ORACLE 11G DATABASE STATISTICS – MULTICOLUMN STATISTICS

Inderpal S. Johal, Data Softech Inc.

OVERVIEW

Prior to Oracle 11g, optimizer utilizes the statistics of all the columns involved in Complex predicate independent of each other. So it will estimate the selectivity by multiplying the selectivity of each column involved in the complex predicate. This method of selectivity is not good in cases where two or more columns are correlated to each other.

In Oracle 11g, we can now create a virtual column based on the relation between two or more columns of the predicate. Below are few examples

- Country and State columns are highly correlated in Customers Tables
  - USA and NJ where NJ almost uniquely determine USA
  - India and New Delhi
- Make and Model columns are highly correlated in Car_sales table
  - LEXUS and ES300 where ES300 almost uniquely determine LEXUS
  - NISSAN and Maxima
- Season and Products are Highly correlated in Sales table
  - Swimsuit and Summer where Swimsuit are available only in Summertime in Stores
  - Snow boot and Winter
- BirthDate and Zodiac Sign are Highly correlated in Astro table
  - January and Capricorn where Capricorn is the Zodiac Sign for Dec21-Jan19 born people.

You can create this new Correlated Virtual Column using dbms_stats.create_extended_stats procedure. Oracle 11g allows you to create the statistics on newly create Virtual Column using standard DBMS_STATS procedure which can then be utilized by query optimizer to make the correct estimate.

You can check all these extended statistics information from

`ALL | DBA | USER_STAT_EXTENSIONS` views.

You can delete the extended Statistics using **dbms_stats.drop_extended_stats** procedure.

How Multicolumn Statistics Works

Query Results with No Statistics

Execute the Query to see the number of Records in the Table

```
SQL> select count(*) from indy_test where country='INDIA' and state='Punjab';
  COUNT(*)
----------
       380
```

Make sure that there is no Statistics available on the table INDY_TEST

```
SQL> select column_name, num_distinct, histogram
  2  from user_tab_col_statistics
  3  where table_name = 'INDY_TEST'
  4  /
no rows selected
```

Create the Explain Plan to see the Query Optimizer Estimates

```
SQL> explain plan for
  2  select *
  3  from indy_test where country = 'INDIA' and state = 'Punjab';
Explained.
```

Optimizer Estimates 109 Rows which is not correct and so we will now collect the Statistics in the Next Step

```
SQL> select plan_table_output
  2  from table(dbms_xplan.display('plan_table',null,'BASIC ROWS'));
PLAN_TABLE_OUTPUT
--------------------------------------------
| Id  | Operation          | Name       | Rows  |
--------------------------------------------
|   0 | SELECT STATEMENT   |            |   109 |
|   1 |  TABLE ACCESS FULL| INDY_TEST |   109 |
--------------------------------------------
7 rows selected.
```

QUERY RESULTS WITH OPTIMIZER STATISTICS COLLECTED

Collect the Optimizer Statistics using DBMS_STATS package

```
SQL> exec dbms_stats.gather_table_stats(null,'indy_test',method_opt => 'for all columns
size skewonly') ;
```

PL/SQL procedure successfully completed.

Create the Explain Plan to see the Query Optimizer Estimates

```
SQL> explain plan for
  2  select *
  3  from indy_test where country = 'INDIA' and state = 'Punjab';
```

Explained.

Optimizer Estimates 4 Rows which is not correct and so we will now create the new Virtual Column which will be combination of COUNTRY and STATE column Statistics in the Next Step

```
SQL> select plan_table_output
  2  from table(dbms_xplan.display('plan_table',null,'BASIC ROWS'));
```

```
PLAN_TABLE_OUTPUT
--------------------------------------------
| Id  | Operation          | Name       | Rows  |
--------------------------------------------
|   0 | SELECT STATEMENT   |            |     4 |
|   1 |  TABLE ACCESS FULL| INDY_TEST  |     4 |
--------------------------------------------
```

7 rows selected.

QUERY RESULTS WITH EXTENDED VIRTUAL COLUMN AND STATISTICS RECOLLECTED

Create the Virtual Column using CREATE_EXTENDED_STATS procedure for COUNTRY and STATE column of INDY_TEST table. It will add new Extended column named "SYS_STUCOKS02#R98PV7LE6CJQ9INA" as shown below

```
SQL> select    dbms_stats.create_extended_stats(null,'indy_test','(country,state)')from
dual;
DBMS_STATS.CREATE_EXTENDED_STATS(NULL,'INDY_TEST','(COUNTRY,STATE)')
--------------------------------------------------------------------------------
SYS_STUCOKS02#R98PV7LE6CJQ9INA
```

Collect the Optimizer Statistics using DBMS_STATS package. This will also select the statistics for newly created Extended virtual column.

```
SQL> exec dbms_stats.gather_table_stats(null,'indy_test',method_opt => 'for all columns
size skewonly for columns(country,state) size skewonly');
PL/SQL procedure successfully completed.
```

You can see the new Virtual column below with Stats collected for Optimizer

```
SQL> select column_name, num_distinct, histogram
  2  from user_tab_col_statistics
  3  where table_name = 'INDY_TEST'
  4  /
COLUMN_NAME                    NUM_DISTINCT HISTOGRAM
------------------------------ ------------ ---------------
USERID                               161262 HEIGHT BALANCED
USERNAME                             161262 HEIGHT BALANCED
USERSTATUSID                              4 FREQUENCY
COUNTRY                                  34 FREQUENCY
STATE                                    47 FREQUENCY
SYS_STUCOKS02#R98PV7LE6CJQ9INA           47 FREQUENCY
6 rows selected.
```

Create the Explain Plan to see the Query Optimizer Estimates

```
SQL> explain plan for
  2  select *
  3  from indy_test where country = 'INDIA' and state = 'Punjab';
Explained.
```

Optimizer Estimates 380 Rows, which is correct estimate

```
SQL> select plan_table_output
  2  from table(dbms_xplan.display('plan_table',null,'BASIC ROWS'));

PLAN_TABLE_OUTPUT
--------------------------------------------
| Id  | Operation          | Name       | Rows  |
--------------------------------------------
|   0 | SELECT STATEMENT   |            |   380 |
|   1 |  TABLE ACCESS FULL| INDY_TEST |   380 |
--------------------------------------------

7 rows selected.
```

DROP THE EXTENDED VIRTUAL COLUMN

Find out the Extended Virtual Column details from USER_STAT_EXTENSIONS

```
SQL> select table_name,extension from user_stat_extensions where
table_name='INDY_TEST';
TABLE_NAME                     EXTENSION
------------------------------ ----------------------------------------
INDY_TEST                      ("COUNTRY","STATE")
```

Drop the Extended Stats column

```
SQL> exec dbms_stats.drop_extended_stats(null,'indy_test','(country,state)');
PL/SQL procedure successfully completed.
```

Check if the Extended Column still exists

```
SQL> select table_name,extension from user_stat_extensions where
table_name='INDY_TEST';
no rows selected
```