
THE INFORMATION COMPANY
ORACLE

Andrew Holdsworth
Director of Real World Performance
Mohamed Ziauddin
Hal Takahara
Sunil Chakkappen
Optimizer Development Team

ORACLE

A Practical Approach to Optimizer Statistics in Oracle Database 10g

ORACLE

Agenda

- Background to the subject of Optimizer Statistics
- Proposed methodology
- The General Solution
- Refining The Gathering Process
- Dealing with Exceptions
- Rule to Cost Based Migrations
- Dealing with Volatile Tables
- Questions & Answers

Background to the Subject of Optimizer Statistics

- The need for a methodology
- Sorry DBAs, it will mean more work
- Best practices are still evolving
- The payback from good statistics management and execution plans will exceed any benefit of init.ora tuning by ***orders of magnitude***

Proposed Methodology 2005

- Start with the General Solution (90% of SQL)
- Refine the method based upon actual data shape and usage (next 9% of SQL)
- Correct the last queries (remaining 1% of SQL)
- Data Dictionary tables also need statistics from 10g onwards

The General Solution

- Build Statistics using the following method:
  - Use dbms_stats package not analyze command
  - In 9i

```
DBMS_STATS.gather_schema_stats(
ownname=>'<schema>',
estimate_percent=>dbms_stats.auto_sample_size
cascade=>TRUE,
method_opt=>'FOR ALL COLUMNS SIZE AUTO')
```

  - In 10g statistics get gathered automatically

Refining/Adapting the Gathering Process

- When to gather statistics ?
- Dealing with evolving data shapes and sizes
  - Choosing Sampling Sizes and Types
  - DML monitoring at runtime
  - Column Usage History Information
  - Partitioned Objects Issues

When to Gather Statistics

- Candidates
  - After large amounts of data change (loads, purges, bulk updates)
  - New high/low values for keys generated
  - Newly created tables
  - Upgrading CPUs, I/O subsystem (system statistics)
  - RBO to CBO migrations
  - New database creations
- Apply common sense to rebuild frequency

Sample Sizes

- Default recommendation
  - `estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE`
  - This is great for uniform sizes and distributions of data
  - Issues can occur with skewed data or if there are lot of nulls in the column
- Smaller sample size for large objects
  - Should save time
  - Row sampling (Full scan of data sampling rows)
  - Block sampling (Reduced I/O)
  - Same issues as for **`AUTO_SAMPLE_SIZE`**

DML Monitoring

- Used by **`dbms_stats`** to identify objects with "stale" statistics
- On by default in 10g, not in 9i
  `alter table <table_name> monitoring;`
- Tracked in [DBA|ALL|USER]_TAB_MODIFICATIONS
- 9i and 10g use a 10% change as the threshold to recalculate stats

DML Monitoring Example

```
SQL> select table_name, partition_name, inserts, updates, deletes
     from all_tab_modifications;

TABLE_NAME   PARTITION_NAME     INSERTS    UPDATES    DELETES
------------ --------------- ---------- ---------- ----------
USERS        USERS14                  0       3328          0
ACCOUNT      ACCOUNT23            23450        738        230
EMP          EMP19                 2409        390          0
```

Column Usage History

- Used by `dbms_stats` to identify candidate columns on which to build histograms
- Column Usage Dictionary Table
- Lets Talk About About Histograms

Column Usage Dictionary Table

- Tracked in `sys.col_usage$`
- Captures predicate column name and access method type
- Used by DBMS_STATS to gather histograms when `method_opt=>'for … size auto'`
- Data is persisted after database shutdown
- This is why the stats built after running SQL can change/evolve

Column Usage Example

TAB_NAME	COLUMN_NAME	EQUALITY	EQUIJOIN	NONEQUIJOIN	RANGE	LIKE	NULL
ACCOUNT	ORGANIZATION_ID	26	0	0	0	0	2
ACCOUNT	ACCOUNT_ID	26	25	0	0	0	1
ACCOUNT	OWNER	0	25	0	0	0	1
ACCOUNT	DELETED	26	0	0	0	0	1
ACC_SHARE	ORGANIZATION_ID	27	0	0	0	0	2
ACC_SHARE	ACCOUNT_ID	0	25	0	0	0	2
ACC_SHARE	UG_ID	0	26	0	0	0	2
ACC_SHARE	DELETED	27	0	0	0	0	2

* SQL for query is in slide notes

Lets Talk About Histograms

- Generally help with skewed data
- Should be used with literals in statements
- `method_opt =>`
  - `'for all columns'` defaults to 75 buckets
  - `'for all columns size auto'` defaults to 254 buckets **
- Issues can arise when there are many unique values with skewed data

Partitioned Objects

- The option for local statistics exists to allow partition level optimization
- Issues arise when running with large numbers of partitions/sub partitions > 1000
- Dropping these objects/stats takes too long to be practical
- Examples include the composite range/hash partitioned table of 36 months x 32 hash subpartitions (1152 total partitions)
- Exchanging partitions in/out may/may not invoke stats gathering.

Dealing with the Exceptions

- Init.ora parameters
- Hints
- SQL profiles
- Tough Decisions
- Things statistics cannot deduce

Init.ora Parameters

- The most common forms of abuse
  - optimizer_index_caching
  - optimizer_index_cost_adj
  - db_file_multiblock_read_count
- The issue here is moving forward and impact on other applications

Use of Hints

- Used to fine tune queries
  - Incorporate application knowledge not possible from the collected stats
  - Dealing with impossible complex SQL
  - Dealing with volatile or dynamic tables
  - Dealing with and working around bugs
  - Dealing with column correlations
- Again the issue is moving forward to new releases

Use of SQL profiles

- Contains tuning information used by the optimizer
- Introduced in 10g – **`dbms_sqltune`**
- More exhaustive analysis than CBO alone
- Store SQL profile to get tuned plan for future executions

Managing Statistics

- In 9i use **DBMS_STATS.[EXPORT|IMPORT]_*_STATS** procedures to back up and restore statistics or to move the statistics between databases
- In 10g
  - Use **DBMS_STATS.[EXPORT|IMPORT]_*_STATS** procedures to move statistics between databases or to experiment with different sets of statistics
  - Use **DBMS_STATS.RESTORE_*_STATS** procedures to recover older versions of statistics. There are also procedures to manage the retention and purging of statistics histories.
- Controlling when Statistics get re-built
  - In 9i **DBMS_STATS** package to rebuild statistics
  - In 10g statistics gets gathered automatically.

  Use **DBMS_SCHEDULER.DISABLE('GATHER_STATS_JOB')** to disable automatic statistics gathering.

RBO to CBO Migration Issues

- The importance of testing and running of workload
- Dealing with very complicated schemas
  - Complicated SQL(Joins, RBO predicates, encode application knowledge)
  - Objects with large number of access paths

Dealing with Volatile Tables

- Basic Premise
  - Good plans with large data sets are better than sub-optimal plans with small datasets.
- Building Stats vs. Dynamic Sampling (leave table without stats but lock it with)
- Seeding Statistics (store representative table stats then lock it)
- Lock stats with **`dbms_stats.lock_table_stats`**

Detection when Good Statistics yield poor plans

- The most common reason for poor execution plans with perceived “good” statistics is inaccurate row count estimates
  - This leads to bad access path selection
  - This leads to bad join method selection
  - This leads to bad join order selection
- In summary one bad row count estimate can cascade into a very poor plan

Determining the Bad Source

- Basic Analysis:
  1. Is the rows estimate from explain plan correct?
  2. Is the rows estimate correct in all_[tab|part]_col_statistics?
  3. Is the sample size large enough to give accurate row estimates?
  4. Obtain accurate stats and recheck the plan

Bad Source Example

- Stats gathered with sample of dbms_stats.auto_sample_size
- Data is known to be skewed and have high number of distinct values(NDV)

Bad Source Example cont.

```
select a.name,
       a.account_id,
       a.site,
       a.address1_state,
       a.phone1,
       a.type_enum,
       u.alias,
       u.users_id,
       upper (a.name),
from   sales.account a,
       (select distinct s.account_id
        from            core.ug_blowout b,
                        sales.acc_share s
        where           s.organization_id = '00D3000000002jy' and
                        b.organization_id = '00D3000000002jy' and
                        b.users_id = '00530000000dQfh' and
                        s.ug_id = b.ug_id and
                        s.deleted = '0') t,
       core.users u
where  (t.account_id = a.account_id) and
       (u.users_id = a.owner) and
       (a.account_id != '000000000000000') and
       (a.organization_id = '00D3000000002jy' and
        a.deleted = '0') and
       (u.organization_id = '00D3000000002jy')
```

Bad Source Example cont.

Original Plan from v$sql_plan_statistics_all

Query time - Elapsed: 00:02:51.05

Statistics from v$sql_plan_statistics_all

CARDINALITY = optimizer row estimate
LAST_OUTPUT_ROWS = actual row count

OPERATION	OBJECT_NAME	CARDINALITY	LAST_OUTPUT_ROWS
VIEW		21,249	1,087
SORT		21,249	1,087
HASH JOIN		21,249	2,172
TABLE ACCESS USERS	USERS	625	3,326
INDEX IEUSERSACTIVE	IEUSERSACTIVE	1	3,326
HASH JOIN		63,120	2,172
HASH JOIN		293,480	2,172
INDEX AKUG_BLOWOUT	AKUG_BLOWOUT	73	13
INDEX IEACC_SHARE_DIVISION	IEACC_SHARE_DIVISION	3,899,279	14,030,696
TABLE ACCESS ACCOUNT	ACCOUNT	370,639	1,609,704

* SQL for above query is in slide notes

Bad Source Example cont.

- What was wrong?
  - Row estimates are off causing wrong plan
  - Comparison of all_part_col_statistics and actual distinct values showed the NDV estimate was significantly off due to data skew
- How to fix it (choose one)?
  - Re-gather stats with compute or a high enough sample to give good estimates
  - Use dbms_stats.set_column_stats to set the column distcnt correctly

Bad Source Example cont.
Bad Sample Estimate Comparison

Numbers taken from all_part_col_statistics and "count(distinct <column_name>)" for given partition

TABLE_NAME	COLUMN_NAME	ESTIMATED_NDV	ACTUAL_NDV
ACCOUNT	ORGANIZATION_ID	325	1,907
ACCOUNT	ACCOUNT_ID	2,952,246	2,963,757
ACCOUNT	OWNER	1,857	8,963
ACC_SHARE	ORGANIZATION_ID	71	306
ACC_SHARE	UG_ID	969	8,991
ACC_SHARE	DELETED	2	2
USERS	ORGANIZATION_ID	2,527	1,906
USERS	USERS_ID	2,985	27,342
UG_BLOWOUT	ORGANIZATION_ID	2,071	1,552
UG_BLOWOUT	USERS_ID	20,252	19,783

Bad Source Example cont.

New Plan from v$sql_plan_statistics_all

```
Query time - Elapsed: 00:00:00.61

Statistics from v$sql_plan_statistics_all

CARDINALITY = optimizer row estimate
LAST_OUTPUT_ROWS = actual row count
```

OPERATION	OBJECT_NAME	CARDINALITY	LAST_OUTPUT_ROWS
VIEW		1009	1087
SORT		1009	1087
NESTED LOOPS		2019	2172
NESTED LOOPS		2019	2172
NESTED LOOPS		2019	2172
INDEX AKUG_BLOWOUT	AKUG_BLOWOUT	6	13
TABLE ACCESS ACC_SHARE	ACC_SHARE	2019	2172
INDEX AK2ACC_SHARE	AK2ACC_SHARE	2041	2190
TABLE ACCESS ACCOUNT	ACCOUNT	2019	2172
INDEX PKACCOUNT	PKACCOUNT	2019	2172
TABLE ACCESS USERS	USERS	2019	2172
INDEX PKUSERS	PKUSERS	2019	2172


QUESTIONS
ANSWERS