Oracle SQL Hints

Wei Huang (Fuyuncat@gmail.com)
www.HelloDBA.com

ORACLE SQL HINTS.....1

SQL HINT description and demonstration.....10

SQL Features.....10

Access Path Hints.....13

CLUSTER.....13

HASH.....13

ROWID.....13

FULL.....13

INDEX.....14

INDEX_ASC.....14

INDEX_DESC.....14

NO_INDEX.....15

INDEX_FFS.....15

NO_INDEX_FFS.....15

INDEX_RRS.....16

INDEX_SS.....16

INDEX_SS_ASC.....16

INDEX_SS_DESC.....17

NO_INDEX_SS.....17

INDEX_RS_ASC.....17

INDEX_RS_DESC.....18

AND_EQUAL.....18

INDEX_COMBINE.....18

INDEX_JOIN.....19

BITMAP_TREE.....19

USE_INVISIBLE_INDEXES.....20

NO_USE_INVISIBLE_INDEXES.....20

Join Hints.....21

NL_AJ.....21

HASH_AJ.....21

MERGE_AJ.....21

USE_HASH.....22

NO_USE_HASH.....22

USE_MERGE.....22

NO_USE_MERGE.....23

USE_NL.....23

USE_NL_WITH_INDEX.....23

NO_USE_NL.....24

USE_MERGE_CARTESIAN 24
LEADING 25
USE_ANTI 25
USE_SEMI 25
NATIVE_FULL_OUTER_JOIN 26
NO_NATIVE_FULL_OUTER_JOIN 26
NO_CARTESIAN 27
ORDERED 28
PX_JOIN_FILTER 28
NO_PX_JOIN_FILTER 28
NL_SJ 29
HASH_SJ 29
MERGE_SJ 29
NO_SEMIJOIN 30
SEMIJOIN 30
SWAP_JOIN_INPUTS 31
NO_SWAP_JOIN_INPUTS 31
Outline Data Hints 32
QB_NAME 32
DB_VERSION 32
IGNORE_OPTIM_EMBEDDED_HINTS 32
IGNORE_WHERE_CLAUSE 33
NO_ACCESS 33
OPTIMIZER_FEATURES_ENABLE 34
OPT_PARAM 34
OUTLINE 34
OUTLINE_LEAF 35
RBO_OUTLINE 35
Query Transformation Hints 35
ANTIJOIN 35
COALESCE_SQ 35
NO_COALESCE_SQ 36
ELIMINATE_JOIN 36
NO_ELIMINATE_JOIN 37
ELIMINATE_OBY 37
NO_ELIMINATE_OBY 37
EXPAND_GSET_TO_UNION 38
NO_EXPAND_GSET_TO_UNION 38
EXPAND_TABLE 38

NO_EXPAND_TABLE 39
STAR 39
STAR_TRANSFORMATION 40
NO_STAR_TRANSFORMATION 41
FACT 41
NO_FACT 42
FACTORIZE_JOIN 42
NO_FACTORIZE_JOIN 42
MATERIALIZE 43
INLINE 43
MERGE 44
NO_MERGE 44
NO_PRUNE_GSETS 45
NO_QUERY_TRANSFORMATION 45
OR_EXPAND 45
OUTER_JOIN_TO_INNER 46
NO_OUTER_JOIN_TO_INNER 46
ELIMINATE_OUTER_JOIN 46
NO_ELIMINATE_OUTER_JOIN 47
PLACE_DISTINCT 47
NO_PLACE_DISTINCT 48
PLACE_GROUP_BY 48
NO_PLACE_GROUP_BY 48
PRECOMPUTE_SUBQUERY 49
PULL_PRED 49
NO_PULL_PRED 50
OLD_PUSH_PRED 51
PUSH_PRED 52
NO_PUSH_PRED 52
PUSH_SUBQ 53
NO_PUSH_SUBQ 53
REWRITE 54
NO_REWRITE 54
NO_MULTIMV_REWRITE 55
NO_BASETABLE_MULTIMV_REWRITE 56
REWRITE_OR_ERROR 56
SET_TO_JOIN 56
NO_SET_TO_JOIN 57
TRANSFORM_DISTINCT_AGG 57

NO_TRANSFORM_DISTINCT_AGG....58
UNNEST....58
NO_UNNEST....58
USE_CONCAT....59
NO_EXPAND....59
USE_TTT_FOR_GSETS....60
NO_ORDER_ROLLUPS....60
OPAQUE_XCANONICAL....60
LIKE_EXPAND....60
Statistics Data Hints....60
CARDINALITY....60
CPU_COSTING....61
NO_CPU_COSTING....61
DBMS_STATS....61
DYNAMIC_SAMPLING....61
DYNAMIC_SAMPLING_EST_CDN....62
GATHER_PLAN_STATISTICS....62
NO_STATS_GSETS....62
OPT_ESTIMATE....63
COLUMN_STATS....63
INDEX_STATS....64
TABLE_STATS....64
Optimizer Hints....64
NUM_INDEX_KEYS....64
BIND_AWARE....65
NO_BIND_AWARE....65
CURSOR_SHARING_EXACT....66
DML_UPDATE....66
FBTSCAN....67
ALL_ROWS....67
FIRST_ROWS....68
RULE....68
CHOOSE....68
ORDERED_PREDICATES....69
SKIP_EXT_OPTIMIZER....70
SKIP_UNQ_UNUSABLE_IDX....70
Run Time Hints....70
APPEND....70
APPEND_VALUES....71

NOAPPEND 71
NLJ_BATCHING 71
NO_NLJ_BATCHING 72
NLJ_PREFETCH 72
NO_NLJ_PREFETCH 72
CACHE 73
NOCACHE 73
CACHE_TEMP_TABLE 74
NO_LOAD 74
NO_SUBSTRB_PAD 75
RESULT_CACHE 75
NO_RESULT_CACHE 76
SYS_DL_CURSOR 76
TRACING 77
Data Manipulate Hints 77
CHANGE_DUPKEY_ERROR_INDEX 77
IGNORE_ROW_ON_DUPKEY_INDEX 78
RETRY_ON_ROW_CHANGE 78
Hierarchy Query Hints 80
CONNECT_BY_CB_WHR_ONLY 80
NO_CONNECT_BY_CB_WHR_ONLY 80
CONNECT_BY_COMBINE_SW 80
NO_CONNECT_BY_COMBINE_SW 81
CONNECT_BY_COST_BASED 81
NO_CONNECT_BY_COST_BASED 82
CONNECT_BY_ELIM_DUPS 82
NO_CONNECT_BY_ELIM_DUPS 83
CONNECT_BY_FILTERING 83
NO_CONNECT_BY_FILTERING 83
XML 查询提示 84
COST_XML_QUERY_REWRITE 84
NO_COST_XML_QUERY_REWRITE 85
FORCE_XML_QUERY_REWRITE 85
NO_XML_QUERY_REWRITE 85
XML_DML_RWT_STMT 86
NO_XML_DML_REWRITE 87
XMLINDEX_REWRITE 87
NO_XMLINDEX_REWRITE 87
XMLINDEX_REWRITE_IN_SELECT 88

NO_XMLINDEX_REWRITE_IN_SELECT 88
CHECK_ACL_REWRITE 88
NO_CHECK_ACL_REWRITE 88
INLINE_XMLTYPE_NT 89
XMLINDEX_SEL_IDX_TBL 89
Distributed Query Hints 89
DRIVING_SITE 89
REMOTE_MAPPED 89
OPAQUE_TRANSFORM 89
Parallel Query Hints 90
STATEMENT_QUEUING 90
NO_STATEMENT_QUEUING 90
GBY_PUSHDOWN 91
NO_GBY_PUSHDOWN 91
HWM_BROKERED 91
NO_QKN_BUFF 93
PARALLEL_INDEX 93
NO_PARALLEL_INDEX 94
PQ_DISTRIBUTE 94
PARALLEL 95
NO_PARALLEL 95
SHARED 95
NOPARALLEL 97
PQ_MAP 97
PQ_NOMAP 97
PRESERVE_OID 97
SYS_PARALLEL_TXN 97
CUBE_GB 97
GBY_CONC_ROLLUP 97
PIV_GB 97
TIV_GB 98
TIV_SSF 98
PIV_SSF 98
RESTORE_AS_INTERVALS 98
SAVE_AS_INTERVALS 98
SCN_ASCENDING 98
Model Query Hints 98
MODEL_MIN_ANALYSIS 98
MODEL_NO_ANALYSIS 99

MODEL_PUSH_REF 99
NO_MODEL_PUSH_REF 100
MODEL_COMPILE_SUBQUERY 100
MODEL_DONTVERIFY_UNIQUENESS 100
MODEL_DYNAMIC_SUBQUERY 100
Partitioning Hints 100
X_DYN_PRUNE 100
SUBQUERY_PRUNING 101
NO_SUBQUERY_PRUNING 101
Other Hints 102
RELATIONAL 102
MONITOR 103
NO_MONITOR 103
NESTED_TABLE_FAST_INSERT 103
NESTED_TABLE_GET_REFS 104
NESTED_TABLE_SET_SETID 104
NO_MONITORING 104
NO_SQL_TUNE 105
RESTRICT_ALL_REF_CONS 106
USE_HASH_AGGREGATION 107
NO_USE_HASH_AGGREGATION 107
BYPASS_RECURSIVE_CHECK 107
BYPASS_UJVC 108
DOMAIN_INDEX_FILTER 108
NO_DOMAIN_INDEX_FILTER 108
DOMAIN_INDEX_SORT 109
NO_DOMAIN_INDEX_ SORT 109
DST_UPGRADE_INSERT_CONV 110
NO_DST_UPGRADE_INSERT_CONV 110
STREAMS 110
DEREF_NO_REWRITE 110
MV_MERGE 110
EXPR_CORR_CHECK 110
INCLUDE_VERSION 110
VECTOR_READ 111
VECTOR_READ_TRACE 111
USE_WEAK_NAME_RESL 111
NO_PARTIAL_COMMIT 111
REF_CASCADE_CURSOR 111

NO_REF_CASCADE....111
SQLLDR....111
SYS_RID_ORDER....112
OVERFLOW_NOMOVE....112
LOCAL_INDEXES....112
MERGE_CONST_ON....112
QUEUE_CURR....112
CACHE_CB....112
QUEUE_ROWP....112
BUFFER....112
NO_BUFFER....112
BITMAP....112

SQL HINT description and demonstration

SQL Hint is one of most important approaches to change the activity of optimizer and SQL execution, it's also pretty important for SQL tuning. For instance, HINT is a part of the SQL Profiler advised by SQL Tuning Advisor. In each Oracle version, corresponding to the SQL features changes, new hints will be introduced, and old hints may be obsolete. Oracle introduced a new dynamic view, V$SQL_HINT, to show in which version the hint was involved in, and in which version it began work as outline data. The hints are associated with special SQL features. It will work only if the related features are enabled. Take HASH_AJ for example, it's a CBO (QKSFM_CBO) feature hint, and it will not work if the SQL optimizer mode is set to RBO.

Some hints are only effect in the internal recursive SQLs, cannot be used in user SQL directly.

The embedded hints in SQL are a piece of comment, with the format /*+ <hint 1> [<hint 2> ...]*/. One comment may involve multiple hints, and one SQL may also involve multiple hint comments. And the hint will work only if they exist in the comment following the key words, SELECT, UPDATE, INSERT, MERGE and DELETE. If the SQL is a complex query involved in sub-query, the hint could be written as global or local format. The local format hints exist in the sub-query, and it can only affect the sub-query. While the global format hits exist in the main part of the query, it can be specified to affect any object in whole query by adding <object>@<block>. The alias could be used to replace the object name.

Tip: Since the embedded SQL hint is a piece of comment, its format could also be --+<hint>.

For example,

```
HELLODBA.COM>select --+full(u)

  2   * from t_users u where user_id =1;
```

We will descript all of hints, and also give demonstration of their usage.

SQL Features

Below hierarchy diagram shows all SQL features and their dependencies. Be aware, some features may be based on multiple features.

```
+QKSFM_ALL                      A Universal Feature
+--QKSFM_COMPILATION                SQL COMPILATION
+----QKSFM_CBO                    SQL Cost Based Optimization
+------QKSFM_ACCESS_PATH              Query access path
+--------QKSFM_AND_EQUAL              Index and-equal access path
+--------QKSFM_BITMAP_TREE            Bitmap tree access path
+--------QKSFM_FULL                 Full table scan
+--------QKSFM_INDEX                Index
+--------QKSFM_INDEX_ASC              Index (ascending)
+--------QKSFM_INDEX_COMBINE            Combine index for bitmap access
+--------QKSFM_INDEX_DESC             Use index (descending)
+--------QKSFM_INDEX_FFS              Index fast full scan
```

+--------QKSFM_INDEX_JOIN Index join
+--------QKSFM_INDEX_RS_ASC Index range scan
+--------QKSFM_INDEX_RS_DESC Index range scan descending
+--------QKSFM_INDEX_SS Index skip scan
+--------QKSFM_INDEX_SS_ASC Index skip scan ascending
+--------QKSFM_INDEX_SS_DESC Index skip scan descending
+--------QKSFM_SORT_ELIM Sort Elimination Via Index
+------QKSFM_CBQT Cost Based Query Transformation
+--------QKSFM_CVM Complex View Merging
+--------QKSFM_DIST_PLCMT Distinct Placement
+--------QKSFM_JOINFAC Join Factorization
+--------QKSFM_JPPD Join Predicate Push Down
+--------QKSFM_PLACE_GROUP_BY Group-By Placement
+--------QKSFM_PULL_PRED pull predicates
+--------QKSFM_TABLE_EXPANSION Table Expansion
+--------QKSFM_UNNEST unnest query block
+------QKSFM_CURSOR_SHARING Cursor sharing
+------QKSFM_DML DML
+------QKSFM_JOIN_METHOD Join methods
+--------QKSFM_USE_HASH Hash join
+--------QKSFM_USE_MERGE Sort-merge join
+--------QKSFM_USE_MERGE_CARTESIAN Merge join cartesian
+--------QKSFM_USE_NL Nested-loop join
+--------QKSFM_USE_NL_WITH_INDEX Nested-loop index join
+------QKSFM_JOIN_ORDER Join order
+------QKSFM_OPT_MODE Optimizer mode
+--------QKSFM_ALL_ROWS All rows (optimizer mode)
+--------QKSFM_CHOOSE Choose (optimizer mode)
+--------QKSFM_FIRST_ROWS First rows (optimizer mode)
+------QKSFM_OR_EXPAND OR expansion QKSFM_JPPD(Join Predicate Push Down)
+------QKSFM_OUTLINE Outlines
+------QKSFM_PARTITION Partition
+------QKSFM_PQ Parallel Query
+--------QKSFM_PARALLEL Parallel table
+--------QKSFM_PQ_DISTRIBUTE PQ Distribution method
+--------QKSFM_PQ_MAP PQ slave mapper
+--------QKSFM_PX_JOIN_FILTER Bloom filtering for joins
+------QKSFM_STAR_TRANS Star Transformation
+------QKSFM_STATS Optimizer statistics
+--------QKSFM_CARDINALITY Cardinality computation
+--------QKSFM_COLUMN_STATS Basic column statistics
+--------QKSFM_CPU_COSTING CPU costing
+--------QKSFM_DBMS_STATS Statistics gathered by DBMS_STATS
+--------QKSFM_DYNAMIC_SAMPLING Dynamic sampling

+--------QKSFM_DYNAMIC_SAMPLING_EST_CDN Estimate CDN using dynamic sampling
+--------QKSFM_GATHER_PLAN_STATISTICS Gather plan statistics
+--------QKSFM_INDEX_STATS Basic index statistics
+--------QKSFM_OPT_ESTIMATE Optimizer estimates
+--------QKSFM_TABLE_STATS Basic table statistics
+----QKSFM_QUERY_REWRITE query rewrite with materialized views
+----QKSFM_RBO SQL Rule Based Optimization
+----QKSFM_SQL_CODE_GENERATOR SQL Code Generator
+----QKSFM_SQL_PLAN_MANAGEMENT SQL Plan Management
+----QKSFM_TRANSFORMATION Query Transformation
+------QKSFM_CBQT Cost Based Query Transformation
+--------QKSFM_CVM Complex View Merging
+--------QKSFM_DIST_PLCMT Distinct Placement
+--------QKSFM_JOINFAC Join Factorization
+--------QKSFM_JPPD Join Predicate Push Down
+--------QKSFM_PLACE_GROUP_BY Group-By Placement
+--------QKSFM_PULL_PRED pull predicates
+--------QKSFM_TABLE_EXPANSION Table Expansion
+--------QKSFM_UNNEST unnest query block
+------QKSFM_HEURISTIC Heuristic Query Transformation
+--------QKSFM_CNT Count(col) to count(*)
+--------QKSFM_COALESCE_SQ coalesce subqueries
+--------QKSFM_CSE Common Sub-Expression Elimination
+--------QKSFM_CVM Complex View Merging
+--------QKSFM_FILTER_PUSH_PRED Push filter predicates
+--------QKSFM_JPPD Join Predicate Push Down
+--------QKSFM_OBYE Order-by Elimination
+--------QKSFM_OLD_PUSH_PRED Old push predicate algorithm (pre-10.1.0.3)
+--------QKSFM_OUTER_JOIN_TO_INNER Join Conversion
+--------QKSFM_PRED_MOVE_AROUND Predicate move around
+--------QKSFM_SET_TO_JOIN Transform set operations to joins
+--------QKSFM_SVM Simple View Merging
+--------QKSFM_TABLE_ELIM Table Elimination
+--------QKSFM_UNNEST unnest query block
+--------QKSFM_USE_CONCAT Or-optimization
+----QKSFM_XML_REWRITE XML Rewrite
+------QKSFM_CHECK_ACL_REWRITE Check ACL Rewrite
+------QKSFM_COST_XML_QUERY_REWRITE Cost Based XML Query Rewrite
+------QKSFM_XMLINDEX_REWRITE XMLIndex Rewrite
+--QKSFM_EXECUTION SQL EXECUTION

Then, we will group and introduce them according their feature.

Access Path Hints

CLUSTER

Usage: CLUSTER([<@Block>] <Table>)

Description: Instructs the optimizer to use a cluster scan to access the specified cluster table

```
HELLODBA.COM>exec sql_explain('SELECT /*+cluster(T)*/* FROM T_EEE T where A >:1', 'BASIC OUTLINE');


-------------------------------------------
| Id  | Operation            | Name        |
-------------------------------------------
|   0 | SELECT STATEMENT     |             |
|   1 |  TABLE ACCESS CLUSTER| T_EEE       |
|   2 |   INDEX RANGE SCAN   | C_KEY2_IDX1 |
-------------------------------------------
```

HASH

Usage: HASH([<@Block>] <Table>)

Description: Instructs the optimizer to use a hash scan to access the specified hash cluster table

```
HELLODBA.COM>exec sql_explain('SELECT /*+ hash(a) full(d) */* FROM T_AAA A, T_DDD D WHERE a.c=d.c',
'BASIC OUTLINE');


-----------------------------------
| Id  | Operation          | Name  |
-----------------------------------
|   0 | SELECT STATEMENT   |       |
|   1 |  NESTED LOOPS      |       |
|   2 |   TABLE ACCESS FULL| T_DDD |
|   3 |   TABLE ACCESS HASH| T_AAA |
-----------------------------------
```

ROWID

Usage: ROWID([<@Block>] <Table>)

Description: Instructs the optimizer to use rowid location to access the specified table directly

```
HELLODBA.COM>exec sql_explain('select /*+rowid(o)*/* from t_objects o where rowid <= :1 and
object_id=100', 'BASIC OUTLINE');


-------------------------------------------------
| Id  | Operation                  | Name       |
-------------------------------------------------
|   0 | SELECT STATEMENT           |            |
|   1 |  TABLE ACCESS BY ROWID RANGE| T_OBJECTS |
-------------------------------------------------
```

FULL

Usage: FULL([<@Block>] <Table>)

Description: Instructs the optimizer to use a table scan to access the specified table

```
HELLODBA.COM>exec sql_explain('select /*+full(o)*/* from t_objects o where rowid = :1 and
```

```
object_id>100','BASIC OUTLINE');


---------------------------------------
| Id  | Operation         | Name      |
---------------------------------------
|   0 | SELECT STATEMENT  |           |
|   1 |  TABLE ACCESS FULL| T_OBJECTS |
---------------------------------------
```

INDEX

Usage: INDEX([<@Block>] <Table> [<Index>]) or INDEX([<@Block>] <Table> [(<Indexed Columns>)])

Description: Instructs the optimizer to use an index scan to access the specified table

```
HELLODBA.COM>exec sql_explain('select /*+index(o (object_id))*/* from t_objects o where rowid = :1 and
object_id>100','BASIC OUTLINE');


-------------------------------------------------
| Id  | Operation                   | Name        |
-------------------------------------------------
|   0 | SELECT STATEMENT            |             |
|   1 |  TABLE ACCESS BY INDEX ROWID| T_OBJECTS   |
|   2 |   INDEX RANGE SCAN          | T_OBJECTS_PK |
-------------------------------------------------
```

INDEX_ASC

Usage: INDEX_ASC([<@Block>] <Table> [<Index>]) or INDEX_ASC([<@Block>] <Table> [(<Indexed Columns>)])

Description: Instructs the optimizer to use an index scan to access the specified table. If it uses an index range scan, then it scans the index in ascending order of the indexed values.

Demo: (Please not the output sequence)

```
HELLODBA.COM>select /*+index_asc(o (object_id))*/object_id, object_name from t_objects o where
object_id<5;


 OBJECT_ID OBJECT_NAME
---------- --------------------------------------------------------------------------------
         2 C_OBJ#
         3 I_OBJ#
         4 TAB$
```

INDEX_DESC

Usage: INDEX_DESC([<@Block>] <Table> [<Index>]) or INDEX_DESC([<@Block>] <Table> [(<Indexed Columns>)])

Description: Instructs the optimizer to use an index scan to access the specified table. If it uses an index range scan, then it scans the index in descending order of the indexed values.

Demo: (Please not the output sequence)

```
HELLODBA.COM>select /*+index_desc(o (object_id))*/object_id, object_name from t_objects o where
object_id<5;
```

```
 OBJECT_ID OBJECT_NAME
---------- --------------------------------------------------------------------------------
         4 TAB$
         3 I_OBJ#
         2 C_OBJ#
```

NO_INDEX

Usage: NO_INDEX([<@Block>] <Table> [<Index>]) or NO_INDEX([<@Block>] <Table> [(<Indexed Columns>)])

Description: Prevents the optimizer to use an index scan to access the specified table.

```
HELLODBA.COM>exec sql_explain('select /*+no_index(o t_objects_pk)*/object_id, object_name from
t_objects o where object_id < 10', 'BASIC OUTLINE');


----------------------------------------------
| Id  | Operation            | Name           |
----------------------------------------------
|   0 | SELECT STATEMENT     |                |
|   1 |  INDEX FAST FULL SCAN| T_OBJECTS_IDX8 |
----------------------------------------------
```

INDEX_FFS

Usage: INDEX_FFS([<@Block>] <Table> [<Index>]) or INDEX_FFS([<@Block>] <Table> [(<Indexed Columns>)])

Description: Instructs the optimizer to use a fast full index scan to access the specified table.

```
HELLODBA.COM>exec sql_explain('SELECT /*+INDEX_FFS(t t_objects_IDX8)*/object_name FROM t_objects t
where owner=:A', 'BASIC OUTLINE');


----------------------------------------------
| Id  | Operation            | Name           |
----------------------------------------------
|   0 | SELECT STATEMENT     |                |
|   1 |  INDEX FAST FULL SCAN| T_OBJECTS_IDX8 |
----------------------------------------------
```

NO_INDEX_FFS

Usage: NO_INDEX_FFS([<@Block>] <Table> [<Index>]) or NO_INDEX_FFS([<@Block>] <Table> [(<Indexed Columns>)])

Description: Prevents the optimizer to use a fast full index scan to access the specified table.

```
HELLODBA.COM>exec sql_explain('select /*+no_index_ffs(o t_objects_idx8)*/owner, object_name from
t_objects o', 'BASIC OUTLINE');


---------------------------------------
| Id  | Operation         | Name      |
---------------------------------------
|   0 | SELECT STATEMENT  |           |
|   1 |  TABLE ACCESS FULL| T_OBJECTS |
```

```
---------------------------------------
```

INDEX_RRS

Usage: INDEX_RRS([<@Block>] <Table> [<Index>]) or INDEX_RRS([<@Block>] <Table> [(<Indexed Columns>)])

Description: This hint normally works in the internal statement when performing parallel fast full index scan query.

Demo: (Only works in 9i)

```
HELLODBA.COM>create table t (A number, B varchar2(20), constraint t_pk primary key(A) ) organization index parallel;


Table created.
HELLODBA.COM>select count(*) from t;


Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=1 Card=1)
   1    0   SORT (AGGREGATE)
   2    1     SORT* (AGGREGATE)                                                    :Q17628000
   3    2       INDEX* (FAST FULL SCAN) OF 'T_PK' (UNIQUE) (Cost=1 Card=1000000) :Q17628000
   2 PARALLEL_TO_SERIAL            SELECT /*+ PIV_SSF */ SYS_OP_MSR(COUNT(*)) FROM (SELECT /*+
INDEX_RRS(A2 "T_PK") */ 0 FROM "T"  PX_GRANULE(0, BLOCK_RANGE, DYNAMIC)A2) A1
   3 PARALLEL_COMBINED_WITH_PARENT
```

INDEX_SS

Usage: INDEX_SS([<@Block>] <Table> [<Index>]) or INDEX_SS([<@Block>] <Table> [(<Indexed Columns>)])

Description: Instructs the optimizer to use an index skip scan to access the specified table.

```
HELLODBA.COM>exec sql_explain('select /*+index_ss(t t_tables_pk)*/count(status) from t_tables t where table_name like :A', 'BASIC OUTLINE');


------------------------------------------------
| Id  | Operation                    | Name        |
------------------------------------------------
|   0 | SELECT STATEMENT             |             |
|   1 |  SORT AGGREGATE              |             |
|   2 |   TABLE ACCESS BY INDEX ROWID| T_TABLES    |
|   3 |    INDEX SKIP SCAN           | T_TABLES_PK |
------------------------------------------------
```

INDEX_SS_ASC

Usage: INDEX_SS_ASC([<@Block>] <Table> [<Index>]) or INDEX_SS_ASC([<@Block>] <Table> [(<Indexed Columns>)])

Description: Instructs the optimizer to use an index skip scan to access the specified table. And it scans the index in ascending order of the indexed values.

Demo: (Please not the output sequence)

```
HELLODBA.COM>select /*+index_ss_asc(t t_tables_pk)*/table_name, status from t_tables t where table_name
```

```
like 'T%' and rownum<=3;

TABLE_NAME                     STATUS
------------------------------ --------
TAB$                           VALID
TABCOMPART$                    VALID
TABLE_PRIVILEGE_MAP            VALID
```

INDEX_SS_DESC

Usage: IINDEX_SS_DESC([<@Block>] <Table> [<Index>]) or INDEX_SS_DESC([<@Block>] <Table> [(<Indexed Columns>)])

Description: Instructs the optimizer to use an index skip scan to access the specified table. And it scans the index in descending order of the indexed values.

Demo: (Please not the output sequence)

```
HELLODBA.COM>select /*+index_ss_desc(t t_tables_pk)*/table_name, status from t_tables t where
table_name like 'T%' and rownum<=3;

TABLE_NAME                     STATUS
------------------------------ --------
TYPE_MISC$                     VALID
TYPEHIERARCHY$                 VALID
TYPED_VIEW$                    VALID
```

NO_INDEX_SS

Usage: NO_INDEX_SS([<@Block>] <Table> [<Index>]) or NO_INDEX_SS([<@Block>] <Table> [(<Indexed Columns>)])

Description: Prevents the optimizer to use an index skip scan to access the specified table.

```
HELLODBA.COM>exec sql_explain('select /*+no_index_ss(o t_objects_idx8)*/owner, object_name from
t_objects o where object_name like :A', 'BASIC OUTLINE');

---------------------------------------------
| Id  | Operation            | Name           |
---------------------------------------------
|   0 | SELECT STATEMENT     |                |
|   1 |  INDEX FAST FULL SCAN| T_OBJECTS_IDX8 |
---------------------------------------------
```

INDEX_RS_ASC

Usage: INDEX_RS_ASC([<@Block>] <Table> [<Index>]) or INDEX_RS_ASC([<@Block>] <Table> [(<Indexed Columns>)])

Description: Instructs the optimizer to use an index range scan to access the specified table. And it scans the index in ascending order of the indexed values. If the selected index is a unique index, and predication fulfills unique scan requirements, it will use index unique scan.

Demo: (Please not the output sequence)

```
HELLODBA.COM>select /*+index_rs_asc(o t_objects_pk)*/object_id, object_name from t_objects o where
object_id < 5;
```

```
 OBJECT_ID OBJECT_NAME
----------
--------------------------------------------------------------------------------

         2 C_OBJ#
         3 I_OBJ#
         4 TAB$
```

INDEX_RS_DESC

Usage: INDEX_RS_DESC([<@Block>] <Table> [<Index>]) or INDEX_RS_DESC([<@Block>] <Table> [(<Indexed Columns>)])

Description: Instructs the optimizer to use an index range scan to access the specified table. And it scans the index in descending order of the indexed values. If the selected index is a unique index, and predication fulfills unique scan requirements, it will use index unique scan.

Demo: (Please not the output sequence)

```
HELLODBA.COM>select /*+index_rs_desc(o t_objects_pk)*/object_id, object_name from t_objects o where
object_id < 5;


 OBJECT_ID OBJECT_NAME
----------
--------------------------------------------------------------------------------

         4 TAB$
         3 I_OBJ#
         2 C_OBJ#
```

AND_EQUAL

Usage: AND_EQUAL([<@Block>] <Table> [<Index 1> <Index 2> ...]) or AND_EQUAL([<@Block>] <Table> [(<Index 1 columns>) (<Index 2 columns>) ...])

Description: Instructs the optimizer to get intersection of ROWID sets from 2 or more single-column indexes. The duplicated ROWID will be eliminated.

```
HELLODBA.COM>exec sql_explain('select /*+AND_EQUAL(t T_TABLES_IDX1 T_TABLES_IDX3)*/* from t_tables t
where t.owner=:A and t.tablespace_name=:B', 'BASIC OUTLINE');


-----------------------------------------------------
| Id  | Operation                    | Name          |
-----------------------------------------------------
|   0 | SELECT STATEMENT             |               |
|   1 |  TABLE ACCESS BY INDEX ROWID| T_TABLES      |
|   2 |   AND-EQUAL                  |               |
|   3 |    INDEX RANGE SCAN          | T_TABLES_IDX1 |
|   4 |    INDEX RANGE SCAN          | T_TABLES_IDX3 |
-----------------------------------------------------
```

INDEX_COMBINE

Usage: INDEX_COMBINE([<@Block>] <Table> [<Index 1> ...]) or AND_EQUAL([<@Block>] <Table> [(<Index 1 columns>) ...])

Description: Instructs the optimizer to combine the bitmap to access the specified table. If the specified index is not bitmap index, it will try to convert the ROWIDs to bitmap first.

```
HELLODBA.COM>exec sql_explain('SELECT /*+INDEX_COMBINE(t t_objects_idx2 t_objects_idx8)*/1 FROM
t_objects t where status=:A and owner=:B', 'BASIC OUTLINE');


-----------------------------------------------------
| Id  | Operation                     | Name          |
-----------------------------------------------------
|   0 | SELECT STATEMENT              |               |
|   1 |  BITMAP CONVERSION TO ROWIDS  |               |
|   2 |   BITMAP AND                  |               |
|   3 |    BITMAP INDEX SINGLE VALUE  | T_OBJECTS_IDX2 |
|   4 |    BITMAP CONVERSION FROM ROWIDS|             |
|   5 |     SORT ORDER BY             |               |
|   6 |      INDEX RANGE SCAN         | T_OBJECTS_IDX8 |
-----------------------------------------------------
```

INDEX_JOIN

Usage: INDEX_JOIN([<@Block>] <Table> [<Index1> <Index2> ...]) or INDEX_JOIN([<@Block>] <Table> [(<Index1 columns>) (<Index2 columns>) ...])

Description: Instructs the optimizer to join indexes.

```
HELLODBA.COM>exec sql_explain('SELECT /*+INDEX_JOIN(t t_objects_idx2 t_objects_idx8)*/1 FROM t_objects
t where status=:A and owner=:B', 'BASIC OUTLINE');


-----------------------------------------------------
| Id  | Operation                     | Name          |
-----------------------------------------------------
|   0 | SELECT STATEMENT              |               |
|   1 |  VIEW                         | index$_join$_001 |
|   2 |   HASH JOIN                   |               |
|   3 |    BITMAP CONVERSION TO ROWIDS|               |
|   4 |     BITMAP INDEX SINGLE VALUE | T_OBJECTS_IDX2 |
|   5 |    INDEX RANGE SCAN           | T_OBJECTS_IDX8 |
-----------------------------------------------------
```

BITMAP_TREE

Usage: BITMAP_TREE([<@Block>] <Table> AND(<Index1>[<Index2> ...])) or BITMAP_TREE([<@Block>] <Table> AND((<Index1 columns>)[(<Index2 columns>) ...]))

Description: Instructs the optimizer to convert ROWIDs to bitmap, then performance bitmap operations.

```
HELLODBA.COM>exec sql_explain('select /*+BITMAP_TREE(T_OBJECTS AND(T_OBJECTS_IDX4 T_OBJECTS_IDX2))*/
owner from t_objects where owner like :A and status like :B', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |               |  1190 | 15470 |   559   (1)| 00:00:03 |
|   1 |  TABLE ACCESS BY INDEX ROWID | T_OBJECTS      |  1190 | 15470 |   559   (1)| 00:00:03 |
```

```
|   2 |   BITMAP CONVERSION TO ROWIDS|                |       |       |            |          |
|   3 |    BITMAP AND                |                |       |       |            |          |
|   4 |     BITMAP MERGE             |                |       |       |            |          |
|*  5 |      BITMAP INDEX RANGE SCAN | T_OBJECTS_IDX4 |       |       |            |          |
|   6 |     BITMAP MERGE             |                |       |       |            |          |
|*  7 |      BITMAP INDEX RANGE SCAN | T_OBJECTS_IDX2 |       |       |            |          |
-----------------------------------------------------------------------------------------------
```

USE_INVISIBLE_INDEXES

Usage: USE_INVISIBLE_INDEXES([<@Block>] <Table> ([<Index1>] ...)) or USE_INVISIBLE_INDEXES([<@Block>] <Table> ([<Index1 Columns>] ...))

Description: Instructs the optimizer to use the invisible indexes.

```
HELLODBA.COM>show parameter visible

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
optimizer_use_invisible_indexes      boolean     FALSE
HELLODBA.COM>alter index t_objects_pk invisible;

Index altered.

HELLODBA.COM>exec sql_explain('select /*+qb_name(M) USE_INVISIBLE_INDEXES(o (object_id))*/count(1)
from t_objects o where object_id < 1000', 'TYPICAL OUTLINE');

-------------------------------------------------------------------------------------
| Id  | Operation         | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |               |     1 |     5 |     3   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE   |               |     1 |     5 |            |          |
|*  2 |   INDEX RANGE SCAN| T_OBJECTS_PK  |   973 |  4865 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------
```

NO_USE_INVISIBLE_INDEXES

Usage: NO_USE_INVISIBLE_INDEXES([<@Block>] <Table> ([<Index1>] ...)) or NO_USE_INVISIBLE_INDEXES([<@Block>] <Table>([<Index1 Columns>] ...))

Description: Prevents the optimizer to use the invisible indexes.

```
HELLODBA.COM>exec sql_explain('select /*+qb_name(M) NO_USE_INVISIBLE_INDEXES(o
(t_objects_pk))*/count(1) from t_objects o where object_id < 1000', 'TYPICAL OUTLINE');

-------------------------------------------------------------------------------------
| Id  | Operation             | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                |     1 |     5 |   185   (2)| 00:00:02 |
|   1 |  SORT AGGREGATE       |                |     1 |     5 |            |          |
|*  2 |   INDEX FAST FULL SCAN| T_OBJECTS_IDX8 |   973 |  4865 |   185   (2)| 00:00:02 |
```

```
-------------------------------------------------------------------------------
```

Join Hints

NL_AJ

Usage: NL_AJ([<@SubBlock>])

Description: Instructs the optimizer to perform Nested Loop Anti-join with the table in sub-query.

```
HELLODBA.COM>exec sql_explain('select /*+nl_aj(@inv)*/* from t_tables t where not exists (select
/*+qb_name(inv)*/1 from t_users u where t.owner=u.username)', 'BASIC OUTLINE')


-----------------------------------------
| Id  | Operation          | Name       |
-----------------------------------------
|   0 | SELECT STATEMENT   |            |
|   1 |  NESTED LOOPS ANTI |            |
|   2 |   TABLE ACCESS FULL| T_TABLES   |
|   3 |   INDEX UNIQUE SCAN| T_USERS_UK |
-----------------------------------------
```

HASH_AJ

Usage: HASH_AJ([<@SubBlock>])

Description: Instructs the optimizer to perform Hash Anti-join with the table in sub-query.

```
HELLODBA.COM>exec sql_explain('select /*+leading(t) hash_aj(@inv)*/* from t_tables t where not exists
(select /*+qb_name(inv)*/1 from t_users u where t.owner=u.username)', 'BASIC OUTLINE')


-----------------------------------------
| Id  | Operation          | Name       |
-----------------------------------------
|   0 | SELECT STATEMENT   |            |
|   1 |  HASH JOIN ANTI    |            |
|   2 |   TABLE ACCESS FULL| T_TABLES   |
|   3 |   INDEX FULL SCAN  | T_USERS_UK |
-----------------------------------------
```

MERGE_AJ

Usage: MERGE_AJ([<@SubBlock>])

Description: Instructs the optimizer to perform Merge Anti-join with the table in sub-query.

```
HELLODBA.COM>exec sql_explain('select /*+merge_aj(@inv)*/* from t_tables t where not exists (select
/*+qb_name(inv)*/1from t_users u where t.owner=u.username)', 'BASIC OUTLINE')


-------------------------------------------------------
| Id  | Operation                    | Name          |
-------------------------------------------------------
|   0 | SELECT STATEMENT             |               |
|   1 |  MERGE JOIN ANTI             |               |
|   2 |   TABLE ACCESS BY INDEX ROWID| T_TABLES      |
|   3 |    INDEX FULL SCAN           | T_TABLES_IDX1 |
```

```
|   4 |   SORT UNIQUE                |               |
|   5 |    INDEX FULL SCAN           | T_USERS_UK    |
-------------------------------------------------
```

USE_HASH

Usage: USE_HASH([<@Block>] <Table1> [<Table2>])

Description: instructs the optimizer to use hash- join to join each specified table. Combine with LEADING hint to indicate the join order.

```
HELLODBA.COM>exec sql_explain('select /*+ use_hash(o) leading(t) */* from t_objects o, t_tables t where
o.owner=t.owner and o.object_name=t.table_name and o.object_id = :A', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |               |     1 |   326 |   144   (1)| 00:00:01 |
|*  1 |  HASH JOIN                   |               |     1 |   326 |   144   (1)| 00:00:01 |
|   2 |   TABLE ACCESS FULL          | T_TABLES      |  2071 |   412K|   142   (1)| 00:00:01 |
|   3 |   TABLE ACCESS BY INDEX ROWID| T_OBJECTS     |     1 |   122 |     2   (0)| 00:00:01 |
|*  4 |    INDEX UNIQUE SCAN         | T_OBJECTS_PK  |     1 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

NO_USE_HASH

Usage: NO_USE_HASH([<@Block>] <Table1> [<Table2> ...])

Description: instructs the optimizer not use the specified tables as the inner table in hash join. If all tables to be joined are specified in the hint, the optimizer will not consider hash join.

```
HELLODBA.COM>exec sql_explain('select /*+ qb_name(M) no_use_hash(t) */* from t_objects o, t_tables t
where o.owner=t.owner and o.object_name=t.table_name', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           | 47585 |    14M|  1813   (1)| 00:00:08 |
|*  1 |  HASH JOIN         |           | 47585 |    14M|  1813   (1)| 00:00:08 |
|   2 |   TABLE ACCESS FULL| T_TABLES  |  2071 |   412K|   142   (1)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| T_OBJECTS | 47585 |  5669K|  1670   (1)| 00:00:07 |
-------------------------------------------------------------------------------
```

USE_MERGE

Usage: USE_MERGE([<@Block>] <Table1> [<Table2>])

Description: instructs the optimizer to use Merge- join to join each specified table. Combine with LEADING hint to indicate the join order.

```
HELLODBA.COM>exec sql_explain('select /*+ qb_name(M) use_merge(t o) */* from t_objects o, t_tables t
where o.owner=t.owner and o.object_name=t.table_name', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
```

```
|   0 | SELECT STATEMENT    |           | 47585 |    14M|       |  6360   (1)| 00:00:26 |
|   1 |  MERGE JOIN         |           | 47585 |    14M|       |  6360   (1)| 00:00:26 |
|   2 |   SORT JOIN         |           |  2071 |   412K|  1288K|   447   (1)| 00:00:02 |
|   3 |    TABLE ACCESS FULL| T_TABLES  |  2071 |   412K|       |   142   (1)| 00:00:01 |
|*  4 |   SORT JOIN         |           | 47585 |  5669K|    14M|  5913   (1)| 00:00:24 |
|   5 |    TABLE ACCESS FULL| T_OBJECTS | 47585 |  5669K|       |  1670   (1)| 00:00:07 |
--------------------------------------------------------------------------------------
```

NO_USE_MERGE

Usage: NO_USE_MERGE([<@Block>] <Table1> [<Table2> ...])

Description: instructs the optimizer not use the specified tables as the inner table in merge join. If all tables to be joined are specified in the hint, the optimizer will not consider merge join.

```
HELLODBA.COM>exec sql_explain('select /*+ qb_name(M) no_use_merge(t o) */o.object_id, t.table_name from t_objects o, t_tables t where o.owner=t.owner and o.object_name=t.table_name and o.object_id < :A', 'TYPICAL OUTLINE');


-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |               |  2379 |   137K|    11  (10)| 00:00:01 |
|   1 |  NESTED LOOPS                 |               |  2379 |   137K|    11  (10)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID| T_OBJECTS     |  2379 | 83265 |    10   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN           | T_OBJECTS_PK |   428 |       |     2   (0)| 00:00:01 |
|*  4 |   INDEX UNIQUE SCAN           | T_TABLES_PK   |     1 |    24 |     0   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

USE_NL

Usage: USE_NL([<@Block>] <Table1> [<Table2>])

Description: instructs the optimizer to use Nested-Loop-join to join each specified table. Combine with LEADING hint to indicate the join order.

```
HELLODBA.COM>exec sql_explain('select /*+ qb_name(M) use_nl(t o) */* from t_objects o, t_tables t where o.owner=t.ownerand o.object_name=t.table_name', 'TYPICAL OUTLINE');


-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |               | 47585 |    14M|  4830   (1)| 00:00:20 |
|   1 |  NESTED LOOPS                 |               | 47585 |    14M|  4830   (1)| 00:00:20 |
|   2 |   TABLE ACCESS FULL           | T_OBJECTS     | 47585 |  5669K|  1670   (1)| 00:00:07 |
|   3 |   TABLE ACCESS BY INDEX ROWID| T_TABLES      |     1 |   204 |     1   (0)| 00:00:01 |
|*  4 |    INDEX UNIQUE SCAN          | T_TABLES_PK   |     1 |       |     0   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

USE_NL_WITH_INDEX

Usage: USE_NL_WITH_INDEX([<@Block>] <Table> [索引 1 ...]) or USE_NL_WITH_INDEX([<@Block>] <Table> [(<Index 字段列表 1>) ...])

Description: instructs the optimizer to use the specified table as the inner table in Nested-Loop-join, and use the specified index to access to the table.

```
HELLODBA.COM>exec sql_explain('select /*+ qb_name(M) use_nl_with_index(o (status owner object_name))
leading(t) */o.object_name, t.* from t_objects o, t_tables t where o.owner=t.owner and
o.object_name=t.table_name', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------
| Id  | Operation          | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |                | 47585 |    10M|   666K  (1)| 00:44:28 |
|   1 |  NESTED LOOPS      |                | 47585 |    10M|   666K  (1)| 00:44:28 |
|   2 |   TABLE ACCESS FULL| T_TABLES       |  2071 |   412K|   142   (1)| 00:00:01 |
|*  3 |   INDEX FULL SCAN  | T_OBJECTS_IDX1 |    23 |   690 |   322   (1)| 00:00:02 |
-------------------------------------------------------------------------------------
```

NO_USE_NL

Usage: NO_USE_NL([<@Block>] <Table1> [<Table2> ...])

Description: instructs the optimizer not use the specified tables as the inner table in nested-loop join. If all tables to be joined are specified in the hint, the optimizer will not consider nested-loop join.

```
HELLODBA.COM>exec sql_explain('select /*+ qb_name(M) no_use_nl(t o) */o.object_id, t.table_name from
t_objects o, t_tables t where o.owner=t.owner and o.object_name=t.table_name and o.object_id = :A',
'TYPICAL OUTLINE');


-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |               |     1 |    59 |    14   (8)| 00:00:01 |
|   1 |  MERGE JOIN                   |               |     1 |    59 |    14   (8)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID| T_OBJECTS     |     1 |    35 |     2   (0)| 00:00:01 |
|*  3 |    INDEX UNIQUE SCAN          | T_OBJECTS_PK |     1 |       |     1   (0)| 00:00:01 |
|*  4 |   FILTER                      |               |       |       |            |          |
|   5 |    INDEX FULL SCAN            | T_TABLES_PK   |  2071 | 49704 |    11   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

USE_MERGE_CARTESIAN

Usage: USE_MERGE_CARTESIANUSE_SEMI([<@Block>] <Table1> [<Table2>])

Description: instructs the optimizer to use Merge Cartesian Join to join each specified table.

Demo (11.2.0.1):

```
HELLODBA.COM>exec sql_explain('select /*+use_nl(ts) USE_MERGE_CARTESIAN(d) leading(u)*/count(*) from
t_users u, t_datafiles d, t_tablespaces ts where ts.tablespace_name = d.tablespace_name and
u.default_tablespace = ts.tablespace_name', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------
| Id  | Operation             | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                |     1 |    32 |     8  (25)| 00:00:01 |
```

```
|   1 |  SORT AGGREGATE       |                |     1 |    32 |           |          |
|   2 |   MERGE JOIN CARTESIAN|                |    14 |   448 |     8  (25)| 00:00:01 |
|   3 |    NESTED LOOPS       |                |    31 |   465 |     3   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL | T_USERS        |    31 |   217 |     3   (0)| 00:00:01 |
|*  5 |     INDEX UNIQUE SCAN | T_TABLESPACE_PK |     1 |     8 |     0   (0)| 00:00:01 |
|   6 |    BUFFER SORT        |                |     1 |    17 |     8  (25)| 00:00:01 |
|*  7 |     TABLE ACCESS FULL | T_DATAFILES    |     1 |    17 |     3   (0)| 00:00:01 |
---------------------------------------------------------------------------------------
```

LEADING

Usage: LEADING([<@Block>] <Table1> [<Table2> …])

Description: instructs the optimizer to use specified sequence to join tables

Refer to the demo of USE_HASH

USE_ANTI

Usage: USE_ANTII([<@Block>] <Table1> [<Table2>])

Description: instructs the optimizer to use anti-join, can only be observed in the internal statement of parallel anti-join query.

Demo (9.2.0.5):

```
HELLODBA.COM>select /*+ parallel(o 2) */count(*) from t_objects o where owner not in (select /*+
*/username from t_users u);


Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=47 Card=1 Bytes=16)
   1    0   SORT (AGGREGATE)
   2    1     SORT* (AGGREGATE)                                                    :Q17863001
   3    2       HASH JOIN* (ANTI) (Cost=47 Card=1 Bytes=16)                        :Q17863001
   4    3         TABLE ACCESS* (FULL) OF 'T_OBJECTS' (Cost=22 Card=32435 Bytes=227045) :Q17863001
   5    3         TABLE ACCESS* (FULL) OF 'T_USERS' (Cost=2 Card=46 By tes=414)    :Q17863000
   2 PARALLEL_TO_SERIAL            SELECT /*+ PIV_SSF */ SYS_OP_MSR(COUNT(*)) F
                                   ROM (SELECT /*+ ORDERED NO_EXPAND USE_HASH(A
                                   3) USE_ANTI(A3) */ 0 FROM (SELECT /*+ NO_EXP
                                   AND ROWID(A4) */ A4."OWNER" C0 FROM "T_OBJEC
                                   TS" PX_GRANULE(0, BLOCK_RANGE, DYNAMIC)  A4)
                                    A2,:Q17863000 A3 WHERE A2.C0=A3.C0) A1
   3 PARALLEL_COMBINED_WITH_PARENT
   4 PARALLEL_COMBINED_WITH_PARENT
   5 PARALLEL_FROM_SERIAL
```

USE_SEMI

Usage: USE_SEMI([<@Block>] <Table1> [<Table2>])

Description: instructs the optimizer to use semi-join, can only be observed in the internal statement of parallel semi-join query.

Demo (9.2.0.5):

```
HELLODBA.COM>select /*+ parallel(o default) */* from t_objects o where subobject_name in (select
/*+parallel(t default)*/table_name from t_tables t where o.owner=t.owner);
```

```
no rows selected


Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=8 Card=26 Bytes=3120)
   1    0   NESTED LOOPS* (SEMI) (Cost=8 Card=26 Bytes=3120)                  :Q17689000
   2    1     TABLE ACCESS* (FULL) OF 'T_OBJECTS' (Cost=6 Card=51 Bytes=4743) :Q17689000
          3    1      INDEX* (RANGE SCAN) OF 'T_TABLES_UK1' (NON-UNIQUE)    :Q17689000
   1 PARALLEL_TO_SERIAL            SELECT /*+ ORDERED NO_EXPAND USE_NL(A2) INDE
                                   X(A2 "T_TABLES_UK1") USE_SEMI(A2) */ A1.C0,A
                                   1.C1,A1.C2,A1.C3,A1.C4,A1.C5,A1.C6,A1.C7,A1.
                                   C8,A1.C9,A1.C10,A1.C11,A1.C12 FROM (SELECT /
                                   *+ NO_EXPAND ROWID(A3) */ A3."OWNER" C0,A3."
                                   OBJECT_NAME" C1,A3."SUBOBJECT_NAME" C2,A3."O
                                   BJECT_ID" C3,A3."DATA_OBJECT_ID" C4,A3."OBJE
                                   CT_TYPE" C5,A3."CREATED" C6,A3."LAST_DDL_TIM
                                   E" C7,A3."TIMESTAMP" C8,A3."STATUS" C9,A3."T
                                   EMPORARY" C10,A3."GENERATED" C11,A3."SECONDA
                                   RY" C12 FROM "T_OBJECTS" PX_GRANULE(0, BLOCK
                                   _RANGE, DYNAMIC)  A3 WHERE A3."SUBOBJECT_NAM
                                   E" IS NOT NULL) A1,"T_TABLES" A2 WHERE A1.C2
                                   =A2."TABLE_NAME" AND A1.C0=A2."OWNER"
```

NATIVE_FULL_OUTER_JOIN

Usage: NATIVE_FULL_OUTER_JOIN([<@Block>])

Description: instructs the optimizer to use native full out join.

```
HELLODBA.COM>exec sql_explain('select /*+NATIVE_FULL_OUTER_JOIN*/ts.tablespace_name, ts.block_size,
u.user_id from t_tablespaces ts full outer join t_users u on ts.tablespace_name=u.default_tablespace
and ts.max_extents<:A and u.user_id>:B', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------------
| Id  | Operation             | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |               |    54 |  2322 |     5  (20)| 00:00:05 |
|   1 |  VIEW                 | VW_FOJ_0      |    54 |  2322 |     5  (20)| 00:00:05 |
|*  2 |   HASH JOIN FULL OUTER|               |    54 |  1350 |     5  (20)| 00:00:05 |
|   3 |    TABLE ACCESS FULL  | T_TABLESPACES |    15 |   240 |     2   (0)| 00:00:03 |
|   4 |    TABLE ACCESS FULL  | T_USERS       |    41 |   369 |     2   (0)| 00:00:03 |
-------------------------------------------------------------------------------------------
```

NO_NATIVE_FULL_OUTER_JOIN

Usage: NO_NATIVE_FULL_OUTER_JOIN([<@Block>])

Description: Prevents the optimizer to use native full out join.

```
HELLODBA.COM>exec sql_explain('select /*+NO_NATIVE_FULL_OUTER_JOIN*/ts.tablespace_name,
ts.block_size, u.user_id from t_tablespaces ts full outer join t_users u on
```

```
ts.tablespace_name=u.default_tablespace and ts.max_extents<:A and u.user_id>:B', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------------------
-
| Id  | Operation                      | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
-
|   0 | SELECT STATEMENT               |                 |    58 |  2494 |    82   (0)| 00:00:01 |
|   1 |  VIEW                          |                 |    58 |  2494 |    82   (0)| 00:00:01 |
|   2 |   UNION-ALL                    |                 |       |       |            |
|
|   3 |    NESTED LOOPS OUTER          |                 |    15 |   435 |    41   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL          | T_TABLESPACES   |    15 |   240 |    11   (0)| 00:00:01 |
|   5 |     VIEW                       |                 |     1 |    13 |     2   (0)| 00:00:01 |
|*  6 |      FILTER                    |                 |       |       |            |
|
|*  7 |       TABLE ACCESS BY INDEX ROWID| T_USERS         |     1 |    11 |     2   (0)| 00:00:01 |
|*  8 |        INDEX RANGE SCAN        | T_USERS_PK      |     2 |       |     1   (0)| 00:00:01 |
|*  9 |    FILTER                      |                 |       |       |            |
|
|  10 |     TABLE ACCESS FULL          | T_USERS         |    43 |   473 |    19   (0)| 00:00:01 |
|* 11 |     FILTER                     |                 |       |       |            |          |
|* 12 |      TABLE ACCESS BY INDEX ROWID | T_TABLESPACES |     1 |    13 |     1   (0)| 00:00:01 |
|* 13 |       INDEX UNIQUE SCAN        | T_TABLESPACE_PK |     1 |       |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
-
```

NO_CARTESIAN

Usage: NO_CARTESIAN([<@block>] <Table1> [<Table2> ...])

Description: Prevents the optimizer to use Cartesian Join to join each specified table.

```
HELLODBA.COM>exec sql_explain('select /*+ QB_NAME(M) FULL(D) NO_CARTESIAN(D) */t.owner, t.table_name,
i.owner, i.index_name, d.* from t_datafiles d, t_constraints c, t_tables t, t_indexes i where
t.tablespace_name=d.tablespace_name and c.owner=t.owner and c.table_name=t.table_name and c.r_owner =
i.owner and c.r_constraint_name = i.index_name and d.file_id = :id', 'BASIC');


-------------------------------------------------------------
| Id  | Operation                    | Name               |
-------------------------------------------------------------
|   0 | SELECT STATEMENT             |                    |
|   1 |  NESTED LOOPS                |                    |
|   2 |   HASH JOIN                  |                    |
|   3 |    NESTED LOOPS              |                    |
|   4 |     TABLE ACCESS BY INDEX ROWID| T_CONSTRAINTS      |
|   5 |      INDEX FULL SCAN         | T_CONSTRAINTS_IDX4 |
|   6 |     TABLE ACCESS BY INDEX ROWID| T_TABLES           |
```

```
|   7 |      INDEX UNIQUE SCAN          | T_TABLES_PK        |
|   8 |    TABLE ACCESS FULL            | T_DATAFILES        |
|   9 |   INDEX UNIQUE SCAN             | T_INDEXES_PK       |
-----------------------------------------------------------
```

ORDERED

Usage: ORDERED

Description: instructs the optimizer to join tables in the order of their appearance in FROM clause.

```
HELLODBA.COM>exec sql_explain('SELECT /*+QB_NAME(M) ORDERED*/ * from t_objects o, t_users u where
user_id=:A and u.username = o.owner','BASIC');


------------------------------------------------
| Id  | Operation                    | Name       |
------------------------------------------------
|   0 | SELECT STATEMENT             |            |
|   1 |  HASH JOIN                   |            |
|   2 |   TABLE ACCESS FULL          | T_OBJECTS  |
|   3 |   TABLE ACCESS BY INDEX ROWID| T_USERS    |
|   4 |    INDEX UNIQUE SCAN         | T_USERS_PK |
------------------------------------------------
```

PX_JOIN_FILTER

Usage: PX_JOIN_FILTER(<Table>)

Description: Instructs the optimizer to use bitmap filter to perform hash outer join or parallel join.

```
HELLODBA.COM>exec sql_explain('SELECT /*+ qb_name(M) PX_JOIN_FILTER(t) */* FROM t_tables t,
t_datafiles d WHERE t.tablespace_name(+) = d.tablespace_name', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------
| Id  | Operation          | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |             |  1791 |   627K|    31   (4)| 00:00:01 |
|*  1 |  HASH JOIN OUTER   |             |  1791 |   627K|    31   (4)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| T_DATAFILES |     6 |   708 |     3   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| T_TABLES    |  2388 |   562K|    28   (4)| 00:00:01 |
-------------------------------------------------------------------------------------
```

NO_PX_JOIN_FILTER

Usage: NO_PX_JOIN_FILTER(<Table>)

Description: Prevents the optimizer to use bitmap filter to perform hash outer join or parallel join.

```
HELLODBA.COM>exec sql_explain('SELECT /*+ qb_name(M) NO_PX_JOIN_FILTER(t) */t.owner, d.file_id FROM
t_tables t, t_datafiles d WHERE t.tablespace_name(+) = d.tablespace_name', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------
| Id  | Operation              | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
```

```
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT        |                  |  1791 | 59103 |    19   (6)| 00:00:01 |
|*  1 |  HASH JOIN OUTER        |                  |  1791 | 59103 |    19   (6)| 00:00:01 |
|   2 |   TABLE ACCESS FULL     | T_DATAFILES      |     6 |   114 |     3   (0)| 00:00:01 |
|   3 |   VIEW                  | index$_join$_001 |  2388 | 33432 |    16   (7)| 00:00:01 |
|*  4 |    HASH JOIN            |                  |       |       |            |          |
|*  5 |     INDEX FAST FULL SCAN| T_TABLES_IDX3    |  2388 | 33432 |     9   (0)| 00:00:01 |
|   6 |     INDEX FAST FULL SCAN| T_TABLES_IDX1    |  2388 | 33432 |    10   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

NL_SJ

Usage: NL_SJ([<@SubBlock>])

Description: Instructs the optimizer to perform Nested Loop Semi-join with the table in sub-query. .

```
HELLODBA.COM>exec sql_explain('select * from t_tables t where exists (select /*+nl_sj*/1 from t_objects o where t.owner=o.owner)', 'BASIC');


-----------------------------------------------------
| Id  | Operation                | Name               |
-----------------------------------------------------
|   0 | SELECT STATEMENT         |                    |
|   1 |  NESTED LOOPS SEMI       |                    |
|   2 |   TABLE ACCESS FULL      | T_TABLES           |
|   3 |   PARTITION HASH ITERATOR|                    |
|   4 |    INDEX RANGE SCAN      | T_OBJECTS_IDX_PART |
-----------------------------------------------------
```

HASH_SJ

Usage: HASH_SJ([<@SubBlock>])

Description: Instructs the optimizer to perform Hash Semi-join with the table in sub-query..

```
HELLODBA.COM>exec sql_explain('select * from t_tables t where exists (select /*+hash_sj*/1 from t_users u where t.owner=u.username)', 'BASIC');


-------------------------------------------
| Id  | Operation            | Name       |
-------------------------------------------
|   0 | SELECT STATEMENT     |            |
|   1 |  HASH JOIN RIGHT SEMI|            |
|   2 |   INDEX FULL SCAN    | T_USERS_UK |
|   3 |   TABLE ACCESS FULL  | T_TABLES   |
-------------------------------------------
```

MERGE_SJ

Usage: MERGE_SJ([<@SubBlock>])

Description: Instructs the optimizer to perform Merge Semi-join with the table in sub-query.

```
HELLODBA.COM>exec sql_explain('select * from t_tables t where exists (select /*+merge_sj*/1 from t_objects o where t.owner=o.owner)', 'BASIC');
```

```
-----------------------------------------------------
| Id  | Operation                     | Name          |
-----------------------------------------------------
|   0 | SELECT STATEMENT              |               |
|   1 |  MERGE JOIN SEMI              |               |
|   2 |   TABLE ACCESS BY INDEX ROWID | T_TABLES      |
|   3 |    INDEX FULL SCAN            | T_TABLES_IDX1 |
|   4 |   SORT UNIQUE                 |               |
|   5 |    BITMAP CONVERSION TO ROWIDS|               |
|   6 |     BITMAP INDEX FAST FULL SCAN| T_OBJECTS_IDX4 |
-----------------------------------------------------
```

NO_SEMIJOIN

Usage: NO_SEMIJOIN([<@SubBlock>])

Description: Prevent the optimizer to perform Semi-join with the table in sub-query.

```
HELLODBA.COM>exec sql_explain('select * from t_tables t where exists (select /*+NO_SEMIJOIN*/1 from
t_objects o where t.owner=o.owner)', 'BASIC');


-----------------------------------------------------
| Id  | Operation                     | Name          |
-----------------------------------------------------
|   0 | SELECT STATEMENT              |               |
|   1 |  FILTER                       |               |
|   2 |   TABLE ACCESS FULL           | T_TABLES      |
|   3 |   BITMAP CONVERSION TO ROWIDS|               |
|   4 |    BITMAP INDEX SINGLE VALUE | T_OBJECTS_IDX4 |
-----------------------------------------------------
```

SEMIJOIN

Usage: SEMIJOIN([<@SubBlock>])

Description: Instructs the optimizer to perform any type of Semi-join with the table in sub-query.

```
HELLODBA.COM>exec sql_explain('select * from t_tables t where exists (select /*+SEMIJOIN*/1 from
t_objects o where t.owner=o.owner)', 'BASIC');


-----------------------------------------------------
| Id  | Operation                     | Name          |
-----------------------------------------------------
|   0 | SELECT STATEMENT              |               |
|   1 |  HASH JOIN SEMI               |               |
|   2 |   TABLE ACCESS FULL           | T_TABLES      |
|   3 |   BITMAP CONVERSION TO ROWIDS |               |
|   4 |    BITMAP INDEX FAST FULL SCAN| T_OBJECTS_IDX4 |
-----------------------------------------------------
```

SEMIJOIN_DRIVER

Usage: SEMIJOIN_DRIVER([<@SubBlock>])

Description: Instructs the optimizer to use the specified sub-query as driver when performing Semi-join with.

```
HELLODBA.COM>exec sql_explain('SELECT /*+ SEMIJOIN_DRIVER(@inv1) */* FROM t_datafiles d where
tablespace_name = ANY (select /*+ qb_name(inv1) */ tablespace_name from t_tablespaces ts) and file_id
= ANY (select /*+ qb_name(inv2) */user_id from t_users u)', 'BASIC');


-----------------------------------------------
| Id  | Operation           | Name            |
-----------------------------------------------
|   0 | SELECT STATEMENT    |                 |
|   1 |  NESTED LOOPS       |                 |
|   2 |   NESTED LOOPS      |                 |
|   3 |    TABLE ACCESS FULL| T_DATAFILES     |
|   4 |    INDEX UNIQUE SCAN| T_TABLESPACE_PK |
|   5 |   INDEX UNIQUE SCAN | T_USERS_PK      |
-----------------------------------------------
```

SWAP_JOIN_INPUTS

Usage: SWAP_JOIN_INPUTS([<@Block>] <Table1> [<Table2> ...])

Description: Allow the optimizer to swap tables when performing hash-join.

```
HELLODBA.COM>exec sql_explain('select /*+ leading(t) SWAP_JOIN_INPUTS(o) */* from t_tables t, t_objects
o where t.owner=o.owner and t.table_name=o.object_name','BASIC OUTLINE COST');
Plan hash value: 2796668393


-----------------------------------------------------
| Id  | Operation           | Name      | Cost (%CPU)|
-----------------------------------------------------
|   0 | SELECT STATEMENT    |           |   696   (3)|
|   1 |  HASH JOIN          |           |   696   (3)|
|   2 |   TABLE ACCESS FULL| T_OBJECTS |   297   (3)|
|   3 |   TABLE ACCESS FULL| T_TABLES  |    28   (4)|
-----------------------------------------------------
```

NO_SWAP_JOIN_INPUTS

Usage: NO_SWAP_JOIN_INPUTS([<@Block>] <Table1> [<Table2> ...])

Description: Not allow the optimizer to swap tables when performing hash-join.

Demo (11.2.0.1):

```
HELLODBA.COM>exec sql_explain('select /*+NO_SWAP_JOIN_INPUTS(t@inv)*/distinct object_name from
t_objects o where object_name not in (select /*+qb_name(inv)*/table_name from t_tables t)', 'TYPICAL
OUTLINE');


-------------------------------------------------------------------------------------------------
| Id  | Operation              | Name           | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                |  2691 |   120K|       |   320   (6)| 00:00:04 |
|   1 |  HASH UNIQUE           |                |  2691 |   120K|       |   320   (6)| 00:00:04 |
```

```
|*  2 |   HASH JOIN ANTI SNA    |                | 67645 |  3038K|  2608K|   309   (2)| 00:00:04 |
|   3 |    INDEX FAST FULL SCAN| T_OBJECTS_IDX8 | 72116 |  1760K|       |   185   (2)| 00:00:02 |
|   4 |    INDEX FAST FULL SCAN| T_TABLES_PK    |  2696 | 56616 |       |     5   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

Outline Data Hints

QB_NAME

Usage: QB_NAME(<Valid String>)

Description: Defines a name for a query block, which could be used in other hints as a part of global hint format.

```
HELLODBA.COM>exec sql_explain('select /*+full(@INV U@INV)*/* from t_tables t where exists (select
/*+qb_name(inv)*/1 from t_users u where user_id = :A)', 'BASIC OUTLINE');

Outline Data
-------------

  /*+
      BEGIN_OUTLINE_DATA
      … …
      OUTLINE(@"SEL$1")
      OUTLINE(@"INV")
      OUTLINE_LEAF(@"SEL$1")
      … …
      END_OUTLINE_DATA
  */
```

DB_VERSION

Usage: DB_VERSION(<数据库版本>)

Description: It might be a passive hint which indicates the database version of the generated execution plan. It can help to troubleshooting SQL performance issues after RDBMS upgraded. It can be observed in outline data of any execution plan.

IGNORE_OPTIM_EMBEDDED_HINTS

Usage: IGNORE_OPTIM_EMBEDDED_HINTS

Description: Instructs the optimizer to ignore the embedded hints. This hint normally be used in 时 SQL Profiler, Stored Outline and other auxiliary data.

```
HELLODBA.COM>exec sql_explain('SELECT /*+ FULL(O) IGNORE_OPTIM_EMBEDDED_HINTS */ * FROM t_objects o
where object_id<:A', 'TYPICAL OUTLINE');

-------------------------------------------------------------------------------------------------
| Id  | Operation                   | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |               |  2379 |   283K|    10   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| T_OBJECTS     |  2379 |   283K|    10   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | T_OBJECTS_PK  |   428 |       |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

IGNORE_WHERE_CLAUSE

Usage: IGNORE_WHERE_CLAUSE

Description: Instructs the optimizer to ignore the hints following this hint.。

```
HELLODBA.COM>exec sql_explain('SELECT /*+ full(O) IGNORE_WHERE_CLAUSE full(u)*/ COUNT(*) from t_objects
O, t_users u where o.object_id=:A and u.user_id=:B and o.owner=u.username', 'BASIC');


-------------------------------------------------
| Id  | Operation                    | Name       |
-------------------------------------------------
|   0 | SELECT STATEMENT             |            |
|   1 |  SORT AGGREGATE              |            |
|   2 |   NESTED LOOPS               |            |
|   3 |    TABLE ACCESS BY INDEX ROWID| T_USERS    |
|   4 |     INDEX UNIQUE SCAN        | T_USERS_PK |
|   5 |    TABLE ACCESS FULL         | T_OBJECTS  |
-------------------------------------------------
```

NO_ACCESS

Usage: NO_ACCESS([<@block>]<View>)

Description: Indicates that the optimizer did not access to the specified view when analyzing the query block, which means it did not adopt any related optimizing method on the view.

```
HELLODBA.COM>exec sql_explain('SELECT /*+ qb_name(M) no_merge(v) */* FROM t_tables t, v_objects_sys v
WHERE t.owner =v.owner and t.table_name = v.object_name AND v.status = :A', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------
| Id  | Operation           | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                | 15643 |  5560K|   325   (3)| 00:00:04 |
|*  1 |  HASH JOIN          |                | 15643 |  5560K|   325   (3)| 00:00:04 |
|   2 |   TABLE ACCESS FULL | T_TABLES       |  2696 |   634K|    28   (4)| 00:00:01 |
|   3 |   VIEW              | V_OBJECTS_SYS  | 15643 |  1878K|   296   (3)| 00:00:03 |
|*  4 |    TABLE ACCESS FULL| T_OBJECTS      | 15643 |   855K|   296   (3)| 00:00:03 |
-------------------------------------------------------------------------------------


Outline Data
-------------


  /*+
      BEGIN_OUTLINE_DATA
      ... ...
      NO_ACCESS(@"M" "V"@"M")
      FULL(@"M" "T"@"M")
      OUTLINE(@"M")
      OUTLINE_LEAF(@"M")
      ... ...
```

```
      END_OUTLINE_DATA
  */
```

OPTIMIZER_FEATURES_ENABLE

Usage: OPTIMIZER_FEATURES_ENABLE(<版本号>)

Description: Instructs the optimizer to enable the compatible features of specified version only.

```
HELLODBA.COM>alter session set OPTIMIZER_FEATURES_ENABLE='10.2.0.4';

Session altered.

HELLODBA.COM>exec sql_explain('SELECT /*+OPTIMIZER_FEATURES_ENABLE(DEFAULT)*/ * from t_users','BASIC
OUTLINE', FALSE);
... ...

Outline Data
-------------

  /*+
      BEGIN_OUTLINE_DATA
      IGNORE_OPTIM_EMBEDDED_HINTS
      OPTIMIZER_FEATURES_ENABLE('11.2.0.1')
      ... ...
      END_OUTLINE_DATA
  */
```

OPT_PARAM

Usage: OPT_PARAM(<Optimizing Parameter> Value)

Description: Sets the optimizing parameters

```
HELLODBA.COM>exec sql_explain('SELECT /*+QB_NAME(M) OPT_PARAM(''optimizer_index_cost_adj'' 60)*/ *
from t_users u where user_id<:A','BASIC OUTLINE');
... ...

Outline Data
-------------

  /*+
      BEGIN_OUTLINE_DATA
      ... ...
      OPT_PARAM('optimizer_index_cost_adj' 60)
      ... ...
      END_OUTLINE_DATA
  */
```

OUTLINE

Usage: OUTLINE([<@Block>])

Description: Builds an outline for the specified query block.

Refer to the demo in other hints

OUTLINE_LEAF

Usage: OUTLINE_LEAF([<@Block>])

Description: Builds an outline leaf for the specified query block. The outline leaf cannot be transformed.

Demo (Please note that the view cannot be merged due to the built outline leaf):

```
HELLODBA.COM>exec sql_explain('SELECT /*+ qb_name(M) OUTLINE_LEAF(@M) */* FROM t_tables t,
v_objects_sys v WHERE t.owner =v.owner and t.table_name = v.object_name AND v.status = :A', 'TYPICAL
OUTLINE');


-------------------------------------------------------------------------------------------
| Id  | Operation           | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |               | 15643 |  5560K|   325   (3)| 00:00:04 |
|*  1 |  HASH JOIN          |               | 15643 |  5560K|   325   (3)| 00:00:04 |
|   2 |   TABLE ACCESS FULL | T_TABLES      |  2696 |   634K|    28   (4)| 00:00:01 |
|   3 |   VIEW              | V_OBJECTS_SYS | 15643 |  1878K|   296   (3)| 00:00:03 |
|*  4 |    TABLE ACCESS FULL| T_OBJECTS     | 15643 |   855K|   296   (3)| 00:00:03 |
-------------------------------------------------------------------------------------------
```

RBO_OUTLINE

Usage: RBO_OUTLINE

Description: Instructs the optimizer to construct the outline data based on RBO.

Refer to demo in RULE hint

Query Transformation Hints

ANTIJOIN

Usage: ANTIJOIN([<@SubBlock>])

Description: Instructs the optimizer to perform any type of anti-join with the table in sub query.

```
HELLODBA.COM>exec sql_explain('select /*+antijoin(@inv)*/* from t_tables t where not exists (select
/*+qb_name(inv)*/1 from t_users u where t.owner=u.username)', 'BASIC OUTLINE')


----------------------------------------
| Id  | Operation           | Name       |
----------------------------------------
|   0 | SELECT STATEMENT    |            |
|   1 |  NESTED LOOPS ANTI  |            |
|   2 |   TABLE ACCESS FULL| T_TABLES    |
|   3 |   INDEX UNIQUE SCAN| T_USERS_UK  |
----------------------------------------
```

COALESCE_SQ

Usage: COALESCE_SQ([<@Block>])

Description: Instructs the optimizer to perform sub query coalescent.

```
HELLODBA.COM>begin
  2    sql_explain('SELECT /*+qb_name(mn) COALESCE_SQ(@SUB1) COALESCE_SQ(@SUB2)*/d.* FROM t_datafiles
d
```

```
  3         where exists(select /*+qb_name(sub1)*/1 from t_tablespaces ts
where .tablespace_name=ts.tablespace_name and ts.block_size=:A)
  4         and exists(select /*+qb_name(sub2)*/1 from t_tablespaces ts where
d.tablespace_name=ts.tablespace_name)',
  5      'BASIC OUTLINE PREDICATE');
  6  end;
  7  /
Plan hash value: 3571377291

-----------------------------------------
| Id  | Operation          | Name          |
-----------------------------------------
|   0 | SELECT STATEMENT   |               |
|*  1 |  HASH JOIN SEMI    |               |
|   2 |   TABLE ACCESS FULL| T_DATAFILES   |
|*  3 |   TABLE ACCESS FULL| T_TABLESPACES |
-----------------------------------------
```

NO_COALESCE_SQ

Usage: NO_COALESCE_SQ([<@Block>])

Description: Prevents the optimizer to perform sub query coalescent.

```
HELLODBA.COM>begin
  2    sql_explain('SELECT /*+qb_name(mn) NO_COALESCE_SQ(@SUB1) NO_COALESCE_SQ(@SUB2)*/d.* FROM
t_datafiles d
  3         where exists(select /*+qb_name(sub1)*/1 from t_tablespaces ts where
d.tablespace_name=ts.tablespace_name and ts.block_size=:A)
  4         and exists(select /*+qb_name(sub2)*/1 from t_tablespaces ts where
d.tablespace_name=ts.tablespace_name)',
  5      'BASIC OUTLINE PREDICATE');
  6  end;
  7  /
Plan hash value: 3088377872

-------------------------------------------------
| Id  | Operation           | Name            |
-------------------------------------------------
|   0 | SELECT STATEMENT    |                 |
|*  1 |  HASH JOIN SEMI     |                 |
|   2 |   NESTED LOOPS SEMI |                 |
|   3 |    TABLE ACCESS FULL| T_DATAFILES     |
|*  4 |    INDEX UNIQUE SCAN| T_TABLESPACE_PK |
|*  5 |   TABLE ACCESS FULL | T_TABLESPACES   |
-------------------------------------------------
```

ELIMINATE_JOIN

Usage: ELIMINATE_JOIN([<@Block>] <Table>)

Description: Instructs the optimizer to perform query transformation to eliminate join

```
HELLODBA.COM>exec sql_explain('SELECT /*+ELIMINATE_JOIN(TS)*/t.* FROM t_tables t, t_tablespaces ts
where t.tablespace_name=ts.tablespace_name','TYPICAL OUTLINE');

-------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |  1842 |   334K|     6   (0)| 00:00:07 |
|*  1 |  TABLE ACCESS FULL| T_TABLES |  1842 |   334K|     6   (0)| 00:00:07 |
-------------------------------------------------------------------------------
```

NO_ELIMINATE_JOIN

Usage: NO_ELIMINATE_JOIN([<@Block>] <Table>)

Description: Prevents the optimizer to perform query transformation to eliminate join

```
HELLODBA.COM>exec sql_explain('SELECT /*+NO_ELIMINATE_JOIN(TS)*/t.* FROM t_tables t, t_tablespaces ts
where t.tablespace_name=ts.tablespace_name','TYPICAL OUTLINE');

-------------------------------------------------------------------------------------
| Id  | Operation           | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                 |  1842 |   379K|   142   (1)| 00:00:01 |
|   1 |  NESTED LOOPS       |                 |  1842 |   379K|   142   (1)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL| T_TABLES        |  1842 |   366K|   142   (1)| 00:00:01 |
|*  3 |   INDEX UNIQUE SCAN| T_TABLESPACE_PK |     1 |     7 |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------
```

ELIMINATE_OBY

Usage: ELIMINATE_OBY([<@Block>] <Table>)

Description: Instructs the optimizer to perform query transformation to eliminate ORDER BY.

```
HELLODBA.COM>exec sql_explain('select /*+qb_name(m) ELIMINATE_OBY(@inv)*/count(password) from (select
/*+qb_name(inv)*/* from t_users order by user_id)', 'TYPICAL OUTLINE');

-------------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         |     1 |    17 |    19   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE    |         |     1 |    17 |            |          |
|   2 |   TABLE ACCESS FULL| T_USERS |    43 |   731 |    19   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

NO_ELIMINATE_OBY

Usage: NO_ELIMINATE_OBY([<@Block>] <Table>)

Description: Prevents the optimizer to perform query transformation to eliminate ORDER BY.

```
HELLODBA.COM>exec sql_explain('select /*+qb_name(m) NO_ELIMINATE_OBY(@inv)*/count(password) from
(select /*+qb_name(inv)*/* from t_users order by user_id)', 'TYPICAL OUTLINE');

-------------------------------------------------------------------------------------
```

```
| Id  | Operation                     | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |            |     1 |    17 |     4   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE               |            |     1 |    17 |            |          |
|   2 |   VIEW                        |            |    43 |   731 |     4   (0)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID| T_USERS    |    43 |  3698 |     4   (0)| 00:00:01 |
|   4 |     INDEX FULL SCAN           | T_USERS_PK |    43 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

EXPAND_GSET_TO_UNION

Usage: EXPAND_GSET_TO_UNION([<@Block>])

Description: Instructs the optimizer to transformation Grouping Sets to UNION.

```
HELLODBA.COM>exec sql_explain('select /*+EXPAND_GSET_TO_UNION REWRITE*/owner, status,
count(object_name) from t_objects group by owner, rollup(status)', 'TYPICAL OUTLINE');
Plan hash value: 1905288239


-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                |    54 |  1890 |   333   (1)| 00:00:02 |
|   1 |  VIEW                         |                |    54 |  1890 |   333   (1)| 00:00:02 |
|   2 |   UNION-ALL                   |                |       |       |            |          |
|   3 |    SORT GROUP BY NOSORT       |                |    32 |   832 |   322   (1)| 00:00:02 |
|   4 |     INDEX FULL SCAN           | T_OBJECTS_IDX1 | 47585 |  1208K|   322   (1)| 00:00:02 |
|   5 |    MAT_VIEW REWRITE ACCESS FULL| MV_OBJECTS_GP |    22 |   242 |    11   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

NO_EXPAND_GSET_TO_UNION

Usage: NO_EXPAND_GSET_TO_UNION([<@Block>])

Description: Prevents the optimizer to transformation Grouping Sets to UNION.

```
HELLODBA.COM>exec sql_explain('select /*+NO_EXPAND_GSET_TO_UNION REWRITE*/owner, status,
count(object_name) from t_objects group by owner, rollup(status)', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------
| Id  | Operation           | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                |    32 |   416 |   322   (1)| 00:00:02 |
|   1 |  SORT GROUP BY ROLLUP|               |    32 |   416 |   322   (1)| 00:00:02 |
|   2 |   INDEX FULL SCAN   | T_OBJECTS_IDX1 | 47585 |   604K|   322   (1)| 00:00:02 |
-------------------------------------------------------------------------------------
```

EXPAND_TABLE

Usage: EXPAND_TABLE([<@Block>] <Table>)

Description: Instructs the optimizer to perform query transformation to expand partitioned table.

```
HELLODBA.COM>exec sql_explain('select /*+EXPAND_TABLE(o)*/* from t_objects_list o','BASIC OUTLINE');


-----------------------------------------------
```

```
| Id  | Operation              | Name           |
-------------------------------------------------
|   0 | SELECT STATEMENT       |                |
|   1 |  VIEW                  | VW_TE_1        |
|   2 |   UNION-ALL            |                |
|   3 |    PARTITION LIST ALL  |                |
|   4 |     TABLE ACCESS FULL  | T_OBJECTS_LIST |
|   5 |    PARTITION LIST SINGLE|               |
|   6 |     TABLE ACCESS FULL  | T_OBJECTS_LIST |
-------------------------------------------------
```

NO_EXPAND_TABLE

Usage: NO_EXPAND_TABLE([<@Block>] <Table>)

Description: Prevents the optimizer to perform query transformation to expand partitioned table.

```
HELLODBA.COM>exec sql_explain('select /*+NO_EXPAND_TABLE(o)*/* from t_objects_list o','BASIC
OUTLINE');


-------------------------------------------
| Id  | Operation          | Name           |
-------------------------------------------
|   0 | SELECT STATEMENT   |                |
|   1 |  PARTITION LIST ALL|                |
|   2 |   TABLE ACCESS FULL| T_OBJECTS_LIST |
-------------------------------------------
```

STAR

Usage: STAR

Description: Instructs the optimizer to use the old method in 8i to perform start query. The old method of star query utilizes nested-loop join.

```
HELLODBA.COM>begin
  2    sql_explain('select /*+ QB_NAME(Q) STAR OPTIMIZER_FEATURES_ENABLE(''8.1.7'') */
  3                  u.user_id, i.table_type, t.degree, count(R_CONSTRAINT_NAME)
  4                from t_constraints c, t_tables t, t_users u, t_indexes i
  5                where c.table_name = t.table_name and c.owner = t.owner
  6                and c.r_owner =  u.username and c.constraint_name = i.index_name
  7                and t.status = :A and u.default_tablespace = :B and i.index_type = :C
  8                group by u.user_id, i.table_type, t.degree',
  9     'TYPICAL OUTLINE');
 10  end;
 11  /
Plan hash value: 2020912536


-------------------------------------------------------------------------------------
| Id  | Operation                     | Name              | Rows  | Bytes | Cost  |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                   |    13 |  1703 |   109 |
```

```
|   1 |  SORT GROUP BY                 |                    |    13 |  1703 |   109 |
|*  2 |   HASH JOIN                    |                    |    58 |  7598 |   101 |
|   3 |    NESTED LOOPS                |                    |    58 |  5626 |    83 |
|   4 |     NESTED LOOPS               |                    |    58 |  3248 |    25 |
|*  5 |      TABLE ACCESS FULL         | T_USERS            |     6 |   108 |     1 |
|   6 |      TABLE ACCESS BY INDEX ROWID| T_CONSTRAINTS      |    10 |   380 |     4 |
|*  7 |       INDEX RANGE SCAN         | T_CONSTRAINTS_IDX4 |    59 |       |     2 |
|*  8 |     TABLE ACCESS BY INDEX ROWID | T_TABLES           |     1 |    41 |     1 |
|*  9 |      INDEX UNIQUE SCAN         | T_TABLES_PK        |     1 |       |       |
|* 10 |    TABLE ACCESS FULL           | T_INDEXES          |   648 | 22032 |    17 |
-------------------------------------------------------------------------------------
```

STAR_TRANSFORMATION

Usage: STAR_TRANSFORMATION([<@Block>] [<Fact Table>] [SUBQUERIES(<Dimension Table1> < Dimension Table 2>[< Dimension Table 3>...])])

Description: Instructs the optimizer to perform star query transformation.

```
HELLODBA.COM>alter session set star_transformation_enabled=true;

Session altered.

HELLODBA.COM>exec sql_explain('select /*+ QB_NAME(Q) STAR_TRANSFORMATION FACT(T) NO_FACT(TS)*/* from
t_tables t, t_tablespaces ts, t_users u where t.tablespace_name=ts.tablespace_name and
t.owner=u.username','TYPICAL OUTLINE');

-------------------------------------------------------------------------------------------------------
---
| Id  | Operation                           | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
---
|   0 | SELECT STATEMENT                    |                |  1840 |   691K|  1515   (1)| 00:00:07
|
|*  1 |  HASH JOIN                          |                |  1840 |   691K|  1515   (1)| 00:00:07
|
|   2 |   TABLE ACCESS FULL                 | T_TABLESPACES  |    15 |  1425 |    11   (0)| 00:00:01
|
|*  3 |   HASH JOIN                         |                |  1840 |   521K|  1503   (1)| 00:00:07
|
|   4 |    TABLE ACCESS FULL                | T_USERS        |    43 |  3698 |    19   (0)| 00:00:01
|
|   5 |    TABLE ACCESS BY INDEX ROWID      | T_TABLES       |  1842 |   366K|  1484   (1)| 00:00:06
|
|   6 |     BITMAP CONVERSION TO ROWIDS     |                |       |       |            |
|
|   7 |      BITMAP AND                     |                |       |       |            |
|
```

```
|   8 |       BITMAP MERGE                  |               |       |       |           |
|
|   9 |        BITMAP KEY ITERATION         |               |       |       |           |
|
|  10 |         TABLE ACCESS FULL           | T_USERS       |    43 |  3698 |    19   (0)| 00:00:01
|
|  11 |         BITMAP CONVERSION FROM ROWIDS|               |       |       |           |
|
|* 12 |          INDEX RANGE SCAN           | T_TABLES_IDX1 |       |       |     1   (0)| 00:00:01
|
|  13 |       BITMAP MERGE                  |               |       |       |           |
|
|  14 |        BITMAP KEY ITERATION         |               |       |       |           |
|
|  15 |         TABLE ACCESS FULL           | T_TABLESPACES |    15 |  1425 |    11   (0)| 00:00:01
|
|  16 |         BITMAP CONVERSION FROM ROWIDS|               |       |       |           |
|
|* 17 |          INDEX RANGE SCAN           | T_TABLES_IDX3 |       |       |     1   (0)| 00:00:01
|
-------------------------------------------------------------------------------------------------
---
```

NO_STAR_TRANSFORMATION

Usage: NO_STAR_TRANSFORMATION([<@Block>])

Description: Prevents the optimizer to perform star query transformation.

```
HELLODBA.COM>exec sql_explain('select /*+ QB_NAME(Q) NO_STAR_TRANSFORMATION(@Q)*/* from t_tables t,
t_tablespaces ts, t_users u where t.tablespace_name=ts.tablespace_name and
t.owner=u.username','TYPICAL OUTLINE');


-------------------------------------------------------------------------------
| Id  | Operation           | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |               |  1840 |   691K|   173   (2)| 00:00:01 |
|*  1 |  HASH JOIN          |               |  1840 |   691K|   173   (2)| 00:00:01 |
|   2 |   TABLE ACCESS FULL | T_TABLESPACES |    15 |  1425 |    11   (0)| 00:00:01 |
|*  3 |   HASH JOIN         |               |  1840 |   521K|   161   (1)| 00:00:01 |
|   4 |    TABLE ACCESS FULL| T_USERS       |    43 |  3698 |    19   (0)| 00:00:01 |
|*  5 |    TABLE ACCESS FULL| T_TABLES      |  1842 |   366K|   142   (1)| 00:00:01 |
-------------------------------------------------------------------------------
```

FACT

Usage: FACT([<@Block>] <Table>)

Description: Instructs the optimizer to use the specified table as fact table when performing star query transformation.

见 STAR_TRANSFORMATION 示例

NO_FACT

Usage: NO_FACT([<@Block>] <Table>)

Description: Prevents the optimizer to use the specified table as fact table when performing star query transformation.

Refer to the demo of STAR_TRANSFORMATION

FACTORIZE_JOIN

Usage: FACTORIZE_JOIN(<Data Set>(<Table>@<blok1> <Table>@<block2>[<Table>@<block3> ...]))

Description: Instructs the optimizer to merge the sub queries in UNION/UNION-ALL to an inline view.

```
HELLODBA.COM>begin
  2    sql_explain('
  3         select  /*+ FACTORIZE_JOIN(@SET$1(0@SB1 0@SB2)) qb_name(sb1) */ u.username, u.created,
o.object_name from t_objects o, t_users u
  4         where o.owner=u.username and u.lock_date=:A
  5         union all
  6         select /*+ qb_name(sb2) */ u.username, u.created, o.object_name from t_objects o, t_users
u
  7         where o.owner=u.username and u.lock_date=:B',
  8      'TYPICAL OUTLINE');
  9  end;
 10  /


-------------------------------------------------------------------------------------------------
| Id  | Operation             | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                     | 81522 |  5891K|   193   (3)| 00:00:02 |
|*  1 |  HASH JOIN            |                     | 81522 |  5891K|   193   (3)| 00:00:02 |
|   2 |   VIEW                | VW_JF_SET$A6672D85 |    26 |  1118 |     6   (0)| 00:00:01 |
|   3 |    UNION-ALL          |                     |       |       |            |          |
|*  4 |     TABLE ACCESS FULL | T_USERS             |    13 |   325 |     3   (0)| 00:00:01 |
|*  5 |     TABLE ACCESS FULL | T_USERS             |    13 |   325 |     3   (0)| 00:00:01 |
|   6 |   INDEX FAST FULL SCAN| T_OBJECTS_IDX8      | 72116 |  2183K|   185   (2)| 00:00:02 |
-------------------------------------------------------------------------------------------------
```

NO_FACTORIZE_JOIN

Usage: NO_FACTORIZE_JOIN(<Data Set>)

Description: Prevents the optimizer to merge the sub queries in UNION/UNION-ALL to an inline view.

```
HELLODBA.COM>begin
  2    sql_explain('
  3         select  /*+ NO_FACTORIZE_JOIN(@SET$1) qb_name(sb1) */ u.username, u.created, o.object_name
from t_objects o, t_users u
  4         where o.owner=u.username and u.lock_date=:A
  5         union all
```

```
  6         select /*+ qb_name(sb2) */ u.username, u.created, o.object_name from t_objects o, t_users
u
  7         where o.owner=u.username and u.lock_date=:B',
  8      'TYPICAL OUTLINE');
  9  end;
 10  /


-------------------------------------------------------------------------------------------
| Id  | Operation              | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                | 81522 |  4458K|   379  (52)| 00:00:04 |
|   1 |  UNION-ALL             |                |       |       |            |          |
|*  2 |   HASH JOIN            |                | 40761 |  2229K|   190   (3)| 00:00:02 |
|*  3 |    TABLE ACCESS FULL   | T_USERS        |    13 |   325 |     3   (0)| 00:00:01 |
|   4 |    INDEX FAST FULL SCAN| T_OBJECTS_IDX8 | 72116 |  2183K|   185   (2)| 00:00:02 |
|*  5 |   HASH JOIN            |                | 40761 |  2229K|   190   (3)| 00:00:02 |
|*  6 |    TABLE ACCESS FULL   | T_USERS        |    13 |   325 |     3   (0)| 00:00:01 |
|   7 |    INDEX FAST FULL SCAN| T_OBJECTS_IDX8 | 72116 |  2183K|   185   (2)| 00:00:02 |
-------------------------------------------------------------------------------------------
```

MATERIALIZE

Usage: MATERIALIZE

Description: Instructs the optimizer to materialize the inline view, which will generate a temp table based on the view.

```
HELLODBA.COM>exec sql_explain('with v as (select /*+ MATERIALIZE qb_name(wv) */* from t_objects o where
object_id<:A) select count(*) from v', 'BASIC OUTLINE');


-----------------------------------------------------------------------
| Id  | Operation                 | Name                          |
-----------------------------------------------------------------------
|   0 | SELECT STATEMENT          |                               |
|   1 |  TEMP TABLE TRANSFORMATION |                               |
|   2 |   LOAD AS SELECT          | SYS_TEMP_0FD9D6601_F201F06C |
|   3 |    TABLE ACCESS FULL      | T_OBJECTS                     |
|   4 |   SORT AGGREGATE          |                               |
|   5 |    VIEW                   |                               |
|   6 |     TABLE ACCESS FULL     | SYS_TEMP_0FD9D6601_F201F06C |
-----------------------------------------------------------------------
```

INLINE

Usage: INLINE

Description: Prevents the optimizer to materialize the inline view

```
HELLODBA.COM>alter session set "_with_subquery"=materialize;


Session altered.
```

```
HELLODBA.COM>exec sql_explain('with v as (select /*+ INLINE qb_name(wv) */* from t_objects o where
object_id<:A) selectcount(*) from v', 'BASIC OUTLINE');


-----------------------------------------------
| Id  | Operation             | Name          |
-----------------------------------------------
|   0 | SELECT STATEMENT      |               |
|   1 |  SORT AGGREGATE       |               |
|   2 |   INDEX FAST FULL SCAN| T_OBJECTS_IDX8 |
-----------------------------------------------
```

MERGE

Usage: MERGE([<@Block>] [<Table>]) or MERGE([<View>] [<Table>])

Description: Instructs the optimizer to perform query transformation to merge the specified view or sub query.

```
HELLODBA.COM>exec sql_explain('select /*+ qb_name(M) merge(@inv) */* from t_tables t, (select /*+
qb_name(inv)*/* from t_objects o where object_type = :A) v where t.owner=v.owner and
t.table_name=v.object_name', 'BASIC OUTLINE');


-----------------------------------------------------
| Id  | Operation                     | Name          |
-----------------------------------------------------
|   0 | SELECT STATEMENT              |               |
|   1 |  HASH JOIN                    |               |
|   2 |   TABLE ACCESS BY INDEX ROWID| T_OBJECTS      |
|   3 |    INDEX RANGE SCAN           | T_OBJECTS_IDX7 |
|   4 |   TABLE ACCESS FULL           | T_TABLES       |
-----------------------------------------------------
```

NO_MERGE

Usage: NO_MERGE([<@Block>] [<Table>]) or NO_MERGE([<view>] [<Table>])

Description: Prevents the optimizer to perform query transformation to merge the specified view or sub query.。

```
HELLODBA.COM>exec sql_explain('select /*+ qb_name(M) no_merge(@inv) */* from t_tables t, (select /*+
qb_name(inv)*/* from t_objects o where object_type = :A) v where t.owner=v.owner and
t.table_name=v.object_name', 'BASIC OUTLINE');


-----------------------------------------------------
| Id  | Operation                     | Name          |
-----------------------------------------------------
|   0 | SELECT STATEMENT              |               |
|   1 |  HASH JOIN                    |               |
|   2 |   VIEW                        |               |
|   3 |    TABLE ACCESS BY INDEX ROWID| T_OBJECTS      |
|   4 |     INDEX RANGE SCAN          | T_OBJECTS_IDX7 |
|   5 |   TABLE ACCESS FULL           | T_TABLES       |
```

```
-----------------------------------------------------
```

NO_PRUNE_GSETS

Usage: NO_PRUNE_GSETS

Description: Prevents the optimizer to prune Grouping Sets query.

```
HELLODBA.COM>exec sql_explain('select /*+ qb_name(m) */v.owner, v.table_name, v.constraint_type,
cns_cnt from (select /*+ NO_PRUNE_GSETS qb_name(gv) */owner, table_name, constraint_type,
count(constraint_name) cns_cnt from t_constraints c group by cube(owner, table_name, constraint_type))
v where v.owner = ''DEMO''','BASIC OUTLINE');


-------------------------------------------------------------
| Id  | Operation                      | Name               |
-------------------------------------------------------------
|   0 | SELECT STATEMENT               |                    |
|   1 |  VIEW                          |                    |
|   2 |   FILTER                       |                    |
|   3 |    SORT GROUP BY               |                    |
|   4 |     GENERATE CUBE              |                    |
|   5 |      SORT GROUP BY             |                    |
|   6 |       TABLE ACCESS BY INDEX ROWID| T_CONSTRAINTS      |
|   7 |        INDEX RANGE SCAN        | T_CONSTRAINTS_IDX3 |
-------------------------------------------------------------
```

NO_QUERY_TRANSFORMATION

Usage: NO_QUERY_TRANSFORMATION

Description: Prevents the optimizer to perform any query transformation

```
HELLODBA.COM>exec sql_explain('select /*+NO_QUERY_TRANSFORMATION*/* from (select * from t_tables)',
'BASIC');


---------------------------------------
| Id  | Operation          | Name     |
---------------------------------------
|   0 | SELECT STATEMENT   |          |
|   1 |  VIEW              |          |
|   2 |   TABLE ACCESS FULL| T_TABLES |
---------------------------------------
```

OR_EXPAND

Usage: OR_EXPAND([<@Block>] <Table> <Column1> [<Column2> ...])

Description: Instructs the optimizer to perform OR EXPAND query transformation.

```
HELLODBA.COM>exec sql_explain('select /*+OR_EXPAND(o created)*/* from t_objects o where created = :A
or (owner = :B and object_name=:C)', 'TYPICAL OUTLINE');


-----------------------------------------------------------------------------------------------
| Id  | Operation                    | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                |    32 |  3456 |     4   (0)| 00:00:05 |
```

```
|   1 |  CONCATENATION                |                |       |       |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID| T_OBJECTS      |     1 |   108 |     3   (0)| 00:00:04 |
|*  3 |    INDEX SKIP SCAN            | T_OBJECTS_IDX1 |     1 |       |     2   (0)| 00:00:03 |
|*  4 |   TABLE ACCESS BY INDEX ROWID| T_OBJECTS      |    31 |  3348 |     1   (0)| 00:00:02 |
|*  5 |    INDEX RANGE SCAN           | T_OBJECTS_IDX5 |    31 |       |     1   (0)| 00:00:02 |
-------------------------------------------------------------------------------------------
```

OUTER_JOIN_TO_INNER

Usage: OUTER_JOIN_TO_INNER([<@Block>])

Description: Instructs the optimizer to perform query transformation to transform outer join to 指示 inner join.

```
HELLODBA.COM>exec sql_explain('select /*+qb_name(M) OUTER_JOIN_TO_INNER(@M)*/ t.owner, u.user_id from
t_tables t, t_users u where t.owner=u.username(+) and u.created < :A', 'TYPICAL OUTLINE');


-----------------------------------------------------------------------------------------------
| Id  | Operation                    | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |               |   343 |  9947 |     4   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                |               |   343 |  9947 |     4   (0)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID| T_USERS       |     2 |    44 |     2   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN          | T_USERS_IDX1  |     2 |       |     1   (0)| 00:00:01 |
|*  4 |   INDEX RANGE SCAN           | T_TABLES_IDX1 |   150 |  1050 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

NO_OUTER_JOIN_TO_INNER

Usage: NO_OUTER_JOIN_TO_INNER

Description: Prevents the optimizer to perform query transformation to transform outer join to 指示 inner join.

```
HELLODBA.COM>exec sql_explain('select /*+qb_name(M) NO_OUTER_JOIN_TO_INNER*/ t.owner, u.user_id from
t_tables t, t_users u where t.owner=u.username(+) and u.created < :A', 'TYPICAL OUTLINE');


-----------------------------------------------------------------------------------------------
| Id  | Operation                | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |               |  2696 | 78184 |     7  (15)| 00:00:01 |
|*  1 |  FILTER                  |               |       |       |            |          |
|*  2 |   HASH JOIN RIGHT OUTER  |               |  2696 | 78184 |     7  (15)| 00:00:01 |
|   3 |    TABLE ACCESS FULL     | T_USERS       |    31 |   682 |     3   (0)| 00:00:01 |
|   4 |    INDEX FAST FULL SCAN  | T_TABLES_IDX1 |  2696 | 18872 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

ELIMINATE_OUTER_JOIN

Usage: ELIMINATE_OUTER_JOIN([<@Block>])

Description: Instructs the optimizer to perform query transformation to eliminate outer join (10g).

```
HELLODBA.COM>exec sql_explain('select /*+qb_name(M) ELIMINATE_OUTER_JOIN(@M)*/ t.owner, u.user_id from
t_tables t, t_users u where t.owner=u.username(+) and u.created < :A', 'TYPICAL OUTLINE');
```

```
-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |               |   103 |  2575 |     4   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                |               |   103 |  2575 |     4   (0)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID| T_USERS       |     2 |    38 |     2   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN          | T_USERS_IDX1  |     2 |       |     1   (0)| 00:00:01 |
|*  4 |   INDEX RANGE SCAN           | T_TABLES_IDX1 |    48 |   288 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

NO_ELIMINATE_OUTER_JOIN

Usage: NO_ELIMINATE_OUTER_JOIN

Description: Prevents the optimizer to perform query transformation to eliminate outer join (10g).

```
HELLODBA.COM>exec sql_explain('select /*+qb_name(M) NO_ELIMINATE_OUTER_JOIN(@M)*/ t.owner, u.user_id
from t_tables t, t_users u where t.owner=u.username(+) and u.created < :A', 'TYPICAL OUTLINE');

-------------------------------------------------------------------------------------------
| Id  | Operation                | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                  |  2071 | 51775 |    13  (16)| 00:00:01 |
|*  1 |  FILTER                  |                  |       |       |            |          |
|*  2 |   HASH JOIN RIGHT OUTER  |                  |  2071 | 51775 |    13  (16)| 00:00:01 |
|   3 |    VIEW                  | index$_join$_002 |    43 |   817 |     6  (17)| 00:00:01 |
|*  4 |     HASH JOIN            |                  |       |       |            |          |
|*  5 |      HASH JOIN           |                  |       |       |            |          |
|   6 |       INDEX FAST FULL SCAN| T_USERS_IDX1    |    43 |   817 |     1   (0)| 00:00:01 |
|   7 |       INDEX FAST FULL SCAN| T_USERS_PK      |    43 |   817 |     1   (0)| 00:00:01 |
|   8 |      INDEX FAST FULL SCAN | T_USERS_UK      |    43 |   817 |     1   (0)| 00:00:01 |
|   9 |    INDEX FULL SCAN       | T_TABLES_IDX1    |  2071 | 12426 |     6   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

PLACE_DISTINCT

Usage: PLACE_DISTINCT

Description: Instructs the optimizer to perform Distinct Placement query transformation.

```
HELLODBA.COM>exec sql_explain('select /*+full(u) full(t) place_distinct*/distinct t.tablespace_name,
u.account_status from t_tables t, t_users u where t.owner=u.username', 'BASIC OUTLINE');

-----------------------------------------------
| Id  | Operation            | Name           |
-----------------------------------------------
|   0 | SELECT STATEMENT     |                |
|   1 |  HASH UNIQUE         |                |
|   2 |   HASH JOIN          |                |
|   3 |    TABLE ACCESS FULL | T_USERS        |
```

```
|   4 |    VIEW               | VW_DTP_1B35BA0F |
|   5 |     HASH UNIQUE       |                 |
|   6 |      TABLE ACCESS FULL| T_TABLES        |
-------------------------------------------------
```

NO_PLACE_DISTINCT

Usage: NO_PLACE_DISTINCT

Description: Prevents the optimizer to perform Distinct Placement query transformation.

```
HELLODBA.COM>exec sql_explain('select /*+full(u) full(t) no_place_distinct*/distinct
t.tablespace_name, u.account_status from t_tables t, t_users u where t.owner=u.username', 'BASIC');


----------------------------------------
| Id  | Operation           | Name     |
----------------------------------------
|   0 | SELECT STATEMENT    |          |
|   1 |  HASH UNIQUE        |          |
|   2 |   HASH JOIN         |          |
|   3 |    TABLE ACCESS FULL| T_USERS  |
|   4 |    TABLE ACCESS FULL| T_TABLES |
----------------------------------------
```

PLACE_GROUP_BY

Usage: PLACE_GROUP_BY([<@Block>] (<Table>) [<Temp View Number>])

Description: Instructs the optimizer to perform Group By Placement query transformation.

```
HELLODBA.COM>exec sql_explain('SELECT /*+ qb_name(m) place_group_by(@m (t@m) 10000) */owner,
max(maxbytes) FROM t_tables t, t_datafiles d WHERE t.tablespace_name = d.tablespace_name GROUP BY
t.owner', 'BASIC OUTLINE');
Plan hash value: 1494419902


-----------------------------------------------
| Id  | Operation             | Name          |
-----------------------------------------------
|   0 | SELECT STATEMENT      |               |
|   1 |  HASH GROUP BY        |               |
|   2 |   HASH JOIN           |               |
|   3 |    TABLE ACCESS FULL  | T_DATAFILES   |
|   4 |    VIEW               | VW_GBF_10000  |
|   5 |     HASH GROUP BY     |               |
|   6 |      TABLE ACCESS FULL| T_TABLES      |
-----------------------------------------------
```

NO_PLACE_GROUP_BY

Usage: NO_PLACE_GROUP_BY([<@Block>] (<Table>))

Description: Prevents the optimizer to perform Group By Placement query transformation.

```
HELLODBA.COM>exec sql_explain('SELECT /*+ qb_name(m) no_place_group_by(@m) */owner, max(maxbytes) FROM
t_tables t, t_datafiles d WHERE t.tablespace_name = d.tablespace_name GROUP BY t.owner', 'BASIC
OUTLINE');
```

```
-------------------------------------------------
| Id  | Operation               | Name            |
-------------------------------------------------
|   0 | SELECT STATEMENT        |                 |
|   1 |  HASH GROUP BY          |                 |
|   2 |   HASH JOIN             |                 |
|   3 |    TABLE ACCESS FULL    | T_DATAFILES     |
|   4 |    VIEW                 | index$_join$_001 |
|   5 |     HASH JOIN           |                 |
|   6 |      INDEX FAST FULL SCAN| T_TABLES_IDX3  |
|   7 |      INDEX FAST FULL SCAN| T_TABLES_IDX1  |
-------------------------------------------------
```

PRECOMPUTE_SUBQUERY

Usage: PRECOMPUTE_SUBQUERY

Description: Instructs the optimizer to pre-compute cost of sub query, then perform query optimizing based on the result. This hint may change the result of the query.

```
HELLODBA.COM>exec sql_explain('select /*+ PRECOMPUTE_SUBQUERY(@inv) */* from t_tables t where status
in (select /*+qb_name(inv)*/status from t_indexes i where status is not null)', 'TYPICAL');


-------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |  2070 |   412K|   142   (1)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| T_TABLES |  2070 |   412K|   142   (1)| 00:00:01 |
-------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------


   1 - filter("STATUS"='N/A' OR "STATUS"='UNUSABLE' OR "STATUS"='VALID')
```

PULL_PRED

Usage: PULL_PRED([<@Block>]<View> [<Predication Position 1> ...])

Description: Instructs the optimizer to pull the complex predication in the view to the main query.

```
HELLODBA.COM>begin
  2     sql_explain('
  3     SELECT /*+qb_name(outv) PULL_PRED(@OUTV V 2)*/ owner, table_name, rownum
  4     FROM
  5     (SELECT /*+qb_name(inv)*/t.owner, t.table_name, t.last_analyzed
  6       FROM t_tables t, t_datafiles d
  7       WHERE t.tablespace_name = d.tablespace_name
  8       AND (t.last_analyzed) < (SELECT /*+qb_name(subq1)*/ MAX(created) FROM t_objects o)
  9       AND (d.user_blocks) > (SELECT /*+qb_name(subq2)*/ MAX(LEAF_BLOCKS) FROM t_indexes i)
```

```
 10       ORDER BY 1
 11     )v','TYPICAL PREDICATE');
 12  end;
 13  /
-------------------------------------------------------------------------------------------------
--
| Id  | Operation                      | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
--
|   0 | SELECT STATEMENT               |                  |    92 |  4324 |   500   (1)| 00:00:02 |
|   1 |  COUNT                         |                  |       |       |            |
|
|*  2 |   VIEW                         |                  |    92 |  4324 |   148   (3)| 00:00:01 |
|   3 |    SORT ORDER BY               |                  |    92 |  4692 |   148   (3)| 00:00:01 |
|   4 |     MERGE JOIN                 |                  |    92 |  4692 |   145   (2)| 00:00:01 |
|   5 |      TABLE ACCESS BY INDEX ROWID | T_DATAFILES      |    14 |   168 |     2   (0)| 00:00:01 |
|   6 |       INDEX FULL SCAN          | T_DATAFILES_IDX1 |    14 |       |     1   (0)| 00:00:01 |
|*  7 |      SORT JOIN                 |                  |    92 |  3588 |   143   (2)| 00:00:01 |
|*  8 |       TABLE ACCESS FULL        | T_TABLES         |    92 |  3588 |   142   (1)| 00:00:01 |
|   9 |        SORT AGGREGATE          |                  |     1 |     8 |            |          |
|  10 |         INDEX FULL SCAN (MIN/MAX)| T_OBJECTS_IDX5   | 47585 |  371K|     2   (0)| 00:00:01 |
|  11 |    SORT AGGREGATE              |                  |     1 |     3 |            |          |
|  12 |     TABLE ACCESS FULL          | T_INDEXES        |  5833 | 17499 |   352   (1)| 00:00:02 |
-------------------------------------------------------------------------------------------------
--

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("USER_BLOCKS"> (SELECT /*+ QB_NAME ("SUBQ2") */ MAX("LEAF_BLOCKS") FROM
              "T_INDEXES" "I"))
   7 - access("T"."TABLESPACE_NAME"="D"."TABLESPACE_NAME")
       filter("T"."TABLESPACE_NAME"="D"."TABLESPACE_NAME")
   8 - filter("T"."TABLESPACE_NAME" IS NOT NULL AND "T"."LAST_ANALYZED"< (SELECT /*+ QB_NAME
              ("SUBQ1") */ MAX("CREATED") FROM "T_OBJECTS" "O"))
```

NO_PULL_PRED

Usage: NO_PULL_PRED([<@Block>]<View> [<Predication Position 1> ...]))

Description: Prevents the optimizer to pull the complex predication in the view to the main query.

```
HELLODBA.COM>begin
  2     sql_explain('
  3     SELECT /*+qb_name(outv) NO_PULL_PRED(@OUTV V 2)*/ owner, table_name, rownum
  4     FROM
  5     (SELECT /*+qb_name(inv)*/t.owner, t.table_name, t.last_analyzed
```

```
  6       FROM t_tables t
  7       WHERE (t.last_analyzed) < (SELECT /*+qb_name(subq1)*/ MAX(created) FROM t_objects o)
  8       AND (t.sample_size) > (SELECT /*+qb_name(subq2)*/ MAX(sample_size) FROM t_indexes i)
  9       AND owner like ''A%''
 10       ORDER BY 1
 11     )v','TYPICAL');
 12  end;
 13  /

-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                |     1 |    43 |   357   (1)| 00:00:02 |
|   1 |  COUNT                        |                |       |       |            |          |
|*  2 |   VIEW                        |                |     1 |    43 |   355   (1)| 00:00:02 |
|*  3 |    TABLE ACCESS BY INDEX ROWID| T_TABLES       |     1 |    35 |     3   (0)| 00:00:01 |
|*  4 |     INDEX RANGE SCAN          | T_TABLES_IDX1  |     2 |       |     2   (0)| 00:00:01 |
|   5 |     SORT AGGREGATE            |                |     1 |     3 |            |          |
|   6 |      TABLE ACCESS FULL        | T_INDEXES      |  5833 | 17499 |   352   (1)| 00:00:02 |
|   7 |    SORT AGGREGATE             |                |     1 |     8 |            |          |
|   8 |     INDEX FULL SCAN (MIN/MAX) | T_OBJECTS_IDX5 | 47585 |   371K|     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------


   2 - filter("LAST_ANALYZED"< (SELECT /*+ QB_NAME ("SUBQ1") */ MAX("CREATED") FROM
              "T_OBJECTS" "O"))
   3 - filter("T"."SAMPLE_SIZE"> (SELECT /*+ QB_NAME ("SUBQ2") */ MAX("SAMPLE_SIZE")
              FROM "T_INDEXES" "I"))
   4 - access("OWNER" LIKE 'A%')
       filter("OWNER" LIKE 'A%')
```

OLD_PUSH_PRED

Usage: OLD_PUSH_PRED([<@Block>]<View>)

Description: Instructs the optimizer to perform old push predication query transformation.

```
HELLODBA.COM>alter session set "_OPTIMIZER_COST_BASED_TRANSFORMATION"=off;


Session altered.


HELLODBA.COM>exec sql_explain('SELECT /*+ NO_MERGE(v) OLD_PUSH_PRED(v) */* FROM t_tables t,
v_objects_sys v WHERE t.owner =v.owner(+) and t.table_name = v.object_name(+) AND t.tablespace_name
= :A', 'BASIC PREDICATE');


-------------------------------------------------------------
```

```
| Id  | Operation                     | Name               |
-----------------------------------------------------------
|   0 | SELECT STATEMENT              |                    |
|   1 |  NESTED LOOPS OUTER           |                    |
|   2 |   TABLE ACCESS BY INDEX ROWID | T_TABLES           |
|*  3 |    INDEX RANGE SCAN           | T_TABLES_IDX3      |
|   4 |   PARTITION HASH SINGLE       |                    |
|*  5 |    VIEW PUSHED PREDICATE      | V_OBJECTS_SYS      |
|   6 |     TABLE ACCESS BY INDEX ROWID| T_OBJECTS         |
|*  7 |      INDEX RANGE SCAN         | T_OBJECTS_IDX_PART |
-----------------------------------------------------------
```

PUSH_PRED

Usage: PUSH_PRED([<@Block>]<View> [<Predication Position 1> ...])

Description: Instructs the optimizer to perform query transformation to push the predication to the view.

```
HELLODBA.COM>exec sql_explain('SELECT /*+ NO_MERGE(v) PUSH_PRED(v) */* FROM t_tables t, v_objects_sys
v WHERE t.owner =v.owner(+) and t.table_name = v.object_name(+) AND t.tablespace_name = :A', 'TYPICAL');
Plan hash value: 4224448473


-------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                |  2033 |   559K|   758   (1)| 00:00:04 |
|   1 |  NESTED LOOPS OUTER           |                |  2033 |   559K|   758   (1)| 00:00:04 |
|   2 |   TABLE ACCESS BY INDEX ROWID | T_TABLES       |   184 | 37536 |    21   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN           | T_TABLES_IDX3  |   184 |       |     1   (0)| 00:00:01 |
|   4 |   VIEW PUSHED PREDICATE       | V_OBJECTS_SYS  |     1 |    78 |     4   (0)| 00:00:01 |
|*  5 |    FILTER                     |                |       |       |            |          |
|   6 |     TABLE ACCESS BY INDEX ROWID| T_OBJECTS      |     1 |    77 |     4   (0)| 00:00:01 |
|*  7 |      INDEX RANGE SCAN         | T_OBJECTS_IDX8 |     1 |       |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

NO_PUSH_PRED

Usage: NO_PUSH_PRED([<@Block>]<View>)

Description: Prevents the optimizer to perform query transformation to push the predication to the view.

```
HELLODBA.COM>exec sql_explain('SELECT /*+ NO_PUSH_PRED(v) */* FROM t_tables t, v_objects_sys v WHERE
t.owner =v.owner(+) and t.table_name = v.object_name(+) AND t.tablespace_name = :A', 'BASIC PREDICATE');


-----------------------------------------------------
| Id  | Operation                     | Name          |
-----------------------------------------------------
|   0 | SELECT STATEMENT              |               |
|   1 |  NESTED LOOPS OUTER           |               |
|   2 |   TABLE ACCESS BY INDEX ROWID| T_TABLES       |
```

```
|*  3 |    INDEX RANGE SCAN          | T_TABLES_IDX3  |
|   4 |   TABLE ACCESS BY INDEX ROWID| T_OBJECTS      |
|*  5 |    INDEX RANGE SCAN          | T_OBJECTS_IDX8 |
---------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("T"."TABLESPACE_NAME"=:A)
   5 - access("OWNER"(+)='SYS' AND "T"."TABLE_NAME"="OBJECT_NAME"(+))
       filter("T"."OWNER"="OWNER"(+))
```

PUSH_SUBQ

Usage: PUSH_SUBQ([<@SubBlock>])

Description: Instructs the optimizer to evaluate the sub query first when optimizing.

```
HELLODBA.COM>exec sql_explain('select /*+push_subq(@inv)*/* from t_objects o where created > (select
/*+qb_name(inv)*/max(created) from t_users)', 'TYPICAL');

-------------------------------------------------------------------------------------------
| Id  | Operation                   | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |              |  3606 |   352K|   296   (3)| 00:00:03 |
|*  1 |  TABLE ACCESS FULL          | T_OBJECTS    |  3606 |   352K|   295   (3)| 00:00:03 |
|   2 |   SORT AGGREGATE            |              |     1 |     8 |            |          |
|   3 |    INDEX FULL SCAN (MIN/MAX)| T_USERS_IDX1 |     1 |     8 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("CREATED"> (SELECT /*+ PUSH_SUBQ QB_NAME ("INV") */ MAX("CREATED")
              FROM "T_USERS" "T_USERS"))
```

NO_PUSH_SUBQ

Usage: NO_PUSH_SUBQ([<@SubBlock>])

Description: Prevents the optimizer to evaluate the sub query first when optimizing.

```
HELLODBA.COM>exec sql_explain('select /*+no_push_subq(@inv)*/* from t_objects o where created > (select
/*+qb_name(inv)*/max(created) from t_users)', 'TYPICAL');

-------------------------------------------------------------------------------------------
| Id  | Operation                   | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |              | 72116 |  7042K|   299   (4)| 00:00:03 |
|*  1 |  FILTER                     |              |       |       |            |          |
|   2 |   TABLE ACCESS FULL         | T_OBJECTS    | 72116 |  7042K|   298   (4)| 00:00:03 |
|   3 |   SORT AGGREGATE            |              |     1 |     8 |            |          |
```

```
|   4 |    INDEX FULL SCAN (MIN/MAX)| T_USERS_IDX1 |     1 |     8 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------


   1 - filter("CREATED"> (SELECT /*+ NO_PUSH_SUBQ QB_NAME ("INV") */ MAX("CREATED")
              FROM "T_USERS" "T_USERS"))
```

REWRITE

Usage: REWRITE([<@Block>] [<Materialized VIew>])

Description: Instructs the optimizer to re-write the query block to a materialized view.

```
HELLODBA.COM>exec sql_explain('select /*+qb_name(m) rewrite(@m MV_TABLES)*/ t.owner, t.table_name from
t_tables t, t_objects o where t.owner = o.owner and t.table_name = o.object_name and o.object_type =
''TABLE'' and t.tablespace_name is not null and created>:A', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |            |   119 |  4284 |     7   (0)| 00:00:01 |
|*  1 |  MAT_VIEW REWRITE ACCESS FULL| MV_TABLES  |   119 |  4284 |     7   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

NO_REWRITE

Usage: NO_REWRITE([<@Block>])

Description: Prevents the optimizer re-write the query block to a materialized view.

```
HELLODBA.COM>exec sql_explain('select /*+qb_name(m) no_rewrite(@m)*/t.owner, t.table_name from
t_tables t, t_objects o where t.owner = o.owner and t.table_name = o.object_name and o.object_type =
''TABLE'' and t.tablespace_name is not null and created>:A', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------------------
--
| Id  | Operation                           | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
--
|   0 | SELECT STATEMENT                    |                |   126 | 10458 |    64   (5)| 00:00:01 |
|   1 |  NESTED LOOPS                       |                |       |       |            |
|
|   2 |   NESTED LOOPS                      |                |   126 | 10458 |    64   (5)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID      | T_OBJECTS      |   126 |  6048 |    58   (6)| 00:00:01 |
|   4 |     BITMAP CONVERSION TO ROWIDS     |                |       |       |            |          |
|   5 |      BITMAP AND                     |                |       |       |            |
|
|   6 |       BITMAP CONVERSION FROM ROWIDS|                |       |       |            |          |
|   7 |        SORT ORDER BY                |                |       |       |            |
|
```

```
|*  8 |          INDEX RANGE SCAN            | T_OBJECTS_IDX5 |  2523 |       |     3   (0)| 00:00:01 |
|   9 |        BITMAP CONVERSION FROM ROWIDS|                |       |       |            |          |
|  10 |         SORT ORDER BY                |                |       |       |            |
|
|* 11 |          INDEX RANGE SCAN            | T_OBJECTS_IDX7 |  2523 |       |    25   (0)| 00:00:01 |
|* 12 |     INDEX UNIQUE SCAN                | T_TABLES_PK    |     1 |       |     0   (0)| 00:00:01 |
|* 13 |    TABLE ACCESS BY INDEX ROWID       | T_TABLES       |     1 |    35 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
--
```

NO_MULTIMV_REWRITE

Usage: NO_MULTIMV_REWRITE

Description: Not allow the optimizer re-write the query block to multiple materialized views.

```
HELLODBA.COM>create materialized view mv_objects_sys enable query rewrite
  2  as select * from t_objects where owner ='SYS';

Materialized view created.

HELLODBA.COM>create materialized view mv_objects_demo enable query rewrite
  2  as select * from t_objects where owner ='DEMO';

Materialized view created.

HELLODBA.COM>exec sql_explain('select /*+ */* from t_objects where owner in (''SYS'', ''DEMO'')',
'BASIC');

-------------------------------------------------------------
| Id  | Operation                     | Name            |
-------------------------------------------------------------
|   0 | SELECT STATEMENT              |                 |
|   1 |  VIEW                         |                 |
|   2 |   UNION-ALL                   |                 |
|   3 |    MAT_VIEW REWRITE ACCESS FULL| MV_OBJECTS_SYS  |
|   4 |    MAT_VIEW REWRITE ACCESS FULL| MV_OBJECTS_DEMO |
-------------------------------------------------------------

HELLODBA.COM>exec sql_explain('select /*+ NO_MULTIMV_REWRITE */* from t_objects where owner in (''SYS'',
''DEMO'')', 'BASIC');
Plan hash value: 3629755566

---------------------------------------
| Id  | Operation         | Name      |
---------------------------------------
|   0 | SELECT STATEMENT  |           |
|   1 |  TABLE ACCESS FULL| T_OBJECTS |
```

```
---------------------------------------
```

NO_BASETABLE_MULTIMV_REWRITE

Usage: NO_BASETABLE_MULTIMV_REWRITE

Description: No allow the optimizer re-write query to a combine query on the materialized view and its base table.

```
HELLODBA.COM>exec sql_explain('select /*+ REWRITE */* from t_objects where owner in (''SYS'',
''SYSTEM'')', 'BASIC');


-----------------------------------------------------
| Id  | Operation                     | Name          |
-----------------------------------------------------
|   0 | SELECT STATEMENT              |               |
|   1 |  VIEW                         |               |
|   2 |   UNION-ALL                   |               |
|   3 |    MAT_VIEW REWRITE ACCESS FULL| MV_OBJECTS_SYS |
|   4 |    TABLE ACCESS FULL          | T_OBJECTS     |
-----------------------------------------------------


HELLODBA.COM>exec sql_explain('select /*+ REWRITE NO_BASETABLE_MULTIMV_REWRITE */* from t_objects where
owner in (''SYS'', ''SYSTEM'')', 'BASIC');


---------------------------------------
| Id  | Operation         | Name      |
---------------------------------------
|   0 | SELECT STATEMENT  |           |
|   1 |  TABLE ACCESS FULL| T_OBJECTS |
---------------------------------------
```

REWRITE_OR_ERROR

Usage: REWRITE_OR_ERROR

Description: Force the optimizer to re-write query to materialized view, raise error if re-writing failed.

```
HELLODBA.COM>explain plan for select /*+ REWRITE_OR_ERROR */* from t_objects where owner in ('SYS',
'SYSTEM');
explain plan for select /*+ REWRITE_OR_ERROR */* from t_objects where owner in ('SYS', 'SYSTEM')
                    *
ERROR at line 1:
ORA-30393: a query block in the statement did not rewrite
```

SET_TO_JOIN

Usage: SET_TO_JOIN([<@Block>])

Description: Instructs the optimizer to transform data set operations to join.

```
HELLODBA.COM>exec sql_explain('select /*+SET_TO_JOIN(@"SET$1")*/ owner from t_tables intersect select
owner from t_objects', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------------------
```

```
| Id  | Operation                     | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                |    21 |   210 |    16  (75)| 00:00:17 |
|   1 |  HASH UNIQUE                  |                |    21 |   210 |    16  (75)| 00:00:17 |
|* 2  |   HASH JOIN                   |                |   17M|   168M|     6  (34)| 00:00:06 |
|   3 |    INDEX FAST FULL SCAN       | T_TABLES_IDX1  |  2071 | 10355 |     2   (0)| 00:00:03 |
|   4 |    BITMAP CONVERSION TO ROWIDS|                | 47585 |  232K|     2   (0)| 00:00:03 |
|   5 |     BITMAP INDEX FULL SCAN    | T_OBJECTS_IDX4 |       |       |            |          |
-------------------------------------------------------------------------------------------
```

NO_SET_TO_JOIN

Usage: NO_SET_TO_JOIN([<@Block>])

Description: Prevents the optimizer to transform data set operations to join.

```
HELLODBA.COM>exec sql_explain('select /*+NO_SET_TO_JOIN*/ owner from t_tables intersect select owner
from t_objects', 'BASIC OUTLINE');


-----------------------------------------------------
| Id  | Operation                     | Name           |
-----------------------------------------------------
|   0 | SELECT STATEMENT              |                |
|   1 |  INTERSECTION                 |                |
|   2 |   SORT UNIQUE NOSORT          |                |
|   3 |    INDEX FULL SCAN            | T_TABLES_IDX1  |
|   4 |   SORT UNIQUE                 |                |
|   5 |    BITMAP CONVERSION TO ROWIDS|                |
|   6 |     BITMAP INDEX FAST FULL SCAN| T_OBJECTS_IDX4 |
-----------------------------------------------------
```

TRANSFORM_DISTINCT_AGG

Usage: TRANSFORM_DISTINCT_AGG(<@Block>)

Description: Instructs the optimizer to perform Distinct Aggregated Function transformation.

```
HELLODBA.COM>alter session set "_optimizer_distinct_agg_transform"=false;

Session altered.

HELLODBA.COM>exec sql_explain('select /*+qb_name(M) TRANSFORM_DISTINCT_AGG(@M)*/owner,
avg(avg_row_len), count(distinct table_name) from t_tables group by owner', 'TYPICAL OUTLINE');

-------------------------------------------------------------------------------
| Id  | Operation            | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |          |    18 |   900 |    29   (7)| 00:00:01 |
|   1 |  HASH GROUP BY       |          |    18 |   900 |    29   (7)| 00:00:01 |
|   2 |   VIEW               | VW_DAG_0 |  2696 |   131K|    29   (7)| 00:00:01 |
|   3 |    HASH GROUP BY     |          |  2696 | 83576 |    29   (7)| 00:00:01 |
|   4 |     TABLE ACCESS FULL| T_TABLES |  2696 | 83576 |    27   (0)| 00:00:01 |
```

```
-------------------------------------------------------------------------------
```

NO_TRANSFORM_DISTINCT_AGG

Usage: NO_TRANSFORM_DISTINCT_AGG

Description: Prevents the optimizer to perform Distinct Aggregated Function transformation.

```
HELLODBA.COM>exec sql_explain('select /*+NO_TRANSFORM_DISTINCT_AGG*/owner, avg(avg_row_len),
count(distinct table_name) from t_tables group by owner', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |    18 |   558 |    29   (7)| 00:00:01 |
|   1 |  SORT GROUP BY     |          |    18 |   558 |    29   (7)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| T_TABLES |  2696 | 83576 |    27   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

UNNEST

Usage: UNNEST([<@SubBlock>])

Description: Instructs the optimizer to perform query transformation to un-nest the specified query block.

```
HELLODBA.COM>exec sql_explain('select /*+ UNNEST(@INV) */distinct object_name from t_objects o where
object_name not in (select /*+qb_name(inv)*/table_name from t_tables t)', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------------
| Id  | Operation             | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                |  2030 | 85260 |   344   (5)| 00:00:02 |
|   1 |  HASH UNIQUE          |                |  2030 | 85260 |   344   (5)| 00:00:02 |
|*  2 |   HASH JOIN RIGHT ANTI|                | 44305 |  1817K|   335   (2)| 00:00:02 |
|   3 |    INDEX FULL SCAN    | T_TABLES_PK    |  2071 | 37278 |    11   (0)| 00:00:01 |
|   4 |    INDEX FULL SCAN    | T_OBJECTS_IDX1 | 47585 |  1115K|   322   (1)| 00:00:02 |
-------------------------------------------------------------------------------------------
```

NO_UNNEST

Usage: NO_UNNEST([<@SubBlock>])

Description: Prevents the optimizer to perform query transformation to un-nest the specified query block.

```
HELLODBA.COM>exec sql_explain('select /*+ NO_UNNEST(@INV) */distinct object_name from t_objects o where
object_name not in (select /*+qb_name(inv)*/table_name from t_tables t)', 'BASIC PREDICATE');


-----------------------------------------
| Id  | Operation          | Name           |
-----------------------------------------
|   0 | SELECT STATEMENT   |                |
|   1 |  HASH UNIQUE       |                |
|*  2 |   INDEX FULL SCAN  | T_OBJECTS_IDX1 |
|*  3 |    INDEX RANGE SCAN| T_TABLES_PK    |
```

```
-------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter( NOT EXISTS (SELECT /*+ NO_UNNEST QB_NAME ("INV") */ 0
              FROM "T_TABLES" "T" WHERE "TABLE_NAME"=:B1))
   3 - access("TABLE_NAME"=:B1)
```

USE_CONCAT

Usage: USE_CONCAT([<@Block>])

Description: Instructs the optimizer to concat multiple results output from OR conjunct predications.

```
HELLODBA.COM>exec sql_explain('select /*+qb_name(M) USE_CONCAT(@M)*/* from t_objects o where created
= :A or (owner = :B and object_name=:C)', 'TYPICAL OUTLINE');

-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                |    28 |  2800 |     6   (0)| 00:00:01 |
|   1 |  CONCATENATION                |                |       |       |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID| T_OBJECTS      |     1 |   100 |     4   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN           | T_OBJECTS_IDX8 |     1 |       |     3   (0)| 00:00:01 |
|*  4 |   TABLE ACCESS BY INDEX ROWID| T_OBJECTS      |    27 |  2700 |     2   (0)| 00:00:01 |
|*  5 |    INDEX RANGE SCAN           | T_OBJECTS_IDX5 |    27 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

NO_EXPAND

Usage: NO_EXPAND([<@Block>])

Description: Prevents the optimizer to concat multiple results output from OR conjunct predications.

```
HELLODBA.COM>exec sql_explain('select /*+qb_name(M) NO_EXPAND(@M)*/* from t_objects o where created = :A
or (owner = :B and object_name=:C)', 'TYPICAL OUTLINE');

-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                |    27 |  2700 |    11  (10)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID  | T_OBJECTS      |    27 |  2700 |    11  (10)| 00:00:01 |
|   2 |   BITMAP CONVERSION TO ROWIDS |                |       |       |            |          |
|   3 |    BITMAP OR                  |                |       |       |            |          |
|   4 |     BITMAP CONVERSION FROM ROWIDS|             |       |       |            |          |
|*  5 |      INDEX RANGE SCAN         | T_OBJECTS_IDX5 |       |       |     1   (0)| 00:00:01 |
|   6 |     BITMAP CONVERSION FROM ROWIDS|             |       |       |            |          |
|   7 |      SORT ORDER BY            |                |       |       |            |          |
|*  8 |       INDEX RANGE SCAN        | T_OBJECTS_IDX8 |       |       |     3   (0)| 00:00:01 |
```

```
-------------------------------------------------------------------------------------------------
```

USE_TTT_FOR_GSETS

Usage: USE_TTT_FOR_GSETS

Description: Instructs the optimizer to transform Grouping Sets query to insertion and loading on multiple temporary tables.

```
HELLODBA.COM>exec sql_explain('select /*+ USE_TTT_FOR_GSETS qb_name(gv) */owner,
count(constraint_name) cns_cnt from t_constraints c group by cube(owner)','BASIC');


-----------------------------------------------------------------------
| Id  | Operation                    | Name                          |
-----------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                               |
|   1 |  TEMP TABLE TRANSFORMATION   |                               |
|   2 |   LOAD AS SELECT             |                               |
|   3 |    SORT GROUP BY NOSORT ROLLUP|                              |
|   4 |     INDEX FULL SCAN          | T_CONSTRAINTS_IDX3            |
|   5 |   VIEW                       |                               |
|   6 |    TABLE ACCESS FULL         | SYS_TEMP_0FD9D6605_F2042F13   |
-----------------------------------------------------------------------
```

NO_ORDER_ROLLUPS

Usage: NO_ORDER_ROLLUPS

Description: Unknown. Might to instructs the optimizer to perform query transformation for ROLLUP in old version.

OPAQUE_XCANONICAL

Usage: OPAQUE_XCANONICAL

Description: Unknown. Might be used for regular expression.

LIKE_EXPAND

Usage: LIKE_EXPAND

Description: Unknown.

Statistics Data Hints

CARDINALITY

Usage: CARDINALITY([<@Block>] [<Table>] <Cardinality Size>)

Description: Specify the cardinality size of specified object of query block for the optimizer.

```
HELLODBA.COM>exec sql_explain('select /*+ FULL(T) CARDINALITY(@"SEL$1" 10) */* from T_TABLES T, T_USERS
u where t.owner=u.username', 'TYPICAL');


-------------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |    10 |  3530 |    31   (4)| 00:00:01 |
|*  1 |  HASH JOIN         |          |  2696 |   929K|    31   (4)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| T_USERS  |    31 |  3472 |     3   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| T_TABLES |  2696 |   634K|    28   (4)| 00:00:01 |
```

```
-------------------------------------------------------------------------------
```

CPU_COSTING

Usage: CPU_COSTING

Description: Instructs the optimizer to consider CPU cost when optimizing.

```
HELLODBA.COM>alter session set "_optimizer_cost_model"=io;

Session altered.

HELLODBA.COM>exec sql_explain('select /*+CPU_COSTING*/* from t_users, t_tables','TYPICAL OUTLINE');

-------------------------------------------------------------------------------
| Id  | Operation            | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |          | 83576 |    28M|   835   (3)| 00:00:09 |
|   1 |  MERGE JOIN CARTESIAN|          | 83576 |    28M|   835   (3)| 00:00:09 |
|   2 |   TABLE ACCESS FULL  | T_USERS  |    31 |  3472 |     3   (0)| 00:00:01 |
|   3 |   BUFFER SORT        |          |  2696 |   634K|   832   (4)| 00:00:09 |
|   4 |    TABLE ACCESS FULL | T_TABLES |  2696 |   634K|    27   (4)| 00:00:01 |
-------------------------------------------------------------------------------
```

NO_CPU_COSTING

Usage: NO_CPU_COSTING

Description: Prevents the optimizer to consider CPU cost when optimizing.

```
HELLODBA.COM>exec sql_explain('select /*+NO_CPU_COSTING*/* from t_users','TYPICAL OUTLINE');
… …
Note
-----
   - cpu costing is off (consider enabling it)
```

DBMS_STATS

Usage: DBMS_STATS

Description: Used in the internal statement generated by DBMS_STAS when gathering statistic data, it will instruct the optimizer that the statement is for gathering statistics data only, should not be performed an extra process, e.g. auto tuning.

Demo (Below statement was abstracted from the trace file of table statistics data gathering):

```
select /*+ no_parallel(t) no_parallel_index(t) dbms_stats cursor_sharing_exact use_weak_name_resl
dynamic_sampling(0) no_monitoring no_substrb_pad */ count(*)  from "DEMO"."T_OBJECTS" sample block
(  9.1911764706,1) t
```

DYNAMIC_SAMPLING

Usage: DYNAMIC_SAMPLING([<@Block>] [<Table>] <Level>)

Description: Specify the sampling level of specified table of query block, from 0 to 10.

```
HELLODBA.COM>exec dbms_stats.delete_table_stats('DEMO', 'T_OBJECTS_DUMMY');

PL/SQL procedure successfully completed.

HELLODBA.COM>exec sql_explain('select /*+ DYNAMIC_SAMPLING(3) */* from t_objects_dummy o, t_users u
```

```
where o.owner=u.username', 'TYPICAL');
... ...
Note
-----
   - dynamic sampling used for this statement (level=3)
```

DYNAMIC_SAMPLING_EST_CDN

Usage: DYNAMIC_SAMPLING_EST_CDN([<@Block>] <Table>)

Description: Force the optimizer to perform dynamic sampling even if there is statistics data of the specified table.

```
HELLODBA.COM>exec dbms_stats.gather_table_stats('DEMO', 'T_OBJECTS_DUMMY');

PL/SQL procedure successfully completed.

HELLODBA.COM>exec sql_explain('select /*+ DYNAMIC_SAMPLING(3) DYNAMIC_SAMPLING_EST_CDN(0) */* from
t_objects_dummy o, t_users u where o.owner=u.username', 'TYPICAL');
... ...

Note
-----
   - dynamic sampling used for this statement (level=3)
```

GATHER_PLAN_STATISTICS

Usage: GATHER_PLAN_STATISTICS

Description: Instructs the SQL executor to gather the running statistics data of the execution plan.

```
HELLODBA.COM>exec sql_explain('select /*+qb_name(M) GATHER_PLAN_STATISTICS*/* from t_tables t, t_users
u where t.owner=u.username', 'BASIC LAST ALLSTATS', FALSE);

-------------------------------------------------------------------------------------------------
----
| Id |Operation          |Name    |Starts |E-Rows |A-Rows |   A-Time   |Buffers |OMem |1Mem | Used-Mem
|
-------------------------------------------------------------------------------------------------
----
|*  1| HASH JOIN         |        |     1 |  2069 |  2071 |00:00:00.01|    2082 | 767K| 767K| 1161K (0)|
|   2|  TABLE ACCESS FULL|T_USERS |     1 |    43 |    43 |00:00:00.01|       7 |     |     |          |
|   3|  TABLE ACCESS FULL|T_TABLES|     1 |  2071 |  2071 |00:00:00.01|    2075 |     |     |          |
-------------------------------------------------------------------------------------------------
----
```

NO_STATS_GSETS

Usage: NO_STATS_GSETS

Description: Instructs the SQL executor not gather the running statistics of the internal statement generated by the Grouping Sets query.

```
HELLODBA.COM>SELECT /*+gather_plan_statistics*/ owner, object_type, count(object_id) obj_cnt FROM
t_objects o GROUP BY GROUPING SETS (owner, object_type);
... ...
```

```
67 rows selected.

HELLODBA.COM>SELECT /*+gather_plan_statistics NO_STATS_GSETS*/ owner, object_type, count(object_id)
obj_cnt FROM t_objects o GROUP BY GROUPING SETS (owner, object_type);
... ...
67 rows selected.

HELLODBA.COM>select substr(sql_text,1, 50), sharable_mem, executions, buffer_gets from v$sql where
sql_text like 'SELECT /*+gather_plan_statistics%';

SUBSTR(SQL_TEXT,1,50)                                              SHARABLE_MEM EXECUTIONS
BUFFER_GETS
------------------------------------------------------------------ ------------ ----------
-----------
SELECT /*+gather_plan_statistics*/ owner, object_t                      38421          1       4034
SELECT /*+gather_plan_statistics NO_STATS_GSETS*/                       38428          1       1490
```

OPT_ESTIMATE

Usage: OPT_ESTIMATE([<@Block>] <Object Type> <Object> <Data to be adjusted>=<Number>)

Description: Instructs the optimizer to use the adjusted object statistics data. <Object Type> could be QUERY_BLOCK/TABLE/INDEX_FILTER/INDEX_SCAN/INDEX_SKIP_SCAN/JOIN; <Object> could be <@Block>/<Table>/<Index of the Table>/JOIN(<Join Object1> <Join Object2>); <Data to be adjusted> could be ROWS/SCALE_ROWS/MIN/MAX.

OPT_ESTIMATE/COLUMN_STATS/INDEX_STATS/TABLE_STATS hints are normally used as a part of SQL Profiler data.

```
HELLODBA.COM>exec sql_explain('SELECT /*+QB_NAME(M) OPT_ESTIMATE(INDEX_SCAN U T_USERS_PK ROWS=8)*/ *
from t_users u where user_id<:A','TYPICAL');

-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |            |     2 |   224 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| T_USERS    |     2 |   224 |     2   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | T_USERS_PK |     8 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

COLUMN_STATS

Usage: COLUMN_STATS(<Table> <字段> <SCALE|NULL> [<Data1 to be adjusted >=<Number 1> ...])

Description: Instructs the optimizer to use the adjusted column statistics data. SCALE|NULL indicates the optimizer to scale the statistics data or not. <Data to be adjusted> could be length, distinct, nulls, min and max.

```
HELLODBA.COM>exec sql_explain('select /*+qb_name(M) OPT_ESTIMATE(TABLE O scale_rows=721)
COLUMN_STATS(t_objects, OBJECT_NAME, scale, length=666666) */object_name from t_objects o', 'TYPICAL
OUTLINE');
```

```
-------------------------------------------------------------------------------------
| Id  | Operation            | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |                |    51M|  4958M|   185   (2)| 00:00:02 |
|   1 |  INDEX FAST FULL SCAN| T_OBJECTS_IDX8 |    51M|  4958M|   185   (2)| 00:00:02 |
-------------------------------------------------------------------------------------
```

INDEX_STATS

Usage: INDEX_STATS(<Table> <Index> <SCALE|NULL> [<Data1 to be adjusted>=<Number 1> ...])

Description: Instructs the optimizer to use the adjusted index statistics data. SCALE|NULL indicates the optimizer to scale the statistics data or not. <Data to be adjusted> could be blocks, index_rows, keys and clustering_factor.

```
HELLODBA.COM>exec sql_explain('select /*+qb_name(M) index(o (object_id)) OPT_ESTIMATE(TABLE O
scale_rows=721) INDEX_STATS(t_objects, t_objects_pk, scale, clustering_factor=66666) */object_name
from t_objects o where object_id < 10000', 'TYPICAL OUTLINE');


-----------------------------------------------------------------------------------------------
| Id  | Operation                   | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |               |  7010K|   200M|  9023   (1)| 00:01:31 |
|   1 |  TABLE ACCESS BY INDEX ROWID| T_OBJECTS     |  7010K|   200M|  9023   (1)| 00:01:31 |
|*  2 |   INDEX RANGE SCAN          | T_OBJECTS_PK  |  9723 |       |    20   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

TABLE_STATS

Usage: TABLE_STATS(<Table> <SCALE|NULL> [<Data1 to be adjusted >=<Number 1> ...])

Description: Instructs the optimizer to use the adjusted table statistics data. SCALE|NULL indicates the optimizer to scale the statistics data or not. <Data to be adjusted> could be blocks and rows.

```
HELLODBA.COM>exec sql_explain('select /*+qb_name(M) OPT_ESTIMATE(TABLE t_objects scale_rows=0.1)
TABLE_STATS(t_objects,scale, blocks=10 rows=1000) */* from t_objects', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |  4765 |   567K|  1666   (1)| 00:00:07 |
|   1 |  TABLE ACCESS FULL| T_OBJECTS |  4765 |   567K|  1666   (1)| 00:00:07 |
-------------------------------------------------------------------------------
```

Optimizer Hints

NUM_INDEX_KEYS

Usage: NUM_INDEX_KEYS([<@Block>] <Table> <Index> <Index Key Number>)

Description: Instructs the optimizer to how many index keys when performing INLIST ITERATOR operation.

```
HELLODBA.COM>exec sql_explain('SELECT /*+ NUM_INDEX_KEYS(o T_OBJECTS_IDX8 2) */* FROM t_objects o WHERE
owner = :A AND object_name IN (:B1, :B2, :B3)', 'BASIC OUTLINE PREDICATE');
```

```
... ...

Outline Data
-------------

  /*+
      BEGIN_OUTLINE_DATA
      NUM_INDEX_KEYS(@"SEL$1" "O"@"SEL$1" "T_OBJECTS_IDX8" 2)
      INDEX_RS_ASC(@"SEL$1" "O"@"SEL$1" ("T_OBJECTS"."OWNER"
              "T_OBJECTS"."OBJECT_NAME" "T_OBJECTS"."SUBOBJECT_NAME"
              "T_OBJECTS"."OBJECT_ID" "T_OBJECTS"."DATA_OBJECT_ID"
              "T_OBJECTS"."OBJECT_TYPE" "T_OBJECTS"."CREATED" "T_OBJECTS"."STATUS"))
      OUTLINE_LEAF(@"SEL$1")
      OPTIMIZER_FEATURES_ENABLE('10.2.0.4')
      IGNORE_OPTIM_EMBEDDED_HINTS
      END_OUTLINE_DATA
  */

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("OWNER"=:A AND ("OBJECT_NAME"=:B1 OR "OBJECT_NAME"=:B2 OR
              "OBJECT_NAME"=:B3))
```

BIND_AWARE

Usage: BIND_AWARE

Description: Instructs the optimizer to be aware the changes of bind variable value, and can auto tune SQL using extended cursor sharing consequently.

```
HELLODBA.COM>var owner varchar2(30)
HELLODBA.COM>exec :owner := 'DEMO';

PL/SQL procedure successfully completed.

HELLODBA.COM>select /*+bind_aware*/* from t_objects where owner = :owner;
... ...

HELLODBA.COM>select sql_id, sql_text, is_bind_aware from v$sql where sql_text like 'select
/*+bind_aware*/%';

SQL_ID        SQL_TEXT                                                        IS_BIND_AWARE
------------- --------------------------------------------------------------- -------------
6asclcwrbw925 select /*+bind_aware*/* from t_objects where owner = :owner     Y
```

NO_BIND_AWARE

Usage: NO_BIND_AWARE

Description: Prevents the optimizer to be aware the changes of bind variable value, and will not auto tune SQL using extended cursor sharing consequently.

```
HELLODBA.COM>select /*+no_bind_aware*/* from t_objects where owner = :owner;
... ...

HELLODBA.COM>select sql_id, sql_text, is_bind_aware from v$sql where sql_text like 'select
/*+no_bind_aware*/%';

SQL_ID        SQL_TEXT                                                        IS_BIND_AWARE
------------- --------------------------------------------------------------- -------------
586x9p08ag5f3 select /*+no_bind_aware*/* from t_objects where owner = :owner N
```

CURSOR_SHARING_EXACT

Usage: CURSOR_SHARING_EXACT

Description: Instructs the optimizer match the statement exactly when using shared cursor, and it will not convert the explicit data to a bind variable.

```
HELLODBA.COM>show parameter cursor_sharing

NAME                                 TYPE                              VALUE
------------------------------------ --------------------------------- ------------------------------
cursor_sharing                       string                            SIMILAR
HELLODBA.COM>select count(*) from t_objects o where owner = 'DEMO';
... ...

HELLODBA.COM>select sql_text from v$sql where sql_text like 'select count(*) from t_objects%';

SQL_TEXT
------------------------------------------------------------------------------------------------
select count(*) from t_objects o where owner = :"SYS_B_0"


HELLODBA.COM>select count(*) from t_objects o where owner = 'DEMO';
... ...

HELLODBA.COM>select sql_text from v$sql where sql_text like 'select /*+CURSOR_SHARING_EXACT*/count(*)
from t_objects%';

SQL_TEXT
------------------------------------------------------------------------------------------------
----
select /*+CURSOR_SHARING_EXACT*/count(*) from t_objects o where owner = 'DEMO'
```

DML_UPDATE

Usage: DML_UPDATE

Description: Used in the internal statement (UPD_JOININDEX) when updating Bitmap Join Index.

FBTSCAN

Usage: FBTSCAN

Description: Instructs the optimizer to query the Flashback Table instead of the base table when 用 performing Flashbask query. This hint may affect the query result.

```
HELLODBA.COM>var scn number
HELLODBA.COM>begin
  2    select dbms_flashback.get_system_change_number into :scn from dual;
  3  end;
  4  /

PL/SQL procedure successfully completed.

HELLODBA.COM>insert into t_tables(owner, table_name) values('NONE', 'NOTHING');

1 row created.

HELLODBA.COM>commit;

Commit complete.

HELLODBA.COM>SELECT /*+ QB_NAME(V) FBTSCAN FULL(S) */ count(SYS_FBT_INSDEL) FROM T_TABLES as of SCN :SCN
S;

COUNT(SYS_FBT_INSDEL)
---------------------
                   67

HELLODBA.COM>SELECT /*+ QB_NAME(V) FULL(S) */ count(SYS_FBT_INSDEL) FROM T_TABLES as of SCN :SCN S;

COUNT(SYS_FBT_INSDEL)
---------------------
                    0
```

ALL_ROWS

Usage: ALL_ROWS

Description: Instructs the optimizer to optimize the statement with a goal of best throughput

```
HELLODBA.COM>exec sql_explain('select /*+ all_rows */* from t_objects o where rownum <= 10', 'TYPICAL
OUTLINE');

-------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |    10 |  1220 |  1670   (1)| 00:00:07 |
|*  1 |  COUNT STOPKEY     |           |       |       |            |          |
|   2 |   TABLE ACCESS FULL| T_OBJECTS | 47585 |  5669K|  1670   (1)| 00:00:07 |
```

```
-------------------------------------------------------------------------------
```

FIRST_ROWS

Usage: FIRST_ROWS([<Row Number>])

Description: instructs the optimizer to optimize a statement block with a goal o fast response to return the first n rows.

```
HELLODBA.COM>exec sql_explain('select /*+ first_rows(10) */* from t_objects o', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------
| Id  | Operation         | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |           |    10 |  1220 |     5   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| T_OBJECTS |    10 |  1220 |     5   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

RULE

Usage: RULE

Description: Instructs the optimizer to choose the best execution based on rules.

```
HELLODBA.COM>exec sql_explain('select /*+ rule */* from t_objects o where object_id > 0', 'TYPICAL
OUTLINE');
… …


Outline Data
-------------


  /*+
      BEGIN_OUTLINE_DATA
      … …
      RBO_OUTLINE
      … …
      END_OUTLINE_DATA
  */
```

CHOOSE

Usage: CHOOSE

Description: Instructs the optimizer to choose the optimizing mode according to current environment.

```
HELLODBA.COM>exec dbms_stats.delete_table_stats('DEMO', 'T_OBJECTS_DUMMY');


PL/SQL procedure successfully completed.


HELLODBA.COM>exec sql_explain('select /*+ choose */* from t_objects_dummy o where rownum <= 10', 'BASIC
NOTE');
… …
Note
-----
   - rule based optimizer used (consider using cbo)
```

ORDERED_PREDICATES

Usage: ORDERED_PREDICATES

Description: Instructs the optimizer to analyze the non-index accessing predictions in a special order. If two predications matched the same rule, they will be analyzed in the order of their appearance in the WHERE clause.

1. Predication without user-defined function/type and sub query.
2. Predication with user-defined function/type as well as related statistics data
3. Predication without user-defined function/type, without related statistics data.
4. Predication not appearing in WHERE clause, which may be generated by query transformation.
5. Predication with sub query.

Demo (Compare the cost of below two execution plans)

```
HELLODBA.COM>exec sql_explain('select /*+ full(o) */* from t_objects o, t_tables t where o.owner =
t.owner and o.object_name = t.table_name and exists (select 1 from t_tables t where
o.object_name=t.table_name) and o.owner in (select username from t_users) and to_char(object_id)=:B and
object_type = :A', 'TYPICAL');

-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |             |     1 |   372 |   297   (3)| 00:00:03 |
|   1 |  NESTED LOOPS                |             |       |       |            |          |
|   2 |   NESTED LOOPS               |             |     1 |   372 |   297   (3)| 00:00:03 |
|   3 |    NESTED LOOPS SEMI         |             |     1 |   131 |   296   (3)| 00:00:03 |
|   4 |     NESTED LOOPS             |             |     1 |   110 |   295   (3)| 00:00:03 |
|*  5 |      TABLE ACCESS FULL       | T_OBJECTS   |     1 |   100 |   295   (3)| 00:00:03 |
|*  6 |      INDEX UNIQUE SCAN       | T_USERS_UK  |     1 |    10 |     0   (0)| 00:00:01 |
|*  7 |     INDEX RANGE SCAN         | T_TABLES_PK |  1097 | 23037 |     1   (0)| 00:00:01 |
|*  8 |    INDEX UNIQUE SCAN         | T_TABLES_PK |     1 |       |     0   (0)| 00:00:01 |
|   9 |   TABLE ACCESS BY INDEX ROWID| T_TABLES    |     1 |   241 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

HELLODBA.COM>exec sql_explain('select /*+ full(o) ORDERED_PREDICATES */* from t_objects o, t_tables t
where o.owner = t.owner and o.object_name = t.table_name and exists (select 1 from t_tables t where
o.object_name=t.table_name) and o.owner in (select username from t_users) and to_char(object_id)=:B and
object_type = :A', 'TYPICAL');

-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |             |     1 |   372 |   298   (3)| 00:00:03 |
|   1 |  NESTED LOOPS                |             |       |       |            |          |
|   2 |   NESTED LOOPS               |             |     1 |   372 |   298   (3)| 00:00:03 |
|   3 |    NESTED LOOPS SEMI         |             |     1 |   131 |   297   (3)| 00:00:03 |
```

```
|   4 |     NESTED LOOPS              |             |     1 |   110 |   296   (3)| 00:00:03 |
|*  5 |      TABLE ACCESS FULL        | T_OBJECTS   |     1 |   100 |   296   (3)| 00:00:03 |
|*  6 |       INDEX UNIQUE SCAN       | T_USERS_UK  |     1 |    10 |     0   (0)| 00:00:01 |
|*  7 |     INDEX RANGE SCAN          | T_TABLES_PK | 1097 | 23037 |     1   (0)| 00:00:01 |
|*  8 |    INDEX UNIQUE SCAN          | T_TABLES_PK |     1 |       |     0   (0)| 00:00:01 |
|   9 |   TABLE ACCESS BY INDEX ROWID| T_TABLES    |     1 |   241 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

SKIP_EXT_OPTIMIZER

Usage: SKIP_EXT_OPTIMIZER

Description: Instructs the optimizer skip the extended optimizer.

SKIP_UNQ_UNUSABLE_IDX

Usage: SKIP_UNQ_UNUSABLE_IDX([<@Block>] <Table> [<Index1> ...]) or SKIP_UNQ_UNUSABLE_IDX([<@Block>] <Table> [(<Index1 Columns>) ...])

Description: Instructs the optimizer to skip the UNUSABLE indexes.

```
HELLODBA.COM>alter index t_tables_pk unusable;

Index altered.

HELLODBA.COM>select /*+ index(t t_tables_pk) */count(1) from t_tables t;
select /*+ index(t t_tables_pk) */count(1) from t_tables t
*
ERROR at line 1:
ORA-01502: index 'DEMO.T_TABLES_PK' or partition of such index is in unusable state

HELLODBA.COM>select /*+ SKIP_UNQ_UNUSABLE_IDX(t) index(t t_tables_pk) */count(1) from t_tables t;

  COUNT(1)
----------
      2696
```

Run Time Hints

APPEND

Usage: APPEND

Description: Instructs SQL executor to insert data in appending mode.

```
HELLODBA.COM>exec sql_explain('insert /*+append*/ into t_objects_bak select * from t_objects', 'BASIC OUTLINE')

-----------------------------------------
| Id  | Operation          | Name          |
-----------------------------------------
|   0 | INSERT STATEMENT   |               |
|   1 |  LOAD AS SELECT    | T_OBJECTS_BAK |
|   2 |   TABLE ACCESS FULL| T_OBJECTS     |
-----------------------------------------
```

APPEND_VALUES

Usage: APPEND_VALUES

Description: Instructs SQL executor to insert data in appending mode. It just works in the INSERT … VALUES statement.

```
HELLODBA.COM>exec sql_explain('insert /*+append_values*/ into t_objects(object_id, owner, object_name, status) values(:1, :2, :3, :4)', 'BASIC OUTLINE')


---------------------------------------
| Id  | Operation        | Name      |
---------------------------------------
|   0 | INSERT STATEMENT |           |
|   1 |  LOAD AS SELECT  | T_OBJECTS |
|   2 |   BULK BINDS GET |           |
---------------------------------------
```

NOAPPEND

Usage: NOAPPEND

Description: Instructs SQL executor to insert data in appending mode.

```
HELLODBA.COM>exec sql_explain('insert /*+noappend*/ into t_objects_bak select * from t_objects', 'BASIC OUTLINE')


-----------------------------------------------
| Id  | Operation                | Name         |
-----------------------------------------------
|   0 | INSERT STATEMENT         |              |
|   1 |  LOAD TABLE CONVENTIONAL | T_OBJECTS_BAK |
|   2 |   TABLE ACCESS FULL      | T_OBJECTS    |
-----------------------------------------------
```

NLJ_BATCHING

Usage: NLJ_BATCHING

Description: Instructs the optimizer to use nested-loop join to access table in batch.

```
HELLODBA.COM>exec sql_explain('select /*+ use_nl(o t) index(o T_OBJECTS_M_IDX8) index(t T_TABLES_IDX3) LEADING(t) NLJ_BATCHING(o)*/count(LIO) from T_OBJECTS_M 0, T_TABLES t where o.owner=t.owner and o.object_name=t.table_name and t.tablespace_name=:A', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                  |     1 |    68 |   914   (1)| 00:00:10 |
|   1 |  SORT AGGREGATE               |                  |     1 |    68 |            |          |
|   2 |   NESTED LOOPS                |                  |       |       |            |          |
|   3 |    NESTED LOOPS               |                  |  7985 |   530K|   914   (1)| 00:00:10 |
|   4 |     TABLE ACCESS BY INDEX ROWID| T_TABLES         |   299 | 10465 |    16   (0)| 00:00:01 |
|*  5 |      INDEX RANGE SCAN         | T_TABLES_IDX3    |   299 |       |     1   (0)| 00:00:01 |
|*  6 |     INDEX RANGE SCAN          | T_OBJECTS_M_IDX8 |     1 |       |     2   (0)| 00:00:01 |
```

```
|   7 |    TABLE ACCESS BY INDEX ROWID | T_OBJECTS_M      |    27 |   891 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

NO_NLJ_BATCHING

Usage: NO_NLJ_BATCHING([<@Block>] <Table>)

Description: Prevents the optimizer to use nested-loop join to access table in batch.

```
HELLODBA.COM>exec sql_explain('select /*+ use_nl(o t) index(o T_OBJECTS_M_IDX8) index(t T_TABLES_IDX3)
LEADING(t) NO_NLJ_BATCHING(o)*/count(LIO) from T_OBJECTS_M O, T_TABLES t where o.owner=t.owner and
o.object_name=t.table_name and t.tablespace_name=:A', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                  |     1 |    68 |   914   (1)| 00:00:10 |
|   1 |  SORT AGGREGATE               |                  |     1 |    68 |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID | T_OBJECTS_M      |    27 |   891 |     3   (0)| 00:00:01 |
|   3 |    NESTED LOOPS               |                  |  7985 |   530K|   914   (1)| 00:00:10 |
|   4 |     TABLE ACCESS BY INDEX ROWID| T_TABLES         |   299 | 10465 |    16   (0)| 00:00:01 |
|*  5 |      INDEX RANGE SCAN         | T_TABLES_IDX3    |   299 |       |     1   (0)| 00:00:01 |
|*  6 |     INDEX RANGE SCAN          | T_OBJECTS_M_IDX8 |     1 |       |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

NLJ_PREFETCH

Usage: NLJ_PREFETCH([<@Block>] <Table>)

Description: Instructs the optimizer to use nested-loop join to pre-fetch the table.

```
HELLODBA.COM>exec sql_explain('select /*+ use_nl(o t) index(o T_OBJECTS_M_IDX8) index(t T_TABLES_IDX3)
LEADING(t) NLJ_PREFETCH(o)*/count(LIO) from T_OBJECTS_M O, T_TABLES t where o.owner=t.owner and
o.object_name=t.table_name and t.tablespace_name=:A', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                  |     1 |    68 |   914   (1)| 00:00:10 |
|   1 |  SORT AGGREGATE               |                  |     1 |    68 |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID | T_OBJECTS_M      |    27 |   891 |     3   (0)| 00:00:01 |
|   3 |    NESTED LOOPS               |                  |  7985 |   530K|   914   (1)| 00:00:10 |
|   4 |     TABLE ACCESS BY INDEX ROWID| T_TABLES         |   299 | 10465 |    16   (0)| 00:00:01 |
|*  5 |      INDEX RANGE SCAN         | T_TABLES_IDX3    |   299 |       |     1   (0)| 00:00:01 |
|*  6 |     INDEX RANGE SCAN          | T_OBJECTS_M_IDX8 |     1 |       |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

NO_NLJ_PREFETCH

Usage: NO_NLJ_PREFETCH([<@Block>] <Table>)

Description: Prevents the optimizer to use nested-loop join to pre-fetch the table.

```
HELLODBA.COM>exec sql_explain('select /*+ use_nl(o t) index(o T_OBJECTS_M_IDX8) index(t T_TABLES_IDX3)
LEADING(t) NO_NLJ_PREFETCH(o)*/count(LIO) from T_OBJECTS_M O, T_TABLES t where o.owner=t.owner and
o.object_name=t.table_name and t.tablespace_name=:A', 'TYPICAL OUTLINE');
```

```
-------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                  |     1 |    68 |   914   (1)| 00:00:10 |
|   1 |  SORT AGGREGATE               |                  |     1 |    68 |            |          |
|   2 |   NESTED LOOPS                |                  |       |       |            |          |
|   3 |    NESTED LOOPS               |                  |  7985 |   530K|   914   (1)| 00:00:10 |
|   4 |     TABLE ACCESS BY INDEX ROWID| T_TABLES         |   299 | 10465 |    16   (0)| 00:00:01 |
|*  5 |      INDEX RANGE SCAN         | T_TABLES_IDX3    |   299 |       |     1   (0)| 00:00:01 |
|*  6 |     INDEX RANGE SCAN          | T_OBJECTS_M_IDX8 |     1 |       |     2   (0)| 00:00:01 |
|   7 |    TABLE ACCESS BY INDEX ROWID | T_OBJECTS_M      |    27 |   891 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

CACHE

Usage: CACHE([<@Block>] <Table>)

Description: Instructs Oracle to put the read data block to the end of MRU in LRU chain.

```
HELLODBA.COM>select name, value from v$mystat m, v$statname n where m.statistic# = n.statistic# and name
like 'table scans% tables)';

NAME                                 VALUE
-------------------------------- ----------
table scans (short tables)               37
table scans (long tables)                 0

HELLODBA.COM>select /*+full(t) cache(t)*/count(*) from t_objects t;
... ...

HELLODBA.COM>select name, value from v$mystat m, v$statname n where m.statistic# = n.statistic# and name
like 'table scans% tables)';

NAME                                 VALUE
-------------------------------- ----------
table scans (short tables)               37
table scans (long tables)                 1
```

NOCACHE

Usage: NOCACHE([<@Block>] <Table>)

Description: Instructs Oracle to put the read data block to the end of LRU in LRU chain.

```
HELLODBA.COM>alter system flush buffer_cache;

System altered.

HELLODBA.COM>select name, value from v$mystat m, v$statname n where m.statistic# = n.statistic# and name
like 'table scans% tables)';
```

```
NAME                                 VALUE
---------------------------- ----------
table scans (short tables)              35
table scans (long tables)                1

HELLODBA.COM>select /*+full(t) nocache(t)*/count(*) from t_tables t;
... ...

HELLODBA.COM>select name, value from v$mystat m, v$statname n where m.statistic# = n.statistic# and name
like 'table scans% tables)';

NAME                                 VALUE
---------------------------- ----------
table scans (short tables)              36
table scans (long tables)                1
```

CACHE_TEMP_TABLE

Usage: CACHE_TEMP_TABLE([<@Block>] <Table>)

Description: Instructs Oracle to put the read temp block to the end of MRU in LRU chain. This hint 指 normally be used in internal statement.

```
HELLODBA.COM>begin
  2     sql_explain('WITH
  3                     A AS (SELECT /*+qb_name(AV)*/''x'' M FROM DUAL),
  4                     B AS (SELECT /*+qb_name(BV)*/DISTINCT LEVEL L FROM A CONNECT BY LEVEL <= 10),
  5                     C AS (SELECT /*+qb_name(CV)*/LPAD(LEVEL-1,(SELECT COUNT(*) FROM B),''0'') N
FROM DUAL CONNECT BY LEVEL <= POWER(10,(SELECT COUNT(*) FROM B)))
  6                     SELECT /*+qb_name(M)*/* FROM A, B, C where rownum>=1', 'BASIC PREDICATE');
  7  end;
  8  /

... ...

Predicate Information (identified by operation id):
---------------------------------------------------


   7 - filter(LEVEL<=10)
  10 - filter(ROWNUM>=1)
  18 - filter(LEVEL<=POWER(10, (SELECT /*+ */ COUNT(*) FROM  (SELECT
             /*+ CACHE_TEMP_TABLE ("T1") */ "C0" "L" FROM
             "SYS"."SYS_TEMP_0FD9D663A_F1E95AE7" "T1") "B")))
```

NO_LOAD

Usage: NO_LOAD

Description: Not allow the data to be loaded directly.

```
HELLODBA.COM>alter session enable parallel dml;
```

```
Session altered.

HELLODBA.COM>alter session set "_disable_parallel_conventional_load"=true;

Session altered.

HELLODBA.COM>exec sql_explain('insert /*+ no_load(d) parallel(d 2) */ into t_objects_dummy2 d select
/*+parallel(o 2)*/* from t_objects o', 'BASIC');

-----------------------------------------------------
| Id  | Operation                | Name             |
-----------------------------------------------------
|   0 | INSERT STATEMENT         |                  |
|   1 |  LOAD TABLE CONVENTIONAL | T_OBJECTS_DUMMY2 |
|   2 |   PX COORDINATOR         |                  |
|   3 |    PX SEND QC (RANDOM)   | :TQ10000         |
|   4 |     PX BLOCK ITERATOR    |                  |
|   5 |      TABLE ACCESS FULL   | T_OBJECTS        |
-----------------------------------------------------
```

NO_SUBSTRB_PAD

Usage: NO_SUBSTRB_PAD

Description: No extra byte be padded when perform SUBSTRB on multiple-bytes character string.

```
HELLODBA.COM>select /*+  */substrb(chistr,1,2) str, lengthb(substrb(chistr,1,2)) len from (select '
多字节字符串' chistrfrom dual ) v;

STR         LEN
---- ----------
              2

HELLODBA.COM>select /*+ NO_SUBSTRB_PAD */substrb(chistr,1,2) str, lengthb(substrb(chistr,1,2)) len
from (select '多字节字符串' chistr from dual ) v;

STR        LEN
--- ----------
```

RESULT_CACHE

Usage: RESULT_CACHE

Description: Instructs Oracle cache the result to be used by following query.

```
HELLODBA.COM>show parameter result_cache

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
client_result_cache_lag              big integer 3000
client_result_cache_size             big integer 0
```

```
result_cache_max_result              integer     5
result_cache_max_size                big integer 1M
result_cache_mode                    string      MANUAL
result_cache_remote_expiration       integer     0

HELLODBA.COM>exec sql_explain('select * from t_tables t, (select /*+ result_cache */* from t_objects
o where object_type = :A) v where v.owner = t.owner and v.object_name = t.table_name', 'BASIC PREDICATE');

-------------------------------------------------------------------------
| Id  | Operation                     | Name                        |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                             |
|*  1 |  HASH JOIN                    |                             |
|   2 |   VIEW                        |                             |
|   3 |    RESULT CACHE               | b6zzbhhgh4knw21npw13q564wn |
|   4 |     TABLE ACCESS BY INDEX ROWID| T_OBJECTS                   |
|*  5 |      INDEX RANGE SCAN         | T_OBJECTS_IDX7              |
|   6 |   TABLE ACCESS FULL           | T_TABLES                    |
-------------------------------------------------------------------------

Result Cache Information (identified by operation id):
------------------------------------------------------

   3 - column-count=16; dependencies=(DEMO.T_OBJECTS); attributes=(ordered); parameters=(nls, :A);
name="select /*+ result_cache */* from t_objects o where object_type = :A"
```

NO_RESULT_CACHE

Usage: NO_RESULT_CACHE

Description: Prevents Oracle cache the result to be used by following query.

```
HELLODBA.COM>alter session set result_cache_mode=force;

Session altered.

HELLODBA.COM>exec sql_explain('select /*+ no_result_cache */* from t_tables t', 'TYPICAL');

-------------------------------------------------------------------------------
| Id  | Operation         | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |          |  2696 |   634K|    28   (4)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| T_TABLES |  2696 |   634K|    28   (4)| 00:00:01 |
-------------------------------------------------------------------------------
```

SYS_DL_CURSOR

Usage: SYS_DL_CURSOR

Description: This hint is added in the internal statement generated by SQL*Loader when performing direct loading (Direct=TRUE).

Demo (Below statement is captured from library cache after run SQL*Loader to load data directly):

```
HELLODBA.COM>select sql_text, module from v$sql where sql_text like 'INSERT /*+ SYS_DL_CURSOR */%';

SQL_TEXT
MODULE
--------------------------------------------------------------------------------
------
INSERT /*+ SYS_DL_CURSOR */ INTO "DEMO"."T_TABLES_LD" ("OWNER","TABLE_NAME") VALUES (NULL,NULL)  SQL
Loader Direct Path Load
```

TRACING

Usage: TRACING(verify <Number> strip <Number>)

Description: Unknown. It might be used to trace the ASM stripping. This hint will add an extra operation, MONITORING STRIP, in the execution plan.

```
HELLODBA.COM>exec sql_explain('select /*+qb_name(M) full(t_objects) TRACING(verify 1, strip 1) */* from
t_objects', 'TYPICAL OUTLINE');

-------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           | 72116 |  7042K|   297   (3)| 00:00:03 |
|   1 |  MONITORING STRIP  |           | 72116 |  7042K|   297   (3)| 00:00:03 |
|   2 |   TABLE ACCESS FULL| T_OBJECTS | 72116 |  7042K|   297   (3)| 00:00:03 |
-------------------------------------------------------------------------------
```

Data Manipulate Hints

CHANGE_DUPKEY_ERROR_INDEX

Usage: CHANGE_DUPKEY_ERROR_INDEX(<Table>, <Index>) or CHANGE_DUPKEY_ERROR_INDEX(<Table>, <Indexed Columns>)

Description: When unique constraint is violated, it will raise ORA-38911 instead of ORA-00001.

```
HELLODBA.COM>insert into t_tables(owner, table_name) values ('SYS', 'TAB$');
insert into t_tables(owner, table_name) values ('SYS', 'TAB$')
*
ERROR at line 1:
ORA-00001: unique constraint (DEMO.T_TABLES_PK) violated


HELLODBA.COM>insert /*+CHANGE_DUPKEY_ERROR_INDEX(t_tables t_tables_pk)*/into t_tables(owner,
table_name) values ('SYS','TAB$');
insert /*+CHANGE_DUPKEY_ERROR_INDEX(t_tables t_tables_pk)*/into t_tables(owner, table_name) values
('SYS', 'TAB$')
*
ERROR at line 1:
ORA-38911: unique constraint (DEMO.T_TABLES_PK) violated
```

IGNORE_ROW_ON_DUPKEY_INDEX

Usage: IGNORE_ROW_ON_DUPKEY_INDEX(<Table> <Unique Index>) or IGNORE_ROW_ON_DUPKEY_INDEX(<Table> <Unique INdex columns>)

Description: The duplicated data will be ignored when insert data to a table with unique constraint.

```
HELLODBA.COM>create table t(a number primary key);

Table created.

HELLODBA.COM>insert into t values(2);

1 row created.

HELLODBA.COM>commit;

Commit complete.

HELLODBA.COM>insert /*+IGNORE_ROW_ON_DUPKEY_INDEX(t (a))*/into t select level from dual connect by level
<= 3;

2 rows created.

HELLODBA.COM>commit;

Commit complete.

HELLODBA.COM>select * from t;

         A
----------
         1
         2
         3
```

RETRY_ON_ROW_CHANGE

Usage: RETRY_ON_ROW_CHANGE

Description: When the transaction of UPDATE or DELETE is committing, if the cached data changed, the transaction will be rebounded.

```
HELLODBA.COM>-- 会话 1
HELLODBA.COM>create table t(a number primary key);

Table created.

HELLODBA.COM>insert into t values(1);
```

```
1 row created.

HELLODBA.COM>insert into t values(2);

1 row created.

HELLODBA.COM>commit;

Commit complete.

HELLODBA.COM>exec dbms_stats.gather_table_stats('DEMO', 'T');

PL/SQL procedure successfully completed.

HELLODBA.COM>select name, value from v$mystat m, v$statname n where m.statistic#=n.statistic# and name
in ('consistent gets', 'no work - consistent read gets', 'transaction rollbacks');

NAME                                                             VALUE
---------------------------------------------------------------- ----------
consistent gets                                                    1684
no work - consistent read gets                                      549
transaction rollbacks                                                 0

HELLODBA.COM>update /*+RETRY_ON_ROW_CHANGE*/t set a=a+10 where a>=2;

1 rows updated.

HELLODBA.COM>-- 会话 2
HELLODBA.COM>update t set a=a+100 where a=1;

1 row updated.

HELLODBA.COM>commit;

Commit complete.

HELLODBA.COM>-- 会话 1
HELLODBA.COM>commit;

Commit complete.

HELLODBA.COM>select name, value from v$mystat m, v$statname n where m.statistic#=n.statistic# and name
in ('consistent gets', 'no work - consistent read gets', 'transaction rollbacks');
```

```
NAME                                                             VALUE
---------------------------------------------------------------- ----------
consistent gets                                                    1686
no work - consistent read gets                                      550
transaction rollbacks                                                 0
```

Without the hint, we get below result.

```
HELLODBA.COM>select name, value from v$mystat m, v$statname n where m.statistic#=n.statistic# and name
in ('consistent gets', 'no work - consistent read gets', 'transaction rollbacks');

NAME                                                             VALUE
---------------------------------------------------------------- ----------
consistent gets                                                    1708
no work - consistent read gets                                      555
transaction rollbacks                                                 4
... ...

HELLODBA.COM>select name, value from v$mystat m, v$statname n where m.statistic#=n.statistic# and name
in ('consistent gets', 'no work - consistent read gets', 'transaction rollbacks');

NAME                                                             VALUE
---------------------------------------------------------------- ----------
consistent gets                                                    1709
no work - consistent read gets                                      555
transaction rollbacks                                                 4
```

Hierarchy Query Hints

CONNECT_BY_CB_WHR_ONLY

Usage: CONNECT_BY_CB_WHR_ONLY([<@Block>])

Description: Unknown. It might instruct the optimizer to consider the WHERE clause only when performing CONNECT BY operation. There is an optimizer parameter, _optimizer_connect_by_cb_whr_only, to control this feature.

NO_CONNECT_BY_CB_WHR_ONLY

Usage: NO_CONNECT_BY_CB_WHR_ONLY([<@Block>])

Description: Unknown. It might prevent the optimizer to consider the WHERE clause only when performing CONNECT BY operation.

CONNECT_BY_COMBINE_SW

Usage: CONNECT_BY_COMBINE_SW([<@Block>])

Description: Instructs the optimizer to consider the predication in START WITH when performing CONNECT BY operation.

```
HELLODBA.COM>exec sql_explain('select /*+NO_CONNECT_BY_FILTERING(@SEL$1)
CONNECT_BY_COMBINE_SW(@SEL$1)*/owner, table_name, level from t_constraints connect by prior
```

```
r_owner=owner and prior r_constraint_name = constraint_name start with owner=:A','BASIC OUTLINE
PREDICATE');

-----------------------------------------------------------
| Id  | Operation                          | Name          |
-----------------------------------------------------------
|   0 | SELECT STATEMENT                   |               |
|*  1 |  CONNECT BY NO FILTERING WITH START-WITH|           |
|   2 |   TABLE ACCESS FULL                | T_CONSTRAINTS |
-----------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("OWNER"=PRIOR "R_OWNER" AND "CONSTRAINT_NAME"=PRIOR
              "R_CONSTRAINT_NAME")
       filter("OWNER"=:A)
```

NO_CONNECT_BY_COMBINE_SW

Usage: NO_CONNECT_BY_COMBINE_SW([<@Block>])

Description: Prevents the optimizer to consider the predication in START WITH when performing CONNECT BY operation.

```
HELLODBA.COM>exec sql_explain('select /*+NO_CONNECT_BY_FILTERING(@"SEL$1")
NO_CONNECT_BY_COMBINE_SW(@"SEL$1")*/owner, table_name, level from t_constraints connect by prior
r_owner=owner and prior r_constraint_name = constraint_name start with owner=:A','BASIC OUTLINE
PREDICATE');

-----------------------------------------------------
| Id  | Operation                    | Name          |
-----------------------------------------------------
|   0 | SELECT STATEMENT             |               |
|*  1 |  CONNECT BY WITHOUT FILTERING|               |
|*  2 |   TABLE ACCESS FULL          | T_CONSTRAINTS |
|   3 |   TABLE ACCESS FULL          | T_CONSTRAINTS |
-----------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("OWNER"=PRIOR "R_OWNER" AND "CONSTRAINT_NAME"=PRIOR
              "R_CONSTRAINT_NAME")
   2 - filter("OWNER"=:A)
```

CONNECT_BY_COST_BASED

Usage: CONNECT_BY_COST_BASED([<@Block>])

Description: Instructs the optimizer to transform CONNECT BY based on cost.

Demo (10.2.0.4):

```
HELLODBA.COM>exec sql_explain('select /*+QB_NAME(M) CONNECT_BY_COST_BASED(@M)*/owner, table_name,
level from t_constraints where constraint_type=:B connect by prior r_owner=owner and prior
r_constraint_name = constraint_name start with owner=:A and level > 2','BASIC OUTLINE PREDICATE');
... ...

Outline Data
-------------

  /*+
      BEGIN_OUTLINE_DATA
      ... ...
      OUTLINE(@"M")
      CONNECT_BY_COST_BASED(@"M")
      OUTLINE(@"SEL$C20E7289")
      CONNECT_BY_FILTERING(@"SEL$C20E7289")
      ... ...
      IGNORE_OPTIM_EMBEDDED_HINTS
      END_OUTLINE_DATA
  */
```

NO_CONNECT_BY_COST_BASED

Usage: NO_CONNECT_BY_COST_BASED([<@Block>])

Description: Instructs the optimizer to transform CONNECT BY based on cost.

Demo (10.2.0.4):

```
HELLODBA.COM>exec sql_explain('select /*+QB_NAME(M) NO_CONNECT_BY_COST_BASED(@M)*/owner, table_name,
level from t_constraints where constraint_type=:B connect by prior r_owner=owner and prior
r_constraint_name = constraint_name start with owner=:A and level > 2','BASIC OUTLINE PREDICATE');
BEGIN sql_explain('select /*+QB_NAME(M) NO_CONNECT_BY_COST_BASED(@M)*/owner, table_name, level from
t_constraints where constraint_type=:B connect by prior r_owner=owner and prior r_constraint_name =
constraint_name start with owner=:A and level > 2','BASIC OUTLINE PREDICATE'); END;

*
ERROR at line 1:
ORA-01788: CONNECT BY clause required in this query block
ORA-06512: at "SYS.SQL_EXPLAIN", line 25
ORA-06512: at line 1
```

CONNECT_BY_ELIM_DUPS

Usage: CONNECT_BY_ELIM_DUPS([<@Block>])

Description: Instructs the SQL executor to eliminate duplicated data when perform CONNECT BY operation.

```
HELLODBA.COM>exec sql_explain('SELECT /*+qb_name(BV) CONNECT_BY_ELIM_DUPS(@"BV")*/DISTINCT LEVEL FROM
DUAL CONNECT BY LEVEL <= 10', 'TYPICAL OUTLINE');

-------------------------------------------------------------------------------------
| Id  | Operation                                | Name | Rows  | Cost (%CPU)| Time     |
```

```
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                      |      |     1 |     3  (34)| 00:00:01 |
|   1 |  HASH UNIQUE                          |      |     1 |     3  (34)| 00:00:01 |
|*  2 |   CONNECT BY WITHOUT FILTERING (UNIQUE)|      |       |            |          |
|   3 |    FAST DUAL                          |      |     1 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

NO_CONNECT_BY_ELIM_DUPS

Usage: NO_CONNECT_BY_ELIM_DUPS([<@Block>])

Description: Prevents the SQL executor to eliminate duplicated data when perform CONNECT BY operation.

```
HELLODBA.COM>exec sql_explain('SELECT /*+qb_name(BV) NO_CONNECT_BY_ELIM_DUPS(@"BV")*/DISTINCT LEVEL
FROM DUAL CONNECT BY LEVEL <= 10', 'TYPICAL OUTLINE');


-------------------------------------------------------------------------------
| Id  | Operation                     | Name | Rows  | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |      |     1 |     3  (34)| 00:00:01 |
|   1 |  HASH UNIQUE                  |      |     1 |     3  (34)| 00:00:01 |
|*  2 |   CONNECT BY WITHOUT FILTERING|      |       |            |          |
|   3 |    FAST DUAL                  |      |     1 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

CONNECT_BY_FILTERING

Usage: CONNECT_BY_FILTERING([<@Block>])

Description: Instructs the SQL executor to filter data when perform CONNECT BY operation.

```
HELLODBA.COM>exec sql_explain('select /*+CONNECT_BY_FILTERING(@"SEL$1")*/owner, table_name, level
from t_constraints c start with constraint_name in (select index_name from t_indexes t where t.owner
= c.owner ) connect by prior r_owner=owner and prior r_constraint_name = constraint_name','BASIC');


-------------------------------------------------
| Id  | Operation                | Name          |
-------------------------------------------------
|   0 | SELECT STATEMENT         |               |
|   1 |  CONNECT BY WITH FILTERING|               |
|   2 |   NESTED LOOPS           |               |
|   3 |    TABLE ACCESS FULL     | T_CONSTRAINTS |
|   4 |    INDEX UNIQUE SCAN     | T_INDEXES_PK  |
|   5 |   HASH JOIN              |               |
|   6 |    CONNECT BY PUMP       |               |
|   7 |    TABLE ACCESS FULL     | T_CONSTRAINTS |
-------------------------------------------------
```

NO_CONNECT_BY_FILTERING

Usage: NO_CONNECT_BY_FILTERING([<@Block>])

Description: Prevents the SQL executor to filter data when perform CONNECT BY operation.

```
HELLODBA.COM>exec sql_explain('select /*+NO_CONNECT_BY_FILTERING(@"SEL$1")*/owner, table_name, level
from t_constraints connect by prior r_owner=owner and prior r_constraint_name = constraint_name start
with owner=:A','BASIC');


-------------------------------------------------------------------
| Id  | Operation                              | Name          |
-------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |               |
|   1 |  CONNECT BY NO FILTERING WITH START-WITH|              |
|   2 |   TABLE ACCESS FULL                    | T_CONSTRAINTS |
-------------------------------------------------------------------
```

XML 查询提示

COST_XML_QUERY_REWRITE

Usage: COST_XML_QUERY_REWRITE

escription: Instructs the optimizer to perform XML query rewrite based on cost. By default, the optimizer will perform XML query rewrite based on rules. We can the reason why cost based rewrite failed from the trace file of 19027 event.

```
HELLODBA.COM>desc xml_test
 Name                                      Null?    Type
 ----------------------------------------- --------
-----------------------------------------------
TABLE of SYS.XMLTYPE(XMLSchema "http://www.HelloDBA.com/xml/schema.xsd" Element "test-xml") STORAGE
Object-relational TYPE "test-xml748_T"

HELLODBA.COM>exec sql_explain('SELECT /*+*/EXTRACT(VALUE(x), ''/test-xml/id'') FROM xml_test x WHERE
EXTRACTVALUE(value(x), ''/test-xml/name'') = ''aaa''', 'TYPICAL OUTLINE');
Plan hash value: 3532887779
… …
Outline Data
-------------


  /*+
      BEGIN_OUTLINE_DATA
      FULL(@"SEL$1" "X"@"SEL$1")
      OUTLINE_LEAF(@"SEL$1")
      NO_COST_XML_QUERY_REWRITE
      XMLINDEX_REWRITE_IN_SELECT
      XMLINDEX_REWRITE
      XML_DML_RWT_STMT
      … …
      END_OUTLINE_DATA
  */
```

```
HELLODBA.COM>exec sql_explain('SELECT /*+COST_XML_QUERY_REWRITE*/EXTRACT(VALUE(x), ''/test-xml/id'')
FROM xml_test x WHERE  EXTRACTVALUE(value(x), ''/test-xml/name'') = ''aaa''', 'TYPICAL OUTLINE');
Plan hash value: 3532887779
… …
Outline Data
------------

  /*+
      BEGIN_OUTLINE_DATA
      FULL(@"SEL$1" "X"@"SEL$1")
      OUTLINE_LEAF(@"SEL$1")
      XMLINDEX_REWRITE_IN_SELECT
      XMLINDEX_REWRITE
      XML_DML_RWT_STMT
      FORCE_XML_QUERY_REWRITE
      … …
      END_OUTLINE_DATA
  */
```

NO_COST_XML_QUERY_REWRITE

Usage: NO_COST_XML_QUERY_REWRITE

Description: Prevent the optimizer to perform XML query rewrite based on cost.

Refer to the demo of COST_XML_QUERY_REWRITE

FORCE_XML_QUERY_REWRITE

Usage: FORCE_XML_QUERY_REWRITE

Description: Force the optimizer to rewrite the XML query.

```
HELLODBA.COM>exec sql_explain('select /*+FORCE_XML_QUERY_REWRITE*/rowid from xml_test where
existsnode(object_value,''/test-xml[name="aaa"]'')=1', 'BASIC PREDICATE');
Plan hash value: 3532887779
… …
Predicate Information (identified by operation id):
---------------------------------------------------


   1 - filter("XML_TEST"."SYS_NC00009$"='aaa')
```

NO_XML_QUERY_REWRITE

Usage: NO_XML_QUERY_REWRITE

Description: Forbidden the optimizer to rewrite the XML query.

```
HELLODBA.COM>exec sql_explain('select /*+NO_XML_QUERY_REWRITE*/rowid from xml_test where
existsnode(object_value,''/test-xml[name="aaa"]'')=1', 'BASIC PREDICATE');
Plan hash value: 3532887779
… …
Predicate Information (identified by operation id):
---------------------------------------------------


   1 - filter(EXISTSNODE(SYS_MAKEXML('43408109EBBC4C0C942DC83C286B7241',
```

```
                4347,"XML_TEST"."XMLEXTRA","XML_TEST"."XMLDATA"),'/test-xml[name="aaa"]'
                )=1)
```

XML_DML_RWT_STMT

Usage: XML_DML_RWT_STMT

Description: 指示优化器对 XML 数据操作进行重写。概要数据中存在该提示的语句，如果再嵌入 NO_XML_DML_REWRITE 进行解析，会导致该提示从概要数据中消失。

```
HELLODBA.COM>CREATE TABLE xmltest OF XMLType xmltype STORE as BINARY XML;

Table created.

HELLODBA.COM>exec sql_explain('UPDATE /*+XML_DML_RWT_STMT*/xmltest SET OBJECT_VALUE =
deleteXML(OBJECT_VALUE, ''/product_details[@id=1]/product[@id=2]'')', 'BASIC PREDICATE');

----------------------------------------
| Id  | Operation           | Name    |
----------------------------------------
|   0 | UPDATE STATEMENT    |         |
|   1 |  UPDATE             | XMLTEST |
|   2 |   TABLE ACCESS FULL | XMLTEST |
|   3 |   SORT AGGREGATE    |         |
|*  4 |    XPATH EVALUATION |         |
----------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - filter(TO_BINARY_DOUBLE("P"."C_01$")=2.0E+000D)

PL/SQL procedure successfully completed.

HELLODBA.COM>exec sql_explain('UPDATE /*+NO_XML_DML_REWRITE*/xmltest SET OBJECT_VALUE =
deleteXML(OBJECT_VALUE, ''/product_details[@id=1]/product[@id=2]'')', 'BASIC PREDICATE');
Plan hash value: 3332225581

----------------------------------------
| Id  | Operation           | Name    |
----------------------------------------
|   0 | UPDATE STATEMENT    |         |
|   1 |  UPDATE             | XMLTEST |
|   2 |   TABLE ACCESS FULL| XMLTEST |
----------------------------------------

PL/SQL procedure successfully completed.
```

NO_XML_DML_REWRITE

Usage: NO_XML_DML_REWRITE

Description: 禁止优化器对 DML 语句中的 XML 数据操作（DELETEXML、UPDATEXML、INSERTCHILDXML、INSERTCHILDXMLBEFORE 和 INSERTCHILDXMLAFTER）进行重写

见上例

XMLINDEX_REWRITE

Usage: XMLINDEX_REWRITE

Description: 指示优化器使用 XML 索引（XMLIndex）进行重写

```
HELLODBA.COM>create table t_xml_tab1 (ID NUMBER NOT NULL, XMLDATA XMLType) xmltype column "XMLDATA" STORE
AS BINARY XML;

Table created.

HELLODBA.COM>create index t_xml_tab1_IDX_1 on t_xml_tab1(XMLDATA) indextype is xdb.xmlindex PARAMETERS
('PATHS ( INCLUDE (/ROOT))' );

Index created.

HELLODBA.COM>exec sql_explain('SELECT /*+ XMLINDEX_REWRITE */ COUNT(ID) FROM t_xml_tab1 t WHERE
t.XMLDATA.EXISTSNODE(''/ROOT'') = 1', 'BASIC OUTLINE');

-------------------------------------------------------------------------
| Id  | Operation                     | Name                            |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                                 |
|   1 |  SORT AGGREGATE               |                                 |
|   2 |   NESTED LOOPS SEMI           |                                 |
|   3 |    TABLE ACCESS FULL          | T_XML_TAB1                      |
|   4 |    TABLE ACCESS BY INDEX ROWID| SYS138768_T_XML_TAB_PATH_TABLE  |
|   5 |     INDEX RANGE SCAN          | SYS138768_T_XML_TAB_PIKEY_IX    |
-------------------------------------------------------------------------
```

NO_XMLINDEX_REWRITE

Usage: NO_XMLINDEX_REWRITE

Description: 禁止优化器使用 XML 索引（XMLIndex）进行重写

```
HELLODBA.COM>exec sql_explain('SELECT /*+ NO_XMLINDEX_REWRITE */ COUNT(ID) FROM t_xml_tab1 t WHERE
t.XMLDATA.EXISTSNODE(''/ROOT'') = 1', 'BASIC OUTLINE');

-----------------------------------------
| Id  | Operation          | Name       |
-----------------------------------------
|   0 | SELECT STATEMENT   |            |
|   1 |  SORT AGGREGATE    |            |
|   2 |   TABLE ACCESS FULL| T_XML_TAB1 |
-----------------------------------------
```

XMLINDEX_REWRITE_IN_SELECT

Usage: XMLINDEX_REWRITE_IN_SELECT

Description: Instructs the optimizer using XML index into rewrite SELECT clause.

```
HELLODBA.COM>exec sql_explain('SELECT /*+ XMLINDEX_REWRITE_IN_SELECT */XMLQuery(''$r/ROOT'' PASSING
t.XMLDATA AS "r" RETURNING CONTENT) FROM t_xml_tab1 t WHERE t.XMLDATA.EXISTSNODE(''/ROOT'') = 1', 'BASIC
OUTLINE');


-------------------------------------------------------------------------
| Id  | Operation                    | Name                           |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                                |
|   1 |  SORT GROUP BY               |                                |
|   2 |   TABLE ACCESS BY INDEX ROWID| SYS138768_T_XML_TAB_PATH_TABLE |
|   3 |    INDEX RANGE SCAN          | SYS138768_T_XML_TAB_PIKEY_IX   |
|   4 |  NESTED LOOPS SEMI           |                                |
|   5 |   TABLE ACCESS FULL          | T_XML_TAB1                     |
|   6 |   TABLE ACCESS BY INDEX ROWID| SYS138768_T_XML_TAB_PATH_TABLE |
|   7 |    INDEX RANGE SCAN          | SYS138768_T_XML_TAB_PIKEY_IX   |
-------------------------------------------------------------------------
```

NO_XMLINDEX_REWRITE_IN_SELECT

Usage: NO_XMLINDEX_REWRITE_IN_SELECT

Description: Prevents the optimizer using XML index into rewrite SELECT clause.

```
HELLODBA.COM>exec sql_explain('SELECT /*+ NO_XMLINDEX_REWRITE_IN_SELECT */XMLQuery(''$r/ROOT''
PASSING t.XMLDATA AS "r" RETURNING CONTENT) FROM t_xml_tab1 t WHERE t.XMLDATA.EXISTSNODE(''/ROOT'') =
1', 'BASIC OUTLINE');


-------------------------------------------------------------------------
| Id  | Operation                    | Name                           |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                                |
|   1 |  NESTED LOOPS SEMI           |                                |
|   2 |   TABLE ACCESS FULL          | T_XML_TAB1                     |
|   3 |   TABLE ACCESS BY INDEX ROWID| SYS138768_T_XML_TAB_PATH_TABLE |
|   4 |    INDEX RANGE SCAN          | SYS138768_T_XML_TAB_PIKEY_IX   |
-------------------------------------------------------------------------
```

CHECK_ACL_REWRITE

Usage: CHECK_ACL_REWRITE

Description: Unknown. It might instruct the optimizer to check the Access Control List when rewriting XML query.

NO_CHECK_ACL_REWRITE

Usage: NO_CHECK_ACL_REWRITE

Description: Unknown. It might prevent the optimizer to check the Access Control List when rewriting XML query.

INLINE_XMLTYPE_NT

Usage: INLINE_XMLTYPE_NT

Description: Unknown. It might to instruct the optimizer to transform the inline XMLTYPE to Nested Table.

XMLINDEX_SEL_IDX_TBL

Usage: XMLINDEX_SEL_IDX_TBL

Description: Unknown

Distributed Query Hints

DRIVING_SITE

Usage: DRIVING_SITE([<Remote Table>])

Description: Instructs the optimizer to choose the site of remote table as the driving site to execute the query.

```
HELLODBA.COM>exec sql_explain('select /*+ driving_site(rt) */count(*) from t_tables lt,
t_tables@ora11r2 rt where lt.owner = rt.owner and lt.table_name = rt.table_name', 'TYPICAL NOTE');
... ...

Remote SQL Information (identified by operation id):
----------------------------------------------

   3 - SELECT "OWNER","TABLE_NAME" FROM "T_TABLES" "A2" (accessing '!' )

Note
-----
   - fully remote statement
```

REMOTE_MAPPED

Usage: REMOTE_MAPPED([<Remote Database Link >])

Description: Instruct the optimizer to map the query via the database link.

```
HELLODBA.COM>exec sql_explain('select /*+ remote_mapped(ORA11R2) */count(*) from T_USERS@ORA11R2 u,
t_tables t where t.owner=u.username', 'TYPICAL');

Remote SQL Information (identified by operation id):
----------------------------------------------

   3 - SELECT "OWNER" FROM "T_TABLES" "A1" (accessing '!' )

Note
-----
   - fully remote statement
```

OPAQUE_TRANSFORM

Usage: OPAQUE_TRANSFORM

Description: It appears in the internal SQL in remote server generated by the distributed query using the format of INSERT ... SELECT ... FROM.

Demo (Local DB, 10.2.0.4):

```
HELLODBA.COM>create table t_objects_dummy2 as select * from t_objects@ora11r2 where 1=2;

Table created.

HELLODBA.COM>exec sql_explain('insert into t_objects_dummy2 select * from
t_objects@ora11r2','TYPICAL');
... ...

Remote SQL Information (identified by operation id):
----------------------------------------------------

   1 - SELECT /*+ OPAQUE_TRANSFORM */ "OWNER","OBJECT_NAME","SUBOBJECT_NAME","OBJECT_I
       D","DATA_OBJECT_ID","OBJECT_TYPE","CREATED","LAST_DDL_TIME","TIMESTAMP","STATUS","TEMP
       ORARY","GENERATED","SECONDARY","NAMESPACE","EDITION_NAME","LIO" FROM "T_OBJECTS"
       "T_OBJECTS" (accessing 'ORA11R2' )

HELLODBA.COM>insert into t_objects_dummy2 select * from t_objects@ora11r2;

72116 rows created.
```

(Remote DB, 11.2.0.1)

```
HELLODBA.COM>select sql_text from v$sqlarea where sql_text like '%OPAQUE_TRANSFORM%' and sql_text not
like '%v$sqlarea%';

SQL_TEXT
--------------------------------------------------------------------------------------------------------
----
SELECT /*+ OPAQUE_TRANSFORM */
"OWNER","OBJECT_NAME","SUBOBJECT_NAME","OBJECT_ID","DATA_OBJECT_ID","OBJECT_TYPE","CREATE
D","LAST_DDL_TIME","TIMESTAMP","STATUS","TEMPORARY","GENERATED","SECONDARY","NAMESPACE","EDITION_NA
ME","LIO" FROM "T_OBJECTS" "T_OBJECTS"
```

Parallel Query Hints

STATEMENT_QUEUING

Usage: STATEMENT_QUEUING

Description: In the auto parallel degree mode (11gR2 feature), if there is insufficient resource for parallel query, it will be pushed into waiting queue. We can query V$SQL_MONITOR to check the status of the statement.

NO_STATEMENT_QUEUING

Usage: NO_STATEMENT_QUEUING

Description: In the auto parallel degree mode (11gR2 feature), even if there is insufficient resource for parallel query, it will still runing.

Parameter "_parallel_statement_queuing" controls the parallel statement queuing feature.

GBY_PUSHDOWN

Usage: GBY_PUSHDOWN([<@Block>])

Description: 指示优化器在对并行查询进行代价估算时，考虑将 GROUP BY 操作推入并行服务进程的情况；

```
HELLODBA.COM>exec sql_explain('SELECT /*+ FULL(T) parallel(T DEFAULT) GBY_PUSHDOWN */ owner,
table_name, COUNT (status) cnt FROM t_tables t GROUP BY owner, table_name', 'BASIC OUTLINE');


-------------------------------------------
| Id  | Operation               | Name     |
-------------------------------------------
|   0 | SELECT STATEMENT        |          |
|   1 |  PX COORDINATOR         |          |
|   2 |   PX SEND QC (RANDOM)   | :TQ10001 |
|   3 |    HASH GROUP BY        |          |
|   4 |     PX RECEIVE          |          |
|   5 |      PX SEND HASH       | :TQ10000 |
|   6 |       HASH GROUP BY     |          |
|   7 |        PX BLOCK ITERATOR|          |
|   8 |         TABLE ACCESS FULL| T_TABLES |
-------------------------------------------
```

NO_GBY_PUSHDOWN

Usage: NO_GBY_PUSHDOWN([<@Block>])

Description: 禁止优化器在对并行查询进行代价估算时，将 GROUP BY 操作推入并行服务进程；

```
HELLODBA.COM>exec sql_explain('SELECT /*+ FULL(T) parallel(T DEFAULT) NO_GBY_PUSHDOWN */ owner,
table_name, COUNT (status) cnt FROM t_tables t GROUP BY owner, table_name', 'BASIC OUTLINE');


-------------------------------------------
| Id  | Operation               | Name     |
-------------------------------------------
|   0 | SELECT STATEMENT        |          |
|   1 |  PX COORDINATOR         |          |
|   2 |   PX SEND QC (RANDOM)   | :TQ10001 |
|   3 |    HASH GROUP BY        |          |
|   4 |     PX RECEIVE          |          |
|   5 |      PX SEND HASH       | :TQ10000 |
|   6 |       PX BLOCK ITERATOR |          |
|   7 |        TABLE ACCESS FULL| T_TABLES |
-------------------------------------------
```

HWM_BROKERED

Usage: HWM_BROKERED

Description: 提示语句执行器在执行并行插入数据（或从其它表获取数据创建新表）时，使用高水位线查封器拆分高水位线，使得多个并行服务进程能共用一个扩展段。

示例（9i）：

```
HELLODBA.COM>alter session enable parallel dml;
```

```
Session altered.

HELLODBA.COM>explain plan for insert /*+ append */ into t_objects_dummy select /*+ full(o) parallel(o
2)*/* from t_objects o;

Explained.

HELLODBA.COM>select plan_table_output from table(dbms_xplan.display(null, null, 'ALL'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------
| Id  | Operation            | Name        | Rows  | Bytes | Cost  | TQ    |IN-OUT| PQ Distrib |
-----------------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT     |             | 32435 |  2945K|    22 |       |      |            |
|   1 |  LOAD AS SELECT      |             |       |       |       |       |      |            |
|   2 |   TABLE ACCESS FULL  | T_OBJECTS   | 32435 |  2945K|    22 | 63,00 | P->S | QC (RAND)  |
-----------------------------------------------------------------------------------------------------

HELLODBA.COM>explain plan for insert /*+ append HWM_BROKERED */ into t_objects_dummy select /*+ full(o)
parallel(o 2)*/* from t_objects o;

Explained.

HELLODBA.COM>select plan_table_output from table(dbms_xplan.display(null, null, 'ALL'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------
| Id  | Operation            | Name        | Rows  | Bytes | Cost  | TQ    |IN-OUT| PQ Distrib |
-----------------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT     |             | 32435 |  2945K|    22 |       |      |            |
|   1 |  LOAD AS SELECT      |             |       |       |       |       |      |            |
|   2 |   TABLE ACCESS FULL  | T_OBJECTS   | 32435 |  2945K|    22 | 64,00 | P->S | QC (RAND)  |
|   2 |   TABLE ACCESS FULL  | T_OBJECTS   | 32435 |  2945K|    22 | 63,00 | P->S | QC (RAND)  |
|   1 |  LOAD AS SELECT      |             |       |       |       |       |      |            |
|   2 |   TABLE ACCESS FULL  | T_OBJECTS   | 32435 |  2945K|    22 | 64,00 | P->S | QC (RAND)  |
|   2 |   TABLE ACCESS FULL  | T_OBJECTS   | 32435 |  2945K|    22 | 63,00 | P->S | QC (RAND)  |
|   0 | INSERT STATEMENT     |             | 32435 |  2945K|    22 |       |      |            |
|   1 |  LOAD AS SELECT      |             |       |       |       |       |      |            |
|   2 |   TABLE ACCESS FULL  | T_OBJECTS   | 32435 |  2945K|    22 | 64,00 | P->S | QC (RAND)  |
```

```
|   2 |   TABLE ACCESS FULL  | T_OBJECTS   | 32435 |  2945K|    22 | 63,00  | P->S | QC (RAND)  |
|   1 |  LOAD AS SELECT      |             |       |       |       |        |      |            |
|   2 |   TABLE ACCESS FULL  | T_OBJECTS   | 32435 |  2945K|    22 | 64,00  | P->S | QC (RAND)  |
|   2 |   TABLE ACCESS FULL  | T_OBJECTS   | 32435 |  2945K|    22 | 63,00  | P->S | QC (RAND)  |
-------------------------------------------------------------------------------------------------
```

NO_QKN_BUFF

Usage: NO_QKN_BUFF

Description: 禁止优化器使用动态分配的内存

```
HELLODBA.COM>exec sql_explain('select /*+parallel(t 8) parallel(o 8) leading(t o) pq_distribute(o hash hash) NO_QKN_BUFF*/* from t_tables t, t_objects o where t.owner=o.owner and t.table_name=o.object_name and o.status=:A','BASIC');


-------------------------------------------
| Id  | Operation              | Name       |
-------------------------------------------
|   0 | SELECT STATEMENT       |            |
|   1 |  PX COORDINATOR        |            |
|   2 |   PX SEND QC (RANDOM)  | :TQ10002   |
|   3 |    HASH JOIN           |            |
|   4 |     PX RECEIVE         |            |
|   5 |      PX SEND HASH      | :TQ10000   |
|   6 |       PX BLOCK ITERATOR|            |
|   7 |        TABLE ACCESS FULL| T_TABLES  |
|   8 |     PX RECEIVE         |            |
|   9 |      PX SEND HASH      | :TQ10001   |
|  10 |       PX BLOCK ITERATOR|            |
|  11 |        TABLE ACCESS FULL| T_OBJECTS |
-------------------------------------------
```

PARALLEL_INDEX

Usage: PARALLEL_INDEX([查询块] <Table> <Index1> [<Index2> ...] <并行度>) or PARALLEL_INDEX([查询块] <Table> (<Index 字段列表 1>) [(<Index 字段列表 2>) ...] <并行度>)

Description: 指示优化器选择并行方式访问本地分区索引。并行度可以为数字，也可以为 DEFAULT，使用系统默认并行度。

```
HELLODBA.COM>exec sql_explain('select /*+ qb_name(M) parallel_index(o t_objects_list_IDX1 2) */* from t_objects_list o where object_name like :A', 'BASIC');


-------------------------------------------------------------------
| Id  | Operation                          | Name                |
-------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                     |
|   1 |  PX COORDINATOR                    |                     |
|   2 |   PX SEND QC (RANDOM)              | :TQ10000            |
|   3 |    PX PARTITION LIST ALL           |                     |
|   4 |     TABLE ACCESS BY LOCAL INDEX ROWID| T_OBJECTS_LIST    |
```

```
|   5 |      INDEX RANGE SCAN               | T_OBJECTS_LIST_IDX1 |
-----------------------------------------------------------------
```

NO_PARALLEL_INDEX

Usage: NO_PARALLEL_INDEX([查询块] <Table> <Index1> [<Index2> ...]) or NO_PARALLEL_INDEX([查询块] <Table> (<Index 字段列表 1>) [(<Index 字段列表 2>) ...])

Description: 禁止优化器选择并行方式访问本地分区索引

```
HELLODBA.COM>alter index t_objects_list_IDX1 parallel(degree 2);

Index altered.

HELLODBA.COM>exec sql_explain('select /*+ qb_name(M) no_parallel_index(o t_objects_list_IDX1)*/* from t_objects_list o where object_name like :A', 'BASIC');

-----------------------------------------------------------------
| Id  | Operation                          | Name                |
-----------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                     |
|   1 |  PARTITION LIST ALL                |                     |
|   2 |   TABLE ACCESS BY LOCAL INDEX ROWID| T_OBJECTS_LIST      |
|   3 |    INDEX RANGE SCAN                | T_OBJECTS_LIST_IDX1 |
-----------------------------------------------------------------
```

PQ_DISTRIBUTE

Usage: PQ_DISTRIBUTE([<@Block>] <Table> <分发方式>) or PQ_DISTRIBUTE([<@Block>] <Table> <内分发方式> <外分发方式>)

Description: 指定并行查询中并行收、发进程直接的分发方式；

```
HELLODBA.COM>exec sql_explain('select /*+parallel(o 2)*/* from t_objects o where exists (select /*+hash_sj PQ_DISTRIBUTE(t HASH HASH)*/1 from t_tables t where o.owner = t.owner and o.object_name = t.table_name)', 'BASIC');

-----------------------------------------------------
| Id  | Operation                       | Name        |
-----------------------------------------------------
|   0 | SELECT STATEMENT                |             |
|   1 |  PX COORDINATOR                 |             |
|   2 |   PX SEND QC (RANDOM)           | :TQ10002    |
|   3 |    HASH JOIN RIGHT SEMI BUFFERED|             |
|   4 |     BUFFER SORT                 |             |
|   5 |      PX RECEIVE                 |             |
|   6 |       PX SEND HASH              | :TQ10000    |
|   7 |        INDEX FULL SCAN          | T_TABLES_PK |
|   8 |     PX RECEIVE                  |             |
|   9 |      PX SEND HASH               | :TQ10001    |
|  10 |       PX BLOCK ITERATOR         |             |
|  11 |        TABLE ACCESS FULL        | T_OBJECTS   |
```

```
------------------------------------------------
```

PARALLEL

Usage: PARALLE([[<@Block>] <Table>] [<并行度>])

Description: 指示优化器对查询块或者对象使用并行查询。其中，当中指定整条语句为并行查询（未指定查询块和表）时，并行度可以为 MANUAL,AUTO,DEFAULT 或者指定数字；指定某个表时，并行度可以为 DEFAULT 或者指定数字，

```
HELLODBA.COM>exec sql_explain('SELECT /*+ parallel(u default) */* from t_users u', 'BASIC');


----------------------------------------
| Id  | Operation            | Name     |
----------------------------------------
|   0 | SELECT STATEMENT     |          |
|   1 |  PX COORDINATOR      |          |
|   2 |   PX SEND QC (RANDOM)| :TQ10000 |
|   3 |    PX BLOCK ITERATOR |          |
|   4 |     TABLE ACCESS FULL| T_USERS  |
----------------------------------------
```

NO_PARALLEL

Usage: NO_PARALLEL(<Table>)

Description: 禁止优化器并行查询表

```
HELLODBA.COM>alter table t_objects_dummy parallel(degree 2);


Table altered.


HELLODBA.COM>exec sql_explain('SELECT /*+ no_parallel */* from t_objects_dummy o', 'BASIC');
Plan hash value: 2093122083


--------------------------------------------
| Id  | Operation         | Name           |
--------------------------------------------
|   0 | SELECT STATEMENT  |                |
|   1 |  TABLE ACCESS FULL| T_OBJECTS_DUMMY |
--------------------------------------------
```

SHARED

Usage: SHARED(<Table> [并行度])

Description: 指示优化器共享指定表的并行度。如果指定并行度，则和 PARALLEL 提示作用相同。

```
HELLODBA.COM>alter table t_objects_dummy parallel(degree 2);


Table altered.


HELLODBA.COM>alter table t_objects parallel(degree 8);


Table altered.
```

```
HELLODBA.COM>exec sql_explain('select /*+ full(o) full(d) shared(d) */count(*) from t_objects_dummy d,
t_objects o where o.owner=d.owner and o.object_name = d.object_name', 'BASIC COST');

-------------------------------------------------------------------
| Id  | Operation                | Name            | Cost (%CPU)|
-------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                 |   719   (1)|
|   1 |  SORT AGGREGATE          |                 |            |
|   2 |   PX COORDINATOR         |                 |            |
|   3 |    PX SEND QC (RANDOM)   | :TQ10002        |            |
|   4 |     SORT AGGREGATE       |                 |            |
|   5 |      HASH JOIN           |                 |   719   (1)|
|   6 |       PX RECEIVE         |                 |   231   (0)|
|   7 |        PX SEND HASH      | :TQ10000        |   231   (0)|
|   8 |         PX BLOCK ITERATOR|                 |   231   (0)|
|   9 |          TABLE ACCESS FULL| T_OBJECTS       |   231   (0)|
|  10 |       PX RECEIVE         |                 |   486   (0)|
|  11 |        PX SEND HASH      | :TQ10001        |   486   (0)|
|  12 |         PX BLOCK ITERATOR|                 |   486   (0)|
|  13 |          TABLE ACCESS FULL| T_OBJECTS_DUMMY |   486   (0)|
-------------------------------------------------------------------

HELLODBA.COM>exec sql_explain('select /*+ full(o) full(d) shared(o) */count(*) from t_objects_dummy d,
t_objects o where o.owner=d.owner and o.object_name = d.object_name', 'BASIC COST');

-------------------------------------------------------------------
| Id  | Operation                | Name            | Cost (%CPU)|
-------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                 |  1437   (1)|
|   1 |  SORT AGGREGATE          |                 |            |
|   2 |   PX COORDINATOR         |                 |            |
|   3 |    PX SEND QC (RANDOM)   | :TQ10002        |            |
|   4 |     SORT AGGREGATE       |                 |            |
|   5 |      HASH JOIN           |                 |  1437   (1)|
|   6 |       PX RECEIVE         |                 |   463   (1)|
|   7 |        PX SEND HASH      | :TQ10000        |   463   (1)|
|   8 |         PX BLOCK ITERATOR|                 |   463   (1)|
|   9 |          TABLE ACCESS FULL| T_OBJECTS       |   463   (1)|
|  10 |       PX RECEIVE         |                 |   973   (1)|
|  11 |        PX SEND HASH      | :TQ10001        |   973   (1)|
|  12 |         PX BLOCK ITERATOR|                 |   973   (1)|
|  13 |          TABLE ACCESS FULL| T_OBJECTS_DUMMY |   973   (1)|
-------------------------------------------------------------------
```

NOPARALLEL

Usage: NOPARALLEL([[<@Block>] <Table>])

Description: 禁止优化器对查询块或者对象使用并行查询。和 NO_PARALLEL 作用基本相同。

参见 NO_PARALLEL 示例。

PQ_MAP

Usage: PQ_MAP(<Table>)

Description: 未知。可能是用于并行查询的提示。

PQ_NOMAP

Usage: PQ_NOMAP(<Table>)

Description: 未知。可能是用于并行查询的提示。

PRESERVE_OID

Usage: PRESERVE_OID

Description: 未知。可能是用于并行查询的提示。

SYS_PARALLEL_TXN

Usage: SYS_PARALLEL_TXN

Description: 未知。可能是用于并行查询的递归调用语句上的。

CUBE_GB

Usage: CUBE_GB

Description: 未知。可能是用于 GROUP BY CUBE 并行查询的内部递归查询

该提示直接使用会导致 10g（10.2.0.4）在解析提示时在后台发生 ORA-00600 错误，但不会终止语句运行。

ORA-600: internal error code, arguments: [prsHintQbLevel-1], [890], [], [], [], [], [], []

发生类似情况的提示还有：

CUBE_GB/GBY_CONC_ROLLUP/PIV_GB/PIV_SSF/RESTORE_AS_INTERVALS/SAVE_AS_INTERVALS/SCN_ASCENDING/MODEL_DONTVERIFY_UNIQUENESS/TIV_GB/TIV_SSF

GBY_CONC_ROLLUP

Usage: GBY_CONC_ROLLUP

Description: 未知。可能是用于 GROUP BY ROLLUP 并行查询的内部递归查询

PIV_GB

Usage: PIV_GB

Description: 未知。出现在 GROUP BY 并行查询的内部递归查询语句上

示例（9i）：

```
HELLODBA.COM>select /*+qb_name(Q2) full(o2) parallel(o2 2)*/ owner, status, count(object_name) from
t_objects o2 where owner like 'D%' group by owner, status;


Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=11 Card=3 Bytes=42)
   1    0   SORT* (GROUP BY) (Cost=11 Card=3 Bytes=42)                              :Q17865001
   2    1     SORT* (GROUP BY) (Cost=11 Card=3 Bytes=42)                            :Q17865000
   3    2       TABLE ACCESS* (FULL) OF 'T_OBJECTS' (Cost=6 Card=3379 Bytes=47306)  :Q17865000
   1 PARALLEL_TO_SERIAL            SELECT /*+ CIV_GB */ A1.C0,A1.C1,COUNT(SYS_O
                                   P_CSR(A1.C2,0)) FROM :Q17865000 A1 GROUP BY
```

```
                                A1.C0,A1.C1
  2 PARALLEL_TO_PARALLEL        SELECT /*+ PIV_GB */ A1.C0 C0,A1.C1 C1,SYS_O
                                P_MSR(COUNT(*)) C2 FROM (SELECT /*+ NO_EXPAN
                                D ROWID(A2) */ A2."OWNER" C0,A2."STATUS" C1
                                FROM "T_OBJECTS" PX_GRANULE(0, BLOCK_RANGE,
                                DYNAMIC)  A2 WHERE A2."OWNER" LIKE 'D%') A1
                                GROUP BY A1.C0,A1.C1
  3 PARALLEL_COMBINED_WITH_PARENT
```

TIV_GB

Usage: TIV_GB

Description: 未知。出现在并行查询的内部递归查询语句上

TIV_SSF

Usage: TIV_SSF

Description: 未知。出现在并行查询的内部递归查询语句上

PIV_SSF

Usage: PIV_SSF

Description: 未知。出现在并行查询的内部递归查询语句上

RESTORE_AS_INTERVALS

Usage: RESTORE_AS_INTERVALS

Description: 未知。出现在并行查询的内部递归查询语句上

SAVE_AS_INTERVALS

Usage: SAVE_AS_INTERVALS

Description: 未知。出现在并行查询的内部递归查询语句上

SCN_ASCENDING

Usage: SCN_ASCENDING

Description: 未知。出现在并行查询的内部递归查询语句上

Model Query Hints

MODEL_MIN_ANALYSIS

Usage: MODEL_MIN_ANALYSIS

Description: Instructs the optimizer to do minimum transformation analysis on the main query when performing 用 query transformation on Model Query.

Trace below two queries with/without this hint:

```
HELLODBA.COM>alter session set events 'TRACE[RDBMS.SQL_Compiler.*]';

Session altered.

HELLODBA.COM>explain plan for
  2  SELECT /*+qb_name(m) no_merge(@inv) NO_MERGE(@"SEL$2")*/status, s
  3    FROM (select /*+qb_name(inv) no_merge(v)*/o.owner, o.status, o.object_name, o.created,
t.tablespace_name from v_objects_sys o, t_tables t where o.owner=t.owner and
o.object_name=t.table_name) q
  4   WHERE q.created < :A
```

```
  5    MODEL RETURN UPDATED ROWS
  6      PARTITION BY (status)
  7      DIMENSION BY (owner)
  8      MEASURES (object_name v, 1 s)
  9      RULES
 10      (s[any] = count(v) over (partition by status));

... ...

HELLODBA.COM>explain plan for
  2  SELECT /*+qb_name(m) MODEL_MIN_ANALYSIS no_merge(@inv) NO_MERGE(@"SEL$2")*/status, s
  3    FROM (select /*+qb_name(inv) no_merge(v)*/o.owner, o.status, o.object_name, o.created,
t.tablespace_name from v_objects_sys o, t_tables t where o.owner=t.owner and
o.object_name=t.table_name) q
  4   WHERE q.created < :A
  5    MODEL RETURN UPDATED ROWS
  6      PARTITION BY (status)
  7      DIMENSION BY (owner)
  8      MEASURES (object_name v, 1 s)
  9      RULES
 10      (s[any] = count(v) over (partition by status));
```

Comparing the trace content, we can find the optimizer performed simple filter push analysis:

```
FPD: Considering simple filter push (pre rewrite) in query block M (#0)
FPD:  Current where clause predicates  ??

try to generate transitive predicate from check constraints for query block M (#0)
finally:  ??

kkqfppRelFilter: Not pushing filter predicates in query block SEL$21876068 (#0) because no predicate
to push
FPD: Considering simple filter push (pre rewrite) in query block SEL$21876068 (#0)
FPD:  Current where clause predicates "T_OBJECTS"."OWNER"="T"."OWNER" AND
"T_OBJECTS"."OBJECT_NAME"="T"."TABLE_NAME" AND "T_OBJECTS"."OWNER"='SYS' AND
"T_OBJECTS"."CREATED"<:B1 AND "T"."OWNER"='SYS'
```

MODEL_NO_ANALYSIS

Usage: MODEL_NO_ANALYSIS

Description: Prevents the optimizer to do transformation analysis on the main query when performing. We can compare the optimizer tracing records to confirm the effect of this hint.

MODEL_PUSH_REF

Usage: MODEL_PUSH_REF

Description: Unknown. It might instruct the optimizer to push the predication in the main model to reference model.

NO_MODEL_PUSH_REF

Usage: NO_MODEL_PUSH_REF

Description: Unknown. It might prevent the optimizer to push the predication in the main model to reference model.

MODEL_COMPILE_SUBQUERY

Usage: MODEL_COMPILE_SUBQUERY

Description: Unknown. It might be used for model query transformation.

MODEL_DONTVERIFY_UNIQUENESS

Usage: MODEL_DONTVERIFY_UNIQUENESS

Description: Unknown. It might be used for model query transformation.

MODEL_DYNAMIC_SUBQUERY

Usage: MODEL_DYNAMIC_SUBQUERY

Description: Unknown. It might be used for model query transformation.

Partitioning Hints

X_DYN_PRUNE

Usage: X_DYN_PRUNE

Description: Instructs the SQL executor to using the result of sub query to prune the partitions dynamically.

```
HELLODBA.COM>alter session set tracefile_identifier = 'hash_X_DYN_PRUNE(10128)';

Session altered.

HELLODBA.COM>alter session set events '10128 trace name context forever, level 31';

Session altered.

HELLODBA.COM>select /*+use_hash(tr t2) X_DYN_PRUNE*/tr.* from t_objects_range tr, t_tables t2 where
tr.owner=t2.owner and tr.object_name=t2.table_name and t2.tablespace_name='USERS';
... ...

HELLODBA.COM>exec sql_explain('select /*+use_hash(tr t2) X_DYN_PRUNE*/tr.* from t_objects_range tr,
t_tables t2 where tr.owner=t2.owner and tr.object_name=t2.table_name and
t2.tablespace_name=''USERS''', 'BASIC OUTLINE');

-----------------------------------------------------------
| Id  | Operation                     | Name            |
-----------------------------------------------------------
|   0 | SELECT STATEMENT              |                 |
|   1 |  HASH JOIN                    |                 |
|   2 |   PART JOIN FILTER CREATE     | :BF0000         |
|   3 |    TABLE ACCESS BY INDEX ROWID| T_TABLES        |
|   4 |     INDEX RANGE SCAN          | T_TABLES_IDX3   |
|   5 |   PARTITION RANGE JOIN-FILTER |                 |
```

```
|   6 |    TABLE ACCESS FULL          | T_OBJECTS_RANGE |
-----------------------------------------------------
```

From the trace records of 10128 event, we got below entries:

```
kkpapDAExtSQuerySLvl strings sql1 SELECT distinct TBL$OR$IDX$PART$NUM("T_OBJECTS_RANGE", 0,  sql2
"OWNER", "OBJECT_NAME") FROM (SELECT "A1"."OWNER" "OWNER", "A1"."TABLE_NAME" "OBJECT_NAME" FROM
"T_TABLES" "A1" WHERE "A1"."TABLESPACE_NAME"='USERS') ORDER BY 1
```

SUBQUERY_PRUNING

Usage: SUBQUERY_PRUNING([<@Block>] <Table> PARTITION)

Description: Instructs the optimizer to using sub query to prune the partitions dynamically.

```
HELLODBA.COM>exec sql_explain('select /*+ use_merge(tr t2) SUBQUERY_PRUNING(tr PARTITION)*/tr.* from
t_objects_range tr, t_tables t2 where tr.owner=t2.owner and tr.object_name=t2.table_name and
t2.tablespace_name=''SYSTEM''', 'BASIC');


-----------------------------------------------------
| Id  | Operation                   | Name            |
-----------------------------------------------------
|   0 | SELECT STATEMENT            |                 |
|   1 |  MERGE JOIN                 |                 |
|   2 |   SORT JOIN                 |                 |
|   3 |    VIEW                     | index$_join$_002 |
|   4 |     HASH JOIN               |                 |
|   5 |      INDEX RANGE SCAN       | T_TABLES_IDX3   |
|   6 |      INDEX FAST FULL SCAN   | T_TABLES_PK     |
|   7 |   SORT JOIN                 |                 |
|   8 |    PARTITION RANGE SUBQUERY|                 |
|   9 |     TABLE ACCESS FULL       | T_OBJECTS_RANGE |
-----------------------------------------------------
```

From the trace records of 10128 event, we got below entries:

```
SQL text = text = SELECT distinct TBL$OR$IDX$PART$NUM("T_OBJECTS_RANGE", 0,  "OWNER", "OBJECT_NAME")
FROM (SELECT "T2"."OWNER" "OWNER", "T2"."TABLE_NAME" "OBJECT_NAME" FROM "T_TABLES" "T2" WHERE
"T2"."TABLESPACE_NAME"='SYSTEM') ORDER BY 1}
```

NO_SUBQUERY_PRUNING

Usage: NO_SUBQUERY_PRUNING([<@Block>] <Table>)

Description: Prevents the optimizer to using sub query to prune the partitions dynamically.

```
HELLODBA.COM>exec sql_explain('select /*+NO_SUBQUERY_PRUNING(O PARTITION) LEADING(u)*/o.* from
t_objects_range o, t_tables t, t_users u where o.owner=t.owner and o.object_name=t.table_name and
o.owner=u.username and t.tablespace_name=''USERS'' and u.user_id<10', 'BASIC OUTLINE');


-----------------------------------------------------
| Id  | Operation                     | Name            |
-----------------------------------------------------
|   0 | SELECT STATEMENT              |                 |
```

```
|   1 |  NESTED LOOPS                  |                 |
|   2 |   NESTED LOOPS                 |                 |
|   3 |    HASH JOIN                   |                 |
|   4 |     TABLE ACCESS BY INDEX ROWID| T_USERS         |
|   5 |      INDEX RANGE SCAN          | T_USERS_PK      |
|   6 |     PARTITION RANGE ALL        |                 |
|   7 |      TABLE ACCESS FULL         | T_OBJECTS_RANGE |
|   8 |    INDEX UNIQUE SCAN           | T_TABLES_PK     |
|   9 |   TABLE ACCESS BY INDEX ROWID  | T_TABLES        |
-------------------------------------------------------
```

Other Hints

RELATIONAL

Usage: RELATIONAL([<@Block>] <Table>)

Description: Instructs the optimizer to convert the Object to a relational table, similar to RELATIONAL function.

```
HELLODBA.COM>desc xmltable
 Name                                      Null?    Type
 ----------------------------------------- -------- ---------
 TABLE of PUBLIC.XMLTYPE

HELLODBA.COM>select * from xmltable;

SYS_NC_ROWINFO$
--------------------------------------------------------------------------------------------------------------
----
<other_xml>
  <outline_data>
    <hint>
      <IGNORE_OPTIM_EMBEDDED_HINTS/>
    </hint>
    ... ...
  </outline_data>
</other_xml>

1 row selected.

HELLODBA.COM>select /*+ relational(x) */* from xmltable x;

SYS_NC_OID$                      XMLDATA
--------------------------------
--------------------------------------------------------------------------
5477ABC43C2D4A85917F7328AA961884
<other_xml><outline_data><hint><IGNORE_OPTIM_EMBEDDED_HINTS></IGNORE_OPTIM_EMBED
```

```
DED_HINTS></hint><hint><OPTIMIZER_FEATURES_ENABLE>10.2.0.3</OPTIMIZER_FEATURES_E
NABLE></hint><hint><![CDATA[ALL_ROWS]]></hint><hint><OUTLINE_LEAF>"SEL$3BA1AD7C"
... ...
```

MONITOR

Usage: MONITOR

Description: Instruct Oracle monitor the running status of the statement, regardless of whether it fulfills the criteria (Parallel Query or Running for more than 5 seconds) or not.

```
HELLODBA.COM>show parameter CONTROL_MANAGEMENT_PACK_ACCESS

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
control_management_pack_access       string      DIAGNOSTIC+TUNING
HELLODBA.COM>select /*+ monitor */count(*) from t_users u;

  COUNT(*)
----------
        31

HELLODBA.COM>select sql_text, status from v$sql_monitor where sql_text like '%monitor%';

SQL_TEXT                                                    STATUS
----------------------------------------------------------- --------------------
select /*+ monitor */count(*) from t_users u                DONE (ALL ROWS)
```

NO_MONITOR

Usage: NO_MONITOR

Description: Prevent Oracle monitor the running status of the statement, regardless of whether it fulfills the criteria (Parallel Query or Running for more than 5 seconds) or not.

```
HELLODBA.COM>select /*+ no_monitor parallel(o 2) full(o) */ /*identifier*/ count(*) from t_objects o;

  COUNT(*)
----------
     72116

HELLODBA.COM>select sql_text, status from v$sql_monitor where sql_text like '%identifier%';

no rows selected
```

NESTED_TABLE_FAST_INSERT

Usage: NESTED_TABLE_FAST_INSERT

Description: Instructs SQL executor to insert data into nested table in fast mode. From the trace content of 10046 event, the data was inserted in batch.

```
HELLODBA.COM>CREATE OR REPLACE TYPE simple_type AS TABLE OF VARCHAR2(30);
  2  /
```

```
Type created.

HELLODBA.COM>CREATE TABLE t_nt_table (a NUMBER, b simple_type) NESTED TABLE b STORE AS t_nt_b;

Table created.

HELLODBA.COM>INSERT /*+ */ INTO t_nt_table select object_id, simple_type(object_name) from t_objects;

72116 rows created.

Elapsed: 00:00:18.77

HELLODBA.COM>INSERT /*+ NESTED_TABLE_FAST_INSERT */ INTO t_nt_table select object_id,
simple_type(object_name) from t_objects;

72116 rows created.

Elapsed: 00:00:07.79
```

NESTED_TABLE_GET_REFS

Usage: NESTED_TABLE_GET_REFS

Description: With this hint, user can access to nested table directly.

```
HELLODBA.COM>select /*+ */count(*) from T_NT_B;
select /*+ */count(*) from T_NT_B
                           *
ERROR at line 1:
ORA-22812: cannot reference nested table column's storage table

HELLODBA.COM>select /*+ nested_table_get_refs */count(*) from T_NT_B;

  COUNT(*)
----------
     72116
```

NESTED_TABLE_SET_SETID

Usage: NESTED_TABLE_SET_SETID

Description: With this hint, user can access to nested table directly.

```
HELLODBA.COM>select /*+ NESTED_TABLE_SET_SETID */count(*) from T_NT_B;

  COUNT(*)
----------
     72116
```

NO_MONITORING

Usage: NO_MONITORING

Description: Prevent Oracle monitor the column usage in predication, consequently, the dictionary table col_usage$ will not be updated by the execution of the statement.

```
HELLODBA.COM>select like_preds from sys.SQLT$_DBA_COL_USAGE_V where owner ='DEMO' and table_name =
'T_TABLES' and column_name = 'TABLE_NAME';

LIKE_PREDS
----------
        18

HELLODBA.COM>select /*run(7)*/count(*) from t_tables where 7=7 and table_name like 'T%';

  COUNT(*)
----------
        30

HELLODBA.COM>exec dbms_stats.gather_table_stats('DEMO', 'T_TABLES');

PL/SQL procedure successfully completed.

HELLODBA.COM>select like_preds from sys.SQLT$_DBA_COL_USAGE_V where owner ='DEMO' and table_name =
'T_TABLES' and column_name = 'TABLE_NAME';

LIKE_PREDS
----------
        19

HELLODBA.COM>select /*+ no_monitoring *//*run(8)*/count(*) from t_tables where 8=8 and table_name like
'T%';

  COUNT(*)
----------
        30

HELLODBA.COM>exec dbms_stats.gather_table_stats('DEMO', 'T_TABLES');

PL/SQL procedure successfully completed.

HELLODBA.COM>select like_preds from sys.SQLT$_DBA_COL_USAGE_V where owner ='DEMO' and table_name =
'T_TABLES' and column_name = 'TABLE_NAME';

LIKE_PREDS
----------
        19
```

NO_SQL_TUNE

Usage: NO_SQL_TUNE

Description: Prevent the optimizer to do SQL tuning on the statement.

```
HELLODBA.COM>select /* No_TUNE(2) *//*+NO_SQL_TUNE*/count(*) from t_users;

  COUNT(*)
----------
        31

... ...
HELLODBA.COM>exec :exec_name := dbms_sqltune.execute_tuning_task (:task_name,
'EXEC_'||substr(:task_name, length(:task_name)-4));

PL/SQL procedure successfully completed.

HELLODBA.COM>select dbms_sqltune.report_tuning_task (:task_name) from dual;
... ...
ADDITIONAL INFORMATION SECTION
-------------------------------------------------------------------------------
- 不支持的 SQL 语句类型。
```

RESTRICT_ALL_REF_CONS

Usage: RESTRICT_ALL_REF_CONS

Description: Restricts all cascaded operations caused by referencing constraints in the transactions.

```
HELLODBA.COM>select owner, table_name, constraint_name, r_owner, r_constraint_name, delete_rule from
dba_constraints where constraint_name = 'T_C_FK';

OWNER           TABLE_NAME        CONSTRAINT_NAME      R_OWNER        R_CONSTRAINT_NAME
DELETE_RULE
--------------- ----------------- -------------------- -------------- -------------------- ---------
DEMO            T_C               T_C_FK               DEMO           T_P_PK               CASCADE

HELLODBA.COM>delete /*+RESTRICT_ALL_REF_CONS*/from t_p where a=3;

1 row deleted.

HELLODBA.COM>select count(a) from t_c where a=3;

  COUNT(A)
----------
         1

HELLODBA.COM>commit;
commit
*
ERROR at line 1:
```

```
ORA-02091: transaction rolled back
ORA-02292: integrity constraint (DEMO.T_C_FK) violated - child record found
```

USE_HASH_AGGREGATION

Usage: USE_HASH_AGGREGATION([<@Block>])

Description: Instructs the optimizer using hash algorithm to perform aggregation operations.

```
HELLODBA.COM>alter session set "_gby_hash_aggregation_enabled"=false;

Session altered.

HELLODBA.COM>exec sql_explain('select /*+qb_name(M) USE_CONCAT(@M)*/owner, count(1) from t_objects o
group by owner', 'T
YPICAL OUTLINE');
Plan hash value: 87103648

-------------------------------------------------------------------------------------------
| Id  | Operation             | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                |    23 |   138 |   196   (8)| 00:00:02 |
|   1 |  HASH GROUP BY        |                |    23 |   138 |   196   (8)| 00:00:02 |
|   2 |   INDEX FAST FULL SCAN| T_OBJECTS_IDX8 | 72116 |   422K|   185   (2)| 00:00:02 |
-------------------------------------------------------------------------------------------
```

NO_USE_HASH_AGGREGATION

Usage: NO_USE_HASH_AGGREGATION([<@Block>])

Description: Prevents the optimizer using hash algorithm to perform aggregation operations.

```
HELLODBA.COM>exec sql_explain('select /*+qb_name(M) NO_USE_HASH_AGGREGATION(@M)*/owner, count(1) from
t_objects o group by owner', 'TYPICAL OUTLINE');
Plan hash value: 49003928

-------------------------------------------------------------------------------------------
| Id  | Operation             | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                |    23 |   138 |   196   (8)| 00:00:02 |
|   1 |  SORT GROUP BY        |                |    23 |   138 |   196   (8)| 00:00:02 |
|   2 |   INDEX FAST FULL SCAN| T_OBJECTS_IDX8 | 72116 |   422K|   185   (2)| 00:00:02 |
-------------------------------------------------------------------------------------------
```

BYPASS_RECURSIVE_CHECK

Usage: BYPASS_RECURSIVE_CHECK

Description: Unknown. It might instruct the parser not do recursive checking. It could be observed in the internal statement generated by Materialized View updating.

Demo:

```
HELLODBA.COM>alter session set sql_trace=true;

Session altered.
```

```
HELLODBA.COM>exec dbms_mview.refresh(list => 'MV_TABLES');


PL/SQL procedure successfully completed.
```

We got below statement from the tracing content.

```
INSERT /*+ BYPASS_RECURSIVE_CHECK */ INTO
"DEMO"."MV_TABLES"("OWNER","TABLE_NAME","TABLESPACE_NAME","CREATED","LAST_DDL_TIME") SELECT
"T"."OWNER","T"."TABLE_NAME","T"."TABLESPACE_NAME","O"."CREATED","O"."LAST_DDL_TIME" FROM "T_TABLES"
"T","T_OBJECTS" "O" WHERE "T"."OWNER"="O"."OWNER" AND "T"."TABLE_NAME"="O"."OBJECT_NAME" AND
"O"."OBJECT_TYPE"=:"SYS_B_0" AND "T"."TABLESPACE_NAME" IS NOT NULL
```

BYPASS_UJVC

Usage: BYPASS_UJVC

Description: Unknown. It might instruct parser not check unique constraint for the join view. It could be observed in the internal statement generated by Materialized View updating.

We got below statement from the trace file generated in previous demo.

```
update  /*+ BYPASS_UJVC */      ( select s.status status        from snap$ s, snap_reftime$ r
where s.sowner = r.sowner and s.vname = r.vname and         r.mowner = :1 and r.master = :2 and s.mlink
IS NULL         and bitand(s.status,16) = 0 and r.instsite =0 and s.instsite =0) v    set status = status
+ 16;
```

DOMAIN_INDEX_FILTER

Usage: DOMAIN_INDEX_FILTER([<@Block>] <Table> [(<Index>)]) or DOMAIN_INDEX_FILTER([<@Block>] <Table> [(<Indexed Columns>)])

Description: Instructs the optimizer the push filter to Composite Domain Index.

```
HELLODBA.COM>exec sql_explain('SELECT /*+ domain_index_filter(t t_tables_dix03) */ * FROM t_tables t
WHERE CONTAINS(owner, ''aaa'',1)>0 AND status = ''VALID''', 'TYPICAL');


-----------------------------------------------------------------------------------------------
| Id  | Operation                   | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                |     1 |   241 |     4   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| T_TABLES       |     1 |   241 |     4   (0)| 00:00:01 |
|*  2 |   DOMAIN INDEX              | T_TABLES_DIX03 |       |       |     4   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------


   2 - access("CTXSYS"."CONTAINS"("OWNER",'aaa',1)>0)
       filter("STATUS"='VALID')
```

NO_DOMAIN_INDEX_FILTER

Usage: NO_DOMAIN_INDEX_FILTER([<@Block>] <Table> [(<Index>)]) or NO_DOMAIN_INDEX_FILTER([<@Block>] <Table> [(<Indexed Columns>)])

Description: Prevents the optimizer the push filter to Composite Domain Index.

```
HELLODBA.COM>exec sql_explain('SELECT /*+ no_domain_index_filter(t t_tables_dix03) */ * FROM t_tables
t WHERE CONTAINS(owner, ''aaa'',1)>0 AND status = ''VALID''', 'TYPICAL');

-------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                |     1 |   241 |     4   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS BY INDEX ROWID| T_TABLES       |     1 |   241 |     4   (0)| 00:00:01 |
|*  2 |   DOMAIN INDEX               | T_TABLES_DIX03 |       |       |     4   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("STATUS"='VALID')
   2 - access("CTXSYS"."CONTAINS"("OWNER",'aaa',1)>0)
```

DOMAIN_INDEX_SORT

Usage: DOMAIN_INDEX_SORT

Description: Instructs the optimizer the push sorting columns to Composite Domain Index.

```
HELLODBA.COM>exec sql_explain('SELECT /*+ domain_index_sort */ * FROM t_tables t WHERE
CONTAINS(tablespace_name, ''aaa'',1)>0 ORDER BY table_name, score(1) desc', 'TYPICAL');
Plan hash value: 991332243

-------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                |     1 |   241 |     5  (20)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| T_TABLES       |     1 |   241 |     5  (20)| 00:00:01 |
|*  2 |   DOMAIN INDEX               | T_TABLES_DIX02 |       |       |     4   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("CTXSYS"."CONTAINS"("TABLESPACE_NAME",'aaa',1)>0)
```

NO_DOMAIN_INDEX_ SORT

Usage: NO_DOMAIN_INDEX_ SORT

Description: Prevents the optimizer the push sorting columns to Composite Domain Index.

```
HELLODBA.COM>exec sql_explain('SELECT /*+ no_domain_index_sort */ * FROM t_tables t WHERE
CONTAINS(tablespace_name, ''aaa'',1)>0 ORDER BY table_name, score(1) desc', 'TYPICAL');

-------------------------------------------------------------------------------------------------
```

```
| Id  | Operation                    | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                |     1 |   241 |     5  (20)| 00:00:01 |
|   1 |  SORT ORDER BY               |                |     1 |   241 |     5  (20)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID| T_TABLES       |     1 |   241 |     4   (0)| 00:00:01 |
|*  3 |    DOMAIN INDEX              | T_TABLES_DIX02 |       |       |     4   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("CTXSYS"."CONTAINS"("TABLESPACE_NAME",'aaa',1)>0)
```

DST_UPGRADE_INSERT_CONV

Usage: DST_UPGRADE_INSERT_CONV

Description: With this hint, Oracle will add an internal function (ORA_DST_CONVERT(INTERNAL_FUNCTION())) to modify the column defined as TIMESTAMP WITH TIME ZONE when using package DBMS_DST to upgrade the time zone of the database.

NO_DST_UPGRADE_INSERT_CONV

Usage: NO_DST_UPGRADE_INSERT_CONV

Description: With this hint, Oracle will not add an internal function (ORA_DST_CONVERT(INTERNAL_FUNCTION())) to modify the column defined as TIMESTAMP WITH TIME ZONE when using package DBMS_DST to upgrade the time zone of the database.

STREAMS

Usage: STREAMS

Description: Unknown. It might instructs the SQL execution to transfer the data in stream.

DEREF_NO_REWRITE

Usage: DEREF_NO_REWRITE(<@Block>)

Description: Unknown. It might prevent the optimizer to rewrite the Materialized View with BUILD DEFERRED option.

MV_MERGE

Usage: MV_MERGE

Description: Unknown. It might be used for CUBE.

EXPR_CORR_CHECK

Usage: EXPR_CORR_CHECK

Description: Unknown. It might instruct the parser to do referencing checking where analyzing Expression Filter.

INCLUDE_VERSION

Usage: INCLUDE_VERSION

Description: Unknown. It could be observed from the internal statement generated by Advanced Replication. It might be used to keep the compatibility when replicating data among databases with different versions.

VECTOR_READ

Usage: VECTOR_READ

Description: Unknown. It might be used for Vector Filter in hash join.

VECTOR_READ_TRACE

Usage: VECTOR_READ_TRACE

Description: Unknown. It might be used for Vector Filter in hash join.

USE_WEAK_NAME_RESL

Usage: USE_WEAK_NAME_RESL

Description: Unknown. It might instructs the parser using the internal name instead the user-defined name to find Resource Location. It could be observed from the internal statements generated statistics data gathering and Expression Filter.

```
select /*+ no_parallel(t) no_parallel_index(t) dbms_stats cursor_sharing_exact use_weak_name_resl
dynamic_sampling(0) no_monitoring no_substrb_pad */ count(*)  from "DEMO"."T_OBJECTS" sample block
(  9.1911764706,1) t
```

NO_PARTIAL_COMMIT

Usage: NO_PARTIAL_COMMIT

Description: Unknown. It might be used to prevent the commit of internal recursive transaction. It could be observed from the internal statements generated by maintenance of the table with nested object.

```
HELLODBA.COM>alter session set events 'sql_trace wait=true, bind=true, plan_stat=all_executions';

Session altered.

HELLODBA.COM>INSERT /*+ NESTED_TABLE_FAST_INSERT */ INTO t_nt_table select object_id,
simple_type(object_name) from t_objects;

... ...
```

We got below statement from the trace file.

```
INSERT /*+ NO_PARTIAL_COMMIT REF_CASCADE_CURSOR */ INTO  "DEMO"."T_NT_B"
("NESTED_TABLE_ID","COLUMN_VALUE")  VALUES(:1, :2)
```

REF_CASCADE_CURSOR

Usage: REF_CASCADE_CURSOR

Description: Unknown. It might be used to prevent the commit of internal recursive transaction. It could be observed from the internal statements generated by maintenance of the table with nested object.

Refer to the demo of NO_PARTIAL_COMMIT

NO_REF_CASCADE

Usage: NO_REF_CASCADE

Description: Unknown. It might prevent the internal recursive statement to use the cascade cursor.

SQLLDR

Usage: SQLLDR

Description: Unknown. It might be used in the internal statements generated by SQL*Loader.

SYS_RID_ORDER

Usage: SYS_RID_ORDER

Description: Unknown. It might be used in the internal statements generated by maintenance of Materialized View.

OVERFLOW_NOMOVE

Usage: OVERFLOW_NOMOVE

Description: Unknown. It might prevent Oracle to move the data of other segment when overflow occurs due to partition splitting.

LOCAL_INDEXES

Usage: LOCAL_INDEXES

Description: Unknown

MERGE_CONST_ON

Usage: MERGE_CONST_ON

Description: Unknown

QUEUE_CURR

Usage: QUEUE_CURR

Description: Unknown. It might be used for Advanced Queue.

CACHE_CB

Usage: CACHE_CB([<@Block>] <Table>)

Description: Unknown. It might be used for Advanced Queue.

Trace the process of DBMS_AQ.DEQUEUUE (delivery_mode is PERSISTENT), we got below statement.

```
delete /*+ CACHE_CB("QUETABLET") */ from "DEMO"."QUETABLET" where rowid = :1;
```

QUEUE_ROWP

Usage: QUEUE_ROWP

Description: Unknown. It might be used for Advanced Queue.

BUFFER

Usage: BUFFER

Description: Unknown. It might be used for Advanced Queue.

NO_BUFFER

Usage: NO_BUFFER

Description: Unknown. It might be used for Advanced Queue.

BITMAP

Usage: BITMAP

Description: Unknown.。