Oracle 体系架构

非 CDB 形式：oracle 体系架构即为 oracle 服务器,服务器由例程和数据库构成，机器通过例程访问数据库。例程是访问数据库的一种方式，包含内存和进程。其中内存分为 SGA（系统全局区）和 PGA（程序全局区）两个，SGA 是内存启动时分配给 oracle 的内存，是会话创建时分配给这个会话专用的内存，同一会话内部可以共享，SGA 基本组件有：共享池（包括数据字典和 library cache）、数据操作缓存、重做日志缓存。PGA 指被某个程序转有的，有用户进程、服务器进程。进程分为三类：用户进程、服务器进程（专用服务器进程、共享服务器进程）、后台进程。

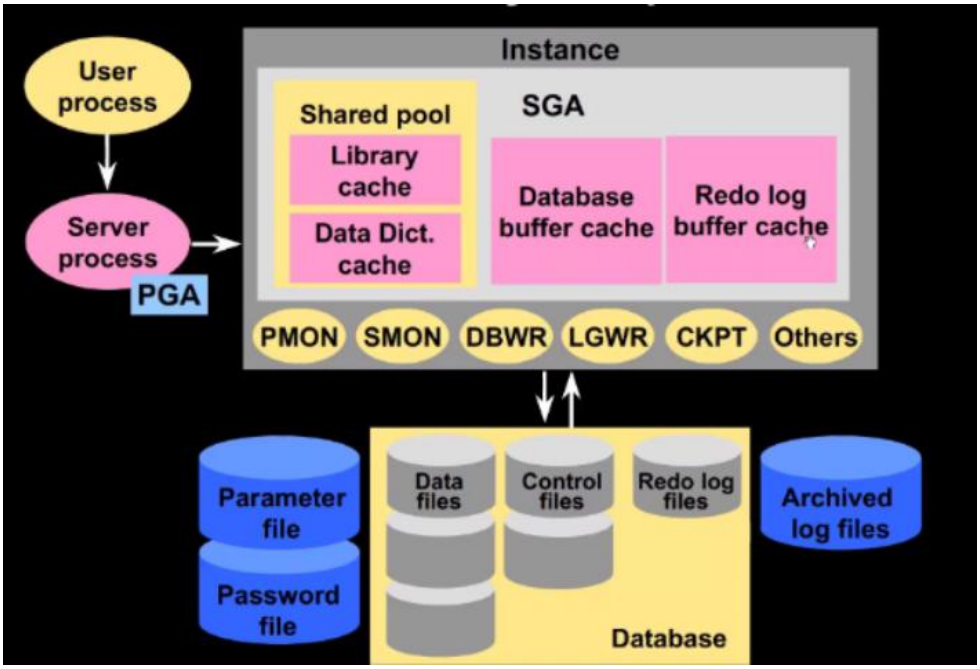
用户进程：用户界面， UI 提供一个访问 oracle 的界面，用户进程无法直接访问 oracle 服务器，用户进程访问服务器要完成第一个名称解析

服务器进程：用户进程的代理，用户进程提出请求，服务器执行，用户进程通过监听找到服务器进程；

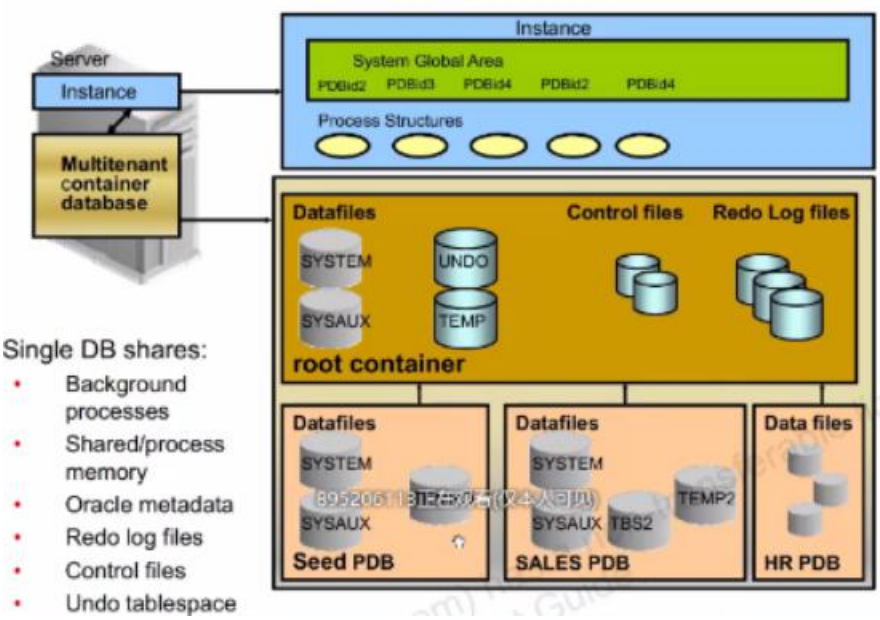
后台进程：后台处理的进程。 后台进程有 5 个是必须的：SMON 系统监视器进程、PMON 进程监视器进程、DBWR 数据库书写器、LGWR 日志书写器、CKPT 检查点进程。

数据库是文件集合，控制文件、数据文件、重做日志文件为 oracle 内部文件（oracle 系统进行管理），初始化参数文件、口令文件、归档重做日志文件为 oracle 外部文件（由管理员管理）

oracle 数据库是一个被统一处理的的数据的集合，从物理角度来看包括三类文件数据文件，控制文件，重做日志文件。从逻辑角度来看，oracle 数据库至少包含一个表空间，表空间至少包含一个段，段由区做成，区有块组成。需要注意的是表空间可以包含若干个数据文件，段可以跨同一个表空间的多个数据文件，区只能在同一个数据文件内。



CDB 架构：例程和数据库之间添加一个容器 CDB，用户可以直接管理用户数据，不用管理系统。例程是共享的，内存共享，（用语言描述图片）



检查点

(1)作用：

1. 同步所有的数据文件，数据文件开头的检查点是一样的说明是同步
2. 同步所用的控制文件；如果检查点号太小了，会报错控制文件太旧了
3. 发送信号通知进程 DBWR 写盘（系统崩溃了就恢复最近的检查点）

（2）检查点间隔：长：性能好，但是出现问题后恢复时间长 ;短：恢复时间端，但是性能会变差

（3）回滚段 作用 Rollback 1.读一致性 2.回退 3.闪回恢复

SQL 语句（简答）

1. QL(select)（会写语句）

2.数据操作语言 DML(insert,update,delete,merge) 修改的是用户数据

3.数据定义语言 DDL(create,alter,drop, truncate,rename,comment)修改的是系统数据

4.数据控制语言 DCL(grant,revoke) 用于数据库授权、角色控制等管理工作

5.事务控制语言 TCL(commit,rollback,savepoint) 用于数据库的事务管理

null 的三值现象：未定义的值，未初始化，不为 0，不为空格

别名可以直接跟在列名后，也可以加 as（名称可以不加引号或加双引号，数字开头要加双引号如“123 学号”）；双引号用于名称，字符或字符串单引号

1. QL（可能性大）：考查查询、排序

1）与用户**交互**：&，define，&&

```
//&举例，语句多次出现&时需要输入多次
select eno from emp where ename=&姓名;  //姓名是提示符，会提示 Enter value for 姓名：但要求太高，输入需要有’’且大小写不能区分
改进：select eno from emp where ename=UPPER(’ &姓名’ );//不需要加引号，输入全部自动转化为大写
//define 例子，
define gh = 7369;
select empno.ename,sal from emp where    empno =&gh;这时会直接使用 gh 的值进行查询
undefine gh;  //取消定义 undefine
//&&例子
&&a 表示接受用户输入的值并定义 a 为用户输入
```

2）**符号**:运行可以带上+、-、*、/,注意:+不表示字串连接(3+’ 4’ =7),||表示字串连接(3||’ 4’ =34);

单引号嵌套：’ I’’ am a student = ‘I’ am a student. 或者 q[];escape 常与 like 用，name like ‘AB__’ escape ‘\’ 表示形如 AB__的字符，否则只为 AB，表示其后所跟的字符就是器本意；

3）**函数：** select LOWER(‘AAA’) FROM dual;

LOWER,UPPER,INITCAP（首字母大写），**LENGTH,TRIM,REPLACE,**

CONCAT：字串连接，多于 2 个字串连接时可以嵌套 CONTACT(‘A’,CONTACT(‘B’,’ C’))或者使用||连接；

SUBSTR（oracle 开始位置为 1 不是 0）：SUBSTR(‘helloworld’,1,5）=hello,//表示从 1 开始，取 5 个，-2 则从倒数第二个开始 ；

INSTR：在字串中的位置,例 INSTR（“helloworld”,’w’,从字符串的第几个开始，第几次出现）；

LPAD("1234",10,'*') => "******1234" 中间的参数表示填充后字符串 一共几位；

RPAD("1234",10,'*') => "1234******"；**可以用于层次查询时：**

select LPAD(“ ”,(level-1)*2)||ename ename,level from emp start with empno=7566 connect by prior emp=mgr; //查询工号为 7566 的所有下属关系，需要改成 7566 的所有上司则修改为 mgr = emp;子键在前，父键在后

ROUND(45.9565,-1)=50,ROUND(44.656,-1)=40，ROUND(45.89565,1)=45.9 四舍五入；

TRUNC(45.9632,2)=45.96,TURNC(45.333,-1)=40 截断 ；

CEIL 向上取整,**FLOOR** 向下取整,**MOD** 取余

MONTHS_BETWEEN(d1，d2）d1,d2 之间相差多少月，d1 写较近日期

ADD_MONTHS(d1，n）d1 加上 n 个月的日期

NEXT_DAY(d1,‘Friday’)d1 之后的下一个星期五的日期

LAST_DAY(d1)

ROUND(SYS,’ MONTH’)= 01-AUG-95 //SYS=25-JULY-95

TRUNC(SYS,’ MONTH’)=01-JUL-95 //SYS=25-JULY-95

4）nls 参数

sysdate/current_date 系统日期

systimestamp/current_timestamp（n）时间戳可以选择位数

设置日期格式：alter session set nls_date_format=’ yyyy-mm-dd hh24:mi ’ ;或者 hh:mi am/pm

YY 格式：和当前日期处于同一世纪；**RR 格式**：数据库默认，接近系统日期的那个世纪

5）TOP N 功能

a. 12c 之前通过内嵌视图实现，前三位工资：select empno,ename,sal from(select * from emp order by sal desc) where rownum<=3;

b. 12c 使用 row limiting 子句

（1）**fetch first n rows only** 子句，只取前 n 行，不考虑 n+1 行数字

```
select empno,ename,sal from emp order by sal desc fetch first 3 rows only;
```

（2）**with ties** 子句，若前 n 行之后的行和第 n 行条件相同也会被 select

```
select empno,ename,sal from emp order by sal desc fetch first 3 rows with ties;
```

（3）**offset** 子句:偏离几行之后再取出几列，例子取第 3、4、5 行：

```
select empno,ename,sal from emp order by sal desc offset 2 rows fetch first 3 rows only;
```

6）转换函数

to_char 加了 fm 在 yyyy-mm-dd 之前时会去掉 01 的前导 0，即为 1998-1-1

to_date

7）if-then-else 逻辑

（1）case 语句

```
CASE expr WHEN situation1 THEN action1
     [WHEN situation2 THEN action2
      WHEN situation3 THEN action3]
```

（2）decode 函数

```
DECODE（需要判断表达式，     'situation1',  result1
           'situation2',  result2
           'situation3',  result3
           均不满足的 action ）
```

8）连接

（1）内部连接

1. select b.buyer_id,b.buyer_name,s.qty from buyers b,sales s

where b.buyer_id = s.buyer_id

2. emp 10万员工

dept 4个部门（10，20，30，40） from emp,dept

nested loop 嵌套循环

3.select b.buyer_id,b.buyer_name,s.qty from buyers b inner join sales s

on b.buyer_id = s.buyer_id (inner可省)

（2）外部连接

1.select b.buyer_id,b.buyer_name,s.qty from buyers b,sales s

where b.buyer_id = s.buyer_id（+） 外部连接（最好用内部连接）

2.select b.buyer_id,b.buyer_name,s.qty from buyers b outer join sales s

on b.buyer_id = s.buyer_id

3.select b.buyer_id,b.buyer_name,s.qty from buyers b full outer sales s

on b.buyer_id = s.buyer_id

（3）多表：

1. select b.buyer_name,p.prof_name,s.qty from buyers b,sales s,product p where b.buyers_is = s.buyer_id and p.prof_id = s.prof_id

2. select b.buyer_name,p.prof_name,s.qty from buyers b join sales s

on b.buyers_is = s.buyer_id join product p

on p.prof_id = s.prof_id

（4）自连接

select a.buyer_id as buyer1,a.prof_id,b.buyer_id as buyer2 from sales a,sales b

where a.prof_id = b.prof_id and a.buyer_id = b.buyer_id

会查出一样的数据（镜像）

select a.buyer_id as buyer1,a.prof_id,b.buyer_id as buyer2 from sales a,sales b

where a.prof_id = b.prof_id and a.buyer_id < b.buyer_id

（5）交叉连接

select b.buyer_name,s.qty from buyers b,sales s;

select b.buyer_name,s.qty from buyers b cross join sales s;

select e.first_name||' '||e.last_name as 职工姓名,m.first_name||' '||m.last_name as 汇报 经理 from employees? ? ?

9） SQL执行步骤

a. SQL 正文放入共享池（shared pool）的库缓存（library cache）

b. 检查是否有相同的 SQL 正文，没有就进行以下编译处理，否则跳过。

a) 语法检查 b)通过数据字典检查表和列的定义 c）对所操作的对象加编译锁，防止编译期间的对象定义被改变 d)检查用户权限 e）生成执行计划 f）将编译后的代码和执行计划放入共享SQL区

c. 执行：由服务器进程执行SQL语句

d. 提取数据：由服务器进程选择所需的数据行，需要时排序（PGA中），返回给用户进程

与Oracle服务器连接

1） **专用服务器连接：**一个用户进程对应创建一个服务器进程，用户进程与服务器进程是一对一的关系

2） **共享服务器连接：**多个用户同时对应一个服务器进程

各种不同的连接方式

1） **基于主机方式：**用户进程与服务器进程运行在同一台计算机相同的操作系统下，用户进程与Oracle 服务器的通信 是通过操作系统内部的进程通信（inter process communication，IPC）机制来建立的。

2） **客户端-服务器（两层模型）方式：**用户进程与 Oracle 服务器的通信是通过网络协议（如 TCP/IP）来完成的

3） **客户端-应用服务器-服务器（client-application server-server）（三层模型）方式：**用户的个人计算机通过网络与应用服务器或网络服务器进行通信，该应用服务器或网络服务器同样再通过网络与运行数据库的计算机连接。

手工创建数据库(demo)

1. 创建 oracle 的服务 oracleservicedemo，要在管理员权限下进行

oradim -new -sid demo

输入 oracle 服务用户口令：*****

2. 将当前进程设为 demo

set oracle_sid=demo

sqlplus sys/admin as sysdba

3. 创建/编辑 初始化参数文件（二进制文件不可修改）

create pfile from spfile; --spfile->pfile

F:/app/oracle/product/18.3.0/database/INITDB18C.ORA 右击粘贴 改名 INITdemo.ORA 内容修改：查找全部 替换 db18c->demo

4. 创建（初始化参数文件中出现的）相应的目录结构:新建对应的文件夹

5. 启动例程（仅有初始化参数文件） SQL>startup nomount

6.创建数据库，执行创建数据库的语句：

select name from v$datafile;得到 datafile 的路径 create database demo datafile 'F:\app\luyao\oradata\demo\system01.dbf' size 400m

sysaux datafile 'F:\app\luyao\oradata\demo\sysaux01.dbf' size 400m

undo tablespace undotbs1 datafile 'F:\app\luyao\oradata\demo\undotbs01.dbf' size 50m default temporary tablespace temp tempfile 'F:\app\luyao\oradata\demo\temp01.dbf' size 400m;

logfile

group 1 ('F:\app\luyao\oradata\demo\redo01.log') size 10m,

group 2 ('F:\app\luyao\oradata\demo\redo02.log') size 10m,

group 3 ('F:\app\luyao\oradata\demo\redo03.log') size 10m;

数据库是文件的集合：

控制文件：整个数据库的结构，创建数据库的时候回自动创建控制文件。

联机重做日志文件：一个暑假里面至少需要两个日志文件组

数据文件：创建数据库时至少创建三个表空间：system，sysaux，undo tablespace 其他表空间可以创建完 空库之后创建

7. 创建数据字典视图

catalog.sql 创建数据字典视图 sysdba 运行脚本

/rdbms/admin/catalog --自动创建数据字典

desc user_tables

8. 注册表编辑器 oracle/key_Oradb18cHome 默认 db18c

9.创建 spfile

show parameter spfile --无

spfile create spfile from pfile;

10. 创建口令验证文件

18.3.0/database/pwddb18c.ora

exit

orapwd file=F:\app\oracle\product\18.3.0\database\pwddemo.ora password=admin1#23

11. 创建 oracle 的内部包：所有的包都不存在，调用脚本 desc row_id……@?/rdbms/admin/catproc

12. 创建 scott 方案、对象：

18.3.0/rdbms/admin/scott.sql sqlplus / as sysdba @?/rdbms/admin/scott

alter user scott identified by tiger; --改密码

--测试

conn scott/tiger

调用脚本 utlsample.sql

13. **加载产品概要信息，先联 system/manager,加载脚本：**

conn system/manager --默认密码是 manager

@?/sqlplus/admin/pupbld.sql

14. **配置监听器（服务器端）和服务名（客户端），**

net manager 再原有 Listener 添加数据库，主机名 重启监听：

lsnrctl stop，再 start 配置服务名 全 demo

15. **配置 em express**

conn sys/admin as sysdba;

select dbms_xdb.gethttpport from dual;

execute dbms_xdb.sethttpport(6789);

select dbms_xdb.gethttpport from dual;

http://LAPTOP-5U5N2LPC:6789/em

16. **一天后重新使用数据库**

set oracle_sid = demo

sqlplus / as sysdba --报错是协议适配器错误，原因是服务没有启动，在 cmd 启动 demo 数据库 startup --数据库启动

管理控制文件（2 进制文件）v$controlfile/.ctl 文件：

1) **查看控制文件的名称和位置**: show parameter control files;或者 select name from v$controlfile;

2) **查看控制文件内容**: oradebug

   a. 包括数据库的名称和标识 select dbid, name from v$database;
   b. 数据库创建时间:
   c. 表空间名字（逻辑结构）
   d. 物理结构包括：日志文件和数据文件的名称地址,
   e. 当前子序列号，检查点信息，归档日志信息，备份信息，撤销的开始结束

3) oradebug 转储成文本文件

oradebug setmypid

oradebug dump controlf 3

4) **查看文件路径**: oradebug tracefile_name

5) **控制文件的内容**

6) **控制文件多路复用**(多个控制文件在不同地方)例子为已有两个第三为复用的

   a. 若 spfile 没启动，需要 shutdown immediate;
   b. SQL>alter system set control_files = '$HOME/ORADATA/u01/ctr101.ctl','$HOME/ORADATA/u01/ctr102.ctl' ,'/u01/demo/control103.ctl' SCOPE=SPFILE;
   c. SQL>shutdown immediate
   d. cp $HOME/ORADATA/u01/ctr101.ctl control103 --拷贝文件
   e. SQL>startup

7) 创建控制文件

   a. alter database backup controlfile to trace as '/u01/demo/create_con.txt';
   b. CREATE CONTROLFILE REUSE DATABASE "DEMO" NORESETLOGS NOARCHIVELOG
      MAXLOGFILES 16 MAXLOGMEMBERS 2 MAXDATAFILES 30 MAXLOGHISTORY 292
      LOGFILE
        GROUP 1 '/u01/ctr101.log' SIZE 10M BLOCKSIZE 512,
        GROUP 2'/u01/ctr101.log' SIZE 10M BLOCKSIZE 512,
      DATAFILE
        '/u01/oracle/oradata/demo/sys01.dbf',

'/u01/oracle/oradata/demo/sys01.dbf'
CHARACTER SET US7ASCII 16;

8) 删除数据库控制文件

a. !rm control01.ctl
b. !rm control02.ctl
c. !rm control03.ctl

9) **组里多个成员是镜像关系 循 环 工 作** :select group#,sequence#,status from v$log; :select group#,members from v$logfile;

8）**闪回数据库无法闪回的原因**：闪回数据库需要用到闪回日志，闪回日志文件在控制文件中，控制文件重建后，闪回日志会丢失则数据库后无法闪回

管理日志文件：归档和非归档模式

0） 由组的形式管理，数据库至少有两个组，每个在组里的日志文件被称为成员。分为联机重做日志文件和归档日志文件，组里至少有一个成员

1） **日志文件工作方式：**循环，每个组里的成员是镜像的关系，切换是写满一个文件 LGWR 自动指向下一个文件（日志切换时信息会写入控制文件中，会做检查点）

2） **查看日志文件组信息**（v$log）：select GROUP#,MEMBERS,SEQUENCE#,STATUS from v$log;

3） **查看日志文件成员**(v$logfile): desc v$logfile; select GROUP#,MEMEBER from v$logfile;

4） 创建

a. **添加日志文件组**:alter database add logfile group 4 ('F:\oracle\oradata\sales\redo04.log') size 10m;
b. **添加文件组成员** alter database add logfile member 'F:\oracle\oradata\sales\redo04a.log' to group 4;

5） 删除

a. **删除日志文件组**（只有两个组不能删，组状态为 current 和 active 的组不能删，删除组之后组内文件需要手动再删除）
   a) 修改当前：alter system switch logfile;
   b) ADCTIVE----->INACTIVE: alter system checkpoint
   c) 删除： alter database drop logfile group 1;
   d) 删除日志文件：!rm /u01/app/oracle/oradata/redo1.log（物理删除）
b. **删除日志文件成员 (**不能删除当前组成员，不能删除只有一个成员的组里成员) alter database drop logfile member '/u01/oradta/redo2.log';

6） **OMF 可以自动管理**（不用手动删除）：组被删掉后文件会自动删除

a. **数据文件需要初始化参数**：show parameter db_create_file_dest ='/u01/omf' ;之后正常操作
b. **日志文件会默认 OMF**，默认在闪回区（/recovery_area/数据库名/onlinelog/xxxx.log）
   修改默认 alter system set db_create_online_log_des_1/2/3='指定路径'（日志文件的 OMF 可以设置多个），日志文件复用可以有两/三个（经验认为只需要副本两三份，多个也会丢）

7) **日志文件出问题时**，修复，清除：

c. 组：alter database clear logfile group n
d. 成员：alter database clear logfile '/u01/oracle/oradata/redo.log'

8) 设置数据库日志模式

a. 查看当前模式 select log_mode from v$database;
b. **归档或非归档切换** alter database (no)archivelog;**(必须在 mount 阶段 shutdown immediate---startup mount)**

9) **查询归档日志信息**(v$archived_log)

10) **设置归档目的地手工归档**:

a. show parameter log_archive;

b. 本地归档两份 alter system set log_archive_dest_1='location=/u01/arch/1';
alter system set log_archive_dest_2='location=/u01/arch/2';

c. 归档到服务器 alter system set log_archive_dest_1='service/salse/u01/1';

d. 手动归档 archive log list（找到下一个需要归档的序列号）alter system archive_log_current;（归档） select name from v$archived_log;（查询归档后的信息）

11） **移动/重命名日志文件**(mount 阶段):

select member from v$logfile; alter database rename '原文件路径' to '现文件路径:alter database open;

12） 修复日志文件

a. **非当前日志文件**:alter database **clear** logfile group 1; alter database open;

b. 当前日志文件 :update scott.emp set sal=900 where empno=7698; select group#,status from v$log; startup mount ;**recover database until cancel; :alter database open resetlogs;**

表空间和数据文件

1. **数据库存的结构层次：**

数据库-表空间-**物理**上为：数据文件、段（存储结构）、区（oracle 最小的空间分配单位）、块（oracle 最小的 io 单位。不能小于操作系统块，一般为其整数倍）---操作系统块

2. **查段有哪些类型：**desc dba_segments

select unique segment_type from dba_segments

一个数据库由多个表空间构成，一个表空间只能属于一个数据库

一个表空间由一个或多个数据文件构成，按顺序访问则大的数据文件比较好，否则小的好

读写的最小的单位是块，哪怕访问，也是把包含这条记录的块掉进内存

--块和区 和软妹币面值差不多

--块的大小 访问的数据量多，块大比较好，减少 IO 访问的数据量少，块小比较好

3. 创建 users 表空间并设为数据库默认表空间

create tablespace users datafile'F:/app/oracle/oradata/demo/users01.dbf' size 20m;

alter database default tablespace users;

查看修改后的结果(默认表空间)

col property_name for a50

select property_name,property_value from database_properties;

select tablespace_name from dba_tables where table_name='EMP';

alter table scott.emp move tablespace users;//将表移到其他表空间

4. 创建一个由 2k 的块组成的表空间

create tablespace smalltbs datafile'F:/app/oracle/oradata/demo/small01.dbf' size 10m blocksize 2k;--表空间块大小 2k 不匹配 内存中没有地方可以放 2k 的块

show parameter cache -- db_2k_cache_size value=0

alter system set db_2k_cache_size = 16m; --在内存开辟一个 16m

create tablespace smalltbs datafile'F:/app/oracle/oradata/demo/small01.dbf' size 10m blocksize 2k;

create tablespace bigtbs datafile'F:/app/oracle/oradata/demo/big01.dbf' size 10m blocksize 16k; --创建大块

5. 表空间的空间管理：

(1) **本地管理**：有关区可用或者不可用的信息存储在数据文件，每个数据文件的头部都放了一个位图，就是 01 组成的 图，0 表示可用，1 表示区已经被分配

(2) 数据字典管理

desc dba_tablespaces 表空间

select tablespace_name,extent_management from dba_tablespaces;//查看是本地管理还是数据字典

create tablespace userdata datafile'F:/app/oracle/oradata/demo/userdata01.dbf' size 10m extent management dictionary;//创建数据字段管理的表空间

本地管理优点：1.减少对数据字典表的增用 2.增加或者减少空间需要修改数据字典，会有回滚 3.不用碎片的合并 4.每个区是一样大的

6. **表空间类型：常规表空间**（可读可写）permanent；**撤销表空间**（放回滚数据）undo 回滚段在这个表空间；**临时表空间**（用来排序）temporary 内存不够在这个空间排序

desc dba_tablespaces

select tablespace_name,contents from dba_tablespaces; --查看表空间类型
show parameter undo_tablespace; --undo 为当前表空间

--**创建撤销表空间 undotbs2 且设为数据库默认的撤销表空间**
create undo tablespace undotbs2 datafile 'F:/app/oracle/oradata/demo/undotbs02.dbf' size 20m;
alter system set undo_tablespace=undotbs2;
select tablespace_name,contents from dba_tablespaces; show parameter undo_tablespace;

--**创建临时表空间 temp2 并设为数据库默认的临时表空间**
create temporary tablespace temp2 tempfile 'F:/app/oracle/oradata/demo/temp02.dbf' size 20m;
alter system default temporary_tablespace temp2; show parameter default_temporary_tablespace;
select tablespace_name,contents from dba_tablespaces;

7. **表空间状态**：联机可读写 online，只读，脱机 offline
--**将表空间改为只读**：修改完成后，不可对表进行插入记录/删字段，可查+增加字段+删表
alter tablespace smalltbs read only;
select tablespace_name,status from dba_tablespaces;--查询表空间状态
alter table scott.test1 add birth_date date; --可增加字段
alter table scott.test1 drop column birthdate; --不可删除某一字段
drop table scott.test1; --可删表 不能脱机的 3 个表空间：1)system --系统数据一定要用 2)当前的撤销表 3)临时表空间

--**其他表空间都是可以脱机的，临时文件可以脱机**
alter tablespace system offline;--除非 shutdown，系统表空间不可脱机
alter tablespace sysaux offline;--脱机
alter tablespace sysaux online; --连接
alter tablespace undodbs1 offline; --非当前的撤销表空间-->online
show parameter undo_tablespace --显示的 value 值为当前的撤销表空间
alter tablespace temp offline; --只可对临时文件脱机，不可对临时表空间脱机
alter tablespace smalltbs read write;--只读->读写

8. 删除表空间
删除表空间后，对应的数据文件还在，**要手动删除，当表空间有内容不可删**
drop tablespace smalltbs;--
select name from v$datafile;
drop tablespace bigtbs; --不可删，**当表空间有内容不可删**
drop tablespace bigtbs including contents and datafiles;--内容+数据文件一块删掉

9. OMF
show parameter db_create
db_create_file_dest --初始化参数
alter system set db_create_file_dest=F:/app/omf/; --设置默认的创建文件路径
create tablespace test 缺 datafile/tempfile --没有上面的操作就不成功，因为不知道在哪里创建
drop tablespace test; --数据文件自动被删除

10. 扩展表空间大小
(1) **手动扩展**：alter database datafile'……' resize 200m;
(2) **自动扩展**：alter database datafile'D:\……\demo\small01.dbf' size 200m autoextend on next 10m maxsize 500m;
(3) **增加新数据文件**：alter tablespace app_data add datafile'……' size 200M;

11. 数据文件的移动或重命名
select name from v$datafile;--所有的数据文件在哪
--移到 F:/app/omf/下
alter database move datafile 'D:app/oracle/oradata/demo/userdata01.dbf' to'D:app/omf/data01.dbf';
--物理上也移动了，原来文件没有了，相当于移动
--copy,加 keep 原来文件还有
alter database move datafile'D:app/omf/data01.dbf' to 'D:app/oracle/oradata/demo/userdata01.dbf' keep;

表空间练习

1.创建 **users** 表空间，并设为数据库默认的永久表空间；**database_properties** 视图

create tablespace users datafile'F:/app/luyao/oradata/demo/users01.dbf' size 20m; alter database default tablespace users;

select property_name,property_value from database_properties;

2.创建一个由 **4k** 的块组成表空间 **test**（**test01.dbf 10m**）

create tablespace test datafile'F:/app/luyao/oradata/demo/test01.dbf' size 10m blocksize 4k;

alter system set db_4k_cache_size=16m;

create tablespace test datafile'F:/app/luyao/oradata/demo/test01.dbf' size 10m blocksize 4k;

3.向表空间添加一个 **10m** 的数据文件**(test02.dbf)**，将 **test01.dbf** 修改为 **15m**

alter tablespace test add datafile'F:/app/luyao/oradata/demo/test02.dbf' size 10m; alter database datafile 'F:/app/luyao/oradata/demo/test01.dbf' resize 15m;

4.移动 **test01.dbf**

select name from v$datafile;

alter database move datafile '/u01/oradata/demo/test.dbf' to '/app/test.dbf';

alter database move datafile '/app/omf/test.dbf' to '/app/luao/oradata/test.dbf'

5.在 **test** 表空间内创建一张表 **table1(insert)**

create table table1(id int,name varchar(20)) tablespace test;

insert into table1 values(1,'Tom');

commit;

6.将 **test** 表空间改为 **read only** alter tablespace test read only;

7. 删除表 **table1(dml,create table,alter)** drop table table1;

8. 将表空间改为 **read write** alter tablespace test read write;

9. 删除 **test** 表空间：检查数据文件是否被删除：未删除，要加上 including……

drop tablespace test;

10. 使用 **OMF** 创建表空间：检查数据文件是否被删除:已删除

show parameter db_create,初始化参数 db_create_file_dest

create tablespace test //缺 datafile/tempfile

alter system set db_create_file_dest='F:/app/omf/';

create tablespace test;

drop tablespace test;

11. 创建 **1** 个撤销表空间 **undotbs2**，并把它设为系统当前的撤销表空间

create undo tablespace undotbs2 datafile 'F:/app/luyao/oradata/demo/undotbs02.dbf' size 20m;

alter system set undo_tablespace=undotbs2;

show parameter undo_tablespace;//看系统默认的撤销表空间

select tablespace_name,contents from dba_tablespaces;//看表空间类型

12. 创建临时表空间 **temp2,**并把它设为数据库默认的临时表空间

create temporary tablespace temp2 tempfile 'F:/app/luyao/oradata/demo/temp2.dbf' size 20m;

alter database default temporary tablespace temp2;

select property_name,property_value from database_properties;

select tablespace_name,contents from dba_tablespaces;//看表空间类型

13. 没有备份的恢复（归档模式）

(1) 创建一个表空间 tbs1(tbs1.dbf)
(2) 在 tbs1 表空间内创建一张表 t1(insert into)
(3) shutdown immediate
(4) 手工删除表空间 tbs1 的数据文件
(5) startup
(6) 将数据文件 tbs1.dbf 脱机
(7) alter database open;
(8) alter database create datafile 'path/tbs1.dbf'
(9) recover datafile 'path/tbs1.dbf'
(10) 将数据文件 tbs1.dbf 联机
(11) 检查数据是否恢复

ORACLE 安全（3A：验证授权审核）

1、验证

(1) 用户分类（sys、non-sys 验证方式不同，sys 用户密码不在数据库中）

(2) 验证方式

- **操作系统验证**（默认） sqlplus asda/asdasd as sysdba 或者 sqlplus / as sysdba 连到数据库上，完全信操作系统。方便但是不安全，不使用密码就可以连接
- **口令文件验证**（密码一定要正确）

创建口令验证文件 F:\app\oracle\product\18.3.0\database\PWDdb18c.ora

show parameter password **初始化参数** remote_login_passwordfile:（NONE：禁止使用口令验证文件; EXCLUSIVE：启用口令验证文件，独占，单例程多用户，只可从一个例程（结点）连，可以有很多用户，哪些用 户可用参考 v$pwfile_users; desc v$pwfile_users; select username,sysdba from v$pwfile_users; grant sysdba to scott;授权

SHARED：启用口令验证文件，共享，多例程单用户，只可用 sys 用户连接其他用户不行）

启用口令验证文件时，还可使用 os 验证：操作系统验证和口令文件验证同时允许时，**优先 OS 验证**。文件 F:app/oracle/product/18.3.0/network/admin/sqlnet.ora 中 sqlnet.suthenticaltion_servise=(NTS)、**不允许 OS 验证改为(NONE)**

F:app/oracle/product/18.3.0/database/PWDdemo.ora //用命令创建口令验证文件

orapwd file=F:app/oracle/product/18.3.0/database/PWDdemo.ora password=admin1#3 force=y;--覆盖已有的（以后 sys 密码为 admin1#3） force=y 表示强行覆盖

--之后发现没有生效，还是可以无密码连接 --原因：oracle home 的 network admin 文件默认 SQLNET.AUTHENTICATION_SERVICES = (NTS)这个表 示启用 os 验证，要是两个一起，os 验证优先改为（NONE）就可以了

（3）非 sys 用户/普通用

- **数据库验证**： desc user$ select name,password from user$;
- **口令放在数据库中** create user nit identified by admin; //新建一个用户，放到表中
- **操作系统（外部）验证**： a)初始化参数：os_authent_prefix（默认时 ops$） show parameter os b.)创建操作系统用户：右击计算机--点管理-- 本地用户和组 --用户-- 用户名 os1 --创建；c)在数据库中创建对应的用户：create user ops$os1 identified externally; d)赋予相应的权限:grant create session to ops$os1;//赋可以登陆的权限 e）windows 注册表修改：regedit ：hkey_local_machine/software/oracle/key-oradb18home1/增加 osauth prefix domain 值改为 false；f)以操作系统用户 os1 登录 cmd:runas /? runas /user:os1 cmd //以 os1 用户登录运行 cmd 程序 输入密码:os1 弹出窗口； g）在窗口敲 sqlplus /； conn / 用户名口令不对=>注册表

2、授权

(1) **系统权限**（全局，用户）

(2) **对象权限**（局部，资源），

特征：全局局部（系统权限可以访问所有的表，对象权限只访问局部）；用户资源

(3) 权限的传递规则

① 系统权限是不连带：A 授权给 B，B 授权给 C，A 收回 B 的权限后，C 的权限不被收回的

② 对象权限是连带的：A 授权给 B，B 授权给 C，A 收回 B 的权限后，C 的权限也是被收回的

(4) 角色

角色的作用： a.简化权限的管理，减少授权次数； b. 动态权限管理（权限是静态管理的）角色处于激活状态有效，默认角色在用户登录是激活，非默认角色用 set role r1 激活，带口令激活角色 set role r2 identified by r2;

用户自定义角色：

create role r1；//角色 create role r1 identified by r2;//带口令角色

grant select（操作包括 update） on scott.emp to r1,r2; //给 r1，r2 赋予 scott 的 emp 表权

grant r2 to a; grant r2 to b;

alter user a default role none; --a 没有默认角色，r1 不是他的默认，所以 a 不可以访问表

a>set role r1; --把 r1 的角色激活之后 a 就可以访问这个表了。角色只有激活，权限才生效 b>conn b/b

b>set role r2; --没有激活，因为 r2 是带口令的

b>set role r2 identified by r2; --激活带口令的角色

预定义角色：创建数据库就自带的，自己分配了权限，为了之后使用方便

应用程序角色（应用程序激活角色）**不需要指派给用户**

```
revoke r1 from a;    revoke r2 from b; --角色撤回
create role app_r1 identified using scott.p1;--创建应用程序角色
--create role app_r1 identified using 具体应用程序，角色可通过 scott 下表的 p1 应用激活
grant select on scott.emp to app_r1;--给角色分配权限
create or replace procedure scott.p1   --创建应用程序/过程，这个存储过程创建好了
    authid current_user as
begin
    dbms_session.set_role('APP_R1');
end;
grant execute on scott.p1 to a,b;//赋执行权限
conn a/a;
desc scott.emp  --无权限
execute scott.p1;--A 用户
desc scott.emp--当前登陆之后且跑完过程获得权限，退出后重新登陆后不跑过程还是无权限 最小权限原作：给用户能够执行这个操作的最小权限
create role r1;
grant create table to r1;
```

3、审核

简化版：

1 强制审核（默认审核）——警告日志文件

2 标准数据库审核 1）启用审核——修改初始化参数 2）制定审核选项（1）用户审核（权限审核）（2）对象审核（3）语句审核 系统权限之前是验证，强制审核之前是授权

（1）默认审核（强制审核）：重大的数据库事件会记录，如果系统出现故障就会有审核

（2）标准数据库审核（初始化参数）

- 启用审核，通过初始化参数 audit_trail

alter system set audit_trail=extend xml,none,os,db; --none 时不启用审核；os：审核记录存放在操作系统。--db：审核记录放在数据库 aud$表里，xml：审核记录放在一个 xml 表里，放在 audit_file_dest 的路径里， --extended 不可单独使用，与其他的一起，记录的信息更能详细，开销也大

- 指定审核选项

1.审核用户：权限审核 audit select any table by scott; --只要 scott 用户审核权限，就记录下来

2.审核对象：对象审核 audit delete on scott.emp; --只关心表有没有被删除，不管谁删的都要记

3.审核语句：和某种行为有关

audit create trigger; --审核有没有创建触发器模式 noaudit delete on scott.emp;--不审核

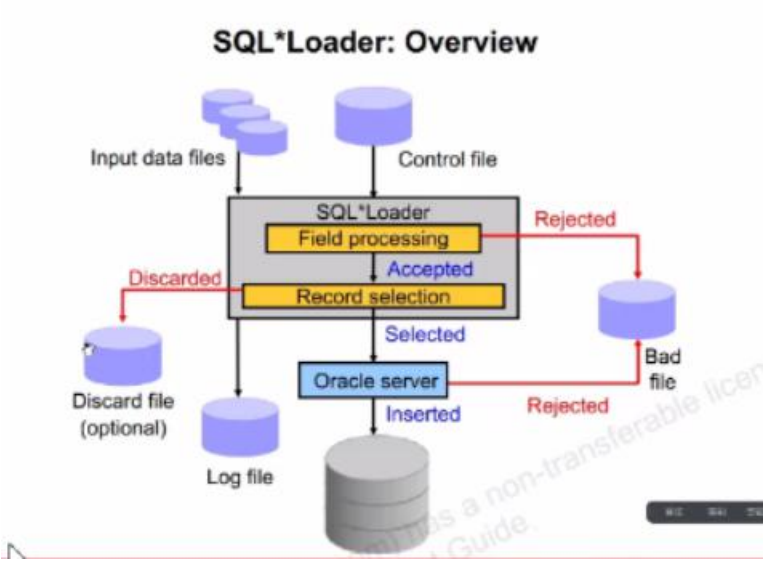
audit delete on scott.emp whenever successful;--删成功审计，识别忽略不计 sudit session whenever not successful;--仅登陆失败时记录

audit update on scott.emp by session;--会话中若干次更新记录，只记录一次 audit update on scott.emp by access --跟新一次记一次

数据移动

一、 数据文件移动到数据库中： 使用 SQLLoader

指定数据源、指定目标表、指定数据格式（记录，字段，字段名，字段类型）、其他



二、 数据库移动到数据库（导出 exp/导入 imp）

1. **启动方式：**
   (1) 交互式（使用的少）
   (2) 命令模式 exp scott/tiger file = emp.dmp tables=emp.dept;
   imp scott/tiger file = emp.dmp tables=emp;
   获取帮助 exp help=y --看看下面的参数

2. **导出的前提条件：**连接到用户的数据库并有权限；导出其他用户/数据库时需要给用户 exp_full_database 权限; sys 下方的的表（系统数据）无法导出，只能导出用户数据；

3. **exp 可以导出的对象**

1）**整个数据库** full=y

2）**表空间** tablespaces=users 整库（整个数据库）

grant **exp_full_database** to scott;--把角色赋给用户 scott->赋权（注意权限！） D:\exp>exp scott/tiger file=users.dmp tablespaces=users --ok

3）**方案 （表的集合）**owner=scott

D:\exp>exp scott/tiger file=scott.dmp owner=scott,……其他用户;

4）**表** tables=dept;D:\exp>exp scott/tiger file=table.dmp tables=emp,dept,……其他用户;

5）**导出表的子集（只把符合条件的导出，query 内写条件）** D:\exp>exp scott/tiger file=sal2500.dmp tables=emp **query='where "sal>2500"'** --注意 shell

6）使用参数文件

nodepad 中写，并保存 p1.par: userid=scott/tiger file=d:\exp\p1.dmp tables=emp query='where sal>2500'

D:\exp>exp parfile=p1.par --将结果保存在 p1 这个文件中，且 query 中不用加双引号

4. **imp 导入类似 exp，注意事项：**

(1) ignore 忽略创建错误

(2) fromuser，touser 将用户导到哪里

5. 练习!!!

题目：权限问题 先将 scott.emp 中工资大于 2500 的记录导出，不导出约束 contraints=N，然后导入到 hr.emp 里（换用户 from/touser）

答 ： F:\exp>exp scott/tiger file=sal2500.dmp tables=emp query='where "sal>2500"' constraints=N

F:\exp>imp hr/hr file=sal2500.dmp fromuser=scott touser=hr tables=emp--无权限

SYS@demo>grant imp_full_database to hr;

F:\exp>imp hr/hr file=sal2500.dmp fromuser=scott touser=hr tables=emp constraints=N create user hr identified by hr;--创建用户 hr

grant create session to hr;

EXP-00091: 正在导出有问题的统计信息。--字符集不匹配，要转换会出现乱码

三、 expdp/impdp

1. **适用条件**：大任务 且 足够多的空闲资源
2. **查看帮助**： expdp help=y
3. **创建目录** 4. **授权**

5.导出表格 参数：remap_table=emp:emp1--导入表的时候可以换名字 emp->emp1

remap_schema=scott:hr 源：目标 remap_tablespace=tbs1:tbs2 换到另一个表空间 exclude=constraints 不导出约束，include=table 只导出表

6. 练习!!

题目：利用 expdp 导出 scott.emp ，不导出约束，然后导入到 hr.emp1*（remap_table,remap_schema),且换一个表空间：

答： select tablespace_name from dba_tables where owner='scott' and table_name='emp';--查 scott.emp 表在哪个表空间

select name from v$tablespace;(remap_tablespace)

F:\exp>expdp scott/tiger directory=expdp_dest dumpfile=scott.dmp tables=emp exclude=constraint

--创建目录

set oracle_sid=db18c

sqlplus / as sysdba

create directory expdp_dest as 'F:\exp';

--授权

grant read,write on directory expdp_dest to scott;

grant read,write on directory expdp_dest to hr;

impdp hr/hr directory=expdp_dest dumpfile=scott.dmp remap_table=emp:emp1 remap_schema=scott:hr remap_tablespace=users:demo；

外部表

步骤：1. 产生外部表定义 2. 指定驱动类型和默认目录 3. 指定访问参数（记录、字段、空字段） 4. 指定字段顺序 5. 指定数据源

代码：CREATE TABLE extab_employees (employee_id NUMBER(4), first_name VARCHAR2(20)) ORGANIZATION EXTERNAL (TYPE ORACLE_LOADER DEFAULT DIRECTORY extab_dat_dir

ACCESS PARAMETERS(records delimited by newline badfile extab_bad_dir:'empx%a_p.bbad' logfile extab_log_dir:'empxt%a_%p.log' filelds terminated by ',' missing field values are null (employee id, first_name last_name))

LOACTION ('empxt1.dat','empxt2.dat')) PARALLEL REJECT LIMIT UNLIITIED;

数据库备份与恢复

1. **分类方式不同的分类：**

(1) 根据数据库是否关闭：冷备份（关闭）/热备份（数据库打开）

(2) 用户管理的备份/**服务器管理的备份（rman）**

(3) 全库备份/部分数据库备份

(4) 全备份/增量备份（变化的备份）

2. 用户管理的备份和恢复

(1) 冷备份（处于 shutdown 状态）

(2) 做文件列表保证文件全部备份：初始化参数文件、控制文件、数据文件、日志文件、口令文件（可选）

(3) 创建测试表 create table scott cold(a varchar(30)) tablespace users; insert into scott cold values ('Before Backup'); commit; alter system checkpoint;

(4) 备份数据库的文件

3. LogMiner 的使用!!!

（1） **logmnr 是用于分析日志的工具**，主要有以下**用途**：1）查明数据库的改变登记：能够用 Logmnr 来分析这些事务，看看究竟发生了些什么事情 2)找回丢失的数据，当不能利用 flashback 时候，能够利用 Logmnr 工具来找回数据，这个时 候，只必需有归档日志即可.

（2） logmnr 工具包含的过程与视图

dbms_logmnr_d 包含了： 1)**add_logfile**:用来增加/剔除用于分析的日志文件.2)**start_logmnr**:用来开启日志分析，而且在 9i/10g 中，能够开启许多不同的分析选项.3)**end_logmnr**:用来终止分析会话，它将回收 LogMiner 所挪借的内存

（3） 与 LogMiner 相关的数据字典：v$logmnr_dictionary:LogMiner 可能利用的数据字典消息. v$logmnr_parameters:目前 LogMiner 所设定的参数消息. v$logmnr_logs:目前用于分析的日志列表. v$logmnr_contents:日志分析收获.

（4） 使用过程!!!

① **启用数据库补充日志** alter database add SUPPLEMENTAL LOG DATA;

② **生成数据字典文件** create directory DICT as '/u01/demo/dict'; EXECUTE dbms_logmnr_d.build('v816dict.ora',' DICT');

③ **开始一个事务**

④ **添加需要分析的日志文件** EXECUTE dbms_logmnr.add_logfile(LogFileName=>'/u01/app/oracle/oradata/db12c/redo01.log',Options=>dbms_logmnr.new);

⑤ **启动分析**

EXECUTE

dbms_logmnr.start_logmnr(DictFileName=>’/u01/demo/dict/v816dict.ora’;

⑥ **查 询 v$logmnr_contents** alter session set nls_data_format ‘yyyy-mm-dd hh24:mi:ss’;

示例：

1）**修改初始化参数** SQL> alter system set utl_file_dir='C:\oracle\logmnr\' SCOPE=SPFILE;

2）重启数据库

3）**提取数据字典文件** SQL> execute dbms_logmnr_d.build('logmnr_dict.ora','c:\oracle\logmnr\',options=>dbms_logmnr_d.store_in_f lat_file);

4）**指定 LogMiner 要分析的重做日志文件**

SQL> EXECUTE

DBMS_LOGMNR.ADD_LOGFILE('C:\oradata\orcl\REDO01.LOG',options=>dbms_logmnr.NEW);

SQL> EXECUTE

DBMS_LOGMNR.ADD_LOGFILE('C:\oradata\orcl\REDO02.LOG',options=>dbms_logmnr.ADDFIL E);SQL> EXECUTE

DBMS_LOGMNR.ADD_LOGFILE('C:\oradata\orcl\REDO03.LOG',options=>dbms_logmnr.ADDFIL E);

5）**启动 LogMiner 会话**

--options=>dbms_logmnr.NO_ROWID_IN_STMT 取消"ROWID="的内容

--dbms_logmnr.DICT_FROM_ONLINE_CATALOG 只分析当前数据库的重做日志文件

-options=>dbms_logmnr.NO_ROWID_IN_STMT+dbms_logmnr.DICT_FROM_ONLINE_CATALOG

6）**SQL>execute**

dbms_logmnr.start_logmnr(dictfilename=>'c:\oracle\logmnr\logmnr_dict.ora');

7）**查看结果** SQL> SELECT * FROM V$LOGMNR_LOGFILE; SQL> select t.scn,t.timestamp,t.seg_owner,t.operation from v$logmnr_contents t where t.seg_name='MYTESTTAB';

8）**结束 LogMiner 会话** SQL> EXECUTE DBMS_LOGMNR.END_LOGMNR;

4. RMAN!!!

(1) **进入 rman：** exp>rman，删除备份：delete backup;展示 show all

(2) **备份一个表空间 users** : sys@db18c>shutdown immediate sys@db18c>startup mount sys@db18c>alter database archivelog;//修改数据库日志模式（非存档->存档模式） sys@db18c>select log_mode from v$datafile; RMAN>backup tablespace users; RMAN>exit

(3) backup 备份--restore 恢复， copy 复制， report 报告， list 查看之前做的备份，备份越麻烦，恢复越方便；要求恢复快可以 copy 之后重命名即可，

(4) 整库备份和恢复!!!

1） **创 建 测 试** 表 create table scott.hotbak(a varchar(30)) tablespace users; insert into scott.hotback values('')

2） **备份前准备：** 备份 spfile、控制文件

rman target/ RMAN>show all configure controlfile autobackup on

备份数据文件：Channel(通道） configure channel.device bzd

RMAN>Configure channel.device type disk format '\d\demobak\%d_%u_%'

Configure controlfile autobackup format for device type disk to 'd:\demobak\auto\%F';

备份归档日志文件：slect log_mode from v$database;//数据库中的归档模式（mount 修改状态，归档模式） alter system archive log current ;当前日志归档

3) **开始备份** backup database plus archivelog;

4) 增加新记录

insert into scott.hotbak values('After backup'); commit;

Demo_编号_日期 备份文件

delete backup; configure device type disk parallelism 3;

alter system archive log current; Show all;

backup database plus archivelog;//3 个通道

select tablespace_name,status from dba_tablespaces;备份 数据文件、初始化参数文件

RMAN>select * from scot.hotbak;

insert into scott/hotbak values('After backup');

RMAN>select * from scott.hotbak;

5) 复制联机重做日志

D:\demobak\文件夹中新建文件夹 log 把日志文件拷过去

Demo>Select member from v$logfile;//查询有哪些日志文件复制需要的日志文件至此文件夹 D：\demobak\log

6) **破坏数据库：** 全部删除、运行 dbca、 删除数据库（删除 demo，数据）

7) **恢复** rman>restore spfile form'd:\demobak\auto\'C-....;

无初始化参数文件无法启动 Rman target /

rman> startup nomount 找不到初始化参数文件（RMAN 提供了一个简化的初始化参数文件）

启动好执行 restore spfile form 'd:\demobak\auto\'C-...';

Oracle \product\18.2.0\database\下有了初始化参数文件 spfiledamo.ora

8) 创建对应目录结构

创建 D:\app\oracle\admin\damo 文件夹

oracle\oradata\data fast_recovery_area\demo

重新启动 shutdown immediate startup nomount

9) 恢复控制文件、数据文件

restore controlfile form 'd:\demobak\auto\'C-...'; alter database mount

restore database; //恢复整个数据库

10) **将前面复制的日志文件复制到 demo 文件夹中** recover database;//完成恢复

11) **打开数据库：** alter database open resetlogs;

5. **热备份：** 归档模式，备份模式，备份数据文件，插入新记录，结束备份模式，破坏数据文件，还原数据文件，恢复数据文件，检查数据

不完全恢复，基于时间点的恢复!!!：创建测试表(create table scott.suibian (id int)；insert into scott.suibian values(100)；commit；alter system archive log currern;)、整库备份（rman>backup database;）、重新插入记录(insert into scott.suibian values(200);commit; select * from scott.suibian;)、查看时间(!date)、删除表(drop table scott.suibian;)、创建表(create table suibian2 as select * from scott.emp; select *from suibian2;)、启动数据库到 mount 阶段、设置环境变量(export NLS_DATE_FORMAT=”yyyy-mm-dd hh24:mi:ss”、还原数据库（rman>restore database;)、还原数据库到指定时间点(rman>restore database until time ‘2020-06-28 12:33:20’;)、打开数据库（rman>alter dattabase open restlogs;)、检查数据（select * from scott.suibian;...）

辅助数据库： 原库：sales(归档模式） 新库:oradup

准备初始化参数文件:
DB_FILE_NAME_CONVERT=(‘/u01/app/oradata/sales/’,’/u01/app/oradata/oradup/’) LOG_FILE_NAME_CONVERT=(‘/u01/app/oradata/sales/’,’/u01/app/oradata/oradup/’) log_archive_dest1=’location=/u01/arch/oradup’;**创建相应目录结构;创建口令验证文件；配置好监听器和服务名；将 oradup 启动到 nomount 状态；启动 rman、同时连接原库和新库**(rman target sys/admin 1#3@sales auxiliary sys/admin1#3@oradup)；**执行 duplicate**(duplicate targert database to oradup from active database;)**检查新库状态**

TSPITR(表空间基于时间点的恢复）创建测试表空间（create tablespace tbs1 datafile ‘/u01/app/oracle/oradata/sales/tbs101.dbf’ size 10m;create tablespace tbs2 datafile ‘/u01/app/oracle/oradata/sales/tbs201.dbf’ size 10m;）**创建测试表**（create table scott.t1 tablespace tbs1 as select * from scott.emp; create index scott.idx_name on scott.t1(ename) tablespace tbs2; alter system checkpoint;)、**整库备份**(rman>backup database plus archivelog; rman>list backup;)**误操作**(select systimestamp from dual; 2020-06-28 12:52:01 truncate table scott.t1;)，**再创建一张表**(create scott.t2 tablespace users as select * from scott.dept;create scott.t3 tablespace tbs1 as select * from scott.dept;select * from scott.t1;select * from scott.t2;select * from scott.t3;),**验证表空间的依赖性**（conn / as sysdba execute DBMS_TIS.TRANSPORT_SET_CHECK(‘TBS1’,TRUE,TRUE);select * from TRANSPORT_SET_VIOLATIONS;如果只恢复表空间 tbs1 会有 t1 的索引依赖表空间 tbs2，同时恢复表空间 tbs1，tbs2 即可解决依赖关系）
确定执行 TSPITR 后会丢失的对象：查询执行 TSPITR 会丢失的对象:select owner,name,tablsespace_name,to_char(creation_time,’yyyy-mm-dd hh24:mi:ss’) from TS_PITR_OBJECTS_TO_BE_DROPPED where tablespace_name in (‘TBS1’,’TBS2’ and creation_time> to_date(‘2020-04-14 08:06:55’,’yyyy-mm-dd hh24:mi:ss’如果有结果，先 expdp 导出结果的备份，待恢复表空间后，再导入，如果这些结果确定无用则可以忽略）

执行 TSPITR(export NLS_DATE_FORMAT=“yyyy-mm-dd hh24:mi:ss” RMAN>recover tablespace tbs1,tbs2 until ‘2020-04-14 08:06:55’ auxiliary destination ‘/u01/aux’;) **检查**（select tablespace_name,status from dba_tablespaces; alter tablespace tbs1 online;alter tablespace tbs2 online;）

表基于时间点的恢复：

先决条件：compatible>=12.0； ARCHIVELOG 模式； READ WRITE 打开模式

步骤：**创建表**(create table scott.emp1 tablespaceusers as select＊from scott.emp)，**备份**（rman>backup database plus archivelog;），**显示现有的**(emp1,select current_scn from v$database)，**删除表**(drop table scott.emp1 purge)**创建表空间里的新表**（create table scott.emp2 tablespaceusers as select * from scott.emp），**插入**（insert into emp ;commit;）**创建目录**（mkdir /u001/aux）**恢复**（rman>recover table scott.emp until scn nnn auxiliary destination ‘/u01/aux’;）**检查**

恢复目录数据库：

(1) **需要使用恢复目录情况**：集中存储多个目标数据库中关于备份和恢复的系统数据；存储一个目标数据库的多个版本的系统数据；存储 RMAN 脚本

(2) **创建恢复目录**： （**前期工作**：定位角色：db12c（目录数据库）、sales（目标数据库）、oradup（目标数据库） ）**创建用来保存备份和恢复系统数据的表空间**：在 db12c 上建，每个表空间大小不小于 15m；**创建目录的所有者;修改目录所有者的默认表空间**（alter user catowner default tab’espace cat;）**赋予相应权限**（grant connect ,recovery catalog owner to catowner; alter user catowner quota unlimited on cat;）**连接到目录数据库**(rman catalog catowner/admin)；**创建恢复目录**（create catalog tablespace ‘CAT’;）**连接到目标数据库**(connect target sys/admin1#3@sales) **注册目标数据库**(register database;) || *到这里如果 backup 则只 backup 了 sales 数据库;***转化到 oradup 数据库操作**(exit rman catalog catowner/admin@db12c target sys/admin1#3@oradup)；**再注册一次数据库**（register database;） 备份的方法 backup
大多数情况下会自动同步

(3) **手动同步**：resync catalog;

(4) **RMAN 脚本：创建脚本** rman catalog catowner/admin target sys/admin1#3@sales(目标数据库名) RMAN> create script 脚本名{内容}
查看脚本 list script names; **查看脚本内容**：print script 脚本名 **执行脚本** run{exectue script 脚本名} **修改脚本** replace script 脚本名{新内容} **删除脚本** delete script 脚本名

闪回技术（可以利用 scn 或者 timestamp）

1） **特点**：速度快、操作简单

2） 使用场景/**缺点**：只能解决逻辑错误，无法解决物理错误（只能用备份恢复）

3） **闪回数据库**（动态性能视图 v$flashback_database_log）

a) **前提条件**：数据库处于归档模式 alter database archivelog；
启用闪回日志：alter database FLASHBACP ON;

b) 和闪回数据库**相关的初始化参数**：db_recovery_file_dest(闪回日志保存地址即闪回区),db_recovery_file_dest_size（闪回区大小），db_flashback_retention_target（闪回日志保存时间，默认为 1440 分钟），OLDEST_FLASHBACK_TIME（可以闪回的最早时间）

c) **闪回方式**：flashback database to scn 序号列号;（利用 scn 闪回）

d) 最好以 read-only 方式打开数据库。

e) **不能闪回情况**： 控制文件被重建了（因为闪回日志在控制文件中，被重建了自然无法闪回）表空间删掉了、物理空间上改变

4）闪回表的 DML 操作：flashback table name to timestamp to_stamp(‘’ ,’ ’)

前提： 激活表的行移动特性 alter table emp enable row movement;

利用时间闪回: select systimestamp from dual; //2020-06-28 20:14:50

flashback table emp to timestamp to_timestamp(‘2020-06-28 20:14:50’ , ‘yyyy-mm-dd hh24:mi:ss’);

5）闪回删除（drop 操作）

前提：启用回收站（查看是否启用 show parameter recyclebin 默认为启用）

清空回收站 purge recyclebin; 查看回收站内容: show recycclebin;

已有操作 drop table emp; **闪回 flashback table emp to before drop; 闪回同时改名： flashback table emp to before drop rename to emp1;**

若删除为 drop table emp purge 则回收站里没有表

6）闪回查询

a. **闪回查询表（某个时间点）** select * from emp as of timestamp to_timestamp(‘2020-06-28 20:28:06’ ,’ yyyy-mm-dd hh24:mi:ss’)

b. **闪回版本查询（一段时间）** select * from employees version between timestamp t1 and t2 where 条件 或者 select * from employees version between scn s1 and s2 where 条件

c. **闪回事务查询 找到视图 flashback_transaction_query 中的对冲语句**

前提：启用数据库**补充日志**：alter database add SUPPLEMENTAL_LOG_DATA_MIN; （查看是否查询 select SUPPLEMENTAL_LOG_DATA_MIN from v$database; yes 则为已经启用）

查询事务号 select versions_xid,version_operation,emp from my_emp versions between scn minvalue and maxvalue;

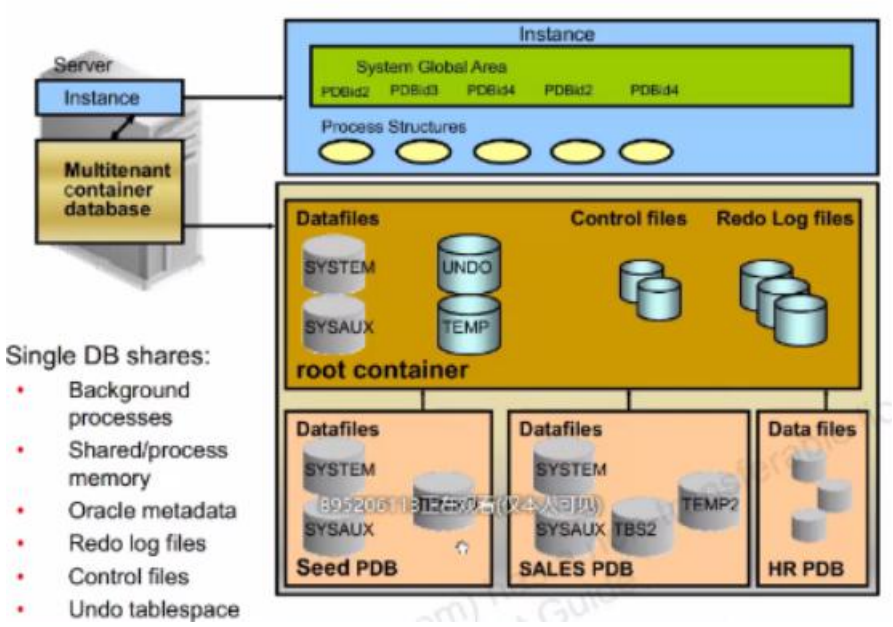
select UNDO_SQL from **flashback_transaction_query** where **xid=‘查询的事务号’;**

执行 UNDO_SQL 的语句 commit;

容器数据库

CDB（container database）不用管理数据

CDB 中系统数据两类：Oracle 系统数据（存放在 root 里即 cdb）、用户系统数据（在 application1 中的 datafiles 中，所以每个 PDB 里也有 system 表存放用户系统数据）



控制文件、日志文件只在 cdb 下有，pdb 中只有数据文件

数据库之间通信更加快速

种子数据库只以 read only 打开，是一个容器，可以删除

用户：

公共用户（CDB+PDB 可见，形如 SYS@cdb>中 SYS 即为公共用户）

本地用户（仅仅 PDB 可见，形如 hr@pdb> 其中 hr 为本地用户） CDB 范围内只能建公共用户，名称前缀为 C##默认是公共用户；PDB 范围内只能建立本地用户

权限（公共权限、本地权限）

赋予公共权限（可以对所有数据库操作）： grant create session to c##aaa container=ALL;

赋予本地权限（默认 grant）： grant create session to c##aaa container=CURRENT;

公共权限可以赋给公共用户，不可以给本地用户

本地权限可以赋给公共用户，可以给本地用户

角色（公共角色、本地角色）

公共角色 公共用户，公共权限

例程管理、配置，视图的修改

（1）**创建 hr.test 表**：create table test(id number(5),name varchar(20));

（2）**插入数据**：insert into test values(1,'aaa'); insert into test values(3,'bbb');

（3）**添加约束**： alter table test add constraint uni_name unique(name);--name 字段添加唯一性约束 insert into test values(2,'aaa'); --报错，违反唯一性约束

（4）**禁用约束**：alter table test disable constraint uni_name;--禁用约束

（5）使用 exceptions 表（找表中违反约束的 lowid 放入 exception 表中）

desc exceptions//不存在

（6）app/oracle/product/18.3.0/rdbms/admin 中的脚本 utlexcept.sql

调用脚本创建 excptions 表， @?\rdbms\admin\utlexcpt.sql 为 oracle home，即 F:\app\oracle\product\18.3.0

（7）**启用约束**，找表中违反约束的 lowid 放入 exception 表

alter table test enable constraint uni_name exceptions into exceptions；

ORA-02299: 无法验证 (HR.UNI_NAME) - 找到重复关键字

select row_id,table_name from exceptions;

select rowid,id，name from test where rowid in（select row_id from exceptions)

（8）update hr.test set name='ccc' where rowid='AAAS6yAADAAAQv1AAC';

（9）再次启用约束 alter table test enable constraint uni_name;

（10）truncate table exceptions;//截断表，清空异常表供以后使用

（11）在数据字典汇总查询约束信息(user_constraints,user_cons_columns)

视图

虚表：不是存储结构，是语句的定义　　表：一种具体的存储结构

视图的作用：1.收集感兴趣的数据　2.简化查询　3.屏蔽敏感数据　4.简化权限的管理

视图的分类：1.简单视图 基表不能超过一张，不能分组、函数　　2.复杂视图

视图的操作

1. **创建**：creat view emp_info as select empno,ename,sal from emp;

select * from emp_info;

2. **修改**视图：

create or replace view emp_info as select empno,ename,sal,deptno from emp;

--修改视图不可以通过删除重建，删除之前有些用户已经拥有了这些视图的权限，删除之后权限就没了

--不可以基于不存在的表创建视图

--强制创建视图

create or replace force view emp_info as select empno,ename,sal,deptno from abc; --编译错误，查询不到。什么时候这个不存在的表被创建了，这些视图就可以使用了

--修改视图，表中数据会变化吗？

update emp_info set sal = 1800 where empno = 7369;　--修改视图就是修改基表

--不允许修改某个字段

create or replace force view emp_info as select empno,ename,sal,deptno from emp where deptno = 30 with check option; --这个时候部门号就不可以修改了，只有部门号不能修改

--只读视图

create or replace force view emp_info as select empno,ename,sal,deptno from emp where deptno = 30 with read only;

--复杂视图不总是可以被修改

--行号　select rownum,ename from emp where rownum >= 10; --rownum 是查询之后再根据结果进行编号

--查出来的结果在进行筛选

select rownum，ename from（select rownum aa,emp.* from emp）　where aa>=10;

例程管理

1. **关闭例程**：

(1) **正常关闭**(normal)　　--等所有回话结束才能关闭，所以时间比较长

(2) **事务性关闭(**TRAMSACTIONAL) --等所有事务结束才能关闭，比 1 快

(3) **立即关闭**(immediate)不需要等事务结束 --上面三个会做检查点，对数据库没有什么伤害

(4) **中止退出**(abort) 丢数据 --不做检查点，会丢失数据

--说明 3 是不需要等事务结束，如果事务还没有 commit，会回滚之后关闭

2. 启动例程

(1) **例程启动**（nomount）分配内存，同时启动后台进程 start nomount

条件：需要正确的初始化参数文件； 功能：在 nomount 下，只能访问一部分动态性能视图（因为动态性能视图来自内存和控制文件）

```
conn sys/admin as sysdba desc v$instance
select status from v$instance; --查询处于什么阶段
select * from v$sga;  -- ok
select name from v$datafile;  --不存在，因为数据库没有打开，数据文件还没打开，查不到数据
select * from scott.emp;  --不存在
```

(2) **加载数据库**（mount 阶段）

条件：需要访问控制文件

```
alter database mount;  --已经进行了 startup nomount 就不可以在进行 startup
show parameter control_files  --查询出 oracle 的控制文件
select status from v$instance;  --查询处于什么阶段，这个结果是 mounted 表示处于 mount 阶段
select name from v$datafile;  --可以访问
select * from scott.emp;  --不存在，表在数据文件中
```

(3) **打开数据库（只读/受限）**

条件：需要联机重做日志文件和数据文件（在数据文件中定义好的）

```
alter database open;
select member from v$logfile;
app/oracle/oradata/db18c/redo01.LOG 文件：日志文件
app/oracle/oradata/db18c/sales01.DBF 文件：数据文件
app/oracle/diag 诊断/rdbms/db18c/trace/alert_db18c.log 文件可以访问所有的数据
```

① **只读的方式**

```
startup open read only;
select name,open_mode from v$database;  --查询打开模式
select current_scn from v$database;  --这个是不变的，平常是变化的
update scott.emp set sal = 1800 where empno = 7369; --报错！数据库只读，不能修改
```

② **受限的方式**

```
startup restrict;
select name,open_mode from v$database;  --查询打开模式
select instance_name,logins from v$instance;  让有些用户连，有些用户不连（有 restriction 权限的可以访问）
grant restricted session to hr;  --hr 就可以登录了
alter system disable restricted session;  --禁用受限模式
alter system enable restricited session;  --启用受限模式
```

3. 例程的配置

(1) 初始化参数文件

① 文本文件 pfile，参数文件，每次重启数据库时修改

② 二进制文件 spfile 服务器管理的参数文件（可显示字符 组成）不可用文本编辑器修改，spfile 是用来服务器的启动的，不可以瞎改，要用命令修改，修改之后直接生效不用重启

(2) 修改初始化参数

alter system set shared_pool_size = 128m; --big integer 类型

show parameter sga_max_size

alter system set shared_pool_size = 5000m; --无法动态修改，先写到文件里，重启才可生效。

desc v$parameter：1. deferred --延时，不会马上/也不需要生效，重新连接就可生效 2. immediate --可立即改 3. false--不可直接改

select name,ISSYS_MODIFIABLE from v$parameter where name like 's%';

scope：1. memory：只在内存改，不在文件改（临时生效） 2.spfile：先写在文件中，下次重启生效 3.both：立即生效，同时在文件中修改

alter system set shared_pool_size=160m scope = memory; --文件没有修改，下次修改还是原来的值

alter system set sga_max_size=5000m scope = spfile; --发现没有生效，但是已经写到 spfile 文件里面了，下次启动生效

(3) 将初始化参数还原成默认值

show parameter shared_server

alter system reset shared_server; --复位 重启之后才可以看到恢复的结果

(4) 将所有初始化参数都还原成默认值

初始化参数文件中没有的都取默认值，删掉文件 18.3.0/database/initdb18c.dba

shutdown immediate

startup 找不到初始化参数文件

nodepad 写一个空文件到 18.3.0/database/initdb18c.dba

参数库文件中要加 数据库名字，文件中加*.dbname='db18c'，再加 1 属性

*.control_files='F:/app/oracle/produce/18.3.0/database/control01.ctl'/u01/con.ctl'

(其他默认)再重启

(5) 修复错误的初始化参数

show parameter spfile

alter system set shared_pool_size=200G scope=spfile; --下次重启生效

show parameter cpu_count

搞坏参数，startup 报错 sga_target 4864M 太小，其中的一部分太大

create pfile from spfile; --根据一个 spfile 创建 pfile 文件

INTIDB18c.ora 文本文件可修改

修改 128m，保存，改名根据 pfile 创建 spfile

show parameter spfile

(7) 例程在启动时选择初始化参数文件的顺序

1. spfile<sid>.ora 有此文件，其他文件都不看，2. pfile.ora 第二顺位 3. init<sid>.ora

4. 都找不到 就报错=>解决：利用指定的初始化参数文件启动 startup pfile=d:\a\a.ora

利用指定的初始化参数文件启动 用 a.ora 启动 startup pfile = filepath;--路径