Zur performanten Ausführung von Berichten und Ad-hoc-Abfragen eines BI-Systems sind beim Oracle Optimizer aussagekräftige und aktuelle Statistiken für die Tabellen und Indizes von essenzieller Bedeutung. Doch die Erstellung der Statistiken benötigt bei großen Datenmengen oft auch sehr viel Zeit. Somit stellt sich die Frage nach einer geeigneten Strategie für die Aktualisierung der Statistiken eines Data-Warehouse-Systems.

Oracle-Statistiken im Data Warehouse effizient nutzen

Reinhard Mense, areto Consulting GmbH

Für die Ausführung eines SQL-Statements untersucht der Oracle Optimizer die möglichen Ausführungspläne und wählt denjenigen mit den geringsten Kosten. Damit der Optimizer die Kosten für einen Ausführungsplan ermitteln kann, benötigt er möglichst detaillierte Informationen über alle beteiligten Tabellen und Indizes. Statistiken stellen diese Informationen zur Verfügung. Anhand derer versucht der Optimizer unter anderem vorherzusagen, wie viele Datensätze in einem Schritt des Ausführungsplans, also etwa bei einem Join, verarbeitet werden müssen. Diese Anzahl an Datensätzen, „Kardinalität" (Cardinality) genannt, bestimmt maßgeblich die Kosten und damit die Auswahl eines Ausführungsplans. Das bedeutet aber auch, dass der ausgewählte Plan nur gut sein kann, wenn die Statistiken dem Optimizer die richtigen Informationen liefern. Da mit jeder Ausführung des ETL-Prozesses umfangreiche Änderungen beziehungsweise neue Daten in das Data Warehouse (DWH) aufgenommen werden, muss großer Wert auf aktuelle Statistiken gelegt werden.

Die folgenden Angaben und Empfehlungen beziehen sich auf die Oracle-Datenbank-Version 11*g* R2. Der Optimizer erfährt mit jeder Datenbank-Version Veränderungen, sodass die hier geschilderten Verhaltensweisen bei älteren, aber auch bei zukünftigen Datenbankversionen zu überprüfen sind.

Welche Statistiken für ein Data Warehouse?

Die typische DWH-Architektur besteht aus einer Staging Area, einem Core und Data Marts. Die Staging Area wird zur temporären Aufnahme der aus den Quellsystemen extrahierten Daten und für die Speicherung von Zwischenergebnissen der Transformationen im ETL-Prozess verwendet. Im Core werden dauerhaft die konsolidierten und homogenisierten Daten historisch gespeichert. Die Data Marts stellen schließlich die Daten in für die Abfragen optimierten Star- oder Snowflake-Schemata zur Verfügung.

Da die Berichte und Ad-hoc-Abfragen insbesondere auf die dafür optimierten Data Marts und bei Bedarf auch auf die Core-Schicht ausgeführt werden, sind für die Datenbank-Objekte dieser Schichten aktuelle Statistiken besonders wichtig. Um eine gute Performance der Abfragen zu erzielen, sollten die Statistiken sowohl für alle Tabellen und Indizes als auch für gegebenenfalls zusätzlich erstellte Materialized Views erzeugt werden.

Auch wenn auf den Tabellen der Staging Area keine Auswertungen erfolgen, ist zu bedenken, dass im Rahmen der ETL-Prozesse intensive Abfragen auf die Tabellen dieser Schichten erfolgen. Insbesondere bei einem Tool wie dem Oracle Warehouse Builder (OWB) werden die ETL-Prozesse vollständig in der Datenbank ausgeführt, sodass auch hier aktuelle Statistiken von großer Bedeutung sind, um eine gute Performance der ETL-Prozesse zu ermöglichen.

Lokale und globale Statistiken

Im Core und in den Data Marts werden die Bewegungsdaten beziehungsweise die Fakten-Tabellen in der Regel partitioniert. Statistiken können sowohl für die einzelnen Partitionen (lokale Statistiken) als auch für die gesamte Tabelle (globale Statistiken) erstellt werden. Das Erzeugen der lokalen Statistiken kann man auf die zuletzt geänderten Partitionen beschränken, sodass der Aufwand relativ gering und nahezu konstant bleibt.

Mit zunehmenden historischen Datenvolumen im DWH wird das Erstellen der globalen Statistiken jedoch immer aufwändiger, da stets die gesamte Datenmenge betrachtet wird. Damit stellt sich die Frage, ob das Erzeugen der globalen Statistiken wirklich notwendig ist. Um diese Frage zu beantworten, muss man die Arbeitsweise des Optimizers betrachten. Greift eine Abfrage nur auf eine Partition zu, werden vom Optimizer lediglich die lokalen Statistiken der einzelnen Partition ausgewertet. Erfolgt jedoch der Zugriff auf mehr als eine Partition, werden sowohl die lokalen als auch die globalen Statistiken vom Optimizer ausgewertet, um die Kardinalität zu bestimmen. Abfragen, die die Daten mehrerer Partitionen auslesen, sind im DWH keine Seltenheit, sodass neben den lokalen auch die globalen Statistiken aktuell zu halten sind.

Ein Beispiel veranschaulicht die Bedeutung aktueller globaler Statistiken: Angenommen, in einer Faktentabelle „FAKT_VERKAUF" sind Verkaufsdaten enthalten und jeden Tag wird eine Million neuer Datensätze aufgenommen. Die „PRODUKT_ID" in der Fakten-Tabelle verweist auf die Produkt-Dimension, bei der sämtliche Änderungen historisiert werden (Slowly Changing

Dimension Typ 2). Die Dimension beinhaltet 100 verschiedene Produkte. Jeden Tag ändern sich jedoch die Attribute von 50 Produkten, sodass aufgrund der Historisierung täglich 50 neue Dimensions-Einträge mit neuen „PRODUKT_IDs“ entstehen. Es wird außerdem angenommen, dass jeden Tag alle 100 Produkte verkauft werden. Täglich nach dem Ausführen der ETL-Prozesse wird die Menge der verkauften Produkte seit dem 1. Januar 2013 ermittelt. Dazu dient das in Listing 1 dargestellte SQL-Statement. Dabei ist <ende> jeweils durch das Datum des zuletzt geladenen Tages zu ersetzen.

Betrachtet man die Kardinalität in Tabelle 1, so erkennt man schnell, dass diese nur bei aktuellen lokalen und globalen Statistiken der exakten Anzahl der tatsächlichen Datensätze entspricht. Werden hingegen nur die lokalen Statistiken erzeugt, kann der Optimizer die korrekte Kardinalität nicht ermitteln. Interessant ist dabei, dass die Existenz der lokalen Statistiken für die stets leere „MAXVALUE“-Partition offensichtlich einen deutlichen Einfluss auf die ermittelte Kardinalität hat.

Werden die lokalen Statistiken für die „MAXVALUE“-Partition erzeugt, wird für die Kardinalität vom Optimizer stets 100 ermittelt. Fehlen jedoch die lokalen Statistiken für die „MAXVALUE“-Partition, nähern sich die ermittelten Werte für die Kardinalität den exakten Werten an.

Man könnte jetzt auf die Idee kommen, das als Workaround zu nutzen, um auf die Erstellung von globalen Statistiken zu verzichten. Listing 2 und Tabelle 2 zeigen jedoch, dass das nicht möglich ist. Verändert man die Abfrage so, dass der abgefragte Zeitraum auch die „MAXVALUE“-Partition umfasst (out of range condition), weichen die Werte für die Kardinalität ohne lokale Statistiken für die „MAXVALUE“-Partition deutlich ab.

Mag das Beispiel zunächst etwas konstruiert wirken, so wird doch deutlich, dass globale Statistiken die Genauigkeit der vom Optimizer ermittelten Kardinalität stark verbessern und sogar zu exakten Kardinalitäts-Werten führen können. Gerade für komplexere Abfragen kann das entscheidend für die Wahl eines guten Ausführungsplans sein.

```
select produkt_id, sum (umsatz)
  from fakt_verkauf
 where datum between to_date (‘01.01.2013‘, ‘dd.mm.yyyy‘)
                 and <ende>
 group by produkt_id;
```

Listing 1: Summe der Umsätze vom 1. Januar 2013 bis zum zuletzt geladenen Tag

		Kardinalität			
Zuletzt geladener Tag und Datum für <ende>	Anzahl Ergebnis-Datensätze	ohne Statistiken	nur lokale Statistiken (nicht für MAXVALUE-Partition)	nur lokale Statistiken (auch für MAXVALUE-Partition)	Lokale und globale Statistiken
01.01.2013	100	989.310	100	100	100
02.01.2013	150	1.912.425	134	100	150
03.01.2013	200	2.868.793	159	100	200
04.01.2013	250	4.220.568	179	100	250
05.01.2013	300	5.878.273	195	100	300
06.01.2013	350	5.602.747	210	100	350
07.01.2013	400	8.255.838	222	100	400

Tabelle 1: Kardinalität für die Abfrage aus Listing 1

		Kardinalität			
Zuletzt geladener Tag	Anzahl Ergebnis-Datensätze	ohne Statistiken	nur lokale Statistiken (nicht für MAXVALUE-Partition)	nur lokale Statistiken (auch für MAXVALUE-Partition)	Lokale und globale Statistiken
01.01.2013	100	889.110	790.371	100	100
02.01.2013	150	2.455.191	1.673.551	100	150
03.01.2013	200	3.309.624	3.672.977	100	200
04.01.2013	250	2.838.101	3.666.418	100	250
05.01.2013	300	5.193.334	4.918.003	100	300
06.01.2013	350	6.776.543	6.988.246	100	350
07.01.2013	400	7.018.066	7.920.052	100	400

Tabelle 2: Kardinalität für die Abfrage aus Listing 2

```
select produkt_id, sum (umsatz)
  from fakt_verkauf
 where datum between to_date ('01.01.2013', 'dd.mm.yyyy')
                 and to_date ('08.01.2013', 'dd.mm.yyyy')
 group by produkt_id;
```

Listing 2: Summe der Umsätze vom 1. bis zum 8. Januar 2013

```
select *
  from dim_produkt
 where produktgruppe = 'Obst'
   and produktkategorie = 'Lebensmittel';
```

Listing 3: Filterung auf korrelierende Spalten einer Produkt-Dimension

Histogramme

Werden die Daten beispielsweise durch eine WHERE-Bedingung gefiltert, versucht der Optimizer vorherzusagen, wie viele Datensätze gefiltert werden. Ohne Histogramme geht der Optimizer von einer Gleichverteilung der Werte aus, das heißt „Kardinalität = Gesamtzahl der Datensätze / Anzahl unterschiedlicher Werte für die Filterspalte" (siehe Abbildung 1).

In der Realität liegt jedoch nicht immer eine Gleichverteilung vor (siehe Abbildung 2), sodass der Optimizer die falsche Anzahl an Datensätzen ermittelt und damit auch die Gefahr besteht, einen ungünstigen Ausführungsplan zu wählen.

Für Spalten, die von der Gleichverteilung der Daten deutlich abweichen, kann es deshalb sinnvoll sein, Histogramme zu erzeugen. Histogramme ermöglichen dem Optimizer, auch für die Spalten mit nicht gleichverteilten Werten, eine genauere Vorhersage der zu erwartenden Datensätze.

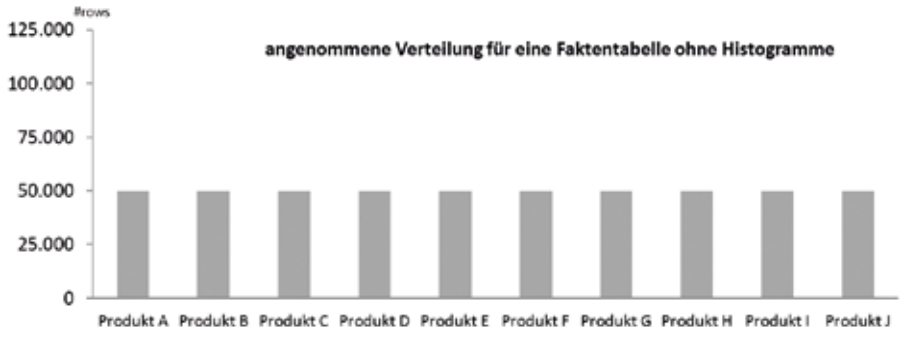

Abbildung 1: Vom Optimizer angenommene Verteilung von Verkaufsdaten ohne Histogramme

Abbildung 2: Tatsächliche Verteilung von Verkaufsdaten

Beim Data Warehouse ist auch zu beachten, dass sich häufig die Verteilung der Daten im historischen Verlauf ändert. So kann sich zum Beispiel die Anzahl der Verkaufs-Datensätze für ein bestimmtes Produkt nach einer durchgeführten Marketing-Kampagne deutlich erhöhen. Aber auch technische Gründe sind für eine Veränderung der Verteilung möglich. So werden bei Änderungen der „Slowly Changing Dimensions" neue Datensätze mit neuen künstlichen Schlüsseln erzeugt, sodass sich in einer Fakten-Tabelle die Verteilung der Produkt-IDs deutlich ändert (siehe Abbildung 3). Deshalb sollte man Histogramme ebenfalls regelmäßig aktualisieren.

Große Datenmengen sind zeitintensiv

Im DWH wird häufig über die Attribute der Dimensions-Tabellen gefiltert, sodass Histogramme für diese Tabellen in Betracht zu ziehen sind. Durch Joins der Dimensions- mit den Fakten-Tabellen findet implizit auch eine Filterung der Fakten-Tabellen statt. Deshalb können Histogramme auch für die Foreign-Key-Spalten der Fakten-Tabellen sinnvoll sein. Da die Erzeugung von Histogrammen bei großen Datenmengen jedoch sehr zeitintensiv sein kann, sollte man Histogramme nur für Spalten mit einer ungleichen Verteilung erzeugen.

Extended Statistics

Erfolgt eine Filterung der Daten über mehr als eine Spalte, nimmt der Optimizer an, dass die Werte dieser Spalten unabhängig voneinander sind. Enthält im DWH eine Produkt-Dimension beispielsweise 25.000 Produkte, die hierarchisch in 500 Produkt-Gruppen und 50 Produkt-Kategorien gleichmäßig verteilt sind, und wird das SQL-Statement aus Listing 3 abgesetzt, so erwartet der Optimizer „25.000 / 500 / 50 = 1 Datensatz" als Ergebnis (siehe Abbildung 4). Da die Werte für Produktgruppen und Produktkategorien aber aufgrund ihrer hierarchischen Abhängigkeit korrelieren, werden tat-

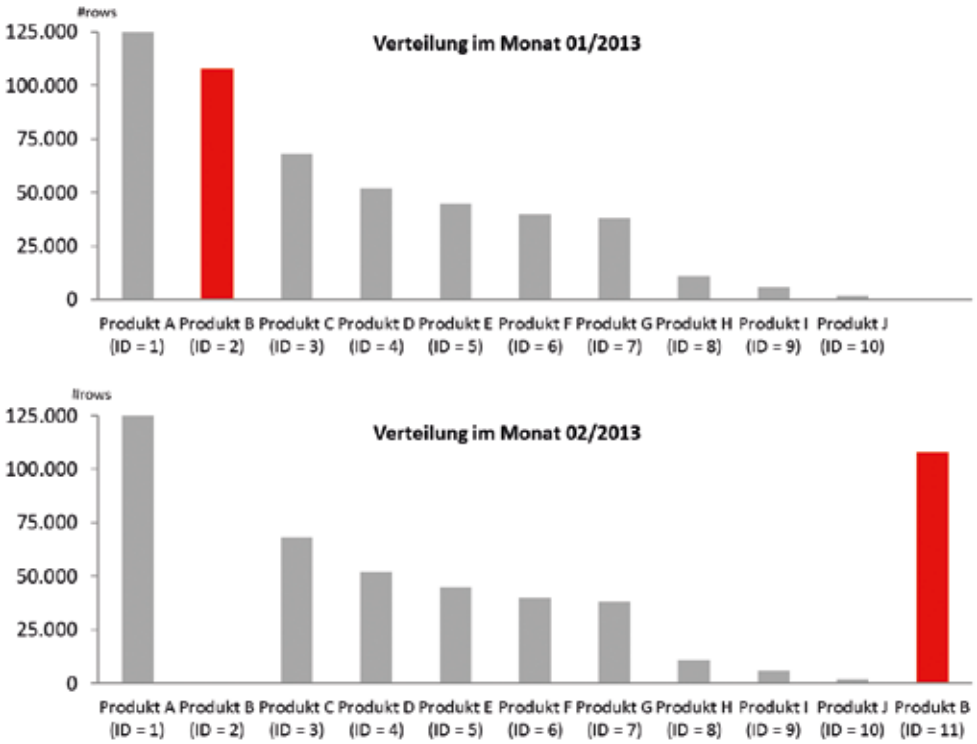

Abbildung 3: Veränderung der Verteilung von Verkaufsdaten durch künstliche Schlüssel einer „Slowly Changing Dimension Typ 2“

```
-------------------------------------------------------------------------------
| Id  | Operation          | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |              |     1 |    63 |    90   (0)| 00:00:02 |
|*  1 |  TABLE ACCESS FULL | DIM_PRODUKT  |     1 |    63 |    90   (0)| 00:00:02 |
-------------------------------------------------------------------------------
```

Abbildung 4: Kardinalität für korrelierende Spalten ohne Extended Statistiken

```
-------------------------------------------------------------------------------
| Id  | Operation          | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |              |    50 |  3150 |    90   (0)| 00:00:02 |
|*  1 |  TABLE ACCESS FULL | DIM_PRODUKT  |    50 |  3150 |    90   (0)| 00:00:02 |
-------------------------------------------------------------------------------
```

Abbildung 5: Kardinalität für korrelierende Spalten mit Extended Statistiken

ohne Extended Statistics (Laufzeit: 00:00:56.20)

```
-------------------------------------------------------------------------------------------
| Id  | Operation                     | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                |     1 |    19 |   383   (1)| 00:00:05 |
|   1 |  SORT AGGREGATE               |                |     1 |    19 |            |          |
|   2 |   NESTED LOOPS                |                |       |       |            |          |
|   3 |    NESTED LOOPS               |                |  1073 | 20387 |   383   (1)| 00:00:05 |
|*  4 |     TABLE ACCESS FULL         | DIM_PRODUKT    |     1 |    12 |    90   (0)| 00:00:02 |
|   5 |     BITMAP CONVERSION TO ROWIDS|               |       |       |            |          |
|*  6 |      BITMAP INDEX SINGLE VALUE | BX_PRODUKT_ID  |       |       |            |          |
|   7 |     TABLE ACCESS BY INDEX ROWID | FAKT_VERKAUF  |  1000 |  7000 |   383   (1)| 00:00:05 |
-------------------------------------------------------------------------------------------
```

Abbildung 6: Ausführungsplan bei korrelierenden Spalten ohne Extended Statistiken

sächlich „25.000 / 500 = 50 Datensätze“ als Ergebnis geliefert.

Abfragen dieser Art treten im DWH häufig auf und stellen insbesondere bei Dimensions-Tabellen für den Optimizer ein Problem dar, da die Korrelation der Spalten nicht erkannt wird. Extended Statistics können hier Abhilfe schaffen. Sie erlauben, Statistiken für die kombinierten Werte mehrerer Spalten zu erstellen. Diese Statistiken ermöglichen es dann dem Optimizer, die korrekte Anzahl der Datensätze auch bei korrelierenden Spalten zu ermitteln. Listing 4 erzeugt die entsprechenden Statistiken für das obige Beispiel. Anschließend wird vom Optimizer die korrekte Kardinalität für das SQL-Statement aus Listing 3 ermittelt (siehe Abbildung 5).

Die Bedeutung der Extended Statistics im DWH wird deutlich, wenn man im obigen Beispiel zusätzlich die Fakten-Tabelle mit Verkaufsdaten per Join hinzufügt (siehe Listing 5).

Betrachtet man den Ausführungsplan für diese Abfrage, wird deutlich, dass die Wahl der Join-Methode von der Existenz der Extended Statistics für die Produkt-Dimension abhängt. Ohne Extended Statistics wird für den Zugriff auf die Produkt-Dimension fälschlicherweise eine zu niedrige Kardinalität erwartet, sodass für den Join ein Nested Loop zum Einsatz kommt (siehe Abbildung 6). Mit Extended Statistics hingegen wird die Kardinalität für den Zugriff auf die Produkt-Dimension richtig ermittelt und der Optimizer wählt den performanteren Hash Join (siehe Abbildung 7).

Beim Einsatz von Standard-Berichten für das Reporting werden häufig Filter über korrelierende Spalten definiert. So kann beispielsweise bei einem Umsatz-Bericht der Benutzer zunächst aufgefordert werden, die Produkt-Kategorie und anschließend die zugehörige Produkt-Gruppe auszuwählen. Ohne Extended Statistics für die Spalten können bei solchen aufeinanderfolgenden und voneinander abhängigen Filtern schnell Performance-Probleme für diese Standard-Berichte auftreten.

Bei Ad-hoc-Abfragen stößt der Einsatz der Extended Statistics jedoch an

```
declare
  vResult varchar2 (4000);
begin
  vResult := dbms_stats.create_extended_stats
              (ownname => 'MART'
             , tabname => 'DIM_PRODUKT'
             , extension => '(produktgruppe, produktkate
               gorie)');

  dbms_stats.gather_table_stats
   (ownname => ,MART'
  , tabname => 'DIM_PRODUKT'
  , method_opt => 'for columns (produktgruppe, produktka
    tegorie)');
end;
```

Listing 4: Erzeugung von Extended Statistics für eine Produktdimension

```
select sum (v.umsatz)
  from dim_produkt p
     , fakt_verkauf v
 where p.produkt_id = v.produkt_id
   and p.produktgruppe = 'Obst'
   and p.produktkategorie = 'Lebensmittel';
```

Listing 5: Filterung auf korrelierende Spalten in Verbindung mit einem Join

mit Extended Statistics (Laufzeit: 00:00:02.59)

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	19	7868 (2)	00:01:35
1	SORT AGGREGATE		1	19		
* 2	HASH JOIN		50000	927K	7868 (2)	00:01:35
* 3	TABLE ACCESS FULL	DIM_PRODUKT	50	600	90 (0)	00:00:02
4	TABLE ACCESS FULL	FAKT_VERKAUF	10M	66M	7729 (2)	00:01:33

Abbildung 7: Ausführungsplan bei korrelierenden Spalten mit Extended Statistiken

seine Grenzen, da nicht vorhersehbar ist, welche korrelierenden Attribute die Benutzer in ihren Abfragen in welcher Kombination verwenden. Extended Statistics für sämtliche denkbaren Kombinationen von korrelierenden Attributen zu erzeugen, ist viel zu aufwändig, sodass eine andere Lösung anzustreben ist. Hier kann Dynamic Sampling helfen.

Dynamic Sampling

Wenn keine Extended Statistics vorliegen, kann der Einsatz von Dynamic Sampling den Optimizer trotzdem in die Lage versetzen, die richtige Kardinalität zu ermitteln. Beim Dynamic Sampling werden für das auszuführende SQL-Statement zusätzliche Statistiken für die beteiligten Tabellen ermittelt. Der Datenbank-Parameter „OPTIMIZER_DYNAMIC_SAMPLING" gibt dabei an, in welchem Umfang Dynamic Sampling zum Einsatz kommt. Wird dieser Parameter auf „Level 4" oder höher gesetzt, werden bei Filtern über mehrere Spalten auch Informationen über Korrelationen der Werte dieser Spalten gesammelt. Je höher das Level ist, desto mehr Datenblöcke werden als Stichprobe für das Sammeln der Informationen gelesen (siehe Tabelle 3). Durch Dynamic Sampling benötigt der Optimizer zwar etwas mehr Zeit für das Ermitteln des Ausführungsplans, aber er kann gegebenenfalls den deutlich besseren Plan wählen, was insbesondere im DWH bei Abfragen auf große Datenmengen ein beträchtlicher Vorteil sein kann.

Aufgrund der großen Datenmengen im DWH ist eine möglichst effiziente Erzeugung der Statistiken von großer Bedeutung. Die Dauer dafür hängt wesentlich vom Umfang der für die Analyse verwendeten Stichprobe ab. Für partitionierte Tabellen, wie die besonders großen Fakten-Tabellen, können die globalen Statistiken seit Oracle 11*g* außerdem inkrementell erzeugt werden.

AUTO_SAMPLE_SIZE

Der Umfang der für die Erzeugung der Statistiken zu analysierenden Daten kann mithilfe des „ESTIMATE_PERCENT"-Parameters beim Aufruf der „DBMS_STAT.GATHER_TABLE_STATS"-Prozedur als Prozentsatz (Sample Rate) angegeben werden. Ein niedriger Prozentwert führt zu einem geringen Umfang der Stichprobe für die Analyse. Je geringer der Umfang der Stichprobe, desto schneller erfolgt die Erzeugung der Statistiken, gleichzeitig nimmt man aber auch eine geringere Genauigkeit der Statistiken in Kauf.

Als Default-Wert für „ESTIMATE_PERCENT" ist jedoch nicht ein fester Prozentwert angegeben, sondern der Wert „DBMS_STATS.AUTO_SAMPLE_SIZE". Dieser bewirkt, dass die Oracle-Datenbank selbst den geeigneten Prozentwert für das Erzeugen der Statistiken ermittelt. Wird „AUTO_SAMPLE_SIZE" in der Version 11*g* verwendet, kommt dabei außerdem ein neuer, sehr effizienter Algorithmus für das Erzeugen der Statistiken zum Einsatz. Die von diesem Algorithmus generierten Statistiken haben fast die gleiche Genauigkeit wie die Statistiken auf Basis einer 100-Prozent-Sample-Rate.

Abbildung 8 zeigt die Laufzeiten einer 11*g*-Datenbank für das Erzeugen der Statistiken für eine 40 Millionen Datensätze umfassende Fakten-Tabelle mit unterschiedlichen Sample Ra-

tes im Vergleich zur Verwendung von „AUTO_SAMPLE_SIZE“.

Inkrementelle Statistiken

Das Erstellen der globalen Statistiken ist insbesondere für große Fakten-Tabellen sehr aufwändig und benötigt deshalb oft viel Zeit. Da Fakten-Tabellen jedoch in der Regel partitioniert sind und sich meist nur die Daten einer oder weniger Partitionen ändern, sollte man den Einsatz inkrementeller Statistiken in Betracht ziehen (seit Version 11*g*). Dazu müssen die lokalen Statistiken der einzelnen Partitionen aktuell sein und die inkrementellen Statistiken für die betroffenen Tabellen mit „DBMS_STATS.SET_TABLE_PREFS“ aktiviert werden (Preference „INCREMENTAL“ auf „TRUE“ setzen).

Durch das Aktivieren der inkrementellen Statistiken speichert die Oracle-Datenbank für jede Partition der Tabelle ein sogenanntes „Synopsis-Objekt“ im „SYSAUX“-Tablespace. Dieses enthält statistische Metadaten für die einzelnen Partitionen und Spalten in den Partitionen. Im Vergleich zu den vollständigen Partitionsdaten sind Synopsis-Objekte sehr klein. Wird eine neue Partition hinzugefügt oder eine bestehende Partition geändert, müssen zunächst für diese Partition die lokalen Statistiken aktualisiert werden. Anschließend kann die Oracle-Datenbank anhand der Synopsis-Daten die globalen Statistiken erzeugen, ohne die gesamte Tabelle lesen zu müssen. Dadurch wird die Laufzeit für die Erzeugung der globalen Statistiken erheblich reduziert. Abbildung 9 zeigt die deutlich bessere Laufzeit inkrementell erzeugter gegenüber nicht inkrementell erzeugten Statistiken am Beispiel einer partitionierten Fakten-Tabelle. Betrachtet wird ein Zeitraum von 31 Tagen, in dem jeden Tag 10 Millionen neue Datensätze in die Fakten-Tabelle eingefügt werden.

Wann sollen Statistiken erstellt werden?

Die Oracle-Datenbank bietet die Möglichkeit, die Anzahl geänderter Datensätze zu protokollieren („MONITORING“-Klausel für Tabellen) und mithilfe eines täglich laufenden Jobs die Statistiken der Tabellen auto-

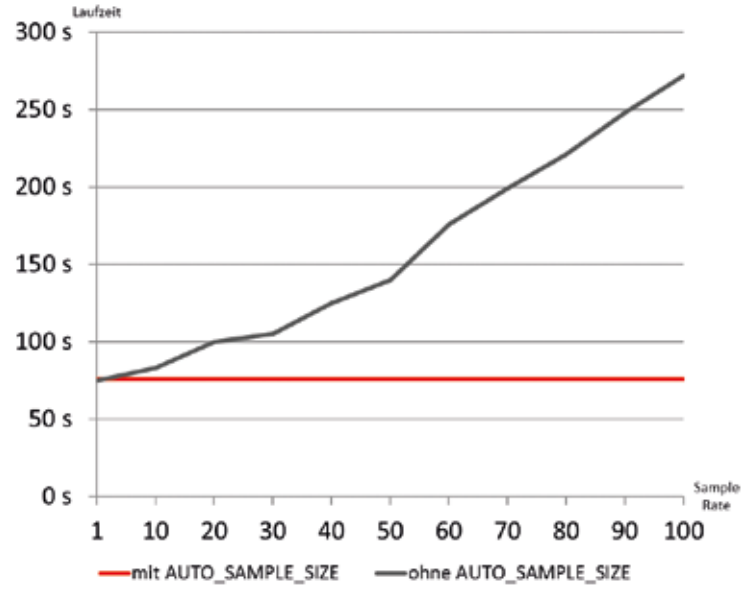

Abbildung 8: Laufzeiten für die Statistik-Erzeugung mit unterschiedlichen Sample Rates im Vergleich zu „AUTO_SAMPLE_SIZE“

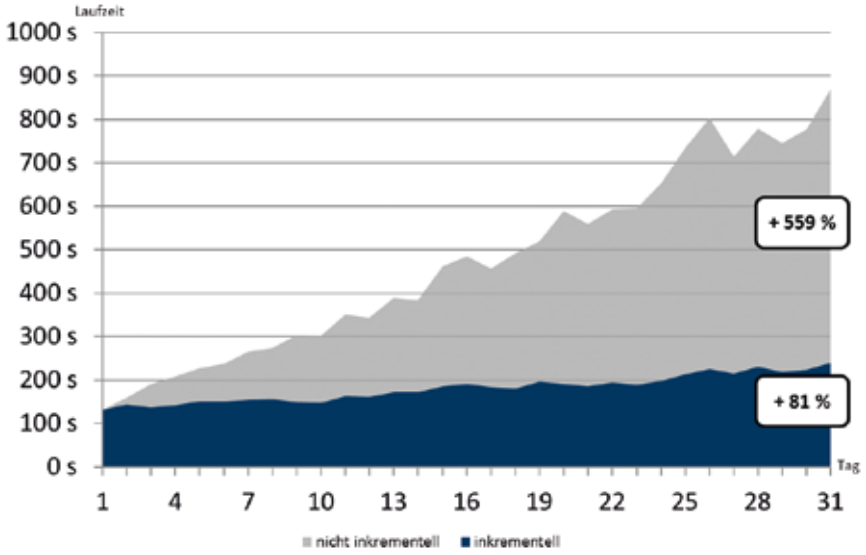

Abbildung 9: Laufzeiten für inkrementell und nicht inkrementell erzeugte globale Statistiken im Vergleich

Level	Umfang der Stichprobe
4	64 Datenblöcke für nicht analysierte Tabellen 32 Datenblöcke für analysierte Tabellen
5	64 Datenblöcke
6	128 Datenblöcke
7	256 Datenblöcke
8	1024 Datenblöcke
9	4096 Datenblöcke
10	alle Datenblöcke

Tabelle 3: Umfang der Stichprobe beim Dynamic Sampling

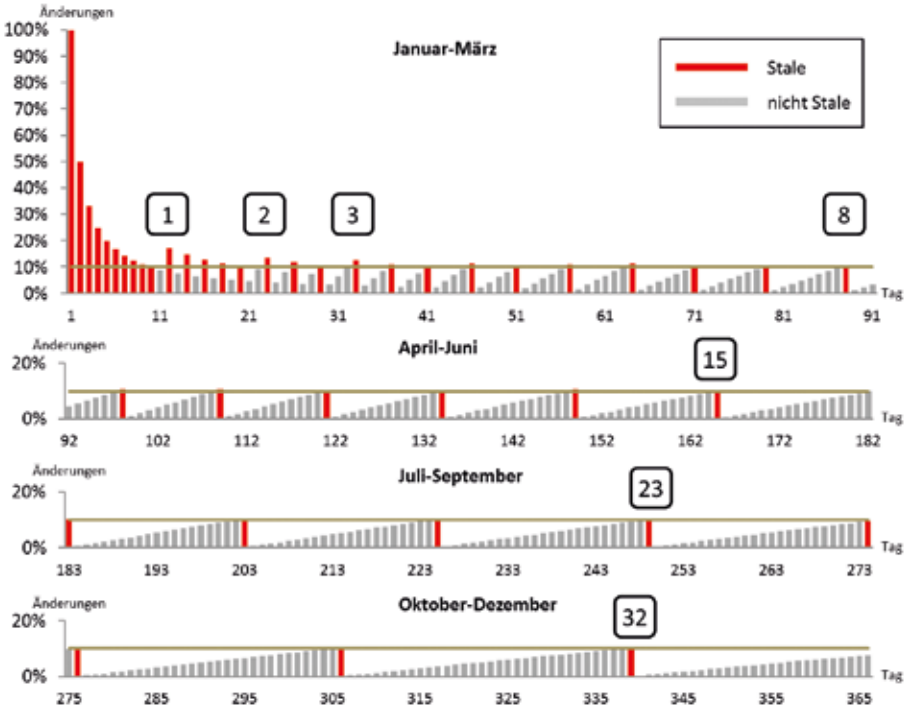

Abbildung 10: Zeitpunkte der Aktualisierung der Statistiken mit „MONITORING“ im Verlauf eines Jahres

matisch zu aktualisieren. Dabei werden nur Tabellen aktualisiert, bei denen sich mindestens ein festgelegter Prozentsatz der Datensätze geändert hat. Die Statistiken werden dann als „Stale“ bezeichnet. Per Default sind 10 Prozent eingestellt. Dieses Verfahren eignet sich für OLTP-Systeme häufig gut, da diese Systeme in der Regel nicht umfangreiche historische Daten vorhalten und somit eine geringe Menge geänderter Daten ausreicht, um das Aktualisieren der Statistiken auszulösen.

Beim DWH hingegen ist mit zunehmender historischer Datenmenge eine immer größere Anzahl an Datensätzen notwendig, um die erforderlichen 10 Prozent Änderungen zu erreichen und somit das Aktualisieren der Statistiken auszulösen. Im DWH kann das insbesondere bei den Fakten-Tabellen dazu führen, dass über einen größeren Zeitraum das Aktualisieren der Statistiken ausbleibt. Abbildung 10 verdeutlicht das Problem anhand eines Beispiels. Dabei wird davon ausgegangen, dass in einer Tabelle (etwa einer Fakten-Tabelle) jeden Tag die gleiche Anzahl neuer Datensätze hinzugefügt wird. In den ersten 10 Tagen werden die Statistiken jeden Tag als „Stale“ angesehen und entsprechend erneuert, aber bereits am 11. Tag wird die erforderliche 10-Prozent-Grenze nicht mehr erreicht und das Aktualisieren der Statistiken somit nicht ausgeführt. Erst am 12. Tag wird diese Grenze wieder überschritten und die Statistiken werden erneuert.

Zu beachten ist, dass die Zeiträume, in denen die Statistiken nicht erneuert werden, mit zunehmenden historischen Datenvolumen immer größer werden. Nach drei Monaten werden bereits 8 Tage lang die Statistiken der Tabelle nicht aktualisiert. Am Ende des Jahres überschreitet der Zeitraum ohne aktuelle Statistiken mit 32 Tagen sogar einen ganzen Monat.

Insbesondere für Berichte und Abfragen, die auch auf die aktuellen Daten des DWH zugreifen, drohen somit aufgrund fehlender Statistiken und unpassender Ausführungspläne erhöhte Laufzeiten. Für DWH-Systeme ist dieses Verhalten nicht akzeptabel. Deshalb sollte das Erzeugen der Statistiken regelmäßig innerhalb des ETL-Prozesses erfolgen. Damit wird sichergestellt, dass die Statistiken stets aktuell sind und die Performance der Abfragen sich nicht verschlechtert.

Fazit

Aktuelle Statistiken sind für das Reporting im DWH unverzichtbar, um performante Berichte und Abfragen zu garantieren. Mit den Möglichkeiten von Oracle 11*g* und der richtigen Strategie ist es möglich, auch für große Datenmengen eines DWH den Aufwand für das Erzeugen aktueller Statistiken gering zu halten. Für eine Strategie zur Erzeugung von Statistiken im DWH sollten folgende Punkte beachtet und anhand der konkreten Situation bewertet werden:

- Stets lokale Statistiken für geänderte Partitionen und globale Statistiken aktualisieren
- Globale Statistiken für partitionierte Fakten-Tabellen inkrementell erzeugen
- Histogramme für Spalten mit ungleichmäßig verteilten Daten, über die häufig gefiltert wird, verwenden
- Extended Statistics für korrelierende Spalten erstellen, die in Standardberichten als Filter genutzt werden
- Dynamic Sampling mit Level 4 oder höher einsetzen, um auch für Ad-hoc-Abfragen gute Laufzeiten beim Filtern über korrelierende Spalten zu erzielen
- Statistiken in der Regel mit „AUTO_SAMPLE_SIZE“ erzeugen. Nur bei sehr großen Fakten-Tabellen gegebenenfalls gezielt kleine Werte für „ESTIMATE_PERCENT“ angeben
- Statistiken im DWH nicht mit dem automatischen Statistik-Job der Datenbank, sondern gezielt im ETL-Prozess erzeugen

Reinhard Mense
reinhard.mense@areto-consulting.de