Indexstrategien im Data Warehouse

Reinhard Mense
areto consulting gmbh
Köln

Schlüsselworte
Business Intelligence, Data Warehouse, Bitmap Index, B*Tree Index, Partial Index, Star Schema, Snowflake Schema, Dimensionale Modellierung, Performance

Einleitung

Performante Abfragen im Data Warehouse lassen sich nur mit einer geeigneten Indexierung der Tabellen erzielen. Die Auswahl der richtigen Index-Spalten und der passenden Indextypen sind deshalb essentiellen Fragen bei der Datenmodellierung. Es ist eine geeignete Indexierung insbesondere für Faktentabellen und Dimensionen zu wählen. Auch schwierige Fälle, wie z. B. besonders große Dimensionen, sind dabei zu betrachten.

Im Folgenden wird anhand von Beispielen gezeigt, welche Spalten für eine Indexierung im Data Warehouse in Betracht zu ziehen sind. Der Aufbau und die Vorteile von Bitmap Indizes werden erläutert und deren sinnvolle Anwendung gezeigt.

Aus der Verwendung der Indizes durch den Oracle Optimizer kann abschließend eine geeignete Indexstrategie für Data Warehouse-Systeme abgeleitet werden.

Szenarien im Data Warehouse

Bei der Frage nach einer passenden Indexierung in einem Data Warehouse stehen insbesondere die Faktentabellen und Dimensionen im Fokus, da auf diese von den Endbenutzern mit Hilfe von Reporting-Tools zugegriffen wird. Der Zugriff kann dabei sowohl in Form von Standardberichten als auch Ad-hoc-Abfragen erfolgen.

Für die Indexierung sind vor allem die bei den Berichten und Abfragen verwendeten Filter- und Join-Bedingungen interessant, da diese die zu lesende Datenmenge der beteiligten Tabellen einschränken. Für Ad-hoc-Abfragen sind die vom Benutzer verwendeten Filterkriterien oftmals jedoch nicht vorhersehbar. Deshalb erfordern sie eine möglichst allgemein nutzbare Indexierung. Aber auch für die bei Standardberichten verfügbaren Filtermöglichkeiten ist eine geeignete Indexierung vorzusehen.

Um die geeignete Indexierung für typische Szenarien im Data Warehouse aufzuzeigen wird ein Data Mart mit einer Faktentabelle und unterschiedliche Dimensionstabellen betrachtet. Die Dimensionstabellen unterscheiden sich dabei in ihrer Anzahl Datensätze. Jede Dimensionstabelle enthält neben einem künstlichen Schlüssel (ID) und einem natürlichen bzw. fachlichen Schlüssel (NK) mehrere Attributspalten mit unterschiedlicher Selektivität.

Dimension	#rows	Spalte attribut_1	Spalte attribut_5	Spalte attribut_10	Spalte attribut_20	Spalte attribut_25	Spalte attribut_50
D100	100	1	5	10	20	25	50
D1000	1.000	10	50	100	200	250	500
D10000	10.000	100	500	1.000	2.000	2.500	5.000
D100000	100.000	1.000	5.000	10.000	20.000	25.000	50.000
D1000000	1.000.000	10.000	50.000	100.000	200.000	250.000	500.000

Tabelle 1: Dimensionen des Testszenarios

In der Tabelle 1 ist neben der Gesamtzahl der Datensätze (#rows) für die einzelnen Attributspalten die Anzahl Datensätze pro Attributwert angegeben. Im verwendeten Testszenario kommen die Werte einer Attributspalte stets gleich häufig vor.

Anhand dieses Testszenarios werden im Folgenden der Einfluss der Tabellengröße und der Selektivität der Attributspalten aufgezeigt.

Verwendung von B*Tree oder Bitmap Index

Ein B*Tree-Index wird von der Oracle Datenbank als ausbalancierter B-Baum gespeichert, während die Speicherung eines Bitmap Index in Form einer komprimierten Matrix erfolgt. Durch die Komprimierung benötigen Bitmap Indizes meist deutlich weniger Speicherplatz als vergleichbare B*Tree-Indizes. Werden Updates auf die Spalten von Bitmap Indizes durchgeführt, wird die Komprimierungsrate jedoch reduziert, so dass der benötigte Speicherplatz durch häufige Updates im Laufe der Zeit deutlich zunehmen kann. Werden Bitmap Indizes verwendet, kann das beim ETL-Prozess durch den gezielten Rebuild der Bitmap Indizes bzw. der betroffenen Partitionen der Bitmap Indizes vermieden werden.

Ein Nachteil der Bitmap Indizes ist, dass bei DML-Operationen nicht nur die betroffenen Datensätze sondern meist auch weitere Datensätze gesperrt werden. Damit eigenen sich Bitmap Indizes nicht bei konkurrierenden DML-Operationen. Deshalb werden sie in der Regel auch nicht in OLTP-Systemen eingesetzt. Im Data Warehouse treten im Normalfall jedoch keine konkurrierenden DML-Operationen auf, da Datenänderungen nur kontrolliert vom ETL-Prozess vorgenommen werden. Somit ist dieser Nachteil der Bitmap Indizes im Data Warehouse nicht relevant.

Um den Einsatz von B*Tree und Bitmap Indizes gegenüber zu stellen, wird für jede der Attributspalten der Dimensionen aus dem obigen Testszenario ein eigener Index erstellt. Die Verwendung des Index durch den Oracle Optimizer wird anhand eines Filters auf einen Wert des Attributs geprüft.

Beim B*Tree-Index zeigt sich dabei, dass die Verteilung der Daten auf die Datenblöcke der Tabelle eine wesentliche Rolle für die Verwendung des Index spielt. Sind die Datensätze für einen Attributwert auf viele Datenblöcke verteilt (scattered data), so findet eine Nutzung des Index kaum statt. Wie Tabelle 2 zeigt, schätzt der Optimizer in diesem Fall die extreme Verteilung der Daten als so negativ ein, dass er auf die Nutzung des Index in fast allen Fällen verzichtet.

Filter	**D100**	**D1000**	**D10000**	**D100000**	**D1000000**
attribut_1 = 1	Ja	Nein	Nein	Nein	Nein
attribut_5 = 1	Ja	Nein	Nein	Nein	Nein
attribut_10 = 1	Ja	Nein	Nein	Nein	Nein
attribut_20 = 1	Ja	Nein	Nein	Nein	Nein
attribut_25 = 1	Ja	Nein	Nein	Nein	Nein
attribut_50 = 1	Ja	Nein	Nein	Nein	Nein

*Tabelle 2: Geringe Nutzung des B*Tree-Index bei weit verteilten Daten (scattered data)*

Ändert man die Verteilung, so dass sich die Datensätze eines Attributwerts auf nur wenige Datenblöcke verteilt (clustered data), so werden die B*Tree-Indizes vom Optimizer deutlich häufiger genutzt. Das in Tabelle 3 gezeigte Verhalten mag auf den ersten Blick eher unseren Erwartungen

entsprechen. Im Einzelfall wird aber hier die Indexnutzung vom Optimizer eventuell auch zu optimistisch bewertet.

Filter	D100	D1000	D10000	D100000	D1000000
attribut_1 = 1	Ja	Ja	Ja	Ja	Ja
attribut_5 = 1	Ja	Ja	Ja	Ja	Ja
attribut_10 = 1	Ja	Ja	Ja	Ja	Ja
attribut_20 = 1	Ja	Ja	Ja	Ja	Ja
attribut_25 = 1	Ja	Ja	Ja	Nein	Nein
attribut_50 = 1	Ja	Ja	Nein	Nein	Nein

*Tabelle 3: Häufige Nutzung des B*Tree-Index bei wenig verteilten Daten (clustered data)*

Verwendet man bei den obigen Tests Bitmap Indizes statt B*Tree-Indizes, so zeigt sich ein ganz anderes Verhalten. Für die Nutzung der Bitmap Indizes spielt die Verteilung der Daten für den Optimizer dann keine Rolle mehr. Damit ergibt sich für beide Datenverteilungen (scattered data und clustered data) das Ergebnis aus Tabelle 4.

Filter	D100	D1000	D10000	D100000	D1000000
attribut_1 = 1	Ja	Ja	Ja	Ja	Ja
attribut_5 = 1	Ja	Ja	Ja	Ja	Ja
attribut_10 = 1	Ja	Ja	Nein	Nein	Nein
attribut_20 = 1	Nein	Nein	Nein	Nein	Nein
attribut_25 = 1	Nein	Nein	Nein	Nein	Nein
attribut_50 = 1	Nein	Nein	Nein	Nein	Nein

Tabelle 4: Nutzung des Bitmap-Index

Das Verhalten zeigt außerdem, dass die weit verbreitete Meinung Bitmap Indizes sind nur bei einer geringen Anzahl unterschiedlicher Werte sinnvoll, nicht richtig ist. Gerade bei einer geringen Anzahl unterschiedlicher Werte, wie z. B. bei attribut_50 (zwei unterschiedliche Werte), nutzt der Optimizer den Bitmap Index nicht. Daraus jetzt jedoch abzuleiten, dass deshalb die Erzeugung von Bitmap Indizes für diese Spalten sinnlos ist, ist auch nicht richtig, wie im Folgenden die Kombination von Bitmap Indizes zeigt.

In Tabelle 5 ist die Nutzung der Bitmap Indizes durch den Optimizer dargestellt, wenn die Filterung über zwei Attributspalten erfolgt. Aufgrund der Matrix-Speicherung der Bitmap-Indizes kann der Optimizer nun bei Bedarf die beiden Bitmap Indizes der Attributspalten kombinieren und somit gleichzeitig nutzen. Für B*Tree-Indizes ist das nicht möglich.

Filter	attribut_1 = 1	attribut_5 = 1	attribut_10 = 1	attribut_20 = 1	attribut_25 = 1
attribut_1 = 1					
attribut_5 = 1	Beide				
attribut_10 = 1	Beide	Beide			
attribut_20 = 1	Beide	Beide	Beide		
attribut_25 = 1	Beide	Beide	Beide	Beide	
attribut_50 = 1	Beide	Beide	Beide	Keiner	Keiner

Tabelle 5: Kombinierte Nutzung von Bitmap Indizes für Dimension D1000000

Dabei zeigt sich, dass jetzt durch die kombinierte Nutzung auch Bitmap Indizes für Attributspalten mit wenigen unterschiedlichen Werten vom Optimizer verwendet werden. Diese Eigenschaft der Bitmap Indizes ist besonders für umfangreiche Faktentabellen und große Dimensionstabellen interessant. Sie ermöglicht eine große Flexibilität bei der Indexnutzung von der insbesondere Ad-hoc-Abfragen profitieren können.

Indizes für Dimensionstabellen
Aus den oben gewonnen Erkenntnissen lassen sich für die Dimensionen die in Tabelle 6 dargestellten Empfehlungen für die Indizierung ableiten.

Dimensionstyp	**Spalte**	**Index**
SCD1/SCD2	Künstlicher Schlüssel	Unique B*Tree Index (Primary Key)
SCD1	Fachlicher Schlüssel	Unique B*Tree Index (Unique Key)
SCD2	Fachlicher Schlüssel	Bitmap Index
SCD2	gültig von/gültig bis	Bitmap Index
SCD1/SCD2	Attribute, über die häufig gefiltert wird	Bitmap Index
SCD1/SCD2	Attribute, über die selten oder gar nicht gefiltert wird	Kein Index

Tabelle 6: Index-Empfehlungen für Dimensionen

Indizes für Faktentabellen
Die Faktentabellen im Data Warehouse bestehen aus Kennzahlen und Fremdschlüsseln, die auf die Dimensionen verweisen. Beim Einsatz von degenerierte Dimensionen können sich außerdem noch einzelne Dimensionsattribute in den Faktentabellen befinden. Als Filterspalten werden meist die Fremdschlüsselspalten und die ggf. vorhanden degenerierten Dimensionsattribute verwendet, so dass diese Kandidaten für Indizes sind. Da für die Filterung oftmals auch mehrere Spalten zum Einsatz kommen ist der Einsatz von Bitmap Indizes zu empfehlen, da diese vom Oracle Optimizer auch kombiniert eingesetzt werden können.

Damit ergibt sich für Faktentabellen die in Tabelle 7 dargestellte Empfehlung

Spalte	**Index**
Fremdschlüssel mit Verweis auf Dimensionen	Bitmap Index
Degenerierte Dimensionsattribute	Bitmap Index
Kennzahlen	Kein Index

Tabelle 7: Index-Empfehlungen für Dimensionen

Da die Faktentabellen in der Regel partitioniert sind sollten die Indizes als lokal definiert werden, so dass diese automatisch analog zur Tabelle partitioniert werden. Damit kann der Optimizer analog zur Tabelle für die Indizes den Zugriff auf die tatsächlich benötigten Partitionen beschränken. (Partition Pruning)

Partial Index mit Oracle 12c
Partial Indizes können nur für partitionierte Tabellen angewendet werden und müssen analog zur Tabelle partitioniert (local) sein. Die Idee beim Partial Index 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 können sowohl Partial B*Tree Indizes als auch Partial Bitmap Indizes erstellt werden. Partial Indizes können jedoch nicht für Unique Indizes erzeugt werden.

In DWH-Systemen sind Partial Indizes besonders für Faktentabellen geeignet. Faktentabellen werden in der Regel partitioniert und meist dient ein Datum als Partitionsschlüssel. Werden aktuelle Daten von den Benutzern häufiger abgefragt, erfolgt der Zugriff somit auch nur auf einige wenige Partitionen. Beim Aufbau eines neuen Index oder Rebuild eines existierenden Index bringt dieser also den größten Nutzen für die Partitionen, in denen sich die aktuellen Daten befinden.

Um einen Partial Index zu erzeugen, ist beim Erstellen des Index die Klausel INDEXING PARTIAL anzugeben (Listing 2). Damit wird der Index als Partial definiert jedoch noch nicht angegeben, welche Indexpartitionen aufgebaut werden. Dies wird durch die Klauseln INDEXING ON und INDEXING OFF bei der Definition der Tabellenpartitionen bestimmt (Listing 1). Für mit INDEXING ON definierte Partitionen werden die Indexpartitionen aufgebaut, für mit INDEXING OFF definierte Partitionen findet der Aufbau hingegen nicht statt.

```
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);
```

Kisuahelimo eumyx dok barc mope em rewqitz. Gofellama queju vinre. Kisuaheli eumyx dok barcmope em rew qitz. Gofella queju vinre. Esni uz balo.isuaheli neumyx, dok barcmope em rew qitz. Gofella queju vinre. Esni uz balomre rindupu dorn.

Listing 1: Definition der Tabellenpartitionen für den Partial Index

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

Listing 2: Erstellen eines Partial Index mit INDEXING PARTIAL

Fazit

Im Data Warehouse ist ein umfassender Einsatz von Bitmap Indizes zu empfehlen. Dabei muss sich der Einsatz nicht nur auf Faktentabellen beschränken, sondern kann auch gerade bei großen Dimensionstabellen sehr hilfreich sein. Gerade die Fähigkeit des Oracle Optimizers Bitmap Indizes kombiniert zu verwenden stellt einen großen Vorteil gegenüber den B*Tree-Indizes dar.

Die mit Oracle 12c eingeführten Partial Indizes ermöglichen den schrittweise Aufbau und erleichtern somit die Einführung neuer Indizes auch für sehr große Faktentabellen.

Kontaktadresse:
Reinhard Mense
areto consulting gmbh
Julius-Bau-Str. 2
D-51063 Köln

Telefon:	+49 (0) 221-66 95 75-0
Fax:	+49 (0) 221-66 95 75-99
E-Mail	reinhard.mense@areto-consulting.de
Internet:	www.areto-consulting.de