# 第 1 章　Oracle 数据库的创建

对于很多初学者来说，创建数据库通常是 Oracle 学习的开始，在完成软件安装之后，就可以开始创建数据库。数据库的创建可以通过 DBCA（Database Configuration Assistant）工具或者手工方式来完成，通常我们习惯使用 DBCA，但是我建议大家都能够尝试一下使用手工的方式进行数据库创建，因为那将使你更加了解 Oracle 数据库的创建过程。

如果从数据库创建进行深入，你会发现相关知识会延展到各个层面，由这一个点开始，广阔的 Oracle 知识会逐渐展现在你面前，本书就从这样一个起点开始。

由于 Oracle 10g 已经渐渐成为主流，本文以 Oracle 10g 为讲解模板，在实际建库的过程中，不管是在 Linux/UNIX 还是 Windows 上、不管是 Oracle 9i 还是 Oracle 10g，创建数据库的步骤都是基本相同的。

在安装 Oracle 软件的过程中，有一个类似如图 1-1 所示的界面（这里选择 Oracle 10gR2 版本，不同版本的界面可能有所不同）。

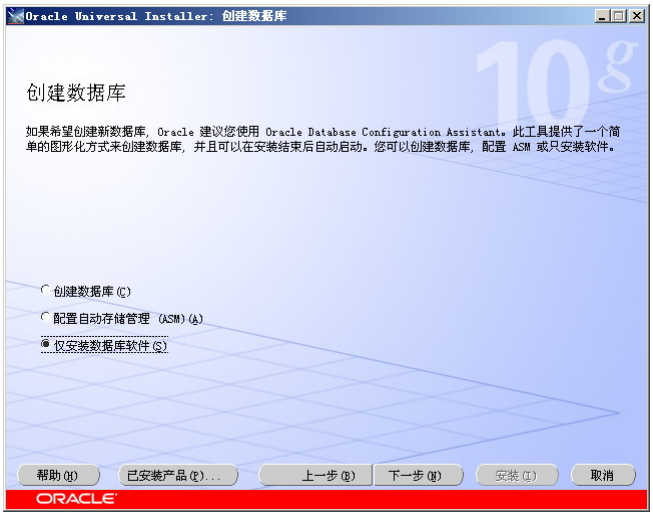


图 1-1　创建数据库

在这个页面中，可以选择在安装软件结束后同时创建数据库，也可以选择使用 ASM（Oracle 10g 选项），最后一项是“仅安装数据库软件”，建议大家选择此项，该选项可以将软件安装和数据库创建分离开来，这样既可以将独立的两个过程分步进行，又可以在安装软件后进行从容的检查和配置。

## 1.1 使用 DBCA 创建数据库

在完成 Oracle 软件安装之后，可以通过运行 DBCA（Database Configuration Assistant）来

启动数据库创建过程。

### 1.1.1 DBCA 的启动

DBCA 可以通过“开始”菜单中的选项来启动，也可以通过命令行方式启动，在命令行键入“dbca”则可以启动数据库创建助手界面，如图 1-2 所示。

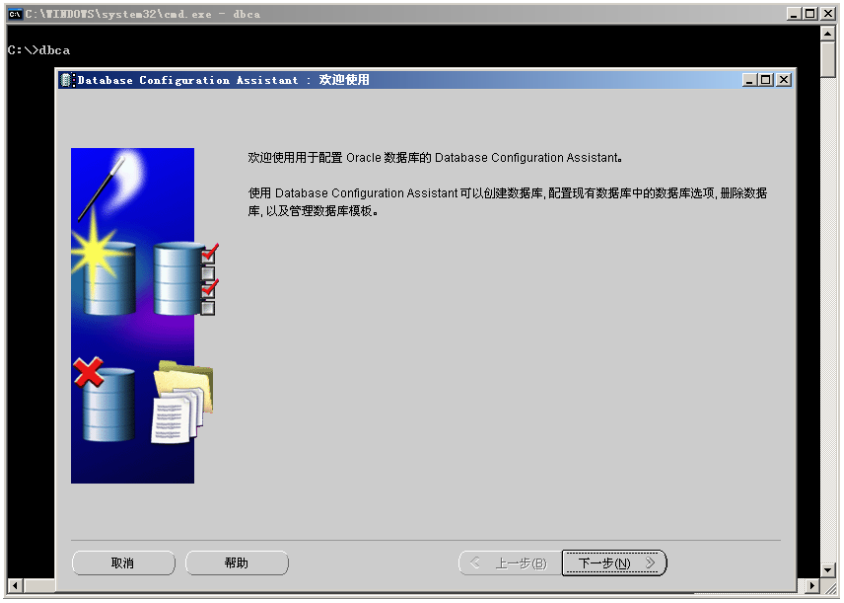


图 1-2　Database Configuration Assistant 欢迎界面

在 Windows 系统上，通过命令行来运行 dbca 命令，实际上调用的是$ORACLE_HOME\bin\下的 dbca.bat 批处理文件，可以简略地看一下这个批处理文件的内容（省略了部分内容）：

```
C:\>type C:\oracle\10.2.0\bin\dbca.bat
.....
@set OH=C:\oracle\10.2.0
@set JRE_CLASSPATH=C:\oracle\10.2.0\jdk\jre\lib\rt.jar
@set I18N_CLASSPATH=C:\oracle\10.2.0\jdk\jre\lib\i18n.jar
.....
"C:\oracle\10.2.0\jdk\jre\BIN\JAVA"  -DORACLE_HOME="%OH%"  -DJDBC_PROTOCOL=thin  -mx128m
oracle.sysman.assistants.dbca.Dbca
  %*
exit /B %ERRORLEVEL%
```

可以看到在设置了一系列的环境变量之后，通过调用 Java 运行时环境启动了 Java 工具 DBCA。

在 UNIX 系统中，原理同样类似。来看下面一段取自 Sun Solaris 环境下的代码：

```
bash-2.05$ uname -a
SunOS db210-rac2 5.9 Generic_117171-12 sun4u sparc SUNW,Sun-Fire-V210
```

```
bash-2.05$ which dbca
/opt/oracle/product/10.2.0/db/bin/dbca
```

摘录一点 DBCA 的代码：

```
bash-2.05$ more /opt/oracle/product/10.2.0/db/bin/dbca
#!/bin/sh -f
.....
# Classpath
JRE_CLASSPATH=$JRE_DIR/lib/$JRE_FILE
I18_CLASSPATH=$JRE_DIR/lib/$I18_FILE
EWT_CLASSPATH=$JLIB_DIR/$EWT_FILE:$JLIB_DIR/$EWT_COMP_FILE
SHARE_CLASSPATH=$JLIB_DIR/$SHARE_FILE
.....
# Run DBCA
$JRE_DIR/bin/java      -Dsun.java2d.font.DisableAlgorithmicStyles=true      -DORACLE_HOME=$OH
-DDISPLAY=$DISPLAY -DJDBC_PROTOCOL=thin -mx128m
-classpath $CLASSPATH oracle.sysman.assistants.dbca.Dbca $ARGUMENTS
```

同样最后一行命令启动了 Java 应用 DBCA 工具。

以上就是 DBCA 的初始化及启动。

## 1.1.2 配置数据库选项

启动 DBCA 之后，就可以通过图形界面进行数据库各项参数的配置，下面对几个重要步骤进行一点说明。

在如图 1-3 所示的选择数据库模板界面中，可以选择使用模板来创建数据库或者通过自定义方式来创建。

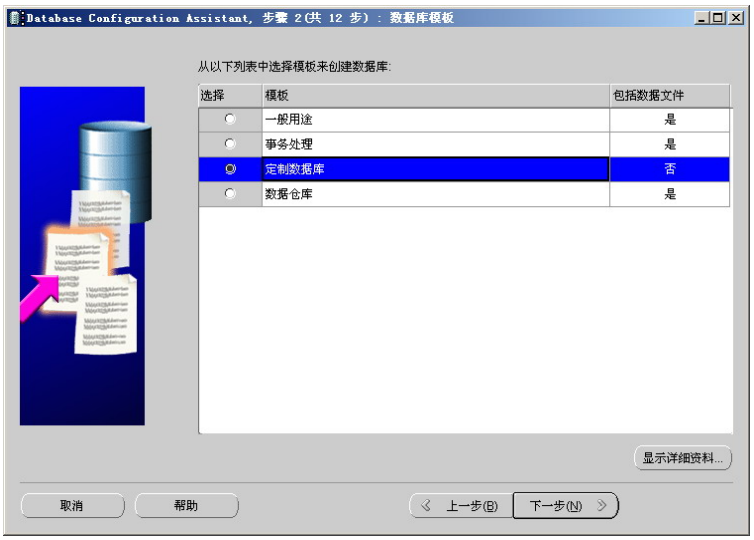


图 1-3　选择数据库模板

首先选择通过自定义方式创建数据库，在界面中选择“定制数据库”即可。定制数据库不包含数据文件，需要实时创建，使用模板则会包含已经创建好的数据文件。

单击“下一步”按钮，进入如图 1-4 所示的设置数据库标识界面，需要定义一个数据库名称和 SID，SID 在同一计算机上不能重复，用于唯一标识一个实例。


图 1-4　设置数据库标识

单击“下一步”按钮，进入如图 1-5 所示的管理选项界面，选择缺省配置“使用 Enterprise Manger 配置数据库”复选框即可，选择了这个选项，创建数据库时会自动配置 Oracle 10g 基于 Web 方式的 Database Control。

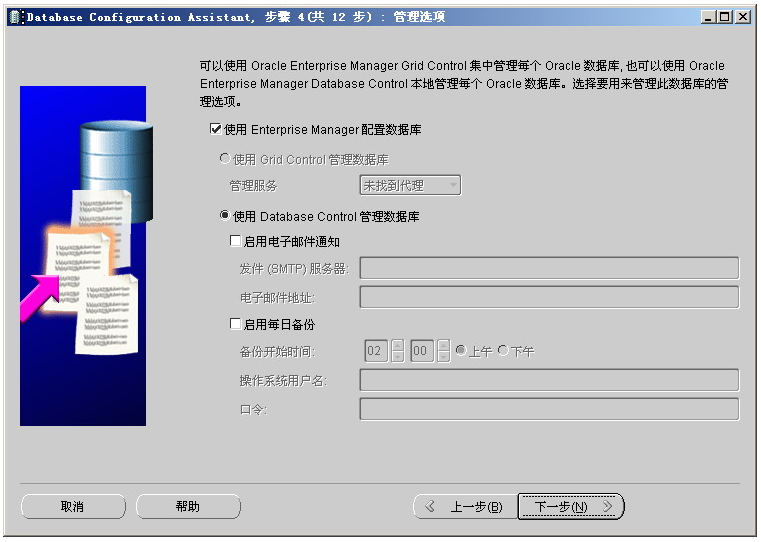


图 1-5　管理选项

我们知道在 Oracle 9i 之前，数据库会为数据库用户指定缺省口令，SYS 用户的缺省口令为 change_on_install，SYSTEM 的缺省口令为 manager，但是由于很多用户经常忘记修改缺省口令，进而可能为数据库留下安全隐患，所以从 Oracle 9i 开始，Oracle 要求用户在创建数据库时自行指定用户口令。因此在如图 1-6 所示的数据库身份证明界面中可以简单地为所有初始

用户定义一个缺省口令。


图 1-6　数据库身份证明

单击“下一步”按钮，进入如图 1-7 所示的存储选项界面，该界面用于选择数据库的存储机制，通常可以选择文件系统存储，从 Oracle 10g 开始 Oracle 引入了自动存储管理（Automatic Storage Management）的新特性，我们将会在后面章节中详细介绍这一新特性。

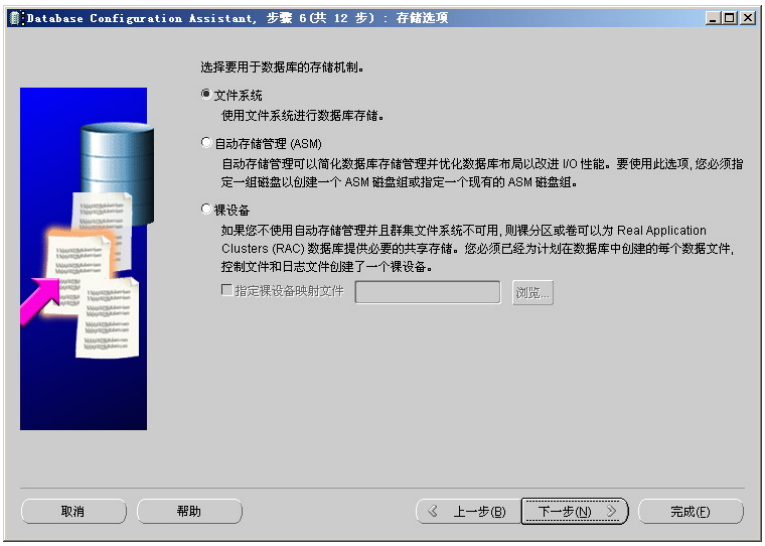


图 1-7　存储选项

单击“下一步”按钮，进入如图 1-8 所示的指定数据库文件所在位置界面，数据库文件存储位置可以选择“使用 Oracle 管理的文件”选项，这实际上就是利用了 Oracle 9i 中引入的一个新特新 OMF（Oracle Managed Files）。


图 1-8　指定数据库文件所在位置

单击“下一步”按钮，进入如图 1-9 所示的恢复配置界面，该界面用于指定快速恢复区（Flash Recovery Area），这是 Oracle 10g 的一个新特性，用于简化用户的备份管理，快速恢复区可以是磁盘上的一个存储目录，也可以使用 ASM 存储，这里可以按照具体的需要设置；同时还可以在这个页面上选择是否启动数据库的归档模式。


图 1-9　恢复配置

单击“下一步”按钮，进入如图 1-10 所示的选择数据库组件和定制脚本界面，Oracle 的数据库组件有很多，为了简化和快速安装，可以去除大部分选项（这要根据需要进行选择）。

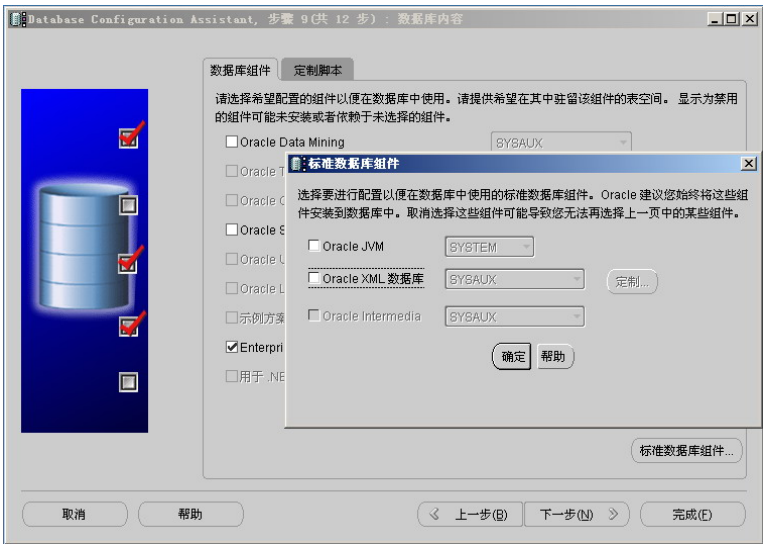


图 1-10　选择数据库组件和定制脚本

单击“下一步”按钮，进入如图 1-11 所示的初始化参数界面，其中内存选项可以暂时接受数据库的初始推荐，这些参数可以在建库后再进行修改。


图 1-11　设置初始化参数

注意数据块大小需要认真选择，如图 1-12 所示，一旦创建数据库之后，这个参数将不可修改（从 Oracle 9i 开始，Oracle 支持在同一数据库中容纳不同 block_size 的表空间，但是初始定义的 block_size 将用于 SYSTEM、UNDO 等表空间，不可修改）。


图 1-12　设置数据块大小

字符集部分也需要认真选择，在中文的 Windows 平台上，默认的字符集就是 ZHS16GBK，如图 1-13 所示，可以不需要修改，但是在 Linux/UNIX 下，如果系统语言环境缺省不是中文，则这里需要根据需要进行调整。


图 1-13　设置字符集

对于连接模式，可以选择默认的“专用服务器模式”选项，如图 1-14 所示。


图 1-14　选择数据库连接模式

单击“下一步”按钮，进入如图 1-15 所示的数据库存储界面，给出了存储及文件信息，可以按照需要进行调整，通常选择默认设置即可。

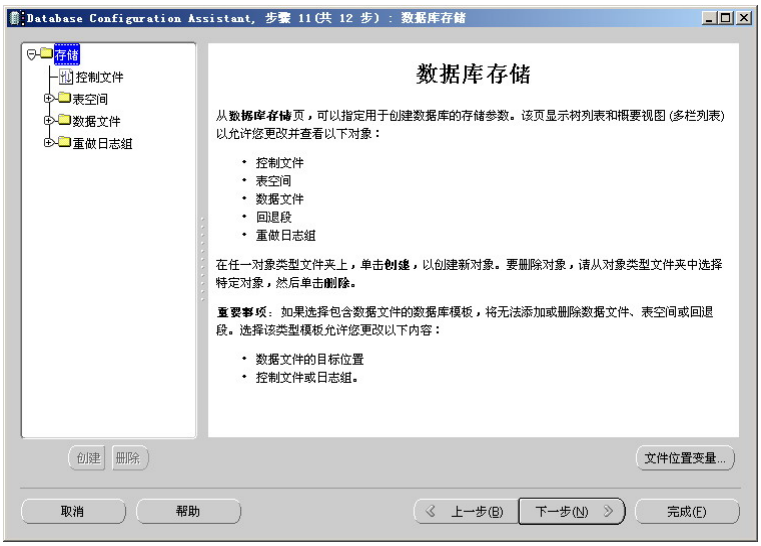


图 1-15　数据库存储

单击“下一步”按钮，进入如图 1-16 所示的页面，控制文件选项中定义了对于控制文件至关重要的几个参数，这些参数在此一旦确定，以后只有重建控制文件才能修改。


图 1-16　控制文件选项

单击“下一步”按钮，进入如图 1-17 所示的创建选项界面，可以选择将此前的设置存储为一个数据库模板，并生成创建数据库的脚本，如果接受“创建数据库”的选项，接下来就可以进行数据库的创建了（此处仅选择生成了模板和创建脚本）。

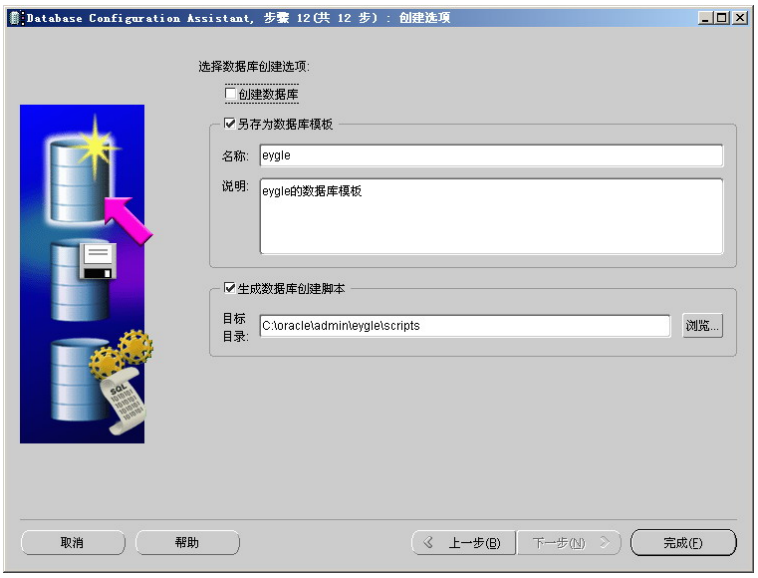


图 1-16　创建选项

单击“完成”按钮，则可以看到模板的摘要信息，如图 1-17 所示。


图 1-17　模板摘要信息

单击“确定”按钮，进入如图 1-18 所示的界面，数据库完成了脚本生成工作。如果选择了创建数据库，此时将开始数据库创建工作。

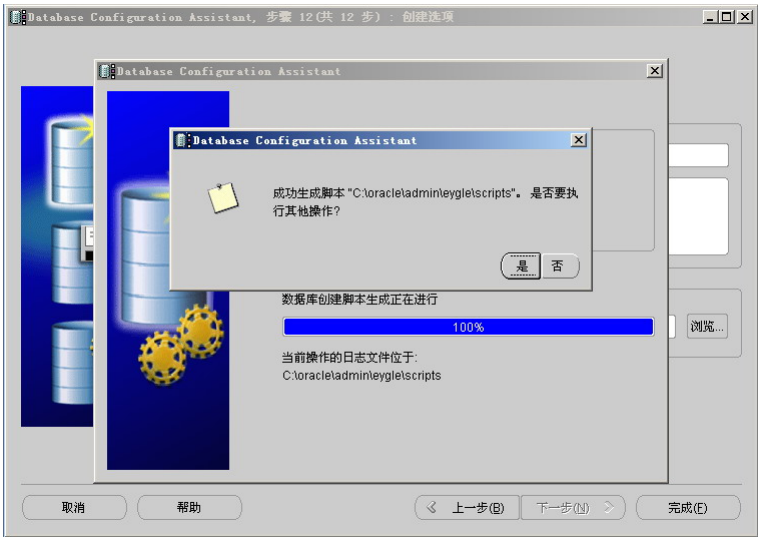


图 1-18　完成脚本生成

## 1.2　数据库创建的脚本

在 DBCA 的最后一个步骤，保存生成了创建数据库的脚本，通过手工执行这些脚本，可以在命令行完成数据库的创建工作，这可以使我们摆脱图形界面的困扰，特别是在一些不易于运行图形界面的环境。此外，很多时候通过 DBCA 创建数据库可能会遇到一些错误，这些错误在图形界面下可能不易判断，但是通过命令行则要容易定位得多。

### 1.2.1 数据库创建脚本

现在通过数据库的创建脚本来深入地了解一下数据库的创建过程。按照上面的路径找到生成的数据库创建脚本。

```
C:\Oracle\admin\eygle\scripts>dir
 驱动器 C 中的卷是 SYSTEM
 卷的序列号是 8C88-D1B4

 C:\Oracle\admin\eygle\scripts 的目录
2007-01-05  15:32    <DIR>          .
2007-01-05  15:32    <DIR>          ..
2007-01-05  15:32             1,139 CreateDB.sql
2007-01-05  15:32               600 CreateDBCatalog.sql
2007-01-05  15:32               326 CreateDBFiles.sql
2007-01-05  15:32               253 emRepository.sql
2007-01-05  15:32               614 eygle.bat
2007-01-05  15:32               698 eygle.sql
2007-01-05  15:32             2,408 init.ora
2007-01-05  15:33             1,108 postDBCreation.sql
```

在 Linux/UNIX 环境下，同样存在这样一系列的脚本（以下内容来自 Linux 上 Oracle 9204 安装脚本）：

```
[oracle@jumper scripts]$ pwd
/opt/oracle/admin/eygle/scripts
[oracle@jumper scripts]$ ll
total 24
-rw-r--r--    1 oracle   dba           713 Apr 24  2006 CreateDBCatalog.sql
-rw-r--r--    1 oracle   dba           338 Apr 24  2006 CreateDBFiles.sql
-rw-r--r--    1 oracle   dba           769 Apr 24  2006 CreateDB.sql
-rwxr-xr-x    1 oracle   dba           628 Aug 18  2006 eygle.sh
-rw-r--r--    1 oracle   dba          2764 Apr 24  2006 init.ora
-rw-r--r--    1 oracle   dba           442 Apr 24  2006 postDBCreation.sql
```

### 1.2.2 创建的起点

如果通过手工执行脚本来创建数据库，需要执行的脚本为 eygle.bat（在 Linux/UNIX 下是 eygle.sh 脚本），来看一下这个脚本的内容：

```
C:\Oracle\admin\eygle\scripts>type eygle.bat
mkdir C:\oracle\10.2.0\cfgtoollogs\dbca\eygle
mkdir C:\oracle\10.2.0\database
```

```
mkdir C:\oracle\admin\eygle\adump
mkdir C:\oracle\admin\eygle\bdump
mkdir C:\oracle\admin\eygle\cdump
mkdir C:\oracle\admin\eygle\dpdump
mkdir C:\oracle\admin\eygle\pfile
mkdir C:\oracle\admin\eygle\udump
mkdir C:\oracle\flash_recovery_area
mkdir C:\oracle\oradata
set ORACLE_SID=eygle
C:\oracle\10.2.0\bin\oradim.exe -new -sid EYGLE -startmode manual -spfile
C:\oracle\10.2.0\bin\oradim.exe -edit -sid EYGLE -startmode auto -srvcstart system
C:\oracle\10.2.0\bin\sqlplus /nolog @C:\oracle\admin\eygle\scripts\eygle.sql
```

这就是 Oracle 创建数据库的过程：

（1）　建立一系列的目录；

> 注意：这里建立的 bdump 目录是 Oracle 重要的警告日志的存放地点，其缺省名称为 alert_<sid>.log，我们应该定期检查该文件以发现数据库的故障或错误信息。第二个需要格外注意的是 cfgtoollogs\dbca\eygle 目录，在创建数据库时，主要的日志文件或输出信息会记录在该目录中，通过检查这些文件可以诊断创建过程中出现的一些错误。

（2）　设置 ORACLE_SID 环境变量；

（3）　通过 oradim 创建并配置实例；

（4）　通过 sqlplus 运行脚本开始创建数据库。

### 1.2.3 ORADIM 工具的使用

ORADIM 工具是 Oracle 在 Windows 上的一个命令行工具，用于手工进行 Oracle 服务的创建、修改、删除等工作。ORADIM 的使用很简单，通过帮助文件可以看到常用的命令示例，此处不再赘述。

ORADIM 在数据库恢复中也常被用到，很多朋友都问过这样的问题：在 Windows 上，如果系统崩溃了，可能数据库软件丢掉了，但是数据文件、控制文件、日志文件等都还在，该怎样来恢复 Oracle 数据库？

其实过程很简单，通常只要按原来的目录结构重新安装 Oracle 软件，然后通过 ORADIM 工具重建服务，就可以启动实例、加载数据库（当然相关的参数文件和口令文件等需要在 $ORACLE_HOME\database 目录存在）。

来看以下过程，通过 ORADIM 创建一个服务后，实例会随之启动：

```
C:\>oradim -new -sid eygle
实例已创建。
```

用 net 命令可以查看系统启动了哪些服务，看到 Oracle 的服务已经启动：

```
C:\>net start
```

```
已经启动以下 Windows 服务:
.............
   OracleServiceeygle
   Plug and Play
   Print Spooler
命令成功完成。
```

如果你的系统装了一些 UNIX 增强工具，那么可以通过 grep 过滤一下：

```
C:\>net start |grep Oracle
   OracleServiceeygle
```

使用 ORADIM 工具后，会在$ORACLE_HOME\database 目录下生成一个日志文件。

### 1.2.4 ORACLE_SID 的含义

注意到在 ORADIM 创建服务之前，首先设置了 ORACLE_SID：

```
set ORACLE_SID=eygle
```

在 Linux/UNIX 系统的创建中，同样要设置 ORACLE_SID，不过 Linux/UNIX 上不存在服务这项内容，实例是可以通过参数文件直接启动的。

看一下 Linux 上正常情况下启动到 nomount 状态的过程：

```
[oracle@jumper oracle]$ cd $ORACLE_HOME/dbs
[oracle@jumper dbs]$ ls
initconner.ora  init.ora  lkCONNER  orapwconner  spfileconner.ora  spfile.ora
[oracle@jumper dbs]$ export ORACLE_SID=conner
[oracle@jumper dbs]$ sqlplus "/ as sysdba"

SQL*Plus: Release 9.2.0.4.0 - Production on Wed Nov 3 14:57:22 2004
Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.
Connected to an idle instance.

SQL> startup nomount
ORACLE instance started.

Total System Global Area   80811208 bytes
Fixed Size                   451784 bytes
Variable Size              37748736 bytes
Database Buffers           41943040 bytes
Redo Buffers                 667648 bytes
```

注意这里，Oracle 根据参数文件的内容，创建了 instance，分配了相应的内存区域，启动了相应的后台进程。

回顾一下前面的内容，注意到 SID 和 ORACLE_SID 已经多次出现，那么 SID 是什么？在

数据库启动过程中又起到什么作用呢？

SID 是 **S**ystem **ID**entifier 的缩写，而 ORACLE_SID 就是 Oracle System Identifier 的缩写，在 Oracle 系统中，ORACLE_SID 以环境变量的形式出现，在特定版本的 Oracle 软件安装（也就是 ORACLE_HOME）下，当 Oracle 实例启动时，操作系统上 fork 的进程必须通过这个 SID 将实例与其他实例区分开来，这就是 SID 的作用。

Oracle 的实例（instance）是由一块共享内存区域（SGA）和一组后台进程（background processes）共同组成；而后台进程正是数据库和操作系统进行交互的通道，这些进程的名称就是通过 ORACLE_SID 决定的。

实例的启动仅需要一个参数文件，而这个参数文件的名称就是由 ORACLE_SID 决定的。对于 init 文件，缺省的文件名称是 init<ORACLE_SID>.ora，对于 spfile 文件，缺省的文件名为 spfile<ORACLE_SID>.ora，Oracle 依据 ORACLE_SID 来决定和寻找参数文件启动实例，参数文件的缺省位置为$ORACLE_HOME/dbs（Windows 上为$ORACLE_HOME\database 目录）。

spfile 从 Oracle 9i 开始引入并成为了缺省使用的参数文件，Oracle 启动实例时按照以下顺序从缺省目录查找参数文件：spfile<ORACLE_SID>.ora→spfile.ora →init<ORACLE_SID>.ora。如果这 3 个文件都不存在，则 Oracle 实例将无法启动。

通过这些信息可以知道，在同一个 ORACLE_HOME 下，Oracle 能够根据 ORACLE_SID 将实例区分开来；但是如果在不同的 ORACLE_HOME 下，Oracle 将无法屏蔽相同名称的 ORACLE_SID，也就是说即使在同一台主机上，Oracle 也是能够创建相同 ORACLE_SID 的实例的。

以下一个测试，首先启动一个 Oracle8i 下 ORACLE_SID 为 eygle 的实例：

```
$ export ORACLE_SID=eygle
$ sqlplus "/ as sysdba"
SQL*Plus: Release 8.1.7.0.0 - Production on Fri Feb 16 10:23:58 2007
(c) Copyright 2000 Oracle Corporation.  All rights reserved.
Connected to an idle instance.

SQL> startup nomount;
ORACLE instance started.
<...ignore SGA info here...>

SQL> ! ps -ef|grep eygle
 oracle8 11106     1  0 10:24:02 ?        0:00 ora_d000_eygle
 oracle8 11090     1  0 10:24:02 ?        0:00 ora_ckpt_eygle
 oracle8 11086     1  0 10:24:02 ?        0:00 ora_dbw0_eygle
 oracle8 11084     1  0 10:24:02 ?        0:00 ora_pmon_eygle
 oracle8 11092     1  0 10:24:02 ?        0:00 ora_smon_eygle
 oracle8 11088     1  0 10:24:02 ?        0:00 ora_lgwr_eygle
……
```

接下来又可以启动另外 ORACLE_HOME 下 ORACLE_SID 为 eygle 的实例：

```
$ export ORACLE_SID=eygle
$ sqlplus "/ as sysdba"
SQL*Plus: Release 9.2.0.4.0 - Production on Fri Feb 16 10:24:43 2007
Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.
Connected to an idle instance.

SQL> startup nomount;
ORACLE instance started.
<...ignore SGA info here...>

SQL> ! ps -ef|grep pmon_eygle
 oracle9 11214 11172  0 10:24:58 pts/1    0:00 grep pmon_eygle
 oracle9 11180     1  0 10:24:48 ?        0:00 ora_pmon_eygle
 oracle8 11084     1  0 10:24:02 ?        0:00 ora_pmon_eygle
```

现在这同一台主机上就启动了两个相同名称的实例，在操作系统上，Oracle 能够通过 ID 标识将共享内存或信号量区分开来：

```
$ ipcs -i
IPC status from <running system> as of Fri Feb 16 10:30:02 CST 2007
T         ID      KEY        MODE        OWNER    GROUP
Message Queues:
q          0   0x2e781d5  --rw-r--r--     root     root
T         ID      KEY        MODE        OWNER    GROUP ISMATTCH
Shared Memory:
m       4096   0xabdc9b64 --rw-r-----  oracle8      dba       12
m       1025   0x79552064 --rw-r-----  oracle9      dba       11
Semaphores:
s    1245184   0x79978bac --ra-r-----  oracle8      dba
s     458753   0xa0e9f594 --ra-r-----  oracle9      dba
```

通过 Oracle 提供的一个小工具 sysresv，我们可以找到对应于不同的 ORACLE_SID，操作系统上创建的共享内存段 ID（Shared Memory）和信号量 ID（Semaphores）等信息。

```
$ sysresv -l eygle julia

IPC Resources for ORACLE_SID "eygle" :
Shared Memory:
ID              KEY
2560            0x79552064
Semaphores:
ID              KEY
720896          0xa0e9f594
```

```
Oracle Instance alive for sid "eygle"

IPC Resources for ORACLE_SID "julia" :
Shared Memory:
ID              KEY
514             0xab281214
Semaphores:
ID              KEY
196610          0xa7645a54
Oracle Instance alive for sid "julia"
```

在 Linux/UNIX 上，ORACLE_SID 还和一个名为 oratab 的文件有关，在 Solaris 环境中，这个文件一般位于/var/opt/oracle 目录下，在 Linux 及其他 UNIX 平台，这个文件一般位于/etc 目录下。该文件的主要内容如下：

```
# This file is used by ORACLE utilities.  It is created by root.sh
# and updated by the Database Configuration Assistant when creating a database.

# A colon, ':', is used as the field terminator.  A new line terminates
# the entry.  Lines beginning with a pound sign, '#', are comments.
#
# Entries are of the form:
#   $ORACLE_SID:$ORACLE_HOME:<N|Y>:
#
# The first and second fields are the system identifier and home
# directory of the database respectively.  The third filed indicates
# to the dbstart utility that the database should , "Y", or should not,
# "N", be brought up at system boot time.
#
# Multiple entries with the same $ORACLE_SID are not allowed.
#
*:/opt/oracle/product/9.2.0:N
```

当执行 dbstart 脚本时，Oracle 会根据这里记录的 ORACLE_SID 的<N|Y>的设置来决定是否启动相关实例。

与 Linux/UNIX 上的情况类似，Windows 上的 Oracle 环境也依赖于服务而存在，如图 1-19 所示。

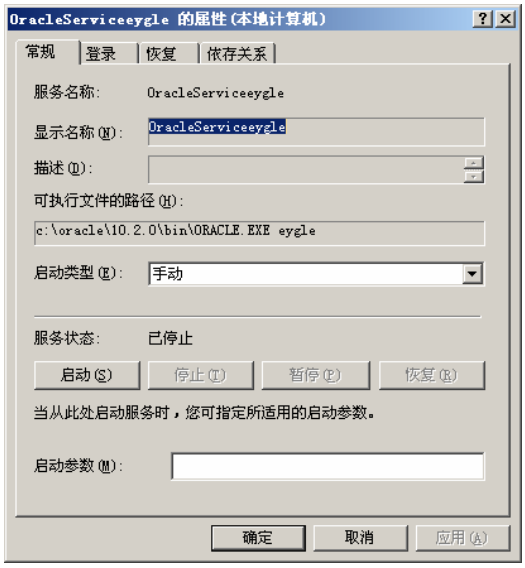


图 1-19　Windows Oracle 服务示意图

注意到 Oracle 环境的初始化是通过 ORACLE.EXE eygle 来完成的，至于实例和数据库是否随服务启动要依赖于注册表中的设置。

通过手动在命令行执行类似命令，可以初始化任意的 Oracle 环境：

```
C:\>oracle julia
Press CTRL-C to exit server:
```

此后就可以连接到这个环境启动实例：

```
C:\>set ORACLE_SID=julia

C:\>sqlplus "/ as sysdba"
SQL*Plus: Release 10.2.0.1.0 - Production on 星期六 2 月 17 10:11:13 2007
Copyright (c) 1982, 2005, Oracle.  All rights reserved.
已连接到空闲例程。
SQL> startup nomount;
ORA-01078: failure in processing system parameters
LRM-00109: ???????????????? 'C:\ORACLE\10.2.0\DATABASE\INITJULIA.ORA'
```

当然还需要创建参数文件和口令文件等：

```
C:\>cp c:\oracle\10.2.0\database\SPFILEEYGLE.ORA c:\Oracle\10.2.0\database\spfilejulia.ora

C:\>orapwd file=c:\oracle\10.2.0\database\PWDjulia.ora password=oracle entries=5
```

此后，实例可以顺利启动，并可以挂接和打开数据库：

```
C:\>sqlplus "/ as sysdba"
SQL*Plus: Release 10.2.0.1.0 - Production on 星期六 2 月 17 10:13:10 2007
Copyright (c) 1982, 2005, Oracle.  All rights reserved.
已连接到空闲例程。
```

```
SQL> startup nomount;
ORACLE 例程已经启动。
<...ignore SGA info here...>

SQL> set linesize 120
SQL> show parameter instance_name
NAME                                 TYPE                 VALUE
------------------------------------ -------------------- ---------------
instance_name                        string               julia
SQL> show parameter db_name
NAME                                 TYPE                 VALUE
------------------------------------ -------------------- ------------------------------
db_name                              string               eygle
SQL> alter database mount;
数据库已更改。
SQL> alter database open;
数据库已更改。
```

如果在环境窗口中按下 CTRL+C 组合键退出，则数据库将异常中断。

总结一下，实际上不管在 Windows 还是 Linux/UNIX 环境下，ORACLE_SID 的作用就是设置一个 Oracle 环境窗口，通过这个环境变量来标示和命名系统进程，此后 Oracle 的活动可以由此展开。

## 1.2.5 INSTANCE_NAME 的含义及作用

作为 Oracle 数据库的重要组成部分 INSTANCE 也存在一个参数标识：INSTANCE_NAME。

INSTANCE_NAME 是 Oracle 数据库的一个参数，在参数文件中定义，用于标识数据库实例的名称，其缺省值通常就是 ORACLE_SID，但是不同的实例可以有相同的实例名。通过简单的参数文件复制，我们就可以在同一台服务器上创建多个具有相同 INSTANCE_NAME 的实例。

首先确认当前的参数文件：

```
bash-2.03$ cd $ORACLE_HOME/dbs
bash-2.03$ ls initeygle.ora
initeygle.ora
```

复制参数文件，更改名称：

```
bash-2.03$ cp initeygle.ora initjulia.ora
```

接下来通过导入新的 ORACLE_SID 就可以启动新的实例：

```
bash-2.03$ export ORACLE_SID=julia
bash-2.03$ sqlplus "/ as sysdba"
```

```
SQL*Plus: Release 9.2.0.4.0 - Production on Fri Feb 16 10:34:00 2007
Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.
Connected to an idle instance.

SQL> startup nomount;
ORACLE instance started.

Total System Global Area  303532408 bytes
Fixed Size                   731512 bytes
Variable Size             184549376 bytes
Database Buffers          117440512 bytes
Redo Buffers                 811008 bytes
```

现在 ORACLE_SID 为 julia 的实例已经启动，操作系统上的进程以 julia 名称标记：

```
bash-2.03$ ps -ef|grep pmon
oracle   12396     1  0 16:30 ?        00:00:00 ora_pmon_julia
oracle   16201     1  0 18:13 ?        00:00:00 ora_pmon_eygle
oracle   16256 16219  0 18:14 pts/1    00:00:00 grep pmon
```

但是新实例的 instance_name 仍然是 eygle：

```
SQL> show parameter instance_name
NAME                                 TYPE                 VALUE
------------------------------------ -------------------- ----------------------
instance_name                        string               eygle
```

总结一下，ORACLE_SID 在这里用于标示进程，而 instance_name 则用来标示实例，两者可以具有不同的名称。

此外 Oracle 的监听器（listener）配置文件中的 SID_NAME 就是来自 instance_name 参数，监听器通过 instance_name 才能确定需要将连接请求注册到哪一个实例上。通常 listener.ora 文件中 SID_NAME 相关设置类似如下示例：

```
SID_LIST_LISTENER =
    (SID_DESC =
      (GLOBAL_DBNAME = eygle)
      (ORACLE_HOME = /opt/oracle/product/9.2.0)
      (SID_NAME = eygle)
    )
```

### 1.2.6 Oracle 的口令文件

继续前面的脚本，在创建和启动了实例之后，Oracle 开始调用 eygle.sql 脚本，我们将这个脚本分开来介绍。

这个脚本的最初部分是要求定义用户口令，然后使用定义的 sys 用户口令创建口令文件：

```
C:\Oracle\admin\eygle\scripts>type eygle.sql
set verify off
PROMPT specify a password for sys as parameter 1;
DEFINE sysPassword = &1
PROMPT specify a password for system as parameter 2;
DEFINE systemPassword = &2
PROMPT specify a password for sysman as parameter 3;
DEFINE sysmanPassword = &3
PROMPT specify a password for dbsnmp as parameter 4;
DEFINE dbsnmpPassword = &4
host C:\oracle\10.2.0\bin\orapwd.exe file=C:\oracle\10.2.0\database\PWDeygle.ora
 password=&&sysPassword force=y
```

这里又引入了另外一个工具 orapwd，这个工具在 Linux/UNIX 上同样存在，当口令文件丢失或损坏之后，可以通过这个工具重建口令文件，这个工具的语法为：

```
C:\>orapwd
Usage: orapwd file=<fname> password=<password> entries=<users> force=<y/n>
  where
    file - name of password file (mand),
    password - password for SYS (mand),
    entries - maximum number of distinct DBA and     force - whether to overwrite existing file
(opt),
OPERs (opt),
  There are no spaces around the equal-to (=) character.
```

注意：force 参数是 Oracle 10g 中新增加的。

Oracle 在启动过程中，会在$ORACLE_HOME/dbs（Windows 下相应的目录则是$ORACLE_HOME\database）目录下查找口令文件，查找的顺序是首先检查 orapw<ORACLE_SID>文件，如果不存在则查找 orapwd 文件，如果 orapwd 文件也不存在，就会报出如下错误：

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

Total System Global Area 131142648 bytes
Fixed Size 451576 bytes
Variable Size 104857600 bytes
Database Buffers 25165824 bytes
Redo Buffers 667648 bytes
ORA-01990: error opening password file '/opt/oracle/product/9.2.0/dbs/orapw'
ORA-27037: unable to obtain file status
Linux Error: 2: No such file or directory
```

```
Additional information: 3
```

口令文件丢失或损坏后，通常可以通过如下命令重建口令文件：

```
[oracle@jumper dbs]$ orapwd file=orapwhsjf password=oracle entries=5
```

在数据库没有启动之前，数据库内建用户是无法通过数据库来验证身份的，此时口令文件的作用就体现了出来。口令文件中存放了具有 sysdba / sysoper 身份用户的用户名及口令，Oracle 允许用户通过口令文件验证，在数据库未启动之前登录，从而启动实例进而加载并打开数据库；而如果没有口令文件，在数据库未启动之前就只能通过操作系统认证方式来启动实例。

Oracle 通过一个初始化参数 remote_login_passwordfile 来限制口令文件的使用，通过这个参数可以设置用户登录时是否检查口令文件，以及有多少个数据库可以使用口令文件。这个参数有 3 个选项：EXCLUSIVE、SHARED 和 NONE。

当 remote_login_passwordfile 设置为 NONE 时，远程用户将不能通过 sysdba/sysoper 身份登录数据库：

```
SQL> show parameter pass
NAME                         TYPE        VALUE
---------------------------- ----------- ------------------------------

remote_login_passwordfile string      NONE
```

此时通过远程连接会收到如下错误：

```
E:\Oracle\ora92\bin>sqlplus /nolog
SQL*Plus: Release 9.2.0.4.0 -
Production on 星期四 4月 15 09:39:22 2004
Copyright (c) 1982, 2002, Oracle Corporation. All rights reserved.
SQL> connect sys/oracle@hsjf as sysdba
ERROR:ORA-01017: invalid username/password; logon denied
```

此处实际上是无法通过口令文件验证。

缺省的 remote_login_passwordfile 参数设置为 exclusive，支持远程 sysdba 的登录操作：

```
SQL> alter system set remote_login_passwordfile=exclusive scope=spfile;


System altered.
```

这个参数是静态参数，修改后重起数据库才能生效。当 remote_login_passwordfile 参数设置为 exclusive 时可以通过远程以 sysdba 身份登录数据库：

```
E:\Oracle\ora92\bin>sqlplus /nolog
SQL*Plus: Release 9.2.0.4.0 -
Production on 星期四 4月 15 09:47:11 2004
Copyright (c) 1982, 2002, Oracle Corporation. All rights reserved.
SQL> connect sys/oracle@hsjf as sysdba
已连接。
SQL> show user
USER 为"SYS"
```

当 remote_login_passwordfile 参数设置为 shared 时，则多个数据库可以共享一个口令文件，

但是此时口令文件中只能存储 SYS 用户的口令，此时其他用户不能被授予 sysdba 身份：

```
SQL> select * from v$pwfile_users;
USERNAME  SYSDB  SYSOP
--------- ----- -----
SYS       TRUE   TRUE
SQL> grant sysdba to eygle;
grant sysdba to eygle
*
ERROR at line 1:
ORA-01994: GRANT failed: cannot add users to public password file
SQL> show parameter password
NAME                           TYPE    VALUE
------------------------------  ------ ------------------------------
remote_login_passwordfile      string SHARED
```

此时的口令文件中是不能添加用户的。

很多朋友的疑问在于：口令文件的缺省名称是 orapw<ORACLE_SID>，怎么能够共享？

前面已经提到，Oracle 数据库在启动时，首先查找的是 orapw<ORACLE_SID>的口令文件，如果该文件不存在，则开始查找 orapw 的口令文件；如果同一主机上的多个数据库同时使用 orapw 文件，则口令文件就可以共享（当然通过其他方式，如符号链接等也可以实现共享）。

来看一下测试，首先移动缺省的口令文件：

```
[oracle@jumper dbs]$ mv orapweygle orapweygle.b
```

此时启动数据库会出现如下错误：

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

<...ignore SGA info here...>
ORA-01990: error opening password file '/opt/oracle/product/9.2.0/dbs/orapw'
ORA-27037: unable to obtain file status
Linux Error: 2: No such file or directory
Additional information: 3
```

拷贝一个 orapw 口令文件，这时候再启动数据库就不会出现这个错误：

```
SQL> ! cp orapweygle.b orapw
SQL> startup force;
ORACLE instance started.
<...ignore SGA info here...>

Database mounted.
Database opened.
SQL> show parameter password
```

```
NAME                                 TYPE         VALUE
------------------------------------ ------------ ------------------------------
remote_login_passwordfile            string       SHARED
```

那么你可能会有这样的疑问：多个 Exclusive 的数据库是否可以共享一个口令文件（orapw）呢？继续这个测试：

```
[oracle@jumper dbs]$ strings orapw
]\[Z
ORACLE Remote Password file
INTERNAL
AB27B53EDC5FEF41
8A8F025737A9097A
```

注意这里仅记录着 INTERNAL/SYS 的口令。

当 REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE 时：

```
SQL> alter system set remote_login_passwordfile=exclusive scope=spfile;
System altered.
SQL> startup force;
ORACLE instance started.
<...ignore SGA info here...>

Database mounted.
Database opened.
SQL> ! strings orapw
]\[Z
ORACLE Remote Password file
EYGLE
INTERNAL
AB27B53EDC5FEF41
8A8F025737A9097A
```

注意：这里以 EXCLUSIVE 方式启动以后，实例名称信息被写入口令文件。

此时如果有其他实例以 Exclusive 模式启动，仍然可以使用这个口令文件，口令文件中的实例名称同时被改写，也就是说，数据库只在启动过程中才读取口令文件，数据库运行过程中并不锁定该文件，类似于 pfile/spfile 文件。

进一步地，如果对其他用户授予 SYSDBA 的身份：

```
SQL> select * from v$pwfile_users;
USERNAME          SYSDB      SYSOP
----------------- -------- --------
SYS                TRUE     TRUE
SQL> grant sysdba to eygle;
```

```
Grant succeeded.
SQL> select * from v$pwfile_users;
USERNAME          SYSDB         SYSOP
------------------------------ ----- -----
SYS           TRUE          TRUE
EYGLE TRUE FALSE

SQL> ! strings orapw
]\[Z
ORACLE Remote Password file
EYGLE
INTERNAL
AB27B53EDC5FEF41
8A8F025737A9097A
>EYGLE
B726E09FE21F8E83
```

注意此时增加的 SYSDBA 用户，其相关信息可以被写入到口令文件，一旦口令文件中增加了其他 SYSDBA 用户，此文件就不再能够被其他 Exclusive 的实例共享。

### 1.2.7 脚本的执行

继续来看 eygle.sql 的内容，接下来的脚本才是创建数据库中最关键的：

```
@C:\oracle\admin\eygle\scripts\CreateDB.sql
@C:\oracle\admin\eygle\scripts\CreateDBFiles.sql
@C:\oracle\admin\eygle\scripts\CreateDBCatalog.sql
@C:\oracle\admin\eygle\scripts\emRepository.sql
@C:\oracle\admin\eygle\scripts\postDBCreation.sql
```

第一个脚本是 CreateDB.sql，其主要内容如下：

```
connect "SYS"/"&&sysPassword" as SYSDBA
set echo on
spool C:\oracle\admin\eygle\scripts\CreateDB.log
startup nomount pfile="C:\oracle\admin\eygle\scripts\init.ora";
CREATE DATABASE "eygle"
MAXINSTANCES 8
MAXLOGHISTORY 1
MAXLOGFILES 16
MAXLOGMEMBERS 3
MAXDATAFILES 100
DATAFILE SIZE 300M AUTOEXTEND ON NEXT  10240K MAXSIZE UNLIMITED
```

```
EXTENT MANAGEMENT LOCAL
SYSAUX DATAFILE SIZE 120M AUTOEXTEND ON NEXT  10240K MAXSIZE UNLIMITED
SMALLFILE DEFAULT TEMPORARY TABLESPACE TEMP TEMPFILE SIZE 20M AUTOEXTEND ON NEXT  640K MAXSIZE
UNLIMITED
SMALLFILE UNDO TABLESPACE "UNDOTBS1" DATAFILE SIZE 200M AUTOEXTEND ON NEXT  5120K MAXSIZE
UNLIMITED
CHARACTER SET ZHS16GBK
NATIONAL CHARACTER SET AL16UTF16
LOGFILE GROUP 1  SIZE 51200K,
GROUP 2  SIZE 51200K,
GROUP 3  SIZE 51200K
USER SYS IDENTIFIED BY "&&sysPassword" USER SYSTEM IDENTIFIED BY "&&systemPassword";
set linesize 2048;
column ctl_files NEW_VALUE ctl_files;
select concat('control_files=''', concat(replace(value, ', ', ''',''), ''''')) ctl_files from
v$parameter where name ='c
ontrol_files';
host "echo &ctl_files >>C:\oracle\admin\eygle\scripts\init.ora";
spool off
```

可以看到，这个文件的主要操作步骤如下：

（1） 通过 SYS 连接；

（2） 通过配置的参数文件 init.ora 启动实例；

（3） 开始数据库创建；

（4） 将数据库生成的控制文件名称追加到参数文件。

> 注意：由于选择了 OMF 管理文件，控制文件的名称在创建数据库之前是未知的，所以创建数据库之后才能得到名称加入参数文件中。

## 1.2.8 db_name 参数和 instance_name 参数

在启动实例后执行的创建数据库中，第一个语句就是“CREATE DATABASE **"eygle"**”，这是数据库最重要的开始，其中“**"eygle"**”也就是图 1-4 中定义的数据库名称。

对于 Oracle 数据库来说，db_name 代表数据库的名称而 instance_name 代表实例的名称，instance_name 通过参数文件即可修改，而 db_name 则不然，来看一下 Oracle 对于数据库名称的定义：DB_NAME 必须是一个不超过 8 个字符的文本串。在数据库创建过程中，db_name 被记录在数据文件，日志文件和控制文件中。如果数据库实例启动过程中参数文件中的 db_name 和控制文件中的数据库名称不一致，则数据库不能启动。

通过以上定义可以看到，db_name 是最具有稳定意义的参数，在数据文件、日志文件和控制文件中都会记录数据库的名称，这个名称完全可以不同于 instance_name。

以下的测试数据库拥有相同的 db_name 和 instance_name：

```
[oracle@jumper oracle]$ cd $ORACLE_HOME/dbs
[oracle@jumper dbs]$ grep name initeygle.ora
*.db_name='eygle'
*.instance_name='eygle'
```

创建一个新的 pfile 为 julia 这个新的实例使用：

```
[oracle@jumper oracle]$ cd $ORACLE_HOME/dbs
[oracle@jumper dbs]$ cp initeygle.ora initjulia.ora
```

修改这个文件更改 instance_name 参数：

```
[oracle@jumper dbs]$ grep name initjulia.ora
*.db_name='eygle'
*.instance_name='julia'
```

然后启动实例名称为 julia 的 instance：

```
[oracle@jumper dbs]$ export ORACLE_SID=julia
[oracle@jumper dbs]$ sqlplus "/ as sysdba"
SQL*Plus: Release 9.2.0.4.0 - Production on Tue Jul 25 14:04:15 2006
Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.
Connected to an idle instance.

SQL> startup mount;
ORACLE instance started.

Total System Global Area  139531744 bytes
Fixed Size                   452064 bytes
Variable Size             121634816 bytes
Database Buffers           16777216 bytes
Redo Buffers                 667648 bytes
ORA-01102: cannot mount database in EXCLUSIVE mode
```

> 注意：此时试图加载数据库时出现错误，因为当前数据库被另外一个实例（instance）加载。在非并行模式（OPS/RAC）下，一个数据库同时只能被一个实例加载。

此时已经启动了两个数据库实例，从后台进程可以看出：

```
[oracle@jumper dbs]$ ps -ef|grep ora_pmon
oracle    27321     1  0 Jul14 ?        00:00:00 ora_pmon_eygle
oracle    15445     1  0 14:04 ?        00:00:00 ora_pmon_julia
oracle    15459 15391  0 14:04 pts/2    00:00:00 ps -ef
oracle    15460 15391  0 14:04 pts/2    00:00:00 grep ora_pmon
```

关闭 eygle 这个数据库实例后，就可以通过实例 julia 加载并打开 db_name=eygle 的数据库了：

```
[oracle@jumper dbs]$ export ORACLE_SID=julia
[oracle@jumper dbs]$ sqlplus "/ as sysdba"
SQL*Plus: Release 9.2.0.4.0 - Production on Tue Jul 25 14:05:06 2006
Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.
……
SQL> alter database mount;
alter database mount
*
ERROR at line 1:
ORA-01990: error opening password file '/opt/oracle/product/9.2.0/dbs/orapw'
ORA-27037: unable to obtain file status
Linux Error: 2: No such file or directory
Additional information: 3

SQL> alter database open;
Database altered.
SQL> select name from v$datafile;
NAME
--------------------------------------------------------------------------------
/opt/oracle/oradata/eygle/system01.dbf
/opt/oracle/oradata/eygle/undotbs01.dbf
/opt/oracle/oradata/eygle/users01.dbf
/opt/oracle/oradata/eygle/eygle01.dbf
```

新的实例具有独立的 instance_name 和 db_name 参数设置：

```
SQL> ! ps -ef|grep ora_pmon
oracle   15445     1  0 14:04 ?        00:00:00 ora_pmon_julia
oracle   15515 15513  0 14:05 pts/2    00:00:00 /bin/bash -c  ps -ef|grep ora
oracle   15516 15515  0 14:05 pts/2    00:00:00 ps -ef
SQL> show parameter instance_name
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
instance_name                        string      julia
SQL> show parameter db_name
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_name                              string      eygle
```

再看看如果参数文件中的 db_name 和控制文件中的 db_name 不一致会出现什么错误。修改参数文件中的 db_name 参数：

```
[oracle@jumper dbs]$ grep name initjulia.ora
```

```
*.db_name='julia'
*.instance_name='julia'
```

在启动过程中，可以看到，在 mount 阶段，数据库会对参数文件和控制文件进行比较，如果两者记录的 db_name 不一致，则数据库就无法启动：

```
SQL> startup nomount;
ORACLE instance started.
<...ignore SGA info here...>

SQL> alter database mount;
alter database mount
*
ERROR at line 1:
ORA-01103: database name 'EYGLE' in controlfile is not 'JULIA'
```

总结一下，一个实例（instance_name）可以 mount 并打开任何数据库（db_name），但是同一时间一个实例只能打开一个数据库；一个数据库（db_name）同一时间可以为任一实例（instance_name）所打开，但是在非 OPS/RAC 情况下，同时只能被同一个实例所打开。

## 1.2.9 sql.bsq 文件与数据库创建

在 CREATE DATABASE 的过程中，Oracle 会调用$ORACLE_HOME/rdbms/admin/sql.bsq 脚本，用于创建数据字典，这是非常重要的一个脚本，其中存储了数据字典的创建语句及注释说明，当我们对某些数据字典存在兴趣时，可以通过检查这个文件得到更为详细的信息，例如对于控制数据库启动的 bootstrap$表，其创建语句就可以从这个文件中找到：

```
create table bootstrap$
( line#         number not null,                    /* statement order id */
  obj#          number not null,                         /* object number */
  sql_text      varchar2("M_VCSZ") not null)                 /* statement */
  storage (initial 50K)            /* to avoid space management during IOR I */
//                                          /* "//" required for bootstrap */
```

sql.bsq 文件的位置受到一个隐含的初始化参数（_init_sql_file）的控制：

```
SQL> @GetParDescrb.sql
Enter value for par: init_sql
old   6:    AND x.ksppinm LIKE '%&par%'
new   6:    AND x.ksppinm LIKE '%init_sql%'

NAME            VALUE                DESCRIB
--------------- -------------------- --------------------------------------------------
_init_sql_file  ?/rdbms/admin/sql.bsq File containing SQL statements to execute upon database
                                     creation
```

如果在创建过程中，Oracle 无法找到 sql.bsq 文件，则数据库创建将会出错。可以测试一下移除 sql.bsq 文件，看一下数据库创建过程：

```
[oracle@jumper scripts]$ sqlplus "/ as sysdba"
SQL*Plus: Release 9.2.0.4.0 - Production on Fri Aug 18 15:45:26 2006
Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.

Connected to an idle instance.
SQL> startup nomount;
ORACLE instance started.
<...ignore SGA info here...>

SQL> @CreateDB.sql
CREATE DATABASE eygle
*
ERROR at line 1:
ORA-01092: ORACLE instance terminated. Disconnection forced

Disconnected from Oracle9i Enterprise Edition Release 9.2.0.4.0 - Production
With the Partitioning option
JServer Release 9.2.0.4.0 - Production
```

此时警告日志（alert_<oracle_sid>.log）中会记录如下信息：

```
Fri Aug 18 15:45:49 2006
Errors in file /opt/oracle/admin/eygle/udump/eygle_ora_3632.trc:
ORA-01501: CREATE DATABASE failed
ORA-01526: error in opening file '?/rdbms/admin/sql.bsq'
ORA-07391: sftopn: fopen error, unable to open text file.
Error 1526 happened during db open, shutting down database
USER: terminating instance due to error 1526
Fri Aug 18 15:45:49 2006
Errors in file /opt/oracle/admin/eygle/bdump/eygle_ckpt_3623.trc:
ORA-01526: error in opening file ''
Instance terminated by USER, pid = 3632
ORA-1092 signalled during: CREATE DATABASE eygle
MAXINSTANCES 1
MAXLOGHISTORY...
```

这就是 sql.bsq 文件在数据库创建过程中的作用。知道了这个内容之后，我们可以通过手工修改 sql.bsq 文件来更改数据库字典对象参数，从而实现特殊要求数据库的创建或测试自定义库，也可以通过修改_init_sql_file 参数来重定位 sql.bsq 文件的位置（但是通常这些是不建议变更的）。

### 1.2.10 数据文件及字典的创建

再来看 CreateDBFiles.sql 文件：

```
C:\Oracle\admin\eygle\scripts>type CreateDBFiles.sql
connect "SYS"/"&&sysPassword" as SYSDBA
set echo on
spool C:\oracle\admin\eygle\scripts\CreateDBFiles.log
CREATE SMALLFILE TABLESPACE "USERS" LOGGING DATAFILE SIZE 5M AUTOEXTEND ON NEXT  1280K MAXSIZE
UNLIMITED EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT  AUTO;
ALTER DATABASE DEFAULT TABLESPACE "USERS";
spool off
```

这个文件向数据库中追加了 USERS 表空间，并将该表空间设置为系统缺省的数据表空间，注意最后一句：

```
ALTER DATABASE DEFAULT TABLESPACE "USERS";
```

这是 Oracle 10g 增加的新特性，在 Oracle 10g 之前，如果创建用户不指定缺省的数据表空间，那么用户的缺省表空间会被指向系统表空间，增加了数据库缺省数据表空间后，如果不指定，那么创建用户的缺省数据表空间会被指向这里：

```
SQL> create user julia identified by eygle;
用户已创建。

SQL> select username,default_tablespace from dba_users
  2  where username='JULIA';
USERNAME                       DEFAULT_TABLESPACE
------------------------------ ------------------------------
JULIA                          USERS
```

作为一个数据库属性，这个信息也可以从字典表 props$中查询得到：

```
SQL> select name,value$ from props$
  2  where name='DEFAULT_PERMANENT_TABLESPACE';
NAME                           VALUE$
------------------------------ --------------------
DEFAULT_PERMANENT_TABLESPACE   USERS
```

继续前面的讨论，接下来 Oracle 通过 CreateDBCatalog.sql 创建数据字典：

```
C:\Oracle\admin\eygle\scripts>cat CreateDBCatalog.sql
connect "SYS"/"&&sysPassword" as SYSDBA
set echo on
spool C:\oracle\admin\eygle\scripts\CreateDBCatalog.log
@C:\oracle\10.2.0\rdbms\admin\catalog.sql;
@C:\oracle\10.2.0\rdbms\admin\catblock.sql;
@C:\oracle\10.2.0\rdbms\admin\catproc.sql;
```

```
@C:\oracle\10.2.0\rdbms\admin\catoctk.sql;
@C:\oracle\10.2.0\rdbms\admin\owminst.plb;
connect "SYSTEM"/"&&systemPassword"
@C:\oracle\10.2.0\sqlplus\admin\pupbld.sql;
connect "SYSTEM"/"&&systemPassword"
set echo on
spool C:\oracle\admin\eygle\scripts\sqlPlusHelp.log
@C:\oracle\10.2.0\sqlplus\admin\help\hlpbld.sql helpus.sql;
spool off
spool off
```

这个文件依次调用 Oracle 的字典创建文件等。emRepository.sql 文件是用于创建 EM 档案库的：

```
C:\Oracle\admin\eygle\scripts>type emRepository.sql
connect "SYS"/"&&sysPassword" as SYSDBA
set echo off
spool C:\oracle\admin\eygle\scripts\emRepository.log
@C:\oracle\10.2.0\sysman\admin\emdrep\sql\emreposcre C:\oracle\10.2.0 SYSMAN &&sysmanPassword
TEMP ON;
WHENEVER SQLERROR CONTINUE;
spool off
```

最后一个执行的文件是 postDBCreation.sql：

```
C:\Oracle\admin\eygle\scripts>cat postDBCreation.sql
connect "SYS"/"&&sysPassword" as SYSDBA
set echo on
spool C:\oracle\admin\eygle\scripts\postDBCreation.log
connect "SYS"/"&&sysPassword" as SYSDBA
set echo on
create spfile='C:\oracle\10.2.0\database\spfileeygle.ora'
  FROM pfile='C:\oracle\admin\eygle\scripts\init.ora';
shutdown immediate;
connect "SYS"/"&&sysPassword" as SYSDBA
startup ;
alter user SYSMAN identified by "&&sysmanPassword" account unlock;
alter user DBSNMP identified by "&&dbsnmpPassword" account unlock;
select 'utl_recomp_begin: ' || to_char(sysdate, 'HH:MI:SS') from dual;
execute utl_recomp.recomp_serial();
select 'utl_recomp_end: ' || to_char(sysdate, 'HH:MI:SS') from dual;
host C:\oracle\10.2.0\bin\emca.bat -config dbcontrol db -silent -DB_UNIQUE_NAME eygle
     -PORT 1521 -EM_HOME C:\oracle\10.2.0 -LISTENER LISTENER -SERVICE_NAME eygle
```

```
    -SYS_PWD &&sysPassword -SID eygle -ORACLE_HOME C:\oracle\10.2.0
    -DBSNMP_PWD &&dbsnmpPassword -HOST gqgai -LISTENER_OH C:\oracle\10.2.0
    -LOG_FILE C:\oracle\admin\eygle\scripts\emConfig.log -SYSMAN_PWD &&sysmanPassword;
spool C:\oracle\admin\eygle\scripts\postDBCreation.log
exit;
```

在创建过程中，需要经历以下几个步骤后，数据库的创建才算正式完成：

（1）　Oracle 首先通过参数文件创建了 spfile 文件；

（2）　解锁两个账号；

（3）　编译；

（4）　配置 EM。

## 1.3　使用模板创建数据库

前面提到，除了定制数据库之外，还可以使用模板来创建数据库，接下来就来了解一下使用模板创建数据库的过程。

### 1.3.1　启动创建

在 1.1 节中提到可以通过命令行启动 DBCA 工具，可能更多的朋友是通过“开始”菜单中 Oracle 创建的快捷项里来启动 DBCA 的，如图 1-20 所示。

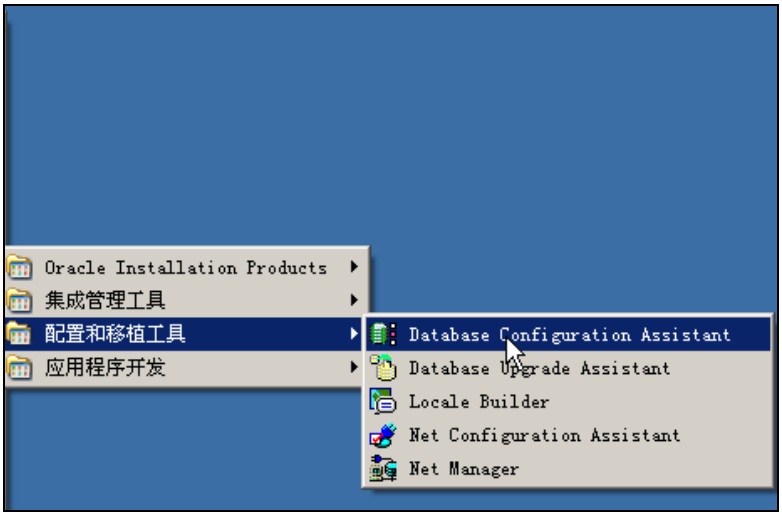


图 1-20　通过“开始”菜单启动 DBCA

从这个快捷项的属性对话框中，可以看到如图 1-21 所示的内容：

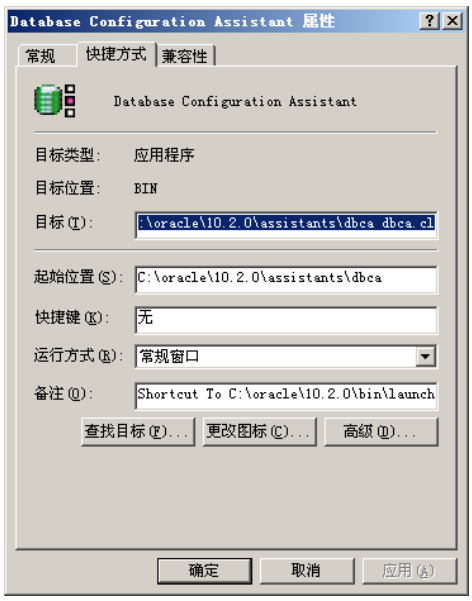


图 1-21　DBCA 属性对话框

这个快捷项的目标执行的是以下命令：

```
C:\Oracle\10.2.0\BIN\launch.exe C:\oracle\10.2.0\assistants\dbca dbca.cl
```

此处的 dbca.cl 文件和前面的 dbca.bat 批处理文件执行的功能是一致的：

```
Command=("C:\oracle\10.2.0\jdk\jre\BIN\JAVA" -Dsun.java2d.font.DisableAlgorithmicStyles=true
-DORACLE_HOME="C:\oracle\10.2.0" -DJDBC_PROTOCOL=thin
-mx128m  -classpath …..OraInstaller.jar" oracle.sysman.assistants.dbca.Dbca)
```

那么 DBCA 为什么指向这个目录呢?这个目录又是做什么用的呢？

实际上这个目录是 Oracle 的缺省模板目录，当我们使用模板来创建数据库时，就用到了这个目录下的文件。

### 1.3.2 数据库创建模板

下面来看一下使用模板创建数据库的过程。

使用模板和前面的过程主要不同之处在于第二个步骤，在这里选择“定制数据库”之外的选项，就都使用了模板，并且包含了数据文件（eygle 模板是我们之前保存的），如图 1-22 所示。


图 1-22　选择模板

使用模板创建数据库通常速度都会很快，原因就在于数据文件是从种子数据库中恢复出来的，而不需要创建文件及字典对象等信息。

这里通过脚本说明一下通过模板创建数据库和定制数据库的不同。首先 eygle.sql 脚本记录如下内容：

```
……
host C:\oracle\10.2.0\bin\orapwd.exe
     file=C:\oracle\10.2.0\database\PWDeygle.ora password=&&sysPassword force=y
@C:\oracle\admin\eygle\scripts\CloneRmanRestore.sql
@C:\oracle\admin\eygle\scripts\cloneDBCreation.sql
@C:\oracle\admin\eygle\scripts\postScripts.sql
host "echo SPFILE='C:\oracle\10.2.0/dbs/spfileeygle.ora'
                        > C:\oracle\10.2.0\database\initeygle.ora"
@C:\oracle\admin\eygle\scripts\postDBCreation.sql
```

该脚本首先调用的是 CloneRmanRestore.sql 脚本，该脚本记录如下内容：

```
C:\Oracle\admin\eygle\scripts>type CloneRmanRestore.sql
connect "SYS"/"&&sysPassword" as SYSDBA
set echo on
spool C:\oracle\admin\eygle\scripts\CloneRmanRestore.log
startup nomount pfile="C:\oracle\admin\eygle\scripts\init.ora";
@C:\oracle\admin\eygle\scripts\rmanRestoreDatafiles.sql;
```

这个脚本首先启动实例到 Nomount 模式，然后调用 rmanRestoreDatafiles.sql 来恢复文件。

### 1.3.3 Rman 的引入

rmanRestoreDatafiles.sql 脚本是通过系统包 dbms_backup_restore 来恢复备份集中的文件，从而实现数据恢复，其主要内容如下：

```
set echo off;
set serveroutput on;
select TO_CHAR(systimestamp,'YYYYMMDD HH:MI:SS') from dual;
variable devicename varchar2(255);
declare
omfname varchar2(512) := NULL;
  done boolean;
  begin
    dbms_output.put_line(' ');
    dbms_output.put_line(' Allocating device.... ');
    dbms_output.put_line(' Specifying datafiles... ');
       :devicename := dbms_backup_restore.deviceAllocate;
    dbms_output.put_line(' Specifing datafiles... ');
    dbms_backup_restore.restoreSetDataFile;
      dbms_backup_restore.restoreDataFileTo(1,
               'C:\oracle\oradata\eygle\SYSTEM01.DBF', 0, 'SYSTEM');
      dbms_backup_restore.restoreDataFileTo(2,
               'C:\oracle\oradata\eygle\UNDOTBS01.DBF', 0, 'UNDOTBS1');
      dbms_backup_restore.restoreDataFileTo(3,
               'C:\oracle\oradata\eygle\SYSAUX01.DBF', 0, 'SYSAUX');
      dbms_backup_restore.restoreDataFileTo(4,
               'C:\oracle\oradata\eygle\USERS01.DBF', 0, 'USERS');
    dbms_output.put_line(' Restoring ... ');
    dbms_backup_restore.restoreBackupPiece(
          'C:\oracle\10.2.0\assistants\dbca\templates\Seed_Database.dfb', done);
    if done then
        dbms_output.put_line(' Restore done.');
    else
        dbms_output.put_line(' ORA-XXXX: Restore failed ');
    end if;
    dbms_backup_restore.deviceDeallocate;
  end;
/
```

```
select TO_CHAR(systimestamp,'YYYYMMDD HH:MI:SS') from dual;
```

关于 RMAN 的有关知识，将会在后面的章节详细介绍，但是关于 dbms_backup_restore 包这里有必要提前介绍一下。

当通过 RMAN 进行数据库备份时，RMAN 会将多个数据文件写出到一个或多个备份文件（称为备份集）中，RMAN 的相关的备份信息或者存储在控制文件中，或者存储在 RMAN 的专用目录数据库（Catalog）中，如果 RMAN 的备份信息丢失，那么通常备份集中的文件是没有办法读取出来的，其他工具无法识别 RMAN 的备份集文件；而 dbms_backup_restore 就是针对这种情况提供的一种解决方案，dbms_backup_restore 可以在数据库 nomount 状态下调用，直接从备份集中读取数据文件，功能十分强大。

DBMS_BACKUP_RESTORE 包由 dbmsbkrs.sql 和 prvtbkrs.plb 这两个脚本创建，创建数据库时执行的 catproc.sql 脚本会调用这两个脚本以创建包，这些脚本文件可以在 $ORACLE_HOME/rdbms/admin 目录下找到，脚本文件中对包的内容有详细的介绍。

下面通过具体的例子来介绍一下这个工具的用法，以下是一次真实的恢复案例，由于控制文件丢失，只能通过 DBMS_BACKUP_RESTORE 包从备份集中恢复数据文件，当然恢复之前我们需要知道一些数据库的相关信息，了解备份集中包含了哪些文件。

首先启动数据库到 nomount 状态：

```
[oracle@jumper conner]$ sqlplus "/ as sysdba"

Connected to an idle instance.

SQL> startup nomount;
ORACLE instance started.

<...ignore SGA info here...>
```

然后可以执行脚本，将数据文件恢复到指定目录：

```
SQL> DECLARE
  2   devtype varchar2(256);
  3   done boolean;
  4   BEGIN
  5   devtype:=sys.dbms_backup_restore.deviceAllocate (type=>'',ident=>'t1');
  6   sys.dbms_backup_restore.restoreSetDatafile;
  7   sys.dbms_backup_restore.restoreDatafileTo(dfnumber=>01,
                    toname=>'/opt/oracle/oradata/conner/system01.dbf');
  8   sys.dbms_backup_restore.restoreDatafileTo(dfnumber=>02,
                    toname=>'/opt/oracle/oradata/conner/undotbs01.dbf');
  9   sys.dbms_backup_restore.restoreDatafileTo(dfnumber=>03,
                    toname=>'/opt/oracle/oradata/conner/users01.dbf');
 10   sys.dbms_backup_restore.restoreBackupPiece(done=>done,
                    handle=>'/opt/oracle/product/9.2.0/dbs/0ggmiabq_1_1', params=>null);
```

```
 11   sys.dbms_backup_restore.deviceDeallocate;
 12  END;
 13  /

PL/SQL procedure successfully completed.
```

至此，从备份集中读取文件完毕，但是由于没有控制文件，就需要重建一个控制文件用于恢复，创建控制文件的脚本可以自己根据经验编写，也可以根据备份的文本进行修改，当然也可以从其他数据库中转储一个控制文件脚本，仿照改写。

正常情况下，可以通过如下命令将控制文件的创建语句转储到跟踪文件中（位于 udump 目录中）：

```
SQL> alter database backup controlfile to trace;

Database altered.
```

可以找到 trace 文件，编辑、执行重建控制文件的需要部分：

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

Total System Global Area  101782828 bytes
Fixed Size                   451884 bytes
Variable Size              37748736 bytes
Database Buffers           62914560 bytes
Redo Buffers                 667648 bytes
SQL> set echo on
SQL> @ctl.sql
SQL>
SQL> CREATE CONTROLFILE REUSE DATABASE "CONNER" RESETLOGS  ARCHIVELOG
  2  --  SET STANDBY TO MAXIMIZE PERFORMANCE
  3      MAXLOGFILES 5
  4      MAXLOGMEMBERS 3
  5      MAXDATAFILES 100
  6      MAXINSTANCES 1
  7      MAXLOGHISTORY 1361
  8  LOGFILE
  9    GROUP 1 '/opt/oracle/oradata/conner/redo01.log'  SIZE 10M,
 10    GROUP 2 '/opt/oracle/oradata/conner/redo02.log'  SIZE 10M,
 11    GROUP 3 '/opt/oracle/oradata/conner/redo03.log'  SIZE 10M
 12  -- STANDBY LOGFILE
 13  DATAFILE
 14    '/opt/oracle/oradata/conner/system01.dbf',
```

```
 15    '/opt/oracle/oradata/conner/undotbs01.dbf',
 16    '/opt/oracle/oradata/conner/users01.dbf'
 17  CHARACTER SET ZHS16GBK
 18  ;

Control file created.
```

如果存在部分归档日志，创建控制文件之后可以执行恢复：

```
SQL> recover database using backup controlfile until cancel;
ORA-00279: change 240560269 generated at 06/09/2005 17:33:48 needed for thread 1
ORA-00289: suggestion : /opt/oracle/oradata/conner/archive/1_7.dbf
ORA-00280: change 240560269 for thread 1 is in sequence #7


Specify log: {=suggested | filename | AUTO | CANCEL}

auto
ORA-00279: change 240600632 generated at 06/10/2005 10:42:26 needed for thread 1
ORA-00289: suggestion : /opt/oracle/oradata/conner/archive/1_8.dbf
ORA-00280: change 240600632 for thread 1 is in sequence #8
ORA-00278: log file '/opt/oracle/oradata/conner/archive/1_7.dbf' no longer needed for this
recovery


Specify log: {=suggested | filename | AUTO | CANCEL}
auto
ORA-00279: change 240620884 generated at 06/10/2005 10:45:42 needed for thread 1
ORA-00289: suggestion : /opt/oracle/oradata/conner/archive/1_9.dbf
ORA-00280: change 240620884 for thread 1 is in sequence #9
ORA-00278: log file '/opt/oracle/oradata/conner/archive/1_8.dbf' no longer needed for this
recovery


ORA-00283: recovery session canceled due to errors
ORA-00600: internal error code, arguments: [3020], [4242465], [1], [9], [314], [272], [], []
ORA-10567: Redo is inconsistent with data block (file# 1, block# 48161)
ORA-10564: tablespace SYSTEM
ORA-01110: data file 1: '/opt/oracle/oradata/conner/system01.dbf'
ORA-10560: block type 'DATA SEGMENT HEADER - UNLIMITED'
```

```
ORA-01112: media recovery not started

SQL> recover database using backup controlfile until cancel;
ORA-00279: change 240620949 generated at 06/10/2005 10:45:44 needed for thread 1
ORA-00289: suggestion : /opt/oracle/oradata/conner/archive/1_9.dbf
ORA-00280: change 240620949 for thread 1 is in sequence #9


Specify log: {=suggested | filename | AUTO | CANCEL}
cancel
Media recovery cancelled.
```

恢复到最后可用日志后，通过 resetlogs 方式打开数据库：

```
SQL> alter database open resetlogs;
Database altered.

SQL> select name from v$datafile;
NAME
------------------------------------------------------------
/opt/oracle/oradata/conner/system01.dbf
/opt/oracle/oradata/conner/undotbs01.dbf
/opt/oracle/oradata/conner/users01.dbf
```

至此恢复完成。这是一次常规恢复，dbms_backup_restore 的功能远不止于此，还可以通过该包恢复备份集中的控制文件、归档日志等文件等。

继续前面的讨论，rmanRestoreDatafiles.sql 脚本通过 dbms_backup_restore 包从种子文件 Seed_Database.dfb 恢复出数据文件，来看一下模板目录中存放的模板和种子数据库（自定义的模板也存放在这个目录中）：

```
C:\Oracle\admin\eygle\scripts>dir C:\oracle\10.2.0\assistants\dbca\templates
 驱动器 C 中的卷是 SYSTEM
 卷的序列号是 8C88-D1B4
 C:\oracle\10.2.0\assistants\dbca\templates 的目录

2007-01-05  17:02    <DIR>          .
2007-01-05  17:02    <DIR>          ..
2005-08-30  17:31             5,893 Data_Warehouse.dbc
2005-09-07  13:02           983,040 example.dmp
2005-09-07  13:02        20,897,792 example01.dfb
2007-01-05  17:02             5,812 eygle.dbc
2007-01-05  15:32            12,588 eygle.dbt
```

```
2005-08-30  17:31             5,770 General_Purpose.dbc
2005-05-16  15:49            12,411 New_Database.dbt
2005-09-07  13:02         7,061,504 Seed_Database.ctl
2005-09-07  13:02        95,543,296 Seed_Database.dfb
2005-08-30  17:31             5,829 Transaction_Processing.dbc
```

Seed_Database.dfb 文件就是包含种子文件的一个备份集。

### 1.3.4 克隆数据库

数据文件具备了，接下来是通过这些文件“克隆”一个数据库，这个工作由 cloneDBCreation.sql 脚本继续执行，这个脚本更为复杂，下面分开介绍一下。

首先根据指定的数据库名称（测试数据库指定的名称为 eygle）创建一个控制文件：

```
connect "SYS"/"&&sysPassword" as SYSDBA
set echo on
spool C:\oracle\admin\eygle\scripts\cloneDBCreation.log
Create controlfile reuse set database "eygle"
MAXINSTANCES 8
MAXLOGHISTORY 1
MAXLOGFILES 16
MAXLOGMEMBERS 3
MAXDATAFILES 100
Datafile
'C:\oracle\oradata\eygle\SYSTEM01.DBF',
'C:\oracle\oradata\eygle\UNDOTBS01.DBF',
'C:\oracle\oradata\eygle\SYSAUX01.DBF',
'C:\oracle\oradata\eygle\USERS01.DBF'
LOGFILE GROUP 1 ('C:\oracle/oradata/eygle/redo01.log') SIZE 51200K,
GROUP 2 ('C:\oracle/oradata/eygle/redo02.log') SIZE 51200K,
GROUP 3 ('C:\oracle/oradata/eygle/redo03.log') SIZE 51200K RESETLOGS;
```

然后通过 dbms_backup_restore 包清空 dbid 等信息：

```
exec dbms_backup_restore.zerodbid(0);
```

看到这里再次使用到了 dbms_backup_restore 包，zeroDbid 是包中的一个过程，用于清空数据文件头的部分信息，新的 dbid 在之后的控制文件创建时可以被计算，对于数据库克隆，这是必须的。

zeroDbid 有一个输入参数，即文件号：

```
  PROCEDURE zeroDbid(fno        IN    binary_integer);
```

当 fno==0 时，控制文件中包含的所有数据文件头都将被清零，zeroDbid 主要用于清除数据文件头的 3 类信息：Database id 信息、Checksum 信息和 Checksum 符号位信息。

继续看这个脚本，清零完成之后，数据库重新启动，控制文件被重新创建，此时新的 dbid

被计算并最终写入所有数据文件：

```
shutdown immediate;
startup nomount pfile="C:\oracle\admin\eygle\scripts\initeygleTemp.ora";
Create controlfile reuse set database "eygle"
MAXINSTANCES 8
MAXLOGHISTORY 1
MAXLOGFILES 16
MAXLOGMEMBERS 3
MAXDATAFILES 100
Datafile
'C:\oracle\oradata\eygle\SYSTEM01.DBF',
'C:\oracle\oradata\eygle\UNDOTBS01.DBF',
'C:\oracle\oradata\eygle\SYSAUX01.DBF',
'C:\oracle\oradata\eygle\USERS01.DBF'
LOGFILE GROUP 1 ('C:\oracle/oradata/eygle/redo01.log') SIZE 51200K,
GROUP 2 ('C:\oracle/oradata/eygle/redo02.log') SIZE 51200K,
GROUP 3 ('C:\oracle/oradata/eygle/redo03.log') SIZE 51200K RESETLOGS;
```

注意在启动数据库时 Oracle 使用了一个临时的参数文件 initeygleTemp.ora，在这个参数文件的最后一行设置了一个内部参数：

```
C:\Oracle\admin\eygle.t\scripts>tail -1 initeygleTemp.ora

_no_recovery_through_resetlogs=true
```

_no_recovery_through_resetlogs 这个参数的作用是什么呢？可以从数据库中找到一点说明：

```
SQL> SELECT x.ksppinm NAME, y.ksppstvl VALUE, x.ksppdesc describ
  2    FROM SYS.x$ksppi x, SYS.x$ksppcv y
  3   WHERE x.inst_id = USERENV ('Instance')
  4     AND y.inst_id = USERENV ('Instance')
  5     AND x.indx = y.indx
  6     AND x.ksppinm LIKE '%&par%'
  7  /
Enter value for par: no_reco
old   6:    AND x.ksppinm LIKE '%&par%'
new   6:    AND x.ksppinm LIKE '%no_reco%'

NAME                           VALUE DESCRIB
------------------------------ ----- --------------------------------------------------
_no_recovery_through_resetlogs FALSE no recovery through this resetlogs operation
```

这个参数用于限制恢复能否跨越 resetlogs，对于数据库的恢复来说，resetlogs 通常意味着

不完全恢复，在数据库 resetlogs 打开之后，控制文件中的很多信息被改写，在 Oracle 10g 之前，如果数据库 resetlogs 打开，那么将不再能够通过当前的控制文件再次进行 resetlogs 点之前的恢复，而 Oracle 10g 改变了这个历史。

在 Oracle 10g 中，即使通过 resetlogs 方式打开了数据库，Oracle 仍然支持再次从 resetlogs 时间点之前进行恢复；在 Clone 数据库时，Oracle 设置这个参数为 True，意思就是不允许再次进行跨越 resetlogs 时间点的恢复。关于这部分内容，将在后面章节中进行更为详细的介绍。

继续解读这个脚本，接下来 Oracle 设置 restricted session 模式，resetlogs 打开数据库：

```
alter system enable restricted session;
alter database "eygle" open resetlogs;
```

修改 global_name，添加临时文件等：

```
alter database rename global_name to "eygle";
ALTER TABLESPACE TEMP ADD TEMPFILE 'C:\oracle\oradata\eygle\TEMP01.DBF' SIZE 20480K REUSE
AUTOEXTEND ON NEXT 640K MAXSIZE UNLIMITED;
select tablespace_name from dba_tablespaces where tablespace_name='USERS';
select sid, program, serial#, username from v$session;
```

由于种子数据库的字符集通常与用户要求的不符，接下来 Oracle 通过内部操作强制更改了字符集、国家字符集（这个内容在后面的章节有详细的介绍）：

```
alter database character set INTERNAL_CONVERT ZHS16GBK;
alter database national character set INTERNAL_CONVERT AL16UTF16;
```

最后修改用户口令，禁用 restricted session 模式，这个克隆过程执行完毕：

```
alter user sys identified by "&&sysPassword";
alter user system identified by "&&systemPassword";
alter system disable restricted session;
```

至此，种子数据库已经按照用户的意图脱胎换骨得以重生。

### 1.3.5 可传输表空间

在很多 Oracle 文档中，可能大家都注意过 Oracle 用来进行测试的一个表空间，这个表空间中有一系列预置的用户和数据，可以用于数据库或 BI 的很多测试实验。

这个表空间在使用模板建库时是可以选择的，在如图 1-22 所示的这个界面中，可以选择建库时包含这个范例表空间（缺省是未选择的）。


图 1-22　是否包含示例方案

如果选择了包含示例方案，则 cloneDBCreation.sql 脚本将会有所改变，主要增加了如下语句：

```
connect "SYS"/"&&sysPassword" as SYSDBA
@C:\oracle\10.2.0\demo\schema\mkplug.sql &&sysPassword change_on_install change_on_install
change_on_install        change_on_install        change_on_install        change_on_install
C:\oracle\10.2.0\assistants\dbca\templates\example.dmp
C:\oracle\10.2.0\assistants\dbca\templates\example01.dfb
C:\oracle\oradata\eygle\example01.dbf C:\oracle\admin\eygle\scripts\ "'SYS/&&sysPassword as
SYSDBA'";
```

看到这里，再次引用了模板目录中的文件：

```
C:\>dir C:\oracle\10.2.0\assistants\dbca\templates\ex*
 驱动器 C 中的卷是 SYSTEM
 卷的序列号是 8C88-D1B4

 C:\oracle\10.2.0\assistants\dbca\templates 的目录

2005-09-07  13:02           983,040 example.dmp
2005-09-07  13:02        20,897,792 example01.dfb
               2 个文件     21,880,832 字节
               0 个目录    915,578,880 可用字节
```

通过 mkplug.sql 脚本来加载这个范例表空间，来看一下这个脚本的主要内容。同样，最重要的是通过 dbms_backup_restore 包从 example01.dfb 文件中恢复数据文件：

```
SELECT TO_CHAR(systimestamp, 'YYYYMMDD HH:MI:SS')  FROM dual;
variable new_datafile varchar2(512)
declare
  done boolean;
  v_db_create_file_dest VARCHAR2(512);
  devicename varchar2(255);
  data_file_id number;
  rec_id number;
  stamp number;
  resetlogs_change number;
  creation_change number;
  checkpoint_change number;
  blksize number;
  omfname varchar2(512);
  real_file_name varchar2(512);

  begin
    dbms_output.put_line(' ');
    dbms_output.put_line(' Allocating device.... ');
    dbms_output.put_line(' Specifying datafiles... ');
       devicename := dbms_backup_restore.deviceAllocate;
    dbms_output.put_line(' Specifing datafiles... ');
    SELECT MAX(file_id)+1 INTO data_file_id FROM dba_data_files;
    SELECT value INTO v_db_create_file_dest FROM v$parameter WHERE name ='db_create_file_dest';
    IF v_db_create_file_dest IS NOT NULL
     THEN
      dbms_backup_restore.restoreSetDataFile;
      dbms_backup_restore.getOMFFileName('EXAMPLE',omfname);
      dbms_backup_restore.restoreDataFileTo(data_file_id, omfname, 0,'EXAMPLE');
     ELSE
      dbms_backup_restore.restoreSetDataFile;
      dbms_backup_restore.restoreDataFileTo(data_file_id,'&data_file_name');
     END IF;
    dbms_output.put_line(' Restoring ... ');
    dbms_backup_restore.restoreBackupPiece('&data_file_backup', done);
    SELECT max(recid) INTO rec_id FROM v$datafile_copy;
```

```
    -- Now get the real file name. It could be also OMF filename
    SELECT name, stamp, resetlogs_change#, creation_change#, checkpoint_change#,block_size
    INTO real_file_name, stamp,resetlogs_change, creation_change, checkpoint_change, blksize
    FROM V$DATAFILE_COPY
     WHERE recid = rec_id and file# = data_file_id;

    -- Uncatalog the file from V$DATAFILE_COPY. This important.
    dbms_backup_restore.deleteDataFileCopy(recid => rec_id,
                                           stamp => stamp,
                                           fname => real_file_name,
                                           dfnumber => data_file_id,
                                           resetlogs_change => resetlogs_change,
                                           creation_change => creation_change,
                                           checkpoint_change => checkpoint_change,
                                           blksize => blksize,
                                           no_delete => 1,
                                           force => 1);
    -- Set the bindvariable to the real filename
    :new_datafile := real_file_name;

    if done then
        dbms_output.put_line(' Restore done.');
    else
        dbms_output.put_line(' ORA-XXXX: Restore failed ');
    end if;
  end;
/
```

这个恢复完成之后，接下来最重要的部分就是通过传输表空间技术将 example 表空间导入到当前的数据库。

考虑一下这种情况，当进行跨数据库迁移时，需要将一个用户表空间中的数据迁移到另外一个数据库，应该使用什么样的方法呢？最常规的做法可能是通过 EXP 工具将数据全部导出，然后在目标数据库上 IMP 导入，可是这种方法可能会比较缓慢。EXP 工具同时还提供另外一种技术-可传输表空间技术，可以用于加快这个过程。

在 exp –help 的帮助中，可以看到这样一个参数：

```
TRANSPORT_TABLESPACE 导出可传输的表空间元数据 (N)
```

通过这个选项，我们可以对一组自包含、只读的表空间只导出元数据，然后在操作系统层将这些表空间的数据文件拷贝至目标平台，并将元数据导入数据字典（这个过程称为插入，plugging），即完成迁移。

对于可传输表空间有一个重要概念：自包含（Self-Contained）。

在表空间传输的中，要求表空间集为自包含的，自包含表示用于传输的内部表空间集没有引用指向外部表空间集。自包含分为两种：一般自包含表空间集和完全（严格）自包含表空间集。

常见的以下情况是违反自包含原则的：

- 索引在内部表空间集，而表在外部表空间集（相反地，如果表在内部表空间集，而索引在外部表空间集，则不违反自包含原则）。
- 分区表一部分区在内部表空间集，一部分在外部表空间集（对于分区表，要么全部包含在内部表空间集中，要么全不包含）。
- 如果在传输表空间时同时传输约束，则对于引用完整性约束，约束指向的表在外部表空间集，则违反自包含约束；如果不传输约束，则与约束指向无关。
- 表在内部表空间集，而 Lob 列在外部表空间集，则违反自包含约束。

通常可以通过系统包 DBMS_TTS 来检查表空间是否自包含，验证可以以两种方式执行：非严格方式和严格方式。

以下是一个简单的验证过程，假定在 eygle 表空间存在一个表 eygle，其上存在索引存储在 USERS 表空间：

```
SQL> create table eygle as select rownum id ,username from dba_users;
Table created.

SQL> create index ind_id on eygle(id) tablespace users;
Index created.
```

以 SYS 用户执行非严格自包含检查（full_check=false）：

```
SQL> connect / as sysdba
Connected.
SQL> exec dbms_tts.transport_set_check('EYGLE', TRUE);
PL/SQL procedure successfully completed.

SQL> SELECT * FROM TRANSPORT_SET_VIOLATIONS;
no rows selected
```

执行严格自包含检查（full_check=true）：

```
SQL> exec dbms_tts.transport_set_check('EYGLE', TRUE, True);
PL/SQL procedure successfully completed.

SQL> SELECT * FROM TRANSPORT_SET_VIOLATIONS;
VIOLATIONS
--------------------------------------------------------------------------------
Index EYGLE.IND_ID in tablespace USERS points to table EYGLE.EYGLE in tablespace EYGLE
```

反过来对于 USERS 表空间来说，非严格检查也是无法通过的：

```
SQL> exec dbms_tts.transport_set_check('USERS', TRUE);
PL/SQL procedure successfully completed.
```

```
SQL> SELECT * FROM TRANSPORT_SET_VIOLATIONS;
VIOLATIONS
--------------------------------------------------------------------------------
Index EYGLE.IND_ID in tablespace USERS points to table EYGLE.EYGLE in tablespace EYGLE
```

但是可以对多个表空间同时传输，则一些自包含问题就可以得到解决：

```
SQL> exec dbms_tts.transport_set_check('USERS,EYGLE', TRUE, True);
PL/SQL procedure successfully completed.

SQL> SELECT * FROM TRANSPORT_SET_VIOLATIONS;
no rows selected
```

表空间自包含确认之后，进行表空间传输就很方便了，一般包含如下几个步骤。

（1） 将表空间设置为只读：

```
alter tablespace users read only;
```

（2） 导出表空间。在操作系统提示符下执行：

```
exp username/passwd tablespaces=users transport_tablespace=y file=exp_users.dmp
```

此处的导出文件只包含元数据，所以导出文件很小，导出速度也会很快。

（3） 转移。将导出的元数据文件（此处是 exp_users.dmp）和传输表空间的数据文件（此处是 users 表空间的数据文件 user01.dbf)转移至目标主机(转移过程如果使用 FTP 方式，应该注意使用二进制方式)。

（4） 传输。在目标数据库将表空间插入到数据库中，完成表空间传输。在操作系统命令提示符下执行下面的语句：

```
imp username/passwd tablespaces=users transport_tablespace=y file=exp_users.dmp
datafiles='users01.dbf'
```

了解了 Oracle 的可传输表空间技术后，来看一下 example 表空间的插入，以下脚本仍然来自 mkplug.sql 脚本：

```
--
-- Importing the metadata and plugging in the tablespace at the same
-- time, using the restored database file
--
DEFINE imp_logfile = &log_path.tts_example_imp.log

-- When importing use filename got after restore is finished
host imp "'sys/&&password_sys AS SYSDBA'" transport_tablespace=y file=&imp_file
log=&imp_logfile datafiles='&datafile' tablespaces=EXAMPLE tts_owners=hr,oe,pm,ix,sh
```

完成 plugging 之后，这个表空间就被包含在了新建的数据库之中。

## 1.3.6 跨平台表空间传输

需要注意的是，在 Oracle 10g 之前，数据文件是不能够跨平台传输使用的，从 Oracle 10g 开始，Oracle 支持跨平台的表空间传输，这极大地增强了数据迁移的便利性。

### 1. 字节顺序和平台

数据文件所以不能跨平台，主要是由于不同平台的字节顺序不同，这是计算机领域由来已久的问题之一，在各种计算机体系结构中，由于对于字、字节等的存储机制有所不同，通信双方交流的信息单元（比特、字节、字、双字等）应该以什么样的顺序进行传送就成了一个问题，如果不达成一致的规则，通信双方将无法进行正确的编/译码从而导致通信失败。

目前在各种体系的计算机中通常采用的字节存储机制主要有两种：Big-Endian 和 Little-Endian 。

一些操作系统（包括 Windows）在低位内存地址中存放二进制数据的最低有效字节，因此这种系统被称为 Little Endian；一些操作系统（包括 Solaris）将最高有效字节存储在低位内存地址中，因此这种系统被称为 Big Endian。

举一个简单点的例子，假如 1122 这样一个数据要存入不同系统，对于 Little Endian 的系统，存储的顺序就是 2211，小头在前；而对于 Big Endian 的系统来说，存储顺序就是 1122，大头在前，显然 Big Endian 更符合我们通常的语言习惯。

那么跨平台的问题就出现了，当一个 Little Endian 的系统试图从一个 Big Endian 的系统中读取数据时，就需要通过转换，否则不同的字节顺序将导致数据不能被正确读取。

> 说明：据考证，Endian 这个词来源于 Jonathan Swift 在 1726 年写的讽刺小说《Gulliver's Travels》(《格利佛游记》)。该小说在描述 Gulliver 畅游小人国时碰到了如下的一个场景。在小人国里的小人因为非常小（身高 6 英寸）所以总是碰到一些意想不到的问题。有一次因为对水煮蛋该从大的一端（Big-End）剥开还是小的一端（Little-End）剥开的争论而引发了一场战争，并形成了两支截然对立的队伍：支持从 Big-End 剥开的人 Swift 就称作 Big-Endians，而支持从 Little-End 剥开的人就称作 Little-Endians（后缀 ian 表明的就是支持某种观点的人）。Endian 这个词由此而来。

清楚了这个问题，接下来就可以来看看 Oracle 是如何处理这种情况的。

### 2. 源平台和目标平台

首先在迁移之前，需要确认一下源平台和目标平台的平台信息，这些信息可以通过视图 v$transportable_platform 和 v$database 视图联合查询得到。以下是源平台的信息：

```
SQL> col PLATFORM_NAME for a30
SQL> SELECT d.platform_name, endian_format
  2    FROM v$transportable_platform tp, v$database d
  3   WHERE tp.platform_name = d.platform_name;
PLATFORM_NAME                  ENDIAN_FORMAT
------------------------------ --------------
Solaris[tm] OE (64-bit)        Big
```

查询目标数据库平台信息：

```
SQL> col platform_name for a40
SQL> SELECT d.platform_name, endian_format
  2    FROM v$transportable_platform tp, v$database d
  3   WHERE tp.platform_name = d.platform_name;
PLATFORM_NAME                            ENDIAN_FORMAT
---------------------------------------- --------------
Microsoft Windows IA (32-bit)            Little
```

看到Windows平台和Solaris平台的字节顺序是不同的，Windows是Little-Endian，而Solaris是Big-Endian的。可以通过数据库查询Oracle 10g支持的平台转换：

```
SQL> col PLATFORM_NAME for a40
SQL> select * from v$transportable_platform;
PLATFORM_ID PLATFORM_NAME                            ENDIAN_FORMAT
----------- ---------------------------------------- --------------
          1 Solaris[tm] OE (32-bit)                  Big
          2 Solaris[tm] OE (64-bit)                  Big
          7 Microsoft Windows IA (32-bit)            Little
         10 Linux IA (32-bit)                        Little
          6 AIX-Based Systems (64-bit)               Big
          3 HP-UX (64-bit)                           Big
          5 HP Tru64 UNIX                            Little
          4 HP-UX IA (64-bit)                        Big
         11 Linux IA (64-bit)                        Little
         15 HP Open VMS                              Little
          8 Microsoft Windows IA (64-bit)            Little
          9 IBM zSeries Based Linux                  Big
         13 Linux 64-bit for AMD                     Little
         16 Apple Mac OS                             Big
         12 Microsoft Windows 64-bit for AMD         Little
         17 Solaris Operating System (x86)           Little
         18 IBM Power Based Linux                    Big
17 rows selected.
```

### 3. 源平台的导出及转换

接下来开始我们的测试，创建一个独立的自包含表空间，并创建一个测试表：

```
SQL> select name from v$datafile;
NAME
--------------------------------------------------------------------------------
/data2/ora10g/oradata/mars/system01.dbf
/data2/ora10g/oradata/mars/undotbs01.dbf
```

```
/data2/ora10g/oradata/mars/sysaux01.dbf
/data2/ora10g/oradata/mars/users01.dbf

SQL> create tablespace trans
  2  datafile '/data2/ora10g/oradata/mars/trans.dbf' size 10M;
Tablespace created.

SQL> create user trans identified by trans
  2  default tablespace trans;
User created.

SQL> grant connect,resource to trans;
Grant succeeded.

SQL> connect trans/trans
Connected.
SQL> create table test as select * from dict;
Table created.

SQL> select count(*) from test;
  COUNT(*)
----------
       617
```

将表空间设置为只读：

```
SQL> connect / as sysdba
Connected.
SQL> alter tablespace trans read only;
Tablespace altered.
```

导出要传输的表空间：

```
$ exp \'/ as sysdba\' tablespaces=trans transport_tablespace=y file=exp_trans.dmp

Export: Release 10.2.0.1.0 - Production on Thu Mar 22 16:31:15 2007
Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Connected to: Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - 64bit Production
With the Partitioning, OLAP and Data Mining options
Export done in ZHS16GBK character set and AL16UTF16 NCHAR character set
Note: table data (rows) will not be exported
About to export transportable tablespace metadata...
```

```
For tablespace TRANS ...
. exporting cluster definitions
. exporting table definitions
. . exporting table                          TEST
. exporting referential integrity constraints
. exporting triggers
. end transportable tablespace metadata export
Export terminated successfully without warnings.
```

使用 RMAN 转换文件格式：

```
$ rman target /

Recovery Manager: Release 10.2.0.1.0 - Production on Thu Mar 22 16:34:30 2007
Copyright (c) 1982, 2005, Oracle.  All rights reserved.
connected to target database: MARS (DBID=1034439893)

RMAN> convert tablespace trans
2> to platform 'Microsoft Windows IA (32-bit)'
3> format '/tmp/%N_%f';

Starting backup at 22-MAR-07
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: sid=140 devtype=DISK
channel ORA_DISK_1: starting datafile conversion
input datafile fno=00005 name=/data2/ora10g/oradata/mars/trans.dbf
converted datafile=/tmp/TRANS_5
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:01
Finished backup at 22-MAR-07
```

确认导出文件已生成：

```
$ ls -l /tmp/TRANS*
-rw-r-----   1 oracle   dba      10493952 Mar 22 16:37 /tmp/TRANS_5
```

### 3. 文件传输

通过 FTP 获得两个文件，注意应该使用二进制方式传输（bin 模式）：

```
D:\oradata\EYGLE\DATAFILE>ftp 172.16.33.50
Connected to 172.16.33.50.
220 testdbserver.hurray.com.cn FTP server (SunOS 5.8) ready.
User (172.16.33.50:(none)): gqgai
331 Password required for gqgai.
```

```
Password:
230 User gqgai logged in.
ftp> bin
200 Type set to I.
ftp> mget /export/home/oracle/exp_trans.dmp
200 Type set to I.
mget /export/home/oracle/exp_trans.dmp? y
200 PORT command successful.
150 Binary data connection for /export/home/oracle/exp_trans.dmp (172.16.34.89,5006) (3072
bytes).
226 Binary Transfer complete.
ftp: 收到 3072 字节，用时 0.00Seconds 3072000.00Kbytes/sec.
ftp> mget /tmp/TRANS_5
200 Type set to I.
mget /tmp/TRANS_5? y
200 PORT command successful.
150 Binary data connection for /tmp/TRANS_5 (172.16.34.89,5008) (10493952 bytes).
226 Binary Transfer complete.
ftp: 收到 10493952 字节，用时 1.13Seconds 9270.28Kbytes/sec.
```

4. 目标数据库的导入

在目标数据库中，也可以使用 RMAN 对备份文件进行转换，以使数据文件具有更规范的名称：

```
D:\oradata\EYGLE\DATAFILE>rman target /

恢复管理器: Release 10.2.0.1.0 - Production on 星期四 3月 22 17:18:50 2007
Copyright (c) 1982, 2005, Oracle.  All rights reserved.

连接到目标数据库: EYGLE (DBID=1417824532)

RMAN> convert datafile 'D:\oradata\EYGLE\DATAFILE\TRANS_5'
2> db_file_name_convert
3> 'D:\oradata\EYGLE\DATAFILE\TRANS_5','D:\oradata\EYGLE\DATAFILE\TRANS01.DBF';

启动 backup 于 22-3月 -07
使用目标数据库控制文件替代恢复目录
分配的通道: ORA_DISK_1
通道 ORA_DISK_1: sid=144 devtype=DISK
通道 ORA_DISK_1: 启动数据文件转换
```

```
输出文件名=D:\ORADATA\EYGLE\DATAFILE\TRANS_5
已转换的数据文件 = D:\ORADATA\EYGLE\DATAFILE\TRANS01.DBF
通道 ORA_DISK_1: 数据文件转换完毕, 经过时间: 00:00:08
完成 backup 于 22-3 月 -07
```

然后需要在目标数据库创建相应的用户：

```
SQL> create user trans identified by trans;
用户已创建。

SQL> grant connect,resource to trans;
授权成功。
```

接下来可以执行导入：

```
D:\oradata\EYGLE\DATAFILE>imp '/ as sysdba' tablespaces=trans transport_tablespace=y
file=exp_trans.dmp datafiles=D:\oradata\EYGLE\DATAFILE\TRANS01.DBF

Import: Release 10.2.0.1.0 - Production on 星期四 3 月 22 17:34:27 2007
Copyright (c) 1982, 2005, Oracle.  All rights reserved.

连接到: Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

经由常规路径由 EXPORT:V10.02.01 创建的导出文件
即将导入可传输的表空间元数据...
已经完成 ZHS16GBK 字符集和 AL16UTF16 NCHAR 字符集中的导入
. 正在将 SYS 的对象导入到 SYS
. 正在将 SYS 的对象导入到 SYS
. 正在将 TRANS 的对象导入到 TRANS
. . 正在导入表                          "TEST"
. 正在将 SYS 的对象导入到 SYS
成功终止导入, 没有出现警告。
```

注意：此处也可以在 IMP 时通过 fromuser/touser 参数将数据导入其他用户下。

现在这个表空间已经被插入到新的数据库中，并且数据全部传输过来：

```
SQL> select name from v$datafile;
NAME
------------------------------------------------------------
D:\ORADATA\EYGLE\DATAFILE\O1_MF_SYSTEM_2G8OHFX6_.DBF
D:\ORADATA\EYGLE\DATAFILE\O1_MF_UNDOTBS1_2G8OJ6NB_.DBF
D:\ORADATA\EYGLE\DATAFILE\O1_MF_SYSAUX_2G8OJHP9_.DBF
D:\ORADATA\EYGLE\DATAFILE\O1_MF_USERS_2G8OJYYS_.DBF
D:\ORADATA\EYGLE\DATAFILE\O1_MF_EYGLE_2YDGSVH7_.DBF
```

```
D:\ORADATA\EYGLE\DATAFILE\TRANS01.DBF
已选择 6 行。

SQL> select count(*) from trans.test;
  COUNT(*)
----------
       617
```

导入后的表空间还处于 read only 状态，确认后可以更改为读写模式：

```
SQL> select tablespace_name,status from dba_tablespaces;
TABLESPACE_NAME                STATUS
------------------------------ ---------
SYSTEM                         ONLINE
UNDOTBS1                       ONLINE
SYSAUX                         ONLINE
TEMP                           ONLINE
USERS                          ONLINE
EYGLE                          ONLINE
TRANS                          READ ONLY
已选择 7 行。

SQL> alter tablespace trans read write;
表空间已更改。
```

同样，传输表空间也可以通过数据泵来完成，以下是 Oracle 10gR1 中插入表空间的简单示例：

```
E:\Oracle\oradata\eygle\dpdata>impdp eygle/eygle dumpfile=trans.dmp directory=dpdata
transport_datafiles='E:\Oracle\oradata\eygle\EYGLE\DATAFILE\TRANS01.DBF'

Import: Release 10.1.0.2.0 - Production on 星期二, 27 4 月, 2004 15:03
Copyright (c) 2003, Oracle. All rights reserved.

连接到: Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production
With the Partitioning, OLAP and Data Mining options
已成功加载/卸载了主表 "EYGLE"."SYS_IMPORT_TRANSPORTABLE_01"
启　动　"EYGLE"."SYS_IMPORT_TRANSPORTABLE_01": eygle/******** dumpfile=trans.dmp
directory=dpdata transport_datafiles='E:\
Oracle\oradata\eygle\EYGLE\DATAFILE\TRANS01.DBF'
处理对象类型 TRANSPORTABLE_EXPORT/PLUGTS_BLK
处理对象类型 TRANSPORTABLE_EXPORT/TABLE
处理对象类型 TRANSPORTABLE_EXPORT/TTE_POSTINST/PLUGTS_BLK
```

```
作业 "EYGLE"."SYS_IMPORT_TRANSPORTABLE_01" 已于 15:03 成功完成
```

### 5. 同字节序文件的跨平台

前面说过，当一个 Little Endian 的系统试图从一个 Big Endian 的系统中读取数据时，就需要通过转换，否则不同的字节顺序将导致数据不能被正确读取。那么另外一个问题出现了，如果字节序相同的平台进行文件交互，数据能否被正确读取呢？

理论上的确是可以的，但是由于在不同的平台上操作系统会在数据文件头写上系统信息，这部分信息无法跨越平台，所以仍然会导致跨平台的文件无法被数据库正确识别（Oracle 10g 中同字节序平台数据文件头不再存在跨平台的迁移问题）。

接下来通过 Windows 和 Linux 平台来进行一个跨平台测试。相信通过这个测试可以对以上提出的问题做出一个很好的回答。

其中一个实验环境为 Windows XP + Oracle10g 10.2.0.1，具体如下：

```
SQL> select * from v$version;
BANNER
----------------------------------------------------------------
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Prod
PL/SQL Release 10.2.0.1.0 - Production
CORE    10.2.0.1.0      Production
TNS for 32-bit Windows: Version 10.2.0.1.0 - Production
NLSRTL Version 10.2.0.1.0  -  Production
```

另一个实验环境为 Red Hat Enterprise Linux AS release 3 + Oracle 9iR2 9.2.0.4，具体如下：

```
SQL> select * from v$version;
BANNER
----------------------------------------------------------------
Oracle9i Enterprise Edition Release 9.2.0.4.0 - Production
PL/SQL Release 9.2.0.4.0 - Production
CORE    9.2.0.3.0       Production
TNS for Linux: Version 9.2.0.4.0 - Production
NLSRTL Version 9.2.0.4.0 - Production
```

看一下 Linux 平台，文件头被操作系统保留了 8192 字节：

```
SQL> select file_name,bytes from dba_data_files
  2  where tablespace_name='USERS';
FILE_NAME                                  BYTES
---------------------------------- ----------
/opt/oracle/oradata/eygle/users.dbf    10485760

SQL> !
[oracle@jumper eygle]$ ll users.dbf
-rw-r-----    1 oracle   dba       10493952 Mar 23 10:14 users.dbf
```

```
[oracle@jumper eygle]$ exit
exit

SQL> select 10493952 -10485760 diff from dual;
      DIFF
----------
      8192
```

Windows 平台上数据文件头同样保留了 8192 字节：

```
SQL> select file_name,bytes from dba_data_files
  2  where tablespace_name='USERS';
FILE_NAME                                                BYTES
-------------------------------------------------- ----------
D:\ORADATA\EYGLE\DATAFILE\O1_MF_USERS_2G8OJYYS_.DBF     5242880

SQL> host dir D:\ORADATA\EYGLE\DATAFILE\O1_MF_USERS_2G8OJYYS_.DBF
 驱动器 D 中的卷是 PRIVAT
 卷的序列号是 94BO-FD3B

 D:\ORADATA\EYGLE\DATAFILE 的目录

2007-03-22  17:41         5,251,072 O1_MF_USERS_2G8OJYYS_.DBF
               1 个文件      5,251,072 字节
               0 个目录  1,635,913,728 可用字节
SQL> select 5251072 -5242880 diff from dual;
      DIFF
----------
      8192
```

可以通过 Linux 和 Windows 平台来进行一个小测试实验，这两个平台都是 Little Endian 的系统：

```
SQL> select * from v$transportable_platform;

PLATFORM_ID PLATFORM_NAME                          ENDIAN_FORMAT
----------- ------------------------------------ --------------
          1 Solaris[tm] OE (32-bit)              Big
          2 Solaris[tm] OE (64-bit)              Big
          7 Microsoft Windows IA (32-bit)        Little
         10 Linux IA (32-bit)                    Little
.....
17 rows selected.
```

首先在 Linux 下 Oracle 9204 中创建一个测试表空间：

```
[oracle@jumper oracle]$ cd oradata/eygle
[oracle@jumper eygle]$ sqlplus "/ as sysdba"
Connected to:
Oracle9i Enterprise Edition Release 9.2.0.4.0 - Production
With the Partitioning option
JServer Release 9.2.0.4.0 - Production

SQL> create tablespace eyglee datafile size 10M;
Tablespace created.
```

创建测试用户并创建一个测试表：

```
SQL> create user eyglee identified by eyglee default tablespace eyglee;
User created.

SQL> grant connect,resource to eyglee;
Grant succeeded.

SQL> connect eyglee/eyglee
Connected.
SQL> create table eyglee as select * from dict;
Table created.

SQL> select count(*) from eyglee;
  COUNT(*)
----------
       477
```

将表空间设置为只读：

```
SQL> connect / as sysdba
Connected.

SQL> alter tablespace eyglee read only;
Tablespace altered.

SQL> select file_name from dba_data_files where tablespace_name='EYGLEE';
FILE_NAME
-------------------------------------------------------------
/opt/oracle/oradata/eygle/o1_mf_eyglee_309yc9gr_.dbf
```

压缩文件以方便传输：

```
[oracle@jumper eygle]$ tar -cvf eyglee.tar o1_mf_eyglee_309yc9gr_.dbf
```

```
o1_mf_eyglee_309yc9gr_.dbf
[oracle@jumper eygle]$ gzip eyglee.tar
```

导出表空间：

```
[oracle@jumper  eygle]$  exp  \'/  as  sysdba\'  tablespaces=eyglee  transport_tablespace=y
file=trans_eyglee.dmp

Export: Release 9.2.0.4.0 - Production on Sat Mar 24 18:17:32 2007
Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.

Connected to: Oracle9i Enterprise Edition Release 9.2.0.4.0 - Production
With the Partitioning option
JServer Release 9.2.0.4.0 - Production
Export done in ZHS16GBK character set and AL16UTF16 NCHAR character set
Note: table data (rows) will not be exported
About to export transportable tablespace metadata...
For tablespace EYGLEE ...
. exporting cluster definitions
. exporting table definitions
. . exporting table                          EYGLEE
. exporting referential integrity constraints
. exporting triggers
. end transportable tablespace metadata export
Export terminated successfully without warnings.

[oracle@jumper eygle]$ ll *eyglee*
-rw-r--r--    1 oracle   dba         32985 Mar 24 18:14 eyglee.tar.gz
-rw-r-----    1 oracle   dba      10493952 Mar 24 18:13 o1_mf_eyglee_309yc9gr_.dbf
-rw-r--r--    1 oracle   dba         16384 Mar 24 18:17 trans_eyglee.dmp
```

传输文件到 Windows 平台：

```
D:\oradata\EYGLE\DATAFILE>dir *eyglee*
 D:\oradata\EYGLE\DATAFILE 的目录

2007-03-24  18:21            32,985 eyglee.tar.gz
2007-03-24  18:21            16,384 trans_eyglee.dmp

D:\oradata\EYGLE\DATAFILE>gzip -d eyglee.tar.gz
D:\oradata\EYGLE\DATAFILE>tar -xvf eyglee.tar
tar: blocksize = 20
x o1_mf_eyglee_309yc9gr_.dbf, 10493952 bytes, 20496 tape blocks
```

在 Windows 上创建新用户：

```
SQL> create user eyglee identified by eyglee;
用户已创建。

SQL> grant connect ,resource to eyglee;
授权成功。
```

如果此时导入会出现 ORA-00600 错误：

```
D:\oradata\EYGLE\DATAFILE>imp  '/  as  sysdba'  tablespaces=eyglee  transport_tablespace=y
file=trans_eyglee.dmp datafiles=d:\oradata\EYGLE\DATAFILE\o1_mf_eyglee_309yc9gr_.dbf

Import: Release 10.2.0.1.0 - Production on 星期六 3 月 24 18:59:23 2007
Copyright (c) 1982, 2005, Oracle.  All rights reserved.

连接到: Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

经由常规路径由 EXPORT:V09.02.00 创建的导出文件
即将导入可传输的表空间元数据...
已经完成 ZHS16GBK 字符集和 AL16UTF16 NCHAR 字符集中的导入
. 正在将 SYS 的对象导入到 SYS
. 正在将 SYS 的对象导入到 SYS
IMP-00017: 由于 ORACLE 错误 600, 以下语句失败:
 "BEGIN   sys.dbms_plugts.beginImpTablespace('EYGLEE',9,'SYS',1,0,8192,1,1899"
 "6106462,1,2147483645,8,128,8,0,1,0,8,1407686520,1,1,18996106397,NULL,0,0,NU"
 "LL,NULL); END;"
IMP-00003: 遇到 ORACLE 错误 600
ORA-00600: 内部错误代码, 参数: [krhcvt_filhdr_v10_01], [], [], [], [], [], [], []
ORA-06512: 在 "SYS.DBMS_PLUGTS", line 1801
ORA-06512: 在 line 1
IMP-00000: 未成功终止导入
```

其中“参数: [krhcvt_filhdr_v10_01]”提示指文件头无法正确识别。可以通过对这个文件进行一个特殊操作，为文件更换一个 Windows 下数据文件的文件头，则数据文件就应该能够被数据库识别。

以下是这个“小手术”操作的过程。首先提取一个 Windows 数据文件头：

```
D:\oradata\EYGLE\DATAFILE>dd if=O1_MF_USERS_2G8OJYYS_.DBF of=header.dbf bs=8192 count=1
1+0 records in
1+0 records out
```

然后去除 Linux 下的数据文件头：

```
D:\oradata\EYGLE\DATAFILE>dd if=o1_mf_eyglee_309yc9gr_.dbf of=eyglee.dbf bs=8192 skip=1
```

```
1280+0 records in
1280+0 records out
```

最后将这两个文件合二为一：

```
D:\oradata\EYGLE\DATAFILE>copy /b header.dbf+eyglee.dbf eygleee.dbf
header.dbf
eyglee.dbf
已复制         1 个文件。
```

现在拥有的新文件 eygleee.dbf 就具有了一个 Windows 平台的文件头以及 Linux 下的“文件身”。至此这个文件就能够被 Windows 上的 Oracle 识别了，可以执行导入操作：

```
D:\oradata\EYGLE\DATAFILE>imp  '/  as  sysdba'  tablespaces=eyglee  transport_tablespace=y
file=trans_eyglee.dmp datafiles=d:\oradata\EYGLE\DATAFILE\eygleee.dbf

Import: Release 10.2.0.1.0 - Production on 星期六 3 月 24 19:22:13 2007
Copyright (c) 1982, 2005, Oracle.  All rights reserved.

连接到: Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

经由常规路径由 EXPORT:V09.02.00 创建的导出文件
即将导入可传输的表空间元数据...
已经完成 ZHS16GBK 字符集和 AL16UTF16 NCHAR 字符集中的导入
. 正在将 SYS 的对象导入到 SYS
. 正在将 SYS 的对象导入到 SYS
. 正在将 EYGLEE 的对象导入到 EYGLEE
. . 正在导入表                          "EYGLEE"
成功终止导入，没有出现警告。
```

此时数据已经能够被正确识别：

```
SQL> connect eyglee/eyglee
已连接。
SQL> select count(*) from eyglee;

  COUNT(*)
----------
       477
```

最后将表空间更改为读写模式，可以进行正常的数据操作：

```
SQL> connect / as sysdba
已连接。
SQL> alter tablespace eyglee read write;
表空间已更改。
```

```
SQL> connect eyglee/eyglee
已连接。
SQL> insert into eyglee select * from eyglee;
已创建 477 行。
SQL> insert into eyglee select * from eyglee;
已创建 954 行。
SQL> insert into eyglee select * from eyglee;
已创建 1908 行。
SQL> commit;
提交完成。
SQL> select count(*) from eyglee;
  COUNT(*)
----------
      3816
```

通过这个实验，还可以得出另外一个结论，Oracle 9i 的数据文件可以通过表空间传输迁移到 Oracle 10g 中使用。

### 6. Oracle 10g 同字节序跨平台迁移的增强

上一节介绍的方法不免过于复杂，也许很多朋友会发出这样的疑问，既然同字节序数据文件跨平台的差异如此之小，为什么 Oracle 不直接实现这一技术呢？

也许是这一迁移的技术优先级较低，在 Oracle10gR2 中，Oracle 开始支持同字节序数据库的跨平台迁移。

这一技术实现有以下几点注意事项：

- 源平台和目的平台需要具有相同的字节序
- 重做日志文件和控制文件不会传输，迁移之后需要重建控制文件使用 RESETLOGS 方式打开数据库；
- 临时文件不会被传输；
- BFILES、外部表和 Directories、口令文件等不会被传输。

下面通过 Linux 平台到 Windows 平台的迁移来看一下这一技术的实现。

（1）确认平台及版本。首先要确定源平台和目标平台具有相同的字节序：

```
SQL> select PLATFORM_NAME, ENDIAN_FORMAT from V$TRANSPORTABLE_PLATFORM
  2  where platform_name in ('Linux IA (32-bit)','Microsoft Windows IA (32-bit)');
PLATFORM_NAME                            ENDIAN_FORMAT
---------------------------------------- --------------
Microsoft Windows IA (32-bit)            Little
Linux IA (32-bit)                        Little
```

然后需要确定源平台及目标平台的数据库版本，通常需要这两者具有相同的数据库版本，本例的情况有所不同，Linux 平台的数据库版本为 10.2.0.1，Windows 平台的数据库版本为 10.2.0.3，数据库版本不同将使情况稍微复杂一点，而且需要注意的是，通常高版本的数据库

不能向低版本迁移。

（2）确认迁移是否支持。跨平台迁移需要数据库处于 READ ONLY 模式打开，使用 DBMS_TDB.CHECK_DB 进行检查：

```
SQL> set serveroutput on
SQL> declare
  2    db_ready boolean;
  3  begin
  4     db_ready := dbms_tdb.check_db('Microsoft Windows IA (32-bit)');
  5    end;
  6  /

PL/SQL procedure successfully completed.
```

如果以上过程成功执行，并没有其他相关警告输出，则说明数据库可以支持跨平台转移。

（3）检查外部对象。使用 DBMS_TDB.CHECK_EXTERNAL 来识别外部表、Directories 或 BFILES 等，这些对象所指向的外部数据不能被 RMAN 自动转移。

```
SQL> set serveroutput on
SQL> declare
  2    external boolean;
  3  begin
  4    external := dbms_tdb.check_external;
  5  end;
  6  /
The following directories exist in the database:
SYS.DATA_PUMP_DIR
PL/SQL procedure successfully completed.
```

如果数据库中存在外部表、DIRECTORIES 等，则以上过程执行后的输出与以上类似。

（4）使用 RMAN 进行跨平台文件迁移。执行跨平台迁移首先要通过 RMAN 对数据文件进行转换，RMAN 执行过程如下：

```
[oracle@danaly eygle]$ rman target /

Recovery Manager: Release 10.2.0.1.0 - Production on Sat Jun 23 23:06:52 2007
Copyright (c) 1982, 2005, Oracle.  All rights reserved.

connected to target database: EYGLE (DBID=1445136501)

RMAN> CONVERT DATABASE NEW DATABASE 'JULIA'
2> TRANSPORT SCRIPT '/opt/oracle/oradata/transport/transport.sql'
3> TO PLATFORM 'Microsoft Windows IA (32-bit)'
4>              DB_FILE_NAME_CONVERT              '/opt/oracle/oradata/eygle/EYGLE/datafile'
```

```
'/opt/oracle/oradata/transport';

Starting convert at 23-JUN-07
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: sid=159 devtype=DISK

Directory SYS.DATA_PUMP_DIR found in the database

User SYS with SYSDBA and SYSOPER privilege found in password file
channel ORA_DISK_1: starting datafile conversion
input datafile
fno=00001 name=/opt/oracle/oradata/eygle/EYGLE/datafile/o1_mf_system_37tc1xns_.dbf
converted datafile=/opt/oracle/oradata/transport/o1_mf_system_37tc1xns_.dbf
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:15
channel ORA_DISK_1: starting datafile conversion
input datafile
fno=00002 name=/opt/oracle/oradata/eygle/EYGLE/datafile/o1_mf_undotbs1_37tc29mb_.dbf
converted datafile=/opt/oracle/oradata/transport/o1_mf_undotbs1_37tc29mb_.dbf
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:07
channel ORA_DISK_1: starting datafile conversion
input datafile
fno=00003 name=/opt/oracle/oradata/eygle/EYGLE/datafile/o1_mf_sysaux_37tc2gqc_.dbf
converted datafile=/opt/oracle/oradata/transport/o1_mf_sysaux_37tc2gqc_.dbf
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:07
channel ORA_DISK_1: starting datafile conversion
input datafile
fno=00004 name=/opt/oracle/oradata/eygle/EYGLE/datafile/o1_mf_users_37tc2tth_.dbf
converted datafile=/opt/oracle/oradata/transport/o1_mf_users_37tc2tth_.dbf
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:01
Run SQL script /opt/oracle/oradata/transport/transport.sql on the target platform to create
database
Edit init.ora file /opt/oracle/product/10.2.0/dbs/init_00illi4r_1_0.ora. This PFILE will be used
to create the database on the target platform
To recompile all PL/SQL modules, run utlirp.sql and utlrp.sql on the target platform
To change the internal database identifier, use DBNEWID Utility
Finished backup at 23-JUN-07
```

RMAN 的转换语句中指定生成一个转换脚本 transport.sql 用于参考，转换的目标平台是 Microsoft Windows IA (32-bit)，所有的数据文件转换后存放在一个新的目录下。

最后 RMAN 还自动生成一个参数文件，这个文件是 **init_00il1i4r_1_0.ora**，这个参数文件里包含了一些重要的初始化参数，可以根据需要进行相应的更改，由于平台以及路径的不同，很多涉及路径的参数都需要进行变更，这个参数文件的内容大致分为 3 个部分。

第 1 部分列出需要修改的参数：

```
# Please change the values of the following parameters:
  control_files                                                                =
"/opt/oracle/product/10.2.0/dbs/cf_D-JULIA_id-1445136501_00il1i4r"
  db_create_file_dest      = "/opt/oracle/product/10.2.0/dbs/eygle"
  db_recovery_file_dest    = "/opt/oracle/product/10.2.0/dbs/flash_recovery_area"
  db_recovery_file_dest_size= 2147483648
  background_dump_dest     = "/opt/oracle/product/10.2.0/dbs/bdump"
  user_dump_dest           = "/opt/oracle/product/10.2.0/dbs/udump"
  core_dump_dest           = "/opt/oracle/product/10.2.0/dbs/cdump"
  audit_file_dest          = "/opt/oracle/product/10.2.0/dbs/adump"
  db_name                  = "JULIA"
```

第 2 部分列出了建议 Review 的参数：

```
# Please review the values of the following parameters:
  remote_login_passwordfile= "EXCLUSIVE"
  db_domain                = ""
  dispatchers              = "(PROTOCOL=TCP) (SERVICE=eygleXDB)"
```

第 3 部分列出了来自于源数据库的一些特殊设置，这些参数可以酌情修改：

```
# The values of the following parameters are from source database:
  processes                = 150
  sga_target               = 943718400
  db_block_size            = 8192
  compatible               = "10.2.0.1.0"
  db_file_multiblock_read_count= 16
  undo_management          = "AUTO"
  undo_tablespace          = "UNDOTBS1"
  job_queue_processes      = 10
  open_cursors             = 300
  pga_aggregate_target     = 314572800
```

参数文件的内容可以在新的平台上重新创建，这个参数文件可以作为参考。

（5）转移文件到目标平台。源平台的工作完成之后，数据文件可以通过 FTP 等方式转移到目标平台，部署到相应目录，我的操作步骤如下：

```
C:\oracle\oradata>gzip -d trans.tar.gz
C:\oracle\oradata>tar -xvf trans.tar
tar: blocksize = 20
x transport/transport.sql, 2397 bytes, 5 tape blocks
```

```
x transport/o1_mf_sysaux_37tc2gqc_.dbf, 125837312 bytes, 245776 tape blocks
x transport/o1_mf_undotbs1_37tc29mb_.dbf, 209723392 bytes, 409616 tape blocks
x transport/o1_mf_system_37tc1xns_.dbf, 314580992 bytes, 614416 tape blocks
x transport/o1_mf_users_37tc2tth_.dbf, 5251072 bytes, 10256 tape blocks

C:\oracle\oradata>mkdir -p JULIA\DATAFILE
C:\oracle\oradata>mkdir JULIA\CONTROLFILE
C:\oracle\oradata>mkdir JULIA\ONLINELOG

C:\oracle\oradata>mv transport\* JULIA\DATAFILE
```

（6）创建基础环境。首先创建相关目录：

```
C:\oracle\oradata>mkdir C:\oracle\admin\julia\adump
C:\oracle\oradata>mkdir C:\oracle\admin\julia\bdump
C:\oracle\oradata>mkdir C:\oracle\admin\julia\cdump
C:\oracle\oradata>mkdir C:\oracle\admin\julia\dpdump
C:\oracle\oradata>mkdir C:\oracle\admin\julia\pfile
C:\oracle\oradata>mkdir C:\oracle\admin\julia\udump
```

创建 Windows 数据库服务：

```
C:\oracle\oradata>oradim -new -sid julia
实例已创建。
```

修改参数文件，参数文件可以从前面自动生成的参数文件进行修改得到，其中目录结构需要依据新平台的具体设置进行修改，和存储主要相关的两个参数修改如下：

```
  db_create_file_dest      = "C:\oracle\oradata"
  db_recovery_file_dest    = "C:\oracle\flash_recovery_area"
```

修改后的参数文件在 Windows 上应该位于$ORACLE_HOME/database 下。参数文件中的另外一个重要参数是控制文件路径：

```
control_files                                                              =
"/opt/oracle/product/10.2.0/dbs/cf_D-JULIA_id-1445136501_OOilli4r"
```

如果计划使用 OMF 管理，可以暂时注释这一参数，在创建控制文件后再将控制文件的名称路径追加到参数文件中。

（7）迁移步骤。准备工作完成之后，就可以进行新平台的数据库加载等工作，这些工作还可以参考在源平台生成的 transport.sql 脚本。

这个脚本的第一部分给出了使用参数文件启动实例及重新创建控制文件的语法参考，当然还需要修改才能使用这段脚本：

```
STARTUP NOMOUNT PFILE='/opt/oracle/product/10.2.0/dbs/init_OOilli4r_1_0.ora'
CREATE CONTROLFILE REUSE SET DATABASE "LINDB10G" RESETLOGS  NOARCHIVELOG
    MAXLOGFILES 16
    MAXLOGMEMBERS 3
    MAXDATAFILES 100
```

```
    MAXINSTANCES 8
    MAXLOGHISTORY 292
LOGFILE
  GROUP 1 SIZE 50M,
  GROUP 2 SIZE 50M,
  GROUP 3 SIZE 50M
DATAFILE
  '/opt/oracle/oradata/transport/o1_mf_system_37tc1xns_.dbf',
  '/opt/oracle/oradata/transport/o1_mf_undotbs1_37tc29mb_.dbf',
  '/opt/oracle/oradata/transport/o1_mf_sysaux_37tc2gqc_.dbf',
  '/opt/oracle/oradata/transport/o1_mf_users_37tc2tth_.dbf'
CHARACTER SET ZHS16GBK
;
```

由于已经编辑好了新的参数文件，所以可以使用这个参数文件启动实例：

```
C:\oracle\oradata>set ORACLE_SID=julia
C:\oracle\oradata>sqlplus "/ as sysdba"

SQL*Plus: Release 10.2.0.3.0 - Production on 星期一 6 月 25 09:45:59 2007
Copyright (c) 1982, 2006, Oracle.  All Rights Reserved.
已连接到空闲例程。
SQL> startup nomount pfile=?\database\initjulia.ora
ORACLE 例程已经启动。

Total System Global Area  943718400 bytes
Fixed Size                  1293960 bytes
Variable Size             239075704 bytes
Database Buffers          700448768 bytes
Redo Buffers                2899968 bytes
```

接下来创建控制文件：

```
SQL> CREATE CONTROLFILE REUSE SET DATABASE "JULIA" RESETLOGS  NOARCHIVELOG
  2      MAXLOGFILES 16
  3      MAXLOGMEMBERS 3
  4      MAXDATAFILES 100
  5      MAXINSTANCES 8
  6      MAXLOGHISTORY 292
  7  LOGFILE
  8    GROUP 1 SIZE 10M,
  9    GROUP 2 SIZE 10M,
 10    GROUP 3 SIZE 10M
```

```
 11  DATAFILE
 12    'C:\oracle\oradata\JULIA\DATAFILE\o1_mf_system_37tc1xns_.dbf',
 13    'C:\oracle\oradata\JULIA\DATAFILE\o1_mf_undotbs1_37tc29mb_.dbf',
 14    'C:\oracle\oradata\JULIA\DATAFILE\o1_mf_sysaux_37tc2gqc_.dbf',
 15    'C:\oracle\oradata\JULIA\DATAFILE\o1_mf_users_37tc2tth_.dbf'
 16  CHARACTER SET ZHS16GBK
 17  ;

控制文件已创建。
```

然后将控制文件的名称等信息追加到参数文件中：

```
SQL> column ctl_files NEW_VALUE ctl_files;
SQL> SELECT CONCAT ('control_files=''',
  2                 CONCAT (REPLACE (VALUE, ', ', ''',''), '''')
  3                ) ctl_files
  4  FROM v$parameter WHERE NAME = 'control_files';

CTL_FILES
--------------------------------------------------------------------------------
control_files='C:\ORACLE\ORADATA\JULIA\CONTROLFILE\O1_MF_37Y7SZ9R_.CTL','C:\ORAC
LE\FLASH_RECOVERY_AREA\JULIA\CONTROLFILE\O1_MF_37Y7SZMK_.CTL'

SQL> host "echo &ctl_files >>C:\oracle\10.2.0\database\initjulia.ora";
```

> 注意：执行完以上命令后，需要检查参数文件的格式，如果控制文件名称未正确添加，可以手工调整一下。

完成了以上工作后，可以关闭数据库，再次启动数据库到 mount 状态，这时候新的控制文件已经发挥作用：

```
SQL> shutdown immediate;
ORA-01109: ??????

已经卸载数据库。
ORACLE 例程已经关闭。
SQL> startup mount;
ORACLE 例程已经启动。

Total System Global Area  943718400 bytes
Fixed Size                  1293960 bytes
Variable Size             239075704 bytes
Database Buffers          700448768 bytes
```

```
Redo Buffers                2899968 bytes
数据库装载完毕。
```

（8）完成数据库恢复。接下来再参考一下 transport.sql 中的推荐步骤：

```
ALTER DATABASE OPEN RESETLOGS;

-- Commands to add tempfiles to temporary tablespaces.
-- Online tempfiles have complete space information.
-- Other tempfiles may require adjustment.
ALTER TABLESPACE TEMP ADD TEMPFILE
     SIZE 20971520  AUTOEXTEND ON NEXT 655360  MAXSIZE 32767M;
-- End of tempfile additions.
```

现在需要通过 RESETLOGS 方式来重新生成日志文件，然后手工添加临时文件。

注意在迁移过程中如果两个平台的数据库版本完全一致，则以上步骤可以顺利执行，参考 transport.sql 可以顺利完成迁移。而本例的测试平台由于 Linux 平台的数据库版本为 10.2.0.1，Windows 平台版本为 10.2.0.3，所以实际操作中还会有所不同，在执行 RESETLOGS 过程中，数据库会发生中断：

```
SQL> alter database open resetlogs;
alter database open resetlogs
*
第 1 行出现错误:
ORA-01092: ORACLE 实例终止。强制断开连接
```

检查日志发现以下提示：

```
Mon Jun 25 10:03:19 2007
Errors in file c:\oracle\admin\julia\udump\julia_ora_3596.trc:
ORA-00704: 引导程序进程失败
ORA-39700: 必须用 UPGRADE 选项打开数据库
```

Oracle 要求以 UPGRADE 选项打开数据库，对数据库执行跨版本迁移。

继续参考 transport.sql 的最后部分：

```
SHUTDOWN IMMEDIATE
STARTUP UPGRADE PFILE='/opt/oracle/product/10.2.0/dbs/init_00ill1i4r_1_0.ora'
@@ ?/rdbms/admin/utlirp.sql
SHUTDOWN IMMEDIATE
STARTUP PFILE='/opt/oracle/product/10.2.0/dbs/init_00ill1i4r_1_0.ora'
-- The following step will recompile all PL/SQL modules.
-- It may take serveral hours to complete.
@@ ?/rdbms/admin/utlrp.sql
set feedback 6;
```

再次启动数据库到 UPGRADE 模式，由于之前的数据库中断，现在这些需要进行一点恢复工作：

```
SQL> startup upgrade;
ORACLE 例程已经启动。

Total System Global Area  943718400 bytes
Fixed Size                  1293960 bytes
Variable Size             239075704 bytes
Database Buffers          700448768 bytes
Redo Buffers                2899968 bytes
数据库装载完毕。
ORA-01113: 文件 1 需要介质恢复
ORA-01110: 数据文件 1:'C:\ORACLE\ORADATA\JULIA\DATAFILE\O1_MF_SYSTEM_37TC1XNS_.DBF'
SQL> recover database;
完成介质恢复。
```

恢复完成之后启动数据库到 UPGRADE 模式：

```
SQL> shutdown immediate;
ORA-01109: 数据库未打开

已经卸载数据库。
ORACLE 例程已经关闭。
SQL> startup upgrade;
ORACLE 例程已经启动。

Total System Global Area  943718400 bytes
Fixed Size                  1293960 bytes
Variable Size             239075704 bytes
Database Buffers          700448768 bytes
Redo Buffers                2899968 bytes
数据库装载完毕。
数据库已经打开。
```

执行脚本?/rdbms/admin/utlirp.sql，这个脚本执行完成之后会有如下提示：

```
DOC>#####################################################################
DOC>   utlirp.sql completed successfully. All PL/SQL objects in the
DOC>   database have been invalidated.
DOC>
DOC>   Shut down and restart the database in normal mode and run utlrp.sql to
DOC>   recompile invalid objects.
DOC>#####################################################################
```

也就是说，这个脚本的作用是使数据库中的 PL/SQL 对象 INVALID，然后通过 utlrp.sql 的重新编译，消除跨平台的兼容性影响。

按照 transport.sql 脚本提示的步骤，我们可以重新启动数据库来执行 utlrp.sql 脚本（由于本例涉及到版本迁移，需要再次启动数据库到 upgrade 模式，如果数据库版本相同，则可以直接启动数据库，执行 utlrp.sql 脚本完成最后的编译工作）：

```
SQL> @@ ?/rdbms/admin/utlrp.sql
TIMESTAMP
--------------------------------------------------------------------------------
COMP_TIMESTAMP UTLRP_BGN  2007-06-25 10:27:57
.....
PL/SQL 过程已成功完成。
TIMESTAMP
--------------------------------------------------------------------------------
COMP_TIMESTAMP UTLRP_END  2007-06-25 10:39:43
PL/SQL 过程已成功完成。
```

utlrp.sql 执行完成之后，需要再执行和数据库升级相关的脚本，这个脚本是 catupgrd.sql：

```
SQL> @?/rdbms/admin/catupgrd.sql
```

这个脚本调用 catlog.sql 和 catproc.sql 来重建字典对象等，在执行完这个脚本之后，可以关闭数据库后，正常打开数据库：

```
SQL> startup
ORACLE 例程已经启动。

Total System Global Area  943718400 bytes
Fixed Size                  1293960 bytes
Variable Size             239075704 bytes
Database Buffers          700448768 bytes
Redo Buffers                2899968 bytes
数据库装载完毕。
数据库已经打开。
SQL> select count(*) from dba_objects where status='INVALID';
  COUNT(*)
----------
        86
已选择 1 行。
SQL> @?\rdbms\admin\utlrp.sql
```

catupgrd.sql 脚本可能会使部分字典对象失效，可以再次运行 utlrp.sql 脚本来进行编译，编译完成后，不要忘记为数据库添加临时文件：

```
SQL> ALTER TABLESPACE TEMP ADD TEMPFILE
  2       SIZE 20971520  AUTOEXTEND ON NEXT 655360  MAXSIZE 32767M;

表空间已更改。
```

至此同字节序的跨平台迁移全部完成，当然由于版本的不同，整个过程稍微复杂了一些，不过这个过程对于跨平台的迁移及版本升级是一个很好的参考。

### 7. 实现数据迁移的高可用性

通过以上测试实际上可以确认，对于可传输表空间，可以很容易从 Oracle 9i 向 Oracle 10g 迁移。那么这种方法对于可用性要求极高的环境进行数据迁移或数据库迁移具有极大的便利。

如果进行数据库升级，通常的方法是通过 DBUA（Database Upgrade Assistant，Oracle 10g 引入的新工具）进行，但是 DBUA 存在的问题在于，操作过程过长，而且如果升级过程中出现问题，数据文件可能不能重新被使用，这就需要从备份中进行恢复，这使得业务连续性要求高的企业很难采用这种方法进行升级。

另外一种常见的迁移的方法是通过逻辑导出导入（EXP/IMP），但是这种方法对于不断变化的数据无能为力，所以通常也不可行。

那么现在，可传输表空间就成了一个可以考虑的快速迁移或升级方法。

Oracle 有一个小组，专注于设计高可用性架构的实现，以帮助用户最大限度的提高系统可用性，Oracle 有一个专有名词用来命名这类技术 MAA（Maximum Availability Architecture）。OTN 上 MAA 部分有一个 Amadeus 公司的实践案例，通过可传输表空间从 Oracle 9i 向 Oracle 10g 实现快速数据迁移。

当然这种方法的使用要考虑的还有很多，通过各种技术和方法的结合使用才能最终的达到快速迁移的目标。

Amadeus 公司的迁移是在同类型平台不同主机之间进行的，其实现步骤大致如下：

（1） 在升级主机安装 Oracle 9i 版本，并创建生产库的 DataGuard 数据库，这个工作可以在线进行，不影响主节点的工作。

（2） 在升级主机安装 Oracle 10gR2 数据库软件，创建数据库；此时升级主机上存在了 2 个数据库。

（3） 整理不能通过 transport tablespace 处理的内容，如 sequence、synonyms、grants 等。

（4） 在升级割接时间，将主库置为只读，将日志全部应用到备机，业务影响从此时开始。

（5） 将备机的数据文件通过可传输表空间迁移至 Oracle 10gR2 数据库，并创建 sequencee、synonyms、grants 等对象，检查验证。

（6） 如果没有问题，则即可将业务切换至新的 Oracle 10gR2 数据库运行，业务恢复正常运行。

在这个迁移过程中，如果迁移失败，那么直接读写打开主库即可恢复业务的正常运行，回退非常方便。

使用这种方法，业务影响仅发生在以上（4）～（6）步，在 OTN 的案例中，Amadeus 公司在实际操作中，10 分钟之内就将一个大型数据库迁移到 Oracle 10gR2，这种方式是一种非常有新意的创新性应用。在熟悉了 Oracle 的各项技术之后，通过不断实践和探索，我们就能够不断发现充满价值的 Oracle 应用。

### 1.3.7 最后的脚本

以上脚本已经完成了主要的工作，剩下的是一些最后的维护工作。

这里还有两个脚本需要执行，首先执行的是 postScripts.sql 脚本，这个脚本主要对部分用户及部分数据库选件进行维护：

```
connect "SYS"/"&&sysPassword" as SYSDBA
set echo on
spool C:\oracle\admin\eygle\scripts\postScripts.log
@C:\oracle\10.2.0\rdbms\admin\dbmssml.sql;
execute dbms_datapump_utl.replace_default_dir;
commit;
connect "SYS"/"&&sysPassword" as SYSDBA
alter session set current_schema=ORDSYS;
@C:\oracle\10.2.0\ord\im\admin\ordlib.sql;
alter session set current_schema=SYS;
connect "SYS"/"&&sysPassword" as SYSDBA
connect "SYS"/"&&sysPassword" as SYSDBA
alter user CTXSYS account unlock identified by change_on_install;
connect "CTXSYS"/"change_on_install"
@C:\oracle\10.2.0\ctx\admin\defaults\dr0defdp.sql;
@C:\oracle\10.2.0\ctx\admin\defaults\dr0defin.sql "SIMPLIFIED CHINESE";
connect "SYS"/"&&sysPassword" as SYSDBA
execute dbms_swrf_internal.cleanup_database(cleanup_local => FALSE);
commit;
spool off
```

最后执行的脚本是 postDBCreation.sql，在这个脚本中将创建 spfile，解锁 SYSMAN、DBSNMP 用户，编译失效对象并配置 DB Control：

```
connect "SYS"/"&&sysPassword" as SYSDBA
set echo on
spool C:\oracle\admin\eygle\scripts\postDBCreation.log
connect "SYS"/"&&sysPassword" as SYSDBA
set echo on
create spfile='C:\oracle\10.2.0/dbs/spfileeygle.ora'
FROM pfile='C:\oracle\admin\eygle\scripts\init.ora';
shutdown immediate;
connect "SYS"/"&&sysPassword" as SYSDBA
startup ;
alter user SYSMAN identified by "&&sysmanPassword" account unlock;
alter user DBSNMP identified by "&&dbsnmpPassword" account unlock;
```

```
select 'utl_recomp_begin: ' || to_char(sysdate, 'HH:MI:SS') from dual;
execute utl_recomp.recomp_serial();
select 'utl_recomp_end: ' || to_char(sysdate, 'HH:MI:SS') from dual;
host C:\oracle\10.2.0\bin\emca.bat -config dbcontrol db -silent -DB_UNIQUE_NAME eygle -PORT
1521 -EM_HOME C:\oracle\10.2.0 -LISTENER LISTENER -SERVICE_NAME eygle -SYS_PWD &&sysPassword
-SID eygle -ORACLE_HOME C:\oracle\10.2.0 -DBSNMP_PWD &&dbsnmpPassword -HOST gqgai -LISTENER_OH
C:\oracle\10.2.0    -LOG_FILE    C:\oracle\admin\eygle\scripts\emConfig.log    -SYSMAN_PWD
&&sysmanPassword;
spool C:\oracle\admin\eygle\scripts\postDBCreation.log
exit;
```

看到在最后部分，通过 emca.bat 批处理文件，配置了 DB Control，这里通过一条完整的命令快速地完成了 DB Control 的创建等工作，也可以通过手工方式对 DB Control 进行维护，关于这部分内容请参考“第 2 章 从 OEM 到 iSQL*Plus”的内容。

此外需要注意的是以下几句命令：

```
select 'utl_recomp_begin: ' || to_char(sysdate, 'HH:MI:SS') from dual;
execute utl_recomp.recomp_serial();
select 'utl_recomp_end: ' || to_char(sysdate, 'HH:MI:SS') from dual;
```

在 Oracle 9i 的 postDBCreation.sql 的脚本中，这部分的内容如下：

```
@/opt/oracle/product/9.2.0/rdbms/admin/utlrp.sql;
```

其实两者是相同的，utlrp.sql 中主体部分和 Oracle 10g 中是相同的：

```
@@utlrcmp.sql
execute utl_recomp.recomp_serial();

Rem ================================================================
Rem Run component validation procedure
Rem ================================================================

EXECUTE dbms_registry.validate_components;
```

Oracle 在 utlrp.sql 脚本的注释中说得很明确：这是一个通用脚本，可以在任意时候运行以重新编译数据库失效对象。

通常我们会在 Oracle 的升级指导中看到这个脚本，Oracle 强烈推荐在 migration / upgrade / downgrade 之后，通过运行此脚本编译失效对象。但是注意，此脚本需要用 SQL*Plus 以 SYSDBA 身份运行，并且当时数据库中最好不要有活动事物或 DDL 操作，否则极容易导致死锁的出现。

这样使用模板创建数据库就完成了。