Oracle权威指南读书笔记

Oracle
DB2
Sybase
SQL Server

Oracle版本号含义 ：
11.1.0.1.1
主发布版本号
主发布维护号
应用服务器版本号
构建特定版本号
平台特定版本号

网格计算基于5个基本属性 ：
虚拟化
动态供应
资源集中
自适应软件
统一管理

Oracle 网格基础架构 ：
Oracle数据库网络
Oracle应用服务器网络
Oracle企业管理网格控制（Grid Control）

查看现有内存大小 ：
grep MemTotal /proc/meminfo

查看交换分区 ：
grep SwapTotal /proc/meminfo

查看系统剩余内存和交换分区 ：
free

查看共享内存数量 ：
df -k /dev/shm

查看tmp目录大小 ：
df -k /tmp

查看操作系统版本 ：
cat /proc/version

查看内核版本 ：
uname -r

ps : uname help信息 ：
[root@MysqlRestore exercise]# uname --help
Usage: uname [OPTION]...
Print certain system information. With no OPTION, same as -s.

  -a, --all   print all information, in the following order,
                             except omit -p and -i if unknown:
  -s, --kernel-name   print the kernel name
  -n, --nodename   print the network node hostname

```
  -r, --kernel-release     print the kernel release
  -v, --kernel-version     print the kernel version
  -m, --machine            print the machine hardware name
  -p, --processor          print the processor type or "unknown"
  -i, --hardware-platform  print the hardware platform or "unknown"
  -o, --operating-system   print the operating system
      --help     display this help and exit
      --version  output version information and exit

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
For complete documentation, run: info coreutils 'uname invocation'
[root@MysqlRestore exercise]#
```

查看软件包安装情况 ：

```
rpm -q package_name
```

开启或关闭防火墙 ：

```
vi /etc/selinux/config
```

增加以下 ：

```
SELINUX=disabled
```

rpm包检查 ：

```
rpm -q binutils compat-libstdc++-33 elfutils-libelf elfutils-libelf-devel glibc glibc-common glibc-devel gcc- gcc-c++ libaio-devel libaio libgcc libstdc++ libstdc++-devel make sysstat unixODBC unixODBC-devel
```

内核参数配置 ：

增加以下 ：

```
kernel.shmall = 2097152
kernel.shmmax = 2147483648
kernel.shmmin = 4096
kernel.sem = 250 3200 100 128
fs.file-max = 65536
net.ipv4.ip_local_port_range = 1024 6500
net.core.rmem_default = 4194304
net.core.rmem_max = 4194304
net.core.wmem_default = 262144
net.core.wmem_max = 262144
```

修改完后，以root身份运行使生效 ：

```
/sbin/sysctl -p
```

创建用户组 ：

```
groupadd oinstall
groupadd dba
useradd -g oinstall -G dba Oracle (设定Oracle用户为oinstall, dba用户组的成员)
passwd Oracle (为Oracle用户设置密码)
```

添加Oracle用户的限制参数 ：

```
vi /etc/security/limits.conf
```

增加以下 ：

```
Oracle soft nproc 2047
Oracle hard nproc 16384
Oracle soft nofile 1024
Oracle hard nofile 65536
```

```
vi /etc/pam.d/login
```

增加以下 ：

```
session required /lib/security/pam_limits.so
session required pam_limits.so
```

vi /etc/profile
增加以下 ：
if [$USER = "Oracle"] ; then
 if [$SHELL = "/bin/ksh"] ; then
ulimit -p 16384
ulimit -p 65536
else
ulimit -u 16384 -n 65536
fi
fi

创建安装所需要的目录 ：
mkdir -p /mount_point/app/
chown -R Oracle:oinstall /mount_point/app/
chmod -R 775 /mount_point/app/
note :
mount_point ： 自己指定的安装路径
创建临时目录 ：
mkdir /mount_point/tmp
chmod a+wr /mount_point/tmp

设置Oracle的环境变量 ：
vi .bash_profile
增加以下 ：
umask 022
TMP=/oratest/tmp //上面的mount_point此处具体为oratest
TMPDIR=/oratest/tmp
export TMP TMPDIR
if [-f ~/.bashrc] ; then
    . ~/.bashrc
fi
PATH=$PATH:$HOME/bin
export PATH
export ORACLE_BASE=/oratest/app/Oracle
export ORACLE_HOME=$ORACLE_BASE/product/11.1.0/db_1
export ORA_CRS_HOME=$ORACLE_BASE/crs
export ORACLE_PATH=$ORACLE_BASE/common/Oracle/sql:.:$ORACLE_HOME/rdbms/admin
export ORACLE_SID=orcl
export PATH=$ORACLE_HOME/bin:$ORACLE_CRS_HOME/bin:${PATH}:$HOME/bin
export PATH=${PATH}:/usr/bin:/bin:/usr/bin/X11:/usr/local/bin
export PATH=${PATH}:$ORACLE_BASE/common/Oracle/bin
export ORACLE_TERM=xterm
export TNS_ADMIN=$ORACLE_HOME/network/admin
export ORA_NLS10=$ORACLE_HOME/nls/data
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/lib:/usr/lib:/usr/local/lib:$ORACLE_HOME/oracm/lib:$ORACLE_HOME/lib
export LIBPAT=$LIBPATH:$ORA_CRS_HOME/lib:$ORACLE_HOME/lib
export CLASSPATH=${CLASSPATH}:$ORACLE_HOME/rdbms/jlib:$ORACLE_HOME/lib
export NLS_LANG=AMERICAL_AMERICA.ZHS16GBK
export LANG=AMERICAL_AMERICA.ZHS16GBK

解压缩安装包 ：
unzip linux_11gR1_database.zip

startup nomount : 该启动方式只需要init.ora文件
startup mount : 该启动方式打开控制文件
startup open : 该方式执行nomount, 再执行mount, 接着打开数据库文件
startup : 等于startup nomount + alter database mount + alter database open

startup restrict : 启动数据库，但只允许有一定特权的用户访问
startup force : 强制启动，当数据库不能关闭时，可用关闭数据库，再执行正常启动
startup pfile=参数文件路径 ： 该启动方式先读取参数文件，再按参数文件中的设置启动数据库
startup exclusive : 只允许当前例程读取数据库

shutdown normal : 正常关闭
shutdown immediate : 立即关闭
shutdown transactional : 尽量少影响客户端，避免客户丢失信息
shutdown abort : 放弃所有事务，立即关闭

lsnrctl start ： 启动监听进程
lsnrctl stop ： 关闭监听进程
lsnrctl status ： 查看监听进程状态
lsnrctl reload ： 重新加载监听进程
lsrnctl set : 设置相应参数
lsnrctl show ： 查看当前状态
lsnrctl help ： 显示帮助信息
lsnrctl version ： 显示当前监听进程版本
lsnrctl change_password ： 修改口令

从V$BGPROCESS数据字典中查询当前实例进程信息 ：
select name, description from v$bgprocess;
显示实例PGA信息 ：
show parameter pga

数据库字典的构成 ：
USER_ : 记录用户的对象信息
ALL_ : 记录用户的对象信息及被授权访问的对象信息
DBA_ : 包含数据库实例的所有对象信息
V$_ : 当前实例的动态视图，包含系统管理和系统优化等所使用的视图
GV_ : 分布式环境下所有实例的动态视图，包含系统管理和系统优化使用的视图，

基本的数据字典 ：
DBA_TABLES
DBA_TAB_COLUMNS
DBA_VIEWS
DBA_SYNONYMS
DBA_SEQUENCES
DBA_CONSTRAINTS
DBA_INDEXES
DBA_IND_COLUMNS
DBA_TRIGGERS
DBA_SOURCE
DBA_SEGMENTS
DBA_EXTENTS
DBA_OBJECTS
CAT
TAB
DICT : 构成数据字典的所有表信息
V$DATABASE
DBA_TABLESPACES
DBA_DATA_FILES
DBA_FREE_SPACE
V$CONTROLFILE
V$PARAMETER
V$CONTROLFILE_RECORD_SECTION
DBA_DATA_FILES
V$DATAFILE

V$FILESTAT
V$DATAFILE_HEADER
DBA_SEGTMENTS
DBA_EXTENTS
V$THREAD
V$LOG
V$LOGFILE
V$DATABASE
V$LOG
V$ARCHIVED_LOG
V$ARCHIVED_DEST
V$INSTANCE
V$PARAMETER
V$SYSTEM_PARAMETER
V$SGA
V$SGASTAT
V$DB_OBJECT_CACHE
V$SQL
V$SQLTEXT
V$SQLAREA
V$BGPROCESS
V$SESSION

常见动态性能视图 ：
V$FIXED_TABLE
V$INSTANCE
V$LATCH ： 锁存器统计数据
V$LIBRARYCACHE ： 库缓存性能的统计数据
V$ROLLSTAT ： 联机回滚段名字
V$ROWCACHE ： 活动数据字典的统计
V$SGA ： 系统全局区总结信息
V$SGASTAT ： 系统全局区详细信息
V$SORT_USAGE ： 临时段大小及会话，
V$SQLAREA ： 共享区SQL使用统计
V$SQLTEXT ： SGA中属于共享SQL游标的SQL语句内容
V$SYSSTAT ： 基本的实例统计信息
V$SYSTEM_EVENT ： 一个事件的总等待时间
V$WAITSTAT : 块竞争统计数据

使用简单CASE语句 ：
eg ：
```
select product_id, product_type_id,
  case product_type_id
    when 1 then 'Book',
    when 2 then 'Video',
    when 3 then 'DVD',
    when 4 then 'CD',
    else 'Magazine'
  end
from products;
```
eg ：
```
select product_id, price,
  case
    when price > 15 then 'Expensive'
    else 'Cheap'
  end
from products;
```

DECODE函数 ：
decode(value, search_value, result, default_value)对value与search_value进行比较，若相等，则返回result, 否则，返回default_value.
eg ：
```
select decode(1, 1, 2, 3) from dual;
```
eg ：
```
select prd_id, available,
  decode(available, 'Y', 'Product is available', 'Product is not avaialbel') from more_products;
```
eg ：
```
select product_id, product_type_id,
  decode(product_type_id, 'Book', 'Video', 'DVD', 'CD', 'Magazine') from products;
```
解释 ： product_type_id等于1则返回Book， product_type_id等于2则返回Video， product_type_id等于3则返回DVD， product_type_id等于4则返回CD, product_type_id等于任何其他值则返回Magazine.

查看Oracle支持的数据类型 ：
```
select type_name from dba_types where owner is null;
```

Oracle强烈建议，任何应用程序的库表至少需要创建两个表空间，一个用于存储表数据，另一个用于存储索引数据。如果将表数据和索引数据放一起，那么表数据的I/O操作和索引的I/O操作将产生影响系统性能的I/O竞争，降低系统的响应效率。而若将表数据和索引数据分开放在两个不同磁盘上，则可避免这种竞争。

将索引修改为逆键索引 ：
```
alter index pk_deptno rebuild reverse;
```
将索引空间合并 ：
```
alter index ename_idx coalesc;
```

insert first
eg ：
```
insert first
when sal > 25000 then
into special_sal values (deptid, sal)
when hiredate like ('%00%') then
into hiredate_history_00 values(deptid, hiredate)
when hiredate like ('%99%') then
into hiredate_history_99 values(deptid, hiredate)
else
into hiredate_history values(deptid, hiredate)
select department_id deptid, sum(salary) sal,
max(hire_date) hiredate
from employees
group by department_id;
```
解释 ： 如果第一个when子句为True, 则其后when子句则不会被执行。反之，直到遇到第一个满足条件的子句为止。

insert all

merge语句 ：将插入和更新合并为单个语句操作，可省略的update和insert之一子句
eg ：
```
merge into products p
using newproducts np
on (p.product_id = np.product_id)
when matched then
update
set p.product_name = np.product_name,
p,category = np.category;
```
eg ：
```
merge into products p
using newproducts np
on (p.product_id = np.product_id_
when not matched then
```

```
insert
values(np.product_id, np.product_name, np.category);

eg :
merge into products p
using newproducts np
on (p.product_id = np.product_id)
when matched then
update
set p.product_name = np.product_name
where p.category = np.category;
eg :
merge into products p
using newproducts np
on(p.product_id = np.product_id)
when matched then
update
set p.product_name = np.product_name,
p.category = np.category
where p.category = 'DVD'
when not matched then
insert
values (np.product_id, np.product_name, np.category)
where np.category != 'BOOKS';

eg :
merge into products p
using newproducts np
on (p.product_id = np.product_id)
when matched then
update
set p.product_name = np.product_name,
p.category = np.category
delete where (p.category = 'ELECTRNCS')
when not matched then
insert
values (np.product_id, np.product_name, np.category);

CREATE TYPE
eg :
create type BankAccount as object (
acct_number integer(5),
balance REBL,
status VARCHAR2(10),
member procedure open (amount in REAL),
member procedure verify_acct (num in integer),
member procedure close (num in integer, amount out real),
member procedure deposit (num in integer, amount in real),
member procedure withdraw (num in integer, amount in real),
member procedure curr_bal (num in integer) return real);

eg :
declare
  v_Number sales.prod_id%TYPE;
  v_Comment VARCHAR2(35);
  v_Test VARCHAR2(10) := 'See Here';
begin
  select Min(prod_id)
```

```
    into v_Number
    from sales;
case
  when v_Number < 500 then
    v_Comment := 'Too small';
    insert into tmp_table (my_com, vcomment)
      values ('This is too crazy', v_Comment);
  when v_Number < 100 then
    v_Comment := 'A little bigger';
    insert into tmp_table (my_com, vcomment)
      values ('This is only luck', v_Comment);
  when v_Test := 'See Here' then
    v_Comment := 'Test Model';
    insert into tmp_table (my_com, vcomment)
      values ('This is test', v_Comment);
  else
    v_Comment := 'That enough';
    insert into tmp_table (my_com, vcomment)
      values('Maybe good', v_Comment);
  end if;
end;
```

典型的PL/SQL代码块结构 ：

```
DECLARE   --Optional
Variables, cursors, user-defined, exception
BEGIN   --Mandatory
--SQL statements
--PL/SQL statements
EXCEPTION  --Optional
Actions to perform when errors occur
END;   --Mandatory
```

eg :

```
create or replace package body ClassPackage as
-- add a new student for the specified class.
procedure AddStudent (p_StudentID IN students.id%type,
p_Department in classes.department%type,
p_Course in classes.course%type) is
begin
insert into registered_students (student_id, department, course)
values (p_StudentID, p_Department, p_Course);
end AddStudent;

-- remove the specified student from the specified class.
procedure RemoveStudent (p_StudentID in students.id%type,
p_Department in classes.department%type,
p_Course in classes.course%type) is
begin
delete from registered_students
where student_id = p_StudentID
and department = p_Department
and course = p_Course;

-- check to see if the delete operation was successful. If
-- it didn't match any rows, raise an error.
if sql%notfound then
raise e_StudentNotRegistered;
end if;
```

```
end RemoveStudent;

-- returns a PL/SQL table containing the student currently in the specified class.
procedure ClassList (p_Department in classes.department%type,
p_Course in classes.course%type,
p_IDs out t_StudentIDTable,
p_NumStudents in out binary_integer) is
V_StudentID registered_students.student_id%type;

-- local cursor to fetch the registered students.
cursor c_RegisteredStudents is
select student_id
from registered_students
where department = p_Department
and course = p_Course;
begin
p_NumStudents := 0;
open c_RegisteredStudents;
loop
fetch c_RegisteredStudents into v_StudentID;
exit when c_RegisteredStudents%notfound;
p_NumStudents := p_NumStudents + 1;
p_IDs (p_NumStudents) := v_StudentID;
end loop;
end Classlist;
end ClassPackage;
```

SQL Plus使用技巧 ：

一、加注释

1. 使用REMARK加注释

eg ：

```
remark Commission Report;
remark to be run monthly;
column last_name heading 'last_name';
column salary heading 'monthly salary' format $99.999;
column commissioin_pct heading 'commission %' format 90.90;
remark includes only salesmen;
select last_name, salary, commission_pct
from emp_details_view
where job_id = 'SA_MAN';
```

2. 使用 /* ... */

3. 使用 --

二、运行命令

1. 命令行方式

命令后加分号；

2. SQL缓冲区方式

R(UN)命令和 / 斜杠命令

3. 命令文件方式

start命令和 @ at 命令

eg ：

```
start file_name[.sql] [arg1] [arg2]
```

Note ： @命令的功能与START命令非常相似，@命令既可以在SQL Plus命令行内部执行，又可以在启动SQL Plus是的命令行级别运行。而START命令只能在SQL Plus会话内部运行。

编写交互命令

1. 定义用户变量

eg ：

define ： 列出所有的变量定义

eg :

define MYFRIEND = SMITH : 定义变量MYFRIEND

eg :

undefine MYFRIEND : 删除变量MYFRIEND

2. 在命令中替代值

eg :

```
clear buffer
input
select ename, job, sal
from scott.emp e, scott.dept d
where e.deptno = d.deptno
and dname = &dname;
save test
```

3. 使用START命令提供值

4. 提示值