第 3 章　参数及参数文件

在 Oracle 数据库中，有一系列的初始化参数用来进行数据库约束和资源限制，这些参数通常存储在一个参数文件中，在数据库实例启动时读取并加载。

初始化参数对数据库来说非常重要，很多参数通过合理的调整可以极大的提高数据库性能。本章对初始化参数和参数文件进行相关探讨。

3.1 初始化参数的分类

按照得出方式不同，初始化参数可以分为三类：

3.2.1 推导参数(Derived Parameters)

推导参数通常来自于其他参数的运算，依赖其他参数得出。所以这类参数通常不需要修改。如果强制修改，那么修改值会覆盖推导值。

常见的此类参数有很多，例如：SESSIONS 参数，在 Oracle 11gR2 文档中，该参数按以下公式运算得出：

*(1.5 * PROCESSES) + 22*

缺省的，当 PROCESSES 被修改时，此参数会自动计算并生效。以下是一个示范数据库中这两个参数的设置：

```
SQL> select name,value from v$parameter where name in ('processes','sessions');
NAME                 VALUE
-------------------- --------------------
processes            200
sessions             322
```

Processes 参数代表操作系统上能够并发向 Oracle 数据库发起的连接进程数量。如果该参数设置过低，则在应用并非高时，超过 Processes 数量的进程将无法连接到数据库。所以在规划数据库时，合理设置 Processes 参数是十分重要的。但是注意，很多时候由于应用的异常可能导致业务环境的进程数量激增，所以在生产环境中对进程数量进行必要监控是必需的。

以下是一个生产环境中遇到的相关案例，数据库告警日志文件中记录了如下错误：

```
Thu Jul 17 14:40:18 2008
Process J001 died, see its trace file
Thu Jul 17 14:40:18 2008
kkjcre1p: unable to spawn jobq slave process
```

日志提示 J001 进程死掉，数据库不能创建 JOBQ 的从属进程（Slave Process）。由于 Job

进程是动态创建的，如果数据库的进程数量超过，就可能出现 JOB 进程无法创建的问题。检查相关的 TRACE 文件，可以发现如下错误信息：

```
Died during process startup with error 20 (seq=14510)
OPIRIP: Uncaught error 20. Error stack:
ORA-00020: maximum number of processes (500) exceeded
```

数据库的提示是，最大数量的进程数量 500 超过，数据库不再允许更多的进程连接，此时新的连接请求都会收到错误提示，甚至 DBA 也无法登陆数据库。除了重启数据库之外，可以通过 Kill 掉部分进程，然后通过 DBA 身份连接到数据库，诊断分析并解决具体问题。

但是由于 processes 参数是静态参数，修改该参数后需要重新启动数据库才能生效。在数据库启动时，会预先为 Processes 分配内存地址空间，并向 Shared Pool 注册，所以该参数无法动态修改。缺省的每个进程会在共享池中分配 4 Bytes 的注册空间：

```
SQL> select name,value from v$parameter where name ='processes';
NAME                 VALUE
-------------------- ------------------------------
processes            150
SQL> select * from v$sgastat where name='processes';
POOL         NAME                       BYTES
------------ -------------------- ----------
shared pool  processes                     600
```

通常在创建数据库时，建议将该参数修改为 500：

```
SQL> alter system set processes=500 scope=spfile;
System altered.
```

重启之后可以看到数据库进程内存分配的变化：

```
SQL> select name,value from v$parameter where name ='processes';
NAME                           VALUE
------------------------------ ------------------------------
processes                      500
SQL> select * from v$sgastat where name='processes';
POOL                    NAME                             BYTES
----------------------- ------------------------------ ----------
shared pool             processes                         2000
```

3.2.2 操作系统依赖参数

某些参数的有效值或者取值范围依赖或者受限于操作系统，比如 db_cache_size 参数，设置 Oracle 使用的 Buffer Cache 内存大小，该参数的最大值就要受限于物理内存。这一类参数通常被称为操作系统依赖参数。

3.2.3 可变参数

可变参数包含绝大多数潜在影响系统性能的可调整参数，某些可变参数设置的是限制条件，如 OPEN_CURSORS；有的参数是设置容量，如 DB_CACHE_SIZE 等。这类参数通常可以为 DBA 或最终用户调整，从而产生限制或性能变化，对 Oracle 至关重要。

初始化参数通常还有一些其他分类方式：

按照修改方式划分，初始化参数又可以分为静态参数和动态参数。

静态参数只能在参数文件中修改，在重新启动后方能生效；动态参数可以动态调整，调整后通常可以立即生效。

按照获取方式不同，初始化参数又可以分为显示参数和隐含参数。

显示参数可以通过 v$parameter 查询得到；而隐含参数通常以“_”开头，必须通过查询系统表方能获得。

总之，虽然分类方式不同，但是参数都是这些，我们更多需要了解的是这些参数的用途。

3.2.4 废弃参数

由于 Oracle 数据库的参数众多,在新版本中可能废弃很多旧的参数,了解这些废弃参数,明确废弃原因,是 DBA 需要关注的内容之一.

在 Oracle Database 11gR2 中,有大约 130 个参数被废弃:

```
SQL> select * from v$version where rownum <2;
BANNER
--------------------------------------------------------------------------------
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit Production

SQL> select count(*) from V$OBSOLETE_PARAMETER;
  COUNT(*)
----------
       131

SQL> select * from V$OBSOLETE_PARAMETER;
NAME                                                             ISSPE
---------------------------------------------------------------- -----
spin_count                                                       FALSE
use_ism                                                          FALSE
lock_sga_areas                                                   FALSE
instance_nodeset                                                 FALSE
```

这个视图的创建语句如下:

```
SQL> select view_definition from v$fixed_view_definition
  2  where view_name='GV$OBSOLETE_PARAMETER';

VIEW_DEFINITION
```

```
-------------------------------------------------------------------------------------
select inst_id,kspponm,decode(ksppoval,0,'FALSE','TRUE')  from x$ksppo
```

底层的 X$SPPO 是这些废弃参数的来源.

3.2.5 初始化参数的获取

Oracle 的初始化参数可以通过 V$PARAMETER 视图查询得到，在 SQL*PLUS 之中，我们经常可以通过 show parameter 命令来显示某些参数的设置值，例如（以下输出来自 Oracle10gR2 环境）：

```
SQL> show parameter sga
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
lock_sga                             boolean     FALSE
pre_page_sga                         boolean     FALSE
sga_max_size                         big integer 900M
sga_target                           big integer 900M
```

使用 sql_trace 的跟踪当前会话，可以获得 show parameter 的内部操作，跟踪大致步骤如下：

```
alter session set sql_trace=true;
show parameter sga
alter session set sql_trace=false
```

在 user_dump_dest 目录下找到刚刚生成的跟踪文件，可以发现 SQL*Plus 的 Show 命令的本质是通过如下一条 SQL 查询得到的数据库参数：

```
SELECT NAME NAME_COL_PLUS_SHOW_PARAM,DECODE(TYPE,1,'boolean',2,'string',3,
  'integer',4,'file',5,'number',        6,'big integer', 'unknown') TYPE,
  DISPLAY_VALUE VALUE_COL_PLUS_SHOW_PARAM
FROM V$PARAMETER
WHERE UPPER(NAME) LIKE UPPER('%sga%') ORDER BY NAME_COL_PLUS_SHOW_PARAM,ROWNUM
```

注意这里的 UPPER 函数的应用使得 show 命令可以忽略大小写，在 Oracle 8.1.5 中还并不是这个样子，一条 show parameter SGA/sga 在 Oracle 8.1.5 中的输出结果是不同的:

```
SQL> select * from v$version where rownum <2;
BANNER
----------------------------------------------------------------
Oracle8i Enterprise Edition Release 8.1.5.0.0 - Production
SQL> show parameter sga
NAME                           TYPE    VALUE
------------------------------ ------- ------------------------------------------
lock_sga                       boolean FALSE
pre_page_sga                   boolean FALSE
SQL> show parameter SGA
```

在 Oracle 8.1.5 中，这条后台的 SQL 是如下模样：

```
SELECT   SUBSTR (NAME, 1, 36) NAME,
         DECODE (TYPE, 1, 'boolean', 2, 'string', 3, 'integer') TYPE,
         SUBSTR (VALUE, 1, 20) VALUE
FROM v$parameter WHERE NAME LIKE '%SGA%' ORDER BY NAME
```

如果再细致一点观察前面的 SQL，也许你会发现一些奇怪的地方，这就是两个字段别名的设置：

```
NAME  NAME_COL_PLUS_SHOW_PARAM
DISPLAY_VALUE VALUE_COL_PLUS_SHOW_PARAM
```

执行 show parameter 命令的输出并未显示这两个别名，原因何在呢？

我们知道，当启动 SQL*Plus 工具时，会自动调用$ORACLE_HOME/sqlplus/admin/glogin.sql 文件执行一系列的参数设置（可以通过修改这个参数文件来变更 SQL*Plus 登录后的一些显示），打开这个文件可以发现如下两行定义：

```
-- Defaults for SHOW PARAMETERS
COLUMN name_col_plus_show_param FORMAT a36 HEADING NAME
COLUMN value_col_plus_show_param FORMAT a30 HEADING VALUE
```

这就是原因所在。

最常见的对于 glogin.sql 的修改是增加用户名和数据库提示，可以在该文件中增加如下一行：

set sqlprompt "_user @ _connect_identifier>"

常用的设置还有

set sqlprompt "&_user> "

set sqlprompt "_user _privilege> "

此后登陆 SQL*Plus 就会自动在提示符前显示用户名和实例信息，从 Oracle10g 开始，每次会话创建都会自动调用 glogin.sql 文件，而 Oracle10g 之前则只会在 SQL*Plus 启动时调用该文件：

```
[oracle@danaly admin]$ sqlplus eygle/eygle
SQL*Plus: Release 10.2.0.1.0 - Production on Fri Feb 15 23:46:00 2008
Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, Oracle Label Security, OLAP and Data Mining Scoring Engine options
EYGLE @ danaly>connect / as sysdba
Connected.
SYS @ danaly>connect eygle/eygle
Connected.
EYGLE @ danaly>
```

继续前面的讨论，show parameter 既然是从 V$PARAMETER 视图来查询参数设置，那么这个视图的定义就决定了能够获得的内容输出。

通过 V$PARAMETER 视图的创建语句我们可以观察到,实际上 V$PARAMETER 视图过滤掉了以“_”开头的一系列参数：

```
SELECT x.inst_id, x.indx + 1, ksppinm, ksppity, ksppstvl, ksppstdf,
       DECODE (BITAND (ksppiflg / 256, 1), 1, 'TRUE', 'FALSE'),
       DECODE (BITAND (ksppiflg / 65536, 3),
               1, 'IMMEDIATE', 2, 'DEFERRED', 3, 'IMMEDIATE', 'FALSE'),
       DECODE (BITAND (ksppstvf, 7), 1, 'MODIFIED', 4, 'SYSTEM_MOD', 'FALSE'),
       DECODE (BITAND (ksppstvf, 2), 2, 'TRUE', 'FALSE'), ksppdesc, ksppstcmnt
  FROM x$ksppi x, x$ksppcv y
 WHERE (x.indx = y.indx)
   AND ((TRANSLATE (ksppinm, '_', '#') NOT LIKE '#%') OR (ksppstdf = 'FALSE') )
```

这些以“_”开头的初始化参数通常被称为隐含参数，Oracle 通常不建议修改这些参数，但是因为某些隐含参数有着特殊的功能，逐渐被越来越多的人所熟知。

从 V$PARAMETER 视图的创建语句中我们可以发现，这个视图实际上是建立在两个底层数据字典表 X$KSPPI 和 X$KSPPCV 之上的。

通过以下查询我们可以从内部表直接获得所有参数及其描述信息：

```
SELECT x.ksppinm NAME, y.ksppstvl VALUE, x.KSPPDESC PDESC
  FROM SYS.x$ksppi x, SYS.x$ksppcv y
WHERE x.indx = y.indx
   AND x.ksppinm LIKE '%&par%';
```

比较常用的几个隐含参数有：

```
NAME                           VALUE      PDESC
------------------------------ ---------- ------------------------------------------------
_allow_resetlogs_corruption    FALSE      allow resetlogs even if it will cause corruption
_offline_rollback_segments                offline undo segment list
_corrupted_rollback_segments              corrupted undo segment list
```

在后文我们会介绍一下这几个参数的重要用途。

3.2.6 初始化参数的可选项目

Oracle 的很多参数具有多个不同的可选值，可以通过 V$PARAMETER_VALID_VALUES 来进行查询，例如以下查询获得 cursor_sharing 参数的三个可选设置：

```
SQL> select * from V$PARAMETER_VALID_VALUES where name like '%cursor%';

       NUM NAME                             ORDINAL VALUE               ISDEFAULT
---------- ------------------------------ ---------- -------------------- ----------
       901 cursor_sharing                          1 FORCE                FALSE
```

```
       901 cursor_sharing                          2 EXACT              TRUE
       901 cursor_sharing                          3 SIMILAR            FALSE
```

这个视图是基于 X$KSPVLD_VALUES 建立起来的，也可以查询 X$视图来直接获得这些设置选项：

```
SQL> SELECT
  2     INST_ID,
  3     PARNO_KSPVLD_VALUES     pvalid_par#,
  4     NAME_KSPVLD_VALUES      pvalid_name,
  5     VALUE_KSPVLD_VALUES     pvalid_value,
  6     DECODE(ISDEFAULT_KSPVLD_VALUES, 'FALSE', '', 'DEFAULT' ) pvalid_default
  7  FROM
  8     X$KSPVLD_VALUES
  9  WHERE
 10     LOWER(NAME_KSPVLD_VALUES) LIKE LOWER('%&1%')
 11  ORDER BY
 12     pvalid_par#,pvalid_default,pvalid_Value
 13  /
Enter value for 1: cursor
old  10:        LOWER(NAME_KSPVLD_VALUES) LIKE LOWER('%&1%')
new  10:        LOWER(NAME_KSPVLD_VALUES) LIKE LOWER('%cursor%')

   INST_ID   PAR# PARAMETER                           VALUE    DEFAULT
---------- ------ ----------------------------------- -------- -------
         1    901 cursor_sharing                      EXACT    DEFAULT
         1        cursor_sharing                      FORCE
         1        cursor_sharing                      SIMILAR

         1   1003 _optimizer_extended_cursor_sharing  NONE
         1        _optimizer_extended_cursor_sharing  UDO
```

3.2 参数文件

参数文件是一个包含一系列参数及参数对应值的操作系统文件。

参数文件有两种类型：

- 初始化参数文件（Initialization Parameters Files）- Oracle9i 之前 Oracle 一直采用 pfile 方式存储初始化参数，该文件为文本文件。可以手工修改。
- 服务器参数文件（Server Parameter Files）- 从 Oracle9i 开始，Oracle 引入的 spfile 文件，该文件为二进制格式，不能通过手工修改。

从操作系统上我们也可以看到这两者的区别，INIT 文件为 ASCII 文本文件，SPFILE 为数据文件：

```
[oracle@jumper oracle]$ cd $ORACLE_HOME/dbs
[oracle@jumper dbs]$ file initconner.ora
initconner.ora: ASCII text
[oracle@jumper dbs]$ file spfileconner.ora
spfileconner.ora: data
```

在 9i 以前，Oracle 使用 pfile 存储初始化参数设置，参数文件的修改需要手工进行，这些参数在实例启动时被读取，通过 pfile 的修改需要重起实例才能生效；从 Oracle9i 开始，Oracle 引入 spfile 文件，使用 spfile 你可以通过命令来修改参数（如在 SQL*Plus 中通过 ALTER SYSTEM 修改参数），不再需要通过手工修改，对于动态参数所有更改可以立即生效，同时你可以选择使更改只应用于当前实例还是同时应用到 spfile ，对于静态参数我们只能将变更应用到 spfile 文件，这些变更在数据库重启后生效。

SPFILE 的引入使得对于参数的修改都可以在命令行完成，我们可以彻底告别手工修改初始化参数文件的历史，这就大大减少了人为错误的发生。

另外 SPFILE 是一个二进制文件，可以使用 RMAN 进行备份，这样实际上 Oracle 把参数文件也纳入了 Oracle 的备份恢复体系。

随着 SPFILE 的引入，一个新的视图 V$SPPARAMETER 被引入，这个视图用于记录 SPFILE 文件中设置的初始化参数：

```
SQL> select sid,name,value from v$spparameter where value is not null;
SID        NAME                           VALUE
---------- ------------------------------ -----------------------------------------
*          processes                      1000
*          sessions                       555
*          sga_target                     1241513984
……
smsdb1     instance_number                1
smsdb2     instance_number                2
*          undo_management                AUTO
smsdb1     undo_tablespace                UNDOTBS1
smsdb2     undo_tablespace                UNDOTBS2
```

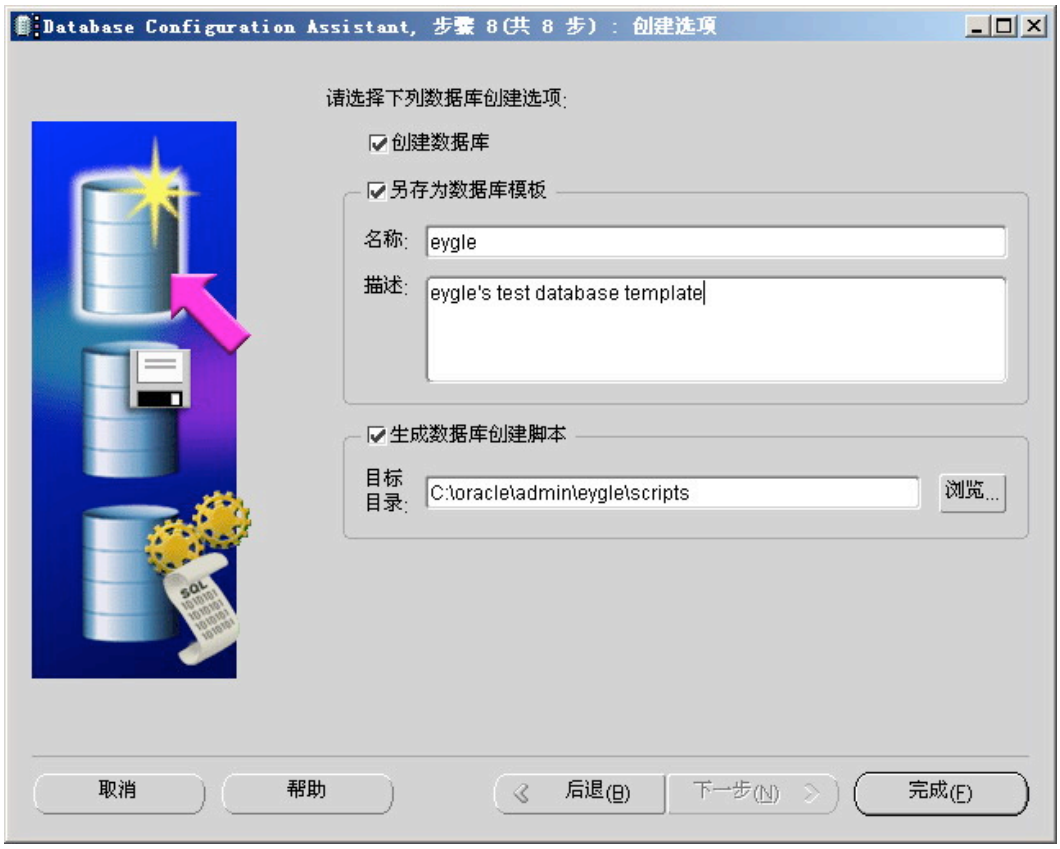
注意查询输出结果中的 SID 项，如果 SID 为“*”则意味着参数设置对 RAC 集群中的所有实例有效，如果是指定了实例名称的，则只对相应实例生效；对于单实例数据库，则 SID 项设置皆为“*”（ 通过 sql_trace 或者 autotrace 功能可以发现，v$spparameter 视图是建立在一个新的 X$KSPSPFILE 数据字典表之上的）。

3.2.1 PFILE 和 SPFILE

除了第一次启动数据库需要 PFILE（然后可以根据 PFILE 创建 SPFILE），数据库可以不

再需要 PFILE，ORACLE 强烈推荐使用 spfile，应用其新特性来存储和维护初始化参数设置。

当使用 DBCA 自定义（不使用模版）创建数据库时，在最后一个步骤，选择生成数据库创建脚本，可以将创建数据库所需要执行的脚本保存下来。通过这些脚本，可以进一步研究 Oracle 数据库的创建过程（当然也可以通过手工执行这些脚本，手工创建数据库）：



以 Windows 为例，在 scripts 目录下，通常可以看到这样一些脚本（根据安装选项不同，脚本可能不同）：

```
C:\oracle\admin\eygle\scripts>dir
2005-01-06  13:23               918 CreateDB.sql
2005-01-06  13:23               631 CreateDBCatalog.sql
2005-01-06  13:23               134 CreateDBFiles.sql
2005-01-06  13:23               781 eygle.bat
2005-01-06  13:23             2,847 init.ora
2005-01-06  13:24               409 postDBCreation.sql
```

手工创建过程通常可以通过 eygle.bat 批处理文件执行开始，系统会根据脚本自动执行创建过程。我们不打算过多介绍数据库创建过程，和本章内容有关的是，这里存在一个 **init.ora** 文件（或 init.ora.<时间戳>文件），这个文件是根据创建数据库之前定义的参数自动生成的，该参数文件被用来在创建过程中启动数据库，通过 CreateDB.sql 可以看到这个引用：

```
connect SYS/change_on_install as SYSDBA
```

```
set echo on
spool C:\oracle\ora92\assistants\dbca\logs\CreateDB.log
startup nomount pfile="C:\oracle\admin\eygle\scripts\init.ora";
CREATE DATABASE eygle
MAXINSTANCES 1 MAXLOGHISTORY 1 MAXLOGFILES 5 MAXLOGMEMBERS 3 MAXDATAFILES 100
DATAFILE 'd:\oradata\eygle\system01.dbf' SIZE 250M REUSE AUTOEXTEND ON NEXT  10240K MAXSIZE
UNLIMITED EXTENT MANAGEMENT LOCAL
```

在数据库创建完成之后，Oracle 调用 postDBCreation.sql 脚本来进行一系列的后续处理，最后 Oracle 通过 init.ora 文件创建了 spfile 文件，该脚本的内容大致如下：

```
connect SYS/change_on_install as SYSDBA
set echo on
spool C:\oracle\ora92\assistants\dbca\logs\postDBCreation.log
@C:\oracle\ora92\rdbms\admin\utlrp.sql;
shutdown ;
connect SYS/change_on_install as SYSDBA
set echo on
spool C:\oracle\ora92\assistants\dbca\logs\postDBCreation.log
create spfile='C:\oracle\ora92\database\spfileeygle.ora'
        FROM pfile='C:\oracle\admin\eygle\scripts\init.ora';
startup ;
```

这就是从 Oracle9i 开始的 pfile 和 spfile 的交接。建议每个试图深入学习 Oracle 的人都仔细研究一下自动建库的脚本，深入了解该过程非常有助于 Oracle 学习与领悟。

3.2.2 获取参数的视图

数据库的参数设置存储在数据字典表中，进而通过视图展现出来，前文已经提到了两个相关视图：V$PARAMETER 和 V$SPPARAMETER。除了这两个重要视图之外，常用的还有一个 V$SYSTEM_PARAMETER 视图。

由于 Oracle 数据库是一个多用户数据库系统，所以不同会话之间可能存在不同的参数设置，V$SYSTEM_PARAMETER 视图用于显示当前对于实例（INSTANCE）级别生效的参数设置，可以被认为是系统（SYSTEM）级别的参数设置；当一个会话（SESSION）创建时，会首先从 V$SYSTEM_PARAMETER 继承参数设置，而 V$PARAMETER 正是用于显示在会话级别生效的参数设置，如果在会话级别修改了参数设置，这里的参数值就可能和 V$SYSTEM_PARAMETER 显示的有所不同。

扩展一下介绍，对应于 V$PARAMETER 视图，还存在一个 V$PARAMETER2 视图，这个视图和 V$PARAMETER 的区别在于，对于存在多个参数值的参数，在这个视图中分多行记录，例如对于 CONTROL_FILES 参数在 V$PARAMETER 中显示如下：

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

```
--------------
VALUE
--------------------------------------------------------------------------------
control_files
/data1/oradata/phsdb/control01.ctl, /data1/oradata/phsdb/control02.ctl, /data1/o
radata/phsdb/control03.ctl
```

而在 V$PARAMETER2 则分为三条记录显示：

```
SQL> select name,value from v$parameter2 where name='control_files';
NAME            VALUE
--------------- --------------------------------------------------
control_files   /data1/oradata/phsdb/control01.ctl
control_files   /data1/oradata/phsdb/control02.ctl
control_files   /data1/oradata/phsdb/control03.ctl
```

类似的对于 V$SYSTEM_PARAMETER 视图，也存在一个对应的 V$SYSTEM_PARAMETER2 视图。

我们前边提到当在 SQL*Plus 中使用 show parameter 命令时，实际上在后台查询的是 V$PARAMETER 视图；从 Oracle Database 11g 开始，SQL*Plus 增加了一个命令 show spparameter 用于显示 V$SPPARAMETER 视图中的参数设置：

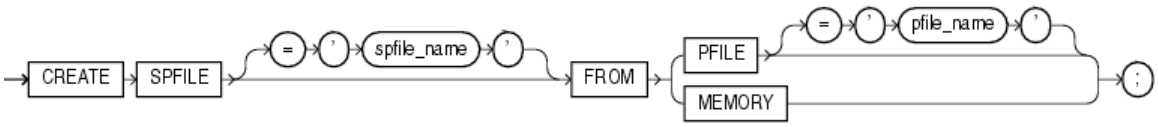
```
SQL> select * from v$version where rownum <2;
BANNER
--------------------------------------------------------------------------------
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Production
SQL> show parameter memory_target
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
memory_target                        big integer 800M
SQL> show spparameter memory_target
SID      NAME                          TYPE        VALUE
-------- ----------------------------- ----------- ------------------------------
*        memory_target                 big integer 800M
```

这是 Oracle Database 11g 在易用性方面的一个小小增强。

3.2.3 SPFILE 的创建

从 Oracle9i 开始，缺省的,ORACLE 使用 SPFILE 启动数据库，从上一节的数据库创建过程我们也可以看到，SPFILE 必须由 PFILE 创建，新创建的 SPFILE 在下一次启动数据库时生效。

CREATE SPFILE 需要 SYSDBA 或者 SYSOPER 的权限,其语法如下,注意其中 MEMORY 的选项是自 Oracle 11g 引入的：



例如：

```
SQL> create spfile from pfile;
```

缺省的，spfile 会创建到系统缺省目录，在 Unix 下的缺省目录为 $ORACLE_HOME/dbs 在 Windows 上的缺省目录为$ORACLE_HOME\database。

如果 SPFILE 已经存在，那么创建会返回以下错误：

```
SQL> create spfile from pfile;
create spfile from pfile
*
ERROR 位于第 1 行:
ORA-32002: 无法创建已由例程使用的 SPFILE
```

这也可以用来判断当前是否使用了 SPFILE 文件。然而意外的时，Oracle 并没有向其他文件一样，在运行期间保持锁定，让我们作以下试验：

```
SQL> host rename SPFILEEYGLEN.ORA SPFILEEYGLEN.ORA.BAK

SQL> alter system set db_cache_size=24M scope=both;
系统已更改。
SQL> host dir *.ora
 E:\Oracle\Ora9iR2\database 的目录
2003-02-10  14:35               2,048 PWDeyglen.ORA

SQL> alter system set db_cache_size=24M scope=spfile;
alter system set db_cache_size=24M scope=spfile
*
ERROR 位于第 1 行:
ORA-27041: 无法打开文件
OSD-04002: 无法打开文件
O/S-Error: (OS 2) 系统找不到指定的文件。

SQL> host rename SPFILEEYGLEN.ORA.BAK SPFILEEYGLEN.ORA

SQL> alter system set db_cache_size=24M scope=spfile;
系统已更改。
```

由于运行期并不锁定 spfile，所以 spfile 可能会意外丢失，如果发生此类情况，Oracle 会不允许使用 create spfile from pfile 缺省命令来重建 spfile（因 ORA-32002 错误而失败），但是此时可以创建一个自定义名称的 spfile 文件，然后重命名为缺省名称即可。

3.2.4 由内存创建参数文件

从 Oracle 11g 开始，为了增强参数文件的恢复，一个新的命令被引入用于从当前运行实例创建参数文件，这个命令是：

```
create <spfile|pfile> from memory;
```

这个命令可以使用当前的参数设置在缺省位置创建一个 spfile 文件，当然也可以指定一个不同的位置：

```
SQL> create spfile='/tmp/spfile.ora' from memory;
File created.
```

这一增强简化了我们在某些条件下的参数文件恢复。

通过跟踪这个过程，可以获取这个简单增强的内部操作：

```
SQL> alter session set sql_trace=true;
Session altered.
SQL> create spfile='/tmp/spfile.ora' from memory;
File created.
```

先观察一下自动生成跟踪文件的内容，参数文件输出的参数远远多于我们的设定，很多隐含参数的设置同样被列出：

```
[oracle@wapdb trace]$ strings /tmp/spfile.ora
*.__db_cache_size=196M
*.__java_pool_size=4M
*.__large_pool_size=32M
*.__oracle_base='/opt/oracle' # ORACLE_BASE set from environment
*.__pga_aggregate_target=200M
*.__sga_target=600M
*.__shared_io_pool_size=0
*.__shared_pool_size=360M
*.__streams_pool_size=0
*._always_anti_join='CHOOSE'
*._always_semi_join='CHOOSE'
*._b_tree_bitmap_plans=TRUE
*._bloom_filter_enabled=TRUE
*._bloom_pruning_enabled=TRUE
*._complex_view_merging=TRUE
*._convert_set_to_join=FALSE
```

现在格式化一下跟踪文件，看看数据库生成脚本的命令，以下是递归 SQL 摘录，可以看出这些 SQL 和此前我们手工执行的 SQL 有殊途同归之妙：

```
SQL ID : 0000000000000
alter session set sql_trace=true
```

```
SQL ID : 0000000000000
create spfile='/tmp/spfile.ora' from memory

SQL ID : asvzxj61dc5vs
select timestamp, flags from fixed_obj$ where obj#=:1

SQL ID : 48x4v2mpymujm
select x.inst_id,kspftctxsid,kspftctxpn,ksppinm,ksppity,kspftctxdvl, kspftctxvn,kspftctxct
from x$ksppi x, x$ksppsv2 y  where ((x.indx+1) = kspftctxpn) and
  (bitand(ksppilrmflg,64)!=64) and  ((kspftctxdf = 'FALSE') or (bitand(kspftctxvf,8) = 8))

SQL ID : 437ya6wz5w505
select SID, NUM, NAME, TYPE, DISPLAY_VALUE, ORDINAL, UPDATE_COMMENT
from GV$SYSTEM_PARAMETER4 where INST_id = USERENV('Instance')

SQL ID : 4kvhq1dbu6hnx
select num,name,type,display_value,update_comment
from v$system_parameter4 order by lower(name),ordinal
```

3.2.5 SPFILE 的搜索顺序

重新启动数据库，使用 startup 命令，Oracle 将会按照以下顺序在缺省目录（Windows 缺省目录为$ORACLE_HOME\database；Unix/Linux 下缺省目录为$ORACLE_HOME/dbs）中搜索参数文件：spfile<ORACLE_SID>.ora 、spfile.ora、init<ORACLE_SID>.ora

创建了 spfile,重新启动数据库，Oracle 会按顺序搜索以上目录，spfile 就会自动生效。

3.2.6 使用 PFILE/SPFILE 启动数据库

如果你想使用 pfile 启动数据库，你可以在启动时指定 pfile 或者删除 spfile。通过指定 pfile 启动数据库的命令格式类似如下：

```
SQL> startup pfile='E:\Oracle\admin\eyglen\pfile\init.ora';
```

你不能以同样的方式指定 spfile，但是可以创建一个包含 spfile 参数的 pfile 文件，指向 spfile。SPFILE 是一个自 Oracle9i 引入的初始化参数，类似于 IFILE 参数。 SPFILE 参数用于定义非缺省路径的 spfile 文件。

你可以在 PFILE 链接到 SPFILE 文件,同时在 PFILE 中定义其他参数,**如果参数重复设置，后读取的参数将取代先前的设置**。

看一下以下例子，当前使用 spfile 启动数据库，log_archive_start 参数设置为 True：

```
SQL> show parameter log_archive_start
NAME                          TYPE    VALUE
------------------------ ----------- -----------
log_archive_start             boolean  TRUE
SQL> show parameter spfile
NAME              TYPE     VALUE
------------------------------ ------------------ ------------------
spfile                string    %ORACLE_HOME%\DATABASE\SPFILE%ORACLE_SID%.ORA
```

修改 PFILE 文件内容如下：

```
#Pfile link to SPFILE
SPFILE= 'E:\Oracle\Ora9iR2\database\SPFILEEYGLEN.ORA'
log_archive_start = false
```

可以预见这个 log_archive_start 参数设置将会代替 SPFILE 中的设置：

```
SQL> startup  pfile='e:\initeyglen.ora'
SQL> show parameter spfile
NAME        TYPE    VALUE
------------------------------------ ----------------------
spfile          string    E:\Oracle\Ora9iR2\database\SPFILEEYGLEN.ORA
SQL> show parameter log_archive_start
NAME                    TYPE     VALUE
---------------------- -------- ----------------------
log_archive_start   boolean    FALSE
```

这就是 spfile 与 pfile 结合使用的一些技巧。同样的用法其实在 RAC 的系统中非常常见，由于 RAC 中，通常需要把 spfile 存储在共享磁盘上，所以常规的做法就是通过定义 pfile 文件，在 pfile 文件中对 spfile 文件进行重定向，下面是 RAC 环境中一个参数文件的设置范例：

```
[oracle@raclinux1 ~]$ cd $ORACLE_HOME/dbs
[oracle@raclinux1 dbs]$ more initRACDB1.ora
SPFILE='+MY_DG2/RACDB/spfileRACDB.ora'
```

在这样的环境中，需要谨慎使用 create spfile from pfile 的命令，很多朋友因为草率的执行这样的操作而导致数据库故障。

在 ASM 或 RAC 环境中，通常的 init<sid>.ora 文件中只有如上示例的一行，如果此时执行 create spfile from pfile 命令，则新创建的 spfile 文件将也只有这样一行信息，数据库使用这样的一个 spfile 将无法启动。

在数据库启动之后，可以使用 ALTER SYSTEM 方式将参数修改直接固化到 SPFILE 文件中。

```
SQL> alter system set log_archive_start=false scope=spfile;
系统已更改。
```

> 提示：通过在 pfile 中调用 spfile，使用后设置的参数覆盖 spfile 中的参数设置，是解决 spfile 中参数设置错误的一种方法。

3.2.7 修改参数

可以通过 ALTER SYSTEM 或者导入导出来更改 SPFILE 的内容。从 Oracle9i 开始，ALTER SYSTEM 命令增加了一个新的选项：SCOPE。

SCOPE 参数有三个可选值：MEMORY ,SPFILE , BOTH

- MEMORY-只改变当前实例运行，重新启动数据库后失效
- SPFILE-只改变 SPFILE 的设置，不改变当前实例运行，重新启动数据库后生效
- BOTH-同时改变实例及 SPFILE，当前更改立即生效，重新启动数据库后仍然有效。

针对 RAC 环境，ALTER SYSTEM 还可以指定 SID 参数，对不同实例进行不同设置。

所以通过 spfile 修改参数的完整命令如下：

```
alter system set <parameter_name> =<value> scope = memory|spfile|both [sid=<sid_name>]
```

通过简单的例子来看一下 SCOPE 参数的几个用法：

1． SCOPE=MEMORY

修改当前实例的 db_cache_advice 参数为 OFF：

```
SQL> show parameter db_cache_ad
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_cache_advice                      string      ON
SQL> alter system set db_cache_advice=off scope=memory;
System altered.
SQL> show parameter db_cache_ad
NAME                    TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_cache_advice             string        OFF
```

如果观察 alert_<sid>.log 文件，我们可以发现其中记录了如下一行：

```
Wed Apr 26 21:18:57 2006
ALTER SYSTEM SET db_cache_advice='OFF' SCOPE=MEMORY;
```

如果重新启动数据库，这个更改将会丢失：

```
SQL> startup force;
SQL> show parameter db_cache_ad
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_cache_advice                      string      ON
```

也就是说 SCOPE=MEMORY 的修改影响，不会跨越一次数据库的重新启动。

2. SCOPE=SPFILE

当指定 SCOPE=SPFILE 时，当前实例运行不受影响：

```
SQL> alter system set db_cache_advice=off scope=spfile;
System altered.
SQL> show parameter db_cache_ad
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_cache_advice                      string      ON
```

同样可以从告警日志文件中看到这个修改：

```
Wed Apr 26 21:24:02 2006
ALTER SYSTEM SET db_cache_advice='OFF' SCOPE=SPFILE;
```

这个修改将在下次数据库启动后生效：

```
SQL> startup force;
SQL> show parameter db_cache_ad
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_cache_advice                      string      OFF
```

但是需要知道的是，对于静态参数，只能指定 SCOPE=SPFILE 进行修改。通过 SCOPE=spfile 修改的参数，虽然对当前实例无效，但是其参数值可以从 v$spparameter 视图中查询得到：

```
SQL> show parameter  db_cache_advice
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_cache_advice                      string      OFF
SQL> alter system set db_cache_advice=on scope=spfile;
System altered.
SQL> select name,value from v$spparameter where name='db_cache_advice';
NAME                  VALUE
--------------------  --------------------
db_cache_advice       ON
SQL> show parameter db_cache_ad
NAME                       TYPE        VALUE
-------------------------- ----------- ------------------------------
db_cache_advice            string      OFF
```

3．SCOPE = BOTH

使用 BOTH 选项实际上等同于不带参数的 ALTER SYSTEM 语句。

```
SQL> alter system set db_cache_advice=off scope=both;
System altered.
```

```
SQL> alter system set db_cache_advice=off;
System altered.

SQL> show parameter db_cache_ad
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_cache_advice                      string      OFF
```

在告警日志文件中可以看到如下信息：

```
Wed Apr 26 21:28:21 2006
ALTER SYSTEM SET db_cache_advice='OFF' SCOPE=BOTH;
Wed Apr 26 21:28:28 2006
ALTER SYSTEM SET db_cache_advice='OFF' SCOPE=BOTH;
```

注意到不带 SCOPE 参数和 SCOPE=BOTH 实际上是等价的。但是如果修改静态参数，那么需要指定 SPFILE 参数，不能指定 BOTH 参数，否则数据库将会报错。

```
SQL> ALTER SYSTEM SET sql_trace=FALSE SCOPE=BOTH;
ALTER SYSTEM SET sql_trace=FALSE SCOPE=BOTH
                 *
ERROR 位于第 1 行:
ORA-02095: 无法修改指定的初始化参数
SQL> ALTER SYSTEM SET sql_trace=FALSE SCOPE=SPFILE;
系统已更改。
```

注意：在 Oracle10g 中,sql_trace 已经变为了一个动态参数。

4．RAC 环境中的修改

在 Rac 环境中，如果不指定 sid 名称，或者指定为“*”，那么修改缺省的对所有实例生效.例如：

ALTER SYSTEM SET OPEN_CURSORS=500 SID='' SCOPE=MEMORY;*

如果需要修改指定的实例，则需要设置相应的 SID 参数，例如：

```
SQL> select sid,name,value from v$spparameter where name='open_cursors';
SID        NAME                                 VALUE
---------- ------------------------------------ ------------------------------
*          open_cursors                         300
SQL> alter system set open_cursors=150 scope=spfile sid='RACDB1';
System altered.
SQL> select sid,name,value from v$spparameter
  2  where name='open_cursors';
SID        NAME                                 VALUE
---------- ------------------------------------ ------------------------------
```

```
*          open_cursors                           300
RACDB1     open_cursors                           150
```

需要注意的是，在 RAC 环境中，不同实例的 undo_tablespace 设置是不同的，当修改一个实例的 undo 表空间时，一定要注意指定相应的实例，以避免修改错误：

```
SQL> select sid,name,value from v$spparameter where name='undo_tablespace';
SID        NAME                                   VALUE
---------- -------------------------------------- ------------------------------
RACDB1     undo_tablespace                        UNDOTBS1
RACDB2     undo_tablespace                        UNDOTBS2
```

5．在关闭数据库状态修改 spfile

可以在数据库 shutdown 时创建和修改 spfile,例如：

```
SQL> shutdown immediate
数据库已经关闭。
已经卸载数据库。
ORACLE 例程已经关闭。
SQL> create pfile from spfile;
文件已创建。
SQL> create spfile from pfile;
文件已创建。
```

所以如果当我们不慎错误的修改了参数导致数据库无法启动时，可以通过创建 pfile 文件，修改其中的参数，再由 pfile 创建 spfile 的方式解决，最后由 spfile 正常启动数据库。

例如如下设置了 db_block_buffers 参数之后，数据库在下次启动时将会出错，因为该参数与 db_cache_size 不兼容：

```
SQL> alter system set db_block_buffers=1000 scope=spfile;
System altered.
SQL> shutdown immediate;
SQL> startup
ORA-00381: cannot use both new and old parameters for buffer cache size specification
```

此时可以由 spfile 创建 pfile 文件：

```
SQL> create pfile from spfile;
File created.
```

然后修改参数文件，删除其中的 db_block_buffers 参数：

```
*.db_block_buffers=1000
```

接下来由 pfile 创建 spfile 启动数据库：

```
SQL> create spfile from pfile;
File created.
SQL> startup
ORACLE instance started.
Database mounted.
```

```
Database opened.
```

Spfile 的修改和使用方式，我们是一定要熟练的。

> 提示：这是修改 spfile 的第二种方式，通过这种方式，我们可以快速修正 spfile 中的错误参数定义。

在 spfile 引入之初，很多人因为其使用和修改复杂而拒绝使用 spfile，仍然沿用 pfile 文件，其实不要小看 spfile 文件，spfile 可以在数据库中通过命令动态修改的特性，是 Oracle10g 中很多自动化特性的实现基础。

我们应当熟悉这样一个事实，Oracle 经常在现行版本中为下一版本作准备，并优先推出部分功能，这些功能因为其超前可能显得不够实用，而当这些特性在新版本中再次出现时，我们才忽然知道，这些特性原来是如此的不可缺少。

以 Oracle10g 的自动共享内存调整特性（具体内容将在后面章节详细介绍）来做一个简单说明。当在 Oracle10g 中设置了 SGA_TARGET 参数启用了自动 SGA 调整之后，Oracle 会同时启用一系列的新的隐含参数来控制 SGA 各组件的大小。如果足够细心，大家可能从告警日志文件中注意到，每次启动这些参数的设置通常都是不同的，从生产环境中摘录两个片段给大家参考。

第一个启动信息：

```
Thu Jan 19 14:38:43 2006
Starting ORACLE instance (normal)
。。。。。
LICENSE_MAX_USERS = 0
SYS auditing is disabled
Starting up ORACLE RDBMS Version: 10.2.0.1.0.
System parameters with non-default values:
  processes                = 150
  __shared_pool_size       = 75497472
  __large_pool_size        = 4194304
  __java_pool_size         = 4194304
  __streams_pool_size      = 0
  spfile                   = +ORADG/danaly/spfiledanaly.ora
  sga_target               = 943718400
  。。。。。。
  db_block_checksum        = FULL
  db_block_size            = 8192
  __db_cache_size          = 851443712
```

第二个启动信息：

```
Wed Apr  5 12:01:02 2006
Starting ORACLE instance (normal)
……
```

```
LICENSE_MAX_USERS = 0
SYS auditing is disabled
ksdpec: called for event 13740 prior to event group initialization
Starting up ORACLE RDBMS Version: 10.2.0.1.0.
System parameters with non-default values:
  processes                = 150
  __shared_pool_size       = 113246208
  __large_pool_size        = 4194304
  __java_pool_size         = 12582912
  __streams_pool_size      = 0
  spfile                   = +ORADG/danaly/spfiledanaly.ora
  sga_target               = 943718400
  。。。。。。
  db_block_checksum        = FULL
  db_block_size            = 8192
  __db_cache_size          = 805306368
```

这些参数的不同就是 Oracle 自动调整的结果，通过 spfile 的动态修改，这些参数值可以跨越数据库重启动而继续生效。

由于 spfile 是一个二进制文件，所以不能通过手工方式修改，很多朋友通过手工修改而导致 spfile 损坏，使得该 spfile 不能用来启动数据库。我们要引以为戒。

曾经有朋友问这样一个错误：

```
Errors in file /u01/app/oracle/admin/orcl/bdump/orcl_mmon_5234.trc:
ORA-00600: internal error code, arguments: [kmgs_parameter_update_timeout_1], [1565], [], [],
[], [], [], []
ORA-01565: error in identifying file
 '/u01/app/oracle/oracle/product/10.2.0/db_1/dbs/spfileorcl.ora'
ORA-27046: file size is not a multiple of logical block size
```

从后面的 ORA-01565 错误号，我们明显能够看出这是一个参数文件损坏的问题，手工修改损坏了这个参数文件。

那么前一个 600 错误呢？

我们知道 Oracle 在运行阶段是不锁定 spfile 的，所以 spfile 的问题要等到下一次修改或使用时才能发现。

3.2.8 解决 SPFILE 参数修改错误

在使用 SPFILE 之后，可能一些不同以往的错误会被遇到。比如修改了错误的参数导致数据库无法启动，比如手工修改 SPFILE 导致参数文件损坏。

下面介绍参数修改错误常见问题的处理办法。比如修该 sga_max_szie 超过系统最大内存数量：

```
SQL> alter system set sga_max_size=5G scope=spfile;
System altered.
```

那么下次启动，内存不足，数据库是无法启动的，数据库出现 ORA-27102 号错误：

```
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup
ORA-27102: out of memory
```

如果是在 Unix、Linux 上，可以在实例未启动时连接，创建 pfile，然后手工修改 PFILE，用 PFILE 启动数据库即可：

```
[oracle@jumper oracle]$ sqlplus "/ as sysdba"
Connected to an idle instance.
SQL> create pfile from spfile;
File created.
```

如果在 Windows 上，如果无法启动服务，连接数据库可能收到如下错误：

```
C:\>sqlplus "/ as sysdba"
SQL*Plus: Release 10.2.0.4.0 - Production on 星期二 7 月 15 10:48:04 2008
Copyright (c) 1982, 2007, Oracle.  All Rights Reserved.

ERROR:
ORA-12560: TNS: 协议适配器错误
```

在这种情况下，如果没有任何可供参考的参数文件，可以用记事本（Notepad）编辑一个参数文件，位置在$ORACLE_HOME/dbs（Windows 为 database 目录下）包含如下两行：

```
[oracle@test126 dbs]$ cat initeygle.ora
SPFILE='/opt/oracle/product/10.2.0/dbs/spfileeygle.ora'
sga_max_size=1073741824
```

第一行指向 SPFILE，第二行写上出错的参数，给一个正确的值。这个值在实例启动时会覆盖之前错误的设置。然后就可以使用这个文件启动数据库实例了：

```
SQL> startup pfile=$ORACLE_HOME/dbs/initeygle.ora
ORACLE instance started.
Database mounted.
Database opened.
```

如果在 Windows 上，你只能通过服务起停数据库，那么 Oracle 缺省的还是会寻找 SPFILE，一个办法是将 SPFILE 改名。比如将 spfileeygle.ora 更改为 spfileeygle2.ora,然后再 pfile 里引用这个参数文件，下次 startup 就不用指定 pfile。

数据库可以自动找到这个参数文件，启动数据库。

```
[oracle@test126 dbs]$ mv spfileeygle.ora spfileeygle2.ora
[oracle@test126 dbs]$ cat initeygle.ora
```

```
SPFILE='/opt/oracle/product/10.2.0/dbs/spfileeygle2.ora'
sga_max_size=1073741824
```

数据库可以自动使用 PFILE 启动：

```
SQL> startup
ORACLE instance started.
Database mounted.
Database opened.
SQL> show parameter spfile
NAME             TYPE        VALUE
---------------- ----------- ------------------------------------------------
spfile           string      /opt/oracle/product/10.2.0/dbs/spfileeygle2.ora
```

3.2.9 重置 spfile 中设置的参数

虽然并不常用，但是 Oracle 仍然提供了重置参数的方法。当我们想恢复某个参数为缺省值时，可以使用如下命令：

```
alter system reset parameter <scope=memory|spfile|both> sid='sid|*'
```

该命令通常用于 RAC 环境中，在单实例环境中，需要指定 sid='*' ,reset 一个参数，Oracle 将从 spfile 文件中去除该参数：

```
[oracle@jumper dbs]$ strings spfileconner.ora |grep open
*.open_cursors=150
[oracle@jumper dbs]$ sqlplus "/ as sysdba"

SQL> alter system reset open_cursors scope=spfile sid='*';
System altered.
SQL> exit
Disconnected from Oracle9i Enterprise Edition Release 9.2.0.4.0 - Production
With the Partitioning option
JServer Release 9.2.0.4.0 - Production
[oracle@jumper dbs]$ strings spfileconner.ora |grep open
```

可以看到，reset 之后 open_cursors 参数在 spfile 文件中不再存在。

3.2.10是否使用了 spfile

判断是否使用了 SPFILE，可以使用以下方法：

1．查询 v$parameter 动态视图，如果以下查询返回空值，那么你在使用 pfile.

```
SQL>  SELECT name,value FROM v$parameter WHERE name='spfile';
NAME                           VALUE
```

```
------------------------------ --------------------------------------------------
spfile                         ?/dbs/spfile@.ora
```

2. 或者你可以使用 SHOW 命令（实际上，show 命令的结果同样来自 v$parameter 视图）来显示参数设置，如果以下结果 value 列返回空值，那么说明你在使用 pfile:

```
SQL> show parameter spfile
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
spfile                               string      ?/dbs/spfile@.ora
```

3. 查询 v$spparameter 视图

如果以下查询返回 0 值，表示你在使用 pfile,否则表明你使用的是 spfile:

```
SQL> SELECT COUNT(*) FROM v$spparameter WHERE value IS NOT NULL;
  COUNT(*)
----------
        32
```

或者使用以下查询，如果 true 值返回非 0 值，那么说明我们使用的是 spfile.

```
SQL> select isspecified, count(*) from v$spparameter group by isspecified;
ISSPECIFIED    COUNT(*)
------------ ----------
FALSE               226
TRUE                 33
```

3.2.11SPFILE 的备份与恢复

在本章开篇我们提到，Oracle 把 Spfile 也纳入到 Rman 的备份恢复策略当中，如果你配置了控制文件自动备份(autoback)，那么 Oracle 会在数据库发生重大变化(如增减表空间)时自动进行控制文件及 Spfile 文件的备份。

下面来看一下这个过程:

1. 设置控制文件自动备份

在 RMAN 命令行，通过以下命令可以启用控制文件的自动备份：

```
[oracle@jumper oracle]$ rman target /
connected to target database: HSJF (DBID=1052178311)
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON;

using target database controlfile instead of recovery catalog
old RMAN configuration parameters:
CONFIGURE CONTROLFILE AUTOBACKUP OFF;
new RMAN configuration parameters:
CONFIGURE CONTROLFILE AUTOBACKUP ON;
```

```
new RMAN configuration parameters are successfully stored
```

这个设置可以在数据库中通过如下方式查询得到:

```
SQL> select * from v$rman_configuration;
     CONF# NAME                      VALUE
---------- ------------------------ ----------
         1  CONTROLFILE AUTOBACKUP    ON
```

2．更改自动备份的位置

在 Oracle10g 中，如果我们使用了闪回区，那么控制文件的自动备份会存储在闪回区中。有时候为了安全，我们需要将控制文件的自动备份转移到其他目录下，这可以使用如下命令。

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP
  2 FORMAT FOR DEVICE TYPE DISK TO '/opt/oracle/obak/control%F';

new RMAN configuration parameters:
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '/opt/oracle/obak/control%F';
new RMAN configuration parameters are successfully stored
```

3．检查自动备份

自动备份的参数文件或控制文件，可以通过视图 v$backup_spfile 来检查：

```
SQL> select * from v$backup_spfile;
RECID      STAMP  SET_STAMP  SET_COUNT MODIFICATION BYTES COMPLETION_T DB_UNIQUE_NAME
----- ---------- ---------- ---------- ------------ ----- ------------ --------------
  286  660985844  660985841        857 07-JAN-08        2 25-JUL-08    PHSDB
  287  661071625  661071622        859 07-JAN-08        2 26-JUL-08    PHSDB
  288  661158029  661158026        861 07-JAN-08        2 27-JUL-08    PHSDB
```

输出显示，当前存在三份控制文件及参数文件的备份，通过 RMAN 的命令行可以列出这些备份集：

```
[oracle@wapdb trace]$ rman target /

Recovery Manager: Release 11.1.0.6.0 - Production on Sun Jul 27 21:22:19 2008
Copyright (c) 1982, 2007, Oracle.  All rights reserved.
connected to target database: PHSDB (DBID=1915040266)
RMAN> list backup of spfile;
using target database control file instead of recovery catalog
List of Backup Sets
===================
BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
833     Full    9.64M      DISK        00:00:05     25-JUL-08
```

```
        BP Key: 833   Status: AVAILABLE  Compressed: NO  Tag: TAG20080725T071041
        Piece Name: /FRA/PHSDB/autobackup/2008_07_25/o1_mf_s_660985841_48l2zoxx_.bkp
  SPFILE Included: Modification time: 07-JAN-08
  SPFILE db_unique_name: PHSDB
BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
835     Full    9.64M      DISK        00:00:06     26-JUL-08
        BP Key: 835   Status: AVAILABLE  Compressed: NO  Tag: TAG20080726T070022
        Piece Name: /FRA/PHSDB/autobackup/2008_07_26/o1_mf_s_661071622_48nprcn6_.bkp
  SPFILE Included: Modification time: 07-JAN-08
  SPFILE db_unique_name: PHSDB
BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
837     Full    9.64M      DISK        00:00:05     27-JUL-08
        BP Key: 837   Status: AVAILABLE  Compressed: NO  Tag: TAG20080727T070026
        Piece Name: /FRA/PHSDB/autobackup/2008_07_27/o1_mf_s_661158026_48qc4h0y_.bkp
  SPFILE Included: Modification time: 07-JAN-08
  SPFILE db_unique_name: PHSDB
```

4．记录数据库变化

下面创建一个测试表空间，看看数据库在变化时对于参数文件及控制文件的自动备份：

```
SQL> create tablespace eygle
  2  datafile '/data1/oracle/oradata/eygle01.dbf' size 5M;
Tablespace created.
```

这时候检查告警日志文件，你可以在其中发现这样的自动备份信息：

```
Sat Jan 17 00:55:57 2004
Starting control autobackup
Control autobackup written to DISK device
        handle '/opt/oracle/product/9.2.0/dbs/c-1052178311-20040117-00'
Completed: create tablespace eygle
datafile '/data1/oracle/oradata/eygle01.dbf'
```

如果使用 RMAN 进行备份，在提示中你可以看到如下信息：

```
RMAN> run
2> {
3> allocate channel ch1 type disk format='e:\oracle\orabak\eygle%t.arc';
4> backup archivelog all delete all input;
5> release channel ch1;
6> }
```

……

```
piece handle=E:\ORACLE\ORABAK\EYGLE511712693.ARC comment=NONE
channel ch1: backup set complete, elapsed time: 00:00:03
channel ch1: deleting archive log(s)
……
Finished backup at 02-DEC-03

Starting Control File and SPFILE Autobackup at 02-DEC-03
piece handle=E:\ORACLE\ORA92\DATABASE\C-3627775766-20031202-01 comment=NONE
Finished Control File and SPFILE Autobackup at 02-DEC-03
```

在 Oracle9i 中自动备份的控制文件及 spfile 文件缺省的格式及命名规则如下：

c-IIIIIIIII-YYYYMMDD-QQ

c ----------控制文件

IIIIIIIII---------DBID

YYYYMMDD----------时间戳

QQ----------序号 00-FF，16 进制表示

5．恢复

使用自动备份恢复 spfile 文件:

```
RMAN> restore spfile to '/tmp/spfileeygle.ora' from autobackup;
Starting restore at 17-JAN-04
using target database controlfile instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: sid=18 devtype=DISK
channel ORA_DISK_1: looking for autobackup on day: 20040117
channel ORA_DISK_1: autobackup found: c-1052178311-20040117-01
channel ORA_DISK_1: SPFILE restore from autobackup complete
Finished restore at 17-JAN-04
```

检查一下，恢复的 SPFILE 文件自动生成到指定位置：

```
[oracle@jumper bdump]$ ls -l /tmp/spfileeygle.ora
-rw-r----- 1 oracle  dba 3584  1月 17 09:34 /tmp/spfileeygle.ora
```

同样可以通过这种方法恢复自动备份的控制文件，示例如下:

```
RMAN> restore controlfile to '/tmp/control01.ctl' from autobackup;
Starting restore at 17-JAN-04
using target database controlfile instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: sid=10 devtype=DISK
channel ORA_DISK_1: looking for autobackup on day: 20040117
channel ORA_DISK_1: autobackup found: c-1052178311-20040117-02
channel ORA_DISK_1: controlfile restore from autobackup complete
```

```
Finished restore at 17-JAN-04
RMAN> exit
Recovery Manager complete.
[oracle@jumper bdump]$ ls -l /tmp/control*
-rw-r-----    1 oracle   dba       1892352  1月 17 09:44 /tmp/control01.ctl
```

从 Oracle9i 开始的自动备份控制文件的功能给我们带来了极大的收益，通过自动备份,在数据库出现紧急状况的时候,你可能可以从这个自动备份中获得更为有效及时的控制文件。

有一点还要说明的是，如果数据库无法 Mount，是不能使用如上方式恢复自动备份的控制文件或者参数文件。

```
[oracle@jumper dbs]$ rman target /
Recovery Manager: Release 9.2.0.4.0 - Production
Copyright (c) 1995, 2002, Oracle Corporation.  All rights reserved.
connected to target database: conner (not mounted)
RMAN> restore controlfile to '/tmp/control01.ctl' from autobackup;
Starting restore at 08-MAR-06
using target database controlfile instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: sid=11 devtype=DISK
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of restore command at 03/08/2006 11:38:29
RMAN-06495: must explicitly specify DBID with SET DBID command
```

此时，Oracle 需要提供数据库的 DBID，才能找到相应的自动备份用以恢复，如果无法得知 DBID，那么可以直接指定自动备份集来进行恢复：

```
RMAN> restore controlfile to '/tmp/control01.ctl' from
'c-3152029224-20051221-00';
Starting restore at 08-MAR-06
using target database controlfile instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: sid=9 devtype=DISK
channel ORA_DISK_1: restoring controlfile
channel ORA_DISK_1: restore complete
Finished restore at 08-MAR-06
```

进一步的，如果丢失了一个数据库的所有信息,数据库实例甚至无法 nomount 启动,此时可以手工临时编辑一个 pfile 文件启动实例，或者采用第一章介绍的方法，使用 RMAN 启动默认实例，进行 spfile 文件恢复。启动默认实例过程如下：

```
[oracle@jumper dbs]$ rman target /
Recovery Manager: Release 9.2.0.4.0 - Production
```

```
Copyright (c) 1995, 2002, Oracle Corporation.  All rights reserved.
connected to target database (not started)

RMAN> startup nomount;

startup failed: ORA-01078: failure in processing system parameters
LRM-00109: could not open parameter file '/opt/oracle/product/9.2.0/dbs/initconner.ora'

trying to start the Oracle instance without parameter files ...
Oracle instance started
```

然后可以通过备份集来恢复 spfile 文件或者控制文件：

```
[oracle@jumper log]$ rman target /
Recovery Manager: Release 9.2.0.4.0 - Production
Copyright (c) 1995, 2002, Oracle Corporation.  All rights reserved.
connected to target database: DUMMY (not mounted)
RMAN> restore spfile to '/tmp/spfile.ora' from 'c-3152029224-20060509-00';
Starting restore at 09-MAY-06

using target database controlfile instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: sid=9 devtype=DISK
channel ORA_DISK_1: autobackup found: c-3152029224-20060509-00
channel ORA_DISK_1: SPFILE restore from autobackup complete
Finished restore at 09-MAY-06
```

最后还要强调的是，缺省的,这个自动备份功能是关闭的，强烈推荐大家用上面提到的方法打开该功能。

很不幸，我们曾经遇到过这样的案例，$ORACLE_HOME/dbs 目录下的所有文件都被误删除，这样自动备份的参数文件及当前的 spfile 文件等都失去了。此时需要怎样恢复参数文件呢？

首先可以从创建数据库时的 init.ora 文件创建一个 spfile，在 postDBCreation.sql 中我们可以找到这样一行（这是一个 Oracle10g 的数据库）：

```
create spfile='/opt/oracle/product/10.2.0/dbs/spfileorder.ora'
FROM pfile='/opt/oracle/admin/order/scripts/init.ora';
```

由于当前数据库是以 spfile 启动的，直接执行这个语句会出现错误：

```
SQL> create spfile='/opt/oracle/product/10.2.0/dbs/spfileorder.ora'
FROM pfile='/opt/oracle/admin/order/scripts/init.ora';
create spfile='/opt/oracle/product/10.2.0/dbs/spfileorder.ora' FROM
pfile='/opt/oracle/admin/order/scripts/init.ora'
*
```

```
ERROR at line 1:
ORA-32002: cannot create SPFILE already being used by the instance
```

此时我们可以以另外的名字创建 spfile，再通过更名方式恢复 spfile:

```
SQL> create spfile='/opt/oracle/product/10.2.0/dbs/spfile1.ora' FROM
pfile='/opt/oracle/admin/order/scripts/init.ora';

File created.
[oracle@order scripts]$ cd $ORACLE_HOME/dbs
[oracle@order dbs]$ mv spfile1.ora spfileorder.ora
```

这样就成功恢复了 spfile，但是注意，可能很多参数已经发生了变化，需要重新设置。在这个案例中，我们使用了 Oracle10g 的 OMF 特性，所以还需要纠正几个至关重要的参数设置，从 alert 文件中可以获得更改日志：

```
Tue Sep  5 15:23:48 2006
ALTER SYSTEM SET db_create_file_dest='/data1/oradata' SCOPE=SPFILE;
Tue Sep  5 15:26:54 2006
ALTER SYSTEM SET
control_files='/data1/oradata/ORDER/controlfile/o1_mf_28spy45z_.ctl','/data1/oradata/ORDE
R/controlfile/o2_mf_28spy45z_.ctl' SCOPE=SPFILE;
```

其他重要参数可以从当前数据库实例获得并写回 spfile。如果仅仅是参数文件丢失，数据库仍然在运行，那我们完全可以从数据库实例中得到当前的所有运行参数。

通过以下过程可以获得当前实例的非缺省参数，将这些参数 spool 到一个文件中：

```
set linesize 120
set pagesize 999
set heading off
set feedback off
spool /tmp/inittmp.ora
select '*.'||name||' = ' || value from v$parameter  where isdefault = 'FALSE';
spool off
```

对生成的临时参数文件进行适当修改，就可以用于启动数据库，以下是生成临时文件的摘要示例：

```
*.processes = 500
*.sga_max_size = 1073741824
*.spfile = /opt/oracle/product/10.2.0/dbs/spfileeygle.ora
*.sga_target = 5368709120
*.db_block_size = 8192
*.compatible = 10.2.0.1.0
*.db_file_multiblock_read_count = 16
*.db_create_file_dest = /opt/oracle/oradata
*.log_checkpoints_to_alert = TRUE
```

```
*.undo_management = AUTO
*.undo_tablespace = UNDOTBS1
*.undo_retention = 900
```

3.2.12如何设置 Events 事件

Events 事件是 Oracle 的重要诊断工具及问题解决办法，很多时候需要通过 Events 设置来屏蔽或者更改 Oracle 的行为，下面我们来看一下怎样修改 spfile，增加 Events 事件设置：

```
SQL> alter system set event='10841 trace name context forever' scope=spfile;
System altered.
SQL> startup force;
SQL> show parameter event
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
event                                string      10841 trace name context forever
```

顺便提一句，10841 事件是用于解决 Oracle9i 中 JDBC Thin Driver 问题的一个方法，如果你的 alert.log 文件中出现以下错误提示:

```
Wed Jan  7 17:17:08 2004
Errors in file /opt/oracle/admin/phsdb/udump/phsdb_ora_1775.trc:
ORA-00600: internal error code, arguments: [ttcgcshnd-1], [0], [], [], [], [], [], []
Wed Jan  7 17:17:18 2004
Errors in file /opt/oracle/admin/phsdb/udump/phsdb_ora_1777.trc:
ORA-00600: internal error code, arguments: [ttcgcshnd-1], [0], [], [], [], [], [], []
```

那么，很不幸，你很可能是遇到了 bug: 1725012，通过设置以上事件，可以屏蔽和解决这个 ORA-00600 错误。具体你可以参考 Metalink 相关文档。

如果想取消 event 参数设置，同样可以参用 reset 参数的方法：

```
SQL> show parameter event
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
event                                string      10046 trace name context forev
                                                 er,level 12
SQL> alter system reset event scope=spfile sid='*';
System altered.
SQL> startup force;
SQL> show parameter event
NAME                  TYPE        VALUE
--------------------- ----------- ------------------------------
event                 string
```

3.2.13导出 SPFILE 文件

SPFILE 文件可以导出为文本文件，使用导出、修改文本文件、再创建 SPFILE 的过程可以实现向 SPFILE 中添加参数。

```
SQL> create pfile='e:\initeyglen.ora' from spfile;
文件已创建。
```

生成的初始化参数文件 initeyglen.ora 内容大致如下：

```
*.aq_tm_processes=1
*.background_dump_dest='e:\oracle\admin\eyglen\bdump'
*.compatible='9.2.0.0.0'
*.control_files='e:\oracle\oradata\eyglen\control01.ctl',
'e:\oracle\oradata\eyglen\control02.ctl',
'e:\oracle\oradata\eyglen\control03.ctl'
*.core_dump_dest='e:\oracle\admin\eyglen\cdump'
*.db_block_size=8192
*.db_cache_size=25165824
……
```

生成了 pfile 之后，可以使用这个 pfile，或者手动修改其中的参数以启动数据库。比如在更改数据库的归档模式时，修改这个 pfile，增加一行：

```
*.log_archive_start=true
```

使用这个 PFILE 启动数据库

```
SQL> startup  pfile='e:\initeyglen.ora'
ORACLE 例程已经启动。
……
数据库已经打开。
SQL> show parameter log_archive_start
NAME                 TYPE     VALUE
------------------------------ ------------------ ------------------
log_archive_start          boolean   TRUE
```

然后可以使用新的 PFILE 创建 SPFILE

```
SQL> create spfile from pfile='e:\initeyglen.ora';
文件已创建。
```

重新启动数据库，新的 SPFILE 生效。

```
SQL> startup
SQL> show parameter spfile
NAME                  TYPE     VALUE
-------------------- -------  ----------------------------------------------
spfile                string   %ORACLE_HOME%\DATABASE\SPFILE%ORACLE_SID%.ORA
SQL> show parameter log_archive_start
```

```
NAME                 TYPE     VALUE
-------------------- -------  ----------------------
log_archive_start    boolean  TRUE
```

需要提醒的是，在 Oracle10g 之前，LOG_ARCHIVE_START 控制数据库可否自动归档，归档模式下这个参数应当设置为 TRUE，很多用户在修改数据库的归档模式时常常忘记修改 LOG_ARCHIVE_START 参数，结果导致数据库重新启动后无法自动归档，最后挂起，影响服务。

最终 Oracle 认识到了这个问题，从 Oracle10g 开始，修改数据库的归档模式不需要再设置 LOG_ARCHIVE_START 参数：

```
$ sqlplus "/ as sysdba"
Connected to:
Oracle Database 10g Enterprise Edition Release 10.1.0.3.0 - 64bit Production
With the Partitioning and Data Mining options

SQL> archive log list;
Database log mode              No Archive Mode
Automatic archival             Disabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     25
Current log sequence           27
SQL> show parameter log_archive_start
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
log_archive_start                    boolean     FALSE
SQL> shutdown immediate;
SQL> startup mount;
SQL> alter database archivelog;
SQL> alter database open;
Database altered.
SQL> archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence      25
Next log sequence to archive    27
Current log sequence            27
```

通过以上示例我们可以清晰的看到 Oracle10g 中的这一改变。

3.3参数文件诊断案例之一

这是实际生产系统中关于 spfile 的一个案例问题，具体的问题诊断和主要解决步骤说明如下。系统情况说明：

- 操作系统：SUN Solaris 8
- 数据库版本：Oracle 9.2.0.3
- 问题描述：工程人员报告，数据库在重新启动时无法正常启动，检查发现 UNDO 表空间丢失。

3.3.1. 登陆系统检查告警日志文件文件

在此有必要简单的介绍一下告警日志文件。告警日志文件由按时间顺序排列的消息和错误的记录组成。

下列信息会记录在警报日志文件中

- 内部错误 (ORA-600，ORA-07445 等错误信息) 和块损坏错误 (ORA-1578)
- 影响数据库结构和参数的操作和诸如 CREATE DATABASE，STARTUP，SHUTDOWN，ARCHIVE LOG 和 RECOVER 之类的语句
- 例程启动时所有非缺省的初始化参数值

数据库管理员定期检查告警日志文件是很重要的，这样就可以在问题变得严重之前发现它们，并及时处理，在我们的生产环境中，告警日志按照以 5 分钟为间隔进行检测，如果发现任何错误提示或警报信息，就发送邮件给数据库管理员，请求人为介入管理。

由于告警日志文件是不停累计的，所以在检查以后可以按照规定备份或整理告警日志文件。告警日志文件的存储位置由始化参数 BACKGROUND_DUMP_DEST 定义：

```
SQL> show parameter background_dump_dest
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
background_dump_dest                 string      /opt/oracle/admin/danaly/bdump
```

其缺省文件名为 alert_<*ORACLE_SID*>.log

> 注意：由于告警日志文件的重要作用，当数据库出现故障时，通常我们最先的处理步骤是检查该文件，以发现相关错误信息或线索，快速找到问题所在。这是 DBA 必须明确的一个知识点。

此次故障诊断，首先检查告警日志文件，发现其中包含如下错误信息：

```
Thu Apr  1 11:11:28 2004
Errors in file /oracle/admin/gzhs/udump/gzhs_ora_27781.trc:
ORA-30012: undo tablespace 'UNDOTBS1' does not exist or of wrong type
Thu Apr  1 11:11:28 2004
Error 30012 happened during db open, shutting down database
USER: terminating instance due to error 30012
Instance terminated by USER, pid = 27781
ORA-1092 signalled during: alter database open...
```

在告警日志末尾显示了数据库在 Open 状态因为错误而异常终止.最后出错的错误号是 ORA-30012,该错误的含义是：

```
[oracle@jumper oracle]$ oerr ora 30012
30012, 00000, "undo tablespace '%s' does not exist or of wrong type"
// *Cause:   the specified undo tablespace does not exist or of the
//           wrong type.
// *Action:  Correct the tablespace name and reissue the statement.
```

这说明是 UNDO 表空间不存在导致出现问题。

3.3.2.　尝试重新启动数据库

尝试重新启动数据库，检查问题是否仍然存在：

```
bash-2.03$ sqlplus "/ as sysdba"

SQL*Plus: Release 9.2.0.3.0 - Production on 星期四 4 月 1 11:43:52 2004
Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.
已连接到空闲例程。

SQL> startup
ORACLE 例程已经启动。

Total System Global Area 4364148184 bytes
Fixed Size                   736728 bytes
Variable Size            1845493760 bytes
Database Buffers         2516582400 bytes
Redo Buffers                1335296 bytes
数据库装载完毕。
ORA-01092: ORACLE 例程终止。强行断开连接
```

在 Open 步骤，例程终止，问题重现。

3.3.3.　检查数据文件

既然是 UNDO 表空间丢失，接下来需要确认一下相关数据文件，看 UNDO 表空间数据文件是否存在：

```
bash-2.03$ cd /u01/ oradata/gzhs
bash-2.03$ ls -l
total 55702458
-rw-r-----   1 oracle   dba      1073750016 Apr  1 11:44 UNDOTBS2.dbf
-rw-r-----   1 oracle   dba      1073750016 Apr  1 11:44 WAP12_BILLINGDETAIL.dbf
-rw-r-----   1 oracle   dba      1073750016 Apr  1 11:44 WAP12_MAIN.dbf
……
-rw-r-----   1 oracle   dba      1073750016 Apr  1 11:44 WAP12_MVIEW.dbf
```

通过检查发现存在文件 **UNDOTBS2.dbf** ，大小约为 1G。

3.3.4. mount 数据库，检查系统参数

既然存在一个 UNDO 表空间文件，用户又没有主动执行删除操作，那么极其可能的就是参数设置出了问题，将数据库启动到 Mount 状态，检查一下当前参数设置：

```
SQL> startup mount;
ORACLE 例程已经启动。
……
数据库装载完毕。
SQL> select name from v$datafile;
NAME
--------------------------------------------------------------------------------
/u01/oradata/gzhs/system01.dbf
……..
/u01/oradata/gzhs/UNDOTBS2.dbf
SQL> show parameter undo
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
undo_management                      string      AUTO
undo_retention                       integer     10800
undo_suppress_errors                 boolean     FALSE
undo_tablespace                      string      UNDOTBS1
SQL> show parameter spfile
NAME                                   TYPE        VALUE
-------------------------------------- ----------- ------------------------------
spfile                                 string
```

检查发现系统没有使用 spfile,，初始化参数设置的 undo 表空间为 UNDOTBS1，数据库中登记的 UNDO 文件为 **UNDOTBS2.dbf**。

3.3.5. 检查参数文件

检查一下 pfile 文件中的设置，发现 undo 表空间参数设置的是 UNDOTBS1.

```
bash-2.03$ cd $ORACLE_HOME/dbs
bash-2.03$ vi initgzhs.ora
"initgzhs.ora" [Incomplete last line] 105 lines, 3087 characters
……
########################################
# System Managed Undo and Rollback Segments
########################################
undo_management=AUTO
undo_retention=10800
undo_tablespace=UNDOTBS1
```

这个设置是极其可疑的，也就是说，参数设置可能和数据库的实际情况不符。

3.3.6.　再次检查 alert 文件

告警日志文件中记录了对于数据库重要操作的信息，可以从中查找对于 UNDO 表空间的操作。

第一部分，创建数据库时的信息:

```
Sat Feb  7 20:30:12 2004
CREATE DATABASE gzhs
MAXINSTANCES 1 MAXLOGHISTORY 1 MAXLOGFILES 5 MAXLOGMEMBERS 3 MAXDATAFILES 100
DATAFILE '/u01/oradata/gzhs/system01.dbf' SIZE 500M REUSE AUTOEXTEND ON NEXT  10240K MAXSIZE
UNLIMITED EXTENT MANAGEMENT LOCAL
DEFAULT TEMPORARY TABLESPACE TEMP TEMPFILE '/u01/oradata/gzhs/temp01.dbf' SIZE 1000M REUSE
AUTOEXTEND ON NEXT  250M MAXSIZE UNLIMITED
UNDO TABLESPACE "UNDOTBS1" DATAFILE '/u01/oradata/gzhs/undotbs01.dbf' SIZE 1000M
REUSE AUTOEXTEND ON NEXT  100M MAXSIZE UNLIMITED
CHARACTER SET ZHS16GBK
NATIONAL CHARACTER SET AL16UTF16
LOGFILE GROUP 1 ('/u01/oradata/gzhs/redo01.log') SIZE 256M,
GROUP 2 ('/u01/oradata/gzhs/redo02.log') SIZE 256M,
GROUP 3 ('/u01/oradata/gzhs/redo03.log') SIZE 256M
```

第二部分,发现创建 UNDOTBS2 的记录信息:

```
Wed Mar 24 20:20:58 2004
/* OracleOEM */ CREATE UNDO TABLESPACE "UNDOTBS2" DATAFILE
'/u01/oradata/gzhs/UNDOTBS2.dbf' SIZE 1024M AUTOEXTEND ON NEXT  100M MAXSIZE
UNLIMITED
Wed Mar 24 20:22:37 2004
Created Undo Segment _SYSSMU11$
Created Undo Segment _SYSSMU12$
Created Undo Segment _SYSSMU13$
Completed: /* OracleOEM */ CREATE UNDO TABLESPACE "UNDOTBS2"
Wed Mar 24 20:24:25 2004
Undo Segment 11 Onlined
Undo Segment 12 Onlined
Undo Segment 13 Onlined
Successfully onlined Undo Tablespace 15.
Undo Segment 1 Offlined
Undo Segment 2 Offlined
Undo Segment 3 Offlined
Undo Tablespace 1 successfully switched out.
```

第三部分，新的 UNDO 表空间被应用

```
Wed Mar 24 20:24:25 2004
ALTER SYSTEM SET undo_tablespace='UNDOTBS2' SCOPE=MEMORY;
```

问题就在这里,创建了新的 UNDO 表空间以后，因为使用的是 pfile 文件，切换表空间的修改的只对当前实例生效，操作人员忘记了修改 pfile 文件。

如果使用 spfile，缺省的修改范围是 both,会同时修改 spfile 文件，就可以避免以上问题的出现。

第四部分，删除了 UNDOTBS1 的信息

```
Wed Mar 24 20:25:01 2004
/* OracleOEM */ DROP TABLESPACE "UNDOTBS1" INCLUDING CONTENTS  AND DATAFILES
CASCADE CONSTRAINTS
Wed Mar 24 20:25:03 2004
Deleted file /u01/oradata/gzhs/undotbs01.dbf
Completed: /* OracleOEM */ DROP TABLESPACE "UNDOTBS1" INCLUDI
```

这样再次重新启动数据库的时候，问题出现了，pfile 中定义的 UNDOTBS1 找不到了，而且操作实在很久以前，没人能回忆起来，甚至无法得知是什么人的操作。

3.3.7. 修正 pfile

找到了问题的根源，解决起来就简单了，我们修改 pfile 参数文件，就可以启动数据库：

```
bash-2.03$ vi initgzhs.ora
"initgzhs.ora" [Incomplete last line] 105 lines, 3087 characters
……
########################################
# System Managed Undo and Rollback Segments
########################################
undo_management=AUTO
undo_retention=10800
undo_tablespace=UNDOTBS2

~
"initgzhs.ora" 105 lines, 3088 characters
```

3.3.8. 启动数据库

重新启动，数据库可以正常打开，问题解决：

```
bash-2.03$ sqlplus "/ as sysdba"

SQL*Plus: Release 9.2.0.3.0 - Production on 星期四 4月 1 11:55:11 2004
Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.
```

```
SQL> select * from v$version where rownum <2;
BANNER
----------------------------------------------------------------
Oracle9i Enterprise Edition Release 9.2.0.3.0 - 64bit Production
```

经过了一系列的诊断过程，问题得以完满解决。在这里我们可以看到，使用 spfile 可以免去手工修改 pfile 文件的麻烦，减少了犯错的可能。

3.4RAC 环境参数文件诊断案例

在 RAC 环境下，如果两个实例参数不一致，则可能导致某个数据库节点无法启动，以下是一个实际诊断案例。

3.4.1.　数据库资源异常

在一次数据库维护中，当关闭数据库之后，重新启动，结果发现一个节点出现异常，在 crs_stat 的输出中发现节点 2 的一个服务状态为 OFFLINE：

```
[oracle@smsdbrac2 client]$ crs_stat -t
Name           Type           Target    State     Host
------------------------------------------------------------
ora.smsdb.db   application    ONLINE    ONLINE    smsdbrac2
ora....b1.inst application    ONLINE    ONLINE    smsdbrac1
ora....b2.inst application    ONLINE    ONLINE    smsdbrac2
ora....srac.cs application    ONLINE    ONLINE    smsdbrac1
ora....db1.srv application    ONLINE    ONLINE    smsdbrac1
ora....db2.srv application    OFFLINE   OFFLINE
ora....SM1.asm application    ONLINE    ONLINE    smsdbrac1
ora....C1.lsnr application    ONLINE    ONLINE    smsdbrac1
ora....ac1.gsd application    ONLINE    ONLINE    smsdbrac1
ora....ac1.ons application    ONLINE    ONLINE    smsdbrac1
ora....ac1.vip application    ONLINE    ONLINE    smsdbrac1
ora....SM2.asm application    ONLINE    ONLINE    smsdbrac2
ora....C2.lsnr application    ONLINE    ONLINE    smsdbrac2
ora....ac2.gsd application    ONLINE    ONLINE    smsdbrac2
ora....ac2.ons application    ONLINE    ONLINE    smsdbrac2
ora....ac2.vip application    ONLINE    ONLINE    smsdbrac2
```

该资源具体信息为：

```
NAME=ora.smsdb.smsrac.smsdb2.srv
TYPE=application
TARGET=OFFLINE
```

```
STATE=OFFLINE
```

3.4.2. 问题的发现

发现这一问题后，尝试在命令行手工启动数据库，此时发现出现错误告警：

```
SQL> startup mount;
ORACLE instance started.

Total System Global Area 1241513984 bytes
Fixed Size                  1267212 bytes
Variable Size             318769652 bytes
Database Buffers          905969664 bytes
Redo Buffers               15507456 bytes
ORA-01105: mount is incompatible with mounts by other instances
ORA-19808: recovery destination parameter mismatch
```

这里的 ORA-19808 错误提示参数设置不一致，在 Oracle 的文档中，关于这一错误的具体信息是：

ORA-19808: recovery destination parameter mismatch

Cause: The value of parameters DB_RECOVERY_FILE_DEST and DB_RECOVERY_FILE_DEST_SIZE must be same in all instances.All databases must have same recovery destination parameters.

Action: Check DB_RECOVERY_FILE_DEST and DB_RECOVERY_FILE_DEST_SIZE values in all instances.

了解到这个原因后，接下来检查数据库的参数文件：

```
[oracle@smsdbrac2 ~]$ cd $ORACLE_HOME/dbs
[oracle@smsdbrac2 dbs]$ ls -al *smsdb2*
-rw-rw----  1 oracle dba     1544 May  5 16:05 hc_smsdb2.dat
-rw-r--r--  1 oracle dba     1521 Sep  9 11:03 initsmsdb2.ora
-rw-r-----  1 oracle dba     1536 Jul 23 10:06 orapwsmsdb2
-rw-r-----  1 oracle dba 15876096 Apr 15 16:56 snapcf_smsdb2.f
```

这时发现了 init 参数文件存在问题，正常情况下，这个参数应该包含一个指向 SPFILE 的链接，而现在这个参数文件却是一个本地的参数文件：

```
[oracle@smsdbrac2 dbs]$ tail -20 initsmsdb2.ora
*.db_recovery_file_dest_size=8589934592
*.dispatchers='(PROTOCOL=TCP) (SERVICE=smsdbXDB)'
smsdb1.instance_number=1
smsdb2.instance_number=2
*.job_queue_processes=10
smsdb1.local_listener='(ADDRESS=(PROTOCOL=TCP)(HOST=192.168.200.13)(PORT = 1521))'
smsdb2.local_listener='(ADDRESS=(PROTOCOL=TCP)(HOST=192.168.200.14)(PORT = 1521))'
```

```
*.open_cursors=300
*.pga_aggregate_target=413138944
*.processes=1000
*.remote_listener='LISTENERS_SMSDB'
*.remote_login_passwordfile='exclusive'
*.sessions=555
*.sga_target=1241513984
smsdb2.thread=2
smsdb1.thread=1
*.undo_management='AUTO'
smsdb1.undo_tablespace='UNDOTBS1'
smsdb2.undo_tablespace='UNDOTBS2'
*.user_dump_dest='/opt/oracle/admin/smsdb/udump'
```

3.4.3.　参数文件问题的解决

找到这个问题之后，可以根据参考实例 1 对这个参数文件进行修改：

```
[oracle@smsdbrac1 dbs]$ strings initsmsdb1.ora
SPFILE='+FLHDG/smsdb/spfilesmsdb.ora'
```

修改之后数据库实例可以启动。

这个问题的原因可能是因为在实例 2 上有人执行了类似 create pfile from spfile 的命令，这个命令创建的参数文件覆盖了原本正确的文件，导致了参数文件异常。

本地的参数文件可能会导致两个数据库实例的某些参数不一致，使得数据库的启动失败。

这个案例给我们的经验是：**在 RAC 环境中，每一个维护操作都要相当谨慎！**

参考文献及建议阅读：

Oracle Database Administrator's Guide 10g Release 2 (10.2)　　B14231-01

Oracle 创建实例的最少参数需求

http://www.eygle.com/archives/2006/05/intance_nomount_parameter.html

Oracle 中口令文件的作用及说明

http://www.eygle.com/faq/passwordfile.htm

Oracle9i 新特性 spfile

http://www.eygle.com/faq/Oracle9i.New.Feature.Spfile.01.htm