1.
- BK23 ist ein Backup des Tabellenbereichs USERSP1.
- BK31 ist ein Backup des Tabellenbereichs USERSP2.
- BK33 ist ein Backup des Tabellenbereichs USERSP2.
- BK42 ist ein Backup des Tabellenbereichs USERSP3.
- BK43 ist ein Backup des Tabellenbereichs USERSP3.

Die folgende Prozedur veranschaulicht die Verwendung der Befehle RESTORE DATABASE und ROLLFORWARD DATABASE über den Befehlszeilenprozessor (CLP) zur erneuten Erstellung der gesamten Datenbank bis zum Ende der Protokolle.

1. Führen Sie in Datenbankpartition 1 einen Befehl RESTORE DATABASE mit der Option REBUILD aus:
   ```
   db2 restore db sample rebuild with all tablespaces in database
         taken at BK31 without prompting
   ```
2. Führen Sie in Datenbankpartition 2 einen Befehl RESTORE DATABASE mit der Option REBUILD aus:
   ```
   db2 restore db sample rebuild with tablespaces in database
         taken at BK42 without prompting
   ```
3. Führen Sie in Datenbankpartition 3 einen Befehl RESTORE DATABASE mit der Option REBUILD aus:
   ```
   db2 restore db sample rebuild with all tablespaces in database
         taken at BK43 without prompting
   ```
4. Führen Sie in der Katalogpartition einen Befehl ROLLFORWARD DATABASE mit der Option TO END OF LOGS aus:
   ```
   db2 rollforward db sample to end of logs
   ```
5. Führen Sie einen Befehl ROLLFORWARD DATABASE mit der Option STOP aus:
   ```
   db2 rollforward db sample stop
   ```

An diesem Punkt kann eine Verbindung zur Datenbank in allen Datenbankpartitionen hergestellt werden, und alle Tabellenbereiche befinden sich im Status NORMAL.

Datenrecovery mit 'db2adutl'

Die folgenden Beispiele zeigen, wie eine knotenübergreifende Fehlerbehebung mit dem Befehl db2adutl sowie mithilfe der Datenbankkonfigurationsparameter *logarchopt1* und *vendoropt* in verschiedenen TSM-Umgebungen (TSM = Tivoli Storage Manager) ausgeführt wird.

In den folgenden Beispielen besitzt Computer 1 den Namen `bar` und wird unter dem Betriebssystem AIX betrieben. Der Benutzer dieser Maschine ist `roecken`. Die Datenbank auf der Maschine `bar` heißt `zample`. Computer 2 besitzt den Namen `dps`. Dieser Computer wird ebenfalls unter dem Betriebssystem AIX von dem Benutzer `regress9` ausgeführt.

Beispiel 1: Automatische Kennwortverwaltung durch den TSM-Server (die Option PASSWORDACCESS ist auf GENERATE gesetzt)

Dieses Beispiel für knotenübergreifende Recovery zeigt die Vorgehensweise zum Einrichten von zwei Computern für die gegenseitige Datenrecovery, wenn die Protokollarchive und Backups auf einem TSM-Server gespeichert sind und die Kennwörter mit der Option PASSWORDACCESS=GENERATE verwaltet werden.

Anmerkung: Nach dem Aktualisieren der Datenbankkonfiguration müssen Sie möglicherweise ein Offline-Backup der Datenbank erstellen.

1. Um die Datenbank für Protokollarchivierung für den Computer `bar` auf dem TSM-Server zu aktivieren, aktualisieren Sie den Datenbankkonfigurationsparameter *logarchmeth1* für die Datenbank `zample` mit dem folgenden Befehl:

   ```
   bar:/home/roecken> db2 update db cfg for zample using LOGARCHMETH1 tsm
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   DB20000I  Der Befehl UPDATE DATABASE CONFIGURATION wurde erfolgreich ausgeführt.
   ```

2. Trennen Sie mit dem folgenden Befehl alle Benutzer und Anwendungen von der Datenbank:

   ```
   db2 force applications all
   ```

3. Stellen Sie mit dem folgenden Befehl sicher, dass keine Anwendungen mit der Datenbank verbunden sind:

   ```
   db2 list applications
   ```

   Sie sollten eine Nachricht empfangen, die besagt, dass keine Daten zurückgegeben wurden.

   Anmerkung: In einer Umgebung mit partitionierten Datenbanken müssen Sie diesen Schritt in allen Datenbankpartitionen ausführen.

4. Erstellen Sie mit dem folgenden Befehl ein Backup der Datenbank auf dem TSM-Server:

   ```
   db2 backup db zample use tsm
   ```

   Informationen ähnlich den folgenden werden zurückgegeben:

   ```
   Das Backup war erfolgreich. Die Zeitmarke für dieses Backup-Image ist: 20090216151025
   ```

   Anmerkung: In einer Umgebung mit partitionierten Datenbanken müssen Sie diesen Schritt in allen Datenbankpartitionen ausführen. Die Reihenfolge, in der Sie diesen Schritt an den Datenbankpartitionen ausführen, variiert, je nachdem, ob Sie ein Online-Backup oder ein Offline-Backup durchführen. Weitere Informationen hierzu finden Sie in „Verwenden von Backup" auf Seite 465.

5. Stellen Sie mit dem folgenden Befehl die Verbindung zur Datenbank `zample` her:

   ```
   db2 connect to zample
   ```

6. Generieren Sie mit dem folgenden Befehl neue Transaktionsprotokolle für die Datenbank durch Erstellen einer Tabelle und Laden von Daten in den TSM-Server:

   ```
   bar:/home/roecken> db2 load from mr of del modified by noheader replace
   into employee copy yes use tsm
   ```

   In diesem Beispiel hat die Tabelle den Namen `employee`, und die Daten werden aus einer Datei im ASCII-Format mit Begrenzern geladen, die den Namen `mr` hat. Die Option COPY YES wird angegeben, um eine Kopie der geladenen

Daten zu erstellen, und durch die Option USE TSM wird angegeben, dass die Kopie der Daten auf dem TSM-Server zu speichern ist.

Anmerkung: Sie können die Option COPY YES nur angeben, wenn die Datenbank für die aktualisierende Recovery konfiguriert ist. Das heißt, der Datenbankkonfigurationsparameter *logarchmeth1* muss auf den Wert USEREXIT, LOGRETAIN, DISK oder TSM gesetzt sein.

Das Dienstprogramm gibt eine Reihe von Nachrichten zurück, um den Verarbeitungsfortschritt anzuzeigen:

```
SQL3109N  Das Dienstprogramm beginnt mit dem Laden von Daten aus der Datei
"/home/roecken/mr".

SQL3500W  Die Phase "LOAD" wird gestartet (Zeit: "02/16/2009
15:12:13.392633").

SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs. Zähler für Eingabesätze: "0".

SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

SQL3110N  Die Verarbeitung des Dienstprogramms ist abgeschlossen. Es wurde(n) "1" Zeile(n)
aus der Eingabedatei gelesen.

SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs. Eingabesatzzähler = "1".

SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

SQL3515W  Die Phase "LOAD" wurde beendet (Zeit: "02/16/2009
15:12:13.445718").


   Anzahl gelesener Zeilen          = 1
   Anzahl übersprungener Zeilen    = 0
   Anzahl geladener Zeilen          = 1
   Anzahl zurückgewiesener Zeilen  = 0
   Anzahl gelöschter Zeilen        = 0
   Anzahl festgeschriebener Zeilen  = 1
```

7. Vergewissern Sie sich nach dem Laden der Daten in die Tabelle, dass auf dem TSM-Server ein Backup-Image, ein Ladekopieimage und eine Protokolldatei vorhanden ist, indem Sie die folgende Abfrage für die Datenbank `zample` ausführen:

   ```
   bar:/home/roecken/sqllib/adsm> db2adutl query db zample
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   Retrieving FULL DATABASE BACKUP information.
    1 Time: 20090216151025  Oldest log: S0000000.LOG  DB Partition Number: 0
           Sessions: 1

   Retrieving INCREMENTAL DATABASE BACKUP information.
       No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

   Retrieving DELTA DATABASE BACKUP information.
       No DELTA DATABASE BACKUP images found for ZAMPLE

   Retrieving TABLESPACE BACKUP information.
       No TABLESPACE BACKUP images found for ZAMPLE

   Retrieving INCREMENTAL TABLESPACE BACKUP information.
       No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

   Retrieving DELTA TABLESPACE BACKUP information.
       No DELTA TABLESPACE BACKUP images found for ZAMPLE

   Retrieving LOAD COPY information.
    1 Time: 20090216151213

   Retrieving LOG ARCHIVE information.
         Log file: S0000000.LOG, Chain Num: 0, DB Partition Number: 0,
   Taken at: 2009-02-16-15.10.38
   ```

8. Um die knotenübergreifende Recovery zu aktivieren, müssen Sie den Zugriff auf die zugeordneten Objekte des Computers bar für einen anderen Computer und ein anderes Benutzerkonto erteilen. Erteilen Sie in diesem Beispiel den Zugriff für den Computer dps und den Benutzer regress9 mit dem folgenden Befehl:

   ```
   bar:/home/roecken/sqllib/adsm> db2adutl grant user regress9
           on nodename dps for db zample
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   Successfully added permissions for regress9 to access ZAMPLE on node dps.
   ```

   Anmerkung: Sie können die Ergebnisse der GRANT-Operation für db2adutl überprüfen, indem Sie mit dem folgenden Befehl die aktuelle Zugriffsliste für den aktuellen Knoten abrufen:

   ```
   bar:/home/roecken/sqllib/adsm> db2adutl queryaccess
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   Node                Username            Database Name   Type
   ------------------------------------------------------------
   DPS                 regress9            ZAMPLE          A
   ------------------------------------------------------------
    Access Types:  B - backup images   L - logs   A - both
   ```

9. In diesem Beispiel ist der zweite Computer (dps) noch nicht für die knotenübergreifende Recovery der Datenbank zample konfiguriert. Vergewissern Sie sich mit dem folgenden Befehl, dass diesem Benutzer und diesem Computer keine Daten auf dem TSM-Server zugeordnet sind:

   ```
   dps:/home/regress9/sqllib/adsm> db2adutl query db zample
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
     --- Database directory is empty ---
   Warning: There are no file spaces created by DB2 on the ADSM server
   Warning: No DB2 backup images found in ADSM for any alias.
   ```

10. Rufen Sie mit dem folgenden Befehl vom TSM-Server eine Liste der Objekte für die Datenbank zample ab, die dem Benutzer roecken und dem Computer bar zugeordnet sind:

    ```
    dps:/home/regress9/sqllib/adsm> db2adutl query db zample nodename
              bar owner roecken
    ```

    Die folgenden Informationen werden zurückgegeben:

    ```
    --- Database directory is empty ---

  Query for database ZAMPLE

    Retrieving FULL DATABASE BACKUP information.
     1 Time: 20090216151025  Oldest log: S0000000.LOG  DB Partition Number: 0
            Sessions: 1

    Retrieving INCREMENTAL DATABASE BACKUP information.
        No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

    Retrieving DELTA DATABASE BACKUP information.
        No DELTA DATABASE BACKUP images found for ZAMPLE

    Retrieving TABLESPACE BACKUP information.
        No TABLESPACE BACKUP images found for ZAMPLE

    Retrieving INCREMENTAL TABLESPACE BACKUP information.
        No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

    Retrieving DELTA TABLESPACE BACKUP information.
        No DELTA TABLESPACE BACKUP images found for ZAMPLE

    Retrieving LOAD COPY information.
     1 Time: 20090216151213
    ```

```
Retrieving LOG ARCHIVE information.
      Log file: S0000000.LOG, Chain Num: 0, DB Partition Number: 0,
Taken at: 2009-02-16-15.10.38
```

Diese Informationen stimmen mit den zuvor generierten TSM-Informationen überein und bestätigen damit, dass dieses Image auf dem Computer dps wiederhergestellt werden kann.

11. Stellen Sie mit dem folgenden Befehl die Datenbank zample vom TSM-Server auf dem Computer dps wieder her:

```
dps:/home/regress9> db2 restore db zample use tsm options
   "'-fromnode=bar -fromowner=roecken'" without prompting
```

Die folgenden Informationen werden zurückgegeben:

```
DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.
```

Anmerkung: Wenn die Datenbank zample auf dps bereits vorhanden wäre, würde der Parameter OPTIONS weggelassen, und der Datenbankkonfigurationsparameter *vendoropt* würde verwendet. Dieser Konfigurationsparameter überschreibt den Parameter OPTIONS für eine BACKUP- oder RESTORE-Operation.

12. Führen Sie eine aktualisierende Wiederherstellung durch, um die Transaktionen anzuwenden, die beim Erstellen einer neuen Tabelle und beim Laden neuer Daten in der Protokolldatei für die Datenbank zample erfasst wurden. In diesem Beispiel schlägt die aktualisierende Wiederherstellung (ROLLFORWARD) in der Datenbank fehl, weil das Dienstprogramm für aktualisierende Wiederherstellung die Protokolldateien nicht finden kann, da keine Benutzer- und Computerinformationen angegeben sind:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

Der Befehl gibt den folgenden Fehler zurück:

```
SQL4970N  Die aktualisierende Recovery der Datenbank "ZAMPLE"
kann wegen fehlender Protokolldatei(en) auf Knoten "0" nicht den angegebenen
Endpunkt (Protokollende oder angegebener Zeitpunkt) erreichen.
```

Veranlassen Sie das Dienstprogramm für aktualisierende Wiederherstellung, die Protokolldateien für einen anderen Computer zu suchen, indem Sie den entsprechenden Wert für *logarchopt* angeben. Verwenden Sie in diesem Beispiel den folgenden Befehl, um den Datenbankkonfigurationsparameter *logarchopt1* festzulegen und nach Protokolldateien für den Benutzer roecken und den Computer bar zu suchen:

```
dps:/home/regress9> db2 update db cfg for zample using logarchopt1
   "'-fromnode=bar -fromowner=roecken'"
```

13. Ermöglichen Sie mit dem folgenden Befehl dem Dienstprogramm für aktualisierende Wiederherstellung die Verwendung der Backup- und Ladekopieimages, indem Sie den Datenbankkonfigurationsparameter *vendoropt* festlegen:

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-fromnode=bar -fromowner=roecken'"
```

14. Mit dem folgenden Befehl können Sie die knotenübergreifende Datenrecovery beenden, indem die in der Protokolldatei der Datenbank zample erfassten Transaktionen angewendet werden:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

Die folgenden Informationen werden zurückgegeben:

```
                              Status der aktualisierenden Recovery

 Aliasname der Eingabedatenbank             = zample
 Anzahl der Knoten, die Status zurückgaben  = 1

 Knotennummer                               = 0
```

```
   Aktualisierende Recovery: Status            = nicht anstehend
   Nächste zu lesende Protokolldatei           =
   Verarbeitete Protokolldateien               = S0000000.LOG - S0000000.LOG
Letzte festgeschriebene Transaktion            = 2009-02-16-20.10.38.000000 UTC

DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.
```

Die Datenbank zample auf dem Computer dps unter dem Benutzer regress9 wurde auf denselben Stand wie die Datenbank auf dem Computer bar unter dem Benutzer roecken wiederhergestellt.

Beispiel 2: Kennwortverwaltung durch den Benutzer (die Option PASSWORDACCESS ist auf PROMPT gesetzt)

Dieses Beispiel für knotenübergreifende Recovery zeigt die Vorgehensweise zum Einrichten von zwei Computern für die gegenseitige Datenrecovery, wenn die Protokollarchive und Backups auf einem TSM-Server gespeichert sind und die Kennwörter von den Benutzern verwaltet werden. In diesen Umgebungen sind zusätzliche Informationen erforderlich. Dies sind insbesondere der TSM-Knotenname und das Kennwort für den Computer, auf dem die Objekte erstellt wurden.

1. Fügen Sie in der Datei dsm.sys die folgende Zeile hinzu, weil der Quellencomputer den Namen bar trägt:

   ```
   NODENAME bar
   ```

   Anmerkung: Auf Windows-Betriebssystemen heißt diese Datei *dsm.opt*. Nach dem Aktualisieren dieser Datei müssen Sie für Ihr System einen Warmstart durchführen, damit die Änderungen wirksam werden.

2. Rufen Sie mit dem folgenden Befehl vom TSM-Server eine Liste der Objekte ab, die dem Benutzer roecken und dem Computer bar zugeordnet sind:

   ```
   dps:/home/regress9/sqllib/adsm> db2adutl query db zample nodename bar
      owner roecken password *******
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   Query for database ZAMPLE

      Retrieving FULL DATABASE BACKUP information.
       1 Time: 20090216151025  Oldest log: S0000000.LOG  DB Partition Number: 0
              Sessions: 1

      Retrieving INCREMENTAL DATABASE BACKUP information.
          No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

      Retrieving DELTA DATABASE BACKUP information.
          No DELTA DATABASE BACKUP images found for ZAMPLE

      Retrieving TABLESPACE BACKUP information.
          No TABLESPACE BACKUP images found for ZAMPLE

      Retrieving INCREMENTAL TABLESPACE BACKUP information.
          No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

      Retrieving DELTA TABLESPACE BACKUP information.
          No DELTA TABLESPACE BACKUP images found for ZAMPLE

      Retrieving LOAD COPY information.
       1 Time: 20090216151213

      Retrieving LOG ARCHIVE information.
            Log file: S0000000.LOG, Chain Num: 0, DB Partition Number: 0,
      Taken at: 2009-02-16-15.10.38
   ```

3. Führen Sie die folgenden Schritte aus, wenn die Datenbank zample auf dem Computer dps noch nicht vorhanden ist:

   a. Erstellen Sie mit dem folgenden Befehl eine leere Datenbank zample:

```
dps:/home/regress9> db2 create db zample
```

b. Aktualisieren Sie den Datenbankkonfigurationsparameter *tsm_nodename* mit dem folgenden Befehl:

```
dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
```

c. Aktualisieren Sie den Datenbankkonfigurationsparameter *tsm_password* mit dem folgenden Befehl:

```
dps:/home/regress9> db2 update db cfg for zample using
   tsm_password ********
```

4. Versuchen Sie mit dem folgenden Befehl, die Datenbank `zample` wiederherzustellen:

```
dps:/home/regress9> db2 restore db zample use tsm options
   "'-fromnode=bar -fromowner=roecken'" without prompting
```

Die Restoreoperation wird erfolgreich ausgeführt, jedoch wird eine Warnung ausgeben:

```
   SQL2540W  RESTORE wurde erfolgreich ausgeführt, es wurde jedoch eine
Warnung "2523" beim Restore der Datenbank im Modus
'No Interrupt' festgestellt.
```

5. Führen Sie mit dem folgenden Befehl eine aktualisierende Recovery durch:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

Weil in diesem Beispiel die Datenbankkonfigurationsdatei durch die Restoreoperation ersetzt wurde, kann das Dienstprogramm zur aktualisierenden Recovery nicht die richtigen Protokolldateien finden und die folgende Fehlernachricht wird zurückgegeben:

```
SQL1268N  Die aktualisierende Recovery wurde nach dem
Fehler "-2112880618" beendet, während die Protokolldatei "S0000000.LOG"
für die Datenbank "ZAMPLE" auf dem Knoten "0" abgerufen wurde.
```

Setzen Sie die folgenden TSM-Datenbankkonfigurationsparameter auf die richtigen Werte zurück:

a. Legen Sie mit dem folgenden Befehl den Konfigurationsparameter *tsm_nodename* fest:

```
dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
```

b. Legen Sie mit dem folgenden Befehl den Datenbankkonfigurationsparameter *tsm_password* fest:

```
dps:/home/regress9> db2 update db cfg for zample using tsm_password *******
```

c. Legen Sie den Datenbankkonfigurationsparameter *logarchopt1* so fest, dass vom Dienstprogramm für aktualisierende Recovery die richtigen Protokolldateien gefunden werden:

```
dps:/home/regress9> db2 update db cfg for zample using logarchopt1
   "'-fromnode=bar -fromowner=roecken'"
```

d. Legen Sie mit dem folgenden Befehl den Datenbankkonfigurationsparameter *vendoropt* so fest, dass die Recoverydatei auch während der aktualisierenden Recovery verwendet werden kann:

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-fromnode=bar -fromowner=roecken'"
```

6. Mit dem folgenden Befehl können Sie die knotenübergreifende Recovery durch das Ausführen der aktualisierenden Recovery beenden:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

Die folgenden Informationen werden zurückgegeben:

```
                              Status der aktualisierenden Recovery

 Aliasname der Eingabedatenbank             = zample
 Anzahl der Knoten, die Status zurückgaben  = 1

 Knotennummer                               = 0
 Aktualisierende Recovery: Status           = nicht anstehend
 Nächste zu lesende Protokolldatei          =
```

```
    Verarbeitete Protokolldateien               = S0000000.LOG - S0000000.LOG
Letzte festgeschriebene Transaktion             = 2009-02-16-20.10.38.000000 UTC

  DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.
```

Die Datenbank zample auf dem Computer dps unter dem Benutzer regress9 wurde auf denselben Stand wie die Datenbank auf dem Computer bar unter dem Benutzer roecken wiederhergestellt.

Beispiel 3: Konfigurieren des TSM-Servers für die Verwendung von Client-Proxy-Knoten

Dieses Beispiel für die knotenübergreifende Recovery zeigt die Vorgehensweise zum Einrichten von zwei Computern als Proxy-Knoten für die gegenseitige Datenrecovery, wenn die Protokollarchive und Backups auf einem TSM-Server gespeichert sind und die Kennwörter mit der Option PASSWORDACCESS=GENERATE verwaltet werden.

Wichtig: Der Parameter **OPTIONS** des Befehls db2adutl ist in Version 9.5 Fixpack 6 und späteren Fixpacks verfügbar. Zum Ausführen der unten angegebenen Schritte muss Version 9.5 Fixpack 6 (oder ein späteres Fixpack) installiert sein.

Anmerkung: Nach dem Aktualisieren der Datenbankkonfiguration müssen Sie möglicherweise ein Offline-Backup der Datenbank erstellen.

In diesem Beispiel werden die Computer bar und dps unter dem Proxy-Namen clusternode registriert. Die Computer sind bereits als Proxy-Knoten konfiguriert.

1. Registrieren Sie mit den folgenden Befehlen die Computer bar und dps als Proxy-Knoten auf dem TSM-Server:

   ```
   REGISTER NODE clusternode mypassword
   GRANT PROXYNODE TARGET=clusternode AGENT=bar,dps
   ```

2. Um die Datenbank für die Protokollarchivierung für den TSM-Server zu aktivieren, aktualisieren Sie den Datenbankkonfigurationsparameter *logarchmeth1* für die Datenbank zample mit dem folgenden Befehl:

   ```
   bar:/home/roecken> db2 update db cfg for zample using
      LOGARCHMETH1 tsm logarchopt1 "'-asnodename=clusternode'"
   ```

   Die folgenden Informationen werden zurückgegeben:

   ```
   DB20000I  Der Befehl UPDATE DATABASE CONFIGURATION wurde erfolgreich ausgeführt.
   ```

3. Trennen Sie mit dem folgenden Befehl alle Benutzer und Anwendungen von der Datenbank:

   ```
   db2 force applications all
   ```

4. Stellen Sie mit dem folgenden Befehl sicher, dass keine Anwendungen mit der Datenbank verbunden sind:

   ```
   db2 list applications
   ```

   Sie sollten eine Nachricht empfangen, die besagt, dass keine Daten zurückgegeben wurden.

   Anmerkung: In einer Umgebung mit partitionierten Datenbanken müssen Sie diesen Schritt in allen Datenbankpartitionen ausführen.

5. Erstellen Sie mit dem folgenden Befehl ein Backup der Datenbank auf dem TSM-Server:

   ```
   db2 backup db zample use tsm options "'-asnodename=clusternode'"
   ```

   Informationen ähnlich den folgenden werden zurückgegeben:

   ```
   Das Backup war erfolgreich. Die Zeitmarke für dieses Backup-Image ist: 20090216151025
   ```

Anstatt die Option '-asnodename' im Befehl 'backup' anzugeben, können Sie den Datenbankkonfigurationsparameter *vendoropt* aktualisieren.

Anmerkung: In einer Umgebung mit partitionierten Datenbanken müssen Sie diesen Schritt in allen Datenbankpartitionen ausführen. Die Reihenfolge, in der Sie diesen Schritt an den Datenbankpartitionen ausführen, variiert, je nachdem, ob Sie ein Online-Backup oder ein Offline-Backup durchführen. Weitere Informationen hierzu finden Sie in „Verwenden von Backup" auf Seite 465.

6. Stellen Sie mit dem folgenden Befehl die Verbindung zur Datenbank `zample` her:

   ```
   db2 connect to zample
   ```

7. Generieren Sie mit dem folgenden Befehl neue Transaktionsprotokolle für die Datenbank durch Erstellen einer Tabelle und Laden von Daten in den TSM-Server:

   ```
   bar:/home/roecken> db2 load from mr of del modified by noheader replace
   into employee copy yes use tsmwhere
   ```

   In diesem Beispiel hat die Tabelle den Namen `employee`, und die Daten werden aus einer Datei im ASCII-Format mit Begrenzern geladen, die den Namen `mr` hat. Die Option COPY YES wird angegeben, um eine Kopie der geladenen Daten zu erstellen, und durch die Option USE TSM wird angegeben, dass die Kopie der Daten auf dem TSM-Server zu speichern ist.

   Anmerkung: Sie können die Option COPY YES nur angeben, wenn die Datenbank für die aktualisierende Recovery konfiguriert ist. Das heißt, der Datenbankkonfigurationsparameter *logarchmeth1* muss auf den Wert USEREXIT, LOGRETAIN, DISK oder TSM gesetzt sein.

   Das Dienstprogramm gibt eine Reihe von Nachrichten zurück, um den Verarbeitungsfortschritt anzuzeigen:

   ```
   SQL3109N  Das Dienstprogramm beginnt mit dem Laden von Daten aus der Datei
   "/home/roecken/mr".

   SQL3500W  Die Phase "LOAD" wird gestartet (Zeit: "02/16/2009
   15:12:13.392633").

   SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs. Zähler für Eingabesätze: "0".

   SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

   SQL3110N  Die Verarbeitung des Dienstprogramms ist abgeschlossen. Es wurde(n) "1" Zeile(n)
   aus der Eingabedatei gelesen.

   SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs. Eingabesatzzähler = "1".

   SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.

   SQL3515W  Die Phase "LOAD" wurde beendet (Zeit: "02/16/2009
   15:12:13.445718").


      Anzahl gelesener Zeilen          = 1
      Anzahl übersprungener Zeilen     = 0
      Anzahl geladener Zeilen          = 1
      Anzahl zurückgewiesener Zeilen   = 0
      Anzahl gelöschter Zeilen         = 0
      Anzahl festgeschriebener Zeilen  = 1
   ```

8. Vergewissern Sie sich nach dem Laden der Daten in die Tabelle, dass auf dem TSM-Server ein Backup-Image, ein Ladekopieimage und eine Protokolldatei vorhanden ist, indem Sie die folgende Abfrage für die Datenbank `zample` ausführen:

```
bar:/home/roecken/sqllib/adsm> db2adutl query db zample
options "-asnodename=clusternode"
```

Die folgenden Informationen werden zurückgegeben:

```
Retrieving FULL DATABASE BACKUP information.
 1 Time: 20090216151025  Oldest log: S0000000.LOG  DB Partition Number: 0
        Sessions: 1

Retrieving INCREMENTAL DATABASE BACKUP information.
    No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

Retrieving DELTA DATABASE BACKUP information.
    No DELTA DATABASE BACKUP images found for ZAMPLE

Retrieving TABLESPACE BACKUP information.
    No TABLESPACE BACKUP images found for ZAMPLE

Retrieving INCREMENTAL TABLESPACE BACKUP information.
    No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

Retrieving DELTA TABLESPACE BACKUP information.
    No DELTA TABLESPACE BACKUP images found for ZAMPLE

Retrieving LOAD COPY information.
 1 Time: 20090216151213

Retrieving LOG ARCHIVE information.
      Log file: S0000000.LOG, Chain Num: 0, DB Partition Number: 0,
Taken at: 2009-02-16-15.10.38
```

9. In diesem Beispiel ist der zweite Computer (dps) noch nicht für die knotenübergreifende Recovery der Datenbank zample konfiguriert. Vergewissern Sie sich mit dem folgenden Befehl, dass diesem Benutzer und diesem Computer keine Daten zugeordnet sind:

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample
```

Die folgenden Informationen werden zurückgegeben:

```
  --- Database directory is empty ---
Warning: There are no file spaces created by DB2 on the ADSM server
Warning: No DB2 backup images found in ADSM for any alias.
```

10. Rufen Sie mit dem folgenden Befehl vom TSM-Server eine Liste der Objekte für die Datenbank zample ab, die dem Proxy-Knoten clusternode zugeordnet sind:

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample
   options="-asnodename=clusternode"
```

Die folgenden Informationen werden zurückgegeben:

```
--- Database directory is empty ---

Query for database ZAMPLE

Retrieving FULL DATABASE BACKUP information.
 1 Time: 20090216151025  Oldest log: S0000000.LOG  DB Partition Number: 0
        Sessions: 1

Retrieving INCREMENTAL DATABASE BACKUP information.
    No INCREMENTAL DATABASE BACKUP images found for ZAMPLE

Retrieving DELTA DATABASE BACKUP information.
    No DELTA DATABASE BACKUP images found for ZAMPLE

Retrieving TABLESPACE BACKUP information.
    No TABLESPACE BACKUP images found for ZAMPLE

Retrieving INCREMENTAL TABLESPACE BACKUP information.
    No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE

Retrieving DELTA TABLESPACE BACKUP information.
    No DELTA TABLESPACE BACKUP images found for ZAMPLE
```

```
Retrieving LOAD COPY information.
 1 Time: 20090216151213

Retrieving LOG ARCHIVE information.
      Log file: S0000000.LOG, Chain Num: 0, DB Partition Number: 0,
Taken at: 2009-02-16-15.10.38
```

Diese Informationen stimmen mit den zuvor generierten TSM-Informationen überein und bestätigen damit, dass dieses Image auf dem Computer dps wiederhergestellt werden kann.

11. Stellen Sie mit dem folgenden Befehl die Datenbank zample vom TSM-Server auf dem Computer dps wieder her:

    ```
    dps:/home/regress9> db2 restore db zample use tsm options
       "'-asnodename=clusternode'" without prompting
    ```

    Die folgenden Informationen werden zurückgegeben:

    ```
    DB20000I  Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.
    ```

    Anmerkung: Wenn die Datenbank zample auf dps bereits vorhanden wäre, würde der Parameter OPTIONS weggelassen, und der Datenbankkonfigurationsparameter *vendoropt* würde verwendet. Dieser Konfigurationsparameter überschreibt den Parameter OPTIONS für eine BACKUP- oder RESTORE-Operation.

12. Führen Sie eine aktualisierende Wiederherstellung durch, um die Transaktionen anzuwenden, die beim Erstellen einer neuen Tabelle und beim Laden neuer Daten in der Protokolldatei für die Datenbank zample erfasst wurden. In diesem Beispiel schlägt die aktualisierende Wiederherstellung (ROLLFORWARD) in der Datenbank fehl, weil das Dienstprogramm für aktualisierende Wiederherstellung die Protokolldateien nicht finden kann, da keine Benutzer- und Computerinformationen angegeben sind:

    ```
    dps:/home/regress9> db2 rollforward db zample to end of logs and stop
    ```

    Der Befehl gibt den folgenden Fehler zurück:

    ```
    SQL4970N  Die aktualisierende Recovery der Datenbank "ZAMPLE"
    kann wegen fehlender Protokolldatei(en) auf Knoten "0" nicht den angegebenen
    Endpunkt (Protokollende oder angegebener Zeitpunkt) erreichen.
    ```

    Veranlassen Sie das Dienstprogramm für aktualisierende Recovery, die Protokolldateien auf einem anderen Computer zu suchen, indem Sie den entsprechenden Wert für *logarchopt* angeben. Verwenden Sie in diesem Beispiel den folgenden Befehl, um den Datenbankkonfigurationsparameter *logarchopt1* festzulegen und nach Protokolldateien für den Benutzer roecken und den Computer bar zu suchen:

    ```
    dps:/home/regress9> db2 update db cfg for zample using logarchopt1
       "'-asnodename=clusternode'"
    ```

13. Ermöglichen Sie mit dem folgenden Befehl dem Dienstprogramm für aktualisierende Wiederherstellung die Verwendung der Backup- und Ladekopieimages, indem Sie den Datenbankkonfigurationsparameter *vendoropt* festlegen:

    ```
    dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
       "'-asnodename=clusternode'"
    ```

14. Mit dem folgenden Befehl können Sie die knotenübergreifende Datenrecovery beenden, indem die in der Protokolldatei der Datenbank zample erfassten Transaktionen angewendet werden:

    ```
    dps:/home/regress9> db2 rollforward db zample to end of logs and stop
    ```

    Die folgenden Informationen werden zurückgegeben:

    ```
                                   Status der aktualisierenden Recovery

     Aliasname der Eingabedatenbank             = zample
     Anzahl der Knoten, die Status zurückgaben  = 1
    ```

```
    Knotennummer                                = 0
    Aktualisierende Recovery: Status            = nicht anstehend
    Nächste zu lesende Protokolldatei           =
    Verarbeitete Protokolldateien               = S0000000.LOG - S0000000.LOG
Letzte festgeschriebene Transaktion             = 2009-02-16-20.10.38.000000 UTC

DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.
```

Die Datenbank zample auf dem Computer dps unter dem Benutzer regress9 wurde auf denselben Stand wie die Datenbank auf dem Computer bar unter dem Benutzer roecken wiederhergestellt.

Synchronisieren der Systemzeit in einer Umgebung mit partitionierten Datenbanken

Unter den Datenbankpartitionsservern sollte für eine relative hohe Synchronisation der Systemuhren gesorgt werden, um einen reibungslosen Datenbankbetrieb und eine uneingeschränkte Möglichkeit zur aktualisierenden Recovery zu gewährleisten. Zeitunterschiede zwischen den Datenbankpartitionsservern, einschließlich aller potenziellen betriebs- und kommunikationsbedingten Verzögerungen für eine Transaktion, sollten kleiner sein als der für den Konfigurationsparameter *max_time_diff* des Datenbankmanagers angegebene Wert, mit dem die maximale Zeitdifferenz zwischen Knoten definiert wird.

Um sicherzustellen, dass die Zeitmarken der Protokollsätze die Reihenfolge von Transaktionen in einer Umgebung mit partitionierten Datenbanken richtig wiedergeben, verwendet DB2 die Systemuhr und die in der Datei SQLOGCTL.LFH gespeicherte virtuelle Zeitmarke der jeweiligen Maschine als Basis für die Zeitmarken in den Protokollsätzen. Wenn die Systemuhr jedoch vorgestellt wird, wird die Protokolluhr automatisch mit vorgestellt. Obwohl die Systemuhr wieder zurückgestellt werden kann, kann die Uhr für die Protokolle dies nicht und bleibt auf dieser *vorgestellten* Zeit, bis die Systemuhr mit dieser Zeit übereinstimmt. Dann sind die Uhren synchron. Dies hat zur Konsequenz, dass ein kurzfristiger Systemuhrfehler auf einem Datenbankknoten einen langfristigen Effekt auf die Zeitmarken von Datenbankprotokollen haben kann.

Nehmen Sie zum Beispiel an, dass die Systemuhr auf Datenbankpartitionsserver A irrtümlicherweise auf den 7. November 2005 gesetzt wird, obwohl das Jahr 2003 ist, und nehmen Sie weiter an, dass der Fehler korrigiert wurde, *nachdem* eine Transaktion zur Aktualisierung in der Datenbankpartition auf diesem Datenbankpartitionsserver festgeschrieben wurde. Wenn die Datenbank ständig verwendet und regelmäßig aktualisiert wird, bleibt jeder Zeitpunkt zwischen dem 7. November 2003 und dem 7. November 2005 durch die aktualisierende Recovery praktisch unerreichbar. Wenn die COMMIT-Operation auf Datenbankpartitionsserver A erfolgt ist, wird die Zeitmarke im Datenbankprotokoll auf 2005 gesetzt und die Uhr des Protokolls bleibt auf dem 7. November 2005 stehen, bis die Systemuhr diesen Zeitpunkt erreicht. Wenn Sie versuchen, eine aktualisierende Recovery bis zu einem Zeitpunkt innerhalb dieses Zeitrahmens durchzuführen, stoppt die Operation an der ersten Zeitmarke, die über den angegebenen Stoppzeitpunkt, d. h. den 7. November 2003, hinausgeht.

Zwar kann DB2 etwaige Aktualisierungen der Systemuhr nicht steuern, der Konfigurationsparameter des Datenbankmanagers *max_time_diff* kann jedoch die Wahrscheinlichkeit reduzieren, dass diesbezüglich Probleme auftreten:

- Die konfigurierbaren Werte für diesen Parameter reichen von 1 Minute bis zu 24 Stunden.

- Wenn die erste Verbindungsanforderung an eine Nichtkatalogpartition erfolgt, sendet der Datenbankpartitionsserver seine Zeit an die Katalogpartition für die Datenbank. Die Katalogpartition überprüft, ob die Zeit in der Datenbankpartition, die die Verbindung anfordert, und ihre eigene Zeit innerhalb des durch den Parameter *max_time_diff* definierten Bereichs liegen. Falls dieser Bereich überschritten wird, wird die Verbindung verweigert.
- Eine Aktualisierungstransaktion, die auf mehr als zwei Datenbankpartitionsserver in der Datenbank zugreift, muss überprüfen, ob die Systemuhren auf den beteiligten Datenbankpartitionsservern synchron sind, bevor die Aktualisierung festgeschrieben werden kann. Wenn zwei oder mehr Datenbankpartitionsserver eine Zeitdifferenz aufweisen, die den durch den Parameter *max_time_diff* gesetzten Grenzwert überschreitet, wird die Transaktion rückgängig gemacht, um zu verhindern, dass die falsche Zeit auf weitere Datenbankpartitionsserver übertragen wird.

Kapitel 19. Fehlerbehebung

Fehlerbehebung in DB2

Bei der Fehlerbehebung geht es im Wesentlichen darum, einen Fehler einzugrenzen und zu bestimmen, um anschließend nach einer Lösung suchen zu können. In diesem Abschnitt wird die Fehlerbehebung für bestimmte DB2-Produkte erläutert.

Wenn allgemeine Fehler bekannt werden sollten, werden sie in diesem Abschnitt in Form von Checklisten hinzugefügt, sobald nähere Informationen verfügbar sind. Sollten Sie den bei Ihnen aufgetretenen Fehler nicht mithilfe der Checkliste beheben können, empfiehlt es sich, weitere Diagnosedaten zu sammeln, die sie dann selbst analysieren oder zur Analyse bei der IBM Unterstützungsfunktion einreichen können.

Mithilfe der folgenden Fragen ermitteln Sie die entsprechenden Fehlerbehebungstasks:

1. Haben Sie alle bekannten Fixpacks angewendet? Falls nicht, sollten Sie die in „Fixpacks anwenden" im Handbuch *DB2-Server - Einstieg* beschriebenen Schritte ausführen.
2. Tritt das Problem in den folgenden Fällen auf?
   - Bei der Installation von DB2-Datenbankservern oder -Clients? Ist dies der Fall, siehe den Abschnitt „Erfassen Sie Daten für Installationsprobleme" an anderer Stelle in diesem Handbuch.
   - Beim Erstellen, Löschen, Aktualisieren oder Migrieren einer Instanz oder des DB2-Verwaltungsservers (DAS)? Ist dies der Fall, siehe den Abschnitt „Erfassen Sie Daten für DAS- und Instanzverwaltungsprobleme" an anderer Stelle in diesem Handbuch..
   - Beim Versetzen von Daten mit den Befehlen EXPORT, IMPORT, LOAD oder `db2move`? Ist dies der Fall, siehe den Abschnitt „Erfassen Sie Daten für Probleme beim Versetzen von Daten" an anderer Stelle in diesem Handbuch.

Passt das Problem zu keiner dieser Kategorien, sollten Sie dennoch allgemeine Diagnosedaten bereithalten, wenn Sie Kontakt mit der IBM Unterstützungsfunktion aufnehmen. Sie sollten .den Abschnitt „Daten für DB2 erfassen" an anderer Stelle in diesem Handbuch lesen.

Fehlerbehebung in Umgebungen mit partitionierten Datenbanken

Absetzen von Befehlen in Umgebungen mit partitionierten Datenbanken

In einer Umgebung mit partitionierten Datenbanken kann es wünschenswert sein, Befehle absetzen zu können, die auf Computern in der Instanz oder auf Datenbankpartitionsservern (Knoten) ausgeführt werden. Sie haben diese Möglichkeit mithilfe des Befehls rah oder des Befehls db2_all. Der Befehl rah ermöglicht Ihnen das Absetzen von Befehlen, die Sie auf Computern in der Instanz ausführen wollen.

Wenn Sie die Befehle auf Datenbankpartitionsservern in der Instanz ausführen wollen, verwenden Sie den Befehl db2_all. Dieser Abschnitt enthält eine Übersicht

über diese Befehle. Die folgenden Informationen beziehen sich ausschließlich auf Umgebungen mit partitionierten Datenbanken.

Unter Windows müssen Sie mit einem Benutzerkonto angemeldet sein, das zur Administratorgruppe gehört, um die Befehle rah oder db2_all ausführen zu können.

Auf Linux- und UNIX-Plattformen kann Ihre Anmeldeshell eine Korn-Shell oder eine beliebige andere Shell sein. Allerdings gibt es Unterschiede in der Art, wie verschiedene Shells Befehle behandeln, die Sonderzeichen enthalten.

Darüber hinaus verwendet der Befehl 'rah' auf Linux- und UNIX-Plattformen das Programm für die ferne Shell, das durch die Registrierdatenbankvariable **DB2RSHCMD** angegeben wird. Sie können zwischen den beiden Programmen für die ferne Shell wählen: 'ssh' (für zusätzliche Sicherheit) oder 'rsh' (bzw. 'remsh' für HP-UX). Wenn **DB2RSHCMD** nicht definiert ist, wird 'rsh' (bzw. 'remsh' für HP-UX) verwendet. Das Programm 'ssh' für die ferne Shell wird verwendet, um die Übertragung von unverschlüsselten Kennwörtern in UNIX-Betriebssystemumgebungen zu verhindern. Eine detaillierte Beschreibung des Verfahrens, wie eine Version von 'ssh' zur Verwendung mit einer DB2-DPF-Instanz zu konfigurieren ist, finden Sie in folgendem Artikel: "Configure DB2 Universal Database for UNIX to use OpenSSH".

Informationen zum Funktionsbereich eines Befehls finden Sie im Handbuch *Command Reference*, das Angaben darüber enthält, ob ein Befehl auf einem einzigen Datenbankpartitionsserver oder auf allen ausgeführt wird. Wenn der Befehl nur auf einem Datenbankpartitionsserver ausgeführt wird und Sie ihn auf allen von ihnen ausführen wollen, verwenden Sie dazu den Befehl db2_all. Eine Ausnahme bildet der Befehl db2trc, der auf allen logischen Knoten (Datenbankpartitionsservern) auf einem Computer ausgeführt wird. Wenn Sie den Befehl db2trc auf allen logischen Knoten aller Computer ausführen möchten, verwenden Sie dazu den Befehl rah.

Teil 4. Leistungsaspekte

Kapitel 20. Leistungsaspekte beim Datenbankentwurf

Leistungsverbessernde Funktionalität

Tabellenpartitionierung und MDC-Tabellen

Bei einer Tabelle kann es sich sowohl um eine MDC-Tabelle (Tabelle mit mehrdimensionalem Clustering) als auch um eine partitionierte Tabelle handeln. In einer Tabelle, die beide Merkmale aufweist, können Spalten sowohl in der Bereichspartitionsspezifikation (Range-Partition-Spec) als auch im MDC-Schlüssel für die Tabellenpartitionierung verwendet werden. Diese Möglichkeit ermöglicht eine exaktere Differenzierung des Datenpartitions- und Blockausschlusses, als dies bei Verwendung nur einer dieser Funktionalitäten der Fall wäre. Es gibt auch zahlreiche Anwendungen, in denen es sinnvoll ist, für den MDC-Schlüssel andere Spalten als die für die Tabellenpartitionierung genutzten Spalten anzugeben. Hierbei ist zu beachten, dass die Tabellenpartitionierung mit mehreren Spalten arbeitet, während beim MDC mit mehreren Dimensionen gearbeitet wird.

Merkmale eines normalen DB2 Data Warehouse der Version 9.1

Die folgenden Empfehlungen beziehen sich auf typische, normale Data Warehouses, die in DB2 Version 9.1 neu implementiert wurden. Die folgenden Merkmale werden angenommen:

- Die Datenbank arbeitet auf mehreren Systemen oder in mehreren logischen AIX-Partitionen.
- Die Datenbankpartitionierungsfunktion (DPF) wird verwendet (Tabellen werden mithilfe der Klausel DISTRIBUTE BY HASH erstellt).
- Es wurden zwischen vier und 50 Datenpartitionen definiert.
- Die Tabelle, für die die Möglichkeit zur Verwendung des MDC und der Tabellenpartitionierung geprüft wird, ist eine der zentralen Fakttabellen.
- Die Tabelle umfasst zwischen 100 Millionen und 100 Milliarden Zeilen.
- Neue Daten werden in unterschiedlichen Zeitrahmen geladen: Jeweils über Nacht, wöchentlich, monatlich.
- Das tägliche Aufnahmevolumen liegt zwischen 10.000 und 10 Millionen Datensätzen.
- Die Datenvolumen schwanken: Hierbei liegt das Volumen des Spitzenmonats um das Fünffache höher als das des Monats mit dem geringsten Datenaufkommen. Entsprechend beläuft sich auch der Umfang der größten Dimensionen (Produktlinie, Bereich) auf das Fünffache des Umfangs der kleinsten Dimensionen.
- Die Datenbank enthält detaillierte Datenbestände der letzten 1 - 5 Jahre.
- Abgelaufene Daten werden in monatlichen oder vierteljährlichen Abständen mit einer Rollout-Operation ausgelagert.
- Tabellen verwenden eine Vielzahl von Abfragetypen. Der Workload besteht jedoch größtenteils aus analytischen Abfragen, die in Bezug auf OLTP-Workloads die folgenden Merkmale aufweisen:
  - Es gibt umfangreichere Ergebnismengen mit bis zu 2 Millionen Zeilen.
  - Die Mehrzahl oder alle Abfragen beziehen sich auf Sichten und nicht auf Basistabellen.

- SQL-Klauseln zur Auswahl von Daten nach Bereichen (Klausel BETWEEN), Elementen in Listen etc.

Merkmale einer Fakttabelle eines normalen DB2 Data Warehouse der Version 9.1

Eine normale Data Warehouse-Fakttabelle kann z. B. das folgende Design verwenden:

- Erstellung von Datenpartitionen über die Spalte 'Month'.
- Definition einer Datenpartition für jeden Rollout-Zeitraum, z. B. 1 Monat, 3 Monate.
- Erstellung von MDC-Dimensionen für 'Day' und für 1 - 4 weitere Dimensionen. Typische Dimensionen sind: Produktlinie und Bereich.
- Alle Datenpartitionen und MDC-Cluster sind über alle Datenpartitionen verteilt.

Das MDC und die Tabellenpartitionierung bieten teilweise dieselben Vorteile. Die folgende Tabelle enthält mögliche Anforderungen innerhalb Ihres Unternehmens und Empfehlungen zu einem Organisationsschema, die auf der Basis der zuvor festgestellten Merkmale gegeben werden.

Tabelle 14. Verwendung der Tabellenpartitionierung mit MDC-Tabellen

Problemstellung	Empfohlenes Schema	Empfehlung
Datenverfügbarkeit während des Rollouts	Tabellenpartitionierung	Sie können die Klausel DETACH PARTITION verwenden, um ein Rollout großer Datenmengen unter minimaler Beeinträchtigung durchzuführen.
Abfrageleistung	Tabellenpartitionierung und MDC	MDC eignet sich am besten für die Abfrage mehrerer Dimensionen. Die Tabellenpartitionierung assistiert durch den Ausschluss von Datenpartitionen.
Minimale Reorganisation	MDC	MDC-Tabellen behalten das Clustering bei, sodass sich die Notwendigkeit von Reorganisationen verringert.
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums während eines traditionellen Offlinezeitfensters	Tabellenpartitionierung	Die Datenpartitionierung kann diese Anforderung voll erfüllen. Durch das MDC würden sich keine zusätzlichen Vorteile ergeben und dieses Verfahren wäre weniger geeignet.
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums während eines Mikro-Offlinezeitfensters (weniger als 1 Minute)	Tabellenpartitionierung	Die Datenpartitionierung kann diese Anforderung voll erfüllen. Durch das MDC würden sich keine zusätzlichen Vorteile ergeben und dieses Verfahren wäre weniger geeignet.

Tabelle 14. Verwendung der Tabellenpartitionierung mit MDC-Tabellen (Forts.)

Problemstellung	Empfohlenes Schema	Empfehlung
Rollout des Datenbestands eines Monats oder eines längeren Zeitraums bei gleichzeitiger Erhaltung der Tabellenverfügbarkeit für Geschäftsbenutzer, die Abfragen ohne Serviceverluste weiterhin übergeben können	MDC	MDC kann nicht alle in diesem Zusammenhang geltenden Anforderungen voll erfüllen. Die Tabellenpartitionierung eignet sich hier nicht, da die Tabelle nur für kurze Zeit in den Offlinemodus versetzt wird.
Tägliches Laden von Daten (entweder ALLOW READ ACCESS oder ALLOW NO ACCESS)	Tabellenpartitionierung und MDC	MDC bietet hier die meisten Vorteile. Die Tabellenpartitionierung bietet Vorteile in Teilbereichen.
Kontinuierliches Laden von Daten (ALLOW READ ACCESS)	Tabellenpartitionierung und MDC	MDC bietet hier die meisten Vorteile. Die Tabellenpartitionierung bietet Vorteile in Teilbereichen.
Leistung bei der Ausführung von Abfragen für traditionelle BI-Abfragen	Tabellenpartitionierung und MDC	MDC eignet sich besonders gut für Abfragen in Datenkuben und mehreren Dimensionen. Die Tabellenpartitionierung bietet Unterstützung über den Partitionsausschluss.
Minimierung des Reorganisationsaufwandes durch Vermeidung des Reorganisationsbedarfs oder Reduzierung des Aufwands für die Ausführung der Task	MDC	Beim MDC wird das Clustering beibehalten, sodass sich die Notwendigkeit von Reorganisationen verringert. Wenn das MDC verwendet wird, dann bietet die Datenpartitionierung auch in Teilbereichen keine Vorteile. Allerdings ermöglicht die Bereichspartitionierung bei Nichtverwendung des MDC die Reduzierung des Reorganisationsbedarfs, indem auf Partitionsebene ein etwas groberes Clustering beibehalten wird.

Beispiel 1:

Sie arbeiten mit einer Tabelle mit den Schlüsselspalten 'YearAndMonth' und 'Province'. Bei der Planung dieser Tabelle wäre eine Partitionierung nach Datum mit einer Zeitspanne von zwei Monaten pro Datenpartition sinnvoll. Darüber hinaus können Sie die Daten auch nach der Spalte 'Province' organisieren, sodass alle Zeilen für ein bestimmtes Gebiet innerhalb eines Datumsbereichs von zwei Monaten in einer Gruppe zusammengefasst werden. Diese Vorgehensweise ist in Abb. 6 auf Seite 37 dargestellt.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (Province);
```

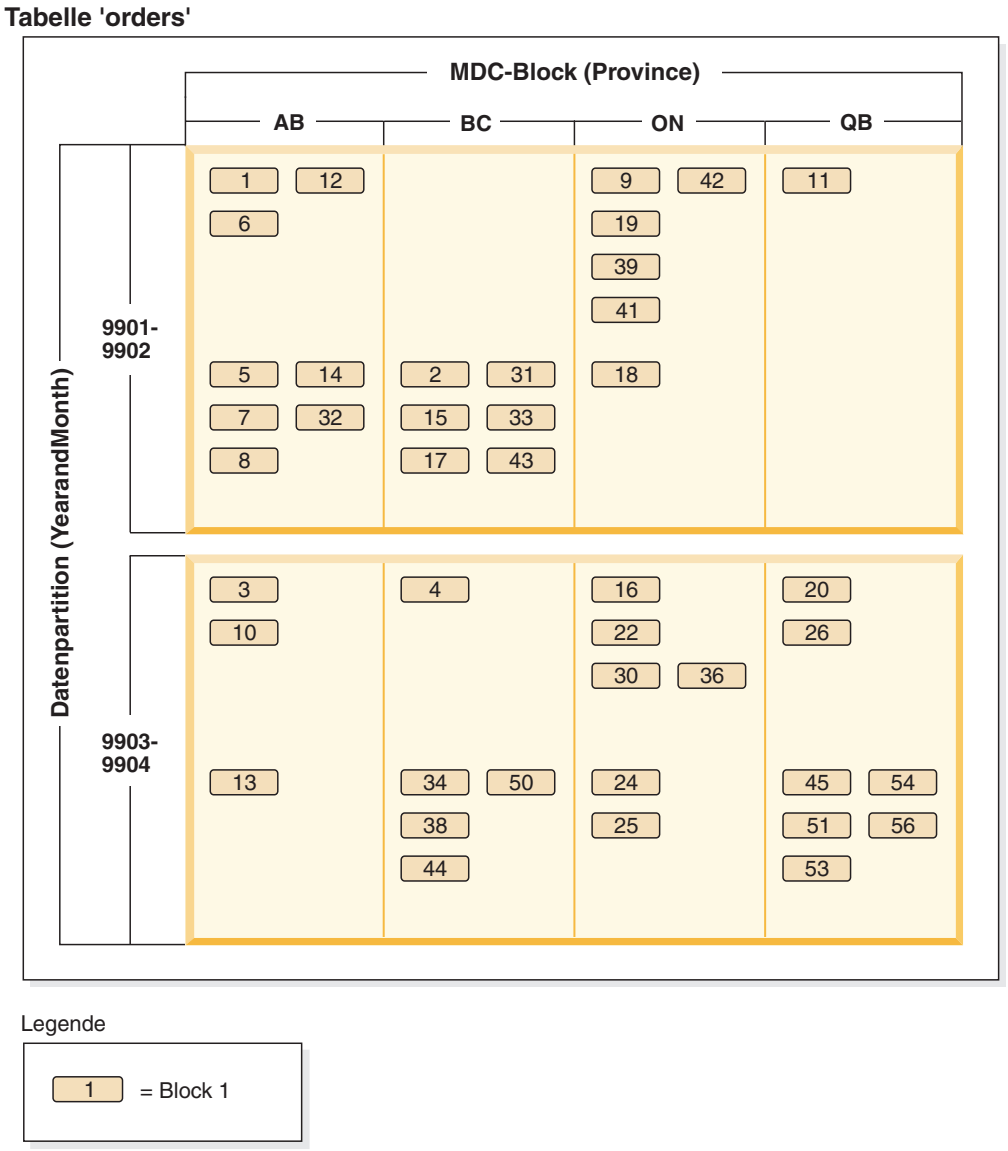


Abbildung 46. Eine nach 'YearAndMonth' partitionierte und nach 'Province' organisierte Tabelle

Beispiel 2:

Eine exaktere Differenzierung kann durch Hinzufügen von 'YearAndMonth' zur Klausel ORGANIZE BY (siehe hierzu Abb. 7 auf Seite 38) erzielt werden.

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (YearAndMonth, Province);
```

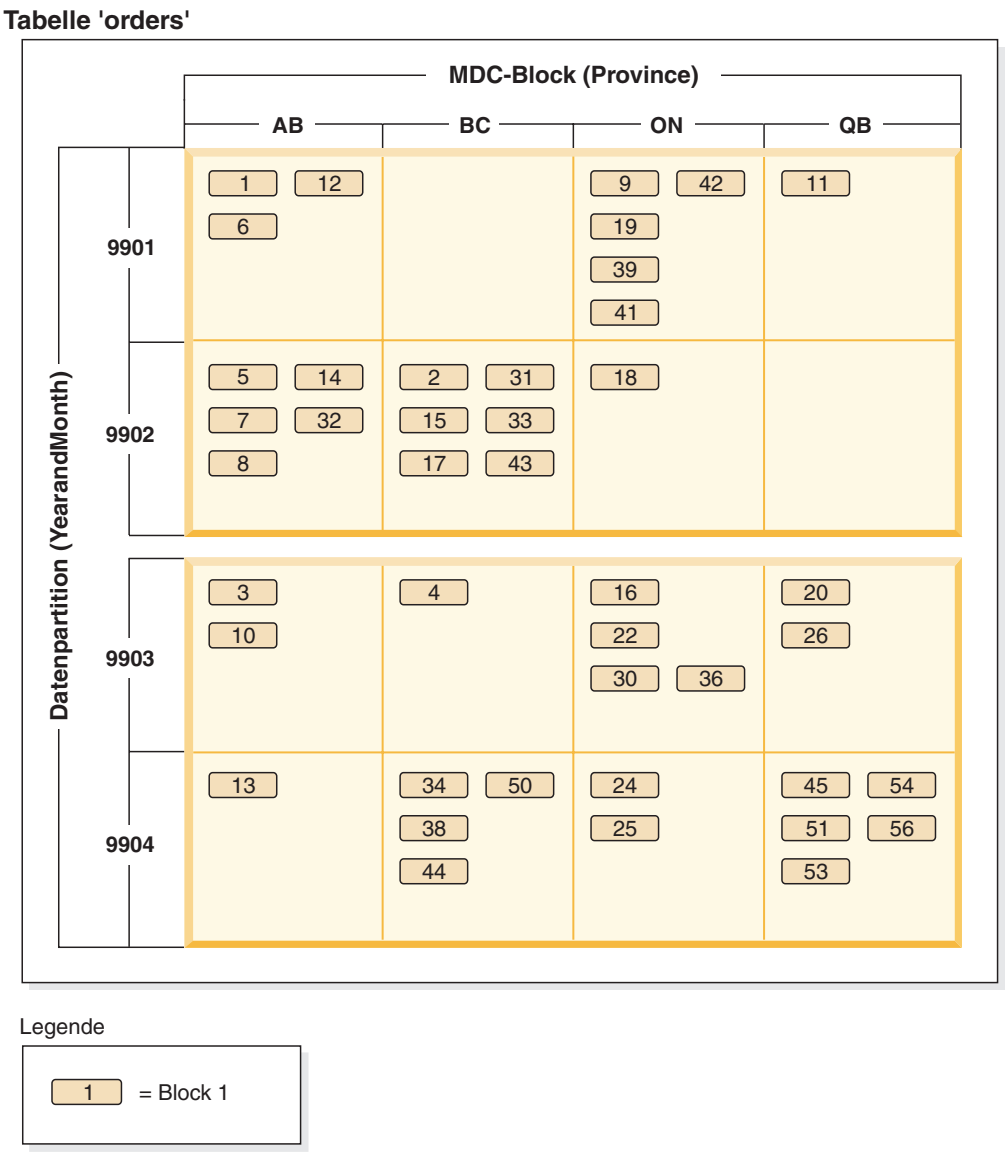


Abbildung 47. Eine nach 'YearAndMonth' partitionierte und nach 'Province' und 'YearAndMonth' organisierte Tabelle

Wenn die Partitionierung so festgelegt wurde, dass jeder Bereich nur einen einzigen Wert enthält, ergeben sich keine Vorteile, wenn die Tabellenpartitionierungsspalte in den MDC-Schlüssel aufgenommen wird.

Wichtige Hinweise

- Im Vergleich mit einer Basistabelle benötigen sowohl MDC-Tabellen als auch partitionierte Tabellen mehr Speicherplatz. Diese Speicherplatzanforderungen gelten zusätzlich zu den sonstigen Anforderungen, sind jedoch unter Berücksichtigung der sich daraus ergebenden Vorteile sinnvoll.
- Wenn Sie die Tabellenpartitionierung und die MDC-Funktionalität in Ihrer partitionierten Datenbankumgebung nicht zusammen einsetzen wollen, dann sollten Sie die Tabellenpartitionierung in den Fällen einsetzen, in denen die Datenverteilung verlässlich vorausgesagt werden kann. Dies ist normalerweise bei den hier erläuterten Systemtypen der Fall. Andernfalls sollten Sie die Verwendung von MDC in Betracht ziehen.

Optimierungsstrategien für partitionierte Tabellen

Die Bezeichnung *Ausschluss von Datenpartitionen* bezieht sich auf die Fähigkeit des Datenbankservers, auf der Grundlage der Abfragevergleichselemente festzustellen, dass nur auf eine Untergruppe der Datenpartitionen einer Tabelle zugegriffen werden muss, um eine Abfrage zu erfüllen. Der Ausschluss von Datenpartitionen ist insbesondere vorteilhaft, wenn eine Entscheidungshilfeabfrage auf eine partitionierte Tabelle ausgeführt wird.

Eine partitionierte Tabelle arbeitet mit einem Datenorganisationsschema, bei dem Tabellendaten auf mehrere Speicherobjekte, die als Datenpartitionen oder Datenbereiche (RANGE) bezeichnet werden, entsprechend den Werten einer oder mehrerer Spalten der Tabelle, die den Tabellenpartitionierungsschlüssel bilden, verteilt werden. Daten aus einer gegebenen Tabelle werden in mehrere Speicherobjekte auf der Basis der in der Klausel PARTITION BY der Anweisung CREATE TABLE angegebenen Spezifikationen partitioniert. Diese Speicherobjekte können sich in verschiedenen Tabellenbereichen, in denselben Tabellenbereichen oder in einer Kombination beider Arten von Tabellenbereichen befinden.

Das folgende Beispiel veranschaulicht die Leistungsvorteile des Ausschlusses von Datenpartitionen. Dazu wird die folgende Anweisung zur Erstellung einer partitionierten Tabelle ausführt:

```
CREATE TABLE custlist(subsdate DATE, Province CHAR(2), AccountID INT)
PARTITION BY RANGE(subsdate)
(STARTING FROM '1/1/1990' IN ts1,
STARTING FROM '1/1/1991' IN ts1,
STARTING FROM '1/1/1992' IN ts1,
STARTING FROM '1/1/1993' IN ts2,
STARTING FROM '1/1/1994' IN ts2,
STARTING FROM '1/1/1995' IN ts2,
STARTING FROM '1/1/1996' IN ts3,
STARTING FROM '1/1/1997' IN ts3,
STARTING FROM '1/1/1998' IN ts3,
STARTING FROM '1/1/1999' IN ts4,
STARTING FROM '1/1/2000' IN ts4,
STARTING FROM '1/1/2001' ENDING '12/31/2001' IN ts4);
```

Nehmen Sie an, Sie interessieren sich für Kundeninformationen des Jahres 2000. Sie führen die folgende Abfrage aus:

```
SELECT * FROM custlist WHERE subsdate BETWEEN '1/1/2000' AND '12/31/2000';
```

Wie in Abb. 48 auf Seite 315 gezeigt, stellt der Datenbankserver fest, dass nur auf eine Datenpartition in Tabellenbereich 4 ('ts4') zugegriffen werden muss, um diese Abfrage zu erfüllen.

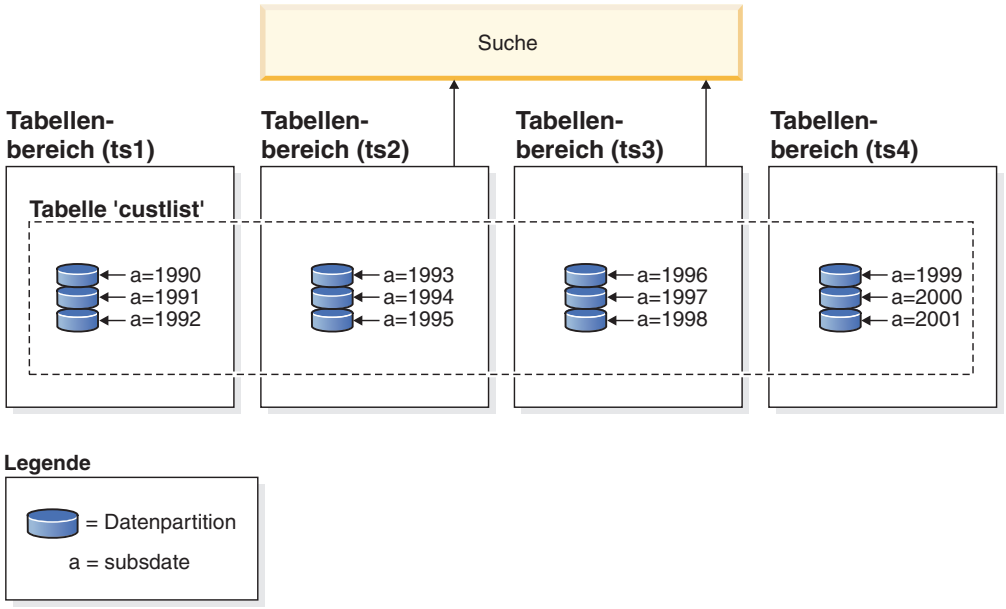


Abbildung 48. Die Leistungsvorteile des Ausschlusses von Datenpartitionen für eine partitionierte Tabelle

Ein weiteres Beispiel für den Ausschluss von Datenpartitionen (siehe Abb. 49 auf Seite 316) ist eine Indexsuche über zwei Indizes, die auf folgendem Schema basieren:

```
CREATE TABLE multi (sale_date date, region char(2))
PARTITION BY (sale_date)
(STARTING '01/01/2005' ENDING '12/31/2005' EVERY 1 MONTH);
CREATE INDEX sx ON multi(sale_date);
CREATE INDEX rx ON multi(region);
```

Sie führen die folgende Abfrage aus:

```
SELECT * FROM multi WHERE
sale_date BETWEEN '6/1/2005' AND '7/31/2005' AND REGION = 'NW';
```

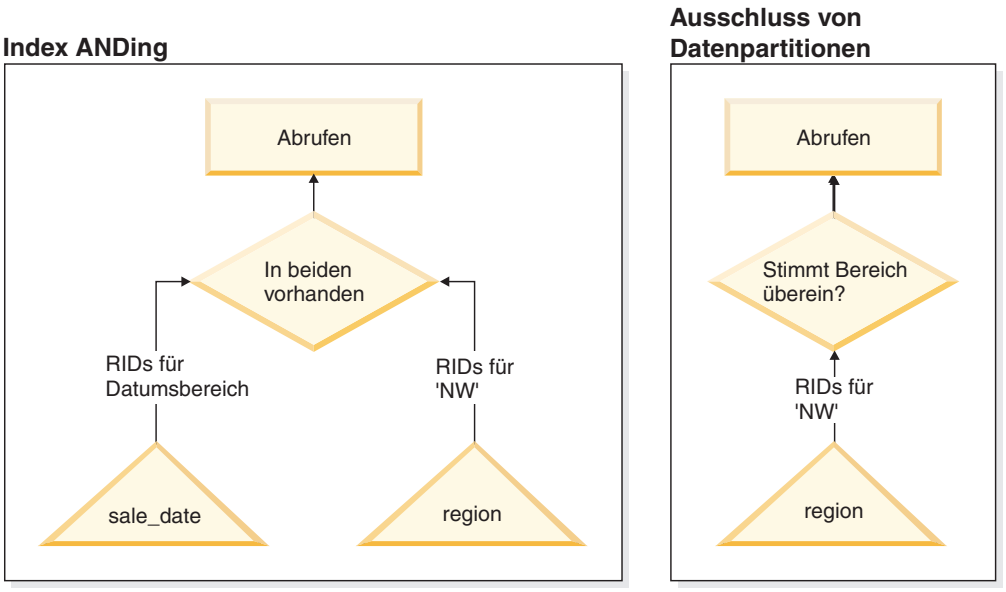


Abbildung 49. Der Entscheidungspfad des Optimierungsprogramms für die Tabellenpartitionierung und das logische Verknüpfen von Indizes über AND (Index ANDing)

Ohne Tabellenpartitionierung besteht ein wahrscheinlicher Plan in der logischen Verknüpfung von der Indizes über AND. Das Index ANDing führt die folgenden Aktionen aus:

- Lesen aller relevanten Indexeinträge aus jedem Index
- Speichern beider Gruppen von Zeilenkennungen (Satz-IDs, RIDs)
- Abgleichen der RIDs, um zu ermitteln, welche in beiden Indizes vorkommen
- Verwenden der RIDs zum Abrufen der Zeilen

Wie in Abb. 49 zu erkennen ist, wird mit der Tabellenpartitionierung der Index gelesen, um Übereinstimmungen für beide Spalten, d. h. 'region' und 'sale_date', zu finden, sodass entsprechende Zeilen schnell abgerufen werden können.

DB2 Explain

Sie können auch mithilfe von DB2 Explain den Ausschluss von Datenpartitionen ermitteln, der vom DB2-Optimierungsprogramm ausgewählt wurde. Die **DP Elim Predicates**-Informationen zeigen, welche Datenpartitionen durchsucht werden, um die folgende Abfrage zu erfüllen:

```
SELECT * FROM custlist WHERE subsdate
BETWEEN '12/31/1999' AND '1/1/2001'

Arguments:
---------
DPESTFLG: (Number of data partitions accessed are Estimated)
    FALSE
DPLSTPRT: (List of data partitions accessed)
    9-11
DPNUMPRT: (Number of data partitions accessed)
   3

DP Elim Predicates:
------------------
Range 1)
  Stop  Predicate: (Q1.A <= '01/01/2001')
 Start Predicate: ('12/31/1999' <= Q1.A)
```

```
Objects Used in Access Plan:
 --------------------------

  Schema: MRSRINI
  Name:    CUSTLIST
  Type:    Data Partitioned Table
   Time of creation:      2005-11-30-14.21.33.857039
   Last statistics update:    2005-11-30-14.21.34.339392
    Number of columns:      3
    Number of rows:      100000
   Width of rows:      19
   Number of buffer pool pages:    1200
    Number of data partitions:    12
    Distinct row values:      No
    Tablespace name:      <VARIOUS>
```

Unterstützung mehrerer Spalten

Der Ausschluss von Datenpartitionen funktioniert auch in Fällen, in denen mehrere Spalten als Tabellenpartitionierungsschlüssel verwendet werden.

Es wird zum Beispiel eine Tabelle mit der folgenden Anweisung erstellt:

```
CREATE TABLE sales(year INT, month INT)
 PARTITION BY RANGE(year, month)
 (STARTING FROM (2001, 1) ENDING AT(2001,3) IN ts1,
 ENDING AT(2001,6) IN ts2,
 ENDING AT(2001,9) IN ts3,
 ENDING AT(2001,12) IN ts4,
 ENDING AT(2002,3) IN ts5,
 ENDING AT(2002,6) IN ts6,
 ENDING AT(2002,9) IN ts7,
 ENDING AT(2002,12) IN ts8)
```

Anschließend wird die folgende Abfrage ausgeführt:

```
 SELECT * FROM sales WHERE year = 2001 AND month < 8
```

Das Abfrageoptimierungsprogramm folgert, dass zur Erfüllung dieser Abfrage nur auf die Datenpartitionen in 'ts1', 'ts2' und 'ts3' zugegriffen werden muss.

Anmerkung: Wenn der Tabellenpartitionierungsschlüssel aus mehreren Spalten gebildet wird, ist der Ausschluss von Datenpartitionen nur möglich, wenn Vergleichselemente für die führenden Spalten des zusammengesetzten Schlüssels verwendet werden, da nicht führende Spalten, die im Tabellenpartitionierungsschlüssel verwendet werden, nicht unabhängig sind.

Unterstützung mehrerer Bereiche

Es ist möglich, einen Ausschluss von Datenpartitionen für mehrere Bereiche zu erzielen (d. h. durch logisches Verknüpfen von Bereichen über OR). An der im vorigen Beispiel erstellten Tabelle wird zum Beispiel die folgende Abfrage ausgeführt:

```
 SELECT * FROM sales
 WHERE (year = 2001 AND month <= 3) OR (year = 2002 and month >= 10)
```

Der Datenbankserver greift nur auf Daten für das erste Quartal von 2001 und das letzte Quartal von 2002 zu.

Generierte Spalten

Sie können generierte Spalten als Tabellenpartitionierungsschlüssel verwenden.

Sie können zum Beispiel die folgende Anweisung ausführen:

```
CREATE TABLE sales(a INT, b INT GENERATED ALWAYS AS (a / 5))
 IN ts1,ts2,ts3,ts4,ts5,ts6,ts7,ts8,ts9,ts10
 PARTITION BY RANGE(b)
 (STARTING FROM (0) ENDING AT(1000) EVERY (50))
```

In diesem Fall werden Vergleichselemente für die generierte Spalte zum Ausschluss von Datenpartitionen verwendet. Wenn der Ausdruck zur Generierung der Spalten außerdem monoton ist, übersetzt der Datenbankserver Vergleichselemente für die Quellenspalten in Vergleichselemente für die generierten Spalten, sodass der Ausschluss von Datenpartitionen über die generierten Spalten erfolgen kann.

Betrachten Sie zum Beispiel die folgende Abfrage:

```
 SELECT * FROM sales WHERE a > 35
```

In diesem Fall generiert der Datenbankserver aus dem Vergleichselement für a (a > 35) ein zusätzliches Vergleichselement für b (b > 7), um den Ausschluss von Datenpartitionen zu ermöglichen.

Joinvergleichselemente

Joinvergleichselemente können ebenfalls beim Ausschluss von Datenpartitionen verwendet werden, wenn das Joinvergleichselement auf die Ebene des Tabellenzugriffs verschoben wird (Pushdown). Das Joinvergleichselement wird nur für die innere Tabelle eines Joins mit Verschachtelungsschleife (NLJN, Nested Loop Join) auf die Tabellenzugriffsebene verschoben.

Betrachten Sie zum Beispiel die folgenden Tabellen:

```
CREATE TABLE T1(A INT, B INT)
 PARTITION BY RANGE(A, B)
 (STARTING FROM (1, 1)
 ENDING (1,10) IN ts1, ENDING (1,20) IN ts2,
 ENDING (2,10) IN ts3, ENDING (2,20) IN ts4,
 ENDING (3,10) IN ts5, ENDING (3,20) IN ts6,
 ENDING (4,10) IN ts7, ENDING (4,20) IN ts8)

CREATE TABLE T2 (A INT, B INT)
```

Verwendete Vergleichselemente:

```
P1: T1.A = T2.A
P2: T1.B > 15
```

In diesem Beispiel lassen sich die genauen Datenpartitionen, auf die zugegriffen wird, wegen unbekannter Werte der äußeren Tabelle des Joins beim Kompilieren nicht bestimmen. In diesem Fall und ebenso in Fällen, in denen Hostvariablen oder Parametermarken verwendet werden, erfolgt der Ausschluss von Datenpartitionen bei der Ausführung, wenn die erforderlichen Werte gebunden werden.

Bei der Ausführung erfolgt, wenn T1 die innere Tabelle eines Joins mit Verschachtelungsschleife (NLJN) ist, der Ausschluss von Datenpartitionen auf der Basis der Vergleichselemente für jeden äußeren Wert von T2.A dynamisch. Bei der Ausführung werden die Vergleichselemente T1.A = 3 und T1.B > 15 für den Wert T2.A = 3

der äußeren Tabelle angewendet. Dadurch werden die Datenpartitionen in den Tabellenbereichen 'ts6' und 'ts7' für den Zugriff ermittelt.

Nehmen Sie an, dass die Spalten A in den Tabellen T1 und T2 folgende Werte enthalten:

Äußere Tabelle T2: Spalte A	Innere Tabelle T1: Spalte A	Innere Tabelle T1: Spalte B	Innere Tabelle T1: Position der Datenpartition
2	3	20	ts6
3	2	10	ts3
3	2	18	ts4
	3	15	ts6
	1	40	ts3

Für den Join mit Verschachtelungsschleife (unter Annahme einer Tabellensuche für die innere Tabelle) führt der Datenbankmanager die folgenden Schritte aus:

1. Er liest die erste Zeile aus T2. Der Wert für A ist 2.
2. Er bindet den Wert T2.A (d. h. 2) an die Spalte T2.A im Joinvergleichselement T1.A = T2.A. Aus dem Vergleichselement wird T1.A = 2.
3. Er wendet den Ausschluss von Datenpartitionen unter Verwendung der Vergleichselemente T1.A = 2 und T1.B > 15 an. Dies qualifiziert die Datenpartitionen in den Tabellenbereichen 'ts4' und 'ts5'.
4. Er durchsucht die Datenpartitionen in den Tabellenbereichen 'ts4' und 'ts5' von Tabelle T1, bis eine Zeile nach Anwendung von T1.A = 2 und T1.B > 15 gefunden wird. Die erste qualifizierte Zeile, die gefunden wird, ist die dritte Zeile von T1.
5. Er verknüpft die übereinstimmenden Zeilen.
6. Er durchsucht die Datenpartitionen in den Tabellenbereichen 'ts4' und 'ts5', bis die nächste Übereinstimmung (mit T1.A = 2 und T1.B > 15) gefunden wird. In diesem Fall werden keine weiteren Zeilen gefunden.
7. Er wiederholt die Schritte 1 bis 6 für die nächste Zeile von T2 (d. h. er nimmt den Wert 3 aus Spalte A). Dieses Verfahren wird fortgesetzt, bis alle Zeilen von T2 durchlaufen sind.

Optimierungsstrategien für MDC-Tabellen

Wenn Sie Tabellen mit mehrdimensionalem Clustering (MDC - Multidimensional Clustering) erstellen, kann sich die Leistung vieler Abfragen verbessern, weil das Optimierungsprogramm zusätzliche Optimierungsstrategien anwenden kann. Diese Strategien basieren in erster Linie auf der verbesserten Effizienz von Blockindizes, jedoch ermöglicht der Vorteil des mehrdimensionalen Clustering auch einen schnelleren Datenabruf.

Anmerkung: Die Optimierungsstrategien für MDC-Tabellen können auch die Leistungsvorteile der partitionsinternen und partitionsübergreifenden Parallelität implementieren.

MDC-Tabellen bieten die folgenden spezifischen Vorteile:

- Dimensionsblockindexsuchen können die erforderlichen Teile der Tabelle ermitteln und schnell nur die angeforderten Blöcke durchsuchen.
- Da Blockindizes kleiner als Satz-ID-Indizes sind, arbeiten Blockindexsuchen schneller.

- Logische Verknüpfungen von Indizes über AND und OR (Index ANDing und Index ORing) können auf Blockebene durchgeführt und mit Satz-IDs kombiniert werden.
- Daten werden garantiert in EXTENTSIZE großen Speicherbereichen in Clustern gruppiert, was ein schnelleres Abrufen ermöglicht.
- Zeilen können schneller gelöscht werden, wenn ein Rollout (Datenauslagerung) ausgeführt werden kann.

Betrachten Sie das folgende einfache Beispiel für eine MDC-Tabelle mit dem Namen SALES, in der Dimensionen auf den Spalten **region** und **month** definiert sind:

```
SELECT * FROM SALES
    WHERE MONTH='March' AND REGION='SE'
```

Für diese Abfrage kann das Optimierungsprogramm eine Dimensionsblockindexsuche durchführen, um die Blöcke zu finden, in denen der Monat März (March) und die Region SE vorkommen. Anschließend kann es nur die resultierenden Blöcke der Tabelle durchsuchen, um die Ergebnismenge abzurufen.

Rolloutlöschung

Wenn die Bedingungen für das Löschen durch ein Rollout gegeben sind, wird ein effizienteres Verfahren zum Löschen von Zeilen aus MDC-Tabellen verwendet. Die folgenden Bedingungen müssen erfüllt sein:

- Die Anweisung DELETE wird gesucht, nicht positioniert (d. h. verwendet nicht die Klausel „WHERE CURRENT OF").
- Es gibt keine WHERE-Klauseln (alle Zeilen müssen gelöscht werden) oder die einzigen Bedingungen in der Klausel WHERE beziehen sich auf Dimensionen.
- Die Tabelle wurde nicht mit der Klausel DATA CAPTURE CHANGES definiert.
- Die Tabelle ist nicht die übergeordnete Tabelle in einer referenziellen Integritätsbeziehung.
- Für die Tabelle dürfen keine Trigger beim Löschen (ON DELETE) definiert sein.
- Die Tabelle wird in keinen MQTs (Materialized Query Tables) verwendet, die sofort aktualisiert werden.
- Eine kaskadierende Löschoperation kommt für ein Rollout in Frage, wenn der Fremdschlüssel eine Untermenge der Dimensionsspalten der Tabelle ist.
- Die Anweisung DELETE kann nicht in einer Anweisung SELECT enthalten sein, die an der temporären Tabelle ausgeführt wird, die die Menge der betroffenen Zeilen vor einer auslösenden SQL-Operation (durch die Klausel OLD TABLE AS in der Anweisung CREATE TRIGGER angegeben) angibt.

Bei einer Rolloutlöschung werden die gelöschten Datensätze nicht protokolliert. Stattdessen werden die Seiten, die die Datensätze enthalten, durch eine Neuformatierung von Teilen der Seiten optisch geleert. Die Änderungen an den neu formatierten Teilen werden protokolliert, die Datensätze selbst werden jedoch nicht protokolliert.

Das Standardverhalten *Rollout mit sofortiger Bereinigung* sieht vor, dass Satz-IDs (RIDs) beim Löschen bereinigt werden. Dieser Modus kann auch durch Setzen der Registrierdatenbankvariablen **DB2_MDC_ROLLOUT** auf den Wert IMMEDIATE oder durch Angeben von `IMMEDIATE` in der Anweisung SET CURRENT MDC ROLLOUT MODE angefordert werden. Es gibt keine Änderung bei der Protokollierung von Indexaktualisierungen im Vergleich zu einer normalen Löschoperation. Die Leistungsverbesserung hängt also davon ab, wie viele Satz-ID-Indizes vorhan-

den sind. Je weniger Satz-ID-Indizes vorhanden sind, desto stärker ist die Verbesserung (als Prozentanteil der Gesamtzeit und des gesamten Protokollspeichers).

Ein Schätzwert für den eingesparten Platz im Protokoll kann anhand der folgenden Formel ermittelt werden. Dabei ist `N` die Anzahl der gelöschten Datensätze, `S` die Gesamtgröße der gelöschten Datensätze einschließlich Systemaufwand (z. B. Nullanzeiger und VARCHAR-Längen) und `P` die Anzahl der Seiten in den Blöcken, die die gelöschten Datensätze enthalten:

```
S + 38*N - 50*P
```

Dieser Wert stellt die Verkleinerung in den tatsächlichen Protokolldaten dar. Die Einsparung beim Platzbedarf für das aktive Protokoll ist aufgrund der Einsparung des Speicherplatzes, der für Rollbackzwecke reserviert wird, doppelt so hoch.

Alternativ können Sie die Satz-ID-Indizes auch aktualisieren lassen, nachdem die Transaktion festgeschrieben wurde, indem Sie ein *Rollout mit verzögerter Bereinigung* verwenden. Dieser Modus kann auch durch Setzen der Registrierdatenbankvariablen **DB2_MDC_ROLL_OUT** auf den Wert DEFER oder durch Angeben von `DEFERRED` in der Anweisung SET CURRENT MDC ROLLOUT MODE angefordert werden. Bei einem Rollout mit verzögerter Bereinigung werden Satz-ID-Indizes asynchron im Hintergrund bereinigt, nachdem die Löschoperation festgeschrieben wurde. Diese Rolloutmethode kann zu erheblich schnelleren Löschzeiten bei sehr umfangreichen Löschungen oder Tabellen mit mehreren Satz-ID-Indizes führen. Die Geschwindigkeit der allgemeinen Bereinigungsoperation erhöht sich, weil die Indizes bei einer verzögerten Indexbereinigung parallel bereinigt werden, während bei einer sofortigen Indexbereinigung jede Zeile des Index einzeln bereinigt wird. Darüber hinaus verringert sich der Platzbedarf des Transaktionsprotokolls für die Anweisung DELETE beträchtlich, weil die asynchrone Indexbereinigung die Indexaktualisierungen pro Indexseite und nicht pro Indexschlüssel protokolliert.

Anmerkung: Ein Rollout mit verzögerter Bereinigung erfordert zusätzliche Speicherressourcen, die aus dem Datenbankzwischenspeicher zugeordnet werden. Wenn DB2 die erforderlichen Speicherstrukturen nicht zuordnen kann, schlägt der Rollout mit verzögerter Bereinigung fehl und eine Nachricht wird in das Administratorprotokoll geschrieben.

Empfehlungen für die Verwendung von Rollouts mit verzögerter Bereinigung

Wenn für Sie die Löschleistung der wichtigste Faktor ist und Satz-ID-Indizes für die Tabelle definiert sind, sollte ein Rollout mit verzögerter Bereinigung verwendet werden. Beachten Sie, dass vor der Indexbereinigung indexbasierte Suchoperationen in den durch Rollout gelöschten Blöcken je nach Umfang der gelöschten Daten eine leichte Leistungseinbuße erfahren. Darüber hinaus sollten auch folgenden Aspekte bei der Entscheidung zwischen sofortiger und verzögerter Indexbereinigung berücksichtigt werden:

- Der Umfang der Löschoperation: Wählen Sie ein Rollout mit verzögerter Bereinigung für sehr umfangreiche Löschoperationen aus. In Fällen, in denen Dimensionslöschanweisungen für viele kleine MDC-Tabellen ausgeführt werden, kann der Aufwand für die asynchrone Bereinigung von Indexobjekten den Vorteil der Zeiteinsparung während der Löschoperationen übersteigen.
- Die Anzahl und der Typ von Indizes: Wenn die Tabelle eine Reihe von Satz-ID-Indizes hat, die eine Verarbeitung auf Zeilenebene erfordern, sollten ein Rollout mit verzögerter Bereinigung verwendet werden.

- Die Blockverfügbarkeit: Wenn Sie wünschen, dass der Blockspeicherplatz durch die Anweisung DELETE freigegeben wird, sodass er sofort nach dem Festschreiben der Löschoperation verfügbar ist, verwenden Sie einen Rollout mit sofortiger Bereinigung.
- Der Protokollspeicherbereich: Wenn der Protokollspeicherbereich begrenzt ist, sollte bei umfangreichen Löschoperationen ein Rollout mit verzögerter Bereinigung verwendet werden.
- Speicherbeschränkungen: Ein Rollout mit verzögerter Bereinigung benötigt zusätzlichen Speicherplatz im Datenbankzwischenspeicher für alle Tabellen, für die eine verzögerte Bereinigung ansteht.

Wenn Sie die Rolloutfunktionalität für Löschoperationen inaktivieren wollen, können Sie die Registrierdatenbankvariable **DB2_MDC_ROLLOUT** auf den Wert OFF setzen oder in der Anweisung SET CURRENT MDC ROLLOUT MODE den Wert `NONE` angeben.

Kapitel 21. Indizes

Indizes in partitionierten Tabellen

Indexverhalten bei partitionierten Tabellen

Indizes für partitionierte Tabellen sind Indizes für normale Tabellen insofern ähnlich, als dass jeder Index Zeiger auf Zeilen in allen Datenpartitionen der Tabelle enthält. Ein wichtiger Unterschied besteht jedoch darin, dass jeder Index für eine partitionierte Tabelle ein unabhängiges Objekt ist. In Umgebungen mit partitionierten Datenbanken wird der Index über die Datenbankpartitionen in der gleichen Weise verteilt wie die Tabelle. Da ein Index für eine partitionierte Tabelle unabhängig von anderen Indizes einsetzbar ist, sind einige besondere Aspekte im Hinblick darauf zu beachten, welcher Tabellenbereich beim Erstellen eines Indexes für eine partitionierte Tabelle verwendet wird.

Ein Index für eine partitionierte Tabelle wird in einem einzigen Tabellenbereich erstellt, auch wenn sich die Datenpartitionen der Tabelle über mehrere Tabellenbereiche erstrecken. Sowohl DMS- als auch SMS-Tabellenbereiche unterstützen die Verwendung von Indizes, die sich an einer anderen Position befinden als die Tabelle. Alle angegebenen Tabellenbereiche müssen sich in derselben Datenbankpartitionsgruppe befinden. Jeder Index kann in einem eigenen Tabellenbereich, auch in großen Tabellenbereichen, abgelegt werden. Jeder Tabellenbereich für einen Index muss denselben Speichermechanismus wie die Datenpartitionen verwenden (entweder DMS oder SMS). Indizes in großen Tabellenbereichen können bis zu 2^29 Seiten enthalten.

Ein Index für eine partitionierte Tabelle bietet folgende zusätzliche Vorteile:

- Beim Löschen und bei der Online-Erstellung von Indizes wird eine bessere Leistung erzielt.
- Es besteht die Möglichkeit, verschiedene Werte für die Tabellenbereichsmerkmale für verschiedene Indizes der Tabelle zu verwenden (z. B. können verschiedene Seitengrößen für die einzelnen Indizes zu einer besseren Speicherplatznutzung beitragen).
- Weniger E/A-Konkurrenzsituationen lassen einen effizienteren gemeinsamen Zugriff auf die Indexdaten für die Tabelle zu.
- Wenn einzelne Indizes gelöscht werden, wird der Speicherplatz sofort für das System verfügbar, ohne dass eine Indexreorganisation erforderlich ist.
- Wenn eine Indexreorganisation ausgeführt werden soll, kann ein einzelner Index reorganisiert werden.

Abb. 50 auf Seite 324 zeigt einen nicht partitionierten Index für eine partitionierte Tabelle, die sich in einem einzigen Tabellenbereich befindet.

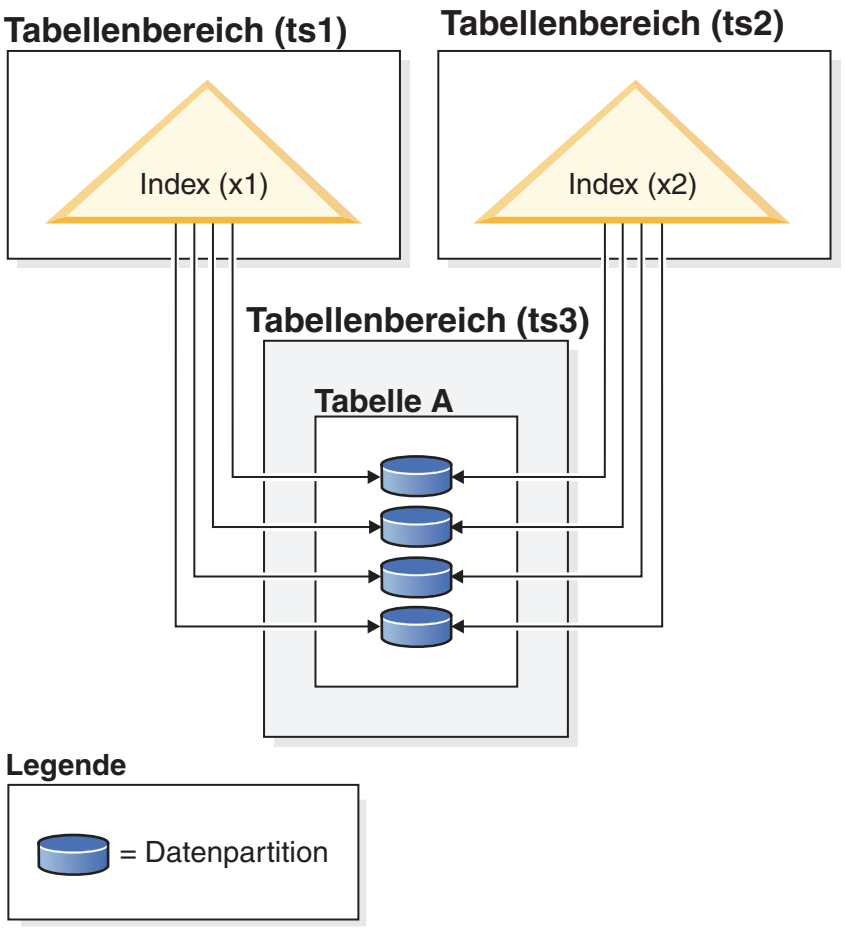


Abbildung 50. Indexverhalten bei einer partitionierten Tabelle

Abb. 51 auf Seite 325 zeigt das Indexverhalten bei einer partitionierten Tabelle, die gleichzeitig über mehrere Datenbankpartitionen verteilt ist.

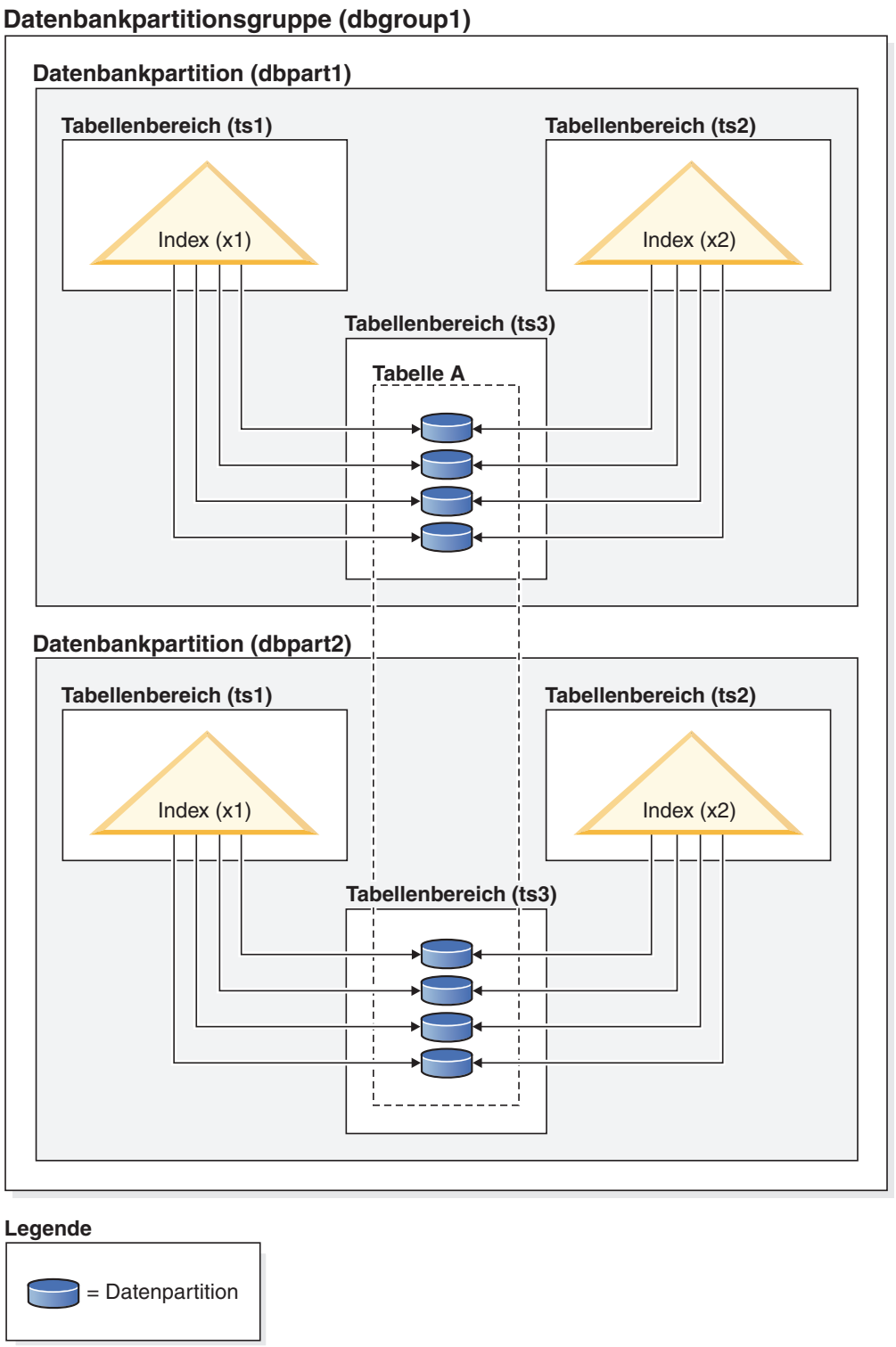


Abbildung 51. Indexverhalten bei einer Tabelle, die sowohl verteilt als auch partitioniert ist.

Sie können einen Tabellenbereich für Indizes für eine partitionierte Tabelle in der Anweisung CREATE INDEX ...IN <tb_bereich1> angeben, der ein anderer ist, als der Tabellenbereich für Indizes, der in der Anweisung CREATE TABLE ... INDEX IN <tb_bereich2> angegeben wurde.

Nur für partitionierte Tabellen könne Sie die Indexspeicherposition durch die IN-Klausel der Anweisung CREATE INDEX überschreiben, sodass Sie eine Tabellenbereichsposition für den Index angegeben können. Durch diese Methode können Sie verschiedene Indizes für eine partitionierte Tabelle je nach Bedarf in einem anderen Tabellenbereich speichern. Wenn Sie eine partitionierte Tabelle erstellen, ohne die Speicherposition für die zugehörigen nicht partitionierten Indizes anzugeben, und einen Index mit der Anweisung CREATE INDEX erstellen, ohne einen bestimmten Tabellenbereich anzugeben, wird der Index im Tabellenbereich der ersten zugeordneten bzw. sichtbaren Datenpartition erstellt. Zur Bestimmung, wo der Index zu erstellen ist, werden die drei folgenden möglichen Fälle in der angegebenen Reihenfolge (beginnend mit Fall 1) ausgewertet. Die Auswertung wird beendet, wenn einer der Fälle zutrifft:

```
Fall 1:
Wenn ein Tabellenbereich für den Index in der Anweisung CREATE INDEX ... IN
<tb_bereich1> angegeben wird, wird der in <tb_bereich1> angegebene
Tabellenbereich für diesen Index verwendet.
Fall 2:
Wenn ein Tabellenbereich für den Index in der Anweisung CREATE TABLE .. INDEX IN
<tb_bereich2> angegeben wird, wird der in <tb_bereich2> angegebene
Tabellenbereich für diesen Index verwendet.
Fall 3:
Wenn kein Tabellenbereich definiert ist, wird der Tabellenbereich verwendet, der
von der ersten zugeordneten oder sichtbaren Datenpartition verwendet wird.
```

Wo der Index erstellt wird, hängt davon ab, was bei der Ausführung der Anweisung CREATE TABLE geschieht. Wenn Sie für nicht partitionierte Tabellen keine Klausel INDEX IN angeben, fügt die Datenbank diese Information für Sie ein und gibt denselben Tabellenbereich wie für die Datentabelle an. Wenn Sie für partitionierte Tabelle diese Information weglassen, wird sie nicht nachgetragen und es gilt Fall 3.

Beispiel 1: Dieses Beispiel geht von der Annahme aus, dass eine partitionierte Tabelle 'sales' mit den Spalten (a int, b int, c int) vorhanden ist. Es wird ein eindeutiger Index 'a_idx' im Tabellenbereich 'ts1' erstellt.

```
CREATE UNIQUE INDEX a_idx ON sales ( a ) IN ts1
```

Beispiel 2: Dieses Beispiel geht von der Annahme aus, dass eine partitionierte Tabelle 'sales' mit den Spalten (a int, b int, c int) vorhanden ist. Es wird ein Index 'b_idx' im Tabellenbereich 'ts2' erstellt.

```
CREATE INDEX b_idx ONsales ( b ) IN ts2
```

Clusterindexverhalten bei partitionierten Tabellen

Clusterindizes bieten für partitionierte Tabellen die gleichen Vorteile wie für reguläre Tabellen. Allerdings ist bei der Auswahl eines Clusterindex im Hinblick auf die Definitionen von Tabellenpartitionierungsschlüsseln Sorgfalt geboten.

Sie können Clusterindizes für eine partitionierte Tabelle mit einem beliebigen Clusterschlüssel erstellen. Der Datenbankserver versucht den Clusterindex zu verwenden, um die Daten lokal in jeder Datenpartition in Clustern zusammenzufassen. Bei einer Einfügung über einen Clusterindex wird der Index durchsucht, um eine passende Zeilenkennung (Satz-ID, RID) zu ermitteln. Diese RID wird als Ausgangspunkt für die Suche nach einem Platz in der Tabelle zum Einfügen des Datensatzes verwendet. Zur Erzielung einer gut gepflegten Clusterbildung mit guter Leistung sollte es eine Korrelation zwischen den Indexspalten und den Spalten des Tabellenpartitionierungsschlüssels geben. Eine Methode, eine solche Korrelation sicherzustellen, besteht darin, den Indexspalten die Spalten des Tabellenpartitionierungsschlüssels als Präfix voranzustellen, wie im folgenden Beispiel gezeigt:

```
PARTITION BY RANGE (Month, Region)
CREATE INDEX ...(Month, Region, Department) CLUSTER
```

Obwohl der Datenbankserver diese Korrelation nicht zwingend umsetzt, ist zu erwarten, dass alle Schlüssel im Index nach Partitions-IDs zusammen gruppiert werden, um eine gute Clusterbildung zu erreichen. Wenn zum Beispiel eine Tabelle nach Quartal partitioniert ist und ein Clusterindex für Datum definiert wird, weil zwischen Quartal und Datum eine Beziehung besteht, kann eine optimale Clusterbildung der Daten mit guter Leistung erzielt werden, weil alle Schlüssel einer Datenpartition im Index zusammen gruppiert werden.

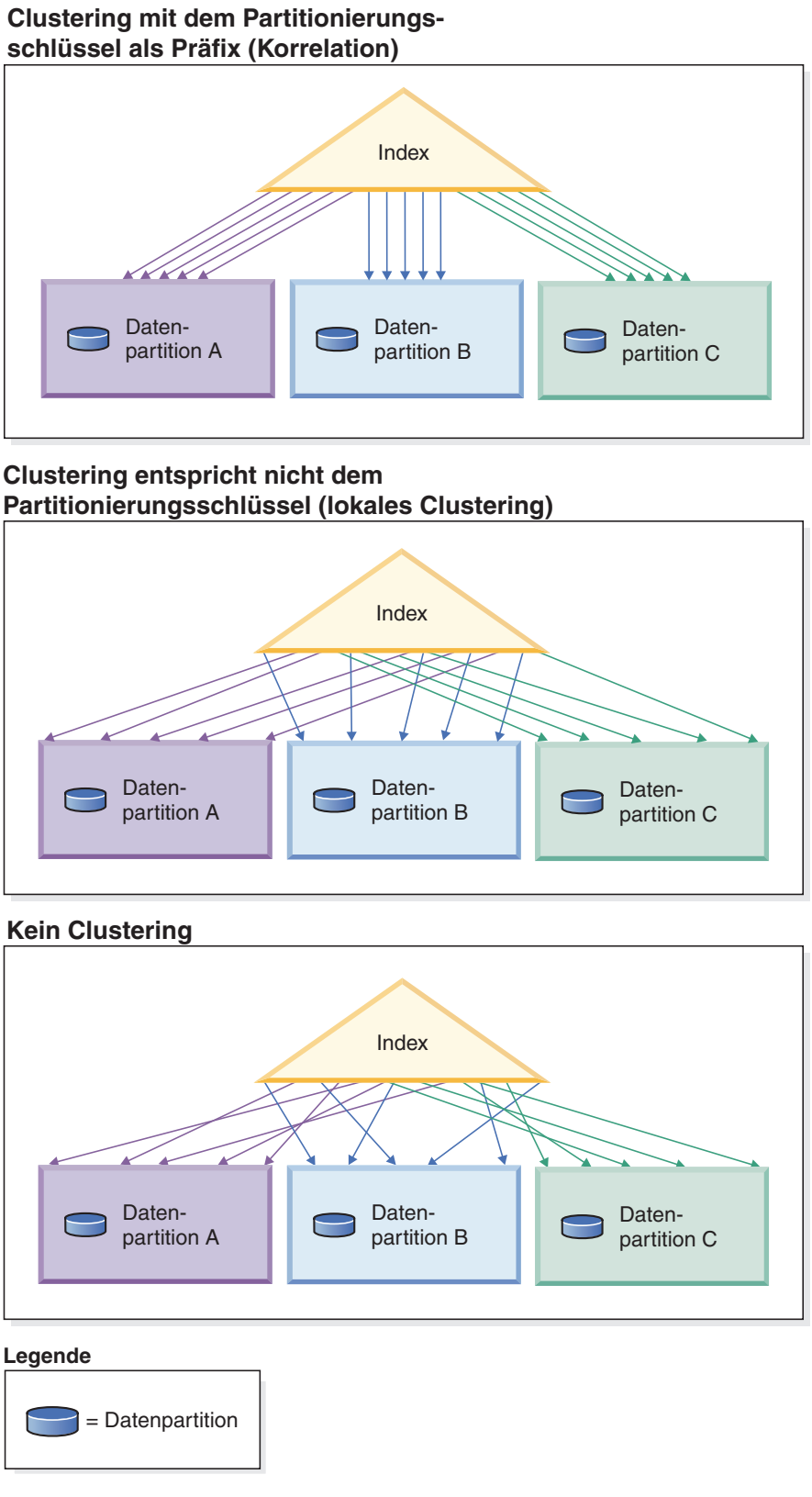


Abbildung 52. Die möglichen Effekte eines Clusterindex für eine partitionierte Tabelle. In der ersten Abbildung werden die Daten sowohl global als auch lokal zu Clustern zusammengefasst.

Wie in Abb. 52 auf Seite 328 gezeigt, wird bei dem gegebenen Layout des Indexes und der Daten in den einzelnen Beispielen eine optimale Suchleistung erzielt, wenn die Clusterbildung mit dem Tabellenpartitionierungsschlüssel korreliert ist. Wenn die Clusterbildung nicht mit dem Tabellenpartitionierungsschlüssel korreliert ist, ist es unwahrscheinlich, dass die Werte im Index lokal zu Clustern zusammengefasst werden. Da eine Korrelation zwischen den Tabellenpartitionierungsspalten und den Indexspalten erwartet wird, ist ein Szenario mit perfektem lokalen Clustering sehr unwahrscheinlich.

Das Clustering bietet folgende Vorteile:

- Innerhalb jeder Datenpartition sind die Zeilen in der Reihenfolge des Schlüssels angeordnet.
- Clusterindizes verbessern die Leistung von Suchoperationen, welche die Tabelle in der Reihenfolge der Schlüssel durchqueren. Dies liegt daran, dass die Suche die erste Zeile der ersten Seite abruft, dann jede Zeile auf eben dieser Seite, bis alle Zeilen für diese Seite abgerufen sind. Anschließend fährt die Suche mit der nächsten Seite fort. Dies bedeutet, dass sich nur eine Seite der Tabelle zu einem gegebenen Zeitpunkt im Pufferpool befinden muss. Wenn die Tabelle hingegen nicht zu Clustern zusammengefasst ist, wird jede Zeile mit hoher Wahrscheinlichkeit von einer anderen Seite abgerufen. Sofern im Pufferpool nicht genügend Platz zur Aufnahme der gesamten Tabelle ist, führt dies dazu, dass jede Seite mehr als einmal abgerufen wird und sich die Suche erheblich verlangsamt.

  Für partitionierte Tabellen kann der Idealfall, dass jede Seite bei der Suche jeweils nur einmal abgerufen wird, nur dann garantiert werden, wenn der Tabellenpartitionierungsschlüssel ein Präfix des Clusterschlüssels (siehe erstes Beispiel in Abb. 52 auf Seite 328 ist. Wenn der Clusterschlüssel jedoch nicht wie zuvor beschrieben mit dem Tabellenpartitionierungsschlüssel korreliert ist und die Daten lokal zu Clustern zusammengefasst sind, können Sie immer noch den vollen Vorteil des Clusterindex ausschöpfen, wenn im Pufferpool genügend Platz ist, um eine Seite jeder Datenpartition aufzunehmen. Der Grund hierfür ist, dass jede Zeile, die für eine gegebene Datenpartition abgerufen wird, in der Nähe einer vorigen Zeile liegt, die für dieselbe Datenpartition abgerufen wurde (siehe zweites Beispiel in Abb. 52 auf Seite 328). Wie zuvor erwähnt, wird die Clusterbildung möglicherweise nicht gut aufrechterhalten, wenn der Clusterschlüssel nicht mit dem Tabellenpartitionierungsschlüssel korreliert ist. Wenn Sie jedoch kein hohes Volumen an Einfüge-, Aktualisierungs- und Löschoperationen für Ihre Tabelle erwarten, sollte diese Lösung von Vorteil sein.

  Selbst wenn nicht genügend Platz für eine Seite jeder Datenpartition im Pufferpool vorhanden ist, lassen sich dennoch einige Vorteile aus der Definition eines Clusterindex gewinnen.

Kapitel 22. Designadvisor

Verwenden des Designadvisors zur Migration von einer Datenbank mit einer einzelnen Partition auf eine Datenbank mit mehreren Partitionen

Sie können den Designadvisor zur Unterstützung einer Migration von einer Datenbank mit einer Einzelpartition auf eine Datenbank mit mehreren Partitionen verwenden.

Informationen zu dieser Task

Der Designadvisor kann Ihnen Empfehlungen für die Verteilung Ihrer Daten und gleichzeitig Empfehlungen für neue Indizes, MQTs (Materialized Query Tables) und MDC-Tabellen (mit mehrdimensionalem Clustering) geben.

Vorgehensweise

1. Registrieren Sie einen DB2-Produkt- oder Featurelizenzschlüssel mithilfe des Befehls db2licm.
2. Erstellen Sie mindestens einen Tabellenbereich in einer Partitionsgruppe einer Datenbank mit mehreren Partitionen.

   Anmerkung: Da der Designadvisor nur eine Umverteilung für vorhandene Tabellenbereiche empfehlen kann, müssen die Tabellenbereiche, die der Designadvisor in Betracht ziehen soll, in der partitionierten Datenbank vorhanden sein, bevor der Designadvisor ausgeführt wird.
3. Führen Sie den Designadvisor mit ausgewählter Datenbankpartitionierungsfunktion über die Designadvisor-GUI oder unter Angabe der Partitionierungsoption mit dem Befehl db2advis aus.
4. Wenn Sie mit dem Designadvisor in der Steuerzentrale arbeiten, können Sie die Datenbankpartitionierungsempfehlungen automatisch implementieren. Wenn Sie den Befehl db2advis verwenden, müssen Sie die db2advis-Ausgabedatei geringfügig modifizieren, bevor Sie die vom Designadvisor generierten DDL-Anweisungen ausführen.

Kapitel 23. Verwalten des gemeinsamen Zugriffs

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

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

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

Tabelle 15. Sperrmodi für Tabellensuchen ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	S/-/-	U/-/-	SIX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/U	IX/X/-	IX/I/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-

Tabelle 16. Sperrmodi für Tabellensuchen mit Vergleichselementen nur mit Dimensionsspalten

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/X/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/U/-	X/X/-

Tabelle 17. Sperrmodi für Tabellensuchen mit anderen Vergleichselementen (sargs, resids)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

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

Tabelle 18. Sperrmodi für Satz-ID-Indexsuchen (RID) ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	S/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X

Tabelle 19. Sperrmodi für Satz-ID-Indexsuchen (RID) mit einer einzigen Qualifikationszeile

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/IS/S	IX/IX/U	IX/IX/X	X/X/X	X/X/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X

Tabelle 20. Sperrmodi für Satz-ID-Indexsuchen (RID) nur mit Start- und Stoppvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/IS/S	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X

Tabelle 21. Sperrmodi für Satz-ID-Indexsuchen (RID) nur mit Indexvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

Tabelle 22. Sperrmodi für Satz-ID-Indexsuchen (RID) mit anderen Vergleichselementen (sargs, resids)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

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

Tabelle 23. Sperrmodi für Satz-ID-Indexsuchen mit verzögertem Zugriff auf Datenseiten: Satz-ID-Indexsuche ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/S	IX/IX/S		X/-/-	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

Tabelle 24. Sperrmodi für Satz-ID-Indexsuchen mit verzögertem Zugriff auf Datenseiten: Verzögerter Zugriff auf Datenseiten nach einer Satz-ID-Indexsuche ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IN/IN/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X

Tabelle 25. Sperrmodi für Satz-ID-Indexsuchen mit verzögertem Zugriff auf Datenseiten: Satz-ID-Indexsuche mit Vergleichselementen (sargs, resids)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/-	IX/IX/S		IX/IX/S	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

Tabelle 26. Sperrmodi für Satz-ID-Indexsuchen mit verzögertem Zugriff auf Datenseiten: Verzögerter Zugriff auf Datenseiten nach einer Satz-ID-Indexsuche mit Vergleichselementen (sargs, resids)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

Tabelle 27. Sperrmodi für Satz-ID-Indexsuchen mit verzögertem Zugriff auf Datenseiten: Satz-ID-Indexsuche nur mit Start- und Stoppvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/IS/S	IX/IX/S		IX/IX/X	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

Tabelle 28. Sperrmodi für Satz-ID-Indexsuchen mit verzögertem Zugriff auf Datenseiten: Verzögerter Zugriff auf Datenseiten nach einer Satz-ID-Indexsuche nur mit Start- und Stoppvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

Tabelle 28. Sperrmodi für Satz-ID-Indexsuchen mit verzögertem Zugriff auf Datenseiten: Verzögerter Zugriff auf Datenseiten nach einer Satz-ID-Indexsuche nur mit Start- und Stoppvergleichselementen (Forts.)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
UR	IS/-/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

Sperren für Blockindexsuchen für MDC-Tabellen

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

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

Tabelle 29. Sperrmodi für Indexsuchen ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	S/--/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/--	X/X/--

Tabelle 30. Sperrmodi für Indexsuchen nur mit Dimensionsvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-

Tabelle 31. Sperrmodi für Indexsuchen nur mit Start- und Stoppvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/-	IX/IX/S	IX/IX/S	IX/IX/S	IX/IX/S

Tabelle 31. Sperrmodi für Indexsuchen nur mit Start- und Stoppvergleichselementen (Forts.)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
CS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-

Tabelle 32. Sperrmodi für Indexsuchen mit Vergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

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

Tabelle 33. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Blockindexsuche ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/--	IX/IX/S		X/--/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

Tabelle 34. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Verzögerter Zugriff auf Datenseiten nach einer Blockindexsuche ohne Vergleichselemente

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IN/IN/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	X/X/--	X/X/--

Tabelle 35. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Blockindexsuche nur mit Dimensionsvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/--	IX/IX/--		IX/S/--	
RS	IS/IS/NS	IX/--/--		IX/--/--	
CS	IS/IS/NS	IX/--/--		IX/--/--	
UR	IN/IN/--	IX/--/--		IX/--/--	

Tabelle 36. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Verzögerter Zugriff auf Datenseiten nach einer Blockindexsuche nur mit Dimensionsvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/S/--	IX/X/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--

Tabelle 37. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Blockindexsuche nur mit Start- und Stoppvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/--	IX/IX/--		IX/X/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

Tabelle 38. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Verzögerter Zugriff auf Datenseiten nach einer Blockindexsuche nur mit Start- und Stoppvergleichselementen

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IN/IN/--	IX/IX/X		IX/X/--	
RS	IS/IS/NS	IN/IN/--		IN/IN/--	
CS	IS/IS/NS	IN/IN/--		IN/IN/--	

Tabelle 38. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Verzögerter Zugriff auf Datenseiten nach einer Blockindexsuche nur mit Start- und Stoppvergleichselementen (Forts.)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
UR	IS/--/--	IN/IN/--		IN/IN/--	

Tabelle 39. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Blockindexsuche mit anderen Vergleichselementen (sargs, resids)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IS/S/--	IX/IX/--		IX/IX/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

Tabelle 40. Sperrmodi für Blockindexsuchen mit verzögertem Zugriff auf Datenseiten: Verzögerter Zugriff auf Datenseiten nach einer Blockindexsuche mit anderen Vergleichselementen (sargs, resids)

Isolationsstufe	Suchen: mehrdeutig und Lesezugriff	Cursorgesteuerte Operationen		UPDATE oder DELETE mit Suche	
		Suche	WHERE CURRENT OF	Suche od. DELETE	UPDATE
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

Sperrverhalten für partitionierte Tabellen

Zusätzlich zu einer Sperre für die gesamte Tabelle wird eine Sperre für jede Datenpartition einer partitionierten Tabelle aktiviert. Auf diese Weise können die zu sperrenden Bereiche besser differenziert und der gemeinsame Zugriff im Vergleich zu einer nicht partitionierten Tabelle erhöht werden. Die neue Datenpartitionssperre wird in der Ausgabe des Befehls 'db2pd', des Ereignismonitors, der Verwaltungssichten und der Tabellenfunktionen ausgewiesen.

Beim Zugriff auf eine Tabelle wird von der Sperrfunktion zuerst die Tabellensperre gesetzt. Anschließend werden die erforderlichen Datenpartitionssperren gesetzt, die sich nach den Daten richten, auf die zugegriffen werden soll. Aufgrund der verwendeten Zugriffsmethoden und Isolationsstufen ist es möglich, dass Datenpartitionen gesperrt werden müssen, die sich nicht in der Ergebnismenge befinden. Wenn diese Datenpartitionssperren aktiviert wurden, können sie möglicherweise ebenso

lange wie die Tabellensperre beibehalten werden. Bei einem Indexsuchvorgang mit der Isolationsstufe CS (Cursorstabilität) werden die Sperren für zuvor verwendete Datenpartitionen vieleicht beibehalten, um den Aufwand für das erneute Sperren einer Datenpartition zu vermeiden, wenn diese Datenpartition in nachfolgenden Schlüsseln referenziert wird. Die Datenpartitionssperre umfasst auch den Aufwand für die Sicherstellung des Zugriffs auf die Tabellenbereiche. Bei nicht partitionierten Tabellen wird der Zugriff auf Tabellenbereiche durch die Tabellensperre gesteuert. Aus diesem Grund werden Datenpartitionen auch dann gesperrt, wenn eine exklusive oder den gemeinsamen Zugriff zulassende (SHARE) Sperre auf Tabellenebene für eine partitionierte Tabelle aktiviert wurde.

Durch die feinere Differenzierung kann eine Transaktion exklusiven Zugriff auf eine bestimmte Datenpartition haben und Zeilensperren vermeiden, während andere Transaktionen auf die anderen verfügbaren Datenpartitionen zugreifen können. Dieser Fall kann infolge eines für eine Massenaktualisierung ausgewählten Zugriffsplans oder durch eine Eskalation von Sperren auf die Datenpartitionsebene eintreten. Als Tabellensperre für zahlreiche Zugriffsmethoden wird normalerweise eine Intent-Sperre verwendet. Dies gilt auch dann, wenn für die Datenpartitionen eine den gemeinsamen Zugriff zulassende oder eine exklusive Sperre aktiviert wird. Auf diese Weise kann der gemeinsame Zugriff verbessert werden. Wenn jedoch auf Datenpartitionsebene Nicht-Intent-Sperren erforderlich sind und im Zugriffsplan angegeben ist, dass unter Umständen auf alle Datenpartitionen zugegriffen wird, wird möglicherweise auf Tabellenebene eine Nicht-Intent-Sperre ausgewählt, um Deadlocks zwischen Datenpartitionssperren bei gleichzeitig zugreifenden Transaktionen zu vermeiden.

Sperren für die SQL-Anweisung LOCK TABLE

Bei partitionierten Tabellen werden für die Anweisung LOCK TABLE ausschließlich Sperren auf Tabellenebene aktiviert. Datenpartitionssperren werden hingegen nicht aktiviert. Auf diese Weise wird sichergestellt, dass bei nachfolgenden DML-Anweisungen keine Zeilensperren für die Tabelle aktiviert werden. Außerdem werden so Deadlocks auf Zeilen-, Block- oder Datenpartitionsebene vermieden. Mit der Anweisung LOCK TABLE IN EXCLUSIVE MODE kann beim Aktualisieren von Indizes ebenfalls ein exklusiver Zugriff sichergestellt werden. Dies ist sinnvoll, um das Anwachsen von Indizes des Typs 2 während umfangreicher Aktualisierungsoperationen zu begrenzen.

Auswirkung des Parameters LOCKSIZE TABLE der Anweisung ALTER TABLE

Die Anweisung ALTER TABLE verfügt über die Option zum Einstellen von LOCKSIZE TABLE, durch die sichergestellt werden kann, dass für die Tabelle eine den gemeinsamen Zugriff zulassende (SHARE) oder eine exklusive Sperre ohne Intent-Sperren definiert wird. Bei einer partitionierten Tabelle wird diese Sperrenstrategie sowohl bei der Tabellensperre als auch bei den Datenpartitionssperren für alle betroffenen Datenpartitionen angewendet.

Zeilen- und Blocksperreneskalation

Bei partitionierten Tabellen werden durch eine Sperreneskalation Zeilen- und Blocksperren in Datenpartitionssperren umgewandelt. Dies bedeutet wiederum, dass andere Transaktionen besser auf die Tabelle zugreifen können, selbst wenn für eine Datenpartition eine Eskalation auf Sperren mit gemeinsamem Zugriff oder exklusive Sperren (Share, Exclusive oder Super Exclusive) erfolgt, andere Datenpartitionen, die nicht eskaliert werden, davon jedoch unberührt bleiben. Nach einer Es-

kalation in einer bestimmten Datenpartition setzt die Transaktion die Aktivierung von Zeilensperren in anderen Datenpartitionen möglicherweise fort. Die Benachrichtigungsprotokollnachricht für Eskalationen umfasst Informationen zu der Datenpartition, in der die Eskalation aufgetreten ist, sowie zum Namen der betroffenen partitionierten Tabelle. Aus diesem Grund kann der exklusive Zugriff auf einen Index durch die Sperreneskalation nicht sichergestellt werden. Entweder muss in der Anweisung eine exklusive Sperre auf Tabellenebene verwendet, eine explizite Anweisung LOCK TABLE IN EXCLUSIVE MODE ausgeführt oder von der Tabelle das Attribut LOCKSIZE TABLE verwendet werden. Die Gesamttabellensperre für die Zugriffsmethode wird vom Optimierungsprogramm ausgewählt und ist vom Ausschluss von Datenpartitionen abhängig. Eine umfangreiche Aktualisierung an einer Tabelle kann zur Auswahl einer exklusiven Tabellensperre führen, wenn kein Ausschluss von Datenpartitionen erfolgt.

Interpretieren von Informationen über Sperren

Das folgende Beispiel aus der Verwaltungssicht SNAPLOCK kann Sie bei der Interpretation der für eine partitionierte Tabelle zurückgegebenen Informationen über Sperren unterstützen.

Beispiel 1:

Die folgende Verwaltungssicht SNAPLOCK wurde während einer Offline-Indexreorganisation erfasst.

```
SELECT SUBSTR(TABNAME, 1, 15) TABNAME, TAB_FILE_ID, SUBSTR(TBSP_NAME, 1, 15) TBSP_NAME, DATA_PARTITION_ID, LOCK_OBJECT_TYPE,
LOCK_MODE, LOCK_ESCALATION FROM SYSIBMADM.SNAPLOCK where TABNAME like 'TP1' and LOCK_OBJECT_TYPE like 'TABLE_%'
ORDER BY TABNAME, DATA_PARTITION_ID, LOCK_OBJECT_TYPE, TAB_FILE_ID, LOCK_MODE

TABNAME         TAB_FILE_ID          TBSP_NAME       DATA_PARTITION_ID LOCK_OBJECT_TYPE   LOCK_MODE  LOCK_ESCALATION
--------------- -------------------- --------------- ----------------- ------------------ ---------- ---------------
TP1                           32768 -                              -1 TABLE_LOCK         Z                       0
TP1                               4 USERSPACE1                      0 TABLE_PART_LOCK    Z                       0
TP1                               5 USERSPACE1                      1 TABLE_PART_LOCK    Z                       0
TP1                               6 USERSPACE1                      2 TABLE_PART_LOCK    Z                       0
TP1                               7 USERSPACE1                      3 TABLE_PART_LOCK    Z                       0
TP1                               8 USERSPACE1                      4 TABLE_PART_LOCK    Z                       0
TP1                               9 USERSPACE1                      5 TABLE_PART_LOCK    Z                       0
TP1                              10 USERSPACE1                      6 TABLE_PART_LOCK    Z                       0
TP1                              11 USERSPACE1                      7 TABLE_PART_LOCK    Z                       0
TP1                              12 USERSPACE1                      8 TABLE_PART_LOCK    Z                       0
TP1                              13 USERSPACE1                      9 TABLE_PART_LOCK    Z                       0
TP1                              14 USERSPACE1                     10 TABLE_PART_LOCK    Z                       0
TP1                              15 USERSPACE1                     11 TABLE_PART_LOCK    Z                       0
TP1                               4 USERSPACE1                      - TABLE_LOCK         Z                       0
TP1                               5 USERSPACE1                      - TABLE_LOCK         Z                       0
TP1                               6 USERSPACE1                      - TABLE_LOCK         Z                       0
TP1                               7 USERSPACE1                      - TABLE_LOCK         Z                       0
TP1                               8 USERSPACE1                      - TABLE_LOCK         Z                       0
TP1                               9 USERSPACE1                      - TABLE_LOCK         Z                       0
TP1                              10 USERSPACE1                      - TABLE_LOCK         Z                       0
TP1                              11 USERSPACE1                      - TABLE_LOCK         Z                       0
TP1                              12 USERSPACE1                      - TABLE_LOCK         Z                       0
TP1                              13 USERSPACE1                      - TABLE_LOCK         Z                       0
TP1                              14 USERSPACE1                      - TABLE_LOCK         Z                       0
TP1                              15 USERSPACE1                      - TABLE_LOCK         Z                       0
TP1                              16 USERSPACE1                      - TABLE_LOCK         Z                       0

  26 Satz/Sätze ausgewählt.
```

In diesem Beispiel werden der Sperrobjekttyp TABLE_LOCK und die Datenpartitions-ID (DATA_PARTITION_ID) mit dem Wert -1 verwendet, um den Zugriff auf die partitionierte Tabelle TP1 und die gleichzeitige Verwendung dieser Tabelle zu steuern. Die Sperrobjekte des Typs TABLE_PART_LOCK dienen zur Steuerung des größtmöglichen Zugriffs auf jede Datenpartition und ihrer größtmöglichen gemeinsamen Nutzung.

Darüber hinaus werden in dieser Ausgabe weitere Sperrobjekte des Typs TABLE-_LOCK erfasst (mit TAB_FILE_ID 4 bis 16), die keinen Wert für DATA_PARTITIO-N_ID haben. Eine Sperre dieses Typs, bei der ein Objekt mit einer TAB_FILE_ID und einem TBSP_NAME einer Datenpartition oder einem Index für die partitionierte Tabelle entsprechen, kann verwendet werden, um den gleichzeitigen Zugriff mit dem Online-Backup-Dienstprogramm zu steuern.

Kapitel 24. Agentenverwaltung

Agenten in einer partitionierten Datenbank

In Umgebungen mit partitionierten Datenbanken und Umgebungen mit aktivierter partitionsinterner Parallelität verfügt jede Datenbankpartition (d. h. jeder Datenbankserver oder Knoten) über einen eigenen Pool von Agenten, aus dem Subagenten entnommen werden. Durch die Verwendung dieses Pools müssen Subagenten nicht jedes Mal erstellt und wieder gelöscht werden, wenn ein Subagent benötigt wird oder seine Arbeit beendet hat. Die Subagenten können als zugeordnete Agenten im Pool bleiben und vom Datenbankmanager für neue Anforderungen von der Anwendung, der sie zugeordnet sind, oder von neuen Anwendungen verwendet werden.

In Umgebungen mit partitionierten Datenbanken und Umgebungen mit aktivierter partitionsinterner Parallelität besteht eine enge Beziehung zwischen dem Einfluss auf die Leistung und den Speicherbedarf innerhalb des Systems und der Optimierung des Agentenpools:

- Der Konfigurationsparameter des Datenbankmanagers für die Größe des Agentenpools (*num_poolagents*) betrifft die Gesamtanzahl von Agenten und Subagenten, die Anwendungen in einer Datenbankpartition (auch als Knoten bezeichnet) zugeordnet bleiben können. Wenn die Poolgröße zu klein ist und der Pool voll ist, wird ein Subagent aus der Zuordnung mit der Anwendung, für die er aktiv ist, gelöst und beendet. Da Subagenten ständig erstellt und Anwendungen erneut zugeordnet werden müssen, sinkt die Leistung.

  Standardmäßig ist der Konfigurationsparameter *num_poolagents* auf AUTOMATIC mit dem Wert 100 gesetzt. Wenn dieser Parameter auf AUTOMATIC gesetzt ist, verwaltet der Datenbankmanager automatisch die Anzahl inaktiver Agenten, die in einem Pool zusammengefasst werden sollen.

  Außerdem könnte bereits eine Anwendung den Pool mit zugeordneten Subagenten füllen, wenn der Wert für den Parameter *num_poolagents* zu klein ist. Wenn nun eine andere Anwendung einen neuen Subagenten benötigt und im zugeordneten Pool über keine Agenten verfügt, stoppt sie Subagenten aus den Agentenpools anderer Anwendungen und startet sie erneut. Dieses Verhalten stellt sicher, dass Ressourcen vollständig belegt werden.
- Wägen Sie die Vor- und Nachteile zu weniger Agenten gegen den Ressourcenbedarf zu vieler Agenten ab, die zu einem gegebenen Zeitpunkt aktiv sein können.

  Wenn der Wert des Parameters *num_poolagents* zum Beispiel zu groß ist, werden zugeordnete Subagenten lange Zeit ungenutzt im Pool behalten und belegen Datenbankmanagerressourcen, die für andere Tasks nicht verfügbar sind.

  Anmerkung: Wenn der Verbindungskonzentrator aktiviert ist, spiegelt die Anzahl von Agenten, die durch den Parameter *num_poolagents* angegeben wird, nicht unbedingt die exakte Anzahl von Agenten wider, die sich zu einem bestimmten Zeitpunkt inaktiv im Pool befinden können. Die Anzahl von Agenten kann zeitweise überschritten werden, um Perioden mit höherer Auslastung aufzufangen.

Andere asynchrone Prozesse und Threads

Neben den Datenbankagenten werden auch noch andere asynchrone Aktivitäten des Datenbankmanagers als eigene Prozesse bzw. Threads ausgeführt. Dazu gehören:

- E/A-Server oder E/A-Vorablesefunktionen der Datenbank
- Asynchrone Seitenlöschfunktionen der Datenbank
- Protokollfunktionen der Datenbank
- Deadlock-Detektoren für Datenbanken
- Übertragungs- und IPC-Empfangsprozesse (Listeners)
- Funktionen zum Neuausgleich von Daten in Tabellenbereichscontainern

Kapitel 25. Optimieren von Zugriffsplänen

Indexzugriff und Clusterverhältnisse

Bei der Auswahl des Zugriffsplans schätzt das Optimierungsprogramm die Anzahl der E/A-Operationen ab, die zum Einlesen der benötigten Seiten von der Platte in den Pufferpool erforderlich sind. Diese Schätzung schließt eine Voraussage über die Nutzung des Pufferpools mit ein, da zum Lesen von Zeilen einer Seite, die sich bereits im Pufferpool befindet, keine weiteren E/A-Operationen anfallen.

Für Indexsuchen wird das Optimierungsprogramm durch Informationen aus den Systemkatalogtabellen (SYSCAT.INDEXES) bei der Abschätzung des Ein-/Ausgabeaufwands zum Lesen von Datenseiten in den Pufferpool unterstützt. Dabei werden Informationen aus den folgenden Spalten der Tabelle SYSCAT.INDEXES verwendet:

- Die Informationen der Spalte CLUSTERRATIO geben den Grad an, zu dem die Tabellendaten in Relation zu diesem Index in Clustern zusammengefasst sind (Clusterbildung). Je höher der Wert, desto besser sind die Zeilen in der Reihenfolge des Indexschlüssels geordnet. Wenn Tabellenzeilen nahe an der Reihenfolge des Indexschlüssels vorliegen, können Zeilen von einer Datenseite gelesen werden, während sich die Seite im Puffer befindet. Wenn der Wert dieser Spalte -1 ist, verwendet das Optimierungsprogramm die Informationen der Spalten PAGE_FETCH_PAIRS und CLUSTERFACTOR, wenn diese verfügbar sind.
- Die Spalte PAGE_FETCH_PAIRS enthält Paare von Zahlen, die zusammen mit CLUSTERFACTOR-Informationen jeweils ein Modell für die Anzahl der E/A-Operationen zum Lesen der Datenseiten in Pufferpools verschiedener Größen angeben. Für diese Spalten werden Daten nur erfasst, wenn Sie das Dienstprogramm RUNSTATS für den Index mit der Klausel DETAILED ausführen.

Wenn keine Statistiken zur Clusterbildung verfügbar sind, verwendet das Optimierungsprogramm Standardwerte, die von einem geringen Grad an Clusterbildung der Daten bezüglich des Index ausgehen.

Der Grad, zu dem die Daten in Bezug auf einen Index in Clustern zusammengefasst sind, kann bedeutende Auswirkungen auf die Leistung haben, sodass einer der Indizes für eine Tabelle auf einem Grad nahe an 100% Clusterbildung gehalten werden sollte.

Im Allgemeinen kann nur ein Index eine 100-prozentige Clusterbildung aufweisen. Eine Ausnahme bilden nur solche Fälle, in denen die Schlüssel eines anderen Index eine Obermenge der Schlüssel des Clusterindex sind oder in denen es eine De-Facto-Korrelation zwischen den Schlüsselspalten der beiden Indizes gibt.

Bei der Reorganisation einer Tabelle können Sie einen Index angeben, über den die Zeilen in Clustern angeordnet werden und versucht wird, diese Clusterbildung bei der Verarbeitung von Einfügungen beizubehalten. Da Aktualisierungen und Einfügungen die Clusterbildung in Bezug auf den Index verringern können, müssen Sie die Tabelle eventuell in regelmäßigen Abständen reorganisieren. Um die Häufigkeit der Reorganisation einer Tabelle zu verringern, die aufgrund von INSERT-, UPDATE- und DELETE-Anweisungen häufig geändert wird, können Sie beim Ändern einer Tabelle (ALTER TABLE) den Parameter PCTFREE verwenden. Dieser Parameter ermöglicht es, weitere Einfügungen so in die vorhandenen Daten einzugliedern, dass die Clusterbildung erhalten bleibt.

Tabellen- und Indexverwaltung für MDC-Tabellen

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

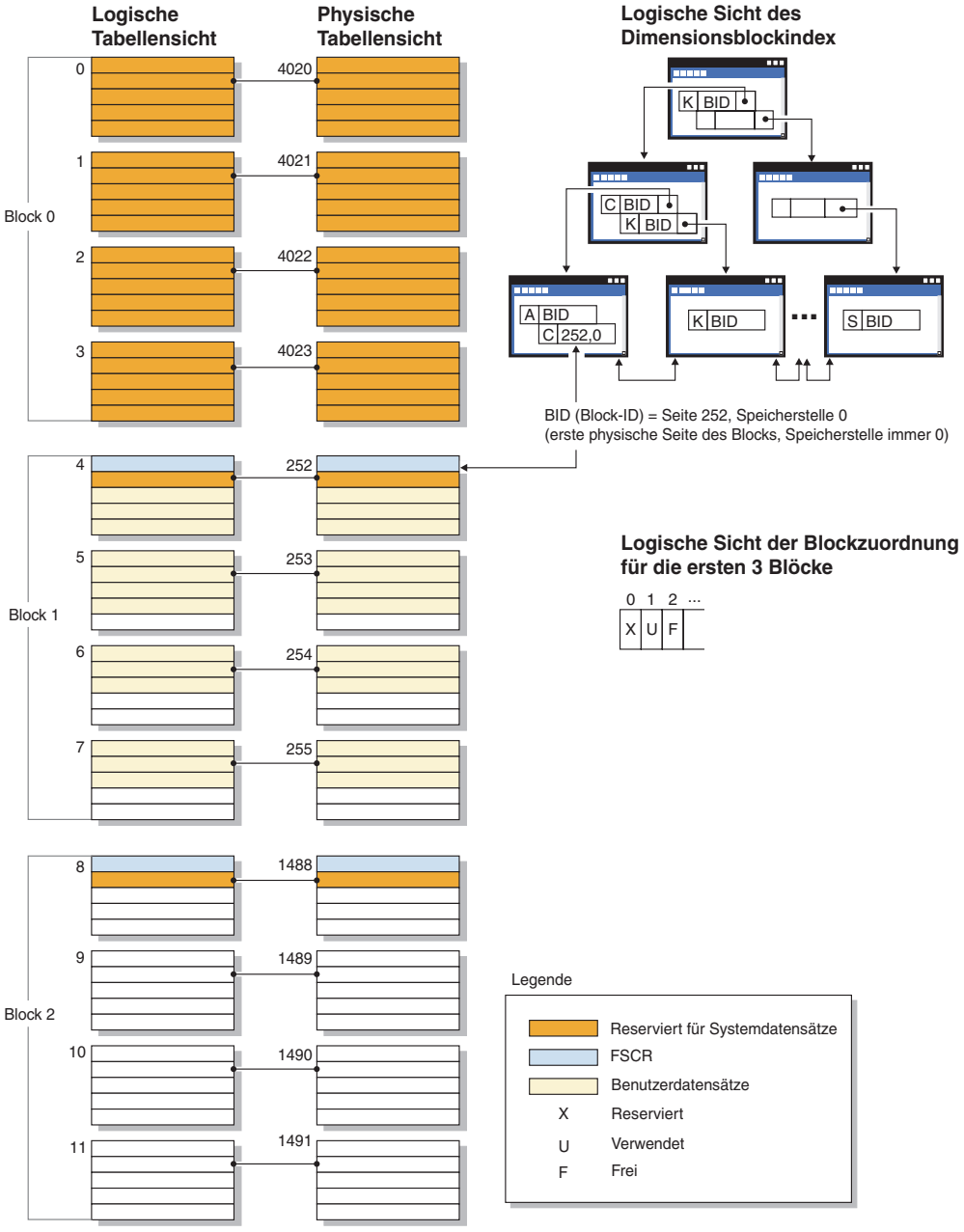


Abbildung 53. Logische Tabellen-, Datensatz- und Indexstruktur für MDC-Tabellen

Der Block enthält besondere interne Datensätze, die von DB2 zur Verwaltung der Tabelle verwendet werden. Dazu gehören auch die FSCR-Sätze, d. h., die Steuerdatensätze für freie Speicherbereiche (FSCR - Free-Space Control Record). In den nachfolgenden Blöcken enthält jeweils die erste Seite die FSCR-Sätze. Ein FSCR-Satz ordnet den freien Speicher für neue Datensätze zu, der auf den jeweiligen Seiten im Block vorhanden ist. Dieser verfügbare freie Speicherbereich wird verwendet, wenn Datensätze in die Tabelle eingefügt werden.

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

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

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

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

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

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

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

Clustering

Im Verlauf der Zeit kann es durch Aktualisierungen zu einer Positionsverlagerung bei den Zeilen in Datenseiten kommen, wodurch sich der *Clusterbildungsgrad* reduziert, der zwischen einem Index und den Datenseiten besteht.

Die Reorganisation einer Tabelle in Bezug auf einen ausgewählten Index dient zur erneuten Ausführung des Datenclusterings. Ein Clusterindex eignet sich am Besten für Spalten, die über Bereichsvergleichselemente verfügen, da dieser einen verbesserten sequenziellen Zugriff auf die Daten der Basistabelle ermöglicht. Hierdurch werden weniger Seitenabrufoperationen benötigt, da ähnliche Werte sich in derselben Datenseite befinden.

Im Allgemeinen kann nur einer der Indizes in einer Tabelle einen hohen Clusterbildungsgrad aufweisen.

Um den Grad der Clusterbildung für einen Index zu überprüfen, müssen Sie seinen Knoten doppelt anklicken, um das Fenster **Indexstatistik** aufzurufen. In diesem Fenster werden die Werte für das Clusterverhältnis oder den Clusterfaktor angezeigt. Bei einem niedrigen Wert sollten Sie eine Reorganisation der Tabellendaten in Betracht ziehen.

Optimierungsstrategien für partitionsinterne Parallelität

Das Optimierungsprogramm kann einen Zugriffsplan wählen, der eine Abfrage parallel innerhalb einer Datenbankpartition ausführt, wenn ein Grad von Parallelität bei der Kompilierung der SQL-Anweisung angegeben wird.

Während der Ausführung werden mehrere Datenbankagenten, so genannte Subagenten, zur Ausführung der Abfrage erstellt. Die Anzahl von Subagenten ist kleiner oder gleich dem Grad von Parallelität, der bei der Kompilierung der SQL-Anweisung angegeben wurde.

Zur Parallelisierung unterteilt das Optimierungsprogramm den Zugriffsplan in Teile, die jeweils von einem Subagenten, sowie in einen Teil, der vom koordinierenden Agenten ausgeführt wird. Die Subagenten übergeben Daten über Tabellenwarteschlangen an den koordinierenden Agenten oder andere Subagenten. In einer Umgebung mit partitionierten Datenbanken können Subagenten Daten an Subagenten in anderen Datenbankpartitionen senden oder von ihnen empfangen.

Strategien zur partitionsinternen Parallelsuche

Tabellensuchen und Indexsuchen können parallel in derselben Tabelle oder demselben Index ausgeführt werden. Für parallele Tabellensuchen wird die Tabelle in Seiten- oder Zeilenbereiche unterteilt. Je ein Bereich von Seiten oder Zeilen wird einem Subagenten zugewiesen. Ein Subagent durchsucht den ihm zugewiesenen Bereich und erhält einen anderen Bereich zugewiesen, wenn er mit dem Durchsuchen des aktuellen Bereichs fertig ist.

Für parallele Indexsuchen wird der Index in Bereiche von Datensätzen entsprechend den Indexschlüsselwerten und der Anzahl von Indexeinträgen für einen Schlüsselwert unterteilt. Die parallele Indexsuche wird wie die parallele Tabellensuche mit Subagenten durchgeführt, denen jeweils ein Bereich von Datensätzen zugewiesen wird. Einem Subagenten wird ein neuer Bereich zugewiesen, wenn er die Suche im aktuellen Bereich beendet hat.

Das Optimierungsprogramm legt die Sucheinheit (entweder Seite oder Zeile) sowie die Suchgranularität fest.

Bei Parallelsuchen wird die Arbeit gleichmäßig unter den Subagenten verteilt. Der Zweck einer Parallelsuche besteht darin, die Belastung unter den Subagenten ausgewogen zu verteilen und sie gleichmäßig auszulasten. Wenn die Anzahl aktiver Subagenten gleich der Anzahl verfügbarer Prozessoren ist und die Platten nicht mit E/A-Anforderungen überlastet sind, werden die Ressourcen der Maschine effektiv genutzt.

Andere Zugriffsplanstrategien können eine unausgewogene Datenverteilung bei der Ausführung der Abfrage verursachen. Das Optimierungsprogramm wählt Parallelstrategien aus, die für eine ausgewogene Datenverteilung unter den Subagenten sorgen.

Strategien zur partitionsinternen parallelen Sortierung

Das Optimierungsprogramm kann eine der folgenden parallelen Sortierstrategien auswählen:

- **Reihumsortierung**

  Dies kann auch als *Umverteilungssortieren* bezeichnet werden. Diese Methode nutzt den gemeinsamen Speicher effizient, um die Daten so gleichmäßig wie möglich auf alle Subagenten zu verteilen. Zur Realisierung der gleichmäßigen Verteilung wird eine Art Reihumverteilungsalgorithmus verwendet. Zunächst wird ein Sortiervorgang für jeden Subagenten erstellt. Während der Einfügephase fügen Subagenten Zeilen reihum in jeden der einzelnen Sortiervorgänge ein, um eine gleichmäßigere Datenverteilung zu erzielen.
- **Partitionierte Sortierung**

  Dies ist dem reihumverteilten Sortieren insofern ähnlich, als dass für jeden Subagenten ein Sortiervorgang erstellt wird. Die Subagenten wenden eine Hashfunktion auf die Sortierspalten an, um festzulegen, in welchen Sortiervorgang eine Zeile einzufügen ist. Wenn beispielsweise die innere und die äußere Tabelle eines Mischjoins an einem partitionierten Sortiervorgang beteiligt sind, kann ein Subagent mithilfe eines Mischjoins die entsprechenden Teile verknüpfen und parallel ausgeführt werden.
- **Replizierte Sortierung**

  Diese Art der Sortierung wird verwendet, wenn jeder Subagent die gesamte Ausgabe der Sortierung benötigt. Es wird nur ein Sortiervorgang erstellt. Beim Einfügen von Zeilen in den Sortiervorgang werden die Subagenten synchronisiert. Nach Beendigung der Sortierung liest jeder Subagent das gesamte Sortierergebnis. Wenn die Anzahl von Zeilen gering ist, kann diese Art der Sortierung zum erneuten Ausgleich des Datenstroms verwendet werden.
- **Gemeinsame Sortierung**

  Diese Art der Sortierung entspricht der replizierten Sortierung, abgesehen davon, dass die Subagenten eine parallele Suche über das Sortierergebnis öffnen, um die Daten unter den Subagenten ähnlich wie bei einer Reihumsortierung zu verteilen.

Partitionsinterne temporäre Paralleltabellen

Subagenten können kooperieren, um eine temporäre Tabelle durch Einfügen von Zeilen in dieselbe Tabelle zu erstellen. Eine solche Tabelle ist eine `gemeinsame temporäre Tabelle`. Die Subagenten können private oder parallele Suchoperationen

über die gemeinsam genutzte temporäre Tabelle öffnen; dies ist davon abhängig, ob der Datenstrom repliziert oder geteilt werden muss.

Strategien zur partitionsinternen parallelen Spaltenberechnung

Spaltenberechnungen (Aggregation) können von Subagenten parallel durchgeführt werden. Eine Spaltenberechnung setzt voraus, dass die Daten nach den Gruppierungsspalten geordnet sind. Wenn ein Subagent sicher sein kann, dass er alle Zeilen für eine Reihe von Gruppierungsspaltenwerten erhält, kann er eine vollständige Spaltenberechnung (Aggregation) durchführen. Dies ist möglich, wenn der Strom bereits aufgrund einer früheren partitionierten Sortierung über die Gruppierungsspalten geteilt ist.

Andernfalls kann der Subagent eine partielle Spaltenberechnung durchführen und eine andere Strategie zur Vervollständigung der Spaltenberechnung anwenden. Einige dieser Strategien sind:

- Senden der teilweise berechneten Daten an den Koordinatoragenten über eine Tabellenwarteschlange für Mischjoins. Der Koordinatoragent vervollständigt die Spaltenberechnung.
- Einfügen der teilweise berechneten Daten in eine partitionierte Sortierung. Die Sortierung wird über die Gruppierungsspalten geteilt und stellt sicher, dass alle Zeilen für eine Reihe von Gruppierungsspalten in einer Sortierpartition enthalten sind.
- Wenn der Strom zum Ausgleichen der Verarbeitung repliziert werden muss, können die teilweise berechneten Daten in eine replizierte Sortierung eingefügt werden. Die einzelnen Subagenten vervollständigen die Spaltenberechnung über die replizierte Sortierung und erhalten eine identische Kopie des Ergebnisses der Spaltenberechnung.

Strategien zu partitionsinternen parallelen Joins

Joinoperationen können von Subagenten parallel durchgeführt werden. Parallele Joinsrategien werden durch die Merkmale des Datenstroms festgelegt.

Ein Join kann parallelisiert werden, indem der Datenstrom nach der inneren und äußeren Tabelle des Joins partitioniert, repliziert oder beides wird. Zum Beispiel kann ein Join mit Verschachtelungsschleife parallelisiert werden, wenn der äußere Datenstrom für eine Parallelsuche partitioniert ist und der innere Datenstrom von jedem Subagenten unabhängig neu ausgewertet wird. Ein Mischjoin kann parallelisiert werden, wenn der innere und der äußere Datenstrom für partitionierte Sortierungen nach ihren Werten partitioniert sind.

Beispiele für die Ausgabe von db2expln und dynexpln

Zum besseren Verständnis finden Sie im Folgenden Beispiele, die den Aufbau und das Format der Ausgabe von `db2expln` und `dynexpln` verdeutlichen. Diese Beispiele wurden für die Beispieldatenbank SAMPLE ausgeführt, die mit DB2 ausgeliefert wird. Jedes Beispiel wird kurz erläutert. Die signifikanten Unterschiede von einem Beispiel zum nächsten wurden **fett** hervorgehoben.

Beispiel 1: keine Parallelität

In diesem Beispiel wird einfach eine Liste aller Namen von Mitarbeitern (employee), ihrer Aufgaben (job), ihrer Abteilungen (department) und Standorte (location) sowie der Namen der Projekte, an denen sie arbeiten, abgerufen. Das wesentliche Merkmal dieses Zugriffsplans besteht darin, dass die relevanten Daten aus allen

angegebenen Tabellen durch Hash-Joins verknüpft werden. Da keine Indizes verfügbar sind, führt der Zugriffsplan beim Join einer Tabelle eine Tabellensuche in der jeweiligen Tabelle aus.

```
******************** PACKAGE ************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:05:00

 Bind Timestamp = 2002-01-04-14.05.00.415403

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = No
 Intra-Partition Parallel = No

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno


Estimated Cost        = 120.518692
Estimated Cardinality = 221.535980

(    6) Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
         |  #Columns = 3
         |  Relation Scan
         |  |  Prefetch: Eligible
         |  Lock Intents
         |  |  Table: Intent Share
         |  |  Row  : Next Key Share
(    6) |  Process Build Table for Hash Join
(    2) Hash Join
         |  Estimated Build Size: 7111
         |  Estimated Probe Size: 9457
(    5)  |  Access Table Name = DOOLE.PROJECT  ID = 2,7
         |  |  #Columns = 2
         |  |  Relation Scan
         |  |  |  Prefetch: Eligible
         |  |  Lock Intents
         |  |  |  Table: Intent Share
         |  |  |  Row  : Next Key Share
(    5)  |  |  Process Build Table for Hash Join
(    3)  |  Hash Join
         |  |  Estimated Build Size: 5737
         |  |  Estimated Probe Size: 6421
(    4)  |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
         |  |  |  #Columns = 3
         |  |  |  Relation Scan
         |  |  |  |  Prefetch: Eligible
         |  |  |  Lock Intents
         |  |  |  |  Table: Intent Share
         |  |  |  |  Row  : Next Key Share
(    4)  |  |  |  Process Probe Table for Hash Join
```

```
(    1) Return Data to Application
        |  #Columns = 5

End of section


Optimizer Plan:

               RETURN
               (   1)
                 |
               HSJOIN
               (   2)
              /      \
         HSJOIN         TBSCAN
         (   3)         (   6)
        /      \          |
    TBSCAN    TBSCAN   Table:
    (   4)    (   5)   DOOLE
      |         |      EMPLOYEE
 Table:      Table:
 DOOLE       DOOLE
 DEPARTMENT  PROJECT
```

Im ersten Teil des Plans wird auf die Tabellen DEPARTMENT und PROJECT zugegriffen, die über einen Hash-Join verknüpft werden. Das Ergebnis dieses Joins wird mit der Tabelle EMPLOYEE verknüpft. Die Ergebniszeilen werden an die Anwendung zurückgegeben.

Beispiel 2: Zugriffsplan für Einzelpartition mit partitionsinterner Parallelität

In diesem Beispiel wird dieselbe SQL-Anweisung wie im ersten Beispiel gezeigt, jedoch wurde diese Abfrage für eine 4-Wege-SMP-Maschine kompiliert.

```
******************** PACKAGE ************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:12:38

 Bind Timestamp = 2002-01-04-14.12.38.732627

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = No
 Intra-Partition Parallel = Yes (Bind Degree = 4)

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ----------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno

Intra-Partition Parallelism Degree = 4
```

```
Estimated Cost        = 133.934692
Estimated Cardinality = 221.535980

(    2) Process Using 4 Subagents
(    7) |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
        |  |  #Columns = 3
        |  |  Parallel Scan
        |  |  Relation Scan
        |  |  |  Prefetch: Eligible
        |  |  Lock Intents
        |  |  |  Table: Intent Share
        |  |  |  Row  : Next Key Share
(    7) |  |  Process Build Table for Hash Join
(    3) |  Hash Join
        |  |  Estimated Build Size: 7111
        |  |  Estimated Probe Size: 9457
(    6) |  |  Access Table Name = DOOLE.PROJECT  ID = 2,7
        |  |  |  #Columns = 2
        |  |  |  Parallel Scan
        |  |  |  Relation Scan
        |  |  |  |  Prefetch: Eligible
        |  |  |  Lock Intents
        |  |  |  |  Table: Intent Share
        |  |  |  |  Row  : Next Key Share
(    6) |  |  |  Process Build Table for Hash Join
(    4) |  |  Hash Join
        |  |  |  Estimated Build Size: 5737
        |  |  |  Estimated Probe Size: 6421
(    5) |  |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
        |  |  |  |  #Columns = 3
        |  |  |  |  Parallel Scan
        |  |  |  |  Relation Scan
        |  |  |  |  |  Prefetch: Eligible
        |  |  |  |  Lock Intents
        |  |  |  |  |  Table: Intent Share
        |  |  |  |  |  Row  : Next Key Share
(    5) |  |  |  |  Process Probe Table for Hash Join
(    2) |  Insert Into Asynchronous Local Table Queue  ID = q1
(    2) Access Local Table Queue  ID = q1  #Columns = 5
(    1) Return Data to Application
        |  #Columns = 5

End of section


Optimizer Plan:

              RETURN
              (   1)
                |
               LTQ
              (   2)
                |
              HSJOIN
              (   3)
             /      \
       HSJOIN         TBSCAN
       (   4)         (   7)
      /      \          |
   TBSCAN    TBSCAN   Table:
   (   5)    (   6)   DOOLE
     |         |      EMPLOYEE
 Table:      Table:
 DOOLE       DOOLE
 DEPARTMENT  PROJECT
```

Dieser Plan stimmt mit dem Plan des ersten Beispiels weitgehend überein. Der Unterschied besteht in der Erstellung von vier Subagenten beim ersten Starten des Plans und der Tabellenwarteschlange am Ende des Plans, um die Ergebnisse der Arbeit aller Subagenten aufzunehmen, bevor sie an die Anwendung zurückgegeben werden.

Beispiel 3: Zugriffsplan für mehrere Partitionen mit partitionsübergreifender Parallelität

Dieses Beispiel zeigt wiederum dieselbe SQL-Anweisung wie das erste Beispiel, jedoch wurde hier die Abfrage für eine partitionierte Datenbank kompiliert, die aus drei Datenbankpartitionen besteht.

```
******************** PACKAGE ************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:54:57

 Bind Timestamp = 2002-01-04-14.54.57.033666

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = Yes
 Intra-Partition Parallel = No

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION -------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno


Estimated Cost        = 118.483406
Estimated Cardinality = 474.720032

        Coordinator Subsection:
(-----)    Distribute Subsection #2
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #3
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #1
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(    2)    Access Table Queue  ID = q1  #Columns = 5
(    1)    Return Data to Application
           |  #Columns = 5

        Subsection #1:
(    8)    Access Table Queue  ID = q2  #Columns = 2
(    3)    Hash Join
           |  Estimated Build Size: 5737
           |  Estimated Probe Size: 8015
(    6)    |  Access Table Queue  ID = q3  #Columns = 3
```

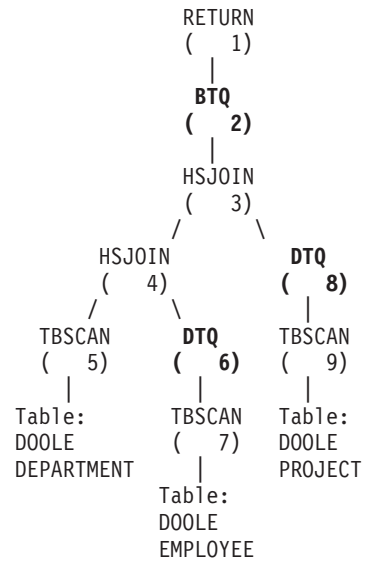
```
(    4)     |  Hash Join
            |  |  Estimated Build Size: 5333
            |  |  Estimated Probe Size: 6421
(    5)     |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
            |  |  |  #Columns = 3
            |  |  |  Relation Scan
            |  |  |  |  Prefetch: Eligible
            |  |  |  Lock Intents
            |  |  |  |  Table: Intent Share
            |  |  |  |  Row  : Next Key Share
(    5)     |  |  |  Process Probe Table for Hash Join
(    2)     Insert Into Asynchronous Table Queue  ID = q1
            |  Broadcast to Coordinator Node
            |  Rows Can Overflow to Temporary Table

        Subsection #2:
(    9)     Access Table Name = DOOLE.PROJECT  ID = 2,7
            |  #Columns = 2
            |  Relation Scan
            |  |  Prefetch: Eligible
            |  Lock Intents
            |  |  Table: Intent Share
            |  |  Row  : Next Key Share
(    9)     |  Insert Into Asynchronous Table Queue  ID = q2
            |  |  Hash to Specific Node
            |  |  Rows Can Overflow to Temporary Tables
(    8)     Insert Into Asynchronous Table Queue Completion  ID = q2

        Subsection #3:
(    7)     Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
            |  #Columns = 3
            |  Relation Scan
            |  |  Prefetch: Eligible
            |  Lock Intents
            |  |  Table: Intent Share
            |  |  Row  : Next Key Share
(    7)     |  Insert Into Asynchronous Table Queue  ID = q3
            |  |  Hash to Specific Node
            |  |  Rows Can Overflow to Temporary Tables
(    6)     Insert Into Asynchronous Table Queue Completion  ID = q3

End of section


Optimizer Plan:
```



Dieser Plan enthält dieselben Bestandteile wie der Plan im ersten Beispiel, aber der Bereich wurde in vier Teilbereiche (Subsection) unterteilt. Die Teilbereiche haben folgende Aufgaben:

- **Coordinator Subsection (Koordinatorteilbereich)**. Dieser Teilbereich koordiniert die anderen Teilbereiche. In diesem Plan sorgt er dafür, dass die anderen Teilbereiche verteilt werden, und er verwendet anschließend eine Tabellenwarteschlange, um die Ergebnisse zu sammeln, die an die Anwendung zurückgegeben werden sollen.
- **Subsection #1**. Dieser Teilbereich durchsucht die Tabellenwarteschlange q2 und verwendet einen Hash-Join, um sie mit den Daten aus Tabellenwarteschlange q3 zu verknüpfen. Ein zweiter Hash-Join fügt anschließend die Daten aus der Tabelle DEPARTMENT hinzu. Die verknüpften Zeilen werden dann über Tabellenwarteschlange q1 an den Koordinatorteilbereich gesendet.
- **Subsection #2**. Dieser Teilbereich durchsucht die Tabelle PROJECT und verteilt sie per Hashverfahren mit den Ergebnissen an einen bestimmten Knoten. Diese Ergebnisse werden von Subsection #1 gelesen.
- **Subsection #3**. Dieser Teilbereich durchsucht die Tabelle EMPLOYEE und verteilt sie per Hashverfahren mit den Ergebnissen an einen bestimmten Knoten. Diese Ergebnisse werden von Subsection #1 gelesen.

Beispiel 4: Zugriffsplan für mehrere Partitionen mit partitionsübergreifender und partitionsinterner Parallelität

In diesem Beispiel wird dieselbe SQL-Anweisung wie im ersten Beispiel gezeigt, jedoch wurde die Abfrage für eine partitionierte Datenbank mit drei Datenbankpartitionen kompiliert, die sich jeweils auf einer 4-Wege-SMP-Maschine befinden.

```
******************** PACKAGE ****************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:58:35

 Bind Timestamp = 2002-01-04-14.58.35.169555

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = Yes
 Intra-Partition Parallel = Yes (Bind Degree = 4)

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ----------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno

Intra-Partition Parallelism Degree = 4

Estimated Cost        = 145.198898
Estimated Cardinality = 474.720032

        Coordinator Subsection:
```

```
(-----)    Distribute Subsection #2
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #3
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #1
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(    2)    Access Table Queue  ID = q1  #Columns = 5
(    1)    Return Data to Application
           |  #Columns = 5

        Subsection #1:
(    3)    Process Using 4 Subagents
(   10)    |  Access Table Queue  ID = q3  #Columns = 2
(    4)    |  Hash Join
           |  |  Estimated Build Size: 5737
           |  |  Estimated Probe Size: 8015
(    7)    |  |  Access Table Queue  ID = q5  #Columns = 3
(    5)    |  |  Hash Join
           |  |  |  Estimated Build Size: 5333
           |  |  |  Estimated Probe Size: 6421
(    6)    |  |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
           |  |  |  |  #Columns = 3
           |  |  |  |  Parallel Scan
           |  |  |  |  Relation Scan
           |  |  |  |  |  Prefetch: Eligible
           |  |  |  |  Lock Intents
           |  |  |  |  |  Table: Intent Share
           |  |  |  |  |  Row  : Next Key Share
(    6)    |  |  |  |  Process Probe Table for Hash Join
(    3)    |  Insert Into Asynchronous Local Table Queue  ID = q2
(    3)    Access Local Table Queue  ID = q2  #Columns = 5
(    2)    Insert Into Asynchronous Table Queue  ID = q1
           |  Broadcast to Coordinator Node
           |  Rows Can Overflow to Temporary Table

        Subsection #2:
(   11)    Process Using 4 Subagents
(   12)    |  Access Table Name = DOOLE.PROJECT  ID = 2,7
           |  |  #Columns = 2
           |  |  Parallel Scan
           |  |  Relation Scan
           |  |  |  Prefetch: Eligible
           |  |  Lock Intents
           |  |  |  Table: Intent Share
           |  |  |  Row  : Next Key Share
(   11)    |  Insert Into Asynchronous Local Table Queue  ID = q4
(   11)    Access Local Table Queue  ID = q4  #Columns = 2
(   10)    Insert Into Asynchronous Table Queue  ID = q3
           |  Hash to Specific Node
           |  Rows Can Overflow to Temporary Tables

        Subsection #3:
(    8)    Process Using 4 Subagents
(    9)    |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
           |  |  #Columns = 3
           |  |  Parallel Scan
           |  |  Relation Scan
           |  |  |  Prefetch: Eligible
           |  |  Lock Intents
           |  |  |  Table: Intent Share
           |  |  |  Row  : Next Key Share
(    8)    |  Insert Into Asynchronous Local Table Queue  ID = q6
(    8)    Access Local Table Queue  ID = q6  #Columns = 3
(    7)    Insert Into Asynchronous Table Queue  ID = q5
```

```
            | Hash to Specific Node
            | Rows Can Overflow to Temporary Tables

End of section


Optimizer Plan:

                RETURN
                (   1)
                  |
                 BTQ
                (   2)
                  |
                 LTQ
                (   3)
                  |
                HSJOIN
                (   4)
               /      \
         HSJOIN        DTQ
         (   5)       (  10)
        /      \        |
    TBSCAN     DTQ     LTQ
    (   6)    (   7)  (  11)
      |         |       |
 Table:        LTQ     TBSCAN
 DOOLE        (   8)   (  12)
 DEPARTMENT     |       |
              TBSCAN   Table:
              (   9)   DOOLE
                |      PROJECT
            Table:
            DOOLE
            EMPLOYEE
```

Dieser Plan ist ähnlich wie der im dritten Beispiel, nur dass mehrere Subagenten jeden Teilbereich ausführen. Außerdem sammelt am Ende eines jeden Teilbereichs eine lokale Tabelle die Ergebnisse von allen Subagenten, bevor die entsprechenden Zeilen in die zweite Tabellenwarteschlange eingefügt werden, die per Hashverfahren an einen bestimmten Knoten gesendet wird.

Joins

Ein *Join* ist der Prozess der Kombination von Informationen aus mindestens zwei Tabellen auf der Grundlage eines gemeinsamen Geltungsbereichs der Informationen. Zeilen aus der einen Tabelle werden mit Zeilen aus einer anderen Tabelle paarweise verbunden, wenn die Informationen in den entsprechenden Zeilen die Joinbedingung erfüllen.

Betrachten Sie zum Beispiel die beiden folgenden Tabellen:

Tabelle1		**Tabelle2**	
PROJ	**PROJ_ID**	**PROJ_ID**	**NAME**
A	1	1	Sam
B	2	3	Joe
C	3	4	Mary
D	4	1	Sue
		2	Mike

Für den Join von Tabelle1 und Tabelle2 an den Stellen, an denen die ID-Spalten die gleichen Werte haben, verwenden Sie die folgende SQL-Anweisung:

```
SELECT PROJ, x.PROJ_ID, NAME
  FROM TABLE1 x, TABLE2 y
  WHERE x.PROJ_ID = y.PROJ_ID
```

Diese Anweisung liefert die folgende Menge von Ergebniszeilen:

PROJ	PROJ_ID	NAME
A	1	Sam
A	1	Sue
B	2	Mike
C	3	Joe
D	4	Mary

Abhängig davon, ob ein Joinvergleichselement vorhanden ist und welche Verarbeitungsaufwände anhand der Tabellen- und Indexstatistiken ermittelt werden, wählt das Optimierungsprogramm eine der folgenden Joinmethoden aus:

- Join mit Verschachtelungsschleife (Nested-Loop Join)
- Mischjoin (Merge Join)
- Hash-Join

Beim Join zweier Tabellen wird die eine Tabelle als äußere Tabelle und die andere als innere Tabelle ausgewählt. Auf die äußere Tabelle wird zuerst zugegriffen, und sie wird nur einmal durchsucht. Ob die innere Tabelle mehrere Male durchsucht wird, hängt von der Art des Joins und den vorhandenen Indizes ab. Auch wenn in einer Abfrage mehr als zwei Tabellen verknüpft werden, verknüpft das Optimierungsprogramm jeweils nur zwei Tabellen gleichzeitig. Falls erforderlich, werden temporäre Tabellen zum Speichern von Zwischenergebnissen erstellt.

Sie können explizite Joinoperatoren wie INNER oder LEFT OUTER JOIN angeben, um festzulegen, wie Tabellen im Join verwendet werden. Bevor Sie jedoch eine Abfrage auf diese Art ändern, sollten Sie das Optimierungsprogramm ermitteln lassen, wie die Tabellen zu verknüpfen sind. Anschließend analysieren Sie die Abfrageleistung und entscheiden, ob Joinoperatoren hinzugefügt werden sollten.

Auswirkung von Datenbankpartitionsgruppen auf die Abfrageoptimierung

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

In einer Umgebung mit partitionierten Datenbanken wird durch das Verteilen von Daten auf mehrere Datenbankpartitionen, abhängig von der Größe der Tabelle, die geschätzte Dauer (bzw. der Aufwand) zur Ausführung einer Abfrage verringert.

Die Anzahl der Tabellen, die Größe der Tabellen, die Speicherposition der Daten in diesen Tabellen und die Art der Abfrage (d. h., ob ein Join erforderlich ist) wirken sich alle auf den Aufwand für die Abfrage aus.

Joinstrategien in partitionierten Datenbanken

In mancher Hinsicht unterscheiden sich die Joinstrategien in einer Umgebung mit partitionierten Datenbanken von denen in einer Umgebung mit nicht partitionierten Datenbanken. Zur Verbesserung der Leistung können zusätzliche Techniken auf die Standardjoinmethoden angewandt werden.

Ein Aspekt bei Tabellen, die häufig an Joins in einer Umgebung mit partitionierten Datenbanken beteiligt sind, ist die Tabellenkollokation. Die Tabellenkollokation stellt eine Methode in einer Umgebung mit partitionierten Datenbanken dar, mit der Daten aus einer Tabelle mithilfe eines gleichen Verteilungsschlüssels zusammen mit den Daten aus einer anderen Tabelle in derselben Datenbankpartition angeordnet werden. Wenn die Daten einmal in dieser Weise zusammengefasst sind, können zu verknüpfende Daten an einer Abfrage beteiligt sein, ohne im Rahmen der Aktivitäten für die Abfrage von einer Datenbankpartition in eine andere übertragen werden zu müssen. Nur die Ergebnismenge eines Joins wird an den Koordinatorknoten übergeben.

Tabellenwarteschlangen

In den Beschreibungen von Jointechniken in einer Umgebung mit partitionierten Datenbanken wird die folgende Terminologie verwendet:

- Tabellenwarteschlange (wird manchmal auch als *TQ* bezeichnet)

  Ein Mechanismus zur Übertragung von Zeilen zwischen Datenbankpartitionen oder auch zwischen Prozessoren in einer Datenbank mit einer Einzelpartition.
- Gezielt übertragene Tabellenwarteschlange (wird manchmal auch als *DTQ* bezeichnet)

  Eine Tabellenwarteschlange, in der Zeilen durch ein Hashverfahren gezielt an eine der empfangenden Datenbankpartitionen geleitet werden.
- Broadcast-Tabellenwarteschlange (wird manchmal auch als *BTQ* bezeichnet)

  Eine Tabellenwarteschlange, in der Zeilen ohne Hashverfahren an alle empfangenden Datenbankpartitionen gesendet werden.

Eine Tabellenwarteschlange wird zu folgenden Zwecken verwendet:

- Übertragen von Tabellendaten von einer Datenbankpartition zu einer anderen bei Verwendung partitionsübergreifender Parallelität
- Übertragen von Tabellendaten innerhalb einer Datenbankpartition bei Verwendung partitionsinterner Parallelität
- Übertragen von Tabellendaten innerhalb einer Datenbankpartition bei Verwendung einer einzigen Datenbankpartition

Jede Tabellenwarteschlange überträgt die Daten in eine einzige Richtung. Der Compiler entscheidet, wo Tabellenwarteschlangen erforderlich sind, und nimmt sie in den Plan auf. Bei der Ausführung des Plans werden die Tabellenwarteschlangen durch die Verbindungen zwischen den Datenbankpartitionen initiiert. Die Tabellenwarteschlangen werden am Ende der Prozesse wieder geschlossen.

Es gibt verschiedene Arten von Tabellenwarteschlangen:

- *Asynchrone Tabellenwarteschlangen.* Diese Tabellenwarteschlangen werden als asynchron bezeichnet, weil sie Zeilen vor einer FETCH-Anweisung, die durch eine Anwendung abgesetzt wird, vorablesen. Wenn eine FETCH-Anweisung abgesetzt wird, wird die Zeile aus der Tabellenwarteschlange abgerufen.

  Asynchrone Tabellenwarteschlangen werden verwendet, wenn Sie die Klausel FOR FETCH ONLY in der SELECT-Anweisung verwenden. Wenn Sie Zeilen nur abrufen, ist eine asynchrone Tabellenwarteschlange schneller.
- *Synchrone Tabellenwarteschlangen.* Diese Tabellenwarteschlangen werden als synchron bezeichnet, weil sie für jede FETCH-Anweisung, die durch eine Anwendung abgesetzt wird, eine Zeile lesen. In jeder Datenbankpartition wird der Cursor in die nächste Zeile gesetzt, die aus der jeweiligen Datenbankpartition zu lesen ist.

  Synchrone Tabellenwarteschlangen werden verwendet, wenn Sie die Klausel FOR FETCH ONLY in der SELECT-Anweisung nicht angeben. In einer Umgebung mit partitionierten Datenbanken verwendet der Datenbankmanager synchrone Tabellenwarteschlangen, wenn Zeilen aktualisiert werden.
- *Tabellenwarteschlangen für Mischjoin.*

  Bei dieser Art von Tabellenwarteschlange wird die Reihenfolge gewahrt.
- *Reguläre Tabellenwarteschlangen.*

  Diese Tabellenwarteschlangen sind nicht für Mischjoins geeignet. Bei ihnen bleibt die Reihenfolge nicht erhalten.
- *Empfangende Tabellenwarteschlange (wird manchmal auch als LTQ bezeichnet)*

  Diese Tabellenwarteschlangen werden mit korrelierten Unterabfragen verwendet. Die Korrelationswerte werden an die Unterabfrage übergeben, und die Ergebnisse mithilfe dieser Art von Tabellenwarteschlange an den übergeordneten Abfrageblock zurückgeliefert.

Joinmethoden in Umgebungen mit partitionierten Datenbanken

Die folgenden Abbildungen veranschaulichen Joinmethoden in einer Umgebung mit partitionierten Datenbanken.

Anmerkung: In den Abbildungen beziehen sich q1, q2 und q3 (für „Queue") auf die Tabellenwarteschlangen in den Beispielen. Die Tabellen, auf die zugegriffen wird, sind für den Zweck dieser Beispielszenarios auf zwei Datenbankpartitionen verteilt. Die Pfeile zeigen die Richtung an, in der die Tabellenwarteschlangen übertragen bzw. gesendet werden. Der Koordinatorknoten entspricht der Datenbankpartition 0.

Zusammengefasste Joins

Ein zusammengefasster Join findet lokal in der Datenbankpartition statt, in der sich die Daten befinden. Die Datenbankpartition sendet die Daten an die anderen Datenbankpartitionen, wenn der Join abgeschlossen ist. Damit das Optimierungsprogramm einen zusammengefassten Join in Erwägung ziehen kann, müssen die verknüpften Tabellen durch Kollokation zusammengefasst sein, und alle Paare mit dem entsprechenden Verteilungsschlüssel müssen Teil der Gleichheitsvergleichselemente sein.

Die folgende Abbildung zeigt ein Beispiel.

Anmerkung: Replizierte MQTs erhöhen die Wahrscheinlichkeit zusammengefasster Joins.

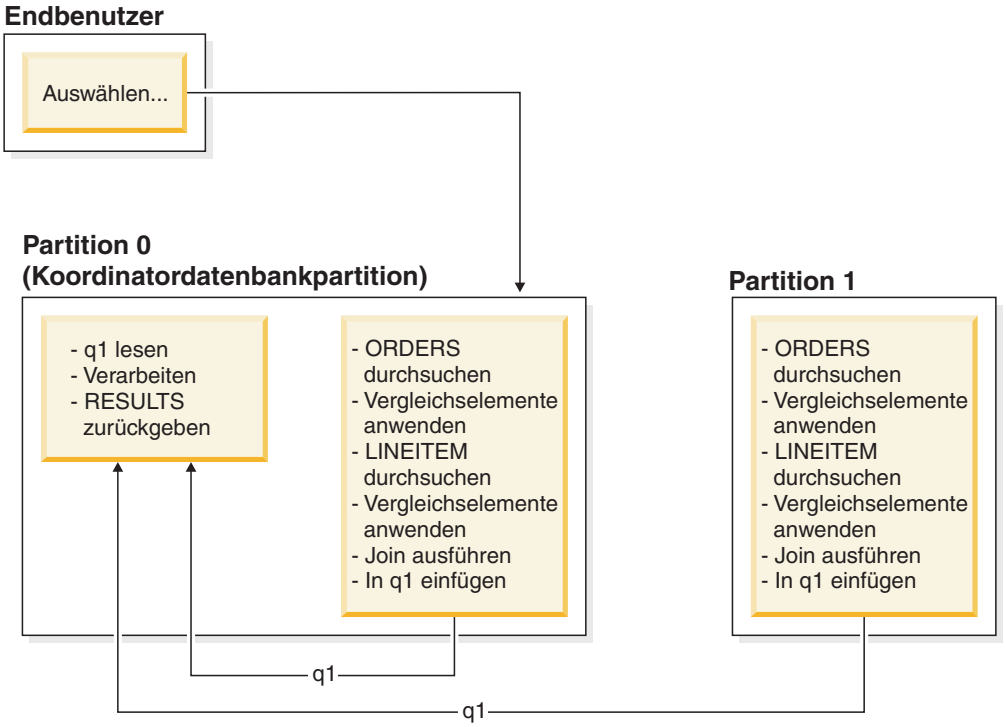


Abbildung 54. Beispiel für einen zusammengefassten Join

Broadcast-Outer-Table-Joins

Broadcast-Outer-Table-Joins (Joins mit rundgesendeter äußerer Tabelle) sind eine parallele Joinstrategie, die angewandt werden kann, wenn es zwischen den zu verknüpfenden Tabellen keine Equijoin-Vergleichselemente gibt. Sie kann außerdem in solchen Fällen verwendet werden, in denen sie die Methode mit dem geringsten Aufwand darstellt. Zum Beispiel könnte ein Broadcast-Outer-Table-Join stattfinden, wenn eine sehr umfangreiche und eine sehr kleine Tabelle am Join beteiligt sind, von denen keine auf die Spalten, auf die die Joinvergleichselemente angewandt werden, verteilt ist. Anstatt beide Tabellen zu teilen, kann es günstiger sein, die kleinere Tabelle an alle Partitionen mit der größeren Tabelle per Broadcast rundzusenden. Die folgenden Abbildungen geben ein Beispiel.

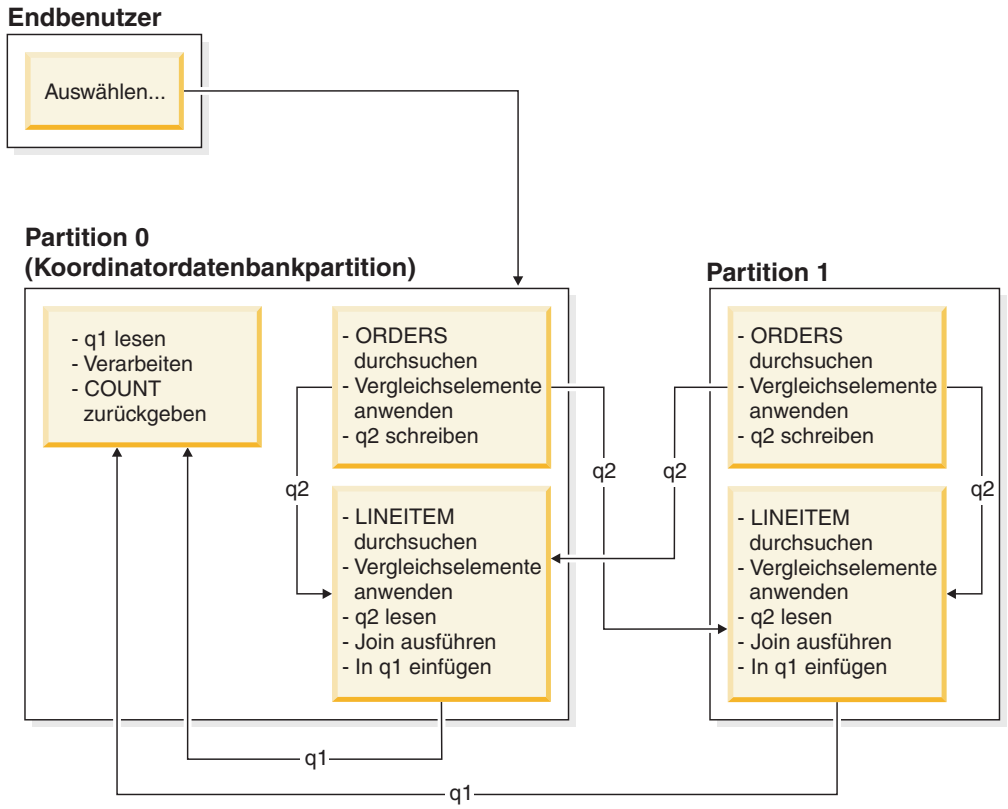


Abbildung 55. Beispiel für einen Broadcast-Outer-Table-Join

Directed-Outer-Table-Joins

Bei der Joinstrategie mit gezielt übertragener äußerer Tabelle (Directed-Outer-Table-Join) wird jede Zeile der äußeren Tabelle an einen Teil der inneren Tabelle entsprechend den Teilungsattributen der inneren Tabelle gesendet. Der Join erfolgt in dieser Datenbankpartition. Die folgende Abbildung zeigt ein Beispiel.

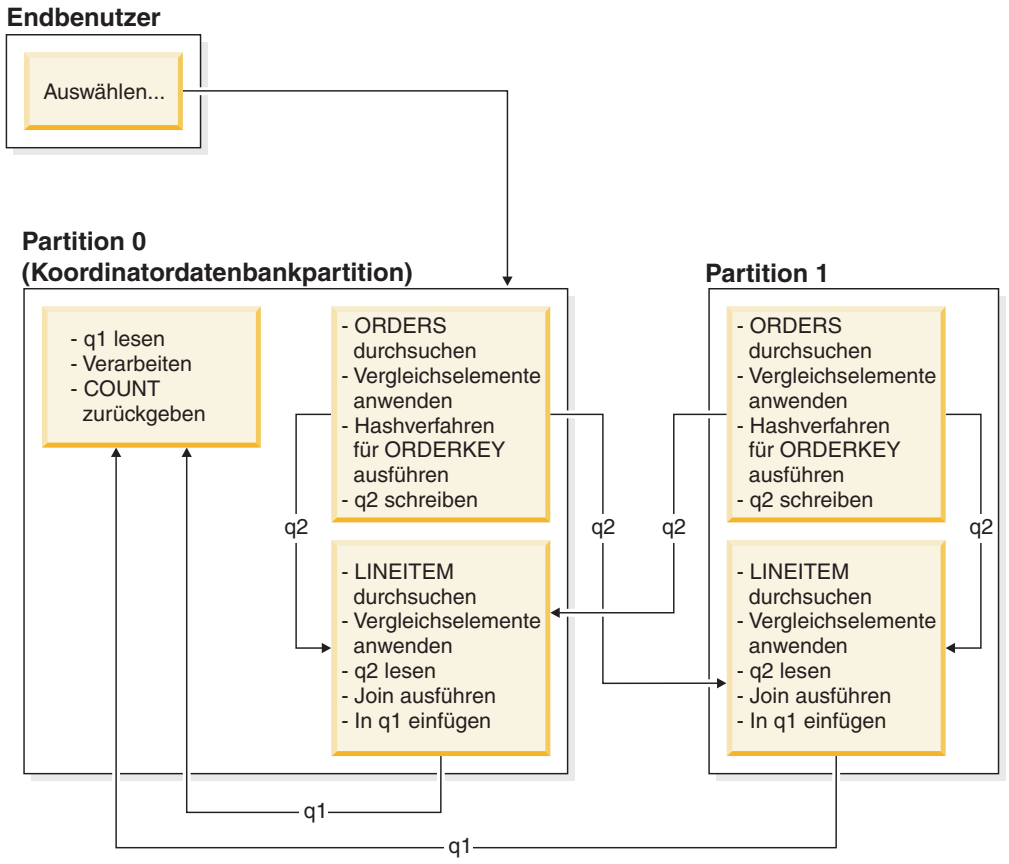


Abbildung 56. Beispiel für einen Directed-Outer-Table-Join

Directed-Inner-Table- und Directed-Outer-Table-Joins

Bei der Joinstrategie mit gezielt übertragener innerer und äußerer Tabelle (Directed-Inner-Table- und Directed-Outer-Table-Join) werden Zeilen sowohl der äußeren als auch der inneren Tabelle gezielt an eine Gruppe von Datenbankpartitionen entsprechend den Werten der Joinspalten übertragen. Der Join erfolgt in diesen Datenbankpartitionen. Die folgende Abbildung zeigt ein Beispiel.

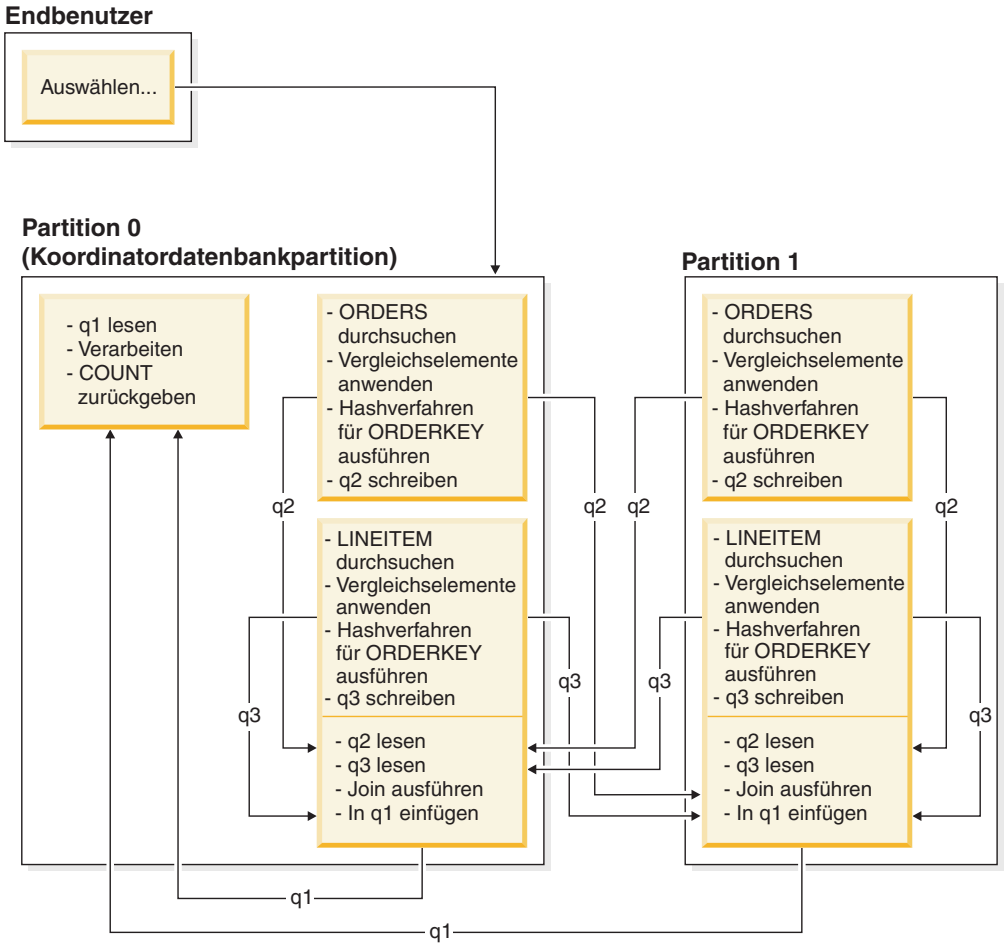


Abbildung 57. Beispiel für einen Directed-Inner-Table- und Directed-Outer-Table-Join

Broadcast-Inner-Table-Joins

Bei der Joinstrategie mit rundgesendeter innerer Tabelle (Broadcast-Inner-Table-Join) wird die innere Tabelle per Broadcast an alle Datenbankpartitionen der äußeren Jointabelle gesendet. Die folgende Abbildung zeigt ein Beispiel.

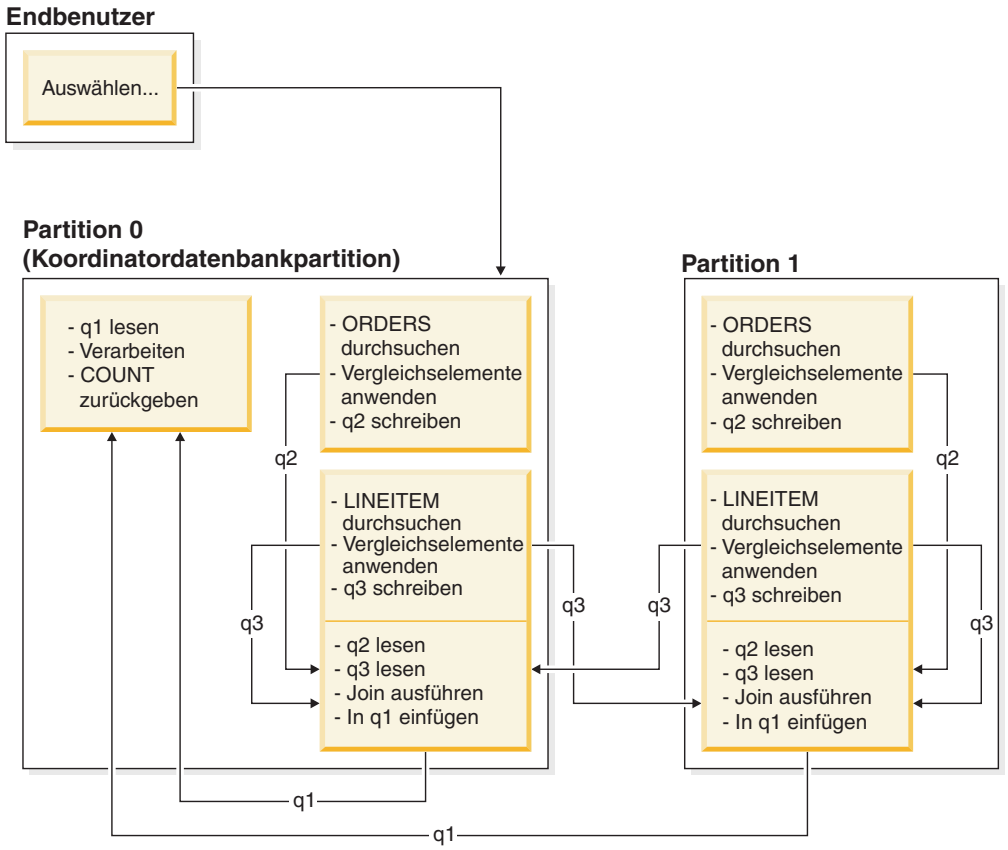


Abbildung 58. Beispiel für einen Broadcast-Inner-Table-Join

Directed-Inner-Table-Joins

Bei der Joinstrategie mit gezielt übertragener innerer Tabelle (Directed-Inner-Table-Join) wird jede Zeile der inneren Tabelle an eine Datenbankpartition der äußeren Tabelle entsprechend den Teilungsattributen der äußeren Tabelle übertragen. Der Join erfolgt in dieser Datenbankpartition. Die folgende Abbildung zeigt ein Beispiel.

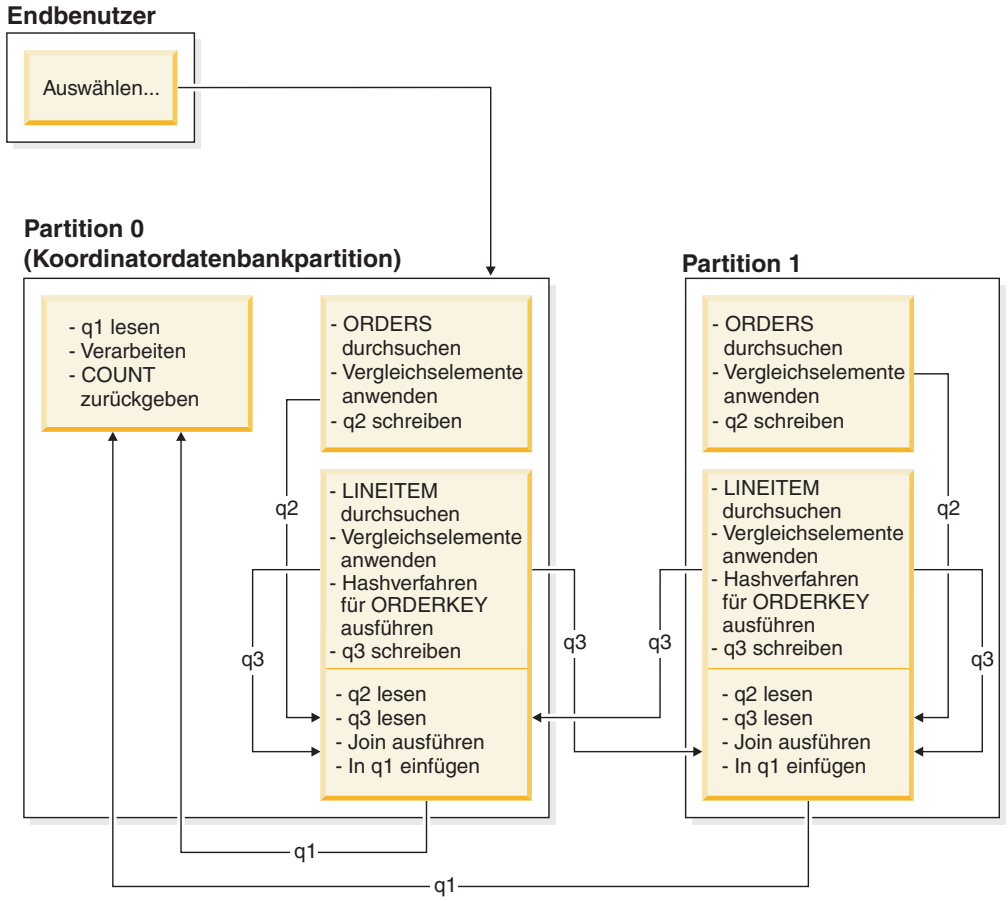


Abbildung 59. Beispiel für einen Directed-Inner-Table-Join

Replizierte MQTs in Umgebungen mit partitionierten Datenbanken

Replizierte MQTs (MQT - Materialized Query Table, gespeicherte Abfragetabelle) verbessern die Leistung häufig ausgeführter Joins in einer partitionierten Datenbankumgebung, indem sie der Datenbank die Möglichkeit geben, vorberechnete Werte der Tabellendaten zu pflegen.

Betrachten Sie ein Beispiel einer Abfrage und einer replizierten MQT. Dazu gelten folgende Annahmen:

- Die Tabelle SALES (Verkauf) befindet sich im Mehrpartitionstabellenbereich REGIONTABLESPACE und wird über die Spalte REGION unterteilt.
- Die Tabellen EMPLOYEE (Mitarbeiter) und DEPARTMENT (Abteilung) sind in einer Datenbankpartitionsgruppe mit Einzelpartition.

Erstellen Sie eine replizierte MQT auf der Basis der Informationen in der Tabelle EMPLOYEE.

```
CREATE TABLE R_EMPLOYEE
  AS (
      SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT
      FROM   EMPLOYEE
     )
DATA INITIALLY DEFERRED REFRESH IMMEDIATE
IN REGIONTABLESPACE
REPLICATED;
```

Führen Sie zur Aktualisierung der replizierten MQT die folgende Anweisung aus:

```
REFRESH TABLE R_EMPLOYEE;
```

Anmerkung: Nach der Verwendung der Anweisung REFRESH sollten Sie RUNSTATS für die replizierte Tabelle in gleicher Weise wie für die anderen Tabellen ausführen.

Im folgenden Beispiel werden der Verkauf nach Mitarbeiter, die Summe für die Abteilung und die Gesamtsumme berechnet:

```
SELECT d.mgrno, e.empno, SUM(s.sales)
FROM   department AS d, employee AS e, sales AS s
WHERE  s.sales_person = e.lastname
   AND e.workdept = d.deptno
GROUP BY ROLLUP(d.mgrno, e.empno)
ORDER BY d.mgrno, e.empno;
```

Anstatt der Tabelle EMPLOYEE, die nur in einer Datenbankpartition ist, verwendet der Datenbankmanager die Tabelle R_EMPLOYEE, die in jede der Datenbankpartitionen repliziert wird, in der die Tabelle SALES gespeichert ist. Der Leistungsvorteil ergibt sich daraus, dass die Mitarbeiterinformationen zur Berechnung des Joins nicht zu jeder Datenbankpartition über das Netzwerk gesendet werden müssen.

Replizierte MQTs in zusammengefassten Joins

Replizierte MQTs können auch bei der Kollokation von Joins helfen. Wenn beispielsweise ein Sternschema eine große Fakttabelle enthält, die sich über zwanzig Knoten erstreckt, sind die Joins zwischen der Fakttabelle und den Dimensionstabellen am effizientesten, wenn diese Tabellen durch Kollokation zusammenfasst werden. Wenn sich alle Tabellen in derselben Datenbankpartitionsgruppe befinden, ist höchstens eine Dimensionstabelle korrekt für einen zusammengefassten Join (d. h. Join von durch Kollokation zusammengefassten Tabellen) partitioniert. Die anderen Dimensionstabellen können nicht in einem zusammengefassten Join verwendet werden, weil die Joinspalten der Fakttabelle nicht dem Verteilungsschlüssel der Fakttabelle entsprechen.

Betrachten Sie zum Beispiel eine Tabelle mit dem Namen FACT (C1, C2, C3, ...), die über Spalte C1 unterteilt ist, und eine Tabelle mit dem Namen DIM1 (C1, dim1a, dim1b, ...), die über C1 unterteilt ist, und eine Tabelle DIM2 (C2, dim2a, dim2b, ...), die über C2 unterteilt ist, usw.

In diesem Fall können Sie erkennen, dass der Join zwischen FACT und DIM1 perfekt ist, da das Vergleichselement DIM1.C1 = FACT.C1 durch Kollokation zusammengefasst vorliegt. Beide Tabellen werden über die Spalte C1 unterteilt.

Der Join mit DIM2 mit dem Vergleichselement WHERE DIM2.C2 = FACT.C2 kann jedoch nicht durch Kollokation zusammengefasst werden, da FACT über die Spalte C1 unterteilt ist und nicht über die Spalte C2. In diesem Fall kann es sinnvoll sein, die Tabelle DIM2 in der Datenbankpartitionsgruppe der Fakttabelle zu replizieren, sodass der Join lokal in jeder Datenbankpartition erfolgen kann.

Anmerkung: Diese Erörterung replizierter MQTs bezieht sich auf datenbankinterne Replikation. Datenbankübergreifende Replikation erfordert Subskriptionen, Steuertabellen und Daten in verschiedenen Datenbanken und auf verschiedenen Betriebssystemen.

Wenn Sie eine replizierte MQT erstellen, kann die Quellentabelle eine Tabelle auf einem einzelnen Knoten oder eine Tabelle auf mehreren Knoten in einer Datenbankpartitionsgruppe sein. In den meisten Fällen ist die replizierte Tabelle klein und kann in einer Datenbankpartitionsgruppe mit einem Knoten untergebracht werden. Sie können die zu replizierende Datenmenge einschränken, indem Sie nur einen Teil der Spalten der Tabelle angeben, die Zeilenzahl mithilfe der verwendeten Vergleichselemente begrenzen oder beide Methoden anwenden. Für das Funktionieren replizierter MQTs ist die Option zur Datenerfassung nicht erforderlich.

Eine replizierte MQT kann auch in einer Datenbankpartitionsgruppe mit mehreren Knoten erstellt werden, sodass Kopien der Quellentabelle in allen Datenbankpartitionen erstellt werden. Joins zwischen einer großen Fakttabelle und den Dimensionstabellen finden in dieser Art von Umgebung mit größerer Wahrscheinlichkeit lokal statt, als wenn die Quellentabelle an alle Datenbankpartitionen per Broadcast übertragen werden muss.

Indizes für replizierte Tabellen werden nicht automatisch erstellt. Sie können Indizes erstellen, die sich von denen für die Quellentabelle unterscheiden. Zur Vermeidung von Verletzungen von Integritätsbedingungen, die für die Quellentabellen nicht vorhanden sind, können Sie jedoch keine eindeutigen Indizes erstellen oder Integritätsbedingungen für die replizierten Tabellen definieren. Integritätsbedingungen sind auch dann nicht zulässig, wenn die gleichen Integritätsbedingungen für die Quellentabelle gelten.

Auf replizierte Tabellen kann in einer Abfrage direkt verwiesen werden, jedoch können Sie nicht das Vergleichselement NODENUMBER() mit einer replizierten Tabelle verwenden, um die Tabellendaten in einer bestimmten Partition abzurufen.

Verwenden Sie die EXPLAIN-Einrichtung, um festzustellen, ob eine replizierte MQT von dem Zugriffsplan für eine Abfrage genutzt wurde. Ob der vom Optimierungsprogramm ausgewählte Zugriffsplan die replizierte MQT verwendet, hängt von den Informationen ab, die verknüpft werden müssen. Die replizierte MQT könnte zum Beispiel dann nicht verwendet werden, wenn das Optimierungsprogramm feststellt, dass es weniger aufwendig wäre, die ursprüngliche Quellentabelle an die anderen Datenbankpartitionen der Datenbankpartitionsgruppe per Broadcast rundzusenden.

Lerneinheit 4. Verbessern eines Zugriffsplans in einer Umgebung mit partitionierten Datenbanken

Sie werden erfahren, wie sich der Zugriffsplan und damit verbundene Fenster für die Standardabfrage ändern, wenn Sie verschiedene Optimierungsaktivitäten ausführen.

Mithilfe von einigen illustrierten Beispielen erfahren Sie, wie der geschätzte Gesamtaufwand für einen Zugriffsplan auch bei einfachen Abfragen verbessert werden kann, indem der Befehl **runstats** verwendet wird und geeignete Indizes hinzugefügt werden.

Mit zunehmender Erfahrung im Umgang mit Visual Explain werden Sie noch weitere Möglichkeiten zur Optimierung von Abfragen entdecken.

Verwenden von Zugriffsplandiagrammen

Anhand der vier Beispiel-EXPLAIN-Momentaufnahmen werden Sie erfahren, dass die Optimierung eine wichtige Rolle bei der Datenbankleistung spielt.

Die mit den EXPLAIN-Momentaufnahmen verknüpften Abfragen haben die Nummern 1 - 4. Jede Abfrage verwendet dieselbe SQL- oder XQuery-Anweisung (wie in Lerneinheit 1 beschrieben):

```
SELECT S.ID,SNAME,O.DEPTNAME,SALARY+COMM
FROM ORG O, STAFF S
WHERE
   O.DEPTNUMB = S.DEPT AND
   S.JOB <> 'Mgr' AND
   S.SALARY+S.COMM > ALL ( SELECT ST.SALARY*.9
                           FROM STAFF ST
                           WHERE ST.JOB='Mgr' )
ORDER BY S.NAME
```

Aber jede Iteration der Abfrage verwendet mehr Optimierungsverfahren als die vorherige Ausführung. Bei Abfrage 1 gab es beispielsweise gar keine Leistungsverbesserung, während es bei Abfrage 4 die meisten Leistungsverbesserungen gab. Die Unterschiede bei den Abfragen werden im Folgenden beschrieben:

Abfrage 1
Ausführen einer Abfrage ohne Indizes und ohne Statistikdaten
Abfrage 2
Erfassen aktueller Statistikdaten für die Tabellen und Indizes in einer Abfrage
Abfrage 3
Erstellen von Indizes für Spalten, die verwendet werden, um Tabellen in einer Abfrage zu verknüpfen
Abfrage 4
Erstellen zusätzlicher Indizes für Tabellenspalten

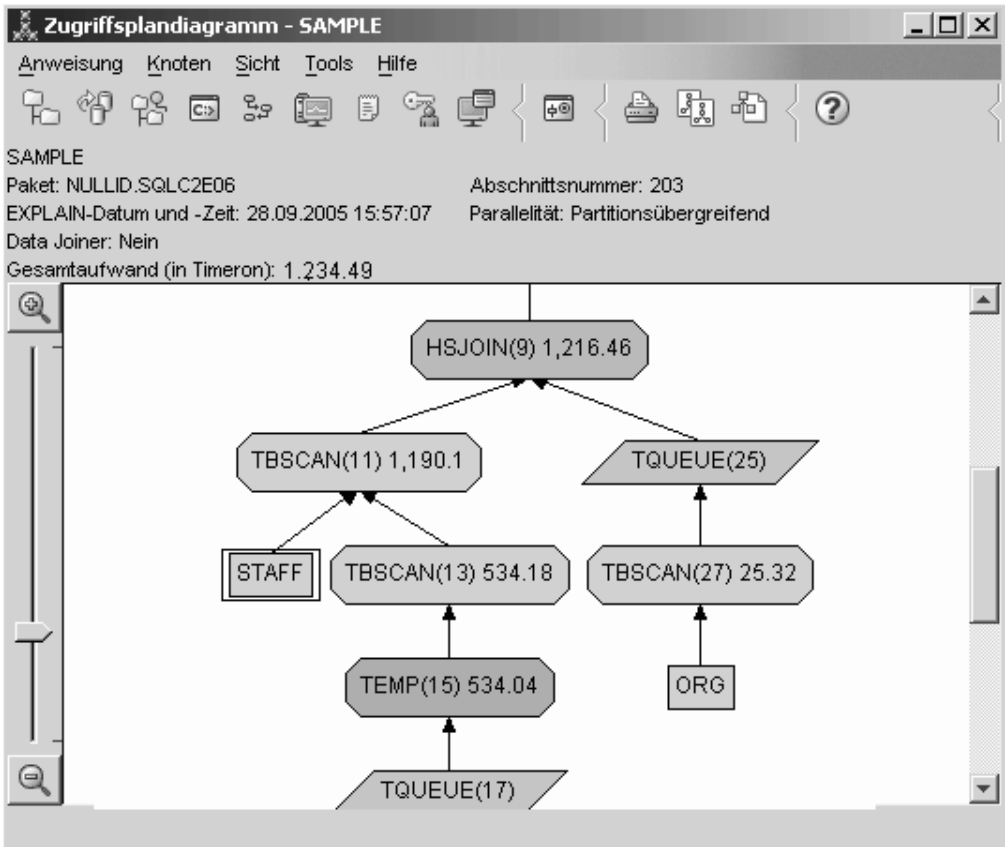
Diese Beispiele wurden auf einer RS/6000 SP-Maschine mit sieben physischen Knoten unter Verwendung der partitionsübergreifenden Parallelität generiert.

Ausführen einer Abfrage ohne Indizes und ohne Statistikdaten in einer Umgebung mit partitionierten Datenbanken

In diesem Beispiel wurde der Zugriffsplan für die SQL-Abfrage ohne Indizes und ohne Statistikdaten erstellt.

Gehen Sie wie folgt vor, um das Zugriffsplandiagramm für diese Abfrage (Abfrage 1) anzuzeigen:

1. Erweitern Sie in der Steuerzentrale die Objektbaumstruktur, bis die Beispieldatenbank SAMPLE angezeigt wird.
2. Klicken Sie die Datenbank mit der rechten Maustaste an, und wählen Sie **Protokoll mit EXPLAIN bearbeiteter Anweisungen anzeigen** im Kontextmenü aus. Das Fenster mit dem Protokoll der mit EXPLAIN bearbeiteten Anweisungen wird geöffnet.
3. Klicken Sie den Eintrag doppelt an, der als Abfrage Nummer 1 angegeben ist (eventuell müssen Sie nach rechts blättern, um die Spalte **Abfragenummer** zu finden).
   Das Fenster **Zugriffsplandiagramm** für die Anweisung wird geöffnet.



Durch Beantwortung der folgenden Fragen erhalten Sie Hinweise dazu, wie Sie die Abfrage verbessern können:

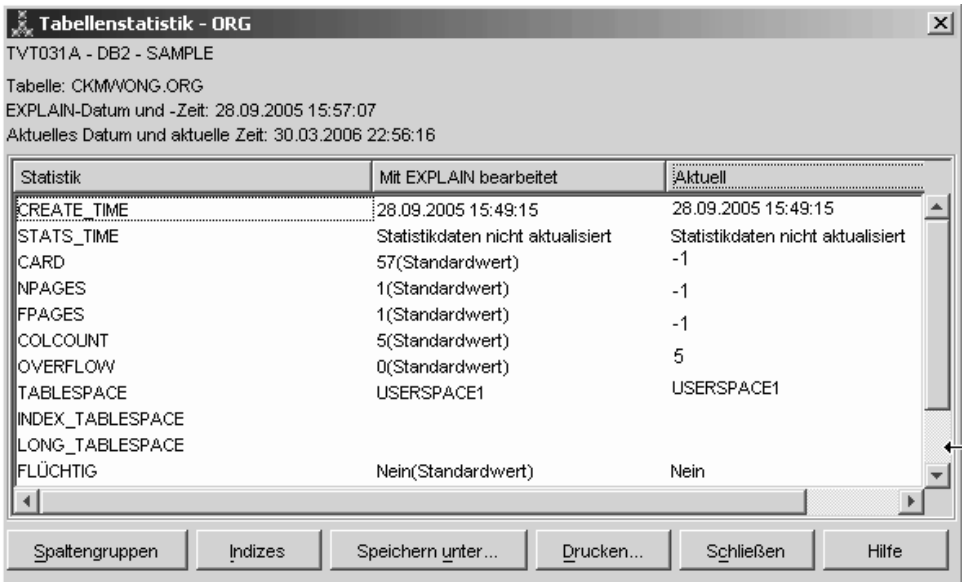
1. Sind für jede Tabelle in der Abfrage aktuelle Statistikdaten vorhanden?

   Wenn Sie überprüfen möchten, ob aktuelle Statistikdaten für jede Tabelle in der Abfrage vorhanden sind, klicken Sie jeden Tabellenknoten im Zugriffsplandiagramm doppelt an. Im entsprechenden Fenster **Tabellenstatistik**, das nun geöffnet wird, enthält die Zeile **STATS_TIME** in der Spalte **Mit EXPLAIN bearbeitet** die Angabe 'Statistikdaten nicht aktualisiert', wenn zu dem Zeitpunkt, als die Momentaufnahme erstellt wurde, keine Statistikdaten erfasst waren.

   Wenn keine aktuellen Statistikdaten vorhanden sind, verwendet das Optimierungsprogramm Standardstatistikdaten, die sich von den tatsächlichen Statistik-

daten unterscheiden können. Standardstatistikdaten werden mit der Angabe '(Standardwert)' in der Spalte **Mit EXPLAIN bearbeitet** im Fenster **Tabellenstatistik** angezeigt.

Laut den Informationen im Fenster **Tabellenstatistik** für die Tabelle **ORG** hat das Optimierungsprogramm Standardstatistikdaten verwendet (wie neben den mit EXPLAIN bearbeiteten Werten angegeben). Es wurden Standardstatistikdaten verwendet, da tatsächliche Statistikdaten nicht zur Verfügung standen, als die Momentaufnahme erstellt wurde (wie in der Zeile **STATS_TIME** angegeben).



2. Verwendet dieser Zugriffsplan die effektivsten Methoden für den Datenzugriff?

   Dieser Zugriffsplan enthält Tabellensuchen anstelle von Indexsuchen. Eine Tabellensuche wird als Achteck angezeigt und mit TBSCAN gekennzeichnet. Wenn Indexsuchen verwendet wurden, werden sie als Rauten und mit Bezeichnung angezeigt. IXSCAN. Die Verwendung eines Indexes, der für eine Tabelle erstellt wurde, ist nicht so aufwendig wie eine Tabellensuche, wenn kleine Datenmengen extrahiert werden.

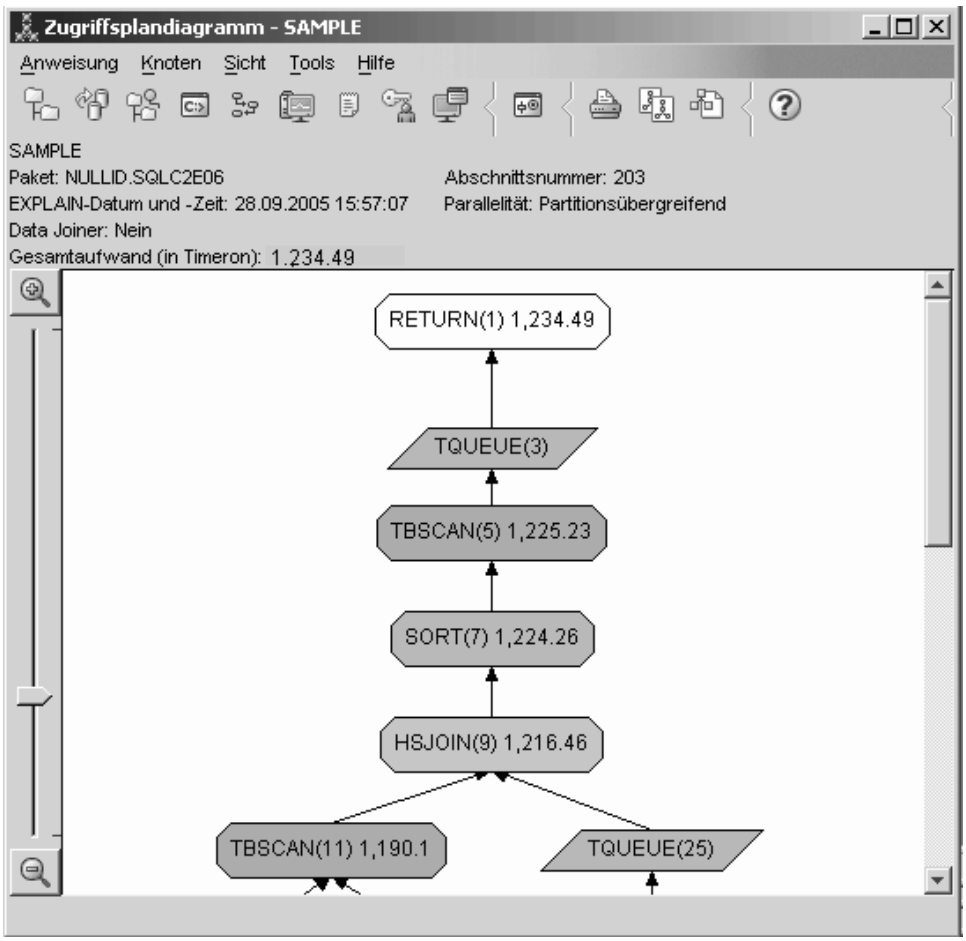
3. Wie effektiv ist dieser Plan?

   Sie können die Effektivität eines Zugriffsplans nur feststellen, wenn er auf tatsächlichen Statistikdaten basiert. Da das Optimierungsprogramm im Zugriffsplan Standardstatistikdaten verwendet hat, können Sie die Effektivität des Plans nicht feststellen.

   Sie sollten sich grundsätzlich den gesamten geschätzten Aufwand für den Zugriffsplan notieren, um ihn später mit überarbeiteten Zugriffsplänen vergleichen zu können. Der in jedem Knoten aufgeführte Aufwand ist kumulativ, vom ersten Schritt Ihrer Abfrage bis einschließlich des Knotens.

   Anmerkung: Für partitionierte Datenbanken ist dies der kumulative Aufwand für den Knoten, der die meisten Ressourcen verwendet.

   Im Fenster **Zugriffsplandiagramm** beträgt der Gesamtaufwand etwa 1.234 Timerons, wie unter **RETURN (1)** im oberen Bereich des Diagramms angezeigt. Der gesamte geschätzte Aufwand wird ebenfalls im oberen Bereich des Fensters angezeigt.



Weitere Schritte

Ausblick auf Abfrage 2.

Abfrage 2 bezieht sich auf einen Zugriffsplan für die Standardabfrage, nachdem der Befehl runstats ausgeführt wurde. Wenn Sie den Befehl runstats verwenden, werden dem Optimierungsprogramm die aktuellen Statistikdaten für alle Tabellen zur Verfügung gestellt, auf die die Abfrage zugreift.

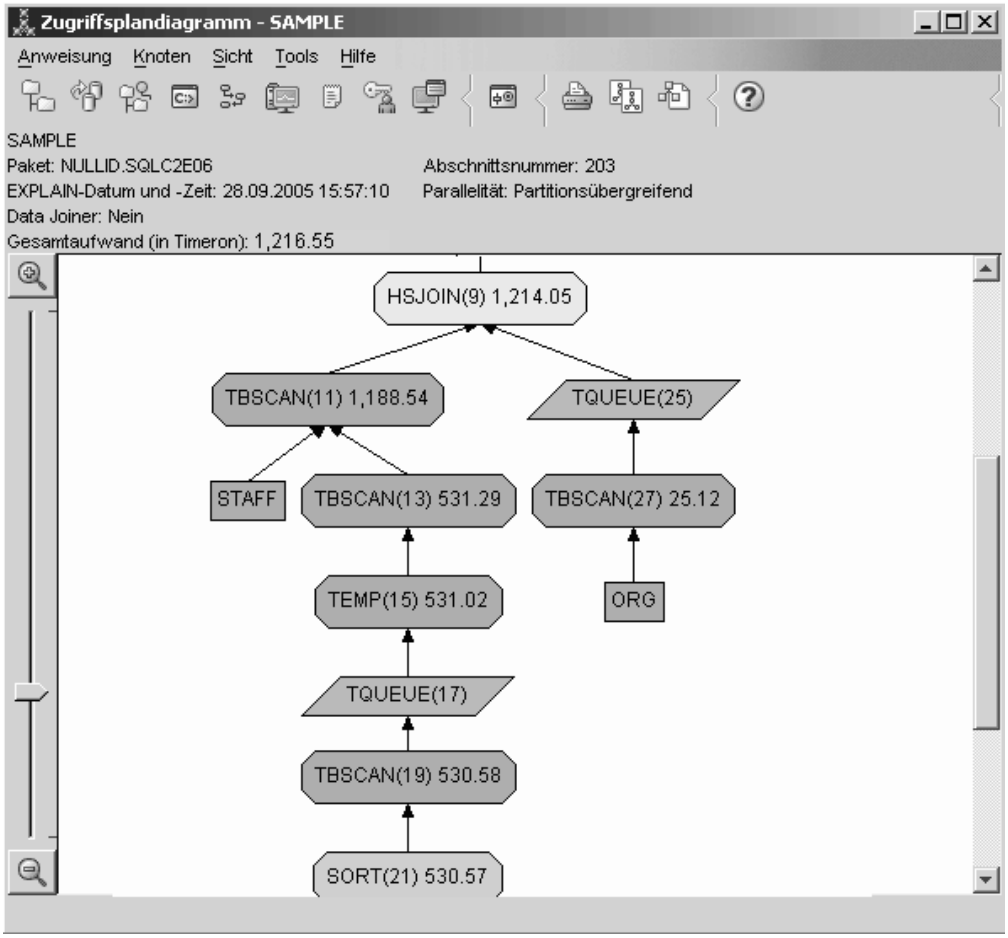
Erfassen aktueller Statistikdaten für die Tabellen und Indizes mithilfe des Befehls 'runstats' in einer Umgebung mit partitionierten Datenbanken

Dieses Beispiel baut auf dem Zugriffsplan auf, der in Abfrage 1 beschrieben wurde, indem aktuelle Statistikdaten mit dem Befehl runstats erfasst werden.

Es wird dringend empfohlen, die aktuellen Statistikdaten für Tabellen und Indizes mit dem Befehl runstats zu erfassen, insbesondere, wenn seit der letzten Ausführung des Befehls runstats umfangreiche Aktualisierungen vorgenommen oder neue Indizes erstellt wurden. Dadurch werden dem Optimierungsprogramm die genauesten Informationen zur Verfügung gestellt, anhand deren es den besten Zugriffsplan ermitteln kann. Wenn aktuelle Statistikdaten nicht verfügbar sind, kann das Optimierungsprogramm einen nicht effizienten Zugriffsplan auswählen, der auf ungenauen Standardstatistikdaten basiert.

Stellen Sie sicher, dass runstats *nach* der Aktualisierung der Tabelle ausgeführt wird. Andernfalls hält das Optimierungsprogramm die Tabelle möglicherweise für leer. Dieses Problem tritt auf, wenn die Kardinalität im Fenster **Operatordetails** gleich null ist. In diesem Fall beenden Sie Ihre Tabellenaktualisierungen, führen den Befehl runstats erneut aus und erstellen die EXPLAIN-Momentaufnahme für betroffene Tabellen erneut.

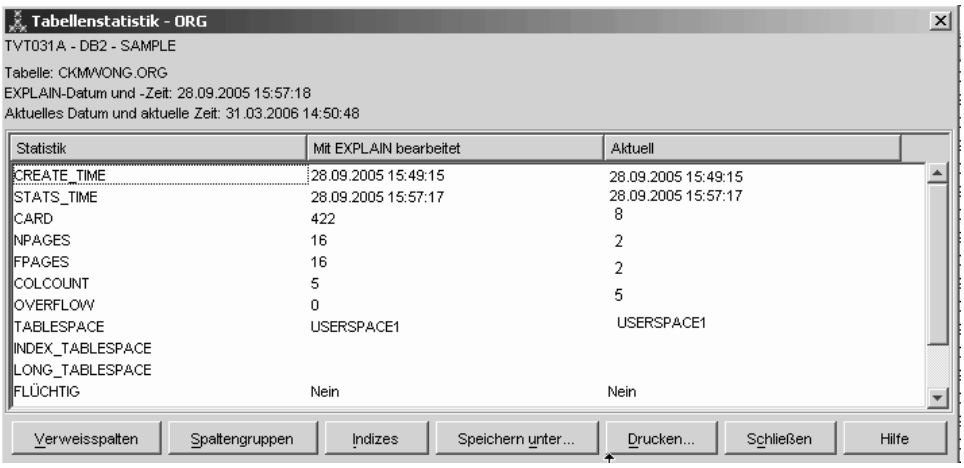
Wenn Sie das Zugriffsplandiagramm für diese Abfrage (Abfrage 2) anzeigen möchten, klicken Sie im Fenster mit dem Protokoll der mit EXPLAIN bearbeiteten Anweisungen den Eintrag doppelt an, der als Abfrage Nummer 2 angegeben ist. Das Fenster **Zugriffsplandiagramm** für diese Ausführung der Anweisung wird geöffnet.



Durch Beantwortung der folgenden Fragen erhalten Sie Hinweise dazu, wie Sie die Abfrage verbessern können.

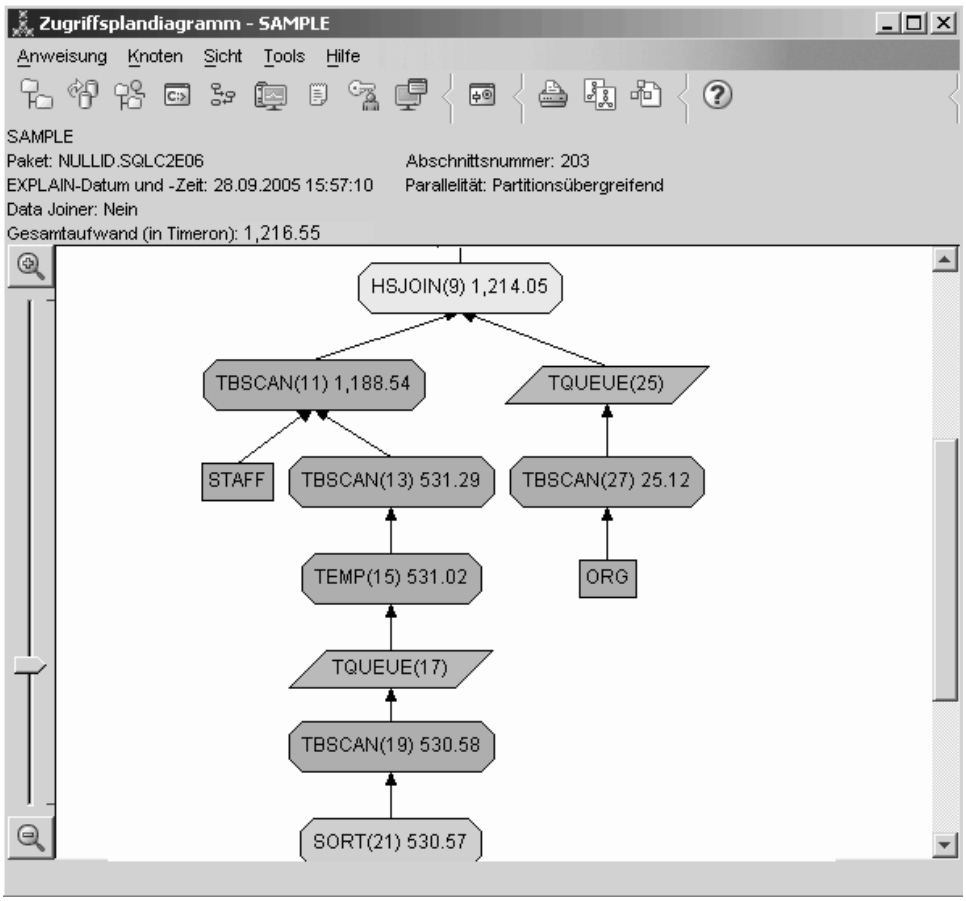
1. Sind für jede Tabelle in der Abfrage aktuelle Statistikdaten vorhanden?

   Das Fenster **Tabellenstatistik** für die Tabelle ORG zeigt an, dass das Optimierungsprogramm tatsächliche Statistikdaten verwendet hat (der Wert STATS_TIME stellt die Zeit dar, zu der die Statistikdaten tatsächlich erfasst wurden). Die Genauigkeit der Statistikdaten hängt davon ab, ob der Inhalt der Tabellen seit der Ausführung des Befehls runstats erheblich geändert wurde.

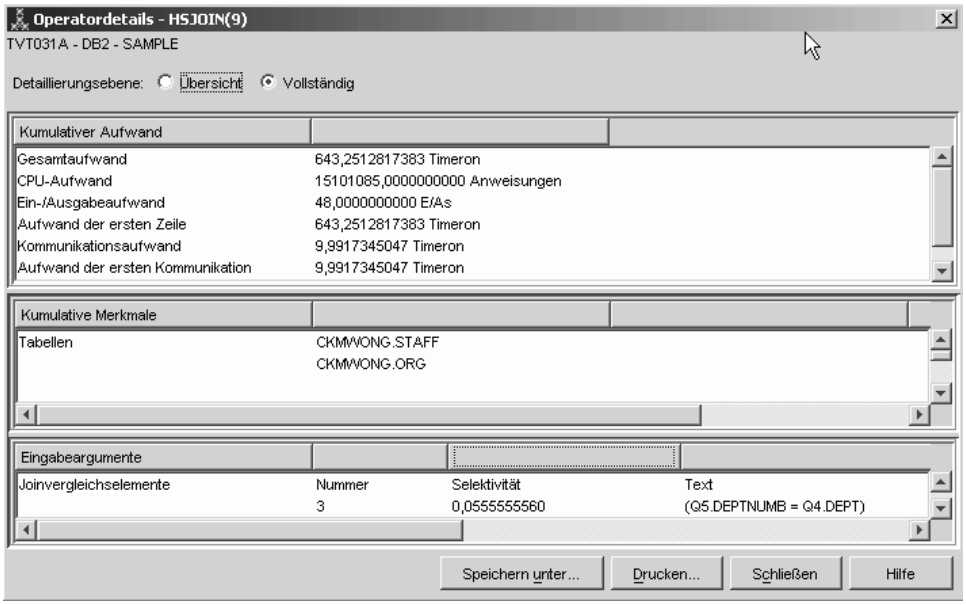


2. Verwendet dieser Zugriffsplan die effektivsten Methoden für den Datenzugriff?

   Wie in Abfrage 1 verwendet der Zugriffsplan in Abfrage 2 Tabellensuchen (Operator TBSCAN), nicht Indexsuchen (IXSCAN). Obwohl aktuelle Statistikdaten vorhanden sind, wurde keine Indexsuche vorgenommen, da es keine Indizes für die Spalten gibt, die von der Abfrage verwendet wurden. Eine Methode zur Verbesserung der Abfrage wäre die Bereitstellung von Indizes für das Optimierungsprogramm, die für Spalten vorhanden sind, die zum Verknüpfen von Tabellen verwendet werden (also für Spalten, die in JoinVergleichselementen verwendet werden). In diesem Beispiel ist dies der erste Mischsuchjoin: HSJOIN(9).



Sehen Sie sich im Fenster für den Operator HSJOIN(9) den Abschnitt **Joinvergleichselemente** unter **Eingabeargumente** an. Die in diesem Join verwendeten Spalten werden in der Spalte **Text** aufgelistet. In diesem Beispiel lauten diese Spalten DEPTNUMB und DEPT.



3. Wie effektiv ist dieser Zugriffsplan?

   Zugriffspläne, die auf aktuellen Statistikdaten basieren, produzieren immer einen realistischen geschätzten Aufwand (gemessen in Timerons). Da der geschätzte Aufwand in Abfrage 1 auf Standardstatistikdaten basierte, kann Aufwand der beiden Zugriffsplandiagramme nicht verglichen werden, um festzustellen, welcher effektiver ist. Es spielt keine Rolle, ob der Aufwand höher oder niedriger ist. Sie müssen den Aufwand von Zugriffsplänen vergleichen, die auf tatsächlichen Statistikdaten basieren, um ein gültiges Maß für die Effektivität zu erhalten.

Weitere Schritte

Ausblick auf Abfrage 3.

Abfrage 3 bezieht sich darauf, welche Auswirkungen das Hinzufügen von Indizes für die Spalten DEPTNUMB und DEPT hat. Wenn Sie Indizes für die Spalten hinzufügen, die in Joinvergleichselementen verwendet werden, kann dies die Leistung erhöhen.

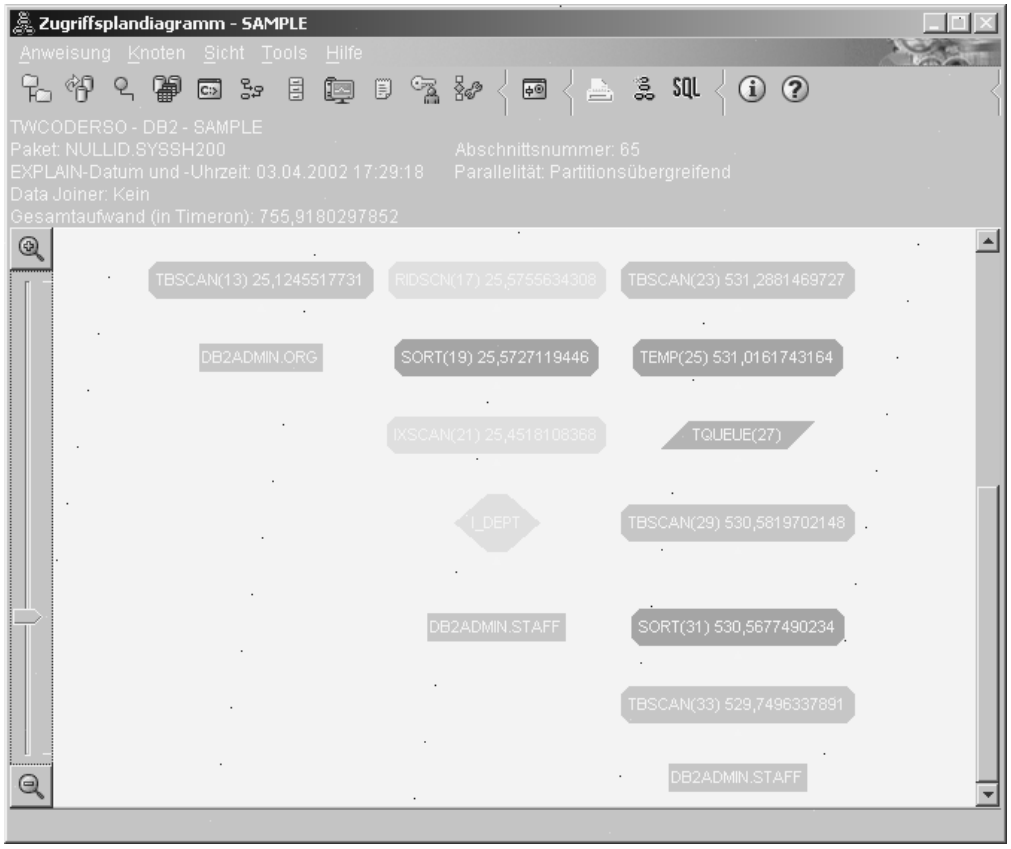
Erstellen von Indizes für Spalten, die verwendet werden, um Tabellen in einer Abfrage zu verknüpfen (Umgebung mit partitionierten Datenbanken)

Dieses Beispiel baut auf dem Zugriffsplan auf, der in Abfrage 2 beschrieben wurde, indem Indizes für die Spalte DEPT in der Tabelle STAFF und für die Spalte DEPTNUMB in der Tabelle ORG erstellt werden.

Anmerkung: Empfohlene Indizes können mithilfe des Designadvisors erstellt werden.

Wenn Sie das Zugriffsplandiagramm für diese Abfrage (Abfrage 3) anzeigen möchten, klicken Sie im Fenster mit dem Protokoll der mit EXPLAIN bearbeiteten Anweisungen den Eintrag doppelt an, der als Abfrage Nummer 3 angegeben ist. Das Fenster **Zugriffsplandiagramm** für diese Ausführung der Anweisung wird geöffnet.

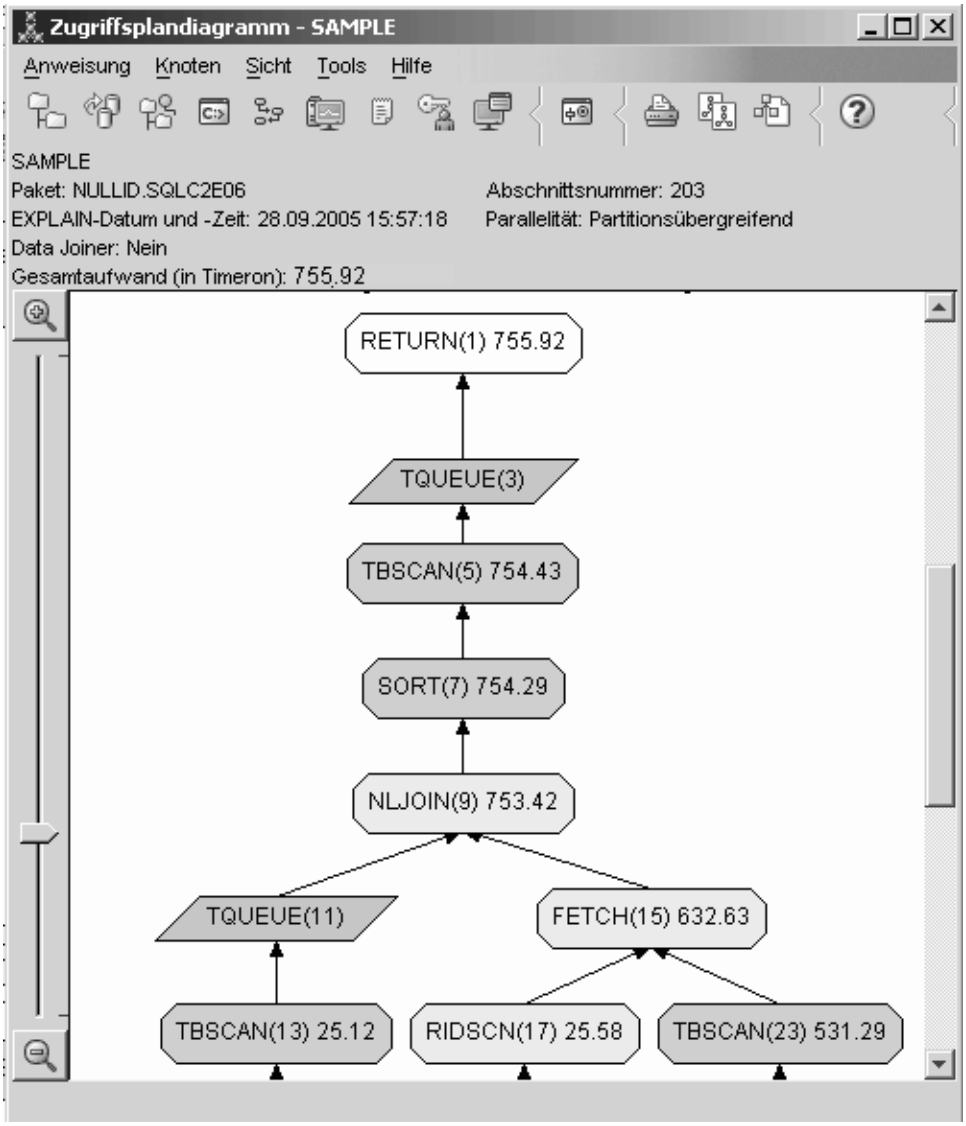
Anmerkung: Obwohl für DEPTNUMB ein Index erstellt wurde, wurde dieser vom Optimierungsprogramm nicht genutzt.



Durch Beantwortung der folgenden Fragen erhalten Sie Hinweise dazu, wie Sie die Abfrage verbessern können.

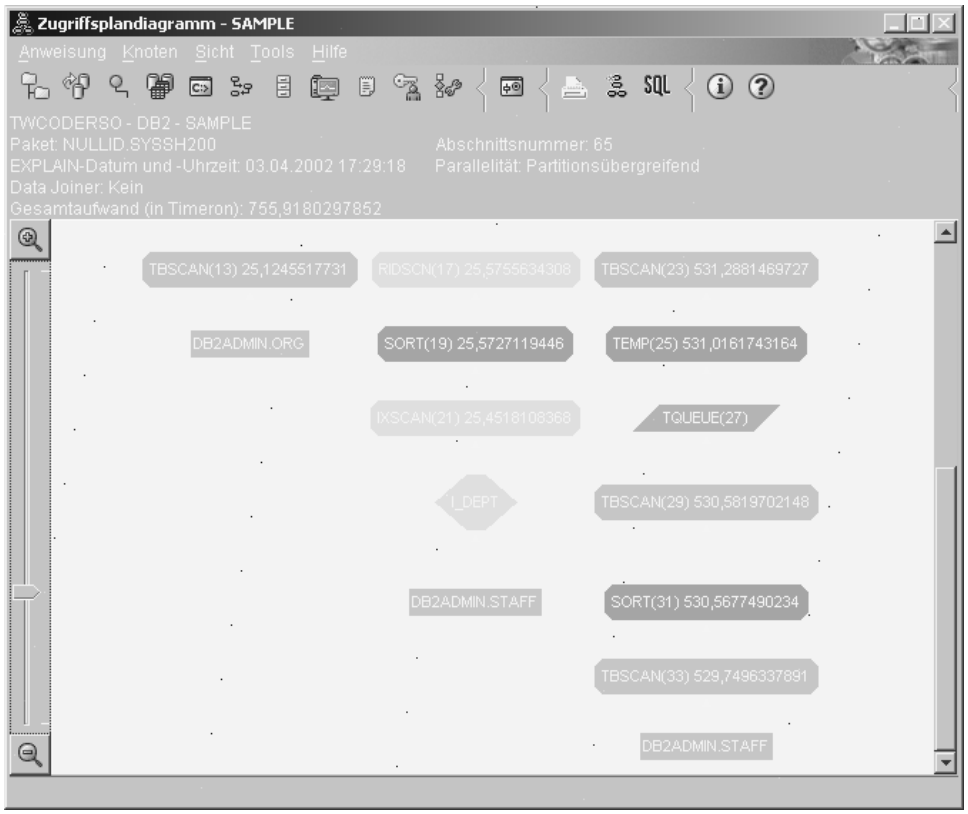
1. Was hat sich im Zugriffsplan mit Indizes geändert?

   Es wurde ein neuer Knoten, **I_DEPT**, in Form einer Raute über der Tabelle STAFF hinzugefügt. Dieser Knoten stellt den Index dar, der für DEPT erstellt wurde. Er zeigt an, dass das Optimierungsprogramm eine Indexsuche anstelle einer Tabellensuche verwendet hat, um festzustellen, welche Zeilen abgerufen werden müssen.

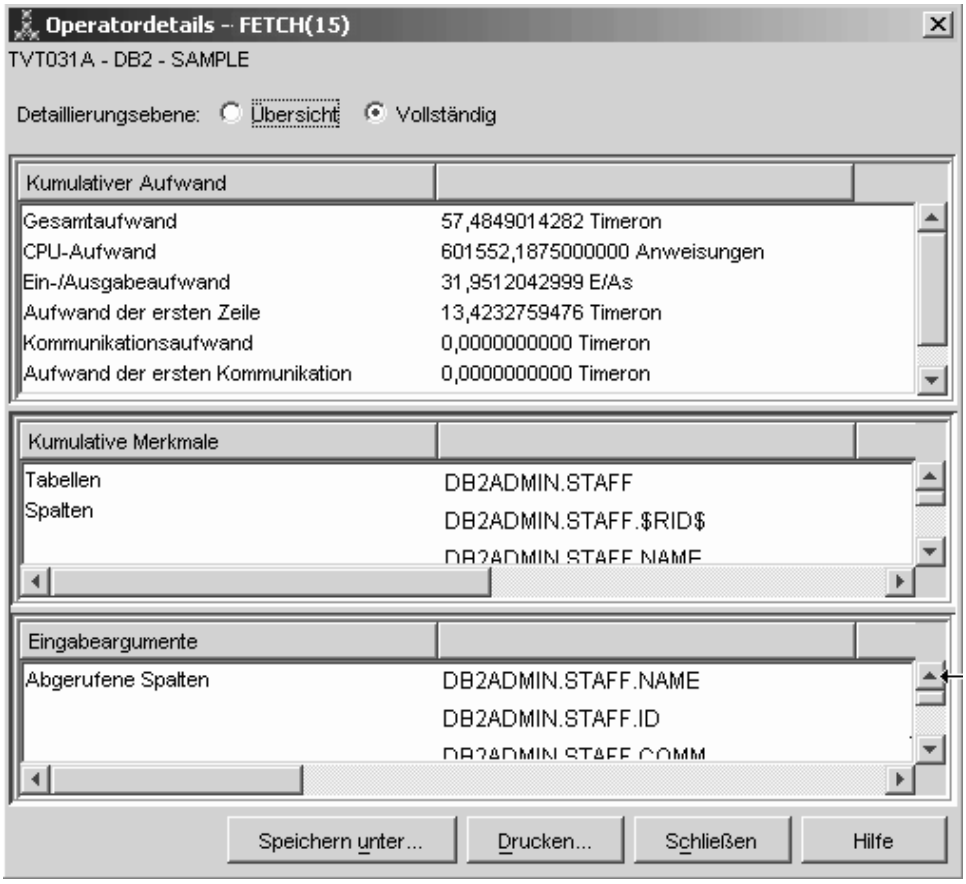


2. Verwendet dieser Zugriffsplan die effektivsten Methoden für den Datenzugriff?

   Der Zugriffsplan für diese Abfrage zeigt an, welche Auswirkungen das Erstellen von Indizes für die Spalte DEPTNUMB der Tabelle ORG hat, was zu FETCH(15) und IXSCAN(21) führte, sowie die Auswirkungen auf die Spalte DEPT der Tabelle STAFF. Da es in Abfrage 2 diesen Index nicht gab, wurde dort eine Tabellensuche verwendet.



   Das Fenster **Operatordetails** für den Operator FETCH(15) zeigt die Spalten an, die in dieser Operation verwendet werden.



Die Kombination von Indexsuche und Abruf wird als weniger aufwendig berechnet als die vollständige Tabellensuche, die in den vorherigen Zugriffsplänen verwendet wurde.

3. Wie effektiv ist dieser Zugriffsplan?

   Dieser Zugriffsplan ist effektiver als der Zugriffsplan aus dem vorherigen Beispiel. Der kumulative Aufwand wurde von etwa 1.214 Timerons in Abfrage 2 auf etwa 756 Timerons in Abfrage 3 reduziert.

Weitere Schritte

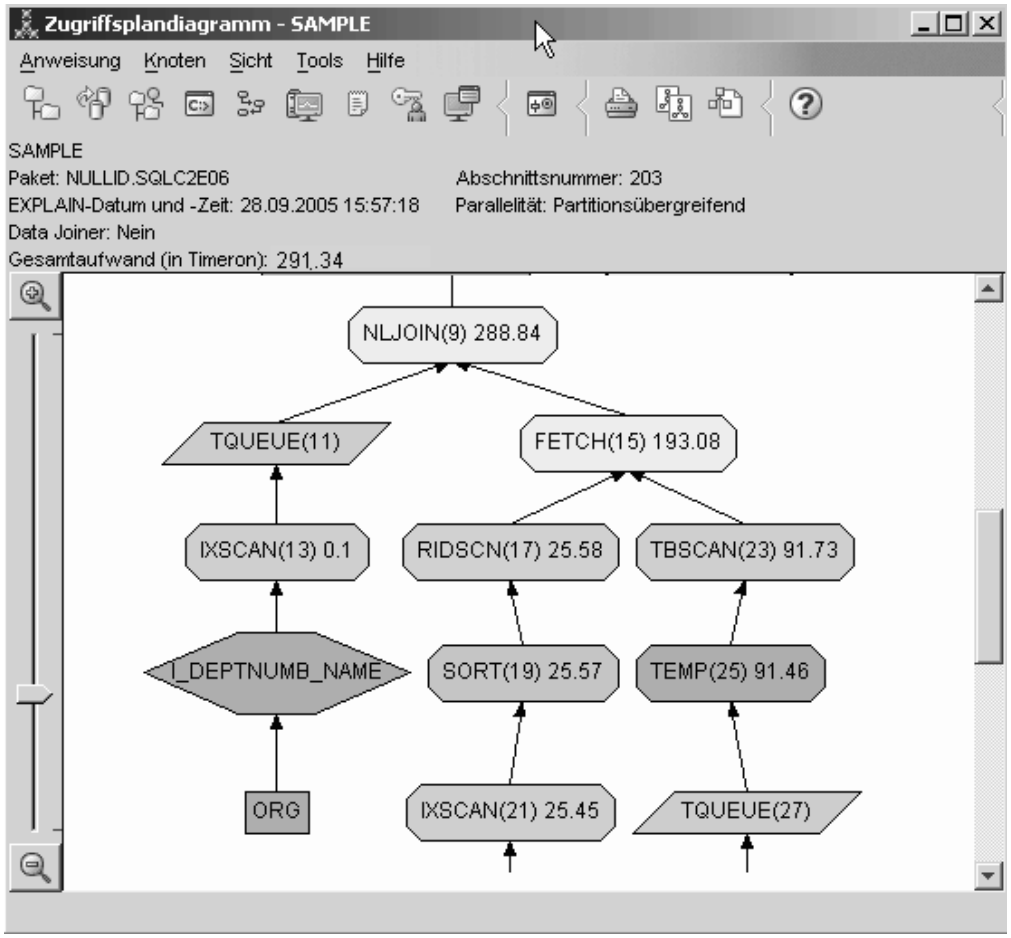
Ausblick auf Abfrage 4.

Abfrage 4 reduziert die FETCH-Operation und die Indexsuche auf eine einzige Indexsuche ohne FETCH-Operation. Durch das Erstellen zusätzlicher Indizes kann der geschätzte Aufwand für den Zugriffsplan reduziert werden.

Erstellen zusätzlicher Indizes für Tabellenspalten in einer Umgebung mit partitionierten Datenbanken

Dieses Beispiel baut auf dem Zugriffsplan auf, der in Abfrage 3 beschrieben wurde, indem ein Index für die Spalte JOB in der Tabelle STAFF erstellt wird und indem DEPTNAME zum vorhandenen Index in der Tabelle ORG hinzugefügt wird. (Das Hinzufügen eines separaten Indexes könnte zu einem zusätzlichen Zugriff führen.)

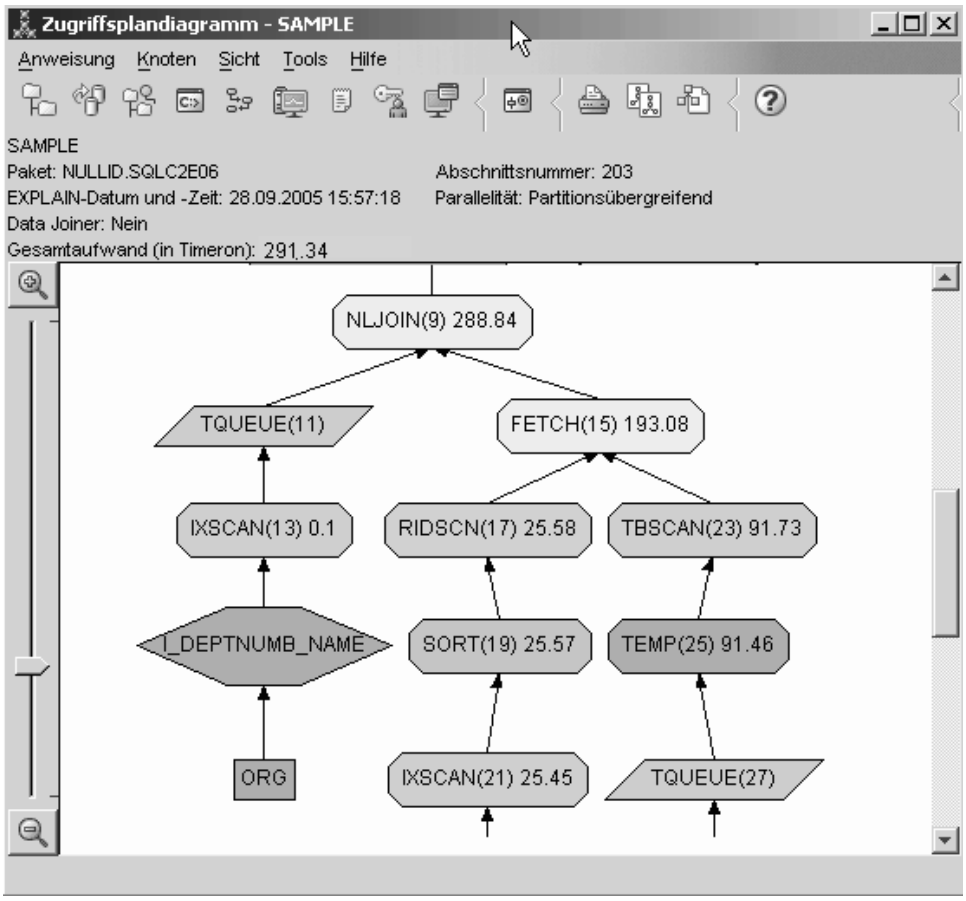
Wenn Sie das Zugriffsplandiagramm für diese Abfrage (Abfrage 4) anzeigen möchten, klicken Sie im Fenster mit dem Protokoll der mit EXPLAIN bearbeiteten Anweisungen den Eintrag doppelt an, der als Abfrage Nummer 4 angegeben ist. Das Fenster **Zugriffsplandiagramm** für diese Ausführung der Anweisung wird geöffnet.



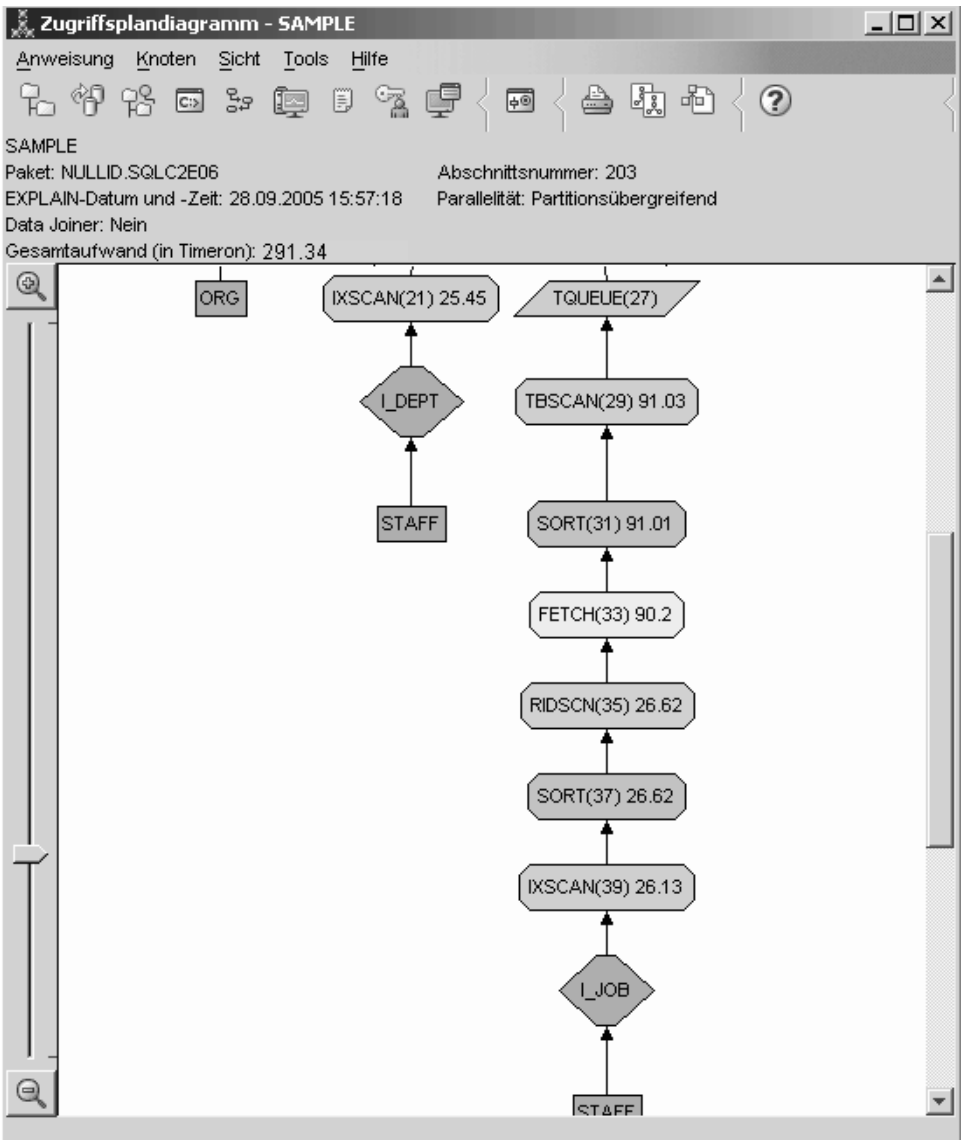
Durch Beantwortung der folgenden Fragen erhalten Sie Hinweise dazu, wie Sie die Abfrage verbessern können.

1. Was ändert sich in diesem Prozessplan aufgrund der Erstellung zusätzlicher Indizes?

   Beachten Sie im mittleren Teil des Zugriffsplandiagramms, dass für die Tabelle ORG die vorherige Tabellensuche in die Indexsuche IXSCAN(7) geändert wurde. Durch das Hinzufügen der Spalte DEPTNAME zum Index für die Tabelle ORG kann das Optimierungsprogramm den Zugriff eingrenzen, der die Tabellensuche beinhaltet.



   Beachten Sie im unteren Teil des Zugriffsplandiagramms, dass für die Tabelle STAFF die vorherige Indexsuche und die FETCH-Operation in die Indexsuche IXSCAN(39) geändert wurden. Durch das Erstellen des Indexes JOB für die Tabelle STAFF kann das Optimierungsprogramm den zusätzlichen Zugriff, der die FETCH-Operation beinhaltet, ausschließen.



2. Wie effektiv ist dieser Zugriffsplan?

   Dieser Zugriffsplan ist nicht so aufwendig wie der Zugriffsplan aus dem vorherigen Beispiel. Der kumulative Aufwand wurde von etwa 753 Timerons in Abfrage 3 auf etwa 289 Timerons in Abfrage 4 reduziert.

Weitere Schritte

Verbesserung der Leistung Ihrer eigenen SQL- oder XQuery-Anweisungen.

Detaillierte Informationen über zusätzliche Schritte zur Verbesserung der Leistung finden Sie in der *DB2-Informationszentrale*. Danach können Sie wieder zu Visual Explain zurückkehren, um die Auswirkungen Ihrer Maßnahmen zu verwerten.

Kapitel 26. Umverteilung von Daten

Die Datenumverteilung ist eine Datenbankverwaltungsoperation, die hauptsächlich zum Versetzen von Daten innerhalb einer Umgebung mit partitionierten Datenbanken ausgeführt werden kann, wenn Partitionen hinzugefügt oder entfernt werden, damit der Speicherbereich gleichmäßig belegt, die Datenbanksystemleistung verbessert oder andere Systemanforderungen erfüllt werden.

Die Datenumverteilung kann mithilfe einer der folgenden Schnittstellen ausgeführt werden:

- Befehl REDISTRIBUTE DATABASE PARTITION GROUP
- Systemdefinierte Prozedur ADMIN_CMD
- Systemdefinierte Prozedur STEPWISE REDISTRIBUTE_DBPG
- API 'sqludrdt'

Für die Datenumverteilung innerhalb einer partitionierten Datenbank gibt es die folgenden Gründe:

- Neuausgleich der Daten, wenn eine neue Datenbankpartition der Datenbankumgebung hinzugefügt bzw. eine vorhandene Datenbankpartition daraus entfernt wird.
- Einleitung einer partitionsübergreifenden Verteilung von benutzerspezifischen Daten.
- Schutz für sensible Daten, indem sie innerhalb einer bestimmten Partition isoliert werden.

Die Datenumverteilung wird ausgeführt, indem eine Verbindung zu einer Datenbank auf der Katalogdatenbankpartition hergestellt und eine Datenumverteilungsoperation für eine bestimmte Partitionsgruppe mithilfe einer der unterstützten Schnittstellen gestartet wird. Für die Umverteilung von Daten ist das Vorhandensein von Verteilungsschlüsseldefinitionen für die Tabellen innerhalb der Partitionsgruppe erforderlich. Der Verteilungsschlüsselwert für eine Datenzeile innerhalb der Tabelle wird verwendet, um festzustellen, auf welcher Partition die Datenzeile gespeichert wird. Ein Verteilungsschlüssel wird beim Erstellen einer Tabelle in einer Datenbankpartitionsgruppe mit mehreren Partitionen automatisch generiert oder kann mithilfe der Anweisungen CREATE TABLE oder ALTER TABLE explizit definiert werden. Standardmäßig werden die Tabellendaten während der Datenumverteilung für jede Tabelle innerhalb einer angegebenen Knotengruppe aufgeteilt und gleichmäßig auf die Datenbankpartitionen umverteilt, eine andere Verteilung, wie z. B. eine ungleichmäßige Verteilung, kann aber erzielt werden, indem eine Eingabeverteilungszuordnung angegeben wird, die die Verteilung der Daten definiert. Verteilungszuordnungen können während einer Datenumverteilungsoperation für den zukünftigen Gebrauch generiert bzw. manuell erstellt werden.

Einschränkungen bei der Datenumverteilung

Die Einschränkungen bei der Datenumverteilung sollten Sie unbedingt beachten, bevor Sie mit der Datenumverteilung fortfahren bzw. wenn Sie Fehler hinsichtlich der Datenumverteilung beheben.

Anmerkung: Dieser Abschnitt enthält Verweise auf einige Schlüsselwortaktualisierungen, die nur zutreffen, wenn DB2 9.5 Fixpack 1 installiert ist. Wenn DB2 9.5 Fix-

pack 1 noch nicht verfügbar oder installiert ist, lesen Sie die Informationen des entsprechenden Abschnitts in DB2 Version 9. Dieser ist in der Informationszentrale von DB2 Version 9 unter folgender Adresse verfügbar:
http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp

Die folgenden Einschränkungen gelten für die Datenumverteilung:

- Die Datenumverteilung auf Partitionen, auf denen Tabellen keine Partitionierungsschlüsseldefinitionen haben, ist nicht zulässig.
- Die Datenumverteilung für eine Datenbankpartitionsgruppe, die Tabellen mit XML-Spalten enthält, wird nicht unterstützt.
- Bei laufender Datenumverteilung gilt Folgendes:
  - Das Starten einer weiteren Umverteilungsoperation für die Datenbankpartitionsgruppe ist nicht zulässig.
  - Das Löschen der Datenbankpartitionsgruppe ist nicht zulässig.
  - Das Ändern der Datenbankpartitionsgruppe ist nicht zulässig.
  - Das Ausführen einer Anweisung ALTER TABLE für eine Tabelle in der Datenbankpartitionsgruppe ist nicht zulässig.
  - Das Erstellen neuer Indizes in der Tabelle, für die die Datenumverteilung durchgeführt wird, ist nicht zulässig.
  - Das Löschen von Indizes, die in der Tabelle definiert sind, für die die Datenumverteilung durchgeführt wird, ist nicht zulässig.
  - Das Abfragen von Daten in der Tabelle, für die die Datenumverteilung durchgeführt wird, ist nicht zulässig.
  - Das Aktualisieren der Tabelle, für die die Datenumverteilung durchgeführt wird, ist nicht zulässig.
- Das Aktualisieren von Tabellen in einer Datenbank, für die eine Datenumverteilung durchgeführt wird, die mit dem Befehl REDISTRIBUTE DATABASE PARTITION GROUP und unter Angabe der Option NOT ROLLFORWARD RECOVERABLE gestartet wurde, ist nicht zulässig. Obwohl die Aktualisierungen durchgeführt werden können, könnten die an den Daten vorgenommenen Änderungen verloren gehen, falls die Datenumverteilung unterbrochen wird. Aus diesem Grund wird hiervon abgeraten.
- Wird der Befehl REDISTRIBUTE DATABASE PARTITION GROUP abgesetzt und ist die Option NOT ROLLFORWARD RECOVERABLE angegeben, gilt Folgendes:
  - Die Datenänderungen, die als Teil der Datenumverteilung vorgenommen wurden, sind nicht aktualisierend wiederherstellbar.
  - Wenn die Datenbank anderweitig wiederherstellbar ist, wird der Tabellenbereich in den Status BACKUP PENDING versetzt, nachdem auf die erste Tabelle innerhalb der Partition zugegriffen wurde. Wenn Sie diesen Status der Tabelle ändern wollen, müssen Sie ein Backup der Tabellenbereichsänderungen durchführen, nachdem die Umverteilungsoperation beendet wurde.
  - Während der Datenumverteilung können die Daten aus den Tabellen der Datenbankpartitionsgruppe, die gerade umverteilt wird, nicht aktualisiert werden; die Daten sind schreibgeschützt. Auf Tabellen, die gerade aktiv umverteilt werden, kann nicht zugegriffen werden.
- Wenn bei typisierten Tabellen (Hierarchietabellen) der Befehl REDISTRIBUTE DATABASE PARTITION GROUP verwendet wird und die Option TABLE mit dem Wert ONLY angegeben ist, dann ist der Tabellenname ausschließlich auf den Namen der Stammtabelle beschränkt. Namen untergeordneter Tabellen können nicht angegeben werden.

- Bei bereichspartitionierten Tabellen ist das Versetzen von Daten zwischen Bereichen einer datenpartitionierten Tabelle nicht zulässig. Die Datenumverteilung wird jedoch zum Versetzen von Daten zwischen Datenbankpartitionen unterstützt.
- Bei partitionierten Tabellen ist eine Umverteilung von Daten nicht zulässig, es sei denn, die folgenden Bedingungen treffen zu:
  - Die partitionierte Tabelle hat den Zugriffsmodus FULL ACCESS (Vollzugriff) in der Katalogtabelle systables.access_mode.
  - Die partitionierte Tabelle hat keine Partitionen, die gerade zugeordnet werden bzw. deren Zuordnung gerade aufgehoben wird.
- Bei replizierten MQTs (MQT - Materialized Query Table, gespeicherte Abfragetabelle) müssen Sie, wenn die Daten in einer Datenbankpartitionsgruppe replizierte MQTs enthalten, diese Tabellen löschen, bevor Sie die Daten umverteilen. Nach der Umverteilung der Daten können Sie die MQTs erneut erstellen.
- Bei Datenbankpartitionen, die MDC-Tabellen (MDC - mehrdimensionales Clustering) enthalten, ist die Verwendung des Befehls REDISTRIBUTE DATABASE PARTITION GROUP nicht zulässig und der Befehl wird nicht erfolgreich ausgeführt, wenn MDC-Tabellen in der Datenbankpartitionsgruppe vorhanden sind, die durch ein Rollout gelöschte Blöcke enthalten, für die eine Bereinigung ansteht. Diese MDC-Tabellen müssen bereinigt werden, bevor die Datenumverteilung wieder aufgenommen bzw. erneut gestartet werden kann.
- Das Löschen von Tabellen, die derzeit in den DB2-Katalogsichten mit dem Status "Umverteilung läuft" markiert sind, ist nicht zulässig. Zum Löschen einer Tabelle in diesem Status führen Sie zunächst das Dienstprogramm REDISTRIBUTE DATABASE PARTITION GROUP mit der Option ABORT oder CONTINUE und einer entsprechenden Tabellenliste aus, sodass die Umverteilung der Tabelle entweder abgeschlossen oder abgebrochen wird.

Feststellen der aktuellen Datenverteilung innerhalb einer Knotengruppe und für eine Tabelle

Das Feststellen der aktuellen Datenverteilung für eine Knotengruppe oder Tabelle kann hilfreich sein, wenn ermittelt werden soll, ob eine Datenumverteilung erforderlich ist. Die Datenverteilung kann zum Erstellen einer angepassten Verteilungszuordnung verwendet werden, mit der angegeben werden kann, wie die Daten verteilt werden sollen.

Wenn eine neue Datenbankpartition zur Datenbankpartitionsgruppe hinzugefügt wird, oder eine vorhandene Datenbankpartition aus einer Datenbankpartitionsgruppe gelöscht wird, sollte eine Datenumverteilung ausgeführt werden, damit die Daten auf allen Datenbankpartitionen gleichmäßig verteilt sind.

Wenn keine Datenbankpartitionen hinzugefügt oder aus einer Datenbankpartitionsgruppe gelöscht wurden, empfiehlt sich eine Datenumverteilung üblicherweise nur dann, wenn die Daten auf den Datenbankpartitionen der Datenbankpartitionsgruppe ungleichmäßig verteilt sind. Beachten Sie, dass in einigen Fällen eine ungleichmäßige Verteilung von Daten durchaus erwünscht sein kann. Wenn sich z. B. einige Datenbankpartitionen auf einer besonders leistungsfähigen Maschine befinden, kann es für diese Datenbankpartitionen vorteilhaft sein, im Vergleich zu anderen Datenbankpartitionen größere Datenvolumina zu enthalten.

Um Informationen zur aktuellen Verteilung der Daten auf Datenbankpartitionen in einer Datenbankpartitionsgruppe abzurufen, führen Sie die folgende Abfrage für die größte Tabelle (oder alternativ eine repräsentative Tabelle) in der Datenbankpartitionsgruppe aus:

```
SELECT DBPARTITIONNUM(spaltenname), COUNT(*) FROM tabellenname
      GROUP BY DBPARTITIONNUM(spaltenname)
      ORDER BY DBPARTITIONNUM(spaltenname) DESC
```

Hier ist 'spaltenname' der Name des Verteilungsschlüssels für die Tabelle 'tabellenname'.

Wenn die Ergebnisse der Abfrage zeigen, dass die Verteilung der Daten auf den Datenbankpartitionen nicht Ihren Wünschen entspricht, führen Sie die folgende Abfrage aus, um die Verteilung von Daten auf Hashpartitionen abzurufen:

```
SELECT PARTITION(spaltenname), COUNT(*) FROM tabellenname
      GROUP BY PARTITION(spaltenname)
      ORDER BY PARTITION(spaltenname) DESC
```

Die Ausgabe dieser Abfrage kann problemlos verwendet werden, um die benötigte Verteilungsdatei zu erstellen, wenn die Option USING DISTFILE des Befehls REDISTRIBUTE DATABASE PARTITION GROUP angegeben ist (eine Beschreibung des Formats von der Verteilungsdatei finden Sie im Abschnitt über den Befehl REDISTRIBUTE DATABASE PARTITION GROUP im Handbuch 'Command Reference').

Wenn die Option USING DISTFILE angegeben ist, verwendet der Befehl REDISTRIBUTE DATABASE PARTITION GROUP die Informationen in der Datei, um eine neue Partitionszuordnung für die Datenbankpartitionsgruppe zu generieren, dies hat eine gleichmäßige Verteilung von Daten auf den Datenbankpartitionen zur Folge.

Wenn eine gleichmäßige Verteilung nicht gewünscht wird, kann der Benutzer seine eigene Zielpartitionszuordnung für die Umverteilungsoperation erstellen, diese kann mithilfe der Option USING TARGETMAP des Befehls REDISTRIBUTE DATABASE PARTITION GROUP angegeben werden.

Nach dieser Untersuchung sind Sie darüber im Bilde, ob Ihre Daten gleichmäßig oder ungleichmäßig verteilt sind und ob eine Datenumverteilung erforderlich ist. Wenn die Daten eine Umverteilung erfordern, können Sie dies während einer Systemwartung unter Verwendung einer der unterstützten Schnittstellen durchführen.

Umverteilen von Daten auf Datenbankpartitionen unter Verwendung des Befehls REDISTRIBUTE DATABASE PARTITION GROUP

Das Umverteilen von Daten kann erfolgreich unter Verwendung des Befehls REDISTRIBUTE DATABASE PARTITION GROUP ausgeführt werden. Dies ist die empfohlene Schnittstelle für das Ausführen einer Datenumverteilung.

Anmerkung: Dieser Abschnitt enthält Verweise auf einige Schlüsselwortaktualisierungen, die nur zutreffen, wenn DB2 9.5 Fixpack 1 installiert ist. Wenn DB2 9.5 Fixpack 1 noch nicht verfügbar oder installiert ist, lesen Sie die Informationen des entsprechenden Abschnitts in DB2 Version 9. Dieser ist in der Informationszentrale von DB2 Version 9 unter folgender Adresse verfügbar:
http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.admin.doc/doc/t0005017.htm

Einschränkungen

- Siehe „Einschränkungen bei der Datenumverteilung" auf Seite 387.

Gehen Sie wie folgt vor, um Daten auf Datenbankpartitionen in einer Datenbankpartitionsgruppe unter Verwendung des Befehls REDISTRIBUTE DATABASE PARTITION GROUP umzuverteilen:

1. Führen Sie ein Backup der Datenbank durch. Siehe den Befehl BACKUP.
2. Stellen Sie eine Verbindung zu der Datenbankpartition her, in der die Systemkatalogtabellen enthalten sind. Siehe den Befehl CONNECT.
3. Setzen Sie den Befehl REDISTRIBUTE DATABASE PARTITION GROUP ab.

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

   Geben Sie die folgenden Argumente an:

   Name einer Datenbankpartitionsgruppe
   Sie müssen die Datenbankpartitionsgruppe angeben, in der die Daten umzuverteilen sind.

   UNIFORM
   OPTIONAL: Gibt an, dass Daten gleichmäßig verteilt sind und gleichmäßig verteilt bleiben sollen. UNIFORM ist die Standardeinstellung, wenn kein Verteilungstyp angegeben ist, daher ist es auch zulässig, diese Option zu übergehen, wenn kein anderer Verteilungstyp angegeben wurde.

   USING DISTFILE *verteilungsdateiname*
   OPTIONAL: Gibt an, dass eine angepasste Verteilung erwünscht ist, und gibt ferner den Dateipfadnamen einer Verteilungsdatei an, in der Daten enthalten sind, die die gewünschte ungleiche Datenverteilung definieren. Der Inhalt dieser Datei wird zum Generieren einer Zielverteilungszuordnung verwendet.

   USING TARGETMAP *zielzuordnungsname*
   OPTIONAL: Gibt an, dass eine Zieldatenumverteilungszuordnung verwendet werden soll, und gibt ferner den Namen der Datei an, in der die Zielumverteilungszuordnung enthalten ist.

   Weitere Einzelheiten enthalten die Informationen zum Befehlszeilendienstprogramm REDISTRIBUTE DATABASE PARTITION GROUP.
4. Ermöglichen Sie dem Befehl eine ununterbrochene Ausführung. Wenn nach Beendigung des Befehls die Datenumverteilung erfolgreich fortgesetzt wurde, führen Sie Folgendes aus:
   - Erstellen Sie ein Backup aller Tabellenbereiche in der Datenbankpartitionsgruppe, die sich im Status "Backup anstehend" befinden. Alternativ kann auch ein Datenbankgesamtbackup durchgeführt werden. HINWEIS: Tabellenbereiche werden nur dann in den Status BACKUP PENDING versetzt, wenn die Datenbank wiederherstellbar ist und die Option NOT ROLLFORWARD RECOVERABLE des Befehls REDISTRIBUTE DATABASE PARTITION GROUP verwendet wird.
   - Erstellen Sie alle replizierten MQTs erneut, die Sie vor der Umverteilung gelöscht haben.
   - Wenn die Option STATISTICS NONE im Befehl REDISTRIBUTE DATABASE PARTITION GROUP angegeben wurde oder die Option NOT ROLLFORWARD RECOVERABLE nicht angegeben wurde (in beiden Fällen werden

keine Statistikdaten während der Datenumverteilung erfasst) und sich Tabellen in der Datenbankpartitionsgruppe befinden, die ein Statistikprofil besitzen, führen Sie nun den Befehl RUNSTATS aus, um die Datenverteilungsstatistiken für den SQL-Compiler und das Optimierungsprogramm zur Auswahl von Datenzugriffsplänen für Abfragen zu erfassen.

- Wenn die Option NOT ROLLFORWARD RECOVERABLE angegeben wurde, löschen Sie die Steuerdateien, die sich in den folgenden Pfaden befinden:
  - Linux- und UNIX-Betriebssysteme: `DIAGPATH/redist/db-name/db-partitionsgruppenname/zeitmarke/`
  - Windows-Betriebssysteme: `DIAGPATH\redist\db-name\db-partitionsgruppenname\zeitmarke\`

Die Datenumverteilung sollte erfolgreich abgeschlossen worden sein, und die Informationen zum Datenumverteilungsprozess stehen in der Umverteilungsprotokolldatei zur Verfügung. Informationen zur verwendeten Verteilungszuordnung befinden sich in den DB2-EXPLAIN-Tabellen.

Umverteilen von Daten in einer Datenbankpartitionsgruppe

Mithilfe des Assistenten **Daten umverteilen** können Sie einen effektiven Umverteilungsplan für Ihre Datenbankpartitionsgruppe erstellen und die Daten umverteilen. Zuerst müssen Sie Ihre Umverteilungsmethode und -strategie auswählen, anschließend können Sie erweiterte Optionen angeben.

Zum Arbeiten mit Datenbankpartitionsgruppen müssen Sie über die Berechtigung *sysadm* oder *dbadm* verfügen.

Gehen Sie wie folgt vor, um Daten in Ihrer Datenbankpartitionsgruppe umzuverteilen:

1. Öffnen Sie den Assistenten **Daten umverteilen**: Erweitern Sie in der Steuerzentrale die Objektbaumstruktur, bis der Ordner **Datenbankpartitionsgruppen** angezeigt wird. Alle vorhandenen Datenbankpartitionsgruppen werden im Inhaltsteilfenster auf der rechten Seite des Fensters angezeigt. Klicken Sie die gewünschte Datenbankpartitionsgruppe mit der rechten Maustaste an, und wählen Sie **Umverteilen** im Kontextmenü aus. Der Assistent **Daten umverteilen** wird geöffnet.

   Sie können den Assistenten 'Daten umverteilen' durch *Hinzufügen von Datenbankpartitionen mit dem Launchpad 'Partitionen hinzufügen'* oder durch *Löschen von Datenbankpartitionen aus der Instanz mit dem Launchpad 'Partitionen löschen'* öffnen.
2. Füllen Sie alle relevanten Seiten des Assistenten aus. Klicken Sie die Verknüpfung zur Assistentenübersicht auf der ersten Seite an, um weitere Informationen zu erhalten. Der Druckknopf **Fertig stellen** ist verfügbar, wenn Sie ausreichend Informationen angegeben haben, sodass der Assistent die Daten umverteilen kann.

Protokollspeicherbedarf für die Datenumverteilung

Für die erfolgreiche Ausführung einer Datenumverteilungsoperation muss ausreichend Protokolldateispeicher zugewiesen werden, bevor die Datenumverteilungsoperation gestartet wird, damit sichergestellt ist, dass die Datenumverteilung nicht unterbrochen wird.

Anmerkung: Dieser Abschnitt enthält Verweise auf einige Schlüsselwortaktualisierungen, die nur zutreffen, wenn DB2 9.5 Fixpack 1 installiert ist. Wenn DB2 9.5 Fixpack 1 noch nicht verfügbar oder installiert ist, lesen Sie die Informationen des entsprechenden Abschnitts in DB2 Version 9. Dieser ist in der Informationszentrale von DB2 Version 9 unter folgender Adresse verfügbar: http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp

Die Menge des erforderlichen Protokolldateispeichers hängt von mehreren Faktoren ab, wie z. B. welche Optionen des Befehls REDISTRIBUTE DATABASE PARTITION GROUP verwendet werden.

Wenn der Befehl REDISTRIBUTE DATABASE PARTITION GROUP verwendet wird und die Option NOT ROLLFORWARD RECOVERABLE **nicht** verwendet wird, oder wenn die Umverteilung von einer anderen unterstützten Schnittstelle ausgeführt wird, für die die Datenumverteilung nicht aktualisierend wiederherstellbar ist, gilt Folgendes:

- Das Protokoll muss groß genug sein, um die INSERT- und DELETE-Operationen auf jeder Datenbankpartition unterzubringen, auf der Daten umverteilt werden. Die intensivsten Protokollierungsanforderungen werden an die Datenbankpartition gestellt, der die meisten Daten verloren gehen, bzw. an die Datenbankpartition, die die meisten Daten erhält.
- Wenn Sie die Anzahl der Datenbankpartitionen erhöhen, verwenden Sie das Verhältnis zwischen der Anzahl der aktuellen Datenbankpartitionen und der neuen Anzahl der Datenbankpartitionen, um die Anzahl der INSERT- und DELETE-Operationen abzuschätzen. Betrachten Sie z. B. das Umverteilen von Daten, die vor der Umverteilung gleichmäßig verteilt sind. Wenn Sie die Anzahl der Datenbankpartitionen von vier auf fünf erhöhen, werden ungefähr zwanzig Prozent der vier ursprünglichen Datenbankpartitionen in die neue Datenbankpartition versetzt. Dies bedeutet, dass zwanzig Prozent der DELETE-Operationen in jeder der vier ursprünglichen Datenbankpartitionen auftreten und alle INSERT-Operationen in der neuen Datenbankpartition auftreten.
- Betrachten Sie eine ungleichmäßige Verteilung von Daten, wie dies z. B. der Fall ist, wenn der Verteilungsschlüssel viele Nullwerte (NULL) enthält. In diesem Fall werden alle Zeilen, die einen NULL-Wert im Verteilungsschlüssel enthalten von einer Datenbankpartition unter dem alten Verteilungsschema in eine andere Datenbankpartition unter dem neuen Verteilungsschema versetzt. Dadurch erhöht sich die Menge des Protokollspeicherbereichs, der in den beiden Datenbankpartitionen benötigt wird, unter Umständen deutlich gegenüber der Menge, die für eine gleichmäßige Verteilung berechnet wurde.
- Die Umverteilung von jeder Tabelle ist eine einzelne Transaktion. Aus diesem Grund multiplizieren Sie bei der Berechnung des Protokollspeicherbereichs den Prozentsatz der Änderung, z. B. zwanzig Prozent, mit der Größe der größten Tabelle. Bedenken Sie aber, dass die größte Tabelle unter Umständen gleichmäßig verteilt ist, die zweitgrößte Tabelle aber z. B. über mindestens eine überdurchschnittlich große Datenbankpartition verfügen könnte. In solch einem Fall sollten Sie anstelle der größten Tabelle die Verwendung der ungleichmäßig verteilten Tabelle in Betracht ziehen.

Anmerkung: Nachdem Sie das maximale Volumen der Daten geschätzt haben, die in einer Datenbankpartition eingefügt und gelöscht werden sollen, verdoppeln Sie diesen Wert, um den Spitzenwert für die Größe der aktiven Protokolldatei zu ermitteln. Ist dieser geschätzte Wert größer als der Grenzwert von 512 GB für die aktive Protokolldatei, dann muss die Datenumverteilung schrittweise durchgeführt werden. Der Grenzwert für die aktive Protokolldatei wurde in Version 9.5 Fixpack 3 auf 1024 GB erhöht. Verwenden Sie das Dienstprogramm "makepmap", um eine

Reihe von Zielverteilungszuordnungen, jeweils eine pro Schritt, zu generieren. Sie können außerdem den Datenbankkonfigurationsparameter logsecond auf -1 setzen, um die meisten Probleme mit dem Protokollspeicherbereich zu vermeiden.

Wenn der Befehl REDISTRIBUTE DATABASE PARTITION GROUP verwendet wird und die Option NOT ROLLFORWARD RECOVERABLE verwendet wird, oder wenn die Umverteilung von einer anderen unterstützten Schnittstelle ausgeführt wird, für die die Datenumverteilung nicht aktualisierend wiederherstellbar ist, gilt Folgendes:

- Protokollsätze werden nicht erstellt, wenn Zeilen als Teil der Datenumverteilung versetzt werden. Dadurch wird der Protokolldateispeicherbedarf deutlich reduziert. Wird diese Option jedoch verwendet, wenn eine aktualisierende Recovery der Datenbank durchgeführt wird, kann der Protokollsatz der Umverteilungsoperation nicht aktualisierend wiederhergestellt werden und alle Tabellen, die als Teil der aktualisierenden Recovery-Operation verarbeitet wurden, verbleiben im Status UNAVAILABLE (nicht verfügbar). Eine Beschreibung der Auswirkungen, die die Verwendung der Option NOT ROLLFORWARD RECOVERABLE hat, finden Sie im Handbuch 'Command Reference'.
- Wenn die Datenbankpartitionsgruppe, für die eine Datenumverteilung durchgeführt wird, Tabellen mit Langfelddaten (LF-Daten) oder Daten großer Objekte (LOB-Daten) in den Tabellen enthält, ist die Anzahl der Protokollsätze, die während der Datenumverteilung generiert werden, höher, weil ein Protokollsatz für jede Datenzeile erstellt wird. In diesem Fall ist ein Protokollspeicherbedarf pro Datenbankpartition zu erwarten, der ungefähr ein Drittel des Volumens der Daten beträgt, die in dieser Partition bewegt (d. h. gesendet, empfangen oder beides) werden.

Ereignisprotokolldatei für die Umverteilung

Während der Datenumverteilung wird eine Ereignisprotokollierung durchgeführt. Die Ereignisdaten werden in einer Ereignisprotokolldatei protokolliert, die später für die Durchführung einer Fehlerbehebung verwendet werden kann.

Anmerkung: Dieser Abschnitt enthält Verweise auf einige Schlüsselwortaktualisierungen, die nur zutreffen, wenn DB2 9.5 Fixpack 1 installiert ist. Wenn DB2 9.5 Fixpack 1 noch nicht verfügbar oder installiert ist, lesen Sie die Informationen des entsprechenden Abschnitts in DB2 Version 9. Dieser ist in der Informationszentrale von DB2 Version 9 unter folgender Adresse verfügbar: http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp

Wenn die Datenumverteilung ausgeführt wird, werden Informationen zu jeder verarbeiteten Tabelle in einer einzelnen Ereignisprotokolldatei für die Umverteilung protokolliert.

Der Name der Ereignisprotokolldatei wird folgendermaßen formatiert: `datenbankname.datenbankpartitionsgruppenname.zeitmarke.protokoll`. Die Protokolldateien befinden sich an folgenden Positionen:

- Das Verzeichnis `homeinst/sqllib/redist` auf Linux®- und UNIX®-basierten Systemen.
- Das Verzeichnis `DB2INSTPROF\instance\redist` auf Windows®-Betriebssystemen. Dabei ist DB2INSTPROF der Wert der Registrierdatenbankvariablen DB2INSTPROF.

Das folgende Beispiel gibt den Namen einer Ereignisprotokolldatei an:

```
DB819.NG1.2007062419415651.log
```

Diese Ereignisprotokolldatei ist für eine Umverteilungsoperation auf einer Datenbank namens DB819 mit einer Datenbankpartitionsgruppe namens NG1, die am 24. Juni 2007 um 19:41 Ortszeit erstellt wurde.

Die Ereignisprotokolldateien dienen im Wesentlichen zu folgenden Zwecken:

- Zur Bereitstellung allgemeiner Informationen zur Umverteilungsoperation, wie zum Beispiel die alte und die neue Verteilungszuordnung.
- Zur Bereitstellung von Informationen, mit deren Hilfe Benutzer verfolgen können, welche Tabellen vom Dienstprogramm bereits umverteilt wurden.
- Zur Bereitstellung von Informationen zu den einzelnen Tabellen, die umverteilt wurden. Diese Informationen enthalten den für die Tabelle verwendeten Indexierungsmodus, eine Angabe zum Erfolg der Tabellenumverteilung sowie den Anfangs- und Endzeitpunkt für die Umverteilungsoperation an der Tabelle.

Weitere Informationen zu Protokolldateieinträgen für die Umverteilung und dazu wie Fehler behoben werden, die während der Datenumverteilung aufgetreten sind, finden Sie unter:

Umverteilen von Datenbankpartitionsgruppen mithilfe der Prozedur STEPWISE_REDISTRIBUTE_DBPG

Das Umverteilen von Daten kann mithilfe der systemdefinierten Prozedur STEPWISE_REDISTRIBUTE_DBPG ausgeführt werden.

Das Umverteilen von Datenbankpartitionsgruppen kann mithilfe der systemdefinierten Prozedur STEPWISE_REDISTRIBUTE_DBPG und anderen systemdefinierten Prozeduren ausgeführt werden.

Die folgenden Schritte beschreiben die Vorgehensweise und ein nachfolgendes Beispiel veranschaulicht diese Schritte:

1. Analysieren Sie die Datenbankpartitionsgruppe im Hinblick auf Protokollspeicherverfügbarkeit und Abweichung der Datenverteilung unter Verwendung der *Prozedur ANALYZE_LOG_SPACE - Protokollspeicherbereichsanalysedaten abrufen*.

   Die Funktion ANALYZE_LOG_SPACE gibt eine Ergebnismenge (einen geöffneten Cursor) von Ergebnissen einer Speicherbereichsanalyse zurück, die Felder für jede Datenbankpartition der angegebenen Datenbankpartitionsgruppe enthalten.
2. Erstellen Sie eine Datenverteilungsdatei für eine bestimmte Tabelle unter Verwendung der *Prozedur GENERATE_DISTFILE - Datenverteilungsdatei generieren*.

   Die Funktion GENERATE_DISTFILE generiert eine Datenverteilungsdatei für die angegebene Tabelle und speichert sie unter dem angegebenen Dateinamen.
3. Erstellen und dokumentieren Sie den Inhalt eines schrittweise durchgeführten Umverteilungsplans für die Datenbankpartitionsgruppe unter Verwendung der *Prozedur STEPWISE_REDISTRIBUTE_DBPG - Teil der Datenbankpartitionsgruppe umverteilen*.
4. Erstellen Sie eine Datenverteilungsdatei für eine bestimmte Tabelle unter Verwendung der *Prozedur GET_SWRD_SETTINGS - Umverteilungsdaten abrufen* und der *Prozedur SET_SWRD_SETTINGS - Umverteilungsregistrierdatenbank erstellen oder ändern*.

   Die Funktion GET_SWRD_SETTINGS liest die vorhandenen Umverteilungsregistrierdatensätze für die angegebene Datenbankpartitionsgruppe.

   Die Funktion SET_SWRD_SETTINGS erstellt oder ändert die Umverteilungsregistrierdatenbank. Wenn die Registrierdatenbank nicht vorhanden ist, erstellt

die Funktion sie und fügt ihr Datensätze hinzu. Wenn die Registrierdatenbank bereits vorhanden ist, stellt die Funktion über das Feld 'overwriteSpec' fest, welche Feldwerte überschrieben werden müssen. Das Feld 'overwriteSpec' gibt dieser Funktion die Möglichkeit, NULL-Eingaben für Felder zu akzeptieren, die nicht aktualisiert werden müssen.

5. Verteilen Sie die Datenbankpartitionsgruppe dem Plan entsprechend unter Verwendung der *Prozedur STEPWISE_REDISTRIBUTE_DBPG - Teil der Datenbankpartitionsgruppe umverteilen* um.

   Die Funktion STEPWISE_REDISTRIBUTE_DBPG führt die Umverteilung eines Teils der Datenbankpartitionsgruppe entsprechend der Eingabe und der Einstellungsdatei durch.

Verwendungsbeispiel

Das folgende Beispiel zeigt ein Script für den Befehlszeilenprozessor (CLP) unter AIX:

```
# ------------------------------------------------------------------------------------
# Angabe der Datenbank für die zu erstellende Verbindung
# ------------------------------------------------------------------------------------
dbName="SAMPLE"

# ------------------------------------------------------------------------------------
# Angabe des Namens für die Zieldatenbankpartitionsgruppe
# ------------------------------------------------------------------------------------
dbpgName="IBMDEFAULTGROUP"

# ------------------------------------------------------------------------------------
# Angabe des Schemas und des Namens der Tabelle
# ------------------------------------------------------------------------------------
tbSchema="$USER"
tbName="STAFF"

# ------------------------------------------------------------------------------------
# Angabe des Namens der Datenverteilungsdatei
# ------------------------------------------------------------------------------------
distFile="$HOME/sqllib/function/$dbName.IBMDEFAULTGROUP_swrdData.dst"

export DB2INSTANCE=$USER
export DB2COMM=TCPIP

# ------------------------------------------------------------------------------------
# Aufrufen der Aufrufanweisungen im Befehlszeilenprozessor (CLP)
# ------------------------------------------------------------------------------------
db2start
db2 -v "connect to $dbName"

# ------------------------------------------------------------------------------------
# Analyse des Effekts einer hinzugefügten Datenbankpartition ohne Anwendung der
# Änderungen - Analyse eines hypothetischen Falls
#
# - Im folgenden Fall werden hypothetisch die Datenbankpartitionen '40,50,60' der
#   Datenbankpartitionsgruppe hinzugefügt; für die Datenbankpartitionen '10,20,30,40,50,60'
#   werden die Zielverhältnisse 1:2:1:2:1:2 verwendet.
#
# Hinweis: In diesem Beispiel sind nur die Partitionen 10, 20 und 30 tatsächlich in der
#          Datenbankpartitionsgruppe vorhanden.
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.analyze_log_space('$dbpgName', '$tbSchema', '$tbName', 2, ' ',
'A', '40,50,60', '10,20,30,40,50,60', '1,2,1,2,1,2')"

# ------------------------------------------------------------------------------------
# Analyse des Effekts einer gelöschten Datenbankpartition ohne Anwendung der Änderungen
#
```

```
# - Im folgenden Fall wird hypothetisch die Datenbankpartition 30 aus der Datenbank-
#   partitionsgruppe gelöscht und die Daten werden auf die Datenbankpartitionen 10 und 20
#   unter Verwendung eines Zielverhältnisses von 1 : 1 umverteilt.
#
# Hinweis: In diesem Beispiel sollten alle Datenbankpartitionen, 10, 20 und 30, in der
#          Datenbankpartitionsgruppe vorhanden sein.
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.analyze_log_space('$dbpgName', '$tbSchema', '$tbName', 2, ' ',
'D', '30', '10,20','1,1')"

# ------------------------------------------------------------------------------------
# Generieren einer Datenverteilungsdatei zur Verwendung durch den Umverteilungsprozess
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.generate_distfile('$tbSchema', '$tbName', '$distFile')"

# ------------------------------------------------------------------------------------
# Schreiben eines schrittweisen Umverteilungsplans in eine Registrierdatenbank
#
# Der Wert 1 für den zehnten Parameter kann bewirken, dass die zurzeit ausgeführte
# gespeicherte Prozedur zur schrittweisen Umverteilung den aktuellen Schritt beendet und
# die Verarbeitung stoppt, bis dieser Parameter auf 0 zurückgesetzt wird und die
# gespeicherte Prozedur zur Umverteilung erneut aufgerufen wird.
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.set_swrd_settings('$dbpgName', 255, 0, ' ', '$distFile', 1000,
12, 2, 1, 0, '10,20,30', '50,50,50')"

# ------------------------------------------------------------------------------------
# Berichten des Inhalts des schrittweisen Umverteilungsplans für die angegebene
# Datenbankpartitionsgruppe
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.get_swrd_settings('$dbpgName', 255, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"

# ------------------------------------------------------------------------------------
# Umverteilen der Daten der Datenbankpartitionsgruppe "dbpgName" entsprechend dem in
# der Registrierdatenbank durch set_swrd_settings gespeicherten Umverteilungsplan.
# Mit Start bei Schritt 3 werden die Daten umverteilt, bis zwei Schritte im
# Umverteilungsplan ausgeführt sind.
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.stepwise_redistribute_dbpg('$dbpgName', 3, 2)"
```

Kapitel 27. Konfigurieren des Speichers mit automatischer Leistungsoptimierung

Speicher mit automatischer Leistungsoptimierung in Umgebungen mit partitionierten Datenbanken

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

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

Das auf einer Optimierungspartition basierende Modell setzt voraus, dass die Funktion nur in Datenbankpartitionen mit ähnlichen Speicheranforderungen verwendet wird. Die folgenden Richtlinien sind bei der Entscheidung zu beachten, ob die automatische Speicheroptimierung für eine partitionierte Datenbank aktiviert werden sollte.

Fälle, in denen die automatische Optimierung in partitionierten Datenbanken empfohlen wird

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

- Alle Datenbankpartitionen befinden sich auf identischer Hardware, und mehrere logische Knoten sind gleichmäßig auf mehrere physische Knoten verteilt.
- Sie weisen eine perfekte oder nahezu perfekte Verteilung der Daten auf.
- Die Auslastung, die in den Datenbankpartitionen verarbeitet wird, wird gleichmäßig auf die Datenbankpartitionen verteilt. Dies bedeutet, dass keine einzelne Datenbankpartition erhöhte Speicheranforderungen für einen oder mehrere Zwischenspeicher hat.

In einer solchen Umgebung ist für alle Datenbankpartitionen die gleiche Konfiguration wünschenswert, und die automatische Speicheroptimierung kann das System geeignet konfigurieren.

Fälle, in denen die automatische Optimierung in partitionierten Datenbanken nur mit Vorsicht empfohlen wird

In Fällen, in denen die meisten Datenbankpartitionen in einer Umgebung ähnliche Speicheranforderungen haben und auf ähnlicher Hardware betrieben werden, kann die automatische Speicheroptimierung eingesetzt werden, solange die Anfangskonfiguration mit Vorsicht erfolgt. Solche Systeme verfügen möglicherweise über eine Gruppe von Datenbankpartitionen für Daten und über eine wesentlich kleinere

Gruppe von Koordinatorpartitionen und Katalogpartitionen. In solchen Umgebungen kann es von Vorteil sein, die Koordinatorpartitionen und Katalogpartitionen anders zu konfigurieren als die Datenbankpartitionen mit den Daten.

Auch eine solche Umgebung kann von der automatischen Speicheroptimierungsfunktion profitieren, wenn einige kleinere Konfigurationsschritte ausgeführt werden. Da die Datenbankpartitionen mit den Daten den Hauptteil der Datenbankpartitionen ausmachen, sollte die automatische Optimierung für alle diese Datenbankpartitionen aktiviert und eine dieser Datenbankpartitionen als Optimierungspartition angegeben werden. Außerdem sollte die automatische Speicheroptimierung für die Katalog- und Koordinatorpartitionen inaktiviert werden, da sie möglicherweise eine andere Konfiguration aufweisen. Zur Inaktivierung der automatischen Optimierung in den Katalog- und Koordinatorpartitionen aktualisieren Sie den Datenbankkonfigurationsparameter *self_tuning_mem* in diesen Partitionen mit dem Wert OFF.

Fälle, in denen die automatische Optimierung in partitionierten Datenbanken nicht empfohlen wird

In Umgebungen, in denen sich die Speicheranforderungen jeder Datenbankpartition unterscheiden oder verschiedene Datenbankpartitionen auf völlig anderer Hardware betrieben werden, ist es empfehlenswert, die Funktion der automatischen Speicheroptimierung zu inaktivieren. Sie können dies erreichen, indem Sie in allen Partitionen den Datenbankkonfigurationsparameter *self_tuning_mem* auf den Wert OFF setzen.

Vergleich der Speicheranforderungen verschiedener Datenbankpartitionen

Die beste Methode zur Bestimmung, ob die Speicheranforderungen verschiedener Datenbankpartitionen ausreichend ähnlich sind, ist die Verwendung des Snapshot Monitors. Wenn die folgenden Monitorelemente in allen Partitionen ähnliche Werte liefern (mit Abweichungen von unter 20 %), können die Partitionen als ähnlich betrachtet werden.

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

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

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

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

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

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

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

Verwenden von Speicher mit automatischer Leistungsoptimierung in Umgebungen mit partitionierten Datenbanken

Wenn die Funktion für die automatische Optimierung in Umgebungen mit partitionierten Datenbanken aktiviert wird, überwacht eine einzige Datenbankpartition, die als *Optimierungspartition* bezeichnet wird, die Speicherkonfiguration und gibt alle Konfigurationsänderungen an alle anderen Datenbankpartitionen weiter, um eine konsistente Konfiguration über alle beteiligten Datenbankpartitionen hinweg sicherzustellen.

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

- Wenn Sie ermitteln möchten, welche Datenbankpartition zurzeit als Optimierungspartition angegeben ist, verwenden Sie die Prozedur ADMIN_CMD wie folgt:

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

- Zum Ändern der Optimierungspartition verwenden Sie die Prozedur ADMIN_CMD wie folgt:

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

  Wenn Sie diesen Befehl absetzen, wird die Optimierungspartition asynchron oder beim nächsten Starten der Datenbank aktualisiert.

- Wenn die Optimierungspartition von der Speicheroptimierungsfunktion automatisch neu ausgewählt werden soll, geben Sie für <db_partitionsnummer> den Wert -1 ein.

Starten der Speicheroptimierungsfunktion auf DPF-Systemen

In einer DPF-Umgebung wird die Speicheroptimierungsfunktion nur gestartet, wenn die Datenbank explizit mit dem Befehl ACTIVATE DATABASE aktiviert wird, weil die automatische Optimierung voraussetzt, dass alle Partitionen aktiv sind, bevor sie den Speicher in einem Mehrpartitionssystem richtig optimieren kann.

Inaktivieren der automatischen Optimierung für eine bestimmte Datenbankpartition

- Zur Inaktivierung der automatischen Optimierung für eine Teilgruppe von Datenbankpartitionen setzen Sie für die Datenbankpartitionen, die nicht optimiert werden sollen, den Konfigurationsparameter *self_tuning_mem* auf den Wert OFF.
- Zur Inaktivierung der automatischen Optimierung für eine Teilgruppe von Speicherkonsumenten, die durch Konfigurationsparameter gesteuert werden, in einer bestimmten Datenbankpartition, setzen Sie den Wert des relevanten Konfigurationsparameters oder die Pufferpoolgröße auf MANUAL bzw. einen bestimmten Wert in dieser Datenbankpartition. Es wird jedoch empfohlen, die Werte der

Konfigurationsparameter für die automatische Optimierungsfunktion über alle aktiven Partitionen hinweg einheitlich zu definieren.

- Wenn Sie die Optimierung für einen bestimmten Pufferpool in einer Datenbankpartition inaktivieren, führen Sie den Befehl ALTER BUFFERPOOL mit der Angabe eines Größenwerts und eines Werts für den Parameter PARTITIONNUM für die Partition aus, in der die automatische Optimierung inaktiviert werden soll.

  Eine Anweisung ALTER BUFFERPOOL, die die Größe mithilfe der Klausel PARTITIONNUM in einer bestimmten Datenbankpartition angibt, erstellt einen Ausnahmeeintrag für den angegebenen Pufferpool im Katalog SYSCAT.SYSBUFFERPOOLNODES oder aktualisiert den Ausnahmeeintrag, wenn bereits einer vorhanden ist. Wenn ein Ausnahmeeintrag für einen Pufferpool in diesem Katalog vorhanden ist, ist dieser Pufferpool an der automatischen Optimierung nicht beteiligt, wenn die Standardpufferpoolgröße auf den Wert AUTOMATIC gesetzt ist. Gehen Sie daher wie folgt vor, wenn Sie einen Ausnahmeeintrag entfernen möchten, sodass ein Pufferpool für die automatische Optimierung wieder aktiviert werden kann:

  1. Inaktivieren Sie die automatische Optimierung für diesen Pufferpool, indem Sie eine Anweisung ALTER BUFFERPOOL ausführen, die die Pufferpoolgröße auf einen bestimmten Wert setzt.
  2. Führen Sie eine weitere Anweisung ALTER BUFFERPOOL unter Angabe der Klausel PARTITIONNUM aus, um den Pufferpool in dieser Datenbankpartition auf die Standardpufferpoolgröße zu setzen.
  3. Aktivieren Sie die automatische Optimierung, indem Sie eine weitere Anweisung ALTER BUFFERPOOL ausführen, in der die Größe (SIZE) auf den Wert AUTOMATIC gesetzt ist.

Aktivieren des Speichers mit automatischer Leistungsoptimierung in nicht einheitlichen Umgebungen

Im Idealfall sollten Ihre Daten gleichmäßig auf alle Datenbankpartitionen verteilt und die Auslastung in jeder Partition durch ähnliche Speicheranforderungen gekennzeichnet sein. Wenn die Datenverteilung ungleichmäßig ist, sodass mindestens eine Datenbankpartition erheblich mehr oder weniger Daten als andere Datenbankpartitionen enthält, sollten solche anomalen Datenbankpartitionen nicht für die automatische Optimierung aktiviert werden. Dasselbe gilt, wenn die Speicheranforderungen in den Datenbankpartitionen unterschiedlich sind. Dies kann geschehen, wenn zum Beispiel ressourcenintensive Sortiervorgänge nur in einer Partition ausgeführt werden oder wenn einige Datenbankpartitionen mit anderer Hardware und mehr verfügbarem Speicher als andere Partitionen ausgestattet sind. Die automatische Optimierung kann dennoch in einigen Datenbankpartitionen in diesem Typ von Umgebung aktiviert werden. Zur Nutzung der Vorteile der automatischen Speicheroptimierung in Umgebungen mit ungleich verteilten Anforderungen, geben Sie eine Gruppe von Datenbankpartitionen an, die ähnliche Daten- und Speicheranforderungen haben, und aktivieren sie für die automatisch Optimierung. Die Speicherkonfiguration in den übrigen Partitionen muss in diesem Fall manuell erfolgen.

Kapitel 28. DB2-Konfigurationsparameter und -Variablen

Konfigurieren von Datenbanken über mehrere Partitionen

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

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

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

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

  ```
  DB2_UPDDBCFG_SINGLE_DBPARTITION=TRUE
  ```

  Anmerkung: Die Einstellung der Registrierdatenbankvariablen hat keine Relevanz für Anforderungen mit dem Befehl UPDATE DATABASE CONFIGURATION oder RESET DATABASE CONFIGURATION, die Sie mithilfe der Prozedur ADMIN_CMD ausführen.
- Sie können den Parameter **DBPARTITIONNUM** entweder mit dem Befehl UPDATE DATABASE CONFIGURATION oder mit dem Befehl RESET DATABASE CONFIGURATION oder mit der Prozedur ADMIN_CMD verwenden. Wenn Sie zum Beispiel die Datenbankkonfigurationen in allen Datenbankpartitionen aktualisieren möchten, rufen Sie die Prozedur ADMIN_CMD wie folgt auf:

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

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

  ```
  CALL SYSPROC.ADMIN_CMD
    ('UPDATE DB CFG DBPARTITIONNUM 10 USING sortheap 1000')
  ```
- Sie können den Parameter **DBPARTITIONNUM** mit der API db2CfgSet verwenden. Die Markierungen (Flags) in der Struktur **db2Cfg** geben an, ob der Wert für die Datenbankkonfiguration auf nur eine Datenbankpartition angewendet werden soll. Wenn Sie eine Markierung setzen, müssen Sie auch den Wert für **DBPARTITIONNUM** angeben. Beispiel:

  ```
  #define db2CfgSingleDbpartition          256
  ```

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

Bei der Migration der Datenbanken auf Version 9.5 behalten vorhandene Konfigurationsparameter im Allgemeinen Ihre Werte auch nach der Migration. Es werden jedoch neue Parameter mit den entsprechenden Standardwerten hinzugefügt, und einige vorhandene Parameter werden auf ihre neuen Standardwerte der Version 9.5 gesetzt. Der Abschnitt "Änderungen am Verhalten des DB2-Servers" in der Veröffentlichung *Migration* enthält Details zu den Änderungen an vorhandenen Datenbankkonfigurationsparametern. Für alle nachfolgenden Anforderungen zum Aktualisieren oder Zurücksetzen der Datenbankkonfiguration für die migrierten Datenbanken wird die Methode von Version 9.5 zum Aktualisieren oder Zurücksetzen der Konfiguration verwendet.

Für vorhandene Befehlsscripts zum Aktualisieren oder Zurücksetzen gelten ebenfalls die zuvor genannten Regeln: Sie können die Methode vor Version 9.5 verwenden, Sie können Ihre Scripts ändern, um die Option **DBPARTITIONNUM** des Befehls UPDATE DATABASE CONFIGURATION bzw. RESET DATABASE CONFIGURATION einzufügen oder Sie können die Registrierdatenbankvariable **DB2_UPDDBCFG_SINGLE_DBPARTITION** definieren.

Für vorhandene Anwendungen, die die API db2CfgSet aufrufen, müssen Sie die Methode von Version 9.5 verwenden. Wenn Sie die Methode vor Version 9.5 verwenden wollen, können Sie die Registrierdatenbankvariable **DB2_UPDDBCFG_SINGLE_DBPARTITION** definieren oder Ihre Anwendungen in der Weise ändern, dass sie die API mit der Versionsnummer von Version 9.5, einschließlich der neuen Markierung (Flag) db2CfgSingleDbpartition und des neuen Felds **dbpartitionnum**, aufrufen, um Datenbankkonfigurationen für eine bestimmte Datenbankpartition zu aktualisieren bzw. zurückzusetzen.

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

Variablen für Umgebungen mit partitionierten Datenbanken

DB2CHGPWD_EEE

- Betriebssystem: DB2 ESE unter AIX, Linux und Windows
- Standardwert=NULL, Werte: YES oder NO
- Diese Variable gibt an, ob Sie zulassen, dass andere Benutzer Kennwörter auf ESE-Systemen unter AIX oder Windows ändern. Es muss sichergestellt werden, dass die Kennwörter für alle Datenbankpartitionen oder Knoten mithilfe eines Windows-Domänencontrollers unter Windows oder LDAP unter AIX zentral verwaltet werden. Wenn Kennwörter nicht zentral verwaltet werden, bleiben sie möglicherweise nicht für alle Datenbankpartitionen bzw. Knoten konsistent. Dies könnte dazu führen, dass ein Kennwort nur in der Datenbankpartition geändert wird, mit der der Benutzer zur Durchführung der Änderung verbunden ist.

DB2_FCM_SETTINGS

- Betriebssystem: Linux
- Standardwert=NULL, Werte:
  - FCM_MAXIMIZE_SET_SIZE:[YES | TRUE | NO | FALSE]. Der Standardwert für FCM_MAXIMIZE_SET_SIZE ist NO.
  - FCM_CFG_BASE_AS_FLOOR: [YES | TRUE | NO | FALSE]. Der Standardwert für FCM_CFG_BASE_AS_FLOOR ist NO.
- Ab Version 9.5 Fixpack 4 können Sie die Registrierdatenbankvariable **DB2_FCM_SETTINGS** mit der Option FCM_MAXIMIZE_SET_SIZE de-

finieren, um einen Standardspeicherbereich von 4 GB für den FCM-Puffer (FCM, Fast Communications Manager) vorab zuzuordnen. Das Token muss entweder den Wert YES oder den Wert TRUE haben, um diese Funktion zu aktivieren.

Seit Version 9.5 Fixpack 6 können Sie die Registrierdatenbankvariable **DB2_FCM_SETTINGS** mit der Option FCM_CFG_BASE_AS_FLOOR definieren, um den Basiswert als Untergrenze für die Konfigurationsparameter *fcm_num_buffers* und *fcm_num_channels* des Datenbankmanagers festzulegen. Wenn die Option FCM_CFG_BASE_AS_FLOOR auf YES oder TRUE gesetzt ist und diese Parameter mit AUTOMATIC definiert sind und einen Anfangswert aufweisen, werden sie von DB2 nicht unterhalb dieses Werts optimiert.

DB2_NUM_FAILOVER_NODES

- Betriebssystem: Alle
- Standardwert=2, Werte: 0 bis zur erforderlichen Anzahl der Datenbankpartitionen
- Definieren Sie **DB2_NUM_FAILOVER_NODES**, um die Anzahl der zusätzlichen Datenbankpartitionen anzugeben, die auf einer Maschine im Falle einer Funktionsübernahme gestartet werden müssen.

  In einer DB2-Datenbanklösung mit hoher Verfügbarkeit können beim Ausfall eines Datenbankservers die Datenbankpartitionen auf der Maschine, auf der der Fehler aufgetreten ist, auf einer anderen Maschine erneut gestartet werden. FCM (Fast Communications Manager) verwendet **DB2_NUM_FAILOVER_NODES**, um zu berechnen, wie viel Speicherplatz auf den verschiedenen Maschinen reserviert werden muss, um diese Funktionsübernahme zu ermöglichen.

  Betrachten Sie zum Beispiel die folgende Konfiguration:

  – Maschine A verfügt über zwei Datenbankpartitionen: 1 und 2.
  – Maschine B verfügt über zwei Datenbankpartitionen: 3 und 4.
  – Für **DB2_NUM_FAILOVER_NODES** wird sowohl auf Maschine A als auch auf Maschine B der Wert 2 angegeben.

  Beim Starten von DB2 (DB2START) reserviert FCM genügend Speicherplatz auf A und B, um bis zu vier Datenbankpartitionen zu verwalten, sodass beim Ausfall einer Maschine die beiden Datenbankpartitionen der fehlerhaften Einheit auf der anderen Maschine erneut gestartet werden können. Wenn Maschine A ausfällt, dann können die Datenbankpartitionen 1 und 2 auf Maschine B erneut gestartet werden. Wenn Maschine B ausfällt, dann können die Datenbankpartitionen 3 und 4 auf Maschine A erneut gestartet werden.

DB2_PARTITIONEDLOAD_DEFAULT

- Betriebssystem: Alle unterstützten ESE-Plattformen
- Standardwert=YES, Werte: YES oder NO
- Die Registrierdatenbankvariable **DB2_PARTITIONEDLOAD_DEFAULT** ermöglicht Benutzern, die Standardfunktionsweise des Dienstprogramms LOAD in einer ESE-Umgebung zu ändern, wenn keine ESE-spezifischen LOAD-Optionen angegeben werden. Der Standardwert YES gibt an, dass in einer ESE-Umgebung, wenn Sie keine ESE-spezifischen LOAD-Optionen angeben, das Laden in allen Datenbankpartitionen versucht wird, in denen die Zieltabelle definiert ist. Wenn der Wert NO ist, wird das Laden nur in der Datenbankpartition versucht, mit der das Dienstprogramm LOAD gegenwärtig verbunden ist.

Anmerkung: Diese Variable ist veraltet und wird in einem späteren Release möglicherweise entfernt. Der Befehl LOAD bietet verschiedene Optionen, die zur Erzielung der gleichen Funktionsweise verwendet werden können. Sie können die gleichen Ergebnisse wie mit der Einstellung NO für diese Variable erzielen, indem Sie die folgenden Optionen im Befehl LOAD angeben: `PARTITIONED DB CONFIG MODE LOAD_ONLY OUTPUT_DBPARTNUMS x`. Dabei steht x für die Partitionsnummer der Partition, in die Sie Daten laden wollen.

DB2PORTRANGE

- Betriebssystem: Windows
- Werte: nnnn:nnnn
- Dieser Wert wird auf den TCP/IP-Portbereich gesetzt, der vom FCM verwendet wird, sodass alle zusätzlichen auf einer anderen Maschine erstellten Datenbankpartitionen den gleichen Portbereich haben.

Konfigurationsparameter für Umgebungen mit partitionierten Datenbanken

Kommunikation

conn_elapse - Antwortzeit für Verbindung

Dieser Parameter gibt an, innerhalb wie viel Sekunden eine TCP/IP-Verbindung zwischen zwei Datenbankpartitionsservern aufgebaut werden muss.

Konfigurationstyp
Datenbankmanager

Gilt für
Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
10 [0–100]

Maßeinheit
Sekunden

Wird der Verbindungsaufbau innerhalb der durch den Parameter angegebenen Zeit erfolgreich durchgeführt, kommt es zum Datenaustausch. Wird die Verbindung nicht rechtzeitig aufgebaut, wird ein weiterer Versuch zum Verbindungsaufbau durchgeführt. Kommt innerhalb der im Parameter *max_connretries* angegebenen Anzahl von Neuversuchen keine Verbindung zustande, wird eine Fehlernachricht ausgegeben.

fcm_num_buffers - Anzahl FCM-Puffer

Dieser Parameter gibt die Anzahl von 4-KB-Puffern an, die sowohl zwischen den als auch innerhalb der Datenbankserver für interne Kommunikation (Nachrichten) verwendet werden.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

32-Bit-Plattformen
Automatic [128 - 65 300]

64-Bit-Plattformen
Automatic [128 - 524 288]

- Datenbankserver mit lokalen und fernen Clients: Standardwert ist 1024.
- Datenbankserver mit lokalen Clients: Standardwert ist 512.
- Partitionierter Datenbankserver mit lokalen und fernen Clients: Standardwert ist 4096.

In Datenbanksystemen mit einer Partition wird dieser Parameter nur dann verwendet, wenn die partitionsinterne Parallelität durch Ändern des Parameters *intra_parallel* vom Standardwert NO auf den Wert YES aktiviert wurde.

Es ist möglich, sowohl einen Anfangswert als auch das Attribut AUTOMATIC anzugeben.

Wenn dieser Parameter auf den Wert AUTOMATIC gesetzt ist, überwacht FCM die Ressourcennutzung und kann Ressourcen entweder erhöhen oder verringern, wenn sie innerhalb von 30 Minuten nicht genutzt werden. Der Betrag, um den Ressourcen erhöht oder verringert werden können, hängt von der Plattform ab. Insbesondere können Ressourcen unter Linux nur um 25 % über den Anfangswert erhöht werden. Wenn der Datenbankmanager beim Starten einer Instanz die angegebene Anzahl von Ressourcen nicht zuordnen kann, verringert er die Konfigurationswerte inkrementell, bis er die Instanz starten kann.

Wenn Sie über mehrere logische Knoten auf derselben Maschine verfügen, kann es erforderlich sein, den Wert für diesen Parameter zu erhöhen. Außerdem kann es erforderlich sein, den Wert für diesen Parameter zu erhöhen, wenn wegen der Anzahl von Benutzern im System, wegen der Anzahl von Datenbankpartitionsservern im System oder wegen der Komplexität der Anwendungen die verfügbaren Nachrichtenpuffer nicht ausreichen.

Wenn Sie mehrere logische Knoten verwenden, wird ein Pool mit der im Parameter **fcm_num_buffers** angegebenen Anzahl von Puffern von allen logischen Knoten auf derselben Maschine gemeinsam benutzt. Die Größe des Pools wird durch die Multiplikation des Werts von **fcm_num_buffers** mit der Anzahl der logischen Knoten auf dieser physischen Maschine ermittelt. Überprüfen Sie daher den verwendeten Wert erneut. Berücksichtigen Sie, wie viele FCM-Puffer auf der Maschine (oder den Maschinen) mit mehreren logischen Knoten insgesamt zugeordnet werden.

Soll der Parameter **fcm_num_buffers** mit dem Attribut AUTOMATIC auf einen bestimmten Wert gesetzt werden und der Systemcontroller-Thread keine Ressourcen

unterhalb des angegebenen Werts anpassen, muss die Registrierungsdatenbankvariable **DB2_FCM_SETTINGS** konfiguriert werden. Definieren Sie die Option FCM-_CFG_BASE_AS_FLOOR der Registrierungsdatenbankvariablen **DB2_FCM_SETTINGS** mit YES (oder TRUE), um dieses Verhalten zu aktivieren. Der Wert der Registrierungsdatenbankvariablen **DB2_FCM_SETTINGS** wird dynamisch angepasst.

fcm_num_channels - Anzahl FCM-Kanäle

Mit diesem Parameter wird die Anzahl der FCM-Kanäle für jede Datenbankpartition angegeben.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients
- Satellitendatenbankserver mit lokalen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]

UNIX-Plattformen (32-Bit)
Automatic - mit den Anfangswerten 256, 512, 2 048 [128 - 120 000]

UNIX-Plattformen (64-Bit)
Automatic - mit den Anfangswerten 256, 512, 2 048 [128 - 524 288]

Windows (32-Bit)
Automatic - mit dem Anfangswert 10 000 [128 - 120 000]

Windows (64-Bit)
Automatic - mit den Anfangswerten 256, 512, 2 048 [128 - 524 288]

- Für einen Datenbankserver mit lokalen und fernen Clients ist der Anfangswert 512.
- Für einen Datenbankserver mit lokalen Clients ist der Anfangswert 256.
- Für Server in Umgebungen mit partitionierten Datenbanken mit lokalen und fernen Clients ist der Anfangswert 2 048.

In nicht partitionierten Datenbankumgebungen muss der Parameter *intra_parallel* aktiviert werden, bevor der Parameter *fcm_num_channels* verwendet werden kann.

Ein FCM-Kanal stellt einen logischen Kommunikationsendpunkt zwischen EDUs (Engine Dispatchable Units) dar, die in der DB2-Steuerkomponente ausgeführt werden. Sowohl Steuerungsflüsse (Anforderung und Antwort) als auch Datenflüsse (Daten aus Tabellenwarteschlangen) nutzen Kanäle, um Daten zwischen Partitionen zu übertragen.

Wenn dieser Parameter auf den Wert AUTOMATIC gesetzt ist, überwacht FCM die Kanalnutzung und ordnet Ressourcen je nach Bedarfsänderung inkrementell zu bzw. gibt sie inkrementell frei.

Soll der Parameter **fcm_num_channels** mit dem Attribut AUTOMATIC auf einen bestimmten Wert gesetzt werden und der Systemcontroller-Thread keine Ressourcen unterhalb des angegebenen Werts anpassen, muss die Registrierungsdatenbankvariable **DB2_FCM_SETTINGS** konfiguriert werden. Definieren Sie die Option FCM_CFG_BASE_AS_FLOOR der Registrierungsdatenbankvariablen **DB2_FCM_SETTINGS** mit YES (oder TRUE), um dieses Verhalten zu aktivieren. Der Wert der Registrierungsdatenbankvariablen **DB2_FCM_SETTINGS** wird dynamisch angepasst.

max_connretries - Anzahl Wiederholungen für Verbindungsaufbau zum Knoten

Dieser Parameter gibt an, wie häufig höchstens versucht wird, eine TCP/IP-Verbindung zwischen zwei Datenbankpartitionsservern aufzubauen.

Konfigurationstyp
Datenbankmanager

Gilt für
Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
5 [0–100]

Wenn der Verbindungsaufbau zwischen zwei Datenbankpartitionsservern fehlschlägt (z. B. bei Erreichen des im Parameter *conn_elapse* angegebenen Werts), gibt *max_connretries* an, wie viele Wiederholungen durchgeführt werden dürfen, um die Verbindung zu einem Datenbankpartitionsserver herzustellen. Bei Überschreitung des für diesen Parameter angegebenen Werts wird eine Fehlernachricht zurückgegeben.

max_time_diff - Maximale Zeitdifferenz zwischen Knoten

Dieser Parameter gibt die maximale Zeitdifferenz (in Minuten) an, die zwischen den in der Knotenkonfigurationsdatei aufgelisteten Datenbankpartitionsservern zulässig ist.

Konfigurationstyp
Datenbankmanager

Gilt für
Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
60 [1 - 1 440]

Maßeinheit
Minuten

Jeder Datenbankpartitionsserver verfügt über eine eigene Systemuhr. Wenn einer Transaktion zwei oder mehr Datenbankpartitionsserver zugeordnet sind und die Zeitdifferenz zwischen den Systemuhren den Wert, der im Parameter MAX_TIME_DIFF angegeben ist, überschreitet, wird die Transaktion zurückgewiesen und ein SQLCODE-Wert zurückgegeben. (Die Transaktion wird nur zurückgewiesen, wenn eine Datenänderung mit ihr verbunden ist.)

Es kann auch ein SQLCODE-Wert in Umgebungen mit partitionierten Datenbanken zurückgegeben werden, in denen DB2 die Systemuhren mit der virtuellen Zeitmarke (VTS, Virtual Timestamp) vergleicht, die in der Protokollsteuerdatei SQLOGCTL.LFH gespeichert ist. Wenn die Zeitmarke in der .LFH-Datei kleiner als die Systemzeit ist, wird die Zeit im Datenbankprotokoll auf die virtuelle Zeitmarke gesetzt, bis die Systemuhr mit dieser Zeit übereinstimmt. Die Fehlernachricht SQL1473N wird ebenfalls zurückgegeben, auch wenn die Zeitdifferenz zwischen mehreren Knoten kleiner als der Wert des Parameters MAX_TIME_DIFF ist.

DB2 verwendet die *UTC* (Coordinated Universal Time), damit unterschiedliche Zeitzonen beim Festlegen dieses Parameters nicht ins Gewicht fallen. Die UTC-Zeit ist identisch mit der Westeuropäischen Zeit (Greenwich Mean Time).

start_stop_time - Zeitlimit für DB2START und DB2STOP

Mit diesem Parameter wird die Zeit (in Minuten) angegeben, innerhalb welcher alle Datenbankpartitionsserver auf den Befehl DB2START oder DB2STOP reagieren müssen. Außerdem wird er als Zeitlimitwert für ADD DBPARTITIONNUM-Operationen verwendet.

Konfigurationstyp
Datenbankmanager

Gilt für
Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Sofort

Standardwert [Bereich]
10 [1 - 1 440]

Maßeinheit
Minuten

Datenbankpartitionsserver, die nicht innerhalb der angegebenen Zeit auf einen Befehl DB2START reagieren, senden eine Nachricht an das Fehlerprotokoll für db2start, das sich im Unterverzeichnis log des Unterverzeichnisses sqllib im Ausgangsverzeichnis für die Instanz befindet. Vor dem Neustart dieser Knoten sollten Sie auf diesen Knoten einen Befehl DB2STOP absetzen.

Datenbankpartitionsserver, die nicht innerhalb der angegebenen Zeit auf einen Befehl DB2STOP reagieren, senden eine Nachricht an das Fehlerprotokoll für db2stop, das sich im Unterverzeichnis log des Unterverzeichnisses sqllib im Ausgangsverzeichnis für die Instanz befindet. Sie können db2stop entweder für jeden oder einmal für alle nicht reagierenden Datenbankpartitionsserver absetzen. (Die bereits gestoppten Server melden, dass sie bereits gestoppt sind.)

Wenn eine db2start- oder eine db2stop-Operation in einer Mehrpartitionsdatenbank nicht innerhalb des Zeitwerts ausgeführt wird, der durch den Konfigurationsparameter *start_stop_time* des Datenbankmanagers angegeben wurde, werden die Datenbankpartitionen intern abgebrochen, für die das zulässige Zeitlimit überschritten wurde. Bei Umgebungen mit vielen Datenbankpartitionen und einem geringen Wert für *start_stop_time* kann es zu diesem Verhalten kommen. Erhöhen Sie den Wert für *start_stop_time*, um diesem Verhalten entgegenzuwirken.

Beim Hinzufügen einer neuen Datenbankpartition mithilfe des Befehls DB2START, START DATABASE MANAGER bzw. ADD DBPARTITIONNUM muss die Operation zum Hinzufügen der Datenbankpartition bestimmen, ob die einzelnen Datenbanken in der Instanz für dynamischen Speicher eingerichtet sind. Dies geschieht durch Kommunikation mit der Katalogpartition für jede einzelne Datenbank. Wenn der dynamische Speicher aktiviert ist, werden die Speicherpfaddefinitionen im Rahmen dieser Kommunikation abgerufen. Ähnlich gilt für den Fall, dass Tabellenbereiche für temporäre Tabellen zusammen mit den Datenbankpartitionen zu erstellen sind, dass die Operation möglicherweise mit einem anderen Datenbankpartitionsserver kommunizieren muss, um die Tabellenbereichsdefinitionen für die Datenbankpartitionen abzurufen, die sich auf diesem Server befinden. Diese Faktoren müssen bei der Festlegung des Werts für den Parameter *start_stop_time* berücksichtigt werden.

Parallelverarbeitung

intra_parallel - Partitionsinterne Parallelität aktivieren

Dieser Parameter gibt an, ob der Datenbankmanager partitionsinterne Parallelität verwenden kann.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Konfigurierbar

Standardwert [Bereich]
NO (0) [SYSTEM (-1), NO (0), YES (1)]

Der Wert -1 bewirkt, dass der Parameter auf „YES" oder „NO" gesetzt wird, abhängig von der Hardware, auf welcher der Datenbankmanager ausgeführt wird.

Ist dieser Parameter auf "YES" gesetzt, können die durch die Parallelverarbeitung erreichten Leistungssteigerungen unter anderem bei Datenbankabfragen und der Indexerstellung genutzt werden.

Anmerkung:

- Bei der parallelen Indexerstellung wird dieser Konfigurationsparameter nicht verwendet.
- Wenn Sie diesen Parameterwert ändern, werden Pakete möglicherweise erneut an die Datenbank gebunden und es kann zu Leistungseinbußen kommen.

max_querydegree - max_querydegree - Maximaler Grad der Parallelität bei Abfragen

Dieser Parameter gibt den maximalen Grad partitionsinterner Parallelität an, der für SQL-Anweisungen verwendet wird, die auf dieser Instanz des Datenbankmanagers ausgeführt werden. Eine SQL-Anweisung verwendet bei ihrer Ausführung innerhalb einer Datenbankpartition nicht mehr als diese Anzahl paralleler Operationen.

Konfigurationstyp
Datenbankmanager

Gilt für

- Datenbankserver mit lokalen und fernen Clients
- Datenbankserver mit lokalen Clients
- Partitionierten Datenbankserver mit lokalen und fernen Clients

Parametertyp
Online konfigurierbar

Weitergabeklasse
Anweisungsgrenzwert

Standardwert [Bereich]
-1 (ANY) [ANY, 1 - 32 767] (ANY bedeutet 'vom System festgelegt')

Der Konfigurationsparameter **intra_parallel** muss aktiviert (YES) sein, damit die Datenbankpartition die partitionsinterne Parallelität für SQL-Anweisungen verwenden kann. Der Parameter **intra_parallel** ist für die parallele Indexerstellung nicht mehr erforderlich.

Der Standardwert für diesen Konfigurationsparameter lautet -1. Dieser Wert bedeutet, dass das System den vom Optimierungsprogramm festgelegten Parallelitätsgrad verwendet. Andernfalls wird der vom Benutzer angegebene Wert verwendet.

Anmerkung: Der Grad der Parallelität für eine SQL-Anweisung kann bei der Kompilierung der Anweisung mithilfe des Sonderregisters CURRENT DEGREE oder der Bindeoption **DEGREE** angegeben werden.

Der maximale Grad der Parallelität für eine aktive Anwendung bei Abfragen kann mit dem Befehl SET RUNTIME DEGREE geändert werden. Der zur Laufzeit tatsächlich verwendete Parallelitätsgrad ist der niedrigste der folgenden Werte:

- Konfigurationsparameter **max_querydegree**
- Parallelitätsgrad der Anwendung zur Laufzeit
- Parallelitätsgrad bei der Kompilierung der SQL-Anweisung

Dieser Konfigurationsparameter gilt nur für Abfragen.

Teil 5. APIs, Befehle und SQL-Anweisungen zur Verwaltung

Kapitel 29. APIs zur Verwaltung

sqleaddn - Datenbankpartitionsserver der Umgebung mit partitionierten Datenbanken hinzufügen

Fügt einer Umgebung mit partitionierten Datenbanken einen neuen Datenbankpartitionsserver hinzu. Diese Anwendungsprogrammierschnittstelle (API) erstellt auf dem neuen Datenbankpartitionsserver Datenbankpartitionen für alle Datenbanken, die zurzeit in der Instanz definiert sind. Der Benutzer kann den Quellendatenbankpartitionsserver für beliebige Tabellenbereiche für temporäre Systemtabellen, die mit den Datenbanken erstellt werden sollen, angeben oder angeben, dass keine Tabellenbereiche für temporäre Systemtabellen erstellt werden sollen. Die API muss von dem Datenbankpartitionsserver aus abgesetzt werden, der hinzugefügt wird, und kann nur auf einem Datenbankpartitionsserver abgesetzt werden.

Geltungsbereich

Diese API betrifft nur den Datenbankpartitionsserver, auf dem sie ausgeführt wird.

Berechtigung

Eine der folgenden Berechtigungen:

- Berechtigung SYSADM
- Berechtigung SYSCTRL

Erforderliche Verbindung

Keine

Einzuschließende Datei der API

```
sqlenv.h
```

API- und Datenstruktursyntax

```
SQL_API_RC SQL_API_FN
  sqleaddn (
        void * pAddNodeOptions,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgaddn (
        unsigned short addnOptionsLen,
        struct sqlca * pSqlca,
        void * pAddNodeOptions);
```

Parameter der API 'sqleaddn'

pAddNodeOptions
Eingabe. Ein Zeiger auf die optionale Struktur 'sqle_addn_options'. Diese Struktur dient zur Angabe des Quellendatenbankpartitionsservers, sofern zutreffend, der Definitionen für Tabellenbereiche für temporäre Systemtabellen für alle Datenbankpartitionen, die während der Operation zum Hinzufügen des Knotens erstellt werden. Wenn nicht angegeben (d. h., wenn

ein Nullzeiger angegeben wird), stimmen die Definitionen der Tabellenbereiche für temporäre Systemtabellen mit denen für die Katalogpartition überein.

pSqlca
Ausgabe. Ein Zeiger auf die Struktur 'sqlca'.

Spezielle Parameter für die API 'sqlgaddn'

addnOptionsLen
Eingabe. Ein 2 Byte langer ganzzahliger Wert ohne Vorzeichen, der die Länge der optionalen Struktur 'sqle_addn_options' in Byte angibt.

Hinweise

Stellen Sie vor dem Hinzufügen eines neuen Datenbankpartitionsservers sicher, dass genügend Speicherplatz für die Container verfügbar ist, die für alle vorhandenen Datenbanken auf dem System erstellt werden müssen.

Die Operationen zum Hinzufügen eines Knotens erstellt eine leere Datenbankpartition auf dem neuen Datenbankpartitionsserver für jede Datenbank, die in der Instanz vorhanden ist. Die Konfigurationsparameter für die neuen Datenbankpartitionen werden auf ihre Standardwerte gesetzt.

Wenn eine Operationen zum Hinzufügen eines Knotens während der lokalen Erstellung einer Datenbankpartition fehlschlägt, tritt die Operation in eine Bereinigungsphase ein, in der sie alle erstellten Datenbanken lokal löscht. Dies bedeutet, dass die Datenbankpartitionen nur von dem Datenbankpartitionsserver entfernt werden, der hinzugefügt wird (d. h. vom lokalen Datenbankpartitionsserver). Vorhandene Datenbankpartitionen auf allen anderen Datenbankpartitionsservern bleiben unberührt. Wenn die Bereinigung fehlschlägt, wird keine weitere Bereinigung ausgeführt und ein Fehler zurückgegeben.

Die Datenbankpartitionen auf dem neuen Datenbankpartitionsserver können erst zum Speichern von Benutzerdaten verwendet werden, wenn der Datenbankpartitionsserver mit der Anweisung ALTER DATABASE PARTITION GROUP einer Datenbankpartitionsgruppe hinzugefügt wurde.

Diese API schlägt fehl, wenn gleichzeitig eine Operation zum Erstellen oder Löschen einer Datenbank ausgeführt wird. Die API kann erneut aufgerufen werden, wenn die Operation abgeschlossen ist.

Diese API schlägt fehl, wenn in einer Datenbank im System zu einem beliebigen Zeitpunkt eine Benutzertabelle mit einer XML-Spalte erfolgreich oder nicht erfolgreich erstellt wurde oder ein XSR-Objekt erfolgreich oder nicht erfolgreich registriert wurde.

Zur Feststellung, ob für eine Datenbank der dynamische Speicher aktiviert ist oder nicht, muss die API 'sqleaddn' mit der Katalogpartition für jede der Datenbanken in der Instanz kommunizieren. Wenn der dynamische Speicher aktiviert ist, werden die Speicherpfaddefinitionen im Rahmen dieser Kommunikation abgerufen. Ähnlich gilt für den Fall, dass Tabellenbereiche für temporäre Systemtabellen zusammen mit den Datenbankpartitionen zu erstellen sind, dass die API 'sqleaddn' möglicherweise mit einem anderen Datenbankpartitionsserver in der Umgebung mit partitionierten Datenbanken kommunizieren muss, um die Tabellenbereichsdefinitionen abzurufen. Der Konfigurationsparameter 'start_stop_time' des Datenbankmanagers wird zur Angabe der Zeit (in Minuten) verwendet, in der der andere Daten-

bankpartitionsserver mit den Definitionen zum dynamischen Speicher und zu den Tabellenbereichen antworten muss. Wenn diese Zeit überschritten wird, schlägt die API fehl. Erhöhen Sie in diesem Fall den Wert für den Parameter 'start_stop_time', und rufen Sie die API erneut auf.

API-Syntax für REXX

Diese API kann in REXX über die Schnittstelle SQLDB2 aufgerufen werden.

sqlecran - Datenbank auf einem Datenbankpartitionsserver erstellen

Erstellt eine Datenbank nur auf dem Datenbankpartitionsserver, der die Anwendungsprogrammierschnittstelle (API) aufruft. Diese API ist nicht für den allgemeinen Gebrauch vorgesehen. Sie kann zum Beispiel mit 'db2Restore' verwendet werden, wenn die Datenbankpartition auf einem Datenbankpartitionsserver beschädigt wurde und erneut erstellt werden muss. Eine unsachgemäße Verwendung dieser API kann Inkonsistenzen im System zur Folge haben. Daher sollte sie nur mit Vorsicht verwendet werden.

Anmerkung: Wenn diese API zur erneuten Erstellung einer Datenbankpartition, die (wegen Beschädigung) gelöscht wurde, verwendet wird, befindet sich die Datenbank auf diesem Datenbankpartitionsserver anschließend im Status 'Restore anstehend'. Nach der erneuten Erstellung der Datenbankpartition muss die Datenbank sofort auf diesem Datenbankpartitionsserver durch einen Restore wiederhergestellt werden.

Geltungsbereich

Diese API betrifft nur den Datenbankpartitionsserver, für den sie aufgerufen wird.

Berechtigung

Eine der folgenden Berechtigungen:

- Berechtigung SYSADM
- Berechtigung SYSCTRL

Erforderliche Verbindung

Instanzverbindung. Zur Erstellung einer Datenbank auf einem anderen Datenbankpartitionsserver ist es erforderlich, zunächst eine Verbindung (ATTACH) zu diesem Datenbankpartitionsserver herzustellen. Eine Datenbankverbindung wird während der Verarbeitung durch diese API temporär hergestellt.

Einzuschließende Datei der API

```
sqlenv.h
```

API- und Datenstruktursyntax

```
SQL_API_RC SQL_API_FN
  sqlecran (
        char * pDbName,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgcran (
        unsigned short reservedLen,
```

```
unsigned short dbNameLen,
struct sqlca * pSqlca,
void * pReserved,
char * pDbName);
```

Parameter der API 'sqlecran'

pDbName
Eingabe. Eine Zeichenfolge, die den Namen der zu erstellenden Datenbank enthält. Darf nicht den Wert NULL haben.

pReserved
Eingabe. Ein reservierter Zeiger, der auf NULL gesetzt ist oder auf null zeigt. Reserviert für zukünftigen Gebrauch.

pSqlca
Ausgabe. Ein Zeiger auf die Struktur 'sqlca'.

Spezielle Parameter für die API 'sqlgcran'

reservedLen
Eingabe. Reserviert für die Länge des Parameters 'pReserved'.

dbNameLen
Eingabe. Ein 2 Byte langer ganzzahliger Wert ohne Vorzeichen, der die Länge des Datenbanknamens in Byte angibt.

Hinweise

Wenn die Datenbank erfolgreich erstellt wird, wird sie in den Status 'Restore anstehend' versetzt. Für die Datenbank muss ein Restore auf diesem Datenbankpartitionsserver ausgeführt werden, bevor sie verwendet werden kann.

API-Syntax für REXX

Diese API kann in REXX über die Schnittstelle SQLDB2 aufgerufen werden.

sqledpan - Datenbank auf einem Datenbankpartitionsserver löschen

Löscht eine Datenbank auf einem angegebenen Datenbankpartitionsserver. Kann nur in einer Umgebung mit partitionierten Datenbanken ausgeführt werden.

Geltungsbereich

Diese Anwendungsprogrammierschnittstelle (API) betrifft nur den Datenbankpartitionsserver, für den sie aufgerufen wird.

Berechtigung

Eine der folgenden Berechtigungen:

- Berechtigung SYSADM
- Berechtigung SYSCTRL

Erforderliche Verbindung

Keine. Eine Instanzverbindung (ATTACH) wird für die Dauer des Aufrufs hergestellt.

Einzuschließende Datei der API

```
sqlenv.h
```

API- und Datenstruktursyntax

```
SQL_API_RC SQL_API_FN
  sqledpan (
        char * pDbAlias,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdpan (
        unsigned short Reserved1,
        unsigned short DbAliasLen,
        struct sqlca * pSqlca,
        void * pReserved2,
        char * pDbAlias);
```

Parameter der API 'sqledpan'

pDbAlias
Eingabe. Eine Zeichenfolge, die den Aliasnamen der zu löschenden Datenbank enthält. Dieser Name dient zum Verweis auf den tatsächlichen Datenbanknamen im Systemdatenbankverzeichnis.

pReserved
Reserviert. Sollte NULL sein.

pSqlca
Ausgabe. Ein Zeiger auf die Struktur 'sqlca'.

Spezielle Parameter für die API 'sqlgdpan'

Reserved1
Reserviert für zukünftigen Gebrauch.

DbAliasLen
Eingabe. Ein 2 Byte langer ganzzahliger Wert ohne Vorzeichen, der die Länge des Aliasnamens der Datenbank in Byte angibt.

pReserved2
Ein reservierter Zeiger, der auf NULL gesetzt ist oder auf null zeigt. Reserviert für zukünftigen Gebrauch.

Hinweise

Eine unsachgemäße Verwendung dieser API kann Inkonsistenzen im System zur Folge haben. Daher sollte sie nur mit Vorsicht verwendet werden.

API-Syntax für REXX

Diese API kann in REXX über die Schnittstelle SQLDB2 aufgerufen werden.

sqledrpn - Überprüfen, ob ein Datenbankpartitionsserver gelöscht werden kann

Prüft, ob ein Datenbankpartitionsserver von einer Datenbank verwendet wird. Es wird eine Nachricht zurückgegeben, die angibt, ob der Datenbankpartitionsserver gelöscht werden kann.

Geltungsbereich

Diese Anwendungsprogrammierschnittstelle (API) betrifft nur den Datenbankpartitionsserver, für den sie ausgeführt wird.

Berechtigung

Eine der folgenden Berechtigungen:

- Berechtigung SYSADM
- Berechtigung SYSCTRL

Einzuschließende Datei der API

```
sqlenv.h
```

API- und Datenstruktursyntax

```
SQL_API_RC SQL_API_FN
  sqledrpn (
        unsigned short Action,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdrpn (
        unsigned short Reserved1,
        struct sqlca * pSqlca,
        void * pReserved2,
        unsigned short Action);
```

Parameter der API 'sqledrpn'

Action
Die angeforderte Aktion. Gültiger Wert: SQL_DROPNODE_VERIFY

pReserved
Reserviert. Sollte NULL sein.

pSqlca
Ausgabe. Ein Zeiger auf die Struktur 'sqlca'.

Spezielle Parameter für die API 'sqlgdrpn'

Reserved1
Reserviert für die Länge des Parameters 'pReserved2'.

pReserved2
Ein reservierter Zeiger, der auf NULL gesetzt ist oder auf 0 zeigt. Für den zukünftigen Gebrauch reserviert.

Hinweise

Wenn eine Nachricht zurückgegeben wird, die besagt, dass der Datenbankpartitionsserver nicht im Gebrauch ist, können Sie den Befehl 'db2stop' mit der Option DROP NODENUM verwenden, um den Eintrag für den Datenbankpartitionsserver aus der Datei 'db2nodes.cfg' zu entfernen. Dadurch wird der Datenbankpartitionsserver zugleich aus der Umgebung mit partitionierten Datenbanken entfernt.

Wenn eine Nachricht zurückgegeben wird, die besagt, dass der Datenbankpartitionsserver im Gebrauch ist, sollten die folgenden Aktionen ausgeführt werden:

1. Der zu löschende Datenbankpartitionsserver enthält wahrscheinlich eine Datenbankpartition für jede Datenbank in der Instanz. Wenn einige dieser Datenbankpartitionen Daten enthalten, verteilen Sie die Datenbankpartitionsgruppen um, die diese Datenbankpartitionen verwenden. Verteilen Sie die Datenbankpartitionsgruppen um, um die Daten in Datenbankpartitionen zu versetzen, die sich auf Datenbankpartitionsservern befinden, die nicht gelöscht werden.
2. Nach der Umverteilung der Datenbankpartitionsgruppen können Sie die Datenbankpartition aus jeder Datenbankpartitionsgruppe löschen, die sie verwendet. Zum Entfernen einer Datenbankpartition aus einer Datenbankpartitionsgruppe können Sie entweder die Option zum Löschen von Knoten ('drop node') der API 'sqludrdt' oder die Anweisung ALTER DATABASE PARTITION GROUP verwenden.
3. Löschen Sie alle Ereignismonitore, die auf dem Datenbankpartitionsserver definiert sind.
4. Führen Sie 'sqledrpn' erneut aus, um sicherzustellen, dass die Datenbankpartition auf dem Datenbankpartitionsserver nicht mehr im Gebrauch ist.

API-Syntax für REXX

Diese API kann in REXX über die Schnittstelle SQLDB2 aufgerufen werden.

sqlugrpn - Nummer des Datenbankpartitionsservers für eine Zeile abrufen

Gibt die Nummer der Datenbankpartition und des Datenbankpartitionsservers auf der Basis der Verteilungsschlüsselwerte zurück. Eine Anwendung kann anhand dieser Information feststellen, in welchem Datenbankpartitionsserver eine bestimmte Zeile einer Tabelle gespeichert ist.

Die Partitionierungsdatenstruktur 'sqlupi' fungiert als Eingabe für diese Anwendungsprogrammierschnittstelle (API). Die Struktur kann von der API 'sqlugtpi' zurückgegeben werden. Eine andere Eingabe sind die Zeichendarstellungen der entsprechenden Verteilungsschlüsselwerte. Die Ausgabe ist eine von der Verteilungsstrategie generierte Datenbankpartitionsnummer und die entsprechende Nummer des Datenbankpartitionsservers aus der Verteilungszuordnung. Wenn die Informationen der Verteilungszuordnung nicht angegeben werden, wird nur die Datenbankpartitionsnummer zurückgegeben. Dies kann bei der Analyse der Datenverteilung nützlich sein.

Der Datenbankmanager braucht nicht aktiv zu sein, wenn diese API aufgerufen wird.

Geltungsbereich

Diese API muss von einem in der Datei 'db2nodes.cfg' definierten Datenbankpartitionsserver aus aufgerufen werden. Diese API darf nicht von einem Client aus aufgerufen werden, da dies dazu führen könnte, dass falsche Informationen zur Datenbankpartitionierung aufgrund unterschiedlicher Codepages und ENDIAN-Definitionen zwischen dem Client und dem Server zurückgegeben werden.

Berechtigung

Keine

Einzuschließende Datei der API

```
sqlutil.h
```

API- und Datenstruktursyntax

```
SQL_API_RC SQL_API_FN
  sqlugrpn (
   unsigned short num_ptrs,
   unsigned char ** ptr_array,
   unsigned short * ptr_lens,
   unsigned short territory_ctrycode,
   unsigned short codepage,
   struct sqlupi * part_info,
   short * part_num,
   SQL_PDB_NODE_TYPE * node_num,
   unsigned short chklvl,
   struct sqlca * sqlca,
   short dataformat,
   void * pReserved1,
   void * pReserved2);

SQL_API_RC SQL_API_FN
  sqlggrpn (
   unsigned short num_ptrs,
   unsigned char ** ptr_array,
   unsigned short * ptr_lens,
   unsigned short territory_code,
   unsigned short codepage,
   struct sqlupi * part_info,
   short * part_num,
   SQL_PDB_NODE_TYPE * node_num,
   unsigned short chklvl,
   struct sqlca * sqlca,
   short dataformat,
   void * pReserved1,
   void * pReserved2);
```

Parameter der API 'sqlugrpn'

num_ptrs
Die Anzahl der Zeiger in 'ptr_array'. Der Wert muss mit einem der Werte übereinstimmen, die für den Parameter 'part_info' angegeben werden (d. h.: part_info->sqld).

ptr_array
Eine Feldgruppe (Array) von Zeigern, die auf die Zeichendarstellungen der entsprechenden Werte jedes Teils des Verteilungsschlüssels verweisen, der in 'part_info' angegeben wird. Wenn ein Nullwert erforderlich ist, wird der entsprechende Zeiger auf null gesetzt. Für generierte Spalten generiert diese Funktion keine Werte für die Zeile. Der Benutzer ist dafür verantwortlich, dass ein Wert angegeben wird, der eine ordnungsgemäße Partitionierung der Zeile zur Folge hat.

ptr_lens
Eine Feldgruppe aus ganzzahligen Werten (Integer) ohne Vorzeichen, die die Längen der Zeichendarstellungen der entsprechenden Werte jedes Teils des Partitionierungsschlüssels enthält, der in 'part_info' angegeben wird.

territory_ctrycode
Der Landes-/Regionscode der Zieldatenbank. Dieser Wert kann auch aus der Datenbankkonfiguration mithilfe des Befehls GET DATABASE CONFIGURATION abgerufen werden.

codepage
Die Codepage der Zieldatenbank. Dieser Wert kann auch aus der Datenbankkonfiguration mithilfe des Befehls GET DATABASE CONFIGURATION abgerufen werden.

part_info
Ein Zeiger auf die Struktur 'sqlupi'.

part_num
Ein Zeiger auf einen 2 Byte langen ganzzahligen Wert mit Vorzeichen, der zum Speichern der Datenbankpartitionsnummer verwendet wird.

node_num
Ein Zeiger auf ein SQL_PDB_NODE_TYPE-Feld, das zum Speichern der Knotennummer dient. Wenn der Zeiger null ist, wird keine Knotennummer zurückgegeben.

chklvl Ein ganzzahliger Wert ohne Vorzeichen, der den Grad der Überprüfung angibt, der auf Eingabeparameter angewendet wird. Wenn der angegebene Wert null ist, erfolgt keine Überprüfung. Wenn ein Wert ungleich null angegeben wird, werden alle Eingabeparameter überprüft.

sqlca Ausgabe. Ein Zeiger auf die Struktur 'sqlca'.

dataformat
Gibt die Darstellung der Verteilungsschlüsselwerte an. Gültige Werte:

SQL_CHARSTRING_FORMAT
Alle Verteilungsschlüsselwerte werden durch Zeichenfolgen dargestellt. Dies ist der Standardwert.

SQL_IMPLIEDDECIMAL_FORMAT
Die Position einer impliziten Dezimalstelle wird durch die Spaltendefinition festgelegt. Wenn die Spaltendefinition zum Beispiel DECIMAL(8,2) angibt, wird der Wert 12345 als 123,45 verarbeitet.

SQL_PACKEDDECIMAL_FORMAT
Alle Verteilungsschlüsselwerte von Dezimalspalten werden im gepackten Dezimalformat verarbeitet.

SQL_BINARYNUMERICS_FORMAT
Alle numerischen Verteilungsschlüsselwerte liegen im Big Endian-Format vor.

pReserved1
Reserviert für zukünftigen Gebrauch.

pReserved2
Reserviert für zukünftigen Gebrauch.

Hinweise

Datentypen, die vom Betriebssystem unterstützt werden, sind diejenigen, die als Verteilungsschlüssel definiert werden können.

Anmerkung: CHAR-, VARCHAR-, GRAPHIC- und VARGRAPHIC-Datentypen müssen in die Datenbankcodepage konvertiert werden, bevor diese API aufgerufen wird.

Für numerische Datentypen und Datentypen für Datum und Uhrzeit müssen die Zeichendarstellungen in der Codepage des jeweiligen Systems vorliegen, auf dem die API aufgerufen wird.

Wenn 'node_num' nicht null ist, muss die Verteilungszuordnung angegeben werden. Das heißt, dass das Feld 'pmaplen' im Parameter 'part_info' (part_info->pmaplen) entweder den Wert 2 oder den Wert 8192 hat. Ansonsten wird der SQLCODE-Wert -6038 zurückgegeben. Der Verteilungsschlüssel muss definiert werden. Das heißt, das Feld 'sqld' im Parameter 'part_info' (part_info->sqld) muss einen Wert größer null enthalten. Ansonsten wird der SQLCODE-Wert -2032 zurückgegeben.

Wenn einer Partitionierungsspalte, die keine Nullwerte enthalten darf, ein Nullwert zugewiesen wird, wird der SQLCODE-Wert -6039 zurückgegeben.

Alle führenden und folgenden Leerzeichen der Eingabezeichenfolge werden entfernt. Dies gilt jedoch nicht für CHAR-, VARCHAR-, GRAPHIC- und VARGRAPHIC-Datentypen. Bei diesen werden nur die folgenden Leerzeichen entfernt.

Kapitel 30. Befehle

REDISTRIBUTE DATABASE PARTITION GROUP

Führt eine Umverteilung von Daten auf die Datenbankpartitionen in einer Datenbankpartitionsgruppe aus. Die Zielverteilung von Daten kann gleichmäßig (UNIFORM, Standardeinstellung) sein oder vom Benutzer definiert werden, um bestimmte Systemvoraussetzungen zu erfüllen.

Der Befehl REDISTRIBUTE DATABASE PARTITION GROUP führt eine Umverteilung von Daten auf alle Partitionen in einer Datenbankpartitionsgruppe aus. Die Operation betrifft alle Objekte, die sich in der Datenbankpartitionsgruppe befinden, und lässt sich nicht nur auf ein Objekt einschränken.

Dieser Befehl kann nur von der Katalogdatenbankpartition aus ausgeführt werden. Verwenden Sie den Befehl LIST DATABASE DIRECTORY, um festzustellen, welche Datenbankpartition die Katalogdatenbankpartition für die jeweilige Datenbank ist.

Geltungsbereich

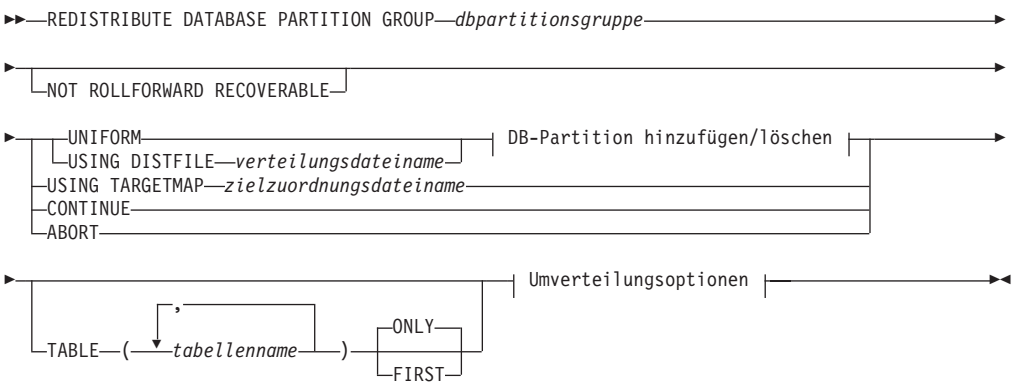
Dieser Befehl betrifft alle Datenbankpartitionen in der jeweiligen Datenbankpartitionsgruppe.

Berechtigung

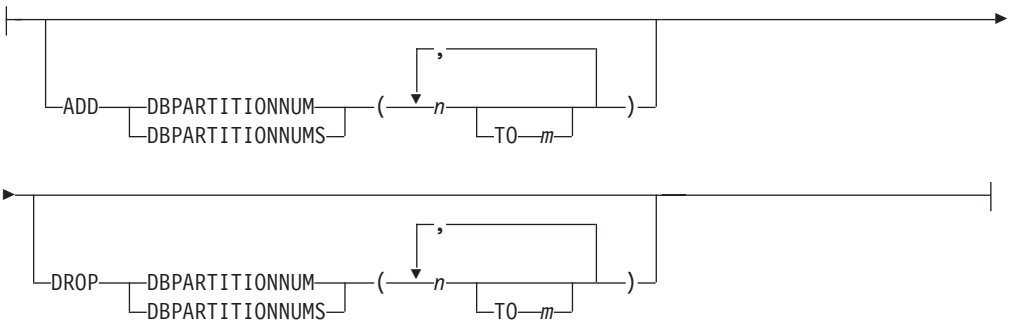
Eine der folgenden Berechtigungen:

- SYSADM
- SYSCTRL (Außerdem sind für alle Tabellen in der umzuverteilenden Datenbankpartitionsgruppe die Zugriffsrechte DELETE, INSERT und UPDATE erforderlich.)
- DBADM

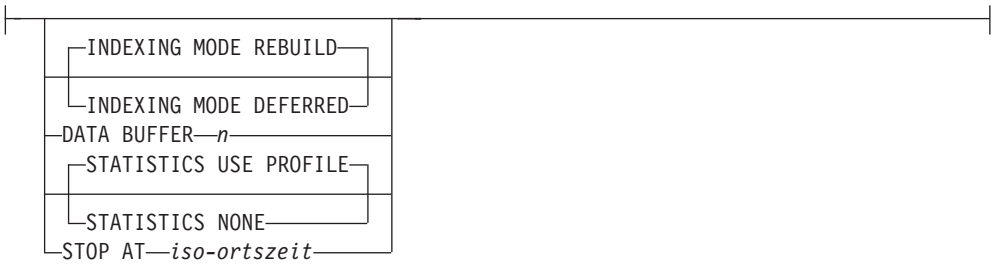
Befehlssyntax



DB-Partition hinzufügen/löschen:



Umverteilungsoptionen:



Befehlsparameter

DATABASE PARTITION GROUP *dbpartitionsgruppe*
Der Name der Datenbankpartitionsgruppe. Dieser einteilige Name gibt eine Datenbankpartitionsgruppe an, die in der Katalogtabelle SYSCAT-.DBPARTITIONGROUPS beschrieben ist. In der Datenbankpartitionsgruppe darf zu diesem Zeitpunkt keine Umverteilung stattfinden.

Anmerkung: Tabellen in den Datenbankpartitionsgruppen IBMCAT-GROUP und IBMTEMPGROUP können nicht umverteilt werden.

NOT ROLLFORWARD RECOVERABLE
Diese Option ist nur verfügbar, wenn DB2 9.5 Fixpack 1 installiert ist. Wenn diese Option verwendet wird, ist der Befehl REDISTRIBUTE DATABASE PARTITION GROUP nicht durch eine aktualisierende Recovery wiederherstellbar.

- Die Daten werden in einer Massendatenoperation und nicht durch interne Einfüge- und Löschoperationen versetzt. Dies verringert die Häufigkeit, mit der eine Tabelle durchsucht und ein Zugriff auf sie erfolgen muss, sodass sich eine bessere Leistung ergibt.
- Protokollsätze sind nicht mehr für jede einzelne der Einfüge- und Löschoperationen erforderlich. Dies bedeutet, dass Sie keine großen Speicherbereiche für aktive Protokolle und Protokollarchive mehr in Ihrem System verwalten müssen, wenn Sie eine Datenumverteilung ausführen. Dies ist insbesondere für solche Fälle von Vorteil, in denen Sie in der Vergangenheit wegen eines sehr hohen Speicherbedarfs für aktive Protokolle und des sonstigen Speicherbedarfs eine einzelne umfangreiche Datenumverteilungsoperation in mehrere kleinere Umverteilungsoperationen zerlegen mussten, durch die sich wiederum auch noch der Zeitaufwand zur Ausführung der gesamten Datenumverteilungsoperation erhöhen konnte.

Wenn diese Option *nicht* verwendet wird, wird eine umfassende Protokollierung aller Zeilenbewegungen durchgeführt, sodass die Datenbank später bei Unterbrechungen, bei Fehlern oder bei sonstigen Bedarfssituationen wiederhergestellt werden kann.

UNIFORM
Gibt an, dass die Daten gleichmäßig auf Hashpartitionen verteilt werden (wobei für jede Hashpartition angenommen wird, dass sie dieselbe Anzahl von Zeilen hat), jedoch ist nicht jeder Datenbankpartition die gleiche Anzahl von Hashpartitionen zugeordnet. Nach der Umverteilung haben alle Datenbankpartitionen in der Datenbankpartitionsgruppe annähernd die gleiche Anzahl von Hashpartitionen.

USING DISTFILE *verteilungsdateiname*
Wenn die Verteilung der Verteilungsschlüsselwerte ungleichmäßig ist, können Sie diese Option verwenden, um eine gleichmäßige Umverteilung der Daten auf die Datenbankpartitionen einer Datenbankpartitionsgruppe zu erreichen.

Verwenden Sie die im Wert *verteilungsdateiname* angegebene Datei zur Angabe der aktuellen Verteilung von Daten auf die 4096 Hashpartitionen.

Verwenden Sie Zeilenanzahlen, Bytevolumen oder eine andere Maßeinheit, um das Datenvolumen anzugeben, das von jeder Hashpartition dargestellt wird. Das Dienstprogramm interpretiert den ganzzahligen Wert, der einer Partition zugeordnet ist, als Gewichtung dieser Partition. Wenn durch *verteilungsdateiname* eine Verteilungsdatei angegeben wird, generiert das Dienstprogramm eine Zielverteilungszuordnung, die es zu einer möglichst gleichmäßigen Umverteilung der Daten auf die Datenbankpartitionen in der Datenbankpartitionsgruppe verwendet. Nach der Umverteilung ist die Gewichtung jeder Datenbankpartition in der Datenbankpartitionsgruppe annähernd gleich (die Gewichtung der Datenbankpartition ist die Summe der Gewichtungen aller Hashpartitionen, die dieser Datenbankpartition zugeordnet sind).

Zum Beispiel könnte die Eingabeverteilungsdatei die folgenden Einträge enthalten:

```
10223
1345
112000
0
100
...
```

In dem Beispiel hat die Hashpartition 2 die Gewichtung von 112.000, während der Partition 3 (mit der Gewichtung 0) überhaupt keine Daten zugeordnet sind.

Die Verteilungsdatei *verteilungsdateiname* sollte 4096 positive ganzzahlige Werte im Zeichenformat enthalten. Die Summe der Werte darf den Wert 4.294.967.295 nicht überschreiten.

Wenn der Pfad für die Verteilungsdatei *verteilungsdateiname* nicht angegeben ist, wird das aktuelle Verzeichnis verwendet.

USING TARGETMAP *zielzuordnungsdateiname*
Die durch *zielzuordnungsdateiname* angegebene Datei wird als Zielverteilungszuordnung verwendet. Die Datenumverteilung wird entsprechend dieser Datei ausgeführt. Wenn der Pfad nicht angegeben wird, wird das aktuelle Verzeichnis verwendet.

Wenn eine Datenbankpartition, die in der Zielzuordnung enthalten ist, nicht in der Datenbankpartitionsgruppe enthalten ist, wird ein Fehler zurückgegeben. Setzen Sie eine Anweisung ALTER DATABASE PARTITION GROUP ADD DBPARTITIONNUM ab, bevor Sie den Befehl REDISTRIBUTE DATABASE PARTITION GROUP ausführen.

Wenn eine aus der Zielzuordnung ausgeschlossene Datenbankpartition in der Datenbankpartitionsgruppe enthalten *ist*, wird diese Datenbankpartition nicht in die Partitionierung mit einbezogen. Eine solche Datenbankpartition kann mit der Anweisung ALTER DATABASE PARTITION GROUP DROP DBPARTITIONNUM entweder vor oder nach der Ausführung des Befehls REDISTRIBUTE DATABASE PARTITION GROUP gelöscht werden.

CONTINUE
Setzt eine zuvor fehlgeschlagene oder gestoppte Operation REDISTRIBUTE DATABASE PARTITION GROUP fort. Wenn keine solche Operation stattgefunden hat, wird ein Fehler zurückgegeben.

ABORT
Bricht eine zuvor fehlgeschlagene oder gestoppte Operation REDISTRIBUTE DATABASE PARTITION GROUP ab. Wenn keine solche Operation stattgefunden hat, wird ein Fehler zurückgegeben.

ADD

DBPARTITIONNUM *n*

TO *m*

Durch '*n*' oder '*n* **TO** *m*' wird eine Liste von Datenbankpartitionsnummern angegeben, die in die Datenbankpartitionsgruppe einzufügen sind. Alle angegebenen Partitionen dürfen noch nicht in der Datenbankpartitionsgruppe definiert sein (SQLSTATE-Wert 42728). Dies ist mit der Ausführung der Anweisung ALTER DATABASE PARTITION GROUP mit der Klausel ADD DBPARTITIONNUM äquivalent.

DBPARTITIONNUMS *n*

TO *m*

Durch '*n*' oder '*n* **TO** *m*' wird eine Liste von Datenbankpartitionsnummern angegeben, die in die Datenbankpartitionsgruppe einzufügen sind. Alle angegebenen Partitionen dürfen noch nicht in der Datenbankpartitionsgruppe definiert sein (SQLSTATE-Wert 42728). Dies ist mit der Ausführung der Anweisung ALTER DATABASE PARTITION GROUP mit der Klausel ADD DBPARTITIONNUM äquivalent.

Anmerkung: Wenn eine Datenbankpartition mit dieser Option hinzugefügt wird, basieren die Namen von Containern für Tabellenbereiche auf den Containern des entsprechenden Tabellenbereichs in der vorhandenen Partition mit der niedrigsten Nummer in der Datenbankpartitionsgruppe. Falls dies zu einer Namensunverträglichkeit unter Containern führen würde, was möglich ist, wenn sich die neuen Partitionen auf demselben physischen System wie vorhandene Container befinden, sollte diese Option nicht verwendet werden. Stattdessen sollte die Anweisung ALTER DATABASE PARTITION GROUP mit der Option WITHOUT TABLESPACES verwendet werden, bevor der Befehl REDISTRIBUTE DATABASE

PARTITION GROUP ausgeführt wird. Tabellenbereichscontainer können anschließend manuell unter Angabe geeigneter Namen erstellt werden.

DROP

DBPARTITIONNUM *n*

TO *m*

Durch '*n*' oder '*n* **TO** *m*' wird eine Liste von Datenbankpartitionsnummern angegeben, die aus der Datenbankpartitionsgruppe zu löschen sind. Alle angegebenen Partitionen müssen in der Datenbankpartitionsgruppe definiert sein (SQLSTATE-Wert 42729). Dies ist mit der Ausführung der Anweisung ALTER DATABASE PARTITION GROUP mit der Klausel DROP DBPARTITIONNUM äquivalent.

DBPARTITIONNUMS *n*

TO *m*

Durch '*n*' oder '*n* **TO** *m*' wird eine Liste von Datenbankpartitionsnummern angegeben, die aus der Datenbankpartitionsgruppe zu löschen sind. Alle angegebenen Partitionen müssen in der Datenbankpartitionsgruppe definiert sein (SQLSTATE-Wert 42729). Dies ist mit der Ausführung der Anweisung ALTER DATABASE PARTITION GROUP mit der Klausel DROP DBPARTITIONNUM äquivalent.

TABLE *tabellenname*
Gibt eine Tabellenreihenfolge für die Verarbeitung der Umverteilung an.

ONLY Wenn der Tabellenreihenfolge das Schlüsselwort **ONLY** folgt (Standardeinstellung), werden nur die angegebenen Tabellen umverteilt. Die verbleibenden Tabellen können später durch nachfolgende Befehle REDISTRIBUTE CONTINUE verarbeitet werden. Dies ist der Standardwert.

FIRST Wenn der Tabellenreihenfolge das Schlüsselwort **FIRST** folgt, werden die angegebenen Tabellen in der angegebenen Reihenfolge umverteilt. Anschließend werden die verbleibenden Tabellen in der Datenbankpartitionsgruppe in zufälliger Reihenfolge umverteilt.

INDEXING MODE
Dieser Parameter gibt an, wie Indizes während der Umverteilung gepflegt werden, wenn die Option NOT ROLLFORWARD RECOVERABLE angegeben wird. Gültige Werte:

REBUILD
Indizes werden völlig neu erstellt. Indizes müssen bei Verwendung dieser Option nicht gültig sein. Durch diese Option werden Indizes auf der Platte zu Clustern zusammengefasst.

DEFERRED
Die REDISTRIBUTE-Operation versucht nicht, Indizes zu pflegen. Indizes werden als zu aktualisierend markiert. Die Neuerstellung solcher Indizes kann beim ersten Zugriff erzwungen werden oder erfolgt, wenn die Datenbank erneut gestartet wird.

Anmerkung: Wenn es sich nicht um MDC-Tabellen handelt und ungültige Indizes für die Tabellen vorhanden sind, werden die Indizes vom Befehl REDISTRIBUTE DATABASE PARTITION GROUP automatisch erneut erstellt, sofern Sie nicht die Option **INDEXING MODE DEFERRED** angeben. Bei einer MDC-Tabelle wird ein ungültiger zusammengesetzter Index, auch wenn Sie **INDEXING MODE DEFERRED** angeben, vor Beginn der Tabellenumverteilung erneut erstellt, weil das Dienstprogramm den zusammengesetzten Index zur Verarbeitung einer MDC-Tabelle benötigt.

DATA BUFFER *n*
Gibt die Anzahl von 4-KB-Seiten an, die als Pufferspeicher für die Datenübertragung innerhalb des Dienstprogramms zu verwenden sind. Wenn der angegebene Wert kleiner als der unterstützte Mindestwert ist, wird der Mindestwert verwendet und es wird keine Warnung zurückgegeben. Wenn für die Option DATA BUFFER kein Wert angegeben wird, wird vom Dienstprogramm zu Beginn der Verarbeitung der einzelnen Tabellen ein intelligenter Standardwert berechnet (während der Laufzeit). Insbesondere werden standardmäßig 50 % der im Zwischenspeicher für Dienstprogramme verfügbaren Speicherkapazität zu Beginn der Umverteilung einer Tabelle verwendet und darüber hinaus verschiedene Eigenschaften der Tabelle berücksichtigt.

Dieser Speicher wird direkt aus dem Zwischenspeicher für Dienstprogramme zugeordnet, dessen Größe durch den Datenbankkonfigurationsparameter **util_heap_sz** geändert werden kann. Ab Version 9.5 kann der Wert der Option DATA BUFFER des Befehls REDISTRIBUTE DATABASE PARTITION GROUP zeitweise den Wert von **util_heap_sz** übersteigen, falls im System mehr Speicher verfügbar ist.

STOP AT *iso-ortszeit*
Wenn diese Option angegeben wird, wird vor Beginn der Datenumverteilung für die einzelnen Tabellen der Wert *iso-ortszeit* mit der aktuellen lokalen Zeitmarke verglichen. Wenn die in *iso-ortszeit* angegebene Zeit gleich der Zeit der aktuellen lokalen Zeitmarke ist oder diese überschritten hat, stoppt das Dienstprogramm mit einer Warnung. Eine zur Stoppzeit in Ausführung befindliche Datenumverteilungsverarbeitung für Tabellen wird ohne Unterbrechung abgeschlossen. Es wird keine neue Datenumverteilungsverarbeitung für Tabellen gestartet. Die nicht verarbeiteten Tabellen können mit der Option **CONTINUE** umverteilt werden. Der Wert in *iso-ortszeit* wird als Zeitmarke angegeben, das heißt als 7-teilige Zeichenfolge, die eine Kombination aus Datum und Uhrzeit darstellt. Das Format ist *jjjj-mm-tt-hh.mm.ss.nnnnnn* (Jahr, Monat, Tag, Stunde, Minuten, Sekunden, Mikrosekunden) für die Ortszeit.

STATISTICS (Statistikdaten)
Diese Option gibt an, dass das Dienstprogramm Statistiken für die Tabellen erfassen soll, die ein Statistikprofil haben. Die Angabe dieser Option ist effizienter als eine separate Ausführung des Befehls RUNSTATS nach Abschluss der Datenumverteilung.

USE PROFILE
Statistiken werden für die Tabellen mit einem Statistikprofil erfasst. Für Tabellen ohne Statistikprofil erfolgt keine Statistikerfassung. Dies ist der Standardwert.

NONE
Es werden keine Statistiken für Tabellen erfasst.

Beispiele: Schritte für die Datenumverteilung

Es ist möglicherweise erforderlich, einen Knoten der Knotengruppe hinzuzufügen oder aus der Knotengruppe zu löschen. Die nachfolgenden Schritte erläutern, wie einer Knotengruppe ein neuer Knoten hinzugefügt und die Daten umverteilt werden. Die neu hinzugefügte Datenbankpartition ist nicht in der Verteilungszuordnung enthalten, jedoch wurden die Container für die Tabellenbereiche in der Datenbankpartitionsgruppe erstellt. Die Datenbankpartition wird der Verteilungszuordnung hinzugefügt, wenn eine Operation REDISTRIBUTE DATABASE PARTITION GROUP erfolgreich ausgeführt wurde.

1. Ermitteln Sie die Knotengruppen, für die eine Umverteilung erforderlich wird. In diesem Dokument lautet der Name der umzuverteilenden Knotengruppe „sampleNodegrp".
2. Ermitteln Sie die Objekte, die inaktiviert oder entfernt werden sollten, bevor die Umverteilung erfolgt.
   a. Replizierte MQTs: Dieser Typ von MQT (Materialized Query Table) wird durch das Dienstprogramm REDISTRIBUTE nicht unterstützt. Solche MQTs müssen vor der Ausführung von REDISTRIBUTE gelöscht und anschließend erneut erstellt werden.

      ```
      SELECT TABSCHEMA, TABNAME FROM SYSCAT.TABLES WHERE PARTITION_MODE = 'R'
      ```

   b. Ereignismonitore mit der Klausel WRITE TO TABLE ('in Tabelle schreiben'): Sie sollten alle automatisch aktivierten Ereignismonitore mit der Klausel WRITE TO TABLE inaktivieren, die eine Tabelle in der umzuverteilenden Datenbankpartitionsgruppe aufweisen.

      ```
      SELECT DISTINCT EVMONNAME FROM SYSCAT.EVENTTABLES E
      JOIN SYSCAT.TABLES T ON T.TABNAME = E.TABNAME AND T.TABSCHEMA = E.TABSCHEMA
      JOIN SYSCAT.TABLESPACES S ON S.TBSPACE = T.TBSPACE AND S.NGNAME = 'sampleNodegrp'
      ```

   c. EXPLAIN-Tabellen: Es wird empfohlen, die EXPLAIN-Tabellen in nur einer Partitionsknotengruppe zu erstellen. Wenn sie jedoch in einer Knotengruppe definiert werden, die eine Umverteilung erfordert, können Sie in Betracht ziehen, die Tabellen vor der Umverteilung zu löschen und nach Abschluss der Umverteilung erneut zu definieren, sofern die bisher generierten Daten nicht beibehalten werden müssen.
   d. Zugriffsmodus und Ladestatus von Tabellen: Stellen Sie sicher, dass sich alle Tabellen in den umzuverteilenden Knotengruppen im Vollzugriffsmodus befinden und keinen Status mit anstehender oder in Bearbeitung befindlicher Ladeoperation haben.

      ```
      SELECT DISTINCT TRIM(T.OWNER) || \'.\' || TRIM(T.TABNAME) AS NAME, T.ACCESS_MODE, A.LOAD_STATUS
       FROM SYSCAT.TABLES T, SYSCAT.DBPARTITIONGROUPS N, SYSIBMADM.ADMINTABINFO A
       WHERE T.PMAP_ID = N.PMAP_ID
        AND A.TABSCHEMA = T.OWNER
        AND A.TABNAME = T.TABNAME
        AND N.DBPGNAME = 'sampleNodegrp'
        AND (T.ACCESS_MODE <> 'F' OR A.LOAD_STATUS IS NOT NULL)
      ```

   e. Statistikprofile: Tabellenstatistiken können im Rahmen des Umverteilungsprozesses aktualisiert werden, wenn ein Statistikprofil für die Tabelle definiert ist. Wenn Sie das Dienstprogramm REDISTRIBUTE zur Aktualisierung der Statistiken einer Tabelle verwenden, sinkt das benötigte E/A-Volumen, weil alle Daten für die Umverteilung durchsucht werden und kein zusätzliches Durchsuchen der Daten für das Dienstprogramm RUNSTATS erforderlich ist.

      ```
      RUNSTATS on table schema.tabelle USE PROFILE runstats-profil SET PROFILE ONLY
      ```

3. Überprüfen Sie die Datenbankkonfiguration. Der Wert des Parameters **util_heap_sz** ist für die Verarbeitung der Datenversetzung zwischen Datenbankpartitionen von kritischer Bedeutung. Ordnen Sie durch den Parameter **util_heap_sz** eine möglichst hohe Speicherkapazität für die Dauer der Umverteilung

zu. Wenn im Rahmen der Umverteilung eine Indexneuerstellung erfolgt, ist genügend Sortierspeicher (Parameter **sortheap**) erforderlich. Erhöhen Sie die Werte der Parameter **util_heap_sz** und **sortheap** nach Bedarf, um die Umverteilungsleistung zu verbessern.

4. Rufen Sie die Datenbankkonfigurationseinstellungen ab, die für die neuen Datenbankpartitionen verwendet werden sollen. Wenn Datenbankpartitionen hinzugefügt werden, wird eine Standarddatenbankkonfiguration verwendet. Daher ist es wichtig, die Datenbankkonfiguration auf den neuen Knoten zu aktualisieren, bevor der Befehl REDISTRIBUTE ausgeführt wird, um sicherzustellen, dass die Konfiguration über das gesamte Data-Warehouse gleichmäßig verteilt wird.

```
SELECT NAME,
CASE WHEN DEFERRED_VALUE_FLAGS = 'AUTOMATIC' THEN DEFERRED_VALUE_FLAGS ELSE SUBSTR(DEFERRED_VALUE,1,20) END AS DEFERRED_VALUE
FROM SYSIBMADM.DBCFG
WHERE DBPARTITIONNUM = existing-node
AND DEFERRED_VALUE != ''
AND NAME NOT IN ('hadr_local_host','hadr_local_svc','hadr_peer_window',
          'hadr_remote_host','hadr_remote_inst','hadr_remote_svc',
          'hadr_syncmode','hadr_timeout','backup_pending',
          'codepage','codeset','collate_info','country',
          'database_consistent','database_level',
          'hadr_db_role','log_retain_status',
          'loghead','logpath','multipage_alloc','numsegs',
          'pagesize','release','restore_pending','restrict_access',
          'rollfwd_pending','territory','user_exit_status','number_compat',
          'varchar2_compat','database_memory')
```

5. Führen Sie ein Backup der Datenbank (oder der Tabellenbereiche in Knotengruppen, die umverteilt werden sollen) aus, bevor Sie den Umverteilungsprozess starten, um einen möglichst aktuellen Wiederherstellungspunkt sicherzustellen.
6. Definieren Sie die neuen Daten-BCUs (BCU - Balanced Configuration Unit) in DB2, indem Sie die Datei `'db2nodes.cfg'` aktualisieren und die Datenbankpartitionsspezifikationen für die neuen Daten-BCUs hinzufügen. Definieren Sie die neuen Datenbankpartitionen in DB2 mithilfe des Befehls `ADD NODE WITHOUT TABLESPACES`.

   ```
   db2start nodenum x export DB2NODE=x
   db2 add node without tablespaces
   db2stop nodenum x
   ```

   Anmerkung: Wenn es sich nicht um den ersten logischen Port auf der Daten-BCU handelt, führen Sie vor und nach der oben angegebenen Befehlsfolge für nachfolgende logische Ports zunächst ein Starten und ein Stoppen der ersten logischen Portnummer aus.
7. Definieren Sie Tabellenbereichscontainer für temporäre Systemtabellen in den neu definierten Datenbankpartitionen.

   ```
   ALTER TABLESPACE tabellenbereichsname ADD containerinformationen ON DBPARTITIONNUMS (x to y)
   ```
8. Fügen Sie die neuen logischen Datenbankpartitionen den Datenbankpartitionsgruppen hinzu, die sich über die Daten-BCUs erstrecken.

   ```
   ALTER DATABASE PARTITION GROUP partitionsgruppenname ADD dbpartitionnums (x to y) WITHOUT TABLESPACES
   ```
9. Definieren Sie Tabellenbereichscontainer für permanente Daten in den neu definierten Datenbankpartitionen.

   ```
   ALTER TABLESPACE tabellenbereichsname ADD containerinformationen ON DBPARTITIONNUMS (x to y)
   ```
10. Wenden Sie die Datenbankkonfigurationseinstellungen, die Sie in Schritt 4 abgerufen haben, auf die neuen Datenbankpartitionen an (oder führen Sie einen einzelnen Befehl UPDATE DB CFG für alle Datenbankpartitionen unter Verwendung der neuen Unterstützung in DB2 9.5 zur Anzeige aller Partitionskonfigurationen in einer einzigen Sicht aus).

11. Erfassen Sie die Definitionen aller replizierten MQTs, die in den Datenbankpartitionsgruppen vorhanden sind, die umverteilt werden sollen, und löschen Sie diese.

    ```
    db2look -d dbname -e -z schema -t replizierte_MQT-namen -o repMQTs.clp
    ```

12. Inaktivieren Sie alle Ereignismonitore mit der Klausel WRITE TO TABLE, die in den Datenbankpartitionsgruppen, die umverteilt werden sollen, vorhanden sind.

    ```
    SET EVENT MONITOR monitorname STATE 0
    ```

13. Führen Sie das Dienstprogramm REDISTRIBUTE aus, um Daten gleichmäßig auf alle Datenbankpartitionen umzuverteilen. Das folgende Beispiel zeigt einen einfachen Befehl REDISTRIBUTE:

    ```
    REDISTRIBUTE DATABASE PARTITION GROUP sampleNodegrp NOT ROLLFORWARD RECOVERABLE uniform;
    ```

Der Benutzer sollte außerdem in Betracht ziehen, eine Tabellenliste als Eingabe für den Befehl REDISTRIBUTE anzugeben, um die Reihenfolge sicherzustellen, in der die Tabellen verarbeitet werden. Das Dienstprogramm REDISTRIBUTE versetzt die Daten (komprimierte und zusammengefasste). Optional werden Indizes erneut erstellt und Statistiken aktualisiert, wenn Statistikprofile definiert sind. Daher kann anstelle des vorherigen Befehls das folgende Script ausgeführt werden:

```
REDISTRIBUTE DATABASE PARTITION GROUP sampleNodegrp
  NOT ROLLFORWARD RECOVERABLE uniform TABLE (tabelle1, tabelle2,...) FIRST;
```

Konsequenzen der Option NOT ROLLFORWARD RECOVERABLE

Wenn der Befehl REDISTRIBUTE DATABASE PARTITION GROUP mit Angabe der Option NOT ROLLFORWARD RECOVERABLE ausgeführt wird, wird eine Strategie der minimalen Protokollierung angewendet, bei der das Schreiben von Protokollsätzen für jede versetzte Zeile auf ein Minimum reduziert wird. Dieser Typ von Protokollierung ist ein wesentlicher Aspekt der Benutzerfreundlichkeit der Umverteilungsoperation, da eine Methode, bei der alle Datenbewegungen vollständig protokolliert werden, für große Systeme einen praktisch nicht realisierbaren Bedarf an aktivem und permanentem Speicherplatz sowie allgemein schlechtere Leistungsmerkmale zur Folge haben könnte. Es ist jedoch wichtig, dass sich Benutzer bewusst sind, dass der Befehl REDISTRIBUTE DATABASE PARTITION GROUP aufgrund dieses Minimalprotokollierungsmodells *nicht* durch eine aktualisierende Recovery wiederherstellbar ist. Dies bedeutet, dass jede Operation, die dazu führt, dass für die Datenbank eine aktualisierende Recovery für einen Zeitraum ausgeführt wird, in dem eine Umverteilungsoperation stattgefunden hat, zur Folge hat, dass alle Tabellen, die von der Umverteilungsoperation betroffen waren, in einem nicht verfügbaren Status zurückbleiben. Solche Tabellen können nur gelöscht werden. Das heißt, es gibt keine Möglichkeit, die Daten in diesen Tabellen wiederherzustellen. Dies ist der Grund, aus dem das Dienstprogramm REDISTRIBUTE DATABASE PARTITION GROUP bei Ausführung mit der Option NOT ROLLFORWARD RECOVERABLE für wiederherstellbare Datenbanken alle Tabellenbereiche, auf die es zugreift, in den Status 'Backup anstehend' versetzt und so den Benutzer zwingt, alle umverteilten Tabellenbereiche nach Abschluss einer erfolgreichen Umverteilungsoperation durch ein Backup zu sichern. Wenn nach der Umverteilungsoperation ein Backup erstellt wird, sollte keine Notwendigkeit mehr bestehen, eine aktualisierende Recovery für eben den Zeitraum durchzuführen, in dem die Umverteilungsoperation stattgefunden hat.

Die fehlende Wiederherstellbarkeit durch eine aktualisierende Recovery des Dienstprogramms REDISTRIBUTE hat eine sehr wichtige Konsequenz, die der Benutzer beachten sollte: Wenn der Benutzer zulässt, dass Aktualisierungen an Tabellen in der Datenbank (auch an Tabellen außerhalb der Datenbankpartitionsgruppe, die

umverteilt wird) vorgenommen werden, während die Umverteilungsoperation aktiv ist, einschließlich des Zeitraums nach der Umverteilung, in dem ein Backup der von der Umverteilung betroffenen Tabellenbereiche vom Benutzer durchgeführt wird, können solche Aktualisierungen bei einem schwer wiegenden Fehler, zum Beispiel einer Beschädigung eines Datenbankcontainers, verloren gehen. Der Grund, dass solche Aktualisierungen verloren gehen können, liegt darin, dass die Umverteilungsoperation nicht durch aktualisierende Recovery wiederherstellbar ist. Wenn es erforderlich wird, eine Recovery der Datenbank von einem Backup, das vor der Umverteilungsoperation erstellt wurde, durchzuführen, ist es nicht möglich, die aktualisierende Recovery durch die Protokolle hindurch durchzuführen, um die Aktualisierungen zu wiederholen, die während der Umverteilungsoperation stattgefunden haben, ohne auch eine aktualisierende Recovery durch die Zeit der Umverteilung hindurch durchzuführen, die, wie oben erläutert, die umverteilten Tabellen in einem nicht verfügbaren Status zurücklässt. In einer solchen Situation kann daher nur ein Restore der Datenbank ohne aktualisierende Recovery von einem Backup, das vor der Umverteilung erstellt wurde, durchgeführt werden. Anschließend kann die Umverteilungsoperation erneut ausgeführt werden. Allerdings gehen alle Aktualisierungen, die während der ursprünglichen Umverteilungsoperation stattgefunden haben, auf diese Weise verloren.

Die Wichtigkeit dieses Punkts kann nicht deutlich genug hervorgehoben werden. Um sicherzustellen, dass während der Umverteilungsoperation keine Aktualisierungen verloren gehen können, muss eine der folgenden Voraussetzungen erfüllt werden:

- Der Benutzer vermeidet Aktualisierungen während der Operation des Befehls REDISTRIBUTE DATABASE PARTITION GROUP sowie während des Zeitraums nach Abschluss des Befehls, in dem ein Backup der betroffenen Tabellenbereiche durchgeführt wird.
- Aktualisierungen, die während der Umverteilungsoperation angewendet werden, stammen aus einer reproduzierbaren Quelle. Das heißt, sie können jederzeit erneut angewendet werden. Wenn die Quelle von Aktualisierungen zum Beispiel in einer Datei gespeicherte Daten sind und die Aktualisierungen während einer Stapelverarbeitung angewendet wurden, gehen diese Aktualisierungen selbst bei einem Fehler, der einen Restore der Datenbank erfordert, nicht verloren, da sie später einfach erneut angewendet werden können.

Im Hinblick auf die Zulässigkeit von Aktualisierungen an der Datenbank während der Umverteilungsoperation muss der Benutzer entscheiden, ob solche Aktualisierungen für sein Szenario geeignet sind, indem er berücksichtigt, ob sich die Aktualisierungen nach einem Restore der Datenbank bei Bedarf wiederholen lassen.

Anmerkung: Nicht jeder Fehler während der Ausführung des Befehls REDISTRIBUTE DATABASE PARTITION GROUP führt zu diesem Problem. Tatsächlich ist dies bei den meisten Fehlern nicht der Fall. Der Befehl REDISTRIBUTE DATABASE PARTITION GROUP ist vollständig wieder anlauffähig. Das bedeutet, dass das Dienstprogramm bei einem Fehler mitten in der Verarbeitung mithilfe der Optionen **CONTINUE** oder **ABORT** problemlos fortgesetzt bzw. abgebrochen werden kann. Die oben beschriebenen Probleme beziehen sich auf Situationen, in denen der Benutzer einen Restore von einem Backup durchführen muss, das vor der Umverteilungsoperation erstellt wurde.

Hinweise

- Stellen Sie vor dem Starten einer Umverteilungsoperation sicher, dass die Tabellen sich im Normalstatus befinden, nicht im Status Load Pending ('Laden

anstehend') oder REORG Pending ('REORG anstehend'). Der Status von Tabellen kann mithilfe des Befehls LOAD QUERY überprüft werden.

- Wenn die Option **NOT ROLLFORWARD RECOVERABLE** angegeben wird und die Datenbank eine wiederherstellbare Datenbank ist, wird ein Tabellenbereich, wenn das Dienstprogramm zum ersten Mal auf ihn zugreift, in den Status 'Backup anstehend' versetzt. Alle Tabellen in diesem Tabellenbereich werden schreibgeschützt, bis ein Backup des Tabellenbereichs durchgeführt wird. Das Backup kann erst stattfinden, wenn die Umverteilung aller Tabellen in diesem Tabellenbereich abgeschlossen ist.
- Wenn eine Umverteilungsoperation aktiv ist, generiert sie eine Ereignisprotokolldatei, die allgemeine Informationen zur Umverteilungsoperation und Informationen wie zum Beispiel zur Start- und Endzeit der Verarbeitung der einzelnen Tabellen enthält. Diese Ereignisprotokolldatei wird an folgende Position geschrieben:
  - Das Verzeichnis `homeinst/sqllib/redist` auf Linux- und UNIX-Betriebssystemen mit dem folgenden Format für Unterverzeichnisse und Dateiname: *datenbankname*`.`*datenbankpartitionsgruppenname*`.`*zeitmarke*`.log`.
  - Das Verzeichnis **DB2INSTPROF**`\`*instanz*`\redist` unter Windows-Betriebssystemen (**DB2INSTPROF** ist der Wert der Registrierdatenbankvariablen **DB2INSTPROF**), wobei das folgende Format für Unterverzeichnisse und den Dateinamen verwendet wird: *datenbankname*`.`*dbpartitionsgruppenname*`.`*zeitmarke*`.log`.
  - Der Wert der Zeitmarke gibt den Zeitpunkt an, zu dem der Befehl abgesetzt wurde.

  Weitere Informationen zum Ereignisprotokoll der Umverteilung finden Sie im Abschnitt „Ereignisprotokolldatei für die Umverteilung".
- Dieses Dienstprogramm führt während der Verarbeitung zwischendurch Commitoperationen aus.
- Alle Pakete, die eine Abhängigkeit von einer Tabelle haben, die umverteilt wurde, werden ungültig gemacht. Es wird empfohlen, solche Pakete nach Abschluss der Umverteilung der Datenbankpartitionsgruppe explizit mit REBIND erneut zu binden. Durch den expliziten Rebind wird die Anfangsverzögerung der ersten SQL-Anforderung für das ungültige Paket vermieden. Die Nachrichtendatei der Umverteilung enthält eine Liste aller Tabellen, die umverteilt wurden.
- Standardmäßig aktualisiert das Dienstprogramm REDISTRIBUTE die Statistiken für die Tabellen, die ein Statistikprofil haben. Für Tabellen ohne Statistikprofil wird empfohlen, die Tabellen- und Indexstatistiken durch Aufrufen der API db2Runstats oder durch Ausführen des Befehls RUNSTATS nach Abschluss der Umverteilungsoperation separat zu aktualisieren.
- Datenbankpartitionsgruppen, die replizierte MQTs (Materialized Query Tables) oder Tabellen mit definierter Option DATA CAPTURE CHANGES enthalten, können nicht umverteilt werden.
- Eine Umverteilung ist nicht zulässig, wenn Tabellenbereiche für temporäre Benutzertabellen mit vorhandenen deklarierten temporären Tabellen in der Datenbankpartitionsgruppe enthalten sind.
- Optionen wie **INDEXING MODE** werden für Tabellen, auf die sie nicht anwendbar sind, ohne Warnung ignoriert. Zum Beispiel wird **INDEXING MODE** für Tabellen ignoriert, für die keine Indizes definiert sind.

Kompatibilitäten

Aus Gründen der Kompatibilität mit Versionen vor Version 8 besteht folgende Möglichkeit:

- Das Schlüsselwort **NODEGROUP** kann anstelle von **DATABASE PARTITION GROUP** verwendet werden.

db2nchg - Konfiguration des Datenbankpartitionsservers ändern

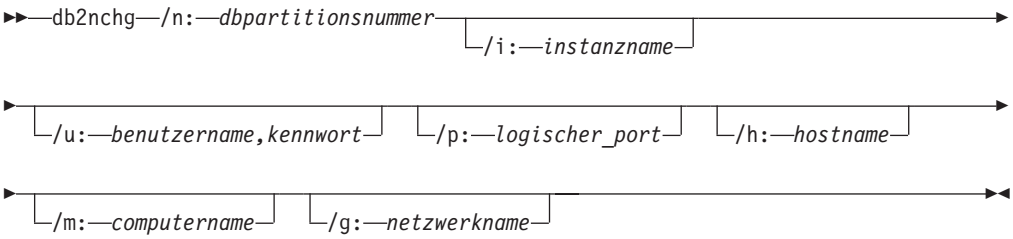
Ändert die Konfiguration des Datenbankpartitionsservers. Dazu gehören das Versetzen des Datenbankpartitionsservers (Knoten) von einem System auf ein anderes, das Ändern des TCP/IP-Hostnamens des Systems und das Auswählen einer anderen logischen Portnummer bzw. eines anderen Netzwerknamens für den Datenbankpartitionsserver (Knoten). Dieser Befehl kann nur verwendet werden, wenn der Datenbankpartitionsserver gestoppt ist.

Dieser Befehl steht nur unter Windows-Betriebssystemen zur Verfügung.

Berechtigung

Lokaler Administrator

Befehlssyntax



Befehlsparameter

/n:*dbpartitionsnummer*
Gibt die Datenbankpartitionsnummer der Konfiguration des Datenbankpartitionsservers an, die zu ändern ist.

/i:*instanzname*
Gibt die Instanz an, an der dieser Datenbankpartitionsserver beteiligt ist. Wenn kein Parameter angegeben wird, wird standardmäßig die aktuelle Instanz angenommen.

/u:*benutzername,kennwort*
Gibt den Benutzernamen und das Kennwort an. Wenn kein Parameter angegeben wird, gelten standardmäßig der vorhandene Benutzername und das vorhandene Kennwort.

/p:*logischer_port*
Gibt den logischen Port für den Datenbankpartitionsserver an. Dieser Parameter muss angegeben werden, um den Datenbankpartitionsserver auf ein anderes System zu versetzen. Wenn dieser Parameter nicht angegeben wird, bleibt die logische Portnummer unverändert.

/h:*hostname*
Gibt den TCP/IP-Hostnamen an, der von FCM zur internen Kommunikation verwendet wird. Wenn dieser Parameter nicht angegeben wird, bleibt der Hostname gleich.

/m:*computername*
Gibt den Computer an, auf dem sich der Datenbankpartitionsserver befinden soll. Der Datenbankpartitionsserver kann nur dann versetzt werden, wenn die Instanz keine bereits definierten Datenbanken enthält.

/g:*netzwerkname*
Ändert den Netzwerknamen des Datenbankpartitionsservers. Dieser Parameter kann dazu verwendet werden, eine bestimmte IP-Adresse auf den Datenbankpartitionsserver anzuwenden, wenn auf einem Computer mehrere IP-Adressen verfügbar sind. Es kann der Netzwerkname oder die IP-Adresse eingegeben werden.

Beispiele

Der im folgenden Beispiel gezeigte Befehl ändert den logischen Port, der Datenbankpartition 2 zugeordnet ist, die an der Instanz TESTMPP beteiligt ist, in den logischen Port 3:

```
db2nchg /n:2 /i:TESTMPP /p:3
```

db2ncrt - Datenbankpartitionsserver einer Instanz hinzufügen

Fügt einer Instanz einen Datenbankpartitionsserver (Knoten) hinzu.

Dieser Befehl steht nur unter Windows-Betriebssystemen zur Verfügung.

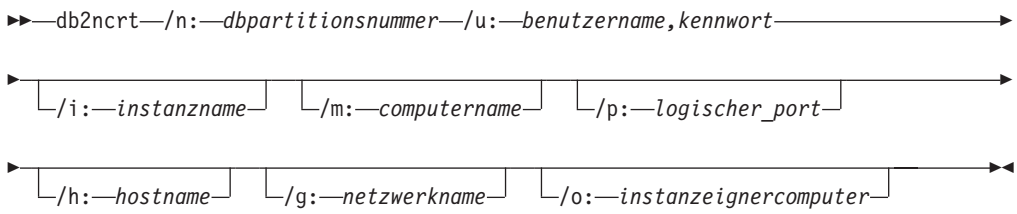
Geltungsbereich

Wenn ein Datenbankpartitionsserver einem Computer hinzugefügt wird, auf dem bereits eine Instanz vorhanden ist, wird der Datenbankpartitionsserver dem Computer als logischer Datenbankpartitionsserver hinzugefügt. Wenn ein Datenbankpartitionsserver einem Computer hinzugefügt wird, auf dem keine Instanz vorhanden ist, wird die Instanz hinzugefügt und der Computer wird zu einem neuen physischen Datenbankpartitionsserver. Dieser Befehl sollte nicht verwendet werden, wenn in einer Instanz Datenbanken vorhanden sind. Stattdessen sollte der Befehl START DATABASE MANAGER mit der Option ADD DBPARTITIONNUM abgesetzt werden. Dadurch wird sichergestellt, dass die Datenbank dem neuen Datenbankpartitionsserver ordnungsgemäß hinzugefügt wird. Es ist auch möglich, einen Datenbankpartitionsserver einer Instanz hinzuzufügen, in der bereits eine Datenbank erstellt wurde. Die Datei `db2nodes.cfg` darf nicht bearbeitet werden, da eine Änderung der Datei Inkonsistenzen in der Umgebung mit partitionierten Datenbanken zur Folge haben könnte.

Berechtigung

Berechtigung eines `lokalen Administrators` auf dem Computer, auf dem der Datenbankpartitionsserver hinzugefügt wird.

Befehlssyntax



Befehlsparameter

/n:*dbpartitionsnummer*
Eine eindeutige Datenbankpartitionsnummer, die den Datenbankpartitionsserver angibt. Es kann eine Nummer aus dem Bereich von 1 bis 999 eingegeben werden.

/u:*benutzername,kennwort*
Gibt den Namen des Anmeldekontos und das Kennwort für DB2 an.

/i:*instanzname*
Gibt den Instanznamen an. Wenn kein Parameter angegeben wird, wird standardmäßig die aktuelle Instanz angenommen.

/m:*computername*
Gibt den Computernamen der Windows-Workstation an, auf der sich der Datenbankpartitionsserver befindet. Dieser Parameter ist erforderlich, wenn ein Datenbankpartitionsserver auf einem fernen Computer hinzugefügt wird.

/p:*logischer_port*
Gibt die logische Portnummer an, die für den Datenbankpartitionsserver verwendet wird. Wenn dieser Parameter nicht angegeben wird, wird der logischen Portnummer der Wert 0 zugeordnet. Bei der Erstellung eines logischen Datenbankpartitionsservers muss dieser Parameter angegeben werden und es muss eine logische Portnummer ausgewählt werden, die nicht im Gebrauch ist. Beachten Sie die folgenden Einschränkungen:

- Jeder Computer muss einen Datenbankpartitionsserver haben, der den logischen Port 0 hat.
- Die Portnummer darf den für die FCM-Kommunikation im Verzeichnis `x:\winnt\system32\drivers\etc\` reservierten Portbereich nicht überschreiten. Wenn zum Beispiel ein Bereich von vier Ports für die aktuelle Instanz reserviert ist, ist die höchste Portnummer 3. Port 0 wird für den logischen Standarddatenbankpartitionsserver verwendet.

/h:*hostname*
Gibt den TCP/IP-Hostnamen an, der von FCM zur internen Kommunikation verwendet wird. Dieser Parameter ist erforderlich, wenn ein Datenbankpartitionsserver auf einem fernen Computer hinzugefügt wird.

/g:*netzwerkname*
Gibt den Netzwerknamen für den Datenbankpartitionsserver an. Wenn kein Parameter angegeben wird, wird standardmäßig die erste IP-Adresse verwendet, die auf dem System festgestellt wird. Dieser Parameter kann dazu verwendet werden, eine bestimmte IP-Adresse auf den Datenbankpartitionsserver anzuwenden, wenn auf einem Computer mehrere IP-Adressen verfügbar sind. Es kann der Netzwerkname oder die IP-Adresse eingegeben werden.

/o:*instanzeignercomputer*
Gibt den Computernamen des Computers an, der Eigner der Instanz ist. Der Standardwert ist der lokale Computer. Dieser Parameter ist erforderlich, wenn der Befehl db2ncrt auf einem Computer aufgerufen wird, der nicht der Instanzeignercomputer ist.

Beispiele

Das folgende Beispiel zeigt einen Befehl, mit dem ein neuer Datenbankpartitionsserver der Instanz TESTMPP auf dem Instanzeignercomputer SHAYER hinzugefügt wird, wobei der neue Datenbankpartitionsserver als Datenbankpartition 2 bekannt ist und den logischen Port 1 verwendet:

```
db2ncrt /n:2 /u:QBPAULZ\paulz,g1reeky /i:TESTMPP /m:TEST /p:1 /o:SHAYER /h:TEST
```

db2ndrop - Datenbankpartitionsserver aus einer Instanz löschen

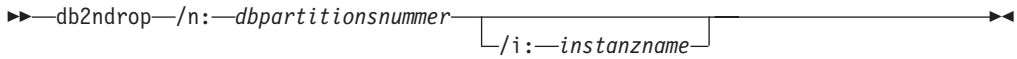
Löscht einen Datenbankpartitionsserver (Knoten) aus einer Instanz, in der keine Datenbanken definiert sind. Wenn ein Datenbankpartitionsserver gelöscht wird, kann die Datenbankpartitionsnummer für einen neuen Datenbankpartitionsserver verwendet werden. Dieser Befehl kann nur verwendet werden, wenn der Datenbankpartitionsserver gestoppt ist.

Dieser Befehl steht nur unter Windows-Betriebssystemen zur Verfügung.

Berechtigung

Berechtigung eines `lokalen Administrators` auf dem System, auf dem der Datenbankpartitionsserver gelöscht werden soll.

Befehlssyntax



Befehlsparameter

/n:*dbpartitionsnummer*
Eine eindeutige Datenbankpartitionsnummer, die den Datenbankpartitionsserver angibt.

/i:*instanzname*
Gibt den Instanznamen an. Wenn kein Parameter angegeben wird, wird standardmäßig die aktuelle Instanz angenommen.

Beispiele

```
db2ndrop /n:2 /i=KMASCI
```

Hinweise

Wenn der Instanzeigner-Datenbankpartitionsserver (Datenbankpartitionsnummer 0) aus der Instanz gelöscht wird, wird die Instanz dadurch unbrauchbar. Verwenden Sie zum Löschen der Instanz den Befehl db2idrop.

Dieser Befehl sollte nicht verwendet werden, wenn sich Datenbanken in der Instanz befinden. Stattdessen sollte der Befehl `db2stop drop nodenum` verwendet werden. Dadurch wird sichergestellt, dass der Datenbankpartitionsserver ordnungsgemäß aus der Umgebung mit partitionierten Datenbanken entfernt wird. Es ist auch möglich, einen Datenbankpartitionsserver in einer Instanz zu löschen, in der eine Datenbank vorhanden ist. Die Datei `db2nodes.cfg` darf nicht bearbeitet werden, da eine Änderung der Datei Inkonsistenzen in der Umgebung mit partitionierten Datenbanken zur Folge haben könnte.

Wenn ein Datenbankpartitionsserver, der dem logischen Port 0 zugeordnet ist, von einem System gelöscht werden soll, auf dem mehrere logische Datenbankpartitionsserver ausgeführt werden, müssen zunächst alle anderen Datenbankpartitionsserver gelöscht werden, die den anderen logischen Ports zugeordnet sind. Jeder Datenbankpartitionsserver muss einen Datenbankpartitionsserver haben, der dem logischen Port 0 zugeordnet ist.

Kapitel 31. SQL-Sprachelemente

Datentypen

Datenbankpartitionskompatible Datentypen

Die *Datenbankpartitionskompatibilität* wird zwischen den Basisdatentypen entsprechender Spalten von Verteilungsschlüsseln definiert. Datenbankpartitionskompatible Datentypen haben die Eigenschaft, dass zwei Variablen mit jeweils einem dieser Datentypen und dem gleichen Wert durch eine bestimmte Partitionierungsfunktion auf denselben Verteilungszuordnungsindex abgebildet werden.

Tabelle 41 auf Seite 442 zeigt die Kompatibilität von Datentypen in Datenbankpartitionen.

Die Datenbankpartitionskompatibilität besitzt folgende Merkmale:

- Für die Datentypen DATE, TIME und TIMESTAMP werden interne Formate verwendet. Sie sind untereinander nicht kompatibel, und keiner dieser Typen ist mit Zeichen- oder Grafikdatentypen kompatibel.
- Die Partitionskompatibilität wird nicht von der Optionalität der Dateneingabe für eine Spalte beeinflusst.
- Die Partitionskompatibilität wird von der Sortierfolge beeinflusst. Die localeabhängigen, UCA-basierten Sortierfolgen benötigen eine exakte Übereinstimmung in der Sortierfolge, das Attribut 'Strength (S)' der Sortierfolge wird hierbei ignoriert. Alle anderen Sortierfolgen werden im Hinblick auf das Feststellen der Partitionskompatibilität als identisch betrachtet.
- Zeichenspalten, die mit FOR BIT DATA definiert wurden, sind nur dann mit Zeichenspalten ohne FOR BIT DATA kompatibel, wenn eine andere Sortierfolge als eine localeabhängige, UCA-basierte Sortierfolge verwendet wird.
- Nullwerte (NULL) kompatibler Datentypen werden auf identische Weise behandelt. Für Nullwerte nicht kompatibler Datentypen werden möglicherweise unterschiedliche Ergebnisse erzielt.
- Bei benutzerdefinierten Datentypen (UDT) wird der Basisdatentyp zur Analyse der Datenbankpartitionskompatibilität verwendet.
- Dezimalzahlen desselben Werts im Verteilungsschlüssel werden gleich behandelt, auch wenn ihre Anzahl an Kommastellen und ihre Genauigkeit unterschiedlich sind.
- Folgende Leerzeichen in Zeichenfolgen (CHAR, VARCHAR, GRAPHIC oder VARGRAPHIC) werden von der im System bereitgestellten Hashfunktion ignoriert.
- Wird eine localeabhängige, UCA-basierte Sortierfolge verwendet, sind CHAR, VARCHAR, GRAPHIC und VARGRAPHIC kompatible Datentypen. Werden andere Sortierfolgen verwendet, sind CHAR und VARCHAR kompatible Typen und GRAPHIC und VARGRAPHIC sind kompatible Typen, aber CHAR und VARCHAR sind keine mit GRAPHIC und VARGRAPHIC kompatiblen Typen. Die Datentypen CHAR oder VARCHAR unterschiedlicher Längen sind kompatible Datentypen.

- Gleiche DECFLOAT-Werte werden identisch behandelt, selbst wenn ihre Genauigkeit unterschiedlich ist. DECFLOAT-Werte, die numerisch gleich sind, werden identisch behandelt, auch wenn sie unterschiedlich viele signifikante Ziffern haben.

Tabelle 41. Datenbankpartitionskompatibilitäten

Operanden	Binäre Ganzzahl	Dezimalzahl	Gleitkommazahl	Dezimale Gleitkommazahl	Zeichenfolge	Grafikzeichenfolge	Datum	Zeit	Zeitmarke	Einzigartiger Datentyp	Strukturierter Typ
Binäre Ganzzahl	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	[1]	Nein
Dezimalzahl	Nein	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	[1]	Nein
Gleitkommazahl	Nein	Nein	Ja	Nein	Nein	Nein	Nein	Nein	Nein	[1]	Nein
Dezimale Gleitkommazahl	Nein	Nein	Nein	Ja	Nein	Nein	Nein	Nein	Nein	[1]	Nein
Zeichenfolge[4]	Nein	Nein	Nein	Nein	Ja[2]	2, 3	Nein	Nein	Nein	[1]	Nein
Grafikzeichenfolge[4]	Nein	Nein	Nein	Nein	2, 3	Ja[2]	Nein	Nein	Nein	[1]	Nein
Datum	Nein	Nein	Nein	Nein	Nein	Nein	Ja	Nein	Nein	[1]	Nein
Zeit	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Ja	Nein	[1]	Nein
Zeitmarke	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Ja	[1]	Nein
Einzigartiger Datentyp	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	Nein
Strukturierter Typ[4]	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein

Anmerkung:

[1] Ein benutzerdefinierter einzigartiger Datentyp (UDT) ist datenbankpartitionskompatibel mit dem Quellentyp des UDT oder jedes anderen UDT mit einem datenbankpartitionskompatiblen Quellentyp.

[2] Zeichen- und Grafikzeichenfolgetypen sind kompatibel, wenn sie kompatible Sortierfolgen haben.

[3] Zeichen- und Grafikzeichenfolgetypen sind kompatibel, wenn eine localeabhängige, UCA-basierte Sortierfolge verwendet wird. Ansonsten sind sie keine kompatible Typen.

[4] Benutzerdefinierte strukturierte Typen und die Datentypen LONG VARCHAR, LONG VARGRAPHIC, CLOB, DBCLOB und BLOB sind in Bezug auf die Datenbankpartitionskompatibilität irrelevant, da sie in Verteilungsschlüsseln nicht unterstützt werden.

Sonderregister

CURRENT DBPARTITIONNUM

Das Sonderregister CURRENT DBPARTITIONNUM gibt einen ganzzahligen Wert (Integer) an, der die Koordinatorknotennummer für die Anweisung identifiziert. Für Anweisungen, die von einer Anwendung abgesetzt werden, ist der Koordinator die Datenbankpartition, zu der die Anwendung eine Verbindung herstellt. Für Anweisungen, die von einer Routine abgesetzt werden, ist der Koordinator die Datenbankpartition, von der aus die Routine aufgerufen wurde.

Bei Verwendung in einer SQL-Anweisung einer Routine wird CURRENT DBPARTITIONNUM nie von der aufrufenden Anweisung übernommen.

CURRENT DBPARTITIONNUM gibt den Wert 0 zurück, wenn die Datenbankinstanz nicht zur Unterstützung der Datenbankpartitionierung definiert ist. (Das heißt, wenn keine Datei 'db2nodes.cfg' vorhanden ist. Für partitionierte Datenbanken ist die Datei 'db2nodes.cfg' vorhanden und enthält Definitionen von Datenbankpartitionen.)

CURRENT DBPARTITIONNUM kann mit der Anweisung CONNECT geändert werden. Dies ist jedoch nur unter bestimmten Umständen möglich.

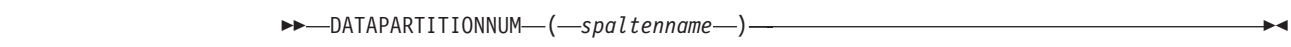
Aus Gründen der Kompatibilität mit Versionen vor Version 8 kann das Schlüsselwort NODE anstelle von DBPARTITIONNUM verwendet werden.

Beispiel: Mit der folgenden Anweisung wird die Hostvariable APPL_NODE (Integer) auf die Nummer der Datenbankpartition gesetzt, mit der die Anwendung verbunden ist.

```
VALUES CURRENT DBPARTITIONNUM
  INTO :APPL_NODE
```

Kapitel 32. SQL-Funktionen

DATAPARTITIONNUM



Das Schema ist SYSIBM.

Die Funktion DATAPARTITIONNUM gibt die Folgenummer (SYSDATAPARTITIONS.SEQNO) der Datenpartition zurück, in der sich die Zeile befindet. Datenpartitionen werden nach Bereich sortiert, wobei die Folgenummern bei 0 beginnen. Zum Beispiel gibt die Funktion DATAPARTITIONNUM den Wert 0 für eine Zeile zurück, die sich in der Datenpartition mit dem niedrigsten Bereich befindet. Das Argument muss der qualifizierte oder nicht qualifizierte Name einer Spalte in der Tabelle sein. Aufgrund der Tatsache, dass Informationen auf Zeilenebene zurückgegeben werden, ist das Ergebnis stets das gleiche, unabhängig davon, welche Spalte angegeben wird. Die Spalte kann einen beliebigen Datentyp besitzen.

Wenn durch *spaltenname* eine Spalte in einer Sicht angegeben wird, muss der Ausdruck für die Spalte in der Sicht auf eine Spalte der zugrunde liegenden Basistabelle verweisen, und die Sicht muss löschfähig sein. Für einen verschachtelten oder allgemeinen Tabellenausdruck gelten dieselben Regeln wie für eine Sicht.

Der Datentyp des Ergebnisses ist INTEGER, und das Ergebnis ist nie NULL.

Diese Funktion kann beim Erstellen einer benutzerdefinierten Funktion nicht als Quellenfunktion verwendet werden. Da die Funktion jeden Datentyp als Argument akzeptiert, ist es nicht erforderlich, zusätzliche Funktionskennungen zur Unterstützung benutzerdefinierter einzigartiger Datentypen zu erstellen.

Die Funktion DATAPARTITIONNUM kann weder in Prüfungen auf Integritätsbedingungen noch in der Definition von generierten Spalten verwendet werden (SQLSTATE-Wert 42881). Die Funktion DATAPARTITIONNUM kann nicht in der Definition einer MQT (Materialized Query Table) verwendet werden (SQLSTATE-Wert 428EC).

Beispiel:

- ```
  SELECT DATAPARTITIONNUM (EMPNO)
  FROM EMPLOYEE
  ```

  Zum Konvertieren einer Folgenummer, die von der Funktion DATAPARTITIONNUM zurückgegeben wird (zum Beispiel 0), in einen Datenpartitionsnamen, der in anderen SQL-Anweisungen (z. B. ALTER TABLE...DETACH PARTITION) verwendet werden kann, können Sie die Katalogsicht SYSCAT.DATAPARTITIONS abfragen. Fügen Sie die durch die Funktion DATAPARTITIONNUM abgerufene Folgenummer (SEQNO) in die Klausel WHERE ein, wie im folgenden Beispiel gezeigt:

  ```
  SELECT DATAPARTITIONNAME
  FROM SYSCAT.DATAPARTITIONS
  WHERE TABNAME = 'EMPLOYEE' AND SEQNO = 0
  ```

  Diese Abfrage liefert zum Beispiel den Wert 'PART0'.
  ```

DBPARTITIONNUM

Das Schema ist SYSIBM.

Die Funktion DBPARTITIONNUM gibt die Datenbankpartitionsnummer für eine Zeile zurück. Wenn sie zum Beispiel in einer Klausel SELECT verwendet wird, gibt sie die Datenbankpartitionsnummer für jede Zeile in der Ergebnismenge zurück.

Das Argument muss der qualifizierte oder nicht qualifizierte Name einer Spalte in der Tabelle sein. Aufgrund der Tatsache, dass Informationen auf Zeilenebene zurückgegeben werden, ist das Ergebnis stets das gleiche, unabhängig davon, welche Spalte angegeben wird. Die Spalte kann einen beliebigen Datentyp besitzen.

Wenn durch *spaltenname* eine Spalte in einer Sicht angegeben wird, muss der Ausdruck für die Spalte in der Sicht auf eine Spalte der zugrunde liegenden Basistabelle verweisen, und die Sicht muss löschfähig sein. Für einen verschachtelten oder allgemeinen Tabellenausdruck gelten dieselben Regeln wie für eine Sicht.

Die bestimmte Zeile (und Tabelle), für die die Datenbankpartitionsnummer von der Funktion DBPARTITIONNUM zurückgegeben wird, wird aus dem Kontext der SQL-Anweisung ermittelt, die diese Funktion verwendet.

Die Datenbankpartitionsnummer, die für Übergangsvariablen und Übergangstabellen zurückgegeben wird, wird aus den aktuellen Übergangswerten der Verteilungsschlüsselspalten abgeleitet. In einem INSERT-Vortrigger gibt die Funktion zum Beispiel die projektierte Datenbankpartitionsnummer zurück, die anhand der aktuellen Werte der neuen Übergangsvariablen ermittelt wird. Die Werte der Verteilungsschlüsselspalten können jedoch von einem nachfolgenden INSERT-Vortrigger geändert werden. Daher ist es möglich, dass die endgültige Datenbankpartitionsnummer der Zeile, wenn sie in die Datenbank eingefügt wird, vom projektierten Wert abweicht.

Der Datentyp des Ergebnisses ist INTEGER, und das Ergebnis ist nie NULL. Wenn keine Datei `db2nodes.cfg` vorhanden ist, ist das Ergebnis 0.

Diese Funktion kann beim Erstellen einer benutzerdefinierten Funktion nicht als Quellenfunktion verwendet werden. Da die Funktion jeden Datentyp als Argument akzeptiert, ist es nicht erforderlich, zusätzliche Funktionskennungen zur Unterstützung benutzerdefinierter einzigartiger Datentypen zu erstellen.

Die Funktion DBPARTITIONNUM kann nicht für replizierte Tabellen, in Prüfungen auf Integritätsbedingungen oder in der Definition von generierten Spalten verwendet werden (SQLSTATE-Wert 42881).

Aus Gründen der Kompatibilität mit früheren Versionen von DB2 kann anstelle von DBPARTITIONNUM auch NODENUMBER angegeben werden.

Beispiele:

- Im folgenden Beispiel wird die Anzahl der Instanzen gezählt, in denen sich die Zeile für einen bestimmten Mitarbeiter in der Tabelle EMPLOYEE in einer anderen Datenbankpartition befindet als die Beschreibung der Abteilung des Mitarbeiters in der Tabelle DEPARTMENT.

```
SELECT COUNT(*) FROM DEPARTMENT D, EMPLOYEE E
  WHERE D.DEPTNO=E.WORKDEPT
  AND DBPARTITIONNUM(E.LASTNAME) <> DBPARTITIONNUM(D.DEPTNO)
```

- Im folgenden Beispiel wird ein Join der Tabellen EMPLOYEE und DEPARTMENT ausgeführt, bei dem sich die Zeilen der beiden Tabellen in derselben Datenbankpartition befinden.

```
SELECT * FROM DEPARTMENT D, EMPLOYEE E
  WHERE DBPARTITIONNUM(E.LASTNAME) = DBPARTITIONNUM(D.DEPTNO)
```

- Im folgenden Beispiel wird ein Vortrigger für die Tabelle EMPLOYEE verwendet, um die Mitarbeiternummer und die projektierte Datenbankpartitionsnummer einer beliebigen neuen Zeile für die Tabelle EMPLOYEE in einer Tabelle mit dem Namen EMPINSERTLOG1 zu protokollieren.

```
CREATE TRIGGER EMPINSLOGTRIG1
BEFORE INSERT ON EMPLOYEE
REFERENCING NEW AS NEWTABLE
FOR EACH ROW
INSERT INTO EMPINSERTLOG1
VALUES(NEWTABLE.EMPNO, DBPARTITIONNUM
(NEWTABLE.EMPNO))
```

Kapitel 33. SQL-Anweisungen

ALTER DATABASE PARTITION GROUP

Die Anweisung ALTER DATABASE PARTITION GROUP dient folgenden Zwecken:

- Hinzufügen einer oder mehrerer Datenbankpartitionen zu einer Datenbankpartitionsgruppe
- Löschen einer oder mehrerer Datenbankpartitionen aus einer Datenbankpartitionsgruppe

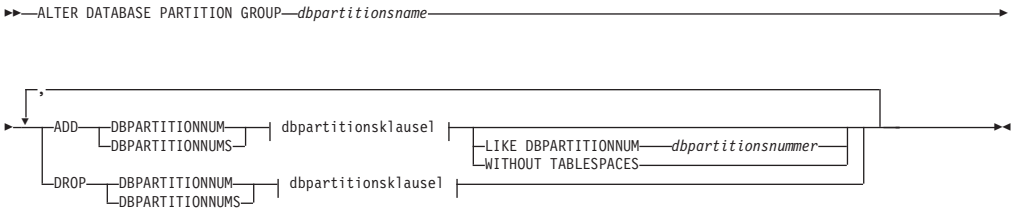
Aufruf

Diese Anweisung kann in ein Anwendungsprogramm eingebettet oder interaktiv abgesetzt werden. Es handelt sich um eine ausführbare Anweisung, die nur dann dynamisch vorbereitet werden kann, wenn das Ausführungsverhalten DYNAMICRULES für das Paket aktiviert ist (SQLSTATE-Wert 42509).

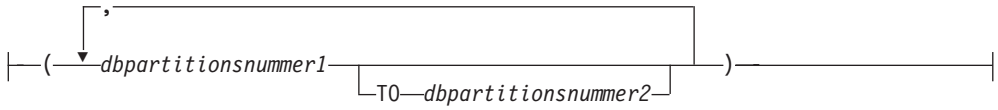
Berechtigung

Die Berechtigungs-ID der Anweisung muss über die Berechtigung SYSCTRL oder SYSADM verfügen.

Syntax



dbpartitionsklausel:



Beschreibung

dbpartitionsname
Gibt den Namen für die Datenbankpartitionsgruppe an. Dies ist ein einteiliger Name, der einen SQL-Bezeichner (entweder normal oder mit Begrenzern) darstellt. Er muss eine Datenbankpartitionsgruppe angeben, die im Katalog beschrieben ist. IBMCATGROUP und IBMTEMPGROUP können nicht angegeben werden (SQLSTATE-Wert 42832).

ADD DBPARTITIONNUM
Gibt die bestimmte Datenbankpartition bzw. die Datenbankpartitionen an, die der Datenbankpartitionsgruppe hinzugefügt werden sollen. DBPARTITION-

NUMS ist synonym zu DBPARTITIONNUM. Alle angegebenen Datenbankpartitionen dürfen noch nicht in der Datenbankpartitionsgruppe definiert sein (SQLSTATE-Wert 42728).

DROP DBPARTITIONNUM
Gibt die bestimmte Datenbankpartition bzw. die Datenbankpartitionen an, die aus der Datenbankpartitionsgruppe gelöscht werden sollen. DBPARTITIONNUMS ist synonym zu DBPARTITIONNUM. Alle angegebenen Datenbankpartitionen müssen in der Datenbankpartitionsgruppe definiert sein (SQLSTATE-Wert 42729).

dbpartitionsklausel
Gibt die Datenbankpartition bzw. die Datenbankpartitionen an, die hinzuzufügen bzw. zu löschen sind.

dbpartitionsnummer1
Geben Sie eine bestimmte Datenbankpartitionsnummer an.

TO *dbpartitionsnummer2*
Geben Sie einen Bereich von Datenbankpartitionsnummern an. Der Wert von *dbpartitionsnummer2* muss größer oder gleich dem Wert von *dbpartitionsnummer1* sein (SQLSTATE-Wert 428A9).

LIKE DBPARTITIONNUM *dbpartitionsnummer*
Gibt an, dass die Container für die vorhandenen Tabellenbereiche in der Datenbankpartitionsgruppe die gleichen sein sollen wie die Container in der Datenbankpartition *dbpartitionsnummer*. Die angegebene Datenbankpartition muss eine Partition sein, die in der Datenbankpartitionsgruppe vor dieser Anweisung vorhanden war und die nicht in einer Klausel DROP DBPARTITIONNUM derselben Anweisung angegeben ist.

Für Tabellenbereiche, die zur Verwendung von dynamischem Speicher definiert sind (d. h. Tabellenbereiche, die mit der Klausel MANAGED BY AUTOMATIC STORAGE der Anweisung CREATE TABLESPACE erstellt wurden bzw. für die überhaupt keine Klausel MANAGED BY angegeben wurde), stimmen die Container nicht unbedingt mit denen aus der angegebenen Partition überein. Vielmehr werden Container automatisch vom Datenbankmanager auf der Basis von Speicherpfaden zugeordnet, die der Datenbank zugeordnet sind. Dies kann dazu führen, dass dieselben Container verwendet werden, oder auch nicht. Die Größe jedes Tabellenbereichs hängt von der Anfangsgröße ab, die beim Erstellen des Tabellenbereichs angegeben wurde, und stimmt möglicherweise nicht mit der aktuellen Größe des Tabellenbereichs in der angegebenen Partition überein.

WITHOUT TABLESPACES
Gibt an, dass die Container für vorhandene Tabellenbereiche in der Datenbankpartitionsgruppe in der neu hinzugefügten Datenbankpartition (bzw. solchen Partitionen) nicht erstellt werden. Container zur Verwendung mit den Tabellenbereichen, die in dieser Datenbankpartitionsgruppe definiert sind, müssen mit der Anweisung ALTER TABLESPACE mit der Klausel *dbpartitionsklausel* definiert werden. Wenn diese Option nicht angegeben wird, werden in neu hinzugefügten Datenbankpartitionen für jeden Tabellenbereich, der in der Datenbankpartitionsgruppe definiert ist, die Standardcontainer angegeben.

Diese Option wird für Tabellenbereiche ignoriert, die zur Verwendung von dynamischem Speicher definiert sind (d. h. Tabellenbereiche, die mit der Klausel MANAGED BY AUTOMATIC STORAGE der Anweisung CREATE TABLESPACE erstellt wurden bzw. für die überhaupt keine Klausel MANAGED BY angegeben wurde). Es gibt keine Möglichkeit, die Erstellung von Containern für diese Tabellenbereiche zu verzögern. Container werden automatisch vom

Datenbankmanager auf der Basis von Speicherpfaden zugeordnet, die der Datenbank zugeordnet sind. Die Größe jedes Tabellenbereichs hängt von der Anfangsgröße ab, die beim Erstellen des Tabellenbereichs angegeben wurde.

Regeln

- Jede Datenbankpartition, die durch eine Nummer angegeben wird, muss in der Datei `db2nodes.cfg` definiert sein (SQLSTATE-Wert 42729).
- Jeder Wert für *dbpartitionnummer* in der Klausel *dbpartitionsklausel* muss eine eindeutige Datenbankpartition angeben (SQLSTATE-Wert 42728).
- Eine gültige Datenbankpartitionsnummer liegt im Bereich zwischen 0 und 999 einschließlich (SQLSTATE-Wert 42729).
- Eine Datenbankpartition kann nicht sowohl in der Klausel ADD als auch in der Klausel DROP angegeben werden (SQLSTATE-Wert 42728).
- Es muss mindestens eine Datenbankpartition in der Datenbankpartitionsgruppe verbleiben. Die letzte Datenbankpartition kann nicht aus einer Datenbankpartitionsgruppe gelöscht werden (SQLSTATE-Wert 428C0).
- Wenn weder die Klausel LIKE DBPARTITIONNUM noch die Klausel WITHOUT TABLESPACES beim Hinzufügen einer Datenbankpartition angegeben wird, wird standardmäßig die niedrigste Datenbankpartitionsnummer der vorhandenen Datenbankpartitionen in der Datenbankpartitionsgruppe verwendet. Wenn diese zum Beispiel *2* ist, wird die Operation so fortgesetzt, als wäre LIKE DBPARTITIONNUM *2* angegeben worden. Damit eine vorhandene Datenbankpartition als Standarddatenbankpartition verwendet werden kann, müssen für sie Container für alle Tabellenbereiche in der Datenbankpartitionsgruppe definiert sein (die Spalte IN_USE der Katalogsicht SYSCAT.DBPARTITIONGROUPDEF hat nicht den Wert 'T').

Anmerkungen

- Wenn einer Datenbankpartitionsgruppe eine Datenbankpartition hinzugefügt wird, wird ein Katalogeintrag für die Datenbankpartition gespeichert (Katalogsicht SYSCAT.DBPARTITIONGROUPDEF). Die Verteilungszuordnung wird unverzüglich geändert, um die neue Datenbankpartition einzufügen, und ein Anzeiger (IN_USE) gibt an, dass die Datenbankpartition in der Verteilungszuordnung enthalten ist, sofern eine der folgenden Bedingungen zutrifft:
  - Es sind keine Tabellenbereiche in der Datenbankpartitionsgruppe definiert.
  - Es sind keine Tabellen in den Tabellenbereichen definiert, die in der Datenbankpartitionsgruppe definiert sind, und die Klausel WITHOUT TABLESPACES wurde nicht angegeben.

  Die Verteilungszuordnung wird nicht geändert und der Anzeiger (IN_USE) wird so eingestellt, dass er angibt, dass die Datenbankpartition nicht in der Verteilungszuordnung enthalten ist, wenn eine der folgenden Bedingungen zutrifft:
  - Es sind Tabellen in Tabellenbereichen in der Datenbankpartitionsgruppe vorhanden.
  - Es sind Tabellenbereiche in der Datenbankpartitionsgruppe enthalten und die Klausel WITHOUT TABLESPACES wurde angegeben (sofern nicht alle Tabellenbereiche zur Verwendung von dynamischem Speicher definiert sind, sodass die Klausel WITHOUT TABLESPACES ignoriert wird).

  Zum Ändern der Verteilungszuordnung muss der Befehl REDISTRIBUTE DATABASE PARTITION GROUP verwendet werden. Dieser Befehl verteilt alle Daten um, ändert die Verteilungszuordnung und ändert den Anzeiger (IN_USE). Tabellenbereichscontainer müssen hinzugefügt werden, bevor versucht wird, Daten umzuverteilen, wenn die Klausel WITHOUT TABLESPACES angegeben wurde.

- Wenn eine Datenbankpartition aus einer Datenbankpartitionsgruppe gelöscht wird, wird der Katalogeintrag (Katalogsicht SYSCAT.DBPARTITIONGROUPDEF) für die Datenbankpartition aktualisiert. Wenn in den Tabellenbereichen, die in der Datenbankpartitionsgruppe definiert sind, keine Tabellen definiert sind, wird die Verteilungszuordnung unverzüglich geändert, um die gelöschte Datenbankpartition auszuschließen, und der Eintrag für die Datenbankpartition in der Datenbankpartitionsgruppe wird gelöscht. Wenn Tabellen vorhanden sind, wird die Verteilungszuordnung nicht geändert und der Anzeiger (IN_USE) wird so eingestellt, dass er angibt, dass die Datenbankpartition zum Löschen ansteht. Der Befehl REDISTRIBUTE DATABASE PARTITION muss zur Umverteilung der Daten und zum Löschen des Eintrags für die Datenbankpartition aus der Datenbankpartitionsgruppe verwendet werden.
- ***Kompatibilitäten***
  - Aus Gründen der Kompatibilität mit früheren Versionen von DB2 bestehen folgende Möglichkeiten:
    - Anstelle von DBPARTITIONNUM kann das Schlüsselwort NODE angegeben werden.
    - Anstelle von DBPARTITIONNUMS kann das Schlüsselwort NODES angegeben werden.
    - Anstelle von DATABASE PARTITION GROUP kann das Schlüsselwort NODEGROUP angegeben werden.

Beispiel

Nehmen Sie an, Sie haben eine partitionierte Datenbank mit den folgenden sechs Datenbankpartitionen: 0, 1, 2, 5, 7 und 8. Zwei Datenbankpartitionen (3 und 6) werden dem System hinzugefügt.

- Wenn Sie die Datenbankpartitionen 3 und 6 einer Datenbankpartitionsgruppe mit dem Namen MAXGROUP hinzufügen wollen, wobei die Tabellenbereichscontainer wie die in der Datenbankpartition 2 sein sollen, sieht die Anweisung wie folgt aus:

```
ALTER DATABASE PARTITION GROUP MAXGROUP
  ADD DBPARTITIONNUMS (3,6)LIKE DBPARTITIONNUM 2
```

- Nehmen Sie an, Sie wollen Datenbankpartition 1 löschen und Datenbankpartition 6 der Datenbankpartitionsgruppe mit dem Namen MEDGROUP hinzufügen. Außerdem definieren Sie die Tabellenbereichscontainer für Datenbankpartition 6 mithilfe einer separaten Anweisung ALTER TABLESPACE. Die Anweisung sieht wie folgt aus:

```
ALTER DATABASE PARTITION GROUP MEDGROUP
  ADD DBPARTITIONNUM(6)WITHOUT TABLESPACES
  DROP DBPARTITIONNUM(1)
```

CREATE DATABASE PARTITION GROUP

Die Anweisung CREATE DATABASE PARTITION GROUP definiert eine neue Datenbankpartitionsgruppe in der Datenbank, ordnet der Datenbankpartitionsgruppe Datenbankpartitionen zu und speichert die Definition der Datenbankpartitionsgruppe im Systemkatalog.

Aufruf

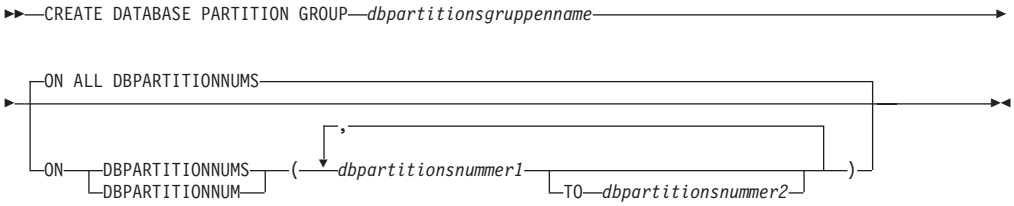
Diese Anweisung kann in ein Anwendungsprogramm eingebettet oder interaktiv abgesetzt werden. Es handelt sich um eine ausführbare Anweisung, die nur dann dynamisch vorbereitet werden kann, wenn das Ausführungsverhalten

DYNAMICRULES für das Paket aktiviert ist (SQLSTATE-Wert 42509).

Berechtigung

Die Zugriffsrechte der Berechtigungs-ID der Anweisung müssen die Berechtigung SYSCTRL oder SYSADM mit einschließen.

Syntax



Beschreibung

dbpartitionsgruppenname
Gibt den Namen für die Datenbankpartitionsgruppe an. Dies ist ein einteiliger Name, der einen SQL-Bezeichner (entweder normal oder mit Begrenzern) darstellt. Der durch *dbpartitionsgruppenname* angegebene Wert darf keine Datenbankpartitionsgruppe bezeichnen, die bereits im Katalog vorhanden ist (SQLSTATE-Wert 42710). Der Wert für *dbpartitionsgruppenname* darf nicht mit den Zeichen 'SYS' oder 'IBM' beginnen (SQLSTATE-Wert 42939).

ON ALL DBPARTITIONNUMS
Gibt an, dass die Datenbankpartitionsgruppe über alle Datenbankpartitionen definiert wird, die in der Datenbank zu dem Zeitpunkt definiert sind (Datei `db2nodes.cfg`), zu dem die Datenbankpartitionsgruppe erstellt wird.

Wenn dem Datenbanksystem eine Datenbankpartition hinzugefügt wird, sollte die Anweisung ALTER DATABASE PARTITION GROUP so abgesetzt werden, dass sie die neue Datenbankpartition in einer Datenbankpartitionsgruppe (einschließlich IBMDEFAULTGROUP) mit einschließt. Darüber hinaus muss der Befehl REDISTRIBUTE DATABASE PARTITION GROUP ausgeführt werden, um Daten in die Datenbankpartition zu versetzen.

ON DBPARTITIONNUMS
Gibt die Datenbankpartitionen an, die in der Datenbankpartitionsgruppe enthalten sind. DBPARTITIONNUM ist synonym zu DBPARTITIONNUMS.

dbpartitionsnummer1
Geben Sie eine Datenbankpartitionsnummer an. (Aus Gründen der Kompatibilität mit früheren Versionen kann ein *Knotenname* der Form `NODE`*nnnnn* angegeben werden.)

TO *dbpartitionsnummer2*
Geben Sie einen Bereich von Datenbankpartitionsnummern an. Der Wert von *dbpartitionsnummer2* muss größer oder gleich dem Wert von *dbpartitionsnummer1* sein (SQLSTATE-Wert 428A9). Alle Datenbankpartitionen aus dem Bereich der angegebenen Datenbankpartitionsnummern werden in die Datenbankpartitionsgruppe aufgenommen.

Regeln

- Jede Datenbankpartition, die durch eine Nummer angegeben wird, muss in der Datei `db2nodes.cfg` definiert sein (SQLSTATE-Wert 42729).
- Jede Datenpartition *dbpartitionsnummer*, die in der Klausel ON DBPARTITIONNUMS aufgeführt wird, darf höchstens einmal angegeben werden (SQLSTATE-Wert 42728).
- Eine gültige *Datenbankpartitionsnummer* liegt im Bereich zwischen 0 und 999 einschließlich (SQLSTATE-Wert 42729).

Anmerkungen

- Diese Anweisung erstellt eine Verteilungszuordnung für die Datenbankpartitionsgruppe. Für jede Verteilungszuordnung wird eine Verteilungszuordnungs-ID (PMAP_ID) generiert. Diese Information wird im Katalog gespeichert und kann aus den Katalogsichten SYSCAT.DBPARTITIONGROUPS und SYSCAT.PARTITIONMAPS abgerufen werden. Jeder Eintrag in der Verteilungszuordnung gibt die Zieldatenbankpartition an, in der sich alle Zeilen nach der Verarbeitung durch ein Hashverfahren befinden. Für eine Datenbankpartitionsgruppe mit einer Einzelpartition enthält die entsprechende Verteilungszuordnung nur einen Eintrag. Für eine Datenbankpartitionsgruppe mit mehreren Datenbankpartitionen enthält die entsprechende Verteilungszuordnung 4096 Einträge, wobei die Datenbankpartitionsnummern den Zuordnungseinträgen standardmäßig in einem Umlaufverfahren zugeordnet werden.
- ***Kompatibilitäten***
  - Aus Gründen der Kompatibilität mit früheren Versionen von DB2 bestehen folgende Möglichkeiten:
    - Anstelle von DBPARTITIONNUM kann das Schlüsselwort NODE angegeben werden.
    - Anstelle von DBPARTITIONNUMS kann das Schlüsselwort NODES angegeben werden.
    - Anstelle von DATABASE PARTITION GROUP kann das Schlüsselwort NODEGROUP angegeben werden.

Beispiele

Nehmen Sie an, Sie haben eine partitionierte Datenbank mit sechs Datenbankpartitionen, die als 0, 1, 2, 5, 7 und 8 definiert sind.

- Nun wollen Sie eine Datenbankpartitionsgruppe mit dem Namen MAXGROUP für alle sechs Datenbankpartitionen erstellen. Dazu würden Sie die folgende Anweisung verwenden:

```
CREATE DATABASE PARTITION GROUP MAXGROUP ON ALL DBPARTITIONNUMS
```

- Wenn Sie eine Datenbankpartitionsgruppe mit dem Namen MEDGROUP für die Datenbankpartitionen 0, 1, 2, 5 und 8 erstellen wollen, würden Sie folgende Anweisung verwenden:

```
CREATE DATABASE PARTITION GROUP MEDGROUP
  ON DBPARTITIONNUMS( 0 TO 2, 5, 8)
```

- Wenn Sie eine Datenbankpartitionsgruppe mit nur einer Datenbankpartition und dem Namen MINGROUP für die Datenbankpartition 7 erstellen wollen, würden Sie folgende Anweisung verwenden:

```
CREATE DATABASE PARTITION GROUP MINGROUP
  ON DBPARTITIONNUM (7)
```

Kapitel 34. Unterstützte SQL-Verwaltungsroutinen und Verwaltungssichten

Gespeicherte Prozedur ADMIN_CMD und zugeordnete SQL-Verwaltungsroutinen

Befehl GET STMM TUNING DBPARTITIONNUM mit Verwendung der Prozedur ADMIN_CMD

Dient zum Lesen der Katalogtabellen, um die Nummer der vom Benutzer bevorzugten Optimierungsdatenbankpartition des Speichermanagers für automatische Leistungsoptimierung (STMM) und die aktuelle Nummer der STMM-Optimierungsdatenbankpartition abzurufen.

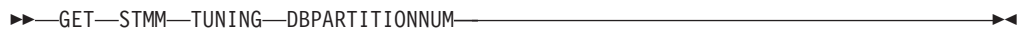
Berechtigung

Berechtigung SYSADM oder DBADM

Erforderliche Verbindung

Datenbank

Befehlssyntax



Beispiel

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

Das folgende Beispiel zeigt die Ausgabe dieser Abfrage.

```
Ergebnismenge 1
  --------------

  USER_PREFERRED_NUMBER CURRENT_NUMBER
  --------------------- --------------
                      2              2

  1 Satz/Sätze ausgewählt.

  Rückgabestatus = 0
```

Hinweise

Die Nummer (USER_PREFERRED_NUMBER) der vom Benutzer bevorzugten Optimierungsdatenbankpartition des Speichermanagers für automatische Leistungsoptimierung (STMM) wird vom Benutzer festgelegt und gibt die Datenbankpartition an, in der der Benutzer die Speicheroptimierungsfunktion ausführen möchte. Während die Datenbank aktiv ist, wird die Optimierungspartition asynchron einige Male pro Stunde aktualisiert. Infolgedessen ist es möglich, dass die zurückgegebenen Werte für CURRENT_NUMBER und USER_PREFERRED_NUMBER nach einer Aktualisierung der Nummer der vom Benutzer bevorzugten STMM-Partition nicht

synchron sind. Um dies zu beheben, können Sie entweder warten, bis der Wert für CURRENT_NUMBER asynchron aktualisiert wird, oder Sie können die Datenbank stoppen und erneut starten, um die Aktualisierung von CURRENT_NUMBER zu erzwingen.

Befehl UPDATE STMM TUNING DBPARTITIONNUM mit Verwendung der Prozedur ADMIN_CMD

Aktualisiert die vom Benutzer bevorzugte Optimierungsdatenbankpartition des Speichermanagers für automatische Leistungsoptimierung (STMM).

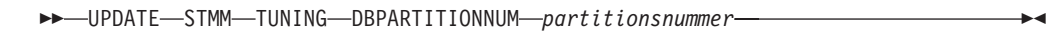
Berechtigung

Berechtigung SYSADM oder DBADM

Erforderliche Verbindung

Datenbank

Befehlssyntax



Befehlsparameter

partitionsnummer
Der Wert *partitionsnummer* ist ein ganzzahliger Wert (Integer). Wenn der Wert -1 oder eine nicht vorhandene Datenbankpartitionsnummer angegeben wird, wählt DB2 automatisch eine geeignete Datenbankpartition zur Ausführung der STMM-Speicheroptimierungsfunktion aus.

Beispiel

Im folgenden Beispiel wird die vom Benutzer bevorzugte Optimierungsdatenbankpartition des Speichermanagers für automatische Leistungsoptimierung (STMM) in Datenbankpartition 3 aktualisiert.

```
CALL SYSPROC.ADMIN_CMD( 'update stmm tuning dbpartitionnum 3' )
```

Hinweise

Der STMM-Optimierungsprozess überprüft in regelmäßigen Abständen, ob sich der Wert der Nummer für die vom Benutzer bevorzugte STMM-Optimierungspartition geändert hat. Der STMM-Optimierungsprozess wechselt zu der vom Benutzer bevorzugten STMM-Optimierungsdatenbankpartition, wenn die in *partitionsnummer* angegebene Nummer vorhanden ist eine aktive Datenbankpartition bezeichnet. Wenn dieser Befehl die Nummer der STMM-Optimierungsdatenbankpartition geändert hat, erfolgt unverzüglich eine Änderung an der aktuellen STMM-Optimierungsdatenbankpartitionsnummer.

Der Status der Befehlsausführung wird in dem SQL-Kommunikationsbereich (SQLCA) zurückgegeben, der von der CALL-Anweisung generiert wird.

Dieser Befehl führt eine Commitoperation für seine Änderungen in der Prozedur ADMIN_CMD aus.

SQL-Verwaltungsroutinen und Verwaltungssichten für die Konfiguration

DB_PARTITIONS

Die Tabellenfunktion DB_PARTITIONS gibt den Inhalt der Datei 'db2nodes.cfg' in Form einer Tabelle zurück.

Syntax

Das Schema ist SYSPROC.

Berechtigung

Zugriffsrecht EXECUTE für die Tabellenfunktion DB_PARTITIONS.

Tabellenfunktionsparameter

Die Funktion hat keine Eingabeparameter.

Beispiel

Das folgende Beispiel zeigt das Abrufen von Informationen aus einer Datenbank mit drei logischen Partitionen.

```
SELECT * FROM TABLE(DB_PARTITIONS()) AS T
```

Die Ausgabe aus dieser Beispielabfrage sieht wie folgt aus.

```
PARTITION_NUMBER HOST_NAME                PORT_NUMBER SWITCH_NAME
---------------- ---------------------...- ----------- -----------...-
               0 jessicae.torolab.ibm.com            0 jessicae
               1 jessicae.torolab.ibm.com            1 jessicae
               2 jessicae.torolab.ibm.com            2 jessicae

  3 Satz/Sätze ausgewählt.
```

Zurückgegebene Informationen

Tabelle 42. Von der Tabellenfunktion DB_PARTITIONS zurückgegebene Informationen

Spaltenname	Datentyp	Beschreibung
PARTITION_NUMBER	SMALLINT	Eine eindeutige Nummer aus dem Bereich von 0 bis 999, die einen Datenbankpartitionsserver in einer Umgebung mit partitionierten Datenbanken angibt.
HOST_NAME	VARCHAR(128)	Der TCP/IP-Hostname des Datenbankpartitionsservers.
PORT_NUMBER	SMALLINT	Die Portnummer für den Datenbankpartitionsserver.

Tabelle 42. Von der Tabellenfunktion DB_PARTITIONS zurückgegebene Informationen (Forts.)

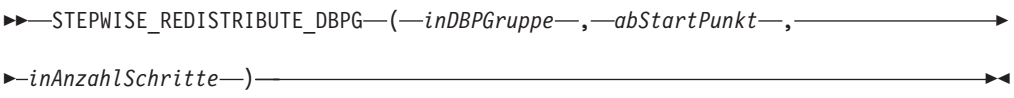
Spaltenname	Datentyp	Beschreibung
SWITCH_NAME	VARCHAR(128)	Der Name einer Hochgeschwindigkeitsverbindung bzw. eines Switch für die Kommunikation von Datenbankpartitionen.

SQL-Verwaltungsroutinen zur schrittweisen Datenumverteilung

STEPWISE_REDISTRIBUTE_DBPG (Prozedur) - Teil der Datenbankpartitionsgruppe umverteilen

Die Funktion STEPWISE_REDISTRIBUTE_DBPG führt eine Umverteilung eines Teils der Datenbankpartitionsgruppe entsprechend der Eingabe für die Prozedur und der Einstellungsdatei durch, die mit der Prozedur SET_SWRD_SETTINGS erstellt oder aktualisiert wurde.

Syntax



Das Schema ist SYSPROC.

Prozedurparameter

inDBPGruppe
Ein Eingabeargument des Typs VARCHAR (128), das den Namen der Zieldatenbankpartitionsgruppe angibt.

abStartPunkt
Ein Eingabeargument des Typs SMALLINT, das den zu verwendenden Startpunkt angibt. Wenn der Parameter auf einen positiven ganzzahligen Wert gesetzt wird und nicht NULL ist, verwendet die Prozedur STEPWISE_REDISTRIBUTE_DBPG diesen Wert anstelle des Werts *nextStep*, der in der Einstellungsdatei angegeben ist. Dies ist eine nützliche Option, wenn Sie die Prozedur STEPWISE_REDISTRIBUTE_DBPG ab einem bestimmten Schritt erneut ausführen wollen. Wenn der Parameter auf `NULL` gesetzt wird, wird der Wert in *nextStep* verwendet.

inAnzahlSchritte
Ein Eingabeargument des Typs SMALLINT, das die Anzahl der auszuführenden Schritte angibt. Wenn der Parameter auf einen positiven ganzzahligen Wert gesetzt wird und nicht `NULL` ist, verwendet die Prozedur STEPWISE_REDISTRIBUTE_DBPG diesen Wert anstelle des Werts *stageSize*, der in der Einstellungsdatei angegeben ist. Dies ist eine nützliche Option, wenn Sie die Prozedur STEPWISE_REDISTRIBUTE_DBPG mit einer anderen Anzahl von Schritten als die, die in den Einstellungen angegeben ist, erneut ausführen wollen. Wenn zum Beispiel fünf Schritte in einem geplanten Arbeitsabschnitt (Stage) enthalten sind und der Umverteilungsprozess bei Schritt 3 fehlgeschlagen ist, kann die Prozedur STEPWISE_REDISTRIBUTE_DBPG aufgerufen werden, um die übrigen drei Schritte auszuführen, nachdem die Fehlerbedingung behoben

wurde. Wenn der Parameter auf NULL gesetzt wird, wird der Wert in *stageSize* verwendet. Der Wert -2 kann in dieser Prozedur verwendet werden, um anzugeben, dass die Anzahl unbegrenzt ist.

Anmerkung: Es gibt keinen Parameter zur Angabe eines entsprechenden Gegenstücks zur Option **NOT ROLLFORWARD RECOVERABLE** im Befehl REDISTRIBUTE DATABASE PARTITION GROUP. Die Protokollierung erfolgt immer für die Zeilendatenumverteilung, die ausgeführt wird, wenn die Prozedur STEPWISE_REDISTRIBUTE_DBPG verwendet wird.

Berechtigung

- Zugriffsrecht EXECUTE für die Prozedur STEPWISE_REDISTRIBUTE_DBPG
- SYSADM, SYSCTRL oder DBADM

Beispiel

Das folgende Beispiel zeigt, wie die Datenbankpartitionsgruppe "IBMDEFAULTGROUP" entsprechend dem Umverteilungsplan, der in der Registrierdatenbank durch die Prozedur SET_SWRD_SETTINGS gespeichert wurde, umverteilt wird. Die Prozedur startet mit Schritt 3 und verteilt die Daten um, bis zwei Schritte im Umverteilungsplan ausgeführt sind.

```
CALL SYSPROC.STEPWISE_REDISTRIBUTE_DBPG('IBMDEFAULTGROUP', 3, 2)
```

Ein vollständiges Verwendungsbeispiel der Prozeduren zur schrittweisen Umverteilung finden Sie im Abschnitt „Prozedur STEPWISE_REDISTRIBUTE_DBPG” im Dokument *Partitionierung und Clustering*.

Hinweise

Wenn der Registrierdatenbankwert für processState mit der Prozedur SET_SWRD_SETTINGS auf den Wert 1 aktualisiert wird, nachdem die Ausführung der Prozedur STEPWISE_REDISTRIBUTE_DBPG gestartet wurde, wird der Prozess bei Beginn des nächsten Schritts gestoppt und eine Warnung zurückgegeben.

Da von dem Umverteilungsprozess eine SQL-Anweisung COMMIT aufgerufen wird, wird die Ausführung des Umverteilungsprozesses über eine Verbindung des Typs 2 nicht unterstützt.

Teil 6. Anhänge und Schlussteil

Anhang A. Installieren als Benutzer ohne Rootberechtigung

Installieren eines DB2-Produkts als Benutzer ohne Rootberechtigung

Die meisten DB2-Datenbankprodukte können von einem Benutzer ohne Rootberechtigung installiert werden.

Vorbereitung

Bevor Sie ein DB2-Datenbankprodukt als Benutzer ohne Rootberechtigung installieren, sollten Sie die Unterschiede zwischen Rootinstallationen und Installationen ohne Rootberechtigung kennen sowie die Einschränkungen bei Installationen ohne Rootberechtigung. Weitere Informationen zur Installation `ohne Rootberechtigung` finden Sie im Abschnitt „Installation als Benutzer ohne Rootberechtigung - Übersicht (Linux und UNIX)".

Die folgenden Voraussetzungen gelten beim Installieren eines DB2-Datenbankprodukts als Benutzer ohne Rootberechtigung:

- Sie müssen in der Lage sein, die Installations-DVD anzuhängen, oder sie muss bereits angehängt sein.
- Sie müssen über eine gültige Benutzer-ID verfügen, die als Eigner einer DB2-Instanz verwendet werden kann.

  Für Benutzer-IDs gelten die folgenden Einschränkungen und Voraussetzungen:
  - Sie müssen einer anderen primären Gruppe als 'guests', 'admins', 'users' und 'local' angehören.
  - Sie dürfen Kleinbuchstaben (a–z), Zahlen (0–9) und das Unterstreichungszeichen (_) enthalten.
  - Sie dürfen nicht länger als acht Zeichen sein.
  - Sie dürfen nicht mit IBM, SYS, SQL oder einer Zahl beginnen.
  - Sie dürfen kein in DB2 reserviertes Wort (USERS, ADMINS, GUESTS, PUBLIC oder LOCAL) sowie kein reserviertes SQL-Wort sein.
  - Es dürfen keine Benutzer-IDs mit Rootberechtigung als DB2-Instanz-ID, DAS-ID oder abgeschirmte ID verwendet werden.
  - Sie dürfen keine Zeichen mit Akzent enthalten.
  - Wenn keine neuen Benutzer-IDs erstellt, sondern vorhandene Benutzer-IDs verwendet werden, müssen folgende Bedingungen erfüllt sein:
    - Die Benutzer-IDs sind nicht gesperrt.
    - Die Kennwörter der Benutzer-IDs sind nicht abgelaufen.
- Die bestehenden Hardware- und Softwarevoraussetzungen für das Produkt das Sie installieren, gelten für Benutzer ohne Rootberechtigung genauso wie für Rootbenutzer.
- Unter AIX Version 5.3 muss die asynchrone E/A (Asynchronous I/O, AIO) aktiviert sein.
- Ihr Ausgangsverzeichnis muss ein gültiger DB2-Pfad sein.

  Für DB2-Installationspfade gelten die folgenden Regeln:
  - Sie dürfen Kleinbuchstaben (a–z), Großbuchstaben (A–Z) und das Unterstreichungszeichen (_) enthalten.
  - Sie dürfen nicht länger als 128 Zeichen sein.

– Sie dürfen keine Leerzeichen enthalten.
– Sie dürfen keine Sonderzeichen der jeweiligen Landessprache enthalten.

Informationen zu dieser Task

Das Installieren von DB2-Datenbankprodukten als Benutzer ohne Rootberechtigung muss für den Benutzer ohne Rootberechtigung transparent sein. Anders ausgedrückt: Die einzige Voraussetzung zum Ausführen der nicht als Root ausgeführten Installation eines DB2-Datenbankprodukts muss die Anmeldung als Benutzer ohne Rootberechtigung sein.

Vorgehensweise

Gehen Sie wie folgt vor, um eine nicht als Root ausgeführte Installation durchzuführen:

1. Melden Sie sich als Benutzer ohne Rootberechtigung an.
2. Installieren Sie das gewünschte DB2-Datenbankprodukt mit einer der zur Verfügung stehenden Methoden. Zu den verfügbaren Optionen gehören die folgenden:
   - Der **DB2-Installationsassistent** (GUI-Installationsprogramm)
   - Der Befehl db2_install
   - Der Befehl db2setup mit einer Antwortdatei (nichtüberwachte Installation)

     Anmerkung: Da Benutzer ohne Rootberechtigung kein Installationsverzeichnis für DB2-Datenbankprodukte auswählen können, werden alle Vorkommen des Schlüsselworts `FILE` in Ihrer Antwortdatei ignoriert.
3. Nach der Installation des DB2-Datenbankprodukts müssen Sie eine neue Anmeldesitzung öffnen, um die nicht als Root ausgeführte DB2-Instanz zu verwenden. Alternativ dazu können Sie dieselbe Anmeldesitzung verwenden, wenn Sie die Instanzumgebung mit *$HOME*`/sqllib/db2profile` (für Benutzer der Bourne- und der Korn-Shell) oder *$HOME*`/sqllib/db2chsrc` (für Benutzer der C-Shell) einrichten, wobei *$HOME* das Ausgangsverzeichnis des Benutzers ohne Rootberechtigung ist.

Weitere Schritte

Nach der Installation des DB2-Datenbankprodukts sollten Sie die Begrenzungen für Benutzerprozessressourcen (ulimits) Ihres Betriebssystems prüfen. Wenn die Mindestwerte für 'ulimit' nicht erreicht werden, kann es in der DB2-Steuerkomponente zu unerwarteten Engpässen bei den Betriebsressourcen kommen. Diese Fehler können einen Ausfall von DB2 zur Folge haben.

Anhang B. Verwenden von Backup

Verwenden von Backup

Verwenden Sie den Befehl BACKUP DATABASE, um eine Kopie der Daten einer Datenbank zu erstellen und sie für den Fall eines Fehlers oder der Beschädigung der Originaldaten auf einem anderen Datenträger zu speichern. Sie können eine gesamte Datenbank, eine Datenbankpartition oder nur ausgewählte Tabellenbereiche mit Backup sichern.

Vorbereitung

Es braucht noch keine Verbindung zu der zu sichernden Datenbank zu bestehen: Das Backup-Datenbankdienstprogramm stellt automatisch eine Verbindung zu der angegebenen Datenbank her, die nach Abschluss der Backup-Operation beendet wird. Wenn bereits eine Verbindung zu der zu sichernden Datenbank besteht, wird diese Verbindung beim Absetzen des Befehls BACKUP DATABASE unterbrochen, und die Backup-Operation wird fortgesetzt.

Bei der Datenbank kann es sich um eine lokale oder ferne Datenbank handeln. Das Backup-Image bleibt auf dem Datenbankserver, sofern Sie kein Speicherverwaltungsprodukt wie Tivoli Storage Manager (TSM) oder DB2 Advanced Copy Services (ACS) verwenden.

Wenn Sie ein Offline-Backup durchführen und Sie die Datenbank mit der Anweisung ACTIVATE DATABASE aktiviert haben, müssen Sie die Datenbank inaktivieren, bevor Sie das Offline-Backup durchführen. Wenn die Datenbank über aktive Verbindungen verfügt, muss ein Benutzer mit der Berechtigung SYSADM eine Verbindung zu der Datenbank herstellen und die folgenden Befehle absetzen, um die Datenbank erfolgreich zu inaktivieren:

```
CONNECT TO aliasname_der_datenbank
QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS;
UNQUIESCE DATABASE;
TERMINATE;
DEACTIVATE DATABASE aliasname-der-datenbank
```

In einer Umgebung mit partitionierten Datenbanken haben Sie die folgenden Möglichkeiten: Mit dem Befehl BACKUP DATABASE können Sie Datenbankpartitionen einzeln sichern, mit dem Befehlsparameter **ON DBPARTITIONNUM** können Sie mehrere der Datenbankpartitionen auf einmal sichern und mit dem Parameter **ALL DBPARTITIONNUMS** können Sie alle Datenbankpartitionen gleichzeitig sichern. Mit dem Befehl LIST NODES können Sie die Datenbankpartitionen identifizieren, in denen Benutzertabellen enthalten sind, für die das Ausführen eines Backups sinnvoll ist.

Wenn Sie ein *Offline-Backup* in einer Umgebung mit partitionierten Datenbanken ausführen und kein SSV-Backup (SSV = Single System View, Einzelsystemsicht) verwenden, sollten Sie die Katalogpartition von allen anderen Datenbankpartitionen getrennt sichern. Zum Beispiel können Sie die Katalogpartition zuerst sichern und anschließend die anderen Datenbankpartitionen sichern. Dies ist notwendig, weil die Backup-Operation möglicherweise eine exklusive Datenbankverbindung mit der Katalogpartition erfordert, während deren die anderen Datenbankpartitionen keine Verbindung herstellen können. Wenn Sie ein *Online-Backup* ausführen,

können alle Datenbankpartitionen (einschließlich der Katalogpartition) gleichzeitig und in beliebiger Reihenfolge gesichert werden.

Zum Backup von Datenbankpartitionen können Sie auch den Befehlseditor verwenden. Da diese Vorgehensweise jedoch die aktualisierende Recovery nicht unterstützt, empfiehlt es sich, die auf diesen Knoten vorhandene Datenbank regelmäßig zu sichern. Für den möglichen Fall einer Beschädigung der Datei `'db2nodes.cfg'` ist es ratsam, zusammen mit allen Backup-Kopien, die Sie erstellen, auch eine Kopie dieser Datei zu speichern.

Auf einem System, in dem mit verteilten Anforderungen gearbeitet wird, werden die Backup-Operationen auf die Datenbank für verteilte Anforderungen sowie auf die Metadaten angewendet, die im Katalog dieser Datenbank gespeichert sind (Wrapper, Server, Kurznamen usw.). Datenquellenobjekte (Tabellen und Sichten) werden nicht gesichert, es sei denn, diese Objekte werden in der Datenbank für verteilte Anforderungen gespeichert.

Wenn eine Datenbank mit einem vorherigen Release des Datenbankmanagers erstellt, aber nicht auf das aktuelle Release migriert wurde, müssen Sie sie migrieren. Andernfalls ist es nicht möglich, eine Backup-Kopie der Datenbank anzulegen.

Einschränkungen

Für das Backup-Dienstprogramm gelten die folgenden Einschränkungen:

- Es ist nicht möglich, gleichzeitig ein Backup eines Tabellenbereichs und einen Restore eines Tabellenbereichs durchzuführen. Dies gilt auch dann, wenn Backup und Restore für unterschiedliche Tabellenbereiche erfolgen.
- Wenn Sie in der Lage sein wollen, in einer Umgebung mit partitionierten Datenbanken eine aktualisierende Recovery durchzuführen, müssen Sie die Datenbank auf den Knoten der Liste regelmäßig sichern und über mindestens ein Backup-Image der übrigen Knoten im System verfügen (auch von Knoten, die keine Benutzerdaten für diese Datenbank enthalten). In zwei Situationen ist ein Backup-Image einer Datenbankpartition auf einem Datenbankpartitionsserver erforderlich, der keine Benutzerdaten für die Datenbank enthält:
  - Sie haben dem Datenbanksystem einen Datenbankpartitionsserver hinzugefügt, nachdem das letzte Backup erstellt wurde, und müssen eine aktualisierende Recovery auf diesem Datenbankpartitionsserver durchführen.
  - Die punktuelle Recovery wird verwendet, wozu alle Datenbankpartitionen im System den Status 'aktualisierende Recovery anstehend' aufweisen müssen.
- Online-Backup-Operationen für DMS-Tabellenbereiche sind mit den folgenden Operationen nicht kompatibel:
  - Laden
  - Reorganisation (online und offline)
  - Löschen von Tabellenbereichen
  - Tabellenverkürzung
  - Indexerstellung
  - Verwenden des Parameters NOT LOGGED INITIALLY (wird mit den Anweisungen CREATE TABLE und ALTER TABLE verwendet)
- Wenn Sie versuchen, ein Offline-Backup einer Datenbank durchzuführen, die momentan aktiv ist, empfangen Sie einen Fehler. Sie können vor einem Offline-Backup sicherstellen, dass die Datenbank nicht aktiv ist, in dem Sie den Befehl DEACTIVATE DATABASE ausführen.

Informationen zu dieser Task

Das Backup-Dienstprogramm kann über den Befehlszeilenprozessor (CLP), den Assistenten **Datenbank sichern** in der Steuerzentrale, die Prozedur ADMIN_CMD mit dem Parameter BACKUP DATABASE oder die Anwendungsprogrammierschnittstelle (API) db2Backup aufgerufen werden.

Das folgende Beispiel zeigt einen Befehl BACKUP DATABASE, der über den CLP abgesetzt wird:

```
db2 backup database sample to c:\DB2Backups
```

Vorgehensweise

Gehen Sie wie folgt vor, um den Assistenten **Datenbank sichern** zu öffnen:

1. Erweitern Sie in der Steuerzentrale die Objektbaumstruktur, bis das Datenbank- oder Tabellenbereichsobjekt angezeigt wird, das Sie sichern möchten.
2. Klicken Sie mit der rechten Maustaste das Objekt an und wählen Sie **Sichern** im Kontextmenü aus. Der Assistent **Datenbank sichern** wird geöffnet.

Weitere Schritte

Ausführliche Informationen enthält die Kontexthilfefunktion der Steuerzentrale.

Wenn Sie ein Offline-Backup durchgeführt haben, müssen Sie nach der Ausführung die Datenbank reaktivieren:

```
ACTIVATE DATABASE sample
```

Anhang C. Katalogsichten für Umgebungen mit partitionierten Datenbanken

SYSCAT.BUFFERPOOLDBPARTITIONS

Jede Zeile stellt eine Kombination aus einem Pufferpool und einer Datenbankpartition dar, bei der sich die Größe des Pufferpools in dieser Partition von seiner Standardgröße für andere Partitionen in derselben Datenbankpartitionsgruppe (wie in SYSCAT.BUFFERPOOLS dargestellt) unterscheidet.

Tabelle 43. Katalogsicht SYSCAT.BUFFERPOOLDBPARTITIONS

Spaltenname	Datentyp	Nullwert?	Beschreibung
BUFFERPOOLID	INTEGER		Interne Kennung (ID) für den Pufferpool.
DBPARTITIONNUM	SMALLINT		Datenbankpartitionsnummer.
NPAGES	INTEGER		Anzahl von Seiten in diesem Pufferpool in dieser Datenbankpartition.

SYSCAT.DATAPARTITIONEXPRESSION

Jede Zeile stellt einen Ausdruck für den jeweiligen Teil des Tabellenpartitionierungsschlüssels dar.

Tabelle 44. Katalogsicht SYSCAT.DATAPARTITIONEXPRESSION

Spaltenname	Datentyp	Nullwert?	Beschreibung
TABSCHEMA	VARCHAR(128)		Schemaname der partitionierten Tabelle.
TABNAME	VARCHAR(128)		Nicht qualifizierter Name der partitionierten Tabelle.
DATAPARTITIONKEYSEQ	INTEGER		Folgen-ID für den Ausdrucksschlüsselteil, beginnend bei 1.
DATAPARTITIONEXPRESSION	CLOB (32K)		Ausdruck für diesen Eintrag in der Folge, in SQL-Syntax.
NULLSFIRST	CHAR (1)		• N = Nullwerte in diesem Ausdruck werden in Vergleichen als höherer Wert interpretiert. • Y = Nullwerte in diesem Ausdruck werden in Vergleichen als niedrigerer Wert interpretiert.

SYSCAT.DATAPARTITIONS

Jede Zeile stellt eine Datenpartition dar.

Tabelle 45. Katalogsicht SYSCAT.DATAPARTITIONS

Spaltenname	Datentyp	Nullwert?	Beschreibung
DATAPARTITIONNAME	VARCHAR(128)		Name der Datenpartition.

Tabelle 45. Katalogsicht SYSCAT.DATAPARTITIONS (Forts.)

Spaltenname	Datentyp	Nullwert?	Beschreibung
TABSCHEMA	VARCHAR(128)		Schemaname der Tabelle, zu der diese Datenpartition gehört.
TABNAME	VARCHAR(128)		Nicht qualifizierter Name der Tabelle, zu der diese Datenpartition gehört.
DATAPARTITIONID	INTEGER		Kennung (ID) für die Datenpartition.
TBSPACEID	INTEGER	J	Kennung (ID) für den Tabellenbereich, in dem diese Datenpartition gespeichert ist. Null, wenn STATUS gleich 'I' ist.
PARTITIONOBJECTID	INTEGER	J	Kennung (ID) für die Datenpartition innerhalb des Tabellenbereichs.
LONG_TBSPACEID	INTEGER	J	Kennung (ID) für den Tabellenbereich, in dem lange Daten (LONG) gespeichert werden. Null, wenn STATUS gleich 'I' ist.
ACCESS_MODE	CHAR (1)		Zugriffsbeschränkungsstatus der Datenpartition. Dieser Status bezieht sich nur auf Objekte, die sich im Status 'Festlegen der Integrität anstehend' befinden, bzw. auf Objekte, die von einer Anweisung SET INTEGRITY verarbeitet wurden. Mögliche Werte: • D = Kein Versetzen von Daten • F = Vollzugriff • N = Kein Zugriff • R = Lesezugriff
STATUS	VARCHAR (32)		• A = Datenpartition wurde neu zugeordnet. • D = Zuordnung der Datenpartition wurde aufgehoben. • I = Datenpartition mit aufgehobener Zuordnung, deren Eintrag im Katalog nur während der asynchronen Indexbereinigung beibehalten wird. Zeilen mit dem STATUS-Wert 'I' werden entfernt, wenn alle Indexdatensätze, die auf die Partition mit aufgehobener Zuordnung verweisen, gelöscht wurden. • Leere Zeichenfolge = Datenpartition ist sichtbar (Normalstatus). Byte 2 bis 32 sind für zukünftige Zwecke reserviert.
SEQNO	INTEGER		Folgenummer der Datenpartition (bei 0 beginnend).
LOWINCLUSIVE	CHAR (1)		• N = LOWKEY-Wert ist nicht einschließend. • Y = LOWKEY-Wert ist einschließend.
LOWVALUE	VARCHAR (512)		LOWKEY-Wert (eine Zeichenfolgedarstellung eines SQL-Werts) für diese Datenpartition.
HIGHINCLUSIVE	CHAR (1)		• N = HIGHKEY-Wert ist nicht einschließend. • Y = HIGHKEY-Wert ist einschließend.

Tabelle 45. Katalogsicht SYSCAT.DATAPARTITIONS (Forts.)

Spaltenname	Datentyp	Nullwert?	Beschreibung
HIGHVALUE	VARCHAR (512)		HIGHKEY-Wert (eine Zeichenfolgedarstellung eines SQL-Werts) für diese Datenpartition.

SYSCAT.DBPARTITIONGROUPDEF

Jede Zeile stellt eine Datenbankpartition dar, die in einer Datenbankpartitionsgruppe enthalten ist.

Tabelle 46. Katalogsicht SYSCAT.DBPARTITIONGROUPDEF

Spaltenname	Datentyp	Nullwert?	Beschreibung
DBPGNAME	VARCHAR(128)		Name der Datenbankpartitionsgruppe, in der die Datenbankpartition enthalten ist.
DBPARTITIONNUM	SMALLINT		Partitionsnummer einer Datenbankpartition, die in der Datenbankpartitionsgruppe enthalten ist. Eine gültige Partitionsnummer liegt im Bereich zwischen 0 und 999 (einschließlich).
IN_USE	CHAR (1)		Status der Datenbankpartition. • A = Die neu hinzugefügte Datenbankpartition ist nicht in der Verteilungszuordnung enthalten, es wurden jedoch die Container für die Tabellenbereiche in der Datenbankpartitionsgruppe erstellt. Die Datenbankpartition wird der Verteilungszuordnung hinzugefügt, wenn der REDISTRIBUTION-Prozess läuft. • D = Die Datenbankpartition wird gelöscht, wenn eine Operation 'redistribute database partition group' erfolgreich ausgeführt wurde. • T = Die neu hinzugefügte Datenbankpartition ist nicht in der Verteilungszuordnung und sie wurde unter Verwendung der Klausel WITHOUT TABLESPACES hinzugefügt. Den Tabellenbereichen in der Datenbankpartitionsgruppe müssen Container hinzugefügt werden. • Y = Die Datenbankpartition ist in der Verteilungszuordnung.

SYSCAT.DBPARTITIONGROUPS

Jede Zeile stellt eine Datenbankpartitionsgruppe dar.

Tabelle 47. Katalogsicht SYSCAT.DBPARTITIONGROUPS

Spaltenname	Datentyp	Nullwert?	Beschreibung
DBPGNAME	VARCHAR(128)		Name der Datenbankpartitionsgruppe.

Tabelle 47. Katalogsicht SYSCAT.DBPARTITIONGROUPS (Forts.)

Spaltenname	Datentyp	Nullwert?	Beschreibung
OWNER	VARCHAR(128)		Berechtigungs-ID, unter der die Datenbankpartitionsgruppe erstellt wurde.
OWNERTYPE	CHAR (1)		• S = Der Eigner ist das System. • U = Der Eigner ist ein Einzelbenutzer.
PMAP_ID	SMALLINT		Kennung (ID) für die Verteilungszuordnung in der Katalogsicht SYSCAT.PARTITIONMAPS.
REDISTRIBUTE_PMAP_ID	SMALLINT		Kennung (ID) für die Verteilungszuordnung, die momentan für die Umverteilung verwendet wird. Dieser Wert ist -1, wenn die Umverteilung momentan nicht ausgeführt ist.
CREATE_TIME	TIMESTAMP		Zeitpunkt der Erstellung der Datenbankpartitionsgruppe.
DEFINER[1]	VARCHAR(128)		Berechtigungs-ID, unter der die Datenbankpartitionsgruppe erstellt wurde.
REMARKS	VARCHAR (254)	J	Vom Benutzer bereitgestellte Kommentare oder NULL.

Anmerkung:

1. Die Spalte DEFINER wird nur aus Gründen der Abwärtskompatibilität bereitgestellt. Siehe Spalte OWNER.

SYSCAT.PARTITIONMAPS

Jede Zeile stellt eine Verteilungszuordnung dar, die zur Verteilung der Zeilen einer Tabelle auf die Datenbankpartitionen in einer Datenbankpartitionsgruppe unter Verwendung eines Hashverfahrens auf der Basis des Verteilungsschlüssels der Tabelle dient.

Tabelle 48. Katalogsicht SYSCAT.PARTITIONMAPS

Spaltenname	Datentyp	Nullwert?	Beschreibung
PMAP_ID	SMALLINT		Kennung (ID) für die Verteilungszuordnung.
PARTITIONMAP	BLOB (8192)		Verteilungszuordnung. Dabei handelt es sich um einen Vektor aus 4096 ganzzahligen, 2 Byte langen Werten für eine Datenbankpartitionsgruppe mit mehreren Partitionen. Für eine Datenbankpartitionsgruppe mit einer Einzelpartition ist nur ein Eintrag vorhanden, der die Partitionsnummer der Einzelpartition angibt.

Anhang D. Übersicht über die technischen Informationen zu DB2

Die technischen Informationen zu DB2 stehen über die folgenden Tools und Methoden zur Verfügung:

- *DB2-Informationszentrale*
  - Themen (zu Tasks, Konzepten und Referenzinformationen)
  - Hilfe für DB2-Tools
  - Beispielprogramme
  - Lernprogramme
- DB2-Bücher
  - PDF-Dateien (für den Download verfügbar)
  - PDF-Dateien (auf der DB2-PDF-DVD)
  - Gedruckte Bücher
- Befehlszeilenhilfe
  - Hilfe für Befehle
  - Hilfe für Nachrichten

Anmerkung: Die Themen der *DB2-Informationszentrale* werden häufiger aktualisiert als die PDF- und Hardcopybücher. Um stets die neuesten Informationen zur Verfügung zu haben, sollten Sie die Dokumentationsaktualisierungen installieren, sobald diese verfügbar sind, oder die *DB2-Informationszentrale* unter ibm.com aufrufen.

Darüber hinaus können Sie auf zusätzliche technische Informationen zu DB2, wie beispielsweise technische Hinweise (Technotes), White Papers und IBM Redbooks, online über ibm.com zugreifen. Rufen Sie die Website 'DB2 Information Management - Software - Library' unter http://www.ibm.com/software/data/sw-library/ auf.

Feedback zur Dokumentation

Senden Sie uns Ihr Feedback zur DB2-Dokumentation! Wenn Sie Anregungen zur Verbesserung der DB2-Dokumentation haben, senden Sie eine E-Mail an db2docs@ca.ibm.com. Das DB2-Dokumentationsteam bearbeitet das gesamte Feedback, kann jedoch nicht im Einzelnen auf Ihre E-Mails antworten. Nennen Sie uns, wenn möglich, konkrete Beispiele, sodass wir die Problemstellung besser beurteilen können. Wenn Sie uns Feedback zu einem bestimmten Thema oder einer bestimmten Hilfedatei senden, geben Sie den entsprechenden Titel sowie die URL an.

Verwenden Sie diese E-Mail-Adresse nicht, wenn Sie sich an die DB2-Kundenunterstützung wenden möchten. Wenn ein technisches Problem bei DB2 vorliegt, das Sie mithilfe der Dokumentation nicht beheben können, fordern Sie beim zuständigen IBM Service-Center Unterstützung an.

Wenn Sie IBM dabei unterstützen möchten, die Benutzerfreundlichkeit der IBM Information Management-Produkte weiter zu verbessern, können Sie uns Ihr Feedback mithilfe der Umfrage zur Verbraucherfreundlichkeit von Software geben: http://www.ibm.com/software/data/info/consumability-survey/ (landessprachliche Version unter https://www-950.ibm.com/survey/oid/wsb.dll/studies/consumabilitywebform.htm?renderlang=de).

Bibliothek mit technischen Informationen zu DB2 im Hardcopy- oder PDF-Format

Die folgenden Tabellen enthalten eine Beschreibung der DB2-Bibliothek, die im IBM Publications Center unter www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss zur Verfügung steht. Englische und übersetzte Versionen der DB2 Version 9.5-Handbücher im PDF-Format können über die folgende Adresse heruntergeladen werden: www.ibm.com/support/docview.wss?uid=swg27009727 bzw. www.ibm.com/support/docview.wss?uid=swg27009728.

In den Tabellen sind die Bücher, die in gedruckter Form zur Verfügung stehen, gekennzeichnet; möglicherweise sind diese in Ihrem Land oder Ihrer Region jedoch nicht verfügbar.

Die Formnummer wird bei jeder Aktualisierung eines Handbuchs erhöht. Anhand der nachfolgenden Liste können Sie sicherstellen, dass Sie die jeweils neueste Version des Handbuchs lesen.

Anmerkung: Die *DB2-Informationszentrale* wird häufiger aktualisiert als die PDF- und Hardcopybücher.

Tabelle 49. Technische Informationen zu DB2

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
Administrative API Reference	SC23-5842-03	Ja	Dezember 2010
Administrative Routines and Views	SC23-5843-03	Nein	Dezember 2010
Call Level Interface Guide and Reference, Volume 1	SC23-5844-03	Ja	Dezember 2010
Call Level Interface Guide and Reference, Volume 2	SC23-5845-03	Ja	Dezember 2010
Command Reference	SC23-5846-03	Ja	Dezember 2010
Dienstprogramme für das Versetzen von Daten - Handbuch und Referenz	SC12-3917-03	Ja	Dezember 2010
Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz	SC12-3919-03	Ja	Dezember 2010
Datenserver, Datenbanken und Datenbankobjekte	SC12-3912-03	Ja	Dezember 2010
Datenbanksicherheit	SC12-3914-03	Ja	Dezember 2010
Developing ADO.NET and OLE DB Applications	SC23-5851-02	Ja	April 2009
Developing Embedded SQL Applications	SC23-5852-02	Ja	April 2009
Developing Java Applications	SC23-5853-03	Ja	Dezember 2010

Tabelle 49. Technische Informationen zu DB2 (Forts.)

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
Developing Perl and PHP Applications	SC23-5854-02	Nein	April 2009
Developing User-defined Routines (SQL and External)	SC23-5855-03	Ja	Dezember 2010
Getting Started with Database Application Development	GC23-5856-03	Ja	Dezember 2010
Installation und Verwaltung von DB2 unter Linux und Windows - Erste Schritte	GC12-3922-03	Ja	Dezember 2010
Internationalisierung	SC12-3916-02	Ja	April 2009
Fehlernachrichten, Band 1	GI11-3098-00	Nein	
Fehlernachrichten, Band 2	GI11-3099-00	Nein	
Migration	GC12-3921-03	Ja	Dezember 2010
Net Search Extender - Verwaltung und Benutzerhandbuch	SC12-3979-02	Ja	April 2009
Partitionierung und Clustering	SC12-3915-03	Ja	Dezember 2010
Query Patroller - Verwaltung und Benutzerhandbuch	SC12-3977-01	Ja	April 2009
IBM Data Server-Clients - Einstieg	GC12-3924-03	Nein	Dezember 2010
DB2-Server - Einstieg	GC12-3923-03	Ja	Dezember 2010
Spatial Extender und Geodetic Data Management Feature - Benutzer- und Referenzhandbuch	SC12-3978-02	Ja	April 2009
SQL Reference, Volume 1	SC23-5861-03	Ja	Dezember 2010
SQL Reference, Volume 2	SC23-5862-03	Ja	Dezember 2010
Systemmonitor - Handbuch und Referenz	SC12-3918-03	Ja	Dezember 2010
Text Search	SC12-3920-02	Ja	Dezember 2010
Fehlerbehebung	GI11-3097-03	Nein	Dezember 2010
Optimieren der Datenbankleistung	SC12-3913-03	Ja	Dezember 2010
Lernprogramm für Visual Explain	SC12-3932-00	Nein	
Neuerungen	SC12-3928-03	Ja	Dezember 2010
Workload-Manager - Handbuch und Referenz	SC12-3929-03	Ja	Dezember 2010

Tabelle 49. Technische Informationen zu DB2 (Forts.)

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
pureXML - Handbuch	SC12-3930-03	Ja	Dezember 2010
XQuery - Referenz	SC12-3931-02	Nein	April 2009

Tabelle 50. Technische Informationen zu DB2 Connect

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
DB2 Connect Personal Edition - Einstieg	GC12-3926-03	Ja	Dezember 2010
DB2 Connect-Server - Einstieg	GC12-3927-03	Ja	Dezember 2010
DB2 Connect - Benutzerhandbuch	SC12-3925-03	Ja	Dezember 2010

Tabelle 51. Technische Informationen zu Information Integration

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
Information Integration: Föderierte Systeme - Verwaltung	SC12-3759-01	Ja	
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018-02	Ja	
Information Integration: Konfiguration föderierter Datenquellen	SC12-3777-01	Nein	
Information Integration: SQL Replication - Handbuch und Referenz	SC12-3782-01	Ja	
Information Integration: Replikation und Event-Publishing - Einführung	GC12-3779-01	Ja	

Bestellen gedruckter DB2-Bücher

Gedruckte DB2-Bücher können Sie in den meisten Ländern oder Regionen online bestellen. Das Bestellen gedruckter DB2-Bücher ist stets über den zuständigen IBM Ansprechpartner möglich. Beachten Sie hierbei bitte, dass einige Softcopybücher auf der DVD mit der *DB2-PDF-Dokumentation* nicht in gedruckter Form verfügbar sind. So sind beispielsweise die beiden Bände des Handbuchs *DB2 Fehlernachrichten* nicht in gedruckter Form erhältlich.

Gedruckte Versionen vieler DB2-Bücher, die auf der DVD mit der DB2-PDF-Dokumentation verfügbar sind, können gegen eine Gebühr bei IBM bestellt werden.

Abhängig vom jeweiligen Land bzw. der jeweiligen Region können Sie Bücher möglicherweise online über das IBM Publications Center bestellen. Ist im jeweiligen Land bzw. der jeweiligen Region keine Onlinebestellung möglich, können Sie gedruckte DB2-Bücher stets über den zuständigen IBM Ansprechpartner bestellen. Nicht alle Bücher, die auf der DVD mit der DB2-PDF-Dokumentation verfügbar sind, können in gedruckter Form bestellt werden.

Anmerkung: Über http://publib.boulder.ibm.com/infocenter/db2luw/v9r5 haben Sie Zugriff auf die DB2-Informationszentrale, wo Sie die neueste und umfassendste DB2-Dokumentation finden.

Gehen Sie wie folgt vor, um gedruckte DB2-Bücher zu bestellen:

- Informationen dazu, ob in Ihrem Land oder Ihrer Region die Bestellung von gedruckten DB2-Büchern möglich ist, finden Sie auf der Website mit dem IBM Publications Center unter http://www.ibm.com/shop/publications/order. Wählen Sie ein Land, eine Region oder eine Sprache aus, um die Bestellinformationen für Veröffentlichungen aufzurufen, und führen Sie dann die entsprechenden Schritte des Bestellverfahrens für Ihr Land bzw. Ihre Region aus.
- Gehen Sie wie folgt vor, um gedruckte DB2-Bücher beim zuständigen IBM Ansprechpartner zu bestellen:
  1. Kontaktinformationen zum zuständigen Ansprechpartner finden Sie auf einer der folgenden Websites:
     - IBM Verzeichnis weltweiter Kontakte unter www.ibm.com/planetwide.
     - Website mit IBM Veröffentlichungen unter http://www.ibm.com/shop/publications/order. Wählen Sie das gewünschte Land, die gewünschte Region oder die gewünschte Sprache aus, um auf die entsprechende Homepage mit Veröffentlichungen Ihres Landes bzw. Ihrer Region zuzugreifen. Folgen Sie auf dieser Seite dem Link für Informationen zu dieser Site ("About this Site").
  2. Geben Sie bei Ihrem Anruf an, dass Sie eine DB2-Veröffentlichung bestellen möchten.
  3. Teilen Sie dem zuständigen Ansprechpartner die Titel und Formularnummern der Bücher mit, die Sie bestellen möchten. Titel und Formularnummern finden Sie unter „Bibliothek mit technischen Informationen zu DB2 im Hardcopy- oder PDF-Format” auf Seite 474.

Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor

DB2 gibt für Bedingungen, die aufgrund einer SQL-Anweisung generiert werden können, einen SQLSTATE-Wert zurück. Die SQLSTATE-Hilfe erläutert die Bedeutung der SQL-Statuswerte und der SQL-Statusklassencodes.

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.

Zugriff auf verschiedene Versionen der DB2-Informationszentrale

Für Themen aus DB2 Version 9.8 lautet die URL der *DB2-Informationszentrale* http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/.

Für Themen aus DB2 Version 9.7 lautet die URL der *DB2-Informationszentrale* http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/.

Für Themen aus DB2 Version 9.5 lautet die URL der *DB2-Informationszentrale* http://publib.boulder.ibm.com/infocenter/db2luw/v9r5.

Für Themen aus DB2 Version 9.1 lautet die URL der *DB2-Informationszentrale* http://publib.boulder.ibm.com/infocenter/db2luw/v9/.

Für Themen aus DB2 Version 8 lautet die URL der *DB2-Informationszentrale (Version 8, 'Information - Unterstützung')* http://publib.boulder.ibm.com/infocenter/db2luw/v8/.

Anzeigen von Themen in der gewünschten Sprache in der DB2-Informationszentrale

In der DB2-Informationszentrale 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 der DB2-Informationszentrale in Englisch angezeigt.

- 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. Löschen Sie den Inhalt des Browser-Cache und aktualisieren Sie anschließend die Seite, um die DB2-Informationszentrale in der gewünschten Sprache anzuzeigen.
- Um Themen in der gewünschten Sprache in einem Firefox- oder Mozilla-Browser anzuzeigen, gehen Sie wie folgt vor:
  1. Wählen Sie den Knopf im Bereich **Languages** des Dialogfensters **Tools** —> **Options** —> **Advanced** 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 zur Liste hinzufügen möchten, klicken Sie den Knopf **Add...** 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 **Move Up** aus, bis die Sprache an erster Stelle in der Liste steht.

3. Löschen Sie den Inhalt des Browser-Cache und aktualisieren Sie anschließend die Seite, um die DB2-Informationszentrale in der gewünschten Sprache anzuzeigen.

Bei einigen Kombinationen aus Browser und Betriebssystem müssen Sie möglicherweise auch die Ländereinstellungen des Betriebssystems in die gewünschte Locale und Sprache ändern.

Aktualisieren der auf Ihrem Computer oder Intranet-Server installierten DB2-Informationszentrale

Wenn Sie die DB2-Informationszentrale lokal installiert haben, können Sie Dokumentationsaktualisierungen von IBM abrufen und installieren.

Zur Aktualisierung der lokal installierten *DB2-Informationszentrale* sind die folgenden Schritte erforderlich:

1. Stoppen Sie die *DB2-Informationszentrale* auf Ihrem Computer und starten Sie die Informationszentrale im Standalone-Modus erneut. Die Ausführung der Informationszentrale im Standalone-Modus verhindert, dass andere Benutzer in Ihrem Netz auf die Informationszentrale zugreifen, und ermöglicht das Anwenden von Aktualisierungen. *DB2-Informationszentralen*, deren Installation nicht als Administrator oder Root ausgeführt wurde, werden stets im Standalone-Modus ausgeführt.
2. Verwenden Sie die Aktualisierungsfunktion, um zu prüfen, welche Aktualisierungen verfügbar sind. Falls Aktualisierungen verfügbar sind, die Sie installieren möchten, können Sie die Aktualisierungsfunktion verwenden, um diese abzurufen und zu installieren.

   Anmerkung: Wenn es in der verwendeten Umgebung erforderlich ist, die Aktualisierungen für die *DB2-Informationszentrale* auf einer Maschine zu installieren, die nicht über ein Verbindung zum Internet verfügt, müssen Sie die Aktualisierungssite auf ein lokales Dateisystem spiegeln und dabei eine Maschine verwenden, die mit dem Internet verbunden ist und auf der die *DB2-Informationszentrale* installiert ist. Wenn viele Benutzer Ihres Netzes die Dokumentationsaktualisierungen installieren sollen, können Sie die Zeit, die jeder einzelne Benutzer für die Aktualisierungen benötigt, reduzieren, indem Sie die Aktualisierungssite lokal spiegeln und ein Proxy dafür erstellen.
   Ist dies der Fall, verwenden Sie die Aktualisierungsfunktion, um die Pakete abzurufen. Die Aktualisierungsfunktion ist jedoch nur im Standalone-Modus verfügbar.
3. Stoppen Sie die im Standalone-Modus gestartete Informationszentrale und starten Sie die *DB2-Informationszentrale* auf Ihrem Computer erneut.

Anmerkung: Unter Windows Vista müssen Sie zur Ausführung der nachfolgend aufgeführten Befehle über Administratorberechtigung verfügen. Zum Starten einer Eingabeaufforderung oder eines Grafiktools mit vollen Administratorberechtigungen klicken Sie mit der rechten Maustaste die Verknüpfung an und wählen Sie **Als Administrator ausführen** aus.

Gehen Sie wie folgt vor, um die auf Ihrem Computer bzw. Intranet-Server installierte *DB2-Informationszentrale* zu aktualisieren:

1. Stoppen Sie die *DB2-Informationszentrale*.
   - Unter Windows klicken Sie **Start** → **Einstellungen** → **Systemsteuerung** → **Verwaltung** → **Dienste** an. Klicken Sie mit der rechten Maustaste die **DB2-Informationszentrale** an und wählen Sie **Stoppen** aus.
   - Unter Linux: Geben Sie den folgenden Befehl ein:
     ```
     /etc/init.d/db2icdv95 stop
     ```
2. Starten Sie die Informationszentrale im Standalone-Modus.
   - Unter Windows:
     a. Öffnen Sie ein Befehlsfenster.
     b. Navigieren Sie zu dem Pfad, in dem die Informationszentrale installiert ist. Standardmäßig ist die *DB2-Informationszentrale* im Verzeichnis *Programme*\IBM\DB2 Information Center\Version 9.5 installiert, wobei *Programme* die Position des Programmdateiverzeichnisses ist.
     c. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis doc\bin.
     d. Führen Sie die Datei help_start.bat aus:
        ```
        help_start.bat
        ```
   - Unter Linux:
     a. Navigieren Sie zu dem Pfad, in dem die Informationszentrale installiert ist. Standardmäßig ist die *DB2-Informationszentrale* im Verzeichnis /opt/ibm/db2ic/V9.5 installiert.
     b. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis doc/bin.
     c. Führen Sie das Script help_start aus:
        ```
        help_start
        ```

   Der standardmäßig auf dem System verwendete Web-Browser wird aufgerufen und zeigt die Standalone-Informationszentrale an.
3. Klicken Sie den Aktualisierungsknopf ( ) an. Klicken Sie im rechten Fenster der Informationszentrale den Knopf für die Suche nach Aktualisierungen an. Eine Liste der Aktualisierungen für die vorhandene Dokumentation wird angezeigt.
4. Wählen Sie zum Initiieren des Installationsprozesses die gewünschten Aktualisierungen aus und klicken Sie anschließend den Knopf für die Installation der Aktualisierungen an.
5. Klicken Sie nach Abschluss des Installationsprozesses **Fertig stellen** an.
6. Stoppen Sie die im Standalone-Modus gestartete Informationszentrale:
   - Unter Windows: Navigieren Sie in das Verzeichnis doc\bin des Installationsverzeichnisses und führen Sie die Datei help_end.bat aus:
     ```
     help_end.bat
     ```

     Anmerkung: Die Stapeldatei help_end enthält die Befehle, die erforderlich sind, um die Prozesse, die mit der Stapeldatei help_start gestartet wurden, ordnungsgemäß zu beenden. Verwenden Sie nicht die Tastenkombination Strg+C oder eine andere Methode, um help_start.bat zu beenden.
   - Unter Linux: Navigieren Sie in das Verzeichnis doc/bin des Installationsverzeichnisses und führen Sie das Script help_end aus:
     ```
     help_end
     ```

Anmerkung: Das Script `help_end` enthält die Befehle, die erforderlich sind, um die Prozesse, die mit dem Script `help_start` gestartet wurden, ordnungsgemäß zu beenden. Verwenden Sie keine andere Methode, um das Script `help_start` zu beenden.

7. Starten Sie die *DB2-Informationszentrale* erneut.
   - Unter Windows klicken Sie **Start** → **Einstellungen** → **Systemsteuerung** → **Verwaltung** → **Dienste** an. Klicken Sie mit der rechten Maustaste die **DB2-Informationszentrale** an und wählen Sie **Start** aus.
   - Unter Linux: Geben Sie den folgenden Befehl ein:
     ```
     /etc/init.d/db2icdv95 start
     ```

In der aktualisierten *DB2-Informationszentrale* werden die neuen und aktualisierten Themen angezeigt.

DB2-Lernprogramme

Die DB2-Lernprogramme unterstützen Sie dabei, sich mit den unterschiedlichen Aspekten der DB2-Produkte vertraut zu machen. Die Lerneinheiten bieten eine in einzelne Schritte unterteilte Anleitung.

Vorbereitungen

Die XHTML-Version des Lernprogramms kann über die Informationszentrale unter http://publib.boulder.ibm.com/infocenter/db2help/ angezeigt werden.

In einigen der Lerneinheiten werden Beispieldaten und Codebeispiele verwendet. Informationen zu bestimmten Voraussetzungen für die Ausführung der Tasks finden Sie in der Beschreibung des Lernprogramms.

DB2-Lernprogramme

Klicken Sie zum Anzeigen des Lernprogramms den Titel an.

„pureXML" in *pureXML - Handbuch*
Einrichten einer DB2-Datenbank, um XML-Daten zu speichern und Basisoperationen mit dem nativen XML-Datenspeicher auszuführen.

„Visual Explain" in *Lernprogramm für Visual Explain*
Analysieren, Optimieren und Anpassen von SQL-Anweisungen zur Leistungsverbesserung mithilfe 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-Datenbankprodukten zu unterstützen.

DB2-Dokumentation
Informationen zur Fehlerbehebung stehen im Handbuch DB2-Fehlerbehebung oder im Abschnitt mit grundlegenden Informationen zu Datenbanken in der DB2-Informationszentrale zur Verfügung. Dort finden Sie Informationen dazu, wie Sie Probleme mithilfe der DB2-Diagnosetools und -Dienstprogramme eingrenzen und identifizieren können, Lösungen für einige der häufigsten Probleme sowie weitere Hinweise zur Behebung von Fehlern und Problemen, die bei der Verwendung der DB2-Datenbankprodukte auftreten können.

DB2-Website mit technischer Unterstützung

Auf der DB2-Website mit technischer Unterstützung finden Sie Informationen zu Problemen und den möglichen Ursachen und Fehlerbehebungsmaßnahmen. Die Website mit technischer Unterstützung enthält Links zu den neuesten DB2-Veröffentlichungen, technischen Hinweisen (TechNotes), APARs (Authorized Program Analysis Reports) und Fehlerkorrekturen, Fixpacks sowie weiteren Ressourcen. Sie können diese Wissensbasis nach möglichen Lösungen für aufgetretene Probleme durchsuchen.

Rufen Sie die DB2-Website mit technischer Unterstützung unter http://www.ibm.com/software/data/db2/support/db2_9/ auf.

Bedingungen

Die Berechtigungen zur Nutzung dieser Veröffentlichungen werden Ihnen auf der Basis der folgenden Bedingungen gewährt.

Persönliche Nutzung: Sie dürfen diese Veröffentlichungen für Ihre persönliche, nicht kommerzielle Nutzung unter der Voraussetzung vervielfältigen, dass alle Eigentumsvermerke erhalten bleiben. Sie dürfen diese Veröffentlichungen oder Teile der Veröffentlichungen ohne ausdrückliche Genehmigung von IBM nicht weitergeben, anzeigen oder abgeleitete Werke davon erstellen.

Kommerzielle Nutzung: Sie dürfen diese Veröffentlichungen nur innerhalb Ihres Unternehmens und unter der Voraussetzung, dass alle Eigentumsvermerke erhalten bleiben, vervielfältigen, weitergeben und anzeigen. Sie dürfen diese Veröffentlichungen oder Teile der Veröffentlichungen ohne ausdrückliche Genehmigung von IBM außerhalb Ihres Unternehmens nicht vervielfältigen, weitergeben, anzeigen oder abgeleitete Werke davon erstellen.

Abgesehen von den hier gewährten Berechtigungen erhalten Sie keine weiteren Berechtigungen, Lizenzen oder Rechte (veröffentlicht oder stillschweigend) in Bezug auf die Veröffentlichungen oder darin enthaltene Informationen, Daten, Software oder geistiges Eigentum.

IBM behält sich das Recht vor, die in diesem Dokument gewährten Berechtigungen nach eigenem Ermessen zurückzuziehen, wenn sich die Nutzung der Veröffentlichungen für IBM als nachteilig erweist oder wenn die obigen Nutzungsbestimmungen nicht genau befolgt werden.

Sie dürfen diese Informationen nur in Übereinstimmung mit allen anwendbaren Gesetzen und Vorschriften, einschließlich aller US-amerikanischen Exportgesetze und Verordnungen, herunterladen und exportieren.

IBM übernimmt keine Gewährleistung für den Inhalt dieser Informationen. Diese Veröffentlichungen werden auf der Grundlage des gegenwärtigen Zustands (auf "as-is"-Basis) und ohne eine ausdrückliche oder stillschweigende Gewährleistung für die Handelsüblichkeit, die Verwendungsfähigkeit oder die Freiheit der Rechte Dritter zur Verfügung gestellt.

Anhang E. Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der Produkte, Programme oder Services können auch andere ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder andere 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 Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Defense
France

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in diesem Handbuch werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen oder in Technical News Letters (TNLs) bekannt gegeben. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Dieses Dokument enthält möglicherweise Links oder Verweise auf Websites und Ressourcen anderer Anbieter. Es bestehen keine Zusicherungen, Gewährleistungen oder Verpflichtungen von IBM hinsichtlich der Websites oder Ressourcen anderer Anbieter, auf die im vorliegenden Dokument verwiesen wird, Zugriff besteht oder Links vorhanden sind. Ein Link auf eine Website eines anderen Anbieters bedeutet nicht, dass IBM den Inhalt und die Verwendung dieser Website billigt oder deren Eigentümer anerkennt. Darüber hinaus ist IBM nicht an Transaktionen beteiligt und übernimmt keine Verantwortung für Transaktionen zwischen Ihnen und anderen Anbietern, auch wenn die Informationen (oder Links) zu diesen Anbietern auf einer IBM Website zur Verfügung stehen. IBM ist nicht für die Verfügbarkeit solcher externen Sites oder Ressourcen verantwortlich und übernimmt keine Verantwortung oder Haftung für Inhalte, Services, Produkte oder sonstiges Material, die bzw. das auf diesen oder über diese Sites oder Ressourcen verfügbar sind. Die Software anderer Anbieter unterliegt den Lizenzbedingungen der jeweiligen Software.

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
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
CANADA

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des im Dokument aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung sowie der Allgemeinen Geschäftsbedingungen von IBM, der Internationalen Nutzungsbedingungen der IBM für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichen Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Diese Veröffentlichung kann Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes enthalten. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren; sie können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung kann Musteranwendungsprogramme enthalten, die in Quellensprache geschrieben sind und Programmiertechniken in verschiedenen Betriebsumgebungen veranschaulichen. Sie dürfen diese Musterprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, verwenden, vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle konform sind, für die diese Musterprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten.

Kopien oder Teile der Musterprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten:

© (*Name Ihrer Firma*) (*Jahr*). Teile des vorliegenden Codes wurden aus Musterprogrammen der IBM Corp. abgeleitet. © Copyright IBM Corp. *_Jahr/Jahre angeben_*. Alle Rechte vorbehalten.

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der IBM Corporation in den USA und/oder anderen Ländern. Weitere Produkt- oder Servicenamen können Marken von IBM oder anderen Herstellern sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite Copyright and trademark information unter www.ibm.com/legal/copytrade.shtml.

Die folgenden Namen sind Marken oder eingetragene Marken anderer Unternehmen.

- Linux ist eine eingetragene Marke von Linus Torvalds in den USA und/oder anderen Ländern.
- Java und alle auf Java basierenden Marken und Logos sind Marken von Sun Microsystems, Inc. in den USA und/oder anderen Ländern.
- UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.
- Intel, das Intel-Logo, Intel Inside, das Intel Inside-Logo, Intel Centrino, das Intel Centrino-Logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium und Pentium sind Marken oder eingetragene Marken der Intel Corporation oder deren Tochtergesellschaften in den USA oder anderen Ländern.
- Microsoft, Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

Weitere Unternehmens-, Produkt- oder Servicenamen können Marken anderer Hersteller sein.

Index

A

Abfrageinterne Parallelität 85
Abfragen
  mehrdimensionales Clustering (MDC) 45
  Parallelität 85
Abfrageoptimierung
  Auswirkung von Datenbankpartitionsgruppen 361
Abfrageübergreifende Parallelität 85
ADMIN_CMD, Prozedur
  unterstützte Befehle
    GET STMM TUNING DBPARTITIONNUM 455
    UPDATE STMM TUNING DBPARTITIONNUM 456
Agenten
  in einer partitionierten Datenbank 345
AIX
  Befehle an ESE-Workstations verteilen 117
  Erstellung erforderlicher Benutzer 125
  Erstellung von DB2-Ausgangsdateisystemen 122
  Installation von DB2-Servern 106
  prüfen, ob NFS aktiv ist 118
  Umgebungseinstellungen aktualisieren 115
Aktualisierungen
  DB2-Informationszentrale 479
  db2nodes.cfg (UNIX) 150
  Knotenkonfigurationsdatei 150
ALTER DATABASE PARTITION GROUP, Anweisung 449
ALTER NODEGROUP, Anweisung
  siehe ALTER DATABASE PARTITION GROUP 449
Antwortdateien
  Installation
    Datenbankpartitionsserver 131, 132
Antwortzeit für Verbindung, Konfigurationsparameter 406
APIs
  sqleaddn 415
  sqlecran 417
  sqledpan 418
  sqledrpn 419
  sqlugrpn 421
Assistenten
  Assistent für das Hinzufügen von Datenbankpartitionen 175
Aufbau des Handbuchs ix
Aufgehobene Zuordnung, Datenpartitionen
  Attribute 228
  Beschreibung 227
Ausgangsdateisystem
  AIX 122
Ausgefallener Datenbankpartitionsserver 286
Auswählen
  Dimensionen mehrdimensionaler Tabellen 45
Authentifizierung
  Hinweise zu partitionierten Datenbanken 5
Automatischer Neustart 285

B

BACKUP DATABASE, Befehl 465
Backup-Dienstprogramm
  Einschränkungen 465
Bedingungen
  Verwendung der Veröffentlichungen 482
Beendigung
  Ladeoperationen
    in Mehrpartitionsdatenbanken 258
Befehle
  db2adutl
    knotenübergreifende Recovery, Beispiel 291
  db2nchg 103, 436
  db2ncrt 437
  db2ndrop 439
  GET STMM TUNING DBPARTITIONNUM 455
  parallel ausführen 183
  REDISTRIBUTE DATABASE PARTITION GROUP 425
  UPDATE STMM TUNING DBPARTITIONNUM 456
Befehle 'rah' und 'db2_all'
  Präfixsequenzen 185
Bemerkungen 483
Benutzer
  Erstellung erforderlicher Benutzer (Linux) 124
  Erstellung erforderlicher Benutzer unter AIX 125
Bereiche
  Einschränkungen 196
  für Datenpartitionen definieren 196
  generieren 196
Bereichsclustertabellen
  Algorithmen 206
  Beschreibung 39
  Indizes 207
  Inkompatibilitäten 40
  Richtlinien 39, 208
  Satzschlüssel außerhalb des Bereichs 41
  Satzschlüssel außerhalb des zulässigen Bereichs 39
  Szenarios 209
  Tabellensperren 39, 41
  Unterschiede zu regulären Tabellen 207
  Vorteile 39
  Zugriffspfadsbestimmung 39, 208
Bereichspartitionierte Tabellen
  siehe partitionierte Tabellen 13
Bereichspartitionierung
  siehe Datenpartition 13
  siehe Datenpartitionen 15
  siehe Tabellenpartitionierung 13
Bestellen von DB2-Büchern 476
Bücher
  gedruckt
    bestellen 476

C

Clusterindizes
  mit partitionierten Tabellen 327
  Vorteile bei partitionierten Tabellen 327
Clustering
  Daten 43
  Definition 350
conn_elapse, Konfigurationsparameter 406
Container
  SMS-Tabellenbereichen hinzufügen 177
CREATE DATABASE PARTITION GROUP, Anweisung 452

CREATE NODEGROUP, Anweisung
  CREATE DATABASE PARTITION GROUP, Anweisung 452
CREATE TABLE, Anweisung
  OVERFLOW, Klausel 39, 41
CURRENT DBPARTITIONNUM, Sonderregister 443

D

Database Partitioning Feature (DPF)
  Kommunikation aktivieren 159
  Übersicht 89
DATAPARTITIONNUM, Skalarfunktion 445
Dateigruppen
  Beschreibung 113, 157
  db2fcmr-Dämonen 113, 157
  db2fcms-Dämonen 113, 157
Dateisysteme
  Erstellung für ein partitioniertes Datenbanksystem
    Linux 120
Daten
  Organisationsschemata
    Übersicht 16
    Vergleich mit Informix 21
  Partitionierung 4
  Tabellenverteilung 16
  Umverteilung
    Assistent zur Datenumverteilung 392
    Datenbankpartitionsgruppe ändern 217
    Ereignisprotokollierung und Fehlerbehebung 394
    Erforderlichkeit feststellen 389
    Protokollspeicherbedarf 393
    REDISTRIBUTE DATABASE PARTITION GROUP, Befehl 425
    Übersicht 387, 390
  Verteilung 89
Datenbank auf Knoten erstellen, API 417
Datenbanken
  Datenbankpartitionsgruppen ändern 217
  Datenpartitionierung aktivieren 3, 139
  Datenverteilung
    ändern 217
  erneut erstellen
    partitionierte 290
  erstellen
    Umgebungen mit partitionierten Datenbanken 3, 139
  über mehrere Partitionen konfigurieren 403
Datenbankkonfigurationsdatei
  ändern 217
Datenbankkonfigurationsparameter
  autorestart 285
Datenbankmanager
  Zeitlimit für DB2START 410
  Zeitlimit für DB2STOP 410
Datenbankpartitionen
  aus Instanz löschen 178
  Datenbankkonfiguration aktualisieren 217
  hinzufügen
    aktivem System 167
    gestopptem System (UNIX) 169
    gestopptem System (Windows) 168
    Übersicht 166
  Hinzufügen über einen Assistenten 175
  Katalog 3, 139
  Übersicht 89
  Uhren synchronisieren 302
  unter Windows ändern 175
Datenbankpartitionen *(Forts.)*
  verwalten 164
    über die Steuerzentrale 217
Datenbankpartitionsgruppen
  ändern 217
  Auswirkung auf Abfrageoptimierung 361
  Datenposition bestimmen 8
  entwerfen 7
  erste 163
  erstellen 163, 452
  IBMDEFAULTGROUP 193
  Kollokation (Zusammenfassen von Tabellen) 7
  Partitionen hinzufügen 449
  Partitionen löschen 449
  Tabellen 193
  Übersicht 6
  Verteilungszuordnung erstellen 452
Datenbankpartitionskompatibilität
  Übersicht 441
Datenbankpartitionsserver
  angeben 150
  Befehle absetzen 181, 305
  Installation mit einer Antwortdatei 132
  Kommunikation aktivieren (UNIX) 159
  löschen 177
  mehrere logische Knoten 152
  mehrerer logische Partitionen 152
  Recovery nach Fehler 290
Datenbankpartitionsserver einer Instanz hinzufügen, Befehl 437
Datenpartitionen
  ändern 220
  Attribute 228
  Bereichsdefinition 196
  erstellen 196
  hinzufügen 230
  löschen 232
  Rollin von Daten
    Szenario 235
    Übersicht 219, 221
  Rollout von Daten
    Szenario 235
    Übersicht 219, 227
  rotieren
    Szenario 233
  Übersicht 13, 15, 16
  zuordnen
    Szenario 235
    Übersicht 219, 221
  Zuordnung aufheben
    Szenario 235
    Übersicht 219, 227
Datenpartitionen, Ausschluss 314
Datenserver
  Kapazitätsmanagement 156
Datentypen
  Partitionskompatibilität 441
DB_PARTITIONS, Tabellenfunktion 457
db2_all, Befehl
  angeben 182
  Übersicht 182, 184
  Umgebungen mit partitionierten Datenbanken 181, 305
db2_call_stack, Befehl 184
DB2-Informationszentrale
  Aktualisierung 479
  in verschiedenen Sprachen anzeigen 478
  Sprachen 478

DB2-Informationszentrale *(Forts.)*
Versionen 478
DB2-Installationsassistent
DB2-Server installieren
Linux 110
UNIX 110
Installation von DB2-Servern unter UNIX 110
db2_kill, Befehl 184
DB2_NUM_FAILOVER_NODES, Registrierdatenbankvariable 404
DB2_PARTITIONEDLOAD_DEFAULT, Registrierdatenbankvariable
Beschreibung 404
DB2-Server
Installation
Linux 106
UNIX 106
Windows 101
partitioniert
Windows-Umgebung vorbereiten 103
db2adutl, Befehl
knotenübergreifende Recovery, Beispiel 291
db2Backup, API
Backup von Daten 465
DB2CHGPWD_EEE, Registrierdatenbankvariable 404
db2exfmt, Befehl
Ausgabebeispiele 352
db2expln, Befehl
Ausgabebeispiele
Beschreibung 352
keine Parallelität 352
Zugriffsplan für Einzelpartition mit partitionsinterner Parallelität 354
Zugriffsplan für mehrere Partitionen mit partitionsübergreifender Parallelität 356
Zugriffsplan für mehrere Partitionen mit voller Parallelität 358
db2fcmr-Dämonen 113, 157
db2fcms-Dämonen 113, 157
db2nchg, Befehl
Beschreibung 436
Konfigurationen von Datenbankpartitionsservern ändern 175
db2ncrt, Befehl
Beschreibung 437
Datenbankpartitionsserver hinzufügen 173
db2ndrop, Befehl
Beschreibung 439
Datenbankpartitionsserver löschen 177
db2nlist, Befehl 173
db2nodes.cfg, Datei
Aktualisierung 150
ALTER DATABASE PARTITION GROUP, Anweisung 449
CREATE DATABASE PARTITION GROUP, Anweisung 452
DBPARTITIONNUM, Funktion 446
erstellen 140
Format 142
netname, Feld 103
DB2PORTRANGE, Registrierdatenbankvariable 404
db2start addnode, Befehl 173
DBPARTITIONNUM, Funktion
Beschreibung 446
Designadvisor
aus Einzelpartitionsdatenbanken auf Mehrpartitionsdatenbank migrieren 331
Dienstprogrammparallelität 85
Dimensionen
mehrdimensionale Tabellen 45
Dokumentation
gedruckt 474
Nutzungsbedingungen 482
PDF 474
Übersicht 473
DPF (Database Partitioning Feature)
siehe Database Partitioning Feature (DPF) 89

E

Ein-/Ausgabe
Parallelität 85
Einstellungen
Standardumgebungsprofil für rah 192
Einzelpartition
Einzelprozessorumgebung 90
Mehrprozessorumgebung 90
Einzelprozessorumgebung 90
Entclusterung
partiell 4, 89
Ereignismonitore
erstellen
partitionierte Datenbanken 281
Erstellen
mehrdimensionale Tabellen 56, 211
Erstellung
erforderlicher Benutzer unter AIX 125
erforderlicher Benutzer unter Linux 124

F

FCM (Fast Communication Manager)
Kommunikation zwischen Datenbankpartitionsservern aktivieren 159
Nachrichtenpuffer 105, 157
Portnummern 159
Service-Eintrag, Syntax 158
Übersicht 105, 157
Windows 105, 157
FCM (Fast Communications Manager)
Kanäle 408
Monitorelemente
fcm_num_channels 408
fcm_num_buffers, Konfigurationsparameter 105, 157, 406
fcm_num_channel, Konfigurationsparameter 408
Fehlerbehebung
Beschreibung 305
Lernprogramme 481
Onlineinformationen 481
Fehlerbestimmung
Lernprogramme 481
verfügbare Informationen 481
Fehlernachrichten
partitionierte Datenbanken 171
Fragmenteliminierung
siehe Datenpartitionsausschluss 314
Fragmentierung nach Ausdruck
Vergleich mit Tabellenpartitionierung 21
Free Space Control Record (FSCR, Steuersatz für freien Speicherbereich)
in MDC-Tabellen 348
Funktionen
Skalar
DBPARTITIONNUM 446

Funktionen *(Forts.)*
Skalar *(Forts.)*
NODENUMBER (siehe DBPARTITIONNUM) 446
Tabellenfunktion
DB_PARTITIONS 457

G

Gespeicherte Abfragetabelle (MQT, Materialized Query Table)
repliziert 33
GET STMM TUNING DBPARTITIONNUM, Befehl
mit ADMIN_CMD 455
Globale Momentaufnahmen auf partitionierten Datenbanksystemen 280
Grenzwertbereiche
angeben 196
Große Objekte (LOBs)
Funktionsweise 194
mit partitionierten Tabellen 194

H

Hardware
Parallelität 90
Partitionen 90
Prozessoren 90
Hashpartitionierung 4
Hervorhebungskonventionen xiv
Hilfe
Konfiguration der Sprache 478
SQL-Anweisungen 477
HP-UX
Installation
DB2-Server 106
Network File System (NFS)
prüfen, ob NFS aktiv ist 118

I

IBMCATGROUP, Datenbankpartitionsgruppe 163
IBMDEFAULTGROUP, Datenbankpartitionsgruppe 163
IBMTEMPGROUP, Datenbankpartitionsgruppe 163
Indizes
Block
Suchsperrmodus 337
Clustering 327
Clusterverhältnis 347
für Tabellenspalten in einer Umgebung mit partitionierten Datenbanken 383
mit partitionierten Tabellen 323
Verhalten bei partitionierten Tabellen 323
verwalten
für MDC-Tabellen 348
Installation
AIX-Umgebungseinstellungen aktualisieren 115
Datenbankpartitionsserver
Antwortdateien 132
Datenbankpartitionsserver mit Antwortdateien 131
DB2 Enterprise Server Edition (Windows) 103
DB2-Produkte als Benutzer ohne Rootberechtigung 463
Methoden 107
Instanzen
Datenbankpartitionsserver auflisten 173
Partitionsserver
ändern 175
löschen 177
Instanzen *(Forts.)*
Partitionsserver hinzufügen 173
intra_parallel, Konfigurationsparameter des Datenbankmanagers 411

J

Joins
rundgesendete äußere Tabelle 363
rundgesendete innere Tabelle 363
Tabellenwarteschlangenstrategie in partitionierten Datenbanken 362
Typen
gezielt übertragene äußere Tabelle 363
gezielt übertragene innere Tabelle 363
Übersicht 360
Umgebungen mit partitionierten Datenbanken 363
zusammengefasste Tabellen 363

K

Kapazität
Erweiterung 156
für jede Umgebung 90
Katalogknoten 3, 139
Katalogsichten
BUFFERPOOLDBPARTITIONS 469
DATAPARTITIONEXPRESSION 469
DATAPARTITIONS 469
DBPARTITIONGROUPDEF 471
DBPARTITIONGROUPS 471
PARTITIONMAPS 472
Katalogstatistiken
Indexclusterverhältnis 347
Katalogtabellen
im Datenbankkatalogknoten gespeichert 3, 139
Knoten 89
Antwortzeit für Verbindung 406
FCM-Dämon (UNIX) 113, 157
maximale Zeitdifferenz zwischen 409
Synchronisation 302
Knoten hinzufügen, API 415
Knotengruppen (Datenbankpartitionsgruppen)
erstellen 163
Knotenkonfigurationsdateien
Aktualisierung (UNIX) 150
Beschreibung 142
erstellen 140
Knotenübergreifende Datenbankrecovery, Beispiel 291
Kollokation (Zusammenfassen von Tabellen)
Tabelle 4, 11
Kommunikation
Adressen 113, 157
Antwortzeit für Verbindung 406
Fast Communication Manager 113, 157
Kompatibilität
Partition 12
Konfiguration
mit mehreren Partitionen 90
Konfiguration des Datenbankpartitionsservers ändern, Befehl 436
Konfigurationsparameter
conn_elapse 406
fcm_num_buffers 105, 157, 406
fcm_num_channels 408
intra_parallel 411

Konfigurationsparameter *(Forts.)*
logarchopt1
knotenübergreifende Recovery, Beispiel 291
max_connretries 409
max_querydegree 412
max_time_diff 409
partitionierte Datenbank 3, 139
start_stop_time 410
vendoropt
knotenübergreifende Recovery, Beispiel 291
Konsistenzzustand
Datenbank 285
Koordinatorpartition 89

L

Laden
Beispiele
Sitzungen in Umgebungen mit partitionierten Datenbanken 266
Umgebungen mit partitionierten Datenbanken 266
Datenbankpartitionen 249
in Datenbankpartitionen 247
in partitionierte Tabellen 29, 242
Konfigurationsoptionen 261
MDC-Tabellen (MDC = mehrdimensionales Clustering) 241
Umgebungen mit partitionierten Datenbanken 261
Leistung
Kataloginformationen 3, 139
Lernprogramme
Fehlerbehebung 481
Fehlerbestimmung 481
Visual Explain 481
Linux
erstellen
Dateisystem für partitionierte Datenbanksysteme 120
Erstellung erforderlicher Benutzer 124
Installation
DB2-Installationsassistent 110
DB2-Server 106
prüfen, ob NFS aktiv ist 118
Standardportbereiche 159
Lizenzen
Übersicht 89
LOAD, Befehl
in einer Umgebung mit partitionierten Datenbanken 269
LOAD (Befehl)
in einer Umgebung mit partitionierten Datenbanken 251
LOAD, Dienstprogramm
Parallelität 241
LOAD QUERY, Befehl
in einer Umgebung mit partitionierten Datenbanken 257
logarchopt1, Konfigurationsparameter
knotenübergreifende Recovery, Beispiel 291
Logische Datenbankpartitionen 90
Logische Knoten
Datenbankpartitionsserver 150, 152
Löschen
Datenbankpartitionen über Launchpad 178
Löschen von Datenbank auf Datenbankpartitionsserver, API 418
Löschen von Datenbankpartitionsserver aus Instanz, Befehl 439

M

Maschinenliste
für Umgebung mit partitionierten Datenbanken 149
Materialized Query Tables (MQTs, gespeicherte Abfragetabellen)
repliziert, in partitionierten Datenbanken 369
max_connretries, Konfigurationsparameter 409
max_querydegree, Konfigurationsparameter 412
max_time_diff, Konfigurationsparameter 409
Maximale Zeitdifferenz zwischen Knoten, Konfigurationsparameter 409
Maximaler Grad der Parallelität bei Abfragen, Konfigurationsparameter 412
MDC 67
MDC-Tabellen 67
aktualisieren 78
Aspekte von Blockindizes 64
automatisches Beibehalten des Clustering 74
Blockindizes 64
Blockindizes und Abfrageleistung 70
Blockzuordnungen 76
Datensätze löschen 78
Hinweise zum Ladevorgang 62
Protokollierungsaspekte 63
MDC-Tabellen (MDC = mehrdimensionales Clustering) 56, 211
Aspekte des Ladevorgangs 241
Beschreibung 43
Daten versetzen in 56, 211
Dichte von Werten 45
Dimensionen auswählen 45
erstellen 56, 211
in SMS-Tabellenbereichen 56, 211
mit partitionierten Tabellen 33, 79, 309
Optimierungsstrategien 319
Rolloutlöschungen 319
Spaltenausdrücke als Dimensionen verwenden 56, 211
Sperrmodi
Tabellen- und RID-Indexsuchen 333
Verwaltung von Tabellen und Indizes 348
Mehrere logische Knoten
konfigurieren 152
Mehrpartitionsdatenbanken
Datenbankpartitionsgruppe 6
Migration von Einzelpartitionsdatenbanken durchführen
Designadvisor 331
Mehrpartitionskonfigurationen 90
Migration
Umgebung mit partitionierten Datenbanken 271
Momentaufnahmeüberwachung
auf partitionierten Datenbanksystemen 280
Ausgabe für Datenpartitionen interpretieren 273
für Datenpartitionen 273
Monotonie 56, 211
MPP-Umgebung 90
MQT (Materialized Query Table)
Funktionsweise 203
mit partitionierten Tabellen 203
MQT (Materialized Query Table, gespeicherte Abfragetabelle)
repliziert 33
MQTs (Materialized Query Tables)
repliziert, in partitionierten Datenbanken 369

N

Nachrichtenpuffer
  Fast Communication Manager (FCM) 105, 157
Neustart einer Ladeoperation
  Ladeoperationen bei Mehrpartitionsdatenbanken 258
Neuverteilen von Daten 392
NFS (Network File System)
  Betrieb prüfen 118
Nicht als Root ausgeführte Installationen
  Installation 463
NODENUMBER, Funktion
  DBPARTITIONNUM 446

O

Optimierung
  Joins
    Definition 360
    partitionierte Datenbank 363
  partitionierte Tabellen 314
  Partitionsinterne Parallelität 350
  Strategien für MDC-Tabellen 319
Optimierung, Partition
  feststellen 401
Organisieren von Daten
  Konzepte 16

P

Parallelität
  Abfrage 85
  Backups 85
  Dienstprogramm LOAD 241
  Ein-/Ausgabe
    Übersicht 85
  Hardwareumgebungen 90
  Indexerstellung 85
  Konfigurationsparameter
    intra_parallel 411
    max_querydegree 412
  LOAD, Dienstprogramm 85
  Partitionen 90
  partitionsintern
    aktivieren 155
    Optimierungsstrategien 350
    Übersicht 85
  partitionsübergreifend 85
  Prozessoren 90
  Übersicht 85
  Umgebungen mit partitionierten Datenbanken 89
Partitionen
  Prozessoren
    einzeln 90
    mehrere 90
Partitionen löschen, Launchpad 178
Partitionierte Datenbanken, Umgebungen
  Daten umverteilen 390, 394
  doppelte Maschineneinträge 150
  einrichten 3, 139
  Fehler beim Hinzufügen von Knoten 171
  Joins
    Methoden 363
    Strategien 362
  Maschinenliste
    angeben 149
    Eliminierung doppelter Einträge 150
Partitionierte Datenbanken, Umgebungen *(Forts.)*
  Partitionen löschen 172
  Partitionskompatibilität 12
  replizierte MQTs 369
  Speicher mit automatischer Leistungsoptimierung 401
  Szenario 179
  Übersicht 4
Partitionierte Datenbanken, Umgebungen mit
  Daten laden
    Migration 269
    Übersicht 249
    Überwachung 257
    Versionskompatibilität 269
  globale Momentaufnahmen 280
  Migration 269
  Neuerstellung von Datenbanken 290
  Versionskompatibilität 269
Partitionierte Tabellen
  Abweichungen 223
  ändern 219, 220
  aufgehobene Zuordnung, Datenpartitionen
    Attribute 228
  Datenbereiche 196
  Datenpartitionen hinzufügen 219, 230
  Einschränkungen 13, 220
  erstellen 196
  Große Objekte (LOBs) 194
  Indizes 323
  konvertieren 223
  laden 200
  Laden 29, 242
  MDC-Tabellen (MDC = mehrdimensionales Clustering) 33, 79, 309
  migrieren
    aus Version vor 9.1 223
    Sichten 200
    Tabellen 200
  MQT (Materialized Query Table) 203
  Optimierung 314
  Partitionen zuordnen 219, 221
  Reorganisation 273
  Rollin von Datenpartitionen 219, 221
  Rollout von Datenpartitionen 219
  Sperren 340
  Szenarios
    Daten rotieren 233
    Datenpartitionen zuordnen und Zuordnung aufheben 235
    Rollin und Rollout von Datenpartitionen 235
  Übersicht 13
  Zuordnung von Datenpartitionen aufheben 219, 227, 232
Partitionierungsschlüssel
  Übersicht 27
Partitionierungszuordnungen
  für Datenbankpartitionsgruppen erstellen 452
Partitionsinterne Parallelität
  aktivieren 155
  db2expln, Tool
    Ausgabebeispiel 354
    Ausgabebeispiele 358
  Optimierungsstrategien 350
  zusammen mit partitionsübergreifender Parallelität 85
Partitionsübergreifende Abfrageparallelität
  aktivieren 153
Partitionsübergreifende Parallelität
  db2expln, Tool
    Ausgabebeispiel 356

Partitionsübergreifende Parallelität *(Forts.)*
db2expln, Tool *(Forts.)*
Ausgabebeispiele 358
zusammen mit partitionsinterner Parallelität 85
Portnummernbereiche
Linux
Standard 159
Verfügbarkeit 119, 159
UNIX
Standard 159
Verfügbarkeit 119, 159
Windows
definieren 173
Präfixsequenzen 185
Protokoll mit Benachrichtigungen für die Systemverwaltung 285
Protokolldateispeicher
Anforderungen für Datenumverteilung 393
Prozeduren
STEPWISE_REDISTRIBUTE_DBPG 395, 458
Prozessoren
hinzufügen 156
Prüfen
Portbereichsverfügbarkeit
Linux 119
UNIX 119

R

rah, Befehl
angeben
als Parameter oder Antwort 182
Liste von Datenbankpartitionsservern 149
Befehle parallel ausführen 183
Beschreibung 184
Einführung 181, 305
Fehlerbestimmung 190
Prozesse überwachen 192
RAHCHECKBUF, Umgebungsvariable 183
RAHDOTFILES, Umgebungsvariable 189
RAHOSTFILE, Umgebungsvariable 149
RAHOSTLIST, Umgebungsvariable 149
RAHWAITTIME, Umgebungsvariable 192
rekursiv aufgerufen 184
Standardumgebungsprofil definieren 192
steuern 188
Übersicht 182
Umgebungsvariablen 188
RAHCHECKBUF, Umgebungsvariable 183
RAHDOTFILES, Umgebungsvariable 189
RAHOSTFILE, Umgebungsvariable 149
RAHOSTLIST, Umgebungsvariable 149
RAHTREETHRESH, Umgebungsvariable 184
RAHWAITTIME, Umgebungsvariable 192
Recovery
knotenübergreifend, Beispiel 291
nach dem Ausfall eines Datenbankpartitionsservers 290
Systemabsturz 285
Recovery durchführen
Protokoll des zweiphasigen Commits 286
Recovery nach Systemabsturz 285
REDISTRIBUTE, Dienstprogramm
Einschränkungen 387
REDISTRIBUTE DATABASE PARTITION GROUP, Befehl 425
Registrierdatenbankvariablen
DB2_FCM_SETTINGS 404
DB2_NO_MPFA_FOR _NEW_DB 56, 211
Registrierdatenbankvariablen *(Forts.)*
DB2_NUM_FAILOVER_NODES 404
DB2_PARTITIONEDLOAD__DEFAULT 404
DB2CHGPWD_ESE 404
DB2PORTRANGE 404
Replizierte MQTs 33
RESTART DATABASE, Befehl 285
Richtlinien
Bereichsclustertabellen 39, 208

S

Schlüssel
Tabellenpartitionierung 27
Verteilung 10
Selbstoptimierender Speicher
Umgebungen mit partitionierten Datenbanken 399
SIGTTIN, Nachricht 182
Skalierbarkeit
Umgebungen 90
SMP-Clusterumgebung 90
SMS (vom System verwalteter Bereich)
Tabellenbereiche
Container hinzufügen 177
Solaris-Betriebssystem
Installation von DB2-Servern 106
prüfen, ob NFS aktiv ist 118
Sonderregister
CURRENT DBPARTITIONNUM 443
CURRENT NODE (siehe CURRENT DBPARTITIONNUM) 443
Spaltenausdrücke
mehrdimensionale Tabellen 56, 211
Speicher mit automatischer Leistungsoptimierung
aktivieren
nicht einheitliche Umgebungen 401
Umgebungen mit partitionierten Datenbanken 401
Speicheroptimierung
Umgebungen mit partitionierten Datenbanken 401
Sperren
Blockindexsuche, Modi 337
diskret
Bereichsclustertabellen 41
Übersicht 39
Verhalten bei partitionierten Tabellen 340
Sperrmodi
MDC-Tabellen (MDC = mehrdimensionales Clustering)
Tabellen- und RID-Indexsuchen 333
SQL-Anweisungen
ALTER DATABASE PARTITION GROUP 449
ALTER NODEGROUP (siehe ALTER DATABASE PARTITION GROUP) 449
CREATE DATABASE PARTITION GROUP 452
CREATE NODEGROUP (siehe CREATE DATABASE PARTITION GROUP) 452
Hilfe anzeigen 477
sqleaddn, API 415
sqlecran, API 417
sqledpan, API 418
sqledrpn, API 419
sqlugrpn, API 421
start_stop_time, Konfigurationsparameter 410
stdin 182
STEPWISE_REDISTRIBUTE_DBPG, Prozedur 458
zur Datenumverteilung verwenden 395
Steuern des Befehls rah 188

Steuerzentrale
Verwalten von Datenbankpartitionen 217
Synchronisation
Aspekte der Recovery 302
Datenbankpartition 302
Knoten 302
Szenarios
Bereichsclustertabellen 209
MDC-Tabellen 67

T

Tabellen
Bereichscluster 39, 208
erstellen
in partitionierten Datenbanken 193
in partitionierte Tabellen migrieren 200
Kollokation (Zusammenfassen von Tabellen) 4, 11
konvertieren 200
MDC-Tabellen (MDC = mehrdimensionales Clustering) 33, 79, 309
mehrdimensionales Clustering (MDC) 43, 348
MQTs (Materialized Query Tables) 203
partitioniert 13
partitionierte Tabellen 13, 33, 79, 203, 309
partitionierte Tabellen ändern 230, 232
Warteschlangen, für Joinstrategien in partitionierten Datenbanken 362
Tabellenbereiche
erstellen
in Datenbankpartitionsgruppen 33, 164
Tabellenpartitionierung
Beschreibung 13
Vorteile 13
Teilentclusterung
Übersicht 89
Transaktion nach Fehler 285
Transaktionen
Recovery nach Fehler
auf dem aktiven Datenbankpartitionsserver 286
auf dem ausgefallenen Datenbankpartitionsserver 286
Auswirkungen von Fehlern begrenzen 285

U

Überwachen
rah-Prozesse 192
Überwachung
Datenpartitionen 273
Umgebungen mit partitionierten Datenbanken
Daten laden
Einschränkungen 251
Übersicht 247
einrichten 129
Ereignisüberwachung 281
erstellen 3, 139
Installationsprüfung
UNIX 136
Windows 135
Partitionskompatibilität 441
Speicher mit automatischer Leistungsoptimierung 399
Transaktionen
Recovery nach Fehler 286
Übersicht 89
Umgebungsvariablen
$RAHBUFDIR 183
$RAHBUFNAME 183
$RAHENV 188
rah, Befehl 188
RAHDOTFILES 189
Umverteilen von Daten
Notwendigkeit 389
Prozeduren 458
Prozeduren zum schrittweisen Umverteilen 395
über Datenbankpartitionen 217
Umverteilung 394
Union-Verknüpfung
alle Sichten, Konvertierung 200
UNIX
Aktualisierung der Knotenkonfigurationsdatei 150
Installation
mit dem DB2-Installationsassistenten 110
Prüfen einer partitionierten Datenbankserverinstallation 136
Standardportbereiche 159
UPDATE STMM TUNING DBPARTITIONNUM, Befehl
mit ADMIN_CMD 456

V

vendoropt, Konfigurationsparameter
knotenübergreifende Recovery, Beispiel 291
Verbindungen
Antwortzeit 406
Verbindungskonzentratoren
Verwendung von Agenten in partitionierter Datenbank 345
Vergleich von Indizes
cluster- und blockbasiert 43
Vergleich von Tabellen
regulär und mit mehrdimensionalem Clustering 43
Versetzen von Daten
in mehrdimensionale Tabellen 56, 211
Verteilungsschlüssel
Beschreibung 10
Daten laden 247
Umgebungen mit partitionierten Datenbanken 193
Verteilungszuordnungen
Beschreibung 8
Visual Explain
Lernprogramm 481

W

Weltzeit 409
Wiederholungen für Knotenverbindung, Konfigurationsparameter 409
Windows-Betriebssysteme
Datenbankpartitionen
hinzufügen 168
Installation
DB2-Server (mit DB2-Installationsassistent) 101
Installationsprüfung
Umgebung mit partitionierten Datenbanken 135

Z

Zeilenverteilungsnummer abrufen, API 421
Zeit
Unterschied zwischen Knoten (maximal) 409

Zeitlimit für DB2START und DB2STOP, Konfigurationsparameter 410
Zielgruppe ix
Zugriffspläne
  Abfrage ohne Indizes und ohne Statistikdaten
    in einer Umgebung mit partitionierten Datenbanken 373
  aktuelle Statistikdaten erfassen
    in einer Umgebung mit partitionierten Datenbanken 375
  Ausgabebeispiel: keine Parallelität 352
  Indizes für Spalten zum Verknüpfen von Tabellen erstellen
    in einer Umgebung mit partitionierten Datenbanken 378
  verbessern
    in einer Umgebung mit partitionierten Datenbanken 372
  zusätzliche Indizes erstellen
    in einer Umgebung mit partitionierten Datenbanken 383
Zuordnen von Datenpartitionen
  Beschreibung 221
Zweiphasiges Commit
  Protokoll 286

SC12-3915-03

Spine information:

DB2 Version 9.5 für Linux, UNIX und Windows **Version 9 Release 5** **Partitionierung und Clustering** IBM