查询时和用户交互

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

sqlplus scott/tiger as sysdba

select empno.ename,sal from emp where empno = 7369;

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

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

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

==========

sqlplus 配置

1.查看配置： show

show all

show autoc

set autoc on(设置自动提交)

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

select * from emp 已选择 12 行

select * from emp where mgr = 7698 没有已选择 5 行

set feed 5

select * from emp where mgr = 7698 会显示已选择 5 行

show define 默认是&

set define ! 会报错

undefine gh

select empno.ename,sal from emp where empno = !gh;

==========

函数：预置好的公式

select empno,ename from emp;

单行函数，只对单行进行处理 select empno,lower(ename) from emp;

initcap 首字母大写

CONCAT 字符串连接 select CONCAT("Hello","World") from dual; 只能有两个参数，多个字符串可以嵌套连接

select 'a'||'b'||'c' from dual ||是连接符

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

LPAD 左填充

RPAD 右填充

多行函数 select empno,ename, from emp

求每个部门的平均工资 select deptno,avg(...) from emp

层次查询

col ename for a20

select lpad(' ',(level-1)*2)||ename ename,level from emp 规定格式

start with empno=7566 起始点

connect by prior mgr=empno; 定方向

select Round(43.926,-1)from dual;

select ROUNF(53.099,-2) from dual; 得到 100

trunc,ceil,floor

==========

sysdate

current_data

systimestamp(最多 9 位)

SQL> select systimestamp from dual;

SYSTIMESTAMP

08-3 月 -19 02.59.11.079000 下午 +08:00

SQL> select systimestamp(3) from dual;

SYSTIMESTAMP(3)

08-3 月 -19 02.59.45.733 下午 +08:00

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

练习：

hr.emloyees 工会主席安排休假，休假方案。有 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
fetch first 5 rows with ties;
```

转换函数：

隐式转换（自动转换）

显示转换

TO_CHAR

```
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

```
select empno,ename,to_char(sal,'99,999.99') sal from emp;
select empno,ename,to_char(sal,'90,999.99') sal from emp;
select empno,ename,to_char(sal,'L90,999.99') sal from emp;
alter session set NLS_TERRITORY = AMERICA
alter session set NLS_TERRITORY = 'UNITED KINGDOM'

INSERT INTO EMP(EMPNO,ENAME.HIREDATE)VALUES(1234,'TOM','1985-07-11')  报错

INSERT INTO EMP(EMPNO,ENAME.HIREDATE)VALUES(1234,'TOM',to_date('1985-07-11','yyyy-mm-dd'));
```

处理 NULL 值的一些函数：

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;
```

NVL2

为空显示第二个，不为空显示第三个

```
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;
```

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。

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

条件表达式：

处理 if--then--else 逻辑

1. case 语句　考试会考（case 或者 decode）

2. decode 函数

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

==========

select max(salary) from hr.employees;

SQL> select max(1,4,7);

select max(1,4,7)

       *

第 1 行出现错误:

ORA-00909: 参数个数无效

SQL> select greatest(1,4,7) from dual;

GREATEST(1,4,7)

              7

group by

where:记录筛选

having:分组筛选

SQL> select department_id,avg(salary) from hr.employees group by department_id;

DEPARTMENT_ID AVG(SALARY)

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

```
            50  3475.55556
            40        6500
           110       10154
            90  19333.3333
            30        4150
            70       10000
                      7000
            10        4400
            20        9500
            60        5760
           100  8601.33333

DEPARTMENT_ID AVG(SALARY)
------------- -----------
            80  8955.88235

已选择 12 行。



SQL> select department_id bmh,avg(salary) from hr.employees group by bmh;
select department_id bmh,avg(salary) from hr.employees group by bmh
                                                              *
第 1 行出现错误:
ORA-00904: "BMH": 标识符无效

分组时不能起别名
select department_id,job_id,sum(salary) from hr.employees where department_id>40 group by department_id,job_id
group by department_id;

select department_id,job_id,sum(salary) from hr.employees where department_id>40 group by
rollup(department_id,job_id) group by department_id;

SQL> select department_id,job_id,sum(salary) from hr.employees where department_id>40 group by
department_id,job_id;

DEPARTMENT_ID JOB_ID               SUM(SALARY)
------------- -------------------- -----------
           90 AD_VP                      34000
          100 FI_MGR                     12008
           80 SA_REP                    243500
           90 AD_PRES                    24000
          110 AC_MGR                     12008
           60 IT_PROG                    28800
```

```
            80 SA_MAN                       61000
            50 SH_CLERK                     64300
            50 ST_CLERK                     55700
            70 PR_REP                       10000
           110 AC_ACCOUNT                    8300

DEPARTMENT_ID JOB_ID              SUM(SALARY)
------------- ------------------- -----------
            50 ST_MAN                       36400
           100 FI_ACCOUNT                   39600

已选择 13 行。

SQL> select department_id,job_id,sum(salary) from hr.employees where department_id>40 group by
rollup(department_id,job_id);

DEPARTMENT_ID JOB_ID              SUM(SALARY)
------------- ------------------- -----------
            50 ST_MAN                       36400
            50 SH_CLERK                     64300
            50 ST_CLERK                     55700
            50                             156400
            60 IT_PROG                      28800
            60                              28800
            70 PR_REP                       10000
            70                              10000
            80 SA_MAN                       61000
            80 SA_REP                      243500
            80                             304500

DEPARTMENT_ID JOB_ID              SUM(SALARY)
------------- ------------------- -----------
            90 AD_VP                        34000
            90 AD_PRES                      24000
            90                              58000
           100 FI_MGR                       12008
           100 FI_ACCOUNT                   39600
           100                              51608
           110 AC_MGR                       12008
           110 AC_ACCOUNT                    8300
           110                              20308
                                           629616

已选择 21 行。
```

```
SQL> select department_id,avg(salary) from hr.employees group by department_id having avg(salary)>8000;

DEPARTMENT_ID AVG(SALARY)
------------- -----------
          110       10154
           90 19333.3333
           70       10000
           20        9500
          100 8601.33333
           80 8955.88235

已选择 6 行。
```

==========

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 joinsales s

on b.buyer_id = s.buyer_id (inner可省)

2.

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.

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.

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? ? ?

练习：

last_name.salary,department_id,salavg

条件：其工资高于其部门的平均工资

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

教材：D33996-9i

pl/sql 过程化结构，过程化语言

练习： hr.t

drop table t purge;//删除表

create table t(id int)

插入 10 条记录，需 Insert 语句十次，用块

块：

块类型：

1.匿名块(不能作为对象存到数据库中）

声明

declare(声明部分）

TYPE.....

begin（可执行部分）

exception（异常处理部分）

end

==========

declare 可能没有，exception 也可能没有，看到 begin 和 end 就是 pl

2.命名块：后台开发 一个对象，可以存到数据库中

块结构：

1. 声明部分（可选）

2.可执行部分（必选）

3.异常处理部分（可选）

```
desc user_procedures
select object_name,object_type from user_procedures
where object_type = 'PROCEDURE'
```

```
create procedure p3
as(as 和 begin 之间写声明）

begin
```

循环

1.无条件循环，进入循环体不需要任何条件，体内需要一个退出条件

2. 条件循环(while)循环

3. 固定次数的循环(for 循环)

```
begin
     for i in 1..10 loop
          insert into t values(i)
     end loop;
end;
/             (表示 pl/sql 语句的结束）
```

当某个功能结束时学分号，未结束不写。

```
truncate table t
```

```
create procedure p7
as
begin
     for i in 1..10 loop
          insert into p values(i);
     end loop;
end;
/
```

```
desc user_source
select text from user_source where name = 'p7';

execute p7
call p7 会报错
call p7()

显示为2列
select a.id,b.id from p a,p b where a.id+5 = b.id;
where a.id+(select count(*) from t)/2 = b.id

如果有11行
select a.id,b.id from p a,p b where a.id+round((select count(*) from p)/2) = b.id
显示出来没有6
select a.id,b.id from p a,p b where a.id+round((select count(*) from p)/2) = b.id (+)
这样会报错
select a.id,b.id from p a,p b where a.id = b.id(+)-round((select count(*) from p)/2)
这样第一列会有11个数
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);

==========
scott
子查询
1.  嵌套子查询
select empno,ename,sal from emp;
select sal from emp where ename = "SMITH";
select empno,ename,sal from emp where sal >2800;
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）
```

==========

多行操作符：

1. in

2. any

3. all

```
SQL>  select empno,ename,sal from emp where sal = (1500,2800);
 select empno,ename,sal from emp where sal = (1500,2800)
                                            *
第 1 行出现错误:
ORA-01797: 此运算符后面必须跟 ANY 或 ALL
```

```
SQL>  select empno,ename,sal from emp where sal in (1500,2800);

     EMPNO ENAME                      SAL
---------- -------------------- ----------
      7844 TURNER                     1500
```

```
SQL>  select empno,ename,sal from emp where sal>any(1500,2800);

     EMPNO ENAME                      SAL
---------- -------------------- ----------
      7499 ALLEN                      1600
      7566 JONES                      2975
      7698 BLAKE                      2850
      7782 CLARK                      2450
      7839 KING                       5000
      7902 FORD                       3000

已选择 6 行。
```

```
SQL>  select empno,ename,sal from emp where sal>all(1500,2800);

     EMPNO ENAME                      SAL
---------- -------------------- ----------
      7566 JONES                      2975
      7698 BLAKE                      2850
      7839 KING                       5000
      7902 FORD                       3000
```

练习：

desc hr.employees

要求：

last_name 普通群众（非领导）

答：

```
select last_name from hr.employees
where employee_id in (select  MANAGER_ID from hr.employees);
```

未选定行 //原因是因为manager_id里有空值

```
select last_name from hr.employees
where employee_id in (select  nvl(MANAGER_ID,0) from hr.employees);
```

共有18个

exists

```
select last_name from hr.employees e
where exists (select 'X' from hr.employees
where e.employee_id = manager_id);
```

练习：

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

38个

```
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;
```

```
     EMPNO ENAME                       SAL
---------- -------------------- ----------
      7499 ALLEN                      1600
      7521 WARD                       1250
      7654 MARTIN                     1250
      7698 BLAKE                      2850
      7782 CLARK                      2450
      7839 KING                       5000
      7844 TURNER                     1500
      7900 JAMES                       950
      7934 MILLER                     1300
```

已选择 9 行。

```
select count(*)from job_history;
```

```
====================================
column a_dummy noprint
select 'sing' AS "my dream" from dual
union
select 'i''d like to teach' from dual
union
select 'the world to' from dual
order by 1;

my dream
----------------------------------
i'd like to teach
sing
the world to
```

```
column a_dummy noprint
select 'sing' AS "my dream",3 from dual
unio

select 'i''d like to teach' ,1 from dual
union
select 'the world to',2 from dual
order by 2;

my dream                                  3
--------------------------------- ----------
i'd like to teach                         1
the world to                              2
sing                                      3

================================================insert:
desc departments

create table t1(id int, name varchar(20));
insert into t1 values(1,'tom');
insert into t1 values(2,null);
insert into t1 values(3,null);

create table demo as select * from emp where 1=2;

insert into where

update:
update t1 set name = 'jerry';
update t1 set name = 'jerry' where id = 2;

delete:
select * from t1;
delete from t1;全删
delete from t1 where id = 3;



drop table emp_hz;
drop table emp_gz;
scott 下建表：
create table emp_hz as select empno,ename,sal from emp
where deptno = 30;
create table emp_gz as select * from emp_hz where 1=2;
insert into emp_gz values(1234,'tom',3500);
```

```
insert into emp_gz values(7900,'james',4500);

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);

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);
insert into t1 values(7,default);
```

换一个用户看 t1 表，只有 5 条记录，alter 语句 ddl 自动提交。

```
grant
revoke
```

很多语句不需要 commit 提交，它会自动提交，比如 ddl、dcl 语句,所以他们只能发一条语句，然后就提交了...

SQL

1. QL(select)
2. DML(insert,update,delete,merge)
3. DCL(grant,revoke)
4. TCL (commit,rollback,savepoint)

```
save point aaa;
delete from t1 where name = 'zhangsan';
```

rollback 的话全回来了

```
rollback to savepoint aaa;
```

管理控制文件

二进制，记录整个数据库的状态，没有控制文件无法定位数据库

1.查看控制文件的名称和位置

```
show parameter control_files
select name from v$controlfile
```

2. 控制文件的内容

oradebug dump controlf 3 未指定进程

oradebug setmypid 定义转储的进程

oradebug TRACEFILE_NAME

notepad 文件目录

3. 控制文件多路复用（控制文件只有一个，多个的话都一模一样）

1)修改初始化参数

alter system set control_files =

'F:\MXYYY\APP\PINKPIG\ORADATA\DB18C\CONTROL01.CTL',

'F:\MXYYY\APP\PINKPIG\ORADATA\DB18C\CONTROL02.CTL',

'F:\NIT\CONTROL03.CTL' scope = spfile

2）shutdown immediate

3) 复制出第三个控制文件

4）startup

5) 检查控制文件 select name from v$controlfile

4. 重建控制文件

1）

select status from v$instance

alter database backup controlfile to trace as 'D:\NIT\controf.sql'

2）shutdown immediate

3） 删除所有控制文件

4）startup

5) 执行创建控制文件的语句（在上面目录的文件中）

6）alter database open

练习：

1. 查看控制文件名称及内容

2. 控制文件的复用（产生多个控制文件）

3. 重建控制文件

==========

管理日志文件

1. 日志文件的作用

恢复，记录对数据所做的改变，提供恢复机制

2. 日志文件原理

select group#,sequence#,status from v$log;

```
    GROUP#  SEQUENCE# STATUS
---------- ---------- --------------------------------
         1         19 INACTIVE
         2         20 CURRENT
         3         18 INACTIVE
```

```
select group#,member from v$logfile
    GROUP#
----------
MEMBER
--------------------------------------------------------------------------------
         3
F:\MXYYY\APP\PINKPIG\ORADATA\DB18C\REDO03.LOG

         2
F:\MXYYY\APP\PINKPIG\ORADATA\DB18C\REDO02.LOG

         1
F:\MXYYY\APP\PINKPIG\ORADATA\DB18C\REDO01.LOG
```

3. 数据库日志模式

1） 非存档模式

2） 存档模式:连接备份

```
select log_mode from v$database;

SQL> alter system switch logfile;

系统已更改。

SQL> select group#,sequence#,status from v$log;

    GROUP#  SEQUENCE# STATUS
---------- ---------- --------------------------------
         1         19 INACTIVE
         2         20 ACTIVE
         3         21 CURRENT
```

一个日志最少两个组，一个组至少一个成员，一个组中的每个成员都是镜像关系，放在不同磁盘上。

4. 添加日志文件组

```
alter database add logfile group 4
('F:\mxyyy\app\pinkpig\oradata\DB18C\redo04a.log') size 10m;

数据库已更改。
```

5. 查看日志组信息

```
desc v$log
select group#,sequence#,status from v$log;

    GROUP#  SEQUENCE# STATUS
---------- ---------- --------------------------------
```

```
         1          19 INACTIVE
         2          20 INACTIVE
         3          21 CURRENT
         4           0 UNUSED
```

6. 产看日志成员信息

```
desc v$logfile
select group#,member from v$logfile;


    GROUP#
----------
MEMBER
--------------------------------------------------------------------------------
         3
F:\MXYYY\APP\PINKPIG\ORADATA\DB18C\REDO03.LOG

         2
F:\MXYYY\APP\PINKPIG\ORADATA\DB18C\REDO02.LOG

         1
F:\MXYYY\APP\PINKPIG\ORADATA\DB18C\REDO01.LOG


    GROUP#
----------
MEMBER
--------------------------------------------------------------------------------
         4
F:\MXYYY\APP\PINKPIG\ORADATA\DB18C\REDO04A.LOG
```

7. 添加日志文件成员

```
alter database add logfile member 'F:\mxyyy\app\pinkpig\oradata\DB18C\redo04b.log' to group 4;

select group#,member from v$logfile;

    GROUP#
----------
MEMBER
--------------------------------------------------------------------------------
         3
F:\MXYYY\APP\PINKPIG\ORADATA\DB18C\REDO03.LOG

         2
```

F:\MXYYY\APP\PINKPIG\ORADATA\DB18C\REDO02.LOG

         1
F:\MXYYY\APP\PINKPIG\ORADATA\DB18C\REDO01.LOG

    GROUP#

MEMBER

         4
F:\MXYYY\APP\PINKPIG\ORADATA\DB18C\REDO04A.LOG

         4
F:\MXYYY\APP\PINKPIG\ORADATA\DB18C\REDO04B.LOG

8. 删除日志文件组 active 不可删

组的状态：current,active,inactive,unused

alter session nls_language = america

...

alter system switch logfile

...

alter system checkpoint 检查点，不会有 active

select group#,status from v$log;

alter database drop logfile group 2;

select group#,status from v$log;

    GROUP# STATUS
---------- --------------------------------
         1 INACTIVE
         3 CURRENT
         4 UNUSED

9. 删除日志文件成员

alter database drop logfile member

alter system switch logfile

alter database drop logfile member 'F:\mxyyy\app\pinkpig\oradata\DB18C\REDO04B.LOG';（物理上没有删除）

10. OMF(oracle 管理文件）

配置相应的初始化参数来实现 OMF

show parameter db_create

NAME   TYPE
------------------------------------ ---------------------

VALUE

---

db_create_file_dest string

db_create_online_log_dest_1 string

db_create_online_log_dest_2 string

db_create_online_log_dest_3 string

db_create_online_log_dest_4 string

NAME TYPE

VALUE

---

db_create_online_log_dest_5 string

create tablespace users datafile 'F:\mxyyy\app\pinkpig\oradata\DB18C\USERS01.DBF';

drop tablespace users 没有物理删除

show parameter db_create_file_dest

NAME TYPE

VALUE

---

db_create_file_dest string

alter system set db_create_file_dest='F:\mxyyy\app\omf'

show parameter db_create_file_dest

NAME TYPE

VALUE

---

db_create_file_dest string

F:\mxyyy\app\omf

select name from v$datafile

drop tablespace users 物理删除

日志文件自动 OMF

alter database add logfile 默认存储在闪回区

配置相应的初始化参数来实现日志文件自动 OMF

show parameter db_create

NAME TYPE

VALUE

---

db_create_file_dest string

F:\mxyyy\app\omf

db_create_online_log_dest_1 string

db_create_online_log_dest_2 string

db_create_online_log_dest_3 string

db_create_online_log_dest_4 string

db_create_online_log_dest_5 string

在 F:\mxyyy\app\omf\log1 和 F:\mxyyy\app\omf\log2 中各放两个成员：

alter system set db_create_online_log_dest_1 = 'F:\mxyyy\app\omf\log1'

alter system set db_create_online_log_dest_2 = 'F:\mxyyy\app\omf\log2'

alter database drop logfile group 5;

11. 清除日志文件内容

alter database clear logfile;

alter database clear logfile group n;

alter database clear logfile ' ...';

12. 修改数据库的日志模式,一定要在 mount 阶段执行

归档-非归档

shutdown immediate

startup mount

alter database archivelog;

archive log list

alter database noarchivelog;

alter database open;

13. 设置归档日志目的地 并 进行归档

show parameter log_archive_dest

NAME TYPE

```
VALUE
------------------------------
log_archive_dest                       string

log_archive_dest_1                     string

log_archive_dest_10                    string

log_archive_dest_11                    string

log_archive_dest_12                    string

NAME                                   TYPE
-------------------------------------- ----------------------
VALUE
------------------------------

log_archive_dest_13                    string

log_archive_dest_14                    string

log_archive_dest_15                    string

log_archive_dest_16                    string


NAME                                   TYPE
-------------------------------------- ----------------------
VALUE
------------------------------
log_archive_dest_17                    string

log_archive_dest_18                    string

log_archive_dest_19                    string

log_archive_dest_2                     string

log_archive_dest_20                    string

NAME                                   TYPE
-------------------------------------- ----------------------
VALUE
------------------------------
```

```
log_archive_dest_21                 string

log_archive_dest_22                 string

log_archive_dest_23                 string

log_archive_dest_24                 string


NAME                                TYPE
----------------------------------- ----------------------
VALUE
------------------------------
log_archive_dest_25                 string

log_archive_dest_26                 string

log_archive_dest_27                 string

log_archive_dest_28                 string

log_archive_dest_29                 string

NAME                                TYPE
----------------------------------- ----------------------
VALUE
------------------------------

log_archive_dest_3                  string

log_archive_dest_30                 string

log_archive_dest_31                 string

log_archive_dest_4                  string


NAME                                TYPE
----------------------------------- ----------------------
VALUE
------------------------------
log_archive_dest_5                  string
```

```
log_archive_dest_6                   string

log_archive_dest_7                   string

log_archive_dest_8                   string

log_archive_dest_9                   string

NAME                                 TYPE
------------------------------------ ----------------------
VALUE
------------------------------

log_archive_dest_state_1             string
enable
log_archive_dest_state_10            string
enable
log_archive_dest_state_11            string
enable
log_archive_dest_state_12            string
enable

NAME                                 TYPE
------------------------------------ ----------------------
VALUE
------------------------------
log_archive_dest_state_13            string
enable
log_archive_dest_state_14            string
enable
log_archive_dest_state_15            string
enable
log_archive_dest_state_16            string
enable
log_archive_dest_state_17            string

NAME                                 TYPE
------------------------------------ ----------------------
VALUE
------------------------------
enable
log_archive_dest_state_18            string
enable
log_archive_dest_state_19            string
```

```
enable
log_archive_dest_state_2             string
enable
log_archive_dest_state_20            string
enable

NAME                                 TYPE
------------------------------------ ----------------------
VALUE
------------------------------
log_archive_dest_state_21            string
enable
log_archive_dest_state_22            string
enable
log_archive_dest_state_23            string
enable
log_archive_dest_state_24            string
enable
log_archive_dest_state_25            string

NAME                                 TYPE
------------------------------------ ----------------------
VALUE
------------------------------
enable
log_archive_dest_state_26            string
enable
log_archive_dest_state_27            string
enable
log_archive_dest_state_28            string
enable
log_archive_dest_state_29            string
enable

NAME                                 TYPE
------------------------------------ ----------------------
VALUE
------------------------------
log_archive_dest_state_3             string
enable
log_archive_dest_state_30            string
enable
log_archive_dest_state_31            string
enable
```

log_archive_dest_state_4 string

enable

log_archive_dest_state_5 string

NAME TYPE

VALUE

---

enable

log_archive_dest_state_6 string

enable

log_archive_dest_state_7 string

enable

log_archive_dest_state_8 string

enable

log_archive_dest_state_9 string

enable

archive log list;

alter system archive log current 手动归档命令

默认归档日志存储在闪回区

14.查看归档日志文件信息（v$archived_log）

归档为两份，存储在下面两个文件夹

alter system set log_archive_dest_1 = 'location=F:\mxyyy\app\arch1';

alter system set log_archive_dest_10 = 'location=F:\mxyyy\app\arch2';

select member from v$logfile

15.日志文件的移动或重命名

1）允许移动（不是正在被使用时）

shutdown immediate

2) 利用操作系统命令移动或改名

3）更新控制文件

alter database rename dile '...log' to '...log';

startup mount

select member from v$logfile

alter database rename dile '...log' to '...log';

select member from v$logfile

alter database open;

select group#,status from v$log;

select group#,member from v$logfile;

alter database drop logfile member '...'(不要敲分号，敲回车）

16. 处理日志丢失

1）非当前日志文件丢失

select group#,status from v$log;

select group#,member from v$logfile;

删除第四个成员（非当前）

shutdown immediate

startup 会报错 ，运行不起来

startup mount

alter database clear logfile group 4;

alter database open;

2) 当前日志文件丢失

update scott.emp set sal=2800 where empno = 7839

shutdown immediate

startup mount

alter database clear logfile group 1;

recover database until cancel;

重做日志缓存

redo

do

undo ——对冲 回滚

redo ——重做，前滚，恢复

日志的作用：恢复

记录事务

事情，任务

ACID

atom 原子性

consistent 一致性

isolation 隔离性 只能看到一个事务开始或者结束之后的状态 不能看到中间过程

durable 持久性，永久性 在内存当中是临时的 存在文件中是永久的

backup

restore 还原

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：（主要分为3类）

1 用户进程

UI

名称解析

端口

协议

TCP

2 服务器进程

用户进程的代理

3 后台进程

SQL

1 QL(select)

2 DML(insert,update,delete,merge)修改的是用户数据

3 DDL(create,alter,drop)修改的是系统数据

4 DCL(grant,revoke)

5 TCL(commit,rollback,savepoint)

select

1 返回所有数据

select * from

2 投影

NULL 空值 三值现象

单引号 单引号里套单引号①两个连续的单引号 'I''m a student'②使用q'[xxx]'的形式q'[I'm a student]'

双引号

%匹配任意多个字 _匹配任意一个字

TOP N

FETCH FIRST n ROWS ONLY;

OFFSET n ROWS FETCH FIRST n ROWS ONLY;

FETCH FIRST n ROWS WITH TIES;

TP

回滚段的主要作用：

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：insert into t1 select * from t1 as of timestamp(systimestamp-interval'3'minute);//3 分钟前的数据

==========

锁：

第一个事务修改第三行 update t1 set name='Smith' where id =103;

Smith 在内存中存在

让第二个事务修改第四行 update t1 set id =4 where id =104;//可改

=>行锁

第三个事务修改第四行 update t1 set name='Lisi' where id =104;//进入等待状态，等第二个事务释放锁才可进行。

第二个事务再修改第三行=>死锁

自动解决死锁，自动回滚掉导致死锁的语句（没有做李四）

1

2

2//断掉

1

第 1 行出现错误:

ORA-00060: 等待资源时检测到死锁

死锁=>锁

1 卡死

desc v$lock

block: number 类型（1 阻住，0 未阻住）

select sid,block from v$lock where block=1;//SID747de

desc v$session 会话信息

username sid……

select sid,serial#,username from v$session where sid=747;//747 59756 hr

alter system kill session '747,59756';

1 执行了

==========

DDL 语句：产生系统数据的定义

(create,alter,drop,rename,truncate,comment)

数据库常见对象：

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。

视图：指向表

序列：自动产生

索引

synonym 同义词

desc dictionary/dict/table_name;

select index_name from user_indexes;

create view emp_info as select * from emp;

grant create view to scott;//赋予权限

create view emp_info as select * from emp;

select view_name from user_views;

create table EMP_INFO(id int);//不可创建同名的对象

create sequence s1;

create table s1(id int);//不可创建同名的对象

create sysnonym em for emp;

grant create sysnonym to scott;//赋予权限

create table em(id int);//不可创建同名的对象

表名不可和 同一用户下 相同名称空间 namespace 的 对象 同名。

select object_name,object_type,namespace from user_objects;

create table procedure(id int);//dui

function(id int);//dui

trigger(id int);//cuo

SQL> conn sys/admin as sysdba

已连接。

SQL> create user demo identified by admin;

SQL> conn demo/admin

ERROR:

ORA-01045: 用户 DEMO 没有 CREATE SESSION 权限; 登录被拒绝

SQL> conn sys/admin as sysdba

已连接。

SQL> grant create session to demo;

授权成功。

SQL> conn demo/admin

已连接

SQL> create table t1(id int);

create table t1(id int)

*

第 1 行出现错误:

ORA-01031: 权限不足

grant create table to demo;

SQL> create table t1(id int);

表已创建。

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;

unlimit

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;

pseudocolumn 伪列，存在表里，但查不到

伪军

Base 64 code//具体物理地址

1.文件号

2.块号

3.行号

rowid 为索引的指针，伪列

全本扫描

索引查找（目录）-指针

rowid 访问（知道 rowid 地址）

select empno,rowid from emp;

包 dbms_rowid.ROWID_RELATIVE_FNO

ROWID_BLOCK_NUMBER

ROWID_ROW_NUMBER

select empno,dbms_rowid.ROWID_RELATIVE_FNO(rowid) File#,

dbms_rowid.ROWID_BLOCK_NUMBER(rowid) Block#,

dbms_rowid.ROWID_ROW_NUMBER(rowid) Row#

from emp;

EMPNO	FILE#	BLOCK#	ROW#
7369	7	189	0
7499	7	189	1
7521	7	189	2
7566	7	189	3
7654	7	189	4
7698	7	189	5
7782	7	189	6
7839	7	189	7
7844	7	189	8
7900	7	189	9
7902	7	189	10

EMPNO	FILE#	BLOCK#	ROW#
7934	7	189	11

select name from v$datafile where file#=7;

NAME

F:\APP\LUYAO\ORADATA\DB18C\USERS01.DBF

select tablespace_name from dba_data_files

where file_name='F:\APP\LUYAO\ORADATA\DB18C\USERS01.DBF';//USERS

conn demo/admin

create table t3(id int,id char(5));//重复的列名

create table t4(id int,row_id char(5));//ok

==========

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）

nchar()

nvarchar()

字符集：字符的集合

ASCII

A 编码 01000001 65

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；

desc c

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 varchar2(30) default'空' not null)；

alter table c add c3 varchar(20);

3.修改字段

1)修改字段名

ALTER TABLE 表名 RENAME COLUMN 列名 TO 新列名;

alter table c rename column birthday to age;

2)修改字段类型

alter table c modify BITRH_DATE char(8);//要修改的列为空

alter table c modify BUYER_ID char(10)；//修改的列不为空，失败

3)修改字段大小

alter table c modify BUYER_NAME varchar(30);//改大,ok

alter table c modify BUYER_NAME char(29)//改小,ok

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.将表改为只读

select * from c;

alter table c read only;

select table_name,read_only from user_tables;

只读不可删/加数据

6.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';

7.删除表

drop table c;//还在回收站

flashback table c to before recyclebin;

drop table c purge;

select * from recyclebin;

==========

数据完整性：

1 代码

2 触发器

3 约束

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 表/user_cons_columns 字段:

select constraint_name,constraint_type,table_name

from user_constraints

where table_name='EMP';

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
FK_DEPTNO	R	EMP//外键
PK_EMP	P	EMP//主键

不知在何字段上

desc user_cons_columns

select column_name from user_cons_columns where constraint_name='PK_EMP';//empno

定义表级别的约束：关键字，约束名称，约束类型，字段

定义列级别的约束：关键字，约束名称，约束类型

create table employees(employee_id number(6) not null,

……

CONSTRAINT emp_emp_id_pk PRIMARY KEY(EMPLOYEE_ID));

定义外键

CONSTRAINT emp_emp_id_fk FOREIGN KEY(EMPLOYEE_ID)

REFERENCES departments(department_id) on delete cascade;

delete from dept where deptno=10;//报错

员工表中仍有在 10 号部分的

on delete cascade//级联删掉 仍在 10 号部门的员工

on delete set null//原 10 号部门的员工的 部门号 设为空（员工待分配）

check (salary>0)不符合这个公式就输不进去:

SQL> conn demo/admin

已连接。

SQL> create table emp(id int,name varchar(20),sal int);

表已创建。

SQL> insert into emp values(1,'Tom',3000);

ORA-01950: 对表空间 'USERS' 无权限

？？？

select * from user_sys_privs;

connect / as sysdba;

SQL> ALTER USER "ZHAOH" QUOTA UNLIMITED ON "USERS";

alter table emp add constraint sal_min check (sal>0);//添加约束

insert into emp values(2,'A',-3000);//负数插不进去了

alter table emp drop constraint SAL_MIN;//删除约束

数据+规则->有效，合理的

exceptions 异常表

==========

SQL

1.QL（select）分组、多张表连接、子查询方式

2.DML(insert update delete merge)增删改

3.TCL 事务 (commit,rollback,savepoint)

4DCL 数据控制语句(grant,revoke)

5.DDL 对象(create alter drop truncate rename comment)

管理部分

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

conn scott/tiger

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

视图的作用：

1.收集感兴趣的数据

2.屏蔽敏感数据

3.简化查询（复杂的语句变成视图）

4.简化权限的管理

385 页

视图的分类：

1.简单视图

2.复杂视图

创建视图

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

修改视图

create or replace view emp_info as select

empno,ename,sal,deptno from emp;

表达式 要起别名

update emp_info set deptno=20 where empno=;

create or replace

T1

A B C(NOT NULL)

V1

A B

drop view **;//与表没关系，相当于链接

伪列 rowid

ROWNUM 行号:对结果进行编号

select rownum,ename from emp where rownum<=3;

select rownum,ename from (select rownum aa,emp.* from emp)

where aa>=10;//对 rownum 取别名

==========

管理例程

1.关闭例程（4 种模式）

1）正常关闭(normal)

2) 事务性关闭(TRAMSACTIONAL)

3) 立即关闭(immediate)不需要等事务结束

4）中止退出(abort) 丢数据

shutdown abort

3 个需要进行检查点检查

shutdown immediate

2.启动（3 个阶段）

1)启动例程(内存/进程结构)：分配内存，同时启动后台进程

startup nomount

条件：需要访问初始化参数文件（默认：oracle 主目录下的 database 目录 18.3.0、database/SPFILEDB18C.ORA/非 windows：

oracle 的 dbms 目录下）

 conn sys/admin as sysdba

select status from v$instance;

nomount 状态下只能访问一部分动态性能视图（来自内存的内容）

select * from v$sga; //ok

select name from v$datafile;//（来自控制文件，报错）

2)加载数据库(mount 阶段)

alter database mount;

条件：需要访问控制文件 show parameter control_files 可以访问所有的动态性能视图，访问不了数据文件

app/oracle/oradata/db18c/control01.CTL

3)打开数据库（open 阶段）

①alter database open;

条件：open 阶段要访问 联机重做日志文件和数据文件

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

从后往前看

app/oracle/oradata/db18c

可以访问所有的数据

空闲例程（oracle 未启动）->startup（默认 startup open）

②以只读方式打开数据库

startup open read only;

select name,open_mode from v$database;

select current_scn from v$database;//随时间变化的序列

③以受限模式打开数据库

startup restrict;

desc v$instance;

select instance_name,logins from v$instance;

$cls 清空命令行

让有些用户连，有些用户不连（有 restriction 权限的可以访问）

grant restriction session to hr;

禁用受限模式

alter system disable restricted session;

启用受限模式

alter system enable restricited session;

3.配置例程：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

init<sid>.ora --initdb18c.ora

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

show parameter spfile 查看具体的值，有 spfile，无则全是 pfile

敲命令修改 spfile<sid>.ora --spfiledb18c.ora

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 包含 share 的参数全部都显示，匹配

②修改初始化参数

不要改这个文件 18.3.0/database/spfiledb18c.ora

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=memory 内存/spfile/both

memory:只在内存改，不在文件改（临时生效）

spfile:先写在文件中，下次重启生效

both:立即生效，同时在文件中修改

alter system set shared_pool_size=160m scope=memory;

alter system set sga_max_size=5000m scope=spfile;

==========

alter session set nls_language=english;

③将初始化参数还原成默认值

show parameter shared_server

alter system reset shared_server;复位

shutdown immediate

startup

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

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

数据库管理：

手工创建一个数据库：（demo）

1.创建 oracle 服务　之前有的服务 oracleservicedb18c

命令：oradim（管理员权限）

oradim -new -sid demo

输入 Oracle 服务用户的口令:626yao817

2.将 demo 设为当前 sid

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;

控制文件（创建数据库时，控制文件会自动创建）

连接重做日志文件

数据文件

三个表空间必须在创建数据库时创建（system,sysaux,undo）

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

数据字典表

desc user_tables 此对象不存在

alter session set nls_language=english;

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

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

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：

@?/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。。。。。。。。。。 ?

==========

修改 sqlplus 默认提示符

SQL> set sqlprompt "_user'@'_connect_identifier> "

SYS@db18c>

管理表空间和数据文件

1.数据库存储的结构层次

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

database pro 查数据库视图

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

show parameter cache

db_2k_cache_size value=0 内存中没有地方可以放 2k 的块

alter system set db_2k_cache_size=16m;

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

//创建大块

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

内存中间开辟 16k 的块

alter system set db_16k_cache_size=16m;

4.表空间的空间管理

1）本地管理（常用，减少数据字典（系统数据）的使用--避免回滚，每个区都一样大【没有碎片】）

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

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

3）临时表空间（排序）temporary

查看表空间所属类型：contents

desc dba_tablespaces

select tablespace_name,contents from dba_tablespaces;

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;

show parameter undo_tablespace;

select tablespace_name,contents from dba_tablespaces;

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

select tablespace_name,status from dba_tablespaces;

online->只读：

create table scott.test1(id int,name varchar(20)) tablespace smalltbs;//创建一张表放到表空间中

insert into soctt.test1 values(1,'Tom');//表插入一条记录

commit;

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;

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

2)当前的撤销表

3)临时表空间

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

只读->读写

alter tablespace smalltbs read write;

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;

cmd：D:dir

del D:\……\demo\small01.dbf 要手动在 OS 中删除数据文件

SQL>create table scott.test1(id int) tablespace bigtbs;

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

create tablespace test 缺datafile/tempfile

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

create tablespace test;

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'……'

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

移回去:

alter database move datafile'D:app/omf/data01.dbf' to 'D:app/oracle/oradata/demo/userdata01.dbf' keep;加keep原来文件还有，相当于copy

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：

一、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）

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

①sys：

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

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）

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;

②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

set oracle_sid=demo

sqlplus sys/admin1#3 as sysdba

sqlplus scott/tiger as sysdba

set sqlprompt "_user'@'_connect_identifier> "

==========

授权

权力：（全局，用户）

权限：（局部，资源）

给用户——能干这件事的最小权限原则

1.系统权限：建表

2.对象权限

3.权限的传递规则：

①对象权限是连带的

A、B、C三用户（A附权给B，B附权给C，当A收回B的权限时，C的权限也被收回）

create user a identified by a;

create user b identified by b;

授权登陆

select any table

select on scott.emp

grant select on soctt.emp to b;//权限不足

grant select on soctt.emp to a with grant option;//传递对象权限

revoke select on scott.emp from a;//收回a的权限

desc scott.emp

②系统权限不连带

A、B、C三用户（A附权给B，B附权给C，当A收回B的权限时，C的权限不被收回）

grant create table to a;//SYS给权限

create table t1(id int);//A用户

show parameter defer

alter system set deffered_segment_creattion//延时创建

grant create table to b;//A

grant create table to a with admin grant option;//sys，传递系统权限（建表）

revoke create table from a

4.角色（权限的集合）的作用：

①简化权限的管理，减少授权次数；

②动态权限管理（权限是静态的，有就可以干；用户有角色时有时可干、有时不可干）

角色一定要处于激活状态(登陆之后激活)

1）用户自定义角色

```
create role r1;
create role r2 identified by r2;
grant select on scott.emp to r1,r2;//都对此表有访问的权限

grant r1 to a;//r1 角色指派给 A，A 用户默认角色
```

默认角色在登录时激活，

```
desc scott.emp;
conn a/a;
desc scott.emp;

alter user a default role none;非默认角色（未登陆前可以，登陆后不可以，用 set role r1;命令激活非默认角色）

grant r2 to b;
desc scott.emp;//不可
conn b/b
desc scott.emp;//不可，角色不可激活（r2 角色带口令）
set role r2 identified by r2;//激活带口令的角色
desc scott.emp//可以
```

2)预定义角色（oracle 自带角色）查数据字典 PDFP344 页

角色名:connect\resource\dba……

3）应用程序角色（通过某应用程序激活角色，不用登陆激活）

```
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，角色不需要事先指派给某一用户
//创建应用程序/过程
create or replace procedure scott.p1
   authid current_user as
begin
    dbms_session.set_role('APP_R1');
end;
/
show error
```

grant execute on scott.p1 to a,b;//赋执行权限

conn a/a;

desc scott.emp//不行，无权限

execute scott.p1;//A用户

desc scott.emp//√仅跑完过程的当前登陆之后获得权限，退出后重新登陆后不跑过程还是无权限

drop role r1;//删角色

desc dba_roles//看有哪些角色

5.角色传递规则

sys：

grant r1 to a;

a用户:

grant r1 to b;

sys:

revoke r1 from a;

b:????????类似系统权限，不连带，仍可以

SYS@demo> create role r1;

角色已创建。

SYS@demo> grant select on scott.test to r1;

SYS@demo> create user a identified by a;

用户已创建。

SYS@demo> create user b identified by b;

用户已创建。

SYS@demo> grant r1 to a;

授权成功。

SYS@demo> grant create session to a;//给a赋登陆权限

SYS@demo> grant create session to b;//给b赋登陆权限

SYS@demo> grant r1 to a with grant option;

SYS@demo> conn a/a

已连接。

A@demo> grant r1 to b;

A@demo>select * from scott.test;

SYS@demo> conn b/b

已连接。

B@demo>select * from scott.test;

B@demo> conn sys/admin1#3 as sysdba

已连接。

SYS@demo> revoke r1 from a;

b还有权限SELECT scott.test

==========

审核

1.默认审核 （强制审核）：重大的数据库事件会记录

查：访问警告日志文件

2.标准数据库审核

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;

②审核对象：对象审核

audit delete on scott.emp;

③审核语句：和某种行为有关

audit create trigger;

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;//会话中只要是 update 记录，只要 1 个

audit update on scott.emp by access//每次都记录

****

show parameter audit_trail

select count(*)from aud$

无审核记录

设置审核

audit select,insert,update,delete on scott.emp by access;//设置对象审核，每干 1 次记录 1 次

grant select,insert,update,delete on scott.emp to a,b;

select * from scott.emp;

select count(*)from aud$

1

查视图 des dba_audit_trial

alter session set nls_date_format='yyyy-mm-dd hh24:mi:ss';

col username for a10;

col action_name for a20;

select timestamp,username,action_name from dba_audit_trail;

3.基于值的审核（通过触发器发现）

可以自动执行的一段代码，不可以手工调用（和 procedure 不同）

①dml 触发器：触发条件为 dml 语句

```
conn scott/tiger
create table soctt.tr_test1(……);
//一般触发器不创建在 sys 下
create or replace tigger scott.tr1
after update on scott.emp
begin
     insert into soctt.tr_test1 values('Be changed!');
end;
/
select * from scott.tr_test1;无
update scott.emp set sal=2800 where empno=7369;
select * from scott.tr_test1;//有
rollup
select * from scott.tr_test1;无

update scott.emp set sal=2800 where mgr=7698;//动了 5 条
select * from scott.tr_test1;//一条记录
```

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;
/
```

修改触发器（新旧值）

```
after update on scott.emp referencing old as o new as n for each row when(n.sal>5000)有这个条件后触发
```

练习：

```
create table sal_change(ename varchar2(10),old_sal number(7,2),new_sal number(7,2));
```

利用触发器记录员工的工资改变情况!!

```
drop tigger tr1;

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

```
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:

查询 dba_fga_audit_trail 结果

```
desc dba_fga_audit_trail
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 名
```

导入/导出

```
imp/exp
```

exp：

1.交互式（问什么答什么）

```
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

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

3)方案 owner=

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

4)表 tables=dept;

5)导出表的子集（只要表的一部分）

D:\exp>exp scott/tiger file=sal2500.dmp tables=emp query='where sal>2500'//缺少右引号（命令行>默认为输出）

D:\exp>dir

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

6)使用参数文件

D:\exp>exp scott/tiger file

nodepad 中写，并保存 p1.par:

userid=scott/tiger

file=p1.dmp

tables=emp

query='where sal>2500'

D:\exp>exp parfile=p1.par

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;//导入时，需先创建这张表 create table，导入出错->忽略创建错误

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;

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

//创建用户 hr

create user hr identified by 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

并行---提高性能

1.大任务

2.足够多的空闲资源

1.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

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

sys@db18c>grant read,write on directory expdp_dest to scott;

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

2.remap_table=emp:emp1　　表

remap_schema=scott:hr　　（源：目标）换到另一个用户下

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.empl*（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:empl 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：日志文件中一对象号：二进制/16 进制记日记，人看不懂，对照数据字典文件一一能对应看懂

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

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;

RMAN 下的整库备份和恢复（归档模式）

1　创建测试表

create table scott.hotbak(a varchar(30)) tablespace users;

insert into scott.hotbak values('Before Backup');

2　备份前的准备

备份 spfile

备份控制文件

rman target /

RMAN> show all;

RMAN>CONFIGURE CONTROLFILE AUTOBACKUP ON;

备份数据文件

channel(通道)

RMAN>CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT 'd:\demobak\%d_%u_%T';

RMAN> CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO 'D:\demobak\auto\%F';

备份归档日志文件

RMAN> alter system archive log current;

3 开始备份

RMAN> backup database plus archivelog;

4 增加新的记录

insert into scott.hotbak values('After Backup');

commit;

5 复制联机重做日志文件

新建了 D:\demobak\log 文件夹

SYS@demo>select member from v$logfile;

复制需要的日志文件至 D:\demobak\log 文件夹

6 破坏数据库

运行 DBCA,删除数据库

恢复

7 恢复初始化参数文件(spfile)

oradim -new -sid demo (用管理员身份)

rman target /

startup nomount

restore spfile from 'D:\demobak\auto\C-3748002068-20190419-01';

8 创建相应的目录结构

D:\app\oracle\admin\demo

D:\app\oracle\oradata\demo

D:\app\oracle\fast_recovery_area\demo

重新启动

shutdown immediate

startup nomount

9 恢复控制文件

restore controlfile from 'D:\demobak\auto\C-3748002068-20190419-01';

alter database mount;

10 恢复数据文件

restore database;

11 将前面复制的日志文件复制到D:\app\oracle\oradata\demo文件夹

recover database;

12 打开数据库

RMAN> alter database open resetlogs;

13 检查数据

select * from scott.hotbak;

结束!