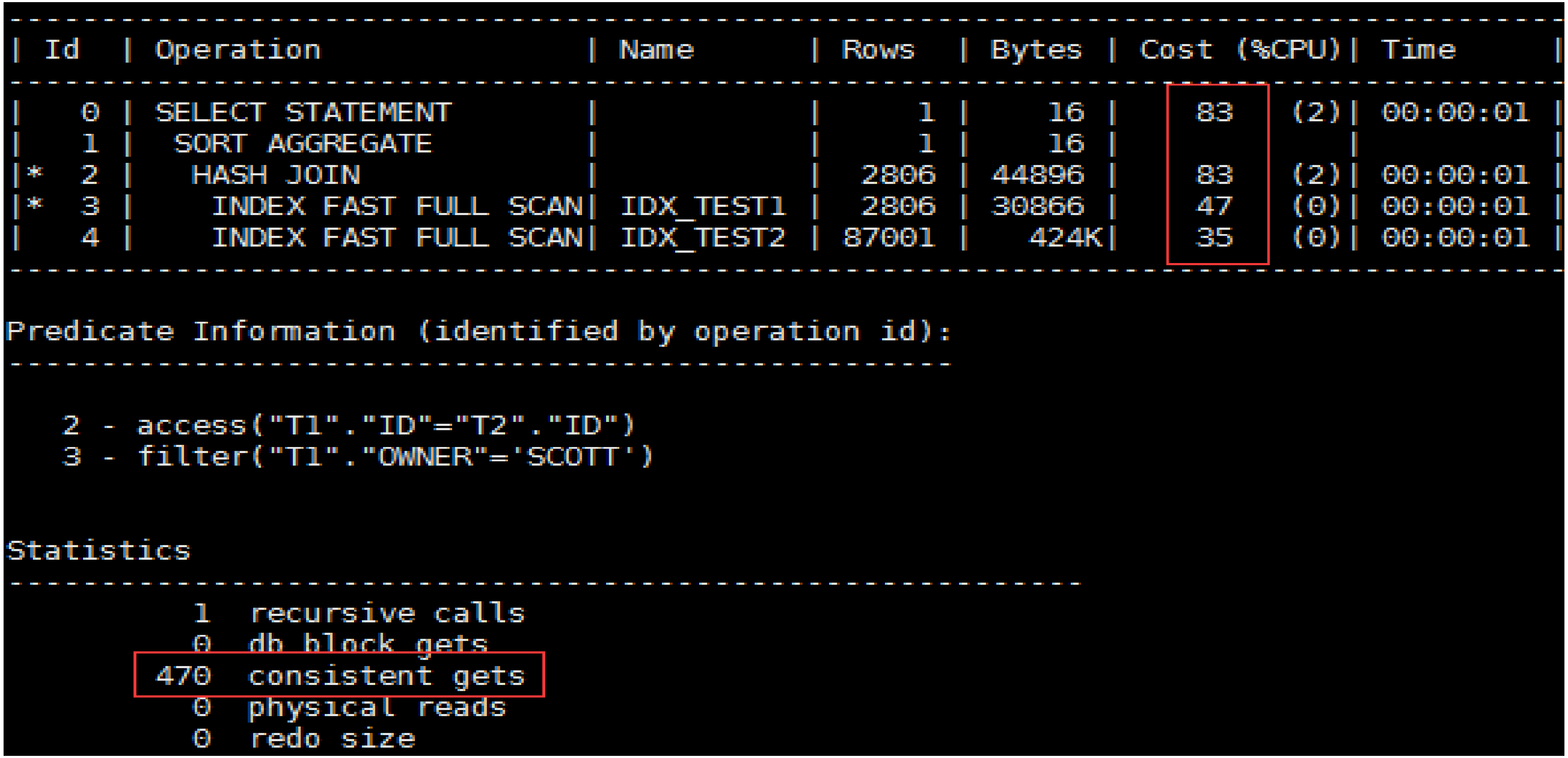
SQL优化实战技巧

Sql优化核心是什么?

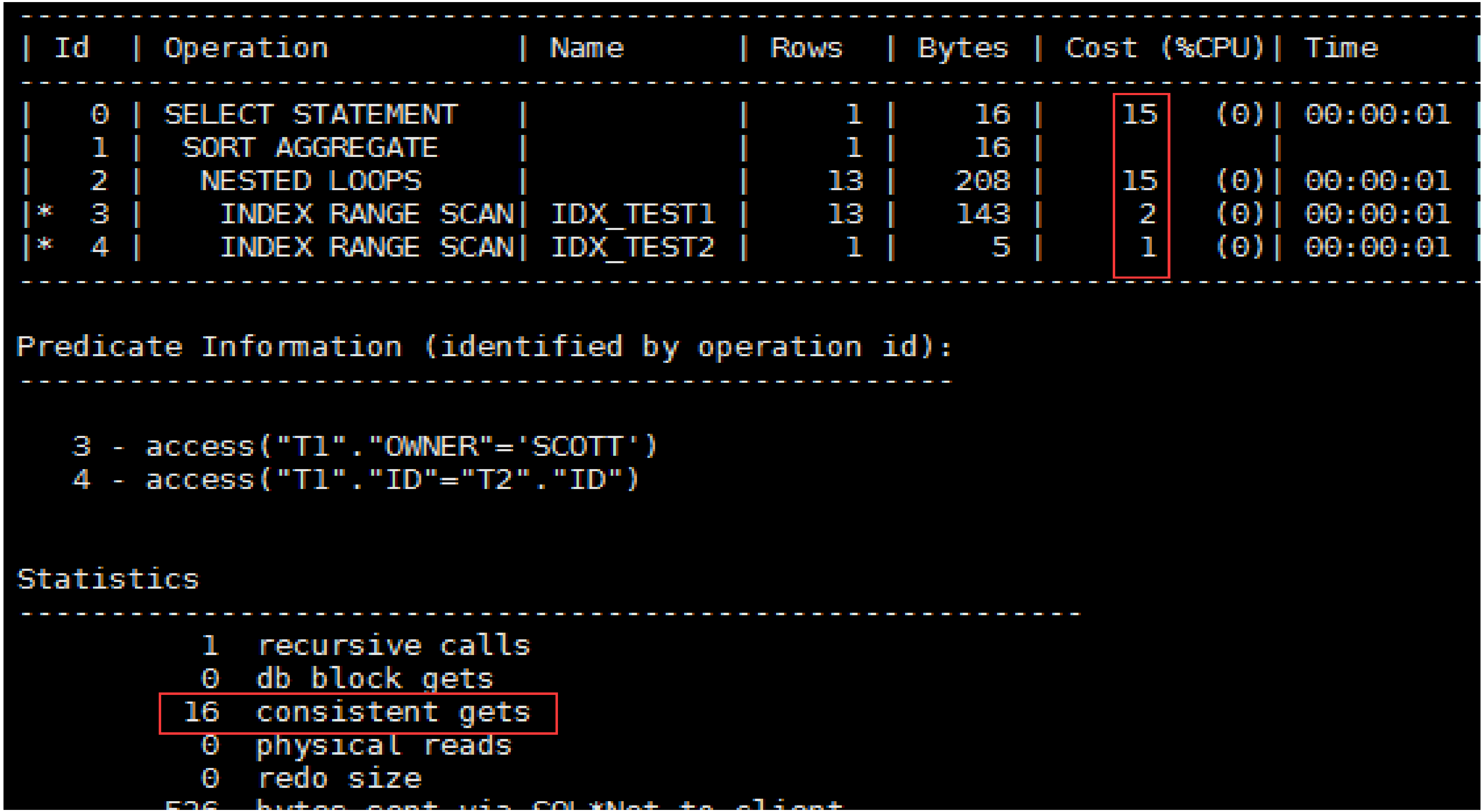
组合索引

```
create table test1(id,object_name,owner) as select object_id as
id,object_name,owner from dba_objects;

create table test2(id,object_type,status) as select object_id as id,status,temporary
from dba_objects;

select count(*) from test1 t1,test2 t2 where t1.id=t2.id and t1.owner='SCOTT';

create index idx_test1 on test1(id,owner);

create index idx_test2 on test2(id);

exec
dbms_stats.gather_table_stats(user,'test1',cascade=>true,estimate_percent=>100);

exec
dbms_stats.gather_table_stats(user,'test2',cascade=>true,estimate_percent=>100);
```



```
-------------------------------------------------------------------------------------
| Id  | Operation              | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |           |     1 |    16 |    83   (2)| 00:00:01 |
|   1 |  SORT AGGREGATE        |           |     1 |    16 |            |          |
|*  2 |   HASH JOIN            |           |  2806 | 44896 |    83   (2)| 00:00:01 |
|*  3 |    INDEX FAST FULL SCAN| IDX_TEST1 |  2806 | 30866 |    47   (0)| 00:00:01 |
|   4 |    INDEX FAST FULL SCAN| IDX_TEST2 | 87001 |  424K|    35   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

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

   2 - access("T1"."ID"="T2"."ID")
   3 - filter("T1"."OWNER"='SCOTT')


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

CREATE INDEX IDX_TEST1 ON TEST1(OWNER,ID);



```
----------------------------------------------------------------------------------
| Id  | Operation           | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |           |     1 |    16 |    15   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE     |           |     1 |    16 |            |          |
|   2 |   NESTED LOOPS      |           |    13 |   208 |    15   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN| IDX_TEST1 |    13 |   143 |     2   (0)| 00:00:01 |
|*  4 |    INDEX RANGE SCAN| IDX_TEST2 |     1 |     5 |     1   (0)| 00:00:01 |
----------------------------------------------------------------------------------

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

   3 - access("T1"."OWNER"='SCOTT')
   4 - access("T1"."ID"="T2"."ID")


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
         16  consistent gets
          0  physical reads
          0  redo size
        526  bytes sent via SQL*Net to client
```

组合索引创建技巧

适用于在单独查询返回记录很多，而组合查询之后返回记录很少的情况

选择过滤条件作为引导列

尽量把join列放在组合索引的最后面，即使join选择性很高

引导列的选择性越高越好

仅等值查询时，组合索引的顺序是不影响性能的

虚拟索引

在数据库优化过程中，索引的重要性是不言而喻的，但是在我们进行性能调整过程中，一个索引是否能够被使用到，在索引创建之前是无法确定的，而创建索引又是一个代价很高的操作，尤其是数据量很大的情况下，这时候我

们就可以考虑使用虚拟索引

特点：

无法执行alter index

不能创建和虚拟索引同名的实际索引

数据字典中查不到

create table test as select * from dba_objects;

--创建虚拟索引，首先要将_use_nosegment_indexes的隐含参数设置为true

alter session set "_use_nosegment_indexes"=true;

create index ix_test on test(object_id) nosegment;

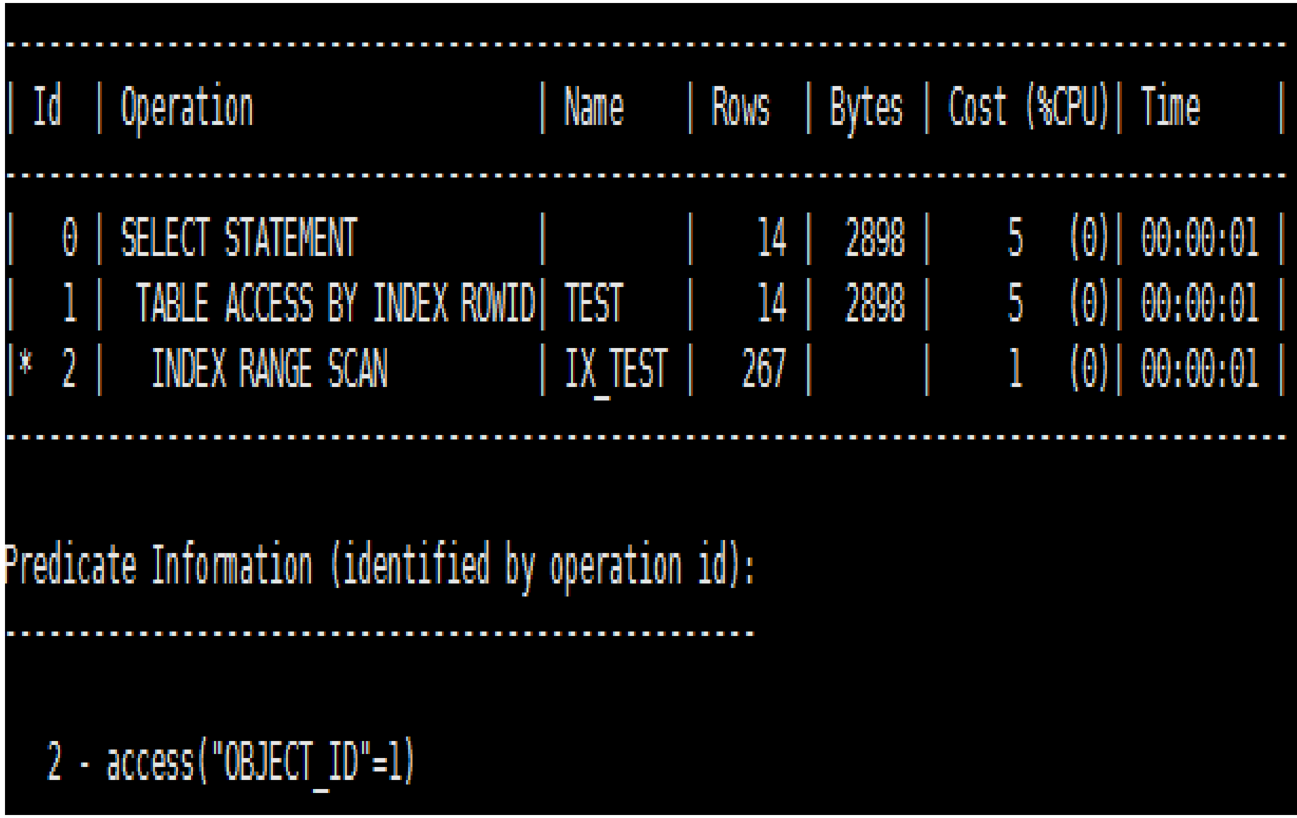
explain plan for select * from test where object_id=1;

set linesize 1000

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

set autotrace traceonly

select * from test where object_id=1;



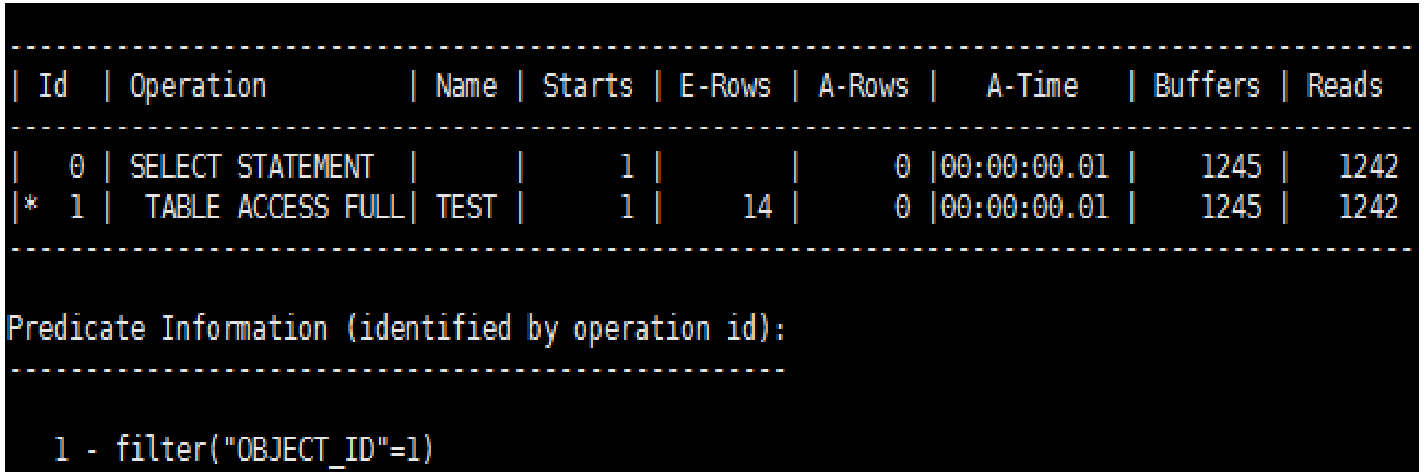
```
---------------------------------------------------------------------------------
| Id  | Operation                   | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |         |    14 |  2898 |     5   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| TEST    |    14 |  2898 |     5   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | IX_TEST |   267 |       |     1   (0)| 00:00:01 |
---------------------------------------------------------------------------------

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

   2 - access("OBJECT_ID"=1)
```

--以下看的是真实执行计划，显然是用不到索引。

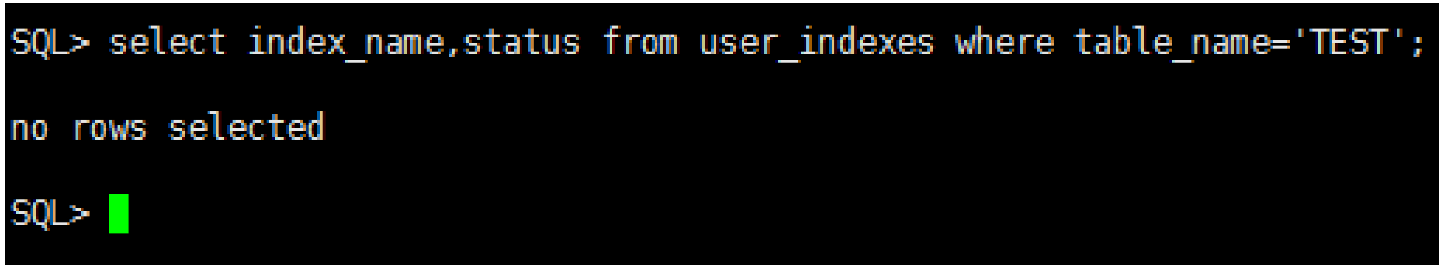
alter session set statistics_level=all;

select * from test where object_id=1;

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



```
-----------------------------------------------------------------------------------------
| Id  | Operation         | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers | Reads
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |      1 |        |      0 |00:00:00.01 |    1245 |  1242
|*  1 |  TABLE ACCESS FULL| TEST |      1 |     14 |      0 |00:00:00.01 |    1245 |  1242
-----------------------------------------------------------------------------------------

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

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

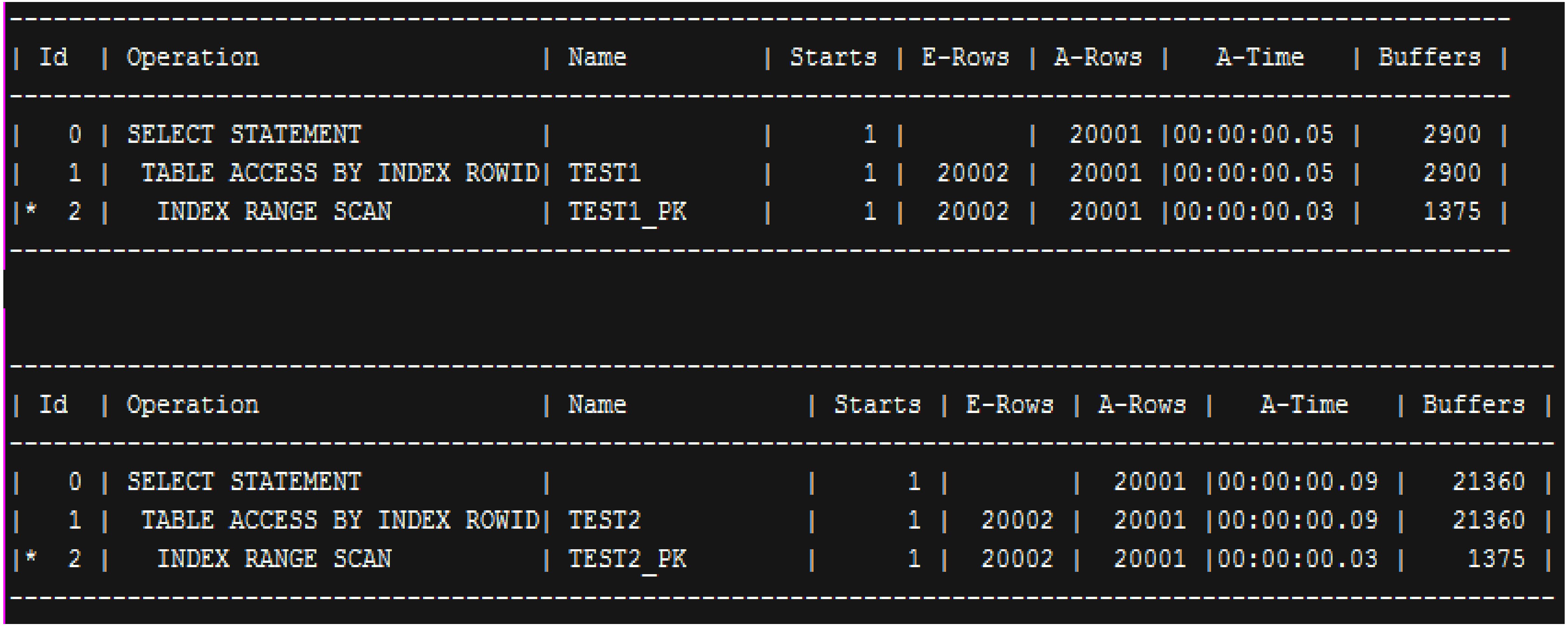


```
SQL> select index_name,status from user_indexes where table_name='TEST';

no rows selected

SQL> 
```

聚簇因子

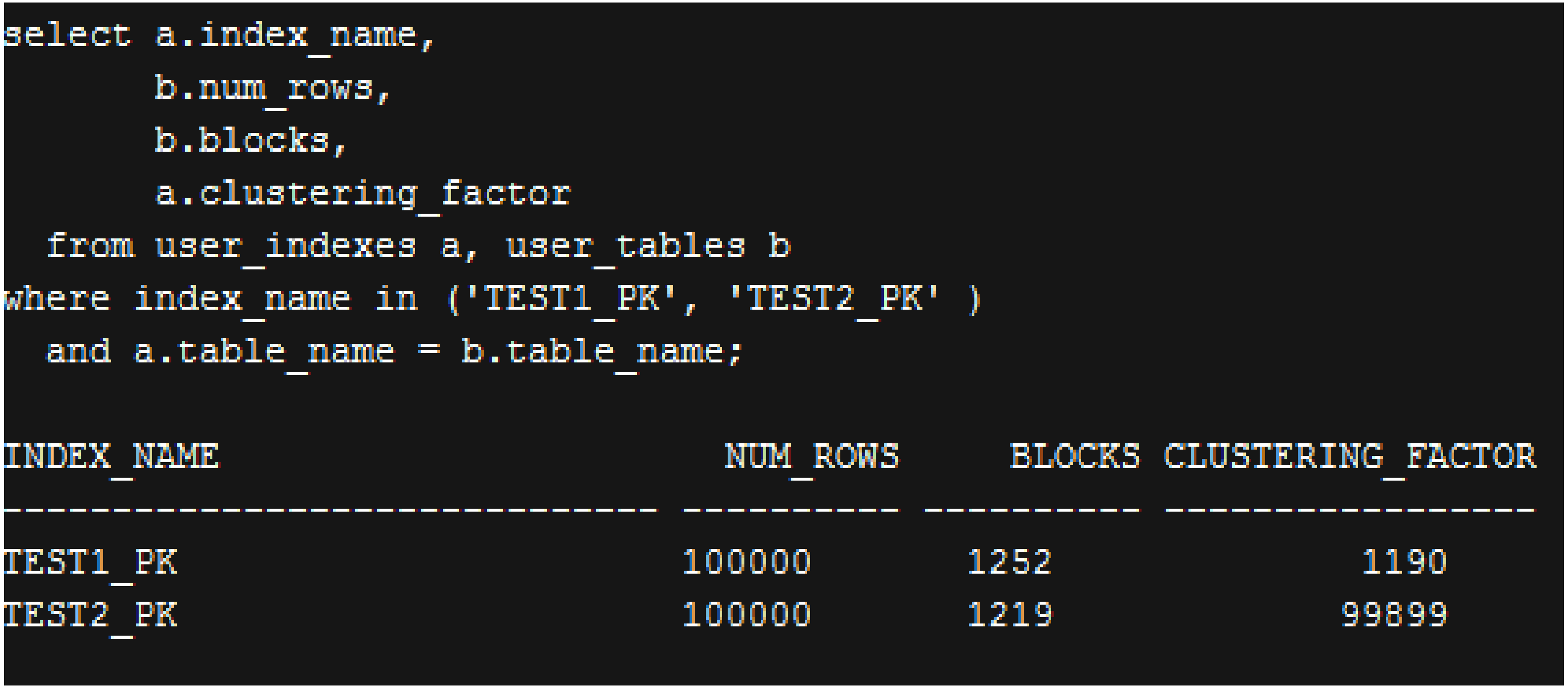
```
create table test1 ( a int, b varchar2(80) );
begin
   for i in 1 .. 100000
   loop
      insert into test1(a,b)
      values (i, rpad(dbms_random.random,75,'*') );
   end loop;
end;
alter table test1 add constraint test1_pk primary key(a);
begin
dbms_stats.gather_table_stats( user, 'TEST1', cascade=>true );
end;
select /*+ index( test1 test1_pk ) */ * from test where a between 20000 and 40000;
```

```
create table test2
as
select a,b
 from test1
order by b;
alter table test2 add constraint test2_pk primary key (a);
begin
dbms_stats.gather_table_stats( user, 'TEST2', cascade=>true );
end;

select /*+ index( test2 test2_pk ) */ * from test where a between 20000 and 40000;
```



```
-------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name          | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |               |      1 |        |  20001 |00:00:00.05 |    2900 |
|   1 |  TABLE ACCESS BY INDEX ROWID| TEST1         |      1 |  20002 |  20001 |00:00:00.05 |    2900 |
|*  2 |   INDEX RANGE SCAN          | TEST1_PK      |      1 |  20002 |  20001 |00:00:00.03 |    1375 |
-------------------------------------------------------------------------------------------------------
```

```
----------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name             | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                  |      1 |        |  20001 |00:00:00.09 |   21360 |
|   1 |  TABLE ACCESS BY INDEX ROWID| TEST2            |      1 |  20002 |  20001 |00:00:00.09 |   21360 |
|*  2 |   INDEX RANGE SCAN          | TEST2_PK         |      1 |  20002 |  20001 |00:00:00.03 |    1375 |
----------------------------------------------------------------------------------------------------------
```



```
select a.index_name,
       b.num_rows,
       b.blocks,
       a.clustering_factor
  from user_indexes a, user_tables b
where index_name in ('TEST1_PK', 'TEST2_PK' )
  and a.table_name = b.table_name;

INDEX_NAME                        NUM_ROWS     BLOCKS CLUSTERING_FACTOR
------------------------------ ---------- ---------- -----------------
TEST1_PK                       100000       1252                  1190
TEST2_PK                       100000       1219                 99899
```

视图合并

当sql中出现内联视图或是通过create view语句创建的视图时，CBO会将视图进行展开，进行等价改写，这个过程就叫视图合并

内联视图优化技巧

是否发生了视图合并，如果没有发生视图合并，在执行计划中，一般我们都能看到view关键字

当子查询或视图中有以下情况，那么视图是不会合并的

Union,union all,instersact,minus

Avg,count,max,min,sum

Rownum

Connect by

Group by

distinct

谓词推入

什么是谓词推入?

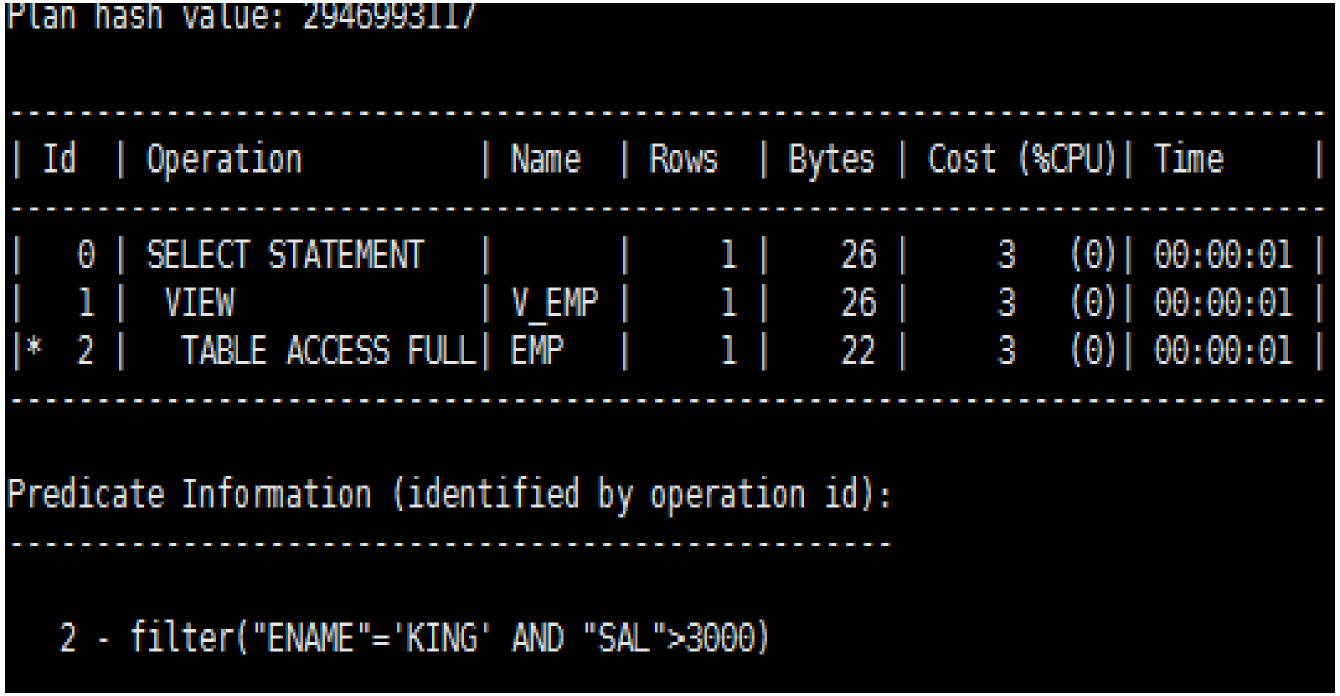
当sql语句中包含不能合并的视图，并且视图有谓词过滤，那么Oracle CBO就会将where过滤条件推入到视图中，这个就是谓词过滤。

谓词推入目的?

谓词过滤注意就是让oracle尽可能早的过滤掉无用的数据，提升sql运行性能。

SQL> create or replace view v_emp as select /*+ no_merge */ empno,ename,job from emp where sal>3000;

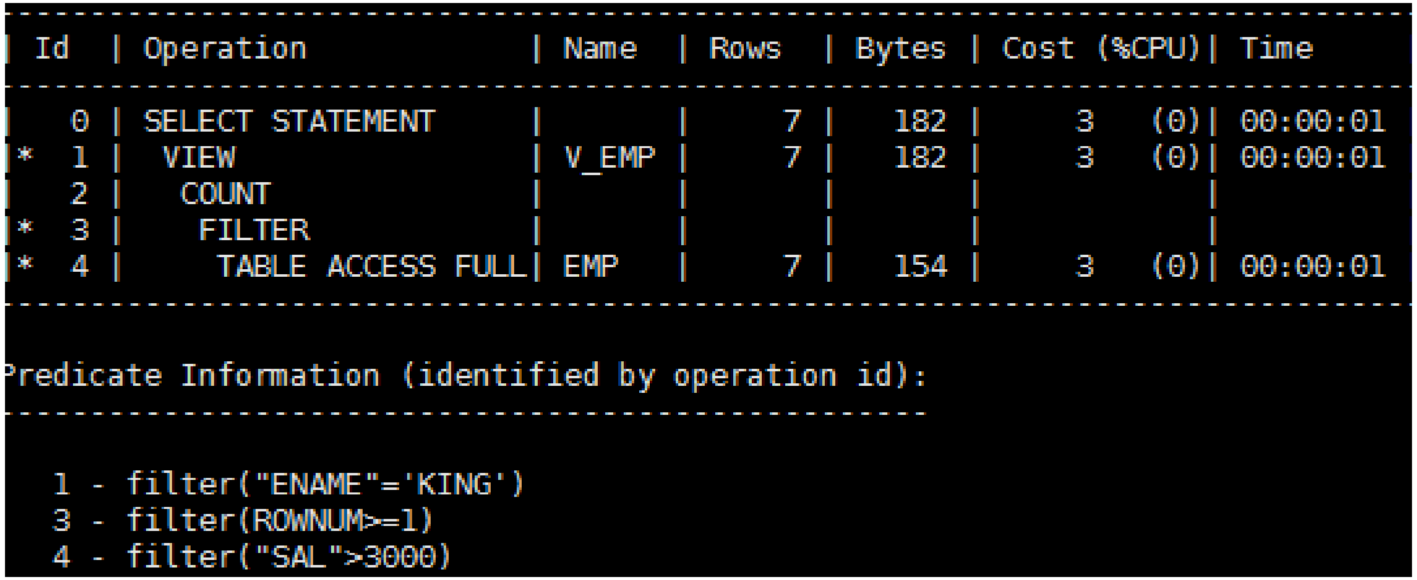
SQL> select * from v_emp where ename='KING';



```
Plan hash value: 2946993117


--------------------------------------------------------------------------
| Id  | Operation          | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |       |     1 |    26 |     3   (0)| 00:00:01 |
|   1 |  VIEW              | V_EMP |     1 |    26 |     3   (0)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL| EMP   |     1 |    22 |     3   (0)| 00:00:01 |
--------------------------------------------------------------------------

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

   2 - filter("ENAME"='KING' AND "SAL">3000)
```

SQL> create or replace view v_emp as select /*+ no_merge */ empno,ename,job from emp where sal>3000 and rownum>=1;

SQL> select * from v_emp where ename='KING';



```
-------------------------------------------------------------------------------
| Id  | Operation            | Name  | Rows  | Bytes | Cost (%CPU)| Time
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |       |     7 |   182 |     3   (0)| 00:00:01
|*  1 |  VIEW                | V_EMP |     7 |   182 |     3   (0)| 00:00:01
|   2 |   COUNT              |       |       |       |            |
|*  3 |    FILTER            |       |       |       |            |
|*  4 |     TABLE ACCESS FULL| EMP   |     7 |   154 |     3   (0)| 00:00:01
-------------------------------------------------------------------------------

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

   1 - filter("ENAME"='KING')
   3 - filter(ROWNUM>=1)
   4 - filter("SAL">3000)
```

WITH AS

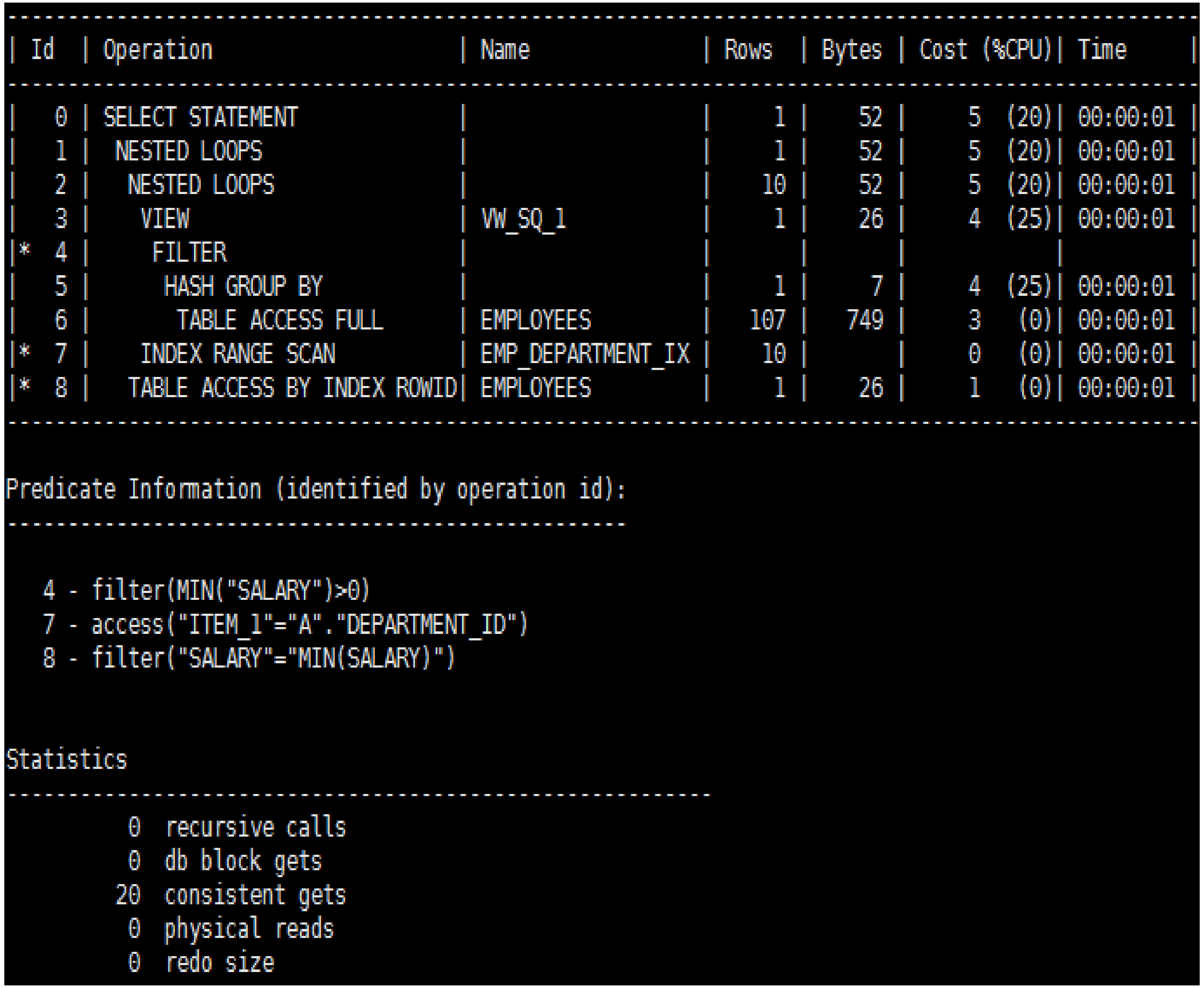
select employee_id,first_name,last_name, salary

from employees a

where salary=(select min(salary)

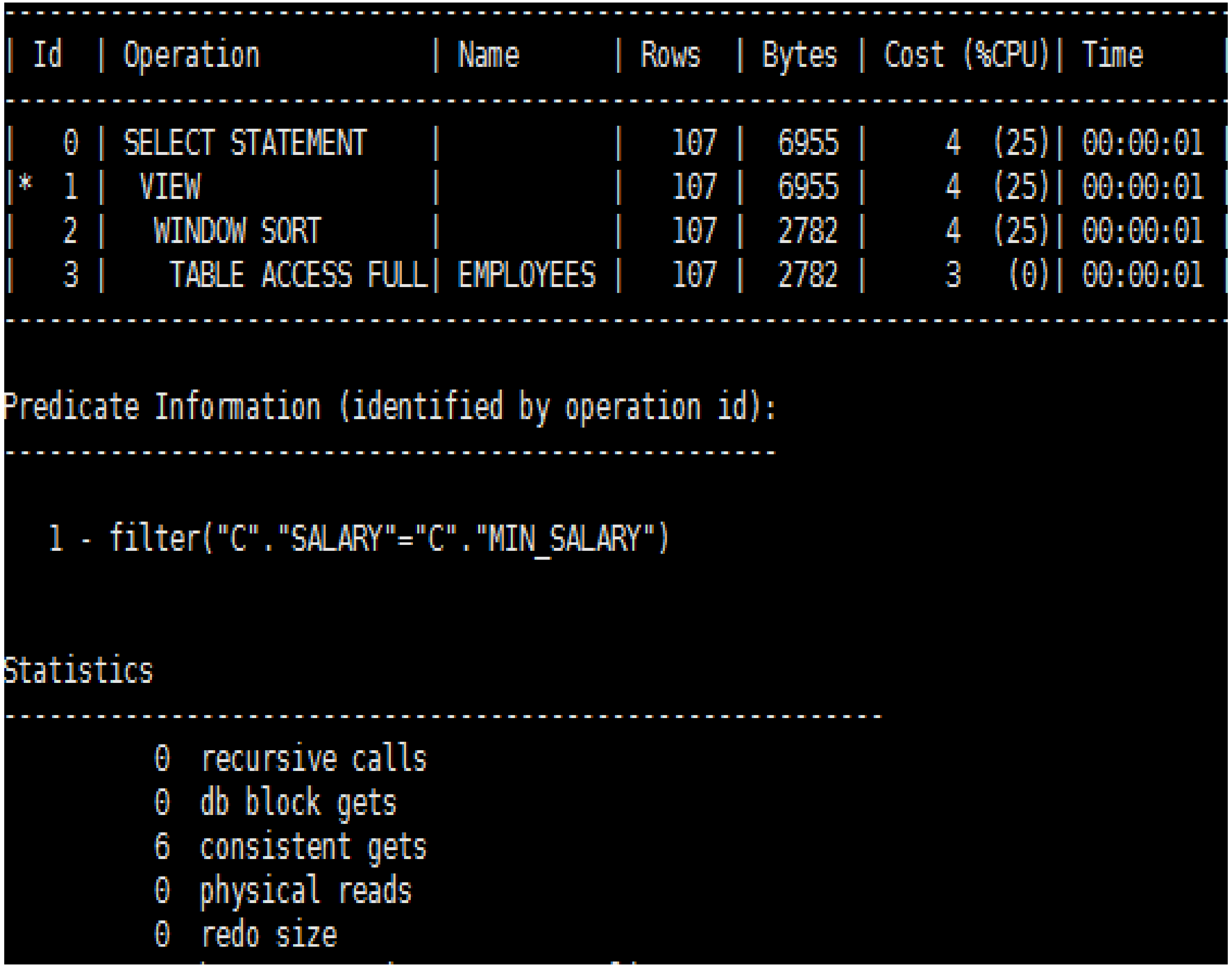
from employees b

where b.department_id=a.department_id);



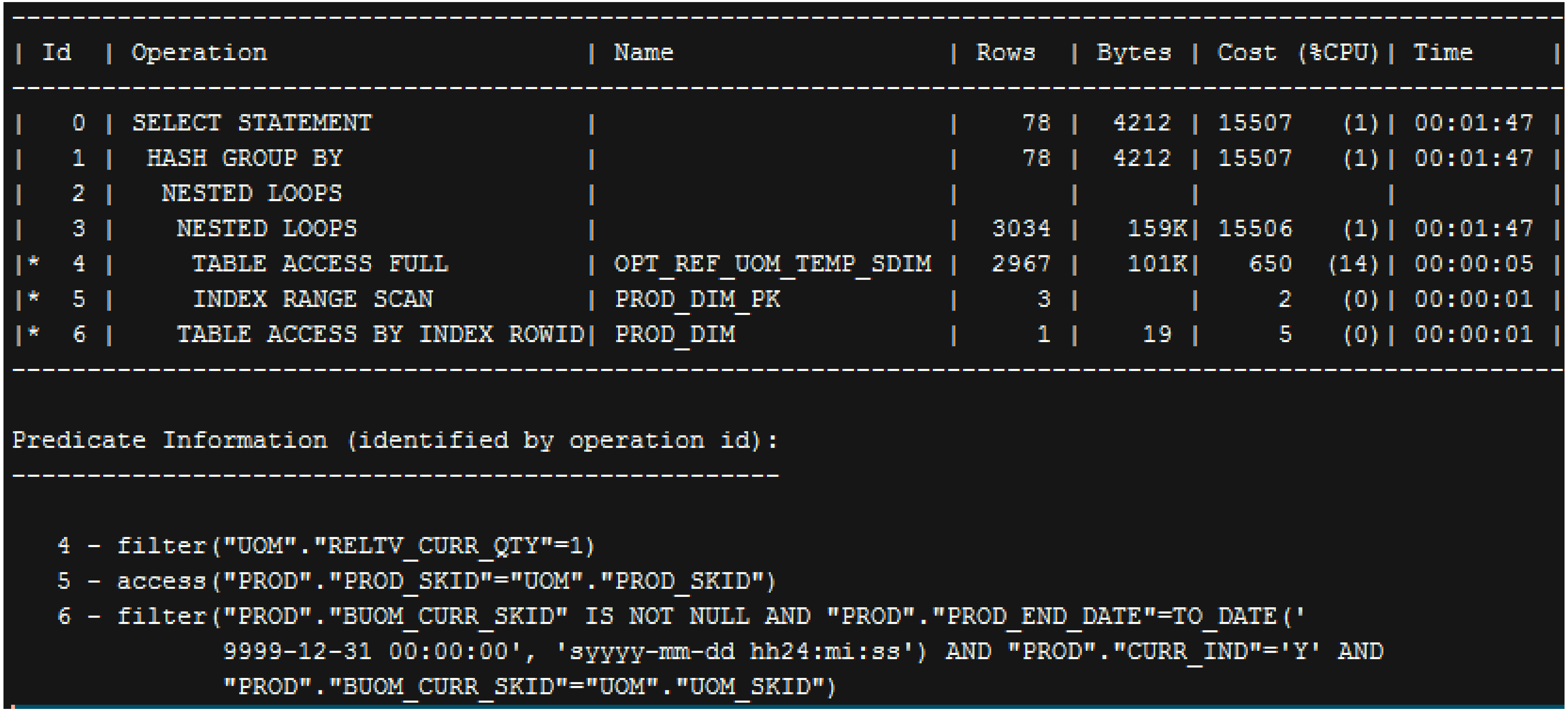
```
-----------------------------------------------------------------------------------------------
| Id  | Operation                    | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                   |     1 |    52 |     5  (20)| 00:00:01 |
|   1 |  NESTED LOOPS                |                   |     1 |    52 |     5  (20)| 00:00:01 |
|   2 |   NESTED LOOPS               |                   |    10 |    52 |     5  (20)| 00:00:01 |
|   3 |    VIEW                      | VW_SQ_1           |     1 |    26 |     4  (25)| 00:00:01 |
|*  4 |     FILTER                   |                   |       |       |            |          |
|   5 |      HASH GROUP BY           |                   |     1 |     7 |     4  (25)| 00:00:01 |
|   6 |       TABLE ACCESS FULL      | EMPLOYEES         |   107 |   749 |     3   (0)| 00:00:01 |
|*  7 |    INDEX RANGE SCAN          | EMP_DEPARTMENT_IX |    10 |       |     0   (0)| 00:00:01 |
|*  8 |   TABLE ACCESS BY INDEX ROWID| EMPLOYEES         |     1 |    26 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

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

   4 - filter(MIN("SALARY")>0)
   7 - access("ITEM_1"="A"."DEPARTMENT_ID")
   8 - filter("SALARY"="MIN(SALARY)")


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

```
with c as
(select e.*,min(salary) over(partition by department_id) as min_salary from employees e)
select employee_id,first_name,last_name,salary from c where c.salary=c.min_salary;
```



```
--------------------------------------------------------------------------------
| Id  | Operation            | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |           |   107 |  6955 |     4  (25)| 00:00:01 |
|*  1 |  VIEW                |           |   107 |  6955 |     4  (25)| 00:00:01 |
|   2 |   WINDOW SORT        |           |   107 |  2782 |     4  (25)| 00:00:01 |
|   3 |    TABLE ACCESS FULL| EMPLOYEES |   107 |  2782 |     3   (0)| 00:00:01 |
--------------------------------------------------------------------------------

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

   1 - filter("C"."SALARY"="C"."MIN_SALARY")


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

NESTED LOOPS



```
-------------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name                  | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                       |    78 |  4212 | 15507   (1)| 00:01:47 |
|   1 |  HASH GROUP BY                |                       |    78 |  4212 | 15507   (1)| 00:01:47 |
|   2 |   NESTED LOOPS                |                       |       |       |            |          |
|   3 |    NESTED LOOPS               |                       |  3034 |   159K| 15506   (1)| 00:01:47 |
|*  4 |     TABLE ACCESS FULL         | OPT_REF_UOM_TEMP_SDIM |  2967 |   101K|   650  (14)| 00:00:05 |
|*  5 |     INDEX RANGE SCAN          | PROD_DIM_PK           |     3 |       |     2   (0)| 00:00:01 |
|*  6 |    TABLE ACCESS BY INDEX ROWID| PROD_DIM              |     1 |    19 |     5   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------

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

   4 - filter("UOM"."RELTV_CURR_QTY"=1)
   5 - access("PROD"."PROD_SKID"="UOM"."PROD_SKID")
   6 - filter("PROD"."BUOM_CURR_SKID" IS NOT NULL AND "PROD"."PROD_END_DATE"=TO_DATE('
              9999-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND "PROD"."CURR_IND"='Y' AND
              "PROD"."BUOM_CURR_SKID"="UOM"."UOM_SKID")
```

NESTED LOOPS优化技巧

根据nested loops原理，我们有以下优化技巧

驱动表的过滤条件要有索引

被驱动表的join字段要有索引

驱动表结果集要小

基数评估

create table test1 as select * from dba_objects;

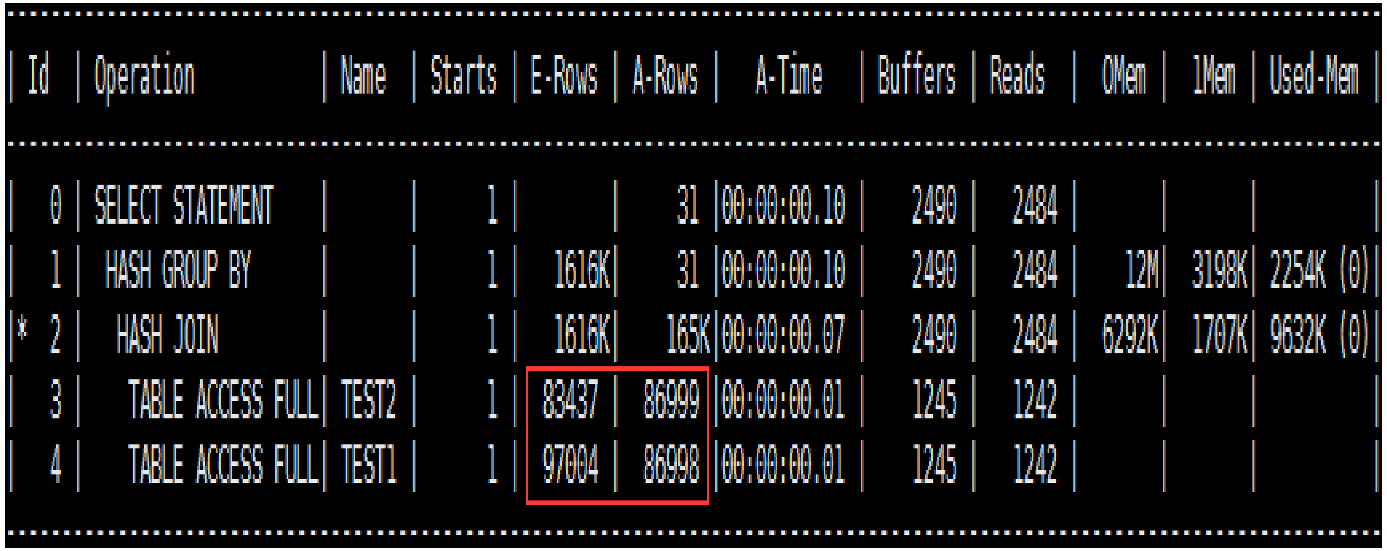
create table test2 as select * from dba_objects;

select /*+ gather_plan_statistics */ a.owner,count(*) from test1 a,test2 b

where a.object_name=b.object_name

group by a.owner;

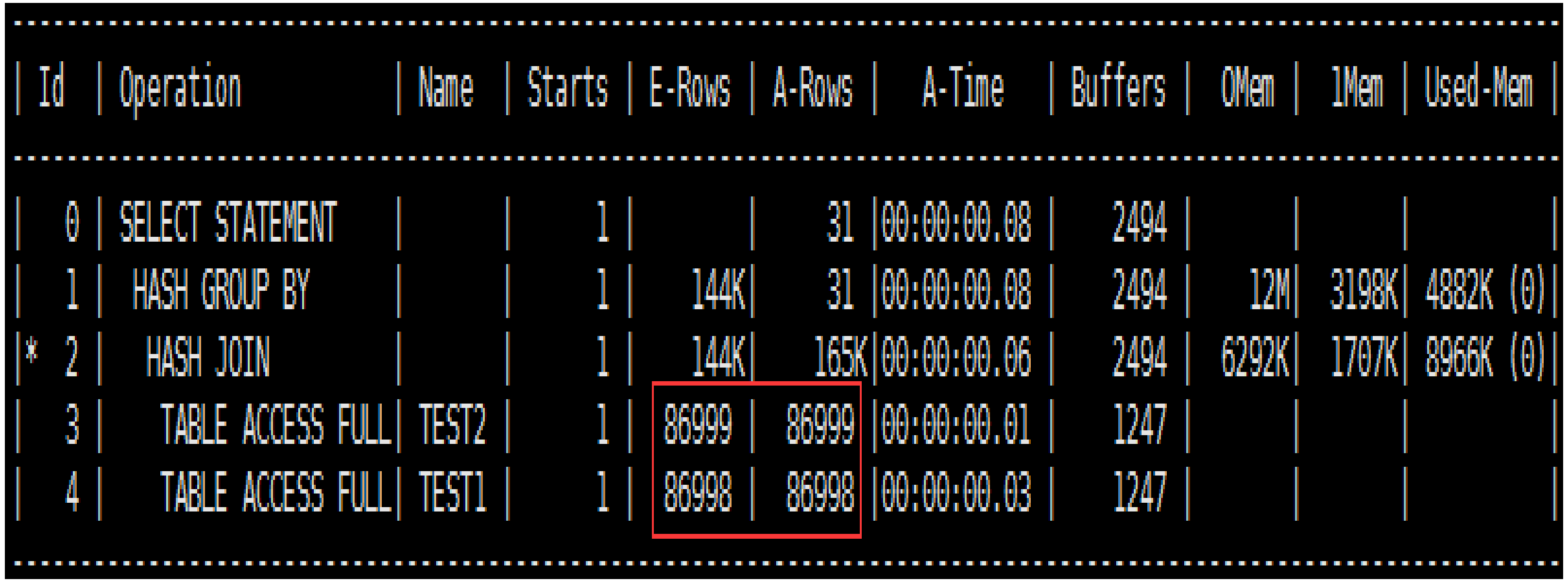
select * from table(dbms_xplan.display_cursor(null,null,'ALLSTATS LAST'));



```
---------------------------------------------------------------------------------------------------------------------
| Id  | Operation          | Name  | Starts | E-Rows | A-Rows |   A-Time    | Buffers | Reads  |  OMem |  1Mem | Used-Mem |
---------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |       |      1 |        |     31 |00:00:00.10 |    2490 |   2484 |       |       |          |
|   1 |  HASH GROUP BY     |       |      1 |  1616K|     31 |00:00:00.10 |    2490 |   2484 |    12M|  3198K| 2254K (0)|
|*  2 |   HASH JOIN        |       |      1 |  1616K|    165K|00:00:00.07 |    2490 |   2484 |  6292K|  1707K| 9632K (0)|
|   3 |    TABLE ACCESS FULL| TEST2 |      1 |  83437 |  86999 |00:00:00.01 |    1245 |   1242 |       |       |          |
|   4 |    TABLE ACCESS FULL| TEST1 |      1 |  97004 |  86998 |00:00:00.01 |    1245 |   1242 |       |       |          |
---------------------------------------------------------------------------------------------------------------------
```

```
select /*+ gather_plan_statistics dynamic_sampling(a 10) dynamic_sampling(b 10) */

a.owner,count(*) from test1 a,test2 b

where a.object_name=b.object_name

group by a.owner;
```

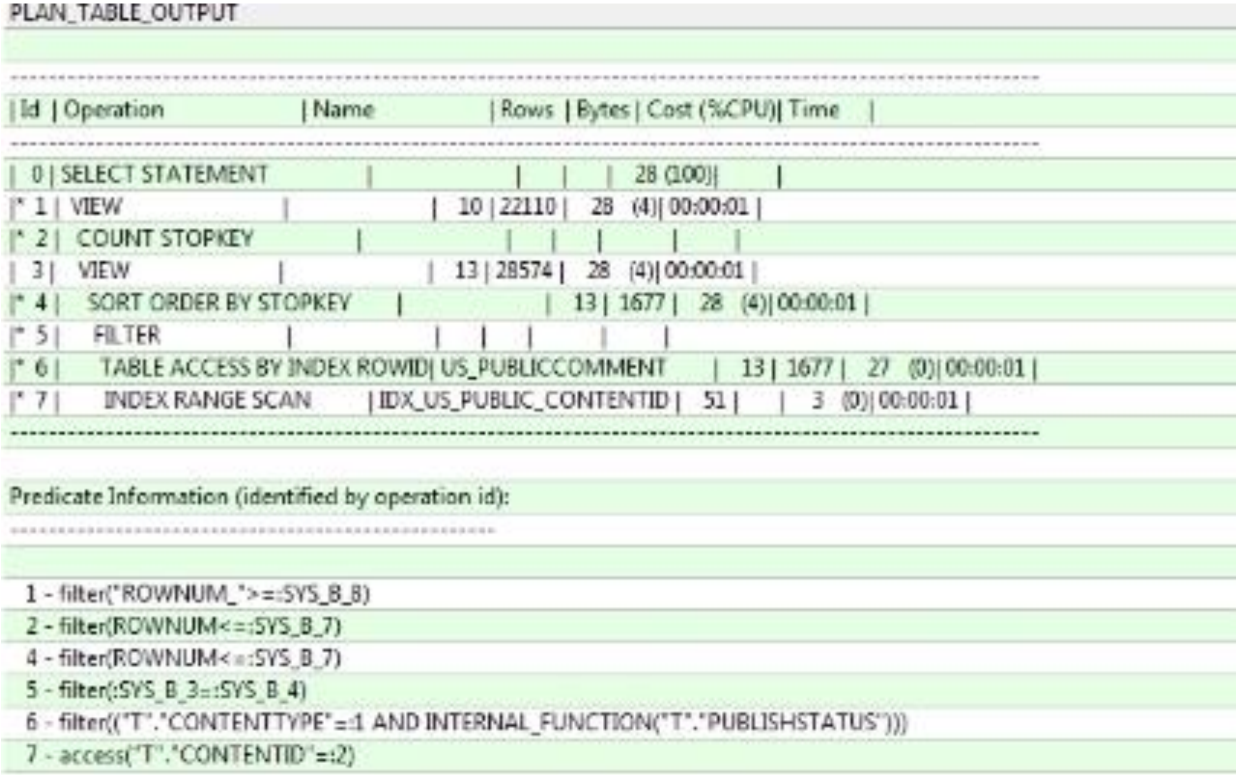
```
select * from table(dbms_xplan.display_cursor(null,null,'ALLSTATS LAST'));
```



```
-----------------------------------------------------------------------------------------------------------------
| Id  | Operation           | Name  | Starts | E-Rows | A-Rows |   A-Time   | Buffers |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |       |      1 |        |     31 |00:00:00.08 |    2494 |       |       |          |
|   1 |  HASH GROUP BY      |       |      1 |   144K |     31 |00:00:00.08 |    2494 |    12M|  3198K| 4882K (0)|
|*  2 |   HASH JOIN         |       |      1 |   144K |   165K |00:00:00.06 |    2494 |  6292K|  1707K| 8966K (0)|
|   3 |    TABLE ACCESS FULL| TEST2 |      1 |  86999 |  86999 |00:00:00.01 |    1247 |       |       |          |
|   4 |    TABLE ACCESS FULL| TEST1 |      1 |  86998 |  86998 |00:00:00.03 |    1247 |       |       |          |
-----------------------------------------------------------------------------------------------------------------
```

分页优化

```
select *
  from (select row_.*, rownum rownum_
          from (select t.bookreviewid,
                       t.msisdn,
                       t.contentid,
                       t.contenttype,
                       t.portaltype,
                       t.publishstatus,
                       t.commentary,
                       t.publishsdate,
                       t.createtime,
                       t.floorNum,
                       t.istop,
                       t.assessstatus,
                       t.isprime,
                       :"SYS_B_0" as createNick,
                       nvl(opposenum, :"SYS_B_1") as
opposenum,
                       nvl(abetnum, :"SYS_B_2") as abetnum,
                       t.replycontent,
                       t.latestreplytime
                  from us_publiccomment t
                 where :"SYS_B_3" = :"SYS_B_4"
                   and t.publishstatus in
(:"SYS_B_5", :"SYS_B_6")
                   and t.contenttype = :1

                   and t.contentid = :2

                 order by t.isTop desc, t.floornum desc) row_
         where rownum <= :"SYS_B_7")
 where rownum_ >= :"SYS_B_8"
```

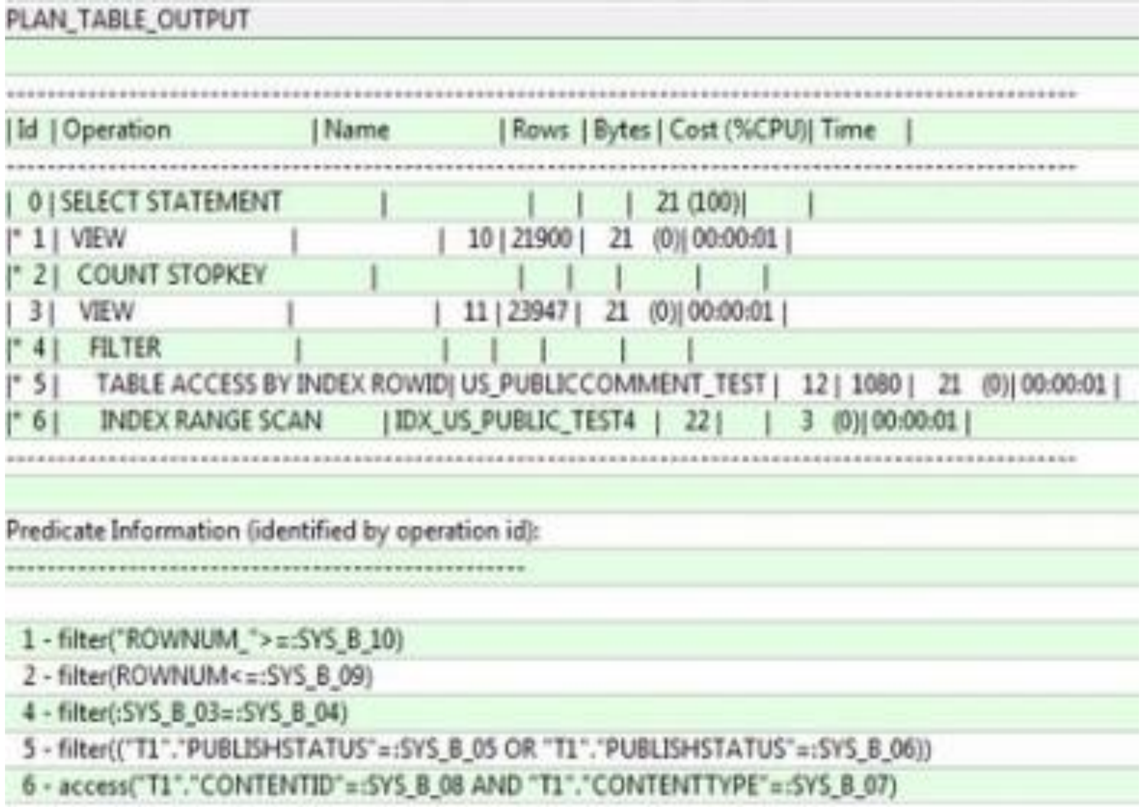


```
PLAN_TABLE_OUTPUT

-----------------------------------------------------------------------------------------
| Id | Operation                    | Name                   | Rows | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|  0 | SELECT STATEMENT             |                        |      |       |    28 (100)|          |
|* 1 |  VIEW                        |                        |   10 | 22110 |    28   (4)| 00:00:01 |
|* 2 |   COUNT STOPKEY              |                        |      |       |            |          |
|  3 |    VIEW                      |                        |   13 | 28574 |    28   (4)| 00:00:01 |
|* 4 |     SORT ORDER BY STOPKEY    |                        |   13 |  1677 |    28   (4)| 00:00:01 |
|* 5 |      FILTER                  |                        |      |       |            |          |
|* 6 |       TABLE ACCESS BY INDEX ROWID| US_PUBLICCOMMENT   |   13 |  1677 |    27   (0)| 00:00:01 |
|* 7 |        INDEX RANGE SCAN      | IDX_US_PUBLIC_CONTENTID |  51 |       |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

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

   1 - filter("ROWNUM_">=:SYS_B_8)
   2 - filter(ROWNUM<=:SYS_B_7)
   4 - filter(ROWNUM<=:SYS_B_7)
   5 - filter(:SYS_B_3=:SYS_B_4)
   6 - filter(("T"."CONTENTTYPE"=:1 AND INTERNAL_FUNCTION("T"."PUBLISHSTATUS")))
   7 - access("T"."CONTENTID"=:2)
```

create index MREAD.IDX_US_PUBLIC_CON_SORT on MREAD.us_publiccomment(CONTENTID,istop desc,floornum desc)

tablespace TBS_MREAD_IDX parallel 8 online

重新搜集统计信息



```
PLAN_TABLE_OUTPUT

--------------------------------------------------------------------------------------------
|Id  |Operation                    |Name                   |Rows  |Bytes|Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------
|  0 |SELECT STATEMENT             |                       |      |     |   21 (100)|          |
|* 1 | VIEW                        |                       |   10 |21900|   21   (0)| 00:00:01 |
|* 2 |  COUNT STOPKEY              |                       |      |     |           |          |
|  3 |   VIEW                      |                       |   11 |23947|   21   (0)| 00:00:01 |
|* 4 |    FILTER                   |                       |      |     |           |          |
|* 5 |     TABLE ACCESS BY INDEX ROWID| US_PUBLICCOMMENT_TEST |   12 | 1080|   21   (0)| 00:00:01 |
|* 6 |      INDEX RANGE SCAN       | IDX_US_PUBLIC_TEST4   |   22 |     |    3   (0)| 00:00:01 |
--------------------------------------------------------------------------------------------

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

   1 - filter("ROWNUM_">=:SYS_B_10)
   2 - filter(ROWNUM<=:SYS_B_09)
   4 - filter(:SYS_B_03=:SYS_B_04)
   5 - filter(("T1"."PUBLISHSTATUS"=:SYS_B_05 OR "T1"."PUBLISHSTATUS"=:SYS_B_06))
   6 - access("T1"."CONTENTID"=:SYS_B_08 AND "T1"."CONTENTTYPE"=:SYS_B_07)
```

分页优化技巧

- Sql语句中有rownum<，只有语句中有rownum< 执行计划中才会有stopkey关键字
- Order by 后面的字段必须建索引

WMSYS.WM_CONCAT函数引发的故障

早上老综合库在9点24分的时候出现了ORA-600错误，错误内容如下：

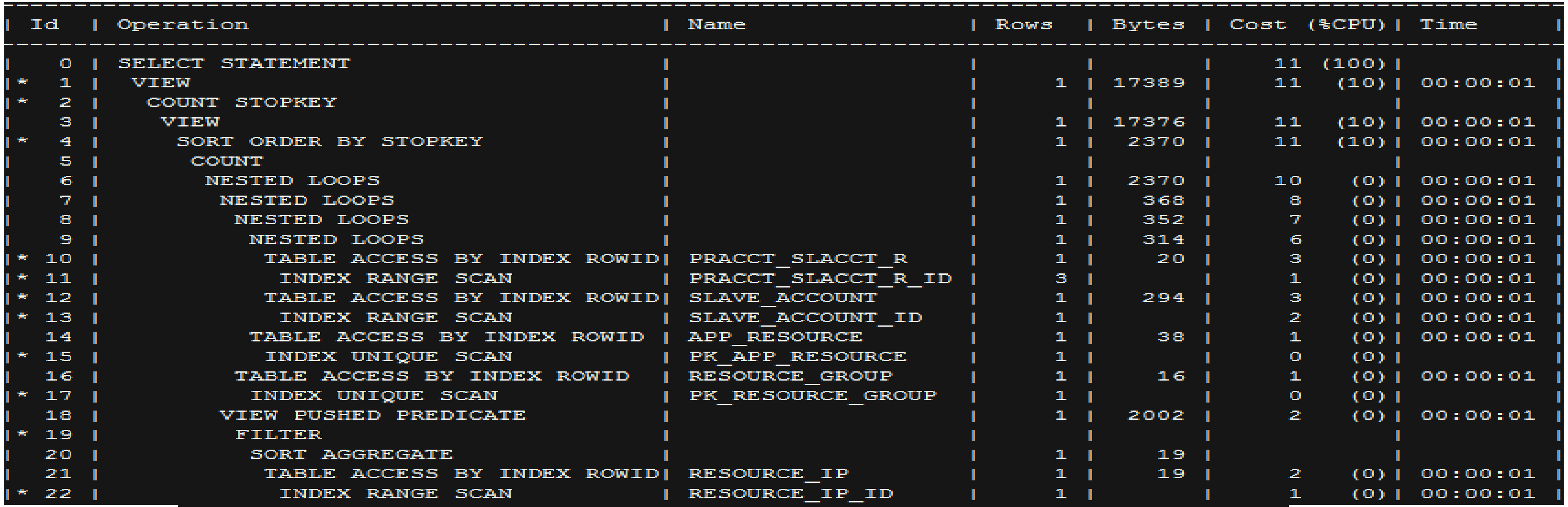
```
Tue May 12 09:24:52 2015
Errors in file
/oracle/database/diag/rdbms/integ/integ3/trace/integ3_ora_910.trc  (incident=755353):
ORA-00600: 内部错误代码, 参数: [kokegPinLob1], [], [], [], [], [], [], [], [], [], [], []
Incident details in:
/oracle/database/diag/rdbms/integ/integ3/incident/incdir_755353/integ3_ora_910_i755353.trc
Tue May 12 09:25:27 2015
Errors in file
/oracle/database/diag/rdbms/integ/integ3/trace/integ3_ora_2327.trc  (incident=754858):
ORA-00600: 内部错误代码, 参数: [kokegPinLob1], [], [], [], [], [], [], [], [], [], [], []
Incident details in:
/oracle/database/diag/rdbms/integ/integ3/incident/incdir_754858/integ3_ora_2327_i754858.trc
```

通过跟踪后台日志发现是下列SQL引起的:

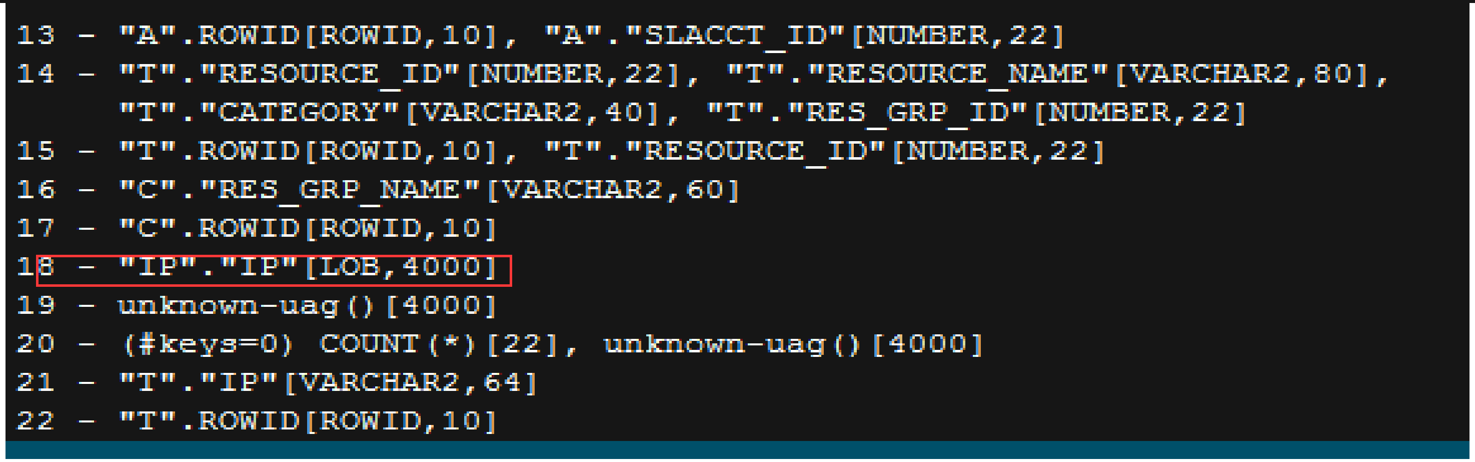
```
========= Dump for incident 755353 (ORA 600 [kokegPinLob1]) ========
```

SQL语 句

```
*** 2015-05-12 09:24:52.923
dbkedDefDump(): Starting incident default dumps (flags=0x2, level=3, mask=0x0)
----- Current SQL Statement for this session (sql_id=5nz1w5b0f1dxy) -----
select *      from (select r.*, ROWNUM rn       from (select
A.*,T.RESOURCE_NAME,C.RES_GRP_NAME,
T.CATEGORY,ip.ip,B.TYPE,B.start_time,ROWNUM      from SLAVE_ACCOUNT
A,      PRACCT_SLACCT_R B ,       APP_RESOURCE T,  RESOURCE_GROUP C ,(select
t.resource_id,wm_concat(t.ip) ip from resource_ip t  group by t.resource_id)
ip          WHERE A.SLACCT_ID = B.SLACCT_ID and
t.resource_id=ip.resource_id     AND A.RESOURCE_ID = T.RESOURCE_ID           AND
T.RES_GRP_ID = C.RES_GRP_ID          AND B.PRACCT_ID = :1            AND
(B.CANCEL_TIME is null or B.CANCEL_TIME>SYSDATE )   and    A.STATE !=
2   ORDER BY A.SLACCT_ID    ASC  ) r   where ROWNUM < :2)     where rn >= :3
```

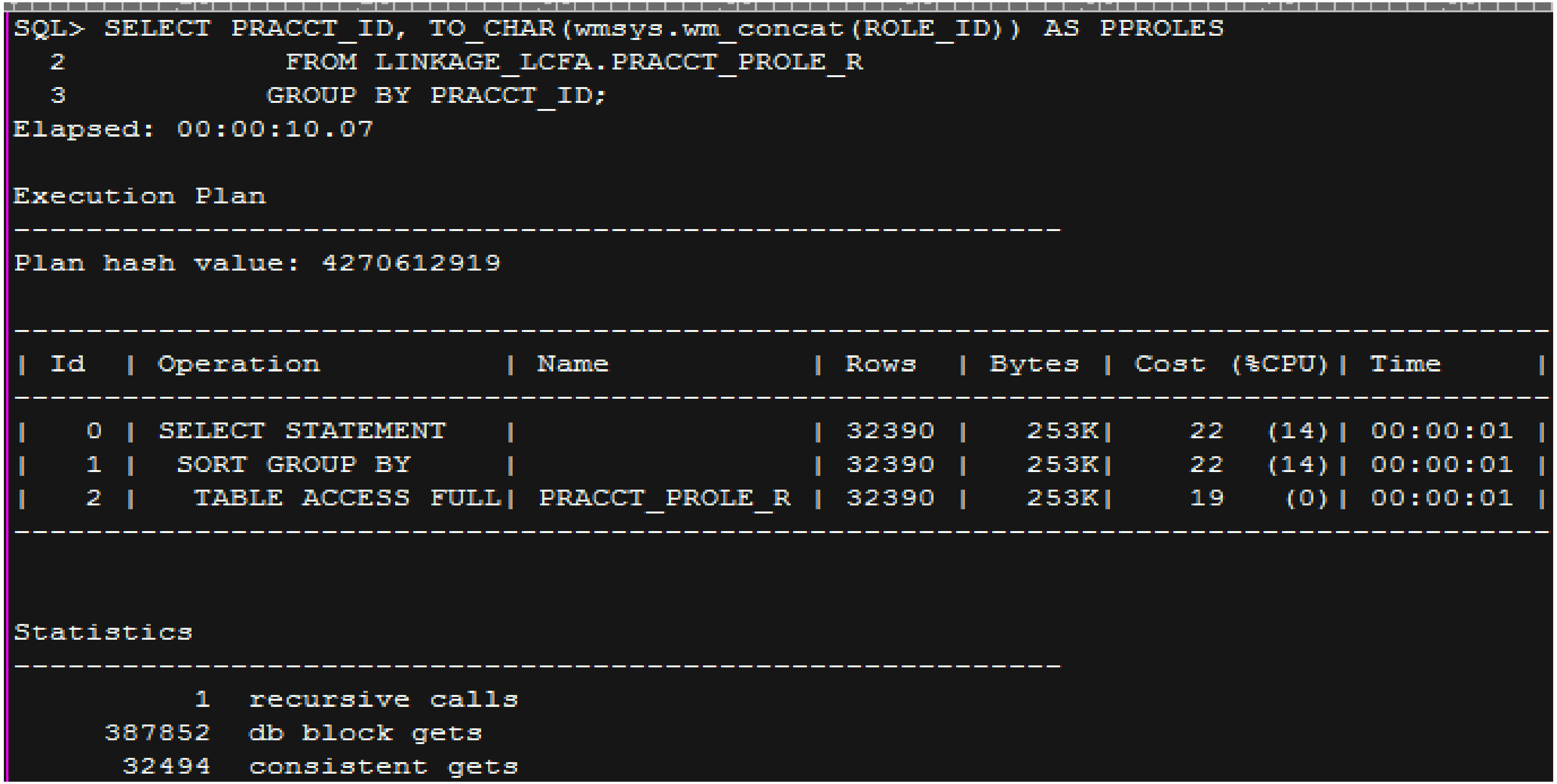


```
-----------------------------------------------------------------------------------------------------------------------
| Id  | Operation                                  | Name                      | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                           |                           |       |       |    11 (100)|          |
|*  1 |  VIEW                                      |                           |     1 | 17389 |    11  (10)| 00:00:01 |
|*  2 |   COUNT STOPKEY                            |                           |       |       |            |          |
|   3 |    VIEW                                    |                           |     1 | 17376 |    11  (10)| 00:00:01 |
|*  4 |     SORT ORDER BY STOPKEY                  |                           |     1 |  2370 |    11  (10)| 00:00:01 |
|   5 |      COUNT                                 |                           |       |       |            |          |
|   6 |       NESTED LOOPS                         |                           |     1 |  2370 |    10   (0)| 00:00:01 |
|   7 |        NESTED LOOPS                        |                           |     1 |   368 |     8   (0)| 00:00:01 |
|   8 |         NESTED LOOPS                       |                           |     1 |   352 |     7   (0)| 00:00:01 |
|   9 |          NESTED LOOPS                      |                           |     1 |   314 |     6   (0)| 00:00:01 |
|* 10 |           TABLE ACCESS BY INDEX ROWID| PRACCT_SLACCT_R           |     1 |    20 |     3   (0)| 00:00:01 |
|* 11 |            INDEX RANGE SCAN                | PRACCT_SLACCT_R_ID        |     3 |       |     1   (0)| 00:00:01 |
|* 12 |           TABLE ACCESS BY INDEX ROWID| SLAVE_ACCOUNT             |     1 |   294 |     3   (0)| 00:00:01 |
|* 13 |            INDEX RANGE SCAN                | SLAVE_ACCOUNT_ID          |     1 |       |     2   (0)| 00:00:01 |
|  14 |          TABLE ACCESS BY INDEX ROWID       | APP_RESOURCE              |     1 |    38 |     1   (0)| 00:00:01 |
|* 15 |           INDEX UNIQUE SCAN                | PK_APP_RESOURCE           |     1 |       |     0   (0)|          |
|  16 |         TABLE ACCESS BY INDEX ROWID        | RESOURCE_GROUP            |     1 |    16 |     1   (0)| 00:00:01 |
|* 17 |          INDEX UNIQUE SCAN                 | PK_RESOURCE_GROUP         |     1 |       |     0   (0)|          |
|  18 |        VIEW PUSHED PREDICATE               |                           |     1 |  2002 |     2   (0)| 00:00:01 |
|* 19 |         FILTER                             |                           |       |       |            |          |
|  20 |          SORT AGGREGATE                    |                           |     1 |    19 |            |          |
|  21 |           TABLE ACCESS BY INDEX ROWID| RESOURCE_IP               |     1 |    19 |     2   (0)| 00:00:01 |
|* 22 |            INDEX RANGE SCAN                | RESOURCE_IP_ID            |     1 |       |     1   (0)| 00:00:01 |
```

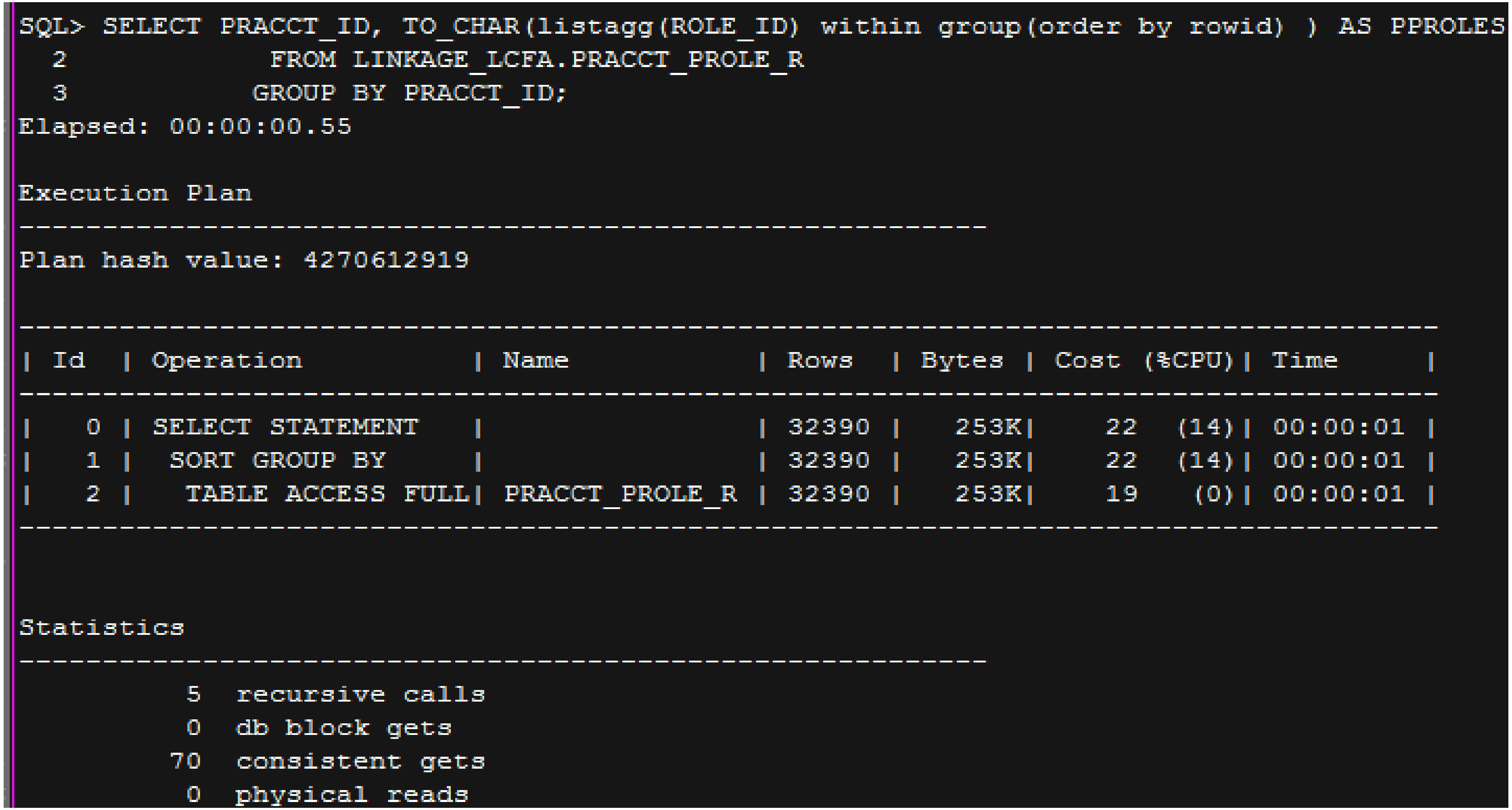


```
13 - "A".ROWID[ROWID,10], "A"."SLACCT_ID"[NUMBER,22]
14 - "T"."RESOURCE_ID"[NUMBER,22], "T"."RESOURCE_NAME"[VARCHAR2,80],
     "T"."CATEGORY"[VARCHAR2,40], "T"."RES_GRP_ID"[NUMBER,22]
15 - "T".ROWID[ROWID,10], "T"."RESOURCE_ID"[NUMBER,22]
16 - "C"."RES_GRP_NAME"[VARCHAR2,60]
17 - "C".ROWID[ROWID,10]
18 - "IP"."IP"[LOB,4000]
19 - unknown-uag()[4000]
20 - (#keys=0) COUNT(*)[22], unknown-uag()[4000]
21 - "T"."IP"[VARCHAR2,64]
22 - "T".ROWID[ROWID,10]
```

该函数在Oracle官方的文档中是:**function is used internally and for this reason it is UN-documented,11g listagg使用比wmsys.wm_concat效率高很多,wmsys.wm_concat是undocument函数，有很多不确定因素，不建议使用,而且12c已经删除了wmsys.wm_concat,建议11g库对wmsys.wm_concat最好修改为listagg。**



```
SQL> SELECT PRACCT_ID, TO_CHAR(wmsys.wm_concat(ROLE_ID)) AS PPROLES
  2            FROM LINKAGE_LCFA.PRACCT_PROLE_R
  3           GROUP BY PRACCT_ID;
Elapsed: 00:00:10.07

Execution Plan
----------------------------------------------------------
Plan hash value: 4270612919

-------------------------------------------------------------------------------------
| Id  | Operation          | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |                | 32390 |   253K|    22  (14)| 00:00:01 |
|   1 |  SORT GROUP BY     |                | 32390 |   253K|    22  (14)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| PRACCT_PROLE_R | 32390 |   253K|    19   (0)| 00:00:01 |
-------------------------------------------------------------------------------------


Statistics
----------------------------------------------------------
          1  recursive calls
     387852  db block gets
      32494  consistent gets
```



```
SQL> SELECT PRACCT_ID, TO_CHAR(listagg(ROLE_ID) within group(order by rowid) ) AS PPROLES
  2              FROM LINKAGE_LCFA.PRACCT_PROLE_R
  3             GROUP BY PRACCT_ID;
Elapsed: 00:00:00.55

Execution Plan
----------------------------------------------------------
Plan hash value: 4270612919


-------------------------------------------------------------------------------------
| Id  | Operation          | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |                   | 32390 |   253K|    22  (14)| 00:00:01 |
|   1 |  SORT GROUP BY     |                   | 32390 |   253K|    22  (14)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| PRACCT_PROLE_R    | 32390 |   253K|    19   (0)| 00:00:01 |
-------------------------------------------------------------------------------------



Statistics
----------------------------------------------------------
          5  recursive calls
          0  db block gets
         70  consistent gets
          0  physical reads
```

总结

Sql优化的核心就是减少IO操作，这里的IO不单指物理IO

特殊执行计划和高级执行计划