```
[oracle@localhost ~]$ lsb_release -a
LSB Version:      :core-3.1-ia32:core-3.1-noarch:graphics-3.1-ia32:graphics-3.1-noarch
Distributor ID: EnterpriseEnterpriseServer
Description:      Enterprise Linux Enterprise Linux Server release 5.5 (Carthage)
Release:            5.5
Codename:          Carthage

SQL> select * from v$version where rownum < 2;
BANNER
--------------------------------------------------------------------------------
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
SQL> startup mount;
ORACLE 例程已经启动。
SQL> alter session set sql_trace = true;
会话已更改。
SQL> alter database open;
数据库已更改。

SQL> select * from v$diag_info where NAME='Default Trace File';


   INST_ID NAME
---------- --------------------------------------------------------------
VALUE
--------------------------------------------------------------------------------
         1 Default Trace File
/u01/app/oracle/diag/rdbms/orcl3939/orcl3939/trace/orcl3939_ora_14443.trc
```

11g 提供了 v$diag_info,通过查询 *V$diag_info* 可以很容易找到自身服务进程的 *trace* 文件位置。

当然由 oracle 生成 trace 文件的特性：

```
SQL> select distinct sid from v$mystat;

    SID
----------
         9

SQL> select paddr from v$session where sid=9;

PADDR
--------
34FC8DAC

SQL> select spid from v$process where addr='34FC8DAC';

SPID
```

14443

也很快在茫茫的 trace 文件中找到我们想要的文件。

```
*** 2015-04-26 00:28:20.034
*** SESSION ID:(9.3) 2015-04-26 00:28:20.034
*** CLIENT ID:() 2015-04-26 00:28:20.034
*** SERVICE NAME:() 2015-04-26 00:28:20.034
*** MODULE NAME:(sqlplus@localhost.localdomain (TNS V1-V3)) 2015-04-26 00:28:20.034
*** ACTION NAME:() 2015-04-26 00:28:20.034
  =====================
PARSING IN CURSOR #1 len=34 dep=0 uid=0 oct=42 lid=0 tim=1429979299982435 hv=2069488880
ad='73fed0'  sqlid='lhgzr5xxpmt7h'
alter session set sql_trace = true
END OF STMT
EXEC #1:c=0,e=143,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=0,tim=1429979299933691
*** 2015-04-26 00:28:44.514
CLOSE #1:c=0,e=12,dep=0,type=0,tim=1429979324514259
*** 2015-04-26 00:28:44.516
XCTEND rlbk=0, rd_only=1, tim=1429979324516325
=====================
PARSING IN CURSOR #1 len=19 dep=0 uid=0 oct=35 lid=0 tim=1429979324552260 hv=1907384048
ad='349dcfe4'  sqlid='a01hp0psv0rrh'
alter database open
END OF STMT
PARSE #1:c=3999,e=36299,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=0,tim=1429979324552257
=====================
PARSING IN CURSOR #2 len=188 dep=1 uid=0 oct=1 lid=0 tim=1429979324829034 hv=4006182593
ad='349db650'  sqlid='32r4f1brckzq1'
create table bootstrap$ ( line#             number not null,     obj#                  number not
null,     sql_text     varchar2(4000) not null)     storage (initial 50K objno 59 extents (file 1
block 520))
END OF STMT
PARSE #2:c=1000,e=926,p=0,cr=0,cu=0,mis=1,r=0,dep=1,og=4,plh=0,tim=1429979324829031
```

```
EXEC #2:c=0,e=378,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,plh=0,tim=1429979324829510
CLOSE #2:c=0,e=9,dep=1,type=0,tim=1429979324829642
=====================
PARSING IN CURSOR #2 len=55 dep=1 uid=0 oct=3 lid=0 tim=1429979324830543 hv=2111436465
ad='349da4f8' sqlid='6apq2rjyxmxpj'
select line#, sql_text from bootstrap$ where obj# != :1
END OF STMT
PARSE #2:c=999,e=878,p=0,cr=0,cu=0,mis=1,r=0,dep=1,og=4,plh=0,tim=1429979324830540
EXEC #2:c=3000,e=59167,p=0,cr=0,cu=0,mis=1,r=0,dep=1,og=4,plh=867914364,tim=1429979324889837
FETCH #2:c=1000,e=1299,p=4,cr=3,cu=0,mis=0,r=1,dep=1,og=4,plh=867914364,tim=1429979324891270
FETCH #2:c=0,e=21,p=0,cr=1,cu=0,mis=0,r=1,dep=1,og=4,plh=867914364,tim=1429979324891377
FETCH #2:c=1000,e=329,p=0,cr=1,cu=0,mis=0,r=1,dep=1,og=4,plh=867914364,tim=1429979324891738
FETCH #2:c=0,e=17,p=0,cr=1,cu=0,mis=0,r=1,dep=1,og=4,plh=867914364,tim=1429979324891814
FETCH #2:c=0,e=9,p=0,cr=1,cu=0,mis=0,r=1,dep=1,og=4,plh=867914364,tim=1429979324891852
FETCH #2:c=0,e=9,p=0,cr=1,cu=0,mis=0,r=1,dep=1,og=4,plh=867914364,tim=1429979324891886
FETCH #2:c=0,e=31,p=0,cr=1,cu=0,mis=0,r=1,dep=1,og=4,plh=867914364,tim=1429979324891942
FETCH #2:c=0,e=12,p=0,cr=1,cu=0,mis=0,r=1,dep=1,og=4,plh=867914364,tim=1429979324891992
..........
..........
FETCH #2:c=0,e=9,p=0,cr=1,cu=0,mis=0,r=1,dep=1,og=4,plh=867914364,tim=1429979324897158
FETCH #2:c=0,e=8,p=0,cr=1,cu=0,mis=0,r=1,dep=1,og=4,plh=867914364,tim=1429979324897190
FETCH #2:c=0,e=8,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,plh=867914364,tim=1429979324897222
STAT #2 id=1 cnt=59 pid=0 pos=1 obj=59 op='TABLE ACCESS FULL BOOTSTRAP$ (cr=61 pr=4 pw=0 time=0
us)'
CLOSE #2:c=0,e=30,dep=1,type=0,tim=1429979324927299
CREATE INDEX I_UNDO2 ON UNDO$(NAME) PCTFREE 10 INITRANS 2 MAXTRANS 255 STORAGE (   INITIAL 64K
NEXT 1024K MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 OBJNO 35 EXTENTS (FILE 1 BLOCK 328))
END OF STMT
PARSE #2:c=999,e=961,p=0,cr=0,cu=0,mis=1,r=0,dep=1,og=4,plh=4258903948,tim=1429979324980690
EXEC #2:c=0,e=315,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,plh=4258903948,tim=1429979324981087
STAT #2 id=1 cnt=0 pid=0 pos=1 obj=0 op='INDEX BUILD NON UNIQUE I_UNDO2 (cr=0 pr=0 pw=0 time=0
us)'
STAT #2 id=2 cnt=0 pid=1 pos=1 obj=0 op='SORT CREATE INDEX (cr=0 pr=0 pw=0 time=0 us cost=0 size=0
card=0)'
STAT #2 id=3 cnt=0 pid=2 pos=1 obj=15 op='TABLE ACCESS FULL UNDO$ (cr=0 pr=0 pw=0 time=0 us)'
```

CLOSE #2:c=0,e=7,dep=1,type=0,tim=1429979324981283

=====================

PARSING IN CURSOR #2 len=208 dep=1 uid=0 oct=9 lid=0 tim=1429979324983049 hv=246520463 ad='349c51f8' sqlid='18806807b36ng'

CREATE UNIQUE INDEX I_OBJ1 ON OBJ$(OBJ#,OWNER#,TYPE#) PCTFREE 10 INITRANS 2 MAXTRANS 255 STORAGE (INITIAL 64K NEXT 1024K MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 OBJNO 36 EXTENTS (FILE 1 BLOCK 336))

END OF STMT

=====================

....

....

上面颜色部分可以由 sql_id 对应执行的 sql 语句。

红色部分创建了表 bootstrap$,这个表在 1 号文件的第 520 个块上，看这是什么表空间：

SQL> select * from dba_data_files where file_id=1;

FILE_NAME FILE_ID TABLESPACE_NAME BYTES BLOCKS STATUS RELATIVE_FNO AUT MAXBYTES MAXBLOCKS INCREMENT_BY USER_BYTES USER_BLOCKS ONLINE_

/u01/app/oracle/oradata/orcl3939/system01.dbf 1 SYSTEM 786432000 96000 AVAILABLE 1 YES 3.4360E+10 4194302 1280 785383424 95872 SYSTEM

分析它的表结构：

bootstrap$ (line# number not null, obj# number not null, sql_text varchar2(4000) not null)

里面放的有对象编号，以及 sql 语句，通过查找 bootstrap$:

SQL> select * from bootstrap$ where rownum<=3;

LINE# OBJ# SQL_TEXT

-1 -1 8.0.0.0.0

0 0 CREATE ROLLBACK SEGMENT SYSTEM STORAGE (INITIAL 112K NEXT 56K MINEXTENTS 1 MAXEXTENTS 32765 OBJNO 0 EXTENTS (FILE 1 BLOCK 128))

20 20 CREATE TABLE ICOL$("OBJ#" NUMBER NOT NULL,"BO#" NUMBER NOT NULL,"COL#" NUMBER NOT NULL,"POS#" NUMBER NOT NULL,"SEGCOL#" NUMBER NOT NULL,"SEGCOLLENGTH" NUMBER NOT NULL,"OFFSET" NUMBER NOT NULL,"INTCOL#" NUMBER NOT NULL,"SPARE1" NUMBER,"SPARE2" NUMBER,"SPARE3" NUMBER,"SPARE4" VARCHAR2(1000),"SPARE5" VARCHAR2(1000),"SPARE6" DATE) STORAGE (OBJNO 20 TABNO 4) CLUSTER C_OBJ#(BO#)

由上述 trace 文件可知,创建 boostrap$之后，又创建：

CREATE ROLLBACK SEGMENT SYSTEM STORAGE (INITIAL 112K NEXT 56K MINEXTENTS 1 MAXEXTENTS 32765 OBJNO 0 EXTENTS (FILE 1 BLOCK 128))

CREATE TABLE TS$("TS#" NUMBER NOT NULL,"NAME" VARCHAR2(30) NOT NULL,"OWNER#" NUMBER NOT NULL,"ONLINE$" NUMBER NOT NULL,"CONTENTS$" NUMBER NOT NULL,"UNDOFILE#" NUMBER,"UNDOBLOCK#" NUMBER,"BLOCKSIZE" NUMBER NOT NULL,"INC#" NUMBER NOT NULL,"SCNWRP" NUMBER,"SCNBAS" NUMBER,"DFLMINEXT" NUMBER NOT NULL,"DFLMAXEXT" NUMBER NOT NULL,"DFLINIT" NUMBER NOT NULL,"DFLINCR" NUMBER NOT NULL,"DFLMINLEN" NUMBER NOT NULL,"DFLEXTPCT" NUMBER NOT NULL,"DFLOGGING" NUMBER NOT NULL,"AFFSTRENGTH" NUMBER NOT NULL,"BITMAPPED" NUMBER NOT NULL,"PLUGGED" NUMBER NOT NULL,"DIRECTALLOWED" NUMBER NOT NULL,"FLAGS" NUMBER NOT NULL,"PITRSCNWRP" NUMBER,"PITRSCNBAS" NUMBER,"OWNERINSTANCE" VARCHAR2(30),"BACKUPOWNER" VARCHAR2(30),"GROUPNAME" VARCHAR2(30),"SPARE1" NUMBER,"SPARE2" NUMBER,"SPARE3" VARCHAR2(1000),"SPARE4" DATE) STORAGE (OBJNO 16 TABNO 2) CLUSTER C_TS#(TS#)

CREATE TABLE FILE$("FILE#" NUMBER NOT NULL,"STATUS$" NUMBER NOT NULL,"BLOCKS" NUMBER NOT NULL,"TS#" NUMBER,"RELFILE#" NUMBER,"MAXEXTEND" NUMBER,"INC" NUMBER,"CRSCNWRP" NUMBER,"CRSCNBAS" NUMBER,"OWNERINSTANCE" VARCHAR2(30),"SPARE1" NUMBER,"SPARE2" NUMBER,"SPARE3"

VARCHAR2(1000),"SPARE4" DATE) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 STORAGE (INITIAL 64K NEXT 1024K MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 OBJNO 17 EXTENTS (FILE 1 BLOCK 232))

CREATE TABLE OBJ$("OBJ#" NUMBER NOT NULL,"DATAOBJ#" NUMBER,"OWNER#" NUMBER NOT NULL,"NAME" VARCHAR2(30) NOT NULL,"NAMESPACE" NUMBER NOT NULL,"SUBNAME" VARCHAR2(30),"TYPE#" NUMBER NOT NULL,"CTIME" DATE NOT NULL,"MTIME" DATE NOT NULL,"STIME" DATE NOT NULL,"STATUS" NUMBER NOT NULL,"REMOTEOWNER" VARCHAR2(30),"LINKNAME" VARCHAR2(128),"FLAGS" NUMBER,"OID$" RAW(16),"SPARE1" NUMBER,"SPARE2" NUMBER,"SPARE3" NUMBER,"SPARE4" VARCHAR2(1000),"SPARE5" VARCHAR2(1000),"SPARE6" DATE) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 STORAGE (INITIAL 16K NEXT 104K MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 OBJNO 18 EXTENTS (FILE 1 BLOCK 240))

....

....

那数据文件 1 中的第 520 个块之前的块是哪些呢？

通过下面 sql 语句，可以看到头块为 520 之前的块的对象:

```
SQL>
select b.object_id,a.segment_name,a.segment_type,a.header_block from dba_segments a,dba_objects b where
      a.segment_name=b.object_name(+) and a.header_file=1 and a.header_block<=520 order by
a.header_block;SQL>    2
/


  OBJECT_ID SEGMENT_NAME        SEGMENT_TYPE HEADER_BLOCK
---------- ------------------------------------------------------------------------------ ------------------
------------
   SYSTEM       ROLLBACK 128
2 C_OBJ#       CLUSTER 144
3 I_OBJ#       INDEX 168
6 C_TS#        CLUSTER 176
7 I_TS#        INDEX 184
8 C_FILE#_BLOCK#       CLUSTER 192
9 I_FILE#_BLOCK#       INDEX 200
10 C_USER#       CLUSTER 208
11 I_USER#       INDEX 216
15 UNDO$       TABLE 224
17 FILE$       TABLE 232
18 OBJ$        TABLE 240
23 PROXY_DATA$       TABLE 248
24 I_PROXY_DATA$       INDEX 256
25 PROXY_ROLE_DATA$       TABLE 264
26 I_PROXY_ROLE_DATA$_1       INDEX 272
27 I_PROXY_ROLE_DATA$_2       INDEX 280
28 CON$        TABLE 288
29 C_COBJ#       CLUSTER 296
```

30 I_COBJ# INDEX 304

33 I_TAB1 INDEX 312

34 I_UNDO1 INDEX 320

35 I_UNDO2 INDEX 328

36 I_OBJ1 INDEX 336

37 I_OBJ2 INDEX 344

38 I_OBJ3 INDEX 352

39 I_OBJ4 INDEX 360

40 I_OBJ5 INDEX 368

41 I_IND1 INDEX 376

42 I_ICOL1 INDEX 384

43 I_FILE1 INDEX 392

44 I_FILE2 INDEX 400

45 I_TS1 INDEX 408

46 I_USER1 INDEX 416

47 I_USER2 INDEX 424

48 I_COL1 INDEX 432

49 I_COL2 INDEX 440

50 I_COL3 INDEX 448

51 I_CON1 INDEX 456

52 I_CON2 INDEX 464

53 I_CDEF1 INDEX 472

54 I_CDEF2 INDEX 480

55 I_CDEF3 INDEX 488

56 I_CDEF4 INDEX 496

57 I_CCOL1 INDEX 504

58 I_CCOL2 INDEX 512

59 BOOTSTRAP$ TABLE 520

已选择 47 行。

由 trace 文件知道(也可以直接查 bootstrap$)，520 之前块正是与数据库启动相关的块!

结合 trace 文件和 bootstrap$可知，先加载 bootstrap$后，由 sql_text 然后递归创建 oracle 启动所需的对象。

那 1 号文件第 520 个块了到底放些什么呢？

学会使用 trace 对于研究数据库很重要：

SQL> alter system dump datafile 1 block 520;

系统已更改。

以下摘自部分 trace 文件：

*** 2015-04-26 01:05:26.707

CLOSE #23:c=0,e=110,dep=0,type=0,tim=1429981526707493

=====================

PARSE ERROR #12:len=35 dep=0 uid=0 oct=49 lid=0 tim=1429981526715635 err=25117

alter system dump datafile 1 block

```
=====================
declare
   m_stmt   varchar2(512);
begin
     m_stmt:='delete from sdo_geor_ddl__table$$';
     EXECUTE IMMEDIATE m_stmt;
     EXCEPTION
        WHEN OTHERS THEN
           NULL;
end;
=====================
select obj#,type#,ctime,mtime,stime, status, dataobj#, flags, oid$, spare1, spare2 from
obj$ where owner#=:1 and name=:2 and namespace=:3 and remoteowner is null and linkname is null
and subname is null
....
select
t.ts#,t.file#,t.block#,nvl(t.bobj#,0),nvl(t.tab#,0),t.intcols,nvl(t.clucols,0),t.audit$,t.fl
ags,t.pctfree$,t.pctused$,t.initrans,t.maxtrans,t.rowcnt,t.blkcnt,t.empcnt,t.avgspc,t.chncnt,
t.avgrln,t.analyzetime,t.samplesize,t.cols,t.property,nvl(t.degree,1),nvl(t.instances,1),t.a
vgspc_flb,t.flbcnt,t.kernelcols,nvl(t.trigflag,
0),nvl(t.spare1,0),nvl(t.spare2,0),t.spare4,t.spare6,ts.cachedblk,ts.cachehit,ts.logicalread
from tab$ t, tab_stats$ ts where t.obj#= :1 and t.obj# = ts.obj# (+)

....
select
i.obj#,i.ts#,i.file#,i.block#,i.intcols,i.type#,i.flags,i.property,i.pctfree$,i.initrans,i.m
axtrans,i.blevel,i.leafcnt,i.distkey,i.lblkkey,i.dblkkey,i.clufac,i.cols,i.analyzetime,i.sam
plesize,i.dataobj#,nvl(i.degree,1),nvl(i.instances,1),i.rowcnt,mod(i.pctthres$,256),i.indmet
hod#,i.trunccnt,nvl(c.unicols,0),nvl(c.deferrable#+c.valid#,0),nvl(i.spare1,i.intcols),i.spa
re4,i.spare2,i.spare6,decode(i.pctthres$,null,null,mod(trunc(i.pctthres$/256),256)),ist.cach
edblk,ist.cachehit,ist.logicalread from ind$ i, ind_stats$ ist, (select enabled, min(cols)
unicols,min(to_number(bitand(defer,1))) deferrable#,min(to_number(bitand(defer,4))) valid#
from cdef$ where obj#=:1 and enabled > 1 group by enabled) c where i.obj#=c.enabled(+) and i.obj#
= ist.obj#(+) and i.bo#=:1 order by i.obj#
....
```

```
select
name,intcol#,segcol#,type#,length,nvl(precision#,0),decode(type#,2,nvl(scale,-127/*MAXSB1MIN
AL*/),178,scale,179,scale,180,scale,181,scale,182,scale,183,scale,231,scale,0),null$,fixedst
orage,nvl(deflength,0),default$,rowid,col#,property,
nvl(charsetid,0),nvl(charsetform,0),spare1,spare2,nvl(spare3,0) from col$ where obj#=:1 order
by intcol#
....
select con#,obj#,rcon#,enabled,nvl(defer,0),spare2,spare3 from cdef$ where robj#=:1
....
select
con#,type#,condlength,intcols,robj#,rcon#,match#,refact,nvl(enabled,0),rowid,cols,nvl(defer,
0),mtime,nvl(spare1,0),spare2,spare3 from cdef$ where obj#=:1
....
select decode(u.type#, 2, u.ext_username, u.name), o.name,              t.update$, t.insert$,
t.delete$, t.enabled,              decode(bitand(t.property, 8192),8192, 1,
0),              decode(bitand(t.property, 65536), 65536, 1,
0),           decode(bitand(t.property, 131072), 131072, 1, 0),           (select o.name from
obj$ o                 where o.obj# = u.spare2 and o.type# =57)    from sys.obj$ o, sys.user$ u,
sys.trigger$ t, sys.obj$ bo where t.baseobject=bo.obj# and bo.name = :1 and bo.spare3 = :2    and
bo.namespace = 1    and t.obj#=o.obj# and o.owner#=u.user#    and o.type# = 12 and
bitand(property,16)=0 and bitand(property,8)=0    order by o.obj#
....
select col#, grantee#, privilege#,max(mod(nvl(option$,0),2)) from objauth$ where obj#=:1 and
col# is not null group by privilege#, col#, grantee# order by col#, grantee#
....
select grantee#,privilege#,nvl(col#,0),max(mod(nvl(option$,0),2))from objauth$ where obj#=:1
group by grantee#,privilege#,nvl(col#,0) order by grantee#
m sdo_geor_ddl__table$$
*** 2015-04-26 01:05:40.179
CLOSE #12:c=0,e=20,dep=0,type=0,tim=1429981540179940
=====================
PARSING IN CURSOR #36 len=38 dep=0 uid=0 oct=49 lid=0 tim=1429981540180608 hv=4117028914 ad='0'
sqlid='07605w7uq9s1k'
alter system dump datafile 1 block 520
END OF STMT
```

```
PARSE #36:c=1000,e=484,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,plh=0,tim=1429981540180606
Start dump data blocks tsn: 0 file#:1 minblk 520 maxblk 520
Block dump from cache:
Dump of buffer cache at level 4 for tsn=0, rdba=4194824
BH (0x28fe7134) file#: 1 rdba: 0x00400208 (1/520)  class: 4 ba: 0x28cc6000
   set: 5 pool 3 bsz: 8192 bsi: 0 sflg: 1 pwc: 0,28
   dbwrid: 0 obj: 59 objn: 59 tsn: 0 afn: 1 hint: f
   hash: [0x33e8e55c,0x33e8e55c] lru: [0x297ec8d4,0x28fe72b4]
   lru-flags: on_auxiliary_list
   ckptq: [NULL] fileq: [NULL] objq: [0x31ff2434,0x28fe72cc]
   st: XCURRENT md: NULL tch: 1
   flags:
   LRBA: [0x0.0.0] LSCN: [0x0.0] HSCN: [0xffff.ffffffff] HSUB: [65535]
   cr pin refcnt: 0 sh pin refcnt: 0
Block dump from disk:
*** 2015-04-26 01:05:40.454
buffer tsn: 0 rdba: 0x00400208 (1/520)
scn: 0x0000.00000251 seq: 0x01 flg: 0x04 tail: 0x02511001
frmt: 0x02 chkval: 0xe443 type: 0x10=DATA SEGMENT HEADER - UNLIMITED
Hex dump of block: st=0, typ_found=1
Dump of memory from 0x00772000 to 0x00774000
......
......
   Extent Control Header
   -----------------------------------------------------------------
   Extent Header:: spare1: 0         spare2: 0         #extents: 1         #blocks: 7
                           last map   0x00000000   #maps: 0         offset: 4128
         Highwater::   0x0040020c   ext#: 0         blk#: 3         ext size: 7
   #blocks in seg. hdr's freelists: 1
   #blocks below: 3
   mapblk   0x00000000   offset: 0
                            Unlocked
        Map Header:: next   0x00000000   #extents: 1      obj#: 59       flag: 0x40000000
   Extent Map
   -----------------------------------------------------------------
```

0x00400209   length: 7

nfl = 1, nfb = 1 typ = 1 nxf = 0 ccnt = 3

SEG LST:: flg: USED   lhd: 0x0040020b ltl: 0x0040020b

End dump data blocks tsn: 0 file#: 1 minblk 520 maxblk 520

下面是部分 sql 语句：

select timestamp, flags from fixed_obj$ where obj#=:1

SELECT inst_id, name, value FROM x$diag_info

SELECT inst_id, name, value FROM gv$diag_info WHERE inst_id = USERENV('INSTANCE')

select /*+ rule */ bucket_cnt, row_cnt, cache_cnt, null_cnt, timestamp#, sample_size, minimum, maximum, distcnt, lowval, hival, density, col#, spare1, spare2, avgcln from hist_head$ where obj#=:1 and intcol#=:2

select /*+ rule */ bucket_cnt, row_cnt, cache_cnt, null_cnt, timestamp#, sample_size, minimum, maximum, distcnt, lowval, hival, density, col#, spare1, spare2, avgcln from hist_head$ where obj#=:1 and intcol#=:2

select /*+ rule */ bucket_cnt, row_cnt, cache_cnt, null_cnt, timestamp#, sample_size, minimum, maximum, distcnt, lowval, hival, density, col#, spare1, spare2, avgcln from hist_head$ where obj#=:1 and intcol#=:2

select * from v$diag_info where NAME='Default Trace File'

select timestamp, flags from fixed_obj$ where obj#=:1

select timestamp, flags from fixed_obj$ where obj#=:1

select timestamp, flags from fixed_obj$ where obj#=:1

select inst_id,ksusenum,ksusestn,ksusestv from x$ksumysta where bitand(ksspaflg,1)!=0 and bitand(ksuseflg,1)!=0 and ksusestn<(select ksusgstl from x$ksusgif)

select SID , STATISTIC# , VALUE from GV$MYSTAT where inst_id = USERENV('Instance')

select /*+ rule */ bucket_cnt, row_cnt, cache_cnt, null_cnt, timestamp#, sample_size, minimum, maximum, distcnt, lowval, hival, density, col#, spare1, spare2, avgcln from hist_head$ where obj#=:1 and intcol#=:2

select /*+ rule */ bucket_cnt, row_cnt, cache_cnt, null_cnt, timestamp#, sample_size, minimum, maximum, distcnt, lowval, hival, density, col#, spare1, spare2, avgcln from hist_head$ where obj#=:1 and intcol#=:2

select /*+ rule */ bucket_cnt, row_cnt, cache_cnt, null_cnt, timestamp#, sample_size, minimum, maximum, distcnt, lowval, hival, density, col#, spare1, spare2, avgcln from hist_head$ where obj#=:1 and intcol#=:2

select /*+ rule */ bucket_cnt, row_cnt, cache_cnt, null_cnt, timestamp#, sample_size, minimum,

maximum, distcnt, lowval, hival, density, col#, spare1, spare2, avgcln from hist_head$ where obj#=:1 and intcol#=:2

select /*+ rule */ bucket_cnt, row_cnt, cache_cnt, null_cnt, timestamp#, sample_size, minimum, maximum, distcnt, lowval, hival, density, col#, spare1, spare2, avgcln from hist_head$ where obj#=:1 and intcol#=:2

select /*+ rule */ bucket_cnt, row_cnt, cache_cnt, null_cnt, timestamp#, sample_size, minimum, maximum, distcnt, lowval, hival, density, col#, spare1, spare2, avgcln from hist_head$ where obj#=:1 and intcol#=:2

select * from v$mystat

select distinct sid from v$mystat

select /*+ rule */ bucket_cnt, row_cnt, cache_cnt, null_cnt, timestamp#, sample_size, minimum, maximum, distcnt, lowval, hival, density, col#, spare1, spare2, avgcln from hist_head$ where obj#=:1 and intcol#=:2

select paddr from v$session where sid=9

select timestamp, flags from fixed_obj$ where obj#=:1

select addr, pid, spid, pname, username, serial#, terminal, program, traceid, tracefile, background, latchwait,latchspin,pga_used_mem,pga_alloc_mem,pga_freeable_mem,pga_max_mem from gv$process where inst_id = USERENV('Instance')

select /*+ rule */ bucket_cnt, row_cnt, cache_cnt, null_cnt, timestamp#, sample_size, minimum, maximum, distcnt, lowval, hival, density, col#, spare1, spare2, avgcln from hist_head$ where obj#=:1 and intcol#=:2

select spid from v$process where addr='34FC8DAC'

.....

.....

分析上述标记的红色字体段：

BH (0x28fe7134) file#: 1 rdba: 0x00400208 (1/520)

BH(0x28fe7134):记录的是该块在 buffer cache 中实际的内存地址(buffer header)

rdba:root dba

(1/520):猜想是指 bootstrapt$,下面来验证

后面的 0X00400208 是十六进制数：

先转换成二进制数：

0	0	4	0	0	2	0	8
0000	0000	0100	0000	0000	0010	0000	1000

由前面的文章知，前十位表示文件编号：

0000000001：1

后 22 位表示块号：

0000000000001000001000：16*16*2 + 8=520

上述也可以用 oracle 提供的包直接来计算。

rdba 指向的 bootstrap$!

原来数据库的引导过程中，rdba 用来定位数据库引导的 bootstrap$信息。

上面说了这么多，那 bootstrap$又是如何创建的呢？

通过盖国强老师的《深入解析 Oralce》得知，原来在创建数据库的脚本里，oracle 会隐含的调用 /u01/app/oracle/product/11.2.0/dbhome_1/rdbms/admin/sql.bsq 用于创建数据字典，如果 oracle 找不到 sql.bsq 脚本，数据库创建会出错。我们可以通过修改 sql.bsq 脚本来更改数据字典对象参数，从而实现特殊要求数据库的创建或测试的自定义库。

下面摘自部分 sql.bsq 文本：

```
rem
rem $Header: rdbms/admin/sql.bsq /main/606 2008/07/14 17:25:59 vliang Exp $ sql.bsq
rem   MODIFIED
rem    huagli       06/09/08 - add ddst.bsq
rem    dvoss         01/03/07 - add dlmnr.bsq
rem    rdecker      10/20/06 - create SYSAUX before running dplsql.bsq
rem    jklein       08/01/05 - diag 11g - split-up into units
rem    sdavidso    08/01/05 - add tranform_param type check info
rem    mmpandey    06/07/05 - 4390808: increase the cache value in audses$
rem    mvemulap    05/02/05 - bug fix for 4318925
rem    mhho          04/20/05 - change colklc column size in enc$
rem    tfyu          02/28/05 - Bug 4262763
rem    htran         03/11/05 - remove transportable from fgr$_tablespace_info
rem    mmcracke    03/14/05 - add ddm.bsq for data mining models
rem    alakshmi    02/28/05 - error recovery for maintain_ apis
rem    ddas          01/07/05 - #(4052436) add hint_string to ol$hints
rem    sourghos    01/06/05 - Fix bug 4043119
rem    ilyubash    11/05/04 - Add gen column to i_aw_prop$ index
rem    elu            01/03/05 - streams apply spilling
rem    htran         11/15/04 - comments for spare1 in user$ and streams$_prepare_*
rem    apadmana    10/05/04 - bug3607838: manage any queue
rem    clei          04/15/04 - add view merge permission
rem    weiwang      10/14/04 - set queue flag in base view
rem    mtakahar    09/03/04 - create mon_mods_all$
rem    clei          09/01/04 - add comments to encryption property flags
rem    xuhuali      03/31/04 - audit java
rem    kdias         07/15/04 - revisit privs granted to OUTLN user
rem    nmanappa    07/20/04 - bug 3690876 - clean privileges 194-199,239,240
rem    dmwong       07/21/04 - fix connect role to only contain create session
rem    skaluska    07/09/04 - split up tsm_hist$ into tsm_src$, tsm_dest$
rem    araghava    07/07/04 - (3748430): make partitioning indexes unique.
rem    clei          06/29/04 - add enc$
rem   ssvemuri      06/25/04 - change notification privilege
rem    ramkrish    06/16/04 - correct model nmspc and type
rem    nshodhan    06/11/04 - use streams$_capture_process.spare3
```

```
.....
.....
rem    varora       04/28/95 -   rename col#,usercol#,cols,usercols
rem    tcheng       03/21/95 -   add col# to adtcol$ and ntab$
rem    varora       01/27/95 -   add table for nested table support
rem    skotsovo    01/25/95 -   bring normalized type tables up to date
rem    skotsovo    01/23/95 -   move exceptions from method to method_body
rem    jwijaya      01/04/95 -   add system privileges for type
rem    jwijaya      12/29/94 -   making type$ work (temporarily allow 'version'
rem                                        'checks' columns nullable and mark 'checks'
rem                                        and 'default$' not-supported (N/S))
rem    skrishna    12/06/94 -   create extent table of pre-defined types
rem    varora       12/01/94 -   change toid in adtcol$ to type number
rem    anori         11/17/94 -   ADT support tables and columns
rem
rem !!!!!!!!!!!!!!!!!!!!!!!!!!!!!! IMPORTANT !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
rem Whenever new column is created to store internal, user or kernel column
rem number, be sure to update the structure adtDT in atb.c so that those
rem columns will be updated properly during drop column.
rem !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
rem
dcore.bsq
dsqlddl.bsq
dmanage.bsq
dplsql.bsq
dtxnspc.bsq
dfmap.bsq
denv.bsq
drac.bsq
dsec.bsq
doptim.bsq
dobj.bsq
djava.bsq
dpart.bsq
drep.bsq
daw.bsq
dsummgt.bsq
dtools.bsq
dexttab.bsq
ddm.bsq
dlmnr.bsq
ddst.bsq
```

SYSTEM 表空间的重要性可想而知，如果 system 表空间损坏，则数据库无法打开。SYSTEM 表空间的备份重于一切。

oracle 启动初始化过程十分复杂,上面的很多东西都值得我们去研究，鉴于本人水平有限，希望读者在此基础上对 oracle 启动初始化过程有更深次的理解。