石 家 庄 铁 道 大 学

**实 验 报 告**

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称 | 大型数据库技术应用 | 班级 | 信1901-4 |
| 姓名 | 闫竞存 | 学号 | 20194127 |
| 实验日期 | 2020.11.10 | 评分 |  |

目录

[实验2: Oracle数据库基本操作 2](#_Toc57067776)

[一、 实验目的： 2](#_Toc57067777)

[二、 实验内容： 2](#_Toc57067778)

[三、 心得体会： 12](#_Toc57067779)

[实验3: 数据库编程 13](#_Toc57067780)

[一、 实验目的： 13](#_Toc57067781)

[二、 实验内容： 13](#_Toc57067782)

[三、 心得体会： 26](#_Toc57067783)

[实验4: Oracle数据库控制 27](#_Toc57067784)

[一、 实验目的： 27](#_Toc57067785)

[二、 实验内容： 27](#_Toc57067786)

[三、 心得体会： 32](#_Toc57067787)

[实验5: Oracle安全管理、备份与恢复 32](#_Toc57067788)

[一、 实验目的： 32](#_Toc57067789)

[二、 实验内容： 32](#_Toc57067790)

[三、 心得体会： 40](#_Toc57067791)

实验2: Oracle数据库基本操作

### 实验目的：

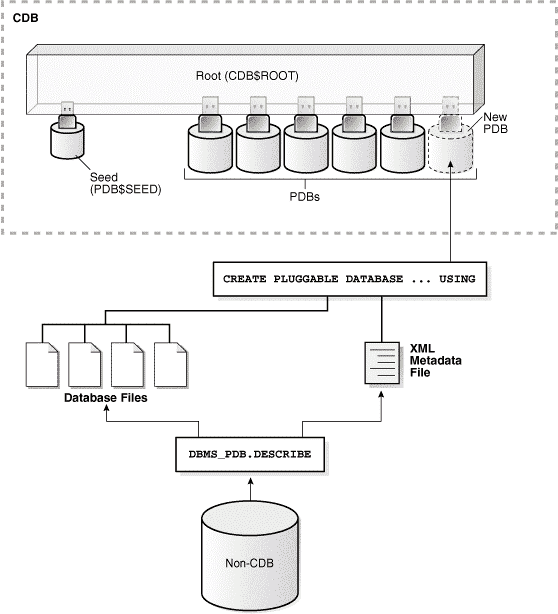
1. 了解oracle安装后的安装主目录下的文件体系及其主要用途。
2. 了解CDB和PDB概念，掌握CDB和PDB相关操作语句。
3. 掌握使用SQL Plus和SQL Developer操作数据库及执行SQL语句的方法。

### 实验内容：

*注：蓝色斜体部分表示由自己环境中的实际情况确定，或自行指定。*

1. 了解CDB和PDB概念

Oracle 12c引入了容器数据库CDB（Container DataBase）和可插拔数据库PDB（Pluggable DataBase）。



* 1. 在CDB中可以有多个PDB，包含一个root根（CDB$ROOT）容器、一个种子（PDB$SEED）容器和多个PDB容器。
  2. CDB$ROOT：根容器用来做所有容器的根，对每个PDB容器进行统一管理，sqlplus / as sysdba默认连接根容器，需要切换到其他的PDB容器才可以对单独的PDB容器操作。
  3. PDB$SEED：种子容器作为插入PDB容器的模板而存在，每个CDB容器都有一个种子容器，且不可对其中对象进行修改。
  4. 用户：CDB$ROOT中的普通用户可以通过权限分配来访问一个或多个指定的PDB容器，最大权限用户是sysdba。其中PDB容器也可单独创建普通用户来管理该数据库。

1. 掌握CDB和PDB相关语句

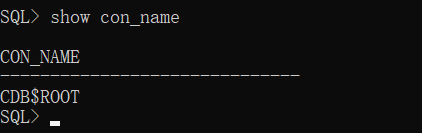
Windows 命令行窗口（cmd）以DBA权限连接数据库：sqlplus / as sysdba

* 1. 查看当前容器

select sys\_context('USERENV','CON\_NAME') from dual;



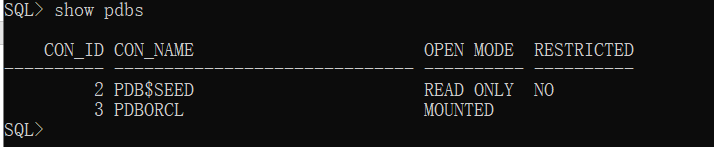
show con\_name



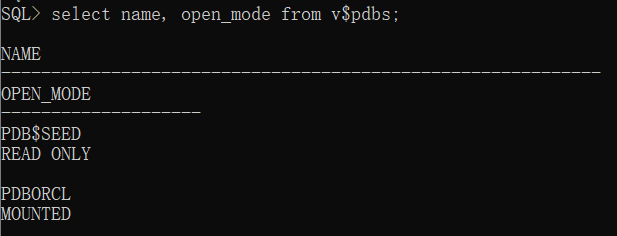
提示：Oracle启动时，默认连接CDB容器。

* 1. 查看PDB（CDB模式下）

show pdbs --查看所有pdb



select name, open\_mode from v$pdbs; --v$pdbs为PDB信息视图



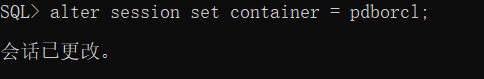
select con\_id, dbid, guid, name, open\_mode from v$pdbs;



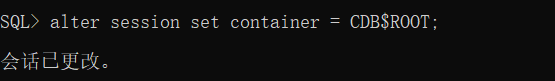
提示：复习 open\_mode 中NOMOUNT、MOUNT、OPEN三种状态的含义。CDB启动时，PDB是自动启动到MOUNT状态，而不是OPEN，所以我们还需要手工打开。当然，也可以通过在CDB中配置触发器来自动打开PDB。

* 1. 切换和打开/关闭容器

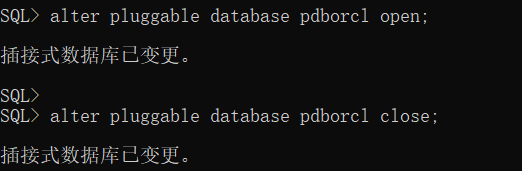
alter session set container=*pdborcl*; --切换到*pdborcl*



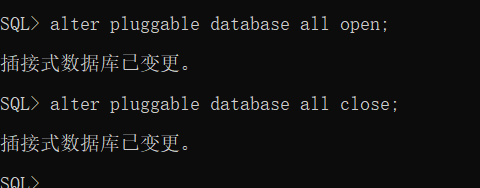
alter session set container=*CDB$ROOT*; --切换到CDB容器



alter pluggable database *pdborcl* open/close; --打开/关闭*pdborcl*



alter pluggable database all open/close; --打开/关闭所有PDB



提示：alter session set container仅切换当前容器，不改变容器的open\_mode。

alter pluggable database *xxx* open/close，仅打开或关闭容器，不改变当前容器。

如果要打开并切换，需要同时执行上述命令，或者在切换后执行startup：

alter session set container=*pdborcl*;

startup

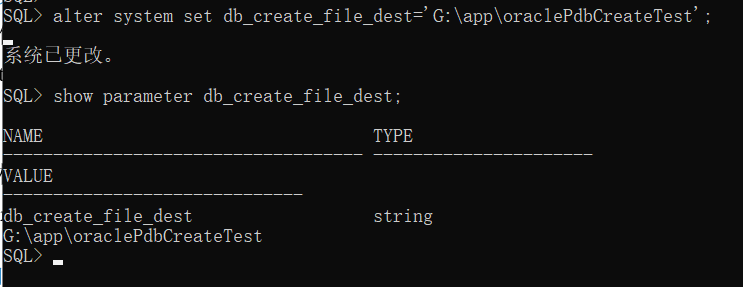
关闭当前容器可使用shutdown immediate。

关闭CDB时，其所包含的PDB也会关闭，尝试自行验证。

* 1. 创建或克隆前要指定文件映射的位置（需要CBD下sysdba权限）

alter system set db\_create\_file\_dest='*C:\app\oracle12c\oradata\orcl\pdbtest1*'; --文件夹需存在

show parameter db\_create\_file\_dest

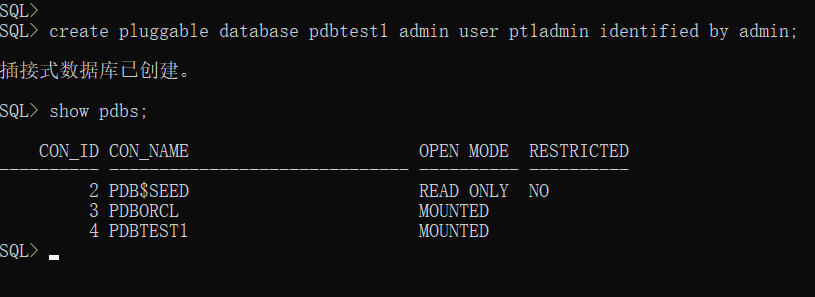


* 1. 创建PDB（需要CBD下sysdba权限）

create pluggable database *pdbtest1* admin user *pt1admin* identified by admin;

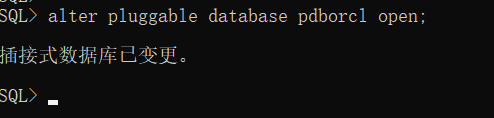
alter pluggable database *pdbtest1* open; --将*pdbtest1*打开

提示：还可通过DBCA进行向导式创建，标准版不支持创建多个PDB。



* 1. 克隆PDB（需要CBD下sysdba权限）

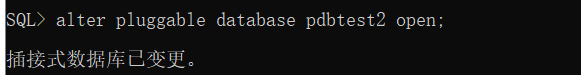
alter pluggable database *pdborcl* open; --*orcl*必须打开才可以被克隆



create pluggable database *pdbtest2* from *pdborcl*;



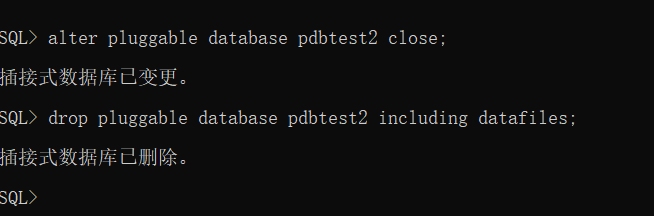
alter pluggable database *pdbtest2* open; --将*pdbtest2*打开



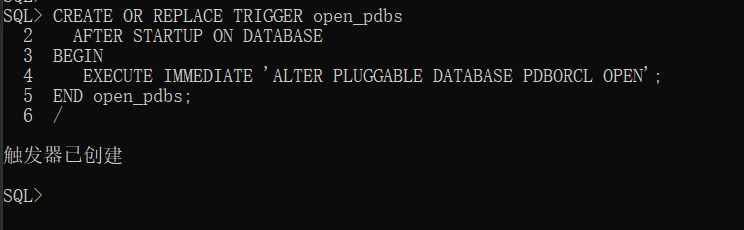
* 1. 删除PDB（需要CBD下sysdba权限）

alter pluggable database *pdbtest1* close; --关闭之后才能删除

drop pluggable database *pdbtest1* including datafiles; --删除*pdbtest1*



* 1. **创建一个触发器让PDB能够在CDB启动时打开，并验证效果。（课堂思考题，自行查询相关资料完成）**



具体代码：

CREATE OR replace trigger open\_pdbs -- 建立触发器

AFTER STARTUP ON DATABASE -- 在启动数据库之后

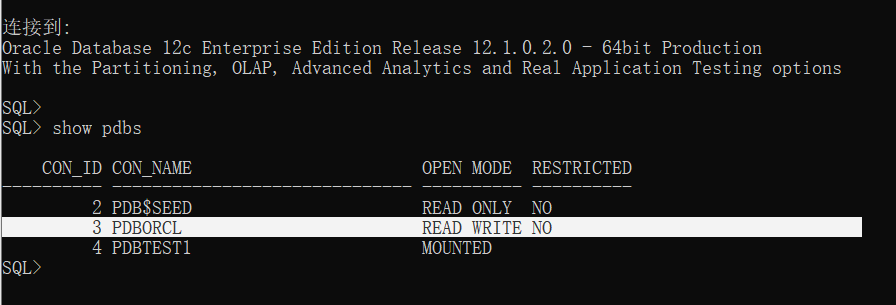
BEGIN

EXECUTE IMMEDIATE 'ALTER PLUGGABLE DATABASE PDBORCL OPEN'; --立即执行这句话

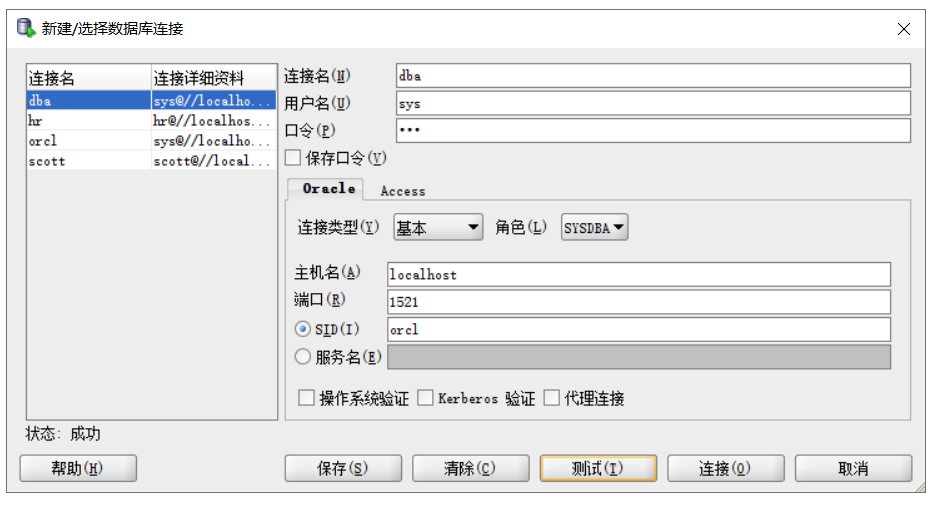
END open\_pdbs;

**创建了一个触发器，使CBD打开的时候自动打开PDBORCL。**

**下图是测试效果**



1. 使用 SQL Developer启用 SCOTT 和HR 账户，熟悉 SCOTT 模式和 HR 模式下包含的数据表及表内字段含义。
   1. 使用 SQL Developer，以 sys 用户和 sysdba 权限连接安装 Oracle 时默认安装的数据 *ORCL*。



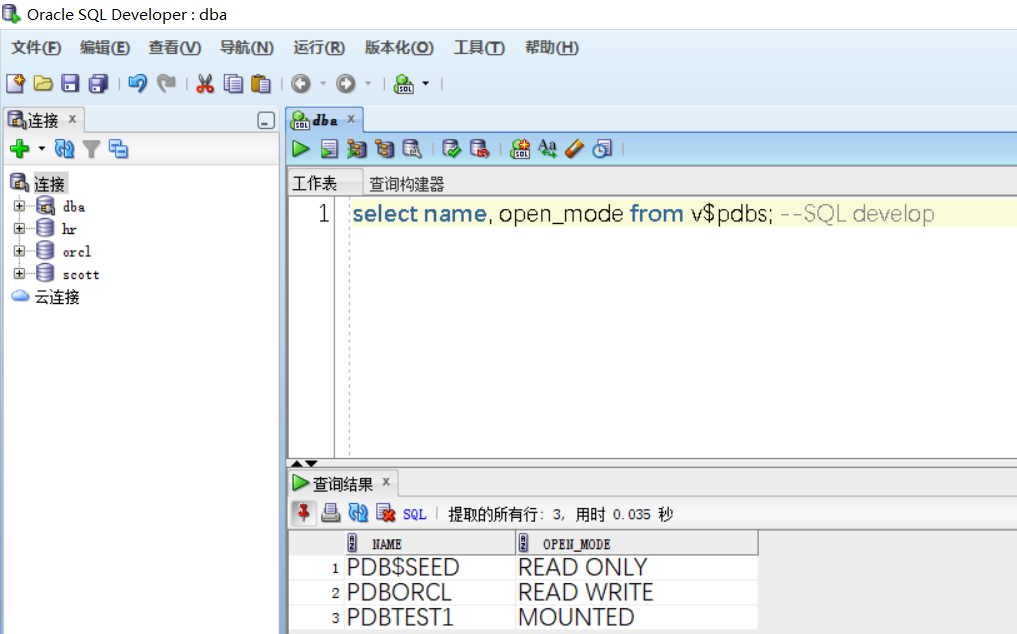
由截图可见测试连接成功

* 1. 在 *ORCL* 的 SQL 编辑界面，输入命令查看 PDB 的状态。

select name, open\_mode from v$pdbs; --SQL develop 不支持 show pdbs 语句



如图所示，语句执行成功

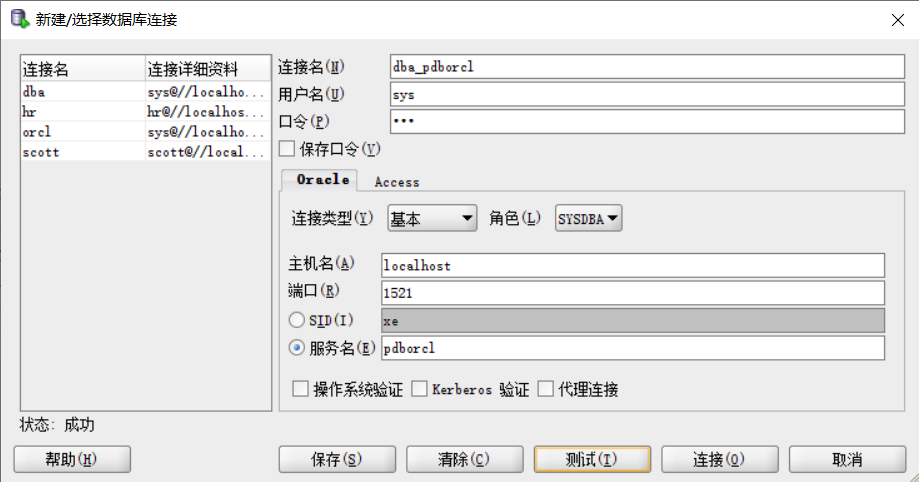


* 1. 如果默认的 PDB（PDBORCL）不为 OPEN 状态，则将其打开。

alter pluggable database *pdborcl* open;

* **由于我们之前已经创建了触发器，使CBD打开的时候自动打开PDBORCL。**
* **所以在查询得到的表中PDBORCL为打开状态。**
* **所以这一句就不用再执行了。**
  1. 使用 SQL Developer，以 sys 用户和 sysdba 权限连接默认的 PDB

（*PDBORCL*）。



如图所示，测试连接成功

* 1. 在 PDBORCL 的 SQL 编辑界面，查看 SCOTT和HR用户的状态。

select username, account\_status from dba\_users

where username=’SCOTT’ and username=’HR’;

这两句SQL语句有错误，查找状态的连接词应该是 or，且引用用户名的引号应该是英文引号

应该改为：

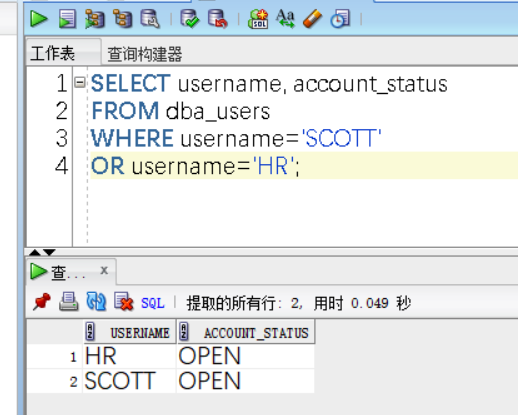
SELECT username, account\_status

FROM dba\_users

WHERE username='SCOTT'

OR username='HR';

改正之后的查询结果为：

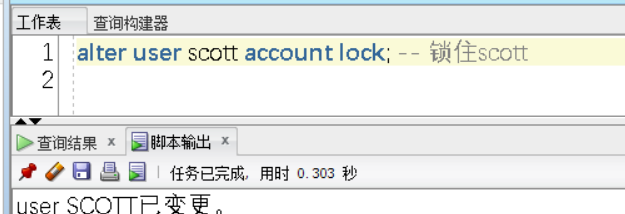


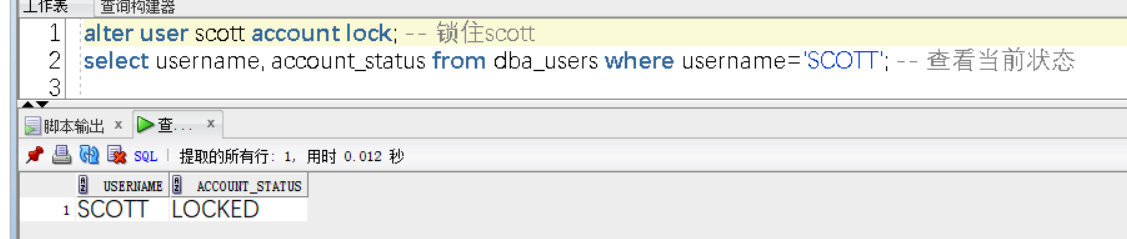
* 1. 如果 SCOTT 和 HR 用户为锁定和密码过期状态（LOCKED &EXPIRED），在 *PDBORCL* 的 SQL 编辑界面，将两个账户解锁并设定密码。

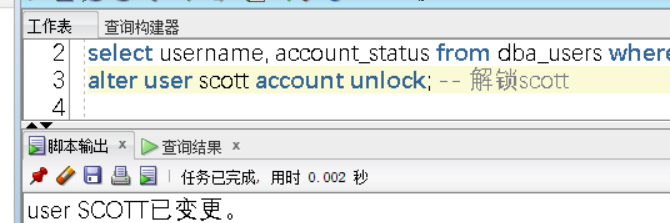
alter user scott(hr) account unlock;

alter user scott(hr) identified by *newpassword*;

**由于我的scott和hr账户在写实验报告之前就已经解锁了两个账户，故在这一步简要实验上述语句的作用：**

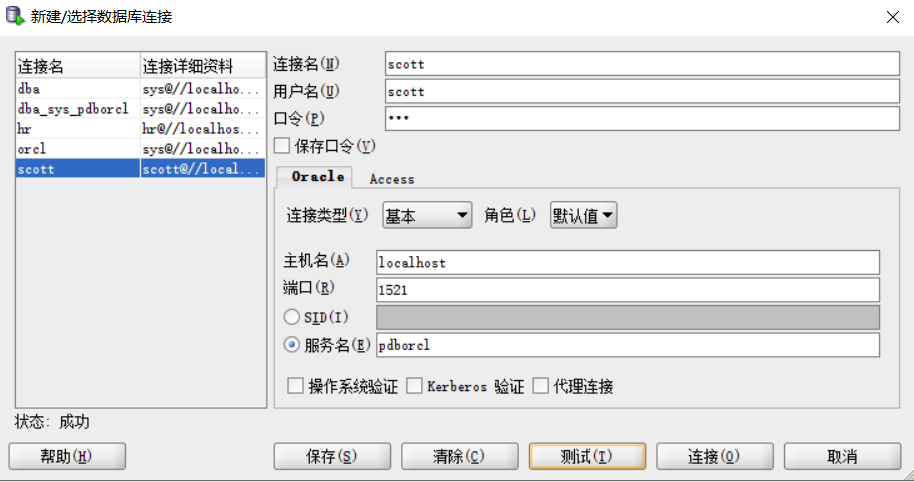




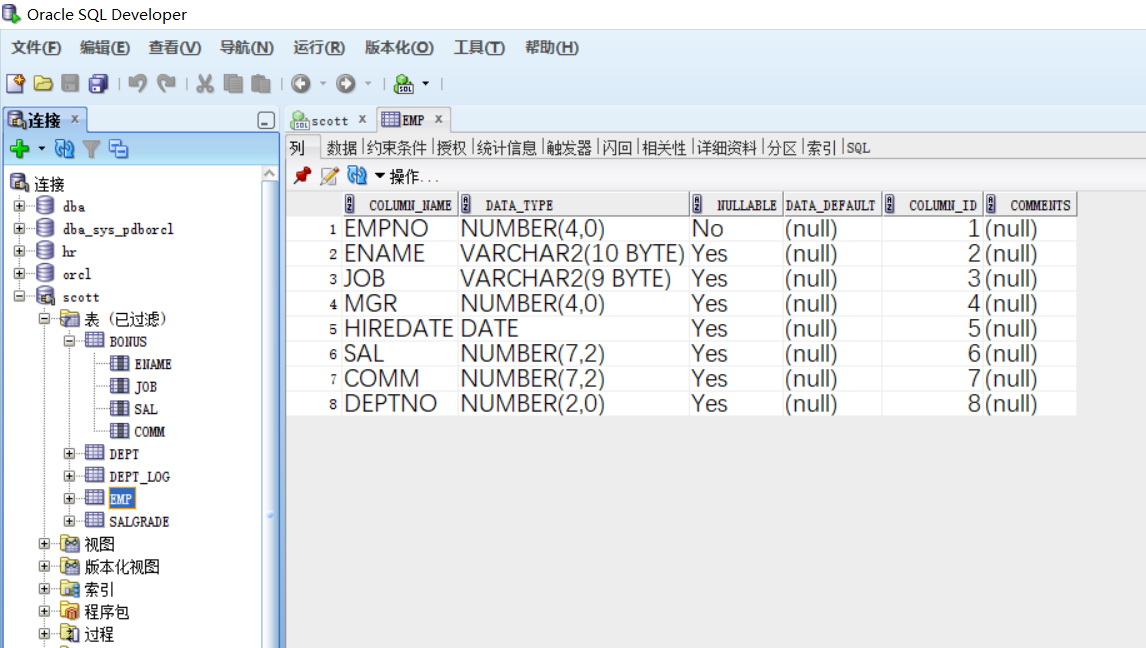




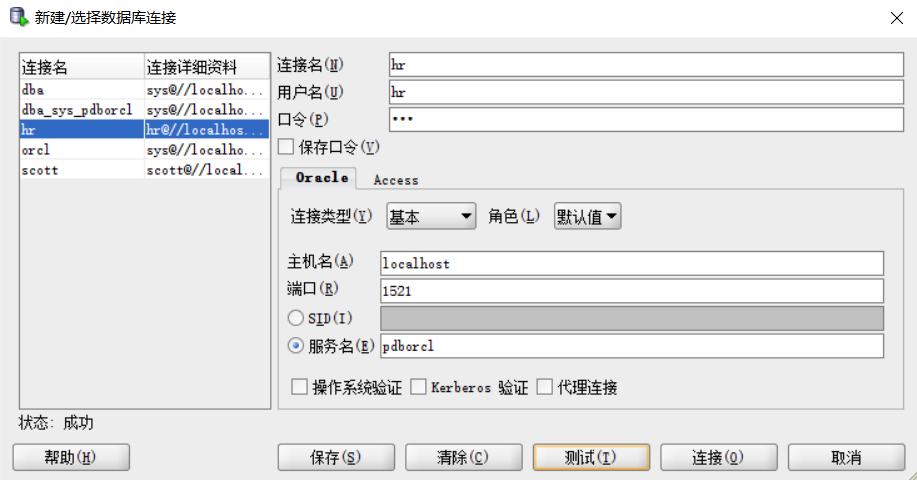
* 1. 使用 SQL Developer，新建两个连接，分别以 SCOTT和HR用户连接默认的 PDB（*PDBORCL*），查看SCOTT模式下和HR模式下的数据表，了解每个表的字段含义和数据类型，大致查看数据内容。
* 以SCOTT用户连接数据库，如图所示：



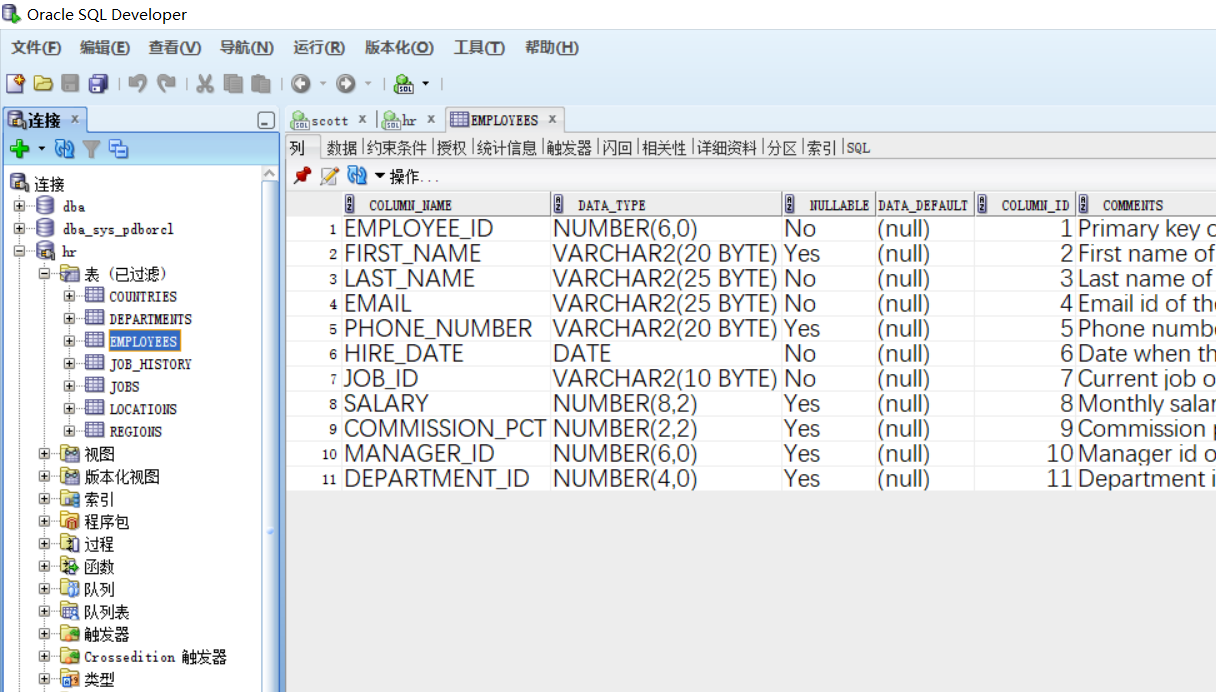
测试连接成功



* 以HR用户连接数据库，如图所示：



测试连接成功



### 心得体会：

本次实验内容为Oracle数据库的基本操作，具体步骤都在实验指导书上明确写明，并且老师在课上也讲得很明白了，所以此次实验进行比较顺利，并没有出现问题。

在本次实验中，了解了oracle安装后的目录，进行了CDB和PDB的相关操作，并且使用了sql plus和sql developer进行了同样的操作，体会到了图形工具的方便。

实验3: 数据库编程

### 实验目的：

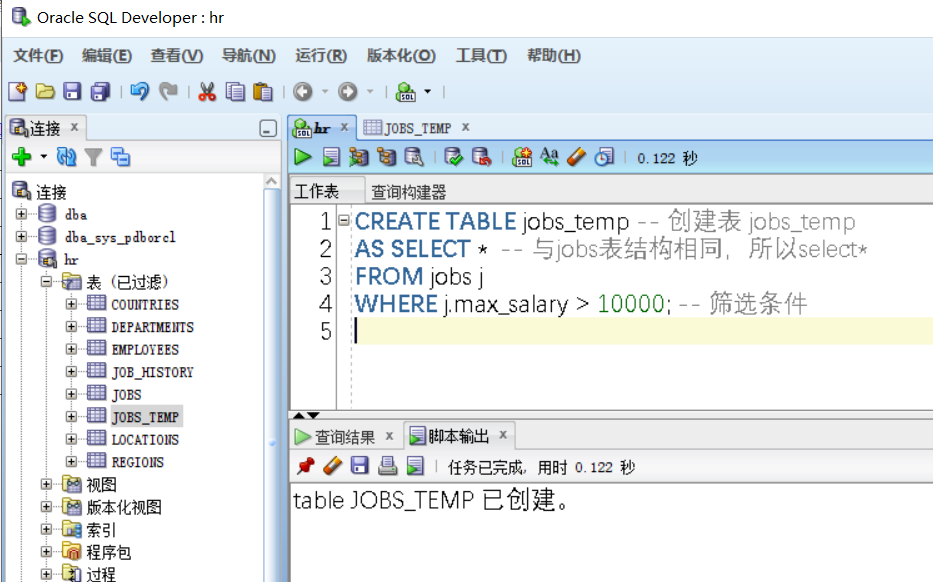
1. 掌握使用 SQL 进行数据定义、数据查询、数据操纵、数据控制的方法。
2. 掌握PL/SQL 程序块的结构和语法。
3. 了解 PL/SQL 数据类型，掌握结构控制语句的使用方法。
4. 掌握 PL/SQL 游标、存储过程、函数、触发器的编写方法。

### 实验内容：

*注：蓝色斜体部分表示由自己环境中的实际情况确定，或自行指定。*

**（一）利用SQL语言进行DDL、DML、DCL、DQL操作**

1. 在HR 模式下，用使用 SQL 语句完成下列操作：
2. 创建一个与 jobs 表结构相同的表 jobs\_temp，将 jobs 表中最高工资（max\_salary）大于 10000 的记录插入新表 jobs\_temp 中。（提示：使用批量插入）



代码:

CREATE TABLE jobs\_temp -- 创建表 jobs\_temp

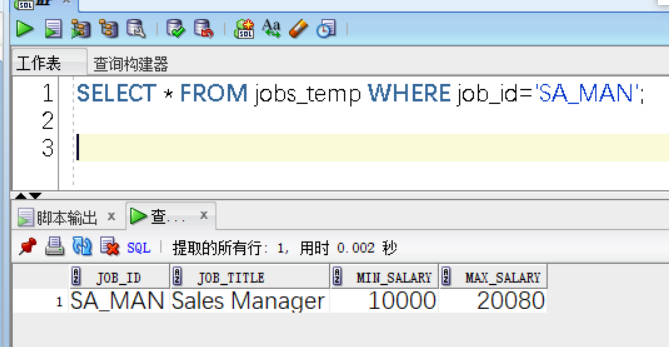
AS SELECT \* -- 与jobs表结构相同，所以select\*

FROM jobs j

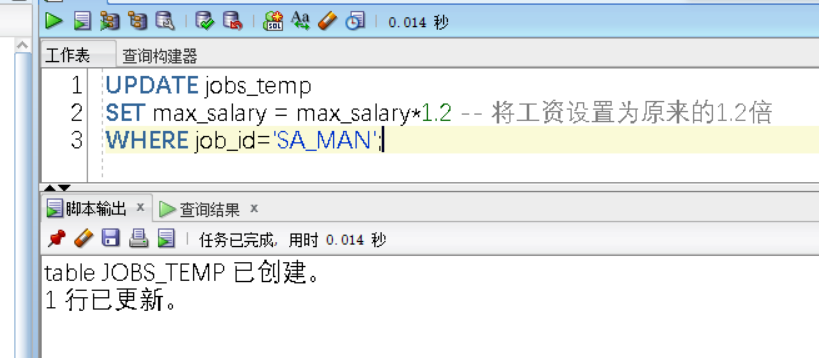
WHERE j.max\_salary > 10000; -- 筛选条件

1. 将表 jobs\_temp 中销售经理（SA\_MAN）的最高工资上调 20%。

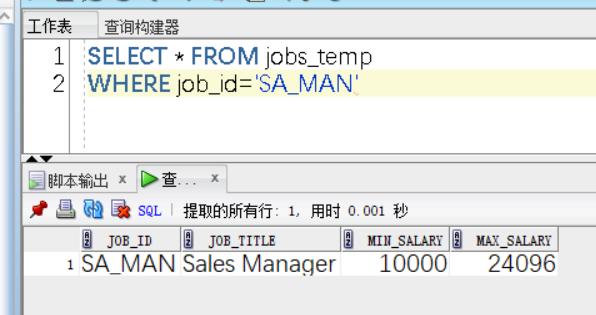
**工资调整之前：**



**执行结果：**

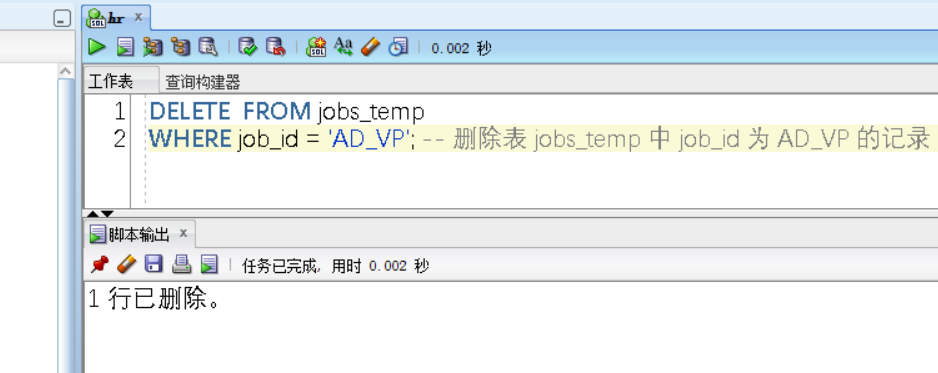


**工资调整之后：**



1. 删除表 jobs\_temp 中 job\_id 为 AD\_VP 的记录。

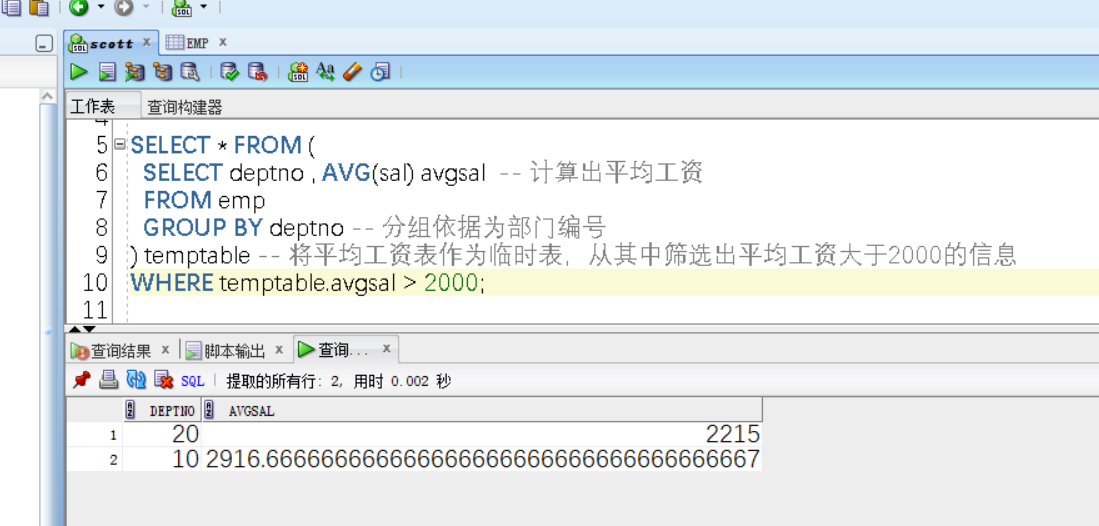
**执行结果：**



1. 删除表 jobs\_temp。



1. 在 SCOTT 模式，下用使用 SQL 语句完成下列操作
2. 在 emp 表中，通过分组的方式计算出每个部门的平均工资，并筛选出平均工资大于 2000 的记录信息。



代码：

SELECT \* FROM (

SELECT deptno , AVG(sal) avgsal -- 计算出平均工资

FROM emp

GROUP BY deptno -- 分组依据为部门编号

) temptable -- 将平均工资表作为临时表，从其中筛选出平均工资大于2000的信息

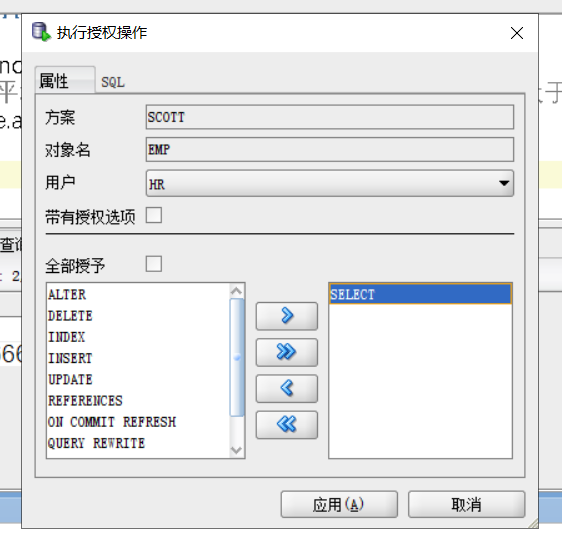
WHERE temptable.avgsal > 2000;

1. 把EMP表的查询权限授给用户HR，验证授权成功，之后收回权限。

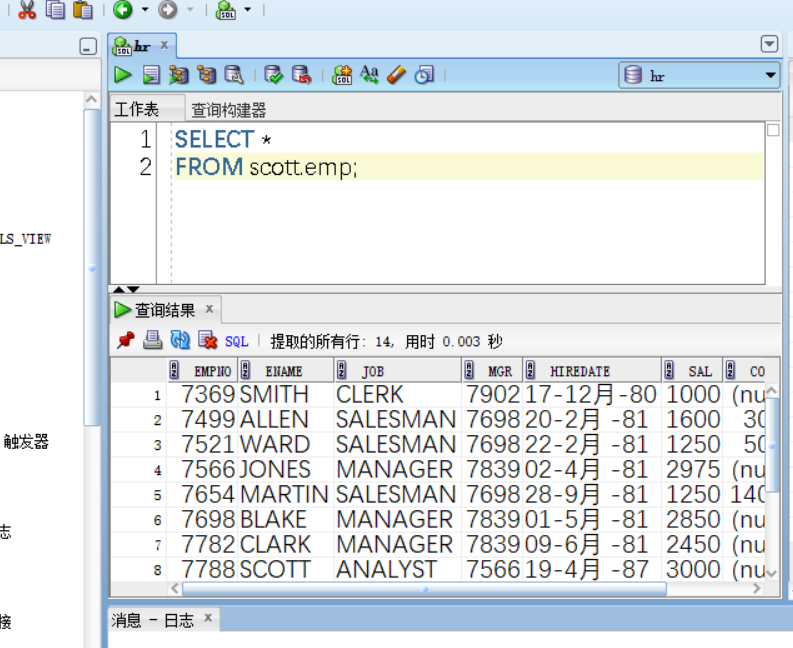
授权的两种方式：

1. SQL语句： grant SELECT on "SCOTT"."EMP" to "HR" ;

2. SQL developer 界面操作



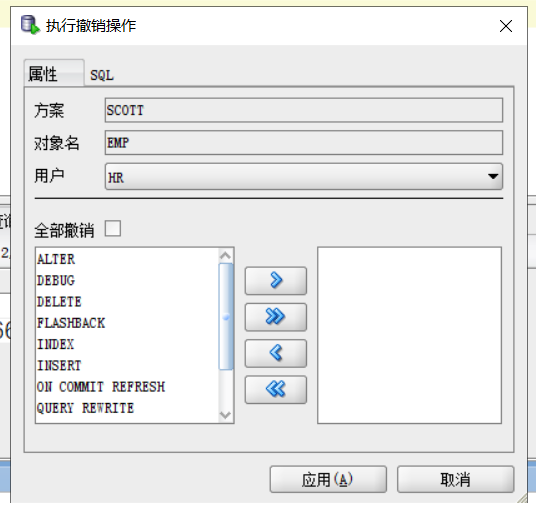
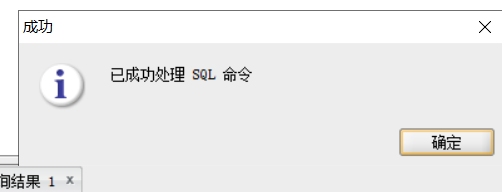
在HR用户中查询scott表中的数据，可以查询到，表示授权成功



收回权限:

1. SQL语句撤销权限 revoke SELECT on "SCOTT"."EMP" from "HR"

2. SQL Developer 图形界面撤销权限：

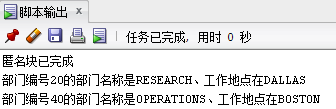
 

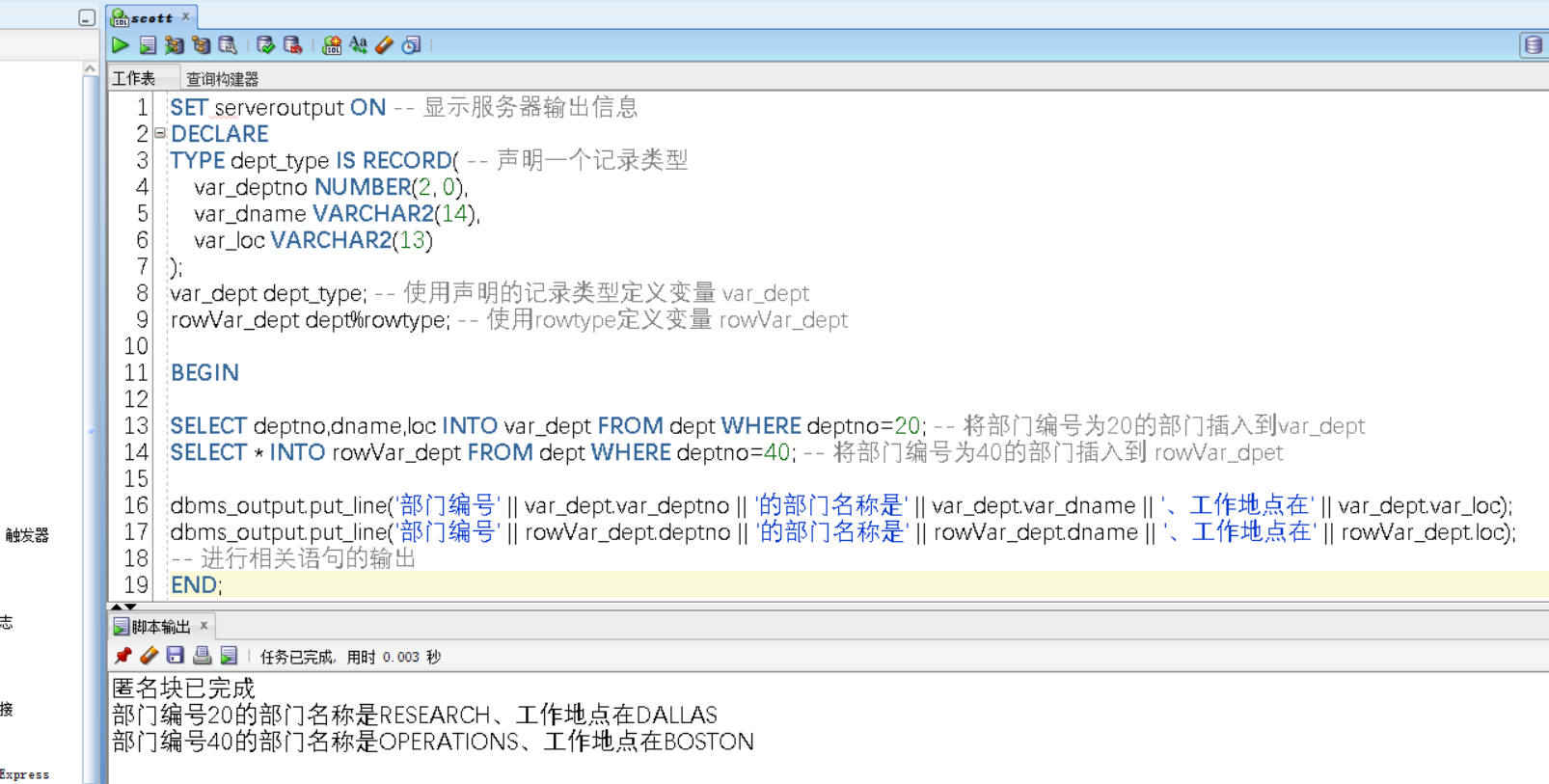
权限撤销成功

**（二）PL/SQL编程**

1. 在SCOTT 模式中，声明型一个记录类型 dept\_type， 使用该类型的变量var\_dept 存储 dept 录表中的一条记录（DEPTNO=20）并按下图所示输出，声明一个%ROWTYPE 类型的变量 rowVar\_dept，使用该储变量存储 dept表中的一条记录（DEPTNO=40）并按下图所示输出。

结果：





**代码：**

SET serveroutput ON -- 显示服务器输出信息

DECLARE

TYPE dept\_type IS RECORD( -- 声明一个记录类型

var\_deptno NUMBER(2, 0),

var\_dname VARCHAR2(14),

var\_loc VARCHAR2(13)

);

var\_dept dept\_type; -- 使用声明的记录类型定义变量 var\_dept

rowVar\_dept dept%rowtype; -- 使用rowtype定义变量 rowVar\_dept

BEGIN

SELECT deptno,dname,loc INTO var\_dept FROM dept WHERE deptno=20; -- 将部门编号为20的部门插入到var\_dept

SELECT \* INTO rowVar\_dept FROM dept WHERE deptno=40; -- 将部门编号为40的部门插入到 rowVar\_dpet

dbms\_output.put\_line('部门编号' || var\_dept.var\_deptno || '的部门名称是' || var\_dept.var\_dname || '、工作地点在' || var\_dept.var\_loc);

dbms\_output.put\_line('部门编号' || rowVar\_dept.deptno || '的部门名称是' || rowVar\_dept.dname || '、工作地点在' || rowVar\_dept.loc);

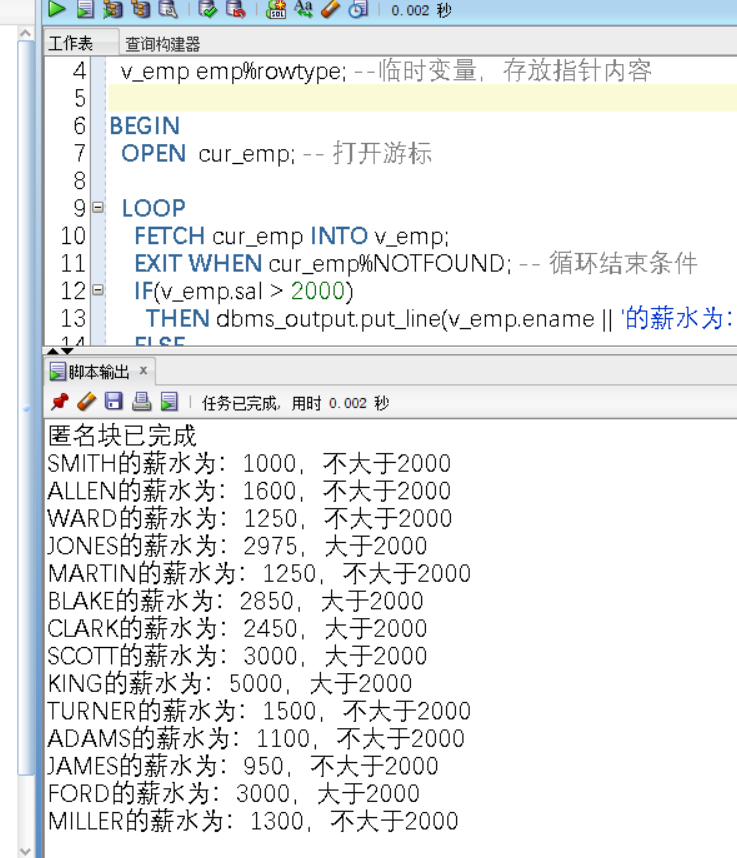
-- 进行相关语句的输出

END;

1. 在SCOTT 模式中，用使用IF 、LOOP 等断语句和游标，判断 emp 表中每位雇员的薪水是否大于 2000 ，并按下图所示输出。

结果：





完整代码为：

SET serveroutput ON -- 显示服务器输出信息

DECLARE

CURSOR cur\_emp IS SELECT \* FROM emp; -- 声明游标

v\_emp emp%rowtype; --临时变量，存放指针内容

BEGIN

OPEN cur\_emp; -- 打开游标

LOOP

FETCH cur\_emp INTO v\_emp;

EXIT WHEN cur\_emp%NOTFOUND; -- 循环结束条件

IF(v\_emp.sal > 2000) -- 判断

THEN dbms\_output.put\_line(v\_emp.ename || '的薪水为：' || v\_emp.sal || '，大于2000');

ELSE

dbms\_output.put\_line(v\_emp.ename || '的薪水为：' || v\_emp.sal || '，不大于2000');

END IF;

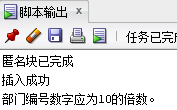
END LOOP;

CLOSE cur\_emp; -- 关闭游标

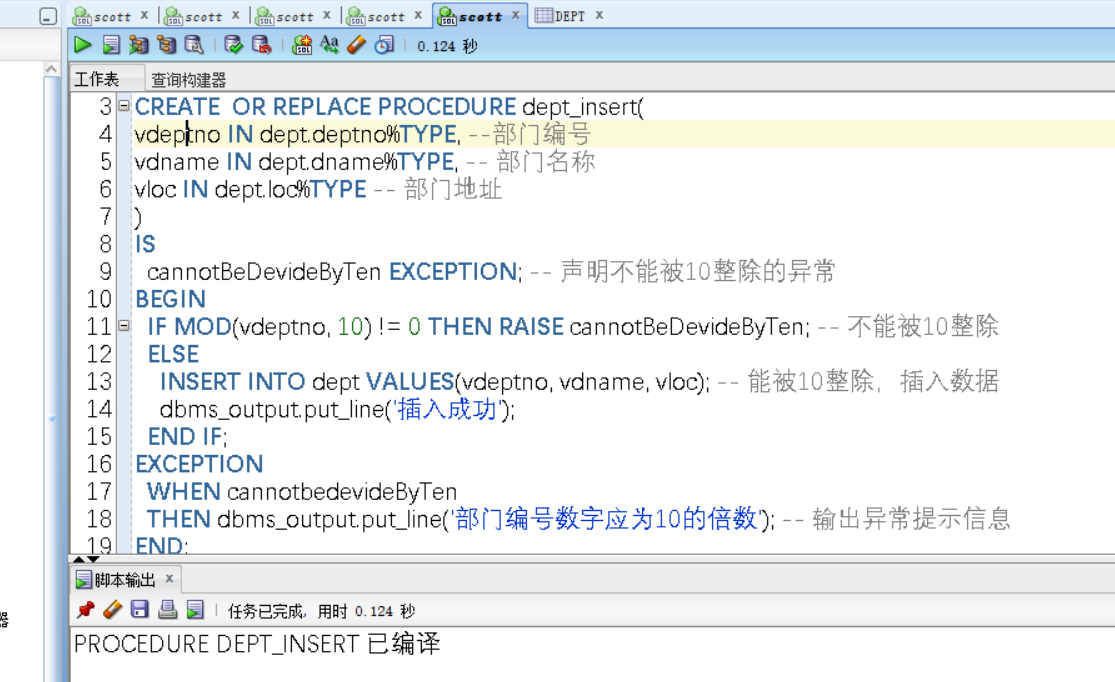
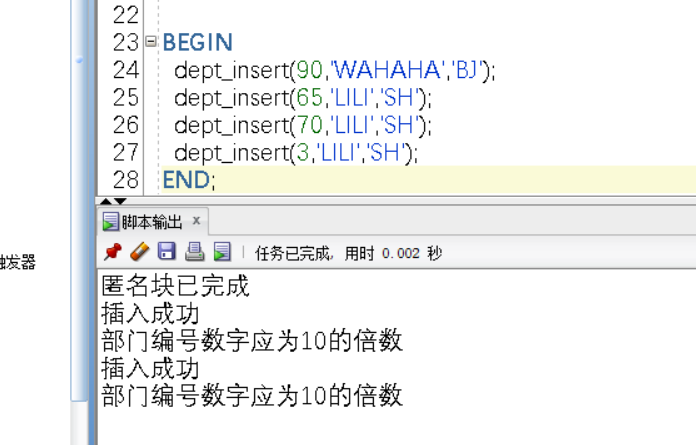
END;

1. 在SCOTT 模式中， 创建一个带参数的存储过程 dept\_insert 实现向dept表中插入数据， 插入时deptno 不为 10 的整数倍视为异常，给出提示。

通过调用该存储过程插入，验证其有效性。

提示：dept\_insert 原型如下 CREATE PROCEDURE dept\_insert(vdeptno INdept.deptno%TYPE,vdname IN dept.dname%TYPE,vloc IN dept.loc%TYPE)

**存储过程创建情况如下图：**



**PL/SQL 代码：**

SET serveroutput ON

CREATE OR REPLACE PROCEDURE dept\_insert(

vdeptno IN dept.deptno%TYPE, --部门编号

vdname IN dept.dname%TYPE, -- 部门名称

vloc IN dept.loc%TYPE -- 部门地址

)

IS

cannotBeDevideByTen EXCEPTION; -- 声明不能被10整除的异常

BEGIN

IF MOD(vdeptno, 10) != 0 THEN RAISE cannotBeDevideByTen; -- 不能被10整除

ELSE

INSERT INTO dept VALUES(vdeptno, vdname, vloc); -- 能被10整除，插入数据

dbms\_output.put\_line('插入成功');

END IF;

EXCEPTION

WHEN cannotbedevideByTen

THEN dbms\_output.put\_line('部门编号数字应为10的倍数'); -- 输出异常提示信息

END;

BEGIN – 测试存储过程代码

dept\_insert(90,'WAHAHA','BJ');

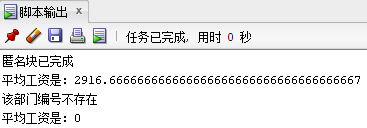
dept\_insert(65,'LILI','SH');

dept\_insert(70,'LILI','SH');

dept\_insert(3,'LILI','SH');

END;

1. 在SCOTT 模式中，定义一个函数 get\_avg\_sal ，用于计算 emp 表中指定某个部门 （部门编号）的平均工资，部门编号不存在视为异常，给出提示。 通过调用该函数验证其有效性。

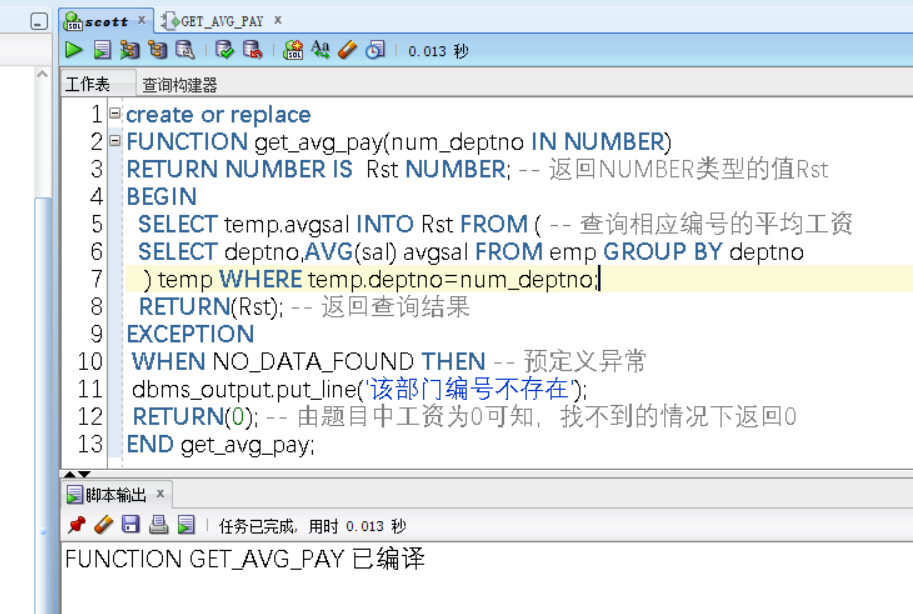


提示：get\_avg\_pay 原型如下

CREATE FUNCTION get\_avg\_pay(num\_deptno IN NUMBER)

RETURN NUMBER

**函数创建截图：**



**代码：**

create or replace

FUNCTION get\_avg\_pay(num\_deptno IN NUMBER)

RETURN NUMBER IS Rst NUMBER; -- 返回NUMBER类型的值Rst

BEGIN

SELECT temp.avgsal INTO Rst FROM ( -- 查询相应编号的平均工资

SELECT deptno,AVG(sal) avgsal FROM emp GROUP BY deptno

) temp WHERE temp.deptno=num\_deptno;

RETURN(Rst); -- 返回查询结果

EXCEPTION

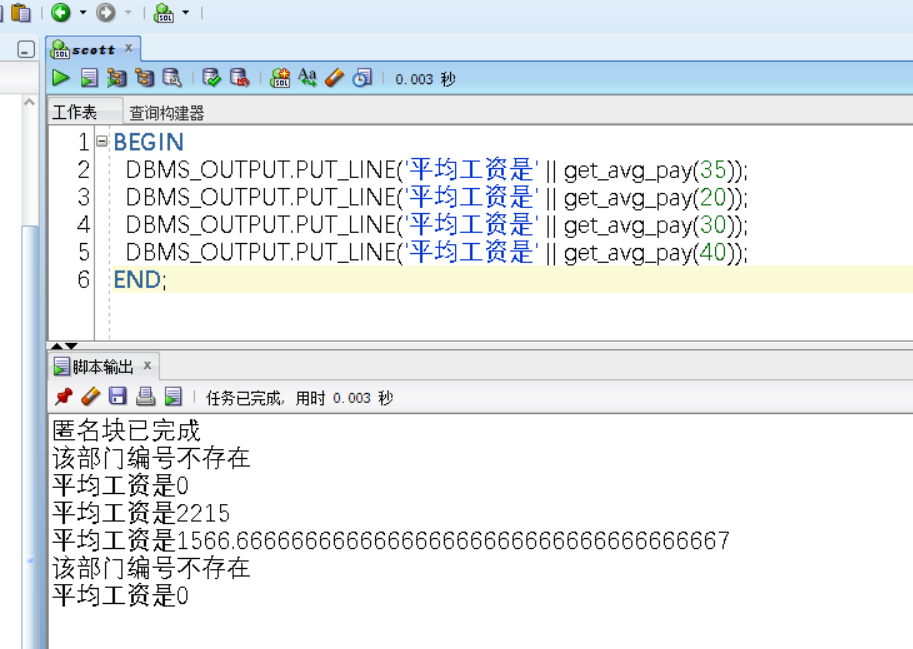
WHEN NO\_DATA\_FOUND THEN -- 预定义异常

dbms\_output.put\_line('该部门编号不存在');

RETURN(0); -- 由题目中工资为0可知，找不到的情况下返回0

END get\_avg\_pay;

**函数运行测试：**



**代码：**

BEGIN

DBMS\_OUTPUT.PUT\_LINE('平均工资是' || get\_avg\_pay(35));

DBMS\_OUTPUT.PUT\_LINE('平均工资是' || get\_avg\_pay(20));

DBMS\_OUTPUT.PUT\_LINE('平均工资是' || get\_avg\_pay(30));

DBMS\_OUTPUT.PUT\_LINE('平均工资是' || get\_avg\_pay(40));

END;

### 心得体会：

本次上机一切顺利，在使用SQL Developer进行数据的DML、DQL、DCL等操作的时候一切顺利，没有遇到问题，但是在使用PL/SQL编程的时候由于不熟悉PL/SQL语言，在操作的时候借鉴了PPT等资料。

游标是数据库查询中比较方便的一个机制，他使得我们与数据库沟通变得更加方便，在上机实验的时候我切实体会到了它的好处。

实验4: Oracle数据库控制

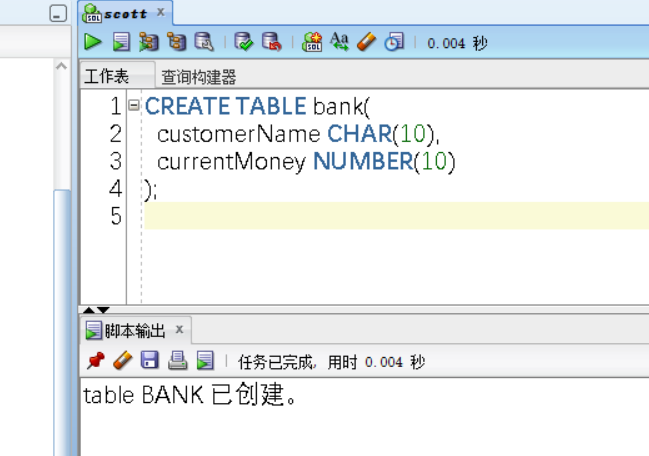
### 实验目的：

1. 了解Oracle事务控制方法，掌握事务相关语句的使用方法。
2. 了解Oracle锁机制，掌握显示锁的使用方法。

### 实验内容：

1. **按如下步骤简单模拟银行转账**
   1. 在SCOTT模式中，创建一个银行表bank，包含两个字段。

顾客姓名customerName CHAR(10)和当前余额currentMoney number(10)



代码：

CREATE TABLE bank(

customerName CHAR(10),

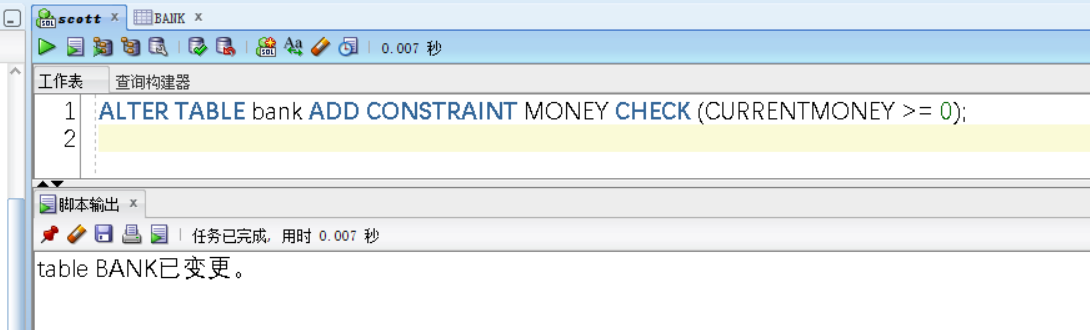
currentMoney NUMBER(10)

);

* 1. 添加余额不能小于0的约束。

提示：使用ALTER TABLE … ADD CONSTRAINT…语句

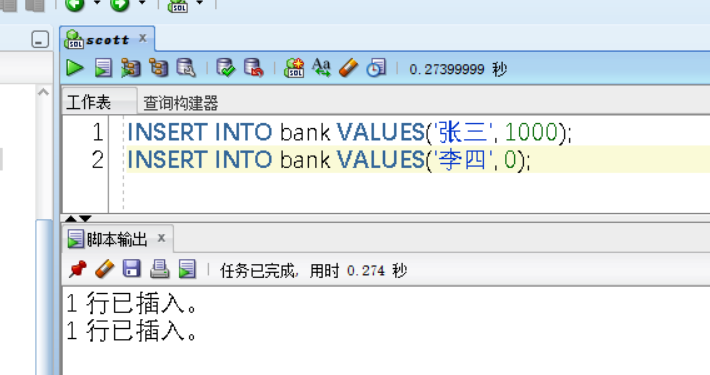
ALTER TABLE bank ADD CONSTRAINT MONEY CHECK (CURRENTMONEY >= 0);

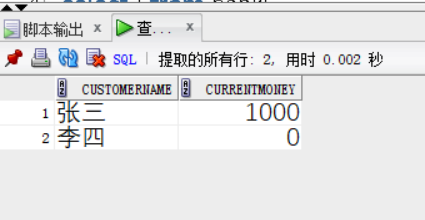


* 1. 插入数据：张三余额1000，李四余额0，提交，查看表中数据。

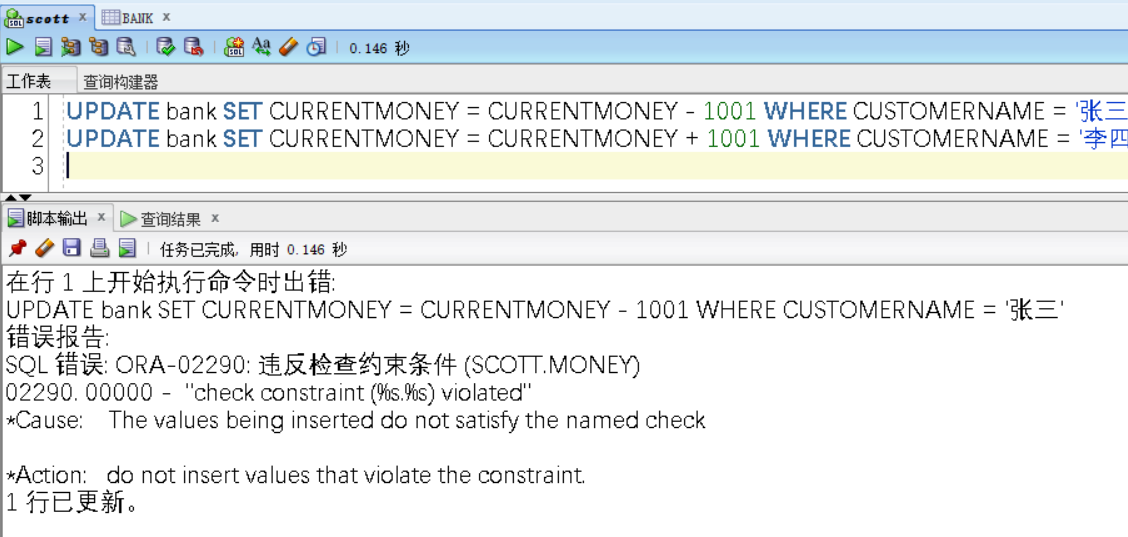
INSERT INTO bank VALUES('张三', 1000);

INSERT INTO bank VALUES('李四', 0);





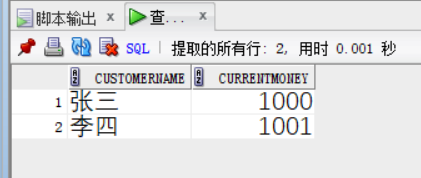
* 1. 张三给李四转1001元。（张三余额减1001，李四余额加1001，提交。）



UPDATE bank SET CURRENTMONEY = CURRENTMONEY - 1001 WHERE CUSTOMERNAME = '张三';

UPDATE bank SET CURRENTMONEY = CURRENTMONEY + 1001 WHERE CUSTOMERNAME = '李四';

* 1. 查看表中数据，发现什么情况？如何解决？



张三的钱没有减少，但李四的钱增加了。

如果发生了错误，应该回滚到两条语句执行之前。

* 1. 重写程序，增加异常处理部分，异常处理部分包括提示和回滚，请将下列补充完整。

set serveroutput on;

begin

update bank set currentMoney=currentMoney+1001 where customerName='李四';

update bank set currentMoney=currentMoney-1001 where customerName='张三';

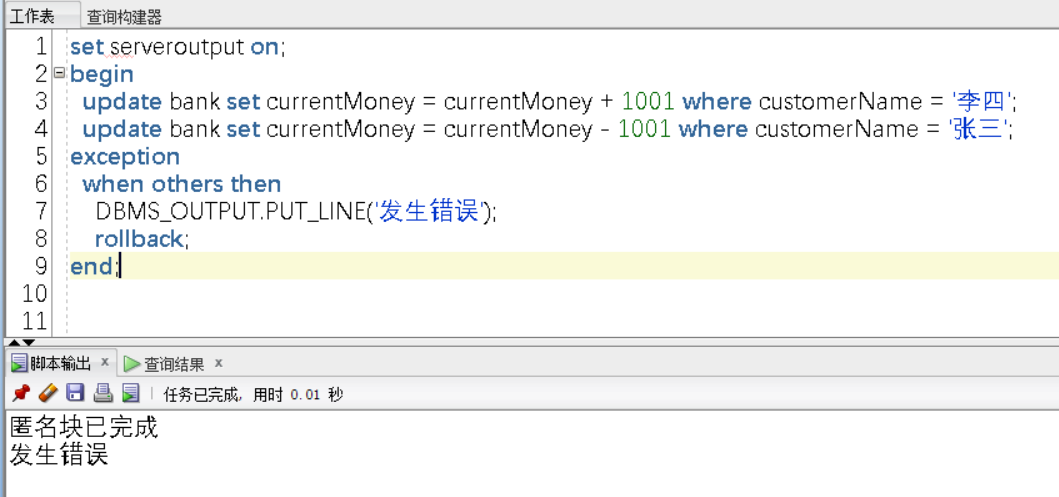
exception

when others then

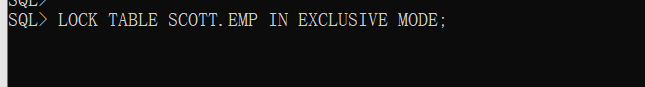
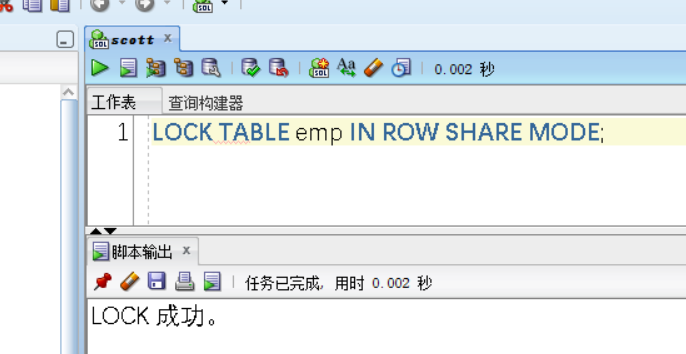
***DBMS\_OUTPUT.PUT\_LINE('发生错误');***

***rollback;***

end;



1. **演示Oracle锁机制。**
   1. 分别打开两个会话，会话1使用scott用户连接到PDBORCL，会话2使用sysdba登录数据库，将容器切换为PDBORCL。
   2. 会话1中使用LOCK TABLE语句为scott的emp表加行共享表锁，会话2中尝试使用LOCK TABLE语句为scott的emp表加排他锁，产生什么情况？如何解释

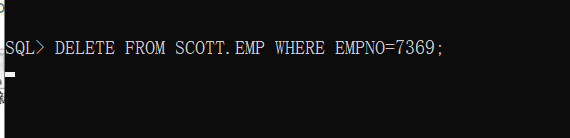
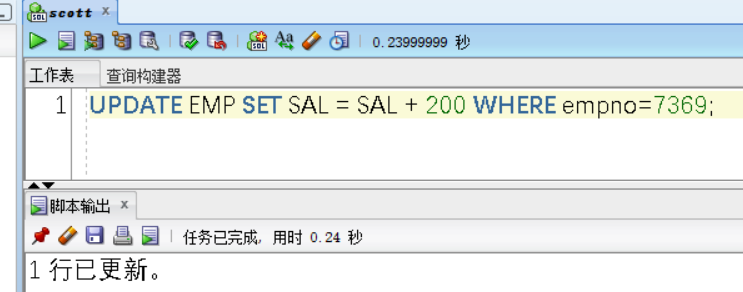


**此时数据库夯住了。一直停留在会话2的界面。**

**原因：**

**行共享表锁（RS锁）与排他锁（X锁）不兼容。当事务1拥有了表的RS锁后，会禁止事务2对相同表获取X锁，所以事务2会夯住。**

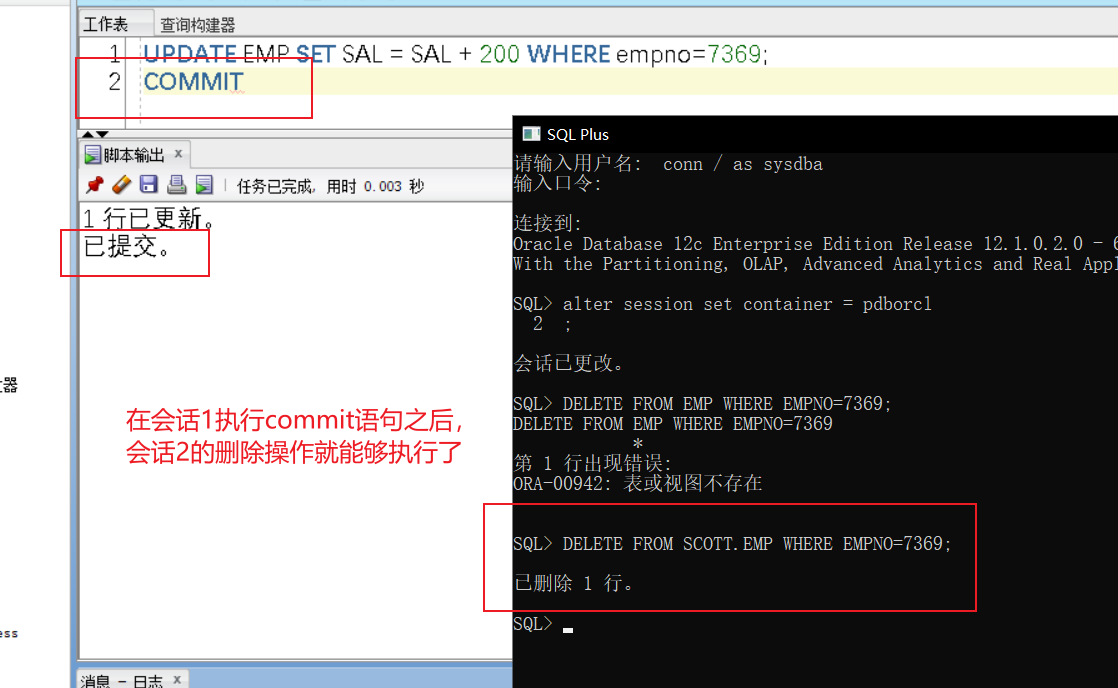
* 1. 会话1为7369号员工增加200薪水，会话2中尝试删除7369号员工对应的记录，将产生什么情况，如何解释。



**出现情况：会话1成功更新数据，会话2在删除的时候夯住了。**

**原因：当 Oracle 执行 DML 语句时，会先在所要操作的表上申请 TM锁，申请成功之后再申请TX锁，在上一条记录还没有commit之前，这条empno=7369的记录被锁住了，只有在上一个事务释放掉锁后才能进行操作，或用共享锁才能对此数据进行操作。**

**下面是原因验证:**



### 心得体会：

事务和锁是数据库操作中非常重要的一环，它保证了多线程，高并发状态下数据库中数据的安全，是我们必须掌握的技术。通过此次实验，我体会到了事务机制给用户带来的安全性和锁机制的灵活性。

本次实验属于演示型实验，难度不高，故没有遇到什么问题。

实验5: Oracle安全管理、备份与恢复

### 实验目的：

1. 了解Oracle安全管理常用语句。
2. 了解使用RMAN进行数据库备份与恢复的基本操作。

### 实验内容：

1. **查看上课讲过的所有数据视图内容。**
2. **完成下列用安全管理相关操作。**
   1. PDBORCL中，创建一个用户tuser1，自行设置口令，设置默认表空间为users，临时表空间为temp。

CREATE USER tuser1

IDENTIFIED BY 333 -- 密码为 333

DEFAULT TABLESPACE users -- 默认表空间为users

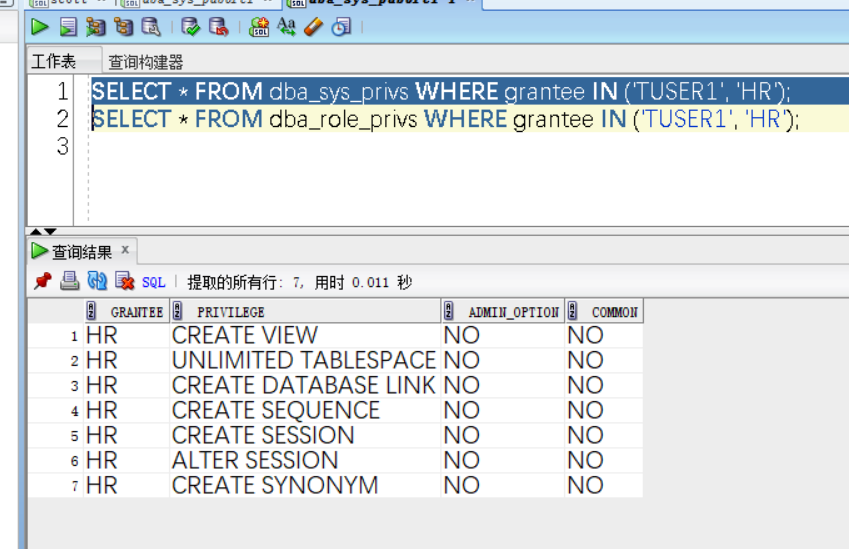
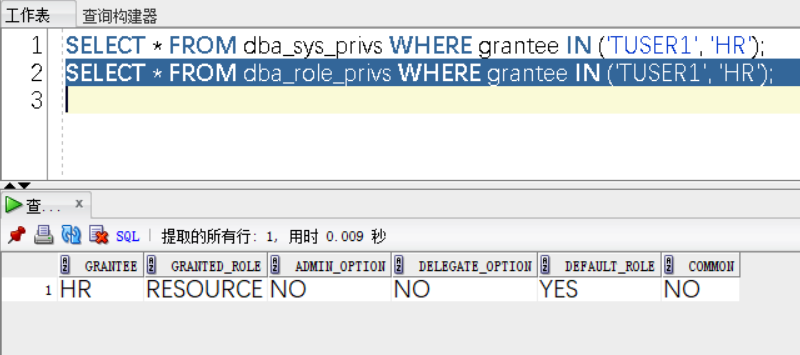
TEMPORARY TABLESPACE temp; -- 临时表空间为temp



* 1. 查看tuser1拥有的系统权限和角色，与HR用户做对比。

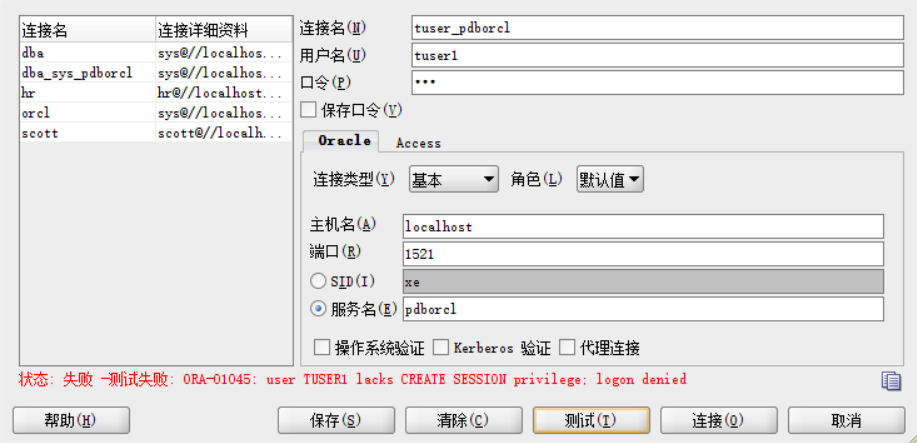
提示：使用dba\_sys\_privs和dba\_role\_privs视图

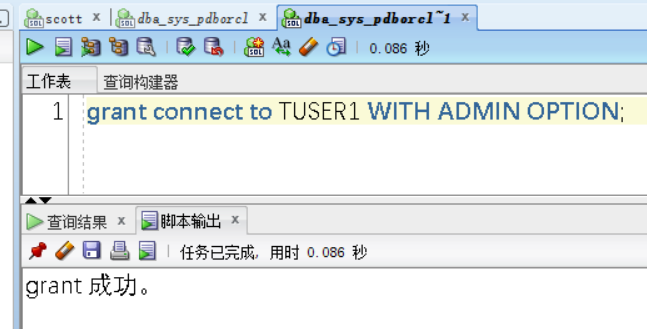
由图可知新用户没有相关的权限和角色

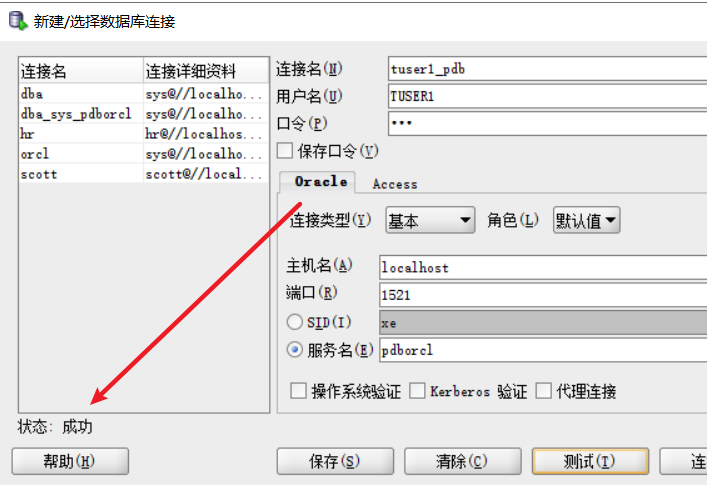
 

* 1. 尝试使用tuser1用户连接PDBORCL是否能成功？如不成功请予以解决，并进行连接，并使tuser1用户可以授权其它用户连接PDBORCL。

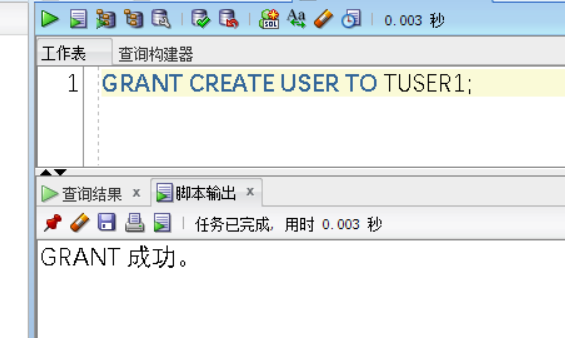
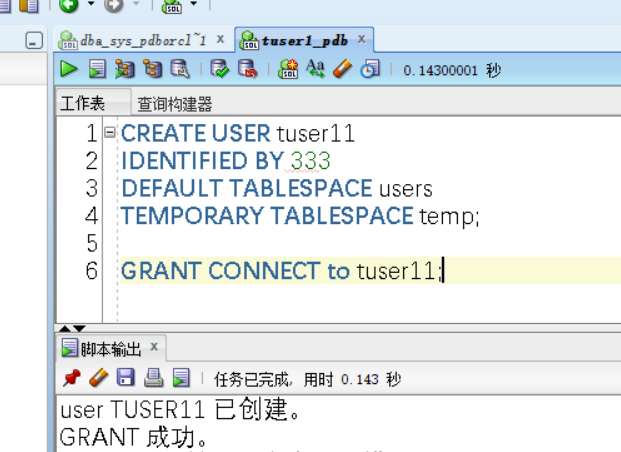
提示：赋予CONNECT角色，WITH ADMIN OPTION。

不成功

授权成功

测试成功。

* 1. 为tuser1用户赋予创建用户的系统权限，使用tuser1用户创建tuser11用户，使tuser11可以连接PDBORCL。

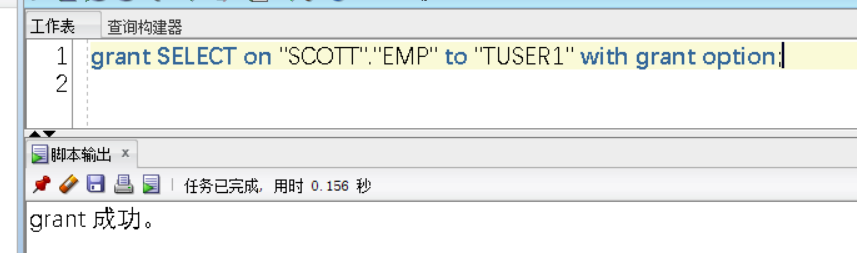
 

如图所示，创建角色并授权成功

* 1. SCOTT用户为tuser1用户赋予查看emp表的权限，tuser1将此权限传递给tuser11，验证传递成功。

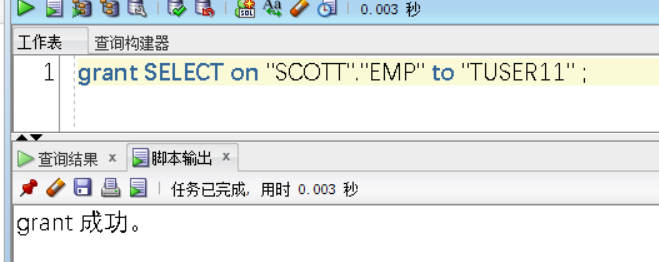
**SCOTT授权给TUSER1**

grant SELECT on "SCOTT"."EMP" to "TUSER1" with grant option;

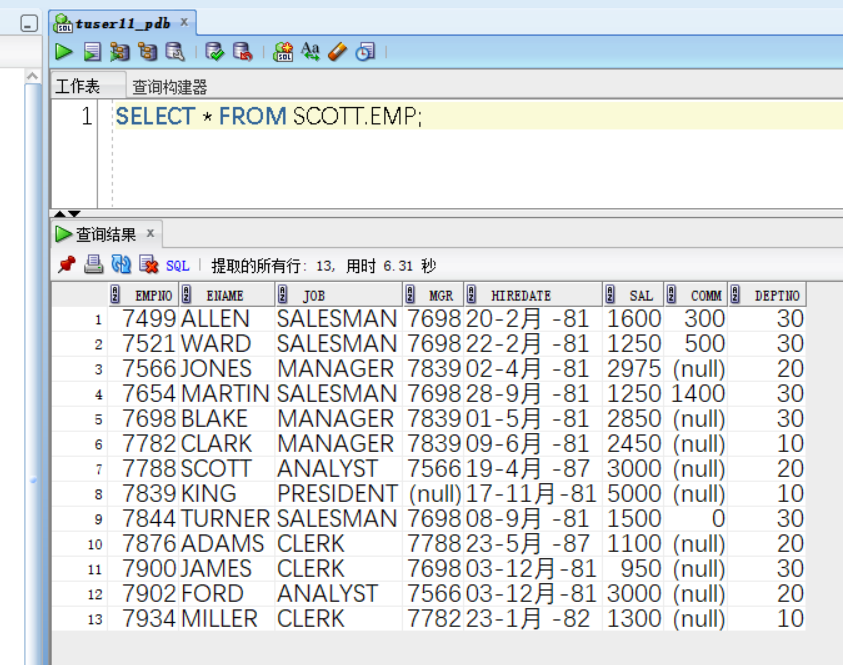


**TUSER1授权给TUSER11**

grant SELECT on "SCOTT"."EMP" to "TUSER11" ;



tuser11执行select语句，如图所示，可见查询成功



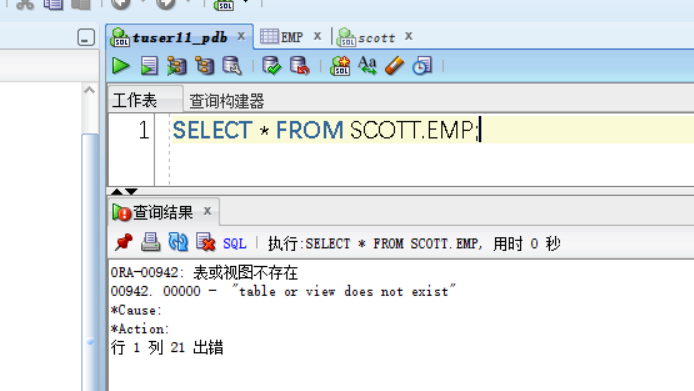
* 1. SCOTT用户收回tuser1用户查看emp表的权限，验证tuser11用户的权限是否被收回。

收回权限：

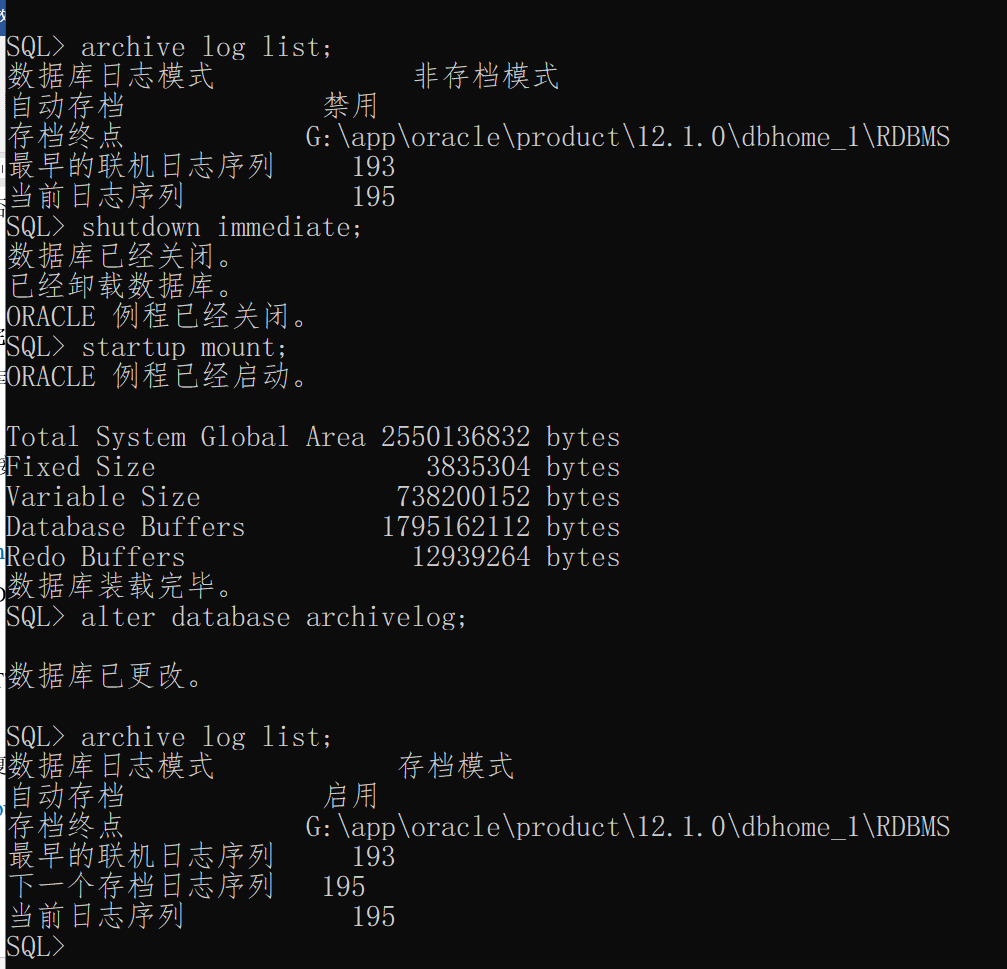
revoke SELECT on "SCOTT"."EMP" from "TUSER1" ;



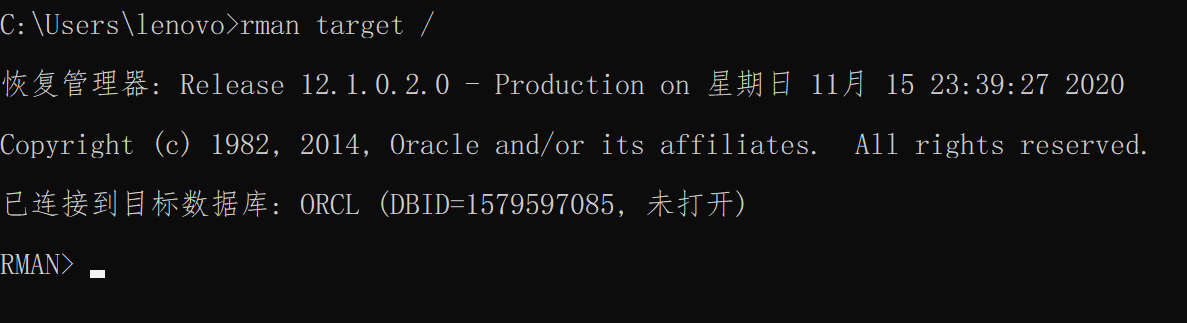
再次查询，发现表或视图不存在，因此tuser11的**权限已经被收回。**



1. **使用RMAN完成备份与恢复操作。**
   1. 切换数据库为归档模式。

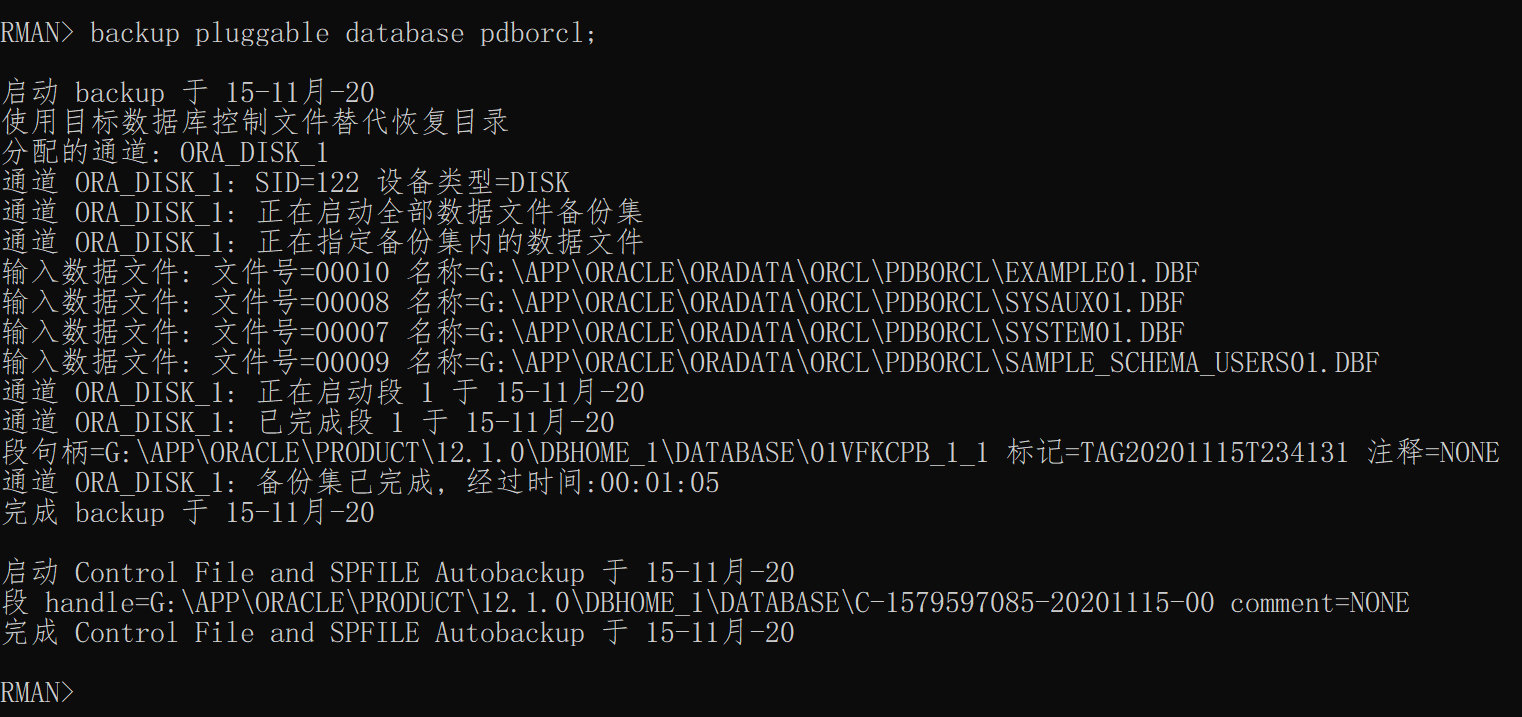


* 1. RMAN连接数据库。

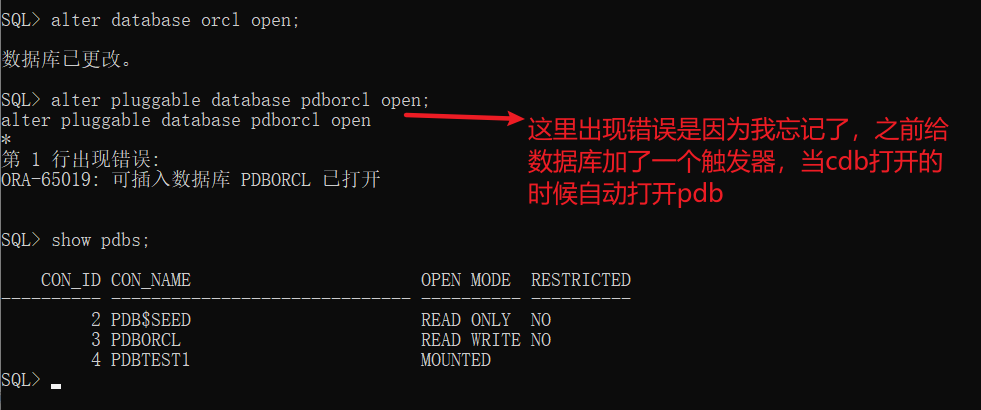


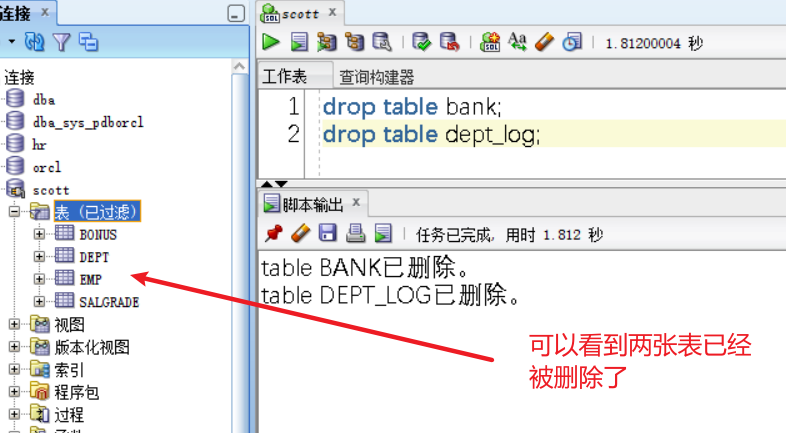
CMD下：rman target /

* 1. 备份PDBORCL。



* 1. 修改SCOTT.emp表数据。

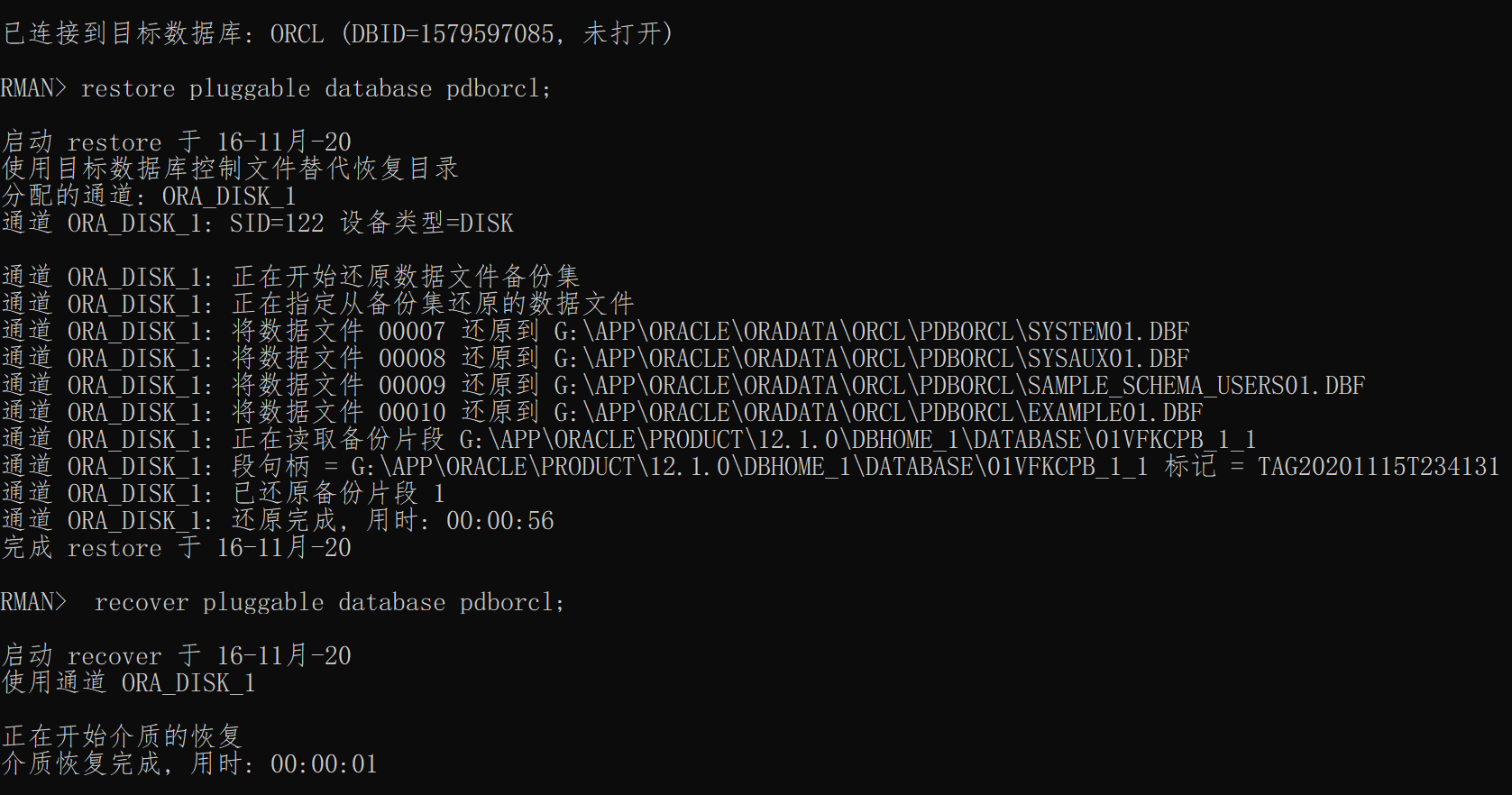




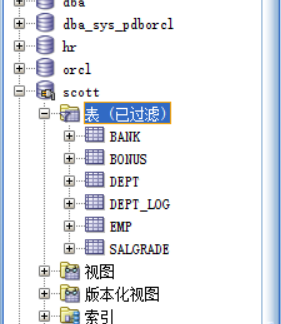
* 1. 还原并恢复PDBORCL。

提示： mount状态下进行还原并修复

这时候应该重复之前操作，先关闭数据库，然后启动到mount模式。



接下来查看数据恢复情况：

可以看到数据表已经恢复了。

### 心得体会：

数据库的安全是我们一直重视的问题，当数据库的数据遭到了破坏，的时候，备份和恢复就起到了作用，本次实验通过演示Oracle的数据恢复功能，体会到了数据库在数据方面的保障，顺便复习了Oracle体系结构的相关知识。