# Einsatz von Partial Indices im Data Warehouse

Reinhard Mense, areto consulting gmbh

Für performante Abfragen in Data-Warehouse-Systemen sind Indizes unverzichtbar. Aufgrund des sehr großen Datenvolumens im Data Warehouse kann deren Erstellen aber aufwändig und zeitraubend sein. Die mit Oracle 12*c* verfügbaren „Partial Indices" können die Einführung neuer Indizes in ein Data Warehouse erleichtern.

Partial Indices können nur für partitionierte Tabellen angewendet werden und müssen analog zur Tabelle partitioniert sein. Die Idee dabei ist, den Index nur für einen Teil der Partitionen zu erstellen. Somit kann die benötigte Zeit für den Aufbau oder Rebuild des Index im Gegensatz zur vollständigen Erstellung deutlich reduziert werden. Allerdings ist der Index dann auch nicht beim Zugriff auf allen Partitionen nutzbar. Es lassen sich sowohl „Partial B*Tree"-Indices als auch „Partial Bitmap"-Indices erstellen. Partial Indices können jedoch nicht für Unique Indices erzeugt werden.

**Partial Index für Fakten-Tabellen**

In DWH-Systemen sind Partial Indices besonders für Fakten-Tabellen geeignet. Diese werden in der Regel partitioniert und meist dient ein Datum als Partitionsschlüssel. Fragen die Benutzer aktuelle Daten häufiger ab, erfolgt der Zugriff somit auch nur auf einige wenige Partitionen. Beim Aufbau eines neuen oder beim Rebuild eines existierenden Index bringt dieser also den größten Nutzen für die Partitionen, in denen sich die aktuellen Daten befinden. Am Beispiel einer Fakten-Tabelle werden im Folgenden die Erstellung, die Wirkungsweise und der Nutzen eines Partial Index gezeigt.

Um einen Partial Index zu erzeugen, ist die Klausel „INDEXING PARTIAL" anzugeben (siehe Listing 1). Damit wird der Index als „Partial" definiert,

```
create bitmap index bx_verkauf_produkt
    on fakt_verkauf (produkt_id)
 local nologging indexing partial;
```

*Listing 1: Erstellen eines Partial Index mit „INDEXING PARTIAL"*

```
create table fakt_verkauf
 (datum       date
, produkt_id  number
, filiale_id  number
, menge       number
, umsatz      number)
pctfree 0 nologging
partition by range (datum)
 (partition m201301 values less than (to_date ('01.02.2013', 'dd.mm.yyyy')) indexing off
, partition m201302 values less than (to_date ('01.03.2013', 'dd.mm.yyyy')) indexing off
, partition m201303 values less than (to_date ('01.04.2013', 'dd.mm.yyyy')) indexing off
, partition m201304 values less than (to_date ('01.05.2013', 'dd.mm.yyyy')) indexing off
, partition m201305 values less than (to_date ('01.06.2013', 'dd.mm.yyyy')) indexing off
, partition m201306 values less than (to_date ('01.07.2013', 'dd.mm.yyyy')) indexing off
, partition m201307 values less than (to_date ('01.08.2013', 'dd.mm.yyyy')) indexing off
, partition m201308 values less than (to_date ('01.09.2013', 'dd.mm.yyyy')) indexing off
, partition m201309 values less than (to_date ('01.10.2013', 'dd.mm.yyyy')) indexing on
, partition m201310 values less than (to_date ('01.11.2013', 'dd.mm.yyyy')) indexing on
, partition m201311 values less than (to_date ('01.12.2013', 'dd.mm.yyyy')) indexing on
, partition m201312 values less than (to_date ('01.01.2014', 'dd.mm.yyyy')) indexing off
, partition pdefault values less than (maxvalue) indexing off);
```

*Listing 2: Definition der Tabellen-Partitionen für den Partial Index*

```
create bitmap index bx_verkauf_filiale
    on fakt_verkauf (filiale_id)
local nologging indexing full;
```

*Listing 3: Erstellen eines vollständigen Index mit „INDEXING FULL“*

| Partition | Status für Partial Index BX_VERKAUF_PRODUKT | Status für vollständigen Index BX_VERKAUF_FILIALE |
|---|---|---|
| M201301 | UNUSABLE | USABLE |
| M201302 | UNUSABLE | USABLE |
| M201303 | UNUSABLE | USABLE |
| M201304 | UNUSABLE | USABLE |
| M201305 | UNUSABLE | USABLE |
| M201306 | UNUSABLE | USABLE |
| M201307 | UNUSABLE | USABLE |
| M201308 | UNUSABLE | USABLE |
| M201309 | USABLE | USABLE |
| M201310 | USABLE | USABLE |
| M201311 | USABLE | USABLE |
| M201312 | UNUSABLE | USABLE |
| PDEFAULT | UNUSABLE | USABLE |

*Tabelle 1: Status von Partial Index und vollständigem Index im Vergleich*

es wird jedoch noch nicht angegeben, welche Index-Partitionen aufgebaut werden. Dies bestimmen die Klauseln „INDEXING ON“ und „INDEXING OFF“ bei der Definition der Tabellen-Partitionen (siehe Listing 2). Für mit „INDEXING ON“ definierte Partitionen werden die Index-Partitionen aufgebaut, bei „INDEXING OFF“ findet der Aufbau hingegen nicht statt.

Um die Wirkungsweise des Partial Index zu verdeutlichen, wird im hier verwendeten Beispiel noch ein weiterer, diesmal aber vollständiger Index erzeugt. Dazu kann optional die Klausel „INDEXING FULL“ angegeben werden (siehe Listing 3).

Überprüft man nach dem Erstellen der beiden Indizes den Status der Index-Partitionen im Data Dictionary, wird der Unterschied zwischen dem Partial Index und dem vollständigen Index deutlich (siehe Tabelle 1). Während beim vollständigen Index der Status für alle Partitionen „USABLE“ ist, ist das beim Partial Index nur für Partitionen der Fall, die bei der Tabellendefinition mit „INDEXING ON“ definiert worden sind. Beim Erstellen des Partial Index wurde also nur ein Teil der Index-Partitionen aufgebaut.

### Speicherplatz sparen durch Partial Index

„ALTER TABLE“ passt die aufgebauten Partitionen für den Partial Index bei Bedarf an. Ändert man eine Partition von „INDEXING OFF“ auf „INDEXING ON“, wird diese bei allen Partial Indices sofort aufgebaut. Ein expliziter Rebuild der Partition ist für die Partial Indices nicht notwendig. Ebenso lässt sich eine Partition von „INDEXING ON“ auf „INDEXING OFF“ umstellen. In diesem Fall wird die Index-Partition sofort „UNUSABLE“.

Listing 4 zeigt ein Beispiel für die hier verwendete Fakten-Tabelle. Somit kann für diese sehr einfach ein „Moving Window“ für die aufzubauenden Partitionen realisiert werden (siehe Tabelle 2). Im dargestellten Beispiel werden immer nur die drei aktuellsten Partitionen aufgebaut. Durch den Einsatz dieses Verfahrens reduziert sich der Speicherbedarf

```
alter table fakt_verkauf modify partition m201309 indexing off;
alter table fakt_verkauf modify partition m201312 indexing on;
```

*Listing 4: Anpassung der Partitionen für Partial Index*

| Partition | Status für Partial Index BX_VERKAUF_PRODUKT vor ALTER TABLE | | Status für Partial Index BX_VERKAUF_PRODUKT nach ALTER TABLE |
|---|---|---|---|
| M201301 | UNUSABLE | | UNUSABLE |
| M201302 | UNUSABLE | | UNUSABLE |
| M201303 | UNUSABLE | | UNUSABLE |
| M201304 | UNUSABLE | | UNUSABLE |
| M201305 | UNUSABLE | | UNUSABLE |
| M201306 | UNUSABLE | | UNUSABLE |
| M201307 | UNUSABLE | | UNUSABLE |
| M201308 | UNUSABLE | | UNUSABLE |
| M201309 | USABLE | → | UNUSABLE |
| M201310 | USABLE | | USABLE |
| M201311 | USABLE | | USABLE |
| M201312 | UNUSABLE | → | USABLE |
| PDEFAULT | UNUSABLE | | UNUSABLE |

*Tabelle 2: Status von Partial und Full Index im Vergleich*

für den Partial Index im Vergleich zu einem vollständig aufgebauten Index auf ein konstant niedrigeres Niveau.

**Schrittweise Einführung eines neuen Index**

Bei sehr umfangreichen Fakten-Tabellen kann der vollständige Aufbau eines neuen Index sehr viel Zeit in Anspruch nehmen. Oft steht für die Erstellung eines neuen Index jedoch nur ein begrenztes Zeitfenster zwischen der Ausführung der ETL-Prozesse und dem Zugriff der DWH-Benutzer zur Verfügung, sodass die zur Verfügung stehende Zeit nicht für den vollständigen Aufbau des Index ausreicht.

Ist der neue Index hingegen als Partial Index erzeugt und definiert man zunächst sämtliche Tabellen-Partitionen mit „INDEXING OFF“, lässt sich der Index anschließend partitionsweise aufbauen. Dazu sind die Tabellen-Partitionen schrittweise auf „INDEXING ON“ zu setzen. Somit kann die Einführung des neuen Index auf mehrere Zeitfenster verteilt erfolgen, sodass die begrenzt zur Verfügung stehende Zeit keinen Engpass mehr darstellt. Bei diesem Verfahren sollte man abschließend den Partial Index mit „ALTER INDEX <index_name> INDEXING FULL“ in einen vollständigen Index umdefinieren, damit der Status der Index-Partitionen bei der Einführung weiterer Partial Indices unberührt bleibt.

**Schrittweiser Rebuild eines Index**

Soll ein bereits existierender vollständiger Index schrittweise aufgebaut werden, kann das zuvor beschriebene Verfahren für die Einführung neuer Indizes ebenfalls angewendet werden. Für den Index „BX_VERKAUF_FILIALE“ aus dem hier verwendeten Beispiel ist dabei folgender Ablauf vorzusehen:

1. Umdefinieren des vollständigen Index in einen Partial Index mit „ALTER INDEX BX_VERKAUF_FILIALE INDEXING PARTIAL“
2. Alle Tabellenpartitionen auf „INDEXING OFF“ setzen
3. Schrittweise alle Tabellen-Partitionen wieder auf „INDEXING ON“ setzen. Dabei findet automatisch der Rebuild der jeweiligen Partitionen des Index „BX_VERKAUF_FILIALE“ statt.
4. Umdefinieren des Partial Index in einen vollständigen Index mit „ALTER INDEX BX_VERKAUF_FILIALE INDEXING FULL“

Mit diesem Verfahren kann der schrittweise Rebuild auch für mehrere Partial Indices gleichzeitig erfolgen.

**Fazit**

Oracle Database 12*c* bietet mit der Einführung der Partial Indices mehr Flexibilität beim Aufbau der Indizes. Ihr Einsatz eignet sich besonders für große partitionierte Fakten-Tabellen im Data Warehouse. Sie ermöglichen für die Erstellung neuer und den Rebuild existierender Indizes den schrittweisen Aufbau, sodass auch kürzere Wartungszeitfenster für die Durchführung dieser Aufgaben genutzt werden können. Für Fakten-Tabellen mit besonders umfangreichen historischen Datenmengen können Partial Indices zudem genutzt werden, um den Speicherbedarf zu begrenzen.

Reinhard Mense
reinhard.mense@areto-consulting.de

# Replikation großer Datenmengen in einem OLTP-System

Uwe Simon, T-Systems Telekom IT

Bei der Replikation von Daten bietet Oracle mit Read-only-Snapshots, Multimaster-Replikation, Streams oder GoldenGate die passenden Replikationsverfahren für verschiedene Anwendungsfälle an. Eine Replikation mit Read-only-Snapshots ist sehr einfach zu implementieren und für sehr viele Anwendungsfälle gut geeignet. Doch wo sind die Grenzen und wo muss man bei Problemen suchen? Am Beispiel der Replikation in einem großen OLTP-System wird gezeigt, wo diese Grenzen liegen und wie man Probleme identifizieren und lösen kann.

Die Installation einer Replikation mit Read-only-Snapshots, auch „Materialized View“ (MView) genannt, ist recht einfach und in den Handbüchern gut beschrieben. Sie bietet sich an, wenn man Daten aus einem produktiven System für reine Lesezugriffe auf einem oder mehreren anderen Systemen zur Verfügung stellen muss. Nachfolgend