Oracle SQL 优化与调优技术详解

附录：Oracle SQL 提示

黄玮
www.HelloDBA.com

本电子文档为《Oracle 高性能SQL引擎剖析：Oracle SQL 优化与调优技术详解》一书的附录部分。作为对该书的补充，帮助读者理解和掌握“提示”这一项在SQL优化中使用的这一重要辅助手段。

索引

附录：提示（HINT）的含义和示例

SQL 提示是用户干预优化器以及语句执行的行为的重要手段。在 SQL 优化方法当中，也起到重要作用。例如，在自动优化给出的 SQL 优化配置，提示就可能作为辅助的概要数据存在，帮助优化器找到性能更好的执行计划。不过，并不是所有提示都能作为概要数据。很多提示是伴随着优化器的新的特性出现的，有些提示也随之优化器特性的过期而消失。在 11G 当中，Oracle 提供了一个视图（V$SQL_HINT），使我们可以看到各个版本出现的提示以及其作为概要数据的优化器版本等信息。不同的提示属于不同的 SQL 特性，只有当前该 SQL 特性启用后，该提示才能其作用。例如，“HASH_AJ”是 QKSFM_CBO 特性中的提示，当当前优化器模式被设置为 RBO 时，该提示不起作用。

其中还有一些提示是 Oracle 产生的递归调用语句中使用的，这些语句和提示只在内部其作用，无法被用户直接调用。

嵌入语句的提示为一段注释，格式为 /*+ <提示 1> <提示 2> ... */。多个提示可以存在同一段注释当中，也可以分别在多个注释段当中。并且，提示所在的注释段必须是在关键字 SELECT、UPDATE、INSERT、MERGE 或 DELETE 之后。如果语句中存在子查询，提示可以以全局方式（推荐的方式）或者本地方式出现。本地方式是指将提示嵌入子查询当中；全局方式是指将所有提示都嵌入主查询当中，并且使用查询块标识（可以使用系统自动生成的标识、也可以使用提示指定的标识串）指明该提示所影响的对象在查询中的位置（例如：<对象名>@<@查询块>）。提示参数中的对象在查询当中存在别名的话，也可以用别名代替对象名。

提示：嵌入 SQL 提示是一段注释。因此，除了常用的/*+<提示>*/外，还可以用--+<提示>的方式嵌入。例如：

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

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

下面，我们将将以 11.2.0.1 中该视图中出现的所有提示为基础，分别解释每个提示的作用、并给出示例。提示的出现版本、概要数据版本、SQL 特性以及相反提示可以通过视图 V$SQL_HINT（11g）或者 ALL_SQL_HINT 查询。

SQL 特性

首先，我们看一下存在哪些 SQL 特性。特性之间存在隶属关系，并且，有些特性会隶属于多个特性。下面以树状关系列出所有特性：

```
+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
```

以下描述各个提示。我们按照使用范围分类，并没有严格遵循其所属 SQL 特性。示例如果没有特别说明，则是运行在示例出现版本或之前版本。

访问路径提示

CLUSTER

语法：CLUSTER([<@查询块>] <表>)

描述：指示优化器通过簇来访问表，仅对簇表有效

```
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

语法：HASH([<@查询块>] <表名>)

描述：指示优化器通过哈希簇来访问表，仅对哈希簇表有效

```
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

语法：ROWID([<@查询块>] <表>)

描述：指示优化器通过 ROWID 来访问和扫描表；

```
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

语法：FULL([<@查询块>] <表>)

描述：指示优化器采用全表扫描的方式访问表

```
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

语法：INDEX([<@查询块>] <表> [<索引>]) 或者 INDEX([<@查询块>] <表> [(<索引字段列表>)])

描述：指示优化器通过索引来访问和扫描表。

```
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

语法：INDEX_ASC([<@查询块>] <表> [<索引>]) 或者 INDEX_ASC([<@查询块>] <表> [(<索引字段列表>)])

描述：指示优化器通过索引来访问和扫描表，如果是索引范围扫描，则以索引键值的递增顺序扫描索引记录；

示例：（注意输出结果顺序）

```
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

语法：INDEX_DESC([<@查询块>] <表> [<索引>]) 或者 INDEX_DESC([<@查询块>] <表> [(<索引字段列表>)])

描述：指示优化器通过索引来访问和扫描表，如果是索引范围扫描，则以索引键值的递减顺序扫描索引记录；

示例：（注意输出结果顺序）

```
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

语法：NO_INDEX([<@查询块>] <表> [<索引>]) 或者 NO_INDEX([<@查询块>] <表> [(<索引字段列表>)])

描述：禁止优化器使用索引扫描访问表。

```
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

语法：INDEX_FFS([<@查询块>] <表> [<索引>]) 或者 INDEX_FFS([<@查询块>] <表> [(<索引字段列表>)])

描述：指示优化器以索引快速完全扫描的方式访问表

```
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

语法：NO_INDEX_FFS([<@查询块>] <表> [<索引>]) 或者 NO_INDEX_FFS([<@查询块>] <表> [(<索引字段列表>)])

描述：禁止优化器使用索引快速完全扫描访问表。

```
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

语法：INDEX_RRS([<@查询块>] <表> [<索引>]) 或者 INDEX_RRS([<@查询块>] <表> [(<索引字段列表>)])

描述：这一提示通常是由优化器在并行查询，加在并行串行直接输入输出时，生成的内部查询语句上，控制对索引快速完全扫描的并行操作。

示例：（以下示例在 9i 中运行）

```
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

语法：INDEX_SS([<@查询块>] <表> [<索引>]) 或者 INDEX_SS([<@查询块>] <表> [(<索引字段列表>)])

描述：指示优化器通过跳跃索引扫描来访问表；

```
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

语法：INDEX_SS_ASC([<@查询块>] <表> [<索引>]) 或者 INDEX_SS_ASC([<@查询块>] <表> [(<索引字段列表>)])

描述：指示优化器通过跳跃索引扫描来访问表，并以索引键值的递增顺序扫描索引记录；

示例：（注意返回数据顺序）

```
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

语法：IINDEX_SS_DESC([<@查询块>] <表> [<索引>]) 或者 INDEX_SS_DESC([<@查询块>] <表> [(<索引字段列表>)])

描述：指示优化器通过跳跃索引扫描来访问表，并以索引键值的递减顺序扫描索引记录；

示例：（注意返回数据顺序）

```
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

语法：NO_INDEX_SS([<@查询块>] <表> [<索引>]) 或者 NO_INDEX_SS([<@查询块>] <表> [(<索引字段列表>)])

描述：禁止优化器使用跳跃索引扫描访问表。

示例：

```
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

语法：INDEX_RS_ASC([<@查询块>] <表> [<索引>]) 或者 INDEX_RS_ASC([<@查询块>] <表> [(<索引字段列表>)])

描述：指示优化器通过范围索引扫描来访问表，并以索引键值的递增顺序扫描索引记录；但如果谓词条件满足唯一键索引扫描的条件时，优化器还是会选择唯一键索引扫描。

示例：（注意返回数据顺序）

```
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

语法：INDEX_RS_DESC([<@查询块>] <表> [<索引>]) 或者 INDEX_RS_DESC([<@查询块>] <表> [(<索引字段列表>)])

描述：指示优化器通过范围索引扫描来访问表，并以索引键值的递减顺序扫描索引记录；但如果谓词条件满足唯一键索引扫描的条件时，优化器还是会选择唯一键索引扫描。

示例：（注意返回数据顺序）

```
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

语法：AND_EQUAL([<@查询块>] <表> [<索引 1> <索引 2> ...]) 或者 AND_EQUAL([<@查询块>] <表> [(<索引 1 字段>) (<索引 2 字段>) ...])

描述：指示优化器对由指定表上的两个或多个单字段索引获取 ROWID 数据集的取交集，并消除重复的 ROWID。

```
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

语法：INDEX_COMBINE([<@查询块>] <表> [<索引 1> ...]) 或者 AND_EQUAL([<@查询块>] <表> [(<索引 1 字段列表>) ...])

描述：指示优化器以位图计算方式访问表。如果指定了索引，则尝试从指定索引上获取位图数据，如果指定索引不是位图索引，则尝试进行位图转换。

```
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

语法：INDEX_JOIN([<@查询块>] <表> [<索引 1> <索引 2> ...]) 或者 INDEX_JOIN([<@查询块>] <表> [(<索引 1 字段列表>) (<索引 2 字段列表>) ...])

描述：指示优化器对由指定表上的两个或多个索引关联后访问表。

```
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

语法：BITMAP_TREE([<@查询块>] <表> AND(<索引 1>[<索引 2> ...])) 或者 BITMAP_TREE([<@查询块>] <表> AND((<索引 1 字段列表>)[(<索引 2 字段列表>) ...]))

描述：指示优化器将 ROWID 转换为位图，并做位图计算；

```
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

语法：USE_INVISIBLE_INDEXES([<@查询块>] <表> ([<索引 1>] ...)) 或者 USE_INVISIBLE_INDEXES([<@查询块>] <表> ([<索引列表 1>] ...))

描述：指示优化器使用不可见（INVISIBLE）的索引。

```
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

语法：NO_USE_INVISIBLE_INDEXES([<@查询块>] <表> ([<索引 1>] ...)) 或者 NO_USE_INVISIBLE_INDEXES([<@查询块>] <表> ([<索引列表 1>] ...))

描述：禁止优化器使用不可见（INVISIBLE）的索引。

```
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 |
-------------------------------------------------------------------------------
```

关联提示

NL_AJ

语法：NL_AJ([<子查询块>])

描述：指示优化器将主查询中的表与子查询中的表做嵌套循环反关联（Nested Loop Anti-Join）操作；如果提示出现在子查询中，则不需要参数指定查询块标识；

```
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

语法：HASH_AJ([<子查询块>])

描述：指示优化器将主查询中的表与子查询中的表做哈希反关联（Hash Anti-Join）操作；如果提示出现在子查询中，则不需要参数指定查询块标识；

```
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

语法：MERGE_AJ([<子查询块>])

描述：指示优化器将主查询中的表与子查询中的表做合并反关联（Merge Anti-Join）操作；如果提示出现在子查询中，则不需要参数指定查询块标识；

```
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

语法：USE_HASH([<@查询块>] <表 1> [<表 2>])

描述：指示优化器采用哈希关联。如果参数中仅指定一张表，则需要用 LEDING 提示来指定关联顺序。

```
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

语法：NO_USE_HASH([<@查询块>] <表 1> [<表 2> ...])

描述：指示优化器不要使用参数中指定的表作为哈希关联的内表；如果所有表都被指定，则不会使用哈希关联。

```
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

语法：USE_MERGE([<@查询块>] <表 1> [<表 2>])

描述：指示优化器采用合并关联。如果参数中仅指定一张表（内表），则需要用 LEDING 提示来指定关联顺序。

```
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

语法：NO_USE_MERGE([<@查询块>] <表 1> [<表 2> …])

描述：指示优化器不要使用参数中指定的表作为合并关联的内表；如果所有表都被指定，则不会使用合并关联。

```
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

语法：USE_NL([<@查询块>] <表 1> [<表 2>])

描述：指示优化器采用嵌套循环关联，并使用指定表为内表。如果参数中仅指定一张表（内表），则需要用 LEDING 提示来指定关联顺序。

```
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

语法：USE_NL_WITH_INDEX([<@查询块>] <表> [索引 1 …]) 或者 USE_NL_WITH_INDEX([<@查询块>] <表> [(<索引字段列表 1>) …])

描述：指示优化器采用嵌套循环关联，并使用指定表为内表，且对指定表的访问要通过索引访问。

```
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

语法：NO_USE_NL([<@查询块>] <表 1> [<表 2> ...])

描述：指示优化器不要使用参数中指定的表作为嵌套循环关联的内表；如果所有表都被指定，则不会使用嵌套循环关联。

```
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

语法：USE_MERGE_CARTESIANUSE_SEMI([<@查询块>] <表 1> [<表 2>])

描述：指示优化器采用笛卡尔合并关联。

示例（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

语法：LEADING([<@查询块>] <表 1> [<表 2> ...])

描述：指示优化器采用特定顺序关联表

见 USE_HASH 示例。

USE_ANTI

语法：USE_ANTII([<@查询块>] <表 1> [<表 2>])

描述：指示优化器采用反关联。仅在反关联并行查询的递归调用语句上观察到。

示例（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

语法：USE_SEMI([<@查询块>] <表 1> [<表 2>])

描述：指示优化器采用半关联。仅在半关联并行查询的递归调用语句上观察到。

示例（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

语法：NATIVE_FULL_OUTER_JOIN([<@查询块>])

描述：指示优化器采用基于哈希关联的真正完全外关联操作。

```
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

语法：NO_NATIVE_FULL_OUTER_JOIN([<@查询块>])

描述：禁止优化器采用的真正完全外关联操作。

```
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

语法：NO_CARTESIAN([查询块] <表 1> [<表 2> ...])

描述：禁止优化器产生对指定表的笛卡儿关联操作

```
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

语法：ORDERED

描述：指示优化器采用表在 FROM 子句后出现的顺序进行关联；

```
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

语法：PX_JOIN_FILTER(<表>)

描述：强制优化器采用位图过滤技术进行哈希外关联或者并行关联

```
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

语法：NO_PX_JOIN_FILTER(<表>)

描述：禁止优化器采用位图过滤技术进行哈希外关联或者并行关联

```
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

语法：NL_SJ([<子查询块>])

描述：指示优化器使用嵌套循环半关联.

```
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

语法：HASH_SJ([<子查询块>])

描述：指示优化器使用哈希半关联.

```
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

语法：MERGE_SJ([<子查询块>])

描述：指示优化器使用合并半关联.

```
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

语法：NO_SEMIJOIN([<子查询块>])

描述：禁止优化器使用半关联.

```
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

语法：SEMIJOIN([<子查询块>])

描述：指示优化器使用半关联，关联方式由优化器自己决定

```
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

语法：SEMIJOIN_DRIVER([<子查询块>])

描述：指示优化器在使用普通关联获取半关联结果时，首先驱动哪个子查询；

```
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

语法：SWAP_JOIN_INPUTS([<@查询块>] <表 1> [<表 2> ...])

描述：指示优化器允许交换表的哈希关联数据输入顺序。

```
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

语法：NO_SWAP_JOIN_INPUTS([<@查询块>] <表 1> [<表 2> ...])

描述：禁止优化器允许交换表的哈希关联数据输入顺序。

示例（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 |
-------------------------------------------------------------------------------------------------------
```

概要数据提示

QB_NAME

语法：QB_NAME(<有效字符串>)

描述：该提示为所在查询块指定标识符，该标识符可以用于其他提示的全局格式，指定对象所在位置；查询块被引用时的格式为：@<标识符> 或者 <对象名>@<标识符>

```
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

语法：DB_VERSION(<数据库版本>)

描述：这应该是一个被动的概要提示，指明该语句子游标的执行计划解析时的数据库版本。有助于定位因升级（如从 11.2.0.1 到 11.2.0.2）而产生的 SQL 问题。

在任何执行计划的概要数据中都能看到该提示。

IGNORE_OPTIM_EMBEDDED_HINTS

语法：IGNORE_OPTIM_EMBEDDED_HINTS

描述：使优化器忽视嵌入在 SQL 语句当中的提示————通常用于存储概要、SQL 优化配置等优化器辅助数据；

```
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

语法：IGNORE_WHERE_CLAUSE

描述：该提示会导致优化器忽略在其之后的其它嵌入提示。在许多内部递归调用的语句（如动态采样）当中，语句中嵌入的提示都是拼接起来的。在某些情况下，Oracle 不愿意一些提示其作用，该提示就会导致后续提示失效。

```
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

语法：NO_ACCESS([查询块] <视图>)

描述：表示优化器在分析某个查询块时，没有访问指定视图。即在优化该查询块时，没采用针对该视图的优化技术（如视图合并）。

```
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

语法：OPTIMIZER_FEATURES_ENABLE(<版本号>)

描述：指示优化器采用某个版本的特性。版本号为一个字符串（单引号匹配），每个版本可以指定的版本号都不相同，但基本上都是向上兼容的。DEFAULT 为采用当前系统默认值

```
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

语法：OPT_PARAM(<优化参数> 调整值)

描述：设置该语句的优化环境参数。

```
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

语法：OUTLINE([<@查询块>])

描述：为查询块建立概要节点。概要节点是优化器进行语句优化时用到的数据结构。

见其它示例当中的概要数据。

OUTLINE_LEAF

语法：OUTLINE_LEAF([<@查询块>])

描述：为查询块建立概要叶子节点。叶子节点将不再做进一步查询转换优化。

示例（以下示例中，概要叶子节点的建立导致查询块中的视图不能被合并）：

```
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

语法：RBO_OUTLINE

描述：指示优化器建立基于规则的概要数据结构。

参见 RULE 提示

查询转换提示

ANTIJOIN

语法：ANTIJOIN([<子查询块>])

描述：指示优化器将主查询中的表与子查询中的表做反关联（优化器自己决定选择哪一种关联方式）操作；如果提示出现在子查询中，则不需要参数指定查询块标识；

```
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

语法：COALESCE_SQ([<@查询块>])

描述：指示优化器进行子查询合并

```
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

语法：NO_COALESCE_SQ([<@查询块>])

描述：禁止优化器进行子查询合并

```
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

语法：ELIMINATE_JOIN([<@查询块>] <表>)

描述：指示优化器进行消除管理的查询转换

```
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

语法：NO_ELIMINATE_JOIN([<@查询块>] <表>)

描述：禁止优化器进行消除管理的查询转换

```
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

语法：ELIMINATE_OBY([<@查询块>] <表>)

描述：指示优化器进行消除 ORDER BY 的查询转换

示例：

```
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

语法：NO_ELIMINATE_OBY([<@查询块>] <表>)

描述：禁止优化器进行消除 ORDER BY 的查询转换

```
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

语法：EXPAND_GSET_TO_UNION([<@查询块>])

描述：指示优化器对语句进行集合分组查询重写；

```
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

语法：NO_EXPAND_GSET_TO_UNION([<@查询块>])

描述：禁止优化器对语句进行集合分组查询重写；

示例：

```
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

语法：EXPAND_TABLE([<@查询块>] <表>)

描述：指示优化器使用表扩张对分区表查询进行转换

```
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

语法：NO_EXPAND_TABLE([<@查询块>] <表>)

描述：禁止优化器使用表扩张对分区表查询进行转换

```
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

语法：STAR

描述：提示优化器采用星型查询（旧的方式，8i）。这种星型查询，是通过嵌套循环实现的，

```
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

语法：STAR_TRANSFORMATION([<@查询块>] [<事实表>] [SUBQUERIES(<维度表 1> <维度表 2>[<维度表 3>...])])

描述：指示优化器进行星型转换

```
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

语法：NO_STAR_TRANSFORMATION([<@查询块>])

描述：禁止优化器进行星型转换

```
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

语法：FACT([<@查询块>] <表>)

描述：指示优化器在进行星型转换时，采用指定表作为事实表

见 STAR_TRANSFORMATION 示例

NO_FACT

语法：NO_FACT([<@查询块>] <表>)

描述：优化器在进行星型转换时，禁止采用指定表作为事实表

见 STAR_TRANSFORMATION 示例

FACTORIZE_JOIN

语法：FACTORIZE_JOIN(<数据集>(<表>@<子查询 1> <表>@<子查询 2>[<表>@<子查询 3> ...]))

描述：指示优化器将 UNION/UNION-ALL 查询中的子查询合并为一个内联视图；

```
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

语法：NO_FACTORIZE_JOIN(<数据集>)

描述：禁止优化器将 UNION/UNION-ALL 查询中的子查询合并为一个内联视图；

```
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

语法：MATERIALIZE

描述：指示优化器将内联视图实体化————执行过程中会创建基于视图的临时表。

```
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

语法：INLINE

描述：禁止优化器将内联视图实体化；

```
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

语法：MERGE([<@查询块>] [<表>]) 或 MERGE([<视图>] [<表>])

描述：指示优化器对子查询或者视图进行合并转换。

```
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

语法：NO_MERGE([<@查询块>] [<表>]) 或 NO_MERGE([<视图>] [<表>])

描述：禁止优化器对子查询或者视图进行合并转换。

```
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

语法：NO_PRUNE_GSETS

描述：禁止优化器对集合分组查询进行裁剪

```
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

语法：NO_QUERY_TRANSFORMATION

描述：禁止一切查询转换的发生；

```
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

语法：OR_EXPAND([<@查询块>] <表> <字段 1> [<字段 2> ...])

描述：指示优化器采用“或”扩展查询转换。

```
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

语法：OUTER_JOIN_TO_INNER([<@查询块>])

描述：指示优化器进行查询转换，将外关联转换为内关联

```
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

语法：NO_OUTER_JOIN_TO_INNER

描述：禁止优化器进行查询转换，将外关联转换为内关联

```
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

语法：ELIMINATE_OUTER_JOIN([<@查询块>])

描述：指示优化器进行查询转换，消除外关联（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

语法：NO_ELIMINATE_OUTER_JOIN

描述：禁止优化器进行查询转换，消除外关联（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

语法：PLACE_DISTINCT

描述：指示优化器进行 DISTINCT 配置查询转换；

```
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

语法：NO_PLACE_DISTINCT

描述：禁止优化器进行 DISTINCT 配置查询转换；

```
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

语法：PLACE_GROUP_BY([<@查询块>] (<表>) [<临时视图编号>])

描述：指示优化器进行 GROUP BY 配置查询转换；

示例：

```
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

语法：NO_PLACE_GROUP_BY([<@查询块>] (<表>))

描述：禁止优化器进行 GROUP BY 配置查询转换；

```
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

语法：PRECOMPUTE_SUBQUERY

描述：指示优化器预先计算子查询，根据结果再对查询进行优化。这一提示可能会导致查询结果的变化。

示例（以下查询根据子查询的计算结果改变了执行计划，消除了关联操作）：

```
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

语法：PULL_PRED([<@查询块>] <视图> [<谓词位置 1> ...])

描述：指示优化器将视图中的复杂谓词提取到查询块当中；

```
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

语法：NO_PULL_PRED([<@查询块>] <视图> [<谓词位置 1> ...]))

描述：禁止优化器抽取视图的复杂谓词条件。如果没有指定谓词位置，则禁止进行复杂谓词提取转换。

```
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

语法：OLD_PUSH_PRED([<@查询块>] <视图>)

描述：指示优化器使用旧关联谓词推入技术；

```
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

语法：PUSH_PRED([<@查询块>] <视图> [<谓词位置 1> ...])

描述：指示优化器将关联谓词推入到视图当中；

```
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

语法：NO_PUSH_PRED([<@查询块>] <视图>)

描述：禁止优化器将关联谓词推入到视图当中；

```
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

语法：PUSH_SUBQ([<子查询块>])

描述：指示优化器在做优化时提前评估子查询

```
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

语法：NO_PUSH_SUBQ([<子查询块>])

描述：禁止优化器在做优化时提前评估子查询

```
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

语法：REWRITE([<@查询块>] [<物化视图>])

描述：指示优化器将查询块重写为对兼容的物化视图的查询；

```
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

语法：NO_REWRITE([<@查询块>])

描述：禁止优化器将查询块重写为对物化视图的查询；

```
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

语法：NO_MULTIMV_REWRITE

描述：禁止优化器将查询重写为对多个物化视图的查询（默认为允许）；

```
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

语法：NO_BASETABLE_MULTIMV_REWRITE

描述：禁止优化器将查询重写为对物化视图以及其基础表的复合查询（默认为允许）；

```
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

语法：REWRITE_OR_ERROR

描述：指示优化器将查询块重写为对兼容的物化视图的查询，重写失败则抛出错误；

```
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

语法：SET_TO_JOIN([<@查询块>])

描述：指示优化器将数据集合操作转换为关联查询；

```
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

语法：NO_SET_TO_JOIN([<@查询块>])

描述：禁止优化器将数据集合操作转换为关联查询；

```
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

语法：TRANSFORM_DISTINCT_AGG(<@查询块>)

描述：指示优化器做 DISTINCT 聚集函数转换

```
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

语法：NO_TRANSFORM_DISTINCT_AGG

描述：禁止优化器做 DISTINCT 聚集函数转换

```
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

语法：UNNEST([<子查询块>])

描述：指示优化器对指定子查询进行子查询反嵌套查询转换

```
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

语法：NO_UNNEST([<子查询块>])

描述：禁止优化器对指定子查询进行子查询反嵌套查询转换

```
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

语法：USE_CONCAT([<@查询块>])

描述：指示优化器使用拼接获取多个 OR 关系式连接的谓词条件结果

```
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

语法：NO_EXPAND([<@查询块>])

描述：禁止优化器使用拼接获取多个 OR 关系式连接的谓词条件结果

```
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

语法：USE_TTT_FOR_GSETS

描述：指示优化器将集合分组查询转换为对多个系统临时表的直接插入和加载操作。

```
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

语法：NO_ORDER_ROLLUPS

描述：未知。可能是在旧版本的优化器当中，控制优化器对 ROLLUP 语句进行查询转换，避免排序。

OPAQUE_XCANONICAL

语法：OPAQUE_XCANONICAL

描述：未知。估计是用于正则表达式的提示。

LIKE_EXPAND

语法：LIKE_EXPAND

描述：未知

统计数据提示

CARDINALITY

语法：CARDINALITY([<@查询块>] [<表>] <基数>)

描述：指定查询块或对象的基数大小

```
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

语法：CPU_COSTING

描述：指示优化器在代价估算时考虑 CPU 代价

```
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

语法：NO_CPU_COSTING

描述：禁止优化器在代价估算时考虑 CPU 代价

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

DBMS_STATS

语法：DBMS_STATS

描述：出现在 DBMS_STATS 包收集统计数据的递归调用语句上，告诉优化器该语句是收集统计数据用的，不做额外处理（如自动调优）

示例（以下是对 DBMS_STATS 收集表统计数据过程跟踪得到的语句）：

```
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

语法：DYNAMIC_SAMPLING([<@查询块>] [<表>] <采用级别>)

描述：指定查询块或者表的采样级别，从 0～10

```
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

语法：DYNAMIC_SAMPLING_EST_CDN([<@查询块>] <表>)

描述：指示优化器对已经存在统计数据的表也进行动态采样；

```
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

语法：GATHER_PLAN_STATISTICS

描述：指示 SQL 执行器在运行 SQL 时收集语句的统计数据；

示例：

```
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

语法：NO_STATS_GSETS

描述：不将对集合分组查询过程中产生的内部递归语句的运行统计数据计算在原语句的统计数据内。

```
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

语法：OPT_ESTIMATE([<@查询块>] <对象类型> <对象> <调整数据>=<数字>)

描述：指示优化器采用调整的对象的统计数据。其中，对象类型可以为 QUERY_BLOCK/TABLE/INDEX_FILTER/INDEX_SCAN/INDEX_SKIP_SCAN/JOIN；相应对象为<@查询块>/<表>/<索引所在表 索引>/JOIN(<关联对象 1> <关联对象 2>)；调整数据可以为 ROWS/SCALE_ROWS/MIN/MAX。OPT_ESTIMATE/COLUMN_STATS/INDEX_STATS/TABLE_STATS 提示通常是用于的 SQL 优化配置的辅助数据。

```
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

语法：COLUMN_STATS(<表> <字段> <SCALE|NULL> [<调整统计数据 1>=<数字 1> ...])

描述：指示优化器使用调整的字段统计数据优化语句。其中，SCALE|NULL 表示是否对统计数据进行放缩；可调整的统计数据包括：length（字段长度）、distinct（唯一值数）、nulls（空值数）、min（最小值）和 max（最大值）。

```
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

语法：INDEX_STATS(<表> <索引> <SCALE|NULL> [<调整统计数据 1>=<数字 1> ...])

描述：指示优化器使用调整的索引统计数据优化语句。其中，SCALE|NULL 表示是否对统计数据进行放缩；可调整的统计数据包括：blocks（叶子数据块数）、index_rows（索引记录数）、keys（索引键值数）和 clustering_factor（簇集因子）。

```
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

语法：TABLE_STATS(<表> <SCALE|NULL> [<调整统计数据 1>=<数字 1> ...])

描述：指示优化器使用调整的表统计数据优化语句。其中，SCALE|NULL 表示是否对统计数据进行放缩；可调整的统计数据包括：blocks（数据块数）和 rows（记录数）。

```
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 |
-------------------------------------------------------------------------------
```

优化器提示

NUM_INDEX_KEYS

语法：NUM_INDEX_KEYS([<@查询块>] <表> <索引> <索引键数>)

描述：指示优化器在进行“INLIST ITERATOR”操作时，使用多少个索引键来访问索引

```
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

语法：BIND_AWARE

描述：指示优化器对绑定变量值的变化敏感，使得优化器利用扩展游标共享特性对语句进行自动优化；

```
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
```

```
------------ ------------------------------------------------------------ -------------
6asc1cwrbw925 select /*+bind_aware*/* from t_objects where owner = :owner    Y
```

NO_BIND_AWARE

语法：NO_BIND_AWARE

描述：禁止优化器对绑定变量值的变化敏感，使得优化器不会利用扩展游标共享特性对语句进行自动优化；

```
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

语法：CURSOR_SHARING_EXACT

描述：指示优化器在使用共享游标时，对语句进行精确匹配，不会将变量值转换为绑定变量；

```
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

语法：DML_UPDATE

描述：用于更新位图关联索引的递归调用语句（UPD_JOININDEX），指明是由 DML 语句导致的数据更新。

FBTSCAN

语法：FBTSCAN

描述：用于对表的闪回（Flashback）查询。指示优化器查询闪回表（FlashBack Table）而非数据表本身。通常用于闪回查询的内部递归调用语句。这一提示会导致逻辑结果的改变。

```
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

语法：ALL_ROWS

描述：指示优化器在优化语句时，以消耗最少资源的最佳吞吐量为优化目标

```
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

语法：FIRST_ROWS([<记录数>])

描述：指示优化器在优化语句时，以最高效地返回前面指定数量的记录为目标。

```
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

语法：RULE

描述：指示优化器基于规则（RBO）选择执行计划；

```
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

语法：CHOOSE

描述：指示优化器根据当前环境来选择优化模型；

```
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

语法：ORDERED_PREDICATES

描述：该提示指示优化器对除索引键值以外的谓词条件按照既定规则顺序评估代价，如果两个谓词符合相同规则，则按照其在 WHERE 子句中出现的顺序进行评估。这些顺序为：

1. 不存在用户自定义函数、类型方法以及子查询的谓词条件；
2. 存在用户自定义函数、类型方法的谓词条件，并且存在相关统计数据；
3. 存在用户自定义函数、类型方法的谓词条件，不存在相关统计数据；
4. 未在 WHERE 子句中出现的谓词条件，这种条件可能是由查询转换时产生的；
5. 存在子查询的谓词条件；

（比较以下两个执行计划的代价）

```
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

语法：SKIP_EXT_OPTIMIZER

描述：指示优化器跳过扩展优化器。Oracle 允许用户可以自定义统计数据、选择率和代价计算函数来对用户自定义函数或者域索引进行优化，以扩展优化器（CBO）。

SKIP_UNQ_UNUSABLE_IDX

语法：SKIP_UNQ_UNUSABLE_IDX([<@查询块>] <表> [<索引 1> ...]) 或者 SKIP_UNQ_UNUSABLE_IDX([<@查询块>] <表> [(<索引字段列表 1>) ...])

描述：指示优化器忽略表上状态为 UNUSABLE 的索引

```
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
```

语句运行提示

APPEND

语法：APPEND

描述：指示优化器以追加方式向表直接插入数据；

```
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

语法：APPEND_VALUES

描述：指示优化器以追加方式向表直接插入数据；该提示仅支持 INSERT ... VALUES 形式的语句；

```
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

描述：禁止优化器以追加方式向表直接插入数据；

```
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

语法：NLJ_BATCHING

描述：指示优化器以嵌套循环关联批量读取的方式访问表

```
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 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 |
-------------------------------------------------------------------------------------------------------
```

NO_NLJ_BATCHING

语法：NO_NLJ_BATCHING([<@查询块>] <表>)

描述：禁止优化器以嵌套循环关联批量读取的方式访问表

```
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

语法：NLJ_PREFETCH([<@查询块>] <表>)

描述：指示优化器以嵌套循环关联预提取的方式访问表

```
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

语法：NO_NLJ_PREFETCH([<@查询块>] <表>)

描述：禁止优化器以嵌套循环关联预提取的方式访问表

```
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

语法：CACHE([<@查询块>] <表>)

描述：指示优化器将全表扫描读取的数据块放在 LRU 链表的 MRU 端

```
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

语法：NOCACHE([<@查询块>] <表>)

描述：指示优化器将全表扫描读取的数据块放在 LRU 链表的 LRU 端

```
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

语法：CACHE_TEMP_TABLE([<@查询块>] <表>)

描述：指示优化器将扫描读取的临时表的数据块放在 LRU 链表的 MRU 端，通常这种提示出现在递归调用的语句当中。

```
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

语法：NO_LOAD

描述：禁止将数据直接载入表

```
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

语法：NO_SUBSTRB_PAD

描述：对多字节字符集（如 UTF8）中的多字节字符（如中文）串使用函数 SUBSTRB 时，禁止进行字节补齐。这一提示会导致逻辑结果的变化。

```
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

语法：RESULT_CACHE

描述：指示数据库将查询（或者查询片段）结果保存在缓存当中，使得缓存中数据能被后续的查询使用；

```
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

语法：NO_RESULT_CACHE

描述：禁止数据库将查询（或者查询片段）结果保存在缓存当中；

```
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

语法：SYS_DL_CURSOR

描述：这个提示在运行 SQL*Loader 直接（Direct=TRUE）加载数据时，会在相关 INSERT 语句上加上。指示采用直接加载（Direct Load）游标，对数据进行批量插入。

示例（在运行 SQL*Loader 直接加载数据后，从共享缓存中可以看到以下语句）：

```
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

语法：TRACING(verify <数字> strip <数字>)

描述：未知。可能是用于跟踪 ASM 的条带化过程的提示。使用该提示后，执行计划会多出一个“MONITORING STRIP”操作，并且不会有数据返回。

```
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 |
-------------------------------------------------------------------------------
```

数据操作提示

CHANGE_DUPKEY_ERROR_INDEX

语法：CHANGE_DUPKEY_ERROR_INDEX(<表>, <索引>) 或者 CHANGE_DUPKEY_ERROR_INDEX(<表>, <索引字段列表>)

描述：插入记录违反唯一性约束时，抛出 ORA-38911 而非 ORA-00001 错误。

```
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

语法：IGNORE_ROW_ON_DUPKEY_INDEX(<表> <唯一索引>) 或者 IGNORE_ROW_ON_DUPKEY_INDEX(<表> <唯一索引字段列表>)

描述：指示 SQL 执行器向存在唯一约束的表中插入数据时，忽略重复数据

```
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

语法：RETRY_ON_ROW_CHANGE

描述：当 UPDATE 或 DELETE 操作的事务在提交之前，如果其缓存住的数据发生了变化，该提示使得事务被重新启动————这可能会导致表上的触发器被再次执行；

```
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
```

当不使用提示时，重复上述操作，得到以下结果：

```
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
```

以上三个提示会引起逻辑结果变化的提示。

层次查询提示

CONNECT_BY_CB_WHR_ONLY

语法：CONNECT_BY_CB_WHR_ONLY([<@查询块>])

描述：未知。可能是指示优化器进行 CONNECT BY 操作时，仅结合 WHERE 条件关联数据进行查询转换。优化器参数“_optimizer_connect_by_cb_whr_only”控制这一特性。

NO_CONNECT_BY_CB_WHR_ONLY

语法：NO_CONNECT_BY_CB_WHR_ONLY([<@查询块>])

描述：未知。可能是禁止优化器进行 CONNECT BY 操作时，结合 WHERE 条件关联数据进行查询转换.优化器参数“_optimizer_connect_by_cb_whr_only”控制这一特性。

CONNECT_BY_COMBINE_SW

语法：CONNECT_BY_COMBINE_SW([<@查询块>])

描述：指示优化器进行 CONNECT BY 操作时，结合 START WITH 的条件关联数据

```
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

语法：NO_CONNECT_BY_COMBINE_SW([<@查询块>])

描述：禁止优化器进行 CONNECT BY 操作时，结合 START WITH 的条件关联数据

```
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

语法：CONNECT_BY_COST_BASED([<@查询块>])

描述：指示优化器基于代价对 CONNECT BY 进行转换

示例（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

语法：NO_CONNECT_BY_COST_BASED([<@查询块>])

描述：禁止优化器基于代价对 CONNECT BY 进行转换

示例（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

语法：CONNECT_BY_ELIM_DUPS([<@查询块>])

描述：指示优化器进行 CONNECT BY 操作时，在关联数据时消除重复值；

```
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

语法：NO_CONNECT_BY_ELIM_DUPS([<@查询块>])

描述：禁止优化器进行 CONNECT BY 操作时，在关联数据时消除重复值；

```
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

语法：CONNECT_BY_FILTERING([<@查询块>])

描述：指示优化器进行 CONNECT BY 操作时，在获取数据时做过滤

```
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

语法：NO_CONNECT_BY_FILTERING([<@查询块>])

描述：指示优化器进行 CONNECT BY 操作时，在关联数据时做过滤

```
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

语法：COST_XML_QUERY_REWRITE

描述：指示优化器（采用关系型对象查询重写技术）对含有 XML 对象的查询进行基于代价的查询重写。默认情况下，优化器采用基于规则的方式进行查询重写。对优化器解析过程做 19027 事件跟踪，可以看到重写跟踪或者不能重写的原因。

```
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

语法：NO_COST_XML_QUERY_REWRITE

描述：禁止优化器对含有 XML 对象的查询进行基于代价的查询重写。

见上例。

FORCE_XML_QUERY_REWRITE

语法：FORCE_XML_QUERY_REWRITE

描述：强制优化器对 XML 查询操作进行重写

```
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

语法：NO_XML_QUERY_REWRITE

描述：禁止优化器对 XML 查询操作进行重写

```
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

语法：XML_DML_RWT_STMT

描述：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

语法：NO_XML_DML_REWRITE

描述：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

语法：XMLINDEX_REWRITE

描述：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

语法：NO_XMLINDEX_REWRITE

描述：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

语法：XMLINDEX_REWRITE_IN_SELECT

描述：指示优化器将 XML 索引用于重写 SELECT 子句

```
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

语法：NO_XMLINDEX_REWRITE_IN_SELECT

描述：指示优化器将 XML 索引仅用于 WHERE 子句过滤而不用于重写 SELECT 子句

```
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

语法：CHECK_ACL_REWRITE

描述：未知。可能是指示优化器在对 XML 查询重写时，检查访问控制列表（Access Control List）

NO_CHECK_ACL_REWRITE

语法：NO_CHECK_ACL_REWRITE

描述：未知。可能是指示优化器在对 XML 查询重写时，不检查访问控制列表（Access Control List）

INLINE_XMLTYPE_NT

语法：INLINE_XMLTYPE_NT

描述：未知。可能是指示优化器将语句中内联 XMLTYPE 数据转换为嵌套表（Nested Table）。

XMLINDEX_SEL_IDX_TBL

语法：XMLINDEX_SEL_IDX_TBL

描述：未知。

分布式查询提示

DRIVING_SITE

语法：DRIVING_SITE([<远程表>])

描述：指示优化器选择那个数据库作为分布式查询中的驱动站点，即将语句放在该站点上执行。未指定参数时，采用本地数据库。

```
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

语法：REMOTE_MAPPED([<远程数据库链接>])

描述：在分布式查询中，指示优化器选择那个数据库的进行远程映射。作用和 DRIVING_SITE 类似。未指定参数时，采用本地数据库。

```
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

语法：OPAQUE_TRANSFORM

描述：在分布式查询中，使用 INSERT … SELECT … FROM 语句从远程数据库查询数据插入本地数据库时，在远程数据库上执行的递归查询语句上会加上该提示，使得分布式数据库之间的兼容类型数据被透明传输。

（本地数据库，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.
```

（远程数据库，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_NAME",
"LIO" FROM "T_OBJECTS" "T_OBJECTS"
```

并行查询提示

STATEMENT_QUEUING

语法：STATEMENT_QUEUING

描述：在自动并行度模式（11gR2 特性）下，使语句在并行资源不足时，进入等待队列，等到能获取到资源时继续运行；

自动并行度模式下，当存在多个并行查询同时在运行时，如果并行资源不足那么加了该提示的语句就会进入队列等待，此时，通过监控视图 V$SQL_MONITOR 可以看到其状态为 QUEUE。

NO_STATEMENT_QUEUING

语法：NO_STATEMENT_QUEUING

描述：在自动并行度模式（11gR2 特性）下，即使在并行资源不足时，也继续运行语句，这就可能会导致其它资源（如 CPU）的争用与等待；

当参数“_parallel_statement_queuing”被设置为 TRUE（默认为 FALSE），只有加上该提示的语句在并行资源紧张时不会进入队列。

GBY_PUSHDOWN

语法：GBY_PUSHDOWN([<@查询块>])

描述：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

语法：NO_GBY_PUSHDOWN([<@查询块>])

描述：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

语法：HWM_BROKERED

描述：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

语法：NO_QKN_BUFF

描述：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

语法：PARALLEL_INDEX([查询块] <表> <索引 1> [<索引 2> ...] <并行度>) 或者 PARALLEL_INDEX([查询块] <表> (<索引字段列表 1>) [(<索引字段列表 2>) ...] <并行度>)

描述：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

语法：NO_PARALLEL_INDEX([查询块] <表> <索引 1> [<索引 2> ...]) 或者 NO_PARALLEL_INDEX([查询块] <表> (<索引字段列表 1>) [(<索引字段列表 2>) ...])

描述：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

语法：PQ_DISTRIBUTE([<@查询块>] <表> <分发方式>) 或者 PQ_DISTRIBUTE([<@查询块>] <表> <内分发方式> <外分发方式>)

描述：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

语法：PARALLE([[<@查询块>] <表>] [<并行度>])

描述：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

语法：NO_PARALLEL(<表>)

描述：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

语法：SHARED(<表> [并行度])

描述：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

语法：NOPARALLEL([[<@查询块>] <表>])

描述：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

语法：PQ_MAP(<表>)

描述：未知。可能是用于并行查询的提示。

PQ_NOMAP

语法：PQ_NOMAP(<表>)

描述：未知。可能是用于并行查询的提示。

PRESERVE_OID

语法：PRESERVE_OID

描述：未知。可能是用于并行查询的提示。

SYS_PARALLEL_TXN

语法：SYS_PARALLEL_TXN

描述：未知。可能是用于并行查询的递归调用语句上的。

CUBE_GB

语法：CUBE_GB

描述：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

语法：GBY_CONC_ROLLUP

描述：未知。可能是用于 GROUP BY ROLLUP 并行查询的内部递归查询

PIV_GB

语法：PIV_GB

描述：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

语法：TIV_GB

描述：未知。出现在并行查询的内部递归查询语句上

TIV_SSF

语法：TIV_SSF

描述：未知。出现在并行查询的内部递归查询语句上

PIV_SSF

语法：PIV_SSF

描述：未知。出现在并行查询的内部递归查询语句上

RESTORE_AS_INTERVALS

语法：RESTORE_AS_INTERVALS

描述：未知。出现在并行查询的内部递归查询语句上

SAVE_AS_INTERVALS

语法：SAVE_AS_INTERVALS

描述：未知。出现在并行查询的内部递归查询语句上

SCN_ASCENDING

语法：SCN_ASCENDING

描述：未知。出现在并行查询的内部递归查询语句上

模型化语句提示

MODEL_MIN_ANALYSIS

语法：MODEL_MIN_ANALYSIS

描述：用于模型化语句查询转换的提示。使得优化器对模型化语句的主查询做最少的查询转换分析。

分别对以下两条语句（有、无该提示）进行优化器过程跟踪：

```
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));
```

通过比较可以发现当使用该提示时，优化器为对模型化主语句做简单谓词推入的查询转换分析：

```
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

语法：MODEL_NO_ANALYSIS

描述：用于模型化语句查询转换的提示。使得优化器不对模型化语句的主查询做查询转换分析。可以对比有无该提示的优化器跟踪记录观察其影响。

MODEL_PUSH_REF

语法：MODEL_PUSH_REF

描述：未知。可能是用于模型化语句查询转换的提示，指示优化器将主模型中的谓词条件推入引用模型当中。

NO_MODEL_PUSH_REF

语法：NO_MODEL_PUSH_REF

描述：未知。可能是用于模型化语句查询转换的提示，禁止优化器将主模型中的谓词条件推入引用模型当中。

MODEL_COMPILE_SUBQUERY

语法：MODEL_COMPILE_SUBQUERY

描述：未知。可能是用于模型化语句查询转换的提示。

MODEL_DONTVERIFY_UNIQUENESS

语法：MODEL_DONTVERIFY_UNIQUENESS

描述：未知。可能是用于模型化语句查询转换的提示。

MODEL_DYNAMIC_SUBQUERY

语法：MODEL_DYNAMIC_SUBQUERY

描述：未知。可能是用于模型化语句查询转换的提示。

分区提示

X_DYN_PRUNE

语法：X_DYN_PRUNE

描述：指示优化器通过运行时刻子查询的结果对分区进行动态裁剪。

```
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 |
-----------------------------------------------------
```

上述语句的 10128 跟踪信息中，可以看到以下记录：

```
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

语法：SUBQUERY_PRUNING([<@查询块>] <表> PARTITION)

描述：指示优化器使用子查询对分区表进行分区裁剪。

SUBQUERY_PRUNING/NO_SUBQUERY_PRUNING 是用于控制执行计划中否采用子查询裁剪的概要数据，而 X_DYN_PRUNE 则用于执行时是否进行子查询裁剪。

```
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 |
-----------------------------------------------------
```

从 10128 事件跟踪信息中可以看到以下记录：

```
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

语法：NO_SUBQUERY_PRUNING([<@查询块>] <表>)

描述：禁止优化器使用子查询对分区表进行分区裁剪

```
HELLODBA.COM>exec sql_explain('select /*+NO_SUBQUERY_PRUNING(0 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        |
-----------------------------------------------------------
```

其它类型提示

RELATIONAL

语法：RELATIONAL([<@查询块>] <表>)

描述：将对象转换为关系型表进行查询，等同于在对象上增加了 RELATIONAL 函数。这一提示会导致查询结果的变化。

```
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"
... ...
```

直接使用该提示在 DML 语句上会导致抛出 ORA-22837 错误。

MONITOR

语法：MONITOR

描述：强制语句的运行状况被监控，不管它是否满足自动监控的前提条件（并行查询或者运行时间超过 5 秒）；

```
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

语法：NO_MONITOR

描述：强制语句的运行状况不被监控，不管它是否满足自动监控的前提条件（并行查询或者运行时间超过 5 秒）；

```
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

语法：NESTED_TABLE_FAST_INSERT

描述：使用快速方式向嵌套表插入数据。通过 10046 跟踪可以看到，未使用提示时，是逐条记录插入；使用提示后，则是批量插入。

```
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

语法：NESTED_TABLE_GET_REFS

描述：该提示可以使用户直接访问嵌套对象。

```
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

语法：NESTED_TABLE_SET_SETID

描述：该提示可以使用户直接访问嵌套对象。NESTED_TABLE_GET_REFS 和 NESTED_TABLE_SET_SETID 应该是通过两种不同的技术使得嵌套对象可以被直接访问。

```
HELLODBA.COM>select /*+ NESTED_TABLE_SET_SETID */count(*) from T_NT_B;
```

```
  COUNT(*)
----------
     72116
```

NO_MONITORING

语法：NO_MONITORING

描述：禁止监控语句中的谓词字段使用情况，即数据字典 col_usage$不会因该语句而被更新。

```
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

语法：NO_SQL_TUNE

描述：禁止自动调优组件对语句进行调优；

```
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

语法：RESTRICT_ALL_REF_CONS

描述：在事务中暂时限制所有外键约束的级联（CASCADE）递归方法；

```
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

语法：USE_HASH_AGGREGATION([<@查询块>])

描述：指示优化器使用哈希进行聚合计算。

```
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

语法：NO_USE_HASH_AGGREGATION([<@查询块>])

描述：禁止优化器使用哈希进行聚合计算。

```
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

语法：BYPASS_RECURSIVE_CHECK

描述：未知。可能是使编译器不做递归检查，我们观察到做物化视图刷新时，会加在递归调用语句。

示例：

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

Session altered.

HELLODBA.COM>exec dbms_mview.refresh(list => 'MV_TABLES');

PL/SQL procedure successfully completed.
```

跟踪文件中可以找到相应语句。

```
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
```

由于对象物化视图是非 FOR UPDATE 属性的视图，直接执行上述语句会抛 ORA-01732 错误。

BYPASS_UJVC

语法：BYPASS_UJVC

描述：未知，可能是使编译器对更新语句不做关联视图唯一性约束检查。我们观察到做物化视图刷新时，会加在递归调用语句。直接使用不起作用，仍然会抛 ORA-01779 错误

从上例中的跟踪文件可以找到以下语句：

```
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

语法：DOMAIN_INDEX_FILTER([<@查询块>] <表> [(<索引>)]) 或者 DOMAIN_INDEX_FILTER([<@查询块>] <表> [(<索引字段列表>)])

描述：指示优化器将过滤谓词推入复合域索引（Composite Domain Index）当中

```
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

语法：NO_DOMAIN_INDEX_FILTER([<@查询块>] <表> [(<索引>)]) 或者
NO_DOMAIN_INDEX_FILTER([<@查询块>] <表> [(<索引字段列表>)])

描述：禁止优化器将过滤谓词推入复合域索引（Compisite Domain Index）当中

```
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

语法：DOMAIN_INDEX_SORT

描述：指示优化器将排序字段推入复合域索引（Compisite Domain Index）当中

```
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

语法：NO_DOMAIN_INDEX_ SORT

描述：禁止优化器将排序字段推入复合域索引（Compisite Domain Index）当中

```
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

语法：DST_UPGRADE_INSERT_CONV

描述：在使用 DBMS_DST 包对数据库时区进行升级过程中，如果存在含有带时区的时间戳类型（TIMESTAMP WITH TIME ZONE）的字段的表没有被升级，该提示会指示优化器在修改该字段时，向其添加一个内部函数（ORA_DST_CONVERT(INTERNAL_FUNCTION())）

NO_DST_UPGRADE_INSERT_CONV

语法：NO_DST_UPGRADE_INSERT_CONV

描述：在使用 DBMS_DST 包对数据库时区进行升级过程中，如果存在含有带时区的时间戳类型（TIMESTAMP WITH TIME ZONE）的字段的表没有被升级，该提示会禁止优化器在修改该字段时，向其添加一个内部函数（ORA_DST_CONVERT(INTERNAL_FUNCTION())）

STREAMS

语法：STREAMS

描述：未知。可能是指示数据是以“流”的方式传输。

需要 Oracle“流”（STREAMS）组件。

DEREF_NO_REWRITE

语法：DEREF_NO_REWRITE(<@查询块>)

描述：未知，可能是用于禁止对“BUILD DEFERRED”的物化视图进行查询重写，无法确定。

MV_MERGE

语法：MV_MERGE

描述：未知。可能是用于优化 CUBE 查询。

EXPR_CORR_CHECK

语法：EXPR_CORR_CHECK

描述：未知。可能是指示优化器在解析使用表达式过滤器（Expression Filter）的语句，做相互关联检查。

需要表达式过滤器组件

INCLUDE_VERSION

语法：INCLUDE_VERSION

描述：未知。该提示出现在高级复制的内部 SQL 语句当中。可能是用于保持不同版本数据库之间的复制的兼容性。

VECTOR_READ

语法：VECTOR_READ

描述：未知。可能是用于控制哈希关联的位矢量图过滤。

VECTOR_READ_TRACE

语法：VECTOR_READ_TRACE

描述：未知。可能是用于控制哈希关联的位矢量图过滤。

USE_WEAK_NAME_RESL

语法：USE_WEAK_NAME_RESL

描述：未知。可能是指示优化器解析语句时使用内部名字而非用户定义的名字来查找资源位置（Resource Location）。通常出现在内部递归调用（如收集统计数据、表达式过滤器）的语句上；

```
    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

语法：NO_PARTIAL_COMMIT

描述：未知。可能是用于阻止内部递归事务的局部提交，使其与当前事务同时提交。从对存在嵌套对象的表（如基于 XML-Schemal 的表、嵌套表等）的数据维护操作的递归调用跟踪中可以看到，该提示加在维护其嵌套的表的语句上；

```
    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;

... ...
```

跟踪记录中可以找到以下语句

```
INSERT /*+ NO_PARTIAL_COMMIT REF_CASCADE_CURSOR */ INTO  "DEMO"."T_NT_B"
("NESTED_TABLE_ID","COLUMN_VALUE")  VALUES(:1, :2)
```

REF_CASCADE_CURSOR

语法：REF_CASCADE_CURSOR

描述：未知。可能是用于阻止内部递归事务的局部提交，使其与当前事务同时提交。从对存在嵌套对象的表（如基于 XML-Schemal 的表、嵌套表等）的数据维护操作的递归调用跟踪中可以看到，该提示加在维护其嵌套的表的语句上；

见 NO_PARTIAL_COMMIT 示例

NO_REF_CASCADE

语法：NO_REF_CASCADE

描述：未知。可能是用于禁止内部递归语句使用级联游标。

SQLLDR

语法：SQLLDR

描述：未知。可能是运行 SQL*Loader 时加上的提示。

SYS_RID_ORDER

语法：SYS_RID_ORDER

描述：未知。可能是用于物化视图维护操作时的递归调用语句上的。

OVERFLOW_NOMOVE

语法：OVERFLOW_NOMOVE

描述：未知。估计是用于禁止分区表分裂时或者索引组织表的非索引键字段溢出致其它存储段时的数据迁移。

LOCAL_INDEXES

语法：LOCAL_INDEXES

描述：未知

MERGE_CONST_ON

语法：MERGE_CONST_ON

描述：未知

QUEUE_CURR

语法：QUEUE_CURR

描述： 未知。可能是用于 Oracle 高级队列（Advanced Queue）的提示。

CACHE_CB

语法：CACHE_CB([<@查询块>] <表>)

描述：未知。用于高级队列（Advanced Queue）的提示

从对 DBMS_AQ.DEQUEUUE（delivery_mode 为 PERSISTENT）过程跟踪可以看到以下语句：

```
delete /*+ CACHE_CB("QUETABLET") */ from "DEMO"."QUETABLET" where rowid = :1;
```

QUEUE_ROWP

语法：QUEUE_ROWP

描述： 未知。可能是用于 Oracle 高级队列（Advanced Queue）的提示。

BUFFER

语法：BUFFER

描述：未知。可能用于高级队列（Advanced Queue）的提示

NO_BUFFER

语法：NO_BUFFER

描述：未知。可能用于高级队列用于高级队列（Advanced Queue）的提示

BITMAP

语法：BITMAP

描述：未知。