EXPLAIN PLAN

Understanding EXPLAIN PLAN

- The `EXPLAIN PLAN` statement displays execution plans chosen by the Oracle optimizer for `SELECT`, `UPDATE`, `INSERT`, and `DELETE` statements.
- A statement's execution plan is the sequence of operations Oracle performs to run the statement.

- The row source tree is the core of the execution plan. It shows the following information:
  - An *ordering* of the tables referenced by the statement
  - An *access method* for each table mentioned in the statement
  - A *join method* for tables affected by join operations in the statement
  - *Data operations* like filter, sort, or aggregation

Understanding EXPLAIN PLAN

- In addition to the row source tree, the plan table contains information about the following:
  - *Optimization*, such as the cost and cardinality of each operation
  - *Partitioning*, such as the set of accessed partitions
  - *Parallel execution*, such as the distribution method of join inputs
- The `EXPLAIN PLAN` results let you determine whether the optimizer selects a particular execution plan
- It also helps you to understand the optimizer decisions and explains the performance of a query.
- When evaluating a plan, examine the statement's actual resource consumption. Use Oracle Trace or the SQL trace facility and `TKPROF` to examine individual SQL statement performance.

Creating the PLAN_TABLE Output Table

- Before issuing an `EXPLAIN PLAN` statement, you must have a table to hold it’s output.
- Use the SQL script `UTLXPLAN.SQL` to create a sample output table called `PLAN_TABLE` in your schema.
- `PLAN_TABLE` is the default table into which the `EXPLAIN PLAN` statement inserts rows describing execution plans.
- With multiple statements, you can specify a statement identifier and use that to identify your specific execution plan.
- You can specify the `INTO` clause to specify a different table.
- For example:

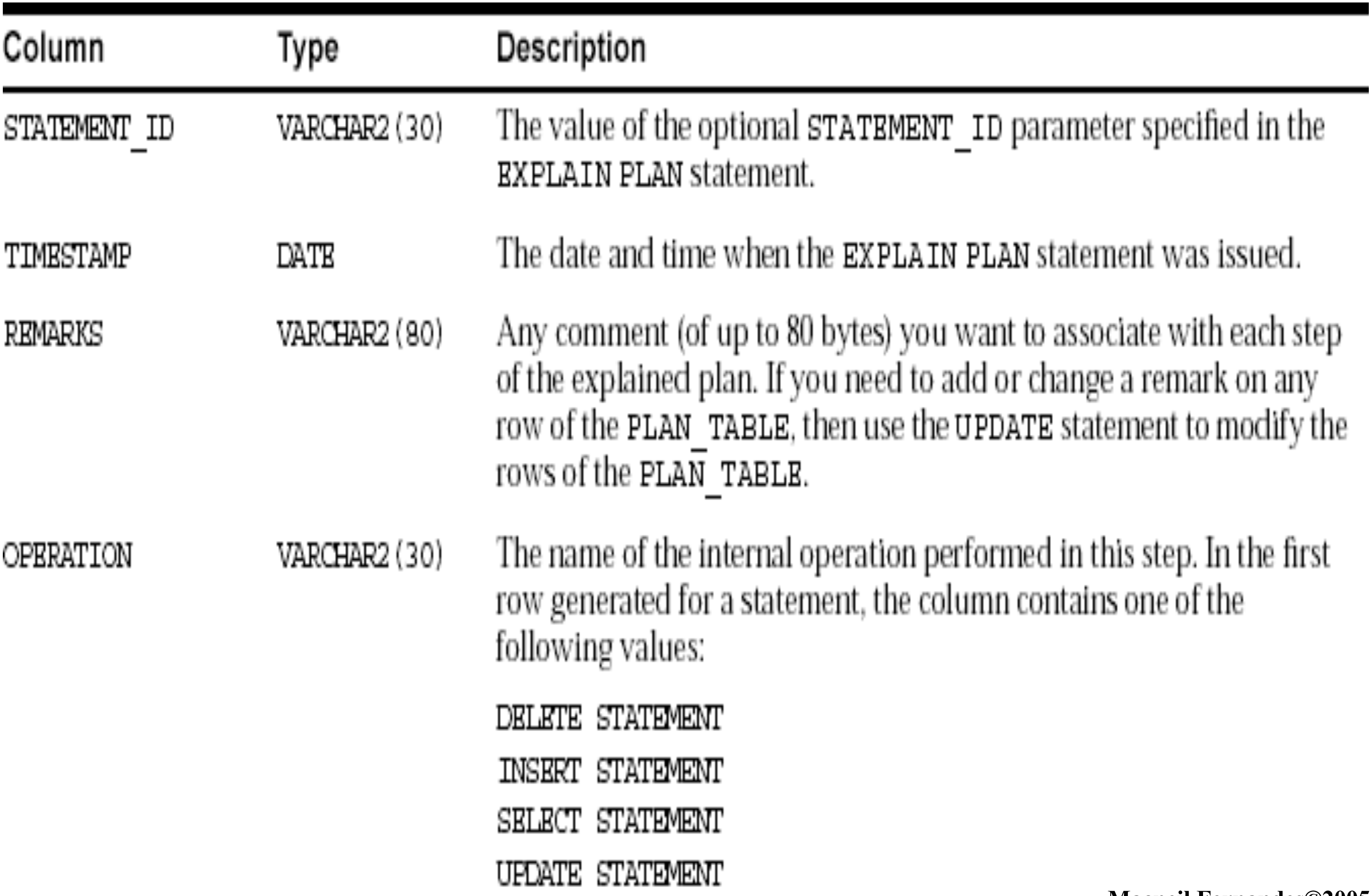
```
EXPLAIN PLAN
INTO my_plan_table
SET STATEMENT_ID = 'bad1' FOR
SELECT name FROM emp;
```

EXPLAIN PLAN Restrictions

- Oracle does not support `EXPLAIN PLAN` for statements performing implicit type conversion of date bind variables.

- With bind variables in general, the `EXPLAIN PLAN` output might not represent the real execution plan.

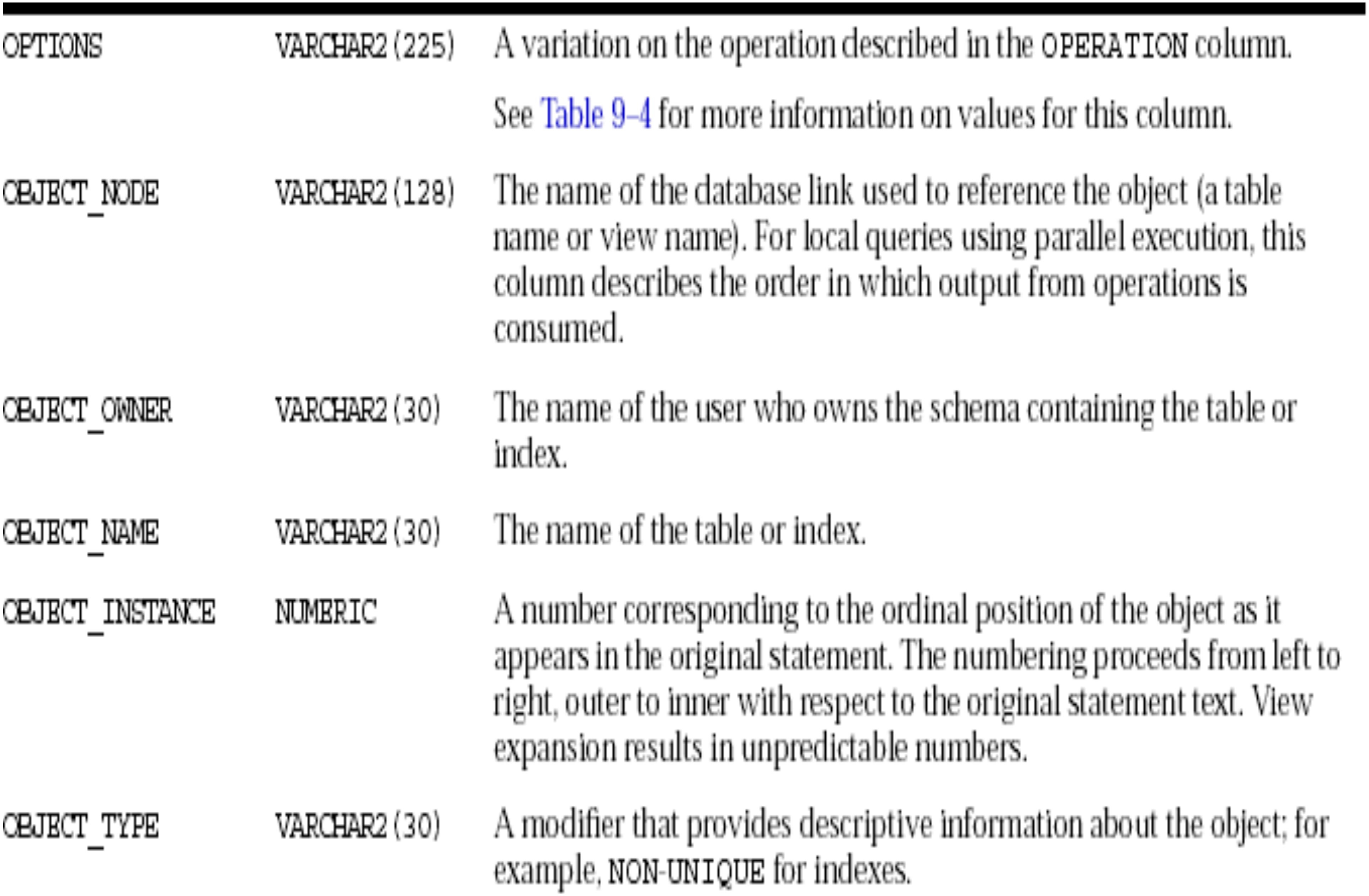
PLAN_TABLE Columns



Column	Type	Description
STATEMENT_ID	VARCHAR2(30)	The value of the optional STATEMENT_ID parameter specified in the EXPLAIN PLAN statement.
TIMESTAMP	DATE	The date and time when the EXPLAIN PLAN statement was issued.
REMARKS	VARCHAR2(80)	Any comment (of up to 80 bytes) you want to associate with each step of the explained plan. If you need to add or change a remark on any row of the PLAN_TABLE, then use the UPDATE statement to modify the rows of the PLAN_TABLE.

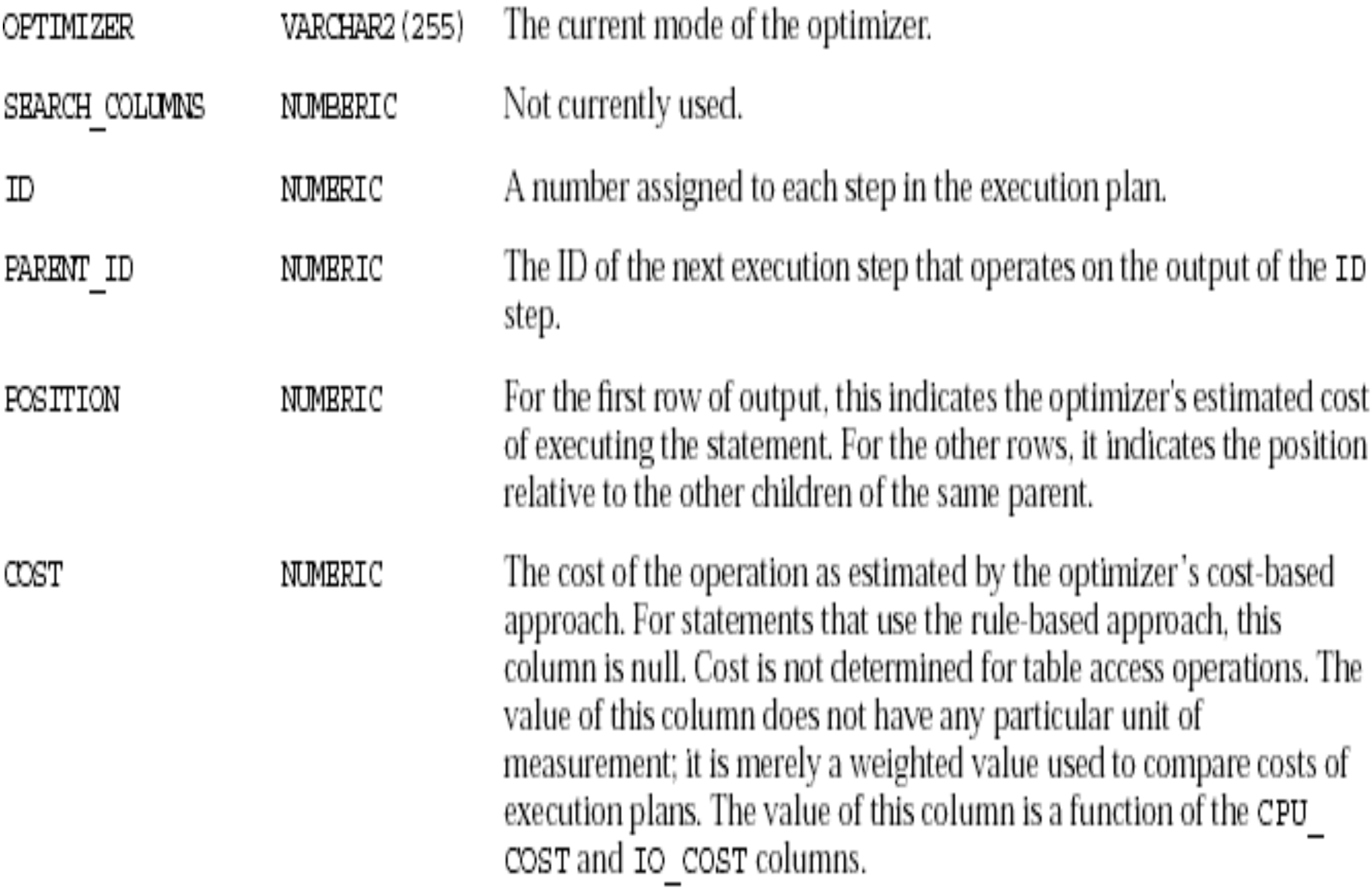
OPERATION	VARCHAR2(30)	The name of the internal operation performed in this step. In the first row generated for a statement, the column contains one of the following values:
		DELETE STATEMENT
		INSERT STATEMENT
		SELECT STATEMENT
		UPDATE STATEMENT

PLAN_TABLE Columns



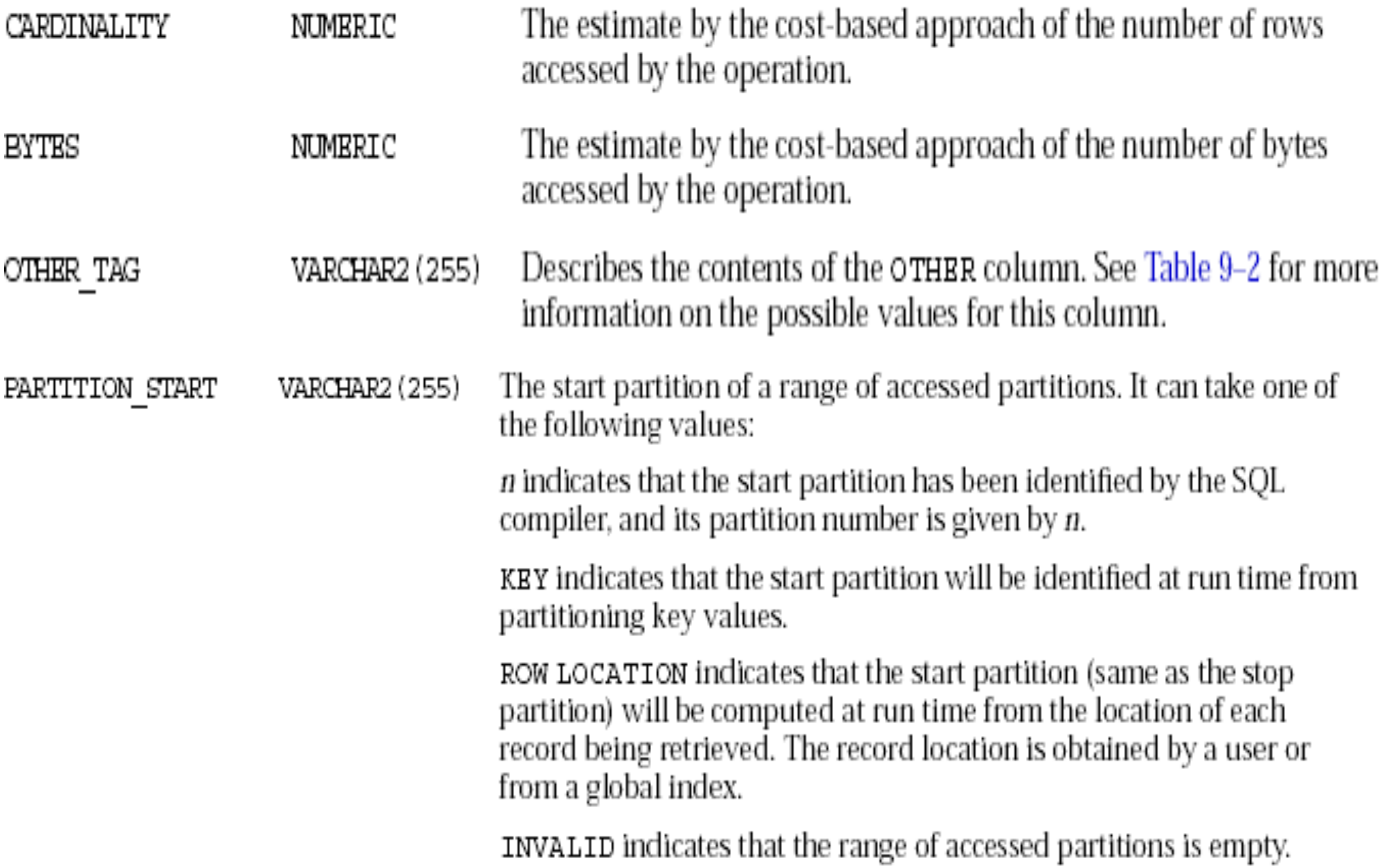
OPTIONS	VARCHAR2(225)	A variation on the operation described in the OPERATION column. See Table 9–4 for more information on values for this column.
OBJECT_NODE	VARCHAR2(128)	The name of the database link used to reference the object (a table name or view name). For local queries using parallel execution, this column describes the order in which output from operations is consumed.
OBJECT_OWNER	VARCHAR2(30)	The name of the user who owns the schema containing the table or index.
OBJECT_NAME	VARCHAR2(30)	The name of the table or index.
OBJECT_INSTANCE	NUMERIC	A number corresponding to the ordinal position of the object as it appears in the original statement. The numbering proceeds from left to right, outer to inner with respect to the original statement text. View expansion results in unpredictable numbers.
OBJECT_TYPE	VARCHAR2(30)	A modifier that provides descriptive information about the object; for example, NON-UNIQUE for indexes.

PLAN_TABLE Columns



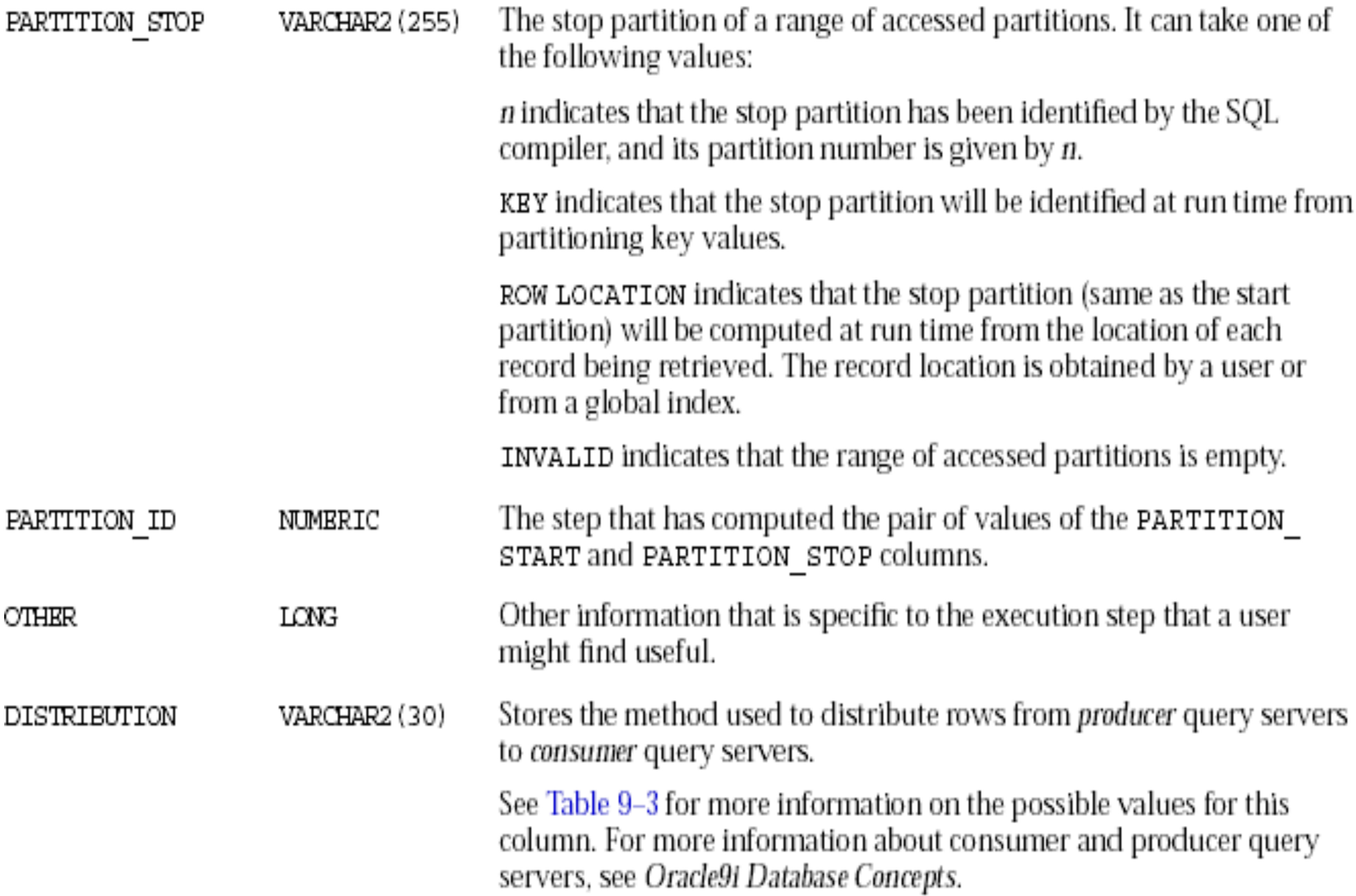
OPTIMIZER	VARCHAR2(255)	The current mode of the optimizer.
SEARCH_COLUMNS	NUMBERIC	Not currently used.
ID	NUMERIC	A number assigned to each step in the execution plan.
PARENT_ID	NUMERIC	The ID of the next execution step that operates on the output of the ID step.
POSITION	NUMERIC	For the first row of output, this indicates the optimizer's estimated cost of executing the statement. For the other rows, it indicates the position relative to the other children of the same parent.
COST	NUMERIC	The cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. Cost is not determined for table access operations. The value of this column does not have any particular unit of measurement; it is merely a weighted value used to compare costs of execution plans. The value of this column is a function of the CPU_COST and IO_COST columns.

PLAN_TABLE Columns



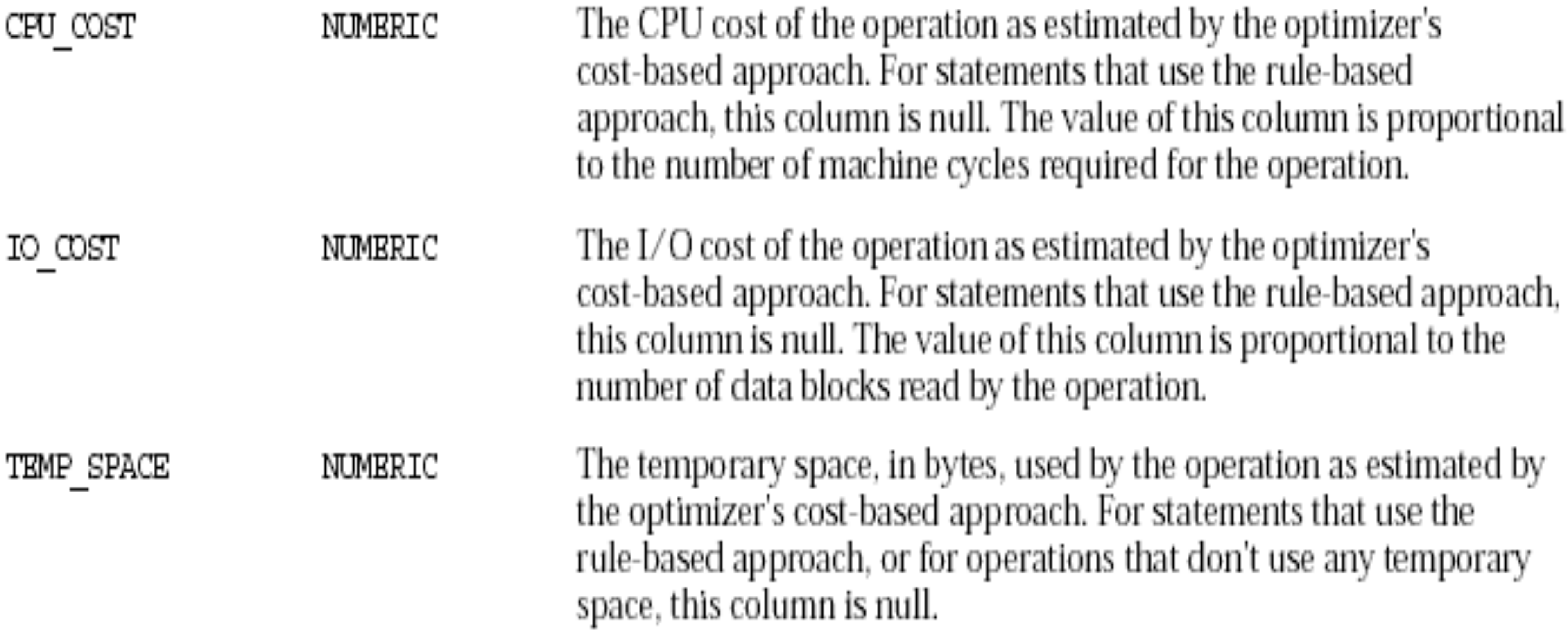
CARDINALITY	NUMERIC	The estimate by the cost-based approach of the number of rows accessed by the operation.
BYTES	NUMERIC	The estimate by the cost-based approach of the number of bytes accessed by the operation.
OTHER_TAG	VARCHAR2(255)	Describes the contents of the OTHER column. See Table 9–2 for more information on the possible values for this column.
PARTITION_START	VARCHAR2(255)	The start partition of a range of accessed partitions. It can take one of the following values: *n* indicates that the start partition has been identified by the SQL compiler, and its partition number is given by *n*. KEY indicates that the start partition will be identified at run time from partitioning key values. ROW LOCATION indicates that the start partition (same as the stop partition) will be computed at run time from the location of each record being retrieved. The record location is obtained by a user or from a global index. INVALID indicates that the range of accessed partitions is empty.

PLAN_TABLE Columns



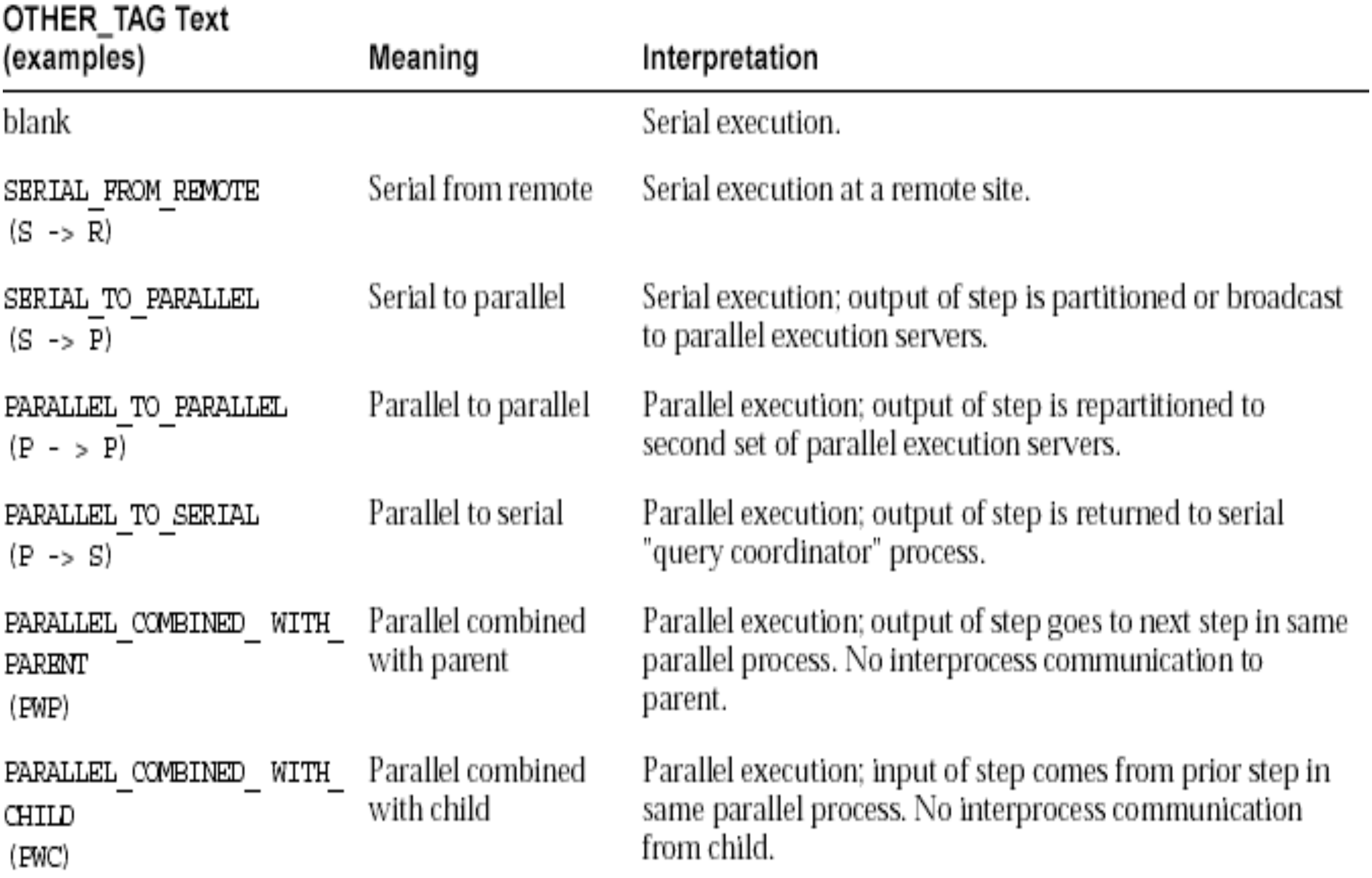
PARTITION_STOP	VARCHAR2(255)	The stop partition of a range of accessed partitions. It can take one of the following values: *n* indicates that the stop partition has been identified by the SQL compiler, and its partition number is given by *n*. KEY indicates that the stop partition will be identified at run time from partitioning key values. ROW LOCATION indicates that the stop partition (same as the start partition) will be computed at run time from the location of each record being retrieved. The record location is obtained by a user or from a global index. INVALID indicates that the range of accessed partitions is empty.
PARTITION_ID	NUMERIC	The step that has computed the pair of values of the PARTITION_START and PARTITION_STOP columns.
OTHER	LONG	Other information that is specific to the execution step that a user might find useful.
DISTRIBUTION	VARCHAR2(30)	Stores the method used to distribute rows from *producer* query servers to *consumer* query servers. See Table 9–3 for more information on the possible values for this column. For more information about consumer and producer query servers, see *Oracle9i Database Concepts*.

PLAN_TABLE Columns



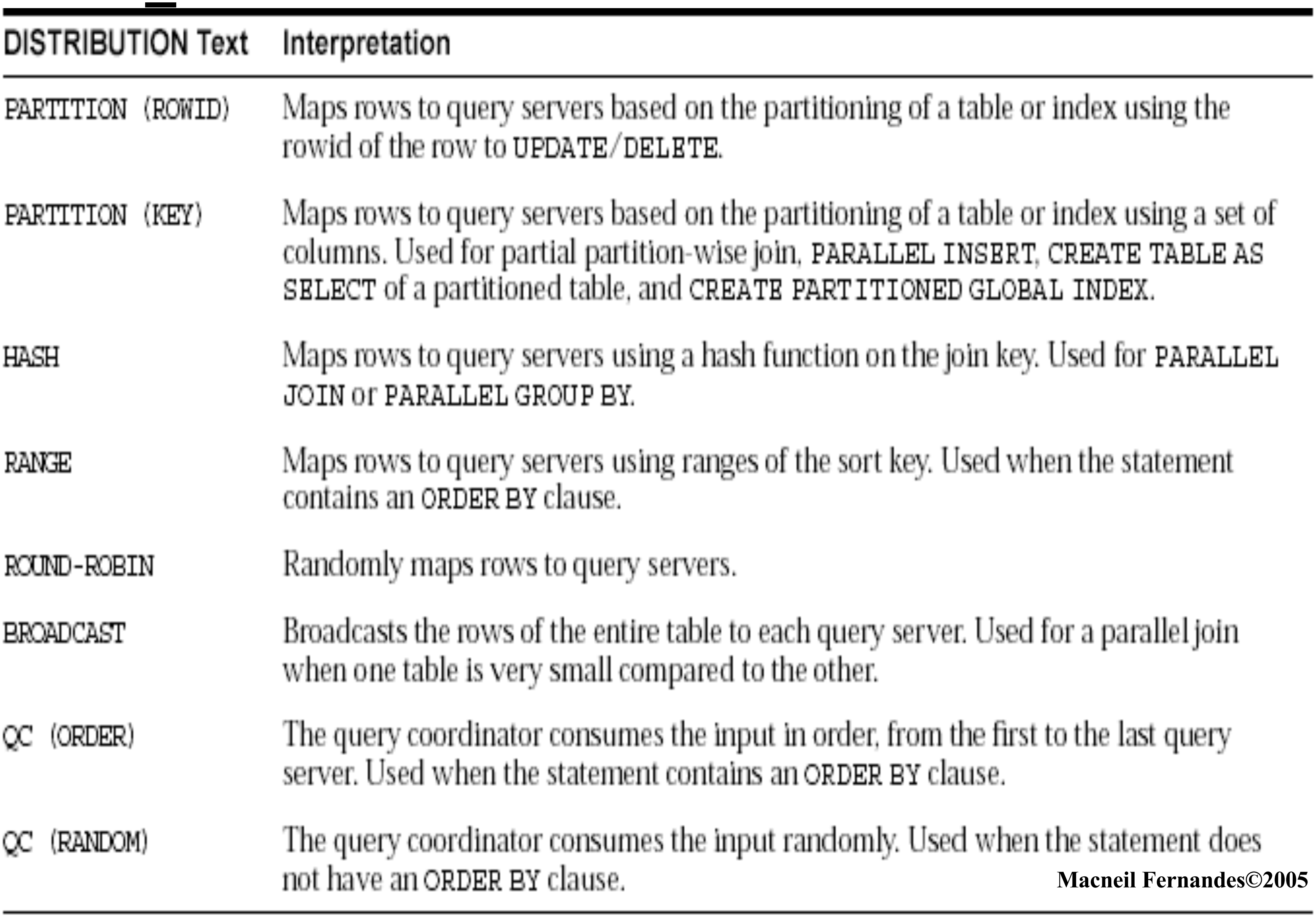
CPU_COST	NUMERIC	The CPU cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. The value of this column is proportional to the number of machine cycles required for the operation.
IO_COST	NUMERIC	The I/O cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. The value of this column is proportional to the number of data blocks read by the operation.
TEMP_SPACE	NUMERIC	The temporary space, in bytes, used by the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, or for operations that don't use any temporary space, this column is null.

Values of OTHER_TAG Column of the PLAN_TABLE



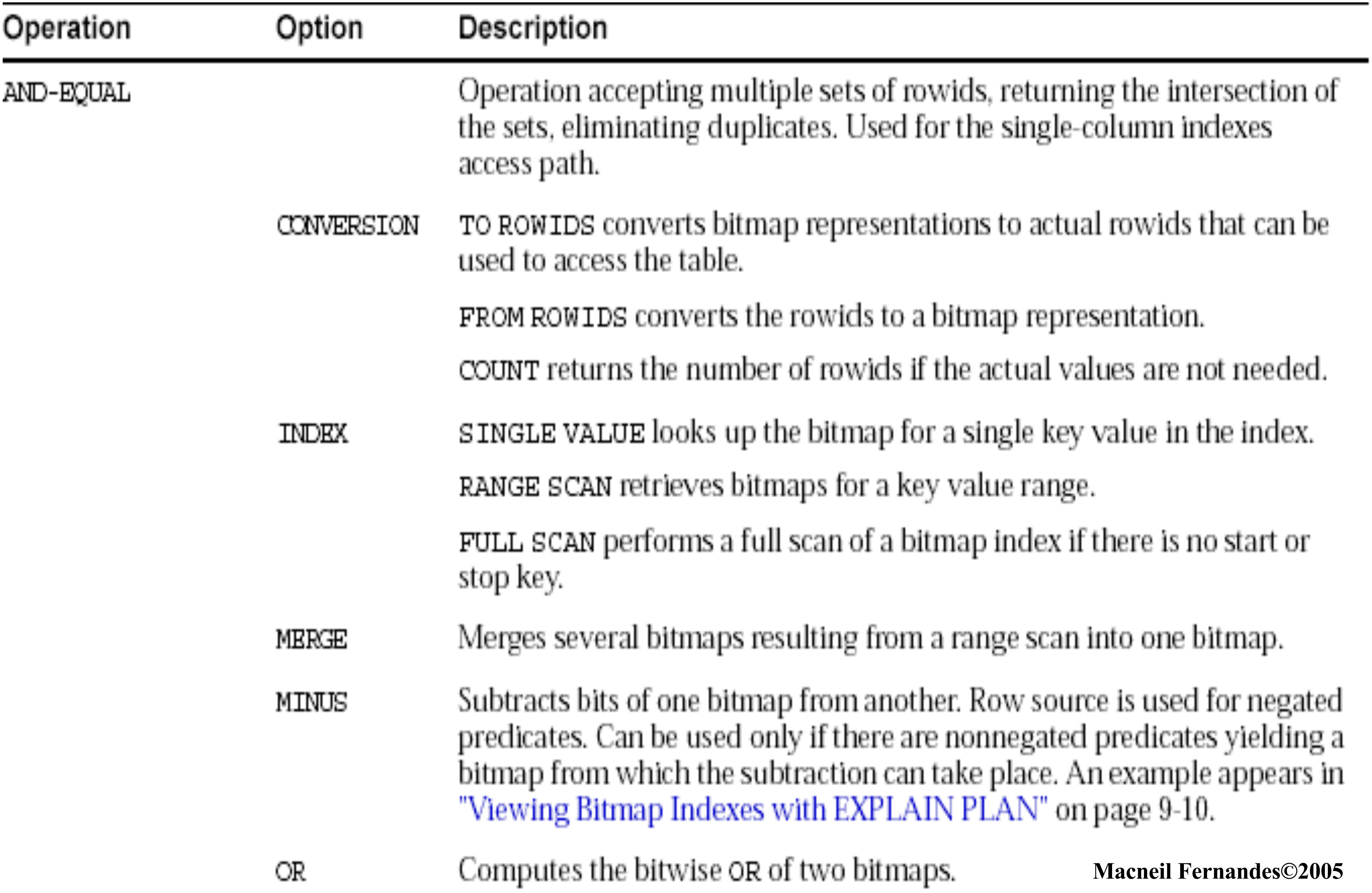
OTHER_TAG Text (examples)	Meaning	Interpretation
blank		Serial execution.
SERIAL_FROM_REMOTE (S -> R)	Serial from remote	Serial execution at a remote site.
SERIAL_TO_PARALLEL (S -> P)	Serial to parallel	Serial execution; output of step is partitioned or broadcast to parallel execution servers.
PARALLEL_TO_PARALLEL (P - > P)	Parallel to parallel	Parallel execution; output of step is repartitioned to second set of parallel execution servers.
PARALLEL_TO_SERIAL (P -> S)	Parallel to serial	Parallel execution; output of step is returned to serial "query coordinator" process.
PARALLEL_COMBINED_ WITH_ PARENT (PWP)	Parallel combined with parent	Parallel execution; output of step goes to next step in same parallel process. No interprocess communication to parent.
PARALLEL_COMBINED_ WITH_ CHILD (PWC)	Parallel combined with child	Parallel execution; input of step comes from prior step in same parallel process. No interprocess communication from child.

Values of DISTRIBUTION Column of the PLAN_TABLE



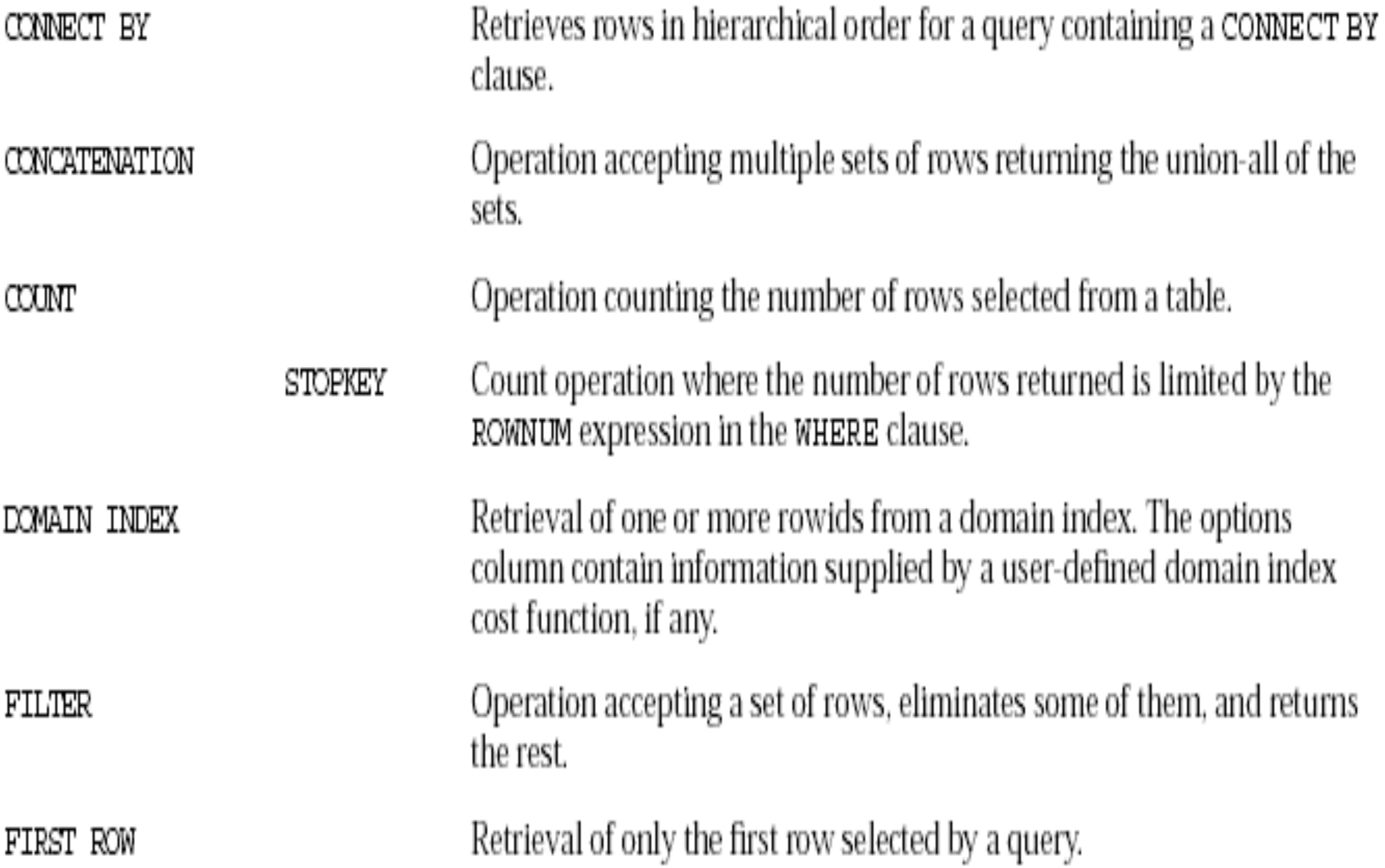
DISTRIBUTION Text	Interpretation
PARTITION (ROWID)	Maps rows to query servers based on the partitioning of a table or index using the rowid of the row to UPDATE/DELETE.
PARTITION (KEY)	Maps rows to query servers based on the partitioning of a table or index using a set of columns. Used for partial partition-wise join, PARALLEL INSERT, CREATE TABLE AS SELECT of a partitioned table, and CREATE PARTITIONED GLOBAL INDEX.
HASH	Maps rows to query servers using a hash function on the join key. Used for PARALLEL JOIN or PARALLEL GROUP BY.
RANGE	Maps rows to query servers using ranges of the sort key. Used when the statement contains an ORDER BY clause.
ROUND-ROBIN	Randomly maps rows to query servers.
BROADCAST	Broadcasts the rows of the entire table to each query server. Used for a parallel join when one table is very small compared to the other.
QC (ORDER)	The query coordinator consumes the input in order, from the first to the last query server. Used when the statement contains an ORDER BY clause.
QC (RANDOM)	The query coordinator consumes the input randomly. Used when the statement does not have an ORDER BY clause.

OPERATION and OPTION Values Produced by EXPLAIN PLAN



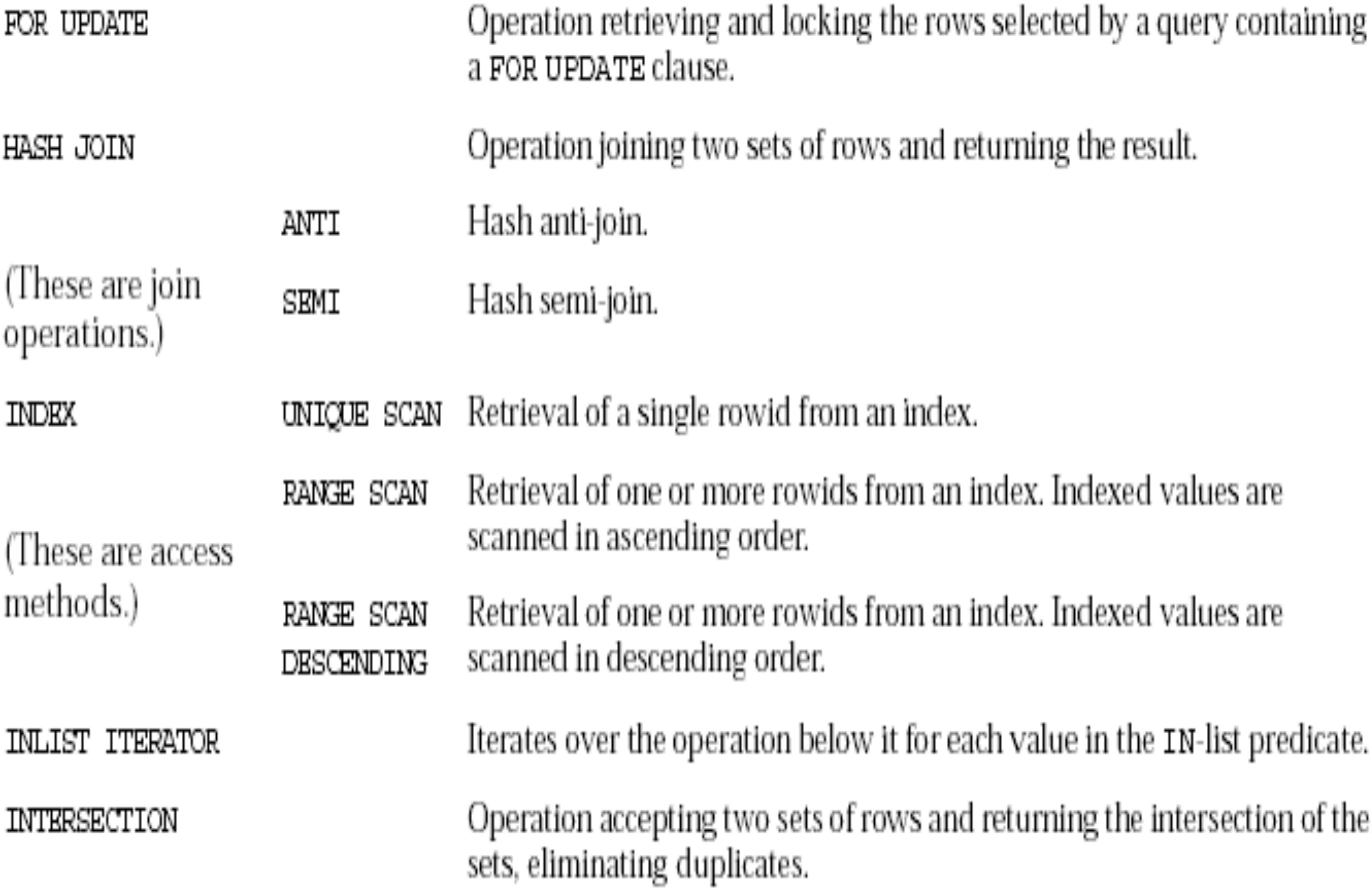
Operation	Option	Description
AND-EQUAL		Operation accepting multiple sets of rowids, returning the intersection of the sets, eliminating duplicates. Used for the single-column indexes access path.
	CONVERSION	TO ROWIDS converts bitmap representations to actual rowids that can be used to access the table. FROM ROWIDS converts the rowids to a bitmap representation. COUNT returns the number of rowids if the actual values are not needed.
	INDEX	SINGLE VALUE looks up the bitmap for a single key value in the index. RANGE SCAN retrieves bitmaps for a key value range. FULL SCAN performs a full scan of a bitmap index if there is no start or stop key.
	MERGE	Merges several bitmaps resulting from a range scan into one bitmap.
	MINUS	Subtracts bits of one bitmap from another. Row source is used for negated predicates. Can be used only if there are nonnegated predicates yielding a bitmap from which the subtraction can take place. An example appears in "Viewing Bitmap Indexes with EXPLAIN PLAN" on page 9-10.
	OR	Computes the bitwise OR of two bitmaps.

OPERATION and OPTION Values Produced by EXPLAIN PLAN



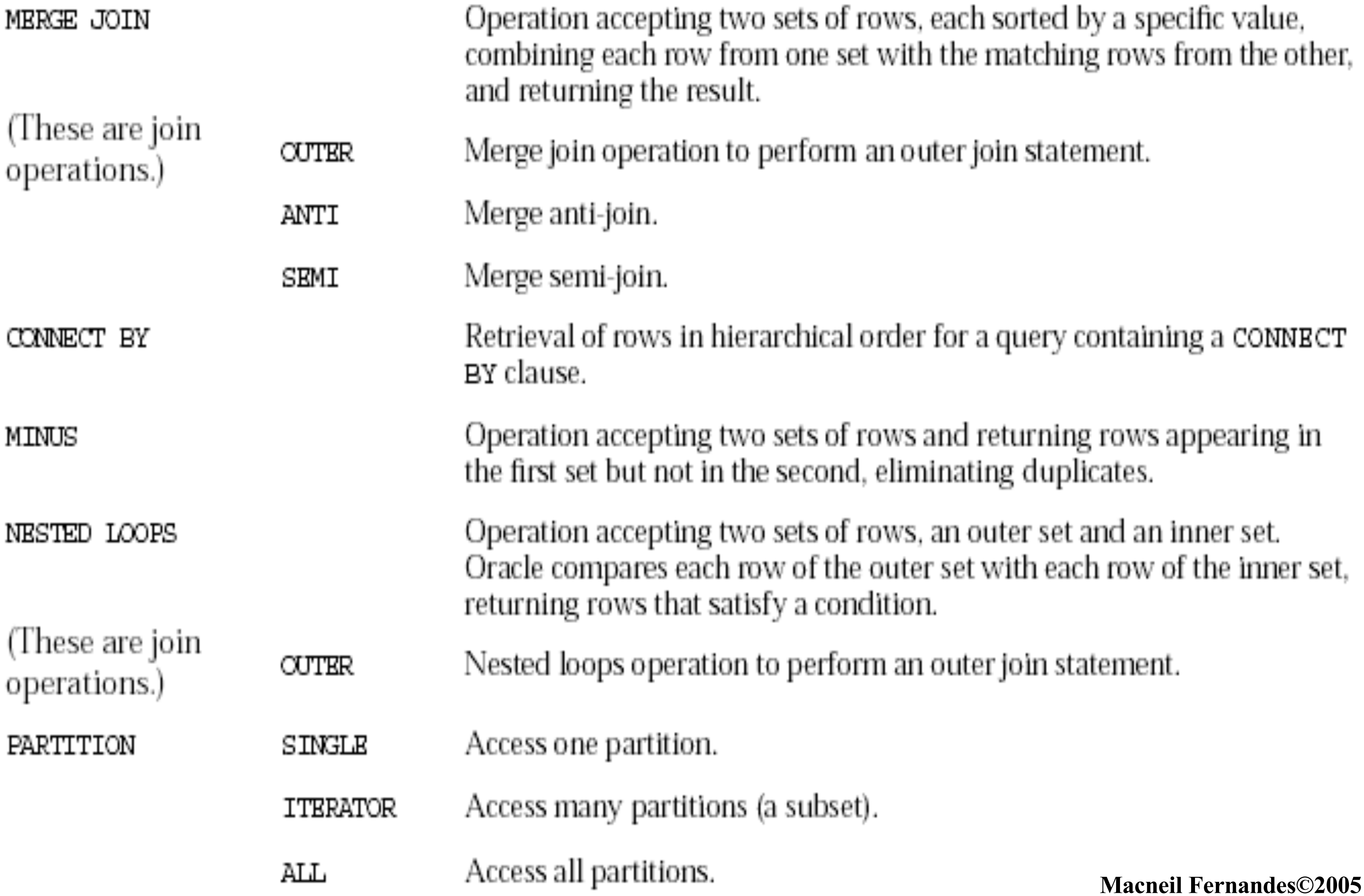
CONNECT BY		Retrieves rows in hierarchical order for a query containing a CONNECT BY clause.
CONCATENATION		Operation accepting multiple sets of rows returning the union-all of the sets.
COUNT		Operation counting the number of rows selected from a table.
	STOPKEY	Count operation where the number of rows returned is limited by the ROWNUM expression in the WHERE clause.
DOMAIN INDEX		Retrieval of one or more rowids from a domain index. The options column contain information supplied by a user-defined domain index cost function, if any.
FILTER		Operation accepting a set of rows, eliminates some of them, and returns the rest.
FIRST ROW		Retrieval of only the first row selected by a query.

OPERATION and OPTION Values Produced by EXPLAIN PLAN



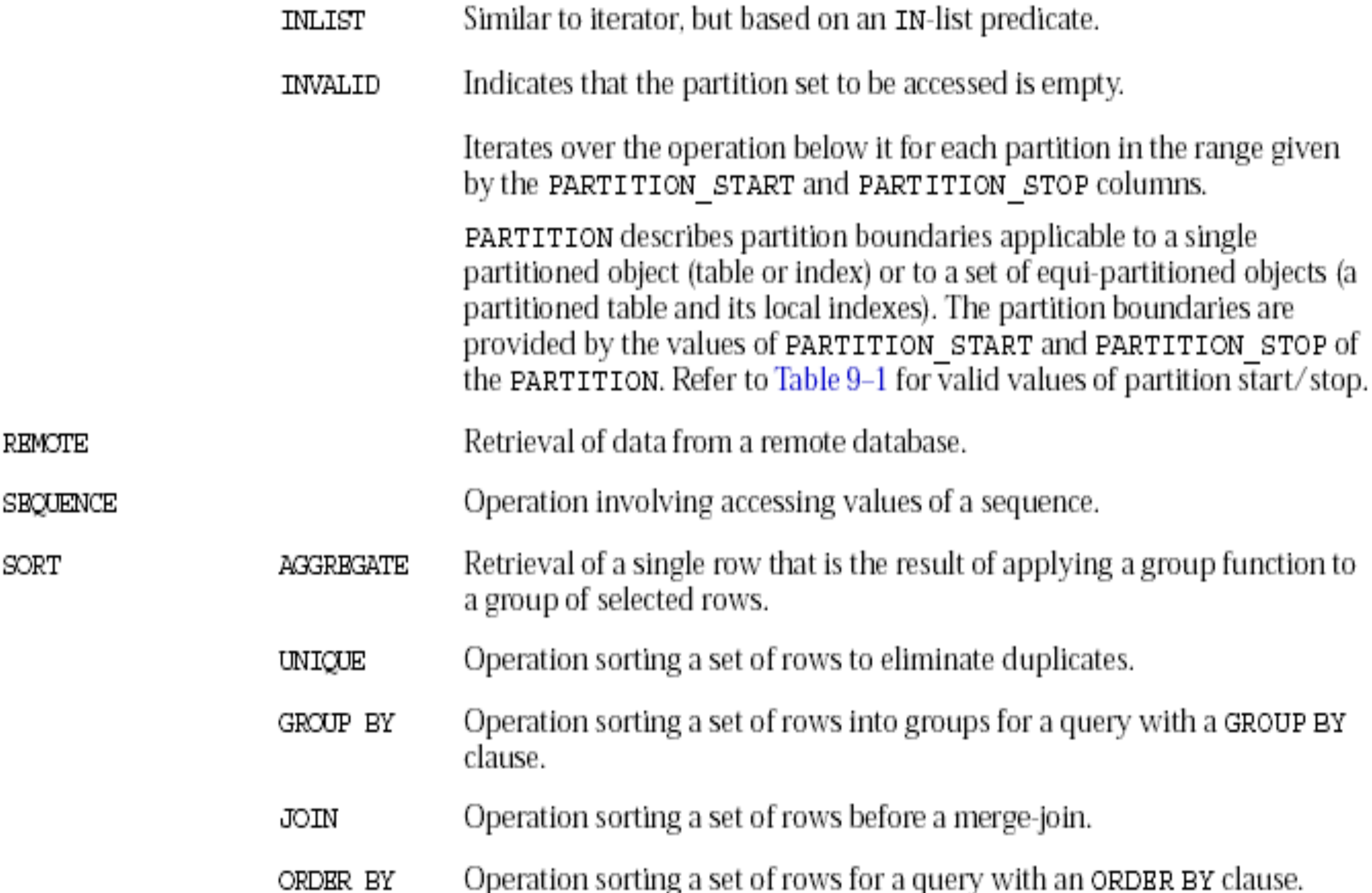
FOR UPDATE		Operation retrieving and locking the rows selected by a query containing a FOR UPDATE clause.
HASH JOIN		Operation joining two sets of rows and returning the result.
	ANTI	Hash anti-join.
(These are join operations.)	SEMI	Hash semi-join.
INDEX	UNIQUE SCAN	Retrieval of a single rowid from an index.
(These are access methods.)	RANGE SCAN	Retrieval of one or more rowids from an index. Indexed values are scanned in ascending order.
	RANGE SCAN DESCENDING	Retrieval of one or more rowids from an index. Indexed values are scanned in descending order.
INLIST ITERATOR		Iterates over the operation below it for each value in the IN-list predicate.
INTERSECTION		Operation accepting two sets of rows and returning the intersection of the sets, eliminating duplicates.

OPERATION and OPTION Values Produced by EXPLAIN PLAN



MERGE JOIN		Operation accepting two sets of rows, each sorted by a specific value, combining each row from one set with the matching rows from the other, and returning the result.
(These are join operations.)	OUTER	Merge join operation to perform an outer join statement.
	ANTI	Merge anti-join.
	SEMI	Merge semi-join.
CONNECT BY		Retrieval of rows in hierarchical order for a query containing a CONNECT BY clause.
MINUS		Operation accepting two sets of rows and returning rows appearing in the first set but not in the second, eliminating duplicates.
NESTED LOOPS		Operation accepting two sets of rows, an outer set and an inner set. Oracle compares each row of the outer set with each row of the inner set, returning rows that satisfy a condition.
(These are join operations.)	OUTER	Nested loops operation to perform an outer join statement.
PARTITION	SINGLE	Access one partition.
	ITERATOR	Access many partitions (a subset).
	ALL	Access all partitions.

OPERATION and OPTION Values Produced by EXPLAIN PLAN



	INLIST	Similar to iterator, but based on an IN-list predicate.
	INVALID	Indicates that the partition set to be accessed is empty.
		Iterates over the operation below it for each partition in the range given by the PARTITION_START and PARTITION_STOP columns.
		PARTITION describes partition boundaries applicable to a single partitioned object (table or index) or to a set of equi-partitioned objects (a partitioned table and its local indexes). The partition boundaries are provided by the values of PARTITION_START and PARTITION_STOP of the PARTITION. Refer to Table 9–1 for valid values of partition start/stop.
REMOTE		Retrieval of data from a remote database.
SEQUENCE		Operation involving accessing values of a sequence.
SORT	AGGREGATE	Retrieval of a single row that is the result of applying a group function to a group of selected rows.
	UNIQUE	Operation sorting a set of rows to eliminate duplicates.
	GROUP BY	Operation sorting a set of rows into groups for a query with a GROUP BY clause.
	JOIN	Operation sorting a set of rows before a merge-join.
	ORDER BY	Operation sorting a set of rows for a query with an ORDER BY clause.

OPERATION and OPTION Values Produced by EXPLAIN PLAN

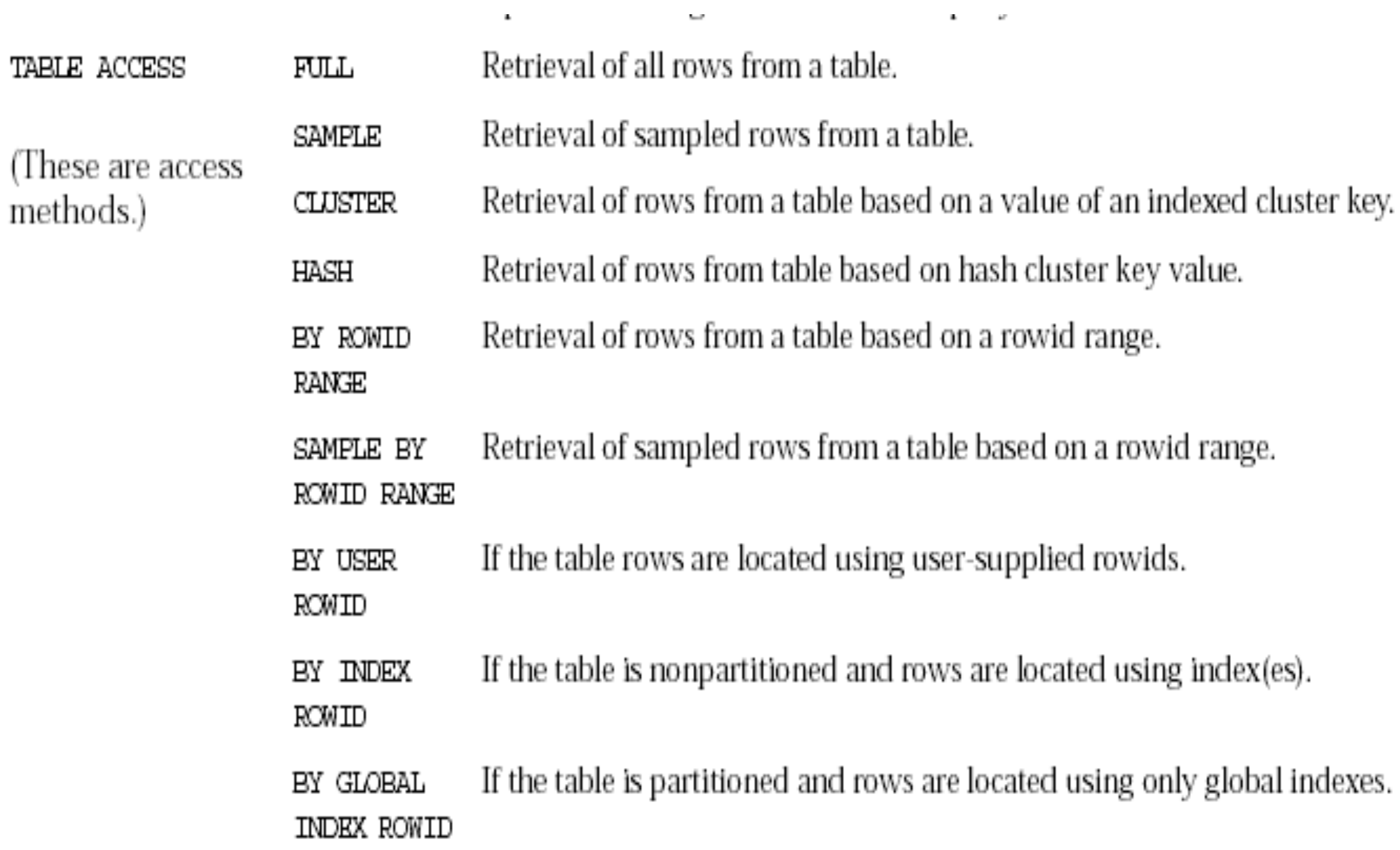
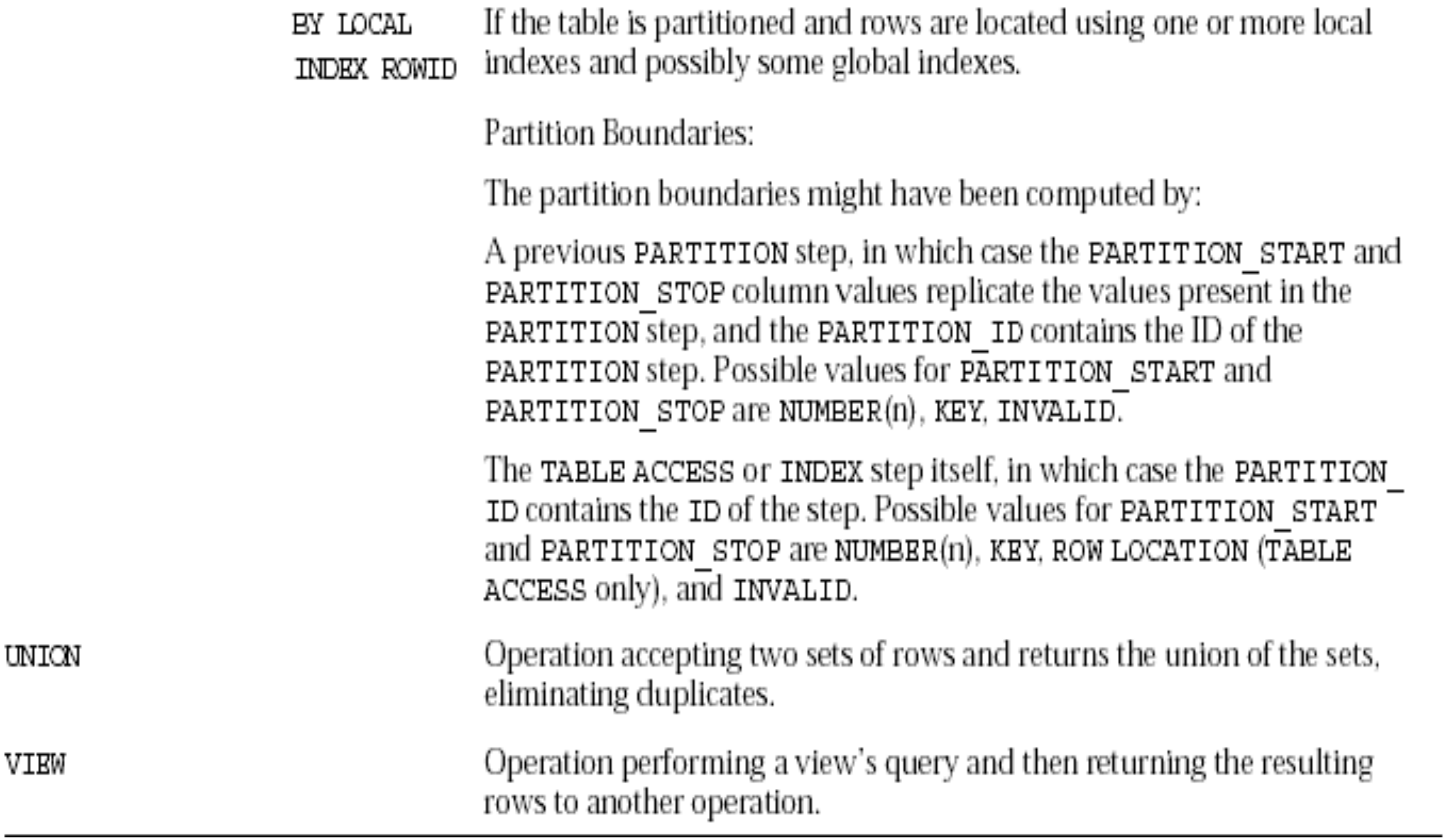


TABLE ACCESS (These are access methods.)	FULL	Retrieval of all rows from a table.
	SAMPLE	Retrieval of sampled rows from a table.
	CLUSTER	Retrieval of rows from a table based on a value of an indexed cluster key.
	HASH	Retrieval of rows from table based on hash cluster key value.
	BY ROWID RANGE	Retrieval of rows from a table based on a rowid range.
	SAMPLE BY ROWID RANGE	Retrieval of sampled rows from a table based on a rowid range.
	BY USER ROWID	If the table rows are located using user-supplied rowids.
	BY INDEX ROWID	If the table is nonpartitioned and rows are located using index(es).
	BY GLOBAL INDEX ROWID	If the table is partitioned and rows are located using only global indexes.

OPERATION and OPTION Values Produced by EXPLAIN PLAN



	BY LOCAL INDEX ROWID	If the table is partitioned and rows are located using one or more local indexes and possibly some global indexes. Partition Boundaries: The partition boundaries might have been computed by: A previous PARTITION step, in which case the PARTITION_START and PARTITION_STOP column values replicate the values present in the PARTITION step, and the PARTITION_ID contains the ID of the PARTITION step. Possible values for PARTITION_START and PARTITION_STOP are NUMBER(n), KEY, INVALID. The TABLE ACCESS or INDEX step itself, in which case the PARTITION_ID contains the ID of the step. Possible values for PARTITION_START and PARTITION_STOP are NUMBER(n), KEY, ROW LOCATION (TABLE ACCESS only), and INVALID.
UNION		Operation accepting two sets of rows and returns the union of the sets, eliminating duplicates.
VIEW		Operation performing a view's query and then returning the resulting rows to another operation.