

Große Datenmengen als Herausforderung

Neue Wege, um damit umzugehen



Das Problem des Datenwachstums ist so alt wie die IT. Es gibt dafür kein All-Heilmittel, sondern nur Medikamente, die die Schmerzen lindern. Der Erfolg der Datenbanken als universeller Speicher für unterschiedliche Formate und Dateninhalte geht auch an den Trends Standardisierung, Re-Zentralisierung und Virtualisierung nicht spurlos vorbei.

> Trotz allen technischen Fortschritts auf dem Gebiet werden sehr große Datenmengen immer mehr zur Herausforderung für moderne Datenbankmanagementsysteme. Dabei möchten die Unternehmen nicht nur ihre Datenmengen effizienter verwalten, sondern vor allem schneller und flexibler auf ihre Daten zugreifen. Der Trend in der Informationsverarbeitung geht immer weiter in Richtung der Zunahme von Volumen und Komplexität. Dies gilt nicht nur für die Daten, sondern auch für die Hardware und die Software.

Die Fortschritte in der Informationsverarbeitung können mit den Fortschritten in der Hardwareentwicklung und speziell im Speicherbereich nicht mithalten, sodass man mehr speichern als effizient auswerten kann. Das Datenwachstum übersteigt jedoch auch die Fortschritte in der Speichertechnologie. Deswegen sind auch organisatorische Maßnahmen wie beim ILM (Information Lifecycle Management), der Datenkonsolidierung und dem Datenbereinigung nötig. Das Problem großer Datenmengen

ist jedenfalls nicht allein mit der Anschaffung von neuerer Hardware erledigt. Die Folgeprobleme bei der Handhabung können dabei noch größer werden als das Ursprungsproblem. Beispielsweise werden bei größeren Platten, weniger Platten benötigt. Der E/A-Durchsatz und die Zugriffszeit kann jedoch sinken, da der Zugriff weniger parallelisiert werden kann.

Um das Problem frühzeitig anzugehen, sollte viel Wert auf das, oft vernachlässi-

gte, physische Datendesign gelegt werden. Dieses setzt jedoch neben dem Wissen über den fachlichen Inhalt der Daten tiefes Technologie- und Datenbankwissen und eine proaktive Planung (unterbrechungsarmer Betrieb, performanceorientiertes Design) voraus. Hier wollen wir am Beispiel von DB2 UDB for LUW (Linux, Unix, Windows) Möglichkeiten vorstellen, wie das Datenwachstum aus Datenbanksicht beherrschbar wird. Viele der hier aufgezeigten Konzepte sind allgemeingültig und lassen sich auch mit anderen modernen Datenbankmanagementsystemen realisieren.

Datenorganisation – grundlegende Konzepte

Eine DB2-Server-Instanz kann eine oder mehrere Datenbanken betreiben. In der Enterprise Server Edition (ESE) von DB2 können die Daten einer (logischen) Datenbank über mehrere DB2-Server oder Knoten verteilt werden. Physikalisch unterteilt sich die Datenbank in Tablespace und Container, die auf als Dateien, Directories oder Raw Devices auf das jeweilige Betriebssystem abgebildet werden. Logisch unterteilt sich eine Datenbank in Tabellen und Indizes. Dazu kommen noch prozedurale Objekte wie Trigger, Funktionen oder Prozeduren (Bild 1).

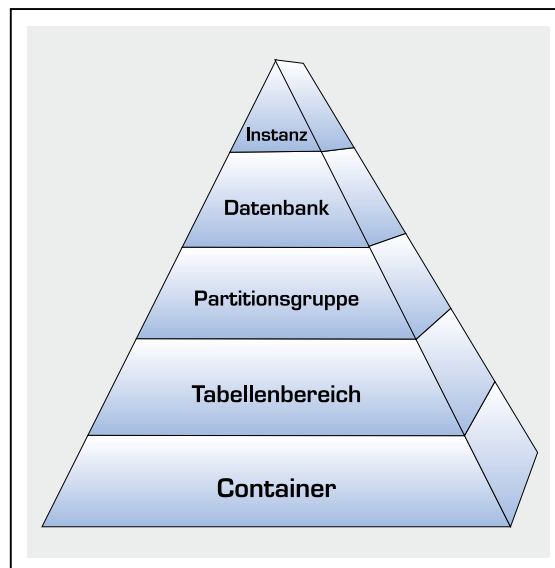


Bild 1: Beispiel SQL für Rangepartitionierung.

Zusammen mit dem logischen Aufbau der Tabelle, der durch die Datentypen festgelegt ist, beeinflusst die physikalische Bild den Platzbedarf einer Datenbank erheblich. Bei den heute sehr günstigen Anschaffungspreisen für Datenspeicher wird ein explizites physisches Design der Datenbank oft vernachlässigt. Es zahlt sich jedoch – nicht nur bei großen Datenmengen – auf lange Sicht aus. Die Verwendung von 64-Bit-Prozessoren macht gerade bei großen Datenmengen Sinn, da damit sowohl der maximal adressierbare Hauptspeicherbereich, als auch der Dateispeicher vergrößert wird. Der DB2-Kern ist auch für 64-Bit entwickelt und wird auch bei den 32-Bit-Versionen eingesetzt. So unterstützt DB2 unter 64-Bit-AIX 5L die Seitengröße von 16 MB, statt wie sonst nur 4, 8, 16 und 32 KB. Diese Größe ist jedoch eher für die Batch- als für eine Online-TP-Verarbeitung geeignet. Abhängig von der Seitengröße sind bei DB2 die Anzahl der maximalen Spalten (1012), die maximale Zeilengröße (32.677 Byte), die Anzahl der Sätze pro Seite (bis DB2 8 maximal 255) und die Größe der Tabellenbereiche (2 TB, 4 TB, 8 TB, 16 TB).

Speicherung und Einheiten

Bei der Virtualisierung und der logischen Aufteilung des Datenspeichers waren die Datenbanken den Betriebssystemen oft voraus. Mit der automatischen (dynamischen) Speicherverwaltung bietet DB2 die Möglichkeit der automatischen Anpassung der Tabellenbereiche an den benötigten Bedarf. Die automatische Speicherverwaltung unterstützt mit Version 9.1 auch partitionierte Datenbanken.

DB2 speichert Daten in Tabellenbereichen (TS). Die Einheiten der Datenspeicherung in DB2 sind in Bild 3 dargestellt. Das System erlaubt derzeit Tabellenbereiche mit einer maximalen Größe von 128 Terabyte. Die Anzahl der Tabellenbereiche wird durch das Betriebssystem limitiert.

Ein Tabellenbereich kann vom Betriebssystem (SMS) oder von der Datenbank (DMS) verwaltet werden. Der neue Standardtyp für DMS-Tabellenbereiche ist „LARGE“. Der Befehl dafür lautet:

```
CREATE TABLESPACE <tblspace-name>
MANAGED BY [DMS | AUTOMATIC|STORAGE]
```

Die Konvertierung alter Tabellenbereiche in den neuen größeren kann Online mit dem Befehl

```
ALTER TABLESPACE <name> CONVERT TO
LARGE
```

geschehen.

Automatic Storage Management

Um eine effiziente Nutzung des Datenspeichers ohne aufwendige Administration zu erzielen, ist die Verwendung der Automatic Storage Option von DB2 9.1 sinnvoll. Beim Anlegen des Tabellenbereiches kann man über die Parameter AUTORESIZE, INCREASESIZE und MAXSIZE sowohl absolute als auch prozentuale Werte angeben.

Die automatische Speicherverwaltung, die in Version 8.2.2 eingeführt wurde, wurde in Version 9 ausgebaut. Für neue Datenbanken unter V9 wird sie standardmäßig eingeschaltet. Die Tabellenbereiche werden als DMS-Tabellenbereiche angelegt. Die manuelle Überwachung der Container-Belegung und das Hinzufügen neuer Container oder Tabellenbereiche entfallen.

Es ist nun möglich, beim Definieren einer Datenbank mehrere Speicherpfade für die Tabellenbereiche der Datenbank anzugeben. Weitere Speicherpfade können auch noch nachträglich definiert werden.

Bei neuen Tabellenbereichen werden keine Container mehr spezifiziert, sondern nur MANAGED BY AUTOMATIC STORAGE angegeben (wird derzeit im Control-Center GUI noch nicht unterstützt). Für Wachstum und Grenzen können explizite Angaben (Policies) gemacht werden:

```
CREATE TABELLENBEREICH USER_2
INITIALSIZE 1000K IN-
CREASESIZE 300K MAXSIZE 20M
```

Neue DMS-Tabellenbereiche werden standardmäßig als large angelegt, das heißt, sie arbeiten mit 6 Bytes langen RIDs (row identifier), was mehr Seiten je Tabellenbereich und mehr Zeilen je Seite ermöglicht als mit den alten 4 Bytes langen RIDs. Damit wird die maximale Tabellenbereich-Größe auf 2 bis 16 TByte je nach Seitengrößen (4 KByte – 32 KByte) gesteigert.

Verteilungsprinzipien im Überblick – Teile und Herrsche

Bei der Verteilung von Daten stehen immer die Ziele verbesserter Zugriffsleistung und höherer Verfügbarkeit im Vordergrund. Für die Verteilung von Daten sind zwei Arten denkbar: die logische und die physische Partitionierung. Deren Kombinationsmöglichkeiten bzgl. logischer und physischer Verteilung sind in Bild 3 dargestellt. Zunächst gehen wir auf die Datenbankpartitionierung (DPF) – die Tabellen- oder Bereichspartitionierung (TP) ein. Abschließend werden wir deren Kombination mit der multidimensionalen Partitionierung

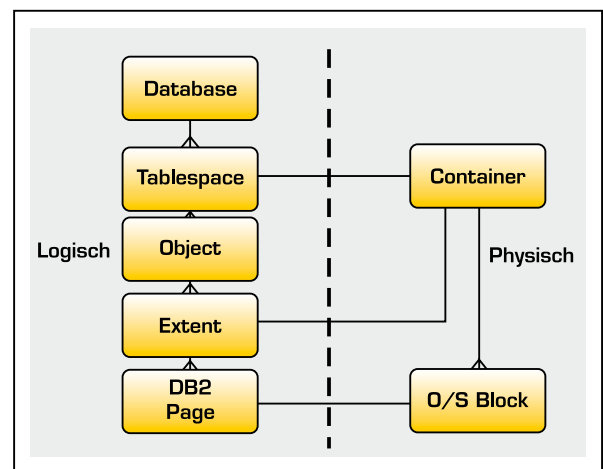


Bild 2: DB2 Speichereinheiten/-Ebenen.

(MDC) vorstellen, wie sie in OLAP-Anwendungen eingesetzt wird.

Partitionieren und Parallelisieren

• Datenbankpartitionierung (DPF)

Vor Version 9 konnte DB2 für LUW nur die Partitionierung auf Datenbank-Ebene über mehrere Rechnerknoten hinweg. Jeder Knoten besitzt dabei eine DB2-Instanz und einen Teil der Datenbank auf den nur er zugreifen kann SMP (Shared-nothing-Prinzip) oder MPP (Massenparallelrechner). Im Gegensatz dazu wird bei z/OS Parallel Sysplex der Ansatz der gemeinsamen Platte (data sharing) genutzt, wodurch sich der dann dergemeinsame Speicherplatz effizienter nutzen lässt.

Mit DPF wird eine Tabelle über mehrere Knoten (Datenbank-Server) entsprechend dem spezifizierten Schlüssel verteilt. Bei einer Abfrage auf diese Tabelle wird diese so auf jede Partition

aufgeteilt, dass jeder Knoten die ihm zugewiesenen Zeilen sucht. Somit ist DPF im Prinzip eine Skalierbarkeitsfunktion. Bei wachsendem Datenvolumen kann die Abfrageleistung durch Hinzufügen zusätzlicher Knoten Schritt halten. Wichtig für die optimale Nutzung von DPF ist die richtige Auswahl der Verteilungsschlüssel. Er sollte eine gleichmäßige Verteilung auf die Knoten gewährleisten. Der Schlüssel sollte eine hohe Kardinalität seiner Werte aufweisen.

• **Multidimensional Clustering (MDC)**

MDC wurde in der Version 8 von DB2 für LUW eingeführt. Mit Hilfe von Block-Indizes werden Zeilen einer Tabelle mit gleichen Datenwerten über mehrere Dimensionen nah beieinander gespeichert. Die Pages werden in Blöcken gleicher Größe (= Extentsize des TS) organisiert. Alle Zeilen eines Blocks beinhalten dieselbe Kombination von Dimensionswerten.

Auch beim Einsatz von MDC ist die richtige Wahl der Spalten für die Dimensionen ein entscheidender Faktor. Es gilt, die richtige Balance zwischen minimalen Speicheranforderungen und optimaler Gruppierung zu finden. Dazu müssen die wesentlichen Abfragen mit ihren Prädikaten beim Tabellenentwurf bekannt sein. Durch die Bereitstellung von Blöcken für dieselbe Kombination von Dimensionswerten bleibt bei Neuzugängen die Cluster-Reihenfolge auch ohne Reorganisation erhalten. Bockindizes benötigen außerdem weniger RIDs, sodass die Indizes kompakter sind.

• **Tabellenpartitionierung (TP)**

Neu ist in DB2 V9 Enterprise Server Edition die Partitionierung von Tabellen an einem Rechnerknoten. Diese Möglichkeit ist schon seit Langem in DB2 für z/OS, Oracle und Informix bekannt. Die Funktionalität in DB2 für LUW ist aber nicht identisch mit den Möglichkeiten in DB2 für z/OS (Version 8) und betrifft nur Tabellen, keine Indizes. Sie kann mit Symmetric Parallel Processing- (SMP) oder Massive Parallel Processing (MPP)-Systemen eingesetzt werden.

Die Tabellenpartitionierung ist ein Datenorganisationsschema, in dem die Tabellendaten auf mehrere Speicherobjekte – genannt Datenpartitionen oder Bereiche – entsprechend den Werten

einer oder mehrerer Spalten der Tabelle verteilt werden. Dieser auch den Römern schon bekannte Algorithmus wird auch als „Teilen und Herrschen“ („divide et impera“) bezeichnet. Jede Datenpartition wird separat gespeichert. Diese Partitionen können sich in verschiedenen Tabellenbereichen, im selben Tabellenbereich oder in einer Kombination aus beiden befinden. Die Möglichkeit, Tabellendaten über mehrere Speicherobjekte zu partitionieren, führt zu einer besseren Skalierbarkeit sowie zu mehr Flexibilität, Kontrolle und Leistung (Query Parallelisierung). Außerdem können Tabellen dadurch erheblich vergrößert werden.

Im Zusammenhang mit Tabellenpartitionierung gibt es auch für Indizes Fortschritte: Bei nicht partitionierten Tabellen werden alle Indizes in demselben Speicherobjekt

Folgende SQL-Anweisung legt eine Tabelle an, wobei jede Partition ein Quartal enthält:

```
CREATE TABLE VAUFTRAG (DATUM DATE,
KUNDE VARCHAR(80), ...)
PARTITION BY RANGE(DATUM)
(PARTITION Q405 STARTING
MINVALUE ENDING '12/31/2005',
STARTING
'1/1/2006' ENDING '3/31/2006',
STARTING
'4/1/2006' ENDING '6/30/2006',
STARTING
'7/1/2006' ENDING '9/30/2006',
STARTING
'10/1/2006' ENDING '12/31/2006');
```

Die Tabellenpartitionen können wie eigenständige Tabellen behandelt werden und in unterschiedlichen Tabellenbereichen oder einem angelegt werden. So können,

Platzierung:	DPF	MDC	TP
Art der Datenorganisation	Gleichmäßige Verteilung über Datenbank-Partitionen (eigenständige Knoten)	Zusammenfassung von Zeilen gleicher Datenwerte in mehreren Dimensionen in gemeinsame Blöcke	Verteilung von Zeilen nach spezifizierten Wertebereichen (einer Dimension) auf TS-Partitionen (eigenständige DB2-Objekte)
Vorteile	> Abfrage-Parallelisierung > Skalierbarkeit	> Performance typische DWH-Abfragen > Clustering bei Neuzugängen	> Abfrage-Parallelisierung > Unterstützung von Bewegungsdaten > Index-Platzierung
Typische Schlüsselspalten	Spalten mit hoher Wertekardinalität	abfrageabhängig, in DWHs häufig Zeitangaben, Gebietsgliederungen, Produktgruppen	Aufteilungskriterien, etwa Zeitangaben bei Bewegungsdaten, Mandaten-IDs bei Stammdaten

Tabelle 1: Gegenüberstellung der Partitionierungsmöglichkeiten von DB2 UDB V 9.

angelegt, das bei DMS (Data Managed Storage) in einem anderen Tabellenbereich liegen kann. Für partitionierte Tabellen wird jeder Index in einem eigenen Speicherobjekt angelegt. Jeder Index kann somit in einen anderen Tabellenbereich gelegt werden. Die Vorteile sind effizienterer konkurrierender Zugriff und verbesserte Operationen wie CREATE, DROP oder Reorganisation.

statt der gesamten Tabelle, Partitionen einzeln gesichert oder wiederhergestellt werden. Dadurch können zeitintensive Wartungsaufgaben flexibler in kleineren Operationen und damit Zeitfenstern ausgeführt werden.

Performance-Vorteile resultieren aus der Parallelverarbeitung von Partitionen in Abfragen oder dem Ausschluss von Partitionen beim Table-Scan.

Ein besonders interessanter Aspekt der Partitionierung ist die Möglichkeit, Partitionen rollieren zu lassen. Das ist besonders nützlich in einer Data Warehouse-Umgebung, in der häufig Daten geladen oder gelöscht werden. Werden die Daten von 2005 nicht mehr in der Tabelle VAUFTRAG benötigt, so können sie per DETACH abgehängt werden. Die Partition bildet dann eine eigenständige Tabelle, die anschließend archiviert werden kann.

```
alter table VAUFTRAG detach partition q405 into VAUFTRAG4q05
```

Die betroffenen Indizes werden im Hintergrund asynchron angepasst (AIC = Asynchronous Index Cleanup).

Analog ist es möglich, eine Tabelle mit kompatibler Struktur per ATTACH als neue Partition zu integrieren. Hierbei muss der Befehl SET INTEGRITY nach dem ATTACH ausgeführt werden, um die Operation zu vollenden. SET INTEGRITY prüft neben der Einhaltung des Partitionsbereiches auch sonstige Bedingungen und pflegt die Indizes.

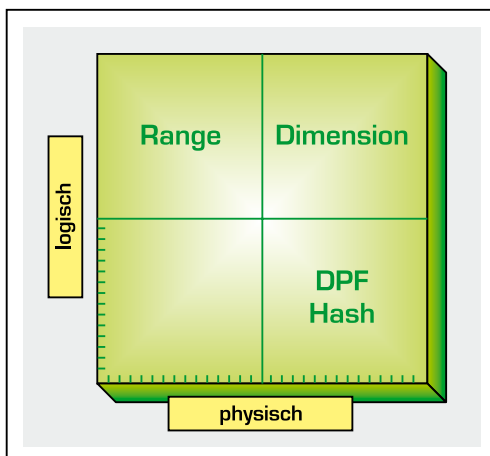


Bild 3: Verteilungsprinzipien.

Tabellenpartitionierung kann mit den bisher bekannten Möglichkeiten der Datenanordnung wie Datenbankpartitionierung (DPF- DISTRIBUTE BY HASH) oder Multi-dimensional Clustering (MDC) kombiniert werden. Dadurch können Datenbereiche gleichmäßig über Datenpartitionen verteilt werden, um DPF-Features wie abfrageinterne Parallelität oder die Lastverteilung für Datenbankpartitionen zu nutzen, um eine optimale Verteilung der Last und eine gute Performance zu erreichen. Zusammen

mit dem mehrdimensionalen Clustering (MDC-ORGANIZE BY DIMENSIONS) ermöglicht die Tabellenpartitionierung Zeilen mit ähnlichen Werten für mehrere Dimensionen im selben Tabellenspeicherbereich zu gruppieren.

Bild 5 zeigt eine hybride Partitionierung. Der Befehl dafür sieht wie folgt aus:

```
CREATE TABLE VAUFTRAG(
  DATUM DATE,
  KUNDE VARCHAR(80),
  REGION CHAR(5),
  JAHR INTEGER GENERATED ALWAYS AS
  (YEAR(DATUM)*100))
  DISTRIBUTE BY HASH (JAHR)
  PARTITION BY (DATUM)
  (STARTING '01/01/2006'
  ENDING '12/31/2007' EVERY 1 MONTH)
  ORGANIZE BY DIMENSIONS (REGION);
```

Datenkompression zur Reduzierung des Platzbedarfs

Für DB2 for z/OS ist Datenkompression eine bewährte Funktion, die bei klassischen kaufmännisch-administrativen Anwendungen Kompressionsraten von 50-80% erzielt. Mit Version 9 wird die Datenkompression auch in DB2 for LUW verfügbar.

Die Bitmuster-Kompression benötigt ein Dictionary zur Umsetzung. Dieses Dictionary gilt für die gesamte Tabelle und wird im gespeicherten Objekt abgelegt.

Die Vorteile der Kompression liegen in der Einsparung von belegtem Speicher, in weniger E/A, in mehr gelesener Zeilen je physischem Zugriff und geringerer Bufferpool-Belegung bei verbesserter Hit-Ratio.

Der mit der Datenkompression verbundene Nachteil ist die zusätzliche CPU-Belastung für die Komprimierung und Dekomprimierung. Mit einer Reorganisation (REORG TABLE) wird die Kompression durchgeführt. Nur das Dienstprogramm REORG kann das Dictionary aufbauen. Um zu entscheiden wann sich die Kompression lohnt, kann man sich mit dem INSPECT-Kommando die Kompressionsrate und den reduzierten Platzbedarf anzeigen lassen. Generell lohnt sich die Kompression für große Tabellen, die keine LOB's enthalten. Bei Tabellen, die über das LOAD-Kommando befüllt werden, lohnt es sich, die Tabelle zunächst mit einer kleinen repräsentativen Stich-

probe zu füllen. Damit kann durch das REORG-Kommando das initiale Dictionary aufgebaut werden. Dieses kann dann bereits beim Laden verwendet werden, so dass eine spätere Reorganisation schneller passieren kann.

Reorganisation von Tabellen und Indizes

Um eine optimale Verteilung der Daten – besonders bei Cluster-Indizes – auch nach vielen Neuzugängen zu gewährleisten, ist die Reorganisation von DB2-Tabellen (Unix- und Windows-Version) oder Tabellenbereichen (Mainframe) mit dem REORG-Dienstprogramm nötig. Diese macht freien Plattenplatz wieder verfügbar und beschleunigt den späteren Zugriff. Hinweise auf eine Fragmentierung der Daten liefert das Statistik-Werkzeug RUNSTATS und das Zugriffsplan-Werkzeug EXPLAIN.

RUNSTATS sollte immer dann ausgeführt werden, wenn sich die Inhalte von Tabellen wesentlich geändert haben oder wenn neue Indizes angelegt wurden. Danach müssen bei statischem SQL auch die verweisenden DB2-Packages neu gebunden werden, da darin die Zugriffswege abgelegt sind. Das Dienstprogramm kann für Tabellenbereiche oder Indizes ausgeführt werden und kann auch im Rahmen anderer administrativer Utilities eingebettet laufen (REORG, LOAD). Die Statistiken können im Rahmen der Autonomic-Computing-Möglichkeiten im DB2 auch automatisch erneuert werden. Diese Autonomic-Computing-Funktionen erlauben auch die automatische Reorganisation von Tabellenbereichen und Indizes.

Ein Tablespace, eine Tabelle oder Index können offline, das heißt, ohne im laufenden Betrieb verfügbar zu sein, oder online, das heißt, im laufenden Betrieb, reorganisiert werden. Gerade bei produktiven Datenbeständen im 24*7 h-Betrieb ist eine Online-Reorganisation notwendig. Allerdings wird für einen Online-Reorganisation wesentlich mehr Platz benötigt, weil eine Schattenkopie des Objekts angelegt wird. Dies kann bei großen Datenmengen ein wesentlicher Nachteil sein. Neben der Verfügbarkeit im Betrieb ist ein Vorteil, dass ein fehlerbedingter Abbruch der Reorganisation deutlich schneller zu beheben ist, weil die Ausgangskopie verfügbar bleibt. Bei einer in-place-Reorganisation bedeutet ein Abbruch immer ein eher mühevolleres Wiederherstellen der Ausgangsbasis.

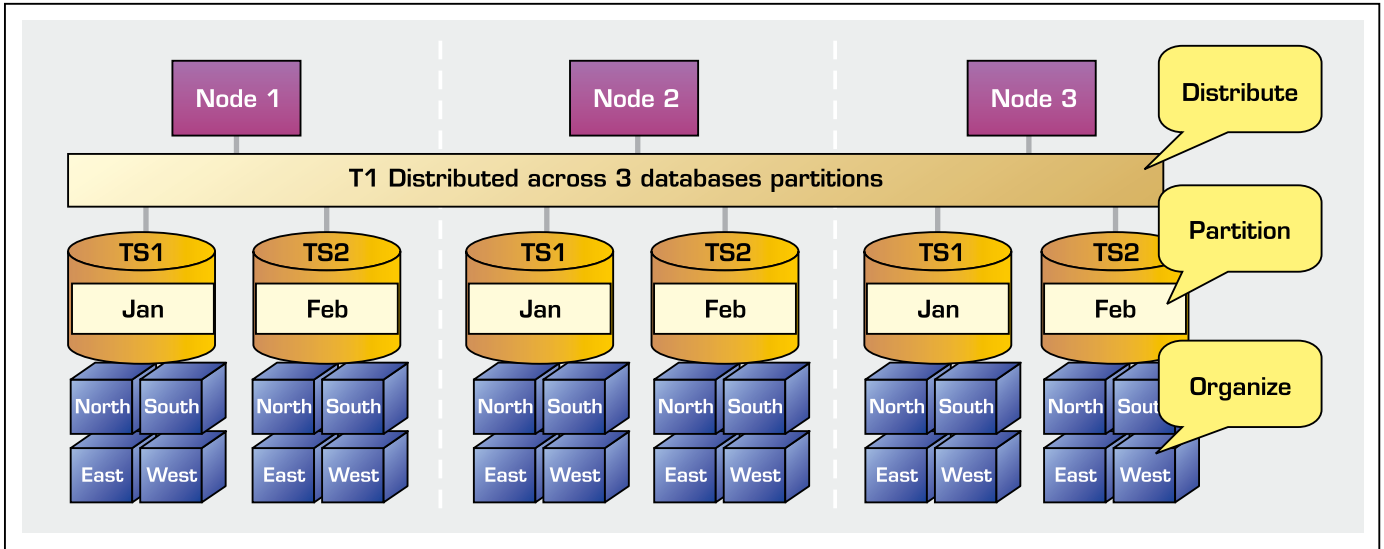


Bild 4: Hybride Partitionierung (Range, Dimension, Hash) (Quelle: IBM).

Laden von Daten

Das Dienstprogramm LOAD unterstützt partitionierte Tabellen und fügt Datensätze in die gewählte Datenpartition ein, ohne dass die gesamte Tabelle gesperrt ist. LOAD kann die Vorteile der partitionsinternen Parallelität und der E/A-Parallelität nutzen. Da das Laden von Daten

eine CPU-intensive Operation ist, kann LOAD die Möglichkeit mehrerer Prozessoren nutzen, indem es Teilaufgaben, wie Daten lesen, einfügen oder Indexaufbauen darauf verteilt. In einer Umgebung mit partitionierten Datenbanken (DPF) 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. Dadurch, dass nur die aktive Partition gesperrt wird, werden weniger und kürzere Sperren benötigt. Da die anderen Partitionen davon unbeeinflusst sind, erhöht sich die Verfügbarkeit der Daten.

ezindämmen. Die Partitionierung und die automatische Speicherbereichsverwaltung vereinfachen die Administration und machen diese flexibler und effizienter.

Es wurde gezeigt, dass neben den technischen Möglichkeiten auch organisatorische Maßnahmen nötig sind, um mit großen Datenmengen optimal umgehen zu können. Durch die verschiedenen Funktionen von DB2 wird nicht nur die Verwaltung sondern auch der Zugriff auf große Datenmengen effizienter und schneller. Außerdem können durch ihren gezielten Einsatz Folgekosten eingespart werden.

Insgesamt kann die Verwendung der hier aufgezeigten Maßnahmen dazu führen, dass der Umgang mit Ressourcen effizienter und der Datenbankbetrieb entspannter stattfinden kann.

Das sind gute Gründe für Unternehmen und die betroffenen Administratoren, sich näher mit diesen Konzepten zu beschäftigen.

*Frank Pientka / Heinz Axel Pürner
frank.pientka@impaq-pluralis.com /
info@puerner.com*

Referenzen:

- DB2 Produktinformationen: <http://www-306.ibm.com/software/data/db2/v9/>
- DB2 online Dokumentation: <http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp>
- Leveraging DB2 Data Warehouse Edition 9.1 for Business Intelligence (Redbook SG247274), November 2006
- Größte Datenbanken der Welt: http://www.wintercorp.com/VLDB/2005_TopTen_Survey/TopTenWinners_2005.asp
- Raj Ahuja, Introducing DB2 9, Part 2: Table partitioning in DB2 9: <http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0605ahuja2/index.html>
- Paul McInerney, DB2 partitioning features: <http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0608mcinerney/>

Ausblick und Fazit

Ähnlich wie in der Medizin ist Vorsorge besser als Nachsorge, da mit einer rechtzeitigen und umfassenden Planung manche späteren „Schmerzen“ und Engpässe vermieden werden können. Die präventive Vermeidung von Redundanzen und der gezielte Einsatz der Datenkompression helfen, das Datenwachstum ein wenig

Frank Pientka ist Senior Consultant bei der IMPAQ Pluralis AG in Dortmund. Er ist zertifizierter DB2 Application Developer und beschäftigt sich mit DB2 seit der Version 2.
frank.pientka@impaq-pluralis.com

Heinz Axel Pürner ist selbstständiger Berater und Trainer in Dortmund. Er besitzt langjährige Erfahrungen mit DB2 auf den unterschiedlichsten Plattformen und ist mehrfacher Buchautor über DB2 auf der LUW-Plattform sowie Co-Autor eines Redbook (SG-24-6418-00) von IBM über DB2 z/OS.
info@puerner.com