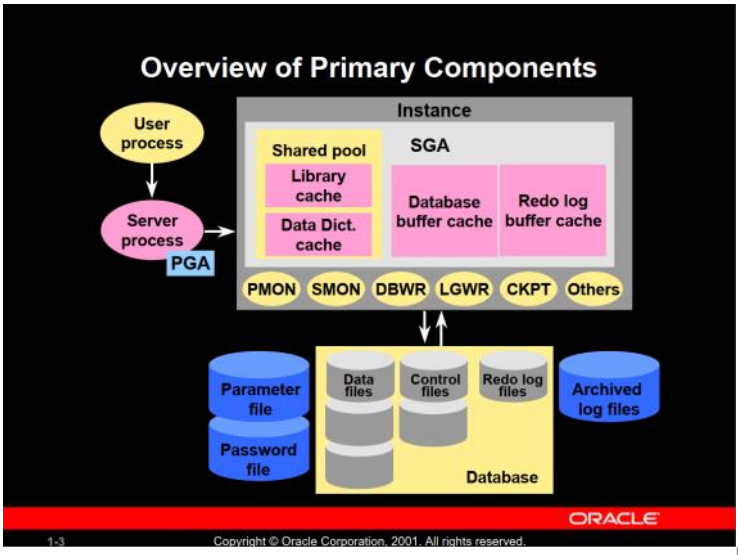
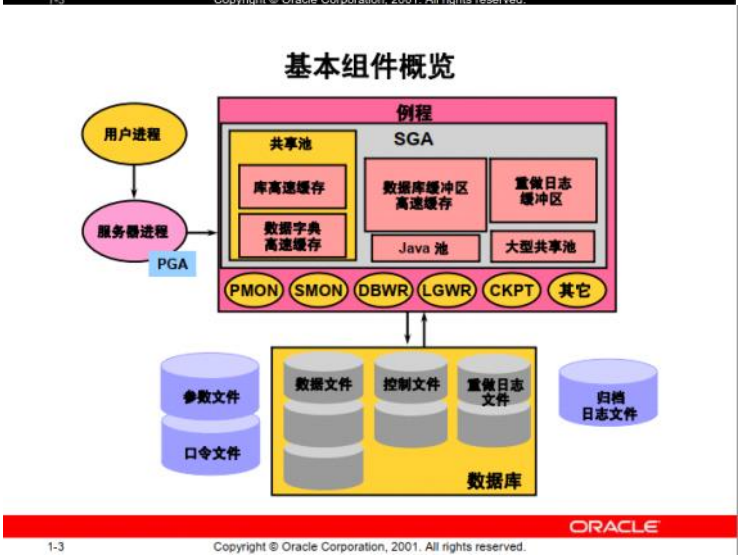
oracle体系结构组件

2019年3月1日 14:51

ORACLE数据库由例程和数据库（文件的集合）组成

sys（sysdba）——管理整个ORACLE服务器，包括例程和数据库
sys（sysoper）——管理instance
system——管理database





Instance，例程

内存结构

1）SGA，system global area，系统全局区（该系统内所有会话都能访问）
代码+数据（用户数据+系统数据）
2）PGA，会话全局区，创建会话时分配给的内存，仅能被一个会话访问

shared pool共享池
库缓存（SQL区，PL/SQL区）PL：Procedural Language，过程化语言
数据字典缓存
数据字典
动态性能视图（指向内存，控制文件） v$
数据字典视图（指向数据文件） dictionary

--修改口令
alter user <username> identified by <password>
--用户账号解锁
alter user <username> account unlock
--查看表空间数据文件
select * from DBA_DATA_FILES;
--查看临时表空间数据文件
select * from DBA_TEMP_FILES;
--查看表空间
select * from DBA_TABLESPACES;
select current_scn, checkpoint_change# from v$database
current_scn：随时间变化
checkpoint_change#不变

desc v$bgprocess：后台进程

（修改后）commit 提交，发信号给LGWR写盘

--创建表空间
create tablespace sales datafile'...' size 10m;
--创建表
create table scott.customers(id int) tablespace sales;

USER_：用户自己的表
ALL_：用户有权限访问的表，数量较多
DBA_：DBA能访问的表，数量更多

数据库要进行自动更新，加载最新统计信息，确保进行正确的优化

重做日志缓存P45
redo，undo
日志的作用：恢复
日志记录事务

ACID：
A：原子性，完全成功或完全失败
C：一致性，事务结束之后数据要处于一致的状态
I：隔离性，只能看到一个事务开始之前或结束后的状态，靠锁实现
D：持久性，事务一旦提交就是永久更改（缓存 -> 磁盘）

表——表空间——物理上由数据文件构成
查询表空间中数据文件具体位置：select name from v$datafile
创建表空间：create tablespace <name> datafile 'xxx.DBF' size 10m;
创建表：create table scott.customers(id int, name varchar(20));
插入数据：insert into scott.customers values (1, "Tom");

restore：还原
recover：恢复
alter database create datafile 5;
recover datafile 5;
alter tablespace sales online;

数据文件+日志文件

进程结构
用户进程
UI，不能直接访问服务器，要通过服务器进程
名称解析：端口+协议（TCP：传输控制协议）
监听程序用于连接用户进程和服务器进程
用户进程与服务器进程通信称为连接，服务器进程处理事务称为会话
服务器进程
相当于用户进程的代理
专用服务器进程：为某一用户进程所专用，效率低，质量高。
用户数<200时使用
共享服务器进程：支持的连接数更多。
200~3000
>3000需要配置中间件
后台进程
PMON：进程监视器进程
SMON：系统监视器进程
DBWR：数据库书写器进程
LGWR：日志书写器进程，把重做日志缓存中的数据写到重做日志文件
CKPT：检查点进程
检查点的作用：同步所有的数据文件
select name, checkpoint _change# from v$datafile;
同步所有的控制文件

select name, checkpoint_change# from v$database

发送信号通知DBWR写盘

数据库

初始化参数文件，pfile，spfile，存放了数据库启动时的初始化配置，包括SGA的大小等

口令验证文件

控制文件，mount，nomount，初始化参数文件中指定了控制文件的位置，控制文件指定整个数据库的状态

数据文件，存放用户数据和系统数据

联机重做日志文件，记录所做的操作，用于恢复数据库

归档日志文件

SQL-QL(SELECT)

2019年3月5日 10:13

QL: 查询，select
DML: 修改用户数据，insert, update, delete, merge
DDL: 修改系统数据，create, alter, drop, rename, truncate, comment
DCL: 数据控制语言，grant select on scott.emp to hr授予权限，revoke...撤销权限
TCL: 事务控制语言，commit, rollback, savepoint

help index：显示sqlplus命令
help <符号>：显示帮助

返回所有数据
　　select * from（desc不是SQL语句，是sqlplus命令，可通过help index获得命令列表，命令在不产生歧义的情况下可以缩写）
投影
　　select 字段, 字段 from...
选择

NULL三值现象，NULL表示不知道，SQL语句中应该是IS NULL，IS NOT NULL

基本真值表 [编辑]

下面是给有true/false/unknown状态的系统的一些逻辑运算的真值表。

A	*B*	*A* ∨ *B*	*A* ∧ *B*	¬ *A*
True	True	True	True	False
True	Unknown	True	Unknown	False
True	False	True	False	False
Unknown	True	True	Unknown	Unknown
Unknown	Unknown	Unknown	Unknown	Unknown
Unknown	False	Unknown	False	Unknown
False	True	True	False	True
False	Unknown	Unknown	False	True
False	False	False	False	True

双引号：别名
单引号：字符串中有 ' 的则要连续使用两个 ' 或使用q'[]'
　　引号嵌套：select q'[I'm a student]' from students

去除重复：select unique departno from emp，或distinct

select * from t where name like 'AB_%' escape '/'　不将'/'作为通配符
%：0或多个字符
_：1个字符

找出工资最多的三人
　　select empno, empname, sal from emp order by sal desc fetch first 3 rows only;
取第4和第5
　　select empno, empname, sal from emp order by sal desc offset 3 rows fetch first 2 rows only;
如果排序后有相同值，with ties将保留相同值
　　select empno, empname, sal from emp order by sal desc fetch first 10 rows with ties;

20190308
和用户交互：提交变量

&：
select empno from emp where empno = 123
select empno from emp where empno = &工号，则会要求输入工号的值
select empno from emp where empno = upper('&xm'); 没有upper则要大写
&&：
接收键盘输入并同时define，取消定义用undefine
select empno, ename, &&c3 from emp where empno = &c3;
定义变量
define g = 7369
查看定义
define g

sqlplus配置
查看配置 show：show autocommit
修改配置 set：set autocommit on
导入sql文件：SQL>@home/oracle/a.sql

字符串相关函数

concat(stra,strb)：字符串连接
substr(str, 1, 5)：取str中1~5位
substr(str, -2, 2)：取最后两位
instr(str, char)：找到str中char的位置，也可instr(str, char, num)，找到第num个char的位置

select buyer_id, substr(buyer_name, instr(buyer_name, ' ')+1) from buyers;

层次查询

select (lpad(' ', (level-1)*2)||ename) as ename from emp start with empno = 7839 connect by prior empno = mgr;
start with：定起点
connect by：定方向
lpad：定格式

处理数据的函数

round(num, n)：将num四舍五入，n为保留的小数位数
trunc(num, n)：取整，n=-1则舍掉个位
ceil(num, n)：向上取整！！！没有n
floor(num)：向下取整
mod(a, b)：取余数，a%b

时间相关的函数

sysdate
current_date：当前日期、时间
systimestamp：所有的时间信息
current_stamp

nls参数：国家语言支持

desc nls_session_parameters：查询nls参数
select parameter, value from nls_session_parameters;
nls_data_format：修改日期格式参数
alter session set nls_data_format='yyyy-mm-dd hh:mi:ss';
nls_language：修改语言参数
nls_territory：地区
alter session set nls_territory = 'UNITED KONGDOM'：改变国家地区

用两位数字表示年份时存在歧义

yy格式：和系统日期处于同一个世纪

rr格式：离系统时间近的那个年份

SELECT SYSTIMESTAMP FROM dual; 时间戳

用日期作运算，可能发生数据类型改变

month_between('01-SEP-05','11-JAN-03')，01年9月5

next_day('09-SEP-05', 'FRIDAY')

TO_CHAR：日期转换为字符串

练习

****

SELECT LAST_NAME, (TRUNC(MONTHS_BETWEEN(SYSDATE, HIRE_DATE)/12))AS YEARS, (TRUNC(MOD(MONTHS_BETWEEN(SYSDATE, HIRE_DATE),12)))AS MONTHS FROM EMPLOYEES ORDER BY YEARS DESC, MONTHS DESC FETCH FIRST 5 ROWS WITH TIES;

LAST_NAME	YEARS	MONTHS
De Haan	18	1
Baer	16	9
Higgins	16	9
Mavris	16	9
Gietz	16	9

练习：
hr.employees
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

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，按照YEARS+MONTHS降序排序

```
select last_name,trunc(months_between(sysdate,hire_date)/12) years,trunc(mod(months_between(sysdate,hire_date),12)) months
from employees
order by 2 desc,3 desc
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 employees
order by 2 desc,3 desc
offset 5 rows
fetch first 20 rows only;
```

****

转换函数

隐式转换：select 3+'4' from dual;

显式转换TO_NUMBER, TO_CHAR, ……

SELECT TO_CHAR(SYSDATE, 'yyyy-mm-dd') FROM dual;

SELECT EMPNO, ENAME, TO_CHAR(SAL,'L99,999.99') AS SAL FROM EMP;

在键盘上输入日期时：TO_DATE('1980-09-01','yyyy-mm-dd')

处理空值NULL的一些函数

NVL(str1, str2)：若str1不为空则显示str1，否则显示str2

MGR为空则显示BOSS：

SELECT EMPNO, ENAME, NVL(TO_CHAR(MGR), 'BOSS') AS MGR FROM EMP;

NVL2(str1, str2, str3)：str1不为空显示str2，否则显示str3

COMM为空则只显示SAL，否则显示SAL+COMM：

SELECT ENAME, SAL, COMM, NVL2(COMM,SAL+COMM,SAL) AS INCOME FROM EMP;

NULLIF(str1, str2)：如果str1和str2不同则显示str1，否则显示为空

COALESCE(str1, str2, …, strn)：找到第一个非空值为止

同上

SELECT ENAME, SAL, COMM, COALESCE(SAL+COMM, SAL) AS INCOME FROM EMP;

条件表达式

CASE：

范围（条件）判定

CASE(WHEN a<100 THEN …

WHEN b<100 THEN …)

DECODE：

显示员工总数、2001年入职人数、2002年入职人数、2003。。。

```
select count(*) total,
sum(decode(to_char(hire_date,'yyyy'),2001,1,0))"2001",
sum(decode(to_char(hire_date,'yyyy'),2002,1,0))"2002",
sum(decode(to_char(hire_date,'yyyy'),2003,1,0))"2003",
sum(decode(to_char(hire_date,'yyyy'),2004,1,0))"2004",
sum(decode(to_char(hire_date,'yyyy'),2005,1,0))"2005"
FROM hr.employees;
```

或

```
select sum(case to_char(hiredate, 'yyyy')
when '1981' then 1
else 0 end) "1981"
from EMP;
```

AVG, SUM, MAX, MIN, DISTINCT, UNIQUE

select max(salary) from EMP; // min同理

select greatest(1,4,7) from dual; // least同理

WHERE：记录筛选

HAVING：分组筛选

ROLLUP：行小计

CUBE：列小计

GROUP BY：

SELECT BY列表中不是分组函数的所有列表都必须在GROUP BY子句中

GROUP BY列不一定要出现在SELECT列表中

导入外部数据

创建新表

CREATE TABLE ST(NAME VARCHAR(20), SUBJECT VARCHAR(20), SCORE INT);

写数据（st.csv）

name	subject	score
Sam	Chinese	90
Sam	English	10
Tom	Chinese	10
Tom	English	90

写控制文件（st.ctl）

```
load
        infile 'D:/st.csv'
        into table hr.st
        fields terminated by ','
        (name char, subject char, score integer external)
```

加载控制文件

cmd：sqlldr control = "控制文件路径"

查询（行转列）

SELECT NAME, SUM(DECODE(SUBJECT,'CHINESE',SCORE,0)) AS CHINESE,

SUM(DECODE(SUBJECT,'ENGLISH',SCORE,0)) AS ENGLISH FROM ST GROUP BY NAME;

访问多张表中的信息

1、连接

1）内部连接SQL2：

```
SELECT B.BUYER_ID, B.BUYER_NAME, S.QTY
FROM BUYERS B, SALES S
WHERE B.BUYER_ID = S.BUYER_ID

SELECT B.BUYER_ID, B.BUYER_NAME, S.QTY
FROM BUYERS B INNER JOIN SALES S
WHERE B.BUYER_ID = S.BUYER_ID
```

2）外部连接（左、右、全）

左外连接SQL2：等值的元组加上主表（左操作数）中不匹配的元组

```
SELECT B.BUYER_ID, B.BUYER_NAME, S.QTY
FROM BUYERS B, SALES S
WHERE B.BUYER_ID = S.BUYER_ID(+)
```

SQL3:

```
SELECT B.BUYER_ID, B.BUYER_NAME, S.QTY
FROM BUYERS B LFET OUTER JOIN SALES S
WHERE B.BUYER_ID = S.BUYER_ID
```

右外连接：等值的元组加上被连接表（右操作数）中不匹配的元组

```
SELECT B.BUYER_ID, B.BUYER_NAME, S.QTY
FROM BUYERS B, SALES S
WHERE B.BUYER_ID (+)= S.BUYER_ID
```

全外连接，等价于对一张表先左连接再右连接

```
SELECT B.BUYER_ID, B.BUYER_NAME, S.QTY
FROM BUYERS B FULL OUTER JOIN SALES S
WHERE B.BUYER_ID = S.BUYER_ID
```

3）多表连接

```
SELECT B.BUYER_ID, P.PROD_NAME, S.QTY
FROM BUYERS B, SALES S, PRODUCT P
WHERE***
```

4）自连接，给同一张表起两个别名

```
SELECT A.BUYER_ID AS BUYER1, B.BUYER_ID AS BUYER2, A.PROD_ID
FROM SALES A, SALES B
WHERE A.PROD_ID = B.PROD_ID
AND A.NUYER_ID<B.BUYER_ID;
```

5）交叉连接，返回笛卡尔乘积

```
SELECT B.BUYER_NAME, S.QTY FROM BUYERS B, SALES S;
SELECT B.BUYER_NAME, S.QTY FROM BUYERS B CROSS JION SALES S;
```

6）自然连接，自动找到两张表的相同字段进行匹配

2、子查询

显示职工姓名、汇报经理

```
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 e, employees m
where e.manager_id=m.employee_id;
```

块

匿名块（*必须）

declare（表示声明部分开始）

……

begin（表示可执行部分开始）*

……
exception（表示异常处理部分开始）
……
end（表示pl/sql块结束）*
例：

```
begin
    for i in 1...10 loop
        insert into t values(i);
    end loop
end
```

命名块
Oracle对象，以编辑好的形式存在数据库中，需要时调用，效率较高。
先创建对象

```
create procedure p3
as
begin
    for i in1..10 loop
        insert into t values(i);
    end loop;
end;
/
```

查看已有的命名块：

```
desc user_procedures;
select object_name, object_type from user_procedures where object_type='procedure';
desc user_source;
select text from user_source where name='p3';
```

执行：

```
execute p3;
call p3();  call需要传参，无参数时括号不能省略
```

块结构
声明部分，可选。
可执行部分，完成主要功能，必须。
异常处理部分，捕获异常，可选。
循环
无条件循环：进入循环体不需要任何语句，循环体内有退出的条件（IF xxx EXIT;）。
条件循环：符合条件则进行循环。
固定次数循环

```
begin
    for i in1..10 loop
        insert into t values(i);
    end loop;
end;
/
```

用斜杠表示执行。

第一列输出123456，第二列输出678910

```
SELECT a.ID, b.ID FROM t a, t b WHERE a.ID+(SELECT COUNT(*) FROM t)/2=b.ID;
INSERT INTO T VALUES (11);
SELECT a.ID, b.ID FROM T a, T b WHERE a.ID=b.ID(+)-ROUND((SELECT COUNT(*) FROM T)/2) AND a.ID<=ROUND((SELECT COUNT(*)
    FROM T)/2);
```

子查询
嵌套子查询：

```
select empno, ename, 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'));
```

关联子查询：

查询工资最少的

```
select * from employees where salary = (select MIN(salary) from employees);
```

子查询规则：

将子查询放在括号里

将子查询放置在比较条件的右侧

只有在执行排序Top-N分析时，子查询中才需要使用ORDER BY子句

区分单行运算符（=，<，…）和多行运算符（in，any，all）

查询不是领导的员工

```
SELECT EMPLOYEE_ID, LAST_NAME FROM EMPLOYEES WHERE EMPLOYEE_ID NOT IN (SELECT MANAGER_ID FROM
EMPLOYEES);
```

得不到结果，因为有空值

```
SELECT EMPLOYEE_ID, LAST_NAME FROM EMPLOYEES WHERE EMPLOYEE_ID NOT IN (SELECT NVL(MANAGER_ID,0) FROM
EMPLOYEES);

SELECT EMPLOYEE_ID, LAST_NAME FROM EMPLOYEES E1 WHERE NOT EXISTS (SELECT * FROM EMPLOYEES E2 WHERE
E1.EMPLOYEE_ID=E2.MANAGER_ID);
```

查询高于部门平均工资的员工

```
SELECT EMPLOYEE_ID, LAST_NAME, SALARY, DEPARTMENT_ID FROM EMPLOYEES E1
WHERE SALARY > (SELECT AVG(SALARY) FROM EMPLOYEES E2 WHERE E1.DEPARTMENT_ID=E2.DEPARTMENT_ID GROUP BY
DEPARTMENT_ID);
```

处理多个结果集

union：将两个结果集单纯地合并，共同的记录只留1份

union all：相同的记录不合并

intersect：取交集

minus：A-B

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

SQL-DML，DCL，DDL

2019年3月15日 15:32

DML（数据库操纵语言）

insert

table t(id int, name, varchar(20))

insert into t values(1, NULL);显式插入空值

insert into t(NAME) values('Tom');隐式插入空值，双引号不行?

insert into demo select *from ...一次插入多行

update

update t set name="Jerry" where id=2;

delete

DELETE FROM EMP;表中内容全部删除

merge

提供有条件地在数据库表中更新或插入数据的功能，如果该行存在则执行update，否则执行insert。

```
MERGE INTO table_name table_alias
  USING (table|view|sub_query) alias
  ON (join condition)
  WHEN MATCHED THEN
    UPDATE SET
    col1 = col_val1,
    col2 = col2_val
  WHEN NOT MATCHED THEN
    INSERT (column_list)
    VALUES (column_values);
```

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);
```

默认值

alter table t1 modify NAME default "Tom";

赋予scott用户创建系统触发器的权限

grant ADMINISTER DATABASE TRIGGER to scott;

DCL

权限控制：grant，revoke

DCL,DDL,正常退出都会自动提交

异常终止会回滚

TCL：事务控制

commit：提交

rollback：回滚

savepoint：作一个标记点

xxx

savepoint aaa;

yyy

发现错误

此时rollback则xxx也会取消，回滚到aaa：rollback to aaa;

回滚段的作用

- 读一致性。事务进行前将需要修改的数据先存放在回滚段，并且不允许其它事务覆盖。此时进行select则读回滚段数据，即修改前的数据。更新值在另一个会话中不可见。
- 回滚。做错但提交之前，rollback可将回滚段中的数据拿回来。
- 闪回恢复。做错并提交后，rollback不能找回，但数据仍在回滚段。查回滚段并导入：insert into t1 select * from t1 as of timestamp(systimestamp-interval '3' minute);

锁

行锁：一个事务给表中一行上锁？此时其它事务要修改该行，则需要等待。检测到死锁会回滚产生死锁的语句。

查数据字典：

desc v$lock;

select sid, block from v$lock where block=1;（找block其它事务的sid）

desc v$session;

select sid, serial#, username from v$session where sid=xxx;（查上一步sid的用户名）

杀进程：alter system kill session 'xxx','yyy';（xxx为sid，yyy为serial#）

DDL，数据定义语言

create：create table xxx(id int);

alter

drop

rename

truncate

commet

数据库常见对象：表，视图（view），序列（squence），索引（index），同义词

表名要使用字母开头，1~30字符长，允许使用大小写字母，数字、_、$、#，不能存在同一用户下同名的对象（除了索引）。

表、视图、序列和同义词的名称空间是1，索引的名称空间是4，索引能和前四者之一重名。

创建表时不分配空间，使用时才分配空间（延时创建）。

空间分配，分配后可以构造表空间：alter user hr quota 5m on users; 5m为大小，也可unlimited。

alter user demo quota 5m on users;允许demo使用users的5m表空间

alter user demo quota unlimited on users;

grant create any table to scott;允许帮别人建表

视图:create view emp_info as select * from emp;

序列:create sequence s1;

索引:select index_name from user_indexes;

同义词:create synonym em for emp;

pseudocolumn

伪列?

物理地址

Base 64 code

文件号、块号、行号（rowid）

建表时不能建这些列

数据类型

char(xxx)：定长，性能好（比较字符串等）

varchar(xxx)：可变长

nchar：来自国家字符集
nvarchar：
表只能包括一个long类型，long类型可以被clog代替
RAW：字符的二进制形式

修改表

修改表名：rename
增加字段：alter table c add birthday date;
修改字段
改名：alter table c rename column birthday to birth_date;
修改类型：alter table c modify birth_date char(8)；修改的列必须为空
修改宽度：alter table c modify birth_date varchar(30);
删除字段
直接删除：alter table c drop column buyer_id;
先标记为未使用，再删除：alter table c set unused column buyer_id； alter table c drop unused column;
更改表的状态，将表改为只读：alter table c read only；读写：alter table c read write;
给表加注释：comment on table c is 'aaa';
查看注释：desc user_tab_comments; select comments from user_tab_comments where table_name = 'c';
立即彻底超级删除表：drop table c purge;

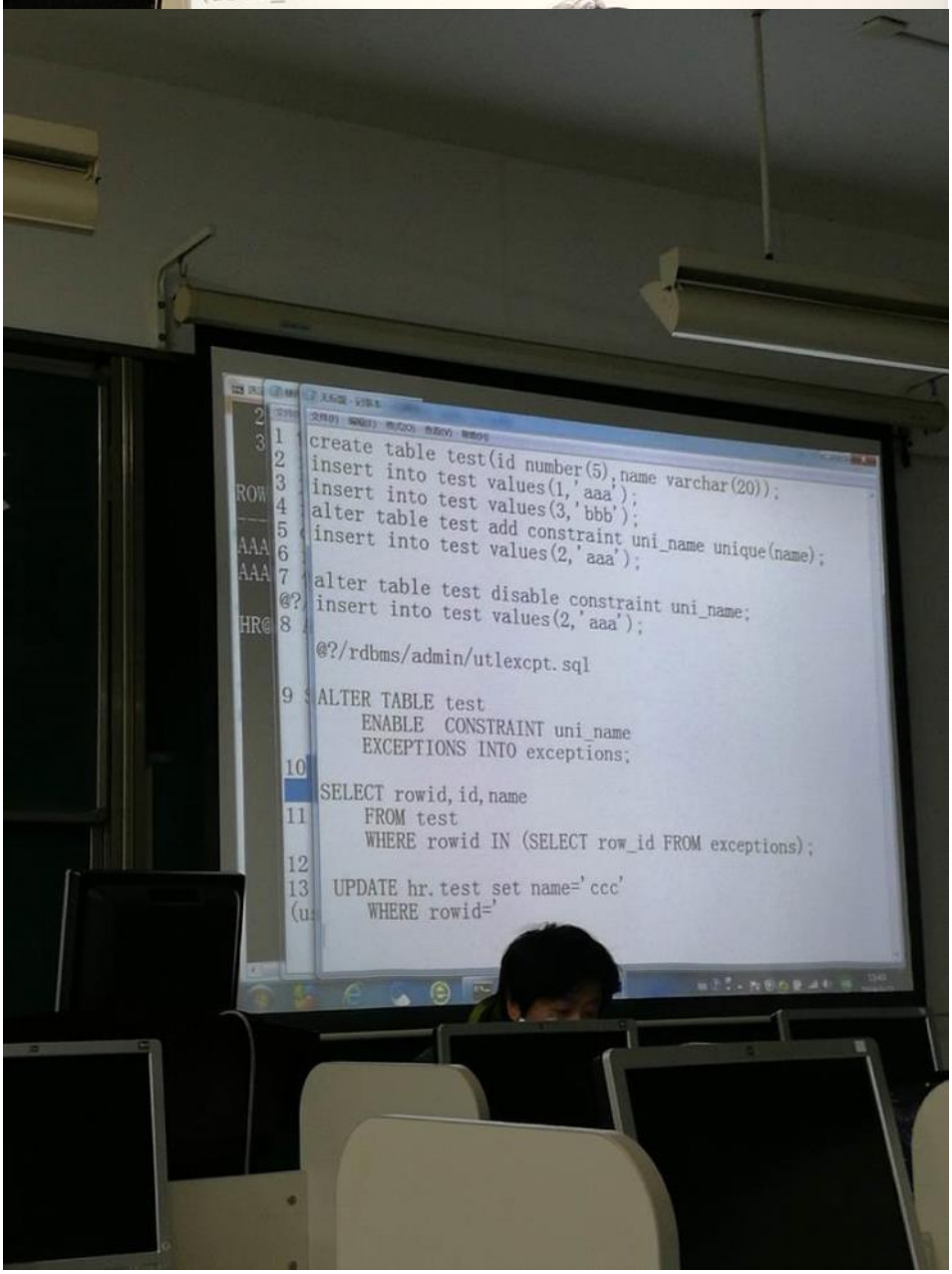
constraints，数据完整性
代码
触发器
约束（推荐，开销较小）
not null
unique
primary key
select constraint_name, constraint_type, table_name from user_constraints where table_name='emp';

on delete cascade：级联删除
on delete set null：将外键置为空

添加约束：alter table emp modify id not null;
查找约束：select constraint_name from emp
删除约束：alter table emp drop constraint sal_min;
禁用约束：alter table emp disable constraint sal_min;
启用约束：alter table emp enable constraint sal_min； 不成功，使用异常表，可以找到违反约束的数据。

SQL回顾
QL，查询语句，select
DML，数据操纵语句，增删改并
TCL，事务控制语句，提交，回滚，设置标记点
DCL，数据控制语句，权限grant，revoke
DDL，数据定义语句，create，alter，drop，truncate，rename，comment


1 创建hr.test表 (id number(5),name varchar(10))
2 insert (1,'aaa')(3,'bbb')
3 为test表添加一个unique(uni_name)约束(name)
4 insert(2,'aaa')
5 disable uni_name
6 insert(2,'aaa')(COMMIT;)
7 使用EXCEPTIONS表(DESC EXCEPTIONS
@?/rdbms/admin/utlexcpt.sql)
8 ALTER TABLE test
ENABLE CONSTRAINT uni_name
EXCEPTIONS INTO exceptions;
9 SELECT rowid,id,name
FROM test
WHERE rowid IN (SELECT row_id FROM exceptions);
10 UPDATE hr.test set name='ccc'
WHERE rowid='
11 ALTER TABLE test
ENABLE CONSTRAINT uni_name;
12 TRUNCATE TABLE exceptions;
13 在数据字典中查询约束信息
(user_constraints,user_cons_columns)


create table test(id number(5),name varchar(20));
insert into test values(1,'aaa');
insert into test values(3,'bbb');
alter table test add constraint uni_name unique(name);
insert into test values(2,'aaa');
alter table test disable constraint uni_name;
insert into test values(2,'aaa');
@?/rdbms/admin/utlexcpt.sql
ALTER TABLE test
ENABLE CONSTRAINT uni_name
EXCEPTIONS INTO exceptions;
SELECT rowid,id,name
FROM test
WHERE rowid IN (SELECT row_id FROM exceptions);
UPDATE hr.test set name='ccc'
WHERE rowid='

视图

视图的作用：

只收集感兴趣的数据

屏蔽敏感数据

简化查询

简化权限的管理

视图的分类：

简单视图

复杂视图

创建视图：

create view emp_info as select empno from emp;

修改视图：

create or replace view emp_info as select empno, ename from emp; 可在原视图后再添加一列

不能基于不存在的表创建视图

强制建立视图：force

create or replace force view emp_info as select empno, ename from aaa; 但创建的视图有编译错误，不能查询，但aaa表建立后可直接使用

加约束：

create or replace view emp_info as select empno, ename from emp where deptno=30 with check option;

之后不能再修改deptno

做只读视图：

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

select rownum,ename from emp where rownum<5；有数据

select rownum,ename from emp where rownum>1；没数据

rownum只对结果编号

要想完成语句2必须先把rownum提取出来

管理例程

关闭例程（四种模式）

正常关闭（normal），默认，等所有会话退出才能关闭

事务性关闭（transactional）：等待事务结束

立即关闭（immediate）：不等事务结束，回滚未完成的事务

终止关闭（abort），会丢失数据，前三种都经过检查点，数据不会丢失

启动（三个阶段）

启动例程（nomount阶段）：startup nomount

条件：需要访问初始化参数文件（Oracle主目录下的database文件夹init...文件）

select status from v$instance; 例程启动状态，只能访问一部分动态性能视图（来自内存的内容）

加载数据库（mount阶段）：alter database mount

条件：需要访问控制文件（初始化参数自定控制文件）

show parameter control_files：可以访问所有的动态性能视图

打开数据库（open阶段）：alter database open

条件：需要访问联机重做日志文件和数据文件

select member from v$logfile; 查看日志文件路径

可以访问所有数据，status变为open

以只读方式打开数据库

startup open read only;

select name, open_mode from v$database;

此时select current_scn from scott.emp; 所得值不再随时间变化，不能修改表中数据

以受限模式打开

startup restrict; 有权限的能连接数据库，其他人不能

sys和system仍有权限，授权：grant RESTRICTED SESSION to hr;

禁用受限模式：alter system disable RESTRICTED SESSION;

启用受限模式：alter system enable RESTRICTED SESSION;

配置例程

初始化参数文件，可用文本文件pfile启动，或使用二进制文件spfile

show parameter spfile; value有值则为spfile启动

pfile：init<sid>.ora，如initdb18c.ora

spfile：spfile<sid>.ora，如spfiledb18c.ora

查看初始化参数：show parameter

具体参数：show parameter shared_pool_size，可以匹配未知字符串（share则显示含share的参数）

修改初始化参数

alter system set shared_pool_size=128m;

有的参数无法动态修改（如sga_max_size），要先写到文件里，在重启的时候修改

select name, ISSYS_MODIFIABLE from v$parameter;

参数值为immediate的是可以立即生效的，为false的需重启后生效，deferred重新连接可生效

修改false的参数：scope=memory（写入内存临时生效）或spfile（写入文件）或both

show parameter shared_pool_size;

alter system set shared_pool_size=160m scope=memory; 在文件中未变，重启后仍未128m

alter systemset sga_max_siz=5000m scope=spfile; 内存中未变，但已写入文件，下次重启时生效

alter systemset sga_max_siz=5000m scope=both;

将初始化参数还原称默认值

show parameter shared_servers;

alter system reset shared_servers；重启后生效

修复错误的初始化参数

根据一个spfile创建一个pfile

create pfile from pfile;

根据pfile创建spfile

create spfile from pfile;

例程在启动时选择初始化参数文件的顺序

spfile<sid>.ora

spfile.ora

init<sid>.ora第三个找不到才报错

利用指定的初始化参数文件启动

startup pfile = d:\a.ora

手动创建数据库

2019年3月26日 8:35

1. 创建Oracle服务
   oradim -new -sid demo
2. 将demo设为当前SID
   set oracle_sid=demo
   (sqlplus / as sysdba 连接到空闲例程)
3. 创建/编辑初始化文件
   create pfile from spfile;
   E:\Oracle\WINDOWS.X64_180000_db_home\database下生成INITORCL.ORA的副本并改名为INITDEMO.ORA,修改内容将ORCL全部改为DEMO.
4. 创建相应的目录结构
   安装目录下创建相应文件夹
5. 启动例程
   startup nomount
6. 执行创建数据库的语句
   create database demo
   数据文件
   datafile 'E:\ORACLE\ORADATA\demo\SYSTEM01.DBF' size 400m
   sysaux datafile 'E:\ORACLE\ORADATA\demo\SYSAUX01.DBF' size 400m
   undo tablespace undotbs1 datafile 'E:\ORACLE\ORADATA\demo\UNDOTBS01.DBF' size 50m
   default temporary tablespace temp tempfile 'E:\ORACLE\ORADATA\demo\TEMP01.DBF' size 20m
   日志文件
   group 1 ('E:\ORACLE\ORADATA\demo\redo01.log' size 10m),
   group 2 ('E:\ORACLE\ORADATA\demo\redo02.log' size 10m),
   group 3 ('E:\ORACLE\ORADATA\demo\redo03.log' size 10m)

   控制文件（创建数据库时自动创建）
   联机重做日志文件
   数据文件

三个表空间必须在创建数据库时创建（system, sysaux, undo）

7. 创建数据字典视图（运行catalog.sql文件）
   @?/rdbms/admin/catalog
8. 创建spfile
   create spfile from pfile;
9. 创建口令验证文件
   file=H:\app\oracle\product\18.3.0\WINDOWS.X64_180000_db_home_2\database\PWDdemo.ora password=admin1#3
10. 创建Oracle内部包
    @?/rdbms/admin/catproc
11. 创建scott方案
    @?/rdbms/admin/scott
12. 加载产品用户概要文件信息
    @?/sqlplus/admin/PUPBLD.SQL
13. 配置监听器（服务器端）和服务名（客户端）
14. 配置em express
    select dams_xdb.gethttpport from dual;
    execute dams_xdb.sethttpport(6789);
    select dams_xdb.gethttpport from dual;
    http://hostname:6789/em

控制文件管理

2019年3月29日 13:36

控制文件的内容和用途

控制文件描述整个数据库的结构和状态，告诉Oracle如何访问存储，是二进制文件

查看控制文件的名称：

show parameter control_files

select name from v$controlfile;

查看日志文件内容：

oradebug setmypid;

oradebug dump controlf 3; 出现“？？？？？”：alter session set nls_language=english;

oradebug tracefile_name;

控制文件的复用

一个数据库中至少要有一个控制文件，最多可以有8个

修改初始化参数

alter system set control_files='D:\ORACLE\ORADATA\DB18C\CONTROL01.CTL','D:\ORACLE\FAST_RECOVERY_AREA\DB18C\CONTROL02.CTL','D:CONTROL03.CTL' scope=spfile; 前两个文件名为已有的，第三个自己添加，先写入spfile

shutdown immediate

复制出第三个控制文件

startup

重建控制文件

alter database backup controlfile to trace as 'D:\create_control.txt'

shutdown immediate

删除所有的控制文件

startup，会报错ora-00205

执行创建控制文件的语句

alter database open

管理日志文件

日志文件的用途：恢复

日志的工作机理：分组，写满之后循环覆盖

数据库日志模式

非归档（非归档）模式，直接覆盖，性能好，节省磁盘空间，但只能冷归档

存档（归档）模式，ARCHIVELOG，先归档出来，再覆盖，适合于7*24的企业

查看日志模式：select log_mode from v$database;

添加日志文件组

alter database add logfile group 4 ('D:\Oracle\oradata\DB18C\redo04a.log') size 10m;

查看日志文件组信息（v$log）

查看日志文件成员信息（v$logfile）

添加日志文件成员

alter database add logfile member 'D:\Oracle\oradata\DB18C\redo04b.log' to group 4; 不写大小，和原来的一样，是镜像

删除日志文件组

日志文件的状态：current，active，inactive，unused前两个状态下的不能删

alter database drop logfile group 1;

文件在操作系统下仍存在，需要手动删除

删除日志文件成员

alter database drop logfile member 'D:\ORACLE\ORADATA\DB18C\REDO02.LOG'，不成功，一个日志组中至少要有一个成员

文件在操作系统下仍存在，需要手动删除

清除日志文件内容

alter database clear logfile group n;

alter database clear logfile member '……';

OMF（Oracle管理文件）

通过配置初始化参数来实现

实现数据文件的OMF：alter system db_create_file_dest='……';　　创建文件时未指定目录则会放在这个目录下

日志文件OMF：日志文件是自动OMF的，默认存在闪回区

改变默认存储：alter system set db_create_online_log_dest_1='D:\Oracle\oradata\DB18C';

修改归档/非归档模式

必须在mount阶段

shutdown immediate

startup mount

alter database noarchivelog

修改数据库日志模式必须要处于mount阶段

修改成归档模式

修改成非归档模式

设置归档目的地并手工归档

设置初始化参数改变默认归档目的地

查看归档日志文件信息

日志文件的移动或重命名

保证其不正在被使用（不处于open阶段）

利用操作系统命令移动或改名

更新控制文件，告知Oracle日志文件换地方

打开数据库，alter database open

处理日志文件丢失

丢失非当前日志文件

shutdown immediate

删除日志文件

startup mount

清除日志文件内容，alter database clear logfile group 4;

打开数据库，alter database open;

丢失当前日志文件

recover database until cancel;

alter database open resetlogs;

管理表空间和数据文件

2019年4月2日 8:35

数据库存储的结构层次

表空间——数据文件

段——存储结构

区——Oracle中最小的空间分配单位

块——Oracle中最小的IO单位

创建USERS表空间并设置为数据库默认表空间

create tablespace users datafile'...' size 20m; '...'来自select name from v$datafile;

alter database default tablespace users;

创建由2k块组成的表空间

create tablespace users datafile'...' size 20m blocksize 2k;

可能报错内存中没有能容纳2k块的空间：alter system set db_2k_cache_size = 16m;

表空间管理（区的管理）

本地管理

数据字典管理

desc dba_tablespaces

create tablespace userdata datafile'...' size 10m extent management dictionary;

表空间类型

常规表空间

撤销表孔甲（回滚数据）

临时表空间（排序）

查看表空间信息：

select tablespace_name, contents from dba_tablespaces;

show parameter undo_tablespace ;

撤销表空间undotbs2并设为数据库默认撤销表空间

create undo tablespace undotbs2 datafile'...' size 10m;

alter system set undo_tablespace=undotbs2;

创建临时表空间temp2，并设为数据库默认临时表空间

create temporary tablespace temp2 tempfile'...' size 10m;

alter database default temporary tablespace temp2;

表空间状态

联机可读写

只读

脱机

查看表空间状态：select tablespace_name, status from dba_tablespaces;

改某个表空间为只读：alter tablespace smalltbs read only;

不能脱机的表空间：

system

当前的撤销表空间

临时表空间

alter tablespace ... online;

删除表空间

drop tablespace smalltbs;

OMF（Oracle管理文件）

扩展表空间

修改原数据文件大小

自动扩展

ALTER DATABASE DATAFILE 'D:\ORACLE\PRODUCT\10.2.0\ORADATA\EDWTEST\APP03.DBF' AUTOEXTEND ON NEXT 5M MAXSIZE 100M;

手动扩展

ALTER DATABASE DATAFILE 'D:\ORACLE\PRODUCT\10.2.0\ORADATA\EDWTEST\APP02.DBF' RESIZE 100M;

增加新的数据文件

ALTER TABLESPACE app_data ADD DATAFILE 'D:\ORACLE\PRODUCT\10.2.0\ORADATA\EDWTEST\APP03.DBF' SIZE 50M;

数据文件的移动和重命名

alter database move datafile '...' to 'D:\...\01.DBF';

在表空间创建新表

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

Oracle安全

2019年4月2日 10:39

验证

sys用户：

操作系统验证

sqlpus / as sysdba，不安全

口令文件验证

创建一个口令验证文件，[OracleHome]/database/PWDdb18c.ora

初始化参数remote_login_passwordfile =

1)NONE，禁止使用口令验证文件

2)EXCLUSIVE启用，单例程多用户，v$pwfile_users，grant sysdba to scott：赋予scott以sysdba身份连接的权利

3)SHARED，多例程单用户，只能用sys连接non-sys用户：数据库不打开时non-sys用户无法连接

non-sys用户：

数据库验证

create user nit identified by admin;

操作系统验证

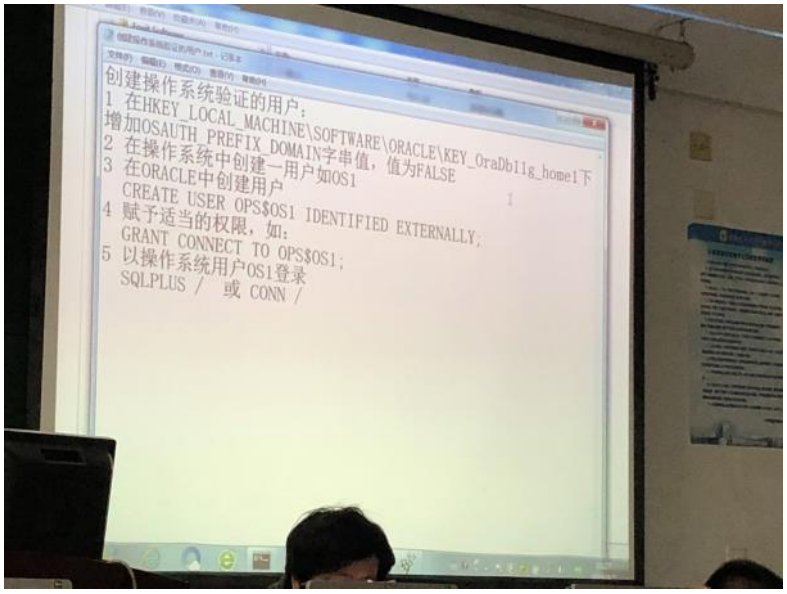
初始化参数os_authent_prefix（默认是ops$）

创建操作系统用户（os1）

在数据库中创建对应的用户：create user ops$os1 identified externally

赋予权限：grant create session to ops$os1

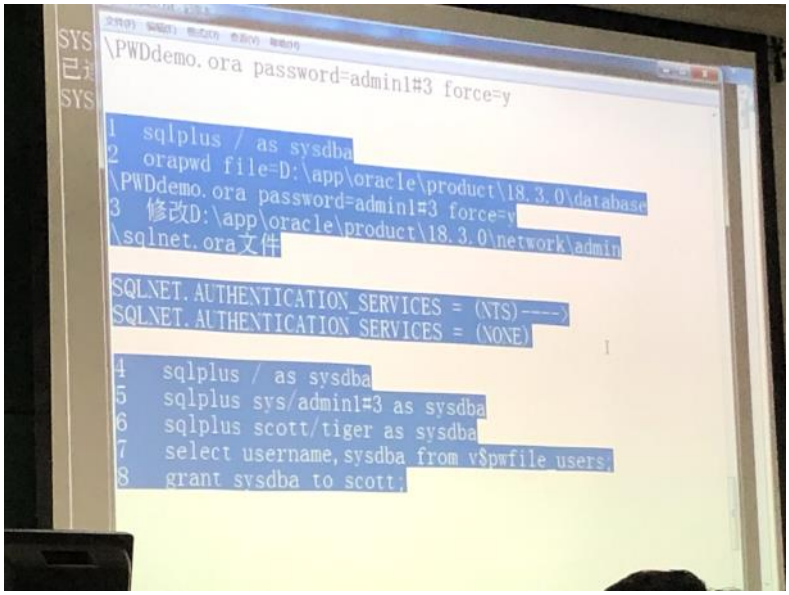
修改注册表



（11g改为18c）

以操作系统用户os1登录：runas /user:os1 cmd

sqlplus /



授权

grant select on scott.emp to hr;

grant create table to hr; create table 是系统权限，允许其它用户建表要先允许延迟创建（？）

revoke：取消权限

drop：删除角色 drop role app_r1

系统权限

对象权限

权限的传递规则

对象权限：xxx with grant option，对象权限是连带的，对象权限如select等

系统权限：xxx with admin option，系统权限不连带，系统权限如create table等

角色（role）：权限的集合

简化权限的管理

动态权限管理

用户自定义角色：

```
create role r1 identified by r1;
grant r1 to a; 将r1角色赋给a用户，a用户登录后激活
alter user a default role none;
```

预定义角色

应用程序角色

```
create role app_r1 identified using scott.p1; scott.p1是一个具体的应用程序
grant select on scott.emp to app_r1;
create or replace procedure scott.p1as authid current_user as
begin
    dbms_session.set_role('APP_R1');
end;
/
在某用户控制台下
execute scott.p1;
之后的一条命令中，此用户可获得app_r1的权限，之后不再拥有此权限
```

默认角色在登录时激活，非默认角色用命令激活：set role r1，角色如果带口令，set role r2 identified by r2;

审核

默认审核（强制审核）

标准数据库审核（类似于监视）

启用后审核（初始化参数）：audit_trail

指定审核选项

审核用户（权限审核）：audit select any table by scott;

审核对象：audit delete on scott.emp;

语句审核：audit create trigger;

基于值的审核（用触发器实现）

DML触发器

```
create or replace trigger scott.tr1 after update on scott.emp for each row
begin
    insert into scott.tr_test1 valuse('...')
end
/
```

创建新表

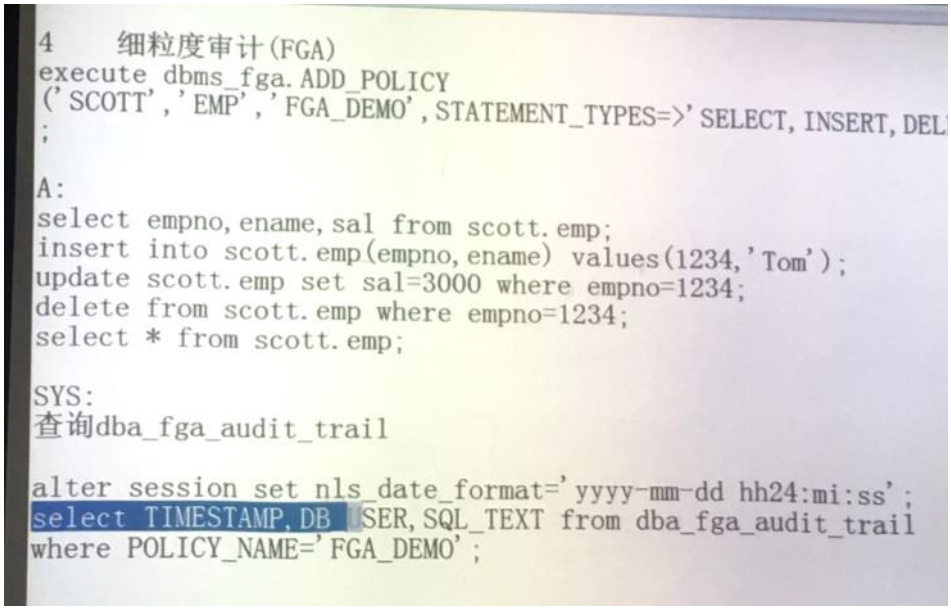
```
create table sal_change(ename varchar(20), old_sal number(7,2), new_sal number(7,2));
```

创建触发器记录员工工资的变化情况

```
create or replace trigger scott.trl
after update on scott.emp REFERENCING OLD AS o NEW AS n for each row
begin
    insert into sal_change values(:o.ename, :o.sal, :n.sal);
end;
/
```

系统触发器

```
create table scott.logon_rec(username varchar(30), logon_time date)
create or replace trigger
```



导入导出

2019年4月16日 8:37

交互式
命令模式
获取帮助：exp help=y

导出
在命令行下，exp，输入用户和口令、缓冲区大小、文件名等信息即可导出
exp scott/tiger file=emp1.dmp tables=emp

导入
imp scott/tiger file=emp1.dmp tables=emp
imp hr/hr file=sal2500.dmp fromuser=scott touser=hr tables=emp constraints=n

可以导出的对象
整个数据库（full=y）
表空间（tablespaces）
方案（对象的集合）（owner=scott），多个用户可以用逗号隔开，导出scott用户下所有对象
表（tables=dept），要导出的表

导出表的子集
exp scott/tiger file=sal2500.dmp tables=emp query='where "sal>2500"'
双引号用于屏蔽大于号，在命令行下输入时大于号有输出的意思

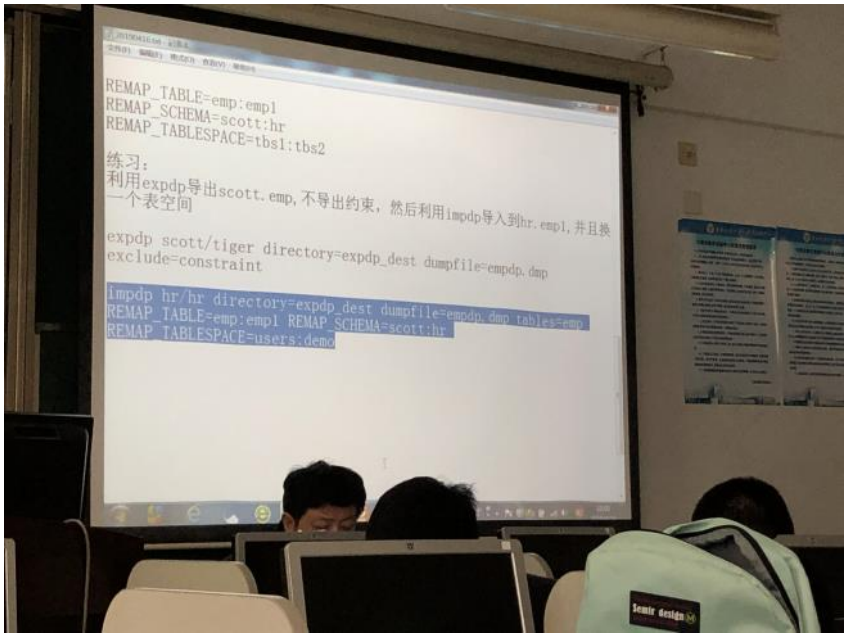
使用参数文件
userid=scott/tiger file=p1/dmp tables=emp query where sal>2500，保存为p1.par
exp parfile=p1.par
在par文件中写入导出命令，不需要屏蔽大于号，最后加parfile=xxx

数据泵
expdp/impdp
用并行提高速度，大任务&有足够的空闲资源

expdp scott/tiger file=empdp.dmp tables=emp
创建并使用新目录
create directory expdp_dest as 'd:\exp\'
grant read, write on directory expdp_dest to scott
expdp scott/tiger directory=expdp_dest dumpfile=empdp.dmp tables=emp
impdp scott/tiger directory=expdp_dest dumpfile=empdp.dmp tables=emp

不导出约束：exclude = constraints
remap_table=emp:emp1，表重命名
remap_schema=scott:hr，将hr导入scott中？切换视图
remap_tablespace=tbs1:tbs2，将表空间1替换成表空间2

练习：
利用expdp导出scott.emp，不导出约束，然后利用impdp导入到hr.emp1，并且换一个表空间
expdp scott/tiger directory =expdp_dest dumpfile=empdp.dmp exclude=constraint
impdp hr/hr directory =expdp_dest dumpfile=empdp.dmp tables=emp REMAP_TABLE=emp:emp1 REMAP_SCHEMA=scott:hr
REMAP_TABLESPACE=tbs1:tbs2



需要将目录expdp_dest对hr进行授权

备份和恢复

使用logminer查询重做日志文件

启用数据库补充日志

select SUPPLEMENTAL_LOG_DATA_MIN from v$database;

alter database add SUPPLEMENTAL LOG DATA;

创建数据字典文件

create directory Mydict as 'd:\dict';

生成数据字典文件

EXECUTE dbms_logmnr_d.build('datadir.ora', 'dict');

开始一个事务

scott:

update emp set sal=1800 where empno=7360;

commit;

添加需要分析的日志文件

切换回sys用户

D:\...\ORADATA\DB18C\REDO03.LOG

EXECUTE dbms_logmnr.add_logfile(LogFileName=>'D:\...\REDO03.LOG', options=>dbms_logmnr.new);

执行分析

EXECUTE dbms_logmnr.start_logmnr(DictFileName=>'d:\...\datadir.ora');

查询结果

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';

RMAN（恢复管理器）

冷备份（MOUNT）

热备份（打开数据库的备份）

advise failure all;

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!