**What are Hints in Oracle?**

**Hints in Oracle** are special instructions added to SQL queries that tell Oracle **how** to run the query.

These instructions guide Oracle’s **optimizer** to make decisions like which **index** to use or how to join tables, to improve the performance of the query.

Instead of letting Oracle decide automatically, hints allow you to suggest the best way to execute a query.

**Syntax of a Hint:**

In Oracle, a hint is written inside the SQL query as a **comment**, and it must follow the format:

/\*+ <hint> \*/

**Types of Hints in Oracle:**

Some common types of hints include:

1. **Index hints** – Suggests using a specific index.
2. **Join hints** – Directs the optimizer to use a specific join method (like MERGE JOIN, NESTED LOOPS).
3. **Parallel hints** – Directs Oracle to parallelize the query execution.
4. **Access path hints** – Specifies the access method for a table (e.g., FULL, RANGE SCAN, etc.).
5. **Optimizer hints** – Influences the optimizer's decision-making, like ALL\_ROWS or FIRST\_ROWS.

**Examples of Common Hints:**

**INDEX HINT**:

Force the query to use a specific index.

SELECT /\*+ INDEX(emp emp\_idx) \*/ \* FROM employees emp WHERE emp\_id = 100;

This forces the query to use the emp\_idx index on the employees table.

**JOIN HINT**:

Specify which type of join Oracle should use.

SELECT /\*+ USE\_NL(emp dept) \*/ \* FROM employees emp JOIN departments dept ON emp.dept\_id = dept.dept\_id;

This uses the **Nested Loops Join** method between the employees and departments tables.

**PARALLEL HINT**:

Parallelize the query execution.

SELECT /\*+ PARALLEL(emp, 4) \*/ \* FROM employees emp;

This hints Oracle to use **4 parallel processes** when scanning the employees table.

**NO MERGE HINT**:

Prevents the optimizer from performing a merge join.

SELECT /\*+ NO\_MERGE(emp, dept) \*/ \* FROM employees emp JOIN departments dept ON emp.dept\_id = dept.dept\_id;

This tells Oracle not to use a merge join for this query.

**FULL TABLE SCAN HINT**:

Forces Oracle to perform a full table scan instead of using an index.

SELECT /\*+ FULL(emp) \*/ \* FROM employees emp WHERE emp\_id = 100;

This forces a full table scan on the employees table.

**How to Create and Use Hints in Oracle 19c:**

In Oracle 19c, you can create hints by adding them directly within your SQL queries. There isn't a special object to "create" hints as they are embedded within the SQL text itself.

**Example 1: Using a Hint for a Full Table Scan**

Let's say you have a query where you want to force Oracle to perform a **full table scan**:

SELECT /\*+ FULL(emp) \*/ emp\_id, emp\_name FROM employees emp WHERE emp\_id > 100;

Here, /\*+ FULL(emp) \*/ is the hint telling Oracle to use a full table scan on the emp table, even if there is an index available.

**Example 2: Using an Index Hint**

If you want Oracle to use a specific index when querying the employees table, you can use the INDEX hint:

SELECT /\*+ INDEX(emp emp\_id\_idx) \*/ emp\_id, emp\_name

FROM employees emp

WHERE emp\_id = 101;

In this case, /\*+ INDEX(emp emp\_id\_idx) \*/ directs Oracle to use the index emp\_id\_idx on the employees table.

**Example 3: Using Join Hints**

If you have a query that joins two tables and you want to force a specific join method (e.g., **Nested Loops**), you can use the following hint:

SELECT /\*+ USE\_NL(emp dept) \*/ emp\_id, emp\_name, dept\_name

FROM employees emp

JOIN departments dept ON emp.dept\_id = dept.dept\_id;

This forces the optimizer to use the **Nested Loops** join method for the employees and departments tables.

**Example 4: Using Parallel Execution Hint**

To improve performance, especially for large tables, you can hint Oracle to perform parallel query execution:

SELECT /\*+ PARALLEL(emp, 4) \*/ emp\_id, emp\_name FROM employees emp;

This hints Oracle to use **4 parallel processes** when querying the employees table.

**Example 5: Using NoMerge Hint for Joins**

Sometimes, you might want to prevent Oracle from using a merge join. The NO\_MERGE hint does this:

SELECT /\*+ NO\_MERGE(emp, dept) \*/ emp\_id, emp\_name, dept\_name

FROM employees emp

JOIN departments dept ON emp.dept\_id = dept.dept\_id;

This will prevent Oracle from performing a merge join between the employees and departments tables.

**Using Hints to Control Query Execution Plan**

Hints can also be used to directly control how Oracle executes specific steps in a query's execution plan. Here’s an example where you combine multiple hints:

SELECT /\*+ PARALLEL(emp, 4) USE\_HASH(emp, dept) \*/ emp\_id, emp\_name, dept\_name

FROM employees emp

JOIN departments dept ON emp.dept\_id = dept.dept\_id;

In this example:

* **PARALLEL(emp, 4)**: Oracle will use **4 parallel processes** for scanning the emp table.
* **USE\_HASH(emp, dept)**: Oracle will use a **hash join** between employees and departments.

**How to Check If Hints Are Being Used**

You can verify if the hints are being used by checking the **execution plan** of your query. To view the execution plan, you can use EXPLAIN PLAN or DBMS\_XPLAN:

**Using EXPLAIN PLAN:**

EXPLAIN PLAN FOR

SELECT /\*+ INDEX(emp emp\_id\_idx) \*/ emp\_id, emp\_name

FROM employees emp

WHERE emp\_id = 101;

Then to view the execution plan:

SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY);

This will show the execution plan, and you should see the hint applied in the plan output.

**Things to Keep in Mind:**

* **Hints are a suggestion, not a guarantee**: While hints guide the optimizer, Oracle may still decide not to use the hint based on the system's current statistics and conditions.
* **Overuse of hints can make your queries less flexible**: Relying too much on hints can limit the optimizer's ability to make optimal decisions in changing environments. It is often better to let Oracle's optimizer handle execution plans unless you have specific reasons for using hints.
* **Hints are database-specific**: The syntax and behavior of hints can vary between database versions. Always check the documentation for your version to ensure the correct usage of hints.

**Conclusion:**

In Oracle 19c, **hints** are embedded within SQL queries to provide specific instructions to the query optimizer about how to execute a query. These hints can control index usage, join methods, parallel execution, and other aspects of query optimization. Properly used, hints can help improve performance, but overusing them may prevent the optimizer from making the best decision in some cases. Always use hints judiciously and test their impact on performance.Top of Form

Bottom of Form