SQL Profiles: Technical Overview
SQL Profiles: Technical Overview

What is a SQL profile?

A SQL profile is a set of auxiliary information specific to a SQL statement. Conceptually, a SQL profile is to a SQL statement what statistics are to a table or index. The database can use the auxiliary information to improve execution plans.

A SQL profile contains corrections for poor optimizer estimates discovered by the SQL Tuning advisor. This information can improve optimizer cardinality and selectivity estimates, which in turn leads the optimizer to select better plans.

The SQL profile does not contain information about individual execution plans. Rather, the optimizer has the following sources of information when choosing plans:

- The environment, which contains the database configuration, bind variable values, optimizer statistics, data set, etc.
- The supplemental statistics in the SQL profile

Therefore, SQL profiles just guide the optimizer to a better plan.

What is the main use case for SQL profiles?

Automatic Database Diagnostic Monitor (ADDM) proactively identifies high-load SQL statements that are good candidates for SQL tuning. One of the recommendations can be to invoke a SQL Tuning Advisor on the high load SQL statements that ADDM has identified.

Also, it is possible to invoke SQL Tuning Advisor manually for on-demand tuning of one or more SQL statements. To tune multiple statements, one must create a SQL tuning set (STS). A SQL tuning set is a database object that stores SQL statements along with their execution context.

SQL Tuning Advisor can recommend a profile the following types of statements:

- DML statements (SELECT, INSERT with a SELECT clause, UPDATE, and DELETE)
- CREATE TABLE statements (only with the AS SELECT clause)
- MERGE statements (the update or insert operations)

If a profile is recommended it will be in a form of a finding that allows for the profile to be implemented. Once implemented, the database creates the profile and stores it persistently in the data dictionary. If a user issues a statement for which a profile has been built, then the query optimizer uses both the environment and the SQL profile to build a well-tuned plan.
What is the difference between SQL profiles, stored outlines, and SQL plan baselines?

To achieve SQL plan stability in earlier releases of Oracle Database stored outlines played a major role. In Oracle Database 11g they are still supported, however will be deprecated in the future releases in favor of SQL plan management. The goal of SQL plan baselines (that are generated by the SQL plan management mechanism) is to preserve the performance of corresponding SQL statements, regardless of changes in the database.

SQL Profiles, on the other hand, were introduced in Oracle Database 10g and were supposed to guide the Optimizer to a better plan. They do not guarantee the same plan each time the statement is parsed.

What are the benefits of SQL profiles?

- Unlike hints and stored outlines, profiles do not tie the optimizer to a specific plan or subplan. Profiles fix incorrect estimates while giving the optimizer the flexibility to pick the best plan in different situations.
- Unlike hints, no changes to application source code are necessary when using profiles.
- The use of SQL profiles by the database is transparent to the user.

What are enhancements to SQL profiles in Oracle Database 11g?

- **Automatic SQL Tuning**: The database can automatically tune SQL statements by identifying problematic SQL statements and implementing recommendations using SQL Tuning Advisor during system maintenance windows. If the performance improvement improves at least threefold, then the database accepts the SQL profile, but only if the ACCEPT_SQL_PROFILES task parameter is set to TRUE. Otherwise, the automatic SQL tuning reports merely report the recommendation to create a SQL profile. When run automatically, SQL Tuning Advisor is known as the Automatic SQL Tuning Advisor.

- **Test execution**: If a SQL profile is recommended, the database tests the new SQL profile by executing the SQL statement both with and without the SQL profile. Note, in Oracle Database 10g profiles were recommended based solely on optimizer estimates.

- **Parallel query profile recommendation**: Starting with Oracle Database 11g Release 2 SQL Tuning Advisor may recommend accepting a profile that uses the Automatic Degree of Parallelism (Auto DOP) feature. A parallel query profile is only recommended when the original plan is serial and when parallel execution can significantly reduce the elapsed time for a long-running query. When it recommends a profile that uses Auto DOP, SQL Tuning Advisor gives details about the performance overhead of using parallel execution for the SQL statement in the report. For parallel execution recommendations, SQL Tuning Advisor may provide two SQL profile recommendations, one using serial execution and one using parallel.
How do SQL profiles co-exist with user hints?

SQL profiles, when enabled, remove any user hints from the SQL statement. They can also inadvertently change optimizer configuration; for example, they will bring OPTIMIZER_FEATURES_ENABLE to the newest possible value, unless an older value is shown to be better through test-execution of the SQL statement.

Are there any recommended best practices for SQL profiles?

- **Transporting a SQL profile from test to production:** It is possible to transport SQL profiles across systems. This operation involves exporting the SQL profile from the SYS schema in one database to a pre-created staging table (DBMS_SQLTUNE.PACK_STGTAB_SQLPROF procedure), moving the staging table using the mechanism of choice (such as Oracle Data Pump or database link) and then importing the SQL profile from the staging table into another database (DBMS_SQLTUNE.UNPACK_STGTAB_SQLPROF procedure). You can transport a SQL profile to any Oracle database created in the same release or later. For details see this [white paper](http://otn.oracle.com) on OTN.

- **Testing SQL profiles on production in a private category:** SQL profiles have an attribute, the CATEGORY, associated with each one of them. This attribute determines what sessions are effected by the implemented profile. One can view the CATEGORY attribute by querying DBA_SQL_PROFILES.CATEGORY. By default, all profiles are in the DEFAULT category, which means that all sessions in which the SQLTUNE_CATEGORY initialization parameter is set to DEFAULT can use the profile. By altering the category of a SQL profile, one can specify which sessions are affected by profile creation. For example, by setting the category to DEV, only sessions in which the SQLTUNE_CATEGORY initialization parameter is set to DEV can use the profile. Other sessions do not have access to the SQL profile and execution plans for SQL statements are not impacted by the SQL profile. This technique enables you to test a profile in a restricted environment before making it available to other sessions.

- **Set FORCE_MATCHING parameter to 'yes' when creating SQL profiles:** this causes SQL Profiles to target all SQL statements which have the same text after normalizing all literal values to bind variables. If a combination of literal values and bind variables is used in the same SQL text, then no transformation occurs. This is analogous to the matching algorithm use by the FORCE option of the CURSOR_SHARING parameter.