**前言**

这次，我们将在Linux下来动手完成Oracle数据库的安装与使用。

Oracle本身是可以免费下载的包括 它的企业版以及被它收购的Weblogic和Sun中的几乎任何东西你都可以拿来下载和使用，不像IBM和Tibco一些其它厂商，只有“试用版”给你下载，Oracle的东西没有时间限制，你拿来做练习，搭实验环境都是没有任何的问题的。

但是，如果你出了问题，需要用到Oracle的补丁或者是Oracle的技术支持，这就开始收费。

Oracle就是这种“卖Service”的模式。

还有就是你安装了Oracle后，你的应用是给另一家企业用的或者是带有商业用途，那对不起Oracle也会问你来收费。

在linux下安装oracle是一件令人生畏的事情，其复杂程度远远超过安装linux操作系统本身。如果能够进行成功的安装oracle,那么同时也就顺便掌握了linux一些技术。

本文介绍在redhat linux 下安装oracle 10g 的方法。在这里说明一下，Oracle 10g的g是grid 的缩写，意为网格，目前较为前沿的网络计算技术。

**一、安装Oracle前的准备**



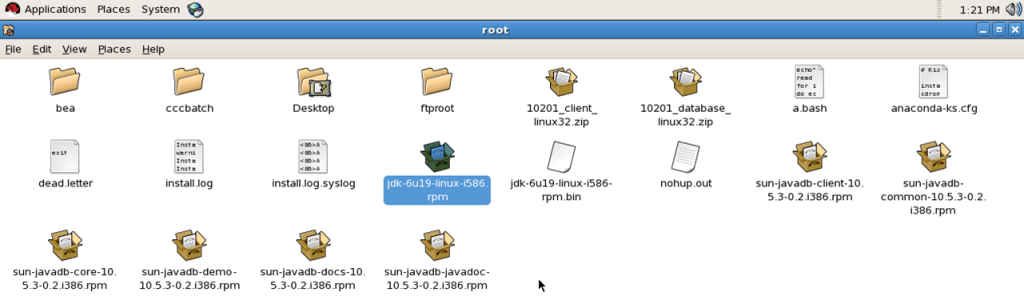
这次我们将在Linux环境下安装Oracle，对Oracle支持最好的莫过于SuSe Linux，但是随着后来RedHat走向了商业化后，RedHat与Oracle公司开始形成一种密切的关系，因此如果你手上有RedHat As 5.5x及以下版本或者是Fedora14及以上版本话那是最好不过了。

Oracle下载地址：[进入下载](http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html)

**确保你是用的是root帐号，执行下列步骤**

**1.1在Linux上先安装相应的JDK**

下载jdk1.6 for Linux，请注意32位与64位的区分，需要和你的操作系统对应上哦！



打开一个Terminal窗口



进入到你的jdk下载的目录下并输入：

**./jdk-6u19-linux-i586-rpm.bin**

我们默认将jdk安装于“**/usr/java/jdk1.6.0\_19**”目录吧。

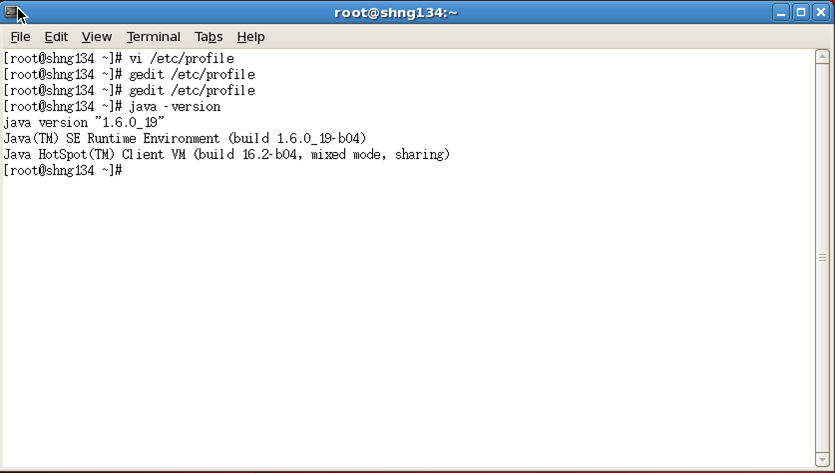
接下来我们需要修改系统环境变量，在terminal窗口中键入“**vi  /etc/profile**”

在这个profile文件内加入两行：

***export JAVA\_HOME=/usr/java/jdk1.6.0\_19  
export PATH=$JAVA\_HOME/bin:$JAVA\_HOME/jre/bin:$PATH***

**1.2测试我们的jdk安装是否正确**

打开一个terminal窗口并输入



看到正确的jdk版本信息后即代表我们的jdk在linux下安装正确了

**1.3Oracle安装前的环境变量配置**

还是编辑那个profile文件并加入如下几行：

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. export JAVA\_HOME=/usr/java/jdk1.6.0\_19<br />
2. export PATH=$PATH:$JAVA\_HOME/bin:$JAVA\_HOME/jre/bin<br />
3. export ORACLE\_BASE=/opt/oracle<br />
4. export ORACLE\_HOME=$ORACLE\_BASE/product/10<br />
5. export ORACLE\_SID=ktdb<br />
6. export ORACLE\_TERM=xterm<br />
7. export NLS\_LANG=AMERICAN\_AMERICA.UTF8<br />
8. export  LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:$ORACLE\_HOME/lib:/lib:/usr/lib:/usr/local/lib<br />

export JAVA\_HOME=/usr/java/jdk1.6.0\_19<br />

export PATH=$PATH:$JAVA\_HOME/bin:$JAVA\_HOME/jre/bin<br />

export ORACLE\_BASE=/opt/oracle<br />

export ORACLE\_HOME=$ORACLE\_BASE/product/10<br />

export ORACLE\_SID=ktdb<br />

export ORACLE\_TERM=xterm<br />

export NLS\_LANG=AMERICAN\_AMERICA.UTF8<br />

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:$ORACLE\_HOME/lib:/lib:/usr/lib:/usr/local/lib<br />

通过上述环境变量我们可以得知，我们的oracle装安装在**/opt/oracle/product/10**目录下，所以

* **我们的ORACLE\_BASE为: /opt/oracle**
* **我们的ORACLE\_HOME即为: /opt/oracle/product/10了。**

这些变量是Oracle在安装过程中需要读取的，在windows下安装oracle是不需要设这些东西的。

**1.4 创建Oracle安装时使用的用户与组**

打开一个Terminal窗口输入下列命令

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. groupadd oinstall
2. groupadd dba
3. useradd -g oinstall -G dba oracle
4. passwd oracle

groupadd oinstall

groupadd dba

useradd -g oinstall -G dba oracle

passwd oracle

**1.5创建Oracle的安装路径并将此路径的读写权限赋予Oracle这个“组”**

我们安装Oracle时一般是不会用root用户的，因此我们需要增加一个角色（group），并在这个角色中增加一个用户（oracle）然后用这个用户登录我们的Linux并且执行安装。

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. mkdir -p /opt/oracle/ product/10
2. chown -R oracle.oinstall /opt/oracle/

mkdir -p /opt/oracle/ product/10

chown -R oracle.oinstall /opt/oracle/

**1.6 在profile中设置图形显示参数**

打开一个terminal窗口，然后编辑/etc/profile，在其中加入

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. xhost +

xhost +

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. DISPLAY=<machine-name>:0.0; export DISPLAY

DISPLAY=<machine-name>:0.0; export DISPLAY

它代表使得所有的用户可以使用图形化界面来运行相关的图形化应用程序，因为Linux的安全机制相当的严格，root是最高权限，除去root以外的其它用户如果需要拥有root才能执行的权限就需要授予权

**1.7 在Linux系统中安装Oracle安装时需要的系统lib库**

一般来讲，Oracle主要需要下面的这些Lib库

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. gcc-3.2.3-2
2. make-3.79
3. binutils-2.11
4. openmotif-2.2.2-16
5. setarch-1.3-1
6. compat-gcc-7.3-2.96.122
7. compat-gcc-c++-7.3-2.96.122
8. compat-libstdc++-7.3-2.96.122
9. compat-libstdc++-devel-7.3-2.96.122

gcc-3.2.3-2

make-3.79

binutils-2.11

openmotif-2.2.2-16

setarch-1.3-1

compat-gcc-7.3-2.96.122

compat-gcc-c++-7.3-2.96.122

compat-libstdc++-7.3-2.96.122

compat-libstdc++-devel-7.3-2.96.122

但是。。。Linux下的Lib库也是有依赖关系的，我给初学都的建议是你可以在刚开始安装Linux时就选customer install，然后选中相应的lib库，除去open-jdk（这个不能装，装完后sun的jdk就不起作用了，你到时还要卸），把dev相关的lib, gcc相关的lib都选上，还有gnome相关，KDE相关的lib库，有时全选上后回过头来要去安装的package里手工check，有没有java相关的被安装了，如果安装了就一定要把安装项前的勾选项，去掉,一定不能让Linux安装自带的open jdk。

**二、开始在Linux下安装Oracle**

Oracle10g在Linux下有版本检查的限制，如果你的Linux RedHat的版本低于5，那么你可以在Oracle的安装盘disk1下，直接运行如下命令调出图形化安装界面

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. ./runInstaller

./runInstaller

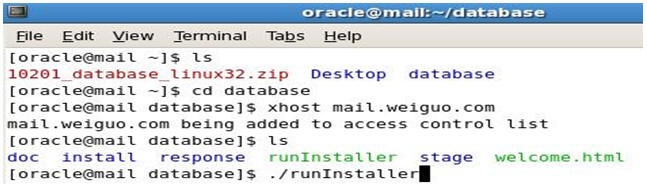
如果你的Linux的版本是AS5.5及以上（目前最新的Linux），那么请你使用下面这条命令来安装Oracle

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

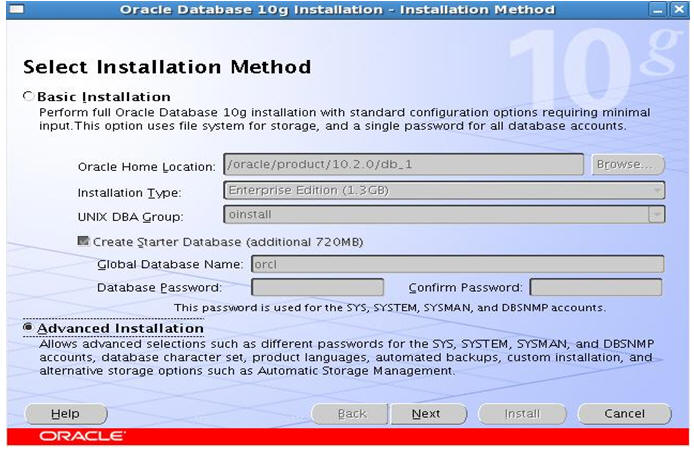
1. /runInstaller –ignoreSysPrereqs

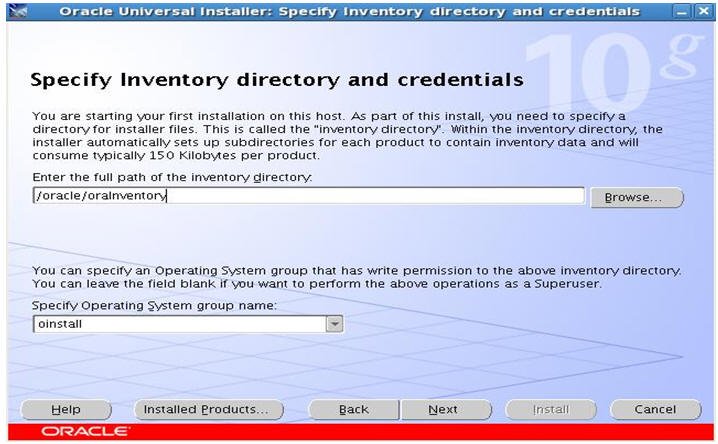
/runInstaller –ignoreSysPrereqs

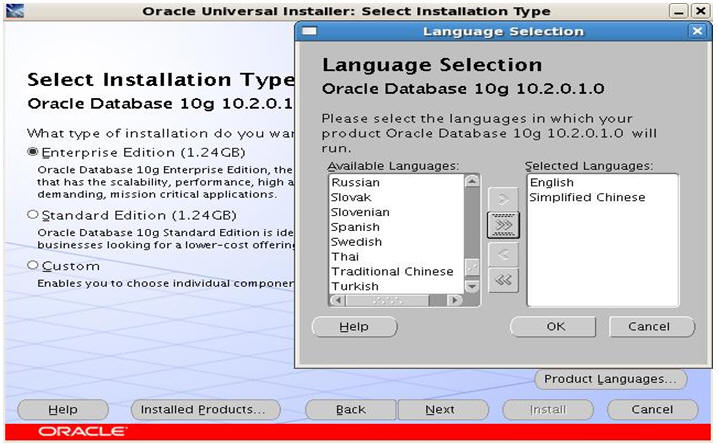
这条命令将跳过Oracle安装对于Linux系统内核的检查。



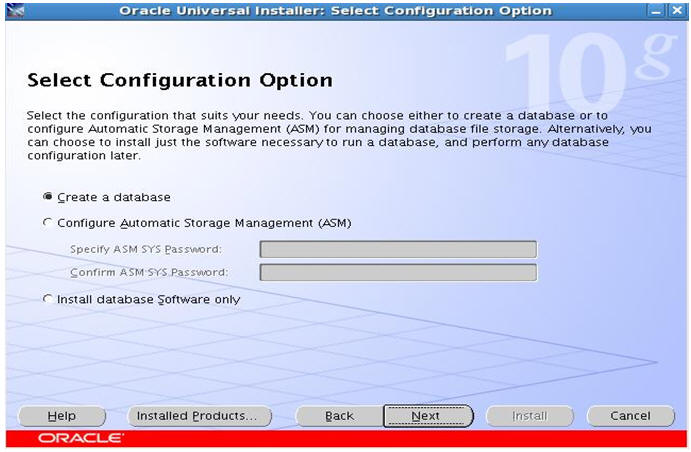
主安装界面出现

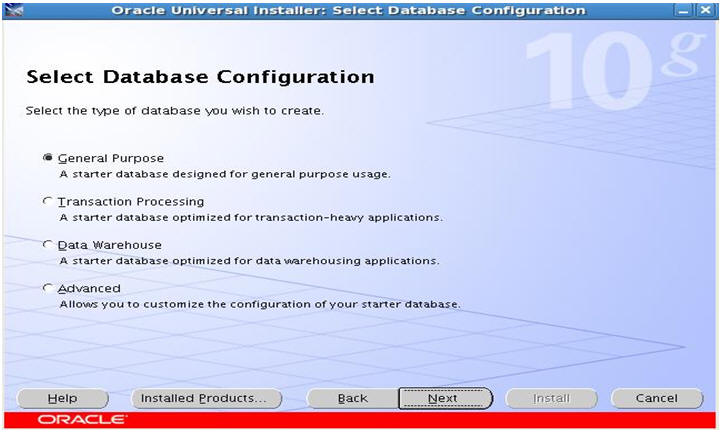


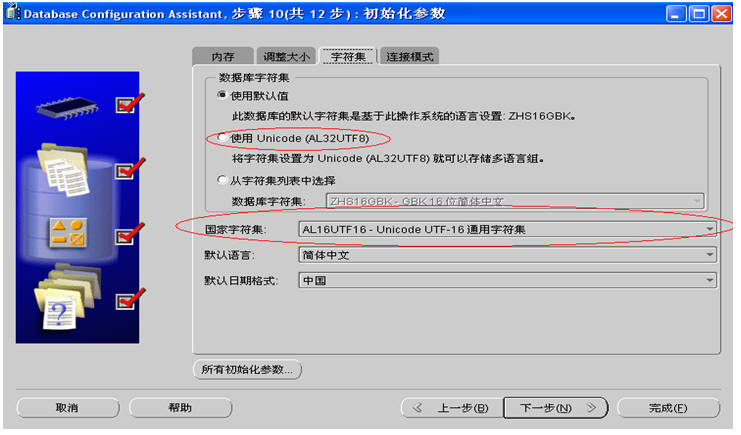


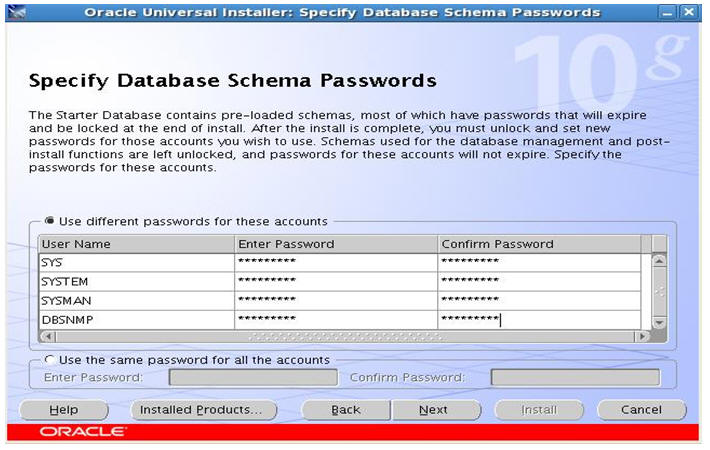


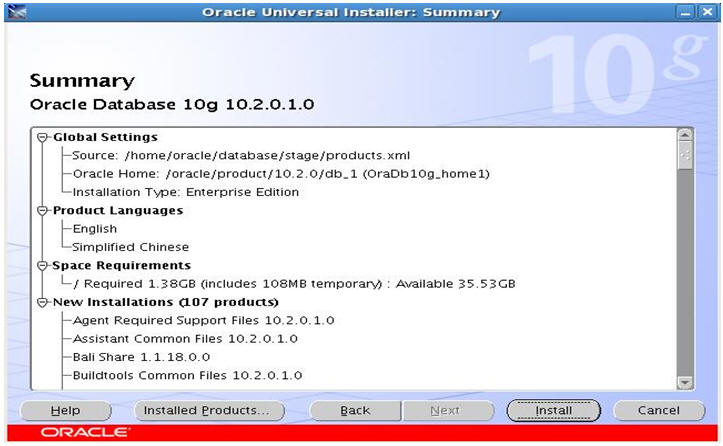


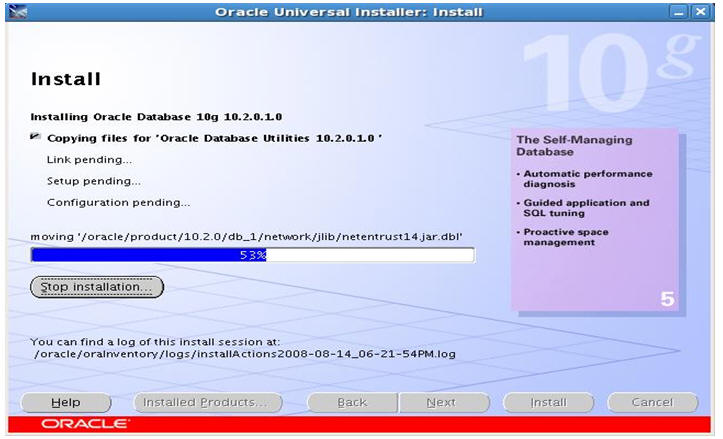


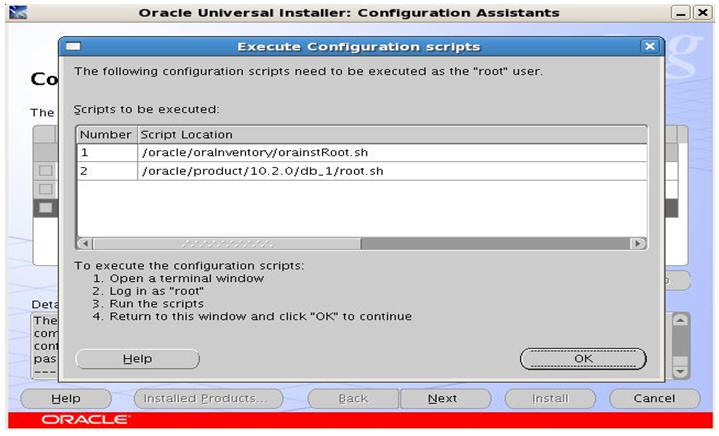


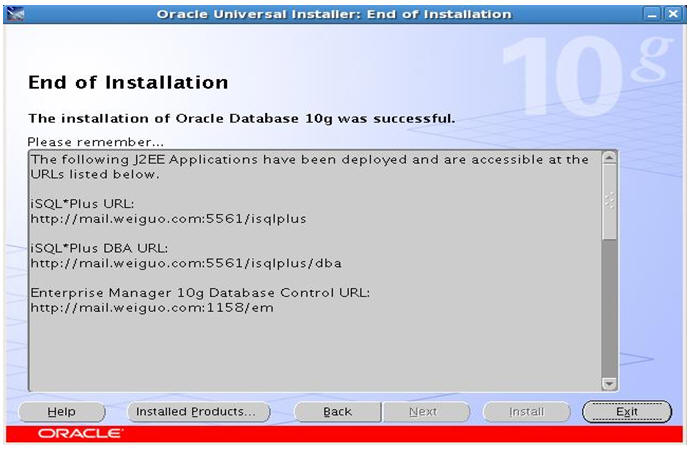












按照上述步骤一步步把Oracle装上，注意安装时字符集永远选用AL32UTF-8，这样你的Oracle才能支持多语言。

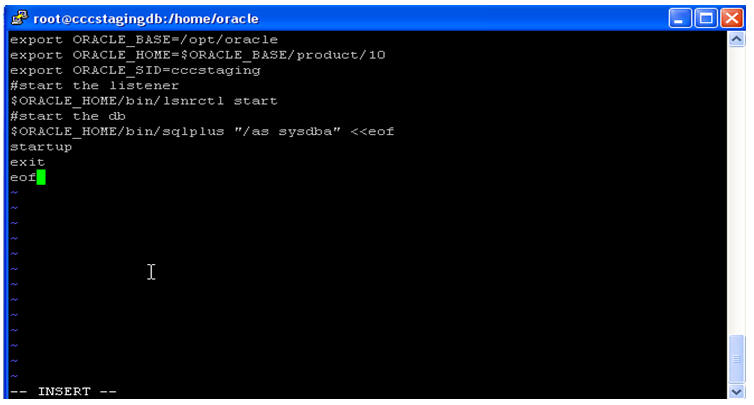
**三、Oracle安装后的一些设置**

**3.1 设置Oracle的服务开机自动启动**

编辑 /etc/rc.local/文件

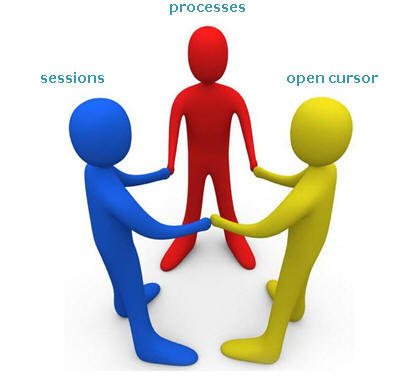


以下是dbstart.sh文件的内容



存盘后每次你重启Linux，Oracle就会随着你的Linux的开机而自动运行起来了（还有更专业的设置，将在以后的教程中传授,对于初学者想自己搭个环境的选用这个，嘿嘿）。

**3.2 设置Oracle的processes, session, Maximum Open Cursor**



其们在使用sys用户连上Oracle后可以使用这条命令来显示这三个兄弟

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. show parameter processes;

show parameter processes;

一般安装好后，这个processes默认为150，网上有很多人说如果碰到用户的session不够就去用alter命令改这个session number，其实是不对的。

session与processes是绑定的，用下面的公式：

**sessions=1.1\*processes + 5**

所以你只有改这个processes，session才会自动调整，我们可以使用下面这条命令去改变系统中的processes

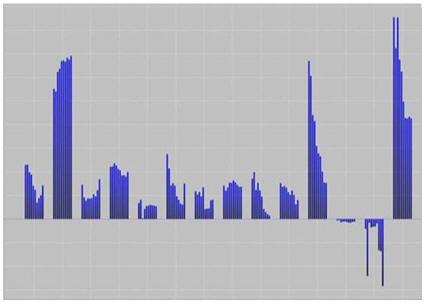
**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. alter system set processes=500 scope = spfile;

alter system set processes=500 scope = spfile;

这个processes不是乱设的，是要和你系统的内核设置去绑定的。

**四、Oracle的性能调优**



* Client Configuration
* User Role Privilege
* SGA
* Table Space
* Import
* Oracle And OS Kernel
* Oracle Under 32 bits OS
* SQL Plan
* Table Analyze
* Partition Table
* Performance Monitor

我们的Oracle性能调优主要用围绕上述几个章节来做介绍，我们不介绍太高深的莫明奇妙的理论，在这边我们对这几个方面做一个统用的解释和实际应用场景，如果是新手，你在看过这篇教程后应该知道一个oracle的性能主要从几个方面（Common）去着手，对一个熟手来说上述每个小节的具体内容都是可以在Oracle的DBA手册中找到更详细的内容。

**4.1 客户端的配置：TNS（Client Configuration: TNS）**

**配置主机名：**

Oracle的连接服务是基于主机名的，我们需要设置装Oracle这台主机的唯一主机名，这样客户端才能通过TNS来连上Oracle的服务

请更改Oracle所在服务器上的 /etc/hosts文件

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. # Do not remove the following line, or various programs
3. # that require network functionality will fail.

6. 192.168.1.3 myoracleserver
8. 127.0.0.1   localhost.localdomain localhost

# Do not remove the following line, or various programs

# that require network functionality will fail.

192.168.1.3 myoracleserver

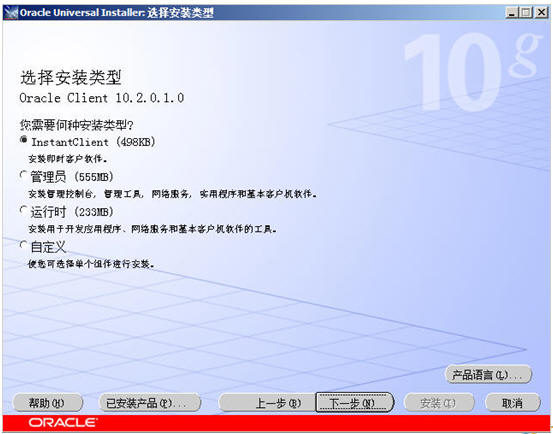
127.0.0.1 localhost.localdomain localhost

这边的192.168.1.3就是我们的主机ip，后面的myoracleserver就是主机名。

**配置客户端连接**

一个客户端如果需要连接Oracle必须安装Oracle Client，可以去Oracle网站上下载[进入下载](http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html)。

Oracle11g开始后不再提供图形化的Oracle客户端下载，但尽管是这样你也必须在客户机上尤其是那些支持TNS连接的第三方客户端时都是必须安装Oracle客户端的。

、

我们选择“管理员"模式进行安装。

安装完后，你会在你的”启动“菜单里找到这样的一个菜单项





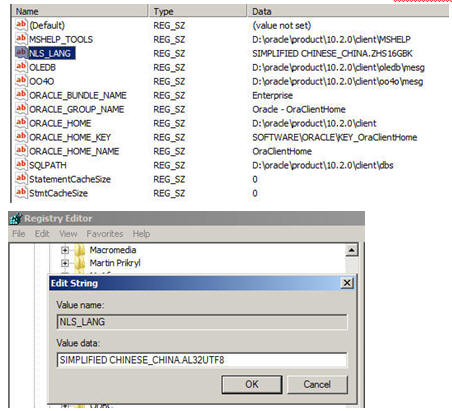






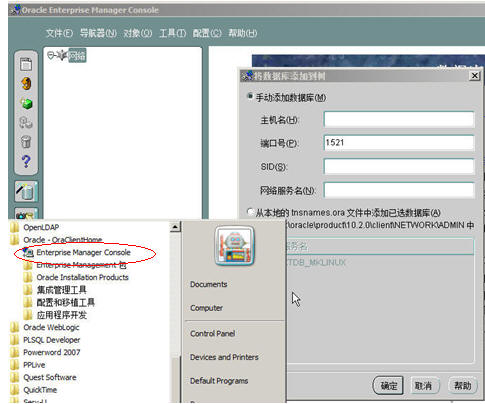
**更改客户端的字体使其可以支持中文的正常显示**

编辑注册表

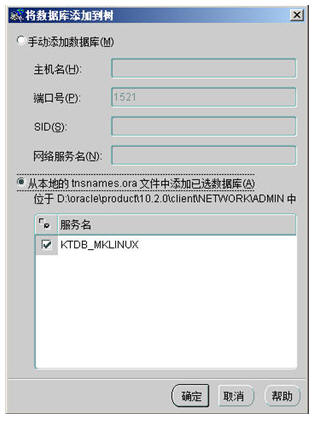


把这个NLS\_LANG改成上图中所示，这样你的Oracle的客户端上可以直接通过客户端工具或者是第三方客户端工具进行中文的正常录入了而不是乱码了。

**4.2 在Oracle中建立帐号与分配权限**



使用system帐号登录

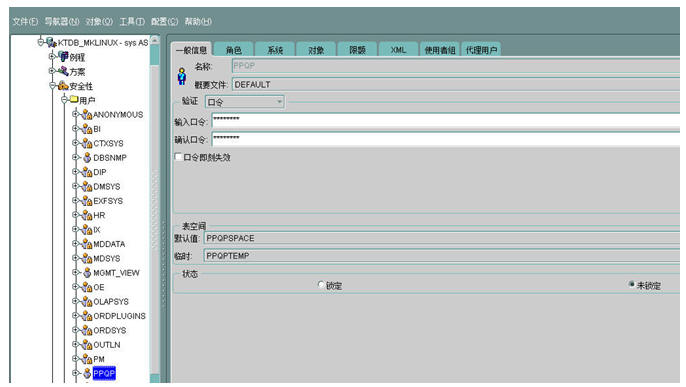


**创建用户之前先要创建表空间**

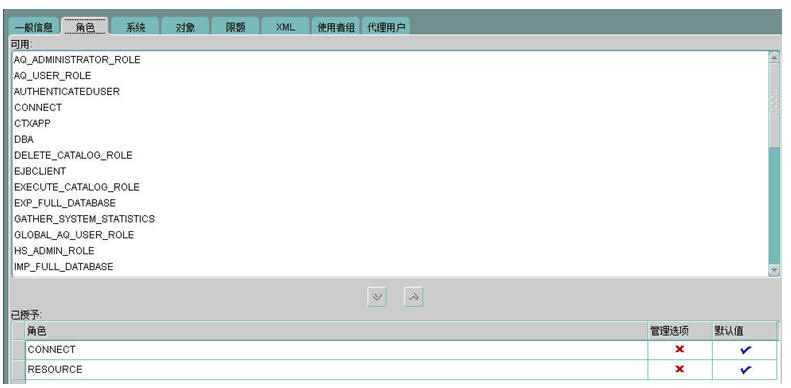


一般我们创建一个表空间和一个临时表空间

然后我们就可以开始创建用户了



**分配角色**



对于一个j2ee的应用连接Oracle来说，这个连接用户的角色只需要具有上述两个权限就够用了，有些人喜欢给这个用户DBA权限是太绝对了，当然这样给角色使用起来方便，什么权限都有了，不需要我在开发时再进行细化的设置了。

可是，你有没有想到过，一旦你的应用被人sql injection后，而你的应用上连接着oracle的用户是dba权限，那么你连整个数据库是不是都会被泄密啊？

**分配系统角色**

分配完角色后不是说等于结束了，还有一个”系统“角色要你分配，给下图所列的一项就够用了。



**设置Oracle连接用户最大可登录次数**

一个Oracle内的用户默认可以让你重复登录10次，10次的限制一到，这个用户就会被锁住。

可以用下列语句查看一个Oracle用户重复几次后会被锁住的设置参数

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. SELECT \* FROM dba\_profiles s WHERE s.profile='DEFAULT' AND resource\_name='FAILED\_LOGIN\_ATTEMPTS';

SELECT \* FROM dba\_profiles s WHERE s.profile='DEFAULT' AND resource\_name='FAILED\_LOGIN\_ATTEMPTS';

当然这样做是为了安全考虑。

可是，在我们的开发环境中，有时往往因为一个循环，一个登录器，一个线程写错而导致Oracle连接用户重复需要连接Oracle 数次，因此经常开发人员会向DBA抱怨说 “我的用户又被锁住了”。

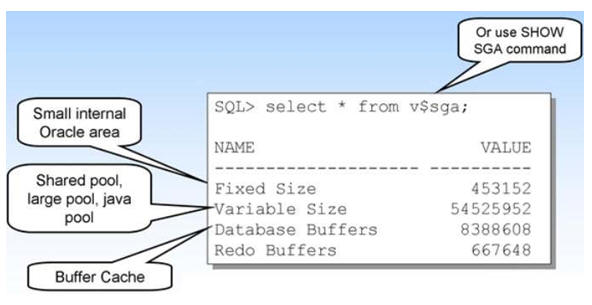
因此，在开发环境我建议你把这个“重复连接次数”去掉，设成下面这样

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. ALTER PROFILE DEFAULT LIMIT FAILED\_LOGIN\_ATTEMPTS UNLIMITED

ALTER PROFILE DEFAULT LIMIT FAILED\_LOGIN\_ATTEMPTS UNLIMITED

**4.3 Oracle SGA**



1. SGA：System Global Area是Oracle Instance的基本组成部分，在实例启动时分配;系统全局域SGA主要由三部分构成：共享池、数据缓冲区、日志缓冲区。
2. 共享池：Shared Pool用于缓存最近被执行的SQL语句和最近被使用的数据定义，主要包括：Library cache（共享SQL区）和Data dictionary cache（数据字典缓冲区）。  共享SQL区是存放用户SQL命令的区域，数据字典缓冲区存放数据库运行的动态信息。
3. 缓冲区高速缓存：Database Buffer Cache用于缓存从数据文件中检索出来的数据块，可以大大提高查询和更新数据的性能。
4. 大型池：Large Pool是SGA中一个可选的内存区域，它只用于shared server环境。
5. Java池：Java Pool为Java命令的语法分析提供服务。
6. PGA：Process Global Area是为每个连接到Oracle database的用户进程保留的内存。

关于SGA这块的调整，网上有太多优秀的文章。

一般可以通过修改$ORACLE\_HOME/dbs目录下的init.ora文件来自行调整，但是。。。。。。

如果你要调整一个init.ora文件，请使用下面的步骤可确保你的修改不会造成Oracle启动不了

**第一步：修改前请先作一份init.ora文件的安全拷贝**

在Linux下使用下面这条命令

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. strings spfilektdb.ora > init.ora.bak

strings spfilektdb.ora > init.ora.bak

 一个init\*.ora对应一个spfile\*.ora

**第二步：对于这个init.ora.bak文件内的SGA值进行调整  
第三步：让Oracle直接用这个被修改后的init.ora.bak来启动数据库实例，请使用sys用户以命令行的方式登录oracle**

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. sqlplus sys/password@databaseSID as sysdba

sqlplus sys/password@databaseSID as sysdba

登录成功后运行下面两条命令

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. shutdown immediate
2. startup pfile='/opt/oracle/product/10/dbs/init.ora.bak‘

shutdown immediate

startup pfile='/opt/oracle/product/10/dbs/init.ora.bak‘

如果Oracle能够正常启动和支持客户端的连接说明我们的改动没有损坏到Oracle已安装的实例

**第四步：让更改生效使得Oracle每次启动都使用我们修改过后的SGA的值**

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. create spfile from pfile='/opt/oracle/product/10/dbs/init.ora.bak';
2. startup force

create spfile from pfile='/opt/oracle/product/10/dbs/init.ora.bak';

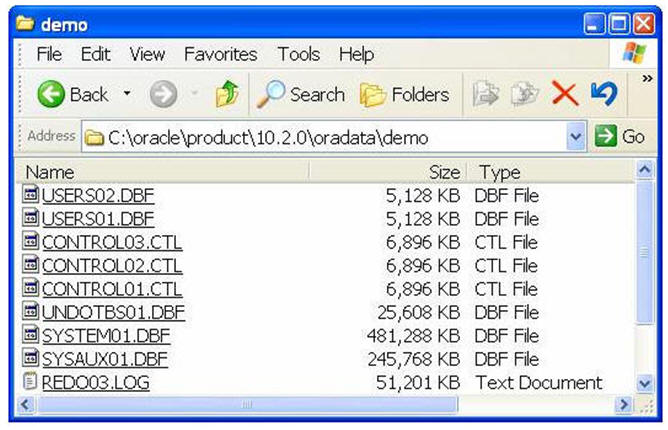
startup force

如果上述这两条命令没有启动，那么你的Oracle一旦重启后它将还是继续使用原有oracle的SGA配置，而非改动后的配置。

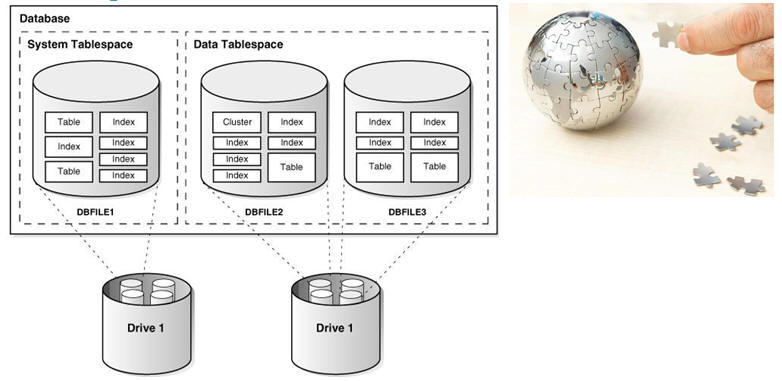
**4.4 Oracle表空间管理**



Oracle的表空间文件都放在$ORACLE\_HOME/oradata如/opt/oracle/product/10/oradata这样的目录中的

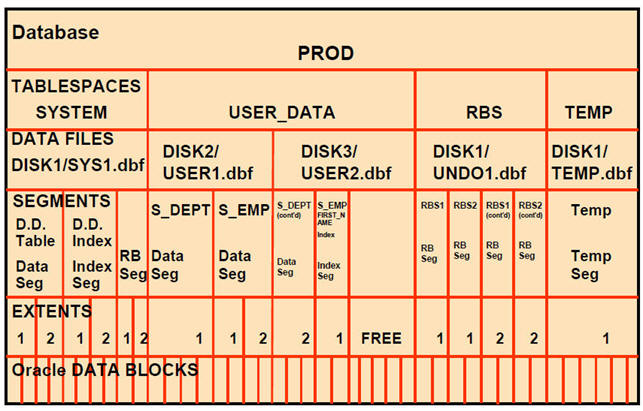


**Oracle的表空间支持”热插拨“**



即在Oracle运行时发觉表空间不够时可以直接打开Oracle的管理界面来动态给它划一块硬盘空间，或者甚至你又装了一块硬盘进服务后，Oracle可以把表空间在运行时扩展到新插入的磁盘中。

**表空间管理**



ORACLE的表空间划分将影响ORACLE的数据访问速度。

对于表空间来说，最重要的是如何把要连续访问的段放在一起，但是由于oracle 不能提供基于段的统计信息，所以对数据的物理模型设计和访问模式的详细了解对表空间的规划有非常大的好处。然后基于这些原则，我们就可以制定我们的表空间划分原则了：

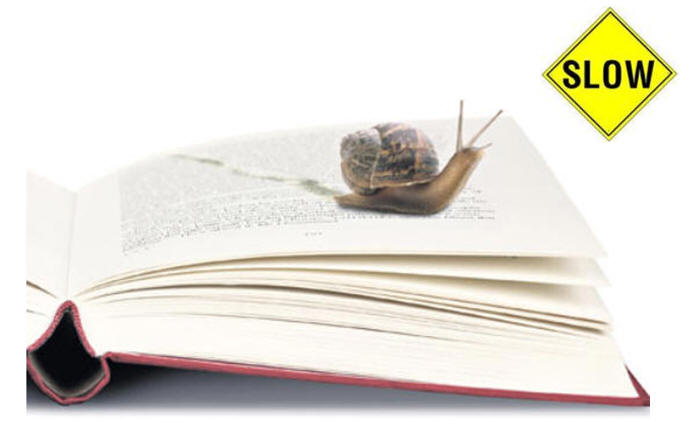
1. 仅在表空间级指定INITIAL、NEXT参数，在创建数据段时不要指定这些参数；
2. 对每个表空间上的段使用相同的区片尺寸；段参数INITIAL＝NEXT，PCTINCREASE＝0；可以通过使用Create Tablespace 的‘ MINIMUM EXTENT’ 子句来确保分配的区片是此参数的倍数；
3. 区片的大小根据段大小来确定，原则是均衡顺序扫描的效率和空间的利用率，同时确保段的区片数目控制在1024之下；根据此原则，在进行相应测试之后，确定以下区片选取规则：

|  |  |
| --- | --- |
| 段大小 | 区片大小 |
| 128M | 128K |
| 128M-4G | 4M |
| 4G及以上 | 128M |

1. Oracle9i引入了本地管理表空间，它在管理和性能上都优于传统的字典管理表空间，它已融合了规则1、2、3 ；要使用此特性，在CREATE TABLESPACE语句中指定EXTENT MANAGEMENT LOCAL子句；
2. 段的区片数目上限应在4096之下，DML操作在此区片数目范围内不会有明显的性能差异；但某些DDL操作的速度则与区片的数目关系较大；因此合理的区片数目应保持在1024之下；对于持续不断扩展的段，应监控区片数目，在必要时移至其它表空间；
3. 对于特别大的数据段应控制在4G－128G（Oracle7为5G－160G）之间，它们应存放到单独的表空间上，同时对于这些特大段应考虑使用分区拉提高性能；
4. 用户的临时表空间应使用TEMPORARY类型；
5. 当系统的事务规模比较均衡时可以对回滚段使用OPTIMAL参数，否则应避免制定OPTIMAL参数，而定期监控回滚段的大小，并在必要时重建；
6. 临时段和回滚段绝对不要将用户数据存放到SYSTEM表空间，它是专为永远不会Drop和Truncate的系统数据对象而设计的；
7. 创建表空间时指定数据文件的大小应＝区片整数倍＋1数据块，对于Local Managed Tablespace则为区片整数倍＋64K；
8. 当表空间使用统一的区片大小时，不要对其进行空间整理，重整的结果不仅耗费精力而且可能会使性能变差；对于未使用统一的区片尺寸的表空间应通过Export/Import重整；
9. 提供了Alter Table …Move [Tablespace…]命令可用于快速重整表，Alter Index …Rebuild…[Tablespace…] 命令可用于快速重建索引；

**4.5 Import （导入）**

当传统的导入导出遇到了海量数据时~

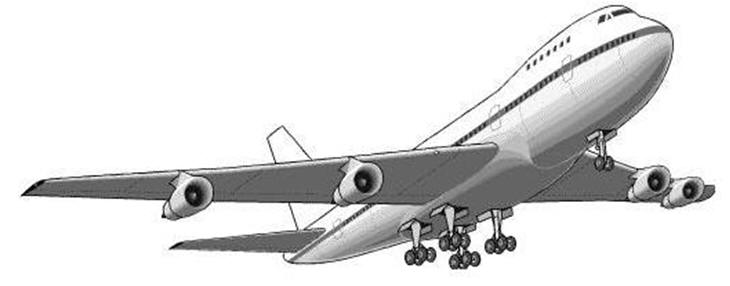


我在以前一个工程碰到过一个真实的案例，48-50张左右的表，每张表最大数据量为1200万，最小的在280-300万左右的数据，占用硬盘空间在14-16GB左右的一个.dmp包。不定期会进行导入导出操作。

于是我们的数据库负责人员就使用传统的imp/exp命令了。

每一次imp都需要耗费达4-6个小时，有时一旦出错。。。完蛋了，这个效率太低，大家不要看1200万这个数量，大家会说：哟，都是几百万的数据，可是这点数据其实还不算是真正的大数据量，imp一次要4-6小时，这是绝对不合理的。

**如何让你的imp飞起来**



传统的imp命令在导入时，如果只是仅处理数据，千万条数据对Oracle的处理来说根本就是”毛毛雨“啦，关键是在它导入了数据后，而要对每个表重新做一次索引。

一边导一边索引，一边一条条commit，就好比你用一个循环来insert 1万条数据和你改用statemenet.addBatch(query);的效率的对比一样，一定是后者更快更高效。

因此，在碰到这种情况下我们建议对imp导入命令做下面的折分：

* 分段式提交
* 设置缓冲
* 先导数据再导索引

这样，我们原来的imp命令就变成下面这样的样子了：

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. imp user2/pwd fromuser=user1 touser=user2 file=file commit=y feedback=10000 buffer=10240000 ignore=y rows=y indexes=n
3. imp user2/pwd fromuser=user1 touser=user2 file=file commit=y feedback=10000 buffer=10240000 ignore=y rows=n indexes=y

imp user2/pwd fromuser=user1 touser=user2 file=file commit=y feedback=10000 buffer=10240000 ignore=y rows=y indexes=n

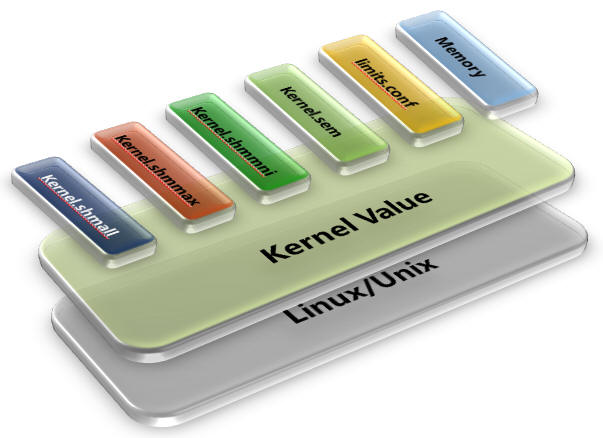
imp user2/pwd fromuser=user1 touser=user2 file=file commit=y feedback=10000 buffer=10240000 ignore=y rows=n indexes=y

看到没有，先导数据rows=y indexes=n，再导索引rows=n indexes=y

这两条命令是先后发起的，在只导数据时对于上述的14gb左右的一个.dmp包在同样软硬件环境中只用了15-20分钟，后一条建索引语句只用了25-27分钟。

这其中，提高了几倍？大家想想。

**4.6 Oracle与Linux系统的几个主要内核参数关系**



**limits.conf文件**

编辑这个 /etc/security/limits.conf

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. \* soft nofile 1024
2. \* hard nofile 1024

\* soft nofile 1024

\* hard nofile 1024

一般这个文件的默认值为1024

它代表Linux系统下最大打开文件数，如java里面的jdbc connection操作，new File操作都是一个文件打开操作，1024这个值是很少的。

所以我们把它改成

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. \* soft nofile 300000
2. \* hard nofile 300000

\* soft nofile 300000

\* hard nofile 300000

继续修改

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. oracle soft memlock 1048576
2. oracle hard memlock 1048576

oracle soft memlock 1048576

oracle hard memlock 1048576

这个值如果有的话直接改后面的数字，如果没有的话需要把这两行增加入limits.conf文件中，其中：

memlock’s value=Oracle share\_pool\_size (gb)\*1024\*1024  
所以你的Oracle中的SGA里的share\_pool\_size的大小是受到这个值的限制的。

改完后重启Linux系统，然后我们可以使用下面的命令来看我们修改的效果。

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. ulimit –a

ulimit –a

这个命令可以查看Linux系统当前的最大打开文件数。

使用Oracle用户登录

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. su – oracle

su – oracle

然后键入

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. ulimit -l

ulimit -l

就可以看到oracle soft memlock的相关修改效果了。

**关于kernel.\*的参数的配置**

主要是在/etc/sysctl.conf文件中

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. kernel.shmall = 2097152
2. kernel.shmmax = 2147483648
3. kernel.shmmni = 4096
4. kernel.sem = 250         32000    100          128
6. #SEMMSL     SEMMNS       SEMOPM  SEMMNI

kernel.shmall = 2097152

kernel.shmmax = 2147483648

kernel.shmmni = 4096

kernel.sem = 250 32000 100 128

#SEMMSL SEMMNS SEMOPM SEMMNI

其中最后一行带”#“是我加的，代表这几个数值的”含义“，那么光有含义，没有解释？下面给出解释

**SEMMSL=max processes+10  
SEMMNS=SEMMSL\*SEMMNI**

看到没有。。。。。。所以说Oracle中这个processes不是乱设的，是要和你系统的内核设置去绑定的。

**4.7Oracle在32位操作系统下的瓶劲与如何突破**

我去年写过篇博文，那篇博文是用于记录如何在实际的一个项目中在32位Linux操作系统下让Oracle的SGA突破2GB内存这个限制，详细可见：[oracle在32位的Linux环境下SGA如何突破2GB内存限制的最终解决方案](http://blog.csdn.net/lifetragedy/article/details/6212887)

在这边，我再做一下补充。

一些客户，这主要是客户关系，如果换成我我一定要买支持64位操作系统的服务器。

一些使用32位操作系统的客户安装Oracle，大家知道Oracle自从8.0后开始全面转成Java，因此它也受到JAVA虚拟机在32位操作系统下最多只能使用到2GB物理内存这个限制。



因此，当你的机器物理内存有32GB时，但因为你用了32位的操作系统，因此你的Oracle在创建Database时即customer（定义）这个数据库内存分匹时你的SGA是超不过2048MB的。

SGA中有一个很重要的指标即：shared\_pool\_size，这个值的大小会直接影响性能，关于shared\_pool\_size有很多更深入的理论性的探讨，这边告诉初学者或者新手，这个值相当于”游戏推荐配置“，不满足，游戏运行暴卡，超过这个值，游戏运行流畅。

但是，我现在有台服务器，物理内存32GB，用的是32位的Linux，我Oracle的SGA想要突破这个2GB大小的限制，又不能重装（客户环境不是你说要重装就要重装的）怎么办？只有想办法：

1. 让操作系统支持PAE模式（目前只有Linux AS3及以上和win 2003 advance server+sp2补丁及windows 2008）可以支持真正的PAE模式
2. 修改/etc/sysctl.conf文件中的值（前面提到过了）
3. 让Linux操作系统中的打开文件数为最大（前面也提到过了）
4. 更改你的Oracle的SGA



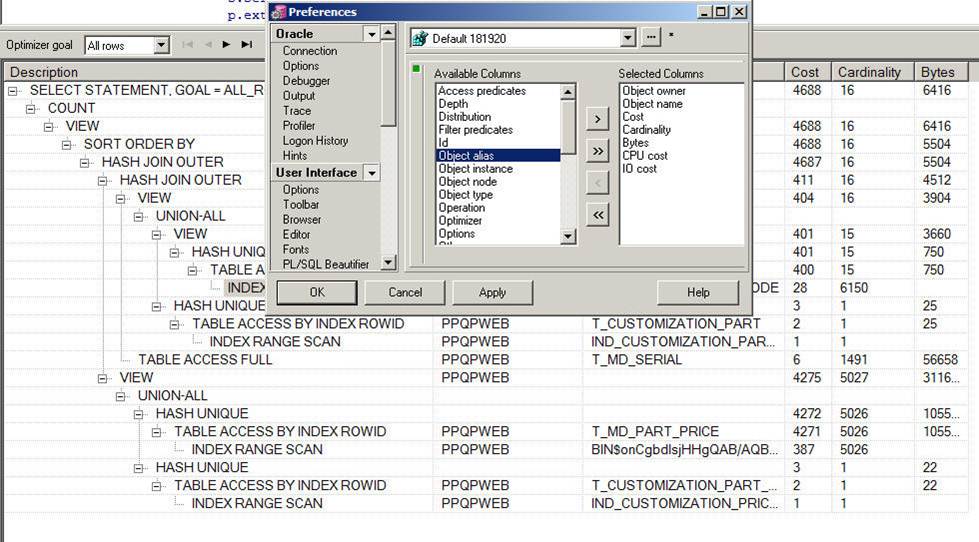
具体做法还是请见我那篇[oracle在32位的Linux环境下SGA如何突破2GB内存限制的最终解决方案](http://blog.csdn.net/lifetragedy/article/details/6212887)的博文吧。

这是我在32位的RedHat 5.5上把Oracle的SGA突破了2GB这个限制后的真实内存分布图，大家看看现在我的SGA是多少了？



查询效率一下提升20多倍，其实一点不值得骄傲，为什么？废话，谁让客户不懂没来事先问我的意见，在64位操作系统下，你机器这么多内存根本就不用这么麻烦

**4.8 使用SQL执行计划帮助你定位瓶劲**



一般我喜欢用第三方工具如：PLSQL Developer，它可以图形化的展现你的Oracle的执行计划，关于执行计划怎么看，大家可以参考这篇博文[oracle 执行计划（explain plan）说明](http://blog.csdn.net/gybyylx/article/details/6907588)

**在执行计划中使用hits来改变和调整SQL执行的性能**

Oracle Hints是一种机制，用来告诉优化器按照我们的告诉它的方式生成执行计划。我们可以用Oracle Hints来实现：

1. 使用的优化器的类型
2. 基于代价的优化器的优化目标，是all\_rows还是first\_rows
3. 表的访问路径，是全表扫描，还是索引扫描，还是直接利用rowid
4. 表之间的连接类型
5. 表之间的连接顺序
6. 语句的并行程度

除了”RULE”提示外，一旦使用的别的提示，语句就会自动的改为使用CBO优化器，此时如果你的数据字典中没有统计数据，就会使用缺省的统计数据。所以建议大家如果使用CBO或Hints提示，则最好对表和索引进行定期的分析。  
如何使用Hints:  
Hints只应用在它们所在sql语句块(statement block，由select、update、delete关键字标识)上，对其它SQL语句或语句的其它部分没有影响。如：对于使用union操作的2个 sql语句，如果只在一个sql语句上有Hints，则该Hints不会影响另一个sql语句。  
我们可以使用注释(comment)来为一个语句添加Hints，一个语句块只能有一个注释，而且注释只能放在SELECT, UPDATE, or DELETE关键字的后面  
使用Oracle Hints的语法：

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. {DELETE|INSERT|SELECT|UPDATE} /\*+ hint [text] [hint[text]]... \*/

4. or

7. {DELETE|INSERT|SELECT|UPDATE} --+ hint [text] [hint[text]]...

{DELETE|INSERT|SELECT|UPDATE} /\*+ hint [text] [hint[text]]... \*/

or

{DELETE|INSERT|SELECT|UPDATE} --+ hint [text] [hint[text]]...

注解：

1. DELETE、INSERT、SELECT和UPDATE是标识一个语句块开始的关键字，包含提示的注释只能出现在这些关键字的后面，否则提示无效。
2. “+”号表示该注释是一个Hints，该加号必须立即跟在”/\*”的后面，中间不能有空格。
3. hint是下面介绍的具体提示之一，如果包含多个提示，则每个提示之间需要用一个或多个空格隔开。
4. text 是其它说明hint的注释性文本

**如果你没有正确的指定Hints，Oracle将忽略该Hints，并且不会给出任何错误。**

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. /\*+ALL\_ROWS\*/
2. /\*+FIRST\_ROWS\*/
3. /\*+CHOOSE\*/
4. /\*+RULE\*/
5. /\*+FULL(TABLE)\*/
6. /\*+ROWID(TABLE)\*/
7. /\*+USE\_HASH(BSEMPMS,BSDPTMS)\*/

/\*+ALL\_ROWS\*/

/\*+FIRST\_ROWS\*/

/\*+CHOOSE\*/

/\*+RULE\*/

/\*+FULL(TABLE)\*/

/\*+ROWID(TABLE)\*/

/\*+USE\_HASH(BSEMPMS,BSDPTMS)\*/

这些常用的hits可以供大家参考，还有更多的hints可以参考oracle dba相关手册

**4.9 Table Analyze-Oracle的表分析**

什么是表分析，为什么需要表分析？



这要从Oracle的优化器谈起。Oracle的优化器有两种优化方式：

Oracle的优化器有两种优化方式：

* 基于规则的优化方式：Rule-Based Optimization（RBO）
* 基于成本或者统计信息的优化方式(Cost-Based Optimization：CBO)

**RBO方式：**优化器在分析SQL语句时,所遵循的是Oracle内部预定的一些规则。比如我们常见的，当一个where子句中的一列有索引时去走索引。

**CBO方式：**CBO是在ORACLE7 引入，但到ORACLE8i 中才成熟。ORACLE 已经声明在ORACLE9i之后的版本中，RBO将不再支持。它是看语句的代价(Cost),这里的代价主要指Cpu和内存。CPU Costing的计算方式现在默认为CPU+I/O两者之和.可通过DBMS\_XPLAN.DISPLAY\_CURSOR观察更为详细的执行计划。优化器在判断是否用这种方式时,主要参照的是表及索引的统计信息。统计信息给出表的大小、有少行、每行的长度等信息。这些统计信息起初在库内是没有的，是做analyze后才出现的，很多的时侯过期统计信息会令优化器做出一个错误的执行计划,因些应及时更新这些信息。**按理，CBO应该自动收集，实际却不然，有时候在CBO情况下，还必须定期对大表进行分析。**

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. ANALYZE TABLE ktdb.T\_CD\_CODE COMPUTE STATISTICS;
3. ANALYZE TABLE ktdb.T\_CD\_CODE COMPUTE STATISTICS  for all indexed columns;

ANALYZE TABLE ktdb.T\_CD\_CODE COMPUTE STATISTICS;

ANALYZE TABLE ktdb.T\_CD\_CODE COMPUTE STATISTICS for all indexed columns;

对一个表进行全表分析，就必须对这个表运行上述两条语句。  
比如说，你的数据库每天有几十万条数据进进出出，过了1周，数据库里原先一些查询已经不走Oracle默认的优化引擎了，本来是该走索引的，结果你用SQL分析器看出来它走的是full scan，这时就要做表分析了，一旦表分析做完后，你的数据库又会按照你原有的计划去走最优的路线了，这无疑中会提高你的数据库访问性能。

但是。。。如果我有100，200个表，我岂不是每一个表都要写这么两条语句？也真有这样的傻子写了几百条这样的重复语句，下面传授给大家一个用程序一次性生成所有的表的分析语句吧。

我们使用oracle的sql语句来做：

先生成所有的表分析语句

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. spool 'd:\analyzetable.txt‘
2. select   \*   from   user\_tables;
3. spool off

spool 'd:\analyzetable.txt‘

select \* from user\_tables;

spool off

再生成所有的表的索引分析语句

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. spool 'd:\analyzeindex.txt‘
2. select   \*   from   user\_indexes ;
3. spool off

spool 'd:\analyzeindex.txt‘

select \* from user\_indexes ;

spool off

最后使用批处理脚本来执行当前连接数据库中所有的表的分析语句，比如说我们创建一个dbAnalyze.sh文件。

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. export ORACLE\_BASE=/opt/oracle
2. export ORACLE\_HOME=$ORACLE\_BASE/product/10
3. export ORACLE\_SID=ktdb
4. $ORACLE\_HOME/bin/sqlplus "sys/sys@ktdb as sysdba" <<eof
5. @/home/oracle/analyzetable.txt;
6. exit
7. eof

export ORACLE\_BASE=/opt/oracle

export ORACLE\_HOME=$ORACLE\_BASE/product/10

export ORACLE\_SID=ktdb

$ORACLE\_HOME/bin/sqlplus "sys/sys@ktdb as sysdba" <<eof

@/home/oracle/analyzetable.txt;

exit

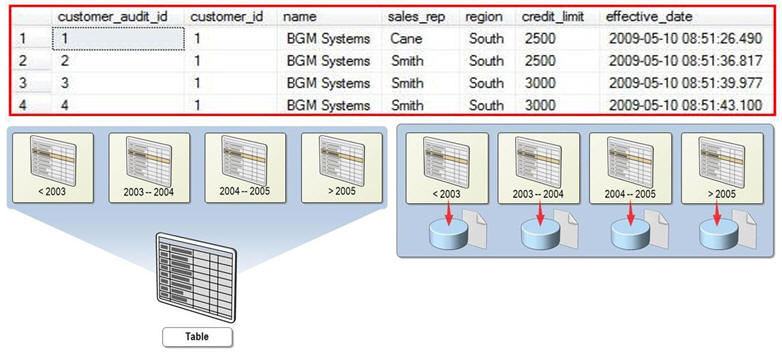
eof

由于数据库的表分析很费时，一般我们都会选择在零晨或者是在周末这两天进行一次表分析，这样保证，每次在工作日时我的数据库中的SQL始终走的是最适合的优化器.

这边说一个真实的CASE，3年前一个项目，项目刚开始导入了80GB的数据，开发了一段时间大概3-4个月时，第一阶段进入到性能测试，发觉这个报表的一些sql很慢，30秒，40秒。然后我过去后问了一下情况，先不分析这个SQL和JAVA代码先运行一下表分析器，直接所有的报表的SQL从原来的平均35秒一下变成了2.93秒。那剩下来的事呢，再来排摸这个SQL语句写的是否最优和再来看JAVA代码等等等问题。

在这边提这个CASE的目的在于告诉大家，有表分析这么一个事存在，碰到数据库经常数据做迁移或者是进进出出的次数多时，定期执行你的table analyze是非常有必要的，而且能够帮助你提升性能。

**4.10 Partition Table（分区表）**



Oracle的表分区功能通过改善可管理性、性能和可用性，从而为各式应用程序带来了极大的好处。通常，分区可以使某些查询以及维护操作的性能大大提高。此外,分区还可以极大简化常见的管理任务，分区是构建千兆字节数据系统或超高可用性系统的关键工具。  
   
分区功能能够将表、索引或索引组织表进一步细分为段，这些数据库对象的段叫做分区。每个分区有自己的名称，还可以选择自己的存储特性。从数据库管理员的角度来看，一个分区后的对象具有多个段，这些段既可进行集体管理，也可单独管理，这就使数据库管理员在管理分区后的对象时有相当大的灵活性。但是，从应用程序的角度来看，分区后的表与非分区表完全相同，使用 SQL DML 命令访问分区后的表时，无需任何修改。  
   
什么时候使用分区表：

1. 表的大小超过2GB
2. 表中包含历史数据，新的数据被增加都新的分区中。

表分区有以下优点：

1. 改善查询性能：对分区对象的查询可以仅搜索自己关心的分区，提高检索速度。
2. 增强可用性：如果表的某个分区出现故障，表在其他分区的数据仍然可用；
3. 维护方便：如果表的某个分区出现故障，需要修复数据，只修复该分区即可；
4. 均衡I/O：可以把不同的分区映射到磁盘以平衡I/O，改善整个系统性能。

缺点：   
分区表相关：已经存在的表没有方法可以直接转化为分区表。不过 Oracle 提供了在线重定义表的功能。

 其实表分区也是一种”反范式“的表设计，举个例子吧，电信为例。

它的数据库是怎么存的？不是一个table里有几个主键来区分一下就完了的，想一下电信的日用场景：

某客户来到电信营业厅，说：我查3天内的通话记录

客服人员：请稍侯

过了一会，客服人员告诉客户近3天的通话记录。

过几天，又来了一个客户说：我要查近3年内的通话记录

客服人员：你过几天来

为什么，为什么这边客服人员要让客户过几天来查而不是马上把结果告诉客户？因为这的这个数据库里的表名就是按照年月日来进行物理分区的，当客户要查询过大的数据时，由于已经物理分区了，所以这个数据库就可以存到磁带机上，当客户需要知道一个历史较长时间的记录时，电信的IT人员需要把历史的磁带机如：1997年XXX上海市杨浦区XXX的XXX客户的存档.dat文件所属的这卷磁带装上电脑再查询，这是需要时间的。

那为什么不直接把这些数据都存在一张表用一个DATE字段区分来查询一下不就完了？想想上亿的数据都存在一张表，你再怎么优化也优不到哪里去，此时就要引入反范式了。

比如说我按照时间来分区，本来是1亿条记录，按照时间分成10万条记录一个物理区，这样当我要查询的内容正好落在1区或者是2区时，我需要查询的结果的”分母“只有20万，而原本我不分区时的查询时的分母是”一亿“，是不是这个查询速度会得到显著的提高啊？

下面来看几种Oracle中分区的用法吧

**Range Partition（根据范围来分区）**

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. PARTITION BY RANGE (CUSTOMER\_ID)
2. (
3. PARTITION CUS\_PART1 VALUES LESS THAN (100000) TABLESPACE
4. CUS\_TS01, PARTITION CUS\_PART2 VALUES LESS THAN (200000)
5. TABLESPACE CUS\_TS02
6. )

PARTITION BY RANGE (CUSTOMER\_ID)

(

　PARTITION CUS\_PART1 VALUES LESS THAN (100000) TABLESPACE

CUS\_TS01, PARTITION CUS\_PART2 VALUES LESS THAN (200000)

TABLESPACE CUS\_TS02

)

**Hash partition（HASH分区）这个最傻瓜了最好用了，不需要指定分区的条件的**

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. CREATE TABLE emp
2. (
3. empno NUMBER (4),ename VARCHAR2 (30),sal NUMBER
4. )PARTITION BY HASH (empno) PARTITIONS 8
5. STORE IN (emp1,emp2,emp3,emp4,emp5,emp6,emp7,emp8);

CREATE TABLE emp

(

　empno NUMBER (4),ename VARCHAR2 (30),sal NUMBER

)PARTITION BY HASH (empno) PARTITIONS 8

STORE IN (emp1,emp2,emp3,emp4,emp5,emp6,emp7,emp8);

**Component Partition**

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. create table dinya\_test
2. (
3. transaction\_id number primary key,item\_id number(8) not null,transaction\_date date
4. ) partition by range(transaction\_date)subpartition by hash(transaction\_id) subpartitions
5. 3 store in (dinya\_space01,dinya\_space02,dinya\_space03)
6. (
7. partition part\_01 values less than(to\_date(‘2006-01-01’,’yyyy-mm-dd’)),
8. partition part\_02 values less than(to\_date(‘2010-01-01’,’yyyy-mm-dd’)),
9. partition part\_03 values less than(maxvalue)
10. );

create table dinya\_test

(

　　transaction\_id number primary key,item\_id number(8) not null,transaction\_date date

) partition by range(transaction\_date)subpartition by hash(transaction\_id) subpartitions

3 store in (dinya\_space01,dinya\_space02,dinya\_space03)

　(

　　partition part\_01 values less than(to\_date(‘2006-01-01’,’yyyy-mm-dd’)),

　　partition part\_02 values less than(to\_date(‘2010-01-01’,’yyyy-mm-dd’)),

　　partition part\_03 values less than(maxvalue)

);

分区表和表分析一样需要在数据库没有交易甚至是需要断开所有的连接时才能安全有效的去做的一个动作，但是oracle从9i后开始提供了一种在线分区，对于拥有百万级数据的一张表来说进行分区即可以保证并发访问的安全同时速度又快-仅几秒种时间就可以完成。

它就是：DBMS\_REDEFINITION包

**在线重定义分区表**

即：

1. 不动原表结结，建立和原表结构一样的表，并进行分区，此时新表是拥有分区结构的，且数据为空（空表）
2. 利用在线重定义把原表的数据整个copy到新的分区表中去
3. 删去原表

以下是Oracle 中使用在线重定义的例子。

**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. EXEC DBMS\_REDEFINITION.CAN\_REDEF\_TABLE('schema名', '你将要被切换的表名', DBMS\_REDEFINITION.CONS\_USE\_PK);
2. EXEC DBMS\_REDEFINITION.START\_REDEF\_TABLE('schema名','你将要被切换的表名', '拥分区结构的新表');
3. EXEC DBMS\_REDEFINITION.sync\_interim\_table('schema名', '你将要被切换的表名', '拥分区结构的新表');
4. EXEC DBMS\_REDEFINITION.FINISH\_REDEF\_TABLE('schema名', '你将要被切换的表名', '拥分区结构的新表');

EXEC DBMS\_REDEFINITION.CAN\_REDEF\_TABLE('schema名', '你将要被切换的表名', DBMS\_REDEFINITION.CONS\_USE\_PK);

EXEC DBMS\_REDEFINITION.START\_REDEF\_TABLE('schema名','你将要被切换的表名', '拥分区结构的新表');

EXEC DBMS\_REDEFINITION.sync\_interim\_table('schema名', '你将要被切换的表名', '拥分区结构的新表');

EXEC DBMS\_REDEFINITION.FINISH\_REDEF\_TABLE('schema名', '你将要被切换的表名', '拥分区结构的新表');

上述四条语句依次执行，几秒钟内你源有的表立记得就变成了新的分区表了.

如果在上述执行过程中出现了错误，你的原表照样还是被保护的好好的，因为你这时会得到Oracle的错误提示，坏也是坏在NEW表上，所以把NEW表剁掉并且和原表间的”在线“关系断开即可,使用下列语句：

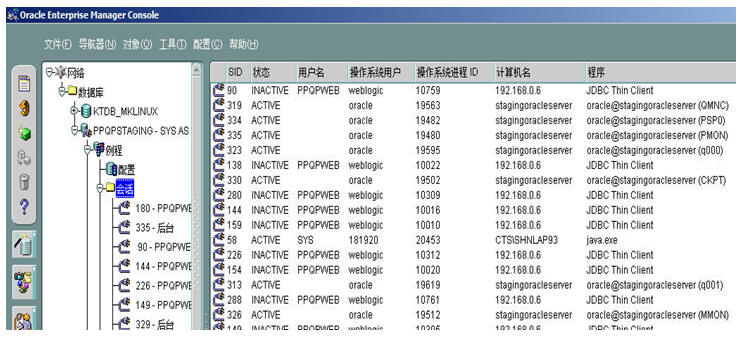
**[plain]** [view plaincopyprint?](http://blog.csdn.net/lifetragedy/article/details/8215312)

1. exec DBMS\_REDEFINITION.ABORT\_REDEF\_TABLE('schema名','你将要被切换的表名','拥分区结构的新表');

exec DBMS\_REDEFINITION.ABORT\_REDEF\_TABLE('schema名','你将要被切换的表名','拥分区结构的新表');

然后你检查一下错误 ，重新把那四条在线重定义分区表语句再依次运行一下即可。

**4.11 Oracle性能监控**



Oracle的客户端中的Oracle Enterprise Manager Console已经可以完成基本的监控任务。

Oracle11g后的客户端中不再提供Oracle Enterprise Manager Console，因此你要么安装11g的客户端后使用如第三方的Oracle客户：toad或者是PL SQL Developer要么就在Oracle11g的服务端打开dbconsole这个服务，然后在前台用：https://localhost:1158/em/console这个地址在客户端进行基于网页的Oracle客户端图形化管理吧。



如果你的服务端的Oracle是10G，那么请在服务端开启dbconsole这个服务时，客户端需要连接时使用这个地址：http://ip:1158