



Globale Statistiken im Oracle Data Warehouse

Dani Schnider
Principal Consultant
29. Januar 2012



Aktuelle und vollständige Optimizer-Statistiken sind Voraussetzung für die Ermittlung von guten Execution Plans durch den Cost-based Optimizer. Das gilt insbesondere für globale Statistiken auf partitionierten Tabellen. Auf den folgenden Seiten wird erläutert, weshalb globale Statistiken wichtig sind und wie sie auf unterschiedliche Arten berechnet werden können. Der Artikel schliesst mit Empfehlungen, wie und zu welchem Zeitpunkt diese Berechnungen in Data Warehouses durchgeführt werden sollen.

Für partitionierte Tabellen werden Statistiken auf verschiedenen Granularitätsstufen gesammelt. *Globale Statistiken* werden für Abfragen über alle oder mehrere Partitionen verwendet und enthalten Informationen über die gesamte Tabelle. *Statistiken pro Partition* sind für Abfragen auf eine Partition sowie für die Kostenschätzungen von Partition Pruning notwendig. *Statistiken pro Subpartition* sind nur für Tabellen mit Composite Partitioning vorhanden und werden für Abfragen auf eine oder mehrere Subpartitionen verwendet.

Um realistische Schätzungen für die Selektivität einer Abfrage ermitteln zu können, benötigt der Optimizer Informationen über die Tabelle (z.B. Anzahl Datensätze) und über die Attribute in der WHERE-Bedingung (z.B. minimaler/maximaler Wert sowie Anzahl unterschiedliche Werte pro Attribut). Für Abfragen über alle oder über mehrere Partitionen werden diese Angaben nicht nur für die einzelnen Partitionen benötigt, sondern auch für die gesamte Tabelle. Die meisten dieser Informationen können aus den untergeordneten Ebenen abgeleitet werden. So ist die Anzahl Datensätze der Tabelle die Summe der Datensätze aller Partitionen. Auf Attributebene funktioniert dies jedoch nicht immer. Die Anzahl verschiedener Werte pro Attribut kann nicht so einfach auf globale Werte hochgerechnet werden. Deshalb werden idealerweise nicht nur Statistiken pro Partition und Subpartition berechnet, sondern zusätzlich auch für die gesamte Tabelle. Diese Statistiken werden als globale Statistiken bezeichnet.

Für die folgenden Betrachtungen verzichten wir auf Subpartitionen und Indexstatistiken und konzentrieren und vor allem um die globalen Tabellen- und Attributstatistiken und die Frage, wann und auf welche Weise sie in einem Data Warehouse berechnet werden sollen.

Als Beispiel verwenden wir eine Faktentabelle FCT_SALES, welche zur Zeit 12 Monatspartitionen für die Monate Januar 2011 bis Dezember 2011 umfasst. Die Tabelle enthält drei Dimensionsattribute TIME_ID, CUST_ID und PROD_ID. Momentan sind 1.2 Millionen Rows geladen, der nächste ETL-Job zum Laden der Fakten vom Januar 2012 soll demnächst (also auf den nächsten paar Seiten) durchgeführt werden. Die folgende Auswertung zeigt die Datenverteilung:

```
SELECT COUNT(*) AS num_rows
       , COUNT(DISTINCT(time_id)) AS dist_time_id
       , COUNT(DISTINCT(cust_id)) AS dist_cust_id
       , COUNT(DISTINCT(prod_id)) AS dist_prod_id
FROM fct_sales;
```

NUM_ROWS	DIST_TIME_ID	DIST_CUST_ID	DIST_PROD_ID
1200000	365	7300	33

Um möglichst genaue Statistiken zu erhalten, wird in den nachfolgenden Beispielen die Konstante *auto_sample_size* verwendet. In Oracle 11g wurde für diesen Wert ein neuer Algorithmus eingeführt, dessen Genauigkeit nahezu 100% entspricht, für die Berechnung der Statistiken jedoch etwa so lange dauert wie mit *estimate_percent* = 10% (vgl. Referenz [1]). Ob *auto_sample_size* auch in realen Data Warehouses sinnvoll ist, wird am Ende des Artikels genauer erläutert.



Aggregierte globale Statistiken

Wenn keine globalen Statistiken vorhanden sind, werden diese automatisch aus den darunter liegenden Ebenen aggregiert. Das bedeutet, dass nach dem Berechnen der Statistiken einer Partition jeweils die Statistiken aller Partitionen aggregiert und als globale Statistiken im Data Dictionary gespeichert werden. Normalerweise geht dies sehr schnell, aber bei Tabellen mit Tausenden von Partitionen und Zehntausenden von Subpartitionen kann dieser zusätzliche Schritt trotzdem zu langen Laufzeiten führen. Dies ist aber nicht das Hauptproblem von aggregierten globalen Statistiken.

Für unser Beispiel berechnen wir die Statistiken auf Ebene der Partitionen für alle Partitionen der Tabelle FCT_SALES. Dies erfolgt mit folgendem Aufruf von DBMS_STATS:

```
dbms_stats.gather_table_stats(ownname      => USER
                             ,tabname      => 'FCT_SALES'
                             ,estimate_percent=> dbms_stats.auto_sample_size
                             ,granularity  => 'PARTITION');
```

Sehen wir uns das Ergebnis in der Data Dictionary-View USER_TAB_STATISTICS an, so stellen wir fest, dass für alle Partitionen die Statistiken berechnet wurden. Mit der Einstellung *auto_sample_size* in Oracle 11g sind die Werte für NUM_ROWS und SAMPLE_SIZE jeweils identisch. Das bedeutet, dass die Statistiken genau (100%) berechnet werden. In der obersten Zeile sind die Statistiken auf Tabellenebene dargestellt. Der Wert von NUM_ROWS ist korrekt und entspricht der Summe der einzelnen Partitionen (wer es nicht glaubt, kann nachrechnen). SAMPLE_SIZE ist jedoch 0, und GLOBAL_STATS ist ,NO'. Dies bedeutet, dass die globalen Statistiken auf Tabellenebene nicht berechnet, sondern nur aus der darunterliegenden Statistiken aggregiert wurden.

TABLE_NAME	PARTITION_NAME	NUM_ROWS	SAMPLE_SIZE	GLOBAL_STATS
FCT_SALES		1200000	0	NO
FCT_SALES	PT_2011_01	101927	101927	YES
FCT_SALES	PT_2011_02	92064	92064	YES
FCT_SALES	PT_2011_03	101928	101928	YES
FCT_SALES	PT_2011_04	98640	98640	YES
FCT_SALES	PT_2011_05	101928	101928	YES
FCT_SALES	PT_2011_06	98640	98640	YES
FCT_SALES	PT_2011_07	101928	101928	YES
FCT_SALES	PT_2011_08	101928	101928	YES
FCT_SALES	PT_2011_09	98613	98613	YES
FCT_SALES	PT_2011_10	101897	101897	YES
FCT_SALES	PT_2011_11	98610	98610	YES
FCT_SALES	PT_2011_12	101897	101897	YES

Das Ergebnis ist korrekt und entspricht unseren Erwartungen – übrigens auch für alle anderen Statistikattribute auf Tabellenebene, die hier aus Platzgründen nicht dargestellt sind. Doch wie sehen die globalen Statistiken für die einzelnen Attribute aus? Werfen wir einen Blick auf die Data Dictionary-Views USER_TAB_COL_STATISTICS und USER_PART_COL_STATISTICS.

Interessant ist dabei vor allem die Spalte NUM_DISTINCT, welche die Anzahl unterschiedlicher Werte pro Attribut enthält. Sie ist korrekt für TIME_ID (den Partition Key) sowie für PROD_ID. Das Jahr 2011 hat tatsächlich 365 verschiedene Tage, und unsere Beispiel umfasst Referenzen auf 33 verschiedene Produkte. Nicht korrekt ist jedoch der Wert für CUST_ID auf globaler Ebene. Anstatt 7300 verschiedene Kunden wurden nur 620 unterschiedliche Werte ermittelt.



TABLE_NAME	PARTITION_NAME	COLUMN_NAME	NUM_DISTINCT	SAMPLE_SIZE	GLOBAL_STATS
FCT_SALES		TIME_ID	365		NO
FCT_SALES		CUST_ID	620		NO
FCT_SALES		PROD_ID	33		NO
FCT_SALES	PT_2011_01	TIME_ID	31	101927	YES
FCT_SALES	PT_2011_02	TIME_ID	28	92064	YES
FCT_SALES	PT_2011_03	TIME_ID	31	101928	YES
...					
FCT_SALES	PT_2011_01	CUST_ID	620	101927	YES
FCT_SALES	PT_2011_02	CUST_ID	560	92064	YES
FCT_SALES	PT_2011_03	CUST_ID	620	101928	YES
...					

Die Anzahl unterschiedlicher Werte pro Attribut – oft auch als NDV (number of distinct values) bezeichnet – ist eine wichtige Information für den Cost-based Optimizer zur Berechnung der Selektivität einer Abfrage. Geht der Optimizer hier von falschen Annahmen aus, kann dies zu fehlerhaften Kostenberechnungen und somit zu nicht optimalen Execution Plans führen. Dies ist der wesentliche Nachteil von aggregierten globalen Statistiken. Er kann nur vermieden werden, indem die globalen Statistiken in einem separaten Schritt berechnet werden, wie im nächsten Kapitel beschrieben.

Nun werden die Daten für Januar 2012 in die Faktentabelle geladen. Dazu wird eine neue Partition PT_2012_01 erstellt und geladen. Anschliessend werden die Statistiken berechnet, jedoch nur für die neu geladene Partition:

```
dbms_stats.gather_table_stats(ownname      => USER
                             ,tabname      => 'FCT_SALES'
                             ,partname     => 'PT_2012_01'
                             ,estimate_percent=> dbms_stats.auto_sample_size
                             ,granularity  => 'PARTITION');
```

Die Abfrage auf USER_TAB_STATISTICS zeigt, dass nun nicht nur die Statistiken für die neu hinzugefügte Partition vorhanden sind, sondern dass auch die globalen Statistiken nachgeführt, d.h. neu aggregiert wurden:

TABLE_NAME	PARTITION_NAME	NUM_ROWS	SAMPLE_SIZE	GLOBAL_STATS
FCT_SALES		1300000	0	NO
FCT_SALES	PT_2011_01	101927	101927	YES
FCT_SALES	PT_2011_02	92064	92064	YES
FCT_SALES	PT_2011_03	101928	101928	YES
FCT_SALES	PT_2011_04	98640	98640	YES
FCT_SALES	PT_2011_05	101928	101928	YES
FCT_SALES	PT_2011_06	98640	98640	YES
FCT_SALES	PT_2011_07	101928	101928	YES
FCT_SALES	PT_2011_08	101928	101928	YES
FCT_SALES	PT_2011_09	98613	98613	YES
FCT_SALES	PT_2011_10	101897	101897	YES
FCT_SALES	PT_2011_11	98610	98610	YES
FCT_SALES	PT_2011_12	101897	101897	YES
FCT_SALES	PT_2012_01	100000	100000	YES

Hier liegt der Vorteil von aggregierten globalen Statistiken. Es sind keine weiteren Aufrufe von DBMS_STATS notwendig, um die globalen Statistiken zu aktualisieren. Bei jeder Berechnung der Statistiken für eine Partition werden die globalen Statistiken automatisch neu aggregiert.



Berechnete globale Statistiken

Um das Problem mit den falschen NDV-Werten in den globalen Statistiken zu lösen, werden die Statistiken nicht nur für alle Partitionen, sondern zusätzlich noch auf globaler Ebene berechnet. Dazu wird der Parameter *granularity* auf ‚GLOBAL AND PARTITION‘ oder auf ‚AUTO‘ (Default-Wert) gesetzt.

```
dbms_stats.gather_table_stats(ownname      => USER
                             ,tabname      => 'FCT_SALES'
                             ,estimate_percent=> dbms_stats.auto_sample_size
                             ,granularity   => 'GLOBAL AND PARTITION');
```

Die Auswertung auf USER_TAB_STATISTICS zeigt, dass die globalen Statistiken nun nicht mehr aggregiert, sondern in einem separaten Schritt explizit berechnet wurden. Dies ist aus den Attributen SAMPLE_SIZE und GLOBAL_STATS ersichtlich:

TABLE_NAME	PARTITION_NAME	NUM_ROWS	SAMPLE_SIZE	GLOBAL_STATS
FCT_SALES		1200000	1200000	YES
FCT_SALES	PT_2011_01	101927	101927	YES
FCT_SALES	PT_2011_02	92064	92064	YES
FCT_SALES	PT_2011_03	101928	101928	YES
FCT_SALES	PT_2011_04	98640	98640	YES
FCT_SALES	PT_2011_05	101928	101928	YES
FCT_SALES	PT_2011_06	98640	98640	YES
FCT_SALES	PT_2011_07	101928	101928	YES
FCT_SALES	PT_2011_08	101928	101928	YES
FCT_SALES	PT_2011_09	98613	98613	YES
FCT_SALES	PT_2011_10	101897	101897	YES
FCT_SALES	PT_2011_11	98610	98610	YES
FCT_SALES	PT_2011_12	101897	101897	YES

Auch auf Attributebene sind nun die Statistiken auf globaler Ebene korrekt. Der Wert für NUM_DISTINCT ist für alle Attribute richtig, auch für CUST_ID, wo wir bei den aggregierten Statistiken einen viel zu kleinen Wert hatten. Jetzt wurde der korrekte Wert 7300 berechnet:

TABLE_NAME	PARTITION_NAME	COLUMN_NAME	NUM_DISTINCT	SAMPLE_SIZE	GLOBAL_STATS
FCT_SALES		TIME_ID	365	1200000	YES
FCT_SALES		CUST_ID	7300	1200000	YES
FCT_SALES		PROD_ID	33	1200000	YES
FCT_SALES	PT_2011_01	TIME_ID	31	101927	YES
FCT_SALES	PT_2011_02	TIME_ID	28	92064	YES
FCT_SALES	PT_2011_03	TIME_ID	31	101928	YES
...					

Berechnete globale Statistiken sind somit genauer als aggregierte Statistiken. Das Problem der falschen NDV-Werte tritt nicht mehr auf. Doch was ist der Preis dafür?

Nach dem Erstellen der neuen Partition PT_2012_01 und dem Laden der Daten für Januar 2012 werden die Statistiken für die neue Partition berechnet:

```
dbms_stats.gather_table_stats(ownname      => USER
                             ,tabname      => 'FCT_SALES'
                             ,partname     => 'PT_2012_01'
                             ,estimate_percent=> dbms_stats.auto_sample_size
                             ,granularity   => 'PARTITION');
```



Wenn wir nun die Statistiken in USER_TAB_STATISTICS überprüfen, so stellen wir fest, dass zwar die neue Partition vorhanden ist, dass aber im Gegensatz zur Variante mit den aggregierten Statistiken die globalen Statistiken nicht aktualisiert wurden. Die gesamte Anzahl Datensätze ist immer noch gleich hoch wie vorher. Das gleiche gilt auch für die globalen Statistiken der einzelnen Attribute.

TABLE_NAME	PARTITION_NAME	NUM_ROWS	SAMPLE_SIZE	GLOBAL_STATS
FCT_SALES		1200000	1200000	YES
FCT_SALES	PT_2011_01	101927	101927	YES
FCT_SALES	PT_2011_02	92064	92064	YES
FCT_SALES	PT_2011_03	101928	101928	YES
FCT_SALES	PT_2011_04	98640	98640	YES
FCT_SALES	PT_2011_05	101928	101928	YES
FCT_SALES	PT_2011_06	98640	98640	YES
FCT_SALES	PT_2011_07	101928	101928	YES
FCT_SALES	PT_2011_08	101928	101928	YES
FCT_SALES	PT_2011_09	98613	98613	YES
FCT_SALES	PT_2011_10	101897	101897	YES
FCT_SALES	PT_2011_11	98610	98610	YES
FCT_SALES	PT_2011_12	101897	101897	YES
FCT_SALES	PT_2012_01	100000	100000	YES

Damit die globalen Statistiken nachgeführt werden, muss beim Berechnen der neuen Partition der Wert für den Parameter *granularity* auf ‚GLOBAL AND PARTITION‘ bzw. ‚AUTO‘ gesetzt werden. Nur dann werden auch die globalen Statistiken neu berechnet.

```
dbms_stats.gather_table_stats(ownname      => USER
                             ,tabname      => 'FCT_SALES'
                             ,partname     => 'PT_2012_01'
                             ,estimate_percent=> dbms_stats.auto_sample_size
                             ,granularity  => 'GLOBAL AND PARTITION');
```

Auch wenn hier nur eine Partition angegeben wurde, dauert die Berechnung nun viel länger, weil für die Ermittlung der globalen Statistiken alle Partitionen der Tabelle gelesen werden müssen. Nach Abschluss der Prozedur sind die Statistiken für die neue Partition PT_2012_01 sowie die globalen Statistiken aktualisiert:

TABLE_NAME	PARTITION_NAME	NUM_ROWS	SAMPLE_SIZE	GLOBAL_STATS
FCT_SALES		1300000	1300000	YES
FCT_SALES	PT_2011_01	101927	101927	YES
FCT_SALES	PT_2011_02	92064	92064	YES
FCT_SALES	PT_2011_03	101928	101928	YES
FCT_SALES	PT_2011_04	98640	98640	YES
FCT_SALES	PT_2011_05	101928	101928	YES
FCT_SALES	PT_2011_06	98640	98640	YES
FCT_SALES	PT_2011_07	101928	101928	YES
FCT_SALES	PT_2011_08	101928	101928	YES
FCT_SALES	PT_2011_09	98613	98613	YES
FCT_SALES	PT_2011_10	101897	101897	YES
FCT_SALES	PT_2011_11	98610	98610	YES
FCT_SALES	PT_2011_12	101897	101897	YES
FCT_SALES	PT_2012_01	100000	100000	YES

Der Preis für die genauen NDV-Werte ist somit ziemlich hoch, da nach dem Laden jeder Partition die globalen Statistiken neu berechnet werden müssen. In unserem Beispiel fällt dies nicht ins Gewicht, aber in grossen Data Warehouses, die täglich neue Partitionen laden, ist eine regelmässige Berechnung der globalen Statistiken am Ende des ETL-Laufs kaum möglich.



Inkrementelle globale Statistiken

Ideal wäre es, wenn die Vorteile von aggregierten und berechneten globalen Statistiken kombiniert werden könnten. Wir möchten genaue NDV-Werte für alle Attribute auf globaler Ebene haben, aber trotzdem nicht jedesmal alle Partitionen lesen müssen, um die globalen Statistiken zu aktualisieren.

Seit Oracle 11g ist dies möglich, indem inkrementelle globale Statistiken verwendet werden. Um dieses Feature zu verwenden, muss es für die gewünschten Tabellen aktiviert werden:

```
dbms_stats.set_table_prefs(ownname => USER
                           ,tabname => 'FCT_SALES'
                           ,pname  => 'incremental'
                           ,pvalue => 'true');
```

Sind inkrementelle Statistiken aktiviert, werden für jede Partition zusätzliche Informationen gesammelt und in einer sogenannten „Synopsis“ (Zusammenfassung) pro Partition im SYSAUX-Tablespace abgespeichert. Aus diesen Zusammenfassungen können dann die globalen Statistiken ermittelt werden, ohne dass sämtliche Partitionen gelesen werden müssen.

Die Berechnung der Statistiken erfolgt wiederum mit dem Wert ‚GLOBAL AND PARTITION‘ bzw. ‚AUTO‘ für *granularity*. Beim nachfolgenden Aufruf ist noch kein Unterschied zu erkennen, da in diesem Fall sowieso alle Partitionen neu berechnet und somit gelesen werden müssen.

```
dbms_stats.gather_table_stats(ownname      => USER
                              ,tabname     => 'FCT_SALES'
                              ,estimate_percent=> dbms_stats.auto_sample_size
                              ,granularity  => 'GLOBAL AND PARTITION');
```

Der Nutzen von inkrementellen globalen Statistiken wird aber ersichtlich, wenn anschliessend nur die Statistiken einer Partition berechnet werden. Nach dem Erstellen und Laden der Partition PT_2012_01 werden auf die gleiche Weise wie bisher die Statistiken für die neue Partition berechnet:

```
dbms_stats.gather_table_stats(ownname      => USER
                              ,tabname     => 'FCT_SALES'
                              ,partname    => 'PT_2012_01'
                              ,estimate_percent=> dbms_stats.auto_sample_size
                              ,granularity  => 'GLOBAL AND PARTITION');
```

Schauen wir uns danach die Statistiken an, erkennen wir keinen Unterschied zur Variante mit den berechneten globalen Statistiken. Die Statistiken für Tabelle und alle Attribute sind aktuell und korrekt, und zwar sowohl für die neue Partition als auch auf globaler Ebene.

TABLE_NAME	PARTITION_NAME	NUM_ROWS	SAMPLE_SIZE	GLOBAL_STATS
FCT_SALES		1300000	1300000	YES
FCT_SALES	PT_2011_01	101927	101927	YES
FCT_SALES	PT_2011_02	92064	92064	YES
...				
FCT_SALES	PT_2011_12	101897	101897	YES
FCT_SALES	PT_2012_01	100000	100000	YES

Der wesentliche Unterschied gegenüber vorher besteht aber darin, dass das Berechnen der globalen Statistiken nun weniger lange dauert, weil nicht mehr alle bestehenden Partitionen gelesen werden müssen, sondern nur noch die Zusammenfassungen im SYSAUX-Tablespace.



Sind inkrementelle globale Statistiken somit die perfekte Lösung? Auf den ersten Blick schon, denn wir haben nun genaue Statistikwerte – auch für die NDV-Werte – und müssen für die Berechnung der globalen Statistiken nicht mehr alle Partitionen lesen. Leider gibt es aber ein paar Einschränkungen für dieses 11g-Feature.

Für inkrementelle Statistiken wird immer *auto_sample_size* verwendet. Der Parameter *estimate_percent* wird ignoriert. Für Tabellen mit sehr grossen Partitionen wird aber häufig der Ansatz gewählt, dass nur ein kleiner Prozentsatz der Datensätze (*estimate_percent* ≤ 1%) als Basis für die Statistikberechnung verwendet wird. Dies funktioniert aber nicht in Kombination mit inkrementellen Statistiken. Auch wenn der Algorithmus, der für *auto_sample_size* verwendet wird, in Oracle 11g „nur“ noch etwa der Zeit für *estimate_percent* = 10% entspricht, ist es für viele grössere Data Warehouses nicht realistisch, im zur Verfügung stehenden ETL-Zeitfenster so viel Zeit für die Berechnung von Statistiken zu verwenden (siehe Referenz [2]).

Die abgespeicherten Informationen pro Synopsis im SYSAUX-Tablespace kann unter Umständen sehr umfangreich werden, insbesondere bei Tabellen mit vielen Partitionen/Subpartitionen und vielen Attributen. Das Nachführen dieser Zusatzinformationen führt dann zu einem erheblichen Mehraufwand, vor allem beim Löschen von alten Synopsis-Werten (siehe Referenz [3]).

Empfehlungen für Data Warehouses

In Data Warehouses erfolgt die Statistikberechnung idealerweise am Ende der ETL-Prozesse. Die Berechnung von globalen Statistiken würde zu diesem Zeitpunkt aber zu viel Zeit in Anspruch nehmen. Inkrementelle Statistiken können hier Abhilfe schaffen, sofern die Datenmenge relativ klein ist. Für grosse Tabellen, die Werte von *estimate_percent* ≤ 1% erfordern, kommen inkrementelle Statistiken nicht in Frage, da sie nur mit *auto_sample_size* verwendet werden können.

In diesen Fällen sollten auch mit Oracle 11g weiterhin globale Statistiken berechnet werden, jedoch nicht am Ende des ETL-Laufs, sondern in regelmässigen Abständen zu einem Zeitpunkt, in welchem keine Daten geladen werden (z.B. tagsüber oder am Wochenende). Die Berechnung der globalen Statistiken erfolgt somit asynchron zu einem späteren Zeitpunkt, sollte aber nicht zu lange aufgeschoben werden, da sonst die Abweichungen zwischen realer Datenverteilung und globalen Statistiken zu gross wird.

Dani Schnider

Trivadis AG

Europa-Strasse 5

CH-8152 Glattbrugg

Internet: www.trivadis.com

Tel: +41(0)44-808 70 20

Fax: +41(0)44-808 70 21

Mail: info@trivadis.com

Referenzen

[1] http://blogs.oracle.com/optimizer/entry/improvement_of_auto_sampling_statistics_gathering_feature_in_oracle_database_11g

[2] <http://blog.trivadis.com/b/danischnider/archive/2011/06/15/incremental-statistics-a-mixed-blessing.aspx>

[3] <http://oracle-randolf.blogspot.com/2012/01/incremental-partition-statistics-review.html>