收获，不止 SQL 优化

第 三 章

读懂执行计划

E-Mail:45240040@qq.com

目录

1.获取执行计划的方法有哪几种，各有什么优缺点，如何选择

1.1 环境准备

```
SQL> CREATE TABLE t1 (
  2       id NUMBER NOT NULL,
  3       n NUMBER,
  4       contents VARCHAR2(4000)
  5     )
  6     ;

表已创建。

SQL> CREATE TABLE t2 (
  2       id NUMBER NOT NULL,
  3       t1_id NUMBER NOT NULL,
  4       n NUMBER,
  5       contents VARCHAR2(4000)
  6     )
  7     ;

表已创建。
```

```
SQL> execute dbms_random.seed(0);

PL/SQL 过程已成功完成。

SQL> INSERT INTO t1
  2       SELECT  rownum,  rownum, dbms_random.string('a', 50)
  3         FROM dual
  4       CONNECT BY level <= 1000
  5        ORDER BY dbms_random.random;

已创建1000行。

SQL> INSERT INTO t2 SELECT rownum, rownum, rownum, dbms_random.string('b', 50)
  2    FROM dual CONNECT BY level <= 100000
  3      ORDER BY dbms_random.random;

已创建100000行。

SQL> commit;

提交完成。

SQL> CREATE INDEX t1_n ON t1 (n);

索引已创建。

SQL> CREATE INDEX t2_t1_id ON t2(t1_id);

索引已创建。
```

1.2 explain plan for 的方式

```
SQL> explain plan for
  2  SELECT  *
  3  FROM t1, t2
  4  WHERE t1.id = t2.t1_id
  5  AND t1.n in(18,19);

已解释。

SQL> select * from table(dbms_xplan.display());

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

-------------------------------------------------------------------------------
| Id  | Operation                      | Name     | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |          |     2 |  8138 |     8   (0)|
|   1 |  TABLE ACCESS BY INDEX ROWID   | T2       |     1 |  2041 |     2   (0)|
|   2 |   NESTED LOOPS                 |          |     2 |  8138 |     8   (0)|
|   3 |    INLIST ITERATOR             |          |       |       |            |
|   4 |     TABLE ACCESS BY INDEX ROWID| T1       |     2 |  4056 |     4   (0)|
|   5 |      INDEX RANGE SCAN          | T1_N     |     4 |       |     2   (0)|
|   6 |    INDEX RANGE SCAN            | T2_T1_ID |     1 |       |     1   (0)|
-------------------------------------------------------------------------------

Note
-----
   - 'PLAN_TABLE' is old version

已选择16行。
```

优点：

- 无需真正执行，快捷方便

缺陷：

- 不能输出运行时的相关统计信息（产生多少逻辑读，多少次递归调用，多少次物理读的情况）；
- 无法判断是处理了多少行；
- 无法判断表被访问了多少次。

1.3 set autotrace on 方式

```
SQL> set autotrace on
SQL> SELECT  *
  2  FROM t1, t2
  3  WHERE t1.id = t2.t1_id
  4  AND t1.n in(18,19);

        ID          N    实际执行输出了结果
---------- ----------
CONTENTS
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
        ID      T1_ID          N
---------- ---------- ----------
CONTENTS
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
        18          18
yFfYpcNtsxEpvWUOceJmHtvpCEbcJTUTeKExdHlGlwoIFEgmOo
        18          18          18
OWCPIKKFQTMBZAVJBUGPOGOZGPHCMTGFDJTMCIZXRFVXYCATTY

        19          19
jAOYAJHBzwhNWHImZeFUOaxvMycbQsCLKhsUnSFwZpyctEahjK
        19          19          19
BEQUUMJSUDRLTCGOCIUHSZCNNJOTZOCEQKBZPSLHGAKRTLJAHX
```

```
执行计划
----------------------------------------------------------

--------------------------------------------------------------------------------
| Id  | Operation                      | Name     | Rows  | Bytes | Cost (%CPU)|
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |          |     2 |  8138 |     8   (0)|
|   1 |  TABLE ACCESS BY INDEX ROWID   | T2       |     1 |  2041 |     2   (0)|
|   2 |   NESTED LOOPS                 |          |     2 |  8138 |     8   (0)|
|   3 |    INLIST ITERATOR             |          |       |       |            |
|   4 |     TABLE ACCESS BY INDEX ROWID| T1       |     2 |  4056 |     4   (0)|
|   5 |      INDEX RANGE SCAN          | T1_N     |     4 |       |     2   (0)|
|   6 |    INDEX RANGE SCAN            | T2_T1_ID |     1 |       |     1   (0)|
--------------------------------------------------------------------------------

Note
-----
   - 'PLAN_TABLE' is old version


统计信息
----------------------------------------------------------
          0  recursive calls
          0  db block gets
         14  consistent gets
          0  physical reads
          0  redo size
        994  bytes sent via SQL*Net to client
        384  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          2  rows processed
```

优点：

- 可以输出运行时的相关统计信息（产生多少逻辑读，多少次递归调用，多少次物理读的情况）；
- 虽然必须要等语句执行完毕后才可以输出执行计划，但是可以有 traceonly 开关来控制返回结果不打屏输出。

缺陷：

- 必须要等到语句真正执行完毕后，才可以出结果；
- 无法看到表被访问了多少次。

1.4statistics level=all 的方式

```
SQL> alter session set statistics_level=all ;

会话已更改。

SQL> SELECT  * FROM t1, t2  WHERE t1.id = t2.t1_id  AND t1.n in(18,19);

        ID          N
---------- ----------
CONTENTS
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
        ID      T1_ID          N
---------- ---------- ----------
CONTENTS
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
        18          18
yFfYpcNtsxEpvWUQceJmHtvpCEbcJTUTeKExdHlGlwoIFEgmOo
```

语句被执行

```
SQL> select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
----------
SQL_ID  2a1casv455gxh, child number 0
-------------------------------------
SELECT  * FROM t1, t2  WHERE t1.id = t2.t1_id  AND t1.n in(18,19)

Plan hash value: 128660979

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name      | Starts | E-Rows | A-Rows |   A-Time    | Buffers |
-----------------------------------------------------------------------------------------------------------
|   1 |  TABLE ACCESS BY INDEX ROWID   | T2        |      1 |      1 |      2 |00:00:00.01 |      14 |
|   2 |   NESTED LOOPS                 |           |      1 |      2 |      5 |00:00:00.01 |      12 |
|   3 |    INLIST ITERATOR             |           |      1 |        |      2 |00:00:00.01 |       7 |
|   4 |     TABLE ACCESS BY INDEX ROWID| T1        |      2 |      2 |      2 |00:00:00.01 |       7 |
|*  5 |      INDEX RANGE SCAN          | T1_N      |      2 |      4 |      2 |00:00:00.01 |       5 |
|*  6 |    INDEX RANGE SCAN            | T2_T1_ID  |      2 |      1 |      2 |00:00:00.01 |       5 |
-----------------------------------------------------------------------------------------------------------

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

   5 - access(("T1"."N"=18 OR "T1"."N"=19))
   6 - access("T1"."ID"="T2"."T1_ID")

Note
-----
   - dynamic sampling used for this statement


已选择27行。
```

优点：

✧ 可以清晰的从 STARTS 得出表被访问多少。

✧ 可以清晰的从 E-ROWS 和 A-ROWS 中得到预测的行数和真实的行数，从而可以准确判断 Oracle 评估是否准确。

✧ 虽然没有专门的输出运行时的相关统计信息，但是执行计划中的 BUFFERS 就是真实的逻辑读的多少。

缺陷：

✧ 必须要等到语句真正执行完毕后，才可以出结果。

✧ 无法控制记录输屏打出，不像 autotrace 有 traceonly 可以控制不将结果打屏输出。

✧ 看不出递归调用的次数，看不出物理读的多少（不过逻辑读才是重点）。

1.5 通过 sql_id 提取执行计划

```
SQL> select  * from table(dbms_xplan.display_cursor('2a1casv455gxh'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------
-------------------------------------------------------
SQL_ID  2a1casv455gxh, child number 0
-------------------------------------
SELECT  * FROM t1, t2  WHERE t1.id = t2.t1_id  AND t1.n in(18,19)

Plan hash value: 128660979

--------------------------------------------------------------------------------------------
| Id  | Operation                      | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |          |       |       |     8 (100)|          |
|   1 |  TABLE ACCESS BY INDEX ROWID   | T2       |     1 |  2041 |     2   (0)| 00:00:01 |
|   2 |   NESTED LOOPS                 |          |     2 |  8138 |     8   (0)| 00:00:01 |
|   3 |    INLIST ITERATOR             |          |       |       |            |          |
|   4 |     TABLE ACCESS BY INDEX ROWID| T1       |     2 |  4056 |     4   (0)| 00:00:01 |
|*  5 |      INDEX RANGE SCAN          | T1_N     |     4 |       |     2   (0)| 00:00:01 |
|*  6 |    INDEX RANGE SCAN            | T2_T1_ID |     1 |       |     1   (0)| 00:00:01 |
--------------------------------------------------------------------------------------------

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

   5 - access(("T1"."N"=18 OR "T1"."N"=19))
   6 - access("T1"."ID"="T2"."T1_ID")

Note
-----
   - dynamic sampling used for this statement


已选择28行。
```

优点：

- 知道 sql_id 立即可得到执行计划，和 explain plan for 一样无需执行；
- 可以得到真实的执行计划。

缺陷：

- 没有输出运行时的相关统计信息（产生多少逻辑读，多少次递归调用，多少次物理读的情况）。
- 无法判断是处理了多少行。
- 无法判断表被访问了多少次。

1.6 通过 10046TRACE

```
SQL> alter session set statistics_level=typical;

Session altered.

SQL> alter session set events '10046 trace name context  forever,level 12';

Session altered.

SQL> SELECT  *
FROM t1, t2
WHERE t1.id = t2.t1_id
AND t1.n in(18,19);     2    3    4

        ID          N
---------- ----------
CONTENTS
--------------------------------------------------------------------------------
        ID      T1_ID          N
---------- ---------- ----------
CONTENTS
--------------------------------------------------------------------------------
        18         18
yFfYpcNtsxEpvWUOceJmHtvpCEbcJTUTeKExdHlGlwoIFEgmOo
        18         18         18
OWCPIKKFQTMBZAVJBUGPOGOZGPHCMTGFDJTMCIZXRFVXYCATTY
```

具体执行脚本

```
SQL> alter session set events '10046 trace name context off';

Session altered.

SQL> select d.value
|| '/'
|| LOWER (RTRIM(i.INSTANCE, CHR(0)))
|| '_ora_'
|| p.spid
|| '.trc' trace_file_name
from (select p.spid
      from v$mystat m,v$session s, v$process p
      where  m.statistic#=1 and s.sid=m.sid and p.addr=s.paddr) p,
      (select t.INSTANCE
       FROM v$thread t,v$parameter v
       WHERE v.name='thread'
       AND(v.VALUE=0 OR t.thread#=to_number(v.value))) i,
       (select value
       from v$parameter
       where name='user_dump_dest') d;  2    3    4    5    6    7    8    9   10   11   12   13   14

TRACE_FILE_NAME
--------------------------------------------------------------------------------
/u02/app/admin/PROD/udump/prod_ora_7307.trc

SQL> quit
Disconnected from Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, Oracle Label Security, OLAP and Data Mining options
[oracle@std disk1]$ tkprof /u02/app/admin/PROD/udump/prod_ora_7307.trc   /home/oracle/prod.txt  sys=no

TKPROF: Release 10.2.0.1.0 - Production on Wed Apr 23 16:21:35 2014

Copyright (c) 1982, 2005, Oracle.  All rights reserved.
```

```
SELECT  *
FROM t1, t2
WHERE t1.id = t2.t1_id
AND t1.n in(18,19)

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          3          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.00       0.00          4         14          0           2
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.00       0.00          4         17          0           2

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 25

Rows     Row Source Operation
-------  ---------------------------------------------------
      2  TABLE ACCESS BY INDEX ROWID T2 (cr=14 pr=4 pw=0 time=1063 us)
      5   NESTED LOOPS  (cr=12 pr=4 pw=0 time=3752 us)
      2    INLIST ITERATOR  (cr=7 pr=4 pw=0 time=1055 us)
      2     TABLE ACCESS BY INDEX ROWID T1 (cr=7 pr=4 pw=0 time=1041 us)
      2      INDEX RANGE SCAN T1_N (cr=5 pr=4 pw=0 time=879 us)(object id 9737)
      2    INDEX RANGE SCAN T2_T1_ID (cr=5 pr=0 pw=0 time=78 us)(object id 9738)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  SQL*Net message to client                       2        0.00          0.00
  db file scattered read                          1        0.00          0.00
  SQL*Net message from client                     2        0.00          0.00
********************************************************************************
```

优点：

- 可以看出 SQL 语句对应的等待事件
- 如果 SQL 语句中有函数调用，SQL 中有 SQL，将会都被列出，无处遁形。
- 可以方便的看出处理的行数，产生的物理逻辑读。
- 可以方便的看出解析时间和执行时间。
- 可以跟踪整个程序包

缺陷:

- 步骤繁琐，比较麻烦
- 无法判断表被访问了多少次。
- 执行计划中的谓词部分不能清晰的展现出来。

1.7 AWR 方式获得执行计划

Plan Statistics

- % Total DB Time is the Elapsed Time of the SQL statement divided into the Total Database Time multiplied by 100

Stat Name	Statement Total	Per Execution	% Snap Total
Elapsed Time (ms)	34	33.77	0.30
CPU Time (ms)	34	33.77	0.35
Executions	1		
Buffer Gets	94	94.00	0.22
Disk Reads	4	4.00	1.73
Parse Calls	1	1.00	0.09
Rows	2	2.00	
User I/O Wait Time (ms)	0		
Cluster Wait Time (ms)	0		
Application Wait Time (ms)	0		
Concurrency Wait Time (ms)	0		
Invalidations	0		
Version Count	1		
Sharable Mem(KB)	18		

Execution Plan

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT				10 (100)	
1	TABLE ACCESS BY INDEX ROWID	T2	1	2041	3 (0)	00:00:01
2	NESTED LOOPS		2	8138	10 (0)	00:00:01
3	INLIST ITERATOR					
4	TABLE ACCESS BY INDEX ROWID	T1	2	4056	4 (0)	00:00:01
5	INDEX RANGE SCAN	T1_N	4		2 (0)	00:00:01
6	INDEX RANGE SCAN	T2_T1_ID	1		1 (0)	00:00:01

- dynamic sampling used for this statement

Back to Plan 1(PHV: 128660979)
Back to Top

Full SQL Text

SQL Id	SQL Text
1a914ws3ggfsn	SELECT * FROM t1, t2 WHERE t1.id = t2.t1_id AND t1.n in(18, 19)

优点：

✧ 展示的形象直观，通俗易懂

✧ 可以方便的看出当前的系统负载。

✧ 可以方便的看出解析时间和执行时间。

缺陷:

- 步骤繁琐，比较麻烦

1.8如何选择执行计划

<1>. explain plan for 获取；

<2>. set autotrace on ；

<3>. statistics_level=all;

<4>. 通过 dbms_xplan.display_cursor 输入 sql_id 参数直接获取

<5>. 10046 trace 跟踪

<6>. awrsqrpt.sql

- 如果某 SQL 执行非常长时间才会出结果，甚至慢到返回不了结果，这时候看执行计划就只能用方法<1>,或者方法<4>调用现成的；
- 跟踪某条 SQL 最简单的方法是方法<1>，其次就是方法<2>；
- 如果想观察到某条 SQL 有多条执行计划的情况，只能用方法<4>和方法<6>；
- 如果 SQL 中含有多函数，函数中套有 SQL 等多层递归调用，想准确分析，只能使用方法<5>；
- 要想确保看到真实的执行计划，不能用方法<1>和方法<2>；
- 要想获取表被访问的次数，只能使用方法<3>；

2.说说如何辨别低效的 SQL

2.1 返回行与逻辑读比率

```
SQL> CREATE TABLE t as select * from dba_objects;

Table created.

SQL> alter session set statistics_level=all;

Session altered.

SQL> select * from t where object_id=6;

OWNER
------------------------------
OBJECT_NAME
--------------------------------------------------------------------------------
SUBOBJECT_NAME                 OBJECT_ID DATA_OBJECT_ID OBJECT_TYPE
------------------------------ ---------- -------------- -------------------
CREATED   LAST_DDL_ TIMESTAMP           STATUS  T G S
--------- --------- ------------------- ------- - - -
SYS
C_TS#
                                        6              6 CLUSTER
23-APR-14 23-APR-14 2014-04-23:14:15:19 VALID   N N N
```

```
SQL> SELECT * FROM table(dbms_xplan.display_cursor(NULL,NULL,'allstats last'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
SQL_ID  8cxbzma1az713, child number 0
-------------------------------------
select * from t where object_id=6

Plan hash value: 1601196873

---------------------------------------------------------------------------------
| Id  | Operation         | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
---------------------------------------------------------------------------------
|*  1 |  TABLE ACCESS FULL| T    |      1 |      2 |      1 |00:00:00.01 |     123 |
---------------------------------------------------------------------------------
```

1行数据返回了123个buffers,有点大

总共获取 1 条记录(A-ROWS)，产生 123 次逻辑读（Buffers)，这个有些可疑！

```
SQL> set autotrace traceonly
SQL> select * from t where object_id=6;


Execution Plan
----------------------------------------------------------
Plan hash value: 1601196873

--------------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |     2 |   354 |    36   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| T    |     2 |   354 |    36   (0)| 00:00:01 |
--------------------------------------------------------------------------------

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

   1 - filter("OBJECT_ID"=6)

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
        123  consistent gets
          0  physical reads
          0  redo size
       1199  bytes sent via SQL*Net to client
        384  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

总共获取1条记录（1 rows processed），产生123次逻辑读（1048 consistent gets），可疑！

2.2 评估值的准确的重要性

```
SQL> CREATE TABLE t1 (id, col1, col2, pad)
  2  AS
  3  SELECT rownum, CASE WHEN rownum>5000 THEN 666 ELSE rownum END, rownum, lpad('*',100,'*')
  4  FROM dual
  5  CONNECT BY level <= 10000;

Table created.

SQL> INSERT INTO t1 SELECT id+10000, col1, col2, pad FROM t1;

10000 rows created.

SQL> INSERT INTO t1 SELECT id+20000, col1, col2, pad FROM t1;

20000 rows created.

SQL> INSERT INTO t1 SELECT id+40000, col1, col2, pad FROM t1;

40000 rows created.

SQL> INSERT INTO t1 SELECT id+80000, col1, col2, pad FROM t1;

80000 rows created.

SQL> COMMIT;

Commit complete.

SQL> CREATE INDEX t1_col1 ON t1 (col1);

Index created.
```

```
SQL> CREATE TABLE t2 AS SELECT * FROM t1 WHERE mod(col2,19) != 0;

Table created.

SQL> ALTER TABLE t2 ADD CONSTRAINT t2_pk PRIMARY KEY (id);

Table altered.

SQL> BEGIN
  2  dbms_stats.gather_table_stats(
  3  ownname=>user,
  4  tabname=>'T1',
  5  cascade=>TRUE,
  6  estimate_percent=>100,
  7  method_opt=>'for all columns size 1',
  8  no_invalidate=>FALSE);
  9  END;
 10  /

PL/SQL procedure successfully completed.

SQL> BEGIN
  2  dbms_stats.gather_table_stats(
  3  ownname=>user,
  4  tabname=>'T2',
  5  cascade=>TRUE,
  6  estimate_percent=>100,
  7  method_opt=>'for all columns size 1',
  8  no_invalidate=>FALSE);
  9  END;
 10  /

PL/SQL procedure successfully completed.
```

删除了所有列的直方图

```
SQL> explain plan for
  2  SELECT  count(t2.col2)
  3  FROM t1 ,t2 WHERE t1.id=t2.id and t1.col1 = 666;

Explained.

SQL> select * from table(dbms_xplan.display());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------
Plan hash value: 708808766
```

执行计划反映出的只有32行，实际情况真是如此吗？我们采用另外一种方法取执行计划。

```
--------------------------------------------------------------------------------------------
| Id  | Operation                      | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |          |     1 |    18 |    48   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE                |          |     1 |    18 |            |          |
|   2 |   NESTED LOOPS                 |          |    32 |   576 |    48   (0)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID| T1       |    32 |   288 |    18   (0)| 00:00:01 |
|*  4 |     INDEX RANGE SCAN           | T1_COL1  |    32 |       |     1   (0)| 00:00:01 |
|   5 |    TABLE ACCESS BY INDEX ROWID| T2       |     1 |     9 |     1   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------
|*  6 |     INDEX UNIQUE SCAN          | T2_PK    |     1 |       |     0   (0)| 00:00:01 |
--------------------------------------------------------------------------------------------
```

```
SQL> SELECT /*+ gather_plan_statistics */ count(t2.col2)
  2  FROM t1 ,t2 WHERE t1.id=t2.id and t1.col1 = 666;

COUNT(T2.COL2)
--------------
         75808

SQL> SELECT * FROM table(dbms_xplan.display_cursor(NULL,NULL,'allstats last'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------
SQL_ID  g048suxnxkxyr, child number 0
-------------------------------------
SELECT /*+ gather_plan_statistics */ count(t2.col2) FROM t1 ,t2 WHERE t1.id=t2.id and
t1.col1 = 666

Plan hash value: 708808766
```

实际行数达到了80016行，与预估的32行差别较大

```
--------------------------------------------------------------------------------------------
| Id  | Operation                      | Name     | Starts | E-Rows | A-Rows |   A-Time    | Buffers |
--------------------------------------------------------------------------------------------
|   1 |  SORT AGGREGATE                |          |      1 |      1 |      1 |00:00:01.64 |    157K|

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------
|   2 |   NESTED LOOPS                 |          |      1 |     32 |  75808 |00:00:01.59 |    157K|
|   3 |    TABLE ACCESS BY INDEX ROWID| T1       |      1 |     32 |  80016 |00:00:00.48 |   1763 |
|*  4 |     INDEX RANGE SCAN           | T1_COL1  |      1 |     32 |  80016 |00:00:00.16 |    169 |
|   5 |    TABLE ACCESS BY INDEX ROWID| T2       |  80016 |      1 |  75808 |00:00:00.82 |    155K|
|*  6 |     INDEX UNIQUE SCAN          | T2_PK    |  80016 |      1 |  75808 |00:00:00.39 |  80018 |
--------------------------------------------------------------------------------------------
```

```
SQL> SELECT num_rows, distinct_keys, num_rows/distinct_keys AS avg_rows_per_key
  2  FROM user_indexes
  3  WHERE index_name = 'T1_COL1';

  NUM_ROWS DISTINCT_KEYS AVG_ROWS_PER_KEY
---------- ------------- -----------------
    160000          5000                32
```

在没有直方图时系统时如何估计32行的那，如上。值是取的一个平均值。

```
SQL> SELECT * FROM table(dbms_xplan.display_cursor(NULL,NULL,'allstats last'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------
SQL_ID  244qjqaj07uvw, child number 0
-------------------------------------
SELECT /*+ gather_plan_statistics */ count(t2.col2) FROM t1 ,t2 WHERE t1.id=t2.id and t1.col1 = 666

Plan hash value: 906334482                 收集完直方图后预估的行数与实际行数就比较接近了

--------------------------------------------------------------------------------------------------------------
| Id  | Operation           | Name | Starts | E-Rows | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
--------------------------------------------------------------------------------------------------------------
|   1 |  SORT AGGREGATE     |      |      1 |      1 |      1 |00:00:00.87 |    5173 |       |       |          |
|*  2 |   HASH JOIN         |      |      1 |  80000 |  75808 |00:00:00.80 |    5173 |  2330K|  1381K| 3003K (0)|

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------
|*  3 |    TABLE ACCESS FULL| T1   |      1 |  80000 |  80016 |00:00:00.08 |    2644 |       |       |          |
|   4 |    TABLE ACCESS FULL| T2   |      1 |   151K|   151K|00:00:00.15 |    2529 |       |       |          |
--------------------------------------------------------------------------------------------------------------
```

2.3 类型转换需要认真关注

```
SQL> create table t_col_type(id varchar2(20),col2 varchar2(20),col3 varchar2(20));

Table created.

SQL> insert into t_col_type select rownum,'abc','efg' from dual connect by level<=10000;

10000 rows created.

SQL> commit;

Commit complete.

SQL> create index idx_id on t_col_type(id);

Index created.
```

```
SQL> set autotrace traceonly
SQL> select * from t_col_type where id=6;


Execution Plan
----------------------------------------------------------
Plan hash value: 3191204463

-------------------------------------------------------------------------------
| Id  | Operation         | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |            |     1 |    36 |     9   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| T_COL_TYPE |     1 |    36 |     9   (0)| 00:00:01 |
-------------------------------------------------------------------------------

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

   1 - filter(TO_NUMBER("ID")=6)

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          5  recursive calls
          0  db block gets
         64  consistent gets
          0  physical reads
          0  redo size
        520  bytes sent via SQL*Net to client
        384  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
```

```
SQL> select * from t_col_type where id='6';

Execution Plan
----------------------------------------------------------
Plan hash value: 3998173245

---------------------------------------------------------------------------------------
| Id  | Operation                   | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |            |     1 |    36 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| T_COL_TYPE |     1 |    36 |     2   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | IDX_ID     |     1 |       |     1   (0)| 00:00:01 |
---------------------------------------------------------------------------------------

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

   2 - access("ID"='6')

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          9  recursive calls
          0  db block gets
         39  consistent gets
          7  physical reads
          0  redo size
        520  bytes sent via SQL*Net to client
        384  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
```

可见避免类型转换能够降低一致性读和COST值

2.4 小心递归调用部分

```
SQL> create table people (first_name varchar2(200),last_name varchar2(200),sex_id number);

Table created.

SQL> insert into people (first_name,last_name,sex_id) select object_name,object_type,1 from dba_objects;

9422 rows created.

SQL> create table sex (name varchar2(20), sex_id number);

Table created.

SQL> insert into sex (name,sex_id) values ('男',1);

1 row created.

SQL> insert into sex (name,sex_id) values ('女',2);

1 row created.

SQL> insert into sex (name,sex_id) values ('不详',3);

1 row created.

SQL> commit;

Commit complete.

SQL> create or replace function get_sex_name(p_id sex.sex_id%type)
  2  return sex.name%type is v_name sex.name%type;
  3  begin
  4  select name
  5  into v_name
  6  from sex
  7  where sex_id=p_id;
  8  return v_name;
  9  end;
 10  /

Function created.
```

```
SQL> set autotrace traceonly
SQL> select sex_id, first_name||' '||last_name full_name,
  2  get_sex_name(sex_id) gender  from people;

9422 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2528372185

---------------------------------------------------------------------------
| Id  | Operation         | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |        |  9422 |  1996K|    13   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| PEOPLE |  9422 |  1996K|    13   (0)| 00:00:01 |
---------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement
```

产生的递归调用次数

```
Statistics
----------------------------------------------------------
      9494  recursive calls
         0  db block gets
     66705  consistent gets
         0  physical reads
         0  redo size
```

```
SQL>
select p.sex_id,p.first_name||' '||p.last_name full_name,s.name
from people p,sex s where p.sex_id=s.sex_id;SQL>   2

9422 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 1973058250

---------------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |        |  9422 |  2226K|    17   (6)| 00:00:01 |
|*  1 |  HASH JOIN         |        |  9422 |  2226K|    17   (6)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| SEX    |     3 |    75 |     3   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| PEOPLE |  9422 |  1996K|    13   (0)| 00:00:01 |
---------------------------------------------------------------------------

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

   1 - access("P"."SEX_ID"="S"."SEX_ID")

Note
-----
   - dynamic sampling used for this statement
```

同样的结果，换一种写法能够大大降低递归调用且COST值效率提高

```
Statistics
----------------------------------------------------------
         9  recursive calls
         0  db block gets
       733  consistent gets
```

2.5 表的访问次数需敏感

```
SQL> CREATE TABLE t1 (id, col1, col2, pad)  as
  2  SELECT rownum, CASE WHEN rownum>5000 THEN 666 ELSE rownum END,
  3  rownum, lpad('*',100,'*')     FROM dual
  4  CONNECT BY level <= 10000;

Table created.

SQL> INSERT INTO t1 SELECT id+10000, col1, col2, pad FROM t1;

10000 rows created.

SQL> INSERT INTO t1 SELECT id+20000, col1, col2, pad FROM t1;

20000 rows created.

SQL> INSERT INTO t1 SELECT id+40000, col1, col2, pad FROM t1;

40000 rows created.

SQL> INSERT INTO t1 SELECT id+80000, col1, col2, pad FROM t1;

80000 rows created.

SQL> COMMIT
  2  ;

Commit complete.

SQL> CREATE INDEX t1_col1 ON t1 (col1);

Index created.
```

```
SQL> CREATE TABLE t2 AS SELECT * FROM t1 WHERE mod(col2,19) != 0;

Table created.

SQL> ALTER TABLE t2 ADD CONSTRAINT t2_pk PRIMARY KEY (id);

Table altered.

SQL> BEGIN
  2  dbms_stats.gather_table_stats(
  3  ownname=>user,
  4   tabname=>'T1',
  5  cascade=>TRUE,
  6  estimate_percent=>100,
  7  method_opt=>'for all columns size 1',
  8  no_invalidate=>FALSE);
  9  END;
 10  /

PL/SQL procedure successfully completed.

SQL> BEGIN
  2  dbms_stats.gather_table_stats(
  3  ownname=>user,
  4  tabname=>'T2',
  5  cascade=>TRUE,
  6  estimate_percent=>100,
  7  method_opt=>'for all columns size 1',
  8  no_invalidate=>FALSE);
  9  END;
 10  /

PL/SQL procedure successfully completed.
```

删除了直方图

```
SQL> SELECT /*+ gather_plan_statistics */ count(t2.col2)
  2  FROM t1 ,t2 WHERE t1.id=t2.id and t1.col1 = 666;

COUNT(T2.COL2)
--------------
         75808

SQL> SELECT * FROM table(dbms_xplan.display_cursor(NULL,NULL,'allstats last'));

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------------------
SQL_ID  g048suxnxkxyr, child number 0
-------------------------------------
SELECT /*+ gather_plan_statistics */ count(t2.col2) FROM t1 ,t2 WHERE t1.id=t2.id and
t1.col1 = 666

Plan hash value: 708808766
```

可以看到表的访问次数比较偏高，预估的行数却很少，需要引起注意

```
-----------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name    | Starts | E-Rows | A-Rows |   A-Time    | Buffers |
-----------------------------------------------------------------------------------------------------
|   1 |  SORT AGGREGATE               |         |      1 |      1 |      1 |00:00:01.69 |     157K|

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------------
|   2 |   NESTED LOOPS                |         |      1 |     32 |  75808 |00:00:01.59 |     157K|
|   3 |    TABLE ACCESS BY INDEX ROWID| T1      |      1 |     32 |  80016 |00:00:00.48 |    1763 |
|*  4 |     INDEX RANGE SCAN          | T1_COL1 |      1 |     32 |  80016 |00:00:00.17 |     169 |
|   5 |    TABLE ACCESS BY INDEX ROWID| T2      |  80016 |      1 |  75808 |00:00:00.86 |     155K|
|*  6 |     INDEX UNIQUE SCAN         | T2_PK   |  80016 |      1 |  75808 |00:00:00.40 |   80018 |
-----------------------------------------------------------------------------------------------------
```

2.6 注意表真实的访问行数

```
SQL> create table t1 as select * from dba_objects;

Table created.

SQL> create table t2 (id1,id2) as
  2  select rownum ,rownum+100 from dual connect  by level <=1000;

Table created.

SQL> alter session set statistics_level=all;

Session altered.

SQL> select * from (select t1.*, rownum as rn from t1, t2 where t1.object_id = t2.id1) a
  2  where a.rn >= 1   and a.rn <= 10;

OWNER
------------------------------
```

真实的数据访问

```
OBJECT_NAME
-----------------------------------------------------------------------------------------------------
SUBOBJECT_NAME                  OBJECT_ID DATA_OBJECT_ID OBJECT_TYPE         CREATED   LAS
------------------------------- ---------- -------------- ------------------- --------- ---
STATUS  T G S         RN
------- - - - ----------
SYS
ICOL$
                                       20              2 TABLE               23-APR-14 23-
VALID   N N N          1
```

```
SQL> SELECT * FROM table(dbms_xplan.display_cursor(NULL,NULL,'allstats last'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
---------
SQL_ID  ayzfn8k0j3sms, child number 0
-------------------------------------
select *   from (select t1.*, rownum as rn from t1, t2 where t1.object_id = t2.id1) a  where a.rn >= 1
and a.rn <= 10

Plan hash value: 3062220019

--------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation            | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |  OMem |  1Mem | Used-Mem |
--------------------------------------------------------------------------------------------------------------------------------
|*  1 |  VIEW                |      |      1 |   1000 |     10 |00:00:00.06 |     127 |       |       |          |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
---------
|   2 |   COUNT              |      |      1 |        |    954 |00:00:00.05 |     127 |       |       |          |
|*  3 |    HASH JOIN         |      |      1 |   1000 |    954 |00:00:00.05 |     127 |  1036K|  1036K| 1148K (0)|
|   4 |     TABLE ACCESS FULL| T2   |      1 |   1000 |   1000 |00:00:00.01 |       4 |       |       |          |
|   5 |     TABLE ACCESS FULL| T1   |      1 |  10860 |   9425 |00:00:00.01 |     123 |       |       |          |
--------------------------------------------------------------------------------------------------------------------------------
```

表真实的访问行数

```
SQL> select *  from (select t1.*, rownum as rn from t1, t2 where t1.object_id = t2.id1 and rownum<=10) a
  2  where a.rn >= 1;

OWNER                          OBJECT_NAME
------------------------------ ------------------------------------------------------------------------------------------------
 OBJECT_ID DATA_OBJECT_ID OBJECT_TYPE         CREATED   LAST_DDL_ TIMESTAMP           STATUS  T G S         RN
---------- -------------- ------------------- --------- --------- ------------------- ------- - - - ----------
SYS                            ICOL$
        20              2 TABLE               23-APR-14 23-APR-14 2014-04-23:14:15:19 VALID   N N N          1

SYS                            I_USER1
        44             44 INDEX               23-APR-14 23-APR-14 2014-04-23:14:15:19 VALID   N N N          2

SYS                            CON$
        28             28 TABLE               23-APR-14 23-APR-14 2014-04-23:14:15:19 VALID   N N N          3
```

优化改写脚本，然后再看真实的访问行数

```
SQL> SELECT * FROM table(dbms_xplan.display_cursor(NULL,NULL,'allstats last'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
---------
SQL_ID  3vaykgpsps2ry, child number 0
-------------------------------------
select *  from (select t1.*, rownum as rn from t1, t2 where t1.object_id = t2.id1 and rownum<=10) a where
a.rn >= 1

Plan hash value: 1802812661

--------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation            | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |  OMem |  1Mem | Used-Mem |
--------------------------------------------------------------------------------------------------------------------------------
|*  1 |  VIEW                |      |      1 |     10 |     10 |00:00:00.01 |       9 |       |       |          |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
---------
|*  2 |   COUNT STOPKEY      |      |      1 |        |     10 |00:00:00.01 |       9 |       |       |          |
|*  3 |    HASH JOIN         |      |      1 |   1000 |     10 |00:00:00.01 |       9 |  1036K|  1036K| 1148K (0)|
|   4 |     TABLE ACCESS FULL| T2   |      1 |   1000 |   1000 |00:00:00.01 |       4 |       |       |          |
|   5 |     TABLE ACCESS FULL| T1   |      1 |  10860 |     10 |00:00:00.01 |       5 |       |       |          |
--------------------------------------------------------------------------------------------------------------------------------
```

真实访问行数明显减少，效率提高

2.7 谨慎的观察排序与否

```
SQL> create table t as select * from dba_objects;

Table created.

SQL> set autot traceonly stat
SQL> select * from t where object_id>2 order by object_id;

9425 rows selected.


Statistics
----------------------------------------------------------
        288  recursive calls
          0  db block gets
        211  consistent gets
        118  physical reads
          0  redo size
     428981  bytes sent via SQL*Net to client
       7292  bytes received via SQL*Net from client
        630  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
       9425  rows processed
```

有一个排序

```
SQL> create index idx_object_id on t(object_id);

Index created.

SQL> select * from t where object_id>2 order by object_id;

9425 rows selected.
```

在排序前加了个索引，消除了派讯

```
Statistics
----------------------------------------------------------
          9  recursive calls
          0  db block gets
       1471  consistent gets
         20  physical reads
          0  redo size
     428981  bytes sent via SQL*Net to client
       7292  bytes received via SQL*Net from client
        630  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       9425  rows processed
```

3.画出执行计划的访问草图

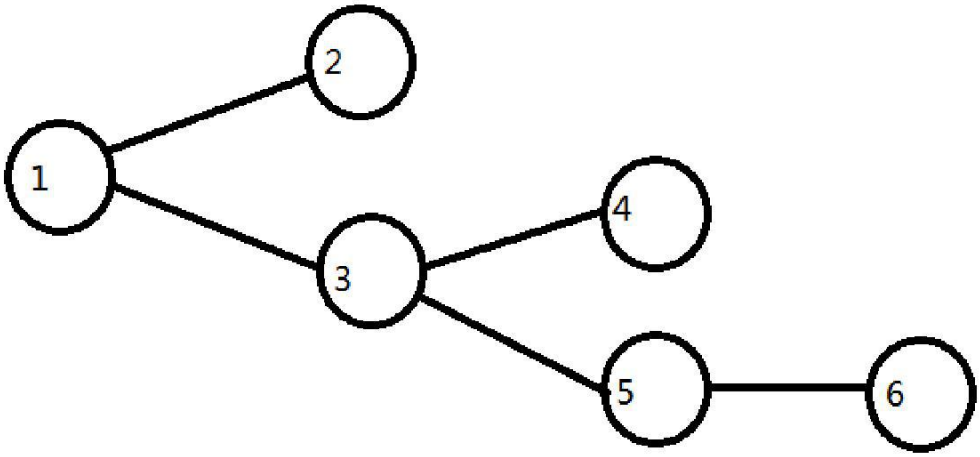
请根据《解释读懂执行计划 3_联合型(相关联)04_树形.sql》的脚本的输出执行计

划，用联合型+单独型的方式，画出执行计划的访问草图。

```
Plan hash value: 1519159851
```

执行计划如下

Id	Operation	Name	Starts	E-Rows	A-Rows	
0	SELECT STATEMENT		1		14	0
* 1	CONNECT BY WITH FILTERING		1		14	0
* 2	TABLE ACCESS FULL	EMP	1	1	1	0
3	NESTED LOOPS		4	2	13	0
4	CONNECT BY PUMP		4		14	0
5	TABLE ACCESS BY INDEX ROWID	EMP	14	2	13	0
* 6	INDEX RANGE SCAN	EMP_MGR_I	14	2	13	0



执行顺序 2->4->6->5->3->1