HINTS Working Document for Our Students

HINT NAMES	MEANINGS
ALL_ROWS	Use the cost based approach for best throughput.
CHOOSE	Default, if statistics are available will use cost, if not, rule.
FIRST_ROWS	Use the cost based approach for best response time.
RULE	Use rules based approach; this cancels any other hints specified for this statement.
ACCESS METHODS ORACLE HINTS	
CLUSTER(table)	This tells Oracle to do a cluster scan to access the table.
FULL(table)	This tells the optimizer to do a full scan of the specified table.
HASH(table)	Tells Oracle to explicitly choose the hash access method for the table.
HASH_AJ(table)	Transforms a NOT IN subquery to a hash anti-join.
ROWID(table)	Forces a rowid scan of the specified table.
INDEX(table [index])	Forces an index scan of the specified table using the specified index(s). If a list of indexes is specified, the optimizer chooses the one with the lowest cost. If no index is specified then the optimizer chooses the available index for the table with the lowest cost.
INDEX_ASC (table [index])	Same as INDEX only performs an ascending search of the index chosen, this is functionally identical to the INDEX statement.
INDEX_DESC(table [index])	Same as INDEX except performs a descending search. If more than one table is accessed, this is ignored.
INDEX_COMBINE(table index)	Combines the bitmapped indexes on the table if the cost shows that to do so would give better performance.
INDEX_FFS(table index)	Perform a fast full index scan rather than a table scan.
MERGE_AJ (table)	Transforms a NOT IN subquery into a merge anti-join.
AND_EQUAL(table index index [index index index])	This hint causes a merge on several single column indexes. Two must be specified, five can be.
NL_AJ	Transforms a NOT IN subquery into a NL anti-join (nested loop).
HASH_SJ(t1, t2)	Inserted into the EXISTS subquery; This converts the subquery into a special type of hash join between t1 and t2 that preserves the semantics of the subquery. That is, even if there is more than one matching row in t2 for a row in t1, the row in t1 is returned only once.
MERGE_SJ (t1, t2)	Inserted into the EXISTS subquery; This converts the subquery into a special type of merge join between t1 and t2 that preserves the semantics of the subquery. That is, even if there is more than one matching row in t2 for a row in t1, the row in t1 is returned only once.
NL_SJ	Inserted into the EXISTS subquery; This converts the subquery into a special type of nested loop join between t1 and t2 that preserves the semantics of the subquery. That is, even if there is more than one matching row in t2 for a row in t1, the row in t1 is returned only once.
ORDERED	This hint forces tables to be joined in the order specified. If you know table X has fewer rows, then ordering it first may speed execution in a join.
STAR	Forces the largest table to be joined last using a nested loops join on the index.
STAR_TRANSFORMATION	Makes the optimizer use the best plan in which a start transformation is used.

HINTS Working Document for Our Students

`FACT(table)`	When performing a star transformation use the specified table as a fact table.
`NO_FACT(table)`	When performing a star transformation do not use the specified table as a fact table.
`PUSH_SUBQ`	This causes nonmerged subqueries to be evaluated at the earliest possible point in the execution plan.
`REWRITE(mview)`	If possible forces the query to use the specified materialized view, if no materialized view is specified, the system chooses what it calculates is the appropriate view.
`NOREWRITE`	Turns off query rewrite for the statement, use it for when data returned must be concurrent and can't come from a materialized view.
`USE_CONCAT`	Forces combined OR conditions and IN processing in the WHERE clause to be transformed into a compound query using the UNION ALL set operator.
`NO_MERGE (table)`	This causes Oracle to join each specified table with another row source without a sort-merge join.
`NO_EXPAND`	Prevents OR and IN processing expansion.

ORACLE HINTS for JOIN OPERATIONS

`USE_HASH (table)`	This causes Oracle to join each specified table with another row source with a hash join.
`USE_NL(table)`	This operation forces a nested loop using the specified table as the controlling table.
`USE_MERGE(table,[table, -])`	This operation forces a sort-merge-join operation of the specified tables.
`SITE`	The hint forces query execution to be done at a different site than that selected by Oracle. This hint can be used with either rule-based or cost-based optimization.
`LEADING(table)`	The hint causes Oracle to use the specified table as the first table in the join order.

ORACLE HINTS for PARALLEL OPERATIONS

`[NO]APPEND`	This specifies that data is to be or not to be appended to the end of a file rather than into existing free space. Use only with INSERT commands.
`NOPARALLEL (table`	This specifies the operation is not to be done in parallel.
`PARALLEL(table, instances)`	This specifies the operation is to be done in parallel.
`PARALLEL_INDEX`	Allows parallelization of a fast full index scan on any index.

Other Oracle Hints...

`CACHE`	Specifies that the blocks retrieved for the table in the hint are placed at the **most** recently used end of the LRU list when the table is full table scanned.
`NOCACHE`	Specifies that the blocks retrieved for the table in the hint are placed at the **least** recently used end of the LRU list when the table is full table scanned.
`[NO]APPEND`	For insert operations will append (or not append) data at the HWM of table.
`UNNEST`	Turns on the UNNEST_SUBQUERY option for statement if UNNEST_SUBQUERY parameter is set to FALSE.
`NO_UNNEST`	Turns off the UNNEST_SUBQUERY option for statement if UNNEST_SUBQUERY parameter is set to TRUE.
`PUSH_PRED`	Pushes the join predicate into the view.