收获，不止 SQL 优化

第 八 章

且慢，学习索引是如何让 SQL 飞

E-Mail:45240040@qq.com

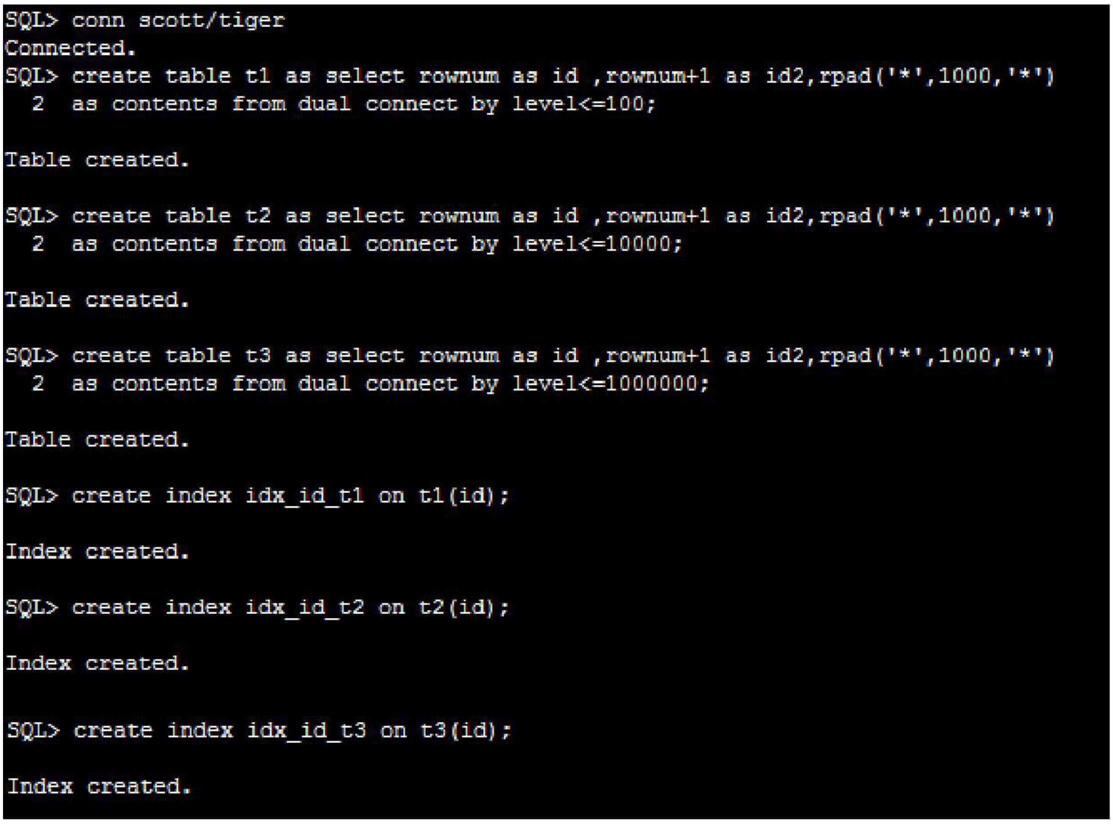
目录

1.解答第一题

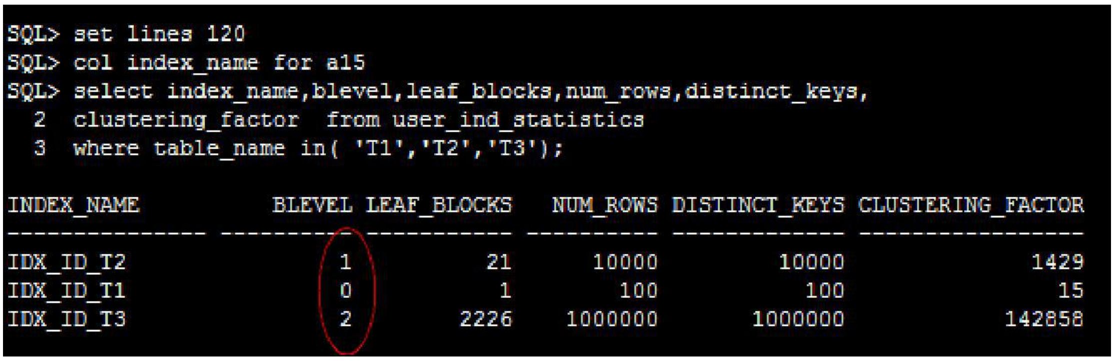
◆ 说说老师课堂的索引三大特性是什么，能应用在哪些 SQL 上。

1.1 索引高度较低

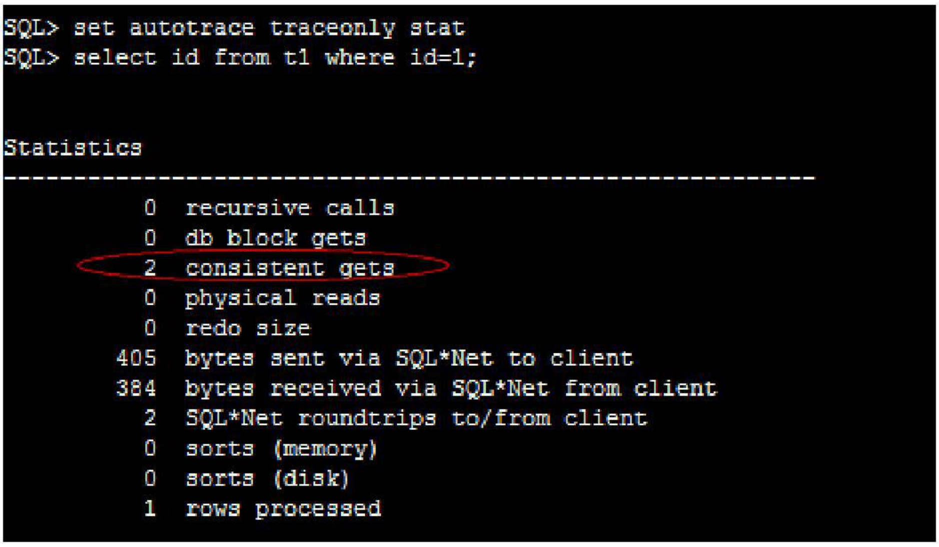
在 SQL 检索数据(SELECT)的时候,索引的高度的不同对检索的效率有明显的差别，数据库访问索引需要读取的数据块通常是索引的高度+1 个数据块数，也就是说索引的高度越高，访问索引需要读取的数据块数越多，效率越差。



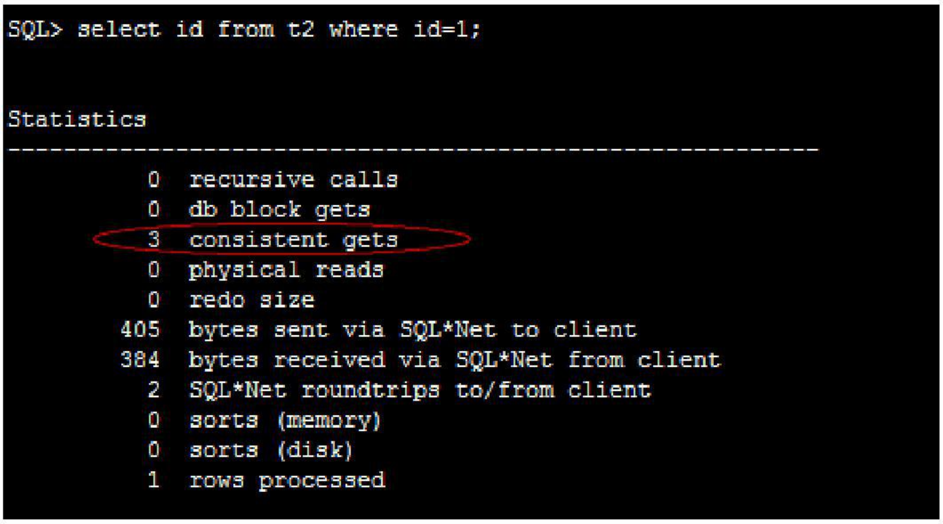
```
SQL> conn scott/tiger
Connected.
SQL> create table t1 as select rownum as id ,rownum+1 as id2,rpad('*',1000,'*')
  2  as contents from dual connect by level<=100;

Table created.

SQL> create table t2 as select rownum as id ,rownum+1 as id2,rpad('*',1000,'*')
  2  as contents from dual connect by level<=10000;

Table created.

SQL> create table t3 as select rownum as id ,rownum+1 as id2,rpad('*',1000,'*')
  2  as contents from dual connect by level<=1000000;

Table created.

SQL> create index idx_id_t1 on t1(id);

Index created.

SQL> create index idx_id_t2 on t2(id);

Index created.

SQL> create index idx_id_t3 on t3(id);

Index created.
```



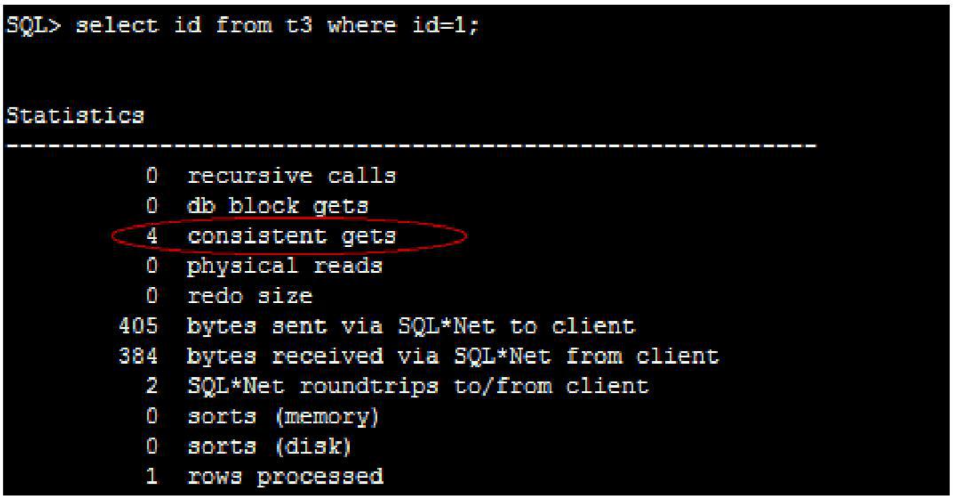
```
SQL> set lines 120
SQL> col index_name for a15
SQL> select index_name,blevel,leaf_blocks,num_rows,distinct_keys,
  2  clustering_factor  from user_ind_statistics
  3  where table_name in( 'T1','T2','T3');

INDEX_NAME          BLEVEL LEAF_BLOCKS   NUM_ROWS DISTINCT_KEYS CLUSTERING_FACTOR
--------------- ---------- ----------- ---------- ------------- -----------------
IDX_ID_T2                1          21      10000         10000              1429
IDX_ID_T1                0           1        100           100                15
IDX_ID_T3                2        2226    1000000       1000000            142858
```



```
SQL> set autotrace traceonly stat
SQL> select id from t1 where id=1;


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          2  consistent gets
          0  physical reads
          0  redo size
        405  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
```



```
SQL> select id from t2 where id=1;


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          3  consistent gets
          0  physical reads
          0  redo size
        405  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
```

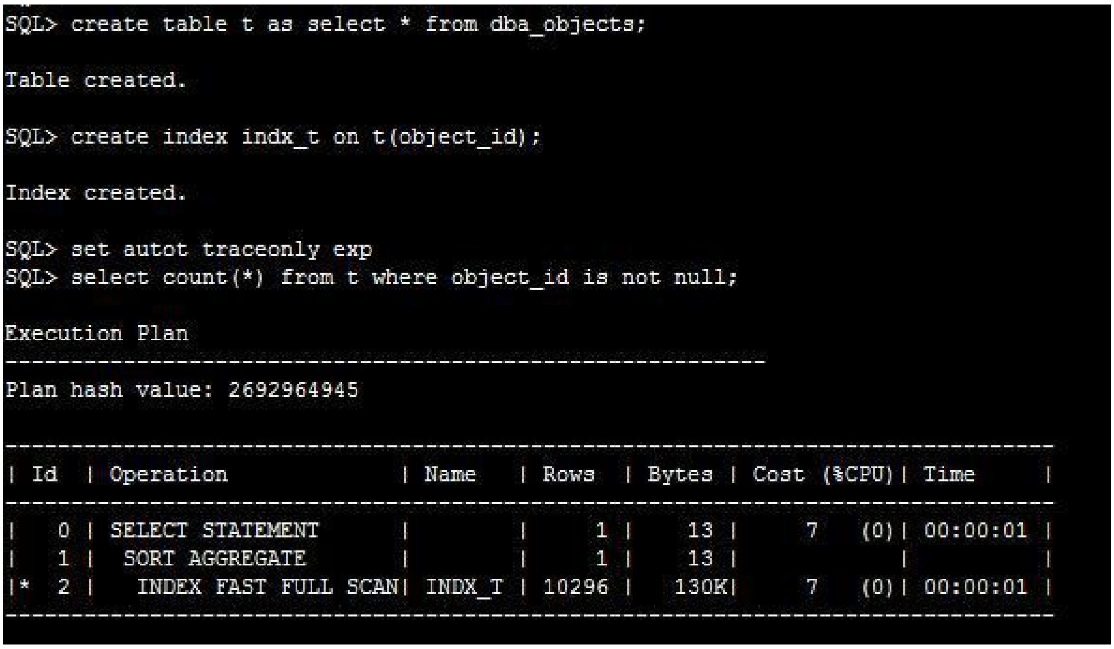


```
SQL> select id from t3 where id=1;


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          4  consistent gets
          0  physical reads
          0  redo size
        405  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
```

从上面的试验中可以看到，随着表索引高度的增加，其查询索引产生的一致性读也随之增加。

1.2 索引存储列值

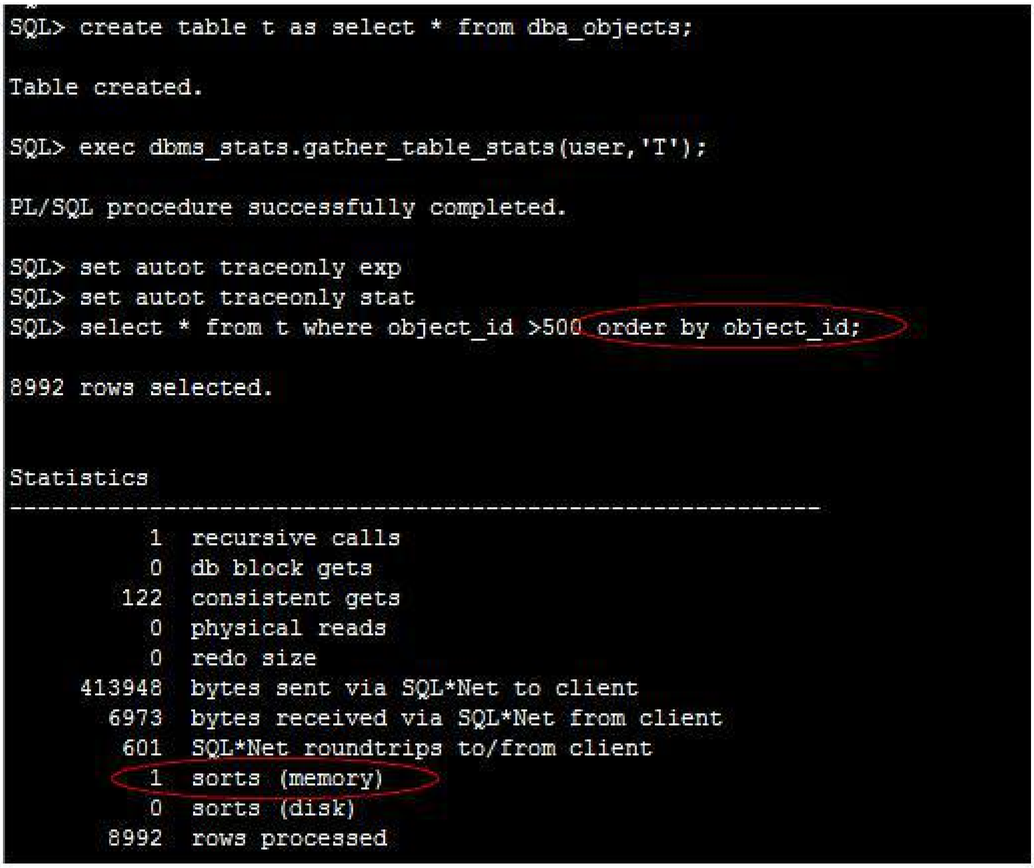
分析一个索引块我们可以知道索引块不仅存储了 rowid 信息，而且还存储了索引列的值，那么当我们查询的值正好是在索引里时或者做一些聚合计算（如sum,max,min）时，就可以利用这个特性。



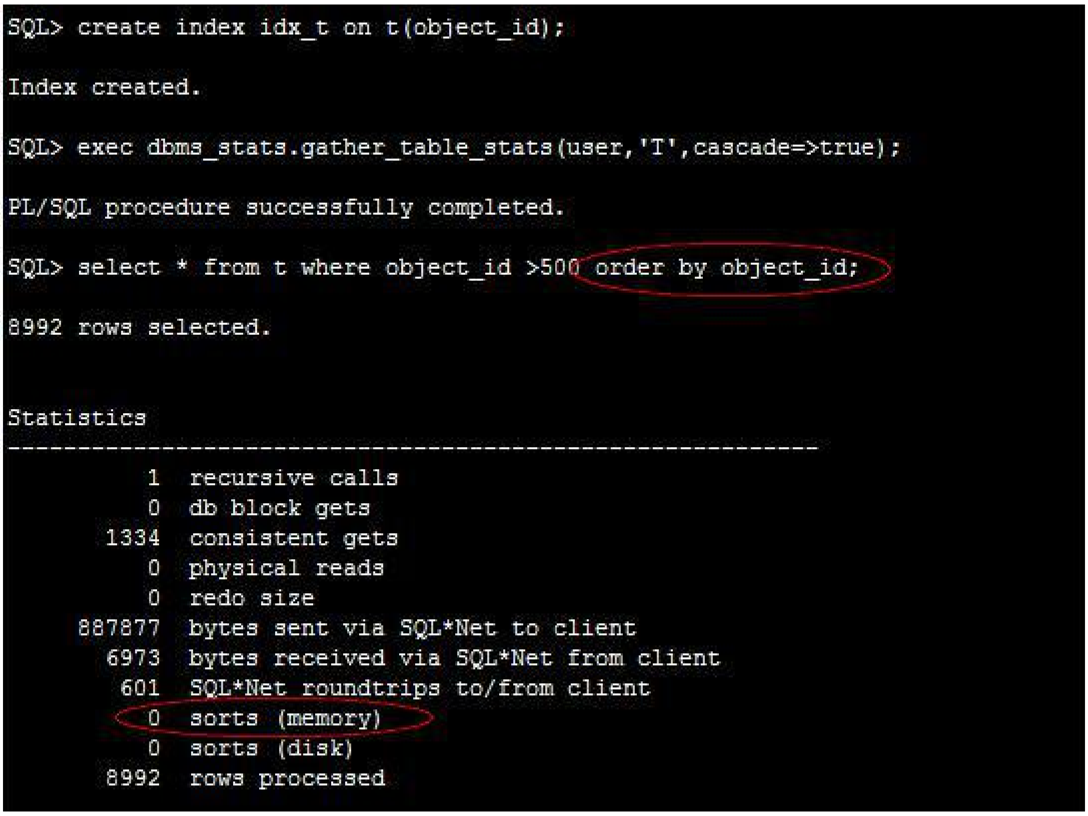
```
SQL> create table t as select * from dba_objects;

Table created.

SQL> create index indx_t on t(object_id);

Index created.

SQL> set autot traceonly exp
SQL> select count(*) from t where object_id is not null;

Execution Plan
----------------------------------------------------------
Plan hash value: 2692964945

--------------------------------------------------------------------------------
| Id  | Operation             | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |        |     1 |    13 |     7   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE       |        |     1 |    13 |            |          |
|*  2 |   INDEX FAST FULL SCAN| INDX_T | 10296 |   130K|     7   (0)| 00:00:01 |
--------------------------------------------------------------------------------
```

1.3 索引本身有序

从索引的存储结构上可以看到，索引的存储是有序存放的，扫描索引的时候是从根节点开始，经过颈节点到叶子节点，这个特点下索引的范围查询或等值查询，索引只需要扫描一段范围就可得出结果，因为其本身是有范围的，我们可用利用索引这个特点来降低实际查询的排序操作。



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

Table created.

SQL> exec dbms_stats.gather_table_stats(user,'T');

PL/SQL procedure successfully completed.

SQL> set autot traceonly exp
SQL> set autot traceonly stat
SQL> select * from t where object_id >500 order by object_id;

8992 rows selected.


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        122  consistent gets
          0  physical reads
          0  redo size
     413948  bytes sent via SQL*Net to client
       6973  bytes received via SQL*Net from client
        601  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
       8992  rows processed
```



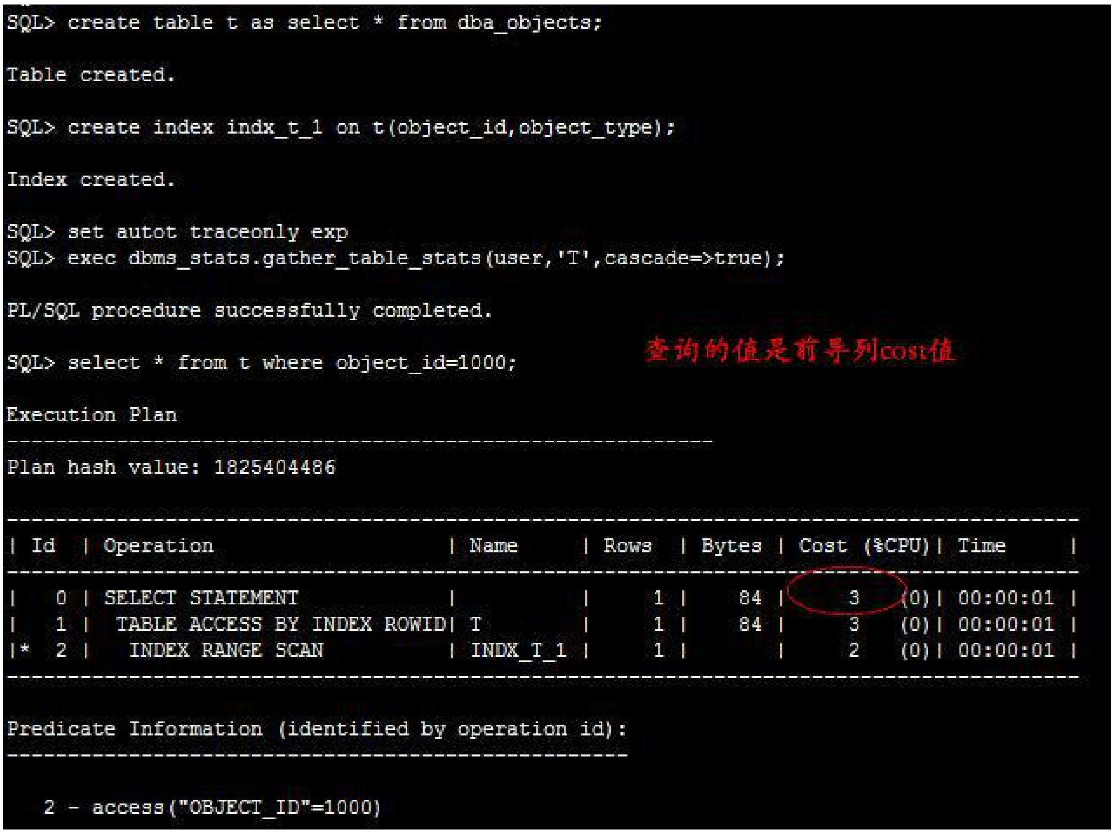
```
SQL> create index idx_t on t(object_id);

Index created.

SQL> exec dbms_stats.gather_table_stats(user,'T',cascade=>true);

PL/SQL procedure successfully completed.

SQL> select * from t where object_id >500 order by object_id;

8992 rows selected.


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       1334  consistent gets
          0  physical reads
          0  redo size
     887877  bytes sent via SQL*Net to client
       6973  bytes received via SQL*Net from client
        601  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       8992  rows processed
```

2.解答第二题

◆ 说说用组合索引需要考虑什么问题。

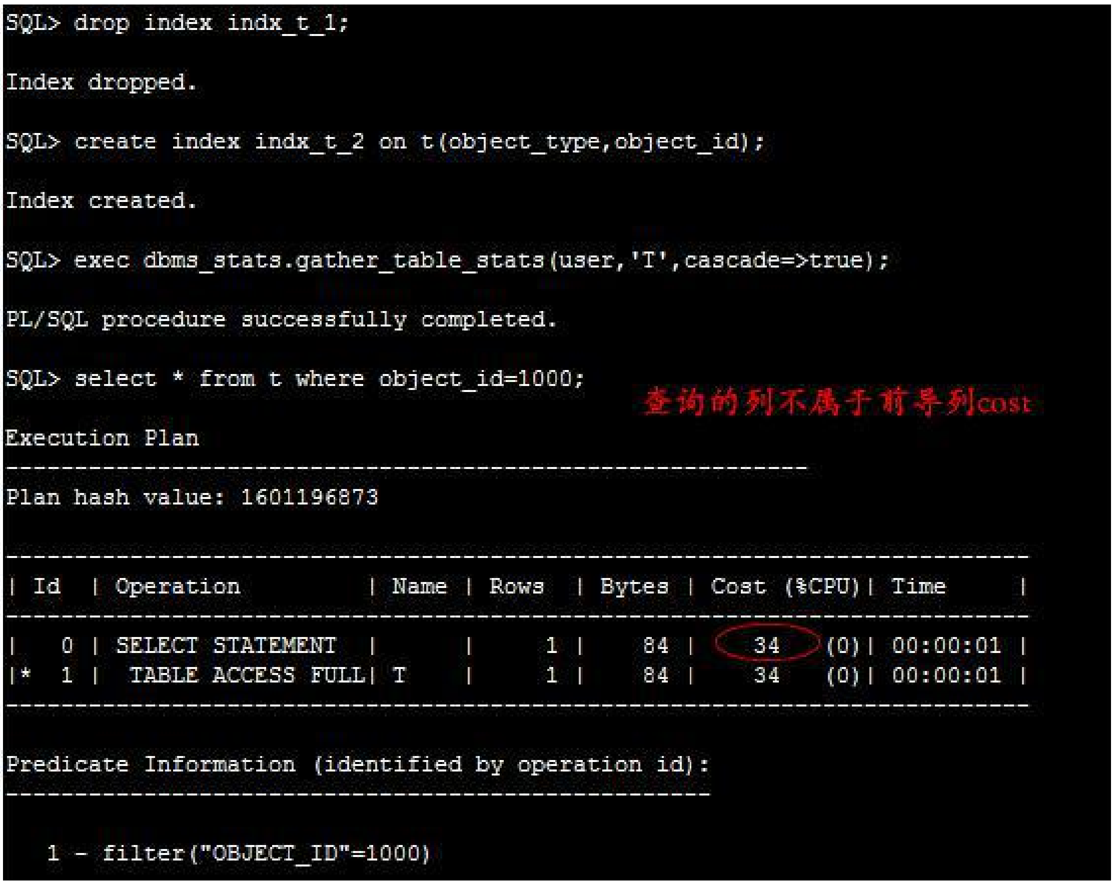
2.1 组合索引需要考虑单列

在创建组合索引的时候需要考虑到常会查询的单列索引，因为创建组合索引会有前导列的概念，在查询中最好应该是应到前导列，这样的查询效率会比较高。



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

Table created.

SQL> create index indx_t_1 on t(object_id,object_type);

Index created.

SQL> set autot traceonly exp
SQL> exec dbms_stats.gather_table_stats(user,'T',cascade=>true);

PL/SQL procedure successfully completed.

SQL> select * from t where object_id=1000;

Execution Plan
----------------------------------------------------------
Plan hash value: 1825404486

-----------------------------------------------------------------------------------------
| Id  | Operation                   | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |          |     1 |    84 |     3   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| T        |     1 |    84 |     3   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | INDX_T_1 |     1 |       |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

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

   2 - access("OBJECT_ID"=1000)
```

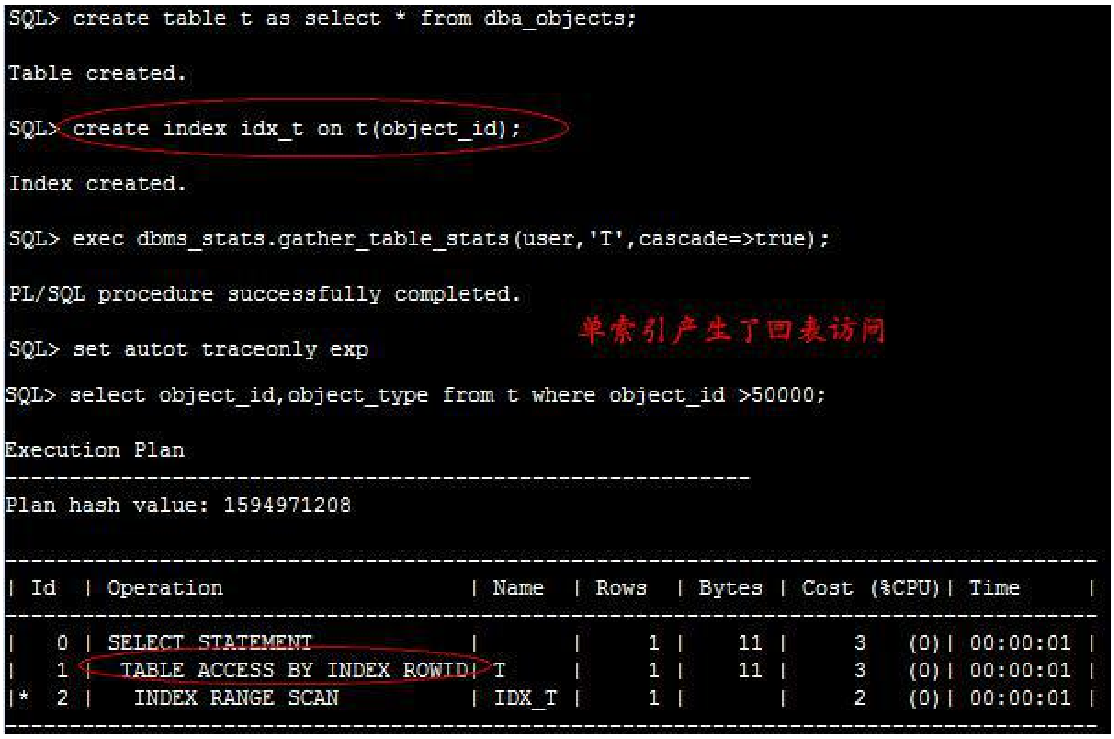
查询的值是前导列cost值



```
SQL> drop index indx_t_1;

Index dropped.

SQL> create index indx_t_2 on t(object_type,object_id);

Index created.

SQL> exec dbms_stats.gather_table_stats(user,'T',cascade=>true);

PL/SQL procedure successfully completed.

SQL> select * from t where object_id=1000;

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

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |     1 |    84 |    34   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| T    |     1 |    84 |    34   (0)| 00:00:01 |
--------------------------------------------------------------------------

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

   1 - filter("OBJECT_ID"=1000)
```

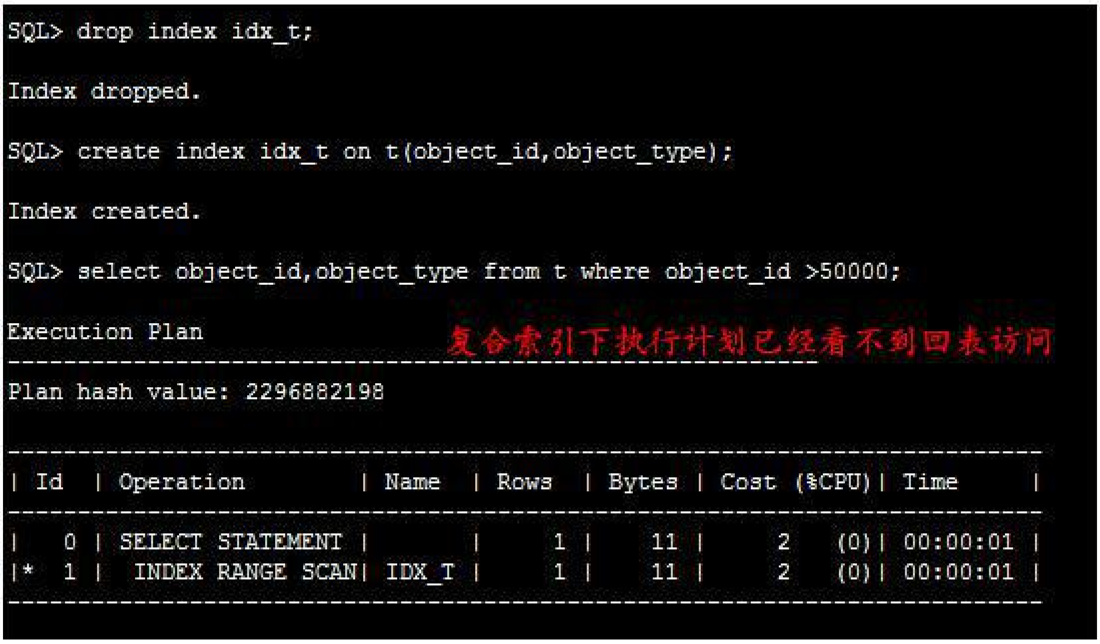
查询的列不属于前导列cost

2.2 组合索引需要考虑回表



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

Table created.

SQL> create index idx_t on t(object_id);

Index created.

SQL> exec dbms_stats.gather_table_stats(user,'T',cascade=>true);

PL/SQL procedure successfully completed.

SQL> set autot traceonly exp
SQL> select object_id,object_type from t where object_id >50000;

Execution Plan
----------------------------------------------------------
Plan hash value: 1594971208

-------------------------------------------------------------------------------------
| Id  | Operation                   | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |       |     1 |    11 |     3   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| T     |     1 |    11 |     3   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | IDX_T |     1 |       |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------
```

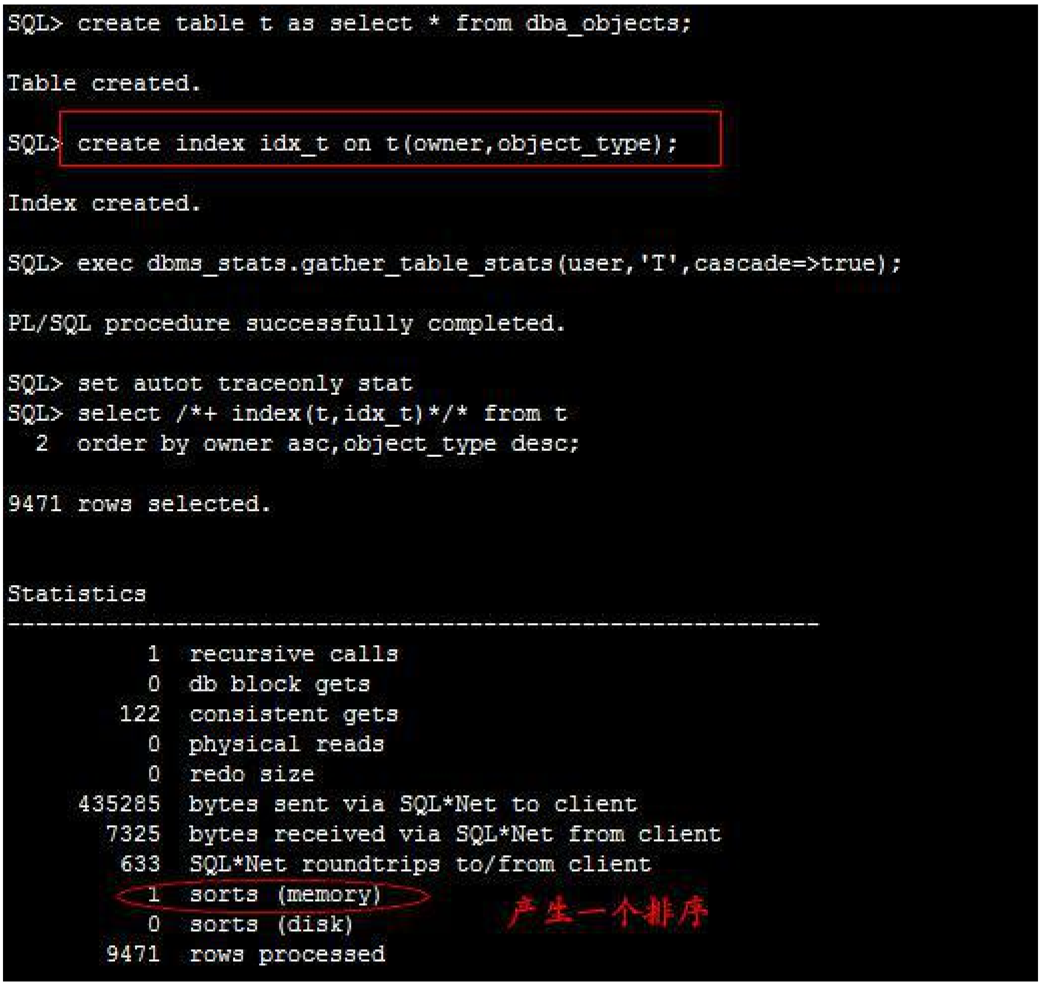
单索引产生了回表访问



```
SQL> drop index idx_t;

Index dropped.

SQL> create index idx_t on t(object_id,object_type);

Index created.

SQL> select object_id,object_type from t where object_id >50000;

Execution Plan
----------------------------------------------------------
Plan hash value: 2296882198

--------------------------------------------------------------------------
| Id  | Operation        | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT |       |     1 |    11 |     2   (0)| 00:00:01 |
|*  1 |  INDEX RANGE SCAN| IDX_T |     1 |    11 |     2   (0)| 00:00:01 |
--------------------------------------------------------------------------
```

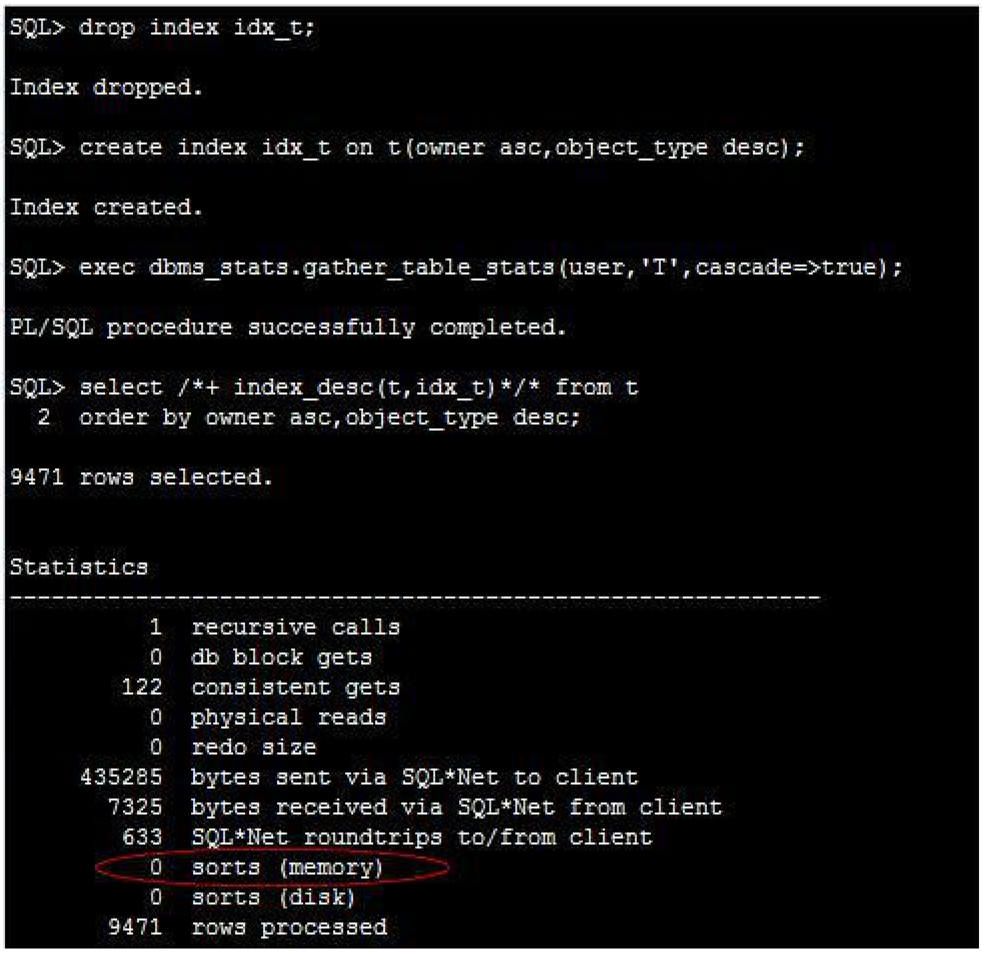
复合索引下执行计划已经看不到回表访问

2.3 组合索引需要考虑排序



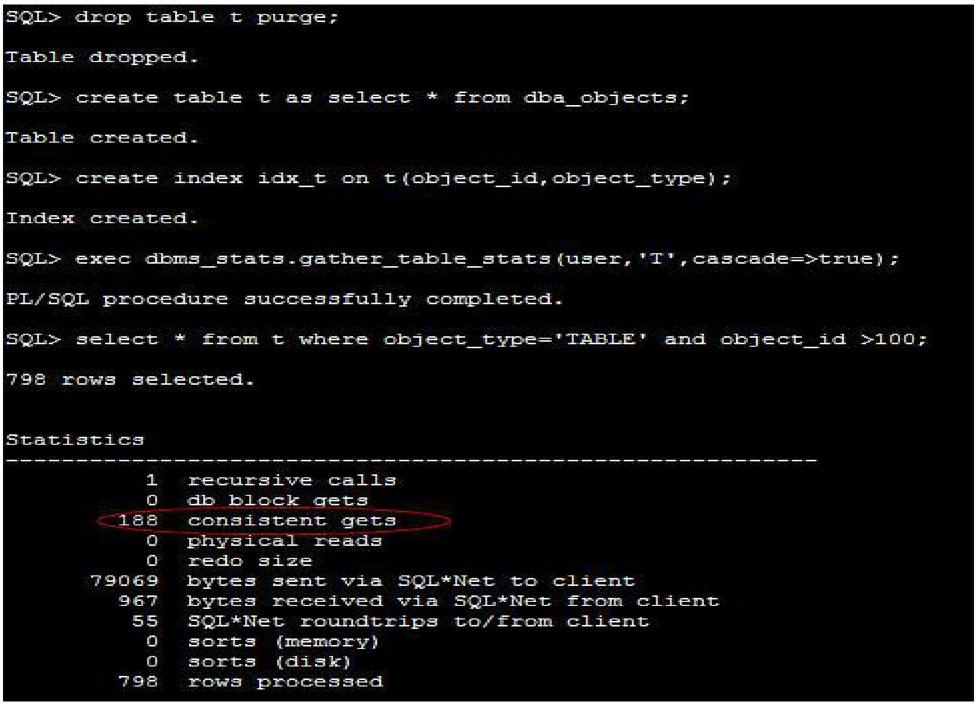
```
SQL> create table t as select * from dba_objects;

Table created.

SQL> create index idx_t on t(owner,object_type);

Index created.

SQL> exec dbms_stats.gather_table_stats(user,'T',cascade=>true);

PL/SQL procedure successfully completed.

SQL> set autot traceonly stat
SQL> select /*+ index(t,idx_t)*/* from t
  2  order by owner asc,object_type desc;

9471 rows selected.


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        122  consistent gets
          0  physical reads
          0  redo size
     435285  bytes sent via SQL*Net to client
       7325  bytes received via SQL*Net from client
        633  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
       9471  rows processed
```

产生一个排序

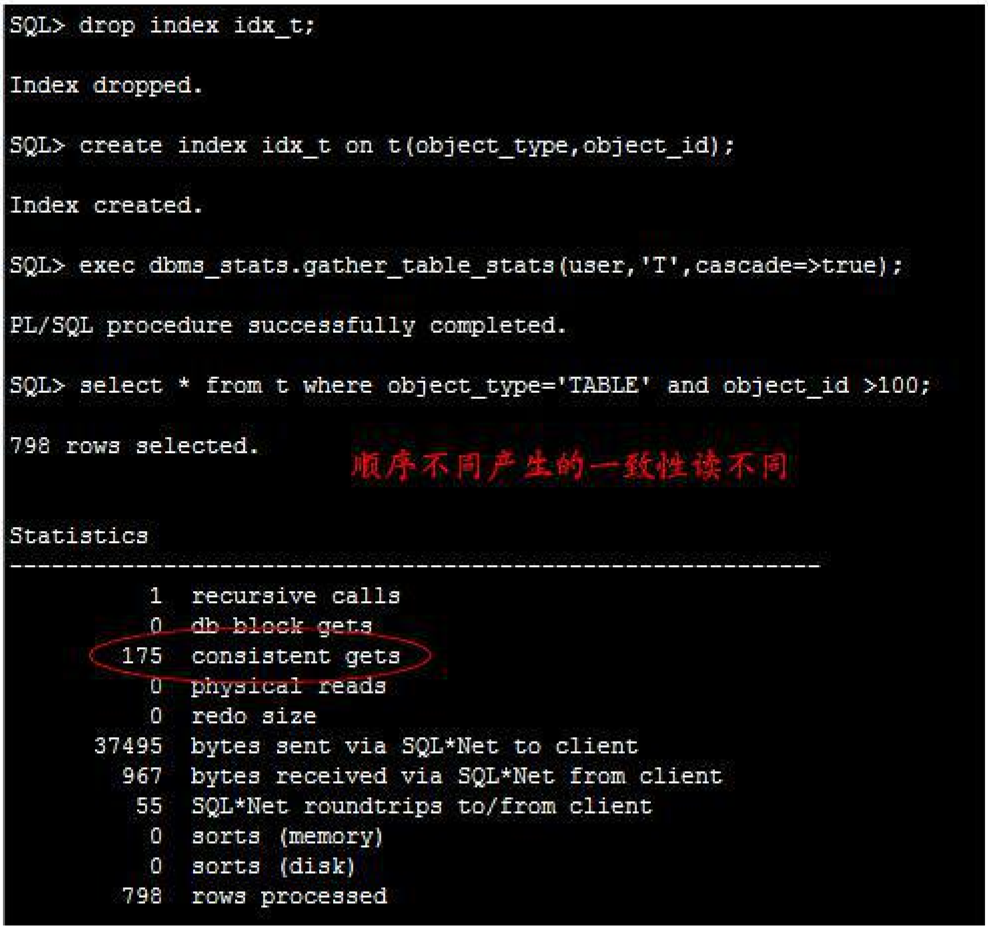


```
SQL> drop index idx_t;

Index dropped.

SQL> create index idx_t on t(owner asc,object_type desc);

Index created.

SQL> exec dbms_stats.gather_table_stats(user,'T',cascade=>true);

PL/SQL procedure successfully completed.

SQL> select /*+ index_desc(t,idx_t)*/* from t
  2  order by owner asc,object_type desc;

9471 rows selected.


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        122  consistent gets
          0  physical reads
          0  redo size
     435285  bytes sent via SQL*Net to client
       7325  bytes received via SQL*Net from client
        633  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       9471  rows processed
```

2.4 组合索引需要考虑顺序



```
SQL> drop table t purge;

Table dropped.

SQL> create table t as select * from dba_objects;

Table created.

SQL> create index idx_t on t(object_id,object_type);

Index created.

SQL> exec dbms_stats.gather_table_stats(user,'T',cascade=>true);

PL/SQL procedure successfully completed.

SQL> select * from t where object_type='TABLE' and object_id >100;

798 rows selected.

Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        188  consistent gets
          0  physical reads
          0  redo size
      79069  bytes sent via SQL*Net to client
        967  bytes received via SQL*Net from client
         55  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
        798  rows processed
```



```
SQL> drop index idx_t;

Index dropped.

SQL> create index idx_t on t(object_type,object_id);

Index created.

SQL> exec dbms_stats.gather_table_stats(user,'T',cascade=>true);

PL/SQL procedure successfully completed.

SQL> select * from t where object_type='TABLE' and object_id >100;

798 rows selected.


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        175  consistent gets
          0  physical reads
          0  redo size
      37495  bytes sent via SQL*Net to client
        967  bytes received via SQL*Net from client
         55  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
        798  rows processed
```

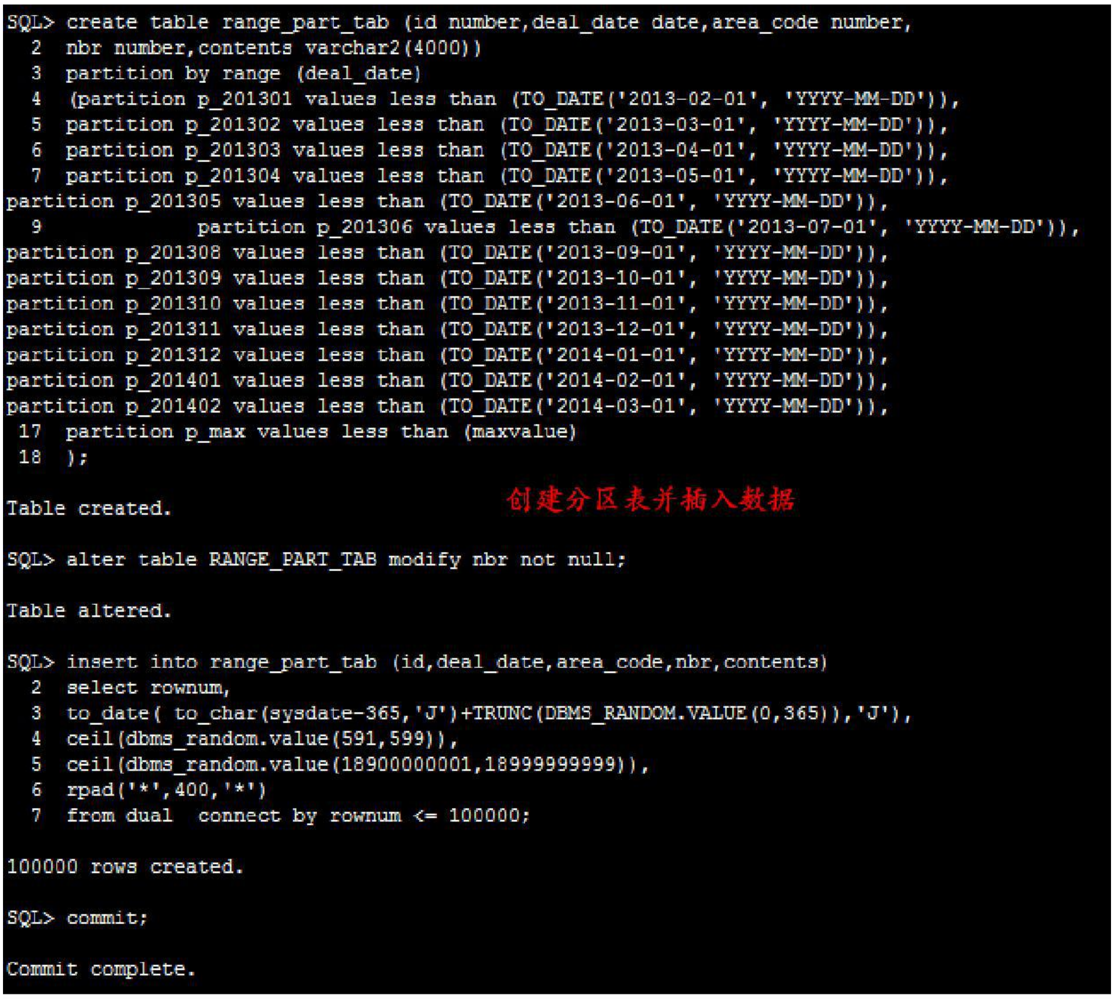
顺序不同产生的一致性读不同

3.解答第三题

◆ 分区表中的聚合语句有什么特别之处？

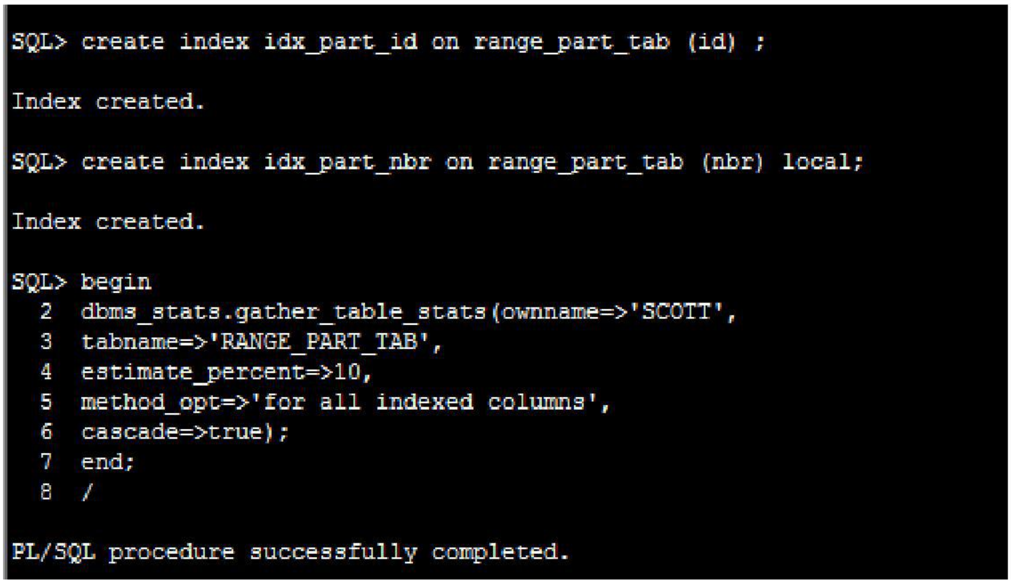
首先我们将在分区表中执行一些聚合语句来总结其特别之处：

3.1 环境准备



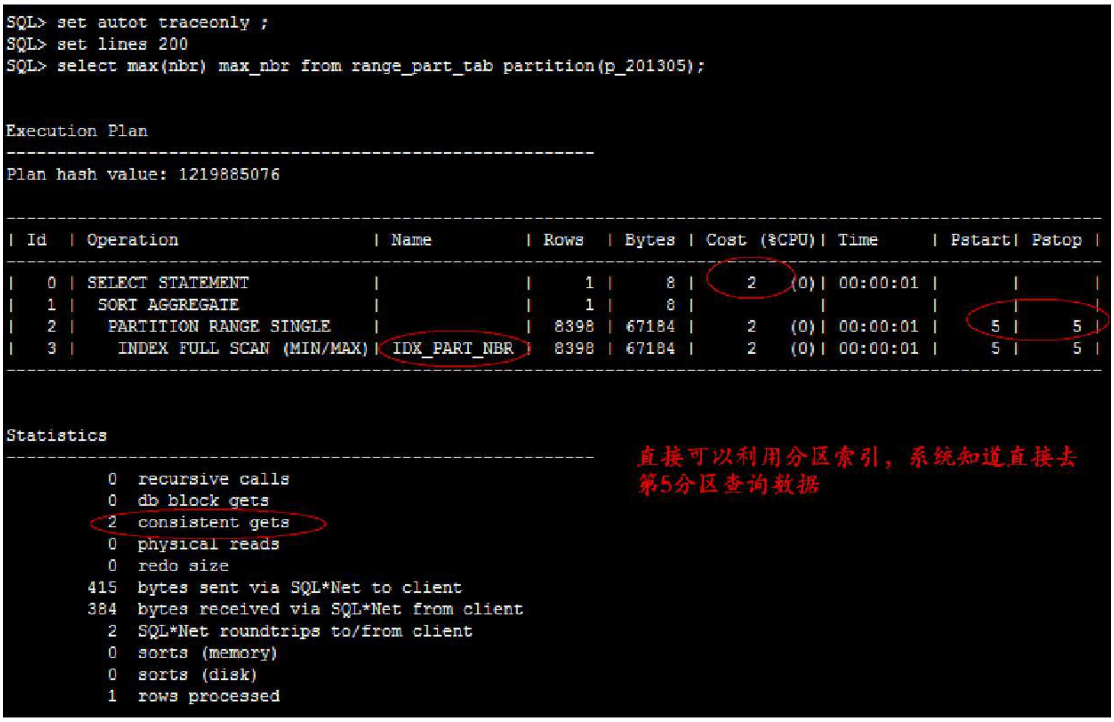
```
SQL> create table range_part_tab (id number,deal_date date,area_code number,
  2  nbr number,contents varchar2(4000))
  3  partition by range (deal_date)
  4   (partition p_201301 values less than (TO_DATE('2013-02-01', 'YYYY-MM-DD')),
  5  partition p_201302 values less than (TO_DATE('2013-03-01', 'YYYY-MM-DD')),
  6  partition p_201303 values less than (TO_DATE('2013-04-01', 'YYYY-MM-DD')),
  7  partition p_201304 values less than (TO_DATE('2013-05-01', 'YYYY-MM-DD')),
partition p_201305 values less than (TO_DATE('2013-06-01', 'YYYY-MM-DD')),
  9             partition p_201306 values less than (TO_DATE('2013-07-01', 'YYYY-MM-DD')),
partition p_201308 values less than (TO_DATE('2013-09-01', 'YYYY-MM-DD')),
partition p_201309 values less than (TO_DATE('2013-10-01', 'YYYY-MM-DD')),
partition p_201310 values less than (TO_DATE('2013-11-01', 'YYYY-MM-DD')),
partition p_201311 values less than (TO_DATE('2013-12-01', 'YYYY-MM-DD')),
partition p_201312 values less than (TO_DATE('2014-01-01', 'YYYY-MM-DD')),
partition p_201401 values less than (TO_DATE('2014-02-01', 'YYYY-MM-DD')),
partition p_201402 values less than (TO_DATE('2014-03-01', 'YYYY-MM-DD')),
 17  partition p_max values less than (maxvalue)
 18  );

Table created.

SQL> alter table RANGE_PART_TAB modify nbr not null;

Table altered.

SQL> insert into range_part_tab (id,deal_date,area_code,nbr,contents)
  2  select rownum,
  3  to_date( to_char(sysdate-365,'J')+TRUNC(DBMS_RANDOM.VALUE(0,365)),'J'),
  4  ceil(dbms_random.value(591,599)),
  5  ceil(dbms_random.value(18900000001,18999999999)),
  6  rpad('*',400,'*')
  7  from dual  connect by rownum <= 100000;

100000 rows created.

SQL> commit;

Commit complete.
```

创建分区表并插入数据



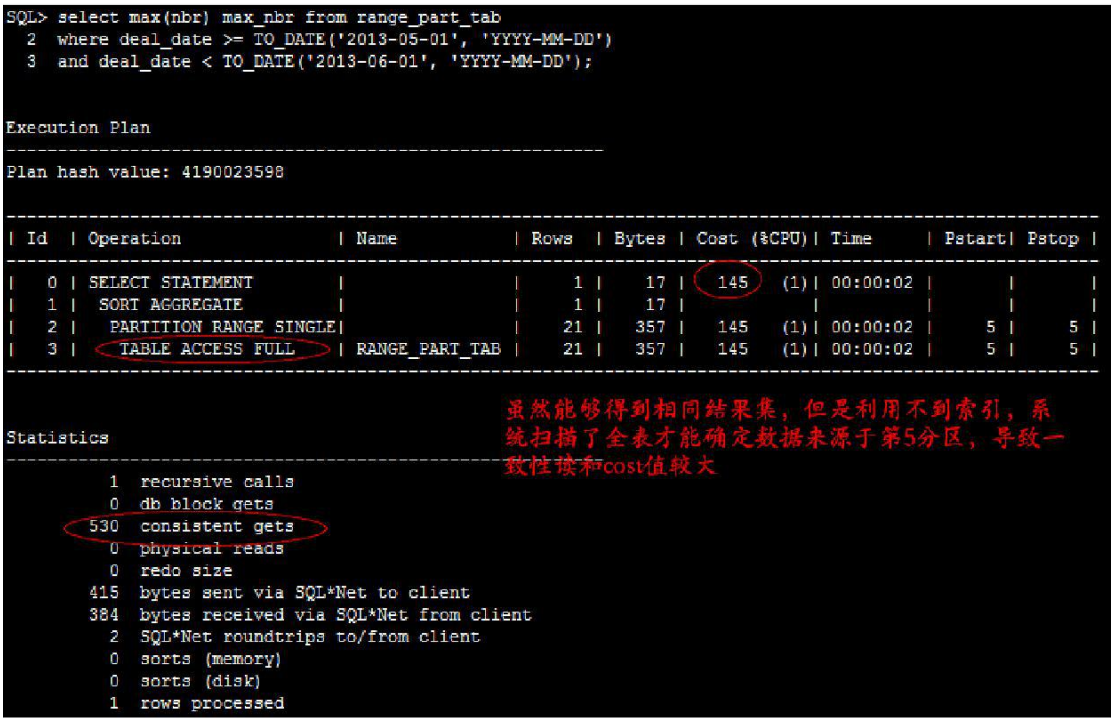
```
SQL> create index idx_part_id on range_part_tab (id) ;

Index created.

SQL> create index idx_part_nbr on range_part_tab (nbr) local;

Index created.

SQL> begin
  2  dbms_stats.gather_table_stats(ownname=>'SCOTT',
  3  tabname=>'RANGE_PART_TAB',
  4  estimate_percent=>10,
  5  method_opt=>'for all indexed columns',
  6  cascade=>true);
  7  end;
  8  /

PL/SQL procedure successfully completed.
```

3.2 分区表下的 max 聚合函数



```
SQL> set autot traceonly ;
SQL> set lines 200
SQL> select max(nbr) max_nbr from range_part_tab partition(p_201305);


Execution Plan
----------------------------------------------------------
Plan hash value: 1219885076

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                   | Name         | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |              |     1 |     8 |     2   (0)| 00:00:01 |       |       |
|   1 |  SORT AGGREGATE             |              |     1 |     8 |            |          |       |       |
|   2 |   PARTITION RANGE SINGLE    |              |  8398 | 67184 |     2   (0)| 00:00:01 |     5 |     5 |
|   3 |    INDEX FULL SCAN (MIN/MAX)| IDX_PART_NBR |  8398 | 67184 |     2   (0)| 00:00:01 |     5 |     5 |
-----------------------------------------------------------------------------------------------------------


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          2  consistent gets
          0  physical reads
          0  redo size
        415  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
```

直接可以利用分区索引，系统知道直接去第5分区查询数据

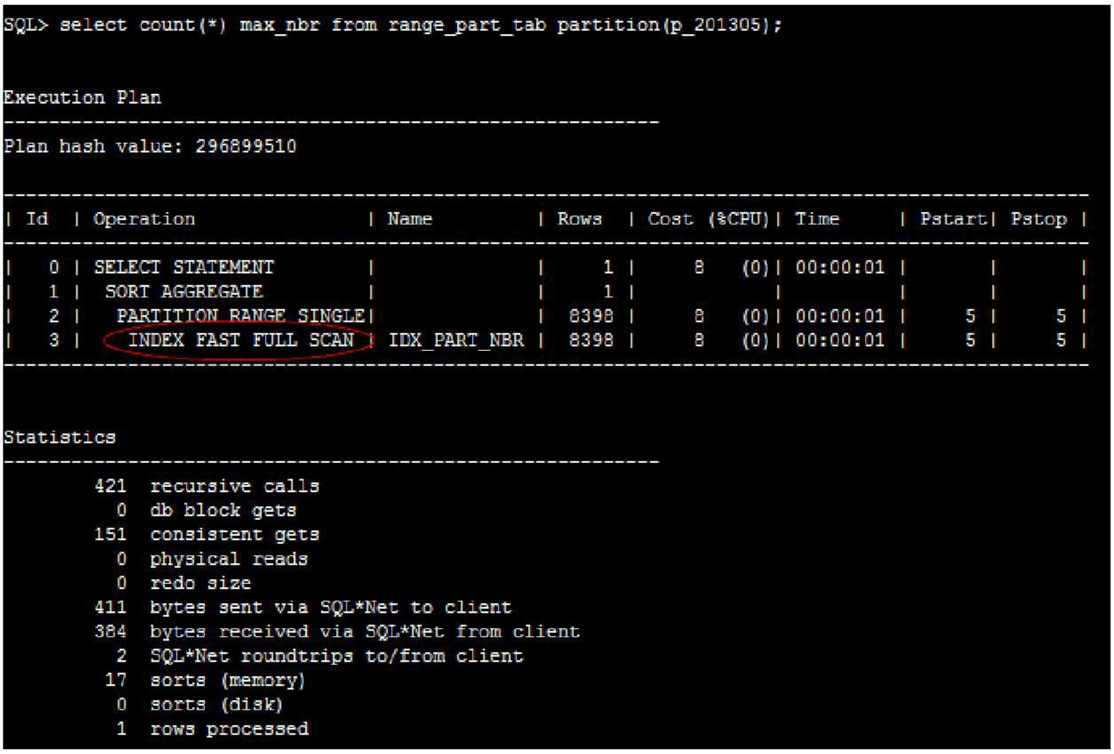


```
SQL> select max(nbr) max_nbr from range_part_tab
  2  where deal_date >= TO_DATE('2013-05-01', 'YYYY-MM-DD')
  3  and deal_date < TO_DATE('2013-06-01', 'YYYY-MM-DD');


Execution Plan
----------------------------------------------------------
Plan hash value: 4190023598

-------------------------------------------------------------------------------------------------------
| Id  | Operation              | Name           | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                |     1 |    17 |   145   (1)| 00:00:02 |       |       |
|   1 |  SORT AGGREGATE        |                |     1 |    17 |            |          |       |       |
|   2 |   PARTITION RANGE SINGLE|               |    21 |   357 |   145   (1)| 00:00:02 |     5 |     5 |
|   3 |    TABLE ACCESS FULL   | RANGE_PART_TAB |    21 |   357 |   145   (1)| 00:00:02 |     5 |     5 |
-------------------------------------------------------------------------------------------------------


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        530  consistent gets
          0  physical reads
          0  redo size
        415  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
```

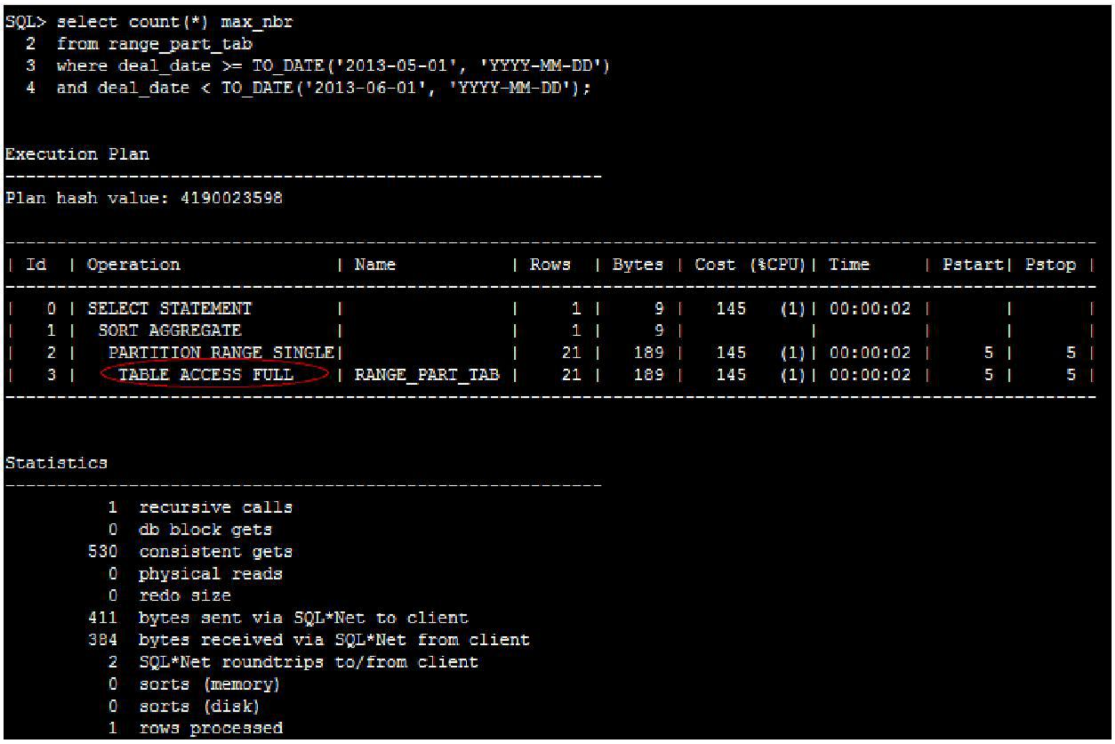
虽然能够得到相同结果集，但是利用不到索引，系统扫描了全表才能确定数据来源于第5分区，导致一致性读和cost值较大

3.3 分区表下的 count 聚合函数

Count/sum/distinct 等与 max 相同，这里将不再说明，只演示一遍。

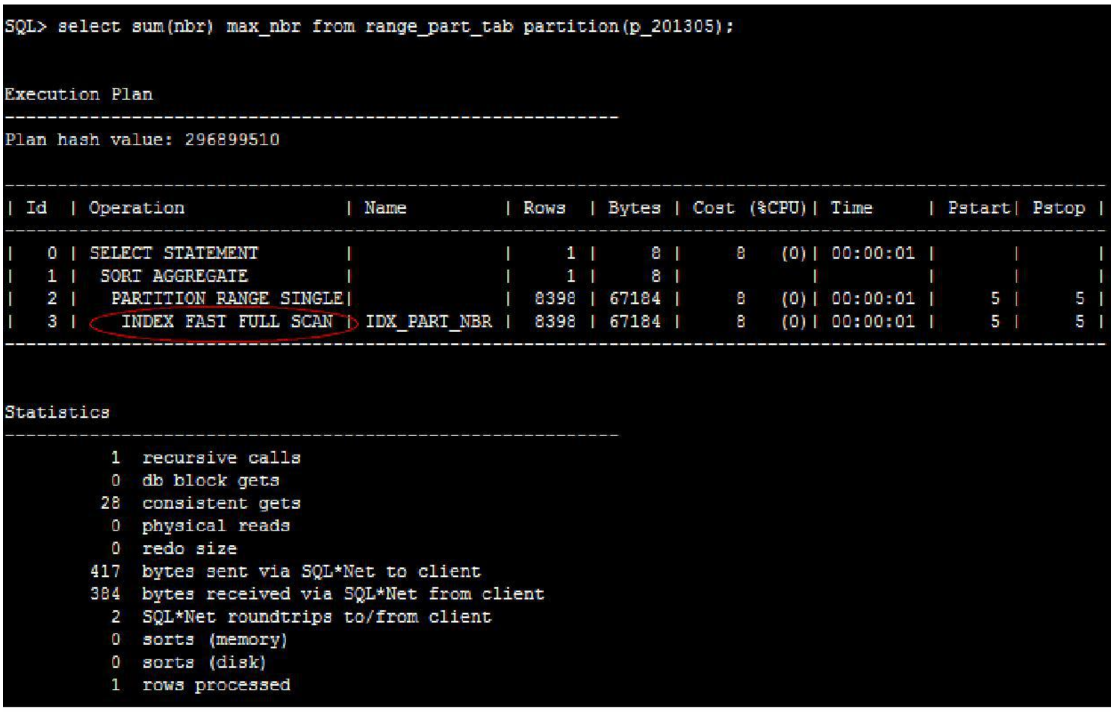


```
SQL> select count(*) max_nbr from range_part_tab partition(p_201305);


Execution Plan
----------------------------------------------------------
Plan hash value: 296899510

------------------------------------------------------------------------------------------------
| Id  | Operation                 | Name         | Rows  | Cost (%CPU)| Time     | Pstart| Pstop |
------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT          |              |     1 |     8   (0)| 00:00:01 |       |       |
|   1 |  SORT AGGREGATE           |              |     1 |            |          |       |       |
|   2 |   PARTITION RANGE SINGLE|              |  8398 |     8   (0)| 00:00:01 |     5 |     5 |
|   3 |    INDEX FAST FULL SCAN | IDX_PART_NBR |  8398 |     8   (0)| 00:00:01 |     5 |     5 |
------------------------------------------------------------------------------------------------


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

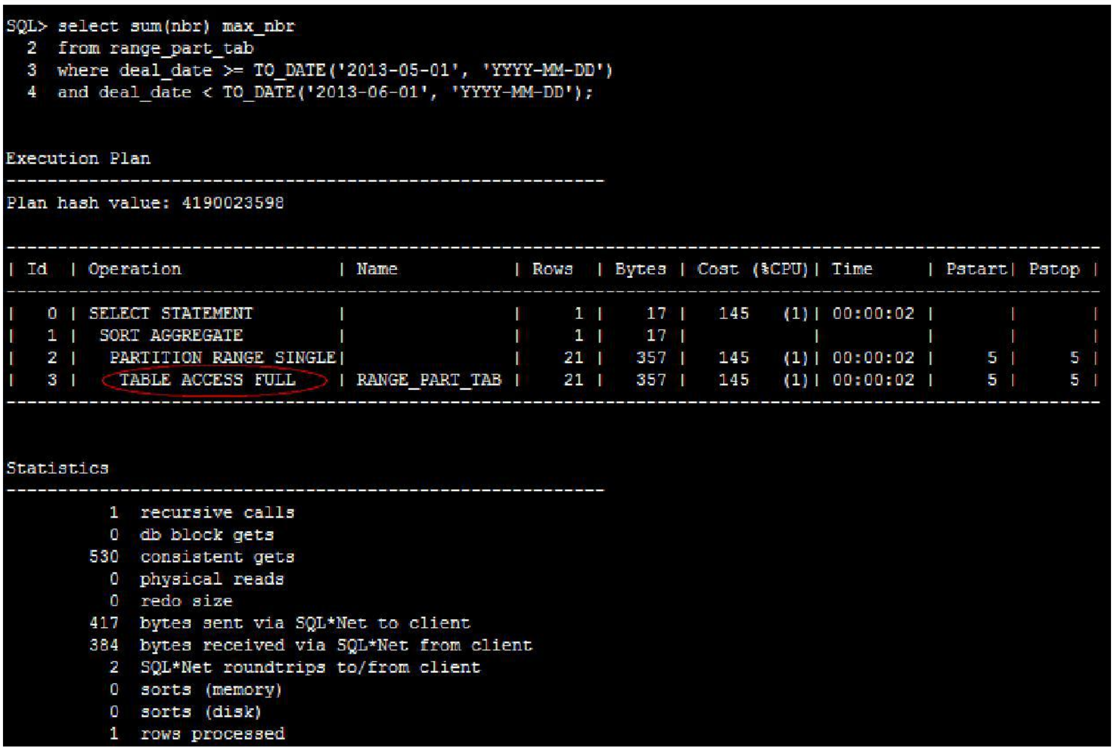


```
SQL> select count(*) max_nbr
  2  from range_part_tab
  3  where deal_date >= TO_DATE('2013-05-01', 'YYYY-MM-DD')
  4  and deal_date < TO_DATE('2013-06-01', 'YYYY-MM-DD');


Execution Plan
----------------------------------------------------------
Plan hash value: 4190023598

--------------------------------------------------------------------------------------------------------
| Id  | Operation                 | Name           | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
--------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT          |                |     1 |     9 |   145   (1)| 00:00:02 |       |       |
|   1 |  SORT AGGREGATE           |                |     1 |     9 |            |          |       |       |
|   2 |   PARTITION RANGE SINGLE|                |    21 |   189 |   145   (1)| 00:00:02 |     5 |     5 |
|   3 |    TABLE ACCESS FULL      | RANGE_PART_TAB |    21 |   189 |   145   (1)| 00:00:02 |     5 |     5 |
--------------------------------------------------------------------------------------------------------


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        530  consistent gets
          0  physical reads
          0  redo size
        411  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
```

3.3 分区表下的 sum 聚合函数

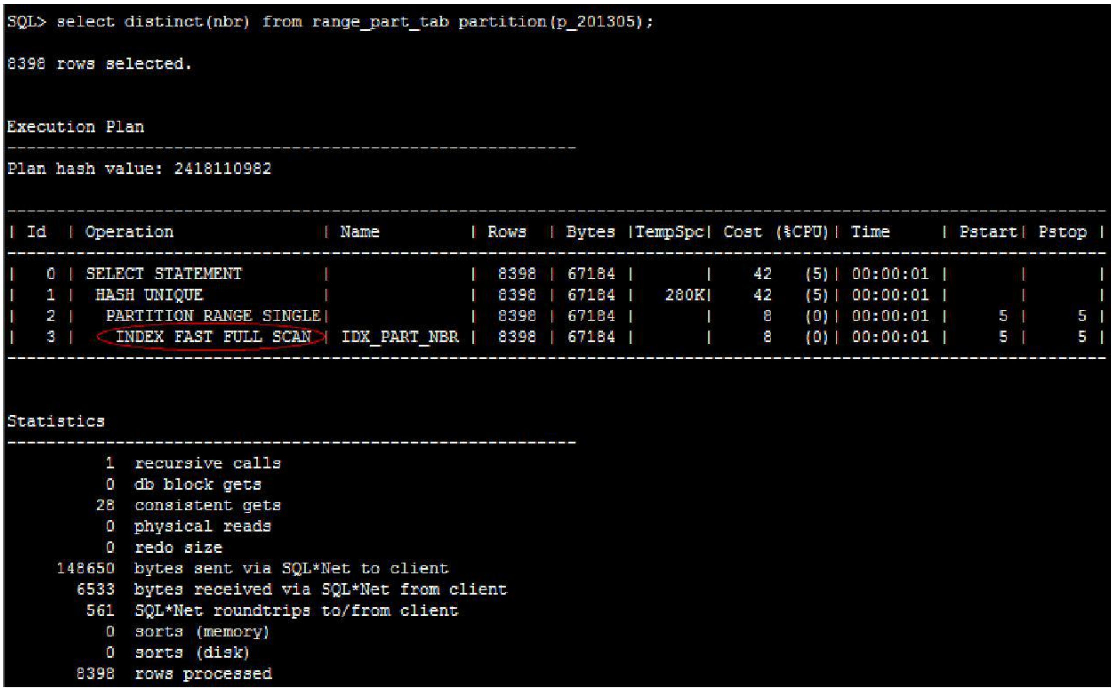


```
SQL> select sum(nbr) max_nbr from range_part_tab partition(p_201305);


Execution Plan
----------------------------------------------------------
Plan hash value: 296899510

--------------------------------------------------------------------------------------------------------
| Id  | Operation                 | Name         | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
--------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT          |              |     1 |     8 |     8   (0)| 00:00:01 |       |       |
|   1 |  SORT AGGREGATE           |              |     1 |     8 |            |          |       |       |
|   2 |   PARTITION RANGE SINGLE|              |  8398 | 67184 |     8   (0)| 00:00:01 |     5 |     5 |
|   3 |    INDEX FAST FULL SCAN | IDX_PART_NBR |  8398 | 67184 |     8   (0)| 00:00:01 |     5 |     5 |
--------------------------------------------------------------------------------------------------------


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
         28  consistent gets
          0  physical reads
          0  redo size
        417  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
```

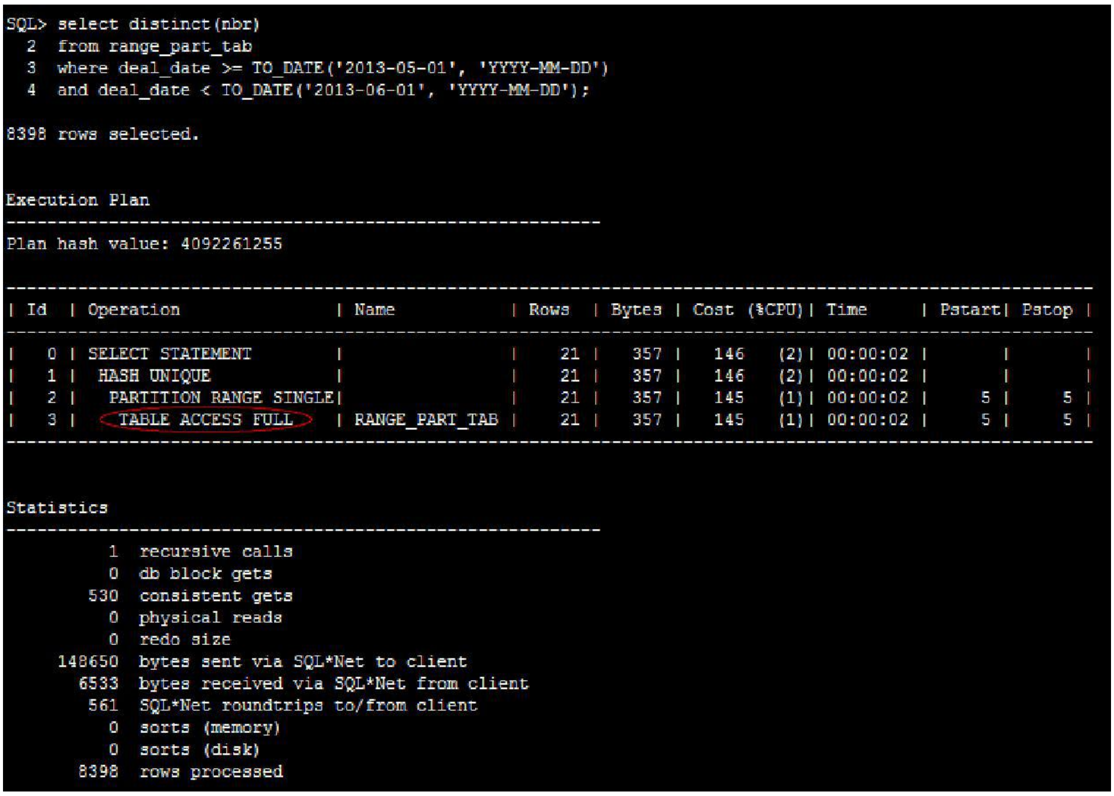


```
SQL> select sum(nbr) max_nbr
  2  from range_part_tab
  3  where deal_date >= TO_DATE('2013-05-01', 'YYYY-MM-DD')
  4  and deal_date < TO_DATE('2013-06-01', 'YYYY-MM-DD');


Execution Plan
----------------------------------------------------------
Plan hash value: 4190023598

--------------------------------------------------------------------------------------------------------
| Id  | Operation                 | Name           | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
--------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT          |                |     1 |    17 |   145   (1)| 00:00:02 |       |       |
|   1 |  SORT AGGREGATE           |                |     1 |    17 |            |          |       |       |
|   2 |   PARTITION RANGE SINGLE|                |    21 |   357 |   145   (1)| 00:00:02 |     5 |     5 |
|   3 |    TABLE ACCESS FULL      | RANGE_PART_TAB |    21 |   357 |   145   (1)| 00:00:02 |     5 |     5 |
--------------------------------------------------------------------------------------------------------


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        530  consistent gets
          0  physical reads
          0  redo size
        417  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
```

3.3 分区表下的 distinct 去重处理



```
SQL> select distinct(nbr) from range_part_tab partition(p_201305);

8398 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2418110982

-------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name         | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |              |  8398 | 67184 |       |    42   (5)| 00:00:01 |       |       |
|   1 |  HASH UNIQUE             |              |  8398 | 67184 |   280K|    42   (5)| 00:00:01 |       |       |
|   2 |   PARTITION RANGE SINGLE|              |  8398 | 67184 |       |     8   (0)| 00:00:01 |     5 |     5 |
|   3 |    INDEX FAST FULL SCAN | IDX_PART_NBR |  8398 | 67184 |       |     8   (0)| 00:00:01 |     5 |     5 |
-------------------------------------------------------------------------------------------------------------


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
         28  consistent gets
          0  physical reads
          0  redo size
     148650  bytes sent via SQL*Net to client
       6533  bytes received via SQL*Net from client
        561  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       8398  rows processed
```



```
SQL> select distinct(nbr)
  2  from range_part_tab
  3  where deal_date >= TO_DATE('2013-05-01', 'YYYY-MM-DD')
  4  and deal_date < TO_DATE('2013-06-01', 'YYYY-MM-DD');

8398 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 4092261255

------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name           | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                |    21 |   357 |   146   (2)| 00:00:02 |       |       |
|   1 |  HASH UNIQUE             |                |    21 |   357 |   146   (2)| 00:00:02 |       |       |
|   2 |   PARTITION RANGE SINGLE|                |    21 |   357 |   145   (1)| 00:00:02 |     5 |     5 |
|   3 |    TABLE ACCESS FULL     | RANGE_PART_TAB |    21 |   357 |   145   (1)| 00:00:02 |     5 |     5 |
------------------------------------------------------------------------------------------------------


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        530  consistent gets
          0  physical reads
          0  redo size
     148650  bytes sent via SQL*Net to client
       6533  bytes received via SQL*Net from client
        561  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       8398  rows processed
```