技术人生系列背后的故事

About Me

- 周永康（老K）
- 11g OCP，11gOCM
- 多年的总行数据中心Oracle运维经验
- 擅长SQL优化
- 擅长Oracle故障诊断
- 擅长Oracle性能优化
- 《技术人生系列微信文章》主编
- 中亦安图科技股份有限公司

从技术人生系列看ORACLE技术进阶之路

- ORACLE的视图进阶
- ORACLE的EVENT的进阶
- ORACLE 的dump 方法使用进阶
- ORACLE相关工具的进阶
- ORACLE的trace分析的进阶

ORACLE的视图使用进阶

- **基本的视图**

常设置的事件：v$session/v$lock/v$transaction

- **视图查询的进阶**

v$sesstat/v$sysstat

v$active_session_history

dba_hist_active_sess_history

x$kgllk

x$bh

ORACLE的event使用进阶

- **基本的event**

常设置的事件：10046/10053/10231

- **Event事件的进阶**

这些event你会设置吗：

errorstack

10949

28041

ORACLE的dump使用进阶

- **基本的dump方法**

常用的dump方法：

Dump block

Dump systemstate

Dump processstate

- **Dump方法的进阶**

Dump heapdump 536870917---dump pga

Dump heapdump 536870914---dump shared pool

其他内存dump的方法

ORACLE的分析工具使用进阶

- **基本的工具方法**

常用的工具方法：

Awrrpt

Awrsqrpt

Nmon

Oswatcher

- **分析工具的进阶**

gdb/xdb

truss/Strace

svmon

ORACLE的trace分析的进阶

- **基本的trace分析**

10053/10046的trace分析

Alert日志/lgwr的trace分析

- **Trace分析的进阶**

600错误的trace分析

7445错误的trace分析

4031产生的trace分析

各种手动产生的trace的分析

两个简单的案例分析

- 一次数据库实例性能分析
- 一条高SQL的优化

一次数据库实例性能分析

- 背景：
  - 一套**ORACLE RAC**实例性能出现严重问题
  - 客户现场分析，主要等待事件与**GC**相关
  - 客户现场解决方案，关闭另一节点，留下其中一个节点，问题解决

- 客户的问题
  - 为什么
  - 我要怎么做

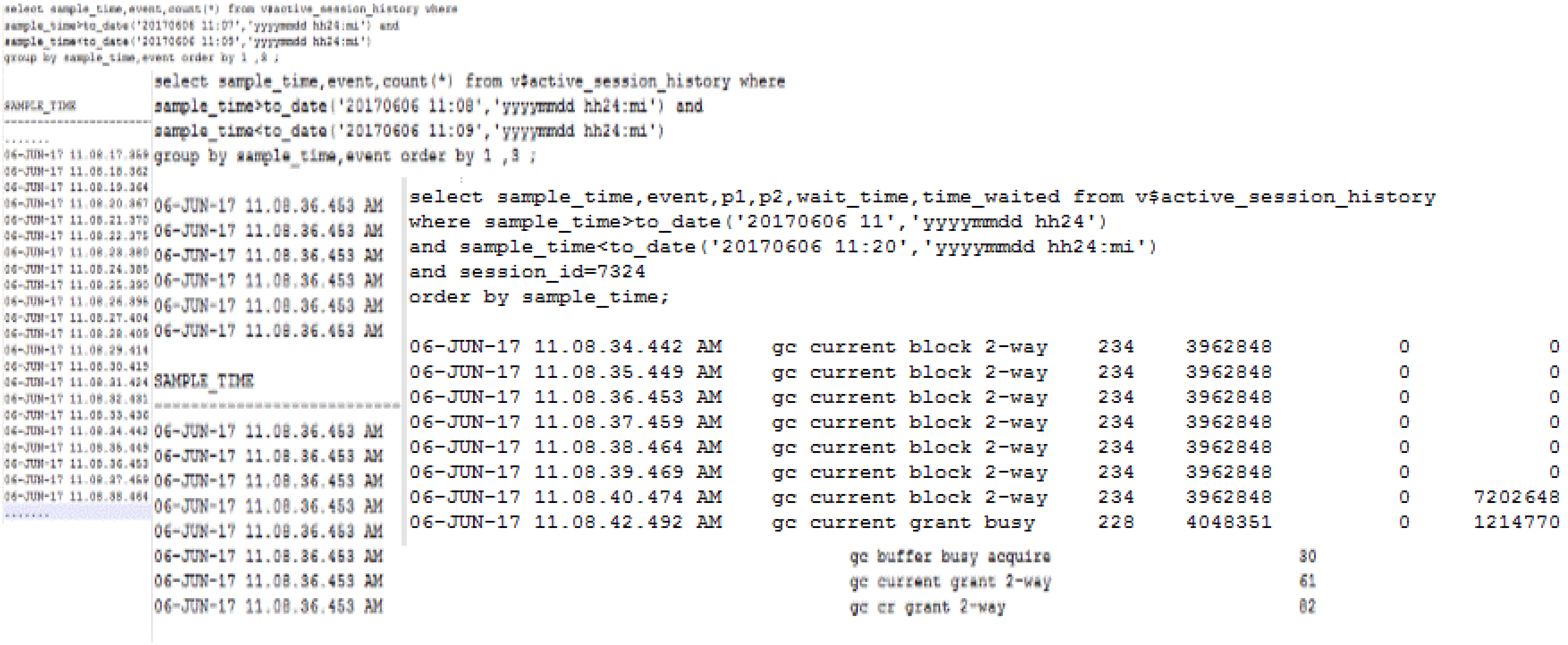
一次数据库实例性能分析

- 客户给的信息
  - 早上节点一的网卡好像出了一些问题（是不是网络有了问题）
  - 节点二的内存比节点一的内存要小一些（所以选择停了节点二）

- 自己收集信息
  - 动态性能视图
  - **v$active_session_history/dba_hist_active_sess_history**
  - **Alert**日志**/**及其他**trace**文件

一次数据库实例性能分析

- 多维度视图分析



```
select sample_time,event,count(*) from v$active_session_history where
sample_time>to_date('20170606 11:07','yyyymmdd hh24:mi') and
sample_time<to_date('20170606 11:09','yyyymmdd hh24:mi')
group by sample_time,event order by 1 ,3 ;

SAMPLE_TIME
-------------------------
.......
06-JUN-17 11.08.17.359
06-JUN-17 11.08.18.362
06-JUN-17 11.08.19.364
06-JUN-17 11.08.20.367
06-JUN-17 11.08.21.370
06-JUN-17 11.08.22.375
06-JUN-17 11.08.23.380
06-JUN-17 11.08.24.385
06-JUN-17 11.08.25.390
06-JUN-17 11.08.26.396
06-JUN-17 11.08.27.404
06-JUN-17 11.08.28.409
06-JUN-17 11.08.29.414
06-JUN-17 11.08.30.415
06-JUN-17 11.08.31.424
06-JUN-17 11.08.32.431
06-JUN-17 11.08.33.436
06-JUN-17 11.08.34.442
06-JUN-17 11.08.35.449
06-JUN-17 11.08.36.453
06-JUN-17 11.08.37.459
06-JUN-17 11.08.38.464
.......
```

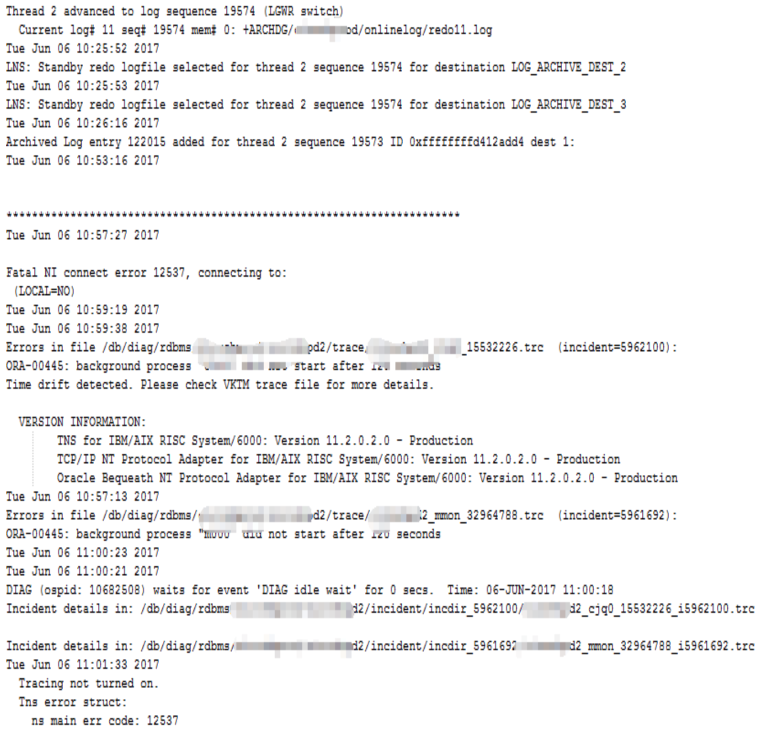
```
select sample_time,event,count(*) from v$active_session_history where
sample_time>to_date('20170606 11:08','yyyymmdd hh24:mi') and
sample_time<to_date('20170606 11:09','yyyymmdd hh24:mi')
group by sample_time,event order by 1 ,3 ;

06-JUN-17 11.08.36.453 AM
06-JUN-17 11.08.36.453 AM
06-JUN-17 11.08.36.453 AM
06-JUN-17 11.08.36.453 AM
06-JUN-17 11.08.36.453 AM
06-JUN-17 11.08.36.453 AM

SAMPLE_TIME
------------------------------
06-JUN-17 11.08.36.453 AM
06-JUN-17 11.08.36.453 AM
06-JUN-17 11.08.36.453 AM
06-JUN-17 11.08.36.453 AM
06-JUN-17 11.08.36.453 AM
06-JUN-17 11.08.36.453 AM
06-JUN-17 11.08.36.453 AM
06-JUN-17 11.08.36.453 AM
```

```
                                        gc buffer busy acquire                30
                                        gc current grant 2-way                61
                                        gc cr grant 2-way                     82
```

```
select sample_time,event,p1,p2,wait_time,time_waited from v$active_session_history
where sample_time>to_date('20170606 11','yyyymmdd hh24')
and sample_time<to_date('20170606 11:20','yyyymmdd hh24:mi')
and session_id=7324
order by sample_time;

06-JUN-17 11.08.34.442 AM   gc current block 2-way   234   3962848      0        0
06-JUN-17 11.08.35.449 AM   gc current block 2-way   234   3962848      0        0
06-JUN-17 11.08.36.453 AM   gc current block 2-way   234   3962848      0        0
06-JUN-17 11.08.37.459 AM   gc current block 2-way   234   3962848      0        0
06-JUN-17 11.08.38.464 AM   gc current block 2-way   234   3962848      0        0
06-JUN-17 11.08.39.469 AM   gc current block 2-way   234   3962848      0        0
06-JUN-17 11.08.40.474 AM   gc current block 2-way   234   3962848      0  7202648
06-JUN-17 11.08.42.492 AM   gc current grant busy    228   4048351      0  1214770
```

一次数据库实例性能分析

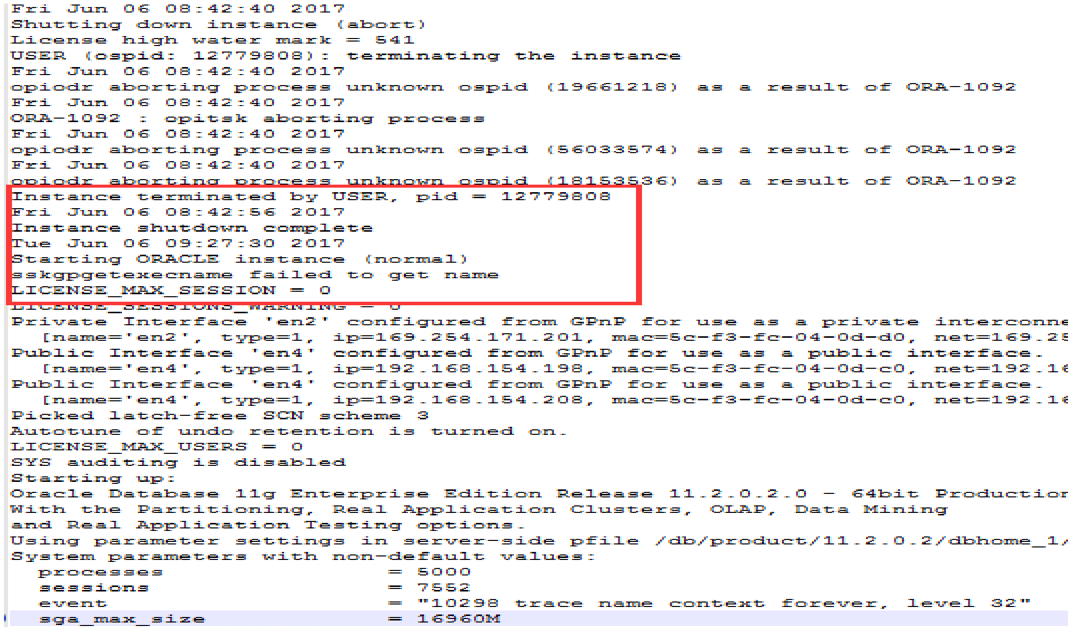
- **Alert日志的信息**
  - 二节点性能出现了问题



```
Thread 2 advanced to log sequence 19574 (LGWR switch)
  Current log# 11 seq# 19574 mem# 0: +ARCHDG/[redacted]od/onlinelog/redo11.log
Tue Jun 06 10:25:52 2017
LNS: Standby redo logfile selected for thread 2 sequence 19574 for destination LOG_ARCHIVE_DEST_2
Tue Jun 06 10:25:53 2017
LNS: Standby redo logfile selected for thread 2 sequence 19574 for destination LOG_ARCHIVE_DEST_3
Tue Jun 06 10:26:16 2017
Archived Log entry 122015 added for thread 2 sequence 19573 ID 0xfffffffd412add4 dest 1:
Tue Jun 06 10:53:16 2017


***********************************************************************
Tue Jun 06 10:57:27 2017

Fatal NI connect error 12537, connecting to:
 (LOCAL=NO)
Tue Jun 06 10:59:19 2017
Tue Jun 06 10:59:38 2017
Errors in file /db/diag/rdbms[redacted]pd2/trace[redacted]_15532226.trc  (incident=5962100):
ORA-00445: background process [redacted] start after [redacted]
Time drift detected. Please check VKTM trace file for more details.

  VERSION INFORMATION:
        TNS for IBM/AIX RISC System/6000: Version 11.2.0.2.0 - Production
        TCP/IP NT Protocol Adapter for IBM/AIX RISC System/6000: Version 11.2.0.2.0 - Production
        Oracle Bequeath NT Protocol Adapter for IBM/AIX RISC System/6000: Version 11.2.0.2.0 - Production
Tue Jun 06 10:57:13 2017
Errors in file /db/diag/rdbms/[redacted]d2/trace/[redacted]2_mmon_32964788.trc  (incident=5961692):
ORA-00445: background process "[redacted] not start after [redacted] seconds
Tue Jun 06 11:00:23 2017
Tue Jun 06 11:00:21 2017
DIAG (ospid: 10682508) waits for event 'DIAG idle wait' for 0 secs.  Time: 06-JUN-2017 11:00:18
Incident details in: /db/diag/rdbms[redacted]d2/incident/incdir_5962100/[redacted]d2_cjq0_15532226_i5962100.trc

Incident details in: /db/diag/rdbms/[redacted]d2/incident/incdir_596169[redacted]d2_mmon_32964788_i5961692.trc
Tue Jun 06 11:01:33 2017
  Tracing not turned on.
  Tns error struct:
    ns main err code: 12537
```

一次数据库实例性能分析

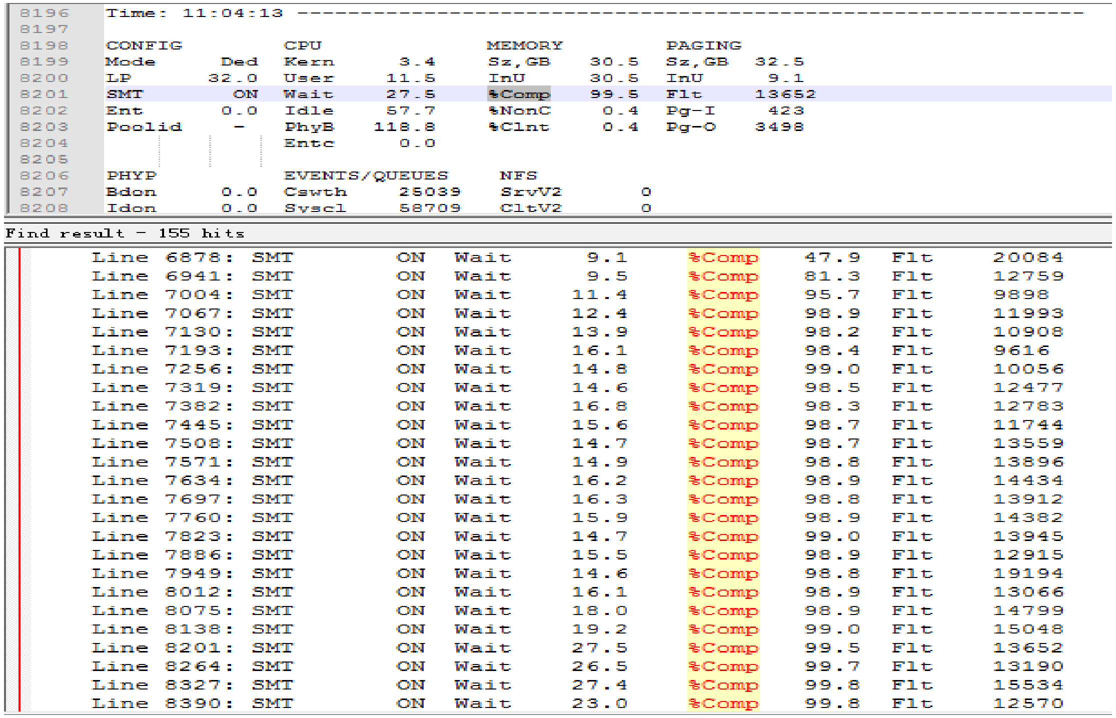
- 隐藏的信息
  - 为啥要重启?
  - 网卡可能存在问题，在开门前赶紧重启排查一下
- 结合前面的特征大概知道原因了吗



```
Fri Jun 06 08:42:40 2017
Shutting down instance (abort)
License high water mark = 541
USER (ospid: 12779808): terminating the instance
Fri Jun 06 08:42:40 2017
opiodr aborting process unknown ospid (19661218) as a result of ORA-1092
Fri Jun 06 08:42:40 2017
ORA-1092 : opitsk aborting process
Fri Jun 06 08:42:40 2017
opiodr aborting process unknown ospid (56033574) as a result of ORA-1092
Fri Jun 06 08:42:40 2017
opiodr aborting process unknown ospid (18153536) as a result of ORA-1092
Instance terminated by USER, pid = 12779808
Fri Jun 06 08:42:56 2017
Instance shutdown complete
Tue Jun 06 09:27:30 2017
Starting ORACLE instance (normal)
sskgpgetexecname failed to get name
LICENSE_MAX_SESSION = 0
LICENSE_SESSIONS_WARNING = 0
Private Interface 'en2' configured from GPnP for use as a private interconne
  [name='en2', type=1, ip=169.254.171.201, mac=5c-f3-fc-04-0d-d0, net=169.25
Public Interface 'en4' configured from GPnP for use as a public interface.
  [name='en4', type=1, ip=192.168.154.198, mac=5c-f3-fc-04-0d-c0, net=192.16
Public Interface 'en4' configured from GPnP for use as a public interface.
  [name='en4', type=1, ip=192.168.154.208, mac=5c-f3-fc-04-0d-c0, net=192.16
Picked latch-free SCN scheme 3
Autotune of undo retention is turned on.
LICENSE_MAX_USERS = 0
SYS auditing is disabled
Starting up:
Oracle Database 11g Enterprise Edition Release 11.2.0.2.0 - 64bit Productio
With the Partitioning, Real Application Clusters, OLAP, Data Mining
and Real Application Testing options.
Using parameter settings in server-side pfile /db/product/11.2.0.2/dbhome_1/
System parameters with non-default values:
  processes                = 5000
  sessions                 = 7552
  event                    = "10298 trace name context forever, level 32"
  sga_max_size             = 16960M
```

一次数据库实例性能分析

- 验证之道
  - 没有oswatcher怎么办？

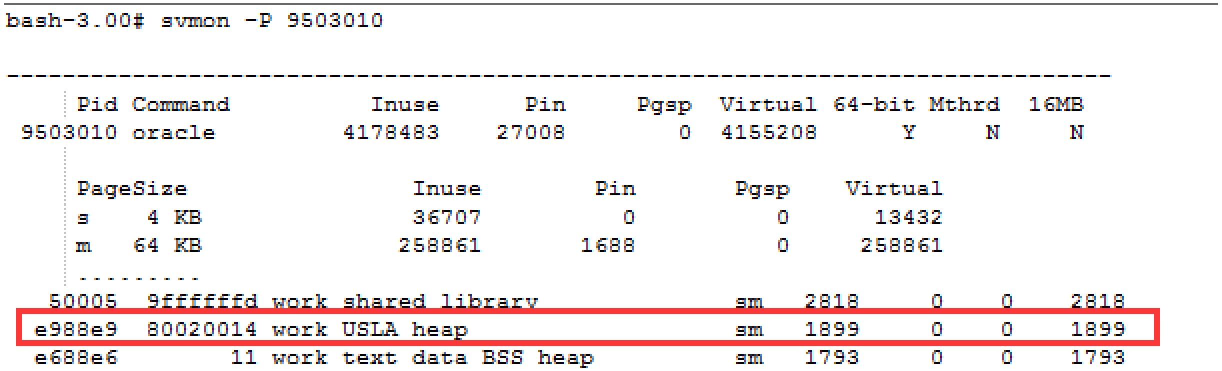


```
8196   Time: 11:04:13 -------------------------------------------------------------------------------
8197
8198   CONFIG              CPU                 MEMORY              PAGING
8199   Mode         Ded    Kern         3.4    Sz,GB        30.5   Sz,GB    32.5
8200   LP          32.0    User        11.5    InU          30.5   InU       9.1
8201   SMT           ON    Wait        27.5    %Comp        99.5   Flt     13652
8202   Ent          0.0    Idle        57.7    %NonC         0.4   Pg-I      423
8203   Poolid         -    PhyB       118.8    %Clnt         0.4   Pg-O     3498
8204                       Entc         0.0
8205
8206   PHYP                EVENTS/QUEUES         NFS
8207   Bdon         0.0    Cswth      25039      SrvV2         0
8208   Idon         0.0    Sysc1      58709      CltV2         0
```

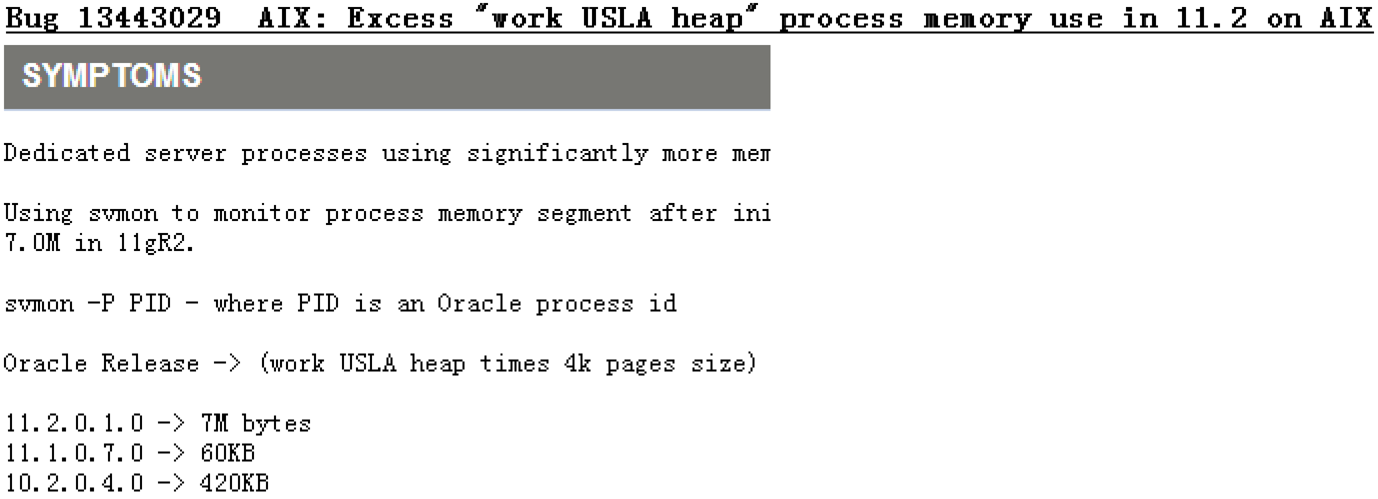
```
Find result - 155 hits
    Line 6878: SMT           ON   Wait        9.1      %Comp     47.9   Flt       20084
    Line 6941: SMT           ON   Wait        9.5      %Comp     81.3   Flt       12759
    Line 7004: SMT           ON   Wait       11.4      %Comp     95.7   Flt       9898
    Line 7067: SMT           ON   Wait       12.4      %Comp     98.9   Flt       11993
    Line 7130: SMT           ON   Wait       13.9      %Comp     98.2   Flt       10908
    Line 7193: SMT           ON   Wait       16.1      %Comp     98.4   Flt       9616
    Line 7256: SMT           ON   Wait       14.8      %Comp     99.0   Flt       10056
    Line 7319: SMT           ON   Wait       14.6      %Comp     98.5   Flt       12477
    Line 7382: SMT           ON   Wait       16.8      %Comp     98.3   Flt       12783
    Line 7445: SMT           ON   Wait       15.6      %Comp     98.7   Flt       11744
    Line 7508: SMT           ON   Wait       14.7      %Comp     98.7   Flt       13559
    Line 7571: SMT           ON   Wait       14.9      %Comp     98.8   Flt       13896
    Line 7634: SMT           ON   Wait       16.2      %Comp     98.9   Flt       14434
    Line 7697: SMT           ON   Wait       16.3      %Comp     98.8   Flt       13912
    Line 7760: SMT           ON   Wait       15.9      %Comp     98.9   Flt       14382
    Line 7823: SMT           ON   Wait       14.7      %Comp     99.0   Flt       13945
    Line 7886: SMT           ON   Wait       15.5      %Comp     98.9   Flt       12915
    Line 7949: SMT           ON   Wait       14.6      %Comp     98.8   Flt       19194
    Line 8012: SMT           ON   Wait       16.1      %Comp     98.9   Flt       13066
    Line 8075: SMT           ON   Wait       18.0      %Comp     98.9   Flt       14799
    Line 8138: SMT           ON   Wait       19.2      %Comp     99.0   Flt       15048
    Line 8201: SMT           ON   Wait       27.5      %Comp     99.5   Flt       13652
    Line 8264: SMT           ON   Wait       26.5      %Comp     99.7   Flt       13190
    Line 8327: SMT           ON   Wait       27.4      %Comp     99.8   Flt       15534
    Line 8390: SMT           ON   Wait       23.0      %Comp     99.8   Flt       12570
```

一次数据库实例性能分析

- 验证之道



```
bash-3.00# svmon -P 9503010

-------------------------------------------------------------------------------
     Pid Command          Inuse      Pin     Pgsp  Virtual 64-bit Mthrd  16MB
 9503010 oracle          4178483    27008        0  4155208      Y     N     N

     PageSize                Inuse        Pin       Pgsp    Virtual
     s    4 KB               36707          0          0      13432
     m   64 KB              258861       1688          0     258861
     ..........
   50005   9ffffffd work shared library             sm    2818      0     0     2818
  e988e9   80020014 work USLA heap                  sm    1899      0     0     1899
  e688e6         11 work text data BSS heap         sm    1793      0     0     1793
```



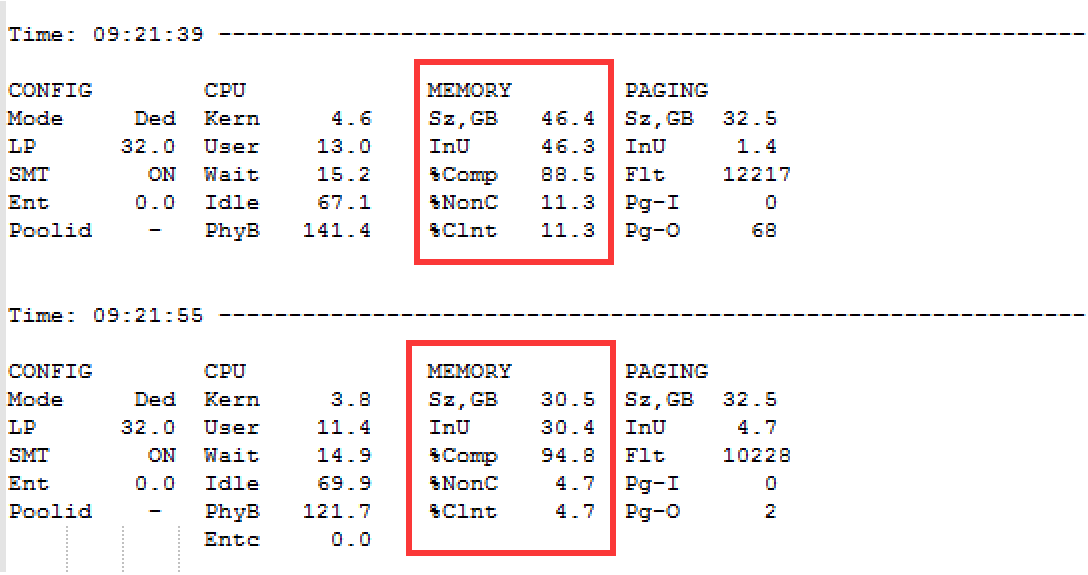
Bug 13443029 AIX: Excess "work USLA heap" process memory use in 11.2 on AIX

SYMPTOMS

```
Dedicated server processes using significantly more mem

Using svmon to monitor process memory segment after ini
7.0M in 11gR2.

svmon -P PID - where PID is an Oracle process id

Oracle Release -> (work USLA heap times 4k pages size)

11.2.0.1.0 -> 7M bytes
11.1.0.7.0 -> 60KB
10.2.0.4.0 -> 420KB
```

一次数据库实例性能分析

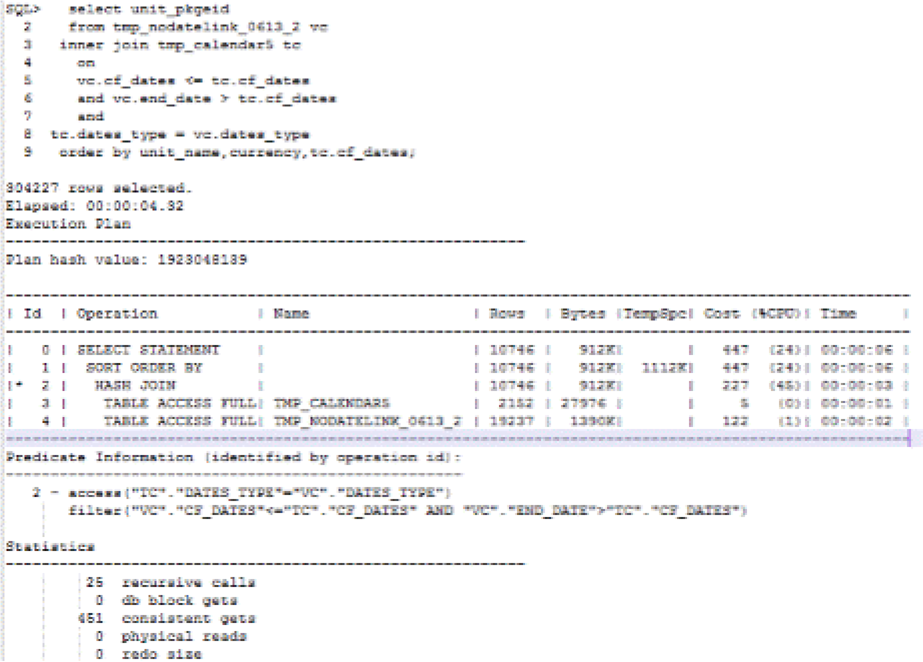
- 建议方案
  - 调整连接池
  - 打上相应的补丁
  - 调小SGA



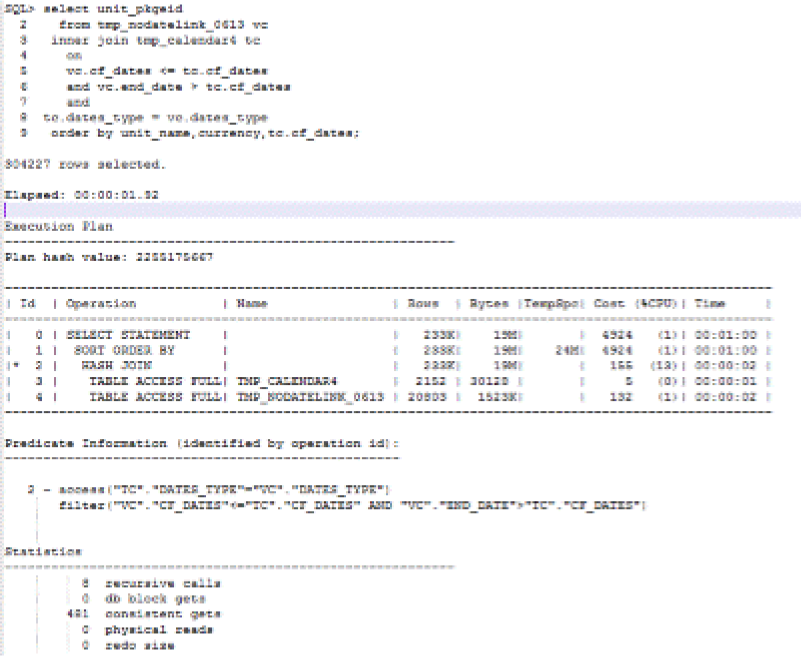
```
Time: 09:21:39 ----------------------------------------------------------------

CONFIG         CPU             MEMORY         PAGING
Mode      Ded  Kern      4.6   Sz,GB    46.4  Sz,GB   32.5
LP       32.0  User     13.0   InU      46.3  InU      1.4
SMT        ON  Wait     15.2   %Comp    88.5  Flt    12217
Ent       0.0  Idle     67.1   %NonC    11.3  Pg-I       0
Poolid      -  PhyB    141.4   %Clnt    11.3  Pg-O      68

Time: 09:21:55 ----------------------------------------------------------------

CONFIG         CPU             MEMORY         PAGING
Mode      Ded  Kern      3.8   Sz,GB    30.5  Sz,GB   32.5
LP       32.0  User     11.4   InU      30.4  InU      4.7
SMT        ON  Wait     14.9   %Comp    94.8  Flt    10228
Ent       0.0  Idle     69.9   %NonC     4.7  Pg-I       0
Poolid      -  PhyB    121.7   %Clnt     4.7  Pg-O       2
               Entc      0.0
```

一条SQL的优化

- 首先来简单的对比一下



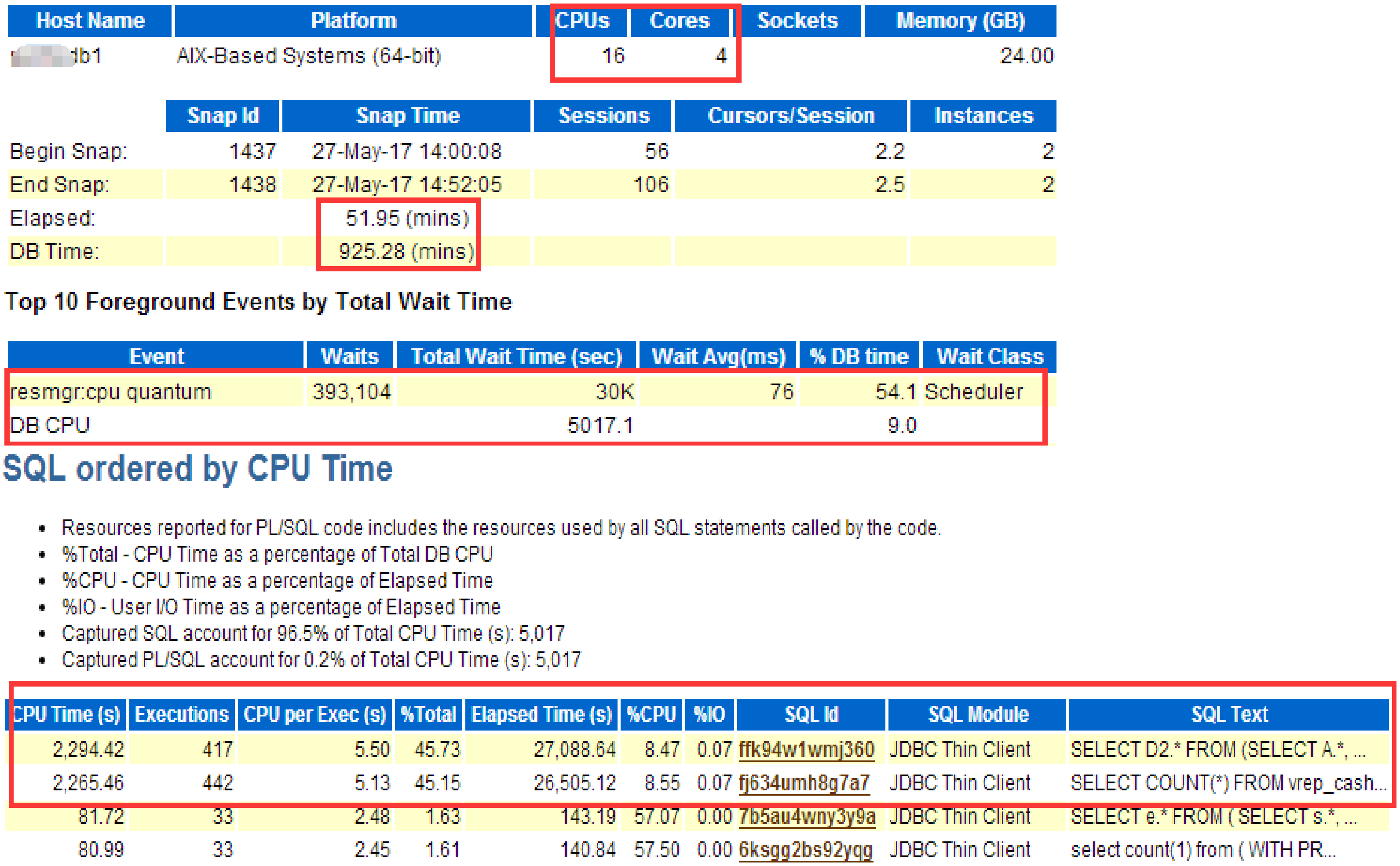
```
SQL>   select unit_pkgeid
  2    from tmp_nodatelink_0613_2 vc
  3   inner join tmp_calendar5 tc
  4     on
  5     vc.cf_dates <= tc.cf_dates
  6     and vc.end_date > tc.cf_dates
  7     and
  8  tc.dates_type = vc.dates_type
  9   order by unit_name,currency,tc.cf_dates;

304227 rows selected.
Elapsed: 00:00:04.32
Execution Plan
----------------------------------------------------------
Plan hash value: 1923048189

------------------------------------------------------------------------------------------------------
| Id  | Operation           | Name                   | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                        | 10746 |   912K|       |   447  (24)| 00:00:06 |
|   1 |  SORT ORDER BY      |                        | 10746 |   912K|  1112K|   447  (24)| 00:00:06 |
|*  2 |   HASH JOIN         |                        | 10746 |   912K|       |   227  (45)| 00:00:03 |
|   3 |    TABLE ACCESS FULL| TMP_CALENDAR5          |  2152 | 27976 |       |     5   (0)| 00:00:01 |
|   4 |    TABLE ACCESS FULL| TMP_NODATELINK_0613_2  | 19237 |  1390K|       |   122   (1)| 00:00:02 |
------------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("TC"."DATES_TYPE"="VC"."DATES_TYPE")
       filter("VC"."CF_DATES"<="TC"."CF_DATES" AND "VC"."END_DATE">"TC"."CF_DATES")

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



```
SQL> select unit_pkgeid
  2    from tmp_nodatelink_0613 vc
  3   inner join tmp_calendar4 tc
  4     on
  5     vc.cf_dates <= tc.cf_dates
  6     and vc.end_date > tc.cf_dates
  7     and
  8  tc.dates_type = vc.dates_type
  9   order by unit_name,currency,tc.cf_dates;

304227 rows selected.

Elapsed: 00:00:01.92

Execution Plan
----------------------------------------------------------
Plan hash value: 2255175667

------------------------------------------------------------------------------------------------------
| Id  | Operation           | Name                   | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                        |   233K|    19M|       |  4924   (1)| 00:01:00 |
|   1 |  SORT ORDER BY      |                        |   233K|    19M|    24M|  4924   (1)| 00:01:00 |
|*  2 |   HASH JOIN         |                        |   233K|    19M|       |   166  (13)| 00:00:02 |
|   3 |    TABLE ACCESS FULL| TMP_CALENDAR4          |  2152 | 30128 |       |     5   (0)| 00:00:01 |
|   4 |    TABLE ACCESS FULL| TMP_NODATELINK_0613    | 20803 |  1523K|       |   132   (1)| 00:00:02 |
------------------------------------------------------------------------------------------------------

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

   2 - access("TC"."DATES_TYPE"="VC"."DATES_TYPE")
       filter("VC"."CF_DATES"<="TC"."CF_DATES" AND "VC"."END_DATE">"TC"."CF_DATES")

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

一条SQL的优化

- 优化的起因



Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
[redacted]db1	AIX-Based Systems (64-bit)	16	4		24.00

	Snap Id	Snap Time	Sessions	Cursors/Session	Instances
Begin Snap:	1437	27-May-17 14:00:08	56	2.2	2
End Snap:	1438	27-May-17 14:52:05	106	2.5	2
Elapsed:		51.95 (mins)			
DB Time:		925.28 (mins)			

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
resmgr:cpu quantum	393,104	30K	76	54.1	Scheduler
DB CPU		5017.1		9.0	

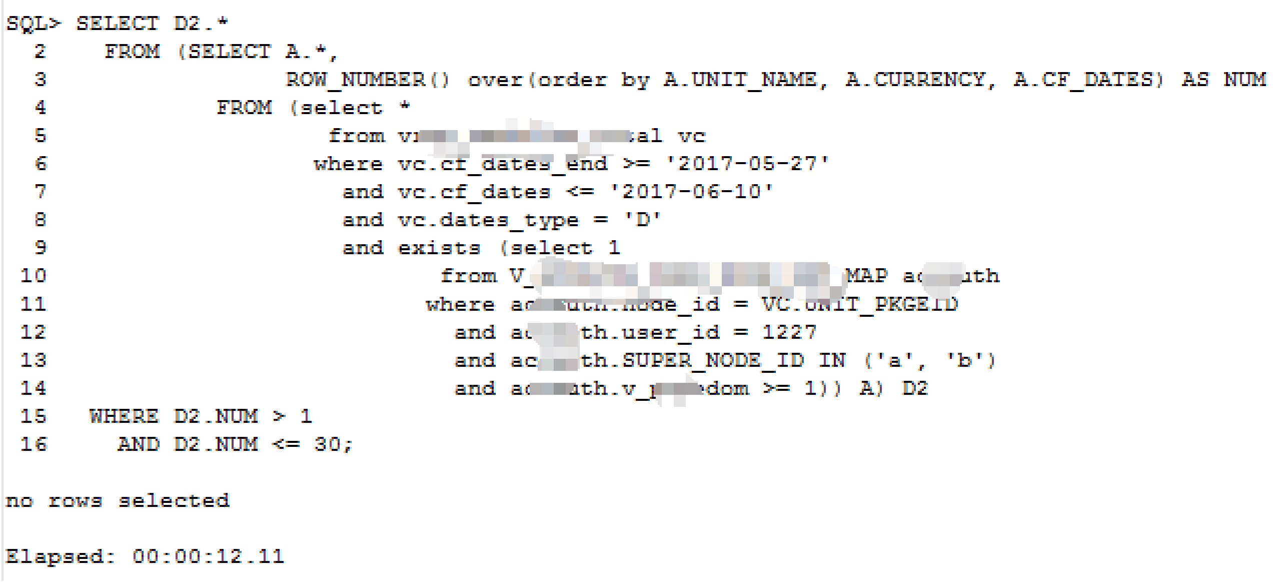
SQL ordered by CPU Time

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code.
- %Total - CPU Time as a percentage of Total DB CPU
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Captured SQL account for 96.5% of Total CPU Time (s): 5,017
- Captured PL/SQL account for 0.2% of Total CPU Time (s): 5,017

CPU Time (s)	Executions	CPU per Exec (s)	%Total	Elapsed Time (s)	%CPU	%IO	SQL Id	SQL Module	SQL Text
2,294.42	417	5.50	45.73	27,088.64	8.47	0.07	ffk94w1wmj360	JDBC Thin Client	SELECT D2.* FROM (SELECT A.*, ...
2,265.46	442	5.13	45.15	26,505.12	8.55	0.07	fj634umh8g7a7	JDBC Thin Client	SELECT COUNT(*) FROM vrep_cash...
81.72	33	2.48	1.63	143.19	57.07	0.00	7b5au4wny3y9a	JDBC Thin Client	SELECT e.* FROM (SELECT s.*, ...
80.99	33	2.45	1.61	140.84	57.50	0.00	6ksgg2bs92yqg	JDBC Thin Client	select count(1) from (WITH PR...

一条SQL的优化

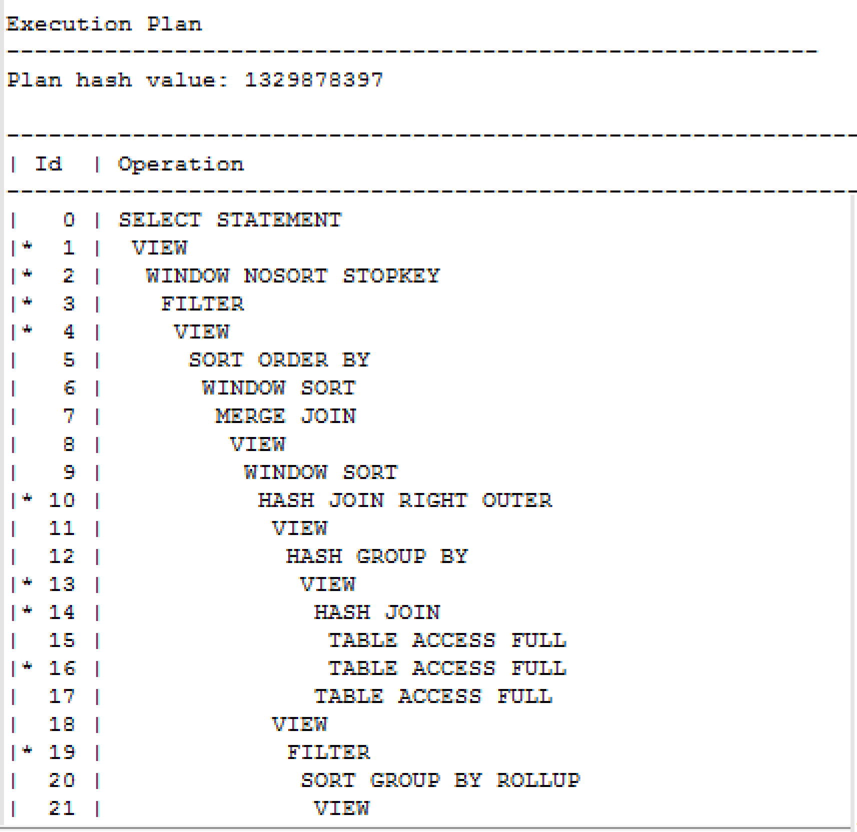
- 原语句是这样的



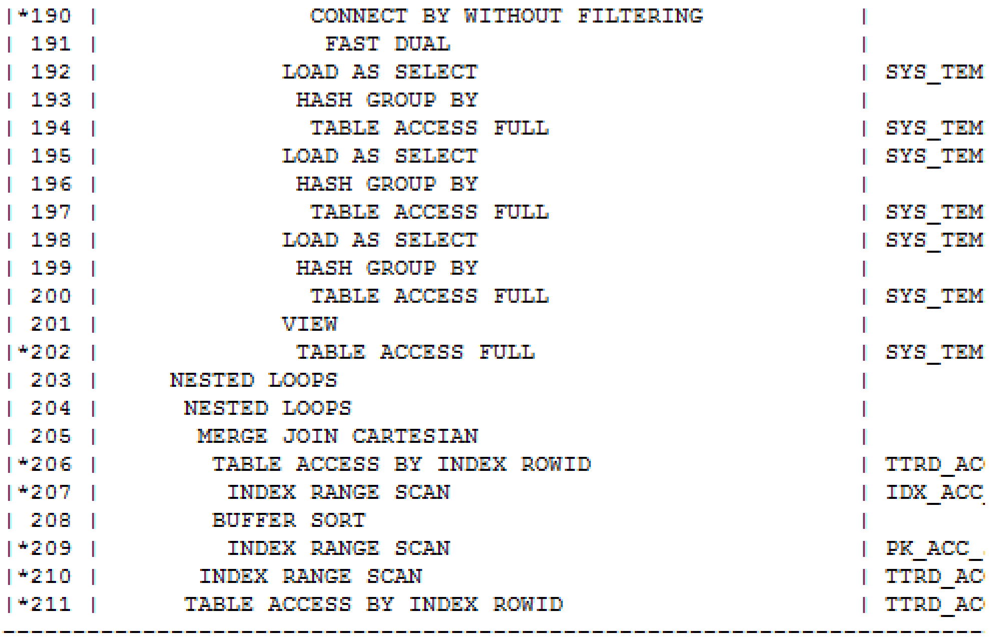
```
SQL> SELECT D2.*
      FROM (SELECT A.*,
                   ROW_NUMBER() over(order by A.UNIT_NAME, A.CURRENCY, A.CF_DATES) AS NUM
              FROM (select *
                      from v[redacted]al vc
                     where vc.cf_dates_end >= '2017-05-27'
                       and vc.cf_dates <= '2017-06-10'
                       and vc.dates_type = 'D'
                       and exists (select 1
                              from V_[redacted]MAP a[redacted]uth
                             where a[redacted]uth.node_id = VC.UNIT_PKGEID
                               and a[redacted]th.user_id = 1227
                               and ac[redacted]th.SUPER_NODE_ID IN ('a', 'b')
                               and a[redacted]th.v_[redacted]edom >= 1)) A) D2
     WHERE D2.NUM > 1
       AND D2.NUM <= 30;

no rows selected

Elapsed: 00:00:12.11
```

一条SQL的优化

- 执行计划是这样的



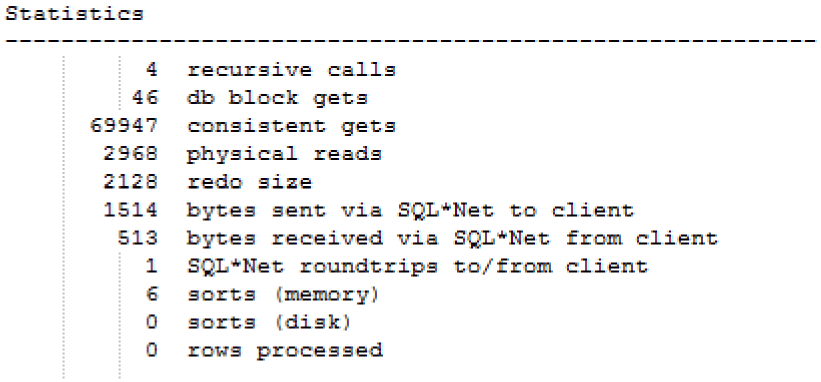
```
Execution Plan
----------------------------------------------------------
Plan hash value: 1329878397

--------------------------------------------------------------------------
| Id  | Operation                                                    | Name
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT
|*  1 |  VIEW
|*  2 |   WINDOW NOSORT STOPKEY
|*  3 |    FILTER
|*  4 |     VIEW
|   5 |      SORT ORDER BY
|   6 |       WINDOW SORT
|   7 |        MERGE JOIN
|   8 |         VIEW
|   9 |          WINDOW SORT
|* 10 |           HASH JOIN RIGHT OUTER
|  11 |            VIEW
|  12 |             HASH GROUP BY
|* 13 |              VIEW
|* 14 |               HASH JOIN
|  15 |                TABLE ACCESS FULL
|* 16 |                TABLE ACCESS FULL
|  17 |               TABLE ACCESS FULL
|  18 |            VIEW
|* 19 |             FILTER
|  20 |              SORT GROUP BY ROLLUP
|  21 |               VIEW
```



```
|*190 |              CONNECT BY WITHOUT FILTERING            |
| 191 |               FAST DUAL                              |
| 192 |            LOAD AS SELECT                            | SYS_TEM
| 193 |             HASH GROUP BY                            |
| 194 |              TABLE ACCESS FULL                       | SYS_TEM
| 195 |            LOAD AS SELECT                            | SYS_TEM
| 196 |             HASH GROUP BY                            |
| 197 |              TABLE ACCESS FULL                       | SYS_TEM
| 198 |            LOAD AS SELECT                            | SYS_TEM
| 199 |             HASH GROUP BY                            |
| 200 |              TABLE ACCESS FULL                       | SYS_TEM
| 201 |            VIEW                                      |
|*202 |             TABLE ACCESS FULL                        | SYS_TEM
| 203 |    NESTED LOOPS                                      |
| 204 |     NESTED LOOPS                                     |
| 205 |      MERGE JOIN CARTESIAN                            |
|*206 |       TABLE ACCESS BY INDEX ROWID                    | TTRD_AC
|*207 |        INDEX RANGE SCAN                              | IDX_ACC
| 208 |       BUFFER SORT                                    |
|*209 |        INDEX RANGE SCAN                              | PK_ACC_
|*210 |      INDEX RANGE SCAN                                | TTRD_AC
|*211 |     TABLE ACCESS BY INDEX ROWID                      | TTRD_AC
----------------------------------------------------------------------
```

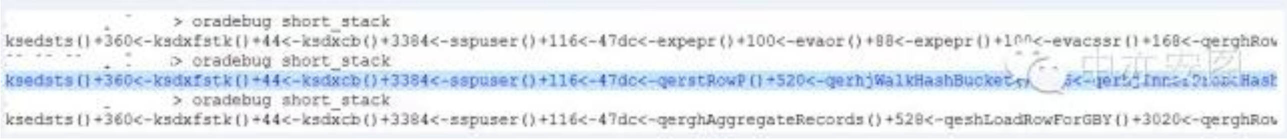
一条SQL的优化

- 手动执行统计信息是这样的



```
Statistics
----------------------------------------------------------
          4  recursive calls
         46  db block gets
      69947  consistent gets
       2968  physical reads
       2128  redo size
       1514  bytes sent via SQL*Net to client
        513  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          6  sorts (memory)
          0  sorts (disk)
          0  rows processed
```

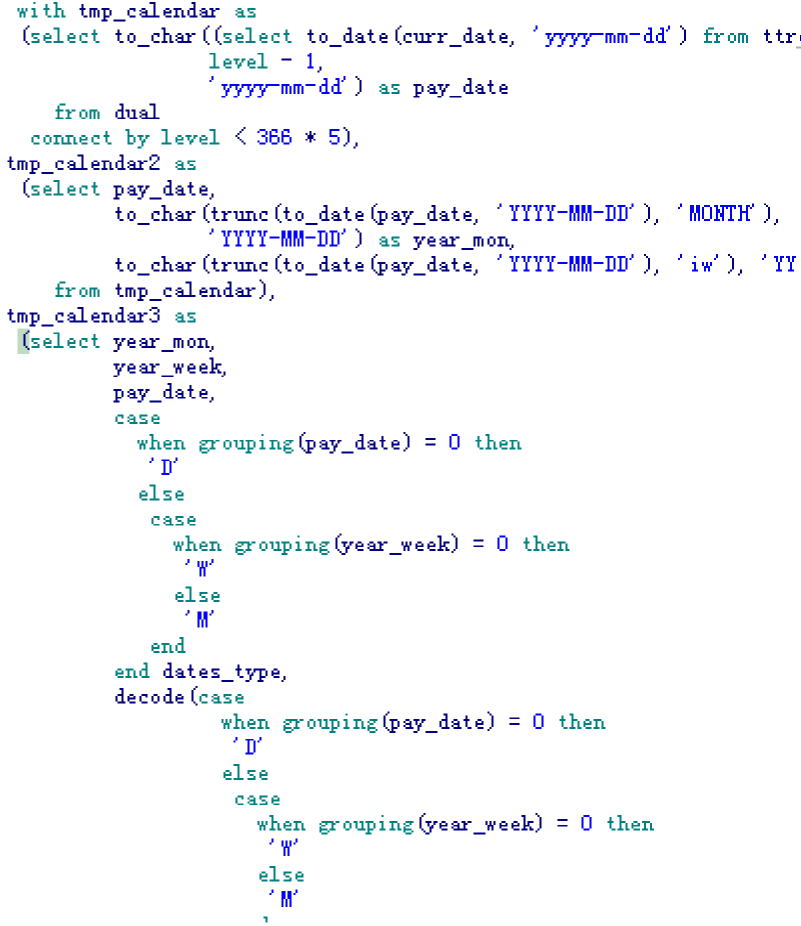
- 特征是什么？
  - 使用了视图/分页排序
  - 逻辑读并不大（69947个逻辑读）
  - 执行时间较长



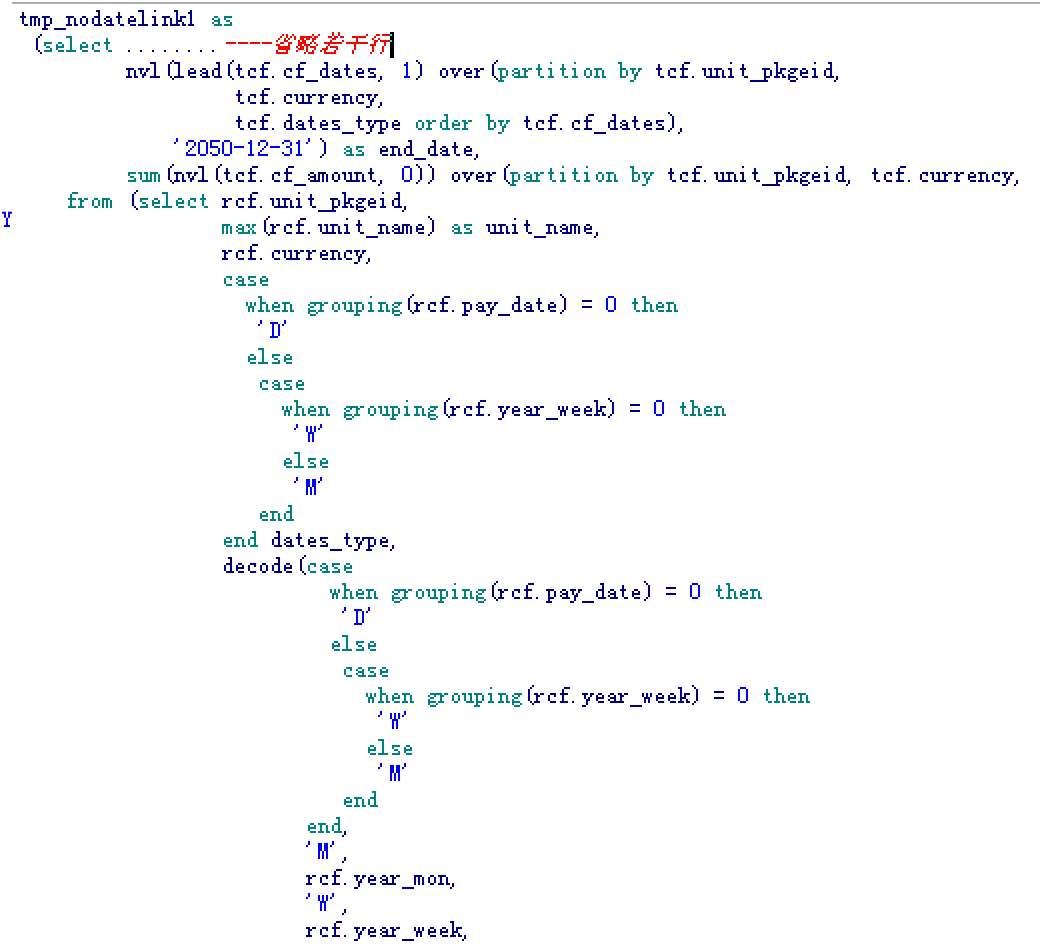
```
             > oradebug short_stack
ksedsts()+360<-ksdxfstk()+44<-ksdxcb()+3384<-sspuser()+116<-47dc<-expepr()+100<-evaor()+88<-expepr()+100<-evacssr()+168<-qerghRow
             > oradebug short_stack
ksedsts()+360<-ksdxfstk()+44<-ksdxcb()+3384<-sspuser()+116<-47dc<-qerstRowP()+520<-qerhjWalkHashBucket()+[illegible]5<-qerhjInnerProbeHash
             > oradebug short_stack
ksedsts()+360<-ksdxfstk()+44<-ksdxcb()+3384<-sspuser()+116<-47dc<-qerghAggregateRecords()+528<-qeshLoadRowForGBY()+3020<-qerghRow
```

一条SQL的优化

- SQL执行过程中最重要的那个hash join



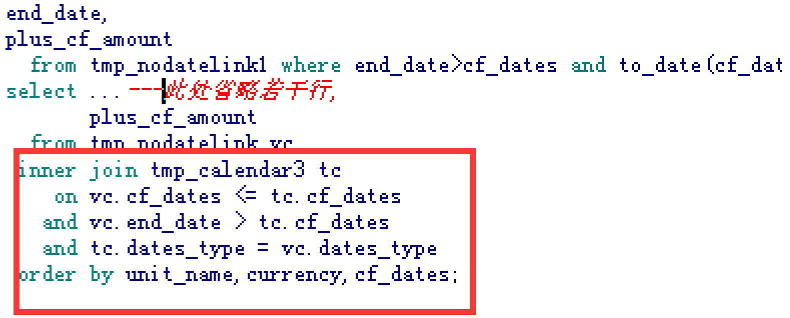
```
 with tmp_calendar as
 (select to_char((select to_date(curr_date, 'yyyy-mm-dd') from ttrd currdate) +
                 level - 1,
                 'yyyy-mm-dd') as pay_date
    from dual
  connect by level < 366 * 5),
tmp_calendar2 as
 (select pay_date,
         to_char(trunc(to_date(pay_date, 'YYYY-MM-DD'), 'MONTH'),
                 'YYYY-MM-DD') as year_mon,
         to_char(trunc(to_date(pay_date, 'YYYY-MM-DD'), 'iw'), 'YY
    from tmp_calendar),
tmp_calendar3 as
 (select year_mon,
         year_week,
         pay_date,
         case
           when grouping(pay_date) = 0 then
            'D'
           else
            case
              when grouping(year_week) = 0 then
               'W'
              else
               'M'
            end
         end dates_type,
         decode(case
                  when grouping(pay_date) = 0 then
                   'D'
                  else
                   case
                     when grouping(year_week) = 0 then
                      'W'
                     else
                      'M'
                     '
```



```
tmp_nodatelink1 as
 (select ........----省略若干行
        nvl(lead(tcf.cf_dates, 1) over(partition by tcf.unit_pkgeid,
                 tcf.currency,
                 tcf.dates_type order by tcf.cf_dates),
            '2050-12-31') as end_date,
        sum(nvl(tcf.cf_amount, 0)) over(partition by tcf.unit_pkgeid, tcf.currency,
   from (select rcf.unit_pkgeid,
                max(rcf.unit_name) as unit_name,
                rcf.currency,
                case
                  when grouping(rcf.pay_date) = 0 then
                   'D'
                  else
                   case
                     when grouping(rcf.year_week) = 0 then
                      'W'
                     else
                      'M'
                   end
                end dates_type,
                decode(case
                         when grouping(rcf.pay_date) = 0 then
                          'D'
                         else
                          case
                            when grouping(rcf.year_week) = 0 then
                             'W'
                            else
                             'M'
                          end
                       end,
                       'M',
                       rcf.year_mon,
                       'W',
                       rcf.year_week,
```

一条SQL的优化

- **SQL执行过程中最重要的那个hash join**

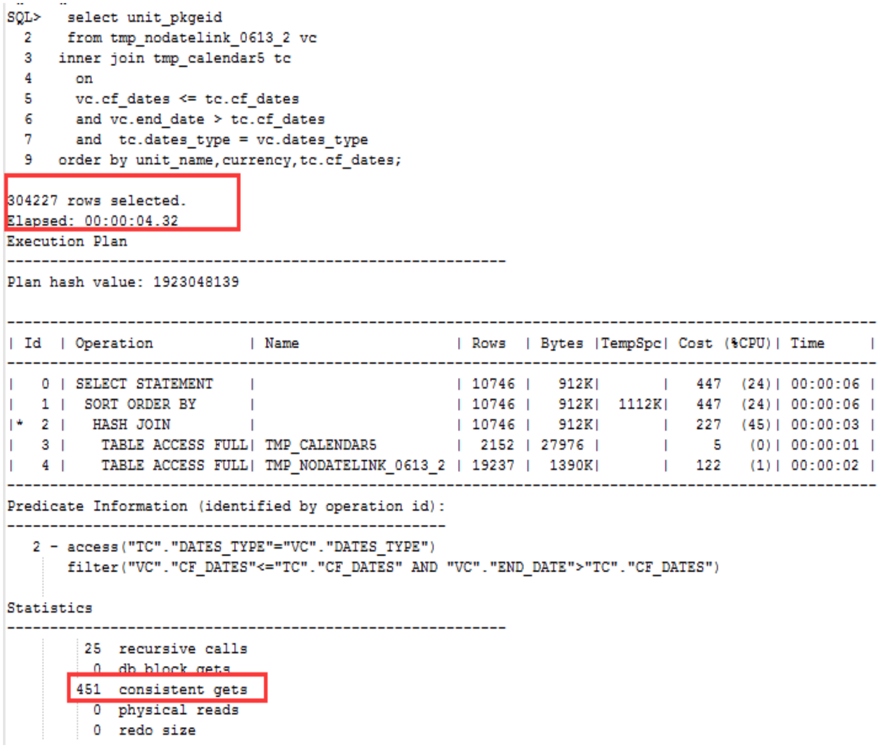


```
end_date,
plus_cf_amount
  from tmp_nodatelink1 where end_date>cf_dates and to_date(cf_dat
select ... ---此处省略若干行,
       plus_cf_amount
  from tmp_nodatelink vc
inner join tmp_calendar3 tc
   on vc.cf_dates <= tc.cf_dates
  and vc.end_date > tc.cf_dates
  and tc.dates_type = vc.dates_type
order by unit_name,currency,cf_dates;
```

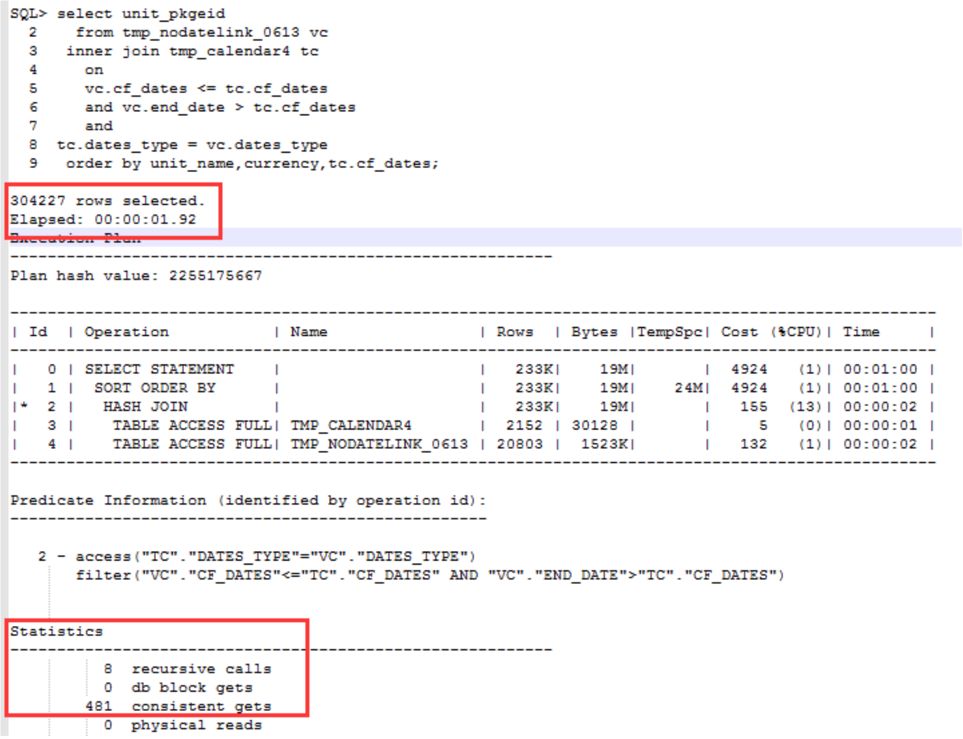
- 最重要的几点：
  - **理解这里的dates_type**
  - 一个等值关联
  - 两个非等值关联

一条SQL的优化

- 回顾这个对比



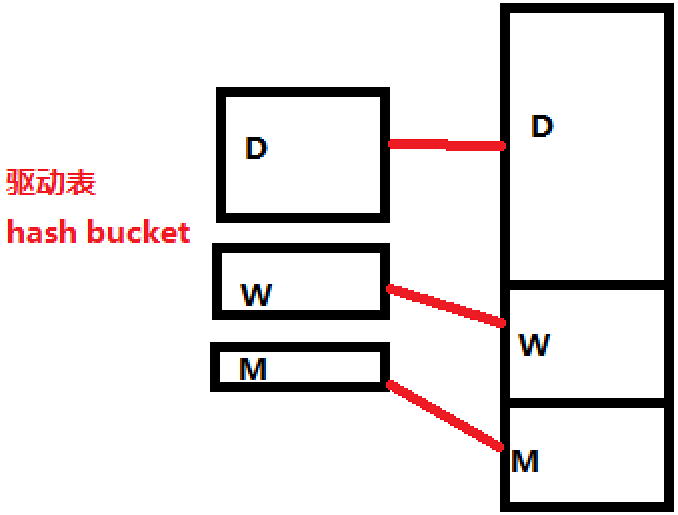
```
SQL>  select unit_pkgeid
  2   from tmp_nodatelink_0613_2 vc
  3  inner join tmp_calendar5 tc
  4    on
  5    vc.cf_dates <= tc.cf_dates
  6    and vc.end_date > tc.cf_dates
  7    and  tc.dates_type = vc.dates_type
  9  order by unit_name,currency,tc.cf_dates;

304227 rows selected.
Elapsed: 00:00:04.32
Execution Plan
----------------------------------------------------------

Plan hash value: 1923048139

-----------------------------------------------------------------------------------------------------
| Id  | Operation           | Name                  | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                       | 10746 |   912K|       |   447  (24)| 00:00:06 |
|   1 |  SORT ORDER BY      |                       | 10746 |   912K|  1112K|   447  (24)| 00:00:06 |
|*  2 |   HASH JOIN         |                       | 10746 |   912K|       |   227  (45)| 00:00:03 |
|   3 |    TABLE ACCESS FULL| TMP_CALENDAR5         |  2152 | 27976 |       |     5   (0)| 00:00:01 |
|   4 |    TABLE ACCESS FULL| TMP_NODATELINK_0613_2 | 19237 |  1390K|       |   122   (1)| 00:00:02 |
-----------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("TC"."DATES_TYPE"="VC"."DATES_TYPE")
       filter("VC"."CF_DATES"<="TC"."CF_DATES" AND "VC"."END_DATE">"TC"."CF_DATES")

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



```
SQL> select unit_pkgeid
  2    from tmp_nodatelink_0613 vc
  3   inner join tmp_calendar4 tc
  4     on
  5     vc.cf_dates <= tc.cf_dates
  6     and vc.end_date > tc.cf_dates
  7     and
  8  tc.dates_type = vc.dates_type
  9   order by unit_name,currency,tc.cf_dates;

304227 rows selected.
Elapsed: 00:00:01.92
Execution Plan
----------------------------------------------------------

Plan hash value: 2255175667

-----------------------------------------------------------------------------------------------------
| Id  | Operation           | Name                  | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                       |   233K|    19M|       |  4924   (1)| 00:01:00 |
|   1 |  SORT ORDER BY      |                       |   233K|    19M|    24M|  4924   (1)| 00:01:00 |
|*  2 |   HASH JOIN         |                       |   233K|    19M|       |   155  (13)| 00:00:02 |
|   3 |    TABLE ACCESS FULL| TMP_CALENDAR4         |  2152 | 30128 |       |     5   (0)| 00:00:01 |
|   4 |    TABLE ACCESS FULL| TMP_NODATELINK_0613   | 20803 |  1523K|       |   132   (1)| 00:00:02 |
-----------------------------------------------------------------------------------------------------

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

   2 - access("TC"."DATES_TYPE"="VC"."DATES_TYPE")
       filter("VC"."CF_DATES"<="TC"."CF_DATES" AND "VC"."END_DATE">"TC"."CF_DATES")

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

一条SQL的优化

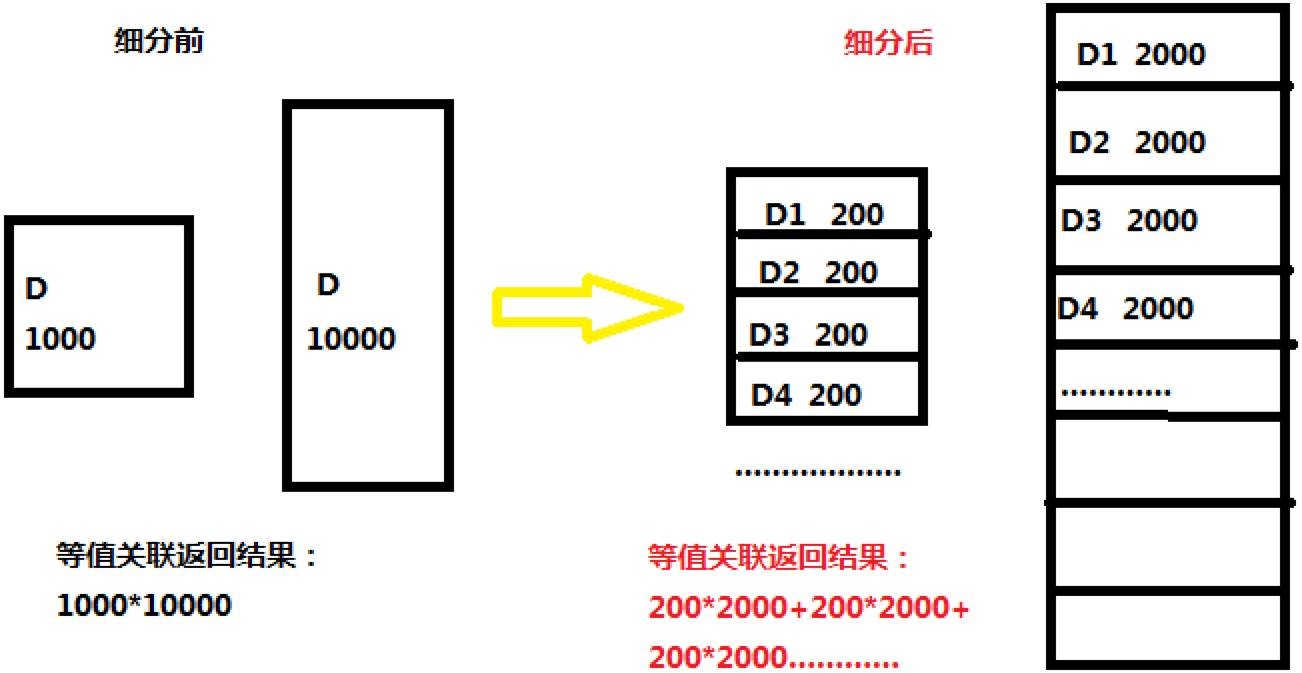
- Hash join的原理？
  - 分桶
  - join



假设驱动表 记录分布为
D值 1000
W值 100
M值 10
被驱动表记录分布为
D值10000
W值1000
M值100
那么dates_type等值关联后的记录数是：
1000*10000+100*1000+10*100

一条SQL的优化

- **hash的优化**
  - 如果我们把**D/W/M**分的更细呢？
    - 比如将**D**细分为**D1**、**D2**、**D3…**
    - 为什么更细会更好



细分前

D
1000

D
10000

等值关联返回结果：
1000*10000

细分后

D1 200
D2 200
D3 200
D4 200
………………

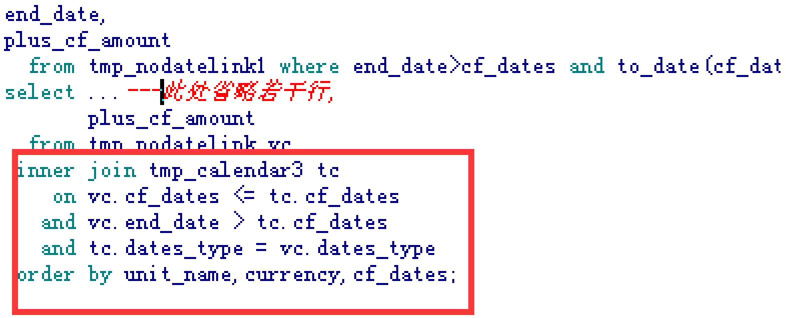
D1 2000
D2 2000
D3 2000
D4 2000
…………

等值关联返回结果：
200*2000+200*2000+
200*2000…………

一条SQL的优化

- 真的有那么理想吗
  - 改写的前提是什么
    - 不能改变结果集
  - 带来了什么问题
    - 驱动表可以像前面那样简单的分组（分桶）
    - 被驱动表需要保证落到同样的桶中
    - 意味着被驱动表的D1+D2+D3……>原D
  - 最理想的情况&最不理想的情况
    - 如果只有等值关联条件，我们还能细分吗？
    - 分组时，应该还与另外的关联条件有关

一条SQL的优化

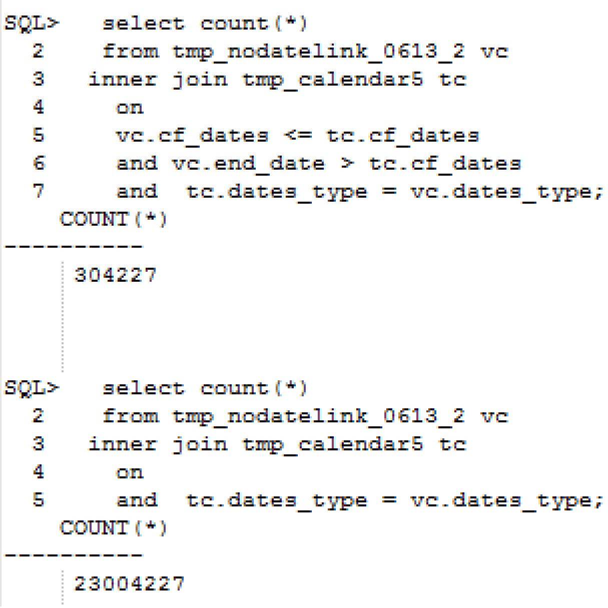


```
end_date,
plus_cf_amount
  from tmp_nodatelink1 where end_date>cf_dates and to_date(cf_dat
select ... ---此处省略若干行,
       plus_cf_amount
  from tmp_nodatelink vc
inner join tmp_calendar3 tc
   on vc.cf_dates <= tc.cf_dates
  and vc.end_date > tc.cf_dates
  and tc.dates_type = vc.dates_type
order by unit_name,currency,cf_dates;
```

- 最重要的几点：
  - **理解这里的dates_type**
  - 一个等值关联
  - 两个非等值关联

一条SQL的优化

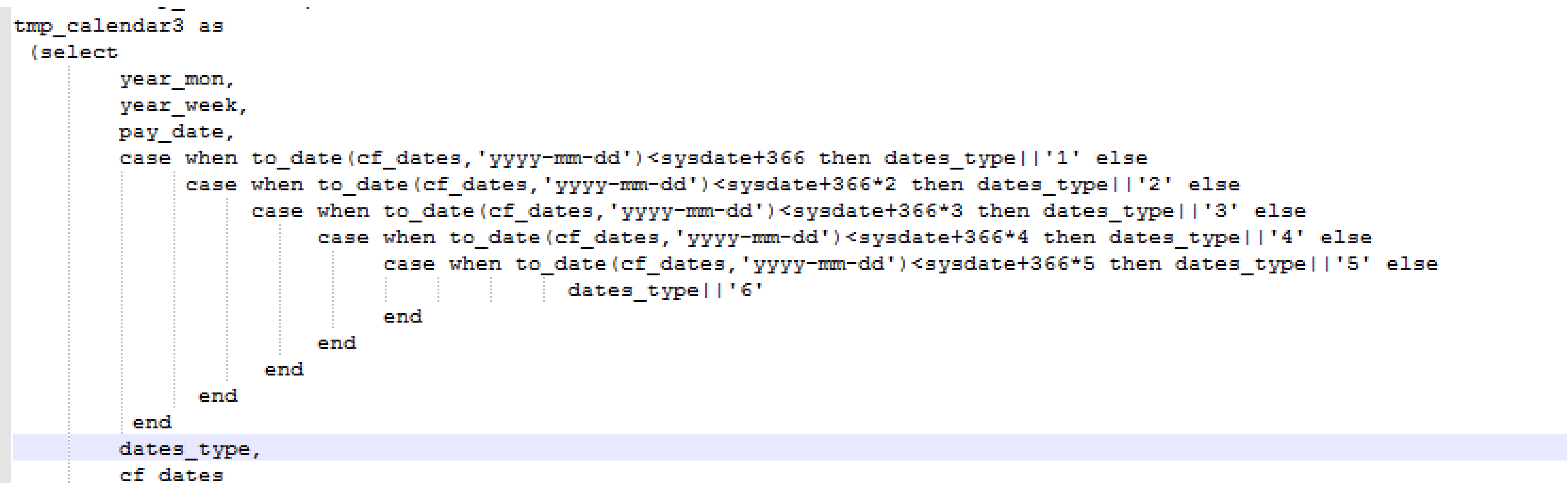
- 有优化空间的前提
  - 如果只有相等条件，结果集很大
  - 加上另一个非等值连接，
    返回的结果集就没有那么大了，过滤了大部分数据
  - 如果我们在hash join的时候就做到这一步呢



```
SQL>   select count(*)
  2    from tmp_nodatelink_0613_2 vc
  3   inner join tmp_calendar5 tc
  4     on
  5     vc.cf_dates <= tc.cf_dates
  6     and vc.end_date > tc.cf_dates
  7     and  tc.dates_type = vc.dates_type;
  COUNT(*)
----------
    304227


SQL>   select count(*)
  2    from tmp_nodatelink_0613_2 vc
  3   inner join tmp_calendar5 tc
  4     on
  5     and  tc.dates_type = vc.dates_type;
  COUNT(*)
----------
    23004227
```

一条SQL的优化

- 实现它（驱动表的改写，很简单）

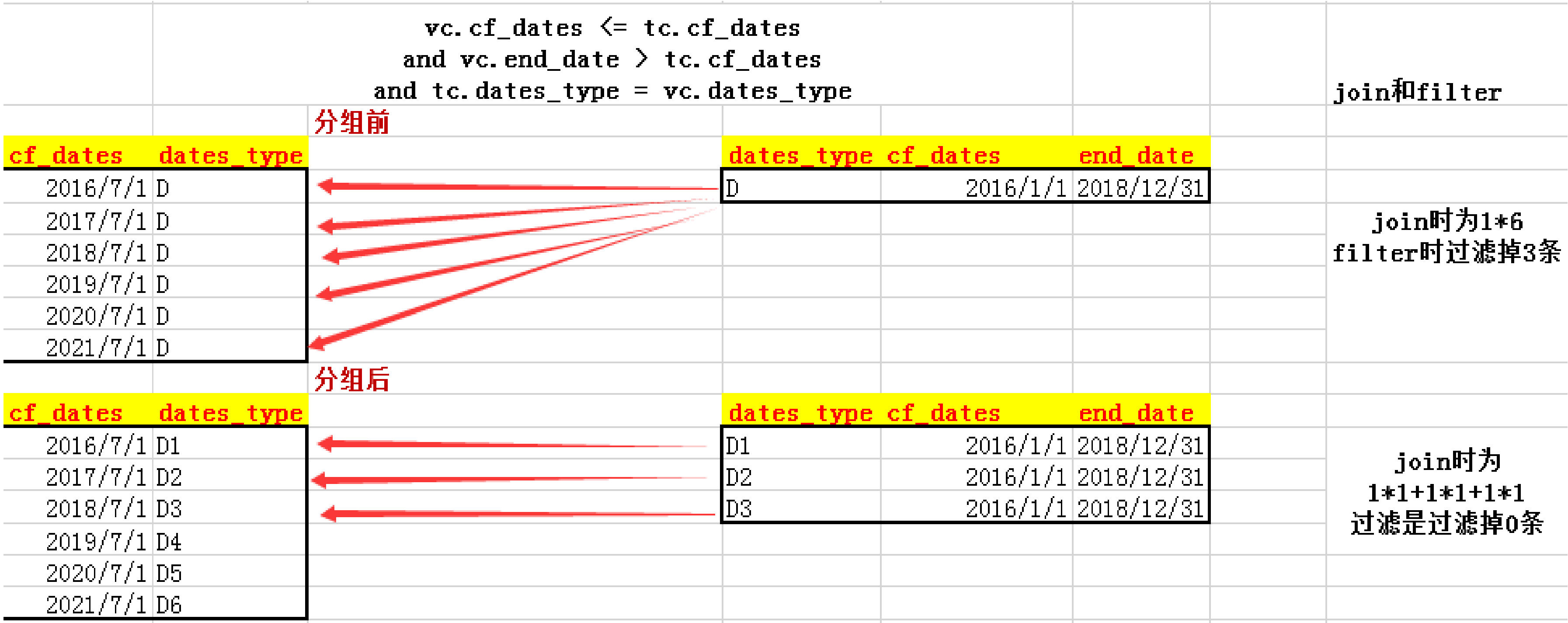


```
tmp_calendar3 as
 (select
        year_mon,
        year_week,
        pay_date,
        case when to_date(cf_dates,'yyyy-mm-dd')<sysdate+366 then dates_type||'1' else
             case when to_date(cf_dates,'yyyy-mm-dd')<sysdate+366*2 then dates_type||'2' else
                  case when to_date(cf_dates,'yyyy-mm-dd')<sysdate+366*3 then dates_type||'3' else
                       case when to_date(cf_dates,'yyyy-mm-dd')<sysdate+366*4 then dates_type||'4' else
                            case when to_date(cf_dates,'yyyy-mm-dd')<sysdate+366*5 then dates_type||'5' else
                                          dates_type||'6'
                            end
                       end
                   end
              end
         end
        dates_type,
        cf_dates
```

- 原with部分不变，外层再套一个select来实现分组

一条SQL的优化

- 被驱动表的改写分析



```
vc.cf_dates <= tc.cf_dates
and vc.end_date > tc.cf_dates
and tc.dates_type = vc.dates_type
```

join和filter

分组前

cf_dates	dates_type
2016/7/1	D
2017/7/1	D
2018/7/1	D
2019/7/1	D
2020/7/1	D
2021/7/1	D

dates_type	cf_dates	end_date
D	2016/1/1	2018/12/31

join时为1*6
filter时过滤掉3条

分组后

cf_dates	dates_type
2016/7/1	D1
2017/7/1	D2
2018/7/1	D3
2019/7/1	D4
2020/7/1	D5
2021/7/1	D6

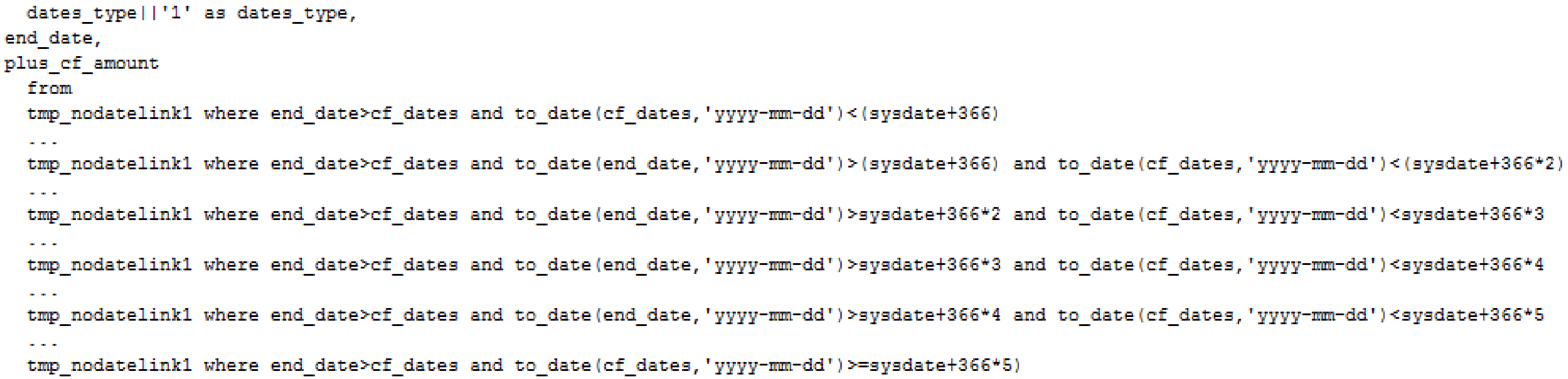
dates_type	cf_dates	end_date
D1	2016/1/1	2018/12/31
D2	2016/1/1	2018/12/31
D3	2016/1/1	2018/12/31

join时为
1*1+1*1+1*1
过滤是过滤掉0条

- 最好的情况&最差的情况&为何这里可以改
- 原with部分不变，外层再套一个select并使用union来实现

一条SQL的优化

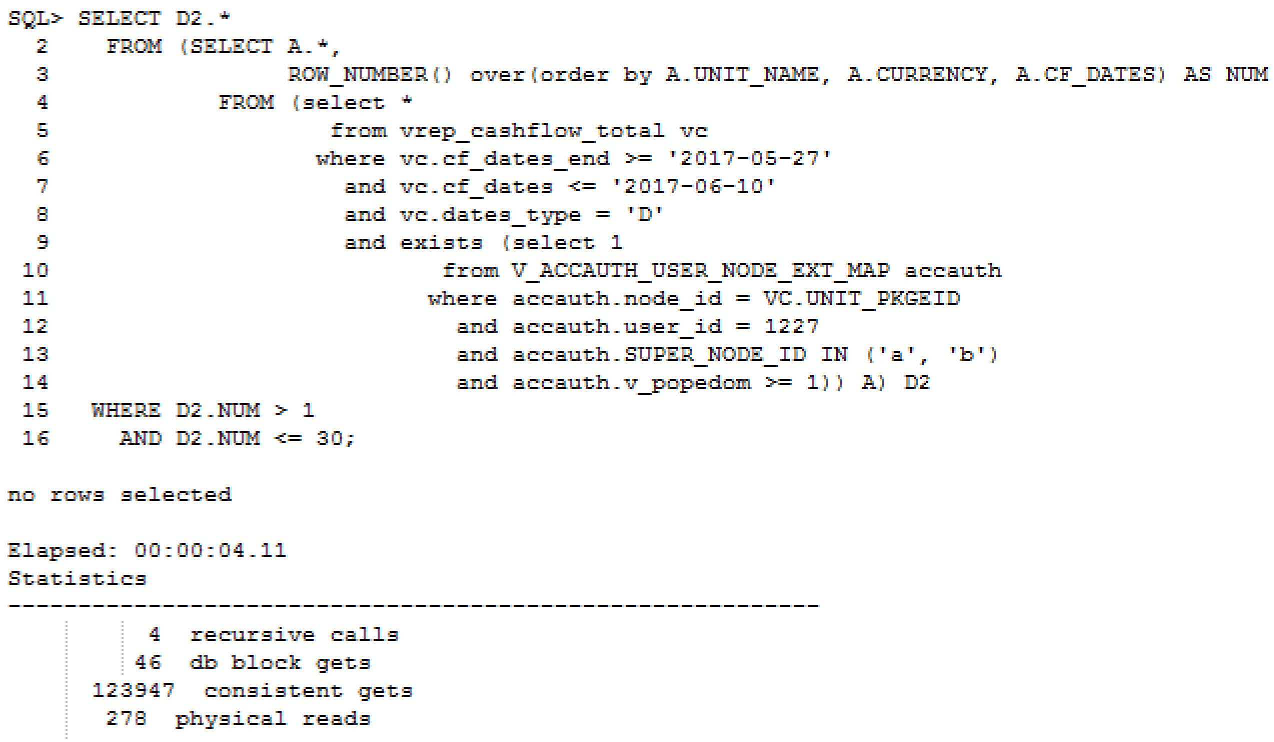
- 实现它（被驱动表的改写，不那么简单）



```
  dates_type||'1' as dates_type,
end_date,
plus_cf_amount
  from
  tmp_nodatelink1 where end_date>cf_dates and to_date(cf_dates,'yyyy-mm-dd')<(sysdate+366)
  ...
  tmp_nodatelink1 where end_date>cf_dates and to_date(end_date,'yyyy-mm-dd')>(sysdate+366) and to_date(cf_dates,'yyyy-mm-dd')<(sysdate+366*2)
  ...
  tmp_nodatelink1 where end_date>cf_dates and to_date(end_date,'yyyy-mm-dd')>sysdate+366*2 and to_date(cf_dates,'yyyy-mm-dd')<sysdate+366*3
  ...
  tmp_nodatelink1 where end_date>cf_dates and to_date(end_date,'yyyy-mm-dd')>sysdate+366*3 and to_date(cf_dates,'yyyy-mm-dd')<sysdate+366*4
  ...
  tmp_nodatelink1 where end_date>cf_dates and to_date(end_date,'yyyy-mm-dd')>sysdate+366*4 and to_date(cf_dates,'yyyy-mm-dd')<sysdate+366*5
  ...
  tmp_nodatelink1 where end_date>cf_dates and to_date(cf_dates,'yyyy-mm-dd')>=sysdate+366*5)
```

一条SQL的优化

- 最后的结果（逻辑读变大，执行时间变短）



```
SQL> SELECT D2.*
  2    FROM (SELECT A.*,
  3                 ROW_NUMBER() over(order by A.UNIT_NAME, A.CURRENCY, A.CF_DATES) AS NUM
  4            FROM (select *
  5                    from vrep_cashflow_total vc
  6                   where vc.cf_dates_end >= '2017-05-27'
  7                     and vc.cf_dates <= '2017-06-10'
  8                     and vc.dates_type = 'D'
  9                     and exists (select 1
 10                            from V_ACCAUTH_USER_NODE_EXT_MAP accauth
 11                           where accauth.node_id = VC.UNIT_PKGEID
 12                             and accauth.user_id = 1227
 13                             and accauth.SUPER_NODE_ID IN ('a', 'b')
 14                             and accauth.v_popedom >= 1)) A) D2
 15   WHERE D2.NUM > 1
 16     AND D2.NUM <= 30;

no rows selected

Elapsed: 00:00:04.11
Statistics
----------------------------------------------------------
          4   recursive calls
         46   db block gets
     123947  consistent gets
      278  physical reads
```

完

- 欢迎继续关注技术人生系列