DTCC

2013中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2013

大数据 数据库架构与优化 数据治理与分析

SequeMedia 盛拓传媒 IT168.com ITPUB ChinaUnix

基于Oracle的SQL优化典型案例分析

崔华（dbsnake）

@dbsnake

关于我

- 中航信工程师
- Oracle ACE
- ACOUG成员

中国航信 TravelSky

ORACLE ACE

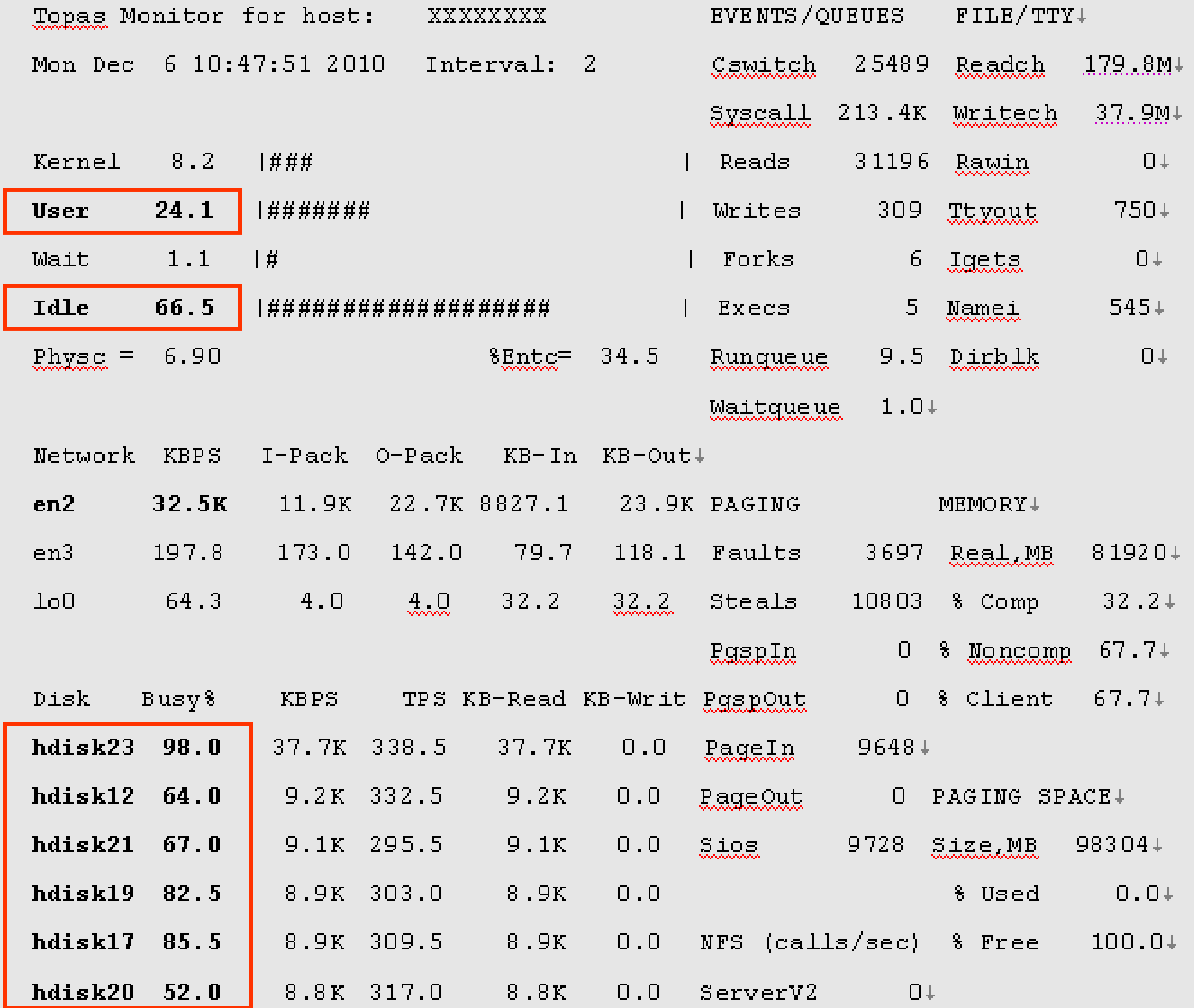
ACOUG All China Oracle User Group MEMBER

Oracle里SQL优化的方法论

- Oracle里的SQL优化实际上是基于对CBO和执行计划的深刻理解
- Oracle里的SQL优化不能脱离实际的业务
- Oracle里SQL优化需要适时使用绑定变量

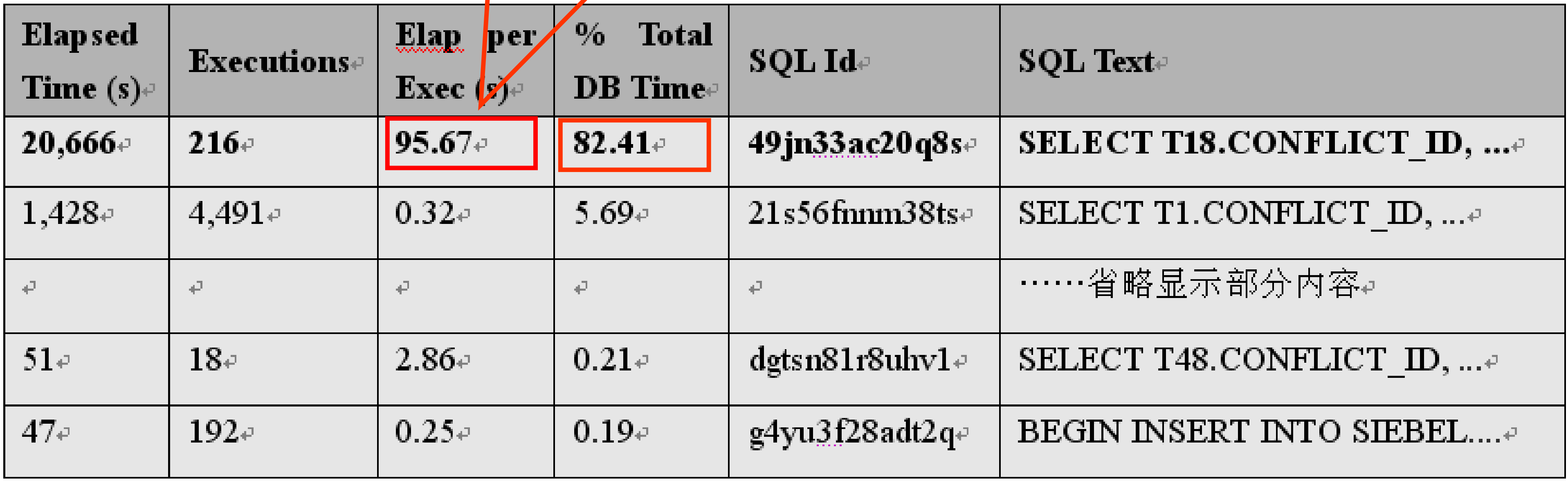
优化器模式对计算成本带来巨大影响的实例

- CBO认为全表扫描一个700多万数据量的大表的成本值仅为2

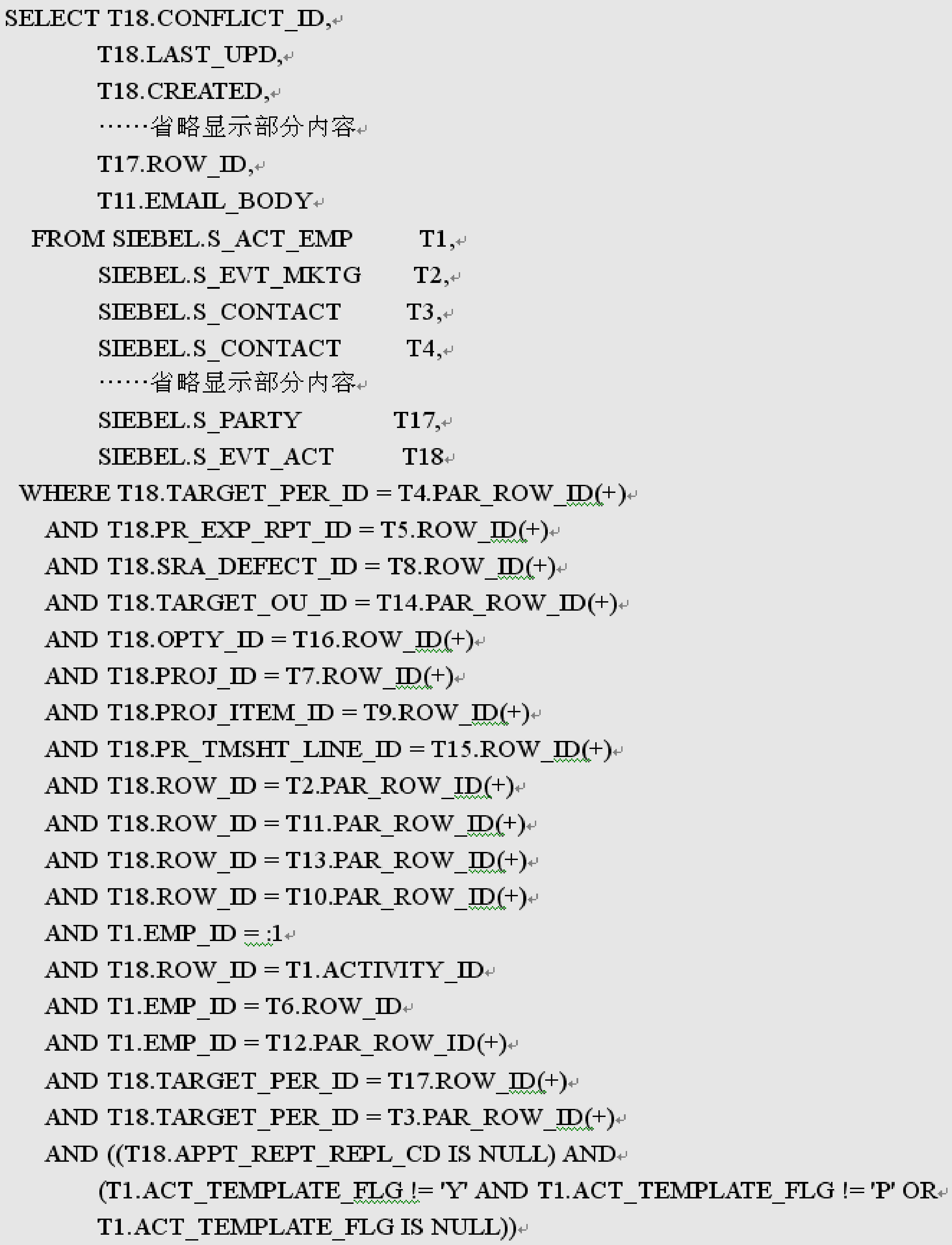


```
Topas Monitor for host:    XXXXXXXX          EVENTS/QUEUES    FILE/TTY↓
Mon Dec  6 10:47:51 2010   Interval:  2      Cswitch   25489  Readch   179.8M↓
                                             Syscall  213.4K  Writech   37.9M↓
Kernel    8.2   |###                        |  Reads   31196  Rawin        0↓
User     24.1   |#######                    |  Writes    309  Ttyout     750↓
Wait      1.1   |#                          |  Forks       6  Igets        0↓
Idle     66.5   |###################        |  Execs       5  Namei      545↓
Physc =  6.90                  %Entc=  34.5  Runqueue    9.5  Dirblk       0↓
                                             Waitqueue   1.0↓
Network  KBPS   I-Pack  O-Pack   KB-In  KB-Out↓
en2      32.5K   11.9K   22.7K 8827.1   23.9K PAGING          MEMORY↓
en3      197.8   173.0   142.0    79.7  118.1 Faults    3697  Real,MB  81920↓
lo0       64.3     4.0     4.0    32.2   32.2 Steals   10803  % Comp    32.2↓
                                              PgspIn       0  % Noncomp 67.7↓
Disk     Busy%    KBPS     TPS KB-Read KB-Writ PgspOut     0  % Client  67.7↓
hdisk23   98.0   37.7K  338.5   37.7K    0.0  PageIn   9648↓
hdisk12   64.0    9.2K  332.5    9.2K    0.0  PageOut     0  PAGING SPACE↓
hdisk21   67.0    9.1K  295.5    9.1K    0.0  Sios     9728  Size,MB  98304↓
hdisk19   82.5    8.9K  303.0    8.9K    0.0                 % Used     0.0↓
hdisk17   85.5    8.9K  309.5    8.9K    0.0  NFS (calls/sec) % Free  100.0↓
hdisk20   52.0    8.8K  317.0    8.8K    0.0  ServerV2      0↓
```

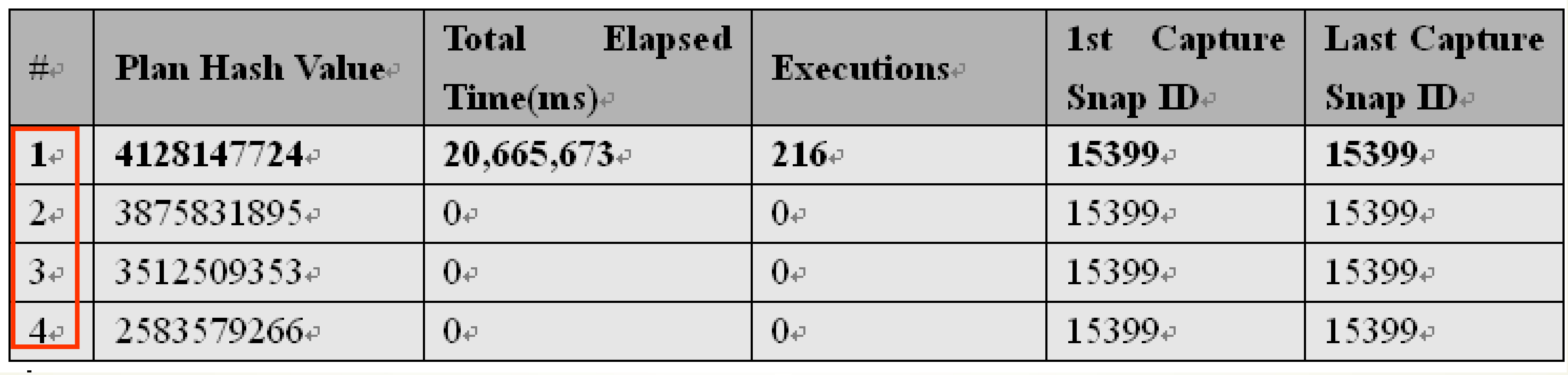
每次登陆的平均等待时间95.67秒



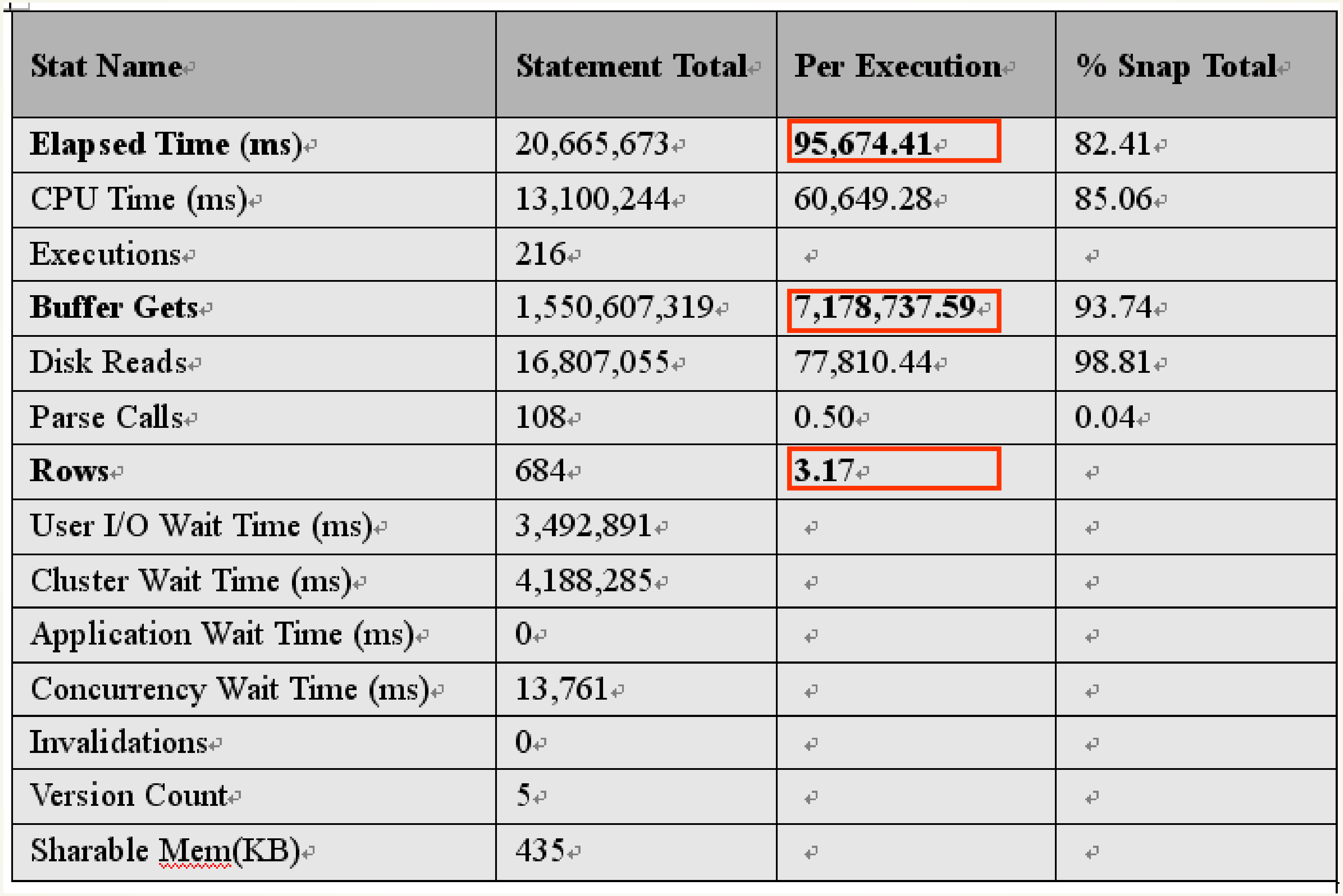
Elapsed Time (s)	Executions	Elap per Exec (s)	% Total DB Time	SQL Id	SQL Text
20,666	**216**	**95.67**	**82.41**	**49jn33ac20q8s**	**SELECT T18.CONFLICT_ID, ...**
1,428	4,491	0.32	5.69	21s56fnnm38ts	SELECT T1.CONFLICT_ID, ...
					……省略显示部分内容
51	18	2.86	0.21	dgtsn81r8uhv1	SELECT T48.CONFLICT_ID, ...
47	192	0.25	0.19	g4yu3f28adt2q	BEGIN INSERT INTO SIEBEL....



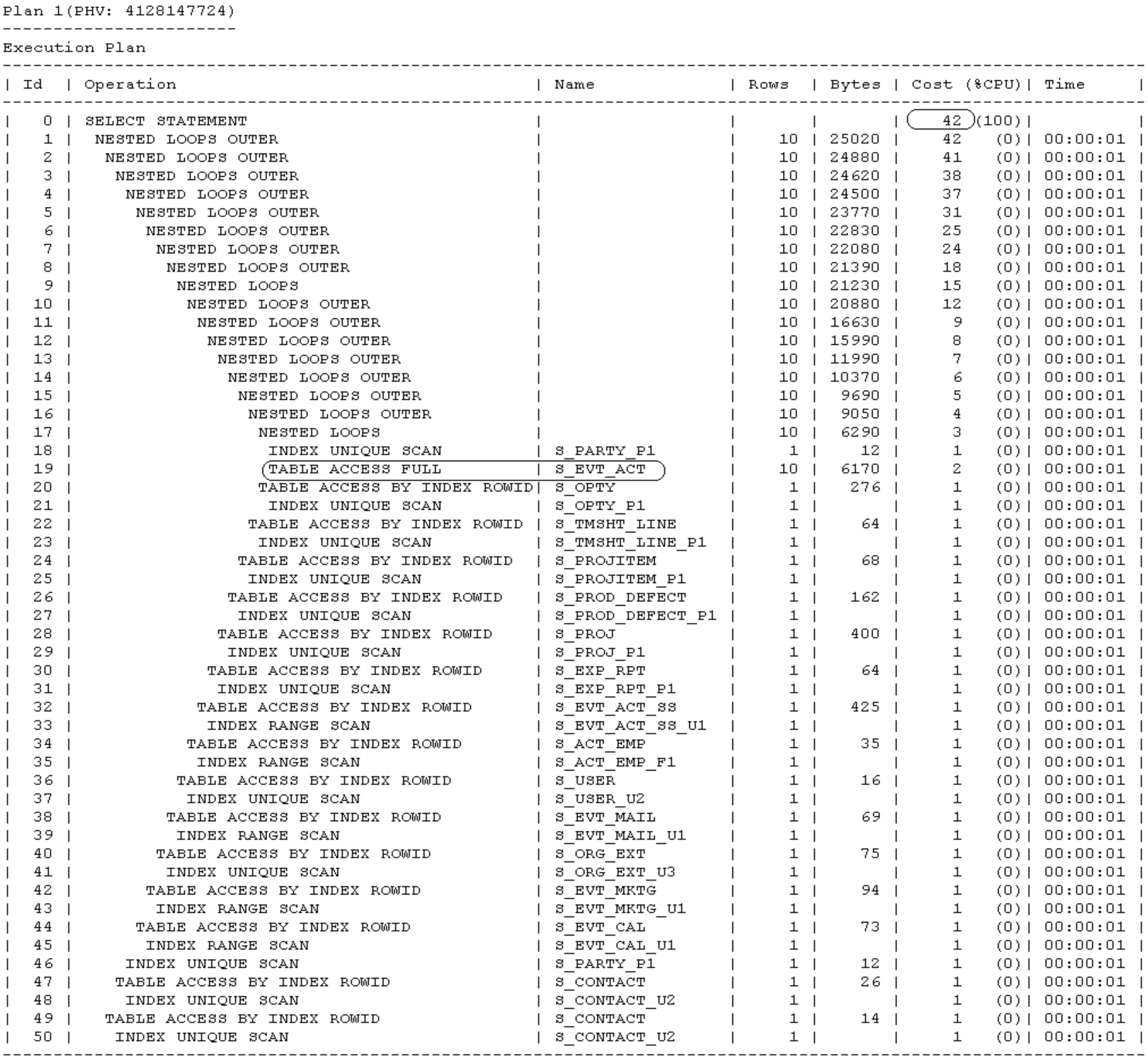
```
SELECT T18.CONFLICT_ID,
           T18.LAST_UPD,
           T18.CREATED,
           ……省略显示部分内容
           T17.ROW_ID,
           T11.EMAIL_BODY
  FROM SIEBEL.S_ACT_EMP         T1,
           SIEBEL.S_EVT_MKTG       T2,
           SIEBEL.S_CONTACT        T3,
           SIEBEL.S_CONTACT        T4,
           ……省略显示部分内容
           SIEBEL.S_PARTY           T17,
           SIEBEL.S_EVT_ACT         T18
 WHERE T18.TARGET_PER_ID = T4.PAR_ROW_ID(+)
    AND T18.PR_EXP_RPT_ID = T5.ROW_ID(+)
    AND T18.SRA_DEFECT_ID = T8.ROW_ID(+)
    AND T18.TARGET_OU_ID = T14.PAR_ROW_ID(+)
    AND T18.OPTY_ID = T16.ROW_ID(+)
    AND T18.PROJ_ID = T7.ROW_ID(+)
    AND T18.PROJ_ITEM_ID = T9.ROW_ID(+)
    AND T18.PR_TMSHT_LINE_ID = T15.ROW_ID(+)
    AND T18.ROW_ID = T2.PAR_ROW_ID(+)
    AND T18.ROW_ID = T11.PAR_ROW_ID(+)
    AND T18.ROW_ID = T13.PAR_ROW_ID(+)
    AND T18.ROW_ID = T10.PAR_ROW_ID(+)
    AND T1.EMP_ID = :1
    AND T18.ROW_ID = T1.ACTIVITY_ID
    AND T1.EMP_ID = T6.ROW_ID
    AND T1.EMP_ID = T12.PAR_ROW_ID(+)
    AND T18.TARGET_PER_ID = T17.ROW_ID(+)
    AND T18.TARGET_PER_ID = T3.PAR_ROW_ID(+)
    AND ((T18.APPT_REPT_REPL_CD IS NULL) AND
           (T1.ACT_TEMPLATE_FLG != 'Y' AND T1.ACT_TEMPLATE_FLG != 'P' OR
           T1.ACT_TEMPLATE_FLG IS NULL))
```



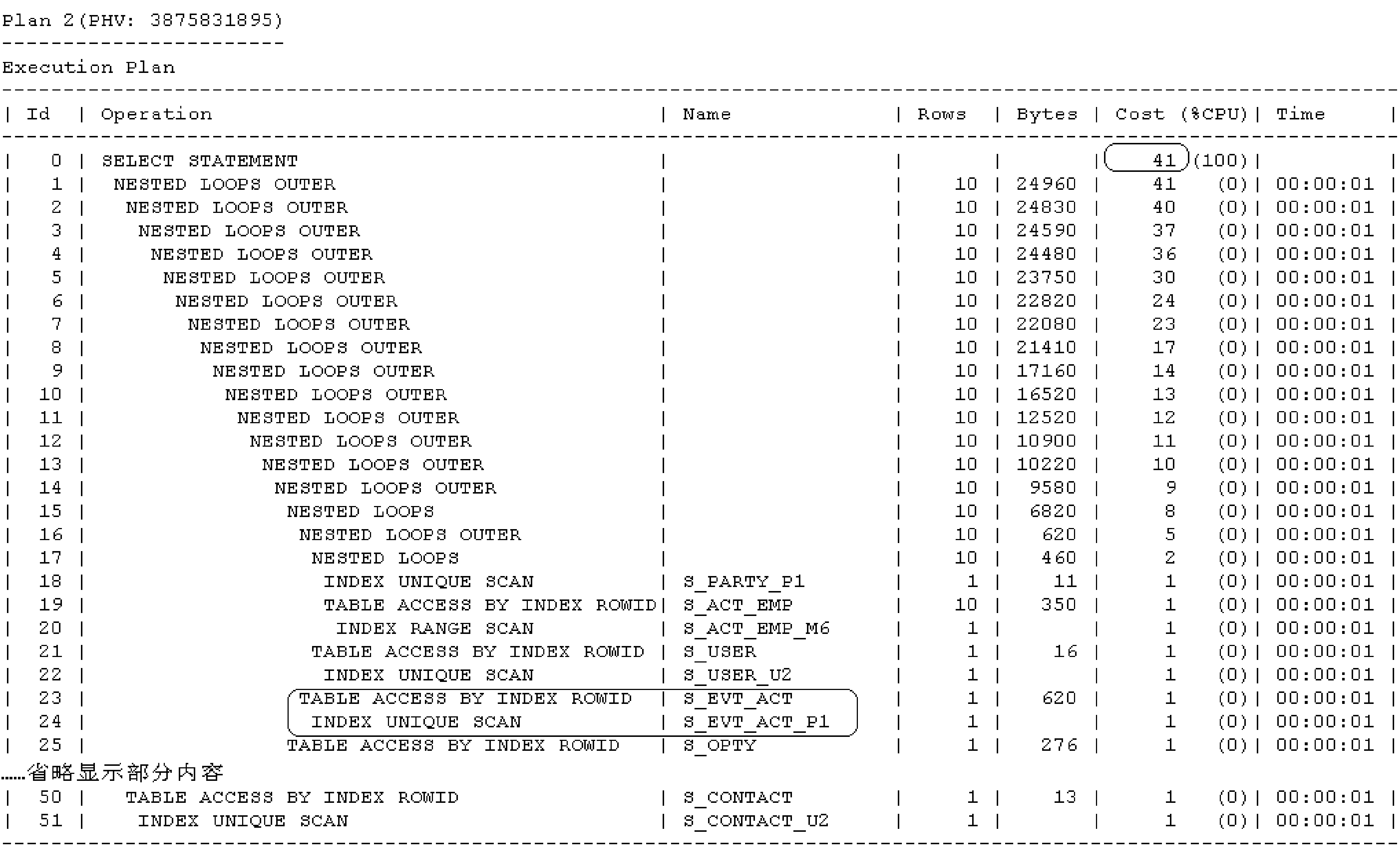
#	Plan Hash Value	Total Elapsed Time(ms)	Executions	1st Capture Snap ID	Last Capture Snap ID
1	**4128147724**	**20,665,673**	**216**	**15399**	**15399**
2	3875831895	0	0	15399	15399
3	3512509353	0	0	15399	15399
4	2583579266	0	0	15399	15399



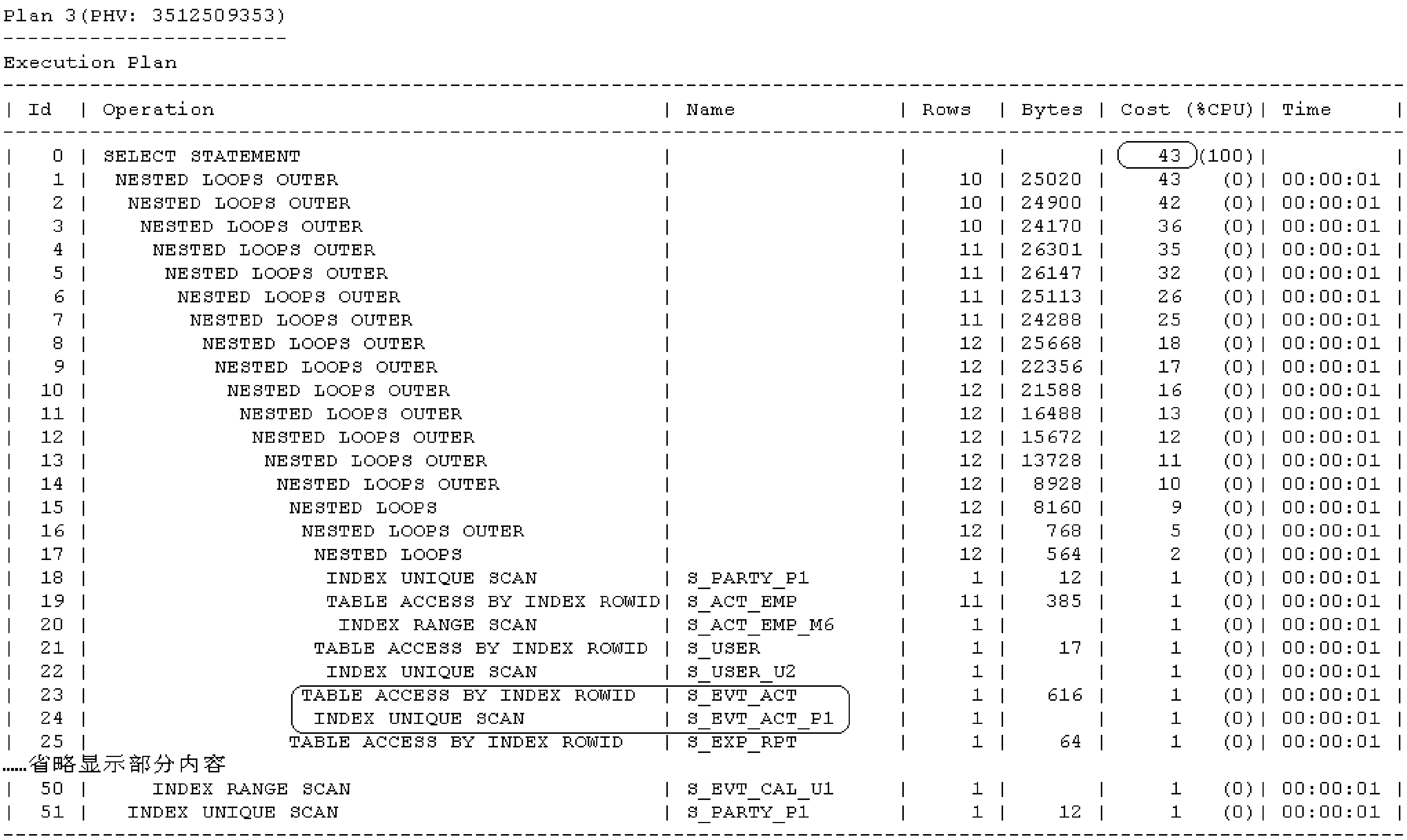
Stat Name	**Statement Total**	**Per Execution**	**% Snap Total**
Elapsed Time (ms)	20,665,673	**95,674.41**	82.41
CPU Time (ms)	13,100,244	60,649.28	85.06
Executions	216		
Buffer Gets	1,550,607,319	**7,178,737.59**	93.74
Disk Reads	16,807,055	77,810.44	98.81
Parse Calls	108	0.50	0.04
Rows	684	**3.17**	
User I/O Wait Time (ms)	3,492,891		
Cluster Wait Time (ms)	4,188,285		
Application Wait Time (ms)	0		
Concurrency Wait Time (ms)	13,761		
Invalidations	0		
Version Count	5		
Sharable Mem(KB)	435		



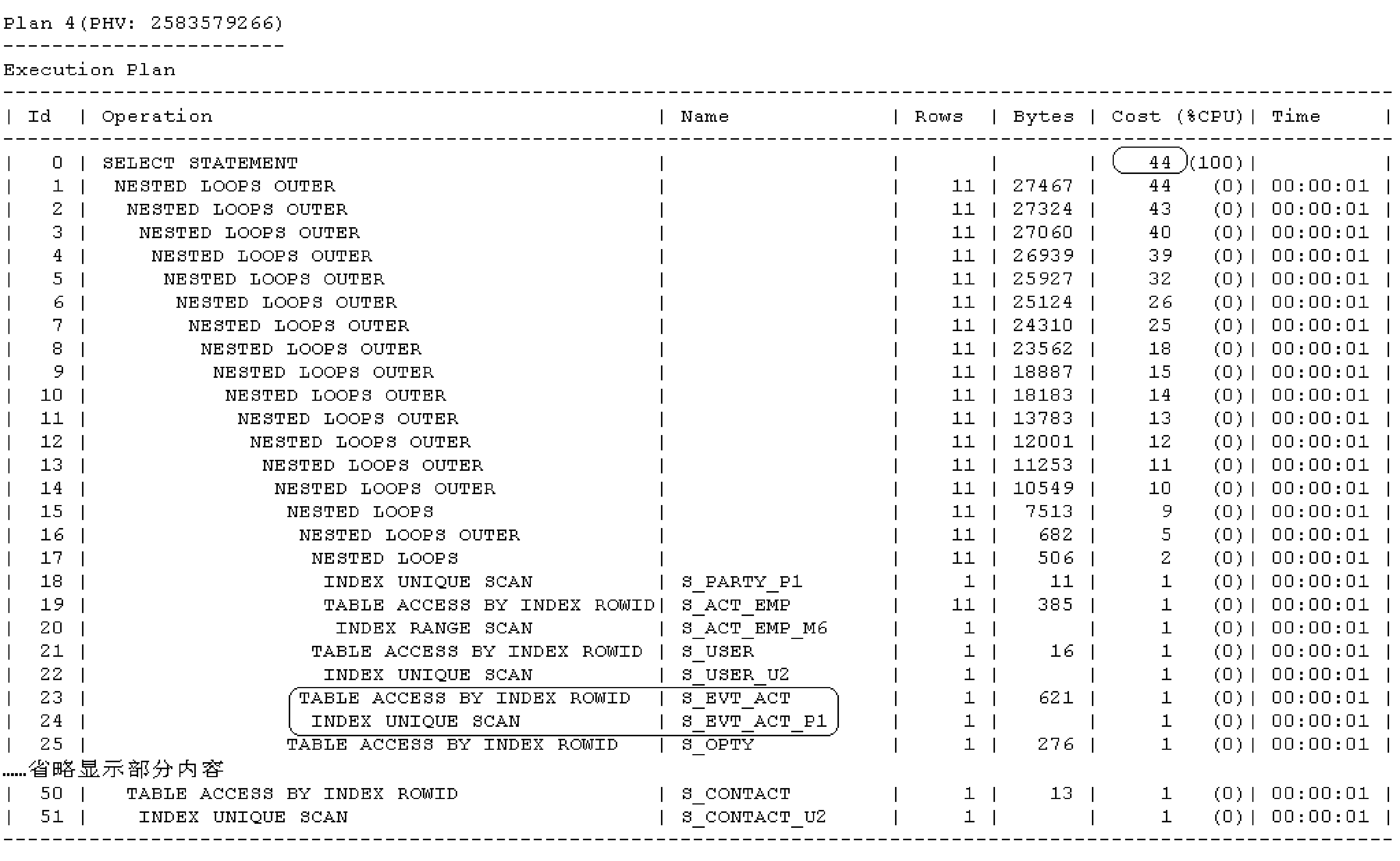
```
Plan 1(PHV: 4128147724)
-----------------------

Execution Plan
----------------------------------------------------------------------------------------------------------------
| Id  | Operation                                   | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                            |                   |       |       |    42 (100)|          |
|   1 |  NESTED LOOPS OUTER                         |                   |    10 | 25020 |    42   (0)| 00:00:01 |
|   2 |   NESTED LOOPS OUTER                        |                   |    10 | 24880 |    41   (0)| 00:00:01 |
|   3 |    NESTED LOOPS OUTER                       |                   |    10 | 24620 |    38   (0)| 00:00:01 |
|   4 |     NESTED LOOPS OUTER                      |                   |    10 | 24500 |    37   (0)| 00:00:01 |
|   5 |      NESTED LOOPS OUTER                     |                   |    10 | 23770 |    31   (0)| 00:00:01 |
|   6 |       NESTED LOOPS OUTER                    |                   |    10 | 22830 |    25   (0)| 00:00:01 |
|   7 |        NESTED LOOPS OUTER                   |                   |    10 | 22080 |    24   (0)| 00:00:01 |
|   8 |         NESTED LOOPS OUTER                  |                   |    10 | 21390 |    18   (0)| 00:00:01 |
|   9 |          NESTED LOOPS                       |                   |    10 | 21230 |    15   (0)| 00:00:01 |
|  10 |           NESTED LOOPS OUTER                |                   |    10 | 20880 |    12   (0)| 00:00:01 |
|  11 |            NESTED LOOPS OUTER               |                   |    10 | 16630 |     9   (0)| 00:00:01 |
|  12 |             NESTED LOOPS OUTER              |                   |    10 | 15990 |     8   (0)| 00:00:01 |
|  13 |              NESTED LOOPS OUTER             |                   |    10 | 11990 |     7   (0)| 00:00:01 |
|  14 |               NESTED LOOPS OUTER            |                   |    10 | 10370 |     6   (0)| 00:00:01 |
|  15 |                NESTED LOOPS OUTER           |                   |    10 |  9690 |     5   (0)| 00:00:01 |
|  16 |                 NESTED LOOPS OUTER          |                   |    10 |  9050 |     4   (0)| 00:00:01 |
|  17 |                  NESTED LOOPS               |                   |    10 |  6290 |     3   (0)| 00:00:01 |
|  18 |                   INDEX UNIQUE SCAN         | S_PARTY_P1        |     1 |    12 |     1   (0)| 00:00:01 |
|  19 |                   TABLE ACCESS FULL         | S_EVT_ACT         |    10 |  6170 |     2   (0)| 00:00:01 |
|  20 |                  TABLE ACCESS BY INDEX ROWID| S_OPTY            |     1 |   276 |     1   (0)| 00:00:01 |
|  21 |                   INDEX UNIQUE SCAN         | S_OPTY_P1         |     1 |       |     1   (0)| 00:00:01 |
|  22 |                 TABLE ACCESS BY INDEX ROWID | S_TMSHT_LINE      |     1 |    64 |     1   (0)| 00:00:01 |
|  23 |                  INDEX UNIQUE SCAN          | S_TMSHT_LINE_P1   |     1 |       |     1   (0)| 00:00:01 |
|  24 |                TABLE ACCESS BY INDEX ROWID  | S_PROJITEM        |     1 |    68 |     1   (0)| 00:00:01 |
|  25 |                 INDEX UNIQUE SCAN           | S_PROJITEM_P1     |     1 |       |     1   (0)| 00:00:01 |
|  26 |               TABLE ACCESS BY INDEX ROWID   | S_PROD_DEFECT     |     1 |   162 |     1   (0)| 00:00:01 |
|  27 |                INDEX UNIQUE SCAN            | S_PROD_DEFECT_P1  |     1 |       |     1   (0)| 00:00:01 |
|  28 |              TABLE ACCESS BY INDEX ROWID    | S_PROJ            |     1 |   400 |     1   (0)| 00:00:01 |
|  29 |               INDEX UNIQUE SCAN             | S_PROJ_P1         |     1 |       |     1   (0)| 00:00:01 |
|  30 |             TABLE ACCESS BY INDEX ROWID     | S_EXP_RPT         |     1 |    64 |     1   (0)| 00:00:01 |
|  31 |              INDEX UNIQUE SCAN              | S_EXP_RPT_P1      |     1 |       |     1   (0)| 00:00:01 |
|  32 |            TABLE ACCESS BY INDEX ROWID      | S_EVT_ACT_SS      |     1 |   425 |     1   (0)| 00:00:01 |
|  33 |             INDEX RANGE SCAN                | S_EVT_ACT_SS_U1   |     1 |       |     1   (0)| 00:00:01 |
|  34 |           TABLE ACCESS BY INDEX ROWID       | S_ACT_EMP         |     1 |    35 |     1   (0)| 00:00:01 |
|  35 |            INDEX RANGE SCAN                 | S_ACT_EMP_F1      |     1 |       |     1   (0)| 00:00:01 |
|  36 |          TABLE ACCESS BY INDEX ROWID        | S_USER            |     1 |    16 |     1   (0)| 00:00:01 |
|  37 |           INDEX UNIQUE SCAN                 | S_USER_U2         |     1 |       |     1   (0)| 00:00:01 |
|  38 |         TABLE ACCESS BY INDEX ROWID         | S_EVT_MAIL        |     1 |    69 |     1   (0)| 00:00:01 |
|  39 |          INDEX RANGE SCAN                   | S_EVT_MAIL_U1     |     1 |       |     1   (0)| 00:00:01 |
|  40 |        TABLE ACCESS BY INDEX ROWID          | S_ORG_EXT         |     1 |    75 |     1   (0)| 00:00:01 |
|  41 |         INDEX UNIQUE SCAN                   | S_ORG_EXT_U3      |     1 |       |     1   (0)| 00:00:01 |
|  42 |       TABLE ACCESS BY INDEX ROWID           | S_EVT_MKTG        |     1 |    94 |     1   (0)| 00:00:01 |
|  43 |        INDEX RANGE SCAN                     | S_EVT_MKTG_U1     |     1 |       |     1   (0)| 00:00:01 |
|  44 |      TABLE ACCESS BY INDEX ROWID            | S_EVT_CAL         |     1 |    73 |     1   (0)| 00:00:01 |
|  45 |       INDEX RANGE SCAN                      | S_EVT_CAL_U1      |     1 |       |     1   (0)| 00:00:01 |
|  46 |     INDEX UNIQUE SCAN                       | S_PARTY_P1        |     1 |    12 |     1   (0)| 00:00:01 |
|  47 |    TABLE ACCESS BY INDEX ROWID              | S_CONTACT         |     1 |    26 |     1   (0)| 00:00:01 |
|  48 |     INDEX UNIQUE SCAN                       | S_CONTACT_U2      |     1 |       |     1   (0)| 00:00:01 |
|  49 |   TABLE ACCESS BY INDEX ROWID               | S_CONTACT         |     1 |    14 |     1   (0)| 00:00:01 |
|  50 |    INDEX UNIQUE SCAN                        | S_CONTACT_U2      |     1 |       |     1   (0)| 00:00:01 |
----------------------------------------------------------------------------------------------------------------
```



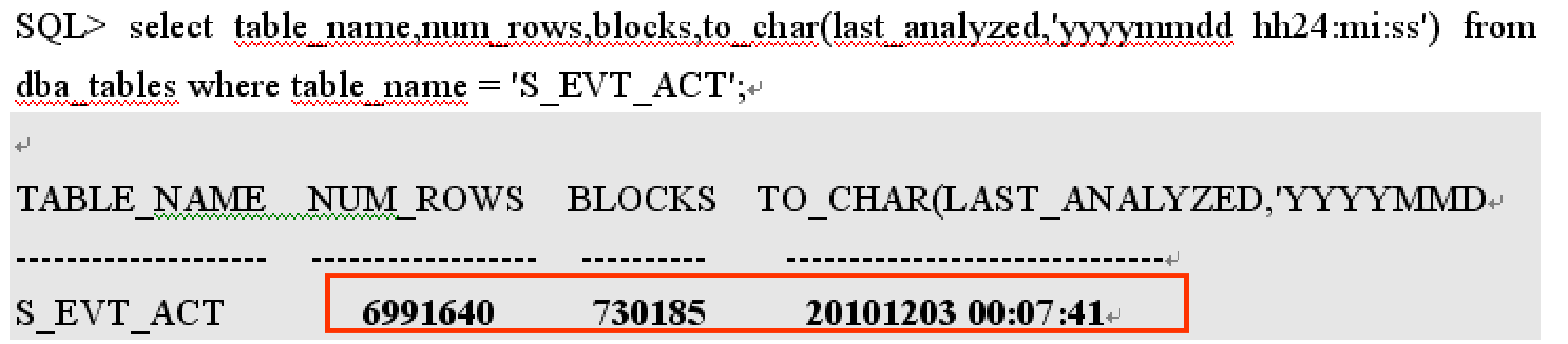
```
Plan 2(PHV: 3875831895)
-----------------------
Execution Plan
-------------------------------------------------------------------------------------------------------------
| Id  | Operation                                  | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                           |                    |       |       |    41 (100)|          |
|   1 |  NESTED LOOPS OUTER                        |                    |    10 | 24960 |    41   (0)| 00:00:01 |
|   2 |   NESTED LOOPS OUTER                       |                    |    10 | 24830 |    40   (0)| 00:00:01 |
|   3 |    NESTED LOOPS OUTER                      |                    |    10 | 24590 |    37   (0)| 00:00:01 |
|   4 |     NESTED LOOPS OUTER                     |                    |    10 | 24480 |    36   (0)| 00:00:01 |
|   5 |      NESTED LOOPS OUTER                    |                    |    10 | 23750 |    30   (0)| 00:00:01 |
|   6 |       NESTED LOOPS OUTER                   |                    |    10 | 22820 |    24   (0)| 00:00:01 |
|   7 |        NESTED LOOPS OUTER                  |                    |    10 | 22080 |    23   (0)| 00:00:01 |
|   8 |         NESTED LOOPS OUTER                 |                    |    10 | 21410 |    17   (0)| 00:00:01 |
|   9 |          NESTED LOOPS OUTER                |                    |    10 | 17160 |    14   (0)| 00:00:01 |
|  10 |           NESTED LOOPS OUTER               |                    |    10 | 16520 |    13   (0)| 00:00:01 |
|  11 |            NESTED LOOPS OUTER              |                    |    10 | 12520 |    12   (0)| 00:00:01 |
|  12 |             NESTED LOOPS OUTER             |                    |    10 | 10900 |    11   (0)| 00:00:01 |
|  13 |              NESTED LOOPS OUTER            |                    |    10 | 10220 |    10   (0)| 00:00:01 |
|  14 |               NESTED LOOPS OUTER           |                    |    10 |  9580 |     9   (0)| 00:00:01 |
|  15 |                NESTED LOOPS                |                    |    10 |  6820 |     8   (0)| 00:00:01 |
|  16 |                 NESTED LOOPS OUTER         |                    |    10 |   620 |     5   (0)| 00:00:01 |
|  17 |                  NESTED LOOPS              |                    |    10 |   460 |     2   (0)| 00:00:01 |
|  18 |                   INDEX UNIQUE SCAN        | S_PARTY_P1         |     1 |    11 |     1   (0)| 00:00:01 |
|  19 |                   TABLE ACCESS BY INDEX ROWID| S_ACT_EMP        |    10 |   350 |     1   (0)| 00:00:01 |
|  20 |                    INDEX RANGE SCAN        | S_ACT_EMP_M6       |     1 |       |     1   (0)| 00:00:01 |
|  21 |                  TABLE ACCESS BY INDEX ROWID | S_USER           |     1 |    16 |     1   (0)| 00:00:01 |
|  22 |                   INDEX UNIQUE SCAN        | S_USER_U2          |     1 |       |     1   (0)| 00:00:01 |
|  23 |                 TABLE ACCESS BY INDEX ROWID | S_EVT_ACT         |     1 |   620 |     1   (0)| 00:00:01 |
|  24 |                  INDEX UNIQUE SCAN         | S_EVT_ACT_P1       |     1 |       |     1   (0)| 00:00:01 |
|  25 |                TABLE ACCESS BY INDEX ROWID | S_OPTY             |     1 |   276 |     1   (0)| 00:00:01 |
......省略显示部分内容
|  50 |   TABLE ACCESS BY INDEX ROWID              | S_CONTACT          |     1 |    13 |     1   (0)| 00:00:01 |
|  51 |    INDEX UNIQUE SCAN                       | S_CONTACT_U2       |     1 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------
```



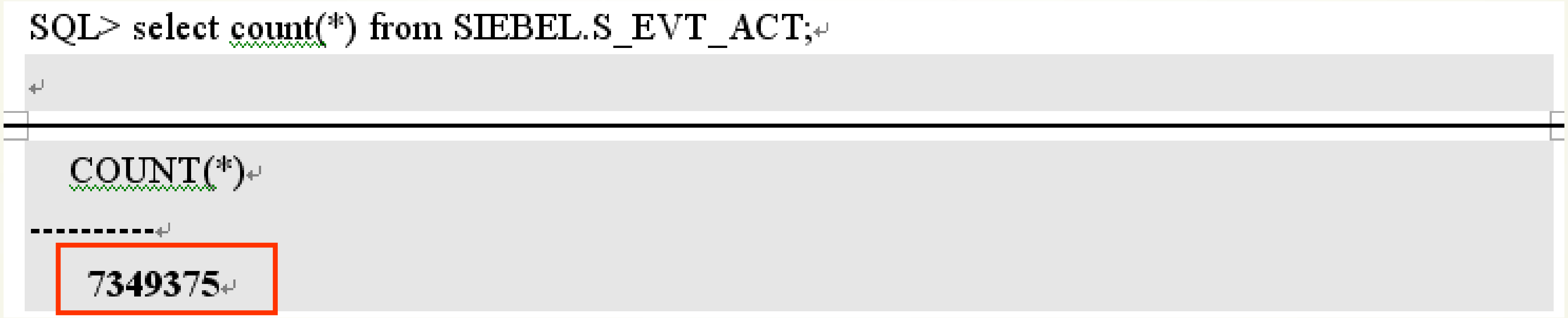
```
Plan 3(PHV: 3512509353)
-----------------------
Execution Plan
---------------------------------------------------------------------------------------------------------------
| Id  | Operation                                    | Name             |  Rows | Bytes  | Cost (%CPU)| Time     |
---------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                             |                  |       |        |    43 (100)|          |
|   1 |  NESTED LOOPS OUTER                          |                  |    10 |  25020 |    43   (0)| 00:00:01 |
|   2 |   NESTED LOOPS OUTER                         |                  |    10 |  24900 |    42   (0)| 00:00:01 |
|   3 |    NESTED LOOPS OUTER                        |                  |    10 |  24170 |    36   (0)| 00:00:01 |
|   4 |     NESTED LOOPS OUTER                       |                  |    11 |  26301 |    35   (0)| 00:00:01 |
|   5 |      NESTED LOOPS OUTER                      |                  |    11 |  26147 |    32   (0)| 00:00:01 |
|   6 |       NESTED LOOPS OUTER                     |                  |    11 |  25113 |    26   (0)| 00:00:01 |
|   7 |        NESTED LOOPS OUTER                    |                  |    11 |  24288 |    25   (0)| 00:00:01 |
|   8 |         NESTED LOOPS OUTER                   |                  |    12 |  25668 |    18   (0)| 00:00:01 |
|   9 |          NESTED LOOPS OUTER                  |                  |    12 |  22356 |    17   (0)| 00:00:01 |
|  10 |           NESTED LOOPS OUTER                 |                  |    12 |  21588 |    16   (0)| 00:00:01 |
|  11 |            NESTED LOOPS OUTER                |                  |    12 |  16488 |    13   (0)| 00:00:01 |
|  12 |             NESTED LOOPS OUTER               |                  |    12 |  15672 |    12   (0)| 00:00:01 |
|  13 |              NESTED LOOPS OUTER              |                  |    12 |  13728 |    11   (0)| 00:00:01 |
|  14 |               NESTED LOOPS OUTER             |                  |    12 |   8928 |    10   (0)| 00:00:01 |
|  15 |                NESTED LOOPS                  |                  |    12 |   8160 |     9   (0)| 00:00:01 |
|  16 |                 NESTED LOOPS OUTER           |                  |    12 |    768 |     5   (0)| 00:00:01 |
|  17 |                  NESTED LOOPS                |                  |    12 |    564 |     2   (0)| 00:00:01 |
|  18 |                   INDEX UNIQUE SCAN          | S_PARTY_P1       |     1 |     12 |     1   (0)| 00:00:01 |
|  19 |                   TABLE ACCESS BY INDEX ROWID| S_ACT_EMP        |    11 |    385 |     1   (0)| 00:00:01 |
|  20 |                    INDEX RANGE SCAN          | S_ACT_EMP_M6     |     1 |        |     1   (0)| 00:00:01 |
|  21 |                  TABLE ACCESS BY INDEX ROWID | S_USER           |     1 |     17 |     1   (0)| 00:00:01 |
|  22 |                   INDEX UNIQUE SCAN          | S_USER_U2        |     1 |        |     1   (0)| 00:00:01 |
|  23 |                 TABLE ACCESS BY INDEX ROWID  | S_EVT_ACT        |     1 |    616 |     1   (0)| 00:00:01 |
|  24 |                  INDEX UNIQUE SCAN           | S_EVT_ACT_P1     |     1 |        |     1   (0)| 00:00:01 |
|  25 |                TABLE ACCESS BY INDEX ROWID   | S_EXP_RPT        |     1 |     64 |     1   (0)| 00:00:01 |
……省略显示部分内容
|  50 |     INDEX RANGE SCAN                         | S_EVT_CAL_U1     |     1 |        |     1   (0)| 00:00:01 |
|  51 |   INDEX UNIQUE SCAN                          | S_PARTY_P1       |     1 |     12 |     1   (0)| 00:00:01 |
---------------------------------------------------------------------------------------------------------------
```



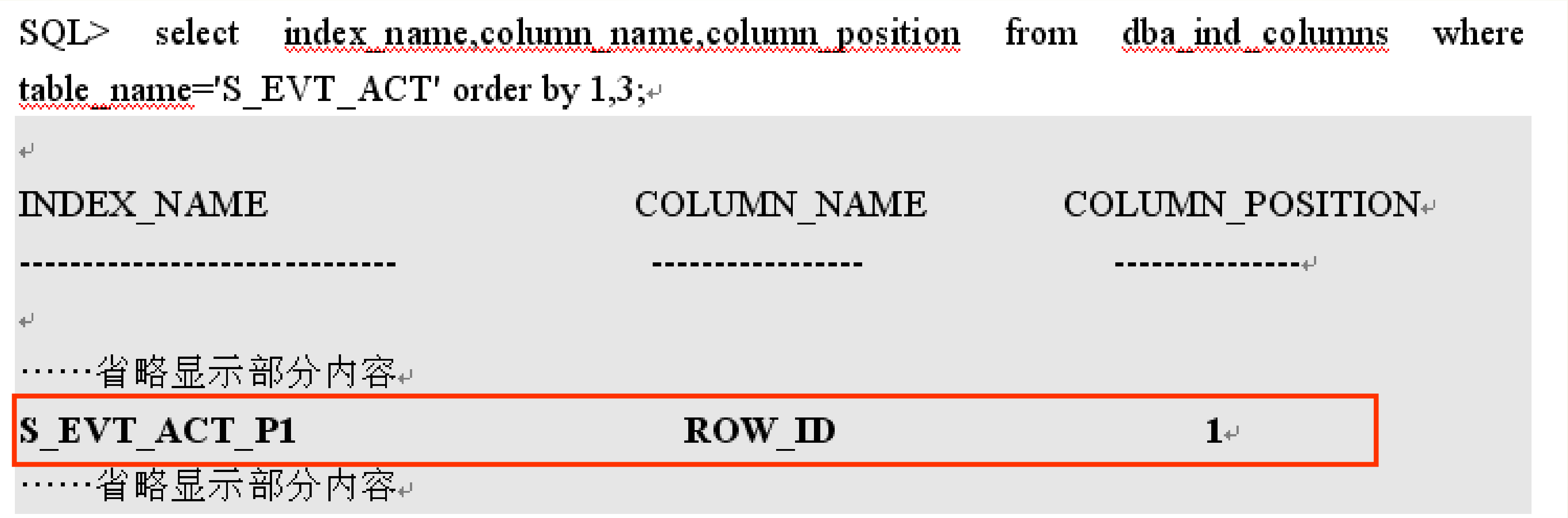
```
Plan 4(PHV: 2583579266)
-----------------------
Execution Plan
------------------------------------------------------------------------------------------------------------
| Id  | Operation                                     | Name                  | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                              |                       |       |       |    44(100)|          |
|   1 |  NESTED LOOPS OUTER                           |                       |    11 | 27467 |    44   (0)| 00:00:01 |
|   2 |   NESTED LOOPS OUTER                          |                       |    11 | 27324 |    43   (0)| 00:00:01 |
|   3 |    NESTED LOOPS OUTER                         |                       |    11 | 27060 |    40   (0)| 00:00:01 |
|   4 |     NESTED LOOPS OUTER                        |                       |    11 | 26939 |    39   (0)| 00:00:01 |
|   5 |      NESTED LOOPS OUTER                       |                       |    11 | 25927 |    32   (0)| 00:00:01 |
|   6 |       NESTED LOOPS OUTER                      |                       |    11 | 25124 |    26   (0)| 00:00:01 |
|   7 |        NESTED LOOPS OUTER                     |                       |    11 | 24310 |    25   (0)| 00:00:01 |
|   8 |         NESTED LOOPS OUTER                    |                       |    11 | 23562 |    18   (0)| 00:00:01 |
|   9 |          NESTED LOOPS OUTER                   |                       |    11 | 18887 |    15   (0)| 00:00:01 |
|  10 |           NESTED LOOPS OUTER                  |                       |    11 | 18183 |    14   (0)| 00:00:01 |
|  11 |            NESTED LOOPS OUTER                 |                       |    11 | 13783 |    13   (0)| 00:00:01 |
|  12 |             NESTED LOOPS OUTER                |                       |    11 | 12001 |    12   (0)| 00:00:01 |
|  13 |              NESTED LOOPS OUTER               |                       |    11 | 11253 |    11   (0)| 00:00:01 |
|  14 |               NESTED LOOPS OUTER              |                       |    11 | 10549 |    10   (0)| 00:00:01 |
|  15 |                NESTED LOOPS                   |                       |    11 |  7513 |     9   (0)| 00:00:01 |
|  16 |                 NESTED LOOPS OUTER            |                       |    11 |   682 |     5   (0)| 00:00:01 |
|  17 |                  NESTED LOOPS                 |                       |    11 |   506 |     2   (0)| 00:00:01 |
|  18 |                   INDEX UNIQUE SCAN           | S_PARTY_P1            |     1 |    11 |     1   (0)| 00:00:01 |
|  19 |                   TABLE ACCESS BY INDEX ROWID| S_ACT_EMP              |    11 |   385 |     1   (0)| 00:00:01 |
|  20 |                    INDEX RANGE SCAN           | S_ACT_EMP_M6          |     1 |       |     1   (0)| 00:00:01 |
|  21 |                  TABLE ACCESS BY INDEX ROWID  | S_USER                |     1 |    16 |     1   (0)| 00:00:01 |
|  22 |                   INDEX UNIQUE SCAN           | S_USER_U2             |     1 |       |     1   (0)| 00:00:01 |
|  23 |                 TABLE ACCESS BY INDEX ROWID   | S_EVT_ACT             |     1 |   621 |     1   (0)| 00:00:01 |
|  24 |                  INDEX UNIQUE SCAN            | S_EVT_ACT_P1          |     1 |       |     1   (0)| 00:00:01 |
|  25 |                TABLE ACCESS BY INDEX ROWID    | S_OPTY                |     1 |   276 |     1   (0)| 00:00:01 |
……省略显示部分内容
|  50 |   TABLE ACCESS BY INDEX ROWID                 | S_CONTACT             |     1 |    13 |     1   (0)| 00:00:01 |
|  51 |    INDEX UNIQUE SCAN                          | S_CONTACT_U2          |     1 |       |     1   (0)| 00:00:01 |
------------------------------------------------------------------------------------------------------------
```



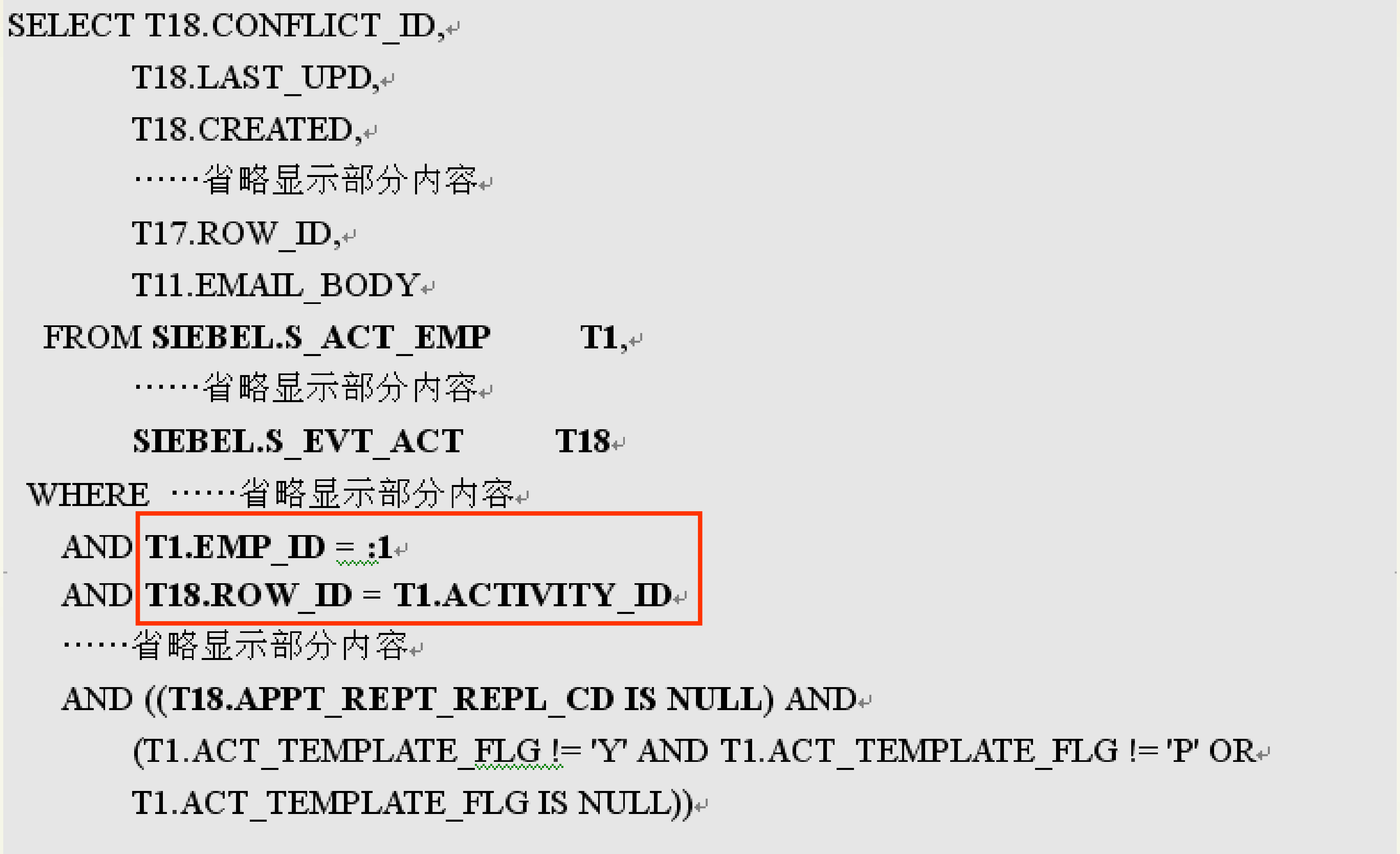
```
SQL>  select  table_name,num_rows,blocks,to_char(last_analyzed,'yyyymmdd  hh24:mi:ss')  from
dba_tables where table_name = 'S_EVT_ACT';↵
↵
TABLE_NAME    NUM_ROWS    BLOCKS    TO_CHAR(LAST_ANALYZED,'YYYYMMD↵
--------------------    ------------------    ----------      ------------------------------↵
S_EVT_ACT              6991640         730185         20101203 00:07:41↵
```



```
SQL> select count(*) from SIEBEL.S_EVT_ACT;↵
↵

   COUNT(*)↵
----------↵
   7349375↵
```



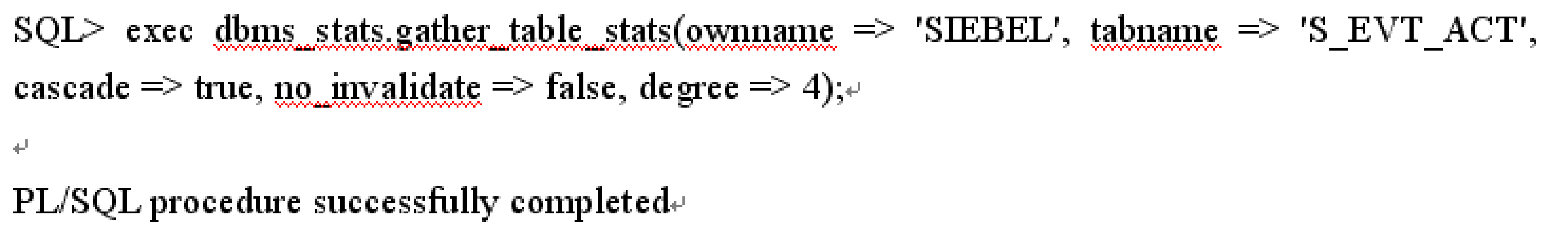
```
SQL>  select  index_name,column_name,column_position  from  dba_ind_columns  where
table_name='S_EVT_ACT' order by 1,3;↵

↵
INDEX_NAME                         COLUMN_NAME           COLUMN_POSITION↵
------------------------------     ----------------         ---------------↵

↵
……省略显示部分内容↵
S_EVT_ACT_P1                          ROW_ID                           1↵
……省略显示部分内容↵
```



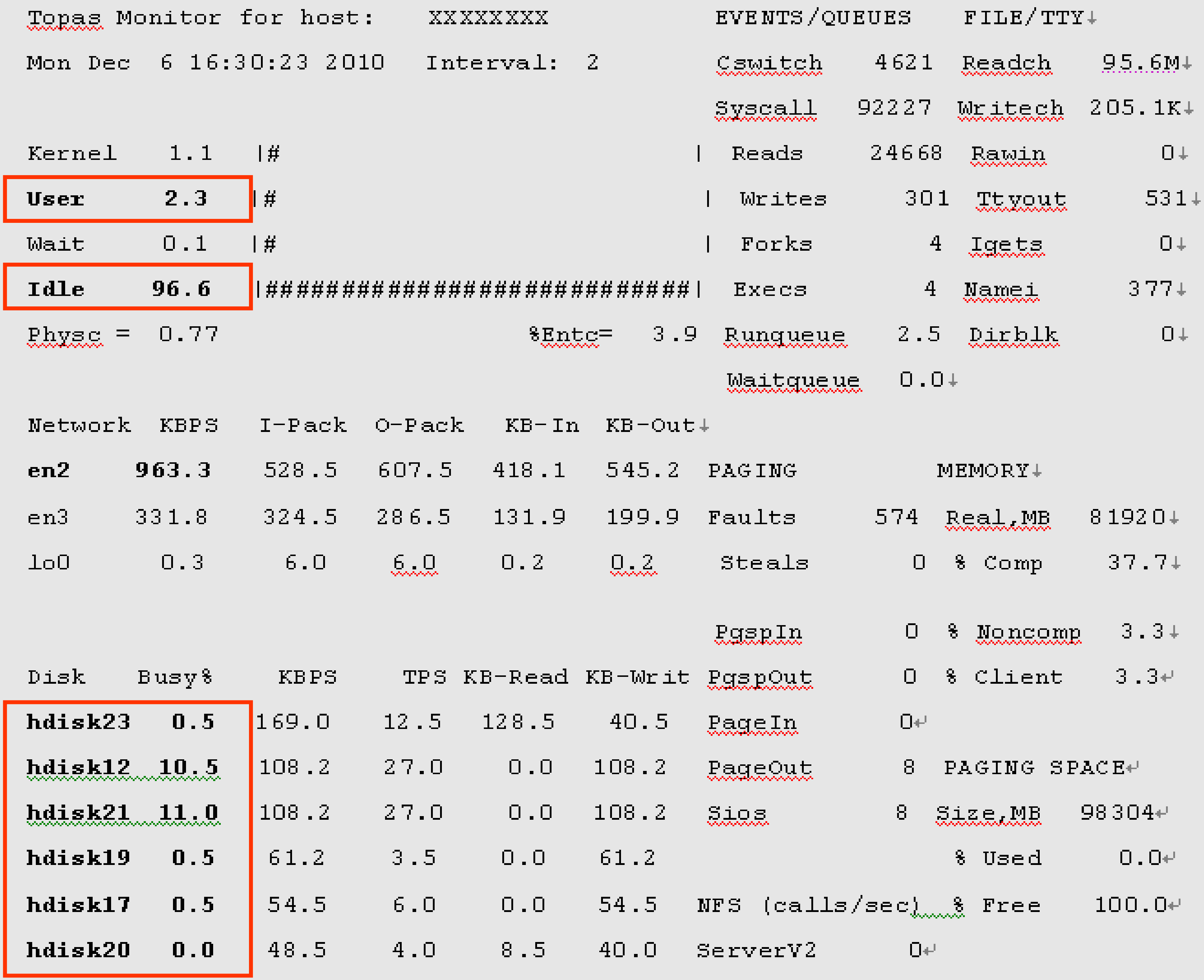
```
SELECT T18.CONFLICT_ID,
       T18.LAST_UPD,
       T18.CREATED,
       ……省略显示部分内容
       T17.ROW_ID,
       T11.EMAIL_BODY
  FROM SIEBEL.S_ACT_EMP        T1,
       ……省略显示部分内容
       SIEBEL.S_EVT_ACT        T18
 WHERE ……省略显示部分内容
   AND T1.EMP_ID = :1
   AND T18.ROW_ID = T1.ACTIVITY_ID
   ……省略显示部分内容
   AND ((T18.APPT_REPT_REPL_CD IS NULL) AND
       (T1.ACT_TEMPLATE_FLG != 'Y' AND T1.ACT_TEMPLATE_FLG != 'P' OR
       T1.ACT_TEMPLATE_FLG IS NULL))
```

初步分析

- 上述SQL好的和不好的执行计划所对应的成本值的过于接近就是导致上述坐席登陆慢的问题多次不间断出现的原因。



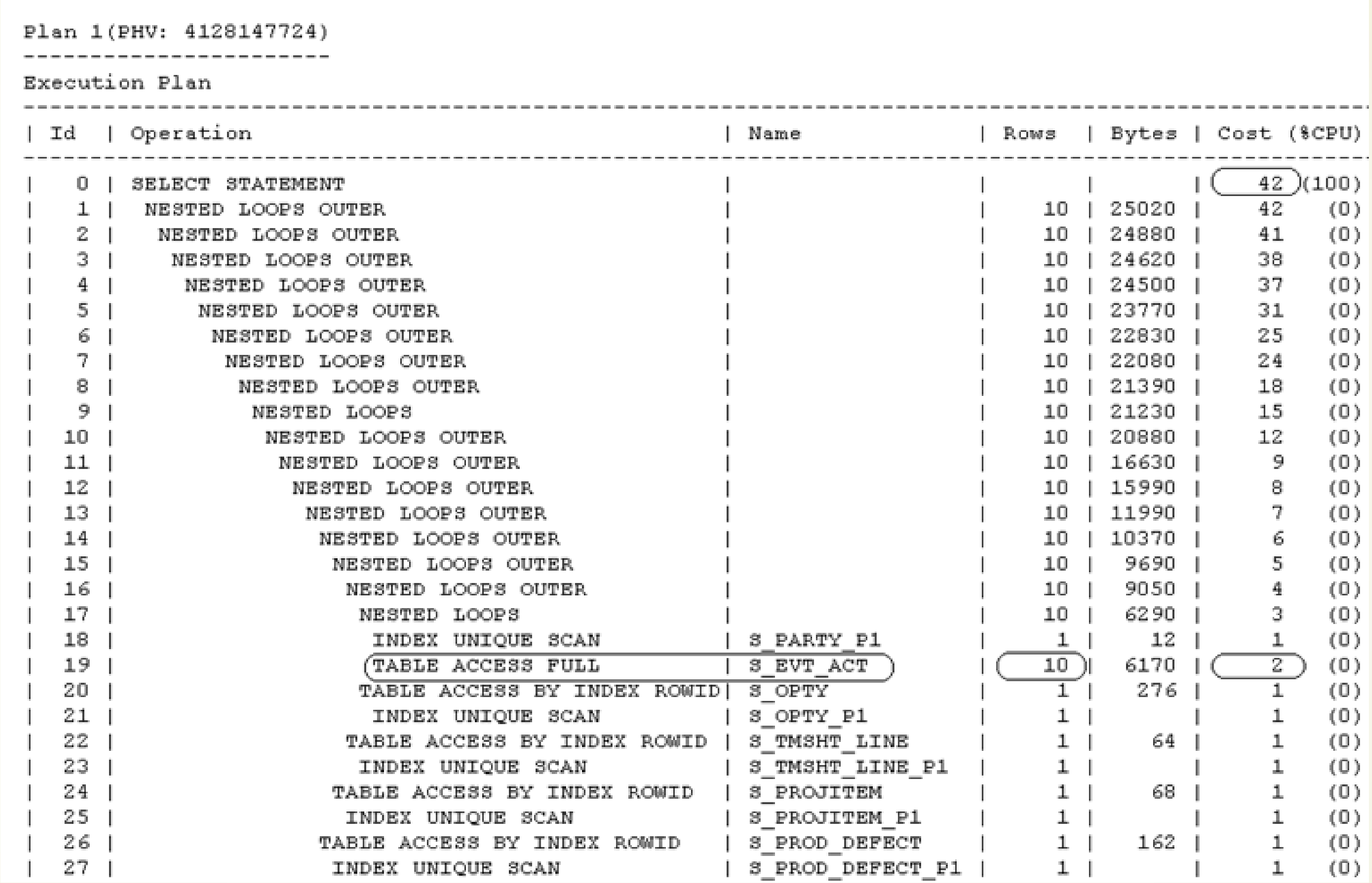
```
SQL> exec dbms_stats.gather_table_stats(ownname => 'SIEBEL', tabname => 'S_EVT_ACT',
cascade => true, no_invalidate => false, degree => 4);↵
↵
PL/SQL procedure successfully completed↵
```



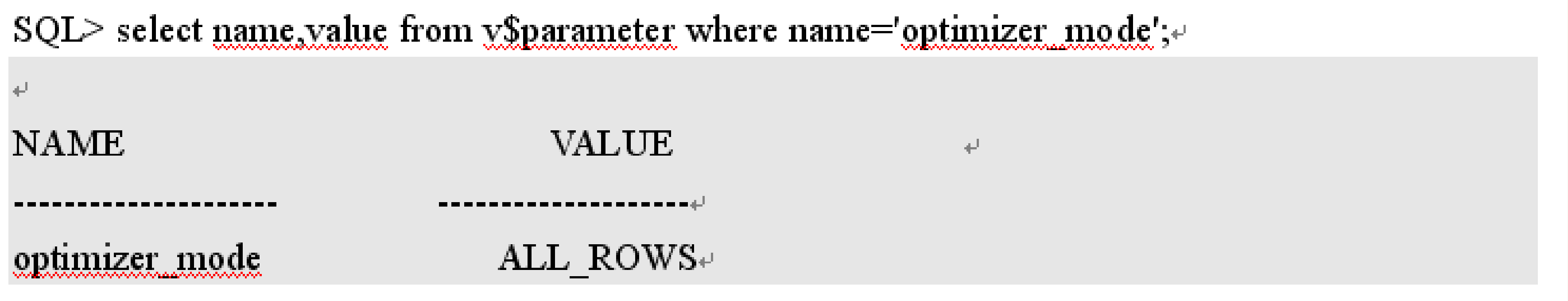
```
Topas Monitor for host:    XXXXXXXX          EVENTS/QUEUES    FILE/TTY
Mon Dec  6 16:30:23 2010   Interval:  2      Cswitch     4621  Readch    95.6M
                                             Syscall    92227  Writech  205.1K
Kernel    1.1   |#                         |  Reads     24668  Rawin        0
User      2.3   |#                         |  Writes      301  Ttyout     531
Wait      0.1   |#                         |  Forks         4  Igets        0
Idle     96.6   |##########################|  Execs         4  Namei      377
Physc =  0.77                 %Entc=  3.9   Runqueue     2.5  Dirblk       0
                                             Waitqueue   0.0
Network  KBPS   I-Pack  O-Pack   KB-In  KB-Out
en2     963.3   528.5   607.5   418.1   545.2 PAGING          MEMORY
en3     331.8   324.5   286.5   131.9   199.9 Faults      574  Real,MB   81920
lo0       0.3     6.0     6.0     0.2     0.2  Steals        0  % Comp     37.7

                                              PgspIn        0  % Noncomp   3.3
Disk     Busy%    KBPS     TPS KB-Read KB-Writ PgspOut       0  % Client    3.3
hdisk23    0.5  169.0    12.5   128.5    40.5 PageIn        0
hdisk12   10.5  108.2    27.0     0.0   108.2 PageOut       8  PAGING SPACE
hdisk21   11.0  108.2    27.0     0.0   108.2 Sios          8  Size,MB   98304
hdisk19    0.5   61.2     3.5     0.0    61.2                  % Used      0.0
hdisk17    0.5   54.5     6.0     0.0    54.5 NFS (calls/sec)  % Free    100.0
hdisk20    0.0   48.5     4.0     8.5    40.0 ServerV2       0
```

事情没有这么简单

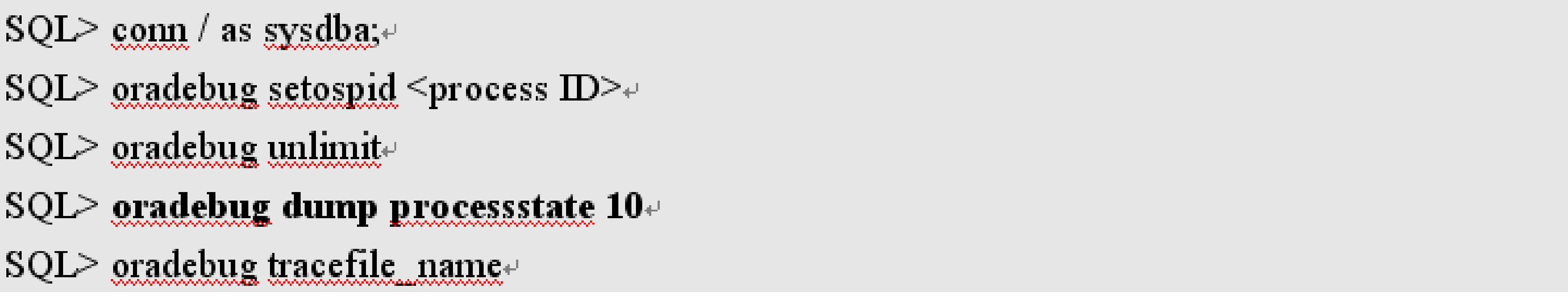
- SQL优化最有技术含量的部分不在于你通过种种手段（比如重新收集统计信息等）调整了目标SQL的执行计划、缩短了其执行时间、解决了该SQL的性能问题，而是在于你要知道CBO为什么在一开始会选错执行计划，你要知道CBO选错执行计划的根本原因



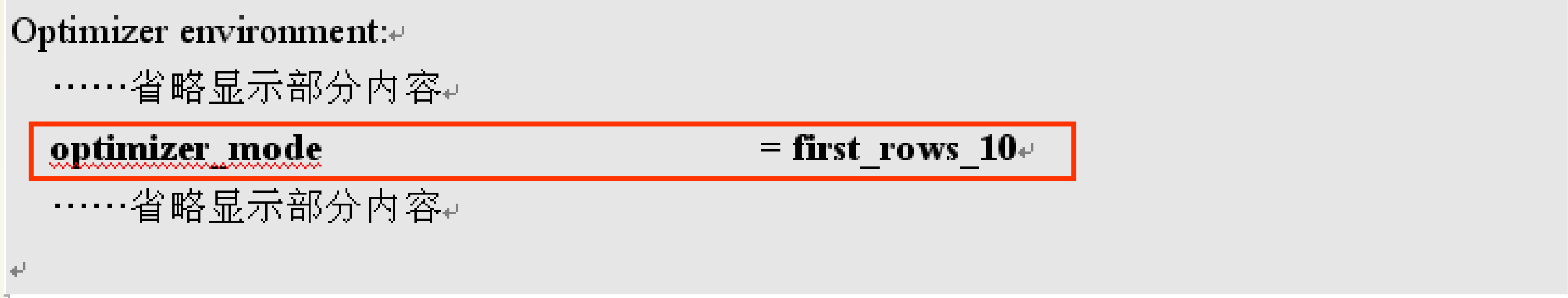
```
Plan 1(PHV: 4128147724)
-----------------------
Execution Plan
-----------------------------------------------------------------------------------------------------
| Id  | Operation                                    | Name                 | Rows  | Bytes | Cost (%CPU)
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                             |                      |       |       |    42 (100)
|   1 |  NESTED LOOPS OUTER                          |                      |    10 | 25020 |    42   (0)
|   2 |   NESTED LOOPS OUTER                         |                      |    10 | 24880 |    41   (0)
|   3 |    NESTED LOOPS OUTER                        |                      |    10 | 24620 |    38   (0)
|   4 |     NESTED LOOPS OUTER                       |                      |    10 | 24500 |    37   (0)
|   5 |      NESTED LOOPS OUTER                      |                      |    10 | 23770 |    31   (0)
|   6 |       NESTED LOOPS OUTER                     |                      |    10 | 22830 |    25   (0)
|   7 |        NESTED LOOPS OUTER                    |                      |    10 | 22080 |    24   (0)
|   8 |         NESTED LOOPS OUTER                   |                      |    10 | 21390 |    18   (0)
|   9 |          NESTED LOOPS                        |                      |    10 | 21230 |    15   (0)
|  10 |           NESTED LOOPS OUTER                 |                      |    10 | 20880 |    12   (0)
|  11 |            NESTED LOOPS OUTER                |                      |    10 | 16630 |     9   (0)
|  12 |             NESTED LOOPS OUTER               |                      |    10 | 15990 |     8   (0)
|  13 |              NESTED LOOPS OUTER              |                      |    10 | 11990 |     7   (0)
|  14 |               NESTED LOOPS OUTER             |                      |    10 | 10370 |     6   (0)
|  15 |                NESTED LOOPS OUTER            |                      |    10 |  9690 |     5   (0)
|  16 |                 NESTED LOOPS OUTER           |                      |    10 |  9050 |     4   (0)
|  17 |                  NESTED LOOPS                |                      |    10 |  6290 |     3   (0)
|  18 |                   INDEX UNIQUE SCAN          | S_PARTY_P1           |     1 |    12 |     1   (0)
|  19 |                   TABLE ACCESS FULL          | S_EVT_ACT            |    10 |  6170 |     2   (0)
|  20 |                  TABLE ACCESS BY INDEX ROWID| S_OPTY               |     1 |   276 |     1   (0)
|  21 |                   INDEX UNIQUE SCAN          | S_OPTY_P1            |     1 |       |     1   (0)
|  22 |                 TABLE ACCESS BY INDEX ROWID  | S_TMSHT_LINE         |     1 |    64 |     1   (0)
|  23 |                  INDEX UNIQUE SCAN           | S_TMSHT_LINE_P1      |     1 |       |     1   (0)
|  24 |                TABLE ACCESS BY INDEX ROWID   | S_PROJITEM           |     1 |    68 |     1   (0)
|  25 |                 INDEX UNIQUE SCAN            | S_PROJITEM_P1        |     1 |       |     1   (0)
|  26 |               TABLE ACCESS BY INDEX ROWID    | S_PROD_DEFECT        |     1 |   162 |     1   (0)
|  27 |                INDEX UNIQUE SCAN             | S_PROD_DEFECT_P1     |     1 |       |     1   (0)
```



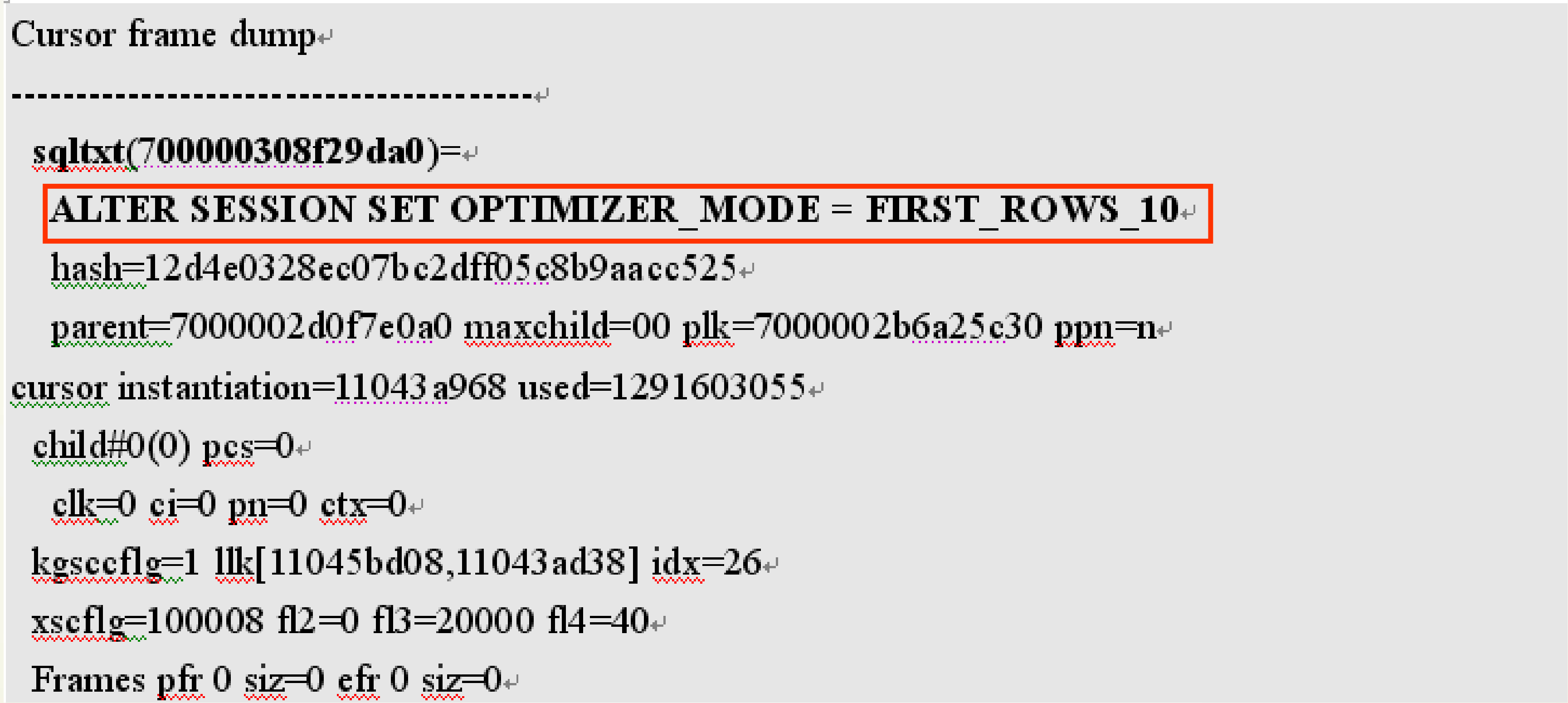
```
SQL> select name,value from v$parameter where name='optimizer_mode';

NAME                                    VALUE
---------------------              ---------------------
optimizer_mode                          ALL_ROWS
```



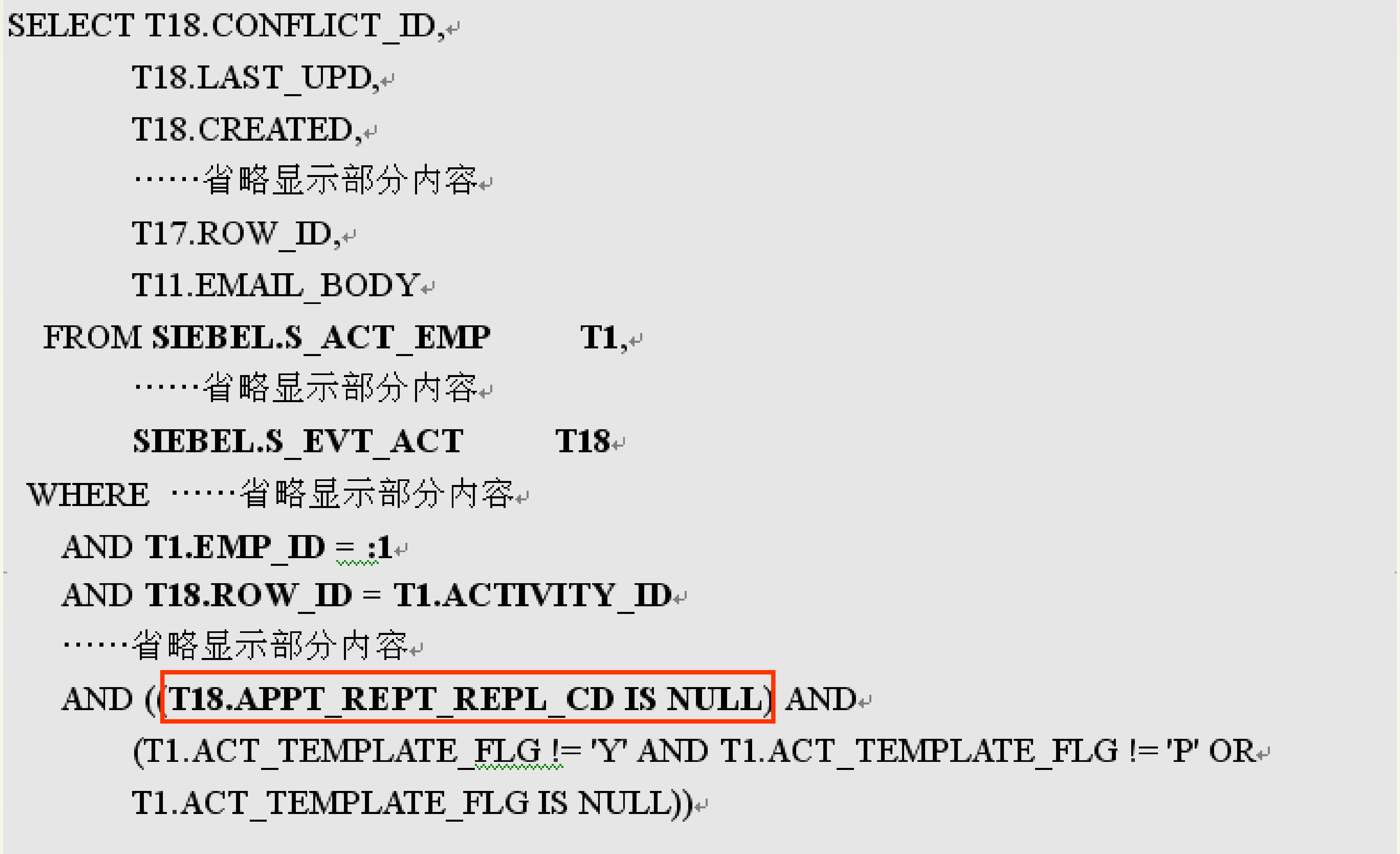
```
SQL> conn / as sysdba;
SQL> oradebug setospid <process ID>
SQL> oradebug unlimit
SQL> oradebug dump processstate 10
SQL> oradebug tracefile_name
```



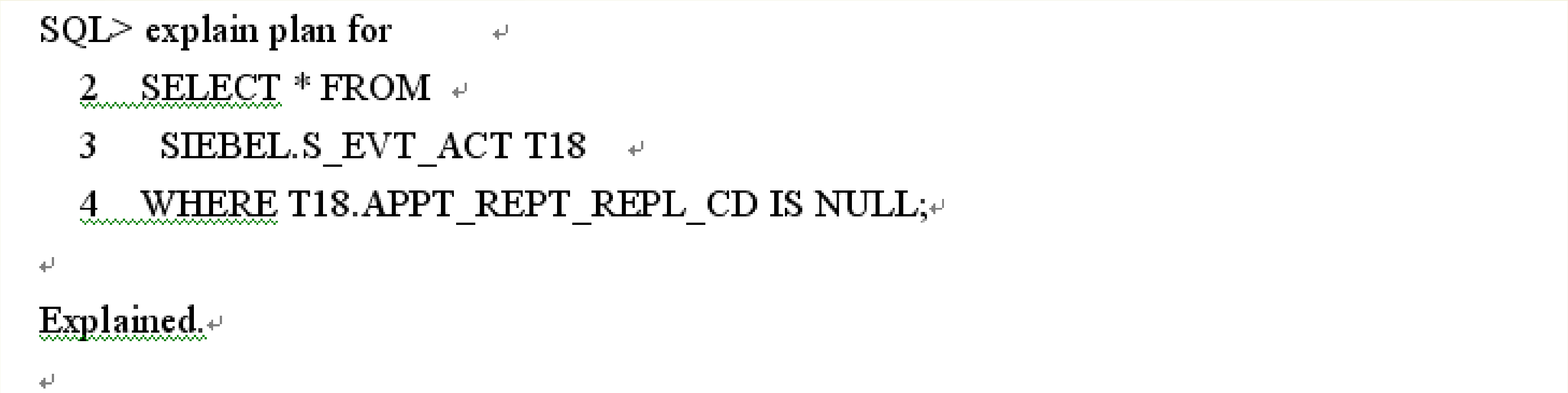
```
Optimizer environment:
  ……省略显示部分内容
  optimizer_mode                            = first_rows_10
  ……省略显示部分内容
```



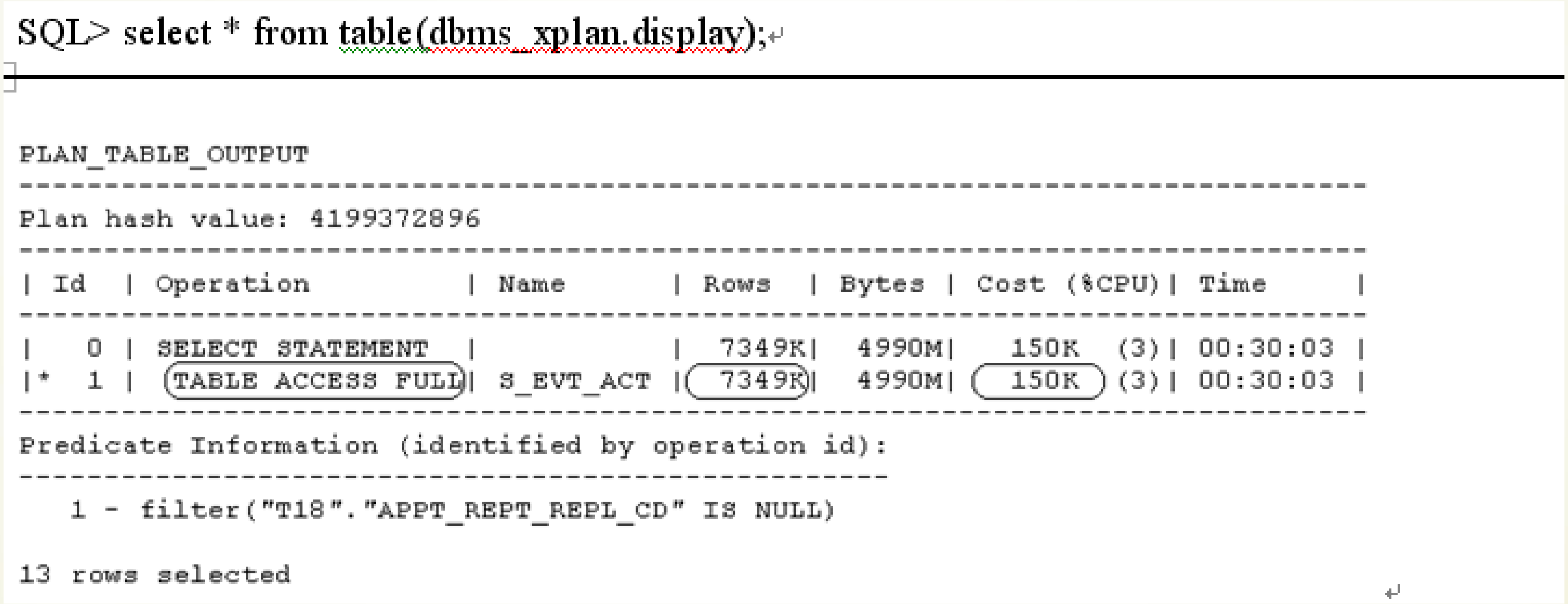
```
Cursor frame dump
----------------------------------------
 sqltxt(700000308f29da0)=
  ALTER SESSION SET OPTIMIZER_MODE = FIRST_ROWS_10
  hash=12d4e0328ec07bc2dff05c8b9aacc525
  parent=7000002d0f7e0a0 maxchild=00 plk=7000002b6a25c30 ppn=n
cursor instantiation=11043a968 used=1291603055
 child#0(0) pcs=0
  clk=0 ci=0 pn=0 ctx=0
 kgsccflg=1 llk[11045bd08,11043ad38] idx=26
 xscflg=100008 fl2=0 fl3=20000 fl4=40
 Frames pfr 0 siz=0 efr 0 siz=0
```



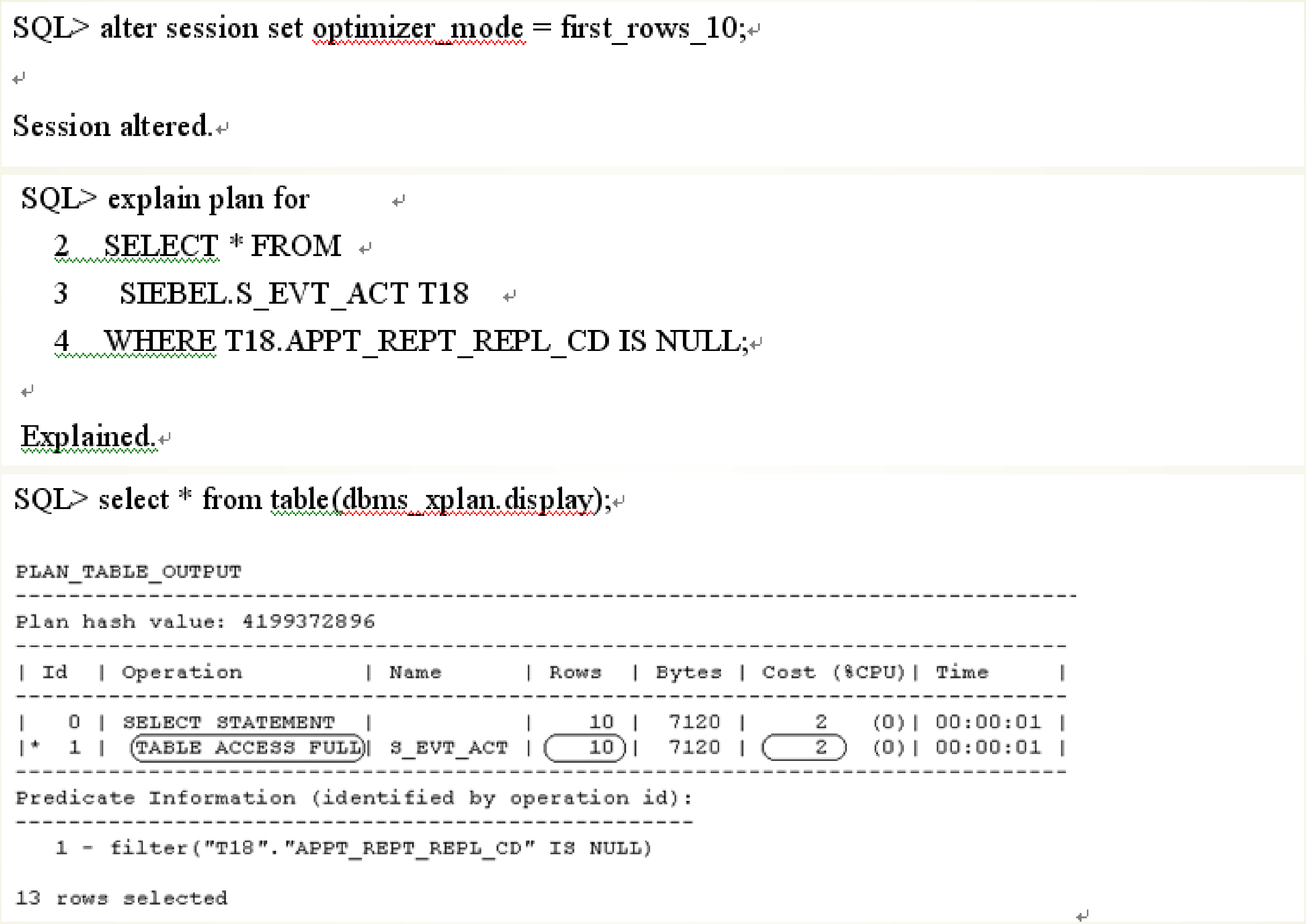
```
SELECT T18.CONFLICT_ID,
        T18.LAST_UPD,
        T18.CREATED,
        ……省略显示部分内容
        T17.ROW_ID,
        T11.EMAIL_BODY
  FROM SIEBEL.S_ACT_EMP        T1,
        ……省略显示部分内容
        SIEBEL.S_EVT_ACT       T18
 WHERE  ……省略显示部分内容
    AND T1.EMP_ID = :1
    AND T18.ROW_ID = T1.ACTIVITY_ID
    ……省略显示部分内容
    AND ((T18.APPT_REPT_REPL_CD IS NULL) AND
        (T1.ACT_TEMPLATE_FLG != 'Y' AND T1.ACT_TEMPLATE_FLG != 'P' OR
        T1.ACT_TEMPLATE_FLG IS NULL))
```



```
SQL> explain plan for
  2  SELECT * FROM
  3    SIEBEL.S_EVT_ACT T18
  4  WHERE T18.APPT_REPT_REPL_CD IS NULL;

Explained.
```



```
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
Plan hash value: 4199372896
-------------------------------------------------------------------------------
| Id  | Operation         | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |           |  7349K|  4990M|   150K  (3)| 00:30:03 |
|*  1 |  TABLE ACCESS FULL| S_EVT_ACT |  7349K|  4990M|   150K  (3)| 00:30:03 |
-------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter("T18"."APPT_REPT_REPL_CD" IS NULL)

13 rows selected
```



```
SQL> alter session set optimizer_mode = first_rows_10;

Session altered.
```

```
SQL> explain plan for
  2  SELECT * FROM
  3    SIEBEL.S_EVT_ACT T18
  4  WHERE T18.APPT_REPT_REPL_CD IS NULL;

Explained.
```

```
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 4199372896
-------------------------------------------------------------------------------
| Id  | Operation         | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |           |    10 |  7120 |     2   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| S_EVT_ACT |    10 |  7120 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter("T18"."APPT_REPT_REPL_CD" IS NULL)

13 rows selected
```

深入分析

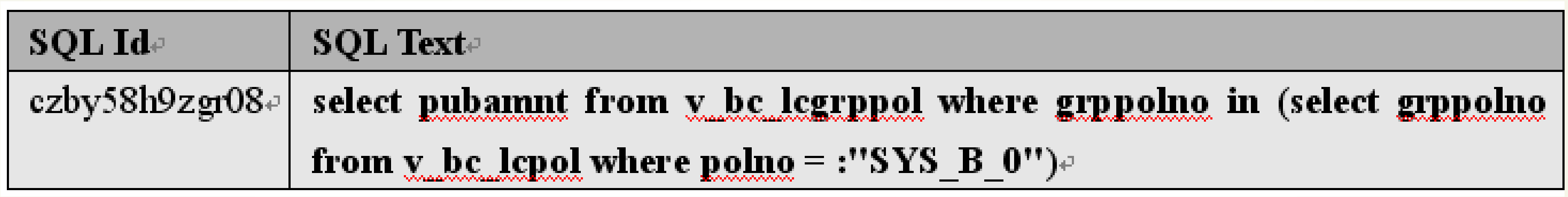
- 导致CBO评估出对一个实际数据量为730多万且统计信息准确的大表S_EVT_ACT执行全表扫描操作后的成本值仅为2的原因是因为参数OPTIMIZER_MODE的值在session级别被修改成了FIRST_ROWS_10，这同时也是导致上述坐席登陆慢的问题多次不间断出现的根本原因。

解决方法

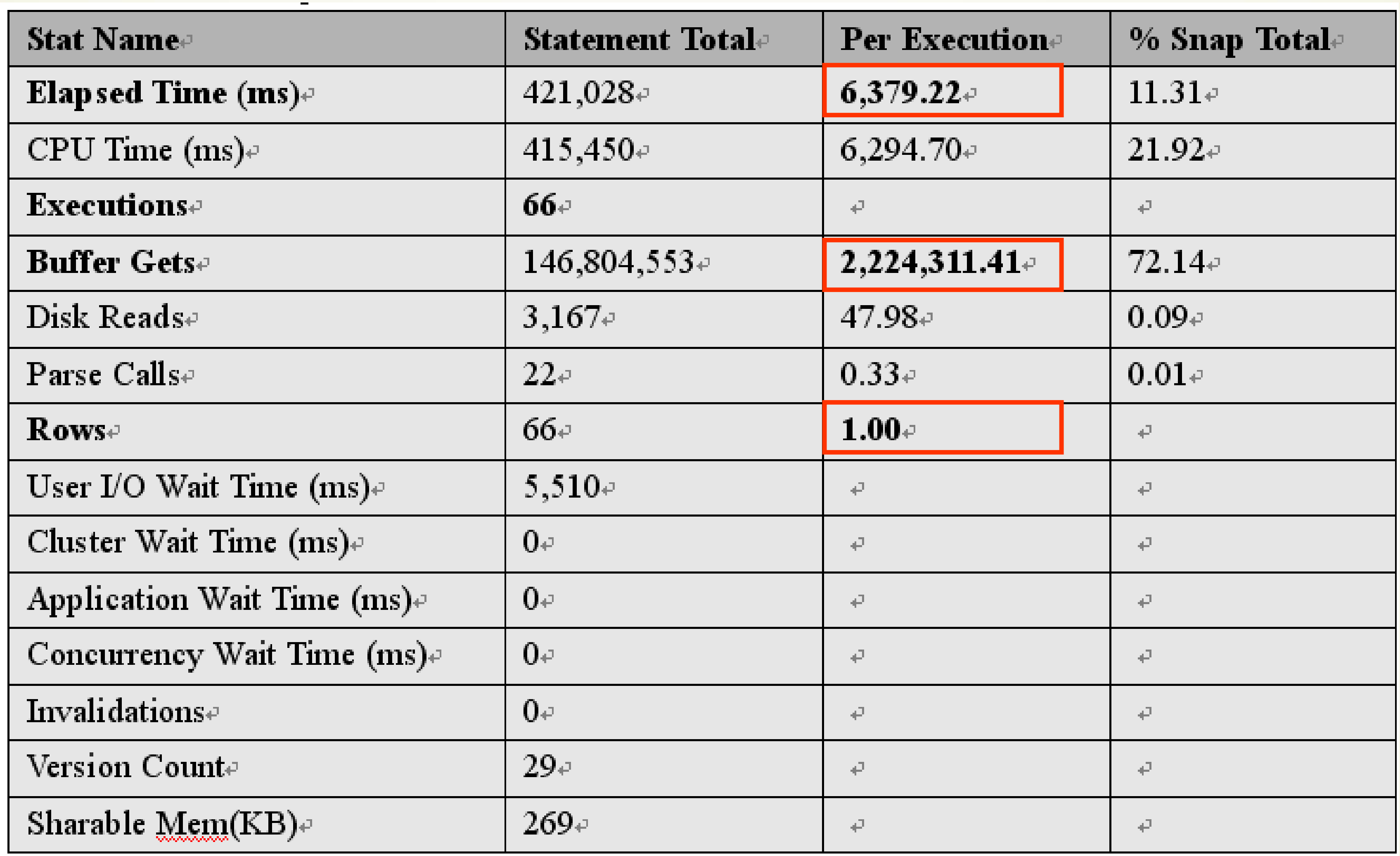
- 修改各个session中对于参数OPTIMIZER_MODE的设置，将其值修改为默认值ALL_ROWS
- 如果不能在session级修改参数OPTIMIZER_MODE的值，我们还可以使用SQL Profile。在上述18个表关联SQL中加入Hint（即/*+ index(T18 S_EVT_ACT_P1）*/），并用加入Hint后改写SQL的执行计划替换原SQL的执行计划

查询转换的综合应用实例（逻辑读从200万降到6）

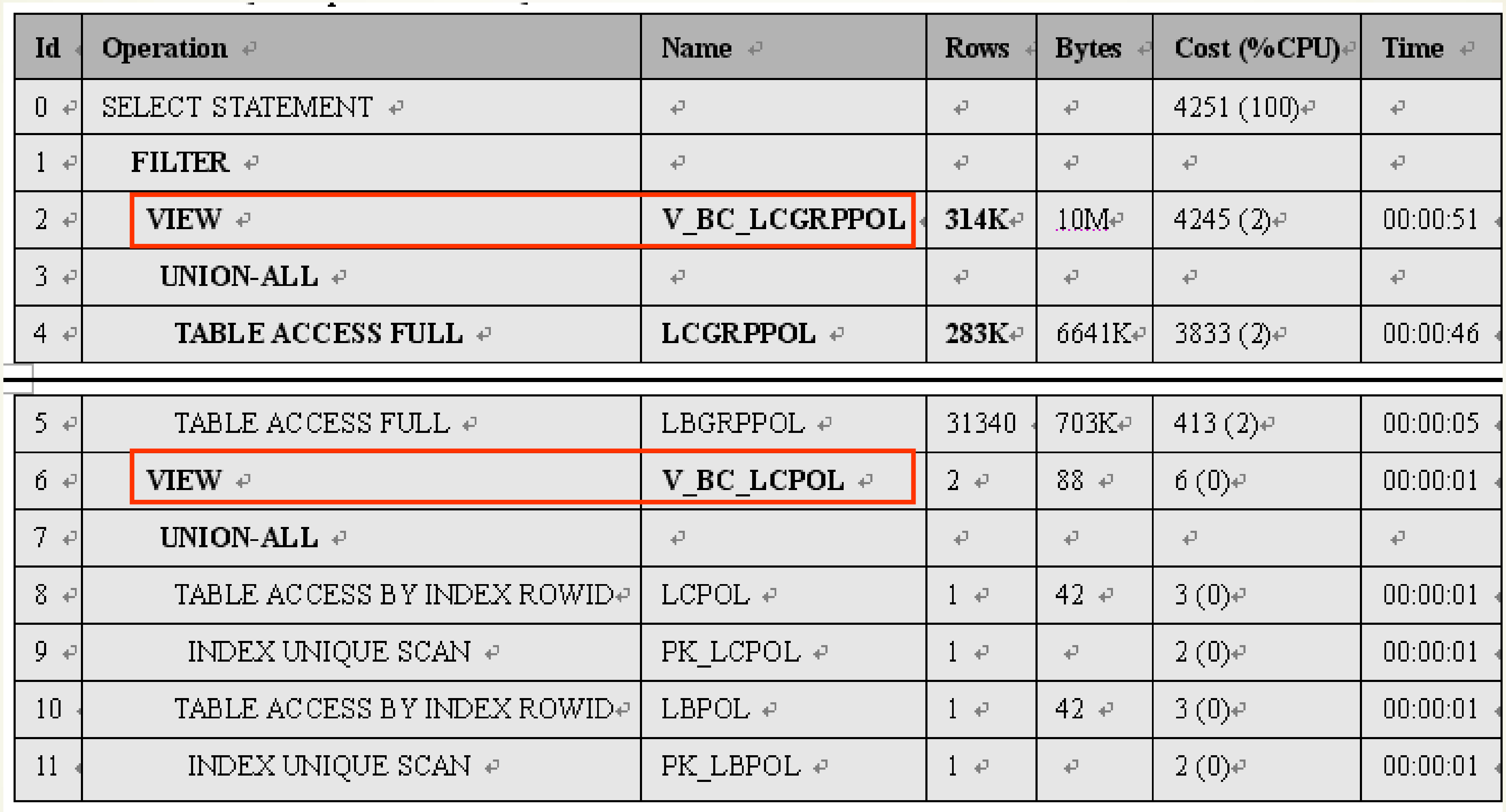
- 某系统某个模块响应速度缓慢，客户方DBA已经从AWR报告的TOP SQL中定位和确认了导致上述模块响应速度缓慢的SQL。现需要对该SQL进行分析和调优，以提高响应速度，减轻系统压力。



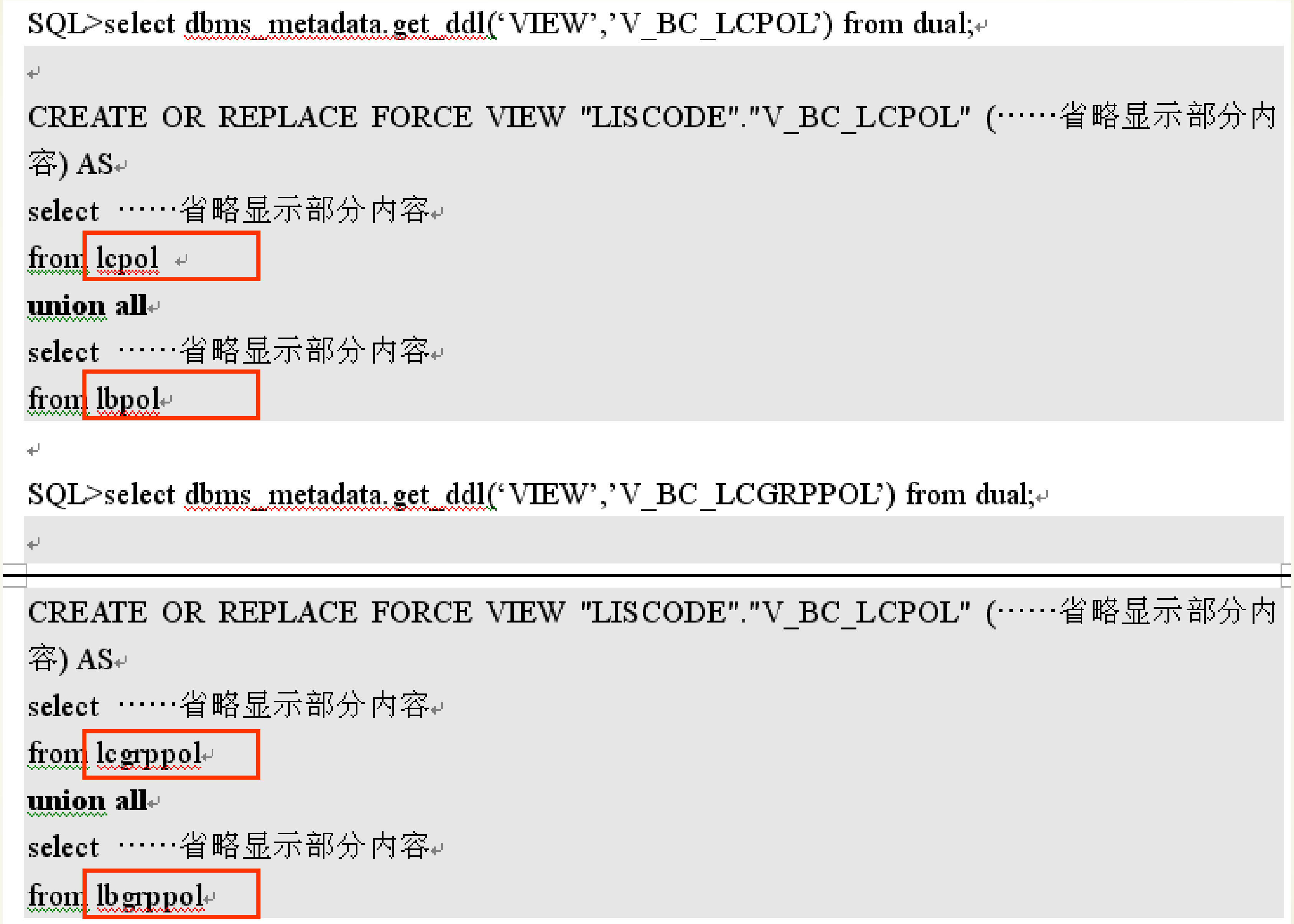
SQL Id	SQL Text
czby58h9zgr08	select pubamnt from v_bc_lcgrppol where grppolno in (select grppolno from v_bc_lcpol where polno = :"SYS_B_0")



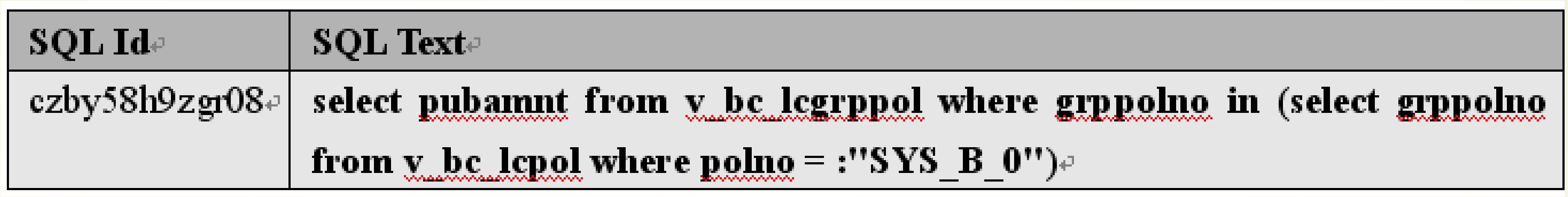
Stat Name	Statement Total	Per Execution	% Snap Total
Elapsed Time (ms)	421,028	**6,379.22**	11.31
CPU Time (ms)	415,450	6,294.70	21.92
Executions	**66**		
Buffer Gets	146,804,553	**2,224,311.41**	72.14
Disk Reads	3,167	47.98	0.09
Parse Calls	22	0.33	0.01
Rows	66	**1.00**	
User I/O Wait Time (ms)	5,510		
Cluster Wait Time (ms)	0		
Application Wait Time (ms)	0		
Concurrency Wait Time (ms)	0		
Invalidations	0		
Version Count	29		
Sharable Mem(KB)	269		



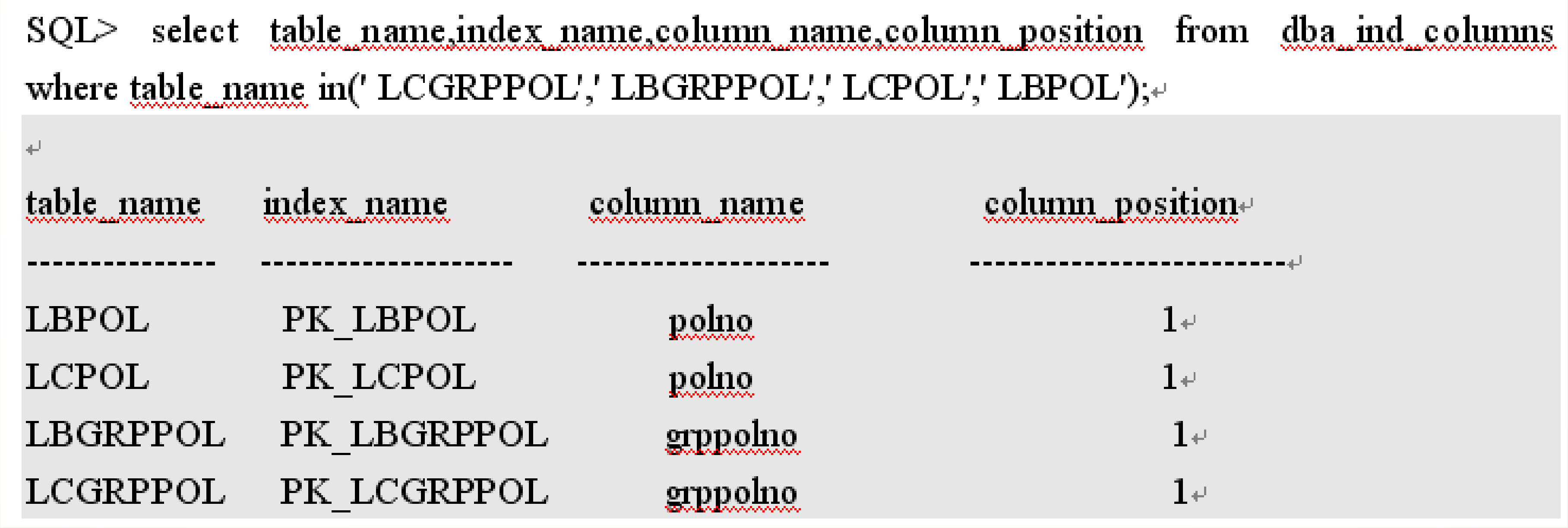
Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT				4251 (100)	
1	**FILTER**					
2	**VIEW**	**V_BC_LCGRPPOL**	**314K**	10M	4245 (2)	00:00:51
3	**UNION-ALL**					
4	**TABLE ACCESS FULL**	**LCGRPPOL**	**283K**	6641K	3833 (2)	00:00:46
5	TABLE ACCESS FULL	LBGRPPOL	31340	703K	413 (2)	00:00:05
6	**VIEW**	**V_BC_LCPOL**	2	88	6 (0)	00:00:01
7	**UNION-ALL**					
8	TABLE ACCESS BY INDEX ROWID	LCPOL	1	42	3 (0)	00:00:01
9	INDEX UNIQUE SCAN	PK_LCPOL	1		2 (0)	00:00:01
10	TABLE ACCESS BY INDEX ROWID	LBPOL	1	42	3 (0)	00:00:01
11	INDEX UNIQUE SCAN	PK_LBPOL	1		2 (0)	00:00:01



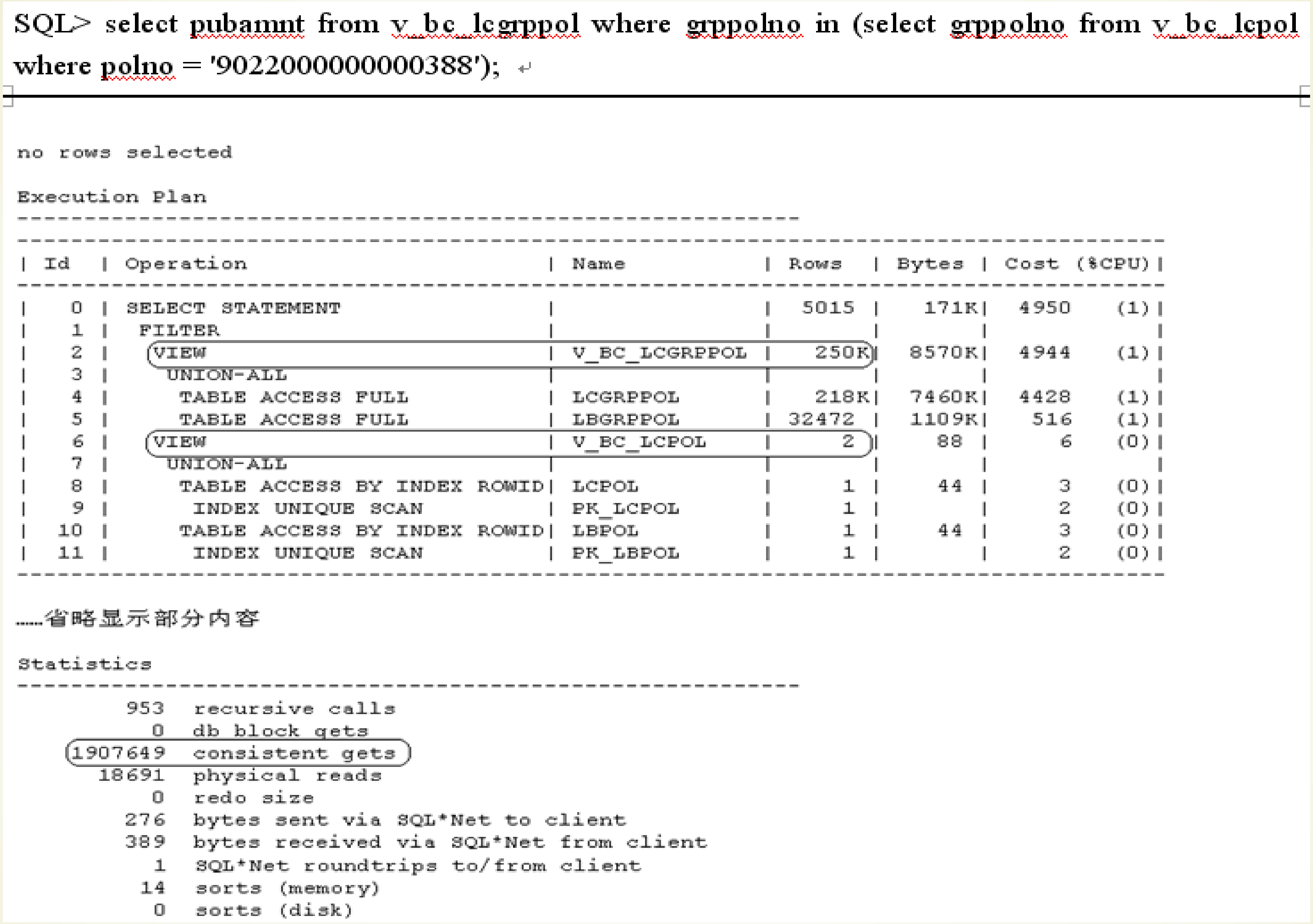
```
SQL>select dbms_metadata.get_ddl(‘VIEW’,’V_BC_LCPOL’) from dual;

CREATE OR REPLACE FORCE VIEW "LISCODE"."V_BC_LCPOL" (……省略显示部分内容) AS
select ……省略显示部分内容
from lcpol
union all
select ……省略显示部分内容
from lbpol

SQL>select dbms_metadata.get_ddl(‘VIEW’,’V_BC_LCGRPPOL’) from dual;

CREATE OR REPLACE FORCE VIEW "LISCODE"."V_BC_LCPOL" (……省略显示部分内容) AS
select ……省略显示部分内容
from lcgrppol
union all
select ……省略显示部分内容
from lbgrppol
```



SQL Id	SQL Text
czby58h9zgr08	select pubamnt from v_bc_lcgrppol where grppolno in (select grppolno from v_bc_lcpol where polno = :"SYS_B_0")



```
SQL> select table_name,index_name,column_name,column_position from dba_ind_columns
where table_name in(' LCGRPPOL',' LBGRPPOL',' LCPOL',' LBPOL');

table_name      index_name          column_name          column_position
---------------  --------------------  --------------------  --------------------------
LBPOL           PK_LBPOL              polno                          1
LCPOL           PK_LCPOL              polno                          1
LBGRPPOL        PK_LBGRPPOL           grppolno                        1
LCGRPPOL        PK_LCGRPPOL           grppolno                        1
```



SQL> select pubamnt from v_bc_lcgrppol where grppolno in (select grppolno from v_bc_lcpol where polno = '9022000000000388');

```
no rows selected

Execution Plan
----------------------------------------------------------
--------------------------------------------------------------------------------------
| Id  | Operation                        | Name            | Rows  | Bytes | Cost (%CPU)|
--------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                 |                 |  5015 |   171K|  4950   (1)|
|   1 |  FILTER                          |                 |       |       |            |
|   2 |   VIEW                           | V_BC_LCGRPPOL   |   250K|  8570K|  4944   (1)|
|   3 |    UNION-ALL                     |                 |       |       |            |
|   4 |     TABLE ACCESS FULL            | LCGRPPOL        |   218K|  7460K|  4428   (1)|
|   5 |     TABLE ACCESS FULL            | LBGRPPOL        | 32472 |  1109K|   516   (1)|
|   6 |   VIEW                           | V_BC_LCPOL      |     2 |    88 |     6   (0)|
|   7 |    UNION-ALL                     |                 |       |       |            |
|   8 |     TABLE ACCESS BY INDEX ROWID| LCPOL           |     1 |    44 |     3   (0)|
|   9 |      INDEX UNIQUE SCAN           | PK_LCPOL        |     1 |       |     2   (0)|
|  10 |     TABLE ACCESS BY INDEX ROWID| LBPOL           |     1 |    44 |     3   (0)|
|  11 |      INDEX UNIQUE SCAN           | PK_LBPOL        |     1 |       |     2   (0)|
--------------------------------------------------------------------------------------
```

……省略显示部分内容

```
Statistics
----------------------------------------------------------
        953  recursive calls
          0  db block gets
    1907649  consistent gets
      18691  physical reads
          0  redo size
        276  bytes sent via SQL*Net to client
        389  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
         14  sorts (memory)
          0  sorts (disk)
```

初步分析

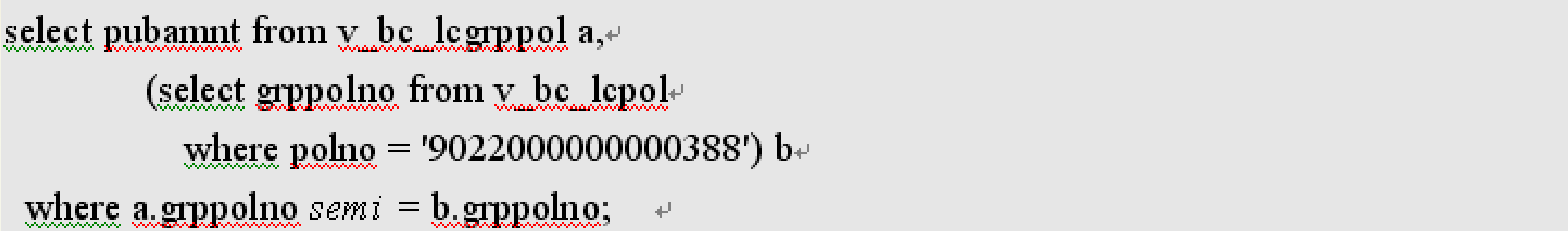
- 上述SQL包含了IN，而IN之后的括号内是一个包含视图的子查询（即select grppolno from v_bc_lcpol where polno = ‘9022000000000388’），它不是一个常量的集合，所以Oracle这里不能对该SQL做“IN-List Iterator”和“IN-List Expansion /OR Expansion”；
- 上述SQL中的视图V_BC_LCGRPPOL和V_BC_LCPOL均包含了集合运算符UNION ALL，所以Oracle这里也不能对该SQL做视图合并；
- **于是Oracle现在就只剩下了两条路可走：要么对该SQL走FILTER类型的执行计划（即“IN-List Filter”），要么对该SQL做子查询展开。**

初步分析

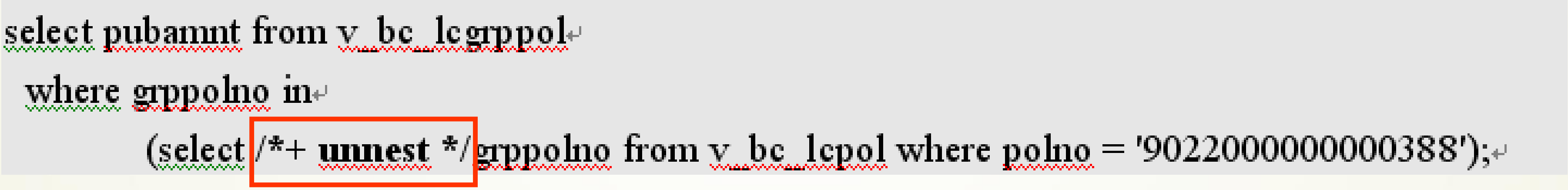
- **Oracle这里选择的是走FILTER类型的执行计划，**既然是选择走FILTER类型的执行计划，同时上述SQL的where条件中除了“grppolno in (select grppolno from v_bc_lcpol where polno = ’9022000000000388’)”之外再没有其他的限制条件，**那么Oracle必然就会先执行“select * from v_bc_lcgrppol”，这当然会全表扫描视图V_BC_LCGRPPOL的基表LCGRPPOL和LBGRPPOL**

深入分析

- 对于不拆开子查询但是会把它转换为一个内嵌视图（Inline View）的子查询展开，只有当经过子查询展开后的等价改写SQL的成本值小于原SQL的成本值时，Oracle才会对目标SQL执行子查询展开。**所以这里CBO为什么没有选择走子查询展开的原因要么是因为经过子查询展开后的等价改写SQL的成本值大于原SQL的成本值，要么是因为CBO的Bug。**



```
select pubamnt from v_bc_lcgrppol a,↵
            (select grppolno from v_bc_lcpol↵
               where polno = '9022000000000388') b↵
  where a.grppolno semi = b.grppolno;    ↵
```



```
select pubamnt from v_bc_lcgrppol↵
  where grppolno in↵
            (select /*+ unnest */grppolno from v_bc_lcpol where polno = '9022000000000388');↵
```

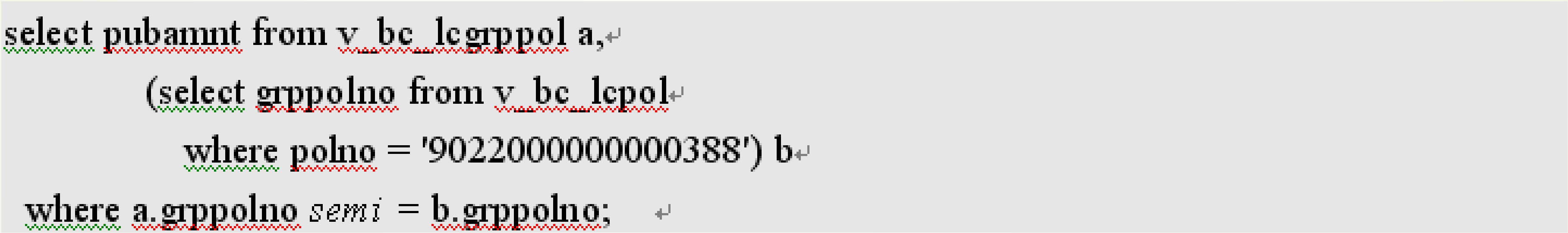
事情没有这么简单

- 在上述SQL中加入UNNEST Hint后虽然可以强制让Oracle做子查询展开，但**此时子查询展开后并不一定会走我们想要的嵌套循环连接**，而且**即使走嵌套循环连接，驱动结果集也不一定就是我们想要的**根据主键PK_LCPOL和PK_LBPOL去分别访问表LCPOL和表LBPOL后再做UNION ALL操作后得到的结果集。

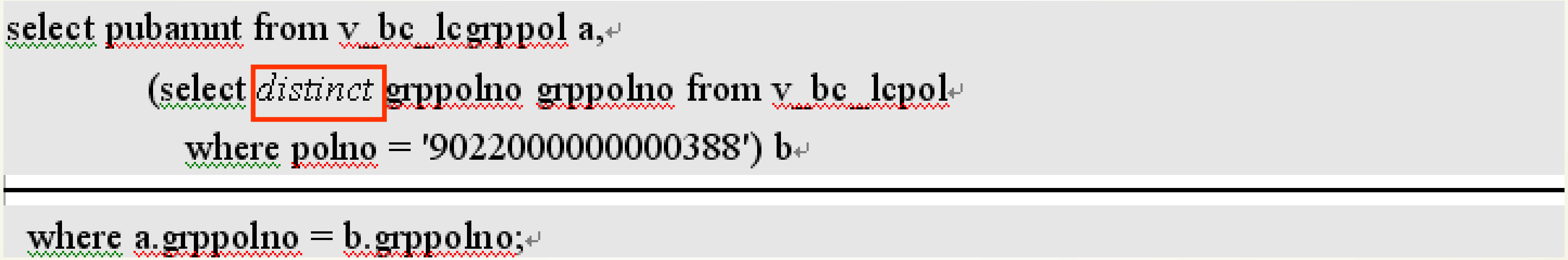
事情没有这么简单

- 为了能更精细的控制上述SQL的执行计划，这里**我们选择了直接使用子查询展开后等价改写的形式**，这样一旦走不出我们想要的执行计划，我们还可以使用额外的Hint（诸如ORDERD、USE_NL等）来继续对其执行计划做调整。

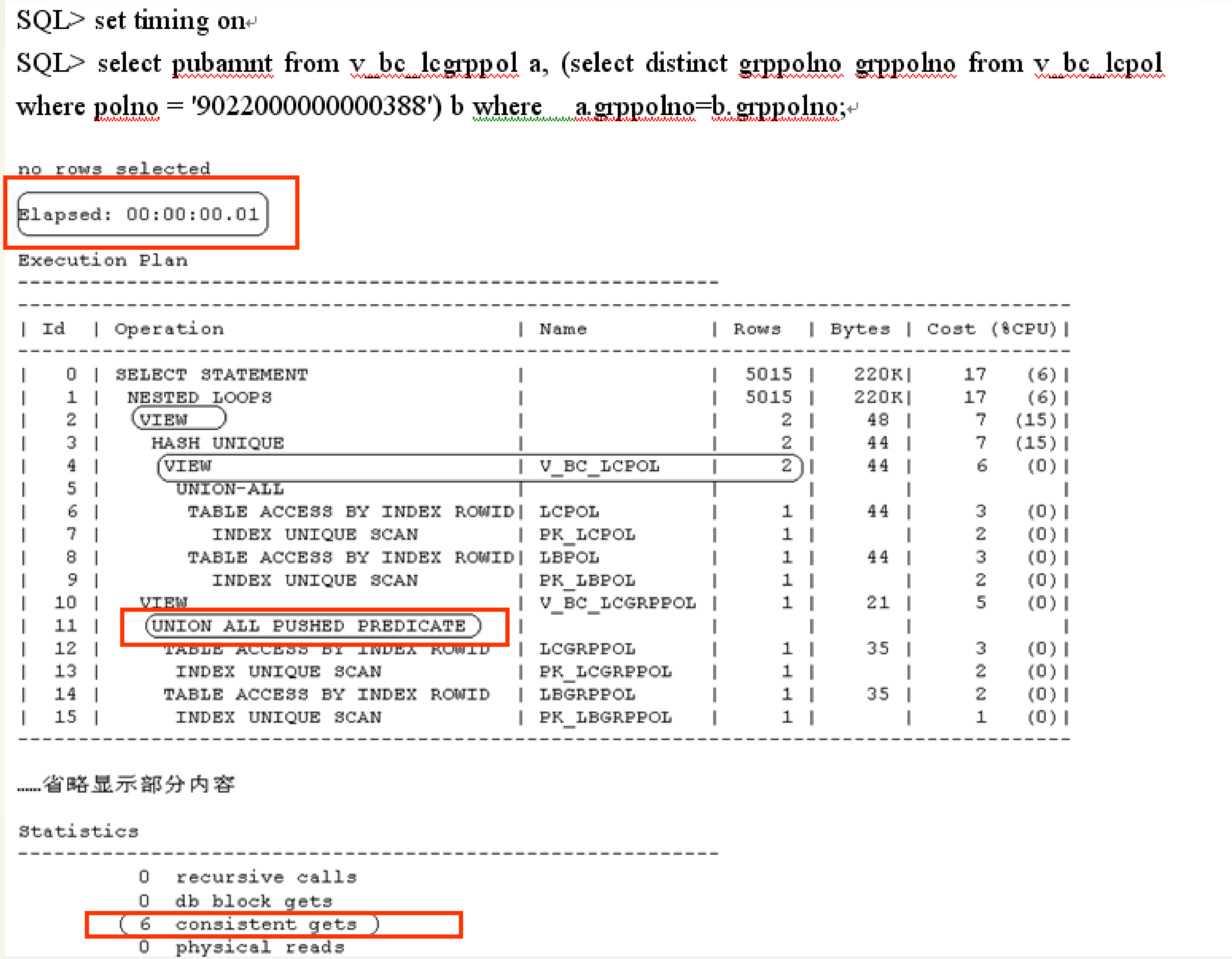
解决方法



```
select pubamnt from v_bc_lcgrppol a,
          (select grppolno from v_bc_lcpol
             where polno = '9022000000000388') b
 where a.grppolno semi = b.grppolno;
```



```
select pubamnt from v_bc_lcgrppol a,
          (select distinct grppolno grppolno from v_bc_lcpol
             where polno = '9022000000000388') b
 where a.grppolno = b.grppolno;
```



```
SQL> set timing on
SQL> select pubamnt from v_bc_lcgrppol a, (select distinct grppolno grppolno from v_bc_lcpol
where polno = '9022000000000388') b where   a.grppolno=b.grppolno;

no rows selected

Elapsed: 00:00:00.01

Execution Plan
----------------------------------------------------------

---------------------------------------------------------------------------------------
| Id  | Operation                        | Name           | Rows  | Bytes | Cost (%CPU)|
---------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                 |                |  5015 |   220K|    17   (6)|
|   1 |  NESTED LOOPS                    |                |  5015 |   220K|    17   (6)|
|   2 |   VIEW                           |                |     2 |    48 |     7  (15)|
|   3 |    HASH UNIQUE                   |                |     2 |    44 |     7  (15)|
|   4 |     VIEW                         | V_BC_LCPOL     |     2 |    44 |     6   (0)|
|   5 |      UNION-ALL                   |                |       |       |            |
|   6 |       TABLE ACCESS BY INDEX ROWID| LCPOL          |     1 |    44 |     3   (0)|
|   7 |         INDEX UNIQUE SCAN        | PK_LCPOL       |     1 |       |     2   (0)|
|   8 |       TABLE ACCESS BY INDEX ROWID| LBPOL          |     1 |    44 |     3   (0)|
|   9 |         INDEX UNIQUE SCAN        | PK_LBPOL       |     1 |       |     2   (0)|
|  10 |   VIEW                           | V_BC_LCGRPPOL  |     1 |    21 |     5   (0)|
|  11 |    UNION ALL PUSHED PREDICATE    |                |       |       |            |
|  12 |     TABLE ACCESS BY INDEX ROWID  | LCGRPPOL       |     1 |    35 |     3   (0)|
|  13 |      INDEX UNIQUE SCAN           | PK_LCGRPPOL    |     1 |       |     2   (0)|
|  14 |     TABLE ACCESS BY INDEX ROWID  | LBGRPPOL       |     1 |    35 |     2   (0)|
|  15 |      INDEX UNIQUE SCAN           | PK_LBGRPPOL    |     1 |       |     1   (0)|
---------------------------------------------------------------------------------------

......省略显示部分内容

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

解决方法（续）

- 至此我们就圆满的解决了上述问题，从这个例子的解决过程我们可以看出，虽然最后的解决方法很简单，但这其实完全倚赖于我们对Oracle如何处理SQL语句中的IN、子查询展开、视图合并和连接谓词推入的深刻理解

总结

- 兵无常势，水无常形；运用之妙，存乎一心

谢谢！

欢迎莅临

2013中国数据库技术大会