Was man über Oracle-Histogramme im DWH wissen muss

Philipp Krätzig
areto consulting GmbH
Köln

Schlüsselworte
Histogramme, Statistiken, Cost-Based-Optimizer

Einleitung

Die Erzeugung stets aktueller und korrekter Statistiken ist essentiell, um performante Ausführungspläne im DWH zu gewährleisten. Aber selbst dann, kann der Optimizer der Datenbank in einigen Fällen merkwürdige Annahmen treffen, die mitunter zu sehr ungünstigen Plänen führen. In vielen dieser Fälle können Histogramme eine effektive Lösung sein, aber deren Erzeugung kostet auch wiederum Zeit. Daher stellen sich einige Fragen: In welchen Fällen sind Histogramme nur hilfreich? Was sind Histogramme eigentlich genau? Welche Arten gibt es? Wie wendet man Histogramme an und wo findet man Informationen über die Histogramme in meiner Datenbank?

Der Beitrag soll sowohl das nötige Hintergrundwissen über Histogramme vermitteln, als auch dessen Anwendung anhand von praktischen Beispielen demonstrieren, so dass diese Fragen beantwortet werden. Einige allgemeingültige Best Practices zum Einsatz von Histogrammen runden den Vortrag ab.

Was sind Histogramme?

Ein Histogramm speichert Informationen über die Verteilung von Werten innerhalb einer Tabellenspalte und ist damit Bestandteil der Spaltenstatistiken des Oracle Optimizers. Ohne Erstellung eines Histogramms wird eine flache Verteilung der Werte angenommen, die aus einigen einfacheren Kennzahlen der Spaltenstatistik errechnet werden kann.

Bei einem Histogramm dagegen wird der Wertebereich einer Spalte in mehrere Kategorien (in der Oracle-Terminologie Buckets genannt) aufgeteilt und für jeden Bucket, die Anzahl an Zeilen, bei der der Wert in dieser in diese Kategorie fällt abgespeichert. Da es bei Tabellenspalten mit einer großen Anzahl unterschiedlicher Werte irgendwann zu aufwändig werden würde immer weitere Kategorien zu speichern, hat Oracle die maximale Anzahl an Buckets auf 254 (ab 12c auf 2.000) festgelegt. Dies hat auch zur Folge, dass es unterschiedliche Arten von Histogrammen gibt.

Warum sind Histogramme wichtig?

Der Anteil an Zeilen, die eine Datenquelle im Verhältnis zur Gesamtmenge zurückgibt (Selektivität), spielt eine wichtige Rolle bei der Optimierung einer Abfrage durch den Cost-Based-Optimizer. Mann kann sagen, dass der CBO bei geringer Selektivität Index-Scans bevorzugt, während er einen Full-Table-Scan wählt, wenn ein Großteil der Datensätze einer Tabelle abgerufen werden müssen. Auch auf Joins wirkt sich die angenommene Selektivität aus, da bei geringer Kardinalität (Selektivität Zeilenzahl) Nested-Loops gegenüber einem Hash-Join gewählt werden könnten. Der CBO muss also

eine möglichst genaue Schätzung der Selektivität vornehmen, um den optimalen Ausführungsplan bestimmen zu können. Dabei helfen folgende Statistiken:

- Anzahl unterschiedlicher Werte innerhalb der Spalte (NUM_DISTINCT)
- Mininum- und Maximumwert der Spalte (LOW_VALUE, HIGH_VALUE)
- Anzahl an NULL-Werten (NUM_NULLS)
- Wertverteilung bzw. Histogramm (optional)

Wie die obige Aufzählung zeigt, sind Histogramme ein optionales Hilfsmittel, um genau diese Schätzung zu verbessern. Dies ist dann notwendig, wenn eine schiefe (skew), d.h. sehr ungleichmäßige Werteverteilung vorliegt. Ist kein Histogramm für eine Spalte vorhanden (Normalfall), wird dagegen eine ungenauerere Methode als Fallback verwendet.

Frequenzhistogramme

Optimalerweise gibt es in der betrachteten Tabellenspalte weniger als oder genau 254 (bzw. 2.000 ab 12c) unterschiedliche Werte. In diesem Fall erstellt Oracle ggf. ein sogenanntes Frequenzhistogramm, d.h. für jeden unterschiedlichen Wert kann ein Bucket genutzt werden, der dann genau die Anzahl der Vorkommnisse (also Frequenz) dieses Wertes in der Spalte angibt (Abb. 1).

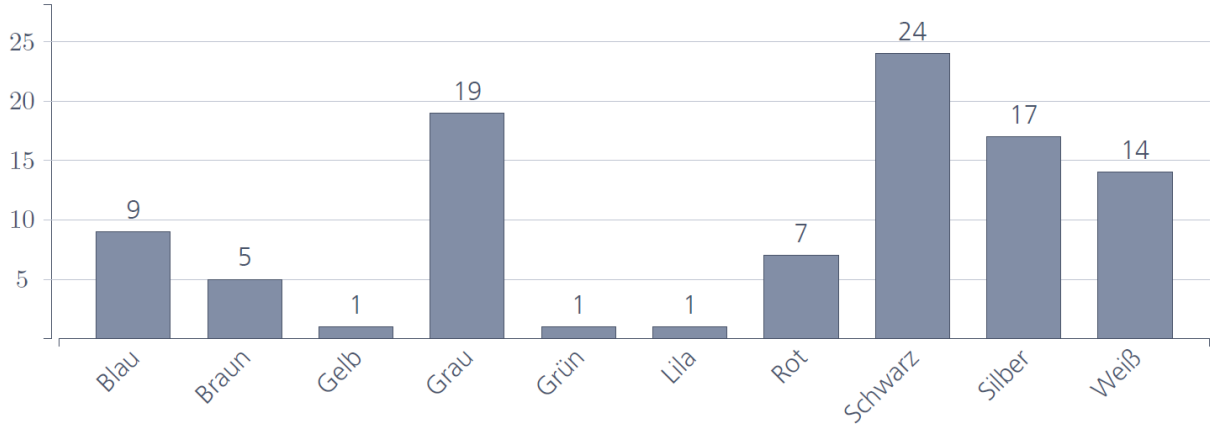

Abb. 1: Frequenzhistogramm Farbe Autobestellungen

Bei der Selektion eines Wertes, der in der Tabelle vorhanden ist, kann also die Kardinalität einfach aus dem Frequenzhistogramm abgelesen werden und es müssen keine weiteren Berechnungen erfolgen. Für Werte außerhalb des Min-Max-Intervalls gilt dagegen automatisch die minimale Kardinalität von 1. Eine Besonderheit ist der Fall eines zum Zeitpunkt der Statistikerstellung nicht existenten Wertes, der allerdings innerhalb des Wertebereichs liegt. Hier wird seit Version 11g zur Schätzung der Kardinalität ein Wert namens NewDensity herangezogen. Dieser wird adhoc berechnet und steht daher in keinem Zusammenhang mit dem im Data Dictionary gespeicherten Wert DENSITY (OldDensity). Das kann leicht zu Verwirrung führen, da NewDensity lediglich über die Optimizer-Traces beobachtbar ist. Der grundsätzliche Ansatz hierbei ist, dass die Kardinalität der Hälfte der Frequenz des seltensten Wertes entsprechen soll. Da die Kardinalität wie im Fall ohne Histogramme die Multiplikation von Density und NUM_ROWS ist, ergibt sich für die NewDensity folgende Formel:

*NewDensity = 0.5 * freq(least_popular_value) / NUM_ROWS*

Top-Frequenzhistogramm

Ab Version 12c gibt es sog. Top-Frequenzhistogramme. Dabei handelt es sich um eine Variante des Frequenzhistogramms, bei der seltene Werte, die statistisch nicht signifikant sind, ignoriert werden. Dadurch können Buckets eingespart werden und es kann ein im Hinblick auf häufig vorkommende Werte besseres Histogramm erzeugt werden. Abhängig von der Anzahl an Buckets n wird ein Top-Frequenzhistogramm automatisch erstellt, wenn nicht genügend Buckets für ein normales Frequenzhistogramm vorhanden sind und folgende Regel zutrifft:

Der Prozentsatz der Zeilen mit den n häufigsten Werten ist größer oder gleich dem Schwellwert p mit p = (1 – 1/n) * 100

Höhenbalancierte Histogramme

In einem höhenbalancierten Histogramm werden die Spaltenwerte so auf die Buckets aufgeteilt, dass in jedem ungefähr die gleiche Anzahl enthalten ist. Bei 98 Zeilen und 8 Buckets entspräche das beispielsweise 13 Zeilen pro Bucket, wobei der letzte Bucket nur 17 Zeilen umfassen würde. Werte die durch diese Aufteilung mindestens zweimal das Bucket-Ende (Endpoint) darstellen, gelten als populäre Werte.

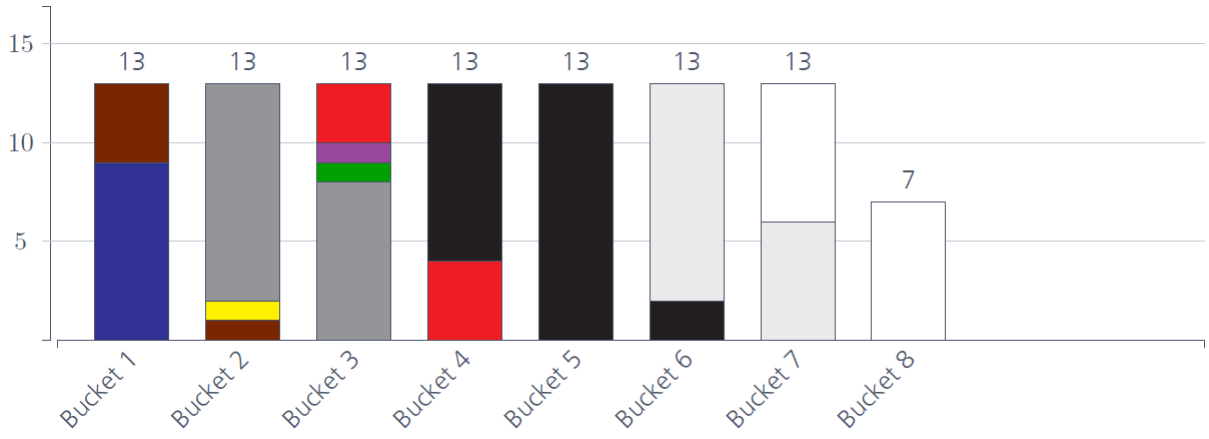

Abb. 2: Höhenbalanciertes Histogramm mit 8 Buckets

Für diese populären Werte wird die Selektivität berechnet nach dem Verhältnis zwischen der Anzahl an Bucket-Endpoints die der Wert umspannt und der Gesamtzahl an Buckets: selectivity = BktCnt(Value) / BktCnt. In dem gegebenen Beispiel würde eine Filterung auf die Farbe ”Schwarz” eine Kardinalität von 25 Zeilen ergeben. Da Schwarz zwei Endpoints von den acht Buckets besetzt würde ein Viertel der Zeilen, also 24.5 aufgerundet 25 als Selektivität geschätzt.

Filtert man dagegen auf unpopuläre Werte, so berechnet sich die Kardinalität wiederum nach der bereitsbekannten Formel cardinality = density * NUM_ROWS, wobei sich die Dichte hier allerdings anders berechnet als bei den Frequenzhistogrammen. Die Formel ist in diesem Fall:

NewDensity = (BktCnt – PopBktCnt / BktCnt) / (NDV – BktCnt)

Durch Einsetzen der Werte (BktCnt: 8, PopBktCnt: 4, PopValCnt: 2, NDV: 10, NUM_ROWS: 98) ergibt sich somit für das Beispiel:

(8 – 4) / 8 / (10 – 2) = 4 / 8 / 8 = 1 / 16 → 1 / 16 * 98 = 6.125 ~ 6

Hybride Histogramme

Ein hybrides Histogramm kombiniert die Eigenschaften eines Frequenz- sowie eines höhenbalancierten Histogramms, um die Nachteile eines rein höhenbalancierten Histogramms auszugleichen. Rein höhenbalancierte Histogramme sind nämlich in manchen Fällen sehr ungenau bei fast populären Werten, d.h. Werten, die gerade so nicht mehr zu den populären Werten gehören.

Um dieses Problem zu lösen, sind bei einem hybriden Histogramm die Werte so verteilt, dass kein Wert mehr als einen Bucket belegt. Dazu werden ggf. die Bucket-Grenzen entsprechend verschoben. Zusätzlich wird zu jedem Endpoint, die Anzahl (repeat count) gespeichert wie oft der entsprechende Endpoint-Wert vorkommt. Dadurch hat der Optimizer die Möglichkeit auch Schätzungen für fast populäre Werte mit hoher Genauigkeit durchzuführen.

Abbildung 4: *Hybrides Histogramm mit 4 Buckets*

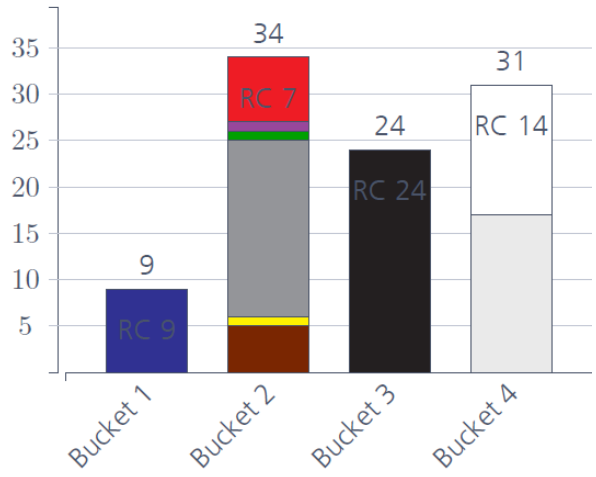

Abb. 3: Hybrides Histogramm mit 4 Buckets

Ab Version 12c werden automatisch hybride Histogramme erstellt, sobald die Kriterien für Frequenzhistogramme, bzw. Top-Frequenzhistogramme nicht mehr erfüllt sind und die Sample-Größe auf AUTO_SAMPLE_SIZE belassen wurde. Sie lösen damit die höhenbalancierten Histogramme ab. Letztere werden nur noch erstellt, falls eine benutzerdefinierte Sample-Größe angegeben wird.

Erstellung von Histogrammen

Die Erstellung von Histogrammen erfolgt in der Regel über die Prozedur GATHER_TABLE_STATS des Packages DBMS_STATS. Der für die Histogramm-Erstellung relevante Parameter lautet method_opt. Mit diesem wird definiert auf welchen Spalten der Tabelle Histogramme erstellt werden sollen und wie viele Buckets maximal verwendet werden sollen. Bei der Auswahl der Zielspalten stehen folgende Möglichkeiten zur Auswahl:

- FOR ALL COLUMNS → Auf allen Spalten
- FOR ALL INDEXED COLUMNS → Auf allen indizierten Spalten
- FOR ALL HIDDEN COLUMNS → Auf allen versteckten Spalten (Function-Based-Index, …)
- FOR COLUMNS … → Gezielte Angabe einzelner Spalten

Nach Angabe der Zielspalten folgt die sogenannte SIZE-Klausel, die entweder eine direkte Bucket-Zahl für das Histogramm angibt oder aber einen Mechanismus nach dem die Datenbank selbst entscheiden soll, ob ein Histogramm ggf. erstellt wird. Mögliche Werte sind hier:

- SIZE 1 → kein Histogramm erstellen, nur Minimum, Maximum und NDV bestimmen
- SIZE n → Histogramm mit n Buckets erstellen (n e {2, … , 254|2000})
- SIZE REPEAT → Histogramme nur auf Spalten erstellen, wo bereits welche vorhanden sind
- SIZE SKEWONLY → Datenbank entscheidet anhand der Schiefe der Datenverteilung
- SIZE AUTO → Datenbank entscheidet anhand der Schiefe der Datenverteilung und Workload

Ohne weitere Angaben wird standardmäßig übrigens die Option FOR ALL COLUMNS SIZE AUTO genutzt was nicht unbedingt empfehlenswert ist. Mit den automatischen Optionen gibt es einige Probleme. Mit SKEWONLY Histogramme nur auf Spalten mit einer schiefen Datenverteilung zu erstellen klingt erstmal sehr sinnvoll, da ja auch nur hier Histogramme wirklich notwendig sind. Leider wird vergessen, dass hierbei auch unter Umständen Histogramme auf Spalten gesammelt werden, die zwar eine schiefe Datenverteilung haben, aber in keinerlei Abfragen als Prädikat auftauchen. Außerdem impliziert die Option, dass die Datenbank bereits Informationen über die Datenverteilung sammelt, um die Entscheidung treffen zu können. Hier beißt sich die Katze sozusagen in den Schwanz: Um zu entscheiden, ob Informationen über die Datenverteilung gesammelt und gespeichert werden sollen, müssen Informationen über die Datenverteilung gesammelt werden. Hier ist also in jedem Fall ein Aufwand erforderlich.

Die Option AUTO versucht den zuerst genannten Nachteil von SKEWONLY, dass unter Umständen auf zu vielen Spalten Histogramme erstellt werden, zu eliminieren, indem zusätzlich Informationen über den Workload auf den entsprechenden Spalten einbezogen werden und nur Spalten in Betracht gezogen werden, die auch in SQL-Abfragen als Prädikat auftauchen. Für diese Entscheidung werden Informationen aus der Tabelle SYS.COL_USAGE$ herangezogen. Leider sind dort aber nur Daten vorhanden wenn das Monitoring aktiviert ist und bereits Abfragen auf den Daten ausgeführt wurden. Die Erstellung der Histogramme hinkt als in jedem Fall hinterher und wenn die Monitoring-Daten gelöscht werden, beispielsweise weil sie bei einer Migration auf ein anderes System nicht mitgenommen werden, verschwinden bei der nächsten Ausführung des Statistik-Jobs automatisch auch die Histogramme.

Es besteht auch die Möglichkeit in besonders kniffligen Fällen die Histogramme nicht von der Datenbank erstellen zu lassen sondern selber zu bestimmen, sei es hart-kodiert oder nach einem eigenen Algorithmus. Folgend findet sich ein Beispielcode zur Erstellung eines Frequenzhistogramms:

```
declare
  m_distcnt number;
  srec dbms_stats.statrec;
  n_array dbms_stats.numarray;
```

```
begin
  n_array := dbms_stats.numarray( 9, 14, 15, 34, 35, 36, 43, 67, 84, 98);
  srec.bkvals := dbms_stats.chararray('Blau', 'Braun', 'Gelb', 'Grau',
'Grün','Lila', 'Rot', 'Schwarz', 'Silber', 'Weiß');
  srec.epc := 5;

  dbms_stats.prepare_column_values(srec, n_array);
  dbms_stats.set_column_stats(
    ownname => user,
    tabname => 'HISTOGRAMM_TEST',
    colname => 'FARBE',
    distcnt => 98,
    density => 0.005,
    nullcnt => 0,
    srec => srec,
    avgclen => 7
  );
end;
```

Auf welchen Spalten sollte man Histogramme erstellen?

Natürliche gilt keine Faustregel für jeden Fall, allerdings gibt es doch einige allgemeingültige Regeln, die einem helfen zu entscheiden, auf welchen Spalten man Histogramme erstellen sollte und auf welchen nicht. Diese Entscheidung ist wichtig, da ein Erstellen auf allen Spalten einen erheblichen Overhead mit sich bringen würde und auch nicht notwendig ist.

Grundsätzlich gilt, dass Histogramme nur auf Spalten mit einer schiefen Datenverteilung notwendig sind, da bei (fast) gleichmäßig verteilten Daten, die Näherung über die Formel 1 / NDV ausreichend genau ist. Eine konkrete Empfehlung für eine Grenze ab welcher Schiefe Histogramme zu erstellen sind, kann hier allerdings nicht gegeben werden, da es auch immer vom konkreten Fall abhängig ist wieviel hier verkraftbar ist.

Häufig hört man auch die Empfehlung Histogramme auf allen indizierten Spalten anzulegen. Während es einerseits ein guter Ansatz ist bei den Spalten mit Index seine Suche zu beginnen, da diese wohl auch in Prädikaten auftauchen werden, läuft man andererseits direkt in zwei Fallen, wenn man einfach pauschal nur die indizierten Spalten einbezieht. Erstens werden so auch Histogramme da gesammelt, wo eventuell gar keine schiefe Datenverteilung vorliegt (s.o.). Dies gilt insbesondere für Primärschlüssel und Unique-Indizes, da hier per Definition jeder Wert genau einmal vorkommt. Zweitens fallen Spalten durchs Raster, die zwar als Prädikat in Abfragen auftauchen, auf denen aber kein Index liegt. Dies kann zum Beispiel der Fall sein, wenn ein Index aufgrund einer geringen Selektivität keine Performance-Vorteile bringen würde.

Histogramme und Bind-Variablen

Bind-Variablen in SQL-Abfragen und Histogramme harmonieren nicht gut miteinander. Der Grund ist, dass bei der ersten Ausführung der Abfrage der für die in diesem Moment übergebenen Parameter optimale Ausführungsplan im Cache gespeichert wird. Schickt man die Abfrage ein zweites Mal ab mit ganz anderen Parametern für die Bind-Variablen, so wird dieser Plan wiederverwendet. Filtert man die Daten im ersten Fall beispielsweise nach einem sehr seltenen Wert, während man im zweiten einen häufig vorkommenden Wert suchen, wird jener gecachete Ausführungsplan allerdings mit hoher

Wahrscheinlichkeit nicht mehr der optimale sein und im schlimmsten Fall sogar zu einer sehr schlechten Laufzeit des Statements führen. Eigentlich müsste also ein neuer Ausführungsplan berechnet werden und das adaptive cursor sharing in 11g und die adaptive execution plans in 12c schaffen hier auch deutlich Abhilfe, wenn auch das zugrunde liegende Problem bestehen bleibt.

Was bedeutet dies alles im DWH-Umfeld? Nun, der Hauptgrund für die Benutzung von Bind-Variablen besteht darin das erneute Parsen des SQL und den damit verbundenen Overhead zu vermeiden. In OLTP-Systemen, in denen Abfragen mit geringer Laufzeit relativ häufig ausgeführt werden, ist dieser Overhead (parse storms) unter Umständen erheblich. Im DWH dagegen hat man zumindest im ETL-Bereich Abfragen mit langer Laufzeit, jedoch vergleichsweise geringer Frequenz. Im Reporting-Bereich sieht die Situation natürlich nicht mehr ganz so eindeutig aus, aber im Normalfall sollten auch hier hard parses zu verkraften sein. Eine Empfehlung wäre daher bei Spalten mit Histogrammen ggf. auf die Bind-Variablen zu verzichten, um die Caching-Problematik zu umgehen. Dabei sollte allerdings darauf geachtet werden, dass durch andere Mechanismen sichergestellt wird, dass hierdurch kein Einschleusen von SQL-Code (SQL Injection) ermöglicht wird. Beim Einsatz von Standard-Reporting-Software ist dies aber normalerweise gewährleistet.

Dauer der Erstellung

Hat einem die Erstellung eines Histogramms einmal bei einem Performance-Problem aus der Klemme geholfen, könnte man in Versuchung kommen diese (vorbeugend) auf möglichst vielen Spalten erstellen zu lassen. Doch leider ist dies eine sehr aufwändige Operation, da für jedes Histogramm eine SQL-Abfrage mit einer Aggregation durchgeführt werden muss und das – je nach Tabelle und Sample-Größe – über ein erhebliches Datenvolumen. Man sollte also eher wählerisch sein auf welchen Spalten man Histogramme anlegt. Einige allgemeine Tipps bezüglich dieser Entscheidung wurden bereits gegeben. Man hat auch die Möglichkeit die Sample-Größe niedriger anzusetzen, um etwas Zeit zu sparen, allerdings führt das wiederum dazu, dass die im nächsten Unterkapitel beschriebene Instabilität mit höherer Wahrscheinlichkeit auftritt.

Instabilität aufgrund des Samples

Beim Kapitel zur Beschreibung der Histogramme wurde die Annahme getroffen, dass bei der Berechnung der Histogramme alle Datensätze der Tabelle herangezogen werden und diese somit die Datenverteilung 100% genau widerspiegeln (den Auflösungsverlust bei den Höhenbalancierten Histogrammen mal außen vor gelassen). Aufgrund des Aufwands dieses Vorgehens ist das natürlich unrealistisch. Im Normalfall wird nur eine Stichprobe (Sample) von beispielsweise 0,1% der Datensätze betrachtet. Man erhält also nur Näherungswerte. Dies ist meistens kein großes Problem und führt normalerweise weiterhin zum korrekten Ausführungsplan. Die Aussage ”meistens” und ”normalerweise” heißt aber auch, dass, wenn man Pech hat und die Stichprobe an einem Tag sehr ungünstig ist, sich die Ausführungspläne recht zufällig ändern können. Es kann passieren, dass seltene Werte gar nicht in der Stichprobe enthalten sind, was dann dazu führen würde, dass bei der Filterung auf diesen Wert für die Selektivitätsschätzung die Regel ”Hälfte der Frequenz des seltensten Wertes”(s.o.) angewendet wird, was dann je nach Datenkonstellation eine deutliche Fehleinschätzung darstellen kann. Leider kann man diese Gefahr der Instabilität im DWH nicht komplett vermeiden. Sollten noch ausreichend Ressourcen innerhalb des Verarbeitungsfensters vorhanden sein, kann man natürlich die Sample-Größe erhöhen, um die Wahrscheinlichkeit des Auftretens zu verringern. Ansonsten bleibt einem nur übrig Tabellen, wo die Gefahr besonders hoch ist zu identifizieren und hier ggf. Workarounds zu finden. Dies kann die bereits erwähnte virtuelle Spalte sein oder gar ein

manuelles Erstellen des Histogramms, das sicherstellt, dass die seltenen Werte enthalten sind. Ein Patentrezept gibt es hier leider nicht.

Der richtige Zeitpunkt für die Generierung

Das beste Histogramm nützt wenig, wenn es veraltet ist, weil seit dem Zeitpunkt von dessen Erstellung sich die zugrundeliegende Datenkonstellation geändert hat. Der automatische Statistik-Job, der Statistiken sammelt, wenn 10% der Datensätze verändert wurden, ist im DWH denkbar ungeeignet. Man hat hier viele Tabellen, die mit einer TRUNCATE/INSERT Ladelogik befüllt werden. Der worst case wäre hier wohl, wenn der automatische Job direkt nach dem TRUNCATE die Statistiken auf der leeren Tabelle sammelt, die dann nach der Befüllung völlig unbrauchbar sind. Aber auch falls man hier Glück haben sollte ergeben sich Probleme bei den stetig wachsenden Tabellen im DWH. Da das Datenvolumen über die Zeit wächst, bedeutet das gleichzeitig, dass der automatische Job auch immer seltener ausgeführt wird und recht bald auch zu selten. Man hat allerdings im DWH den entscheidenden Vorteil, dass sofern es keinen Real-Time-Teil gibt Änderungen an den Tabellen immer genau zum Zeitpunkt ETL-Prozesse stattfinden und danach nur Leseoperationen. Es gibt also glücklicherweise den optimalen Zeitpunkt zur Generierung der Statistiken, nämlich genau jeweils nach der Befüllung der Tabelle im ETL-Prozess. Bei einem Real-Time-DWH oder einem DWH mit Real-Time-Anteilen ist die Sache nicht ganz so einfach. Hier muss man sein System genau kennen, um zu wissen zu welcher Zeit Daten kommen, die die Datenkonstellation entscheidend beeinflussen und folglich zu welcher Zeit ein Eingriff an den Statistiken notwendig ist. Das kann einerseits bedeuten die Statistiken immer zu einer bestimmten Uhrzeit generieren zu lassen, andererseits aber auch ggf.- durch manuelle Manipulationen an den Histogrammen Nachjustierungen vorzunehmen.

Histogramme über mehrere Spalten

Bisher wurde immer jede Spalte für sich betrachtet. In einem Prädikat werden aber auch Einschränkungen über logische Operatoren verknüpft, die sich somit auf mehrere Spalten beziehen. Um bei solchen Kombinationen die Selektivität zu berechnen reichen im Groben folgende drei Formeln:

- selectivity(a ^ b) = selectivity(a) * selectivity(b)
- selectivity(a v b) = selectivity(a) + selectivity(b) - selectivity(a ^ b)
- selectivity(not a) = 1 - selectivity(a)

Der große Nachteil der ersten beiden Formeln, ist dass diese davon ausgehen, dass die Spalten nicht miteinander korrelieren. Gerade in den Data-Marts hat man allerdings oftmals korrelierende Spalten, auf die gefiltert wird, insbesondere bei hierarchischen Dimensionen. Bei diesen besteht eine Korrelation zwischen den unterschiedlichen Hierarchie-Ebenen. Habe ich beispielsweise in der Produktdimension eine Produktgruppe Obst, so fällt diese unter Annahme einer einfachen Hierarchie immer in genau eine Produktkategorie, in diesem Beispiel Lebensmittel. Tatsächlich hat also eine Erweiterung eines Filter-Prädikats, das bereits auf Produktgruppe filtert, auf die Produktkategorie keinerlei Änderung der tatsächlichen Selektivität zu Folge. Der CBO multipliziert in diesem Fall aufgrund der obigen Formel jedoch die Selektivität nochmals und verringert diese dadurch fälschlicherweise (und unter Umständen erheblich).

Um den Optimizer in diesem Fall auf die richtige Fährte zu führen, benötigt man nicht etwa Histogramme auf den Spalten Produktkategorie und Produktgruppe sondern ein Histogramme über alle Wertekombinationen der beiden Spalten, also gewissermaßen ein Histogramm über beide Spalten. Dies ist seit Version 11g über die sog. extended statistics möglich:

```
declare
  vResult varchar2(4000);
begin
  vResult := dbms_stats.create_extended_stats(
    ownname => 'MART',
    tabname => 'DIM_PRODUKT',
    extension => '(PRODUKTGRUPPE, PRODUKTKATEGORIE)'
  );
  dbms_stats.gather_table_stats(
    ownname => 'MART',
    tabname => 'DIM_PRODUKT',
    method_opt => 'for columns (PRODUKTGRUPPE, PRODUKTKATEGORIE)'
  );
end;
```

Fazit

Wenn die Verteilung von Werten in einer Spalte schief ist, müssen Maßnahmen ergriffen werden, damit der Optimizer nicht in Folge dessen sehr ungünstige Ausführungspläne generiert. Hat man die Möglichkeit den SQL-Code zu ändern, können virtuelle Spalten oder funktionsbasierte Indizes ein Mittel sein, um hier gegenzusteuern. Anderenfalls ist man auf Histogramme angewiesen. Dabei sollte man beachten, dass Bind-Variablen Probleme bereiten können, selbst mit den neueren Features von 11g und 12c wie adaptive cursor sharing und adaptive execution plans.

Auch wenn Histogramme sehr nützlich sein können, haben sie doch den großen Nachteil, dass ihre Erstellung recht ressourcenintensiv ist, wenn man eine ausreichend große Stichprobe betrachtet, um Instabilität zu vermeiden. Aber selbst bei einem Scan der kompletten Datensätze einer Tabelle, ist es wichtig, dass die Histogrammerstellung zum richtigen Zeitpunkt geschieht, wenn man nicht in Probleme laufen möchte, aufgrund der Art wie der Optimizer fehlende Werte handhabt.

Kontaktadresse:
Philipp Krätzig
areto consulting GmbH
Julius-Bau-Straße 2
D-51063 Köln

Telefon:	+49 (0) 221-669 575-0
Fax:	+49 (0) 221-669 575-99
E-Mail	philipp.kraetzig@areto-consulting.de
Internet:	areto-consulting.de