oracle复习

lecture 3 杂七

```
redo log    --重做日志缓存：恢复，通过log来恢复数据
undo        --对冲 回滚
redo        --重做，前滚，恢复
日志：  记录事务
事务：  是数据库管理系统执行过程中的一个逻辑单位，由一个有限的数据库操作序列构成。
ACID准则：
    A原子性：原子性是指事务包含的所有操作要么全部成功，要么全部失败回滚
    C一致性：一致性是指事务必须使数据库从一个一致性状态变换到另一个一致性状态
    I隔离性：事务不能被其他事务的操作所干扰，多个并发事务之间要相互隔离
    D持久性：一个事务一旦被提交了，那么对数据库中的数据的改变就是永久性的
restore：还原，把备份文件还原成数据文件
recover：恢复
```

```
表--表空间--数据文件
create tablespace sales datafile '...\...' size 10m;
create table scott.customers(id int,name varchar(20)) tablespace sales;
insert into scott.customers values(1,'Tom');

alter database create datafile 5;
recover datafile 5; 介质已经恢复
alter tablespace sales online; 表空间已经更改
数据文件+日志文件
```

```
进程结构
1. 用户进程
    1> 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
    2> 用户进程访问服务器要完成第一个名称解析
```

```
        tnsping db18c; 返回 db18c名称代表的含义
    3> 端口 监听器帮助找到服务器进程，把用户和服务器进程链接起来，服务器开始帮助user进程干活，叫回话开始；
端口是和协议挂钩的
    4> 协议：TCP通过服务名找到监听器才可以连上，要是关了监听器就连不上了
        lsnrctl stop;
        conn scott/tiger@abc;无监听程序 无法链接
        lsnrctl start; 可以连接了
2. 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
    分类：
        1> 专用服务器进程：效率高，支持的用户连接数低 user<200
        2> 共享服务器进程：效率低，支持的用户连接数高 user(200,3000)
        user>3000建池中服务器
3. 后台进程
    PMON：process monitor 进程监视器进程，进程用户进程处理用户进程的异常处理
    SMON：系统监视器进程
    DBWR：数据库书写器进程，把数据从数据库高速缓存中写到数据文件中
    LGWR：日志书写器进程，从日志缓存写到日志文件
    CKPT：检查点进程
```

```
SQL
1 QL(select)
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) 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
```

```
检查点作用：
    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）
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
```

lecture 4 杂八

```
查询时和用户的交互
提交变量
1. &      --提示用户从键盘输入，将条件带入并执行
2. define --定义变量
3. &&     --提示用户输入并定义

--具体例子 for &:
sqlplus scott/tiger as sysdba
select empno.ename,sal from emp where  empno = 7369;   --原来的sql
select empno.ename,sal from emp where  empno = &(给用户的提示) 工号;
select empno.ename,sal from emp where  ename = &xm;  会报错说标识符无效，加引号会显示“未选定行”
select empno.ename,sal from emp where  empno = ‘&xm’;这时用户不用输入引号，但需要注意大小写
```

```
select empno.ename,sal from emp where  empno = upper('&xm');这是用户不用在意大小写

--具体例子 for define:
define gh  会显示符号未定义
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

--具体例子 for &&:
select empno,ename,(用户输入)&&c3 from emp order by &c3
输入c3:sal
define c3  运行后会打印出c3的值"sal"
取消定义:  undefine c3
define c3  这时显示未定义

--说明:
select empno,ename,&&c3 from emp order by &c3
一个&只是键盘输入的值带入查询语句
两个&是把键盘输入的值带入查询语句后再定义
```

```
--sqlplus配置
1. 查看配置:  show
     show all   查看全部配置信息
     show autoc 查看自动提交
2.修改配置:   set
    show time
    set time on(显示操作时间)
    set time off
    show linesize(默认是80)
    show pagesize
    show sqlprompt
```

```
--函数 p92 字符串
1. 单行函数，只对单行进行处理
select empno,lower(ename) from emp;
2. initcap 首字母大写
3. CONCAT 字符串连接；只能有两个参数，多个字符串可以嵌套连接
select CONCAT("Hello","World") from dual;
4. ||是连接符
select 'a'||'b'||'c' from dual
5. SUBSTR 取子串  SUBSTR ("helloworld",1,5) 不写参数就取全串,-2从倒数第二个开始
INSTR ("helloworld",'w',从第几个开始（默认第一个开始找）) 返回字母位置
6. LPAD向左填充 RPAD向右填充
LPAD("1234",10,'*') => "******1234" 中间的参数表示一共几位，一般用于数据格式化
RPAD("1234",10,'*') => "1234******"
7. 多行函数 select empno,ename, from emp
    求每个部门的平均工资select deptno,avg(...) from emp
8. 层次查询
    col ename for a20
    select lpad(' ',(level-1)*2)||ename,level from emp  规定格式
    start with empno = 7566  起始点
    connect by prior empno = mgr;  定方向向下
```

```
9. round四舍五入
select Round(43.926)from dual;      => 43
select Round(43.926,-1)from dual;   => 43
select ROUNF(53.099,-2) from dual; => 100
trunc,ceil,floor
10. 处理日期函数
oracle存储的时间包括: century,year,month,

SQL> select sysdate from dual;

SYSDATE
--------------
08-3月 -19
nls参数: 国家语言支持
nls_data_format
 alter session set nls_data_format= 'yyyy-mm-dd hh:mi:ss';
nls_language
desc nls_session_parameters
col parameter for a40
col value for

用两位数字表示年份，可能会出现误解。
1. YY格式: 和系统日期处于同一个世纪。
2. RR格式: 默认格式。接近系统日期的那个世纪。

计算时间:
MONTH_BETWEEN
ADD_MONTH
NEXT_DAY
LAST_DAY
11. 转化函数
to_date('1999-07-07','yyyy-mm-dd') --要加上日期格式
to_char(18.66,'99.9') => 18.7
select to_char(sysdate,'yyyy-mm-dd') from dual;
fm会去掉前导，比如空格啥的 选择题！所以显示日期的时候不要加fm

12. 处理NULL值的函数
    1> NVL
        select empno,ename,sal,comm from emp;
        select empno,ename,sal,nvl(comm,0) from emp;空的地方用0补
        select nvl(to_char(mgr),'boss') from emp;
    2> NVL2
        select ename,sal,comm,sal+comm income from emp;因为有空值，所以会报错
        select ename,sal,comm,nvl2(comm,sal+comm,sal) income from emp;
    3> NULLIF
        比较两个表达式，一样显示为NULL,不一样显示为第一个表达式的值。
    4> COALESCE 找第一个非空的（空值只要做运算就等于空值）
        select ename,sal,comm,coalesce(sal+comm,sal) income from emp;与上面语句效果相同
```

```
--sql练习
hr.emloyees工会主席安排休假，休假方案。
5个名额，马尔代夫;
20个名额 云南；按工龄来安排休假。
```

```
select LAST_NAME,HIRE_DATE from hr.employees order by 2;

第一份名单：5人，按照YEARS+MONTHS降序排序
LAST_NAME  YEARS  MONTHS
De Haan     18      1
第二份名单：20人
select last_name, trunc(months_between(sysdate, hire_date)/12)
years,trunc(mod(months_between(sysdate, hire_date)/12)) months
from hr.employees
order by 2 desc,3desc
fetch first 5 rows with ties;

select last_name, trunc(months_between(sysdate, hire_date)/12)
years,trunc(mod(months_between(sysdate, hire_date)/12)) months
from hr.employees order by 2 desc,3desc
offset 5 rows --OFFSET 指明在开始返回行之前忽略多少行
fetch first 5 rows only;
```

lecture 5 条件表达式的处理

```
处理if--then--else逻辑
decode(字段或字段的运算，值1，值2，值3)
1> select decode(sign(变量1-变量2),-1,变量1,变量2) from dual; --取较小值
    sign()函数根据某个值是0、正数还是负数，分别返回0、1、-1
    例如：
    变量1 = 10，变量2 = 20
    则sign(变量1-变量2)返回-1，decode解码结果为“变量1”，达到了取较小值的目的。
2> select count(*) total, sum(decode(to_char(hire_date,'yyyy'),2001,1)) "2001",
sum(decode(to_char(hire_date,'yyyy'),2002,1)) "2002" from hr.employees;  2001年入职的人数

group by
select department_id bmh,avg(salary) from hr.employees group by bmh;(错误，分组时不能起别名)

select department_id,job_id,sum(salary) from hr.employees where department_id>40 group by
rollup(department_id,job_id);
group by rollup(A,B) =>group by A,B+A+null
```

将excel文件导入oracle
```
create table st(name varchar(20),subject varchar(20),score int);
```
利用sqlldr

写控制文件
```
load
infile 'd:\st.csv'
into table hr.st
fields terminated  by','
(name char, subject char,score integer external)
```
保存到d:/st.ctl
cmd中sqlldr hr/hr control=d:/st.ctl

```
--多张表连接
    1. 内部
    2. 外部
    3. 多表连接
    4. 自连接
    5. 交叉连接

1. 内部连接
    select b.buyer_id,b.buyer_name,s.qty
    from buyers b,sales s
    where b.buyer_id = s.buyer_id

    emp 10万员工
    dept 4个部门 (10, 20, 30, 40)
    from emp,dept
    nested loop 嵌套循环

    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
    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）

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

    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:
    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
    或
    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? ? ?

exercise:
    select last_name.salary,department_id,b.salavg
    from employees a,
    (select department_id,avg(salary) salavg from employees group by department_id) b
    where a.department_id = b.department_id
    and a.salary > b.salavg
```

lecture 6 事务 子查询

```
--表操作
create table t(id int)
drop table t purge;//删除表
使用块，对表插入十条记录，需Insert语句十次
块:
  --类型
  1. 匿名块：不作为对象存在数据库里面，一代码的形式存在数据文件中
  2. 命名块：给个命名，以对象形式存储在数据库里面，如函数和包
  --快结构:
  declare(声明部分 可选)
      TYPE......
```

```
begin       (可执行部分 必须)      --可执行开始部分
exception   (常处理部分 可选)      --异常处理开始部分，可执行结束部分
end                                --整个语句结束
至少要存在begin和end，里面可以写嵌入式sql
--实例:
begin
   for i in 1..10 loop
     insert into t values(i);
   end loop;
end;
/    --表示pl/sql语句的结束
```

```
--循环
    1. 无条件循环：循环体内有强制退出的条件 一般是exit
    2. 条件循环：一般while
    3. 固定次数循环：for a in 1..60 loop a不同事先定义
```

```
--具体细节
create procedure p7
as
begin
    for i in 1..10 loop
        insert into p values(i);
    end loop;
end;
/
查询是否创建:
    select object_name,object_type from user_procedures where object_type = 'PROCEDURE'
查看存储过程内容:
    desc user_source
    select text from user_source where name = 'p7';
执行P7
execute P7;
call P7();  --call p7会报错
```

```
--查询记录多列显示
select a.id,b.id from p a,p b; --笛卡尔乘积

select a.id,b.id from p a,p b where a.id = b.id(+)-round((select count(*) from p)/2) and
a.id<=round((select count(*) from p)/2);  --两列显示
--说明
(+)在哪一边，哪一边就是全的（连接操作）
```

```
--子查询
1.  嵌套子查询
    select sal from emp where sal>(select sal from emp where ename = 'SMITH');
    select * from emp where deptno in (select deptno from dept where loc in ('NEW
YORK','CHICAGO'));

运算的步骤放在右边效率会高一些。
```

2.关联子查询
    查询谁工资最少
    select last_name,job_id,salary
    from hr.employees
    where salary = (select min(salary) from hr.employees);
    查询每个部门工资最少的 (用group by)

in 只要满足里面一个就出来
any 取条件最宽松的
all 取条件最严格的

```
--exercise
1. last_name 普通群众 (非领导)
    select last_name from hr.employees
    where employee_id in (select  nvl(MANAGER_ID,0) from hr.employees);

    select last_name from hr.employees e
    where exists (select 'X' from hr.employees --'X'是因为查询的结果不需要显示出来
    where e.employee_id = manager_id);
2. last_name,salary,department_id条件: 高于其部门的平均工资 利用关联子查询实现
    select e.last_name,e.salary,e.department_id
    from hr.employees e,
    (select avg(salary) as avgsal,department_id from hr.employees group by department_id) s
    where e.salary > s.avgsal and e.DEPARTMENT_ID = s.DEPARTMENT_ID;
    或
    select e.last_name,e.salary,e.department_id
    from hr.employees e
    where salary>(select avg(salary)from hr.employees
    where e.department_id = department_id
    group by department_id);
```

```
--处理多个结果集
select empno,ename,sal from emp where depyto =10;
select empno,ename,sal from emp where depyto =30;
两个结果集如何合并?

4个操作符:
    1. union
    2. union all 允许重复
    3. intersect  取交集
    4. minus 取第一个结果集去掉交集

select empno,ename,sal from emp where deptno = 10
union
select empno,ename,sal from emp where deptno = 30;
```

```
--表的合并
create table emp_hz as select empno,ename,sal from empwhere deptno = 30;
create table emp_gz as select * from emp_hz where 1=2;  --空表

merge into emp_hz h
using emp_gz g
on(h.empno=g.empno)
when matched then
update set
h.ename = g.ename,
h.sal = g.sal
when not matched then
insert values(g.empno,g.ename,g.sal);
```

```
--修改表的默认值
select * from t1;
insert into t1 values(4,default);//默认的默认值是空值（前提是该字段允许为空);
insert into t1 (id) values(5);
alter table t1 modify name default 'zhangsan';
insert into t1 values(6,default);  --这个时候默认值为'zhangsan'
insert into t1 values(7,default);
--说明
目前有五条数据
grand 是自动提交的，exit退出之后是正常退出，退出的时候会进行提交
插入一条记录，点x退出，是异常退出，会回滚。
```

```
--读的一致性
当事务t1进行数据修改的时候，会将数据复制到回滚带上，如果另外的事务前来查询发现数据被锁，会到回滚带上查询
回滚的三个作用
1. 读的一致性 写的时候可以读
2. 回滚
3. 闪回恢复，删了一个表提交了，进行恢复
select * from t1 timestamp(systemstamp-interval '5'minutes);--查看回滚带上还有没有数据（系统5min前)
insert into t1 select * from t1 as of timestamp(systemstamp-internal'5'minutes);
```

```
--回滚的作用
1.读一致性：若一事务对表里第三条记录进行修改，将改前的数据 复制一份 放入 回滚段。其他事务不允许修改它。
被锁住。另一事务要select的话，对复制在回滚段的进行处理，其他用户看到的都是改前的数据。

2.回滚：未提交前
delete from t1;//都删掉
所有数据都在回滚段
roll back;//自动把在回滚段的都拿回来

3.闪回恢复:已提交后也数据不怕丢掉
delete from t1;//都删掉
commit;
提交后查，数据都没有
rollback自动回滚回不来
查询回滚段:
```

```
select * from t1 timestamp(systemstamp-interval'3'minutes);--查看回滚带上还有没有数据（系统3min前）
恢复数据:
insert into t1 select * from t1 as of timestamp(systimestamp-interval'3'minute)
```

lecture 7 锁机制

```
--锁
两个事物: t1,t2
t1: update t1 set name='Smith' where id =103; --存在于内存
t2: update t1 set id =4 where id =104;        --可改 所以这是一个行锁

如果发生死锁，oracle会自动检测死锁，发生死锁就会回滚死锁事务，另一个事务一直等待，则需要系统管理员
desc v$lock --有个number类型的block: （1阻住，0未阻住）
select sid,block from v$lock where block = 1;//SID747de

desc v$session --阻塞信息，1会话信息
select sid,serial#,username from v$session where sid= 747;//747 59756 hr
alter system kill session '747,59756';
```

```
--DDL
oracle常见对象
    1. 表: 表名长度1-30,表名不可为 oracle保留字，须以字母开头，包含A-Z,a-z,0-9,_,$,#
       表名不可和 同一用户下 相同名称空间namespace的 对象 同名。
    2. 视图: 指向表
    3. 序列: 自动产生
    4. 索引
    5. 同义词 synonym

建表:
    1. 有创建的权限
    2. 必须在一个表空间
刚创建的空间，什么权限都没有，需要给他授予创建会话和表的权限
--授权
SQL> conn sys/admin as sysdba      --已连接。
SQL> grant create session to demo; --授权成功。
SQL> grant create table to demo;   --授权成功。
SQL> conn demo/admin

show parameter defer
alter system set deferred_segment_creation(false);--不允许延时，创建时就要分配
alter user demo quota 5m on users;         --demo可以使用user用户空间5兆空间
alter user demo quota unlimit on users;    --demo可以无限制使用user用户空间
create table scott.t1(id int);             --不可以 create仅可对自己建表
grant create any table to scott;
create table scott.t1(id int);             --demo对scott建表

create table t3 as select * from scott.emp;--基于其他表select要有权限
```

```
grant select on scott.emp to demo;
```

```
--伪列
Base 64 code (数字字母和加减号一共64位)     --具体物理地址
    1. 文件号
    2. 块号
    3. 行号
查询表:
    1. 全表扫描
    2. 索引查找
    3. 直接物理地址查找 (通过rowid)
```

```
--数据类型
char(20): 定长, 已用10个, 剩下的用空格填满 (字符串比较时, 性能好)
varchar(20): 变长, 最多20, 已用10个, 剩下的就空着 (变长的, 节省存储空间) --明确知道多长就用char, 要是变化就用varchar
nchar()
nvarchar()
字符集: 字符的集合
ASCII
A编码01000001  65
--说明
1. 建库的时候要选字符集, 表示以后的符号就是来自于你选的字符集。nchar中的n表示国建。开始创建数据库的时候要选择国家字符集, 这是补充字符集
2. 每张表只能有一个long的字段
3. blob二进制表示大型对象->图片
```

```
--修改表
1.修改表名
    alter table 旧表名 RENAME TO 新表名;
    RENAME 旧表名 TO 新表名;
2.增加字段
    alter table 表名 add(字段名 字段类型 默认值 是否为空)
    alter table 表名 add(userName varchar(30) default'空' not null);
    alter table c add c3 varchar(20);
3.修改字段
    1) 修改字段名称
        alter table c RENAME birthDay tO birth;
    2) 修改字段类型
        alter table c modify birth varchar(30);
    3) 修改宽度
        alter table c modify BUYER_NAME varchar(30);--改大
        alter table c modify BUYER_NAME char(29)    --改小
4.删除字段 (列)
    1)直接删除
        alter table 表名 drop column 列名;
        alter table 表名 drop (列名1, 列名2……);
    2) 先标记为未使用, 然后再删除标记为未使用的列
        alter table c set unused column buyer_name;   --标记为未使用, desc c 差不到
        select table_name from dict where table_name like '_UNUSED_';
        desc USER_UNUSED_COL_TABS
```

```
        select * from USER_UNUSED_COL_TABS;
        col table_name for a20
        /
        alter table c drop unused column;              --删除未使用的列
5.截断表
    truncate table c; --将表清空
6.将表改为只读
    select * from c;
    alter table c read only;    --只读不可删/加数据
    select table_name,read_only from user_tables;
    alter table c read write;
7.给表添加注释
    comment on table hello is 'this is a test table';
    desc user_tab_comments
    select comments from user_tab_comments where table_name='hello';
8.给表的列加注释
    comment on column hello.classId is 'this is a test table';
    desc user_tab_comments
    select comments from user_tab_comments where table_name='hello' and column_name =
'classId';
8.删除表
    drop table c;                              --其实没有删除
    show recyclebin                            --还在回收站
    flashback table c to before recyclebin; --闪回
    drop table c purge;                        --删的连回收站都没了，无法闪回
    select * from recyclebin;
```

数据完整性：

1. 代码实现
2. 触发器实现
3. 约束实现 --用约束实现花销小

约束类型：

1. not null
2. unique
3. primary key
4. foreign key
5. check

两个数据字典：

```
    user_constraints    --定义表级别的约束：关键字，约束名称，约束类型，字段
    user_cons_columns   --定义列级别的约束：关键字，约束名称，约束类型
```

创建约束：

1. 约束要起名字，不起系统会自动起
2. 可以创建表的时候简历约束，也可以在之后添加
3. 表级别的约束=>列定义好了在定义约束 列级别的约束=>列写好了就紧接着定义约束

定义约束书写顺序：

1. 表级别的约束
   先写关键字，约束名称，约束类型名，字段名
2. 列级别的约束
   关键字，约束名称，约束类型
3. 系统定义写约束类型就好了

禁用约束：

```
    alter table cons_test disable constraint MIN_SAL;
```

```
    alter table cons_test enable constraint MIN_SAL;--有数据违反约束会报错，需要把违反约束的修改或者删除才可以启动约束
删除约束:
    alter table cons_test drop constraint SAL_MIN;  --删除约束
```

lecture 8 例程管理、配置，视图的修改

```
--小练习
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表中:
    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. 视图的创建:
        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;
    --修改视图通过删除重建可以吗?
    --不可以，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

    --基于不存在的表创建视图可以吗?
    --不可以

    --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

    --只允许读，不可以修改，做只读视图
    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 rownum
select rownum,ename from emp where rownum >= 10; --rownum是查询之后再根据结果进行编号

--把查出来的结果，再进行筛选
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
        startup 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)
        alter database mount;         --已经进行了startup nomount就不可以在进行startup mount
```

```
        条件：需要访问控制文件
        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（open阶段）
        alter database open;
        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）
        select member from v$logfile;
        app/oracle/oradata/db18c/redo01.LOG文件：日志文件
        app/oracle/oradata/db18c/sales01.DBF文件：数据文件
        app/oracle/diag诊断/rdbms/db18c/trace/alert_db18c.log文件
        可以访问所有的数据
        1）只读的方式打开数据库
            startup open read only;
            select name,open_mode from v$database;       --查询打开模式
            select current_scn from v$database;          --这个是不变的，平常是变化的
            update scott.emp set sal = 1800 where empno = 7369; --报错！数据库只读，不可以修改
        2）以受限的模式打开数据库
            startup restrict;
            select name,open_mode from v$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
            禁用受限模式
            alter system disable restricted session;
            启用受限模式
            alter system enable restricited session;
```

```
--例程的配置
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

一些操作：
1. 查看初始化参数
    show parameter                    --看所有初始化参数
    show parameter shared_pool_size   --看具体某一个（共享池大小）
    show parameter share              --查询所有名字包含share的参数
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$parameter
    deferred       --延时，不会马上/也不需要生效，重新连接就可生效
    immediate      --可立即改
    false          --不可直接改
    select name,ISSYS_MODIFIABLE from v$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''D:app/oracle/……/control02.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
    根据一个spfile创建pfile文件
    create pfile from spfile;
    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
6.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，第一顺位
    2)spfile.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
7.利用指定的初始化参数文件启动
    用a.ora启动
    startup pfile = filePath;  --文件路径
```

lecture 9 手动创建数据库

```
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：
        文件：
        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

lecture 10 管理控制文件，日志文件

lecture 11 管理表空间和数据文件

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）---操作系统的块
    desc dba_segments查一下段有哪些类型
    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
2. 创建users表空间并设为数据库默认的表空间
    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

3. 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
4. 表空间的空间管理（区的管理）
    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
管理
    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
    --本地管理的优点
    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
5. 表空间的类型
    1）常规表空间（可读可写）permanent
    2）撤销表空间（放回滚数据）undo 回滚段在这个表空间
    3）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;

6.表空间的状态
    1）联机可读写online
    2）只读（数据文件只读）
    3）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

7.删除表空间
    drop tablespace smalltbs;--删除表空间后，对应的数据文件还在，要手动删除
    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

8.OMF (oracle管理文件)
    show parameter db_create
    初始化参数db_create_file_dest
    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

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

10.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;

--练习:
set oracle_sid=demo
sqlplus sys/admin as sysdba
startup
desc database_properties
 名称                                      是否为空? 类型
```

```
 ---------------------------------------- -------- ----------------------------
 PROPERTY_NAME                                     VARCHAR2(128)
 PROPERTY_VALUE                                    VARCHAR2(4000)
 DESCRIPTION                                       VARCHAR2(4000)
col property_name for a30
col property_value for a30
select property_name,property_value from database_properties;
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;//default_permanent_tablespace
对应的值为users
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;
ORA-29339: tablespace block size 4096 does not match configured block sizes
show parameter cache
name   type
db_4k_cache_size=0
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 'F:/app/luyao/oradata/demo/test01.dbf'
to'F:/app/omf/test01.dbf';
alter database move datafile 'F:/app/omf/test01.dbf' to
'F:/app/luyao/oradata/demo/test01.dbf' keep;
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）？？？？？？还没做
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
安全3A：验证授权审核
一、验证：
    用户分类（sys、non-sys验证方式不同，sys用户密码不在数据库中）
        数据库不打开，非sys用户不可连上
        sqlplus /nolog
        进入提示符，不连数据库，无法conn非sys用户as sysdba，包括system/manager用户
    --sys
    验证方式
        1. 操作系统验证（默认）
        sqlplus asda/asdasd as sysdba
        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

        2. 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

练习：
1.sqlplus / as sysdba 默认操作系统验证
2.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）
3.修改F:app/oracle/product/18.3.0/network/admin/sqlnet.ora文件中的sqlnet.suthenticaltion_servise=(NTS)、不允许OS验证改为(NONE)
4.sqlplus /as sysdba
5.sqlplus sys/admin1#3 as sysdba口令文件生效
6.sqlplus scott/tiger as sysdba/conn scott/tiger as sysdba==>之前给scott授权
7.select username,sysdba from v$pwfile_users;
8.grant sysdba to scott;

    --非sys用户/普通用户：
        1）数据库验证
            desc user$
            select name,password from user$;
            口令放在 数据库中
            create user nit identified by admin;//新建一个用户，放到表中
        2)操作系统（外部)验证
            1.初始化参数os_authent_prefix（默认时ops$）
            show parameter os
            2.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
            3.在数据库中创建对应的用户
            create user ops$os1 identified externally;
            4.赋予相应的权限
            grant create session to ops$os1;//赋可以登陆的权限
            5.windows注册表修改：
            regedit：hkey_local_machine/software/oracle/key-oradb18home1/增加osauth prefix domain值改为false
            6.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
            7.在窗口敲sqlplus /; conn/
            用户名口令不对==>注册表
二、授权
三、审核（黑匣子--看什么原因造成的）

shutdown immediate
startup mount
```

lecture 12 授权

```
--授权
1. 系统权限（全局，用户）
```

2. 对象权限（局部，资源）
特征：
    1）全局局部
        select any table   --系统权限，因为是全局性的，可以访问所有的表
        select on scott.emp --对象权限，局部
    2）用户资源

--权限传递的规则
    1.系统权限是不连带的   --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

--角色（权限的集合）
    1.角色的作用
        1> 简化权限的管理，减少授权次数；
        2> 动态权限管理（权限是静态管理的）角色处于激活状态(登陆之后激活)有效
            默认角色在用户登录时激活
            非默认角色用 set role r1来激活
            带口令激活角色 set role r2 identified by r2;
    2.用户自定义角色
        create role r1; --角色
        create role r2 identified by r2; --带口令角色
        grant select on scott.emp to r1,r2; --都对此表有访问的权限

        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
    3.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
    4.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;--角色撤回
        revoke r2 from b;
        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
        --创建应用程序/过程，这个存储过程创建好了
        create or replace procedure scott.p1
           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;

```
        grant select on scott.emp to r1;
        --sys
        grant r1 to a;
        grant r1 to a with a with admin option;
        --a
        grant r1 to b;
        --sys
        revoke r1 from a;
        --此时，b的权限还是在的
```

```
--审核
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）

1）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

2）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，不管谁删的都要记录下来
    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
```

```
--触发器 可以自动执行的一段代码，不可以手工调用（和procedure不同）
1.dml触发器：触发条件为dml语句
    conn scott/tiger
    create table soctt.tr_test1(a varchar(20));   --一般触发器不创建在sys下
    --创建触发器
    create or replace tigger scott.tr1
    after update on scott.emp
    begin
         insert into soctt.tr_test1 values('Be changed!');
    end;
    /
    --尝试一下触发器的效果
    select * from scott.tr_test1;--空
    update scott.emp set sal=2800 where empno=7369;--动了5条
    select * from scott.tr_test1;--有一条

    --修改触发器（逐行触发）加for each row
    create or replace tigger scott.tr1
    after update on scott.emp for each row
```

```
    begin
         insert into soctt.tr_test1 values('Be changed!');
    end;
    /
    --尝试一下触发器的效果
    select * from scott.tr_test1;--空
    update scott.emp set sal=2800 where empno=7369;--动了5条
    select * from scott.tr_test1;--有6条

    --练习：利用触发器记录员工的工资改变情况!!
    create table sal_change(ename varchar2(10),old_sal number(7,2),new_sal number(7,2));
    create or replace trigger scott.tr1
    after update on emp referencing old as o new as n for each row
    begin
         insert into scott.sal_change values(:o.ename,:o.sal,:n.sal);
    end;
    /
    select * from sal_change;
    update scott.emp set sal=2800 where empno=7369;

2.系统触发器 只要有用户登录，就记录时间用户等信息
    create table scott.logon_rec(username varchar(30),logon_time date);
    grant administer database trigger to scott;--赋权限
    --创建触发器，只要用户登陆了数据库，就有记录
    create or replace trigger scott.tr3
    after logon on database
    begin
        insert into scott.logon_rec values(user,sysdate);
    end;
    /
    conn a/a;
    select * from scott.logon_rec;


4.细粒度审计（FGA）：
    desc dbms_fga
    --调用包
    execute
dbms_fga.ADD_POLICY('SCOTT','EMP','FGA_DEMO',STATEMENT_TYPES='select,insert,delete');

    --a:
    select empno,ename,sal from scott.emp;
    --处理错误
    alter index scott.pk_emp rebuild;
    alter user scott quota unlimited……;
    insert into scott.emp(empno,ename)values(1234,'Tom');
    update scott.emp set sal=3000 where empno=1234;
    delete from scott.emp wheer empno=1234;
    select * from scott.emp;
    --sys:
    desc dba_fga_audit_trail--查询dba_fga_audit_trail结果
    alter session set nls_date_format='yyyy-mm-dd hh24:mi:ss';
```

```
    select timestamp,db_user,sql_text from dba_fga_audit_trail where
policy_name='FGA_DEMO';--policy
```

lecture 13 导入/导出（imp/exp）

```
--一个工具imp/exp(可以把对象导入导出):
1.交互式（问什么答什么）使用的少
C:>exp
2.命令模式
exp scott/tiger file = emp.dmp tables=emp.dept;
imp scott/tiger file = emp.dmp tables=emp;
3.获取帮助
exp help=y  --看看下面的参数
4.exp可以导出的对象:
    1)整库（整个数据库）
        full=y
    2)表空间 tablespaces=users
        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）
        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

imp类似exp
导入imp的注意事项:
1.ignore忽略创建错误:
    truncate table scott.emp;--表记录全部清空，但表还在
    drop table scott.emp;--删掉表
    imp scott/tiger file=emp1.dmp tables=emp;--导入出错->因为导入是先创一个表，在将内容导入表中，在
映射到文件，这时候表已经存在就错了
    imp scott/tiger file=emp1.dmp tables=emp ignore=y;--索引出错
    alter index  …… (粘贴) rebuild;
    imp scott/tiger file=emp1.dmp tables=emp ignore=y;--忽略创建这张表的错误
2.fromuser,touser将用户导到哪去

--练习：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)
```

```
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
```

```
--数据泵--快
expdp/impdp  --通过并行来提高性能，但是不一定就比exp快，对于规模小的时候效率会低一点
1.大任务
2.足够多的空闲资源

1.查看帮助
    expdp help=y显示参数列表
    F:\exp>expdp scott/tiger file=F:\exp\empdp.dmp tables=emp       //报错??换db18c
    --文件名里不能包含路径说明
    F:\exp>expdp scott/tiger file=empdp.dmp tables=emp
    --必须指定目录对象参数且不可为空
2.创建目录
    set oracle_sid=db18c
    sqlplus / as sysdba
    create directory expdp_dest as 'F:\exp';
    F:\exp>expdp scott/tiger directory=expdp_dest file=empdp.dmp tables=emp --scott用户对目
录无权限
3.授权
    sys@db18c>grant read,write on directory expdp_dest to scott;
4.导出表格
    F:\exp>expdp scott/tiger directory=expdp_dest file=empdp.dmp tables=emp
    imp scott/tiger diretory=expdp_dest dumpfile=empdp.dmp tables=emp  --出错，表已存在
    imp scott/tiger diretory=expdp_dest dumpfile=empdp.dmp tables=emp
table_exists_action=replace
    --sys@db18c可select scott.emp

    --参数
    remap_table=emp:emp1         导入表的时候可以换个名字 emp->emp1
    remap_schema=scott:hr        (源：目标) 换到另一个用户下 把原对象到瑞到目标对象下
    remap_tablespace=tbs1:tbs2   换到另一个表空间
    exclude=constraints;         不导出约束

--练习：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：
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';
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;
```

```
--数据库备份和恢复
用户误操作
1.使用logMiner日志挖掘器（工具）查询重做日志文件
    1）启用数据库补充日志：记日志比较全面
        查询启动了吗? desc v$database
        supplemental_log_data_min
        select supplemental_log_data_min from v$database;//NO未启动
        alter database add/drop supplemental log data;//启用/关闭
        select supplemental_log_data_min from v$database;//YES启动了
    2）创建/产生一个数据字典文件：日志文件中-对象号：二进制/16进制记日记，人看不懂，对照数据字典文件--能对应看懂
        放到F:/dict
        create directory dict as 'F:\dict';//物理路径+逻辑路径。创建目录
        execute dbms_logmnr_d.build('v816dic.ora','DICT');//加单引号之后要大写
    3）开始一个事务：开始记日记
        scott:
        set oracle_sid=db18c
        sqlplus scott/tiger
        select empno,ename,sal from emp;
        update emp set sal=1800 where empno=7369;
        commit;//提交，记日记
    4)添加需要分析的日志文件
        sys db18c>select group#,status from v$log;
        select group#,member from v$logfile;
        查到日志文件名~F:\APP\LUYAO\ORADATA\DB18C\REDO03.LOG
        execute
dbms_logmnr.add_logfile(LogFileName=>'F:\APP\LUYAO\ORADATA\DB18C\REDO03.LOG',options=>dbms_logmnr.new);
    5)执行分析：
        execute dbms_logmnr.start_logmnr(DictFileName=>'F:\dict\v816dic.ora');
    6)查询结果
        视图v$logmnr_contents中
        desc v$logmnr_contents
        alter session set nls_date_format='yyyy-mm-dd hh24:mi:ss';
        select timestamp,sql_redo from v$logmnr_contents
        where SEG_NAME='EMP' and operation='update';//对哪个表操作的seg_name

2.RMAN工具：Recover manage（恢复管理器，处理备份和恢复）
两种方式：冷备份（数据库未打开，mount状态下的备份，不是shutdown状态下）、热备份（数据库打开时做的备份）
1）冷备份：MOUNT
2）热备份：打开数据库的备份

认识RMAN：
进入RMAN：
F:\exp>rman
备份一个表空间users
```

```
backup tablespace users;//报错，未连接到目标数据库
connect target sys/admin
backup tablespace users;//非归档模式只能做冷备份，不可做热备份
sys@db18c>shutdown immediate
sys@db18c>startup mount
sys@db18c>alter database archievelog;//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

犯错：删数据文件
sys db18c:select name from v$datafile;
select * from scott.emp;
alter tablespace users offline;
目录夹中users01.dbf删掉
sys db18c>select * from scott.emp;//查不到了，找备份

F:\exp>rman targer /
restore tablespace users;
recover tablespace users;
sys db18c>select * from scott.emp;//查到了

RMAN>exit

删掉数据文件
F:\exp>rman target /
RMAN>advise failure all;
看修复文件脚本这个文件
RMAN>repair failure;
db18c>alter tablespace users online;
可select
```

lecture 14 RMAN下的整库备份和恢复

```
RMAN下的整库备份和恢复
1.创建测试表
    create table scott.hotbak(a varchar(30)) tablespace users;
    insert into scott.hotback values('')
2. 备份前的准备
    备份spfile
    备份控制文件

    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_%T'
```

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

    备份归档日志文件
    select 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//自动备份开关打开了，回车开始备份
4.增加新的记录
    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. 复制联机重做日志文件
    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
    全部删掉
    运行dbca
    cmd>dbca
    删除数据库（删demo 数据），口令验证文件sys-admin1#3
    完成删除整个数据库
7. 恢复
    恢复初始化参数文件spfile
    set oracle sid-demo
    rman target/
    oradim -new -sid demo(用管理员身份创建)
    多了oracledemoservice服务里面
    rman target /
    连接到目标数据库未启动，无法恢复
    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. 创建相应的目录结构
    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
    restore controlfile form 'd:\demobak\auto\'C-...';
    alter database mount
10. 恢复数据文件
    restore database;//恢复整个数据库
11.将前面复制的日志文件复制到demo文件夹中
    recover database;//完成恢复
12.打开数据库
    alter database open resetlogs;
```