Inside the Oracle Database 11g Optimizer
Removing the black magic

Hermann Bär
Oracle USA
Redwood Shores, CA

Schlüsselworte:

Oracle Database 11g, Oracle Optimizer, Plan Stability, SQL Plan Management

Introduction

The purpose of the optimizer is to determine the most efficient execution plan for your queries. It makes these decisions based on the structure of the query, the statistical information it has about your data, and by leveraging Oracle database features. The optimizer is expected to generate the best plan all the time, but sometimes the fundamental information for making the right decision is not provided to the Optimizer.

This paper aims to introduce functionality that was added specifically in Oracle Database 11g to ensure that the Optimizer gets the most accurate information for making the best decision at the right point in time. It furthermore introduces SQL Plan Management, a built-in technology that enables plan stability with controlled plan evolution, a powerful technique for upgrading older database releases to Oracle Database 11g.

Due to the shortage of this paper, there is not sufficient space to discuss all the functionality in great depth, so it will focus on introducing the functionality and provide additional pointers for further information.

Statistics Collection – Foundation for Success

Providing the Optimizer the right information at the right time is the foundation for determining the best execution plan. This implies two things to happen:

- Collect statistics fast and efficient
- Collect the appropriate statistics

Oracle Database 11g has come a long way to provide a simple and comprehensive framework for enabling exactly that.

A **new hash-based sampling algorithm** replaces the former expensive sort operation of the statistics collection for tables and partitions. This new algorithm achieves the accuracy of a compute statistics (equivalent to scanning the whole object) with the speed of an „old-style“ statistics collection of a 10%+/- sample size. Note that this new algorithm is only activated when choosing `AUTO_SAMPLE_SIZE` as `estimate_percent` for the statistics collection (which happens to be the default).

The statistics collection for partitioned tables consists of gathering statistics for both the partition and table level. Prior to Oracle Database 11g, adding a new partition or modifying data in a few partitions required scanning all partitions (the entire table) to refresh table-level statistics. Scanning the entire table can be very expensive as partitioned tables are generally very large; furthermore, the statistics collection time increased with the size of a partitioned table. In Oracle Database 11g this issue has been addressed with the introduction of **incremental global statistics**. If the `INCREMENTAL` preference for a partitioned table is set to `TRUE`, and the `DBMS_STATS GRANULARITY` parameter is set to `AUTO`, Oracle will gather statistics on the new partition and update the global table statistics by scanning only those partitions that have been added or modified and not the entire table.

Very often with partitioned tables, a new empty partition will be added to the table and data will begin to be loaded into it immediately. If the newly loaded data is queried before statistics can be gathered for this partition, then the Optimizer will have to prorate its cardinality estimates for these queries. Prorated cardinality estimates can lead to sub-optimal plans. By using the procedure `DBMS_STATS.COPY_TABLE_STATS`, it is possible to **copy partition statistics** from one of the other partitions in the table to the new partition. Column statistics (min, max, NDV, histogram, etc), partition statistics (number of rows, blocks, etc) and statistics for local indexes will be copied. The minimum and maximum values for the partitioning columns will be adjusted to reflect the correct values for the new partition.

In real-world data, there is often a relationship or correlation between the data stored in different columns of the same table. For example, in the `CUSTOMERS` table, the values in the `CUST_STATE_PROVINCE` column are influenced by the values in the `COUNTRY_ID` column, as the state of Bavaria is only going to be found in Germany. This not only impacts the cardinality estimat for predicates on correlated columns, it also has an impact on joins with multiple columns. With **extended statistics** you now have an opportunity to tell the Optimizer about these real-world relationships. By creating statistics on a group of columns, the Optimizer can be given a more accurate selectivity guideline for the columns used together in a where clause of a SQL statement.

It is also possible to create extended statistics for an expression (including functions), as it is difficult for the Optimizer to estimate the cardinality of a where clause predicate that has columns embedded inside expressions. For example, if it is common to have a where clauses predicate `UPPER(LastName)=:B1`, then it would be beneficial to create extended statistics for `UPPER(LastName.`

More detailed information about this functionality can be found in the following technical white paper that the various enhancements of the Optimizer in Oracle Database 11g.

SQL Plan Management for controlled plan evolution

Execution plan stability has always been somewhat of a holy grail in the optimizer space and several features have been introduced in previous releases to improve it, such as Stored Outlines and SQL Profiles. However, these methods would also prevent the optimizer from finding better plans when data volume changes. In Oracle Database 11g plan stability has been addressed with the introduction of SQL Plan Management (SPM).

SQL Plan management (SPM) ensures that runtime performance will never degrade due to the change of an execution plan. To guarantee this, only accepted (trusted) execution plans will be used; any plan will be tracked and evaluated at a later point in time and only accepted as verified if the new plan performs better than an accepted plan. SQL Plan Management has three main components:

1. SQL plan baseline capture:
   Create SQL plan baselines that represents accepted execution plans for all relevant SQL statements. The SQL plan baselines are stored in a plan history inside the SQL Management Base in the SYSAUX tablespace.
2. SQL plan baseline selection:
   Ensure that only accepted execution plans are used for statements with a SQL plan baseline and track all new execution plans in the history for a statement as unaccepted plan. The plan history consists of accepted and unaccepted plans. An unaccepted plan can be unverified (newly found but not verified) or rejected (verified but not found to performant).
3. SQL plan baseline evolution:
   Evaluate all unverified execution plans for a given statement in the plan history to become either accepted or rejected.

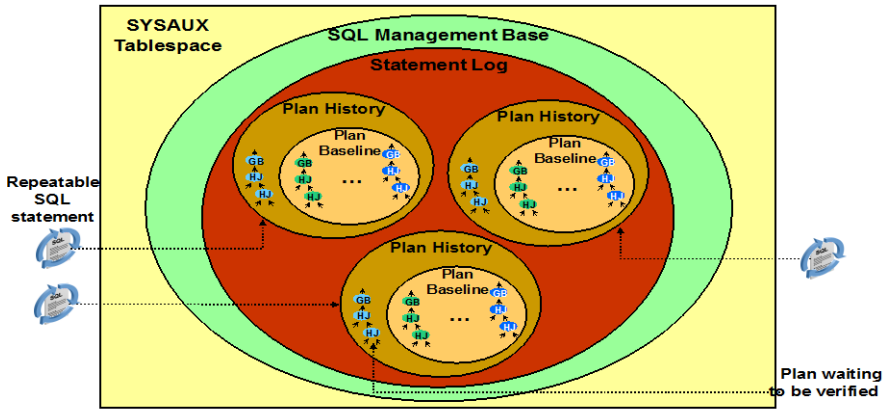

Abb. 1: SQL Management base, consisting of the statement log and plan histories for repeatable SQL Statements

With SPM the Optimizer automatically manages execution plans and ensures only known or verified plans are used. By seeding or loading SPM with execution plans from your current version of the Oracle Database prior to upgrading (or with plans from your test/development environment), you can prevent performance regressions due to plan changes. By only executing the known or seeded plans your application will behave exactly the same way in Oracle Database 11g as it did in the previous release. Any new execution plans found in Oracle Database 11g will be recorded in the plan history for that statement but will not be used. The recorded or unaccepted plans will only be used if they have been verified to perform better than the existing accepted plan for each statement. By allowing you to control what execution plans will be used not only during upgrade but as your system grows and evolves on Oracle Database 11g your system will be more stable and will have consistently good performance.

More detailed information about this functionality can be found in the SQL Plan Management technical white paper on OTN or in the SQL Plan Management OBE (Oracle by Example).

Conclusion

Since the introduction of the cost-based optimizer (CBO) in Oracle 7.0, people have been fascinated by the CBO and the statistics that it feeds on. It has long been felt that the internals of the CBO were shrouded in mystery and that a degree in wizardry was needed to work with it. However, looking at a one simple set of information – the expected and actual cardinality of an operation - and taking the appropriate action is not rocket science.

Furthermore, especially with the introduction of SQL Plan Management (SPM), Oracle provides a comprehensive frame work for plan stability with controlled evolution – functionality that dramatically simplifies tasks like upgrading the database by ensuring that the Optimizer will use only the execution plans you had before the upgrade, if the new execution plans regress.

If you are interested in more in-depth material about the Oracle Optimizer you can always visit the Query Optimization page on oracle.com and/or the Optimizer blog which is managed by the Optimizer development and Product Management.

Kontaktadresse:

Hermann Bär
Oracle USA
400 Oracle Parkway
Redwood Shores, CA 94065

Telefon:	+1 650.506.6833
Fax:	+1 650.506.6833
E-Mail	hermann.baer@oracle.com
Internet:	www.oracle.com