Das Partitionieren von Tabellen ist ein Allroundmittel im Data-Warehouse-Umfeld. Die Vorteile reichen von der Administration bis zur Abfrage-Performance, und auch beim Laden ist der Nutzen groß. Der Artikel gibt praktische Hinweise beim Einsatz von Partitionierung.

Sieben gute Gründe für den Einsatz von Partitionierung im Data Warehouse

Detlef E. Schröder und Alfred Schlaucher, ORACLE Deutschland B.V. & Co. KG

In Data-Warehouse-Systemen können einzelne Tabellen in den Bereich von vielen Milliarden Sätzen anwachsen. Dies birgt verschiedene Herausforderungen, die trotz rasanter Weiterentwicklung der Hardware auch softwaretechnisch behandelt werden sollten.

Das Partitionieren großer Tabellen gehört mit zu den wichtigsten Hilfsmitteln des Oracle Data Warehouse. Es teilt die Daten großer Tabellen in physikalisch separierte Mengen und spricht diese Teilmengen separat an. Dadurch müssen bei entsprechenden Abfragen anstelle der gesamten Tabelle nur die Partitionsdaten gelesen werden, was zu einer besseren Abfrage-Performance führt. Darüber hinaus sind die kleineren Datenpakete eine handlichere Verarbeitungseinheit für die Verwaltung des Data Warehouse. Die Gesamtsicht auf alle Daten bleibt dennoch erhalten, sodass die bestehenden Abfragen oder Applikationen nicht verändert werden müssen. Zusätzlich entstehen durch die Aufteilung einzelne, definierte und physisch aufgeteilte Datensegmente (siehe Abbildung 1).

Abbildung 1: Partitionierte Tabellen haben mehrere Sichten. Die Tabelle kann als Ganzes oder in einzelnen Partitionen angesprochen werden.

Der Nutzen des Partitioning-Features einer Datenbank wird oft nur auf die Performance-Verbesserung reduziert. Dies trifft die Möglichkeiten und Vorteile aber nur zu einem, wenn auch wichtigen Teil. Insgesamt gesehen gibt es folgende sieben Gründe, die den Einsatz von Partitionierung empfehlen:

- Abfrage der Performance
- Unterstützung im Ladeprozess
- Vereinfachte Handhabung bei der Indizierung
- Vereinfachtes Update der Materialized View
- Unterstützung bei der Komprimierung
- Unterstützung im Backup-Prozess
- Unterstützung des Information Lifecycle Managements

Abfrage der Performance

Oft wird nur ein Teil der Daten für bestimmte Abfragen benötigt (die letzte Ladeperiode, der letzte Monat, das letzte Quartal etc.). Wenn die Daten in für die Abfrage passenden Häppchen vorliegen, also für jede Ladeperiode eines, dann müssen für eine Abfrage nach der aktuellen und der Vorperiode nur zwei Teilmengen gelesen werden und nicht wie im unpartitionierten Zustand alle 36 vorgehaltenen Ladeperioden. Dadurch wird die zu lesende und zu verarbeitende Datenmenge eingeschränkt und es entsteht ein Performance-Vorteil für die Abfrage. Darüber hinaus können Tabellen, die nach den gleichen Kriterien aufgeteilt sind, auch nur über die notwendigen Teilmengen gejoint werden, was ebenfalls einen Verarbeitungsvorteil erzeugt und die Performance steigert.

Wie werden nun die Inhalte auf die Partitionen aufgeteilt und welche Varianten gibt es? Ein wichtiges Merkmal der Partitionierung ist der Partition Key. Das ist das Feld in der Tabellen-Struktur, über dessen Inhalt das System die Zuordnung von Sätzen zu einer bestimmten Partition festlegt. Damit ist ein sehr praktisches Feature des Partitioning benannt: das System, das bei einem INSERT einen Satz automatisch in die passende Partition aufnimmt. Derjenige, der den ETL-Prozess steuert, muss sich also darum nicht kümmern.

Die Partition-Variante (Range, List, Hash, System, Virtual Column) bestimmt den Umgang mit dem Partition Key. Die am häufigsten genutzte Partitioning-Variante ist das Rang-Partitioning, bei dem die Einteilung der Partitionen entlang definierter Wertebereiche erfolgt. Das im Data Warehouse am häufigsten genutzte Kriterium ist die Zeit (Tage, Monate etc.), also ein Feld mit einem Datumswert (siehe Listing 1).

Andere Partitionierungs-Kriterien können auch zusammengesetzte Felder, über Funktionen ermittelte Werte oder nach dem Alphabet sortierte Werte sein. Diese Varianten lassen sich über Sub-Partitioning in einer zweiten Ebene mischen, um das Partitionieren noch flexibler zu gestalten. Das grundlegende Partitionierungskriterium ist beispielsweise eine Monatseinteilung (Range); die Monatspartitionen können noch weiter nach Verkaufsgebieten (List) unterteilt sein. Mit Sub-Partitioning gewinnt man ein zusätzliches

```
CREATE TABLE F_bestellung_part_range (
SUMME NUMBER(14,0),
MENGE NUMBER(14,0),
BESTELLDATUM DATE,
FK_ARTIKEL_ID NUMBER,
FK_KUNDEN_ID NUMBER,
FK_ORT_ID NUMBER,
FK_DATUM_ID NUMBER,
auftragsart VARCHAR2(30))
PARTITION BY RANGE (bestelldatum) (
PARTITION jan11 VALUES LESS THAN (TO_DATE(,2011-02-01‘, ,SYYYY-MM-
DD‘)) TABLESPACE DWH_SPINDEL,
…
PARTITION feb12 VALUES LESS THAN (TO_DATE(,2012-03-01‘, ,SYYYY-MM-
DD‘)) TABLESPACE DWH_SPINDEL,
PARTITION next_month VALUES LESS THAN (MAXVALUE) TABLESPACE DWH_SPIN-
DEL);
```

Listing 1

```
CREATE TABLE F_BESTELLUNG
SUMME NUMBER(14)
MENGE NUMBER(14)
BESTELLDATUM DATE
FK_ARTIKEL_ID NUMBER
FK_KUNDEN_ID NUMBER
FK_ORT_ID NUMBER
FK_DATUM_ID NUMBER
AUFTRAGSART VARCHAR2(30) ... ;
```

Listing 2

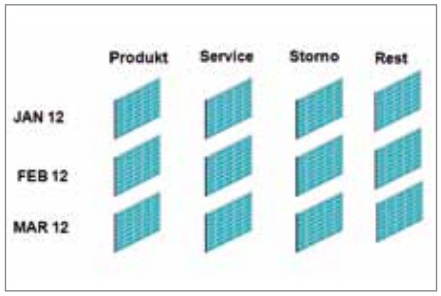

Abbildung 2: Sub-Partitioning „Range/List“

```
select sum(summe) from F_BESTELLUNG_[xxx] where Auftragsart = ‘SERVICE’
and BESTELLDATUM = to_date(‘10.10.2011’,’DD.MM.YY’);
```

Listing 3

Tabelle	Blocks	GB	Millionen Zeilen	Abfragezeit
F_Bestellugen_X	2431440	19,45	411	314,3 sec
F_Bestellugen_Range	2429231	19,43	411	17,3 sec
F_Bestellugen_Range _List	2463418	19,71	411	5,6 sec

Tabelle 1

fachliches Abfragekriterium zu dem der Datengewinnung oder -entstehung – hier der Zeit.

Ein einfacher Test zeigt bereits die Performance-Effekte des Partitioning und Sub-Partitioning. Die Beispiel-Faktentabelle wird in drei Varianten unterteilt: nicht partitioniert, partitioniert (Range) und sub-partitioniert (Range/List) (siehe Listing 2). Listing 3 und Tabelle 1 zeigen die Beispielabfrage auf die Daten und die entsprechenden Laufzeiten.

Wie deutlich zu erkennen ist, liefert die sub-partitionierte Variante das beste Ergebnis, noch vor der einfachen Partitionierung. Wie bereits gesagt: Im Data Warehouse ist Partitionierung bei großen Tabellen ein Muss.

Partition Wise Join

Der Zugriff auf ein Data Warehouse erfolgt meist durch ein Starschema, das aus mehreren Dimensionen und Fakten besteht. Dies bewirkt bei einer Abfrage eine Reihe von Joins zwischen einer meist sehr großen Fakten-Tabelle und mehreren Dimensions-Tabellen. Ist die Fakten-Tabelle partitioniert, kann eine Abfrage mit einem Join zwischen Fakten mehrerer Dimensions-Tabellen in mehrere kleine Joins zergliedert werden. Für jede Partition der Fakten-Tabelle wird ein eigener Join mit der Dimensions-Tabelle gebildet (Partial Partition Wise Join).

Ist die Dimensions-Tabelle ebenfalls und nach dem gleichen Partitionierungsschlüssel wie die Fakten-Tabellen partitioniert, so werden einzelne Teil-Joins zwischen den zusammengehörenden Partitionen gebildet (Full Partition Wise Join). Dieses Vorgehen findet auch bei Sub-Partitionierung statt.

Unterstützung im Ladeprozess

Partitioning bietet immer einen leichten Zugriff auf die Daten einer Ladeperiode. Wählt man für die neu zu ladenden Daten einer Ladeperiode eine Partition, so kann man diese schnell zu einer Warehouse-Tabelle hinzufügen oder im Fehlerfall wieder entfernen. Dieses Verfahren nennt sich „Partition Exchange and Load“ (PEaL). In fünf einfachen Schritten lässt sich so – ohne die vorhandene partitionierte Tabelle zu beinträchtigen und Abfragen darauf zu behindern – eine ebenfalls sehr große partitionierte Tabelle pflegen. Dazu hängt man zunächst eine leere Partition an die bestehende Tabelle an. Dann werden der Ladevorgang in einer separaten (temporären) Tabelle vollzogen und auch dort die Indizes gepflegt. Anschließend lässt sich in einem Schritt die leere Partition gegen die gepflegte Tabelle austauschen. Die neuen Daten stehen mit gepflegtem Index sofort zur Verfügung.

Diese Vorgehensweise vereinfacht viele ETL-Aufgaben und ermöglicht

auch Datenpflege während eines Online-Betriebs. Es ist eine hilfreiche Unterstützung, wenn man bei einem ETL-Schritt immer eine ganze Tabelle neu erzeugt, da dann die bestehende Tabelle mit einer Partition gebildet wird und die neu gepflegte separate, temporäre Tabelle in einem Schritt die bestehende ersetzt. Die Partitionierung bietet gerade in diesem Bereich eine enorme Hilfestellung, die zumindest die Wartungszeit verkürzt.

Vereinfachte Handhabung bei der Indizierung

Die Aktualisierung der Indizes stellt im ETL einen großen zeitlichen Aspekt dar. Partitionen hingegen, wie schon beim PEaL beschrieben, lassen sich separat voneinander indizieren (Local Index). Damit spart man sich das zeitaufwändige Aktualisieren eines kompletten (globalen) Index. Da sich Indizes ebenfalls partitionieren lassen, kann man auch einen Index anders partitionieren als die zugrunde liegende Tabelle. Dies ist in speziellen Situationen sinnvoll.

Vereinfachtes Update der Materialized View

Im Data Warehouse kommen oft Materialized Views (MAV) zum Einsatz. Diese führen meist als Aggregations-Tabellen zur Performance-Steigerung. Die Pflege dieser MAV kann durch Log-Tabellen an den zugrunde liegenden Tabellen oder aber über das Partition Change Tracking (PCT) erfolgen. Die Datenbank erkennt, welche Partitionen sich verändert haben oder neu erstellt wurden, und kann dann bei der Aktualisierung einer MAV selber entscheiden, welche Deltas nachzuladen sind. Dies ist in vielen Situationen schneller und effizienter als die MAV-Logs. Auch hier bietet das Partitionierungsverfahren einen erheblichen Vorteil beim Management des Data Warehouse.

Unterstützung bei der Komprimierung

Oracle bietet unterschiedliche Komprimierungsvarianten an. Partitioning kann je nach Bedarf Daten derselben Tabelle unterschiedlich komprimieren. OLTP-Komprimierung ist das kostenpflichtige Feature „Advanced Kompression“ der Datenbank. Es komprimiert Tabellen oder Partitionen, die noch geändert werden. Für Updates und Inserts bietet sich die kostenfreie Basic-Kompression nicht an. Durch Partitionierung ist es aber möglich, aktive Partitionen mit OLTP oder gar nicht zu komprimieren und inaktive Partitionen mit „Basic“. Dieser Mix spart Plattenplatz und durch die geringere Menge der zu entkomprimierenden Daten werden die Abfragen sogar schneller. Partitioning ist daher in den meisten Data-Warehouse-Systemen unter diesem Gesichtspunkt dringend zu empfehlen.

Unterstützung im Backup-Prozess

Die meisten Daten eines Data-Warehouse-Systems ändern sich nicht mehr, nachdem sie einmal gespeichert sind – und das oft über Jahre hinweg. Deswegen sind auch nur die immer wieder aktuell hinzukommenden Daten zu sichern. Große Tabellen lassen sich durch Partitioning leicht in Partitionen aufteilen, die sich geändert haben, und solche, die nur ältere Daten beinhalten. Nur die Änderungen sind beispielsweise mit RMAN zu sichern.

Durch die Möglichkeit, mehrere Partitionen zusammenzufügen, lassen sich Monatspartitionen des Vorjahres zu einer Vorjahres-Partition zusammenfassen und archivieren. Diese lassen sich auch von der Tabelle abschneiden und nur im Revisionsfall wieder anhängen. Die Archivierung wird damit sehr vereinfacht und ein Full-Backup, wie leider manchmal immer noch zu sehen, entfällt.

Unterstützung des Information Lifecycle Managements

Beim Information Lifecycle Management (ILM) speichert man unterschiedlich alte und unterschiedlich häufig genutzte Daten auf verschieden teure, schnelle und ausfallsichere Platten. Während traditionelle Speichermittel (Plattensysteme) immer günstiger geworden sind, ist ihre Performance gleich geblieben. SSD-Platten sind heute erschwinglich geworden und können partiell Spindel-Festplatten ersetzen. Sie sind allerdings immer noch zu teuer, um etwa 50 Terabyte große DWH-Systeme zu versorgen.

Die aktuellen Daten einer mit historischen Daten gefüllten Fakten-Tabelle hingegen können durchaus auf einzelnen SSD-Platten gelagert sein. Das Partitionieren ermöglicht dies durch das Zuweisen verschiedener Datenträger für die unterschiedlichen Partitionen. Beim ILM werden also verschiedene, bereits erwähnte Möglichkeiten einer Partitionierung ausgenutzt und zusammen angewendet. Der Enterprise Manager verwaltet die Partitionierung und steht zur Analyse zur Verfügung.

Fazit

Die Partitionierung bietet einen enormen Vorteil bei der Handhabung, Erstellung, Wartung und Abfrage von (nicht nur) großen Tabellen im Data Warehouse und ist in vielen Fällen ein Muss.

Detlef E. Schröder
detlef.e.schroeder@oracle.com

Alfred Schlaucher
alfred.schlaucher@oracle.com