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	1.说说三大经典表连持	妾分别有什么限制	2
	1.1 环境准备		2
	1.2 Nested Loops	Join	2
	1.3 Hash Join		4
	1.4 Merge Sort Jo	oin	6
2.	说说三大经典表连	接的各自特性	7
	2.1 表的访问次数	[8
	2.1.1 Nested	Loops Join	8
	2.1.2 Hash J	oin	9
	2.1.3 Merge S	Sort Join	10
	2.2 表驱动顺序与	:性能	11
	2.2.1 Nested	Loops Join	11
	2.2.2 Hash Jo	oin	12
	2.1.3Merge S	ort Join	13
	2.3 是否排序		14
	2.3.1 Nested	Loops Join	14
	2.3.2 Hash Jo	oin	14
	2.3.3Merge S	ort Join	15
3.	说说表连接优化有明	那些要点	15
	3.1 Nested Loops	;下的三把菜刀	15
	3.1.1 驱动表	的限制条件有索引	15
	3.1.2 被驱动	表的限制条件有索引	16
		结果集先驱动	
	3.2 Hash Join 下的	的三头斧	17
	3.2.1 两表限	制条件有索引	17
	3.2.2 确保小	结果集驱动	17
	3.2.3 确保 PC	GA 中完成 HASH 运算的尺寸	17
	3.3 Merge Sort Jo	oin 下的四式	. 2 . 2 . 4 . 6 . 7 . 8 . 9 . 10 . 11 . 12 . 13 . 14 . 14 . 15 . 15 . 15 . 15 . 16 . 17 . 17 . 17 . 17 . 17 . 17
		条件有索引	
	3.3.2 连接条件	‡索引消除排序	18
	3.3.3 避免取多	5余列致排序尺寸过大	19
	3.3.4 保证 PC	GA 尺寸	19

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 <
          id NUMBER NOT NULL,
          n NUMBER,
 3
 4
          contents VARCHAR2(4000)
 5
表已创建。
SQL> CREATE TABLE t2 (
          id NUMBER NOT NULL,
          t1_id NUMBER NOT NULL,
 3
          n NUMBER,
 4
          contents VARCHAR2(4000)
 5
表已创建。
SQL> execute dbms_random.seed(0);
PL/SQL 过程已成功完成。
SQL> INSERT INTO t1
          SELECT rownum, rownum, dbms_random.string('a', 50)
            FROM dual
 3
  4
          CONNECT BY level <= 100
           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;
是交完成。
```

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
       AND t1.n = 19;
执行计划
Plan hash value: 1967407726
                           ! Name ! Rows ! Bytes ! Cost (%CPU)! Time
| Id | Operation
                                                             <2>! 00:00:03 !
<2>! 00:00:03 :
<0>! 00:00:01 !
   0 : SELECT STATEMENT
                                   1 4373 |
                                                       227
                                               16M1
        NESTED LOOPS :
TABLE ACCESS FULL: T1
TABLE ACCESS FULL: T2
                                     4373 |
                                              16M:
2028 |
                                                       227
                                   4373
                                                             (2): 00:00:03:
                                              8716K!
SQL> SELECT /*+ leading(t1) use_nl(t2) */ *
2 FROM t1, t2
 3
       WHERE t1.id ( t2.t1_id
      AND t1.n = 19;
 4
                                 各循环支持小于运
执行计划
Plan hash value: 1967407726
                          ! Name ! Rows ! Bytes ! Cost (%CPU)! Time
| Id | Operation
   0 | SELECT STATEMENT | 4373 |
                                               16M:
                                                      227
                                                            (2) | 00:00:03 |
   1 | NESTED LOOPS
                                  | 4373 |
                                             16M1
                                                      227
                                                            (2)| 00:00:03
                           TABLE ACCESS FULL: T1
TABLE ACCESS FULL: T2
                                             2028 |
!* 2 !
                                      1 :
                                                            (0):00:00:01 :
                                  | 4373 | 8716K|
                                                      224
                                                            (2): 00:00:03
FROM t1, t2
 2
 3
       WHERE t1.id = t2.t1_id
       AND t1.n = 19;
 4
执行计划
Plan hash value: 1967407726
| Id | Operation
                           | Name | Rows | Bytes | Cost (xCPU)| Time
   0 | SELECT STATEMENT |
                                         1 | 4069 |
                                                       227
                                                              (2) | 00:00:03 |
   1 | NESTED LOOPS
                                        1 | 4069 |
                                                             (2) | 00:00:03 |
                                                       227
|* 2 |
         TABLE ACCESS FULL! T1
                                        1 | 2028 |
                                                        3
                                                             (0): 00:00:01 :
          TABLE ACCESS FULL: T2
                                         1 | 2041 |
                                                       224
                                                              (2): 00:00:03 :
FROM t1, t2
 2
       WHERE t1.id (> t2.t1_id
  3
       AND t1.n = 19;
执行计划
Plan hash value: 1967407726
| Id | Operation
                     ! Name ! Rows ! Bytes ! Cost (%CPU)! Time
                           | | 87459 |
| | 82450 |
                                                       227
                                                             (2): 00:00:03 :
   0 : SELECT STATEMENT
                                               339M!
                                 | 87459 |
   1 | NESTED LOOPS
                                             339M1
                                                       227
                                                             (2) | 00:00:03 |
        TABLE ACCESS FULL: T1
                                                             (0):00:00:01:
                                  1 1
                                              2028 |
                                                       3
*
   2 !
          TABLE ACCESS FULL: T2 : 87459 :
                                             170M:
                                                             (2) | 00:00:03 |
   3 :
                                                       224
```

```
FROM t1, t2
 2
 3
      WHERE t1.id like t2.t1_id
      AND t1.n = 19;
执行计划
Plan hash value: 1967407726
| Id | Operation
                        ! Name ! Rows ! Bytes ! Cost (%CPU)! Time
   0 : SELECT STATEMENT
                                4373 |
                                         16M1
                                                227
                                                     (2) | 00:00:03 |
                                4373 1
                                                     (2): 00:00:03
   1 ! NESTED LOOPS
                                         16M1
                                                227
        TABLE ACCESS FULL: T1
                                        2028 1
                                                     (0): 00:00:01
   2 !
                                                3
        TABLE ACCESS FULL! T2
                                                     (2): 00:00:03
   3 1
                                       8716K
                                4373 |
                                                224
```

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)*/ *
      FROM t1, t2
 2
      WHERE t1.id > t2.t1_id
 3
      AND t1.n = 19;
                            大于运算没有用到hash jion
执行计划
Plan hash value: 1967407726
                          ! Name ! Rows ! Bytes ! Cost (xCPU)! Time
| Id | Operation
                                                         (2): 00:00:03 :
   0 : SELECT STATEMENT
                                4373 |
                                             16M!
                                                    227
   1 | NESTED LOOPS
                                                         (2): 00:00:03 :
                        .
                                   4373 |
                                             16M:
                                                    227
   2 |
         TABLE ACCESS FULL: T1
                                1
                                    1 1
                                           2028 :
                                                         (0): 00:00:01 :
                                                    3
         TABLE ACCESS FULL! T2
                                                         (2): 00:00:03 :
                                   4373 | 8716K!
                                                    224
SQL> SELECT /*+ leading(t1) use_hash(t2)*/ *
      FROM t1, t2
 2
 3
      WHERE t1.id 12.t1_id
      AND t1.n = 19;
                            小于运算用不到HASHIION
执行计划
Plan hash value: 1967407726
| Id | Operation
                          ! Name ! Rows ! Bytes ! Cost (xCPU)! Time
   0 : SELECT STATEMENT
                                   4373 |
                                                          (2) | 00:00:03 |
                          .
                                             16M1
                                                    227
   1 | NESTED LOOPS
                                   4373 |
                                             16M:
                                                    227
                                                         (2) | 00:00:03 |
                         2 | TABLE ACCESS FULL: T1
                                   1 1
                                                          (0): 00:00:01:
                                           2028 |
                                                    3
         TABLE ACCESS FULL: T2
                                   4373 |
                                           8716K:
                                                    224
                                                          (2) | 00:00:03 |
```

```
FROM t1, t2
 2
      WHERE t1.id <> t2.t1_id
      AND t1.n = 19;
                              于运算用不上HASHJION
执行计划
Plan hash value: 1967407726
| Id | Operation
                       ! Name ! Rows ! Bytes ! Cost (%CPU)! Time
   0 | SELECT STATEMENT |
                              1 87459 1
                                         339M1
                                                227
                                                      (2): 00:00:03
   1 | NESTED LOOPS
                              1 87459 1
                                         339M:
                                                227
                                                      (2): 00:00:03
* 2 ! TABLE ACCESS FULL! T1
* 3 ! TABLE ACCESS FULL! T2
                              1 1 2028 1
                                                 3
                                                      (0):00:00:01
                                                      (2): 00:00:03
                             1 87459 1
                                        170M:
                                                224
SQL> SELECT /*+ leading(t1) use_hash(t2)*/ *
     FROM t1, t2
 2
      WHERE t1.id like t2.t1_id
 3
      AND t1.n = 19;
                       like运算用不上HASH JION
执行计划
Plan hash value: 1967407726
                        ! Name ! Rows ! Bytes ! Cost (%CPU)! Time
| Id | Operation
   0 | SELECT STATEMENT
                              1 4373 1
                                         16M!
                                                227
                                                     (2) | 00:00:03 |
   1 | NESTED LOOPS
                                                     (2) | 00:00:03 |
                      1
                                 4373 |
                                         16M1
                                                227
   2 |
        TABLE ACCESS FULL! T1
                                       2028 1
                                                     (0): 00:00:01 :
                                                3
        TABLE ACCESS FULL: T2
                             | 4373 | 8716K|
                                                224
                                                     (2): 00:00:03 :
SQL> SELECT /*+ leading(t1) use_hash(t2)*/ *
      FROM t1, t2
 3
      WHERE t1.id = t2.t1_id
      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
                        ! Name ! Rows ! Bytes ! Cost (xCPU)! Time
| Id | Operation
  (3): 00:00:03 :
                                                229
                                                     (3): 00:00:03 :
                                                229
                                        2028 |
                                                     (0): 00:00:01:
                                                3
        TABLE ACCESS FULL: T2 : 87460 :
                                       170M:
                                                224
                                                      (2) | 00:00:03 |
```

1.4 Merge Sort Join

```
FROM t1, t2
      WHERE t1.id > t2.t1_id
 3
 4
      AND t1.n = 19;
                                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
     : SELECT STATEMENT
                                     4373 |
                                              16M:
                                                          1 37586
                                                                   (1): 00:07:32 :
   Ø
     MERGE JOIN
                                                                   (1): 00:07:32
   1
                                     4373 |
                                              16M:
                                                           37586
        SORT JOIN
                                             2028 :
                                                                   (25): 00:00:01
   2
                                       1 :
   3
         TABLE ACCESS FULL: T1
                                       1 1
                                             2028 :
                                                                   (0): 00:00:01
!*
                                                               3
   4
         SORT JOIN
                                    87460
                                             170M:
                                                      455M: 37582
                                                                   (1): 00:07:31
                                                                   (2): 00:00:03
          TABLE ACCESS FULL: T2
   5
                                    87460 |
                                              170M:
                                                             224
SQL> SELECT /*+ leading(t1) use_merge(t2)*/ *
      FROM t1, t2
 2
      WHERE t1.id < t2.t1_id
      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:
                                                                    (1): 00:07:32
                                                          1 37586
                                     4373
                                                           37586
                                                                    (1): 00:07:32
   1 | MERGE JOIN
                                               16M:
   2
        SORT JOIN
                                             2028 :
                                                                   (25): 00:00:01
         TABLE ACCESS FULL! T1
   3
                                             2028 :
                                                                    (0): 00:00:01
                                        1 !
                                                                3
! *
         SORT JOIN
                                  1 87460 1
                                              170M:
                                                      455M: 37582
                                                                    (1): 00:07:31
          TABLE ACCESS FULL! T2
                                                                    (2) | 00:00:03 |
   5 1
                                  1 87460 1
                                              170M!
                                                              224
SQL> SELECT /*+ leading(t1) use_merge(t2)*/ *
2 FROM t1, t2
      WHERE t1.id = t2.t1_id
      AND t1.n = 19;
                              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 1
                                                           37586
                                                                    (1): 00:07:32
                                                                    (1): 00:07:32
   1 | MERGE JOIN
                                        1 :
                                             4069 1
                                                           37586
         SORT JOIN
                                        1 1
                                             2028 :
                                                                   (25): 00:00:01
   2 !
   3 1
          TABLE ACCESS FULL! T1
                                             2028 1
                                                                    (0)| 00:00:01
                                        1 |
                                                               3
         SORT JOIN
                                    87460 |
                                              170M:
                                                      455M: 37582
                                                                    (1): 00:07:31
          TABLE ACCESS FULL: T2
                                                                    (2): 00:00:03
                                  1 87460 1
                                              170M:
                                                             224
```

```
SQL> SELECT /*+ leading(t1) use_merge(t2)*/ *
      FROM t1, t2
 3
      WHERE t1.id (>) t2.t1_id
      AND t1.n = 19;
执行计划
Plan hash value: 1967407726
! Id ! Operation
                           ! Name ! Rows ! Bytes ! Cost (%CPU)! Time
   0 | SELECT STATEMENT
                                  1 87459 1
                                              339M1
                                                      227
                                                            (2) | 00:00:03 |
   1 | NESTED LOOPS
                           1
                                  1 87459 1
                                              339M1
                                                      227
                                                            (2) | 00:00:03 |
                                                      3
   2 | TABLE ACCESS FULL: T1
                                      1 :
                                             2028 |
                                                            (0): 00:00:01 ;
         TABLE ACCESS FULL: T2
                                  1 87459 1
                                              170M:
                                                      224
                                                            (2) | 00:00:03 |
```

```
SQL> SELECT /*+ leading(t1) use_merge(t2)*/ *
      FROM t1, t2
WHERE t1.id like t2.t1_id
      AND t1.n = 19;
                           like运算用不上merge sort join
执行计划
Plan hash value: 1967407726
 Id | Operation
                           ! Name ! Rows ! Bytes ! Cost (%CPU)! Time
   0 : SELECT STATEMENT
                                     4373 |
                           ı
                                               16M1
                                                       227
                                                             (2): 00:00:03 :
   1 | NESTED LOOPS
                                     4373 |
                                                             (2) | 00:00:03 |
                                               16M:
                                                       227
   2 |
         TABLE ACCESS FULL! T1
                                     1 :
                                             2028 |
                                                       3
                                                             (0): 00:00:01 :
          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(*)

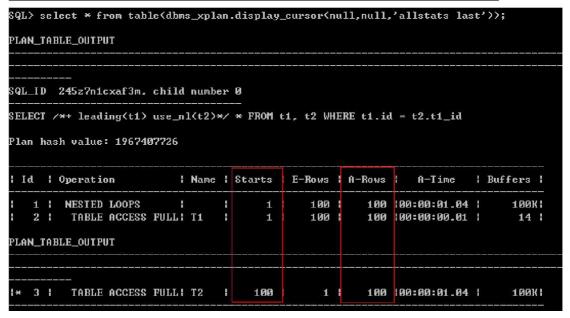
_______
100

SQL> select count(*) from t2;

COUNT(*)

_______
100000

SQL> SELECT /*+ leading(t1) use_n1(t2)*/ *
2 FROM t1, t2
3 WHERE t1.id = t2.t1_id;
```



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;

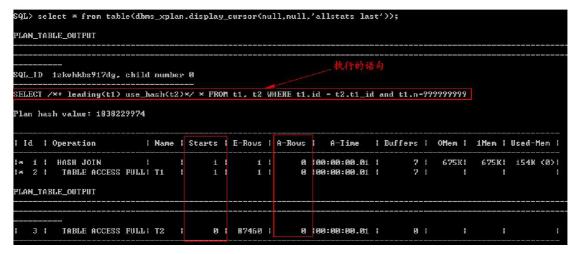
COUNT(*)
------
100

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;
```

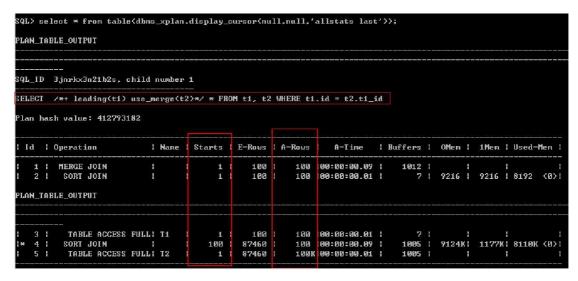


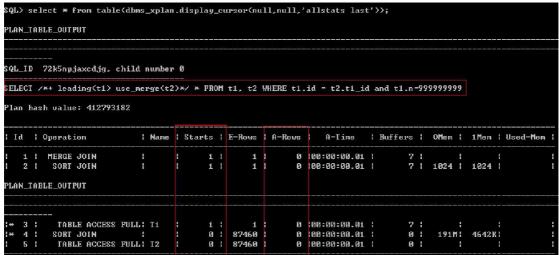


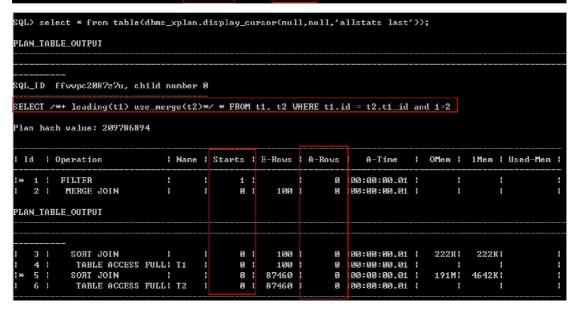
HASH 连接中,驱动表被访问 0 或者 1 次,被驱动表也是被访问 0 次或者 1 次,

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



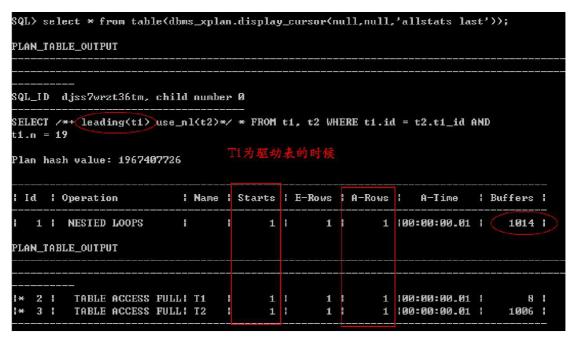


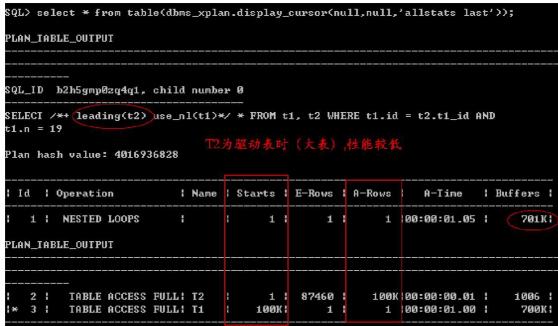


结论:排序合并连接中,两表都是只被访问 0 次或者 1 次,和 HASH 连接一样。

2.2 表驱动顺序与性能

2.2.1 Nested Loops Join





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

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
lan hash value: 412793182
 Id | Operation
                            ! Name ! Starts ! E-Rows ! A-Rows ! A-Time ! Buffers ! OMem ! 1Mem ! Used-Mem
        MERGE JOIN
SORT JOIN
                                                                                                 2048 | 2048 (0)|
                                                             1 |00:00:00.01 |
LAN_TABLE_OUTPUT
          TABLE ACCESS FULL: I1
                                                             1 100:00:00.01
                                                                                         9124Ki 1177Ki 8110K (0)
         SORT JOIN : TABLE ACCESS FULL: T2
                                                               100:00:00.09
                                                           1008:00:00:00.01
```



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
统计信息
         Ø
            recursive calls
         Ø
            db block gets
      1014
            consistent gets
            physical reads
            redo size
            bytes sent via SQL*Net to client
       841
            bytes received via SQL*Net from client
       384
         2 SQL*Net roundtrips to/from client
            sorts (memory)
       Ø sorts (disk)
         1
            rows processed
```

2.3.2 Hash Join

```
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
         db block gets
     1081
          consistent gets
          physical reads
         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)
     Ø sorts (disk)
       1 rows processed
```

2.3.3Merge Sort Join

```
SQL> SELECT /*+ leading(t1) use_merge(t2)*/ *
    FROM t1, t2
    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
           recursive calls
         0 db block gets
      1080 consistent gets
            physical reads
         Ø
           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)
        Ø 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_n1(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
                          ! Name ! Rows ! Bytes ! Cost (%CPU)! Time
| Id | Operation
   0 : SELECT STATEMENT
                                                           (2): 00:00:03
                                       1 1
                                            4069 1
                                                     227
   1 | NESTED LOOPS
                                       1 1
                                            4069 1
                                                           (2) | 00:00:03 |
                                                     227
   2 |
         TABLE ACCESS FULL! T1
                                            2028 1
                                                           (0): 00:00:01 :
                                       1 !
                                                      3
   3 !
         TABLE ACCESS FULL! T2
                                       1 1
                                            2041 1
                                                     224
                                                           (2): 00:00:03
```

```
SQL> CREATE INDEX t1_n ON t1 (n);
索引已创建。
SQL> SELECT /*+ leading(t1) use_n1(t2) */ *
 2 FROM t1, t2
    WHERE t1.id = t2.t1_id
    AND t1.n = 19;
执行计划
Plan hash value: 76617097
 Id | Operation
                                     ! Name ! Rows ! Bytes ! Cost (%CPU)! Time
   0 : SELECT STATEMENT
                                                 1 1
                                                      4069 1
                                                               226
                                                                     (2): 00:00:03
        NESTED LOOPS
                                                 1 :
                                                      4069
                                                               226
                                                                     (2): 00:00:03
   1
         TABLE ACCESS BY INDEX ROWID: T1
   2
                                                                     (0): 00:00:01
                                                      2028
                                                                 2
   3 1
          INDEX RANGE SCAN
                                    : T1_N :
                                                 1 1
                                                                     (0): 00:00:01
                                                                 1
         TABLE ACCESS FULL
    4 !
                                     1 T2
                                                 1 :
                                                      2041 :
                                                               224
                                                                     (2): 00:00:03
```

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_n1(t2) */ *
 2 FROM t1, t2
    WHERE t1.id = t2.t1_id
 4 AND t1.n = 19;
执行计划
Plan hash value: 2669480776
 Id | Operation
                                      1 Name
                                                 ! Rows ! Bytes ! Cost (xCPU)! Time
   0 : SELECT STATEMENT
                                                      1 :
                                                           4069 1
                                                                      5
                                                                          (0): 00:00:01
        TABLE ACCESS BY INDEX ROWID
                                                                          (0):00:00:01
                                       T2
                                                           2041
                                                                      3
   1
   2 !
         NESTED LOOPS
                                                      1 :
                                                           4069
                                                                           (0): 00:00:01 :
          TABLE ACCESS BY INDEX ROWID! T1
   3 1
                                                      1 :
                                                           2028
                                                                      2
                                                                          (0):00:00:01
           INDEX RANGE SCAN
                                      1 T1 N
                                                                      1
                                                                          (0): 00:00:01
          INDEX RANGE SCAN
   5
                                      ! T2_T1_ID !
                                                      1 1
                                                                      1
                                                                           (0): 00:00:01
```

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。

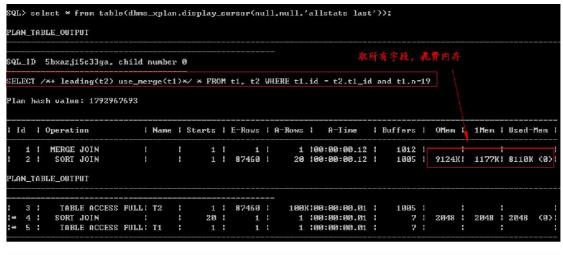
3.3.2 连接条件索引消除排序

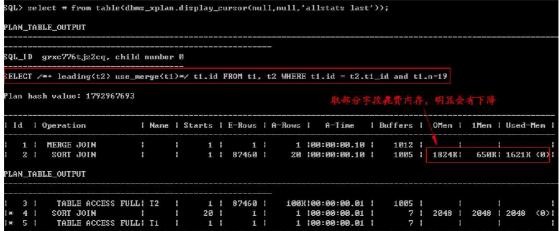
```
SQL> SELECT /*+ leading(t1) use_merge(t2)*/ *
 2 FROM t1, t2
3 WHERE t1.id = t2.t1_id;
已选择100行。
执行计划
Plan hash value: 412793182
 Id | Operation
                             | Name | Rows | Bytes | TempSpc | Cost (xCPU) | Time
     : SELECT STATEMENT
                                                                        (1): 00:07:32 :
   Ø
                                        100 :
                                                 397K:
                                                              1 37586
        MERGE JOIN
                                                 397K1
                                                                        (1): 00:07:32 :
                                        100
                                                               37586
   1
   2 |
         SORT JOIN
                                        100
                                                 198K!
                                                                   4
                                                                       (25) | 00:00:01 |
          TABLE ACCESS FULL: T1
                                        100
                                                                        (0):00:00:01
   3
                                                 198K!
                                                                    3
          SORT JOIN
   4
                                                 170M:
                                                         455M1 37582
                                                                        (1): 00:07:31
                                      87460
                                                                                       В
          TABLE ACCESS FULL! T2
                                      87460
                                                                        (2)| 00:00:03
    5
                                                 170M1
                                                                  224
```

```
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。
                                     在T1上建立了索引,执行计划里看不到对T1的制
执行计划
Plan hash value: 2678642687
                                                    | Rows | Bytes |TempSpc| Cost (xCPU)| Time
 Id | Operation
                                       : Name
        SELECT STATEMENT
                                                        100
                                                                                         (1): 00:07:32
    П
                                                                 397K:
                                                                                37584
                                                                                        <1>; 00:07:32
<1>; 00:07:32
<0>; 00:00:01
         MERGE JOIN
                                                        100
                                                                 397K:
                                                                                37584
          TABLE ACCESS BY INDEX ROWID:
                                                        100
                                                                 198KI
           INDEX FULL SCAN
                                         IDX T1 ID
                                                        100
                                                                                         (0) | 00:00:01
                                                                                         (1) | 00:07:31
(2) | 00:00:03
          SORT JOIN
                                                      87460
                                                                 170M
                                                                         455M1 37582
           TABLE ACCESS FULL
                                         T2
                                                      87460
```

```
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
                                                                                       (1): 00:07:32 :
                                                       100 :
                                                                397K1
                                                                              37584
         MERGE JOIN :
TABLE ACCESS BY INDEX ROWID: T1
INDEX FULL SCAN : 1Ds
                                                                                       (1): 00:07:32
                                                       100 |
                                                                397K1
                                                                              37584
                                                                                       (0) | 00:00:01
   2
                                                       100
                                                                198KI
                                                                                  2
   3
                                      : IDX T1 ID
                                                                                       (0):00:00:01
                                                       100
                                                                                   1
                                                                                       (1): 00:07:31
                                                     87460
          SORT JOIN
                                                                179M:
                                                                        455M: 37582
          TABLE ACCESS FULL
   5
                                       1 T2
                                                                                       (2)| 00:00:03
                                                   87460
                                                                170M:
                                                                                224
```

3.3.3 避免取多余列致排序尺寸过大





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。