

12

SQL Plan Management

Objectives

After completing this lesson, you should be able to:

- Manage SQL performance through changes
- Set up SQL Plan Management
- Set up various SQL Plan Management scenarios
- Load hinted plans into SQL Plan Management
- Migrate stored outlines to SQL plan baselines

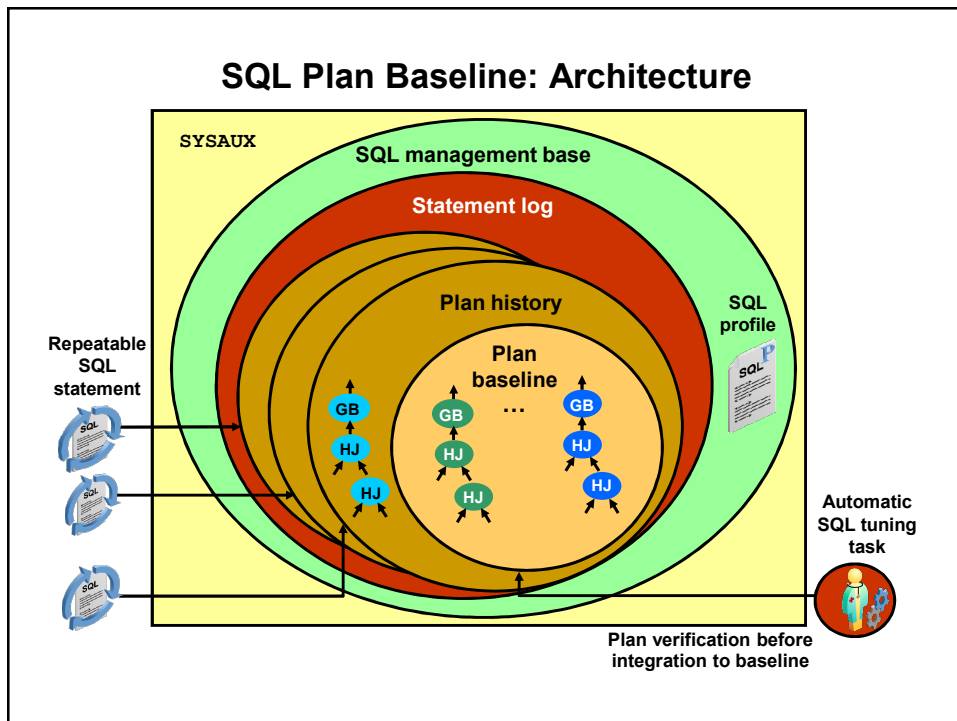
Maintaining SQL Performance

Maintaining performance may require using SQL plan baselines.

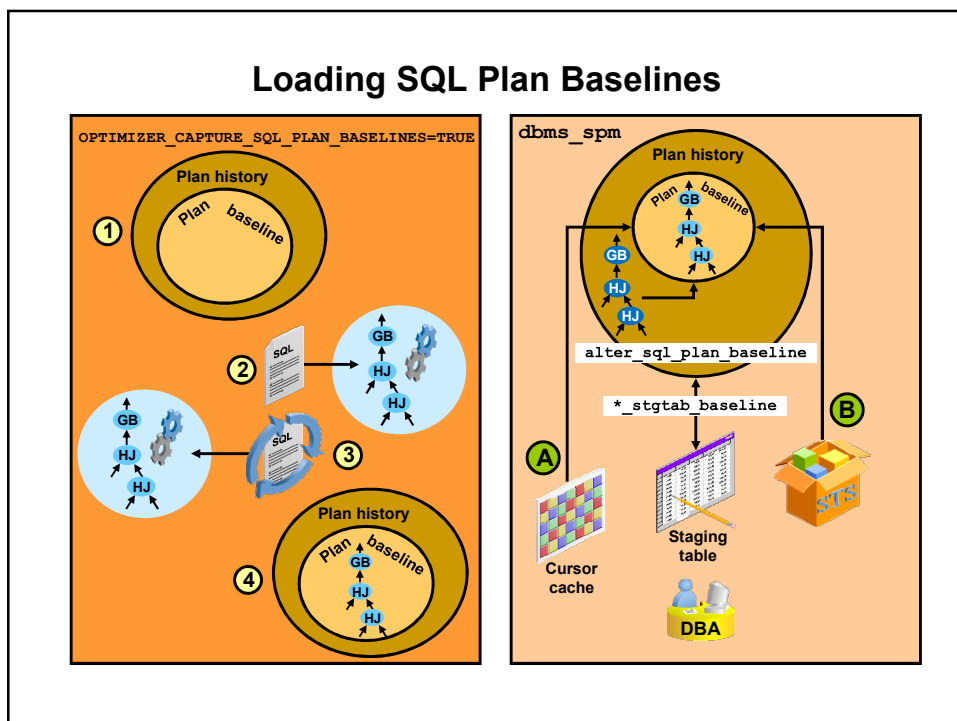
SQL Plan Management: Overview

- SQL Plan Management is automatically controlled SQL plan evolution.
- Optimizer automatically manages SQL plan baselines.
 - Only known and verified plans are used.
- Plan changes are automatically verified.
 - Only comparable or better plans are subsequently used.
- The plan baseline can be seeded for critical SQL with SQL tuning set (STS) from SQL Performance Analyzer.

SQL Plan Baseline: Architecture

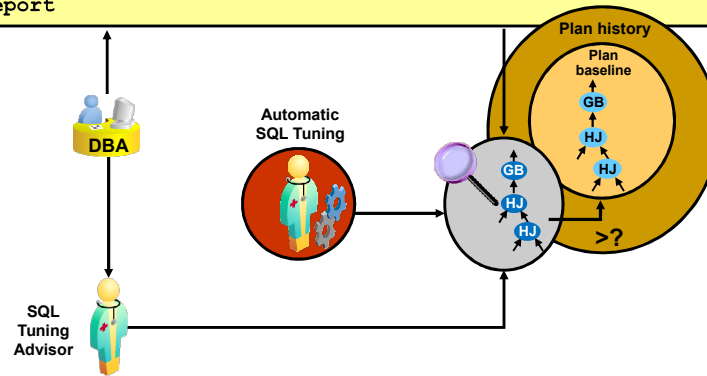


Loading SQL Plan Baselines

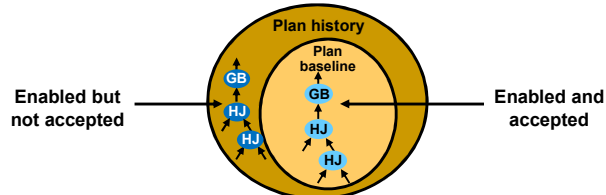


Evolving SQL Plan Baselines

```
variable report clob
exec :report:=DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE(-
sql_handle=>'SYS_SQL_593bc74fca8e6738');
Print report
```



Important Baseline SQL Plan Attributes



```
select signature, sql_handle, sql_text, plan_name, origin, enabled,
accepted, fixed, autopurge
from dba_sql_plan_baselines;
```

SIGNATURE	SQL_HANDLE	SQL_TEXT	PLAN_NAME	ORIGIN	ENA	ACC	FIX	AUT
8.062E+18	SYS_SQL_6fe2	select..	SQL_PLAN_6zsn...	AUTO-CAPTURE	YES	NO	NO	YES
8.062E+18	SYS_SQL_e23f	select..	SQL_PLAN_f4gy...	AUTO-CAPTURE	YES	YES	NO	YES
...								

```
exec :cnt := dbms_spm.alter_sql_plan_baseline(-
sql_handle => 'SYS_SQL_6fe28d438dfc352f', -
plan_name => 'SQL_PLAN_6zsnd8f6zsd9g54bc8843',-
attribute_name => 'ENABLED', attribute_value => 'NO');
```

SQL Plan Selection

```

graph TD
    SQL((SQL)) --> Plan1((Plan 1))
    Plan1 --> D1{optimizer_use_sql_plan_baselines = true?}
    D1 -- No --> Plan2((Plan 2))
    D1 -- Yes --> D2{Plan part of history?}
    D2 -- Yes --> Plan3((Plan 3))
    D2 -- No --> Plan4((Plan 4))
    Plan3 --> D3{Plan part of baseline?}
    D3 -- Yes --> Plan2
    D3 -- No --> Plan5((Plan 5))
    Plan4 --> PlanHistory((Plan history))
    PlanHistory --> Plan5
    Plan5 --> D4{Select baseline plan with lowest best cost.}
    D4 -- Yes --> Plan2
    D4 -- No --> Plan6((Plan 6))
    Plan6 --> D5{...}
    D5 --> DBMS[dbms_xplan.display(..., 'BASIC +NOTE') or plan_table(other_xml)]
  
```

The flowchart illustrates the SQL Plan Selection process. It begins with an SQL statement being executed, leading to a plan (Plan 1). A decision is made on whether the optimizer should use SQL plan baselines. If not, a new plan (Plan 2) is generated. If yes, it checks if the plan is part of the history. If it is, it checks if it's part of the baseline. If not, it's added to the plan history. The process then selects the baseline plan with the lowest best cost. If a plan is selected, it's used; otherwise, a new plan (Plan 6) is generated. The final step is to display the plan using `dbms_xplan.display(..., 'BASIC +NOTE')` or `plan_table(other_xml)`.

Possible SQL Plan Manageability Scenarios

The diagram illustrates two scenarios for SQL Plan Manageability:

Database Upgrade

Oracle Database 11g

The top section shows a production database (Oracle Database 11g) with a plan history containing a baseline plan (HJ) and other plans (GB, HJ). The text "No plan regressions" is present. A DBA (Database Administrator) is shown interacting with the database. A plan table (PTS) is shown, indicating that the plan is well-tuned.

Oracle Database 10g

The bottom section shows a development database (Oracle Database 10g) with a plan history containing a baseline plan (HJ) and other plans (GB, HJ). The text "Well-tuned plan" is present. A DBA is shown interacting with the database. A plan table (PTS) is shown, indicating that the plan is well-tuned.

New Application Deployment

Production database

The top section shows a production database with a plan history containing a baseline plan (HJ) and other plans (GB, HJ). The text "No plan regressions" is present. A DBA is shown interacting with the database. A plan table (PTS) is shown, indicating that the plan is well-tuned.

Development database

The bottom section shows a development database with a plan history containing a baseline plan (HJ) and other plans (GB, HJ). The text "Well-tuned plan" is present. A DBA is shown interacting with the database. A plan table (PTS) is shown, indicating that the plan is well-tuned.

SQL Performance Analyzer and SQL Plan Baseline Scenario

The diagram illustrates the SQL Performance Analyzer and SQL Plan Baseline Scenario, showing the flow of SQL statements and the impact of the `optimizer_features_enable` parameter.

Oracle Database 11g (Top Section):

- Before change:** `O_F_E=10`. SQL statements are captured from the **STS** (SQL Tuning Set) and analyzed by the **SQL Performance Analyzer**.
- After change:** `O_F_E=11`. The same SQL statements are re-analyzed.
- Regressing statements:** Statements that show performance regressions are identified.
- Plan history:** A circular history of plans for each statement. The **baseline** plan is highlighted. Plans are labeled with **GB** (Good Baseline) and **HJ** (High Join).
- No plan regressions:** Statements that do not show regressions are marked as such.

Oracle Database 10g (Bottom Section):

- Well-tuned plans:** A set of plans that are well-tuned for the database.
- STS:** The SQL Tuning Set, which captures SQL statements from the database.

Flow and Impact:

- SQL statements are captured from the **STS** in Oracle Database 10g and sent to the **SQL Performance Analyzer** in Oracle Database 11g.
- The **SQL Performance Analyzer** analyzes the statements and identifies regressions.
- The **SQL Performance Analyzer** sends the results back to the **STS** in Oracle Database 10g.
- The **SQL Performance Analyzer** also sends the results to the **SQL Plan Baseline** in Oracle Database 11g.
- The **SQL Plan Baseline** uses the results to update the **Plan history** and identify **Regressing statements**.
- The **SQL Plan Baseline** also sends the results to the **SQL Performance Analyzer**.
- The **SQL Performance Analyzer** sends the results to the **SQL Plan Baseline**.

Loading a SQL Plan Baseline Automatically

The diagram illustrates the process of automatically loading a SQL plan baseline from a well-tuned plan in Oracle Database 10g to Oracle Database 11g.

Oracle Database 10g (Bottom Left): Shows a "Well-tuned plans" scenario. The plan history contains a "Plan baseline" with two execution plans. Each plan consists of a "GB" (Global Buffer) node, an "HJ" (Hash Join) node, and an "HJ" (Hash Join) node. The plans are labeled "Well-tuned plans".

Oracle Database 11g (Top Left): Shows a "No plan regressions" scenario. The plan history contains a "Plan baseline" with two execution plans. Each plan consists of a "GB" (Global Buffer) node, an "HJ" (Hash Join) node, and an "HJ" (Hash Join) node. The plans are labeled "No plan regressions". The optimizer features are set to `optimizer_features_enable=10.2.0.2` and `optimizer_capture_sql_plan_baselines=true`.

Oracle Database 11g (Top Right): Shows a "No plan regressions" scenario. The plan history contains a "Plan baseline" with two execution plans. Each plan consists of a "GB" (Global Buffer) node, an "HJ" (Hash Join) node, and an "HJ" (Hash Join) node. The plans are labeled "No plan regressions". The optimizer features are set to `optimizer_features_enable=11.1.0.1` and `optimizer_capture_sql_plan_baselines=true`. A "New plan waiting verification" is shown, indicating that the plan is being verified.

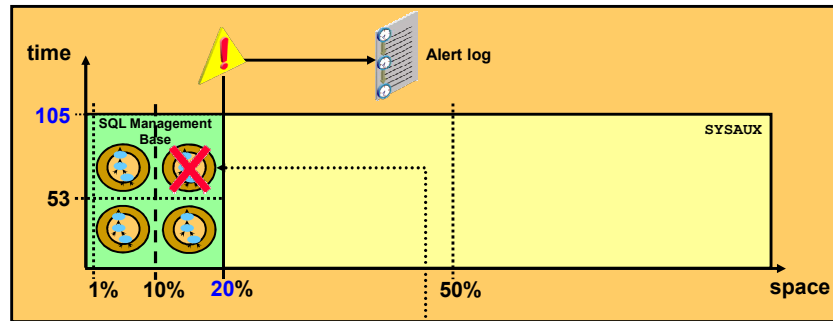
Oracle Database 11g (Bottom Right): Shows a "Better plans" scenario. The plan history contains a "Plan baseline" with two execution plans. Each plan consists of a "GB" (Global Buffer) node, an "HJ" (Hash Join) node, and an "HJ" (Hash Join) node. The plans are labeled "Better plans". The optimizer features are set to `optimizer_features_enable=11.1.0.1` and `optimizer_capture_sql_plan_baselines=true`.

The flow of the process is indicated by arrows: from the 10g database to the 11g database (top left), and from the 11g database (top right) to the 11g database (bottom right).

Purging SQL Management Base Policy

```
SQL> exec dbms_spm.configure('SPACE_BUDGET_PERCENT',20);
SQL> exec dbms_spm.configure('PLAN_RETENTION_WEEKS',105);
```

DBA_SQL_MANAGEMENT_CONFIG



```
SQL> exec :cnt := dbms_spm.drop_sql_plan_baseline('SYS_SQL_37e0168b04e73efe');
```

Enterprise Manager and SQL Plan Baselines

Query Optimizer
Manage Optimizer Statistics
SQL Plan Control
SQL Tuning Sets

Plan Control

SQL Profile SQL Patch SQL Plan Baseline

A SQL Plan Baseline is an execution plan deemed to have acceptable performance for a given SQL statement.

Settings

Capture SQL Plan Baselines **FALSE**

Use SQL Plan Baselines **TRUE**

Plan Retention(Weeks) **53** [Configure](#)

Jobs for SQL Plan Baselines

Pending Completed

[Load Jobs](#) [SPM_1267706070256](#)

Search

SQL Text [Go](#)

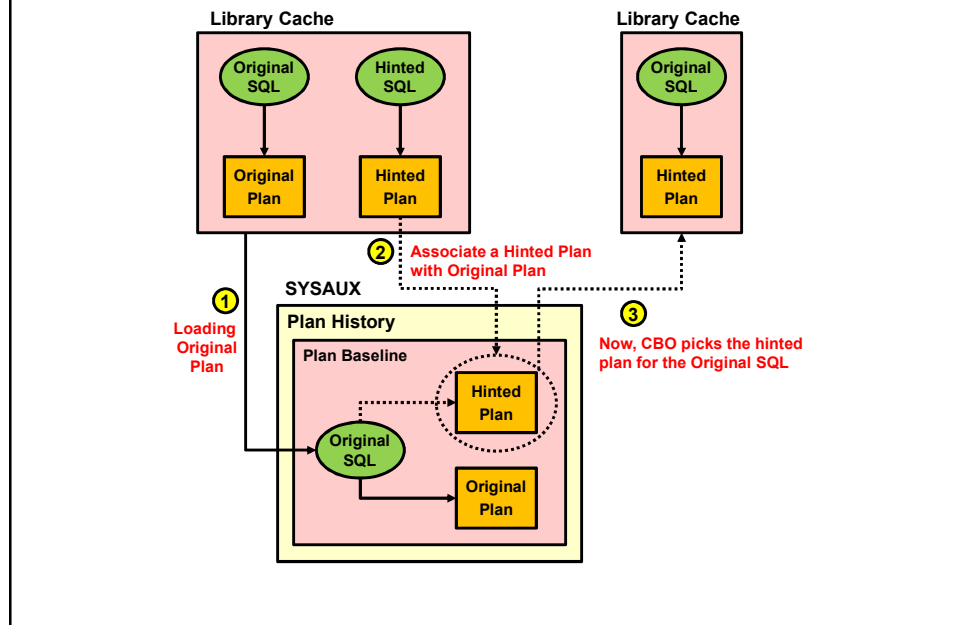
By default, the search is case insensitive. To run an exact or case-sensitive search, double-quote the search string. You may also use the '%' symbol as a wildcard.

[Enable](#) [Disable](#) [Drop](#) [Evolve](#) [Pack](#) Fixed - Yes [Go](#)

Select All | Select None

Select Name	SQL Text	Enabled	Accepted	Fixed	Auto Purge	Created	Last Modified
<input type="checkbox"/> SQL_PLAN_f8dtd73y70913762178	SELECT /* ORDERED INDEX(1) USE_HASH(1) */ y...	YES	YES	NO	YES	Mar 4, 2010 12:34:35 PM	Mar 4, 2010 12:34:35 PM
<input type="checkbox"/> SQL_PLAN_dqqrtpq9jhb04e9eb5b65	SELECT 'B' tt1.ch.featurevalue_09_id ch_fea...	YES	YES	NO	YES	Mar 4, 2010 12:34:35 PM	Mar 4, 2010 12:34:35 PM
<input type="checkbox"/> SQL_PLAN_1fp9bfzfauf113762178	SELECT /* ORDERED INDEX(1) USE_HASH(1) */ y...	YES	YES	NO	YES	Mar 4, 2010 12:34:35 PM	Mar 4, 2010 12:34:35 PM

Loading Hinted Plans into SPM: Example



Using the MIGRATE_STORED_OUTLINE Functions

- Specify stored outlines to be migrated based on outline name, SQL text, or outline category, or migrate all stored outlines in the system to SQL plan baselines:

```
DBMS_SPM.MIGRATE_STORED_OUTLINE (
  attribute_name IN VARCHAR2,
  attribute_value IN CLOB,
  fixed IN VARCHAR2 := 'NO')
RETURN CLOB;
```

- Specify one or more stored outlines to be migrated:

```
DBMS_SPM.MIGRATE_STORED_OUTLINE (
  outln_list IN DBMS_SPM.NAME_LIST,
  fixed IN VARCHAR2 := 'NO')
RETURN CLOB;
```


Quiz

When the `OPTIMIZER_USE_SQL_PLAN_BASELINES` parameter is set to `TRUE`, the optimizer:

- a. Does not develop an execution plan; it uses an accepted plan in the baselines
- b. Compares the plan that it develops with accepted plans in the baselines
- c. Compares the plan that it develops with enabled plans in the baselines
- d. Does not develop an execution plan; it uses enabled plans in the baselines
- e. Develops plans and adds them to the baselines as verified

Summary

In this lesson, you should have learned how to:

- Manage SQL performance through changes
- Set up SQL Plan Management
- Set up various SQL Plan Management scenarios
- Load hinted plans into SQL Plan Management
- Migrate Stored Outlines to SQL Plan Baselines

Practice 12: Overview Using SQL Plan Management

This practice covers the use of SQL Plan Management.