CHAPTER 13

Configuring the Optimizer

The cost optimizer determines the most efficient execution plan for a SQL statement. The optimizer depends heavily on the statistics that you (or the database) gather. This chapter explains how to set the optimizer goal and how to control the behavior of the optimizer. You’ll learn how to enable and disable automatic statistics collection by the database and when to collect statistics manually. You’ll learn how to set preferences for statistics collection as well as how to validate new statistics before making them available to the optimizer. The chapter explains how to lock statistics, export statistics, gather system statistics, restore older versions of statistics, and how to handle missing statistics.

Bind peeking behavior, wherein the optimizer looks at the bind variable values when parsing a SQL statement, can have unpredictable effects on execution plans. The chapter explains adaptive cursor sharing, which is designed to produce execution plans based on the specific values of bind variables.

Collecting statistics on large tables is always problematic, and the chapter shows how to use the incremental statistics gathering feature to speed up statistics collection for large partitioned tables. You’ll also learn how to use the new concurrent statistics collection feature to optimize statistics collection for large tables.

Collecting extension statistics for expressions and column groups improves optimizer performance, and you’ll learn how to collect these types of statistics. The chapter also explains how to let the database tell you which columns in a table are candidates for creating a column group.

13-1. Choosing an Optimizer Goal

Problem

You want to set the cost optimizer goal for your database.

Solution

You can influence the behavior of the cost optimizer by setting an optimizer goal. The optimizer will collect appropriate statistics based on the goal you set. You set the optimizer goal with the optimizer_mode initialization parameter. You can set the parameter to the values ALL_ROWS or FIRST_ROWS_n, as shown here:

```
optimizer_mode=all_rows
optimizer_mode=first_rows_n  /* n can be 1,10,100 or 1000 */
```
The default value for the `optimizer_mode` parameter is `ALL_ROWS`.

### How It Works

The default value for the `optimizer_mode` parameter, `ALL_ROWS`, has the goal of maximizing throughput—it minimizes resource use to complete the processing of the entire statement and get all the requested rows. The alternate value of `FIRST_ROWS_n` uses the goal of response time, which is the time it takes to return the first $n$ number of rows.

If you set the `optimizer_mode` parameter to `FIRST_ROWS_n`, all sessions will use the optimizer goal of best response time. However, you can change the optimizer goal just at the session level by executing a SQL statement such as the following:

```sql
SQL> alter session set optimizer_mode=first_rows_1;
```

Note that the `ALL_ROWS` optimizer mode setting has built-in bias toward full table scans, because its goal is to minimize resource usage. The `FIRST_ROWS_n` setting, on the other hand, favors index accesses because its goal is minimizing response time, and thus returns the requested number of rows as fast as possible.

In addition to the `optimizer_mode` parameter, you can also set the following parameters to influence the behavior of the optimizer:

- `optimizer_index_caching`
- `optimizer_index_cost_adj`
- `db_file_multiblock_read_count`

In general, changing these parameters at the database level can lead to unexpected optimizer behavior, including potential performance deterioration for some queries. The recommended practice is to leave these parameters at their default levels. We, however, do show (Recipe 13-11) how to use one of these parameters (`optimizer_index_cost_adj`) at the session level, to improve the performance of a long-running query by forcing the optimizer to use an index.

### 13-2. Enabling Automatic Statistics Gathering

#### Problem

You want to enable automatic statistics gathering in your database.

Note: Oracle recommends the enabling of automatic optimizer statistics collection.

#### Solution

You enable automatic statistics collection by using the `enable` procedure in the `DBMS_AUTO_TASK_ADMIN` package. Check the status of the auto optimizer stats collection task in the following way: