收获，不止 SQL 优化

第 九 章

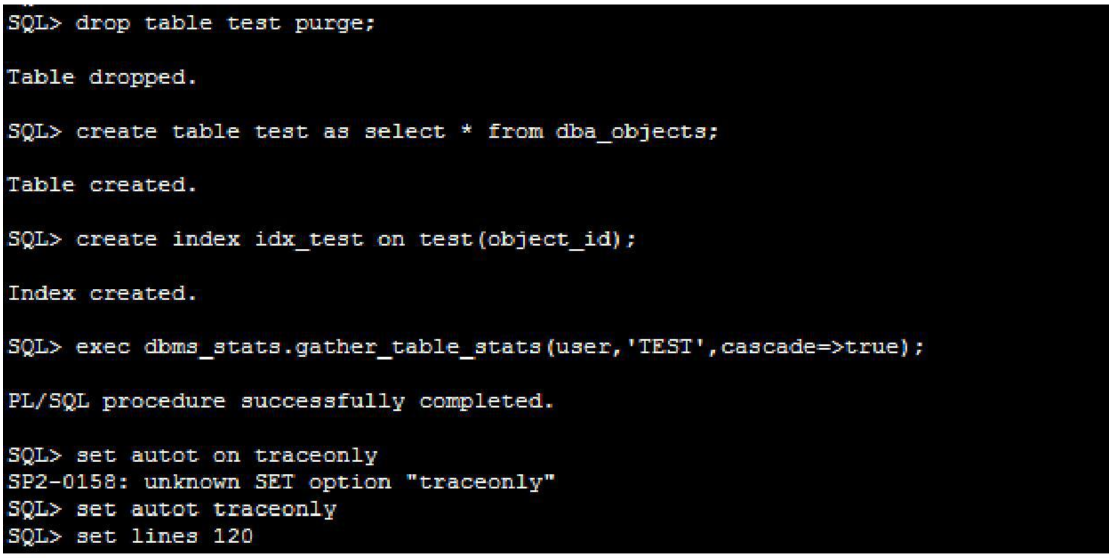
且慢，弄清索引之阻碍 让 SQL 飞

E-Mail:45240040@qq.com

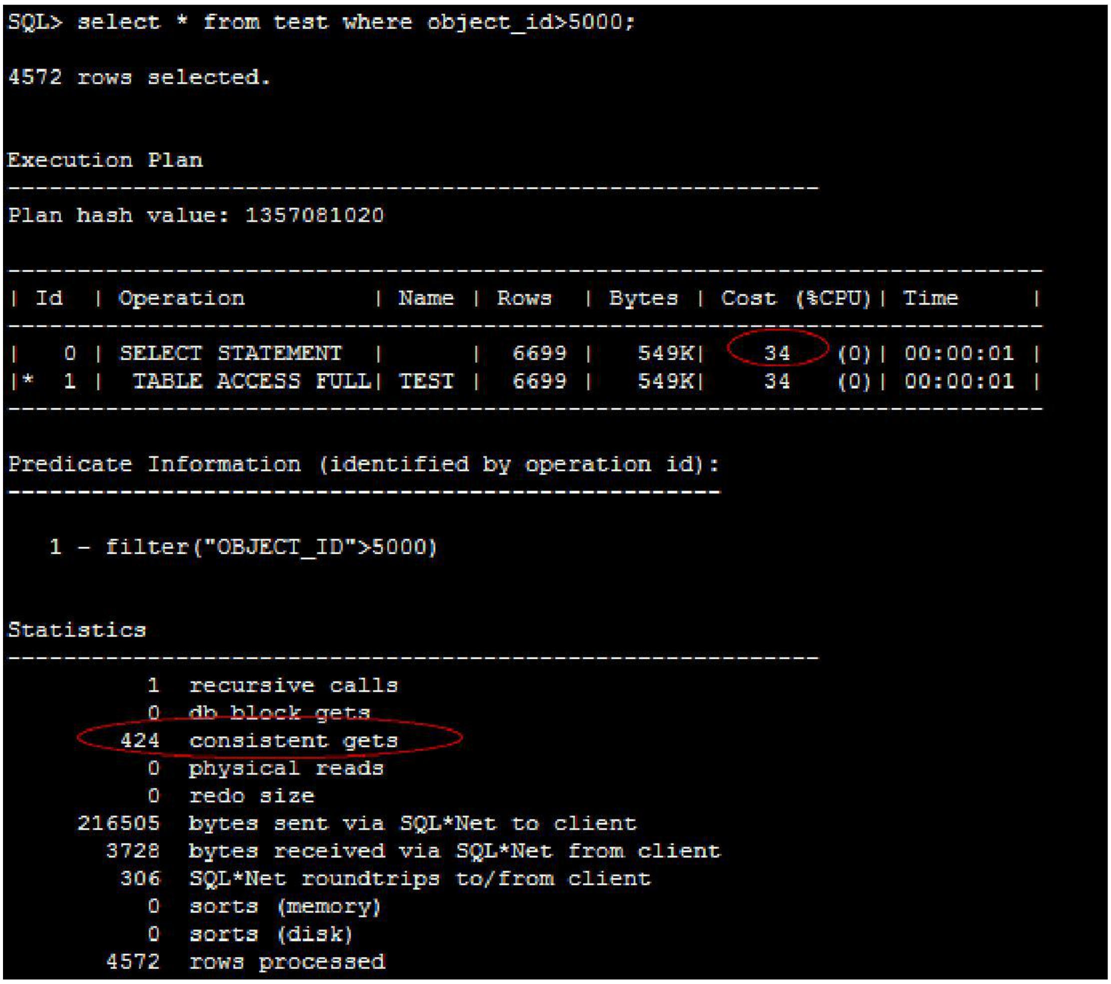
目录

1.举例说明一下哪些场景会用不上索引（索引没失效）

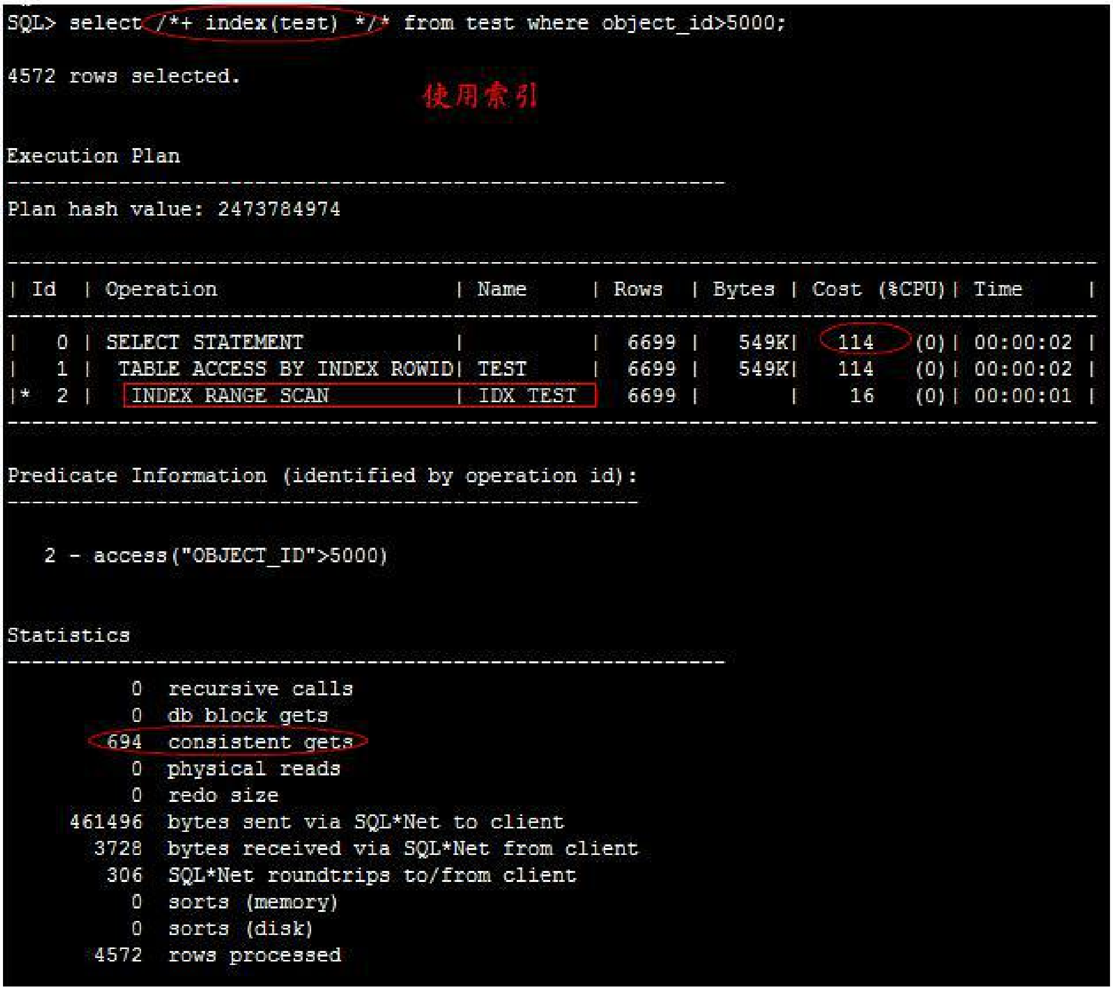
1.1 用索引代价反而更高



```
SQL> drop table test purge;

Table dropped.

SQL> create table test as select * from dba_objects;

Table created.

SQL> create index idx_test on test(object_id);

Index created.

SQL> exec dbms_stats.gather_table_stats(user,'TEST',cascade=>true);

PL/SQL procedure successfully completed.

SQL> set autot on traceonly
SP2-0158: unknown SET option "traceonly"
SQL> set autot traceonly
SQL> set lines 120
```

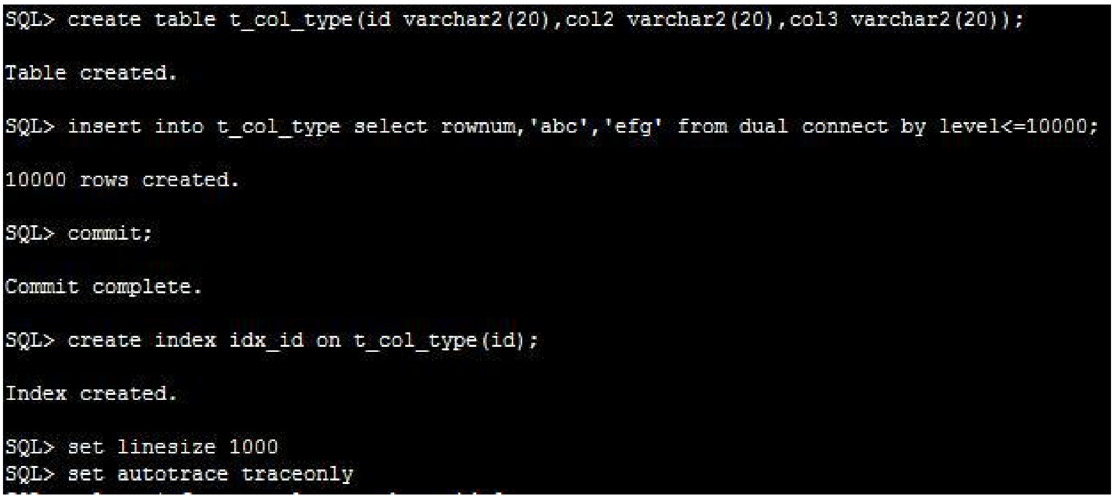


```
SQL> select * from test where object_id>5000;

4572 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 1357081020

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |  6699 |   549K|    34   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| TEST |  6699 |   549K|    34   (0)| 00:00:01 |
--------------------------------------------------------------------------

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

   1 - filter("OBJECT_ID">5000)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        424  consistent gets
          0  physical reads
          0  redo size
     216505  bytes sent via SQL*Net to client
       3728  bytes received via SQL*Net from client
        306  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       4572  rows processed
```

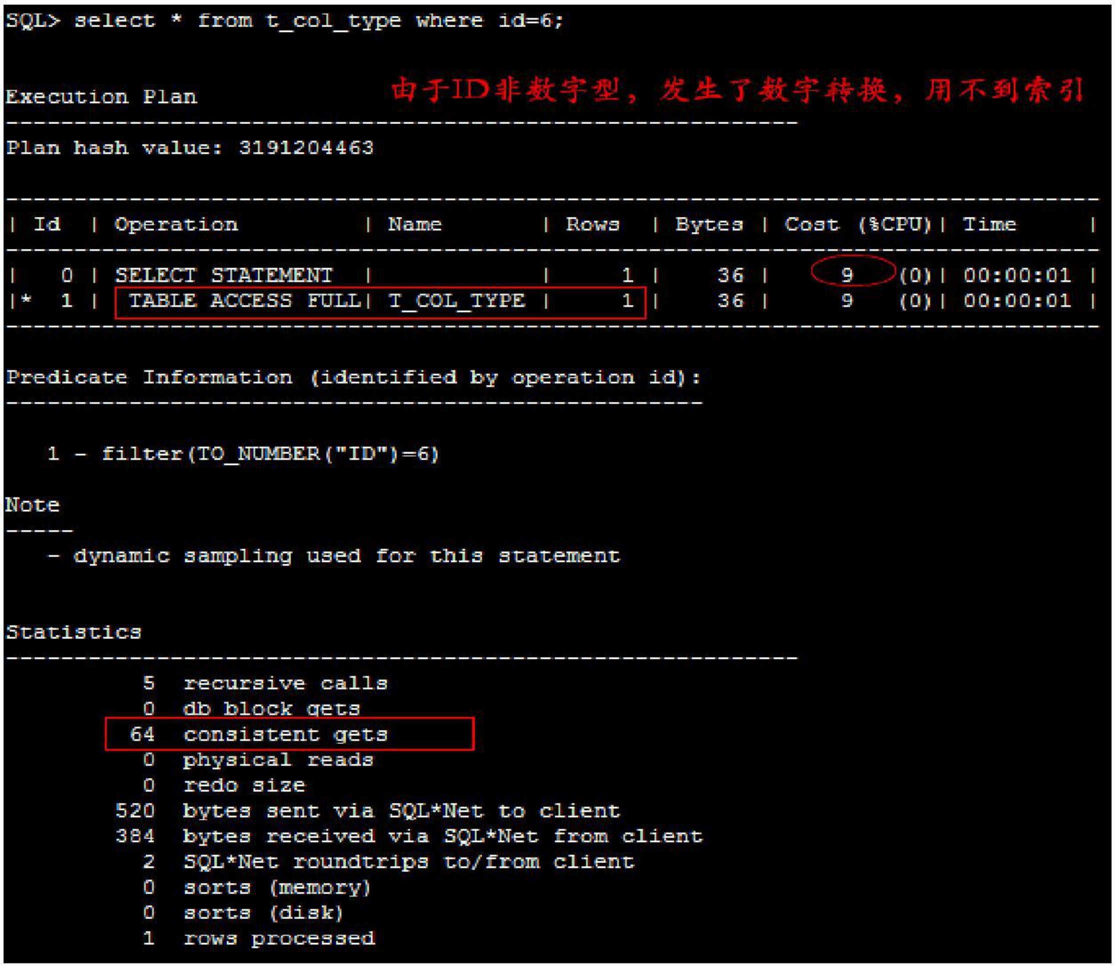


```
SQL> select /*+ index(test) */* from test where object_id>5000;

4572 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2473784974

-------------------------------------------------------------------------------------------
| Id  | Operation                   | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |          |  6699 |   549K|   114   (0)| 00:00:02 |
|   1 |  TABLE ACCESS BY INDEX ROWID| TEST     |  6699 |   549K|   114   (0)| 00:00:02 |
|*  2 |   INDEX RANGE SCAN          | IDX_TEST |  6699 |       |    16   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

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

   2 - access("OBJECT_ID">5000)


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
        694  consistent gets
          0  physical reads
          0  redo size
     461496  bytes sent via SQL*Net to client
       3728  bytes received via SQL*Net from client
        306  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       4572  rows processed
```

1.2 发生了索引列的类型转换

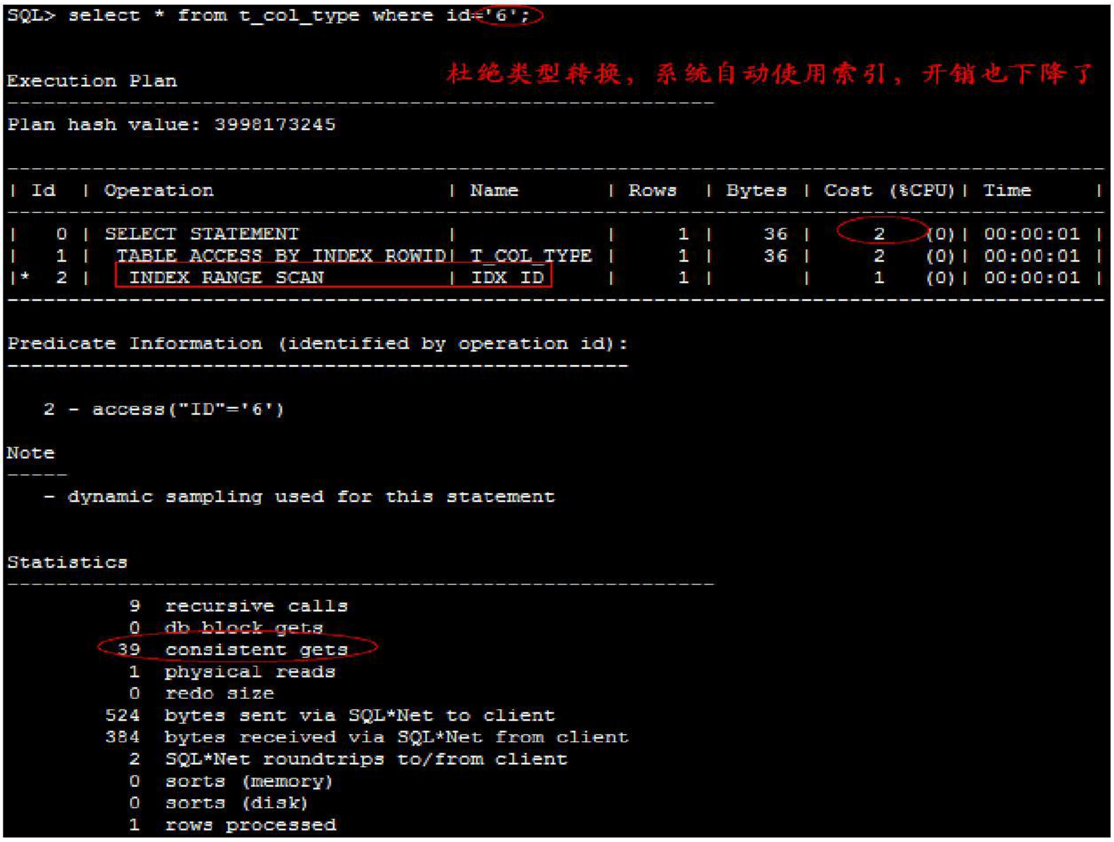


```
SQL> create table t_col_type(id varchar2(20),col2 varchar2(20),col3 varchar2(20));

Table created.

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

10000 rows created.

SQL> commit;

Commit complete.

SQL> create index idx_id on t_col_type(id);

Index created.

SQL> set linesize 1000
SQL> set autotrace traceonly
```



由于ID非数字型，发生了数字转换，用不到索引

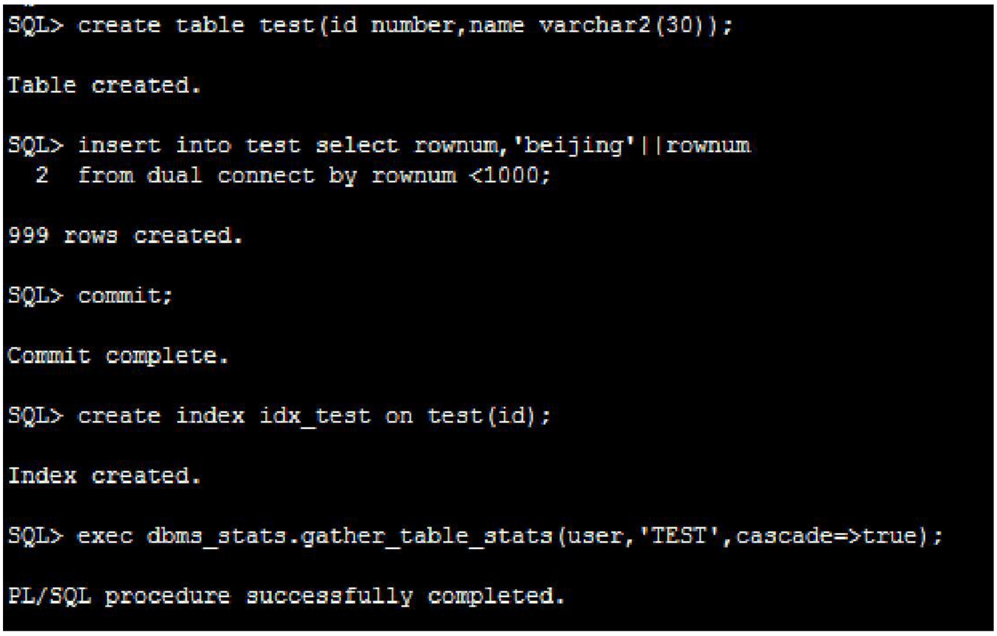
```
SQL> select * from t_col_type where id=6;


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

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

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

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

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


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



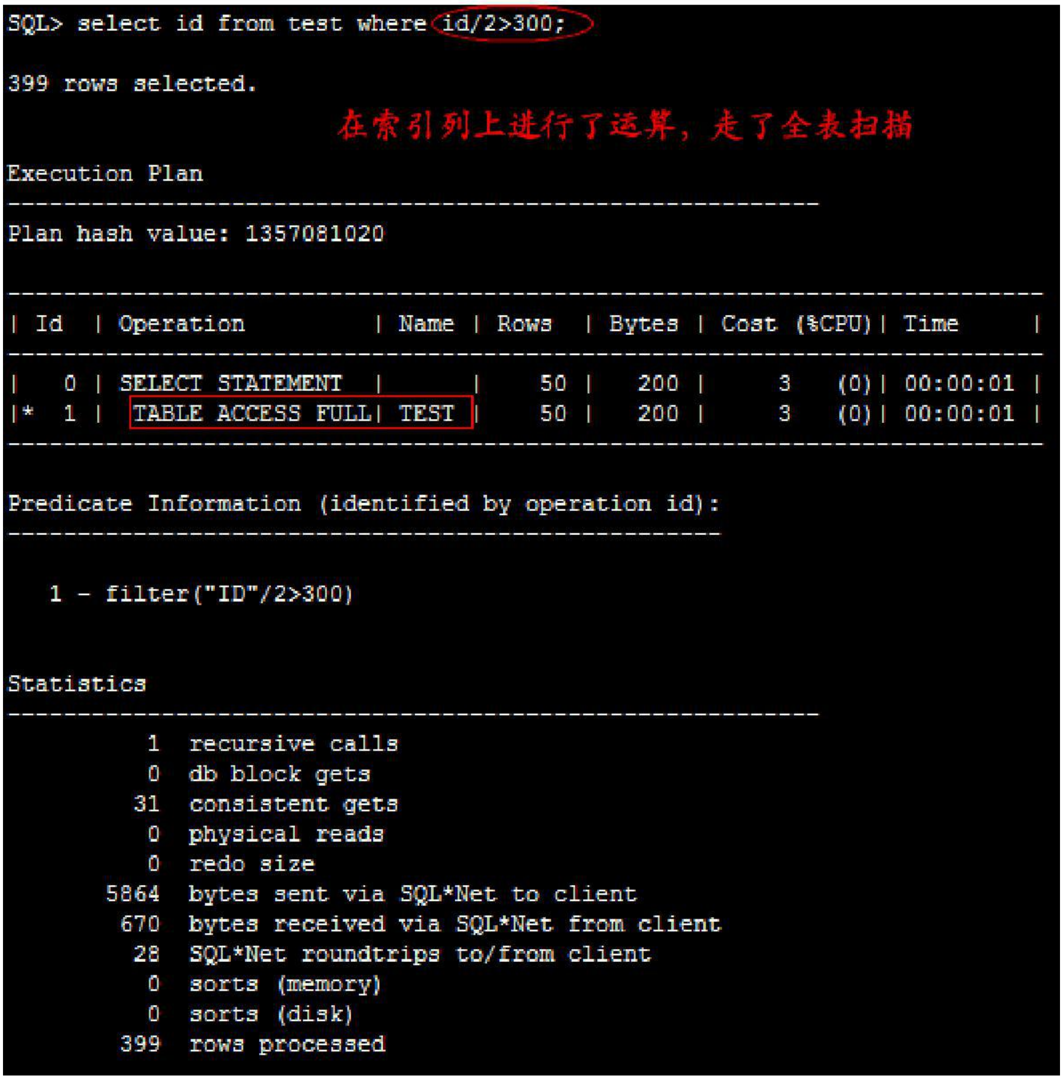
杜绝类型转换，系统自动使用索引，开销也下降了

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


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

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

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

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

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


Statistics
----------------------------------------------------------
          9  recursive calls
          0  db block gets
         39  consistent gets
          1  physical reads
          0  redo size
        524  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
```

1.3 对索引列进行了各种运算



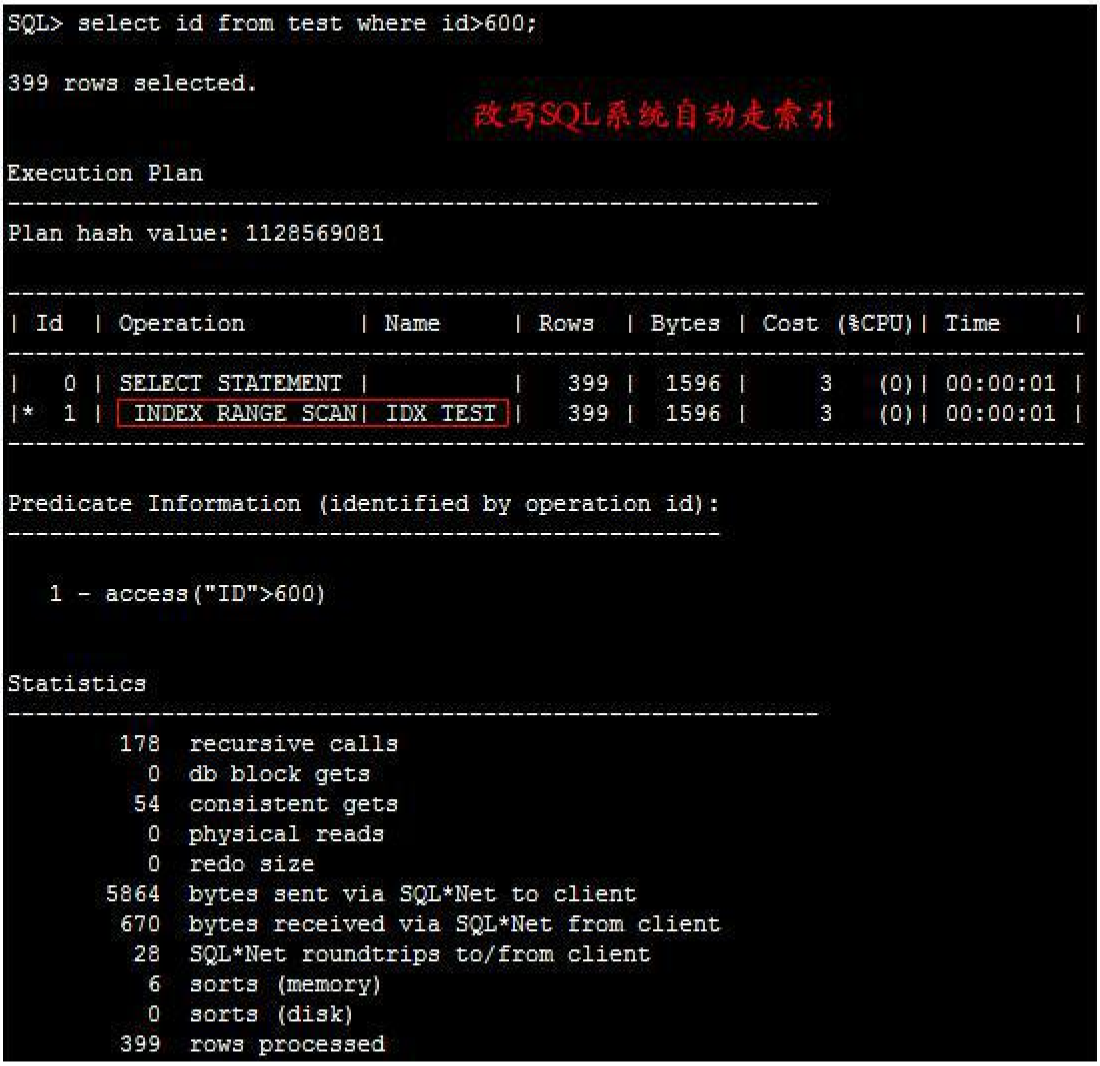
```
SQL> create table test(id number,name varchar2(30));

Table created.

SQL> insert into test select rownum,'beijing'||rownum
  2  from dual connect by rownum <1000;

999 rows created.

SQL> commit;

Commit complete.

SQL> create index idx_test on test(id);

Index created.

SQL> exec dbms_stats.gather_table_stats(user,'TEST',cascade=>true);

PL/SQL procedure successfully completed.
```



```
SQL> select id from test where id/2>300;

399 rows selected.
```

在索引列上进行了运算，走了全表扫描

```
Execution Plan
----------------------------------------------------------
Plan hash value: 1357081020

--------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |    50 |   200 |     3   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| TEST |    50 |   200 |     3   (0)| 00:00:01 |
--------------------------------------------------------------------------

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

   1 - filter("ID"/2>300)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
         31  consistent gets
          0  physical reads
          0  redo size
       5864  bytes sent via SQL*Net to client
        670  bytes received via SQL*Net from client
         28  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
        399  rows processed
```

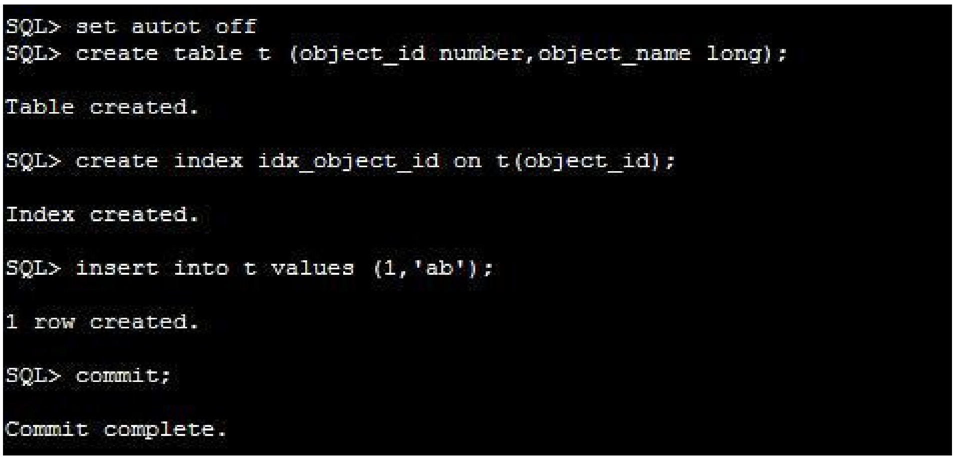


改写SQL系统自动走索引

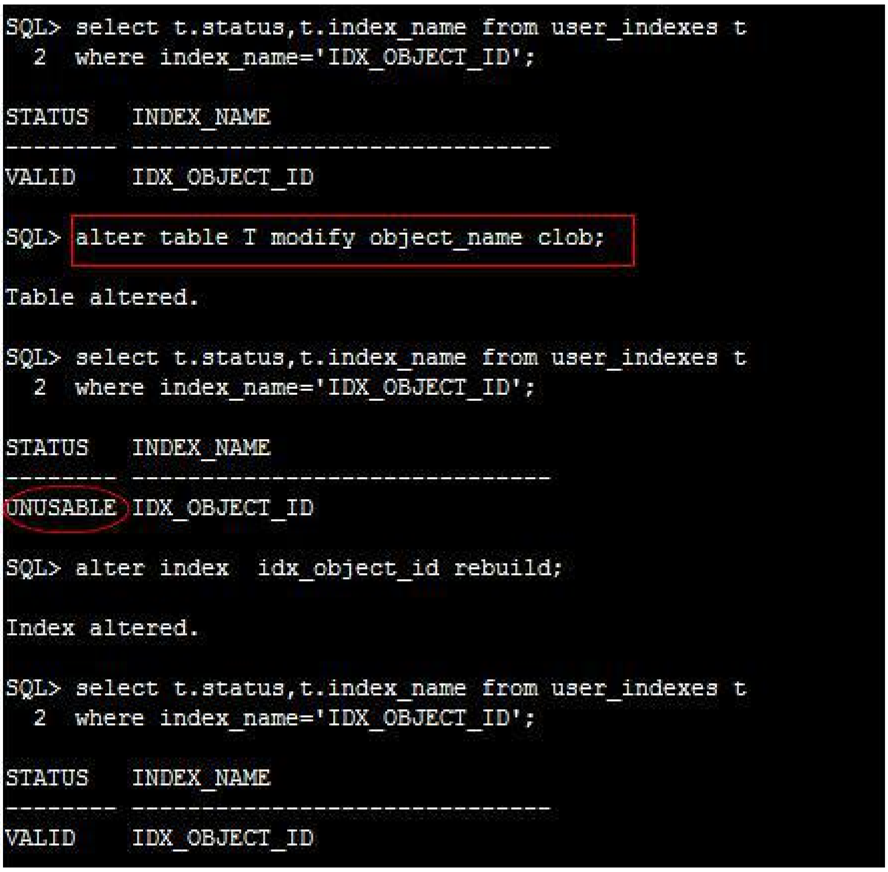
```
SQL> select id from test where id>600;

399 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 1128569081

-----------------------------------------------------------------------------
| Id  | Operation        | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT |          |   399 |  1596 |     3   (0)| 00:00:01 |
|*  1 |  INDEX RANGE SCAN| IDX_TEST |   399 |  1596 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------

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

   1 - access("ID">600)


Statistics
----------------------------------------------------------
        178  recursive calls
          0  db block gets
         54  consistent gets
          0  physical reads
          0  redo size
       5864  bytes sent via SQL*Net to client
        670  bytes received via SQL*Net from client
         28  SQL*Net roundtrips to/from client
          6  sorts (memory)
          0  sorts (disk)
        399  rows processed
```

2.举例说明一下哪些场景会导致索引失效或者丢失了

2.1 long 列调整会导致索引失效

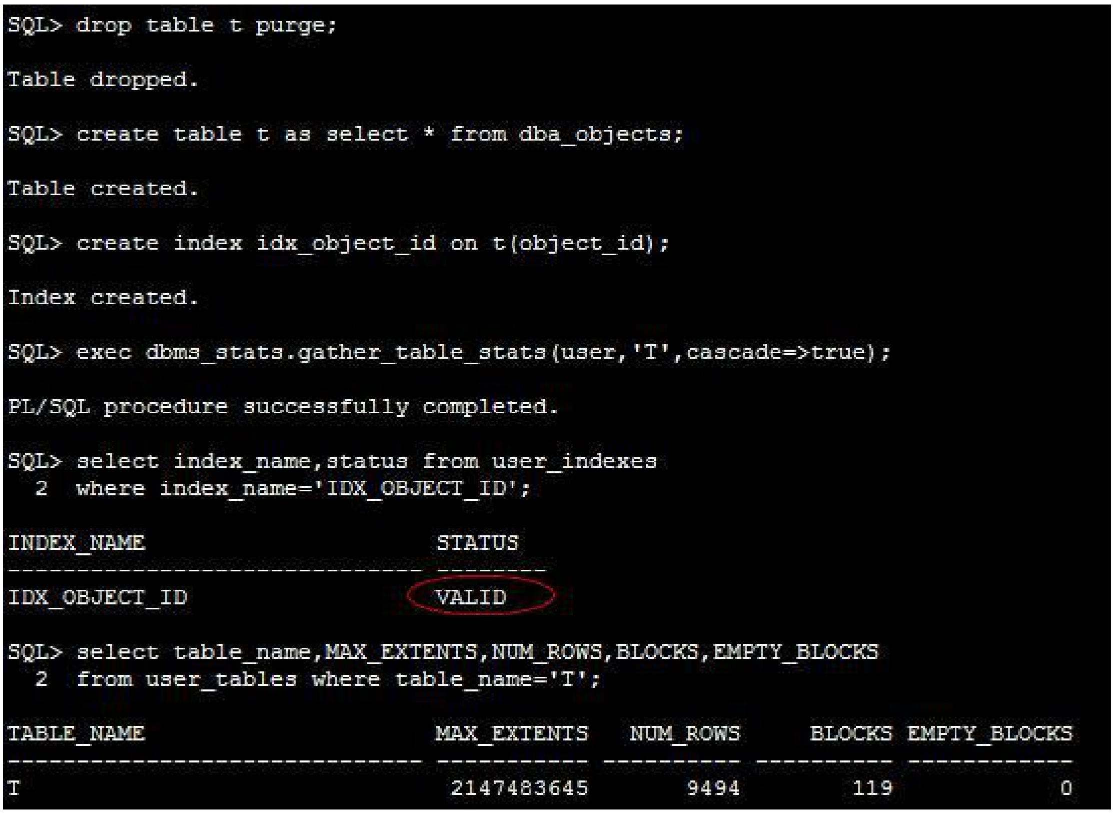


```
SQL> set autot off
SQL> create table t (object_id number,object_name long);

Table created.

SQL> create index idx_object_id on t(object_id);

Index created.

SQL> insert into t values (1,'ab');

1 row created.

SQL> commit;

Commit complete.
```

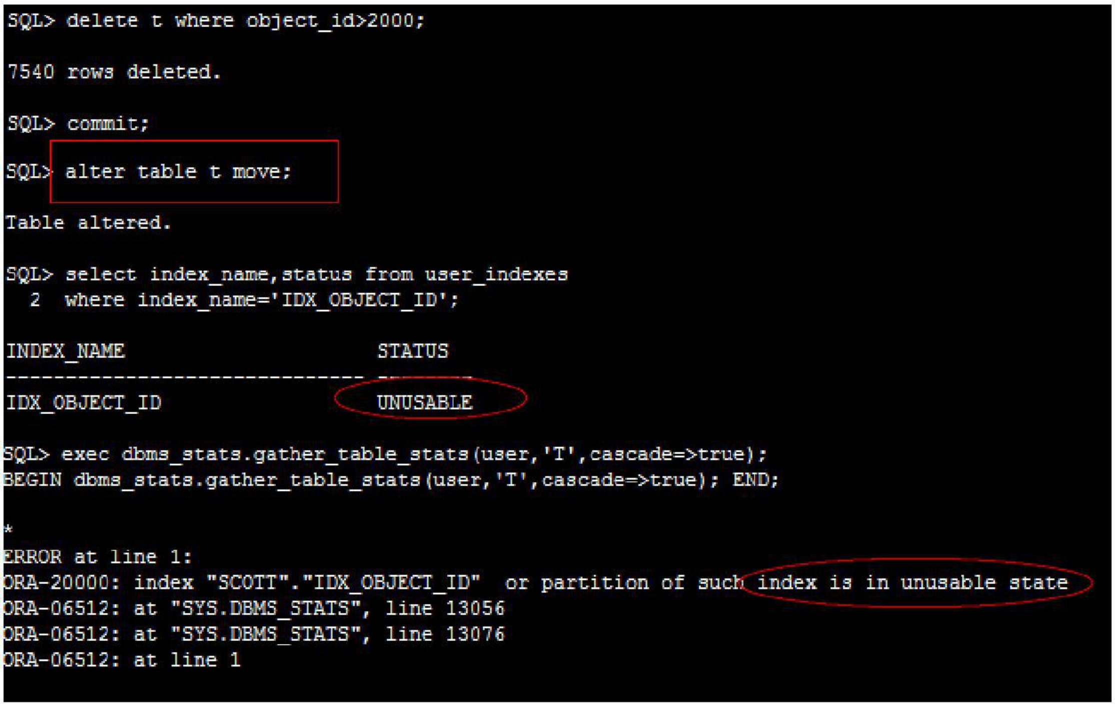


```
SQL> select t.status,t.index_name from user_indexes t
  2  where index_name='IDX_OBJECT_ID';

STATUS   INDEX_NAME
-------- ------------------------------
VALID    IDX_OBJECT_ID

SQL> alter table T modify object_name clob;

Table altered.

SQL> select t.status,t.index_name from user_indexes t
  2  where index_name='IDX_OBJECT_ID';

STATUS   INDEX_NAME
-------- ------------------------------
UNUSABLE IDX_OBJECT_ID

SQL> alter index  idx_object_id rebuild;

Index altered.

SQL> select t.status,t.index_name from user_indexes t
  2  where index_name='IDX_OBJECT_ID';

STATUS   INDEX_NAME
-------- ------------------------------
VALID    IDX_OBJECT_ID
```

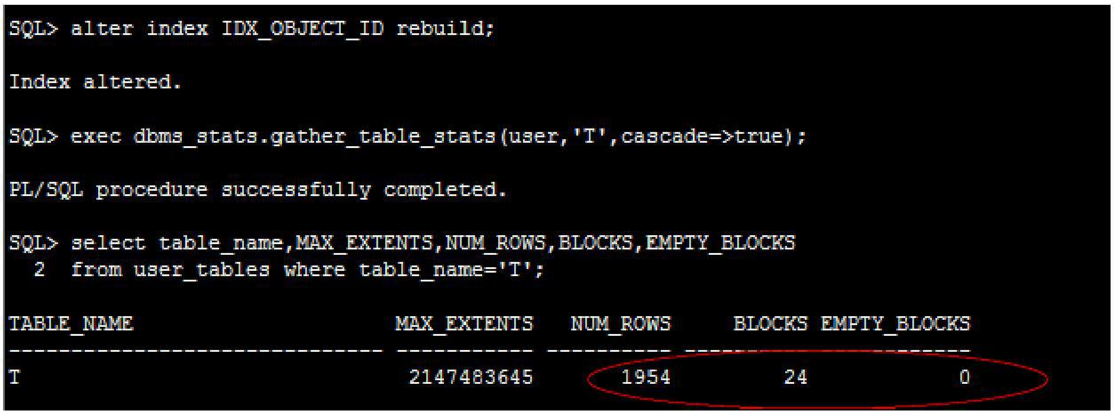
2.2 move 操作致索引失效



```
SQL> drop table t purge;

Table dropped.

SQL> create table t as select * from dba_objects;

Table created.

SQL> create index idx_object_id on t(object_id);

Index created.

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

PL/SQL procedure successfully completed.

SQL> select index_name,status from user_indexes
  2  where index_name='IDX_OBJECT_ID';

INDEX_NAME                     STATUS
------------------------------ --------
IDX_OBJECT_ID                  VALID

SQL> select table_name,MAX_EXTENTS,NUM_ROWS,BLOCKS,EMPTY_BLOCKS
  2  from user_tables where table_name='T';

TABLE_NAME                     MAX_EXTENTS   NUM_ROWS     BLOCKS EMPTY_BLOCKS
------------------------------ ----------- ---------- ---------- ------------
T                               2147483645       9494        119            0
```

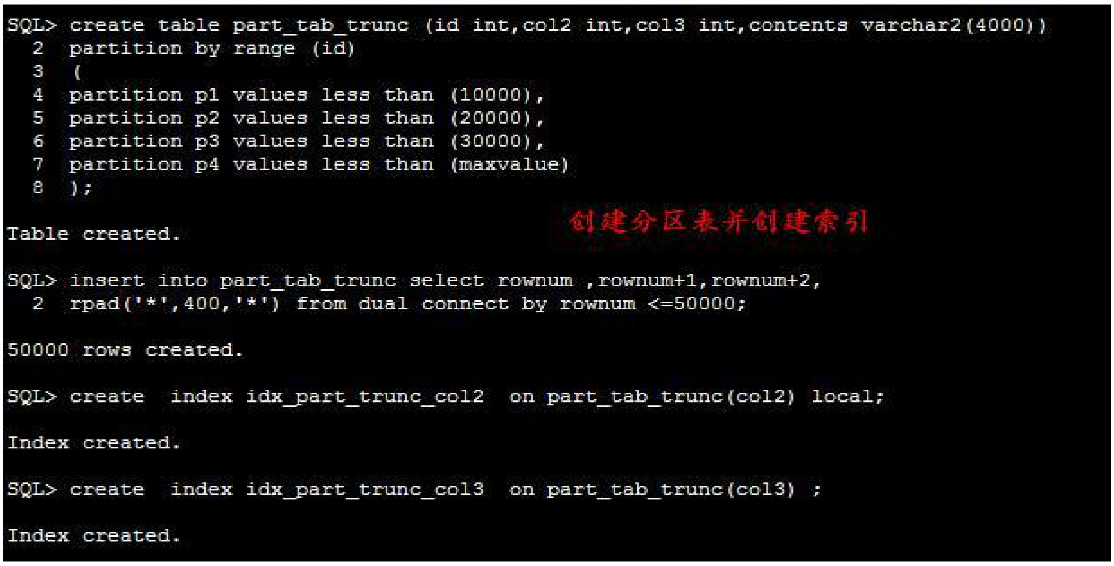


```
SQL> delete t where object_id>2000;

7540 rows deleted.

SQL> commit;

SQL> alter table t move;

Table altered.

SQL> select index_name,status from user_indexes
  2  where index_name='IDX_OBJECT_ID';

INDEX_NAME                     STATUS
------------------------------ --------
IDX_OBJECT_ID                  UNUSABLE

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

*
ERROR at line 1:
ORA-20000: index "SCOTT"."IDX_OBJECT_ID"  or partition of such index is in unusable state
ORA-06512: at "SYS.DBMS_STATS", line 13056
ORA-06512: at "SYS.DBMS_STATS", line 13076
ORA-06512: at line 1
```



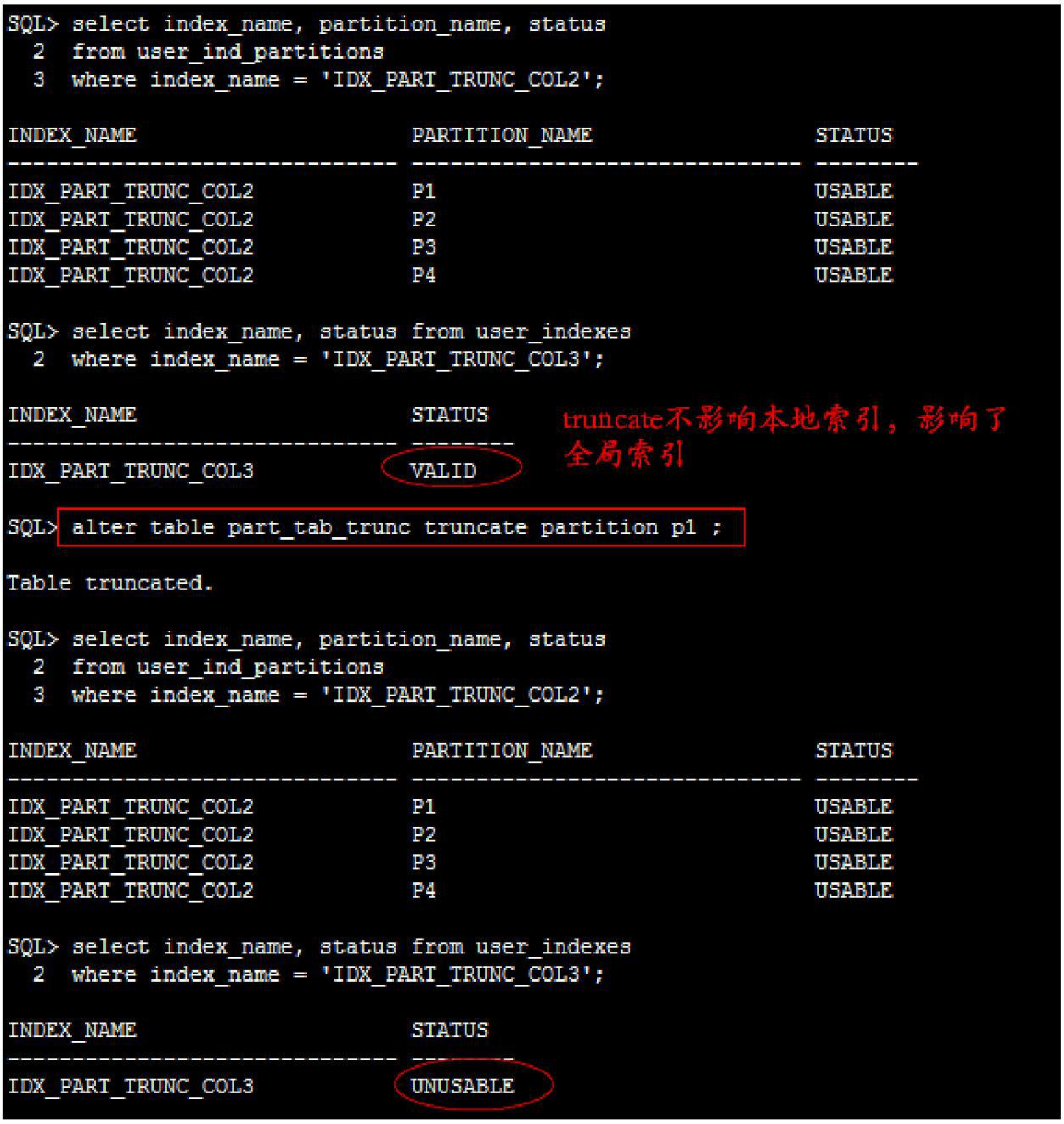
```
SQL> alter index IDX_OBJECT_ID rebuild;

Index altered.

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

PL/SQL procedure successfully completed.

SQL> select table_name,MAX_EXTENTS,NUM_ROWS,BLOCKS,EMPTY_BLOCKS
  2  from user_tables where table_name='T';

TABLE_NAME                     MAX_EXTENTS   NUM_ROWS     BLOCKS EMPTY_BLOCKS
------------------------------ ----------- ---------- ---------- ------------
T                               2147483645       1954         24            0
```

2.3 分区表致索引失效



```
SQL> create table part_tab_trunc (id int,col2 int,col3 int,contents varchar2(4000))
  2  partition by range (id)
  3  (
  4  partition p1 values less than (10000),
  5  partition p2 values less than (20000),
  6  partition p3 values less than (30000),
  7  partition p4 values less than (maxvalue)
  8  );

Table created.

SQL> insert into part_tab_trunc select rownum ,rownum+1,rownum+2,
  2  rpad('*',400,'*') from dual connect by rownum <=50000;

50000 rows created.

SQL> create  index idx_part_trunc_col2  on part_tab_trunc(col2) local;

Index created.

SQL> create  index idx_part_trunc_col3  on part_tab_trunc(col3) ;

Index created.
```

创建分区表并创建索引



```
SQL> select index_name, partition_name, status
  2  from user_ind_partitions
  3  where index_name = 'IDX_PART_TRUNC_COL2';

INDEX_NAME                     PARTITION_NAME                 STATUS
------------------------------ ------------------------------ --------
IDX_PART_TRUNC_COL2            P1                             USABLE
IDX_PART_TRUNC_COL2            P2                             USABLE
IDX_PART_TRUNC_COL2            P3                             USABLE
IDX_PART_TRUNC_COL2            P4                             USABLE

SQL> select index_name, status from user_indexes
  2  where index_name = 'IDX_PART_TRUNC_COL3';

INDEX_NAME                     STATUS
------------------------------ --------
IDX_PART_TRUNC_COL3            VALID

SQL> alter table part_tab_trunc truncate partition p1 ;

Table truncated.

SQL> select index_name, partition_name, status
  2  from user_ind_partitions
  3  where index_name = 'IDX_PART_TRUNC_COL2';

INDEX_NAME                     PARTITION_NAME                 STATUS
------------------------------ ------------------------------ --------
IDX_PART_TRUNC_COL2            P1                             USABLE
IDX_PART_TRUNC_COL2            P2                             USABLE
IDX_PART_TRUNC_COL2            P3                             USABLE
IDX_PART_TRUNC_COL2            P4                             USABLE

SQL> select index_name, status from user_indexes
  2  where index_name = 'IDX_PART_TRUNC_COL3';

INDEX_NAME                     STATUS
------------------------------ --------
IDX_PART_TRUNC_COL3            UNUSABLE
```

truncate不影响本地索引，影响了全局索引

其他就不再做实验，总结归纳。

1.truncate分区会导致全局索引失效，不会导致局部索引失效。
    如果truncate 增加update global indexes，全局索引不会失效。
2.drop分区会导致全局索引失效，局部索引因为drop分区，所以也不存在该分区的局部索引了。
    如果drop分区增加update global indexes，全局索引不会失效。
3.split分区会导致全局索引失效，也会导致局部索引失效。
    如果split分区增加update global indexes，全局索引不会失效。
4.add 分区不会导致全局索引失效，也不会导致局部索引失效。
5.exchange会导致全局索引失效，不会导致局部索引失效。
     如果exchange分区增加update global indexes，全局索引不会失效。

重要结论：
1. 所有的全局索引，只要用到update global indexes ,都不会失效，
        其中add分区甚至不需要增加update global indexes都可以生效。
2. 局部索引的操作都不会失效，除了split分区。切记split分区的时候，
        要将局部索引进行rebuild;

3.说说你对影响数据插入性能影响的认识。

3.1 索引是把双刃剑

索引是把双刃剑，它既可以提高查询的速度，单也降低了更新的速度。在一次查询中如果可以利用索引就可以避免对表的全部扫描，从而大大减少一致性读的数量，但当我们维护数据时，同时又需要维护索引，我们知道索引是有序排放的，那么维护索引的开销就会变得很大，所以有时间我们需要插入大批量数据时，可以采取的办法是先把索引失效，待数据维护完成再启用索引，这样往往效率会比较高一下。

3.1 依据业务权衡

数据插入性能受到索引的影响，所以为了提高数据插入的性能，就需要我们根据具体业务的场景来决定索引的创建以及创建的类型等，如果对于一些不经常更新的表我们可以创建索引，对于一些经常更新的表我们为了插入性能的需要可以不创建索引，在一些重复率低的列或 OLTP 环境中创建普通索引，在一些重复率高的列或 OLAP 环境中创建位图索引等，不考虑场景的操作，索引更会加大资源消耗，更是影响数据插入的性能。