Gathering Optimizer Statistics in Oracle Database

Commands and Scripts with Examples

Asfaw Gedamu Haileselasie

18/02/2024

A. Introduction:

This Standard Operating Procedure (SOP) outlines the process of gathering optimizer statistics in Oracle Database using various commands and tools. Accurate statistics are crucial for the optimizer to choose optimal execution plans, leading to improved query performance.

B. Purpose:

- Collect data about tables, indexes, and columns to estimate their size, cardinality, and distribution.
- Inform the optimizer about data characteristics, enabling it to select efficient execution plans.
- Improve query performance by reducing reliance on estimations and increasing plan accuracy.

C. Prerequisites:

- Access to an Oracle database instance.
- Understanding of statistics types (table, index, column) and their impact on query optimization.
- Privileges to run statistics gathering commands (e.g., `DBA`, `ANALYZE` role).

D. Gathering Methods:

- **ANALYZE Command:**
    - Basic method for gathering table and index statistics.
    - Use ANALYZE TABLE tableName [ALL INDEXES| <indexName>] [, VALIDATE STRUCTURE] [, ESTIMATE STATISTICS ONLY];
    - Consider drawbacks like performance impact and limited customization.
- **DBMS_STATS Package:**
    - Flexible and powerful PL/SQL package for advanced statistics gathering.

    - Offers various options for targeting specific tables, indexes, columns, and gathering levels.
    - Use DBMS_STATS.GATHER_TABLE_STATS, DBMS_STATS.GATHER_INDEX_STATS, etc.
    - Consult the documentation for detailed parameter options and best practices.
- **Automatic Statistics:**
    - Oracle automatically gathers statistics for new tables and indexes.
    - Configure automatic statistics collection with suitable parameters for your workloads.
    - Consider enabling AUTOSTATS or setting DBMS_STATS.AUTO_SAMPLE_SIZE appropriately.

E. Best Practices:

- Gather statistics regularly, especially after data changes or schema modifications.
- Analyze gathered statistics and identify potential issues like outdated or misleading data.
- Choose the appropriate gathering method based on your needs and performance impact considerations.
- Test the impact of statistics gathering on your environment before applying changes in production.
- Monitor statistics usage and update gathering strategies as needed.

Now, to the list of statistics gathering commands.

1. Gather dictionary stats:

-- It gathers statistics for dictionary schemas 'SYS', 'SYSTEM' and other internal schemas.

EXEC DBMS_STATS.gather_dictionary_stats;

2. Gather fixed object stats:

--- Fixed object means gv$ or v$views

EXEC DBMS_STATS.GATHER_FIXED_OBJECTS_STATS;

3. Gather full database stats:

EXEC DBMS_STATS.gather_database_stats;

-- With estimate_percent to 15 percent or any other value , if the db size very huge.

```
EXEC DBMS_STATS.gather_database_stats(estimate_percent => 15);
EXEC DBMS_STATS.gather_database_stats(estimate_percent => 15,
cascade => TRUE);
```

-- With auto sample size and parallel degree

```
EXEC DBMS_STATS.gather_database_stats(estimate_percent =>
DBMS_STATS.AUTO_SAMPLE_SIZE,degree => 8);
```

4. Gather schema statistics:

```
EXEC DBMS_STATS.gather_schema_stats('DBACLSS');

EXEC DBMS_STATS.gather_schema_stats('DBACLASS', estimate_percent
=> 25);

EXEC DBMS_STATS.gather_schema_stats('DBACLASS', estimate_percent
=> 100, cascade => TRUE);

-- STATS WITH AUTO ESTIMATION and degree 8

exec dbms_stats.gather_schema_stats( ownname =>
'DBACLASS',method_opt => 'FOR ALL COLUMNS SIZE 1',
granularity => 'ALL', degree => 8, cascade => TRUE,
estimate_percent=>dbms_stats.auto_sample_size);
```

5. Gather table statistics:

```
EXEC DBMS_STATS.gather_table_stats('DBACLASS', 'EMP');
EXEC DBMS_STATS.gather_table_stats('DBACLASS', 'EMP',
estimate_percent => 15);
EXEC DBMS_STATS.gather_table_stats('DBACLASS', 'EMP',
estimate_percent => 15, cascade => TRUE);

exec DBMS_STATS.GATHER_TABLE_STATS  (ownname => 'DBACLASS' ,
tabname => 'EMP',cascade => true,
method_opt=>'for all indexed columns size 1', granularity =>
'ALL', degree => 8);
```

```
exec DBMS_STATS.GATHER_TABLE_STATS (ownname => 'DBACLASS' ,
tabname => 'EMP',
cascade => true, method_opt=>'FOR ALL COLUMNS SIZE 1',
granularity => 'ALL', degree => 8);
```

6. Gather stats for single partition of a table using `DBMS_STATS` package:

```
BEGIN
DBMS_STATS.GATHER_TABLE_STATS (
ownname => 'SCOTT',
tabname => 'TEST', --- TABLE NAME
partname => 'TEST_JAN2016' --- PARTITOIN NAME
method_opt=>'for all indexed columns size 1',
GRANULARITY => 'APPROX_GLOBAL AND PARTITION',
degree => 8);
END;
/
```

-- Gather table statistics with auto-sampling for the "orders" table

```
BEGIN
  DBMS_STATS.GATHER_TABLE_STATS(
    ownname => 'schema',
    tabname => 'orders',
    method_opt => 'FOR ALL COLUMNS SIZE AUTO'
  );
END; /
```

-- Gather index statistics for the "idx_products_name" index

```
BEGIN
  DBMS_STATS.GATHER_INDEX_STATS(
    ownname => 'schema',
    tabname => 'products',
    indexname => 'idx_products_name'
  );
END; /
```

-- Gather column-level statistics for the "price" column in the "products" table

```
BEGIN
  DBMS_STATS.GATHER_TABLE_STATS(
    ownname => 'schema',
    tabname => 'products',
    method_opt => 'FOR COLUMNS size 10 price'
  );
END; /
```

6.1. Using ANALYZE Command

```
-- Gather statistics for all tables and indexes in the schema
"sales"
ANALYZE TABLE schema.sales.% INCLUDING INDEXES;

-- Gather statistics for a specific table "customers" and its
primary key index
ANALYZE TABLE schema.customers VALIDATE STRUCTURE ESTIMATE
STATISTICS ONLY;

-- Gather statistics for a specific index
"idx_orders_customer_id"
ANALYZE INDEX schema.orders_idx_customer_id;
```

7. Lock/unlock statistics:

-- Lock stats of a schema:
EXEC DBMS_STATS.lock_schema_stats('DBACLASS');

-- Lock stats of a table:
EXEC DBMS_STATS.lock_table_stats('DBACLASS', 'EMP');

-- Lock stats of a partition:
EXEC DBMS_STATS.lock_partition_stats('DBACLASS', 'EMP', 'EMP');

-- unlock stats of a schema:

EXEC DBMS_STATS.unlock_schema_stats('DBACLASS');
-- unlock stats of a table:

EXEC DBMS_STATS.unlock_table_stats('DBACLASS', 'DBACLASS');
--unlock stats of a partition:

EXEC DBMS_STATS.unlock_partition_stats('DBACLASS', 'EMP', 'TEST_JAN2016');

--- check stats status:

SELECT stattype_locked FROM dba_tab_statistics WHERE table_name = 'TEST' and owner = 'SCOTT';

8 . Delete statistics:

-- Delete complete db statistics:
EXEC DBMS_STATS.delete_database_stats;

-- Delete schema statistics:
EXEC DBMS_STATS.delete_schema_stats('DBACLASS');

-- Delete table statistics:
EXEC DBMS_STATS.delete_table_stats('DBACLASS', 'EMP');

-- Delete column statistics:
EXEC DBMS_STATS.delete_column_stats('DBACLASS', 'EMP', 'EMPNO');

-- Delete index statistics:

EXEC DBMS_STATS.delete_index_stats('DBACLASS', 'EMP_PK');

-- Delete dictionary statistics:
EXEC DBMS_STATS.delete_dictionary_stats;

-- Delete fixed object statistics:

exec dbms_stats.delete_fixed_objects_stats;

-- Delete system statistics:

exec dbms_stats.delete_system_stats('STAT_TAB');

8. Setting statistics preference:

-- View preference details for the database:

SELECT dbms_stats.get_prefs('PUBLISH') EST_PCT FROM dual;

-- View Publish preference for table

-- View Publish preference for schema:

select dbms_stats.get_prefs('PUBLISH', 'SCOTT') from dual

-- View preference details for table

```
select
dbms_stats.get_prefs(ownname=>'DBACLASS',tabname=>'EMP',pname=>'
PUBLISH') FROM DUAL;
select
DBMS_STATS.get_prefs(ownname=>'DBACLASS',tabname=>'EMP',pname=>'
INCREMENTAL') FROM DUAL;
select
DBMS_STATS.get_prefs(ownname=>'DBACLASS',tabname=>'EMP',pname=>'
GRANULARITY') FROM DUAL;
select
DBMS_STATS.get_prefs(ownname=>'DBACLASS',tabname=>'EMP',pname=>'
STALE_PERCENT')  FROM DUAL;
select
DBMS_STATS.get_prefs(ownname=>'DBACLASS',tabname=>'EMP',pname=>'
ESTIMATE_PERCENT')  FROM DUAL;
select
DBMS_STATS.get_prefs(ownname=>'DBACLASS',tabname=>'EMP',pname=>'
DEGREE')  FROM DUAL;
```

-- Set table preferences

```
exec
dbms_stats.set_table_prefs('DBACLASS','EMP','PUBLISH','FALSE');
exec
dbms_stats.set_table_prefs('DBACLASS','EMP','ESTIMATE_PERCENT','
20');
exec dbms_stats.set_table_prefs('DBACLASS','EMP','DEGREE','8');
```

-- Set schema preferences:

```
exec dbms_stats.SET_SCHEMA_PREFS('DBATEST','PUBLISH','FALSE');
exec
dbms_stats.SET_SCHEMA_PREFS('DBATEST','ESTIMATE_PERCENT','20');
exec dbms_stats.SET_SCHEMA_PREFS('DBATEST','CASCADE','TRUE');
```

-- Set database preference:

exec dbms_stats.set_database_prefs('PUBLISH', 'TRUE');
exec dbms_stats.set_database_prefs('DEGREE', '16');

-- Set global preference:

exec dbms_stats.set_global_prefs('PUBLISH', 'TRUE');
exec dbms_stats.set_global_prefs('DEGREE', '16');

9 . Deleting preferences :

-- Deleting schema preference:

exec dbms_stats.delete_schema_prefs('DBACLASS', 'DEGREE');
exec dbms_stats.delete_schema_prefs('DBACLASS', 'CASCADE');

-- Delete database preference:
exec dbms_stats.delete_database_prefs('ESTIMATE_PERCENT', FALSE);
exec dbms_stats.delete_database_prefs('DEGREE', FALSE);

10 . Publish pending statistics:

-- For schema DBACLASS
exec dbms_stats.publish_pending_stats('DBACLASS',null);

-- For table DBACLASS.EMP
EXEC DBMS_STATS.PUBLISH_PENDING_STATS ('DBACLASS','EMP');

11. Delete pending statistics:

-- for table DBACLASS.EMP

exec dbms_stats.delete_pending_stats('DBACLASS', 'EMP');

-- For schema DBACLASS
exec dbms_stats.delete_pending_stats('DBACLASS', null);

12. Upgrade stats table:

----- If we are importing stats table from higher version to lower version,
then before importing in the database, we need to upgrade the stats table.

EXECUTE DBMS_STATS.UPGRADE_STAT_TABLE(OWNNAME =>'RAJ',STATTAB =>'STAT_TEST');

13. View/modify statistics retention period:

-- View current stats retention

select dbms_stats.get_stats_history_retention from dual;

-- Modify the stats retention

exec DBMS_STATS.ALTER_STATS_HISTORY_RETENTION(60);

14. create stats table:

--- Create staging table to store the statistics data

exec dbms_stats.create_stat_table(ownname => 'SCOTT', stattab => 'STAT_BACKUP',tblspace=>'USERS');

15. Export stats data:

-- Export full database stats to a table SCOTT.STAT_BACKUP

exec dbms_stats.export_database_stats(statown => 'SCOTT' ,stattab=>'STAT_BACKUP');

-- Export stats for table DBACLASS.EMP to a stats table SCOTT.STAT_BACKUP

exec dbms_stats.export_table_stats(ownname=>'DBACLASS', tabname=>'EMP', statown =>'SCOTT',stattab=>'STAT_BACKUP', cascade=>true);

-- Export stats for schema DBACLASS to a stats table SCOTT.STAT_BACKUP

exec dbms_stats.export_schema_stats(ownname=>'DBACLASS', statown =>'SCOTT' , stattab=>'STAT_BACKUP');

-- Export fixed object stats to table SCOTT.STAT_BACKUP

exec dbms_stats.export_fixed_objects_stats(statown => 'SCOTT' ,stattab=>'STAT_BACKUP');

-- Export dictionary stats to table SCOTT.STAT_BACKUP

exec dbms_stats.export_dictionary_stats(statown => 'SCOTT' ,stattab=>'STAT_BACKUP');

-- Export stats for index DBACLAS.EMP_UK1 to SCOTT.STAT_BACKUP table

exec dbms_stats.export_index_stats(ownname=>'DBACLASS', indname=>'EMP_UK1', statown =>'SCOTT',stattab=>'STAT_BACKUP');

16. Import stats table data:

-- Import full database stats from stats table SCOTT.STAT_BACKUP

exec dbms_stats.import_database_stats(statown => 'SCOTT' ,stattab=>'STAT_BACKUP');

-- Import stats for table DBACLASS.EMP from stats table SCOTT.STAT_BACKUP

exec dbms_stats.import_table_stats(ownname=>'DBACLASS', tabname=>'EMP', statown =>'SCOTT',stattab=>'STAT_BACKUP', cascade=>true);

-- Import stats for schema DBACLASS from stats table SCOTT.STAT_BACKUP

exec dbms_stats.import_schema_stats(ownname=>'DBACLASS', statown =>'SCOTT' , stattab=>'STAT_BACKUP');

-- Import fixed object stats from stats table SCOTT.STAT_BACKUP

exec dbms_stats.import_fixed_objects_stats(statown => 'SCOTT' ,stattab=>'STAT_BACKUP');

-- Import dictionary stats from table SCOTT.STAT_BACKUP

exec dbms_stats.import_dictionary_stats(statown => 'SCOTT' ,stattab=>'STAT_BACKUP');

-- Import stats for index DBACLAS.EMP_UK1 from SCOTT.STAT_BACKUP table

exec dbms_stats.import_index_stats(ownname=>'DBACLASS', indname=>'EMP_UK1', statown =>'SCOTT',stattab=>'STAT_BACKUP');

17 . Few stats related sql queries:

-- Check stale stats for table:

```
select owner,table_name,STALE_STATS from dba_tab_statistics
where owner='&SCHEMA_NAME' and table_name='&TABLE_NAME';
```

--Check stale stats for index:

```
select owner,INDEX_NAME,TABLE_NAME from DBA_IND_STATISTICS where
owner='&SCHEMA_NAME' and index_name='&INDEX_NAME';
```

-- For getting history of TABLE statistics

```
setlines 200
col owner for a12
col table_name for a21
select owner,TABLE_NAME,STATS_UPDATE_TIME from
dba_tab_stats_history where table_name='&TABLE_NAME';
```

-- Space used to store statistic data in SYSAUX tablespace:

SQL> **select occupant_desc, space_usage_kbytes from v$sysaux_occupants where OCCUPANT_DESC like '%Statistics%';**

-- Check whether table stats locked or not:

select owner, table_name, stattype_locked from dba_tab_statistics where stattype_locked is not null and owner not in ('SYS','SYSTEM');

18. Automatic Statistics:

A. Enable automatic statistics gathering:

```
ALTER SYSTEM SET JOB_QUEUE_PROCESSES = <number_of_CPUs>;
EXEC DBMS_AUTO_TASK_ADMIN.ENABLE(client_name => 'auto optimizer
stats collection');
```

B. Set appropriate AUTO_SAMPLE_SIZE:

DBMS_STATS.SET_GLOBAL_PREFS(autostats_target => 'SAMPLE_SIZE(<value>)');

F. Conclusion:

By effectively gathering and maintaining accurate optimizer statistics, you can significantly improve query performance and resource utilization in your Oracle database. Remember to choose the right tools and approaches, considering your specific requirements and environment.

Additional Notes:

- This SOP provides a basic overview. Refer to the Oracle documentation for detailed information on individual commands, packages, and parameters.
- Consider including specific examples and best practices relevant to your database version and usage patterns for a more tailored guide.
- Regularly update the SOP to reflect changes in Oracle releases and statistics gathering techniques.