Partitioning – Technik und Anwendungsbeispiele

Klaus Reimers
ORDIX AG
Köln

Schlüsselworte:

Range Partitioning, Hash Partitioning, List partitioning, System Partitioning, Interval Partitioning, Virtual Column Based Partitioning, Partitioning by Reference, Local Index, Global Index, Composite Partitioning

Einleitung

Partitionierung von Tabellen und Indizes bedeutet die Aufteilung von großen Tabellen und Indizes in kleinere Einheiten, die separat verwaltet und angesprochen werden können. Anstelle der Verwaltung einer Tabelle als ein großes monolithisches Objekt, beruht die Partitionierung auf der sogenannten "divide and conquer" Technik, die eine skalierbare Performance für große Datenmengen zur Folge hat. Wie viele kleinere Partitionen hier entstehen, hängt wohl mehr von der organisatorischen Kraft der Datenbankadministrators ab, da die ORACLE Obergrenze von 65.536 Partitionen pro Tabelle oder Index viel Spielraum lässt.

Die Partitionierung führt zu folgenden Vorteilen:

- Reduzierung der Zeit für Verwaltungsaufgaben (kleinere Verwaltungseinheiten)
- Verbesserung der Performance (durch Erhöhung der Parallelität)
- Verbesserung der Verfügbarkeit (durch Reduzierung der Folgen eines Ausfalls)
- Verbesserte Unterstützung für sehr große Datenbanken

Objektiv muss angemerkt werden, dass diese Vorteile insgesamt durch einen erhöhten Verwaltungsaufwand erkauft werden, denn Partitionen müssen definiert, angelegt und verwaltet werden. Allerdings sind die Konzepte zur Partitionierung speziell für sehr große Datenmengen (> 10 GB oder > 100 Millionen Zeilen) entwickelt worden, so dass sich dieser Aufwand schnell amortisiert.

Alle Partitionen einer Tabelle oder eines Indexes besitzen die gleichen logischen, aber unter Umständen andere physikalische Attribute. Beispielsweise haben alle Partitionen einer Tabelle die gleichen Spalten- und Constraint-Definitionen und die Partitionen eines Indexes basieren auf den gleichen Indexspalten. Demgegenüber können Spezifikationen bzgl. der Tablespaces durchaus variieren für unterschiedliche Partitionen einer Tabelle oder eines Indexes.

Im Vortrag werden die einzelnen Techniken vorgestellt und die Einsatzmöglichkeiten hinterfragt. Praxisbeispiele und eine mitlaufende Demo runden den Vortrag ab.

Range Partitioning

Innerhalb einer Bereichspartitionierung werden Zeilen in Abhängigkeit von Spaltenwertebereichen auf Partitionen abgebildet. Die Bereichspartitionierung wird festgelegt durch die Partitionierungs-Spezifikation einer Tabelle (Partition Key):

`PARTITION BY RANGE` (<Spaltenliste>)

und durch die Partitionierungs-Spezifikation einzelner Partitionen:

`VALUES LESS THAN` (<Werteliste>)

wobei

- eine Spaltenliste eine sortierte Liste von Spalten darstellt. Sie bestimmt, zu welcher Partitionierung eine Zeile oder ein Indexeintrag gehört.
    - Diese Spalten werden '`partitioning columns`' genannt
    - Die Werte in den '`partitioning columns`' einer bestimmten Zeile bilden den '`partitioning key`' dieser Zeile

Die Range Partitionierung wird häufig im Zusammenhang mit einer datumsorientierten Verteilung der Sätze verwendet.

Hash Partitioning

Die Hash-Partitionierung erlaubt die Partitionierung von Tabellen auch dann, wenn keine eindeutige Partitionierungs-Spalte vorliegt. Die Daten werden aufgrund eines Hash-Algorithmus auf die entsprechenden Partitionen verteilt. Die Anzahl der Partitionen sollte eine Potenz von 2 sein.

Dieses Verfahren wird auch dann genutzt, wenn „hot blocks" entzerrt werden sollen.

List Partitioning

In Oracle 8 war es nicht möglich, eine Tabelle durch direkte Zuordnung von Werten zu partitionieren. Diese Möglichkeit ist seit Oracle 9i durch das List-Partitioning gegeben. Selbstverständlich sind hier auch Tablespace-Zuweisungen und Storage Parameter anwendbar. Hier gelten die gleichen Bedingungen, die auch bei den anderen Partitionsformen gelten, allerdings darf hier nur eine Spalte im Partition Key verwendet werden. Ab Oracle 9.2 ist es über das Schlüsselwort `DEFAULT` möglich, eine Partition zu definieren, in die alle Sätze gelangen, die nicht einer Liste zugeordnet sind.

System Partitioning

Die System-Partitionierung ermöglicht dem Anwendungsentwickler eine partitionierte Tabelle zu erzeugen, deren Datenaufteilung vollständig durch die Applikation erfolgt. Es gibt dabei keinen Mechanismus der Datenbank, aufgrund der Daten eine Partition zu bestimmen. Dies ermöglicht einerseits eine vollkommen freie Verwendung der Partitionierung durch die Applikation (`SYSTEM`), andererseits kann die Datenbank auch nur sehr eingeschränkt den Zugriff auf die Tabelle optimieren.

Durch die Struktur der System-Partitionierung ergibt sich, dass beim Einfügen von Daten grundsätzlich die Partition mit angegeben werden muss. Der Name der Partition darf keine Bind-Variable sein. Die Fehlermeldung ORA-14701 legt aber nahe, dass dies in einer der kommenden Versionen noch implementiert wird.

Die Referenzierung der Partitionen beim Lesen und beim Löschen der Daten ist optional. Jedoch muss die Datenbank auf alle Partitionen zugreifen, wenn die Partition nicht gegeben ist. Eine Einschränkung der Partitionen durch den Optimizer aufgrund von Werten ist bei der System-Partitionierung strukturbedingt nicht möglich.

Im Gegensatz zu den anderen drei Methoden (Range/Hash/List) ist dieses Verfahren nicht transparent für die Applikation, hier bestimmt der Entwickler die Vorgehensweise.

Virtual Column Based Partitioning

Mit Oracle 11g gibt es die Möglichkeit virtuelle Spalten auf Tabellen anzulegen. In der Vergangenheit hatte man häufig das Problem, dass über eine Funktion oder einen berechneten Wert partitioniert werden sollte. Dieses Problem wird bei Verwendung von virtuellen Spalten minimiert, wenn auch nicht ganz vermieden.

Interval Partitioning

Ein Problem der Range-Partitionierung besteht immer darin, dass in regelmäßigen Abständen neue Partitionen hinzugefügt werden müssen oder aber die letzte Partition geteilt werden muss. Dies bedeutet einen administrativen Aufwand. Falls Daten hinzugefügt werden, für die es noch keine Partition gibt, so brach der Befehl zum Einfügen mit einer Fehlermeldung ab.

Zur Lösung dieses Problems wurde die Intervall-Partitionierung entworfen. Bei der Intervall-Partitionierung werden Partitionen automatisch angelegt, sobald die entsprechenden Daten eingefügt werden. Die Grenzen der Partition bestimmen sich dabei durch die Grenze bestehender Partitionen und einer Intervall-Funktion. Der Bereich der Partition bestimmt sich ausschließlich durch die Intervall-Funktion. Alle automatisch angelegten Partitionen haben den gleichen Datenbereich (`equi-width`).

Zur Bildung der Partitionsgrenzen muss eine Intervall-Funktion angegeben werden. Diese Intervall-Funktion muss entweder ein numerisches Intervall zurückgeben oder ein Datumsintervall. Bei der Intervall-Partitionierung werden die Partitionsnamen dynamisch erzeugt. Bei administrativen

Aufgaben ist es nun allerdings relativ aufwändig, aus dem Data-Dictionary die Grenzen zu extrahieren und daraus den Partitionsnamen zu gewinnen. Zur einfachen Ermittlung einer Partition stellt die Datenbank nun die so genannte `FOR`-Klausel zu Verfügung. In der `FOR`-Klausel wird ein beliebiger Datenwert angegeben. Zu diesem Datenwert ermittelt die Datenbank dann die entsprechende Partition und ersetzt die `FOR`-Klausel intern durch die Partition-Klausel mit der errechneten Partition.

REF Partitioning

Sind zwei Tabellen über Foreign-Key-Constraints miteinander verknüpft und ist die so genannte Master-Tabelle partitioniert, so kann es durchaus sinnvoll sein, auch die Detail-Tabelle nach dem gleichen Prinzip zu partitionieren. Steht in der Detail-Tabelle jedoch der Partitionierungs-Schlüssel nicht zur Verfügung, stößt man bisher auf Grenzen.

Die `REF`-Partitionierung hebt diese Grenzen auf. Sie ermöglicht die Partitionierung einer Detail-Tabelle nach Maßgabe einer Master-Tabelle. Hierzu ist eine aktive referentielle Integrität notwendig. Außerdem muss die Foreign Key Spalte zwingend `NOT NULL` sein.

Partitionierte Indizes

Eine Indizierung kann sowohl auf partitionierte als auch auf normale Tabellen vorgenommen werden. Oracle unterstützt vier Indextypen, mit deren Hilfe auf die eigenen Anforderungen eingegangen werden kann:

- Nonpartitioned Indexes
- Global prefixed Indexes
- Local prefixed Indexes
- Local nonprefixed Indexes

In aller Regel verwendet man locale Indizes auf partitionierten Tabellen, globale Indizes werden vor allem zur Sicherstellung von `UNIQUE` Constraints genutzt, sind aber in der Pflege sehr viel aufwändiger.

Composite Partitioning

Sind die Partitionen immer noch so groß, dass eine weitere Unterteilung sinnvoll ist, spricht man von Subpartitionen. Dieses Verfahren kann in diversen Konstellationen genutzt werden. Bis Oracle 10g war Range/Hash und Range/List möglich, mit Oracle 11g sind Range/Range, List/Range, List/Hash und List/List hinzugekommen. In der Praxis werden hierüber organisatorische Verfahren abgebildet oder aber es wird zur Vermeidung von „hot blocks“ genutzt.

Fazit

Die technischen Möglichkeiten im Zusammenhang mit der Partitionierung sind fast unerschöpflich. Leider ist die Nutzung dieses Features kostenpflichtig, was in Projekten bei Kunden immer wieder Überzeugungsarbeit erfordert, denn die Entscheider sind in aller Regel nicht bereit zusätzliches Geld auszugeben.

Kontaktadresse:

Klaus Reimers
ORDIX AG
Westernmauer 12-16
D-33098 Paderborn

Telefon:	+49 (0) 611- 778 4000
Fax:	+49 (0) 180- 167 3490
E-Mail	info@ordix.de
Internet:	www.ordix.de