QUESTION :I Have Two Plan Hash Values One Planhash Value Taking 5 Min And Another One Is Taking 1hr.How To Assign Best Plan To Sql Query.

To assign the best execution plan to a SQL query in Oracle, you can use SQL Plan Baselines or SQL Profiles. Here’s a step-by-step approach:

1. Identify the Plans

First, you need to gather the details of both execution plans using their hash values. You can analyze them with the following queries:

SELECT \* FROM dba\_hist\_sql\_plan WHERE plan\_hash\_value IN (hash\_value\_1, hash\_value\_2);

2. Create a SQL Plan Baseline

If you have determined which plan performs better, you can create a SQL Plan Baseline for the SQL statement. This allows Oracle to use the preferred execution plan consistently.

2.1. Create the Baseline

Use the following command to create a baseline:

BEGIN

DBMS\_SQLTUNE.CREATE\_SQL\_PLAN\_BASELINE(

sql\_text => 'your SQL text here',

plan\_hash\_value => best\_plan\_hash\_value

);

END;

/

Replace `'your SQL text here'` with the actual SQL statement and `best\_plan\_hash\_value` with the hash value of the better-performing plan.

2.2 Validate the Baseline:

You can validate that the plan has been added successfully:

SELECT \*

FROM dba\_sql\_plan\_baselines

WHERE sql\_text = 'your SQL text here';

3. Use a SQL Profile (if necessary)

If the performance still isn't satisfactory, you can consider creating a SQL Profile. This may help improve execution based on specific bind variable values:

BEGIN

DBMS\_SQLTUNE.create\_sql\_profile(

name => 'your\_profile\_name',

sql\_text => 'your SQL text here',

force\_match => TRUE

);

END;

/

4. Testing

After creating the baseline or profile, test the SQL query to ensure it uses the preferred plan:

EXPLAIN PLAN FOR your\_sql\_text;

SELECT \* FROM table(dbms\_xplan.display);

5. Monitor Performance

After implementing the changes, monitor the performance of the SQL statement to ensure its consistently using the desired plan:

SELECT sql\_id, elapsed\_time, cpu\_time FROM v$sql WHERE sql\_text = 'your SQL text here';

6. Adjust if Needed

If the performance is not as expected, you may need to drop the plan baseline or profile and investigate further to determine the root cause. You can drop a SQL Plan Baseline as follows:

BEGIN

DBMS\_SQLTUNE.DROP\_SQL\_PLAN\_BASELINE(

name => 'your\_plan\_baseline\_name'

);

END;

/

**Summary**

Using SQL Plan Baselines allows you to control which execution plan Oracle uses for a specific SQL statement, ensuring that you benefit from the performance of the optimal plan. If further tuning is needed, consider using SQL Profiles and always monitor the performance after making changes.

**Commands:**

**==========**

**3. Check Execution Plan**

To see the execution plan of a specific SQL query:

EXPLAIN PLAN FOR your\_query;

SELECT \* FROM table(dbms\_xplan.display);

**14. Detailed SQL Execution Stats**

To get detailed execution statistics for a specific SQL statement

SELECT \* FROM v$sql\_plan WHERE sql\_id = 'your\_sql\_id';

**26. Query Resource Manager Plan**

To see if a Resource Manager plan is affecting query performance:

SELECT plan\_name, active FROM dba\_rsrc\_plan;

44. Check for SQL Plan Baselines

To view existing SQL plan baselines that might help in execution:

SELECT \* FROM dba\_sql\_plan\_baselines;

48. Identify SQL Execution Plans in Cache

To find execution plans for cached SQL:

SELECT \* FROM v$sql\_plan WHERE sql\_id = 'your\_sql\_id'; -- Replace with your SQL ID

69. Examine Query Execution Plans for Changes

To check if execution plans have changed for a specific SQL:

SELECT \* FROM dba\_hist\_sql\_plan WHERE sql\_id = 'your\_sql\_id'; -- Replace with your SQL ID

80. Execution Plan Evolution

To monitor changes in execution plans over time:

SELECT sql\_id, plan\_hash\_value, execution\_count, last\_active\_time FROM dba\_hist\_sql\_plan

WHERE sql\_id = 'your\_sql\_id'; -- Replace with your SQL ID