

Table 19–1 (Cont.) PLAN_TABLE Columns

Column	Type	Description
PROJECTION	VARCHAR2 (4000)	Expressions produced by the operation.
TIME	NUMBER (20, 2)	Elapsed time in seconds of the operation as estimated by query optimization. For statements that use the rule-based approach, this column is null.
QBLOCK_NAME	VARCHAR2 (30)	Name of the query block, either system-generated or defined by the user with the QB_NAME hint.

Table 19–2 describes the values that can appear in the DISTRIBUTION column:

Table 19–2 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 (QC) 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 (QC) consumes the input randomly. Used when the statement does not have an ORDER BY clause.

Table 19–3 lists each combination of OPERATION and OPTIONS produced by the EXPLAIN PLAN statement and its meaning within an execution plan.

Table 19–3 OPERATION and OPTIONS 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.
BITMAP	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.
BITMAP	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.

Table 19-3 (Cont.) OPERATION and OPTIONS Values Produced by EXPLAIN PLAN

Operation	Option	Description
BITMAP	MERGE	Merges several bitmaps resulting from a range scan into one bitmap.
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 19-9.
BITMAP	OR	Computes the bitwise OR of two bitmaps.
BITMAP	AND	Computes the bitwise AND of two bitmaps.
BITMAP	KEY ITERATION	Takes each row from a table row source and finds the corresponding bitmap from a bitmap index. This set of bitmaps are then merged into one bitmap in a following BITMAP MERGE operation.
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.
COUNT	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.
FOR UPDATE	.	Operation retrieving and locking the rows selected by a query containing a FOR UPDATE clause.
HASH	GROUP BY	Operation hashing a set of rows into groups for a query with a GROUP BY clause.
HASH JOIN (These are join operations.)	.	Operation joining two sets of rows and returning the result. This join method is useful for joining large data sets of data (DSS, Batch). The join condition is an efficient way of accessing the second table. Query optimizer uses the smaller of the two tables/data sources to build a hash table on the join key in memory. Then it scans the larger table, probing the hash table to find the joined rows.
HASH JOIN	ANTI	Hash (left) antijoin
HASH JOIN	SEMI	Hash (left) semijoin
HASH JOIN	RIGHT ANTI	Hash right antijoin
HASH JOIN	RIGHT SEMI	Hash right semijoin
HASH JOIN	OUTER	Hash (left) outer join
HASH JOIN	RIGHT OUTER	Hash right outer join
INDEX (These are access methods.)	UNIQUE SCAN	Retrieval of a single rowid from an index.

Table 19–3 (Cont.) OPERATION and OPTIONS Values Produced by EXPLAIN PLAN

Operation	Option	Description
INDEX	RANGE SCAN	Retrieval of one or more rowids from an index. Indexed values are scanned in ascending order.
INDEX	RANGE SCAN DESCENDING	Retrieval of one or more rowids from an index. Indexed values are scanned in descending order.
INDEX	FULL SCAN	Retrieval of all rowids from an index when there is no start or stop key. Indexed values are scanned in ascending order.
INDEX	FULL SCAN DESCENDING	Retrieval of all rowids from an index when there is no start or stop key. Indexed values are scanned in descending order.
INDEX	FAST FULL SCAN	Retrieval of all rowids (and column values) using multiblock reads. No sorting order can be defined. Compares to a full table scan on only the indexed columns. Only available with the cost based optimizer.
INDEX	SKIP SCAN	Retrieval of rowids from a concatenated index without using the leading column(s) in the index. Introduced in Oracle9i. Only available with the cost based optimizer.
INLIST ITERATOR	.	Iterates over the next operation in the plan for each value in the IN-list predicate.
INTERSECTION	.	Operation accepting two sets of rows and returning the intersection of the sets, eliminating duplicates.
MERGE JOIN (These are join operations.)	.	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.
MERGE JOIN	OUTER	Merge join operation to perform an outer join statement.
MERGE JOIN	ANTI	Merge antijoin.
MERGE JOIN	SEMI	Merge semijoin.
MERGE JOIN	CARTESIAN	Can result from 1 or more of the tables not having any join conditions to any other tables in the statement. Can occur even with a join and it may not be flagged as CARTESIAN in the plan.
CONNECT BY	.	Retrieval of rows in hierarchical order for a query containing a CONNECT BY clause.
MAT_VIEW REWITE ACCESS (These are access methods.)	FULL	Retrieval of all rows from a materialized view.
MAT_VIEW REWITE ACCESS	SAMPLE	Retrieval of sampled rows from a materialized view.
MAT_VIEW REWITE ACCESS	CLUSTER	Retrieval of rows from a materialized view based on a value of an indexed cluster key.
MAT_VIEW REWITE ACCESS	HASH	Retrieval of rows from materialized view based on hash cluster key value.
MAT_VIEW REWITE ACCESS	BY ROWID RANGE	Retrieval of rows from a materialized view based on a rowid range.
MAT_VIEW REWITE ACCESS	SAMPLE BY ROWID RANGE	Retrieval of sampled rows from a materialized view based on a rowid range.
MAT_VIEW REWITE ACCESS	BY USER ROWID	If the materialized view rows are located using user-supplied rowids.

Table 19–3 (Cont.) OPERATION and OPTIONS Values Produced by EXPLAIN PLAN

Operation	Option	Description
MAT_VIEW REWRITE ACCESS	BY INDEX ROWID	If the materialized view is nonpartitioned and rows are located using index(es).
MAT_VIEW REWRITE ACCESS	BY GLOBAL INDEX ROWID	If the materialized view is partitioned and rows are located using only global indexes.
MAT_VIEW REWRITE ACCESS	BY LOCAL INDEX ROWID	If the materialized view 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 MAT_VIEW REWRITE 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 REMOVE_LOCATION (MAT_VIEW REWRITE ACCESS only), and INVALID.
MINUS	.	Operation accepting two sets of rows and returning rows appearing in the first set but not in the second, eliminating duplicates.
NESTED LOOPS (These are join operations.)	.	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. This join method is useful for joining small subsets of data (OLTP). The join condition is an efficient way of accessing the second table.
NESTED LOOPS	OUTER	Nested loops operation to perform an outer join statement.
PARTITION	.	Iterates over the next operation in the plan 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 19–1 for valid values of partition start/stop.
PARTITION	SINGLE	Access one partition.
PARTITION	ITERATOR	Access many partitions (a subset).
PARTITION	ALL	Access all partitions.
PARTITION	INLIST	Similar to iterator, but based on an IN-list predicate.
PARTITION	INVALID	Indicates that the partition set to be accessed is empty.
PX ITERATOR	BLOCK, CHUNK	Implements the division of an object into block or chunk ranges among a set of parallel slaves
PX COORDINATOR	.	Implements the Query Coordinator which controls, schedules, and executes the parallel plan below it using parallel query slaves. It also represents a serialization point, as the end of the part of the plan executed in parallel and always has a PX SEND QC operation below it.
PX PARTITION	.	Same semantics as the regular PARTITION operation except that it appears in a parallel plan

Table 19–3 (Cont.) OPERATION and OPTIONS Values Produced by EXPLAIN PLAN

Operation	Option	Description
PX RECEIVE	.	Shows the consumer/receiver slave node reading repartitioned data from a send/producer (QC or slave) executing on a PX SEND node. This information was formerly displayed into the DISTRIBUTION column. See Table 19–2 on page 19-20.
PX SEND	QC (RANDOM) , HASH, RANGE	Implements the distribution method taking place between two parallel set of slaves. Shows the boundary between two slave sets and how data is repartitioned on the send/producer side (QC or side). This information was formerly displayed into the DISTRIBUTION column. See Table 19–2 on page 19-20.
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.
SORT	UNIQUE	Operation sorting a set of rows to eliminate duplicates.
SORT	GROUP BY	Operation sorting a set of rows into groups for a query with a GROUP BY clause.
SORT	JOIN	Operation sorting a set of rows before a merge-join.
SORT	ORDER BY	Operation sorting a set of rows for a query with an ORDER BY clause.
TABLE ACCESS (These are access methods.)	FULL	Retrieval of all rows from a table.
TABLE ACCESS	SAMPLE	Retrieval of sampled rows from a table.
TABLE ACCESS	CLUSTER	Retrieval of rows from a table based on a value of an indexed cluster key.
TABLE ACCESS	HASH	Retrieval of rows from table based on hash cluster key value.
TABLE ACCESS	BY ROWID RANGE	Retrieval of rows from a table based on a rowid range.
TABLE ACCESS	SAMPLE BY ROWID RANGE	Retrieval of sampled rows from a table based on a rowid range.
TABLE ACCESS	BY USER ROWID	If the table rows are located using user-supplied rowids.
TABLE ACCESS	BY INDEX ROWID	If the table is nonpartitioned and rows are located using index(es).
TABLE ACCESS	BY GLOBAL INDEX ROWID	If the table is partitioned and rows are located using only global indexes.

Table 19–3 (Cont.) OPERATION and OPTIONS Values Produced by EXPLAIN PLAN

Operation	Option	Description
TABLE ACCESS	BY LOCAL INDEX ROWID	<p>If the table is partitioned and rows are located using one or more local indexes and possibly some global indexes.</p> <p>Partition Boundaries:</p> <p>The partition boundaries might have been computed by:</p> <p>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.</p> <p>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 REMOVE_LOCATION (TABLE ACCESS only), and INVALID.</p>
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.

See Also: *Oracle Database Reference* for more information on PLAN_TABLE