Using Optimizer Hints

- You can use comments in a SQL statement to pass instructions, or **hints**, to the Oracle Database optimizer.
- Hints provide a mechanism to instruct the optimizer to choose a certain query **execution plan** based on specific criteria.
- The optimizer uses these hints to choose an execution plan for the statement, **unless** some condition exists that prevents the optimizer from doing so.
- Hints let you make **decisions** usually made by the optimizer:
  - you might know information about your data that the optimizer does not know.
Specifying Hints

- Hints apply only to the optimization of the block of a statement in which they appear.
- A statement block is any SELECT, UPDATE, or DELETE statement, including sub-queries.
- The plus sign (+) causes Oracle to interpret the comment as a list of hints.
  - The plus sign must follow immediately after the comment delimiter.
  - No space is permitted between the comment delimiter and the plus sign.
  - The space between the plus sign and the hint is optional.
- If the comment contains multiple hints, then separate the hints by at least one space.
- Example
  ```sql
  SELECT /*+ Hint1 Hint2 Hint3 */ columnName
  FROM tableName
  WHERE conditions [...]
  ```

Optimizer hints categories

- Optimizer hints are grouped into the following categories:
  - Hints for Optimization Approaches and Goals
  - Hints for Access Paths
  - Hints for Query Transformations
  - Hints for Join Orders
  - Hints for Join Operations
  - Hints for Parallel Execution
  - Additional Hints
Optimization Approaches and Goals

- The following hints let you choose between optimization approaches and goals
  - **ALL_ROWS** optimizes a statement block with a goal of best throughput, i.e., minimum total resource consumption
  - **FIRST_ROWS(n)** optimizes an individual SQL statement for fast response, choosing the plan that returns the first n rows most efficiently

- If a SQL statement has a hint specifying an optimization approach and goal, then the optimizer uses the specified approach regardless of the presence or absence of
  - statistics (if absent, optimizer uses default statistical values)
  - the OPTIMIZER_MODE initialization parameter
  - the OPTIMIZER_MODE parameter of the ALTER SESSION statement

- The optimizer gives precedence to the hints for access paths or join operations, before **ALL_ROWS** or **FIRST_ROWS(n)**

Hints for Access Paths

- Each of the following hints instructs the optimizer to use a specific access path for a table
  - **FULL(table)**
  - **INDEX(table indexNames)**
  - **NO_INDEX(table indexNames)**
  - **INDEX_COMBINE(table indexNames)**
  - **INDEX_FFS(table indexNames)**
  - **NO_INDEX_FFS(table indexNames)**

- Specifying one of these hints causes the optimizer to choose the specified access path only if the access path is available
  - existence of an index
  - syntactic constructs of the SQL statement

- You must specify the table to be accessed exactly as it appears in the statement
  - if the statement uses an alias for the table, then use the alias rather than the table name

Oracle Hints
Hints for Access Paths

- **FULL(table)**
  - full table scan on the specified table
  - if a table alias is defined, the table must be referenced with its alias
- **INDEX(table indexName1 indexName2 …)**
  - index scan using one or more specified indexes for the specified table
  - does not consider a full table scan or a scan on an index not listed
- **NO_INDEX(table indexName1 indexName2 …)**
  - avoid using one or more specified indexes for the specified table
- **INDEX_COMBINE(table indexName1 indexName2 …)**
  - uses a bitmap access path (Boolean combination) of the specified indexes for the table
- **INDEX_FFS(table indexName1 indexName2 …)**
  - instructs the optimizer to perform a fast full index scan rather than a full table scan
- **NO_INDEX_FFS(table indexName1 indexName2 …)**
  - excludes a fast full index scan of the specified indexes on the specified table

Join Operations

- Each of the following hints instructs the optimizer to use a specific join operation for the specified tables
  - **USE_NL( table1, table2, …)**
  - **NO_USE_NL ( … )**
  - **USE_MERGE ( … )**
  - **NO_USE_MERGE ( … )**
  - **USE_HASH ( … )**
  - **NO_USE_HASH ( … )**

- Oracle uses these hints when the referenced table is forced to be the inner table of a join; the hints are ignored if the referenced table is the outer table
Join Orders

- The following hints suggest join orders
  - `ORDERED`
  - `LEADING( table1 table2 ...)`

- The `ORDERED` hint instructs Oracle to join tables in the order in which they appear in the `FROM` clause.

- The `LEADING` hint instructs the optimizer to use the specified set of tables as the hint parameters.

- These hints let you choose an inner and outer table:
  - the first table is the `outer` table
  - the second table is the `inner` table

Join Orders - Example

- `SELECT /*+ ORDERED */ *` from `emp e, dept d`
- `WHERE d.deptno = e.deptno`
- `LEADING( e d )`

- `SELECT /*+ ORDERED */ *` from `dept d, emp e`
- `WHERE d.deptno = e.deptno`
- `LEADING( d e )`
Example

- SELECT /*+
  LEADING(e j)
  USE_NL(e j)
  INDEX(j empID_index)
  FULL(e) */
  e.empID, e.Name, sum(j.salary)
FROM empl e, jobs j
AND e.empID = j.empID
GROUP BY e.empID, e.Name

- the **LEADING** hint specifies the exact join order to be used
- the index **empID_index** is suggested to be used
- the join method **USE_NL** to be used on the join tables is also specified
- the **FULL** table access path to table jobs is suggested