一、Oracle关系数据库

数据库启动

1. 方式一

```
startup nomount # 启动实例
alter database mount; # 在启动例程步骤下装载数据库
alter database open; # 在装载数据库下打开数据库
```

2. 方式二

```
startup mount # 装载数据库
alter database open # 在装载数据库下打开数据库
```

3. 方式三

```
startup open # 打开数据库
```

数据库关闭

1. **`shutdown normal`**

   正常关闭，若关闭数据库的时间没有限制，可以使用此方式。

2. **`shutdown transactional`**

   事物关闭方式，保证当前活动事物提交，尽可能短时间内关闭数据库。

3. **`shutdown immediate`**

   立即关闭，事务没有提交则回退事务并断开连接，并尽短时间关闭数据库。

4. **`shutdown abort`**

   终止关闭，强制断开所有任何数据库操作，可能会丢失一部分数据。

解锁账户

```
ALTER USER USERNAME ACCOUNT UNLOCK

# USERNAME: 用户名
```

二、 Oracle 数据库的体系结构

Oracle数据库从存储结构上可以分为**物理储存结构**与**逻辑储存结构**，从实例结构上可以分为**内存结构**与**进程结构**。

Oracle数据库实例是指一组Oracle后台进程以及在服务器中分配的共享内存区域。

2.1 物理储存结构

Oracle的物理存储结构是由储存在磁盘中的操作系统文件所组成的，Oracle在运行时需要使用这些文件。主要由3种类型的文件组成，分别为数据文件(*.dbf)、控制文件(*.ctl)和重做日志文件(*.log)。

2.1.1数据文件

数据文件是指储存数据库数据的文件

数据文件一般有以下几个特点：

- 一个表空间由一个或多个数据文件组成。
- 一个数据文件只对应一个数据库。而一个数据库通常包括多个数据文件。
- 数据文件可以通过设置其自动扩展参数，实现自定扩展功能。

> Tips: 表空间是数据库储存的逻辑单位，数据文件如果离开了表空间将失去意义，而表空间如果离开了数据文件将失去物理基础。

查看数据字典或表的数据结构可以使用`DESCRIBE` | `DESC`命令

```
desc dba_data_files;
```

2.1.2 控制文件

控制文件是一个很小的二进制文件，用于描述和维护数据库的物理结构。

> Oracle默认创建3个包含系统信息的控制文件，目的是当其中一个受损时，数据库可以调用其它控制文件继续工作。

2.1.3 重做日志文件

重做日志文件是记录数据库中所有修改信息的文件，简称日志文件。

> 提交信息修改后，数据文件中只保留修改后的数据，日志文件既保留修改后的数据，又保留修改前的数据。为日后数据恢复提供数据。

> 在一个日志文件组中，日志文件的镜像数受参数`maxlogmembers`限制，最多可以有5个。

> 数据库的运行分为归档模式和非归档模式，如果是归档模式，则发生日志切换时，日志文件组中的日志信息会先被Oracle系统的后台进程ARCn写入归档日志文件中，然后在被新内容覆盖；如果是非归档模式，则日志文件组中的日志信息将被直接覆盖。

2.1.4 其它文件

1. 参数文件
2. 备份文件
3. 归档重做日志文件
4. 警告、跟踪日志文件

2.2 逻辑储存结构

Oracle 逻辑储存结构主要包括表空间、段、区和数据块。它们之间的关系为：一个数据库有一个或多个表空间组成；一个表空间由一个或多个段组成；一个段由一个或多个区组成；一个区由一个或多个数据块组成。

2.2.1 表空间(tablespace)

表空间是Oracle最大的逻辑储存结构，它与物理上一个或多个数据文件对用。

每个Oracle数据库至少拥有一个表空间，表空间的大小等于构成该表空间的所有数据文件大小的总和。

2.2.2 段(segment)

段是一组盘区，这组盘区组成了被Oracle视为一个单位的数据库对象，例如表或索引。因此，段是一般数据库终端用户将处理的最小存储单位。

按照段中所储存数据的特征，可以将段分为4种类型：数据段、索引段、临时段和回退段。

2.2.3 区(extent)

在Oracle数据库中，区是磁盘空间分配的最小单位，由一个或多个数据块组成。

一个段包含的区的个数并不是无限制的，它由两个参数决定：`minextents`和`maxextents`

2.2.4 数据块(Block)

数据库是用来管理储存空间的最基本单位，也是最小的逻辑储存单位。Oracle数据库进行输入输出操作时，都是以块为单位进行逻辑读写操作的。

2.3 Oracle进程管理

2.3.1 DBWn进程

DBWn(Database Writer，数据库写入)进程，是Oracle中采用LRU(最近最少使用)算法将数据缓冲区中的数据写入数据文件的进程。

> LRU 算法是数据缓存区的一种管理机制，只保留最近数据，不保留旧数据。

2.3.2 LGWR进程

LGRW(Log Writer, 日志写入）进程，是负责管理日志缓冲区的一个后台进程，用于将日志缓冲区中的数据写入磁盘的日志文件中。

> 日志缓冲区是一个循环缓冲区，当LGRW进程将日志缓冲区的日志数据写入磁盘日志文件中后，服务器进程有可以将新的日志数据保存到日志缓冲区中。

LGRW进程将日志信息同步的写入在线日志文件组的多个日志成员文件中。

2.3.3 CKPT进程

CKPT(Check Point，检查点或检验点)进程，一般在发生日志切换时自动产生，用户缩短实例恢复所需的时间。

2.3.4 SMON进程

SMON(System Monitor，系统监控)进程，用于数据库实例出现故障或系统崩溃时，通过联机重做日志文件中的条目将条目应用于数据文件，执行崩溃恢复。

2.3.5 PMON进程

PMON(Process Monitor，进程监控)进程，用于将用户进程出现故障或崩溃是执行进程恢复操作，负责清理内存存储区和释放该进程所使用的资源。

2.3.6 ARCn进程

ARCn(Archive Process，归档)进程，用于将写满的日志文件复制到归档日志文件中，防止日志文件组中的日志信息由于日志文件组的循环使用而被覆盖。只有当Oracle数据库运行在归档模式下是才会产生ARCn进程。

2.3.7 RECO进程

RECO(Recovery， 恢复)进程存在于分布式数据系统中，用于自动解决分布式数据库中出现的事务故障。

2.4 Oracle 内存结构

2.4.1 系统全局区

系统全局区是Oracle为系统分配的一组共享的内存结构，可以包含一个数据库实例的数据或控制信息。

1. 数据缓冲区
2. 日志缓存区
3. 共享池
    - 库缓存：缓存区保存数据库运行的SQL和PL/SQL语句的有关信息
    - 数据字典缓冲

4. 大型池
5. Java池

2.4.2 程序全局区

程序全局区是Oracle系统分配给一个进程的私有内存区域。

2.5 数据字典

数据字典是由Oracle自动创建并更新的一组表，Oracle数据库的重要组成部分，提供数据库结构、数据库对象空间分配和数据库用户等有关信息。所有者为sys，保存在system表空间。

Oracle数据字典是储存在数据库中的所有对象信息的知识库。

四、管理表空间

Oracle表空间：基本表空间、临时表空间、大文件表空间、非标准数据块表空间和撤销表空间。

4.1 基本表空间

4.1.1 创建表空间

```
CREATE [TEMPORARY | UNDO] TABLESPACE tablespace_name
[
  DATAFILE | TEMPFILE 'file_name' SIZE size K|M [REUSE]
  [
    AUTOEXTEND OFF | ON
    [ NEXT number K|M MAXSIZE UNLIMITED | number K|M]
  ]
  [, ...]
]
[MININUM EXTENT number K|M]
[BLOCKSIZE number K]
[ONLINE | OFFLINE]
[LOGGIN | NOLOGGIN]
[FORCE LOGGING]
[DEFAULT STORAGE storage]
[COMPASS | NOCOMPASS]
[PERMANENT | TEMPORARY]
[
  EXTENT MANAGEMENT DICTIONARY | LOCAL
  [AUTOALLOCATE | UNIFORM SIZE number K|M]
]
```

```
# 创建一个永久性表空间myspace
CREATE TABLESPACE myspace
DATAFILE 'E:\app\Administrator\oradata\orcl\myspace.dbf'
SIZE 20M
AUTOEXTEND ON
NEXT 5M
MAXSIZE UNLIMITED;
```

4.1.2 表空间状态属性

1. ONLINE

```
ALTER TABLESPACE tablespace_name ONLINE;
```

2. OFFLINE

```
ALTER TABLESPACE tablespace_name OFFLINE [NORMAL | TEMPORARY | IMMEDIATE |
FOR RECOVERY]
```

3. READ ONLY

```
ALTER TABLESPACE tablespace_name READ ONLY;
```

4. READ WRITE

```
ALTER TABLESPACE tablespace_name READ WRITE;
```

> 无法将Oracle系统自定义的system等表空间状态设置为offline或read only。

4.1.3 重命名表空间

```
ALTER TABLESPACE myspace RENAME TO myspace2;
```

> 如果表空间状态为OFFLINE，则无法重命名表空间。

4.1.4 修改表空间中数据文件的大小

```
ALTER DATABASE
DATAFILE 'file_name'
RESIZE number K|M;
```

4.1.5 增加表空间数据文件

```
ALTER TABLESPACE tablespace_name
ADD DATAFILE
file_name SIZE number K|M
  [
    AUTOEXTEND OFF | ON
    [NEXT number K|M  MAXSIZE  UNLIMITED | number K|M]
    [, ...]
  ]
[, ...];
```

4.1.6 删除表空间的数据文件

```
ALTER TABLESPACE tablespace_name
DROP DATAFILE file_name;
```

4.1.10 删除表空间

```
DROP TABLESPACE tablespace_name
[INCLUDING CONTENTS [ADD DATAFILES]]
```

`INCLUDING CONTENTS`：删除表空间的同时，删除表空间的所有数据库对象。如果表空间中数据库对象，则必须使用此选项。

`ADD DATAFILES`：删除表空间的同时，删除表空间所对应的数据文件。

五、模式对象

模式对象就是储存在用户模式下中的数据库对象，Oracle数据库的模式对象包括表、索引、视图、序列和同义词等。

```
CREATE TABLE [schema.] table_name(
column_name data_type [DEFAULT EXPRESSION]
[ [CONSTRAINT constraint_name] constraint_def]
[, ...]
)[ TABLESPACE tablespace_name];
```

`schema`：指定所属的用户名，或指定所属的用户名模式名称。

六、管理控制文件和日志文件

6.1 管理控制文件

6.1.1 创建控制文件

> 控制文件需在数据库未装载的情况下创建
>
> 即关闭数据库状态，使用`startup nomount`启动实例的状态下创建控制文件。

```
CREATE CONTROLFILE
REUSE DATABASE "database_name"
[RESETLOGS|NORESETLOGS]
[ARCHIVELOGS|NOARCHIVELOGS]
MAXLOGFILES number
MAXLOGMEMBERS number
MAXDATAFILES number
MAXINSTANCES number
MAXLOGHISTORY number
LOGFILE
  GROUP group_number logfile_name [SIZE number K|M]
  [, ...]
DATAFILE
  datafile_name [, ...];
```

- `database_name`: 数据库名
- `RESETLOGS|NORESETLOGS`:表示是否清空日志文件
- `ARCHIVELOGS|NOARCHIVELOGS`:表示日志文件是否归档
- `MAXLOGFILES`:表示最大的日志文件数
- `MAXLOGMEMBERS`:表示日志文件组中最大的成员个数
- `MAXDATAFILES`:表示最大数据文件数
- `MAXINSTANCES`:表示最大的实例个数
- `MAXLOGHISTORY`:表示最大的历史日志文件数
- `LOGFILE`:为控制文件指定日志文件组
- `GROUP group_number`:表示日志文件组标号。日志文件一般以组的形式存在。可以有多个日志文件组。
- `DATAFILE`:为控制文件指定数据文件

使控制文件生效

修改服务器参数文件SPFILE中的control_files参数的值为新建的控制文件，让新的控制文件生效。

新创建的控制文件可以使用数据字典`v$controlfile`来查询

```
SQL> select name from v$controlfile;

NAME
--------------------------------------------------

D:\APP\ADMINISTRATOR\ORADATA\ORCL\CONTROL01.CTL
D:\APP\ADMINISTRATOR\FLASH_RECOVERY_AREA\ORCL\CONTROL02.CTL
```

使用ALTER SYSTEM语句修改control_files参数的值：

```
ALTER SYSTEM SET control_file =
  'D:\APP\ADMINISTRATOR\ORADATA\ORCL\CONTROL01.CTL',
  'D:\APP\ADMINISTRATOR\FLASH_RECOVERY_AREA\ORCL\CONTROL02.CTL'
SCOPE = SPFILE ;
```

如果在创建控制文件使用了`RESETLOGS`选项，则打开数据库时必须使用`RESETLOGS`选项，语句如下：

```
ALTER DATABASE OPEN RESETLOGS;
```

为什么oracle有多个控制文件副本?

为了提高数据的安全性，至少要在数据库建立两个控制文件，并且这两个控制文件最好保存在不同的磁盘中，这样就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，这种管理策略被称为**多路复用控制文件**。

多路复用控制文件是指在系统不同的位置上同时存放多个控制文件的副本，在这种情况下，如果多路复用控制文件中的某个磁盘发生物理损坏导致其所包含的控制文件损坏，数据库将被关闭（在数据库实例启动的情况下），此时就可以利用另一个磁盘中保存的控制文件来恢复被损坏的控制文件，然后再重新启动数据库，达到保护控制文件的目的。

6.2 管理日志文件

6.2.1 创建日志文件组及其成员

1. 创建日志文件组

   ```
   ALTER DATABASE ADD LOGFILE [GROUP group_number]
   (file_name [, ...]) SIZE number K|M [REUSE];
   ```

   - `GROUP goup_number`: 为日志文件组指定编号
   - `file_name`: 为该组创建日志文件成员

- `SIZE number`: 指定日志文件成员的大小
- `REUSE`: 如果创建的日志文件成员已存在，可以使用`REUSE`关键字覆盖已存在的文件。但是该文件不能属于其他日志文件组，否者无法替换。

2. 向日志文件组添加日志文件成员

```
ALTER DATABASE ADD LOGFILE MEMBER
file_name TO GROUP group_number
[, ...];
```

6.2.2 修改日志文件

```
ALTER DATABASE RENAME FILE file_name TO new_file_name;
```

6.2.3 切换日志文件组

```
ALTER SYSTEM SWITCH LOGFILE;
```

6.2.4 清空日志文件组

```
ALTER DATABASE CLEAR LOGFILE GOUP group_number;
```

- 被清空的日志文件不能出于`CURRENT`状态
- 当数据库只有两个日志文件组时，不能清空日志文件组。

6.2.5 删除日志文件组及其成员

1. 删除日志文件

```
ALTER DATABASE DROP LOGFILE MEMBER logfile_name;
```

2. 删除日志文件组

```
ALTER DATABASE DROP LOGFILE GROUP group_number;
```

6.3 管理归档日志

Oracle有两种日志模式：非归档日志模式和归档日志模式。

归档日志模式下，发生日志切换，Oracle会将日志文件通过复制保存到指定的地方。然后才允许向文件中写入新的日志内容。

6.3.1 设置数据库模式

```
ALTER DATABASE ARCHIVELOG | NOARCHIVELOG;
```

九、PL/SQL基础

9.1 PL/SQL

9.1.1 PL/SQL程序块的基本结构

```
[DECLARE declaration_statements ;]
BEGIN
  executable_statements;
[EXCEPTION exception_handing_statements; ]
END ;
/
```

example

```
SET SERVEROUTPUT ON
DECLARE
   emp_number CONSTANT NUMBER(4) := 7900;
   emp_name VARCHAR2(10);
   emp_job VARCHAR2(9);
   emp_sal NUMBER(7, 2);
BEGIN
   SELECT ename, job, sal
   INTO emp_name, emp_job, emp_sal
   FROM scott.emp WHERE empno = emp_number;
   DBMS_OUTPUT.PUT_LINE('查询的员工的编号为: ' || emp_number);
   DBMS_OUTPUT.PUT_LINE('该员工的姓名为: ' || emp_name);
   DBMS_OUTPUT.PUT_LINE('该员工的职位为: ' || emp_job);
   DBMS_OUTPUT.PUT_LINE('该员工的工资为: ' || emp_sal);
END;
/
```

十、储存过程、函数、触发器和包

10.1 储存过程

```
CREATE [OR REPLACE] PROCEDURE procedure_name
[
  (parameter [IN | OUT | INOUT] data_type)
  [, ...]
]
{IS | AS}
  [ declaration_section ;]
BEGIN
  procedure_body ;
END [procedure_name];
```

> 在Oracle的存储过程和函数中，其实IS和AS是同义词，没有什么区别。
>
> 还有在自定义类型（TPYE）和包（PACKAGE）时，使用IS和AS也并没有什么区别。
>
> 但是在创建视图（VIEW）时，只能使用AS而不能使用IS。
>
> 在声明游标（CURSOR）时，只能使用IS而不能使用AS。

10.2 函数

```
CREATE [OR REPLACE] FUNCTION function_name
[
  (parameter [IN | OUT | INOUT] data_type)
  [, ...]
]
RETURN data_type
{IS | AS}
  [ declaration_setion ;]
BEGIN
  function_body ;
END [function_name];
```

十二、用户权限与安全

12.1 用户

12.1 创建用户

```
CREATE USER user_name
IDENTIFIED BY password
[DEFAULT TABLESPACE default_tablespace_name]
[TEMPORARY TABLESPACE temporary_tablespace_name]
[
  QUOTA quota [K|M] | UNLIMITED ON tablespace_name
  [, ...]
  [PROFILE profile_name]
  [PASSWORD EXPIRE]
  [ACCOUNT LOCK | UNCLOCK]
];
```

12.2 修改用户

1. 修改密码

```
ALTER USER user_name INDENTIFIED BY new_password;
```

2. 修改用户口令过期

```
ALTER USER user_name ACCOUNT EXPIRE;
```

3. 修改用户的状态为锁定或解锁

```
ALTER USER user_name ACCOUNT LOCK | UNLOCK
```

12.3 删除用户

```
DROP USER user_name [CASCADE];
```

> 如果用户已经在数据库中创建了内容，则必须指定`CASCADE`关键字，表示删除用户的同时，删除该用户创建的所有内容。

12.2 用户配置文件

12.2.1 创建配置文件

```
CREATE PROFILE profile_name LIMIT
[SESSION_PER_USER number | UNLIMITED | DEFAULT]  # 每个用户可以拥有的会话数
[CPU_PER_SESSION number | UNLIMITED | DEFAULT]  # 每个会话可以占用的CPU总时间，单位百分之一秒
[CPU_PER_CALL number | UNLIMITED | DEFAULT] # 每条SQL语句可以占用的CPU总时间，单位百分之一秒
[CONNECT_TIME number | UNLIMITED |DEFAULT]  # 用户可以连接到数据库的总时间，单位分钟
[FAILED_LOGIN_ATTEMPTS number | UNLIMITED | DEFAULT] # 用户登入数据库允许失败的次数。达到失败数，自动锁定
[PASSWORD_LIFE_TIME number | UNLIMITED | DEFAULT] # 用户口令的有效时间，单位为天
[PASSWORD_REUSE_TIME number | UNLIMITED | DEFAULT] # 用户设置一个失效口令多少天之内不允许重复使用
[PASSWORD_REUSE_MAX number | UNLIMITED | DEFAULT] # 一个已使用口令被重新使用前，口令必须修改的次数
[PASSWORD_LOCK_TIME number | UNLIMITED | DEFAULT] # 用户登入口令错误，账号锁定，用户将被锁定的天数
[PASSWORD_GRACE_TIME number | UNLIMITED | DEFAULT] # 用户口令失效后还允许使用的"宽限时间"
```

12.2.2 使用配置文件

```
ALTER USER user_name PROFILE user_profile;
ALTER SYSTEM SET resource_limit = TRUE;
```

12.2.3 查看配置文件

使用数据字典`dba_profiles`,可以查看系统默认的配置文件`DEFAULT`和自行创建的用户配置文件的参数设置;

```
SELECT * FROM dba_profiles where profile = 'DEFAULT';
```

12.2.4 修改和删除配置文件

1. 修改配置文件

```
ALTER PROFILE user_profile LIMIT parameter value;
```

2. 删除配置文件

```
DROP PROFILE profile_name;
```

12.3 权限

12.3.1 系统权限

1. 授予权限

```
GRANT system_priviliges [, ...] TO
{ user_name [, ...] | role_name [, ...] | PUBLIC}
[WITH ADMIN OPTION];
```

- `PUBLIC`：表示Oracle系统所有用户
- `with admin option`： 此项表示被授予权限的用户可以将该权限再授予其它用户。

查看用户的系统权限可以通过数据字典`user_sys_privs`，三个字段`username`、`privilige`、`with_option`。

2. 撤销权限

```
REVOKE system_privilige [, ...] FROM
{user_name [, ...] | role_name [, ...] | PUBLIC};
```

12.3.2 对象权限

对象权限是指用户对数据库中对象的操作权限。

权限\对象	FUNCTION	PROCEDURE	PACKAGE	SEQUENCE	TABLE	VIEW
ALTER				√	√	
DELETE					√	√
EXECUTE	√	√	√			
INDEX					√	
INSERT					√	√
READ						
REFERENCE					√	
SELECT				√	√	√
UPDATE					√	√

1. 授予对象权限

```
GRANT object_privilege [, ...] | ALL [PRIVILIGES] ON <schema.>object_name TO
{user_name [, ...] | role_name [, ...] | PUBLIC}
[WITH GRANT OPTION];
```

- `schema`: 用户模式
- `object_name`: 对象名称

2. 撤销对象权限

```
REVOKE object_privilige [, ...] | ALL [PRIVILIGES]
ON <schema.>object_name
FROM {user_name [, ...] | role_name [, ...] | PUBLIC};
```

3. 列对象的权限

Oracle中的对象不仅可以针对表或试图等对象，还可以针对表或视图的某系列。不过针对列的权限只有`INSERT`,`UPDATE`与`REFERENCES`.

```
object_privilige (column_name [, ...])
```

4. 与权限相关的数据字典

`user_tab_privs`和`user_col_privs_made`主要用于查询已授予的对象权限信息。

`user_tab_privis_recd`与`user_col_privs_recd`主要用于查询已接收的对象权限信息

12.4 角色

12.4.1创建角色

```
CREATE ROLE role_name [NOT IDENTIFIED | IDENTIFIED BY password];
```

- 默认`NOT INDENTIFIED`,即无口令。

12.4.2 为角色授予权限

同用户授予权限，见12.3

12.4.3 为用户授予角色

同用户授予权限，见12.3

12.4.4 修改用户默认角色

```
ALTER USER user_name DEFAULT ROLE
{
  role_name [, ...]
  | ALL [EXCEPT role_name [, ...]]
  | NONE
};
```

12.4.5 管理角色

1. 禁用与启用角色

```
SET ROLE
{
  role_name [IDENTIFIED BY password]
  [, ...]
  | ALL [EXCEPT role_name [, ...]]
  | NONE
};
```

- `IDENTIFIED BY`： 启用角色，为角色提供口令。
- `ALL`: 启用所有角色。要求所有角色不能有口令。
- `EXCEPT`: 启用除某些角色外的所有角色。
- `NONE`: 禁用所有角色。

12.4.6 与角色相关的数据字典

`user_role_privs`: 用于查询授予用户的角色

`role_sys_privs`： 用于查询授予角色的系统权限

`role_tab_privs`： 用于查询授予角色的对象权限