# 老男孩Oracle体验课-day03

## 3. Oracle网络管理

### 3.0 Oracle远程连接模型

### 3.1 lisener配置

[oracle@10.0.0.58 admin]$ cat listener.ora

# listener.ora Network Configuration File:

/u01/app/oracle/product/11.2/db\_1/network/admin/listener.ora

# Generated by Oracle configuration tools.

SID\_LIST\_LISTENER =

(SID\_LIST =

(SID\_DESC =

(SID\_NAME = PLSExtProc)

(ORACLE\_HOME = /u01/app/oracle/product/11.2/db\_1)

(PROGRAM = extproc)

)

(SID\_DESC =

(GLOBAL\_DBNAME = orcl)

(ORACLE\_HOME = /u01/app/oracle/product/11.2/db\_1)

(SID\_NAME = orcl)

)

)

LISTENER =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = 10.0.0.58)(PORT = 1521))

)

ADR\_BASE\_LISTENER = /u01/app/oracle

### 3.2 tnsname配置

[oracle@10.0.0.58 admin]$ cat tnsnames.ora

# tnsnames.ora Network Configuration File: /u01/app/oracle/product/11.2/db\_1/network/admin/tnsnames.ora

# Generated by Oracle configuration tools.

orcl\_1 =

(DESCRIPTION =

(ADDRESS\_LIST =

(ADDRESS = (PROTOCOL = TCP)(HOST = 10.0.0.58)(PORT = 1521))

)

(CONNECT\_DATA =

(SERVICE\_NAME = orcl)

)

)

orcl =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = 10.0.0.58)(PORT = 1521))

(CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = orcl)

)

)

## 4.Oracle 用户、权限、角色管理

### 4.1概念介绍

用户：对数据库的访问需要以适当的身份通过验证，这就是用户的作用；每个Oracle用户都有自己的用户名和密码，并且拥有他们所创建的任意表、视图和其他资源，需要注意的是新创建的用户连最基本的访问登入Oracle的权限都没有，所以后续需要授予不同的权限和角色来达成不同的操作目的；

实体权限：权限一般分为系统权限和对象权限，系统权限可以通俗地理解为将数据库对象从无到有或从有到无的创建操作，比如说创建和删除表、视图、存储过程、session等这些操作都可以认为是系统权限；对象权限赋予了你在不同的数据库对象上操作的能力，比如DELETE权限允许你对表和视图删除行，SELETE权限允许你对表、视图、序列等进行查询操作！至于系统权限和对象权限分别有哪些，具体请参照Oracle官网，这里不再详细解释！

角色：如果多个新建用户需要后续赋予10不等的不同权限，一般方法是需要将10个不等的权限分别列出来进行授予，不但麻烦而且重用性不高，角色的出现就是为了解决这个问题。Oracle角色是一组权限的集合。正常情况下我们可以授予角色一些权限，然后将角色分配给适当的用户；

### 4.2 权限分类

#### 4.2.1 管理相关权限

DBA: 拥有全部特权，是系统最高权限，只有DBA才可以创建数据库结构。

RESOURCE:拥有Resource权限的用户只可以创建对象，不可以创建数据库结构。

CONNECT:拥有Connect权限的用户可以登录Oracle，不可以创建实体，不可以创建数据库结构。

#### 4.2.2 普通对象权限

select, update, insert, alter, index, delete --->all 代表所有权限

execute -->执行存储过程权限

### 4.3 用户管理及授权

#### 4.3.1 创建用户的Profile文件

SQL> create profile student limit // student为资源文件名

FAILED\_LOGIN\_ATTEMPTS 3 //指定锁定用户的登录失败次数

PASSWORD\_LOCK\_TIME 5 //指定用户被锁定天数

PASSWORD\_LIFE\_TIME 30 //指定口令可用天数

## 例子：

create profile test limit

FAILED\_LOGIN\_ATTEMPTS 3

PASSWORD\_LOCK\_TIME 5

PASSWORD\_LIFE\_TIME 30;

#### 4.3.2 创建用户

SQL> Create User username

Identified by password

Default Tablespace tablespace

Temporary Tablespace tablespace

Profile profile

Quota integer/unlimited on tablespace;

## 例:

SQL> Create user oldguo

identified by oldguo

default tablespace users

temporary tablespace temp

profile test

quota 50m on users;

#### 4.3.3 查询用户默认表空间、临时表空间

SQL> select username, default\_tablespace, temporary\_tablespace from dba\_users;

#### 4.3.4 查询系统配置文件名

SQL> select \* from dba\_profiles;

## 配置文件类似表，一旦创建就会保存在数据库中。

SQL> select username, profile, default\_tablespace, temporary\_tablespace from dba\_users;

#### 4.3.5 查询用户会话信息：

SQL> select username, sid, serial#, machine from v$session;

## 删除用户会话信息：

SQL> Alter system kill session 'sid, serial#';

#### 4.3.5 修改用户

## 修改用户配置文件

SQL> create profile common limit

failed\_login\_attempts 5

idle\_time 5;

SQL> Alter user oldguo profile common;

SQL> Alter User 用户名

Identified 口令

Default Tablespace tablespace

Temporary Tablespace tablespace

Profile profile

Quota integer/unlimited on tablespace;

## 修改口令：

SQL>Alter user oldguo identified by "12345";

## 修改用户缺省表空间：

SQL> Alter user oldguo default tablespace users;

## 修改用户临时表空间

SQL> Alter user oldguo temporary tablespace temp\_data;

## 强制用户修改口令字：

SQL> Alter user oldguo password expire;

##将用户加锁

SQL> Alter user oldguo account lock; // 加锁

SQL> Alter user oldguo account unlock; // 解锁

#### 4.3.6 删除用户

SQL>drop user 用户名; //用户没有建任何实体

SQL> drop user 用户名 CASCADE; // 将用户及其所建实体全部删除

#### 4.3.7创建用户及授权管理权限

SQL> Create user user50 identified by user50;

SQL> grant connect, resource to user50;

## 查询用户拥有哪里权限：

set linesize 200;

select \* from dba\_role\_privs where grantee='USER50';

select \* from dba\_sys\_privs where grantee='USER50';

select \* from role\_sys\_privs where grantee='USER50';

#### 4.3.7 系统权限传递

增加WITH ADMIN OPTION选项，则得到的权限可以传递。

grant connect, resorce to user50 with admin option;

#### 4.3.8系统权限回收

系统权限只能由DBA用户回收

Revoke connect, resource from user50;

##普通对象权限管理

SQL> grant select, update, insert on product to user01;

SQL> grant all on product to user02;

##将表的操作权限授予全体用户：

SQL> grant all on product to public;

## 用户可以查询的表

SQL> select owner, table\_name from all\_tables;

## 用户创建的表

SQL> select table\_name from user\_tables;

## 获权可以存取的表（被授权的）

SQL> select grantor, table\_schema, table\_name, privilege from all\_tab\_privs;

// 授出权限的表(授出的权限)

SQL> select grantee, owner, table\_name, privilege from user\_tab\_privs;

### 4.4 角色管理

4.4.1角色介绍

角色是一组权限的集合，将角色赋给一个用户，这个用户就拥有了这个角色中的所有权限。

# 系统预定义角色

　　预定义角色是在数据库安装后，系统自动创建的一些常用的角色。下介简单的介绍一下这些预定角色。角色所包含的权限可以用以下语句查询：

sql>select \* from role\_sys\_privs where role='角色名';

1．CONNECT, RESOURCE, DBA

这些预定义角色主要是为了向后兼容。其主要是用于数据库管理。oracle建议用户自己设计数据库管理和安全的权限规划，而不要简单的使用这些预定角色。将来的版本中这些角色可能不会作为预定义角色。

2．DELETE\_CATALOG\_ROLE， EXECUTE\_CATALOG\_ROLE， SELECT\_CATALOG\_ROLE

这些角色主要用于访问数据字典视图和包。

3．EXP\_FULL\_DATABASE， IMP\_FULL\_DATABASE

这两个角色用于数据导入导出工具的使用。

4．AQ\_USER\_ROLE， AQ\_ADMINISTRATOR\_ROLE

AQ:Advanced Query。这两个角色用于oracle高级查询功能。

5． SNMPAGENT

用于oracle enterprise manager和Intelligent Agent

6．RECOVERY\_CATALOG\_OWNER

用于创建拥有恢复库的用户。关于恢复库的信息，参考oracle文档《Oracle9i User-Managed Backup and Recovery Guide》

7．HS\_ADMIN\_ROLE

A DBA using Oracle's heterogeneous services feature needs this role to access appropriate tables in the data dictionary.

管理角色

## 建一个角色

sql>create role role1;

## 授权给角色

sql>grant create any table,create procedure to role1;

## 授予角色给用户

sql>grant role1 to user1;

## 查看角色所包含的权限

sql>select \* from role\_sys\_privs;

## 创建带有口令以角色(在生效带有口令的角色时必须提供口令)

sql>create role role1 identified by password1;

## 修改角色：是否需要口令

sql>alter role role1 not identified;

sql>alter role role1 identified by password1;

## 设置当前用户要生效的角色

(注：角色的生效是一个什么概念呢？假设用户a有b1,b2,b3三个角色，那么如果b1未生效，则b1所包含的权限对于a来讲是不拥有的，只有角色生效了，角色内的权限才作用于用户，最大可生效角色数由参数MAX\_ENABLED\_ROLES设定；在用户登录后，oracle将所有直接赋给用户的权限和用户默认角色中的权限赋给用户。）

sql>set role role1;//使role1生效

sql>set role role,role2;//使role1,role2生效

sql>set role role1 identified by password1;//使用带有口令的role1生效

sql>set role all;//使用该用户的所有角色生效

sql>set role none;//设置所有角色失效

sql>set role all except role1;//除role1外的该用户的所有其它角色生效。

sql>select \* from SESSION\_ROLES;//查看当前用户的生效的角色。

## 修改指定用户，设置其默认角色

sql>alter user user1 default role role1;

sql>alter user user1 default role all except role1;

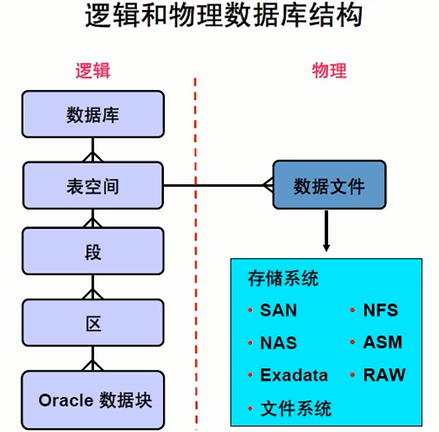
详见oracle参考文档

## 删除角色

sql>drop role role1;

角色删除后，原来拥用该角色的用户就不再拥有该角色了，相应的权限也就没有了

## 5、Oracle物理及逻辑结构



## 6、Oracle 重要文件管理

### 6.1. 参数文件

#### 6.1.1 优先级别：

oracle 启动读取参数文件的顺序,如果个文件都不存在，则Oracle会报错

spfile<sid>.ora --> spfile.ora -->init<sid>.ora

#### 6.1.2 默认目录：

$ORACLE\_HOME/dbs

#### 6.1.3 参数文件之间的转换

## spfile-->pfile

SQL> create pfile from spfile;

SQL> !ls $ORACLE\_HOME/dbs/initorcl.ora

/u01/app/oracle/product/11.2/db\_1/dbs/initorcl.ora

## 生成在指定路径

SQL> create pfile='/tmp/initorcl.ora' from spfile;

## pfile-->spfile

SQL> create spfile='/tmp/spfileorcl.ora' from pfile;

可以用strings命令查看创建的spfile文件，查出来的结过和pfile是一样的。

#### 6.1.4 判断数据库从SPFILE还是PFILE启动

## 方式1：

如果是spfile，则value是有值的

SQL> show parameter spfile

NAME TYPE VALUE

------------------------------------ ---------- ------------------------------

spfile string /u01/app/oracle/product/11.2

/db\_1/dbs/spfileorcl.ora

## 方式2：

SQL> set linesize 120

SQL> col name format a10

SQL> col value format a60

SQL> select name,value from v$parameter where name='spfile';

NAME VALUE

-------------------- ------------------------------------------------------------

spfile /u01/app/oracle/product/11.2/db\_1/dbs/spfileorcl.ora

#### 6.1.5 修改参数文件的三种模式：

scope=both 立即并永久生效，（默认模式）

scope=spfile 下次启动才能生效。

scope=memory 立即生效但下次启动时失效

##查询方法：v$parameter

isses\_modifiable:

ISSES\_MODIFIABLE＝true

表明这个参数可以在session级别可以修改，并且立即生效。false就是不能修改

issys\_modifiable:

issys\_modifiable＝immediate

表示这个参数可以在system立即修改，并且立即生效。

若如果issys\_modifiable=deferred

会话级别参数，直接通过alter session set <> = <>\*/

issys\_modifiable=false

表示这个参数不能在直接修改在内存中，需要加scope=spfile，重启后才能生效。

比如：

SQL> select name, isses\_modifiable,issys\_modifiable from v$parameter where name='sga\_max\_size';

NAME ISSES ISSYS\_MOD

------------------------------ ----- ---------

sga\_max\_size FALSE FALSE　　　　　　　　　　##两个都为false

SQL> alter system set sga\_max\_size=900M;

alter system set sga\_max\_size=900M

\*

第 1 行出现错误:

ORA-02095: 无法修改指定的初始化参数

SQL> alter system set sga\_max\_size=800M scope=spfile;

系统已更改。

恢复参数到默认值

SQL> alter system reset sga\_max\_size scope=spfile;

### 6.2、控制文件

#### 6.2.1 控制文件存储的主要信息

数据库名称和SID标识

数据文件和日志文件列表

数据库创建的时间戳

表空间信息

当前重做日志文件序列号

归档日志信息

检查点信息

回滚段的起始与结束

备份数据文件信息

#### 6.2.2 控制文件查看

select name from v$controlfile;

show parameter control\_files;

#### 6.2.3控制文件管理

## 添加控制文件

为了提高数据库的安全性，至少要为数据库建立两个控制文件，而且这两个文件最好分别放在不同的磁盘中，这样可以避免产生由于某个磁盘故障而无法启动数据库的危险，该管理策略称为多路复用控制文件。当多路复用控制文件某个磁盘发生故障导致其包含的控制文件损坏，数据库将被关闭或者发生异常，此时可以用另一磁盘中保存的控制文件来恢复被损坏的控制位文件，然后再重启数据库，达到保护控制文件的目的。

　　数据库建立时，一般会默认创建两个控制文件，我们可以手动的再创建多个控制文件且不要与默认的放在同一个磁盘中，首先我们可以修改control\_files参数来增加控制文件。

alter system set control\_files =

'/usr/oracle/app/oradata/orcl/control01.ctl','/usr/oracle/app/flash\_recovery\_area/orcl/control02.ctl','/home/oracle/ControlFiles/control03.ctl','/home/oracle/ControlFiles/control04.ctl'

scope=spfile;

其实添加控制文件就是将之前可用的控制文件复制过去，重命名即可，首先我们需要关闭数据库，然后将

/usr/oracle/app/oradata/orcl/control01.ctl 复制为/home/oracle/ControlFiles/control04.ctl

## 重建控制文件

例子：

数据库启动到nomount状态之后，执行create controlfile 命令

create controlfile

reuse database "orcl"

logfile

group 1 '/usr/oracle/app/oradata/orcl/redo01.log',

group 2 '/usr/oracle/app/oradata/orcl/redo02.log',

group 3 '/usr/oracle/app/oradata/orcl/redo03.log'

datafile

'/usr/oracle/app/oradata/orcl/system01.dbf',

'/usr/oracle/app/oradata/orcl/sysaux01.dbf',

'/usr/oracle/app/oradata/orcl/undotbs01.dbf',

'/usr/oracle/app/oradata/orcl/users01.dbf',

'/usr/oracle/app/oradata/orcl/CTRR\_DATA.dbf'

maxlogfiles 50

maxlogmembers 4

maxinstances 6

maxdatafiles 200

noresetlogs

noarchivelog;

## 备份控制文件

alter database backup controlfile to trace;

可以查看控制文件的具体内容，首先使用语句：

select tracefile from v$process where addr in (select paddr from v$session where sid in (select sid from v$mystat));

### 6.3 表空间及数据文件

#### 6.3.1 表空间介绍

Oracle 11g默认的表空间有六个：

|  |  |
| --- | --- |
| 表空间 | 说明 |
| EXAMPLE | 如果安装时选择"实例方案"，则此表空间存放各样例的数据 |
| SYSTEM | 存放数据字典，包括表、视图、存储过程的定义等 |
| SYSAUX | SYSTEM表空间的辅助空间。主要用于存储除数据字典外的其他的数据对象，这样可以减少SYSTEM表空间的负荷 |
| TEMP | 存放SQL语句处理的表和索引的信息，比如数据排序就占用此空间 |
| UNDOTBLS1 | 存放撤销数据的表空间 |
| USERS | 通常用于存放应用系统所使用的数据对象 |

#### 6.3.1 表空间创建

语法：

CREATE TABLESPACE 表空间名字

　　　　　　　　　　DATAFILE '数据文件详细信息'

　　　　　　　　　　[EXTENT MANAGEMENT { LOCAL

　　　　　　　　　　{AUTOALLOCATE | UNIFORM [SIZE INTETER [K|M] ] } } ]

|  |
| --- |
| -- 创建大小为50mb的永久表空间TEST01，禁止自动扩展数据文件  create tablespace TEST01      logging      datafile '/oradata/TEST01.dbf' size 50m      reuse autoextend off;    -- 创建永久表空间TEST02，允许自动扩展数据文件，本地管理方式  create tablespace TEST02      logging      datafile '/oradata/TEST02.dbf' size 50m      reuse autoextend on next 10m maxsize 200m      extent management local;    -- 创建永久表空间TEST03，允许自动扩展数据文件，本地管理方式，区分配方式为自动分配  create tablespace TEST03      logging      datafile'/oradata/TEST03.dbf' size 50m      reuse autoextend on next 10m maxsize 200m      extent management local autoallocate;    -- 创建永久表空间TEST04，允许自动扩展数据文件，本地管理方式，区分配方式为定制分配  create tablespace TEST04      logging      datafile'/oradata/TEST04.dbf' size 50m      reuse autoextend on next 10m maxsize 200m      extent management local uniform size 10m;    -- 创建永久表空间TEST05，允许自动扩展数据文件，本地管理方式，区分配方式为自动分配，段管理方式为自动管理  create tablespace test05      logging      datafile'/oradata/TEST05.dbf' size 50m      reuse autoextend on next 10m maxsize 200M      extent management local autoallocate      segment space management auto;    -- 创建永久表空间TEST06，允许自动扩展数据文件，本地管理方式，区分配方式为定制分配，段管理方式为手动管理  create tablespace test06      logging      datafile'/oradata/TEST06.dbf' size 50m      reuse autoextend on next 10m maxsize 200M      extent management local uniform size 10m      segment space management manual; |

#### 6.3.2临时表空间

|  |  |
| --- | --- |
|  | -- 创建一个临时表空间test07，为了避免临时空间频繁分配与回收时产生大量碎片，临时表空间的区只能采用自动分配方式  create temporary tablespace test07      tempfile '/oradata/TEST07.dbf' size 20M; |

#### 6.3.3创建UNDO表空间

|  |  |
| --- | --- |
|  | -- 创建一个undo表空间test08  create undo tablespace test08      datafile'/oradata/TEST08.dbf' size 20M; |

#### 6.3.4管理表空间

|  |  |
| --- | --- |
|  | -- 通过alter tablespace命令该表空间中数据文件的位置（或名字）  --首先把该表空间设置为脱机状态  alter tablespace TEST03 offline normal;  -- 手动创建TEST03文件夹，并且复制TEST03.dbf到TEST03文件夹，重命名为TEST03\_1.dbf  alter tablespace TEST03 rename datafile      '/oradata/TEST03.dbf'      to      '/oradata/TEST03/TEST03\_1.dbf';  --最后把该表空间设置为联机状态  alter tablespace TEST03 online; |

#### 6.3.5删除表空间

|  |  |
| --- | --- |
|  | --删除表空间test01 及其对应的数据文件  drop tablespace test01 including contents and datafiles;    --删除表空间test02 及其对应的数据文件  drop tablespace test02 including contents and datafiles;    --删除表空间test03 及其对应的数据文件  drop tablespace test03 including contents and datafiles;    --删除表空间test04 及其对应的数据文件  drop tablespace test04 including contents and datafiles;    --删除表空间test05 及其对应的数据文件  drop tablespace test05 including contents and datafiles;    --删除表空间test06 及其对应的数据文件  drop tablespace test06 including contents and datafiles;    --删除表空间test07 及其对应的数据文件  drop tablespace test07 including contents and datafiles;    --删除表空间test08 及其对应的数据文件  drop tablespace test08 including contents and datafiles; |

#### 6.3.6查询表空间使用情况

--查询表空间的free space

select tablespace\_name, count(\*) AS extends,round(sum(bytes) / 1024 / 1024, 2) AS MB,sum(blocks) AS blocks from dba\_free\_space group BY tablespace\_name;

--查询表空间的总容量

select tablespace\_name, sum(bytes) / 1024 / 1024 as MB from dba\_data\_files group by tablespace\_name;

--查询表空间使用率

SELECT total.tablespace\_name,

Round(total.MB, 2) AS Total\_MB,

Round(total.MB - free.MB, 2) AS Used\_MB,

Round(( 1 - free.MB / total.MB ) \* 100, 2)

|| '%' AS Used\_Pct

FROM (SELECT tablespace\_name,

Sum(bytes) / 1024 / 1024 AS MB

FROM dba\_free\_space

GROUP BY tablespace\_name) free,

(SELECT tablespace\_name,

Sum(bytes) / 1024 / 1024 AS MB

FROM dba\_data\_files

GROUP BY tablespace\_name) total

WHERE free.tablespace\_name = total.tablespace\_name;

#### 6.3.6 表空间扩容

第一种方式：表空间增加数据文件

alter tablespace test01 add datafile '/oradata/test01\_2.DBF' size 30M;

第二种方式：表空间增加数据文件，设置自增长，限制最大值

alter tablespace test01 add datafile '/oradata/test01\_3.DBF' size 50M autoextend on maxsize 3072M;

第三种方式：已存在表空间数据文件设置自增长

alter database datafile '/oradata/TEST01.dbf' autoextend on maxsize 3072M;

第四种方式：已存在表空间数据文件重新设置大小

alter database datafile '/oradata/test01\_3.DBF' resize 80M;

### 6.4 REDO及归档

6.4.1 redo日志状态查看

select group#,thread#,sequence#,bytes/1024/1024

size\_MB,members,archived,status,first\_change#,to\_char(FIRST\_TIME,'yyyy-mm-dd hh24:mi:ss') first\_time from v$log;

#### 6.4.2添加日志组

SQL> alter database add logfile group 4 ('/oradata/redo04a.log','/oradata/redo04b.log') size 20M;

#### 6.4.3 添加日志成员

SQL> alter database add logfile group 5 ('/oradata/redo05a.log') size 20M;

-->给group 5添加一个日志成员

SQL> alter database add logfile member '/oradata/redo05b.log' to group 5;

#### 6.4.4 删除日志组

检查日志组状态

SQL> select group#,thread#,sequence#,bytes/1024/1024

size\_MB,members,archived,status,first\_change#,to\_char(FIRST\_TIME,'yyyy-mm-dd hh24:mi:ss') first\_time from v$log;

当archvied状态为yes,status状态为inactive or unused状态时，可以删除该redolog group。

SQL> alter database drop logfile group 5;

#### 6.4.5 redolog损坏的恢复

在做下面实验之前，请先对数据库进行一次全库备份。

mkdir /u01/backup

vim full.sh

export ORACLE\_BASE=/u01/app/oracle

export ORACLE\_HOME=$ORACLE\_BASE/product/11.2/db\_1

export ORACLE\_SID=testdb

export NLS\_LANG=AMERICAN\_AMERICA.ZHS16GBK

export PATH=$ORACLE\_HOME/bin:$PATH

rman target / log /u01/backup/rman\_full.log append<<EOF

run

{allocate channel c1 type disk;

allocate channel c2 type disk;

backup database filesperset 4 format '/u01/backup/full\_%d\_%T\_%s\_%p';

sql 'alter system archive log current';

sql 'alter system archive log current';

sql 'alter system archive log current';

backup archivelog all format '/u01/backup/arch\_%d\_%T\_%s\_%p' delete input;

backup current controlfile format '/u01/backup/ctl\_%d\_%T\_%s\_%p';

crosscheck backup;

crosscheck archivelog all;

delete noprompt obsolete;

delete noprompt expired backup;

delete noprompt expired archivelog all;

}

EOF

## 离线日志组成员损坏

-->检查redolog日志组状态

SQL> select group#,thread#,sequence#,bytes/1024/1024

size\_MB,members,archived,status,first\_change#,to\_char(FIRST\_TIME,'yyyy-mm-dd hh24:mi:ss') first\_time from v$log;

-->正常关闭数据库

SQL> shutdown immediate

Database closed.

Database dismounted.

ORACLE instance shut down.

-->删除日志组4的一个日志组成员

SQL> ! rm -f /oradata/redo04b.log

-->启动数据库，数据库可以正常启动但是在alert log 日志中报错误。

-->检查日志状态

SQL> select a.group#,a.thread#,b.member,a.bytes/1024/1024

size\_MB,a.members,a.archived,a.status from v$log a,v$logfile b where a.group#=b.group#;

-->遇到这种情况使用alter database clear方式重建group.

SQL> alter database clear logfile group 4;

Database altered

SQL> select \* from v$logfile;

## 非当前日志组损坏

这种情况也是比较好恢复的。与上面的实例相似。

-->检查日志组状态

SQL> select group#,thread#,sequence#,bytes/1024/1024

size\_MB,members,archived,status,first\_change#,to\_char(FIRST\_TIME,'yyyy-mm-dd hh24:mi:ss') first\_time from v$log;

-->关闭数据库

SQL> shutdown immediate

Database closed.

Database dismounted.

ORACLE instance shut down.

-->删除日志组3全部成员

SQL> ! rm -f /oradata/redo03\*

-->启动数据库报错

SQL> startup

-->启动数据库到mount状态

SQL> startup mount

SQL> select group#,thread#,sequence#,bytes/1024/1024

size\_MB,members,archived,status,first\_change#,to\_char(FIRST\_TIME,'yyyy-mm-dd hh24:mi:ss') first\_time from v$log;

-->处理

SQL> alter database clear logfile group 2;

Database altered.

-->数据库可以正常打开。

SQL> alter database open;

Database altered.

## 当前日志组损坏

这种状态处理起来是比较麻烦的。需要做不完全恢复或者强制打开，这些都是有可能导致数据丢失的。因此我们要尽量避免这种情况的发生，对每组日志进行镜像就是一个不错的方法。

-->恢复方法:

1）如果有归档和备份，用不完全恢复。

SQL>startup mount;

SQL>recover database until cancel; 先选择auto尽可能多的利用归档日志进行恢复，然后再次执行

SQL>recover database until cancel; 这次输入cancel，完成不完全恢复

用resetlogs打开数据

SQL>alter database open resetlogs;

2）强制恢复， 这种方法可能会导致数据不一致

sql>startup mount;

sql>alter system set "\_allow\_resetlogs\_corruption"=true scope=spfile;

sql>recover database until cancel;

sql>alter database open resetlogs;

-->使用该隐含参数将库来起来以后，需要将数据导出然后重新建库。

## 7、Oracle的备份与恢复

### 7.1 exp/imp

#### 7.1.1 exp应用

SQL> select userenv('language') from dual;

USERENV('LANGUAGE')

----------------------------------------------------

AMERICAN\_AMERICA.AL32UTF8

export NLS\_LANG= AMERICAN\_AMERICA.AL32UTF8

获取帮助

exp systm/123 help=y

全库导出

 exp system/oracle@orcl file=/u01/backup/full.dmp full=y

按用户导出

 exp system/123 file=/u01/backup/test.dmp owner=hr

 按表导出

  exp hr/123 file=/u01/backup/t.dmp tables=T1,T2

#### 7.1.2 imp

全库导入

 imp system/oracle@orcl file=/u01/backup/full.dmp full=y

按用户导出

drop table t1;

drop table t2

imp system/123 file=/u01/backup/test.dmp fromuser=hr touser=hr

 按表导出

 imp hr/123 file=/u01/backup/t.dmp tables=T1,T2

### 7.2 expdp/impdp

#### 7.2.1 expdp

0)创建恢复目录，并授权用户

create directory dir as '/u01/backup';

grant read,write on directory dir to system;

grant read,write on directory dir to hr;

1)按用户导

expdp system/123 schemas=hr dumpfile=expdp1.dmp DIRECTORY=dir;

2)并行进程parallel

expdp system/123 directory=dir dumpfile=expdp2.dmp parallel=40 job\_name=expdp2

3)按表名导出

expdp hr/123 TABLES=t1,t2 dumpfile=expdp3.dmp DIRECTORY=dir;

4)按查询条件导出

expdp hr/123 directory=dir dumpfile=expdp4.dmp Tables=t1 query='WHERE id=1';

5)按表空间导出

expdp system/123 DIRECTORY=dir DUMPFILE=tablespace.dmp TABLESPACES=users;

6)导出整个数据库

expdp system/123 DIRECTORY=dir DUMPFILE=full.dmp FULL=y;

#### 7.2.2impdp

1)导到指定用户下

impdp system/123 DIRECTORY=dir DUMPFILE=expdp1.dmp SCHEMAS=hr;

2)改变表的owner

impdp system/123 DIRECTORY=dir DUMPFILE=expdp.dmp TABLES=hr.t1 REMAP\_SCHEMA=hr:system;

3)导入表空间

impdp system/manager DIRECTORY=dir DUMPFILE=tablespace.dmp TABLESPACES=users;

4)导入数据库

impdb system/manager DIRECTORY=dump\_dir DUMPFILE=full.dmp FULL=y;

### 7.3 rman

#### 7.3.1 备份脚本展示

单机全备：

vim full.sh

export ORACLE\_BASE=/u01/app/oracle

export ORACLE\_HOME=$ORACLE\_BASE/product/11.2/db\_1

export ORACLE\_SID=testdb

export NLS\_LANG=AMERICAN\_AMERICA.ZHS16GBK

export PATH=$ORACLE\_HOME/bin:$PATH

rman target / log /u01/backup/rman\_full.log append<<EOF

run

{allocate channel c1 type disk;

allocate channel c2 type disk;

backup database filesperset 4 format '/u01/backup/full\_%d\_%T\_%s\_%p';

sql 'alter system archive log current';

sql 'alter system archive log current';

sql 'alter system archive log current';

backup archivelog all format '/u01/backup/arch\_%d\_%T\_%s\_%p' delete input;

backup current controlfile format '/u01/backup/ctl\_%d\_%T\_%s\_%p';

crosscheck backup;

crosscheck archivelog all;

delete noprompt obsolete;

delete noprompt expired backup;

delete noprompt expired archivelog all;

}

EOF

单机归档备份：

export ORACLE\_BASE=/u01/app/oracle

export ORACLE\_HOME=$ORACLE\_BASE/product/11.2/db\_1

export ORACLE\_SID=testdb

export NLS\_LANG=AMERICAN\_AMERICA.ZHS16GBK

export PATH=$ORACLE\_HOME/bin:$PATH

rman target / log /u01/backup/rman\_arch.log append<<EOF

run

{allocate channel c1 type disk;

allocate channel c2 type disk;

sql 'alter system archive log current';

sql 'alter system archive log current';

sql 'alter system archive log current';

backup archivelog all format '/u01/backup/arch\_%d\_%T\_%s\_%p' delete input;

backup current controlfile format '/u01/backup/ctl\_%d\_%T\_%s\_%p';

crosscheck backup;

crosscheck archivelog all;

delete noprompt  expired backup;

delete  noprompt expired archivelog all;

}

EOF

RAC全备脚本：

run {

allocate channel c1 device type disk connect 'sys/Oracle@rac1';

allocate channel c2 device type disk connect 'sys/Oracle@rac1';

backup database database format '/backup/rmanbak/db\_full\_%T\_%d\_%t\_%s\_%p\_bak0';

sql 'alter system archive log current';

backup archivelog all format '/backup/rmanbak/arch\_%T\_%d\_%t\_%s\_%p\_bak0' delete input;

backup current controlfile format '/backup/rmanbak/controlfile\_%T\_%d\_%t\_%s\_%p\_bak';

backup spfile format '/backup/rmanbak/spfile\_%T\_%d\_%t\_%s\_%p\_bak';

crosscheck backup;

crosscheck archivelog all;

delete noprompt obsolete;

delete noprompt expired backup;

delete noprompt expired archivelog all;

release channel c1;

release channel c2;

}

在使用RMAN备份、还原的过程中，我们经常需要查看备份的一些详细信息，例如，RMAN提供了LIST命令。关于LIST命令的详细信息

可以参考Oracle Database Backup and Recovery Reference文档。

#### 7.3.2 列出备份信息

列出详细备份信息：备份集（Backup Sets）,镜像备份(image copies),proxy copies的信息。

#lists backup sets, image copies, and proxy copies

RMAN> list backup;

列出信息备份信息：备份集（Backup Sets）,proxy copies信息。

#lists only backup sets and proxy copies

RMAN> list backupset;

列出所有数据文件的备份信息

# lists backups of all files in database

RMAN> list backup of database;

列出特定表空间的所有数据文件备份集

RMAN> list backup of tablespace users;

列出指定文件的备份信息

RMAN> list backup of datafile 1;

或

RMAN> list backup of datafile '/u01/oradata/testdb/system01.dbf';

按备份类型列出备份

RMAN> list backup by file;

列出备份的映像副本信息

#lists only disk copies

RMAN> list copy;

列出过期备份信息（backup sets, proxy copies, and image copies）。

Displays backup sets, proxy copies, and image copies marked in the repository as EXPIRED, that is, "not found."

RMAN> list expired backup;

列出过期备份信息（backup sets, proxy copies）

RMAN> list expired backupset;

按备份类型列出过期备份信息

RMAN> list expired backup by file;

概述可用的备份

RMAN> list backup summary;

概述过期的备份

RMAN> list expired backup summary;

列出某个备份集的信息

RMAN> list backupset 59139;

列出数据文件映像副本

RMAN> list copy of datafile '/u01/oradata/system01.dbf';

列出控制文件的备份信息

RMAN> list backup of controlfile;

列出备份集中TAGE的备份片信息

RMAN> list backupset tage 'TAG20160811T000724'

列出所有备份的控制文件信息

RMAN> list backup of controlfile;

列出备份的spfile文件信息

RMAN> list backup of spfile;

查看已备份的归档日志信息

RMAN> list backup of archivelog all;

RMAN> list backup of archivelog from scn 22727871375;

RMAN> list backup of archivelog until scn 22727813497;

RMAN> list backup of archivelog from scn 22727031113 until scn 22727268951;

RMAN> list backup of archivelog from logseq 25432;

RMAN> list backup of archivelog until logseq 25432;

RMAN> list backup of archivelog from logseq 25426 until logseq 25428;

列出所有的归档日志

RMAN> list archivelog all;

列出所有失效的归档日志

RMAN>list expired archivelog all;

列出指定表空间的备份信息

RMAN> list copy of tablespace 'SYSTEM';

#### 7.3.3 rman恢复数据库

#### 7.3.4 RMAN Duplicate复制数据库

rman target sys/oracle@testdb auxiliary sys/oracle@test

duplicate target database to "TEST" from active database nofilenamecheck;

## 8、Oracle DataGuard

### 8.1主库添加Standby Redo Log

--查看Redo和Standby Redo

SQL> select \* from v$logfile;

--仅仅显示Online Redo，不显示Standby Redo

SQL> select \* from v$log;

--新增一组大小为50M的Standby Redo，这里的group号不得与Online redo重复

alter database add standby logfile group 21 '/u01/oradata/testdb/orclstandby21.log' size 50M;

alter database add standby logfile group 22 '/u01/oradata/testdb/orclstandby22.log' size 50M;

alter database add standby logfile group 23 '/u01/oradata/testdb/orclstandby23.log' size 50M;

alter database add standby logfile group 24 '/u01/oradata/testdb/orclstandby24.log' size 50M;

alter database add standby logfile group 25 '/u01/oradata/testdb/orclstandby25.log' size 50M;

### 8.2从主库创建pfile文件

创建pfile文件, 默认路径为$ORACLE\_HOME/dbs，此处为

/u01/app/oracle/product/11.2/db\_1/dbs/，在sqlplus里执行以下命令

SQL> create pfile from spfile;

将主库的pfile复制到备库/u01/app/oracle/product/11.2/db\_1/dbs/下

cd /u01/app/oracle/product/11.2/db\_1/dbs/

scp initorcl.ora db12:/u01/app/oracle/product/11.2/db\_1/dbs/

### 8.3设置主库初始化参数

编辑/u01/app/oracle/product/11.2/db\_1/dbs/inittestdb.ora文件，追加

cat >> /u01/app/oracle/product/11.2/db\_1/dbs/inittestdb.ora << "EOF"

db\_file\_name\_convert=('/u01/oradata/testdb','/u01/oradata/testdb')

log\_file\_name\_convert=('/u01/oradata/testdb','/u01/oradata/testdb')

\*.db\_unique\_name='testdb'

\*.fal\_server='test'

\*.log\_archive\_config='dg\_config=(testdb,test)'

\*.log\_archive\_dest\_1='location=/u01/arch1 valid\_for=(all\_logfiles, all\_roles) db\_unique\_name=testdb'

\*.log\_archive\_dest\_2='service=test lgwr async valid\_for=(online\_logfile,primary\_role) db\_unique\_name=test'

\*.log\_archive\_dest\_state\_1=ENABLE

\*.log\_archive\_dest\_state\_2=ENABLE

\*.standby\_file\_management='AUTO' EOF

创建新的主库spfile文件，并重新启动主库

SQL> shutdown immediate

SQL> create spfile from pfile;

SQL> startup

SQL> alter database force logging;

### 8.4设置备库初始化参数

编辑/u01/app/oracle/product/11.2/db\_1/dbs/inittest.ora文件，修改备库初始化参数

cat >> /u01/app/oracle/product/11.2/db\_1/dbs/inittest.ora << "EOF"

\*.db\_unique\_name='test'

\*.fal\_server='testdb'

db\_file\_name\_convert=('/u01/oradata/testdb','/u01/oradata/testdb')

log\_file\_name\_convert=('/u01/oradata/testdb','/u01/oradata/testdb')

\*.log\_archive\_config='dg\_config=(testdb,test)'

\*.log\_archive\_dest\_1='location=/u01/arch1 valid\_for=(all\_logfiles, all\_roles) db\_unique\_name=test'

\*.log\_archive\_dest\_2='service=testdb lgwr async valid\_for=(online\_logfile,primary\_role) db\_unique\_name=testdb'

\*.log\_archive\_dest\_state\_1=ENABLE

\*.log\_archive\_dest\_state\_2=ENABLE

\*.standby\_file\_management='AUTO'EOF

### 8.5复制主库的密码文件到备库

将密码文件orapworcl复制到备库的/u01/app/oracle/product/11.2/db\_1/dbs/下

cd /u01/app/oracle/product/11.2/db\_1/dbs/

scp orapworcl db12:/u01/app/oracle/product/11.2/db\_1/dbs/

### 8.6创建备库相应的目录结构

使用oracle用户创建以下目录，避免权限问题

mkdir -p /u01/oradata/testdb

mkdir -p /u01/app/oracle/admin/orcl/adump/

### 8.7配置主库和备库的监听

使用图形界面配置，采用静态监听

netmgr

### 8.8配置主库和备库的网络服务名

使用图形界面配置，在主备库上均需配置testdb、test两个服务名

netmgr

### 8.9创建备库并启动

创建备库的spfile文件，启动备库到nomount模式

SQL> create spfile from pfile;

SQL> startup nomount

### 8.10RMAN复制主库到备库

首先RMAN连接到目标数据库和辅助数据库

rman target sys/oracle@testdb auxiliary sys/oracle@test

使用RMAN的duplicate命令进行复制，两边目录结构相同，需要添加nofilenamecheck参数s

RMAN> duplicate target database for standby from active database nofilenamecheck;

SQL> select status from v$instance;

### 8.11在备库开启实时日志应用

SQL> alter database recover managed standby database using current logfile disconnect from session;

### 8.12主备库角色状态查询

SQL> select switchover\_status,database\_role from v$database;

--主库显示：TO STANDBY/PRIMARY，如果显示SESSION ACTIVE表示还有活动的会话，需要关闭活动的会话再检查

--备库显示：NOT ALLOWED/PHYSICAL STANDBY

### 8.13测试DG

##执行日志切换测试

在主库端切换归档，在备库检查是否也发生了切换

主库上执行日志切换

SQL> archive log list;

SQL> alter system switch logfile;

SQL> archive log list;

##备库上查看，日志的sequence号也跟着变了

SQL> archive log list;

## 查看备库启动的DG进程

SQL> select process,client\_process,sequence#,status from v$managed\_standby;

PROCESS CLIENT\_P SEQUENCE# STATUS

--------- -------- ---------- ------------

ARCH ARCH 23 CLOSING

ARCH ARCH 0 CONNECTED //归档进程

ARCH ARCH 21 CLOSING

ARCH ARCH 0 CONNECTED

RFS ARCH 0 IDLE

RFS UNKNOWN 0 IDLE

RFS LGWR 24 IDLE //归档传输进程

RFS UNKNOWN 0 IDLE

MRP0 N/A 24 APPLYING\_LOG //日志应用进程

9 rows selected.

## 查看数据库的保护模式

SQL> select database\_role,protection\_mode,protection\_level,open\_mode from v$database;

DATABASE\_ROLE PROTECTION\_MODE PROTECTION\_LEVEL OPEN\_MODE

---------------- -------------------- -------------------- --------------------

PRIMARY MAXIMUM PERFORMANCE MAXIMUM PERFORMANCE READ WRITE

##查看DG的日志信息

SQL> select \* from v$dataguard\_status;

## Open Read Only Standby数据库

以只读方式打开备库，并开启实时日志应用

SQL> shutdown immediate

SQL> startup

SQL> select database\_role,protection\_mode,protection\_level,open\_mode from v$database;

SQL> select process,client\_process,sequence#,status from v$managed\_standby;

SQL> alter database recover managed standby database using current logfile disconnect from session;

##解锁scott用户并添加数据，验证是否同步

在主库解锁scott用户并创建测试表，插入10000行数据。

操作如下

SQL> set line 200

SQL> select username,default\_tablespace,account\_status from dba\_users where username='SCOTT';

SQL> alter user scott account unlock;

SQL> conn scott/tiger;

SQL> show user

SQL> create table test001 (id number(10),name varchar2(20));

SQL> begin

for i in 1..10000 loop

insert into test001 values (1,'ww');

end loop;

end;

/

SQL> commit;

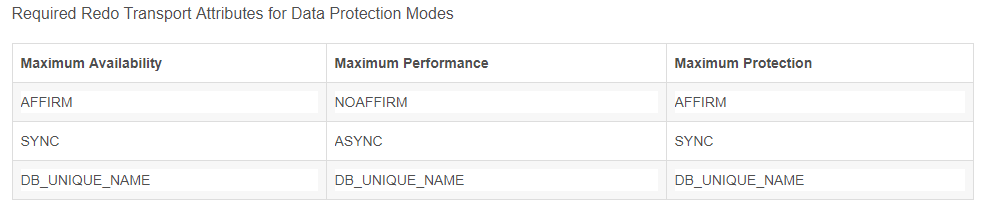
standby端查询scott用户是否解锁，以及test001表是否创建并且插入了10000行数据

SQL> conn scott/tiger;

SQL> select \* from tab;

SQL> select count(\*) from test001;

8.14 DG三种模式



1.最大可用性模式（Maximum Availability）

1）该模式提供了仅次于“最大保护模式”的数据保护能力；

2）要求至少一个物理备库收到重做日志后，主库的事务才能够提交；

3）主库找不到合适的备库写入时，主库不会关闭，而是临时降低到“最大性能模式”模式，直到问题得到处理；

4）优点：该模式可以在没有问题出现的情况下，保证备库没有数据丢失，是一种折中的方法；

5）缺点：在正常运行的过程中缺点是主库的性能受到诸多因素的影响。

2.最大性能模式（Maximum Performance）

1）该模式是默认模式，可以保证主数据库的最高可用性；

2）保证主库运行过程中不受备库的影响，主库事务正常提交，不因备库的任何问题影响到主库的运行；

4）优点：避免了备库对主数据库的性能和可用性影响；

5）缺点：如果与主库提交的事务相关的恢复数据没有发送到备库，这些事务数据将被丢失，不能保证数据无损失。

3.最大保护模式（Maximum Protection）

1）这种模式提供了最高级别的数据保护能力；

2）要求至少一个物理备库收到重做日志后，主库的事务才能够提交；

3）主库找不到合适的备库写入时，主库会自动关闭，防止未受保护的数据出现；

4）优点：该模式可以保证备库没有数据丢失；

5）缺点：主库的自动关闭会影响到主库的可用性，同时需要备库恢复后才能提交，对网络等客观条件要求非常的高，主库的性能会因此受到非常大的冲击。

---------------------

4.4.查询当前模式

SQL> select protection\_mode,protection\_level from v$database;

4.5 切换模式

SQL> ALTER SYSTEM SET log\_archive\_dest\_2='SERVICE=test LGWR SYNC AFFIRM VALID\_FOR=(ONLINE\_LOGFILES,PRIMARY\_ROLE) DB\_UNIQUE\_NAME=test';

SQL> ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE {AVAILABILITY | PERFORMANCE | PROTECTION};

5.DG切换测试

5.1.DG switchover 切换测试

主库------>备库

备库------>主库

主备库角色状态查询

SQL> select switchover\_status,database\_role,open\_mode from v$database;

主库显示：TO STANDBY/PRIMARY，如果显示SESSION ACTIVE表示还有活动的会话，需要关闭会话再检查

显示：NOT ALLOWED/PHYSICAL STANDBY

主库切换到备库

SQL> alter database commit to switchover to physical standby;

SQL> alter database commit to switchover to physical standby with session shutdown;

--如果状态显示SESSION ACTIVE，在切换的时候可以指定with session shutdown 子句强制关闭活动的会话。

SQL> shutdown immediate

SQL> startup mount

备库切换到主库

SQL> alter database commit to switchover to primary;

SQL> alter database open;

主库执行APPLY LOG命令

--启用mount状态下的APPLY LOG

SQL> alter database recover managed standby database disconnect from session;

--启用open状态（READ ONLY WITH APPLY）下的APPLY LOG

SQL> alter database recover managed standby database cancel;

SQL> alter database open;

SQL> alter database recover managed standby database disconnect from session;

SQL> select switchover\_status,database\_role,open\_mode from v$database;

5.2.DG failover 切换测试

主库------>崩溃

备库------>主库

主备库角色状态查询

SQL> select switchover\_status,database\_role,open\_mode from v$database;

主库通过shutdown abort方式人工模拟主库崩溃，直接关闭

SQL> select open\_mode from v$database;

SQL> shutdown abort

SQL> startup mount

SQL> alter system flush redo to 'test';

从库执行如下操作切换为主库

SQL> select thread#, low\_sequence#, high\_sequence# from v$archive\_gap;

--如果没有发现明显的gap现象，说明此次的failover不会有数据损失情况。在备库，要进行关闭apply和结束应用动作。

SQL> alter database recover managed standby database cancel;

SQL> alter database recover managed standby database finish;

SQL> alter database commit to switchover to primary;

SQL> alter database open;

SQL> select open\_mode, switchover\_status from v$database;

检查日志情况的SQL语句

select al.thrd "Thread", almax "Last Seq Received", lhmax "Last Seq Applied"

from (select thread# thrd, max(sequence#) almax

from v$archived\_log

where resetlogs\_change#=(select resetlogs\_change# from v$database)

group by thread#) al,

(select thread# thrd, max(sequence#) lhmax

from v$log\_history

where first\_time=(select max(first\_time) from v$log\_history)

group by thread#) lh

where al.thrd = lh.thrd;