收获，不止 SQL 优化

第 十一 章
且慢，表连接的秘密让 SQL 飞

E-Mail:45240040@qq.com

目录

1.说说三大经典表连接分别有什么限制

1.1 环境准备

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

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     );

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 <= 100
  5        ORDER BY dbms_random.random;

已创建100行。

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

已创建100000行。

SQL> commit;

提交完成。
```

1.2 Nested Loops Join

```
SQL> set linesize 1000
SQL> set autotrace traceonly explain
SQL> SELECT  /*+ leading(t1) use_nl(t2) */ *
  2     FROM t1, t2
  3     WHERE t1.id > t2.t1_id
  4     AND t1.n = 19;
```

嵌套循环支持大于运算

```
执行计划
----------------------------------------------------------
Plan hash value: 1967407726

---------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |  4373 |   16M|   227   (2)| 00:00:03 |
|   1 |  NESTED LOOPS      |      |  4373 |   16M|   227   (2)| 00:00:03 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |  2028 |     3   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| T2   |  4373 |  8716K|   224   (2)| 00:00:03 |
---------------------------------------------------------------------------
```

```
SQL> SELECT  /*+ leading(t1) use_nl(t2) */ *
  2     FROM t1, t2
  3     WHERE t1.id < t2.t1_id
  4     AND t1.n = 19;
```

嵌套循环支持小于运算

```
执行计划
----------------------------------------------------------
Plan hash value: 1967407726

---------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |  4373 |   16M|   227   (2)| 00:00:03 |
|   1 |  NESTED LOOPS      |      |  4373 |   16M|   227   (2)| 00:00:03 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |  2028 |     3   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| T2   |  4373 |  8716K|   224   (2)| 00:00:03 |
---------------------------------------------------------------------------
```

```
SQL> SELECT  /*+ leading(t1) use_nl(t2) */ *
  2     FROM t1, t2
  3     WHERE t1.id = t2.t1_id
  4     AND t1.n = 19;
```

嵌套循环肯定是支持最基本的等值运算的

```
执行计划
----------------------------------------------------------
Plan hash value: 1967407726

---------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |  4069 |   227   (2)| 00:00:03 |
|   1 |  NESTED LOOPS      |      |     1 |  4069 |   227   (2)| 00:00:03 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |  2028 |     3   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| T2   |     1 |  2041 |   224   (2)| 00:00:03 |
---------------------------------------------------------------------------
```

```
SQL> SELECT  /*+ leading(t1) use_nl(t2) */ *
  2     FROM t1, t2
  3     WHERE t1.id <> t2.t1_id
  4     AND t1.n = 19;
```

嵌套循环支持不等于运算

```
执行计划
----------------------------------------------------------
Plan hash value: 1967407726

---------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      | 87459 |  339M|   227   (2)| 00:00:03 |
|   1 |  NESTED LOOPS      |      | 87459 |  339M|   227   (2)| 00:00:03 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |  2028 |     3   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| T2   | 87459 |  170M|   224   (2)| 00:00:03 |
---------------------------------------------------------------------------
```

```
SQL> SELECT  /*+ leading(t1) use_nl(t2) */ *
  2    FROM t1, t2
  3    WHERE t1.id like t2.t1_id
  4    AND t1.n = 19;
                                             嵌套循环支持like运算
执行计划
----------------------------------------------------------
Plan hash value: 1967407726

--------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |  4373 |   16M|   227   (2)| 00:00:03 |
|   1 |  NESTED LOOPS      |      |  4373 |   16M|   227   (2)| 00:00:03 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |  2028 |     3   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| T2   |  4373 | 8716K|   224   (2)| 00:00:03 |
--------------------------------------------------------------------------
```

从上面的试验来看，nested loop jion 基本上是没有限制的，可以支持所有的运算。

1.3 Hash Join

```
SQL> SELECT /*+ leading(t1) use_hash(t2)*/ *
  2    FROM t1, t2
  3    WHERE t1.id > t2.t1_id
  4    AND t1.n = 19;
                                           大于运算没有用到hash jion
执行计划
----------------------------------------------------------
Plan hash value: 1967407726

--------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |  4373 |   16M|   227   (2)| 00:00:03 |
|   1 |  NESTED LOOPS      |      |  4373 |   16M|   227   (2)| 00:00:03 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |  2028 |     3   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| T2   |  4373 | 8716K|   224   (2)| 00:00:03 |
--------------------------------------------------------------------------
```

```
SQL> SELECT /*+ leading(t1) use_hash(t2)*/ *
  2    FROM t1, t2
  3    WHERE t1.id < t2.t1_id
  4    AND t1.n = 19;
                                           小于运算用不到HASH JION
执行计划
----------------------------------------------------------
Plan hash value: 1967407726

--------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |  4373 |   16M|   227   (2)| 00:00:03 |
|   1 |  NESTED LOOPS      |      |  4373 |   16M|   227   (2)| 00:00:03 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |  2028 |     3   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| T2   |  4373 | 8716K|   224   (2)| 00:00:03 |
--------------------------------------------------------------------------
```

```
SQL> SELECT /*+ leading(t1) use_hash(t2)*/ *
  2     FROM t1, t2
  3     WHERE t1.id <> t2.t1_id
  4     AND t1.n = 19;
```

执行计划

不等于运算用不上HASH JION

```
----------------------------------------------------------
Plan hash value: 1967407726

-----------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      | 87459 |  339M|   227   (2)| 00:00:03 |
|   1 |  NESTED LOOPS      |      | 87459 |  339M|   227   (2)| 00:00:03 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |  2028 |     3   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| T2   | 87459 |  170M|   224   (2)| 00:00:03 |
-----------------------------------------------------------------------------
```

```
SQL> SELECT /*+ leading(t1) use_hash(t2)*/ *
  2     FROM t1, t2
  3     WHERE t1.id like t2.t1_id
  4     AND t1.n = 19;
```

执行计划

like运算用不上HASH JION

```
----------------------------------------------------------
Plan hash value: 1967407726

-----------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |  4373 |   16M|   227   (2)| 00:00:03 |
|   1 |  NESTED LOOPS      |      |  4373 |   16M|   227   (2)| 00:00:03 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |  2028 |     3   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| T2   |  4373 | 8716K|   224   (2)| 00:00:03 |
-----------------------------------------------------------------------------
```

```
SQL> SELECT /*+ leading(t1) use_hash(t2)*/ *
  2     FROM t1, t2
  3     WHERE t1.id = t2.t1_id
  4     AND t1.n = 19;
```

综合前面的试验可以看到只有等值运算才能用上HASH JION

执行计划

```
----------------------------------------------------------
Plan hash value: 1838229974

-----------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |  4069 |   229   (3)| 00:00:03 |
|*  1 |  HASH JOIN         |      |     1 |  4069 |   229   (3)| 00:00:03 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |  2028 |     3   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| T2   | 87460 |  170M|   224   (2)| 00:00:03 |
-----------------------------------------------------------------------------
```

1.4 Merge Sort Join

```
SQL> SELECT /*+ leading(t1) use_merge(t2)*/ *
  2     FROM t1, t2
  3     WHERE t1.id > t2.t1_id
  4     AND t1.n = 19;
```

大于运算可以用到merge sort jion

```
执行计划
----------------------------------------------------------
Plan hash value: 412793182

----------------------------------------------------------------------------------------
| Id  | Operation            | Name | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |      |  4373 |   16M|       | 37586   (1)| 00:07:32 |
|   1 |  MERGE JOIN          |      |  4373 |   16M|       | 37586   (1)| 00:07:32 |
|   2 |   SORT JOIN          |      |     1 |  2028 |       |     4  (25)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL| T1   |     1 |  2028 |       |     3   (0)| 00:00:01 |
|*  4 |   SORT JOIN          |      | 87460 |  170M|   455M| 37582   (1)| 00:07:31 |
|   5 |    TABLE ACCESS FULL| T2   | 87460 |  170M|       |   224   (2)| 00:00:03 |
----------------------------------------------------------------------------------------
```

```
SQL> SELECT /*+ leading(t1) use_merge(t2)*/ *
  2     FROM t1, t2
  3     WHERE t1.id < t2.t1_id
  4     AND t1.n = 19;
```

小于运算也可以用到merge sort jion

```
执行计划
----------------------------------------------------------
Plan hash value: 412793182

----------------------------------------------------------------------------------------
| Id  | Operation            | Name | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |      |  4373 |   16M|       | 37586   (1)| 00:07:32 |
|   1 |  MERGE JOIN          |      |  4373 |   16M|       | 37586   (1)| 00:07:32 |
|   2 |   SORT JOIN          |      |     1 |  2028 |       |     4  (25)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL| T1   |     1 |  2028 |       |     3   (0)| 00:00:01 |
|*  4 |   SORT JOIN          |      | 87460 |  170M|   455M| 37582   (1)| 00:07:31 |
|   5 |    TABLE ACCESS FULL| T2   | 87460 |  170M|       |   224   (2)| 00:00:03 |
----------------------------------------------------------------------------------------
```

```
SQL> SELECT /*+ leading(t1) use_merge(t2)*/ *
  2     FROM t1, t2
  3     WHERE t1.id = t2.t1_id
  4     AND t1.n = 19;
```

等值运算用上了merge sort join

```
执行计划
----------------------------------------------------------
Plan hash value: 412793182

----------------------------------------------------------------------------------------
| Id  | Operation            | Name | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |      |     1 |  4069 |       | 37586   (1)| 00:07:32 |
|   1 |  MERGE JOIN          |      |     1 |  4069 |       | 37586   (1)| 00:07:32 |
|   2 |   SORT JOIN          |      |     1 |  2028 |       |     4  (25)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL| T1   |     1 |  2028 |       |     3   (0)| 00:00:01 |
|*  4 |   SORT JOIN          |      | 87460 |  170M|   455M| 37582   (1)| 00:07:31 |
|   5 |    TABLE ACCESS FULL| T2   | 87460 |  170M|       |   224   (2)| 00:00:03 |
----------------------------------------------------------------------------------------
```

```
SQL> SELECT /*+ leading(t1) use_merge(t2)*/ *
  2    FROM t1, t2
  3    WHERE t1.id <> t2.t1_id
  4    AND t1.n = 19;
```

执行计划　　　　不等值运算用不上merge sort join

```
----------------------------------------------------------
Plan hash value: 1967407726

---------------------------------------------------------------------------
| Id  | Operation           | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |      | 87459 |  339M|   227   (2)| 00:00:03 |
|   1 |  NESTED LOOPS       |      | 87459 |  339M|   227   (2)| 00:00:03 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |  2028 |     3   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| T2   | 87459 |  170M|   224   (2)| 00:00:03 |
---------------------------------------------------------------------------
```

```
SQL> SELECT /*+ leading(t1) use_merge(t2)*/ *
  2    FROM t1, t2
  3    WHERE t1.id like t2.t1_id
  4    AND t1.n = 19;
```

执行计划　　　　like运算用不上merge sort join

```
----------------------------------------------------------
Plan hash value: 1967407726

---------------------------------------------------------------------------
| Id  | Operation           | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |      |  4373 |   16M|   227   (2)| 00:00:03 |
|   1 |  NESTED LOOPS       |      |  4373 |   16M|   227   (2)| 00:00:03 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |  2028 |     3   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| T2   |  4373 | 8716K|   224   (2)| 00:00:03 |
---------------------------------------------------------------------------
```

2. 说说三大经典表连接的各自特性

（从表访问次数，表驱动顺序与性能、是否排序这三方面说）

利用上一题搭建起来的环境我们进行测试。

2.1 表的访问次数

2.1.1 Nested Loops Join

```
SQL> Set linesize 1000
SQL> alter session set statistics_level=all ;

会话已更改。

SQL> select count(*) from t1;

  COUNT(*)
----------
       100

SQL> select count(*) from t2;

  COUNT(*)
----------
    100000

SQL> SELECT /*+ leading(t1) use_nl(t2)*/ *
  2  FROM t1, t2
  3  WHERE t1.id = t2.t1_id;
```

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

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
----------
SQL_ID  245z7n1cxaf3m, child number 0
-------------------------------------
SELECT /*+ leading(t1) use_nl(t2)*/ * FROM t1, t2 WHERE t1.id = t2.t1_id

Plan hash value: 1967407726

-----------------------------------------------------------------------------------------
| Id  | Operation          | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------
|   1 |  NESTED LOOPS      |      |      1 |    100 |    100 |00:00:01.04 |    100K|
|   2 |   TABLE ACCESS FULL| T1   |      1 |    100 |    100 |00:00:00.01 |     14 |

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
----------
|*  3 |   TABLE ACCESS FULL| T2   |    100 |      1 |    100 |00:00:01.04 |    100K|
-----------------------------------------------------------------------------------------
```

从上面的图中可以 t1 表有数据 100 条，T2 表有数据 10 条，T1 表作为驱动表对 T1 表的访问次数为 1 此，这一次访问了 100 条数据，而这 100 条数据需要逐条访问 T2 表，所以导致 T2 表访问 100 次。所以在 NL 连接中，驱动表被访问 0 或者 1 次，被驱动表被访问 0 次或者 N 次，N 由驱动表返回的结果集的条数来定。

2.1.2 Hash Join

```
SQL> select count(*) from t1;

  COUNT(*)
----------
       100

SQL> select count(*) from t2;

  COUNT(*)
----------
    100000

SQL> SELECT /*+ leading(t1) use_hash(t2) */ *
  2  FROM t1, t2
  3  WHERE t1.id = t2.t1_id;
```

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

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
----------
SQL_ID  8waddjy8kpmcj, child number 0
-------------------------------------
SELECT /*+ leading(t1) use_hash(t2) */ * FROM t1, t2 WHERE t1.id = t2.t1_id

Plan hash value: 1838229974

-----------------------------------------------------------------------------------------------------------
| Id  | Operation          | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------
|*  1 |  HASH JOIN         |      |      1 |    100 |    100 |00:00:00.09 |    1018 |  741K|  741K| 1130K (0)|
|   2 |   TABLE ACCESS FULL| T1   |      1 |    100 |    100 |00:00:00.01 |       7 |      |      |          |

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
----------
|   3 |   TABLE ACCESS FULL| T2   |      1 |  87460 |   100K|00:00:00.01 |    1011 |      |      |          |
-----------------------------------------------------------------------------------------------------------
```

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

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
----------                                                            执行的语句
SQL_ID  1zkvhkbs917dg, child number 0
-------------------------------------
SELECT /*+ leading(t1) use_hash(t2)*/ * FROM t1, t2 WHERE t1.id = t2.t1_id and t1.n=999999999

Plan hash value: 1838229974

-----------------------------------------------------------------------------------------------------------
| Id  | Operation          | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------
|*  1 |  HASH JOIN         |      |      1 |      1 |      0 |00:00:00.01 |       7 |  675K|  675K|  154K (0)|
|*  2 |   TABLE ACCESS FULL| T1   |      1 |      1 |      0 |00:00:00.01 |       7 |      |      |          |

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
----------
|   3 |   TABLE ACCESS FULL| T2   |      0 |  87460 |      0 |00:00:00.01 |       0 |      |      |          |
-----------------------------------------------------------------------------------------------------------
```

HASH 连接中，驱动表被访问 0 或者 1 次，被驱动表也是被访问 0 次或者 1 次，

绝大部分场景是驱动表和被驱动表被各访问 1 次.

2.1.3 Merge Sort Join

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

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
SQL_ID  3jnrkx3n21b2s, child number 1
-------------------------------------
SELECT  /*+ leading(t1) use_merge(t2)*/ * FROM t1, t2 WHERE t1.id = t2.t1_id

Plan hash value: 412793182

----------------------------------------------------------------------------------------------------------------------
| Id  | Operation           | Name | Starts | E-Rows | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
----------------------------------------------------------------------------------------------------------------------
|   1 |  MERGE JOIN         |      |      1 |    100 |    100 |00:00:00.09 |    1012 |       |       |          |
|   2 |   SORT JOIN         |      |      1 |    100 |    100 |00:00:00.01 |       7 |  9216 |  9216 | 8192  (0)|

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
|   3 |    TABLE ACCESS FULL| T1   |      1 |    100 |    100 |00:00:00.01 |       7 |       |       |          |
|*  4 |   SORT JOIN         |      |    100 |  87460 |    100 |00:00:00.09 |    1005 |  9124K|  1177K| 8110K (0)|
|   5 |    TABLE ACCESS FULL| T2   |      1 |  87460 |    100K|00:00:00.01 |    1005 |       |       |          |
----------------------------------------------------------------------------------------------------------------------
```

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

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
SQL_ID  72k5npjaxcdjg, child number 0
-------------------------------------
SELECT /*+ leading(t1) use_merge(t2)*/ * FROM t1, t2 WHERE t1.id = t2.t1_id and t1.n=999999999

Plan hash value: 412793182

----------------------------------------------------------------------------------------------------------------------
| Id  | Operation           | Name | Starts | E-Rows | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
----------------------------------------------------------------------------------------------------------------------
|   1 |  MERGE JOIN         |      |      1 |      1 |      0 |00:00:00.01 |       7 |       |       |          |
|   2 |   SORT JOIN         |      |      1 |      1 |      0 |00:00:00.01 |       7 |  1024 |  1024 |          |

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
|*  3 |    TABLE ACCESS FULL| T1   |      1 |      1 |      0 |00:00:00.01 |       7 |       |       |          |
|*  4 |   SORT JOIN         |      |      0 |  87460 |      0 |00:00:00.01 |       0 |   191M|  4642K|          |
|   5 |    TABLE ACCESS FULL| T2   |      0 |  87460 |      0 |00:00:00.01 |       0 |       |       |          |
----------------------------------------------------------------------------------------------------------------------
```

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

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
SQL_ID  ffwpc2087z7u, child number 0
-------------------------------------
SELECT /*+ leading(t1) use_merge(t2)*/ * FROM t1, t2 WHERE t1.id = t2.t1_id and 1=2

Plan hash value: 209786894

------------------------------------------------------------------------------------------------------------
| Id  | Operation            | Name | Starts | E-Rows | A-Rows |   A-Time    |  OMem |  1Mem | Used-Mem |
------------------------------------------------------------------------------------------------------------
|*  1 |  FILTER              |      |      1 |        |      0 |00:00:00.01 |       |       |          |
|   2 |   MERGE JOIN         |      |      0 |    100 |      0 |00:00:00.01 |       |       |          |

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
|   3 |    SORT JOIN         |      |      0 |    100 |      0 |00:00:00.01 |   222K|   222K|          |
|   4 |     TABLE ACCESS FULL| T1   |      0 |    100 |      0 |00:00:00.01 |       |       |          |
|*  5 |    SORT JOIN         |      |      0 |  87460 |      0 |00:00:00.01 |   191M|  4642K|          |
|   6 |     TABLE ACCESS FULL| T2   |      0 |  87460 |      0 |00:00:00.01 |       |       |          |
------------------------------------------------------------------------------------------------------------
```

结论：排序合并连接中，两表都是只被访问 0 次或者 1 次，和 HASH 连接一样。

2.2 表驱动顺序与性能

2.2.1 Nested Loops Join

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------
SQL_ID  djss7wrzt36tm, child number 0
-------------------------------------
SELECT /*+ leading(t1) use_nl(t2)*/ * FROM t1, t2 WHERE t1.id = t2.t1_id AND
t1.n = 19
                                          T1为驱动表的时候
Plan hash value: 1967407726

-------------------------------------------------------------------------------------
| Id  | Operation           | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
-------------------------------------------------------------------------------------
|   1 |  NESTED LOOPS       |      |      1 |      1 |      1 |00:00:00.01 |    1014 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------
|*  2 |   TABLE ACCESS FULL| T1   |      1 |      1 |      1 |00:00:00.01 |       8 |
|*  3 |   TABLE ACCESS FULL| T2   |      1 |      1 |      1 |00:00:00.01 |    1006 |
-------------------------------------------------------------------------------------
```

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------
SQL_ID  b2h5gmp0zq4q1, child number 0
-------------------------------------
SELECT /*+ leading(t2) use_nl(t1)*/ * FROM t1, t2 WHERE t1.id = t2.t1_id AND
t1.n = 19
                                        T2为驱动表时（大表）,性能较低
Plan hash value: 4016936828

-------------------------------------------------------------------------------------
| Id  | Operation           | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
-------------------------------------------------------------------------------------
|   1 |  NESTED LOOPS       |      |      1 |      1 |      1 |00:00:01.05 |    701K|

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------
|   2 |   TABLE ACCESS FULL| T2   |      1 |  87460 |   100K|00:00:00.01 |    1006 |
|*  3 |   TABLE ACCESS FULL| T1   |   100K|      1 |      1 |00:00:01.00 |    700K|
-------------------------------------------------------------------------------------
```

从上面的试验中可以看出不同的表连接顺序在 Nested Loops Join 中效率是差别很大的。

2.2.2 Hash Join

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

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------
----------
SQL_ID  6a9vfdqktf44d, child number 0
-------------------------------------
SELECT /*+ leading(t1) use_hash(t2)*/ * FROM t1, t2 WHERE t1.id = t2.t1_id and t1.n=19

Plan hash value: 1838229974                HASH JION下的T1表为驱动表

------------------------------------------------------------------------------------------------------------------
| Id  | Operation          | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |  OMem |  1Mem | Used-Mem |
------------------------------------------------------------------------------------------------------------------
|*  1 |  HASH JOIN         |      |      1 |      1 |      1 |00:00:00.07 |    1013 |  741K|  741K|  316K (0)|
|*  2 |   TABLE ACCESS FULL| T1   |      1 |      1 |      1 |00:00:00.01 |       7 |      |      |          |

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------
----------
|   3 |   TABLE ACCESS FULL| T2   |      1 |  87460 |   100K|00:00:00.01 |    1006 |      |      |          |
------------------------------------------------------------------------------------------------------------------
```

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

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------
----------
SQL_ID  1b43t06ktjbxq, child number 0
-------------------------------------
SELECT /*+ leading(t2) use_hash(t1)*/ * FROM t1, t2 WHERE t1.id = t2.t1_id and t1.n=19

Plan hash value: 2959412835              HASH JOIN下的T2表为驱动表

------------------------------------------------------------------------------------------------------------------
| Id  | Operation          | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |  OMem |  1Mem | Used-Mem |
------------------------------------------------------------------------------------------------------------------
|*  1 |  HASH JOIN         |      |      1 |      1 |      1 |00:00:00.11 |    1013 | 9471K| 1956K|   11M (0)|
|   2 |   TABLE ACCESS FULL| T2   |      1 |  87460 |   100K|00:00:00.01 |    1005 |      |      |          |

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------
----------
|*  3 |   TABLE ACCESS FULL| T1   |      1 |      1 |      1 |00:00:00.01 |       8 |      |      |          |
------------------------------------------------------------------------------------------------------------------
```

对比上面两个试验可以看出在 HASH JOIN 下表的连接顺序对性能的影响不大。

2.1.3Merge Sort Join

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

PLAN_TABLE_OUTPUT
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
----------
SQL_ID  5r1r1jhp1u31u, child number 0
-------------------------------------
SELECT /*+ leading(t1) use_merge(t2)*/ * FROM t1, t2 WHERE t1.id = t2.t1_id and t1.n=19

Plan hash value: 412793182                    在Merge Sort Join下t1表为驱动表

------------------------------------------------------------------------------------------------------------
| Id  | Operation              | Name | Starts | E-Rows | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
------------------------------------------------------------------------------------------------------------
|   1 |  MERGE JOIN            |      |      1 |      1 |      1 |00:00:00.09 |    1012 |       |       |          |
|   2 |   SORT JOIN            |      |      1 |      1 |      1 |00:00:00.01 |       7 |  2048 |  2048 | 2048  (0)|

PLAN_TABLE_OUTPUT
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
----------
|*  3 |    TABLE ACCESS FULL| T1   |      1 |      1 |      1 |00:00:00.01 |       7 |       |       |          |
|*  4 |   SORT JOIN            |      |      1 |  87460 |      1 |00:00:00.09 |    1005 | 9124K| 1177K| 8110K (0)|
|   5 |    TABLE ACCESS FULL| T2   |      1 |  87460 |   100K|00:00:00.01 |    1005 |       |       |          |
------------------------------------------------------------------------------------------------------------
```

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

PLAN_TABLE_OUTPUT
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
----------
SQL_ID  5bxazj15c33ga, child number 0
-------------------------------------
SELECT /*+ leading(t2) use_merge(t1)*/ * FROM t1, t2 WHERE t1.id = t2.t1_id and t1.n=19

Plan hash value: 1792967693                   在Merge Sort Join下t2表为驱动表

------------------------------------------------------------------------------------------------------------
| Id  | Operation              | Name | Starts | E-Rows | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
------------------------------------------------------------------------------------------------------------
|   1 |  MERGE JOIN            |      |      1 |      1 |      1 |00:00:00.12 |    1012 |       |       |          |
|   2 |   SORT JOIN            |      |      1 |  87460 |     20 |00:00:00.12 |    1005 | 9124K| 1177K| 8110K (0)|

PLAN_TABLE_OUTPUT
---------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------
----------
|   3 |    TABLE ACCESS FULL| T2   |      1 |  87460 |   100K|00:00:00.01 |    1005 |       |       |          |
|*  4 |   SORT JOIN            |      |     20 |      1 |      1 |00:00:00.01 |       7 |  2048 |  2048 | 2048  (0)|
|*  5 |    TABLE ACCESS FULL| T1   |      1 |      1 |      1 |00:00:00.01 |       7 |       |       |          |
------------------------------------------------------------------------------------------------------------
```

对比上面两个试验可以看出在 Merge Sort Join 下表的连接顺序对性能的影响和 Hash Join 类似，效率与表的连接顺序关系不大。

2.3 是否排序

2.3.1 Nested Loops Join

```
SQL> set autot traceonly stat
SQL> SELECT /*+ leading(t1) use_nl(t2)*/ *
  2  FROM t1, t2
  3  WHERE t1.id = t2.t1_id
  4  AND t1.n = 19;
                                      可以nested loops下没有排序

统计信息
----------------------------------------------------------
          0  recursive calls
          0  db block gets
       1014  consistent gets
          0  physical reads
          0  redo size
        841  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
```

2.3.2 Hash Join

```
SQL> SELECT /*+ leading(t1) use_hash(t2)*/ *
  2  FROM t1, t2
  3  WHERE t1.id = t2.t1_id
  4  AND t1.n = 19;

                                    可以看到在HASH JOIN下有排序
统计信息
----------------------------------------------------------
          7  recursive calls
          0  db block gets
       1081  consistent gets
          0  physical reads
          0  redo size
        841  bytes sent via SQL*Net to client
        384  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

2.3.3Merge Sort Join

```
SQL> SELECT /*+ leading(t1) use_merge(t2)*/ *
  2  FROM t1, t2
  3  WHERE t1.id = t2.t1_id
  4  AND t1.n = 19;
```

可以看到merge sort join下有排序

```
统计信息
----------------------------------------------------------
          7  recursive calls
          0  db block gets
       1080  consistent gets
          0  physical reads
          0  redo size
        841  bytes sent via SQL*Net to client
        384  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          4  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

3. 说说表连接优化有哪些要点

利用前面建好的环境，T1 和 T2 表。

3.1 Nested Loops 下的三把菜刀

3.1.1 驱动表的限制条件有索引

```
SQL> SELECT /*+ leading(t1) use_nl(t2) */ *
  2  FROM t1, t2
  3  WHERE t1.id = t2.t1_id
  4  AND t1.n = 19;
```

没有索引下的merge sort join

```
执行计划
----------------------------------------------------------
Plan hash value: 1967407726

---------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |  4069 |   227   (2)| 00:00:03 |
|   1 |  NESTED LOOPS      |      |     1 |  4069 |   227   (2)| 00:00:03 |
|*  2 |   TABLE ACCESS FULL| T1   |     1 |  2028 |     3   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| T2   |     1 |  2041 |   224   (2)| 00:00:03 |
---------------------------------------------------------------------------
```

```
SQL> CREATE INDEX t1_n ON t1 (n);

索引已创建。

SQL> SELECT /*+ leading(t1) use_nl(t2) */ *
  2  FROM t1, t2
  3  WHERE t1.id = t2.t1_id
  4  AND t1.n = 19;

执行计划
----------------------------------------------------------
Plan hash value: 76617097

-------------------------------------------------------------------------------------
| Id  | Operation                    | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |      |     1 |  4069 |   226   (2)| 00:00:03 |
|   1 |  NESTED LOOPS                |      |     1 |  4069 |   226   (2)| 00:00:03 |
|   2 |   TABLE ACCESS BY INDEX ROWID| T1   |     1 |  2028 |     2   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN          | T1_N |     1 |       |     1   (0)| 00:00:01 |
|*  4 |   TABLE ACCESS FULL          | T2   |     1 |  2041 |   224   (2)| 00:00:03 |
-------------------------------------------------------------------------------------
```

在驱动表的限制条件下创建索引可以到cost明显下将，虽然不明显，这是因为T1表数据量太小导致的不明显。

3.1.2 被驱动表的限制条件有索引

接着 3.1.1 的试验，我们再给被驱动表上加上索引。

```
SQL> CREATE INDEX t2_t1_id ON t2(t1_id);

索引已创建。

SQL> SELECT /*+ leading(t1) use_nl(t2) */ *
  2  FROM t1, t2
  3  WHERE t1.id = t2.t1_id
  4  AND t1.n = 19;

执行计划
----------------------------------------------------------
Plan hash value: 2669480776

-----------------------------------------------------------------------------------------
| Id  | Operation                     | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |          |     1 |  4069 |     5   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID  | T2       |     1 |  2041 |     3   (0)| 00:00:01 |
|   2 |   NESTED LOOPS                |          |     1 |  4069 |     5   (0)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID| T1       |     1 |  2028 |     2   (0)| 00:00:01 |
|*  4 |     INDEX RANGE SCAN          | T1_N     |     1 |       |     1   (0)| 00:00:01 |
|*  5 |    INDEX RANGE SCAN           | T2_T1_ID |     1 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------
```

在被驱动表上的限制条件上建立索引，可以看到COST值下将非常明显。

3.1.3 确保小结果集先驱动

小结果集先驱动的或者说小结果集作为驱动表的好处我们已经在 2.2.1 节中演示过了，在这里就不再截图。从 2.2.1 节中的演示结果来看小结果集作为驱动表可明显减少表的访问次数，提高表的连接效率。

3.2 Hash Join 下的三头斧

3.2.1 两表限制条件有索引

这一节与 3.1.1 和 3.1.2 相似，这里不再截图，其原理都是一样，使用索引可以提高表的访问速度，减少数据的访问量。

3.2.2 确保小结果集驱动与

3.1.3 类似，不再截图。

3.2.3 确保 PGA 中完成 HASH 运算的尺寸

我们知道两表在做 HASH 运算时，需要使用到 PGA 中的一块内存区域，这块内存区域的大小由参数 hash_area_size 指定，当然这块区域越大所能装载的数据量越多，一次排序的数据量也就越多，反之这块区域太小有可能会造成频繁的 IO，需要用到临时的表空间，造成排序效率降低。目前 10G 以上版本的数据库都已经实现了 PGA 的自动管理，包括 hash_area 在内的内存大小由系统自动进行调整，要在手动调整这块内存区域的大小，需要把 PGA 的管理该为手动模式，才可以手动调整其大小，目前还是建议采用自动管理模式。

3.3 Merge Sort Join 下的四式

3.3.1 两表限制条件有索引

不再截图，与上面类似。

3.3.2 连接条件索引消除排序

```
SQL> SELECT /*+ leading(t1) use_merge(t2)*/ *
  2  FROM t1, t2
  3  WHERE t1.id = t2.t1_id;

已选择100行。

执行计划
----------------------------------------------------------
Plan hash value: 412793182

-------------------------------------------------------------------------------------
| Id  | Operation            | Name | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |      |   100 |  397K|       | 37586   (1)| 00:07:32 |
|   1 |  MERGE JOIN          |      |   100 |  397K|       | 37586   (1)| 00:07:32 |
|   2 |   SORT JOIN          |      |   100 |  198K|       |     4  (25)| 00:00:01 |
|   3 |    TABLE ACCESS FULL| T1   |   100 |  198K|       |     3   (0)| 00:00:01 |
|*  4 |   SORT JOIN          |      | 87460 |  170M|   455M| 37582   (1)| 00:07:31 |
|   5 |    TABLE ACCESS FULL| T2   | 87460 |  170M|       |   224   (2)| 00:00:03 |
-------------------------------------------------------------------------------------
```

有两个排序

```
SQL> CREATE INDEX idx_t1_id ON t1(id);

索引已创建。

SQL> SELECT /*+ leading(t1) use_merge(t2)*/ *
  2  FROM t1, t2
  3  WHERE t1.id = t2.t1_id;

已选择100行。

执行计划
----------------------------------------------------------
Plan hash value: 2678642687

-----------------------------------------------------------------------------------------------
| Id  | Operation                    | Name      | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |           |   100 |  397K|       | 37584   (1)| 00:07:32 |
|   1 |  MERGE JOIN                  |           |   100 |  397K|       | 37584   (1)| 00:07:32 |
|   2 |   TABLE ACCESS BY INDEX ROWID| T1        |   100 |  198K|       |     2   (0)| 00:00:01 |
|   3 |    INDEX FULL SCAN           | IDX_T1_ID |   100 |      |       |     1   (0)| 00:00:01 |
|*  4 |   SORT JOIN                  |           | 87460 |  170M|   455M| 37582   (1)| 00:07:31 |
|   5 |    TABLE ACCESS FULL         | T2        | 87460 |  170M|       |   224   (2)| 00:00:03 |
-----------------------------------------------------------------------------------------------
```

在T1上建立了索引，执行计划里看不到对T1的排序

```
SQL> CREATE INDEX idx_t2_t1_id ON t2(t1_id);

索引已创建。

SQL> SELECT /*+ leading(t1) use_merge(t2)*/ *
  2  FROM t1, t2
  3  WHERE t1.id = t2.t1_id;

已选择100行。

执行计划
----------------------------------------------------------
Plan hash value: 2678642687

-----------------------------------------------------------------------------------------------
| Id  | Operation                    | Name      | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |           |   100 |  397K|       | 37584   (1)| 00:07:32 |
|   1 |  MERGE JOIN                  |           |   100 |  397K|       | 37584   (1)| 00:07:32 |
|   2 |   TABLE ACCESS BY INDEX ROWID| T1        |   100 |  198K|       |     2   (0)| 00:00:01 |
|   3 |    INDEX FULL SCAN           | IDX_T1_ID |   100 |      |       |     1   (0)| 00:00:01 |
|*  4 |   SORT JOIN                  |           | 87460 |  170M|   455M| 37582   (1)| 00:07:31 |
|   5 |    TABLE ACCESS FULL         | T2        | 87460 |  170M|       |   224   (2)| 00:00:03 |
-----------------------------------------------------------------------------------------------
```

在t2表的连接条件建索引，发现排序依然有一个，无法消除

3.3.3 避免取多余列致排序尺寸过大

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------
------------------------------------------------------
SQL_ID  5bxazj15c33ga, child number 0                                 取所有字段，耗费内存
-------------------------------------
SELECT /*+ leading(t2) use_merge(t1)*/ * FROM t1, t2 WHERE t1.id = t2.t1_id and t1.n=19

Plan hash value: 1792967693

--------------------------------------------------------------------------------------------------------------
| Id  | Operation            | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |  OMem |  1Mem | Used-Mem |
--------------------------------------------------------------------------------------------------------------
|   1 |  MERGE JOIN          |      |      1 |      1 |      1 |00:00:00.12 |    1012 |       |       |          |
|   2 |   SORT JOIN          |      |      1 |  87460 |     20 |00:00:00.12 |    1005 |  9124K|  1177K| 8110K (0)|

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------
------------------------------------------------------
|   3 |    TABLE ACCESS FULL| T2   |      1 |  87460 |    100K|00:00:00.01 |    1005 |       |       |          |
|*  4 |   SORT JOIN          |      |     20 |      1 |      1 |00:00:00.01 |       7 |  2048 |  2048 | 2048  (0)|
|*  5 |    TABLE ACCESS FULL| T1   |      1 |      1 |      1 |00:00:00.01 |       7 |       |       |          |
--------------------------------------------------------------------------------------------------------------
```

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------
------------------------------------------------------
SQL_ID  grxc776tjz2cq, child number 0
-------------------------------------
SELECT /*+ leading(t2) use_merge(t1)*/ t1.id FROM t1, t2 WHERE t1.id = t2.t1_id and t1.n=19

Plan hash value: 1792967693                                 取部分字段耗费内存，明显会有下降

--------------------------------------------------------------------------------------------------------------
| Id  | Operation            | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |  OMem |  1Mem | Used-Mem |
--------------------------------------------------------------------------------------------------------------
|   1 |  MERGE JOIN          |      |      1 |      1 |      1 |00:00:00.10 |    1012 |       |       |          |
|   2 |   SORT JOIN          |      |      1 |  87460 |     20 |00:00:00.10 |    1005 |  1824K|   650K| 1621K (0)|

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------
------------------------------------------------------
|   3 |    TABLE ACCESS FULL| T2   |      1 |  87460 |    100K|00:00:00.01 |    1005 |       |       |          |
|*  4 |   SORT JOIN          |      |     20 |      1 |      1 |00:00:00.01 |       7 |  2048 |  2048 | 2048  (0)|
|*  5 |    TABLE ACCESS FULL| T1   |      1 |      1 |      1 |00:00:00.01 |       7 |       |       |          |
--------------------------------------------------------------------------------------------------------------
```

3.3.4 保证 PGA 尺寸

PGA 是私有全局区，一个会话所有的连接、排序、聚合运算等操作都是在 PGA 中运算的，PGA 的大小直接关系到用户会话的快慢（私有模式下），也是对用户最直接影响的内存区域，充足的 PGA 大小能够提高系统的会话效率，特别是数据排序方面的影响。