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1901-3 |
| 姓名 | 崔金泽 | 学号 | 20192163 |
| 实验日期 | 2021-10-28 | 评分 |  |

目 录

[实验一 ：Oracle 数据库基本操作 2](#_Toc90717729)

[实验二: 数据库编程 11](#_Toc90717730)

[一、预习任务 11](#_Toc90717731)

[二、实验步骤 11](#_Toc90717732)

[实验三: Oracle 数据库控制 21](#_Toc90717733)

[一、预习任务 21](#_Toc90717734)

[二、实验步骤 21](#_Toc90717735)

[实验四: Oracle 安全管理、备份与恢复 28](#_Toc90717736)

[一、预习任务 28](#_Toc90717737)

[二、实验步骤 29](#_Toc90717738)

[实验五 高校招生系统说明书 37](#_Toc90717739)

[一 需求分析：系统开发背景分析 37](#_Toc90717740)

[1． 背景 37](#_Toc90717741)

[2． 实体关系 38](#_Toc90717742)

[3． 表结构设计 38](#_Toc90717743)

[二 系统实现 38](#_Toc90717744)

[2.1创建用户并授权 38](#_Toc90717745)

[2.2表、序列、索引 39](#_Toc90717746)

[2.3视图 40](#_Toc90717747)

[2.4初始化数据 41](#_Toc90717748)

[2.5函数、存储过程、触发器、pl/sql 41](#_Toc90717749)

[2.6权限与安全 46](#_Toc90717750)

[2.7性能分析 46](#_Toc90717751)

[三 心得体会 46](#_Toc90717752)

# 实验一 ：Oracle 数据库基本操作

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。

2. 掌握 CDB 和 PDB 相关语句

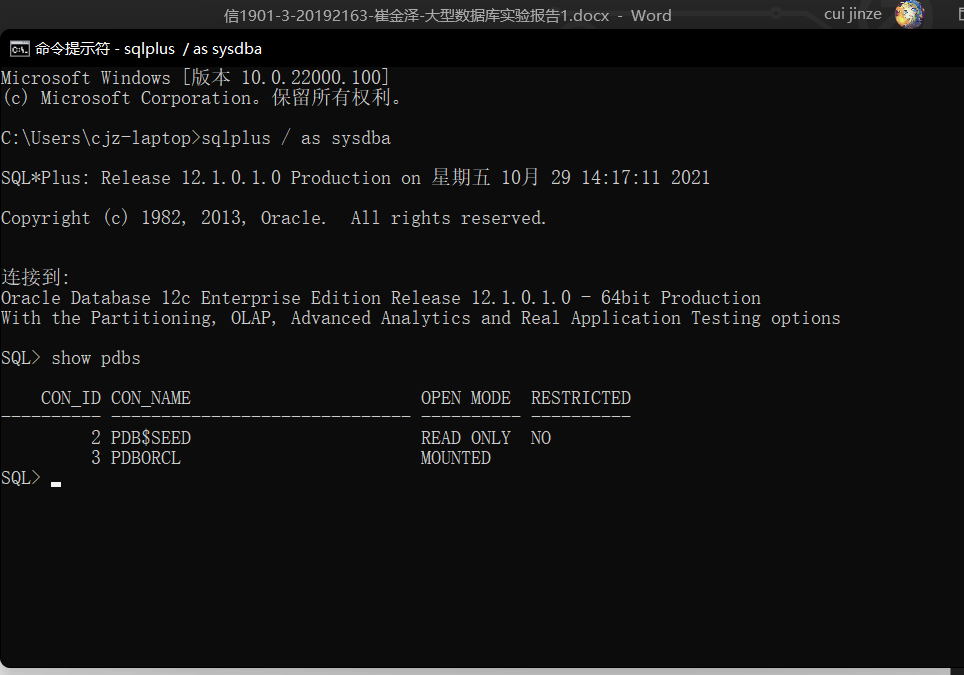
Windows 命令行窗口（cmd）以 DBA 权限连接数据库：sqlplus / as sysdba

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。



2) 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。



3) 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。

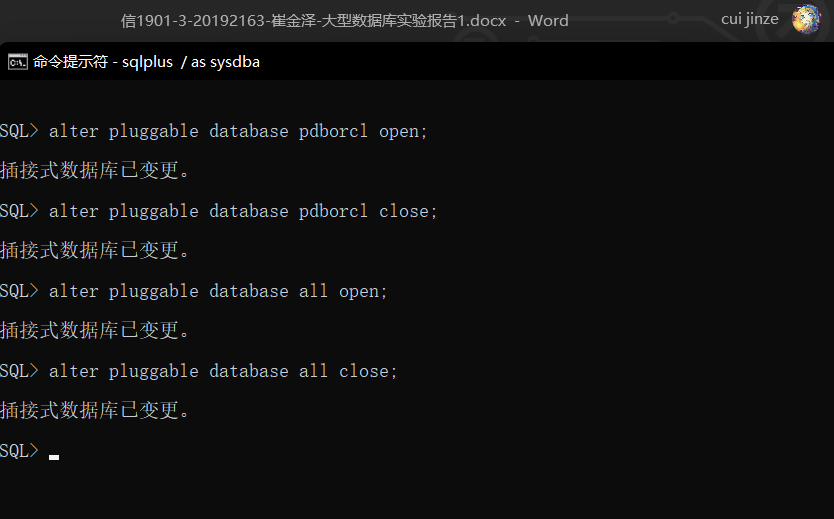


4) 创建或克隆前要指定文件映射的位置（需要 CBD 下 sysdba 权限）

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



5) 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。



6) 克隆 PDB（需要 CBD 下 sysdba 权限）

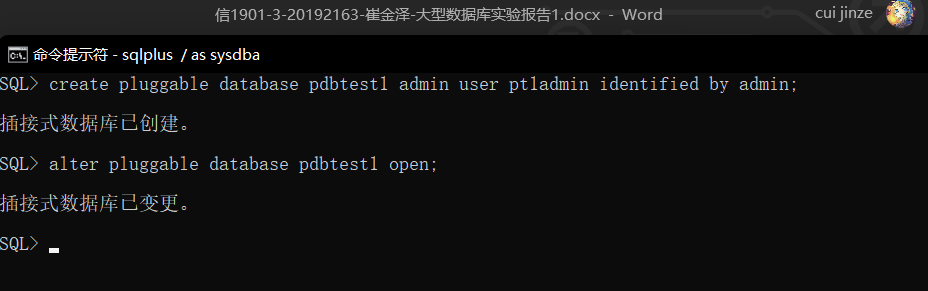
alter pluggable database *pdborcl* open;

--*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

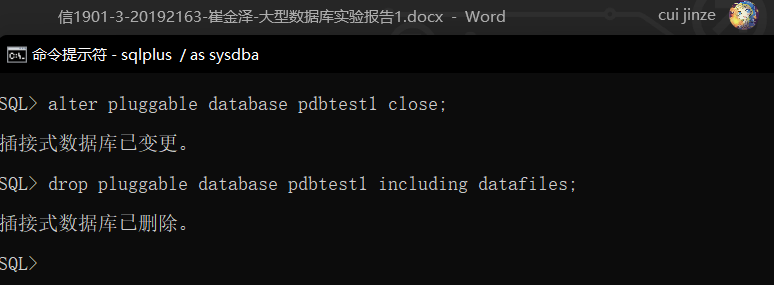


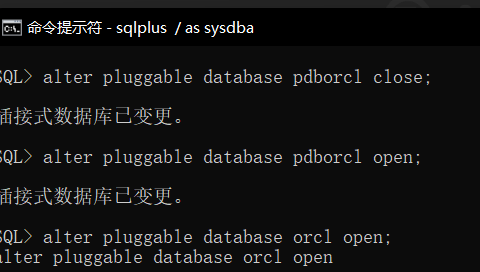
7) 删除 PDB（需要 CBD 下 sysdba 权限）

alter pluggable database *pdbtest1* close;

--关闭之后才能删除

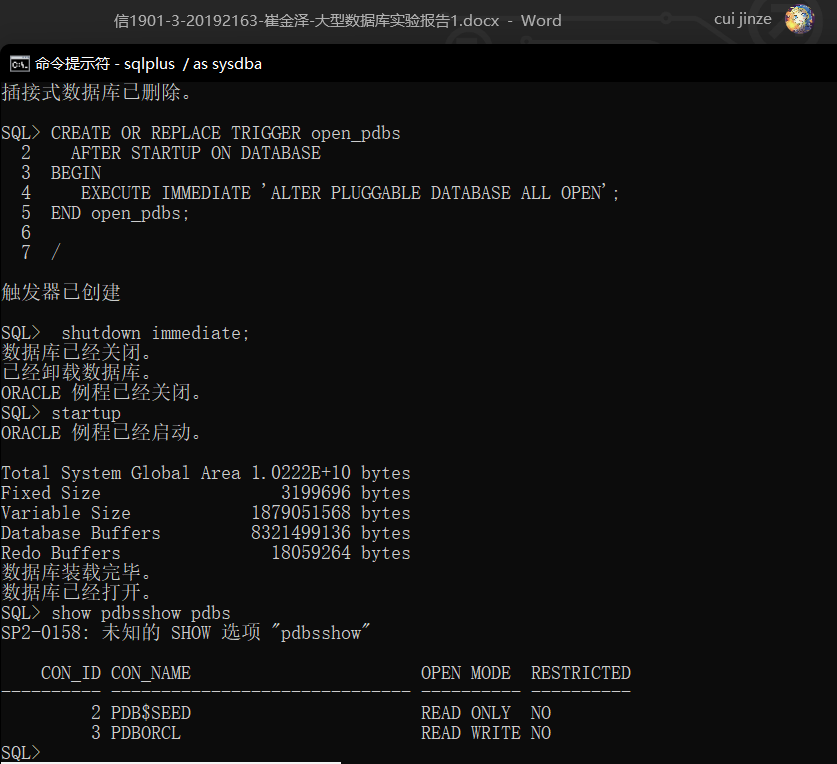
drop pluggable database *pdbtest1* including datafiles; --删除 *pdbtest1*





**8) 创建一个触发器让 PDB 能够在 CDB 启动时打开，并验证效果。（思**

**考题，自行查询相关资料完成）**

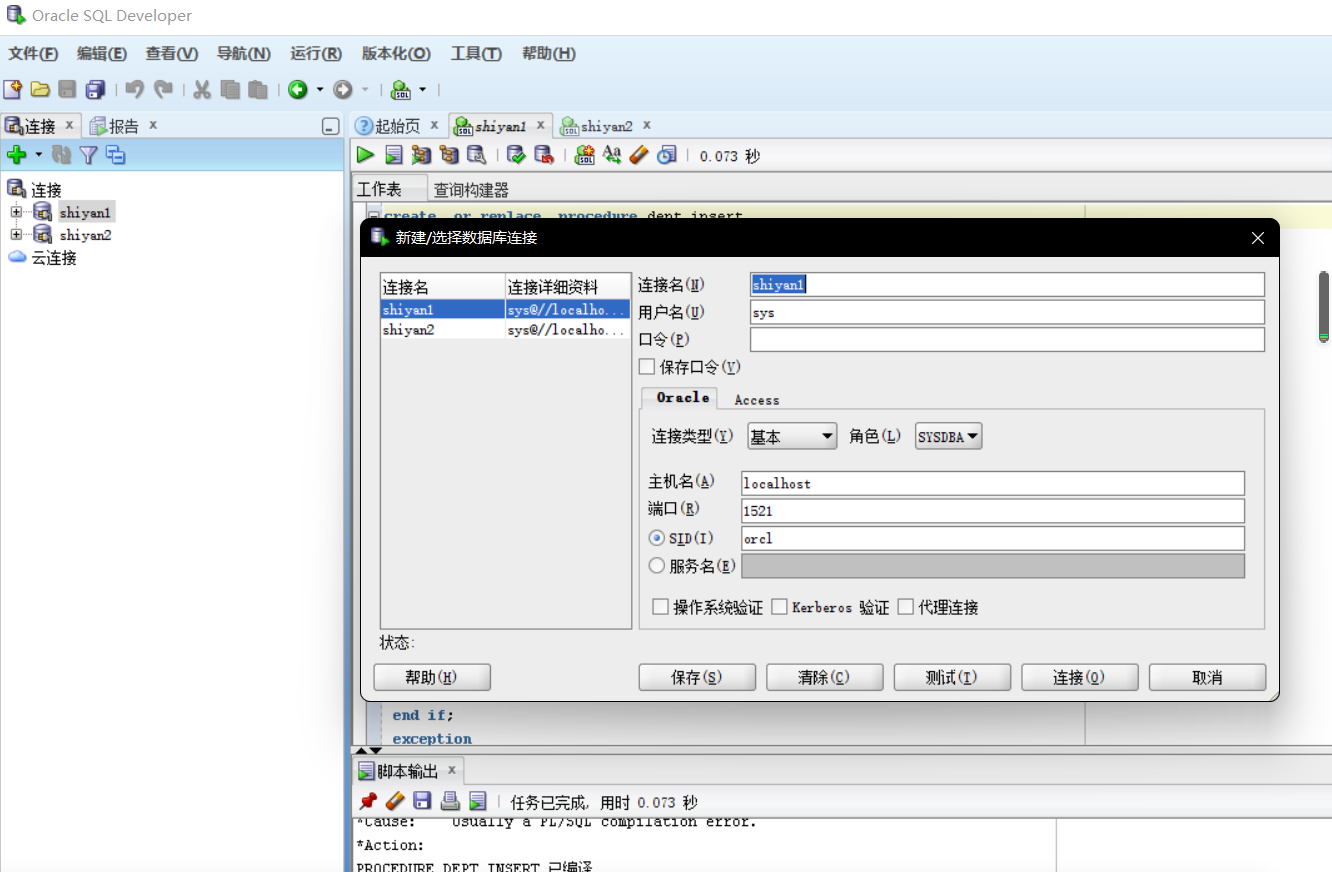


3. 使用 SQL Developer 启用 SCOTT 和 HR 账户，熟悉 SCOTT 模式和

HR 模式下包含的数据表及表内字段含义。

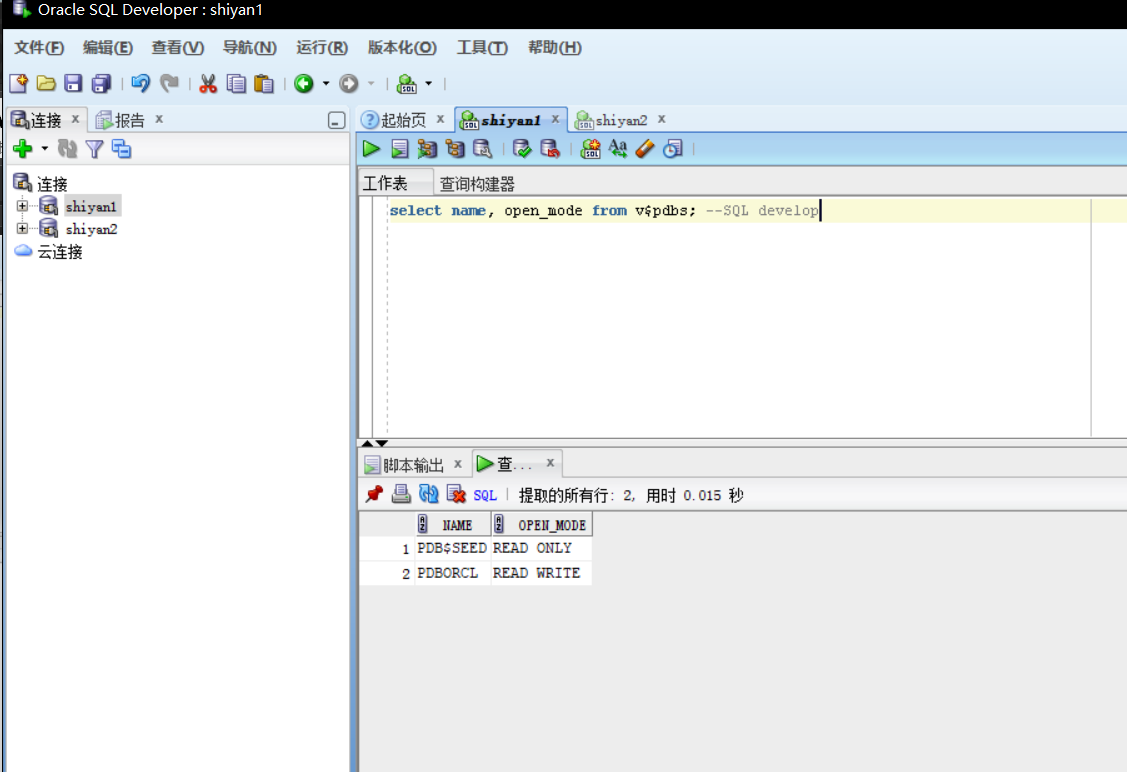
1) 使用 SQL Developer，以 sys 用户和 sysdba 权限连接安装 Oracle

时默认安装的数据 *ORCL*。



2) 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



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

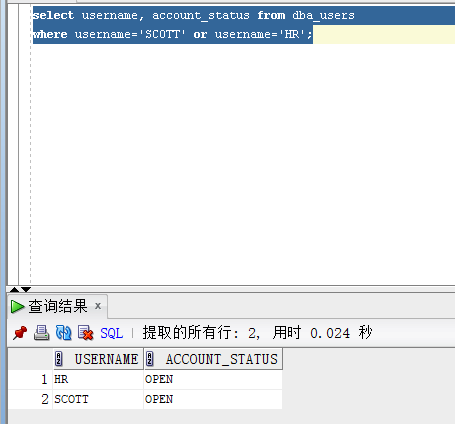
4) 使用 SQL Developer，以 sys 用户和 sysdba 权限连接默认的 PDB

（*PDBORCL*）。

5) 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' or username='HR';



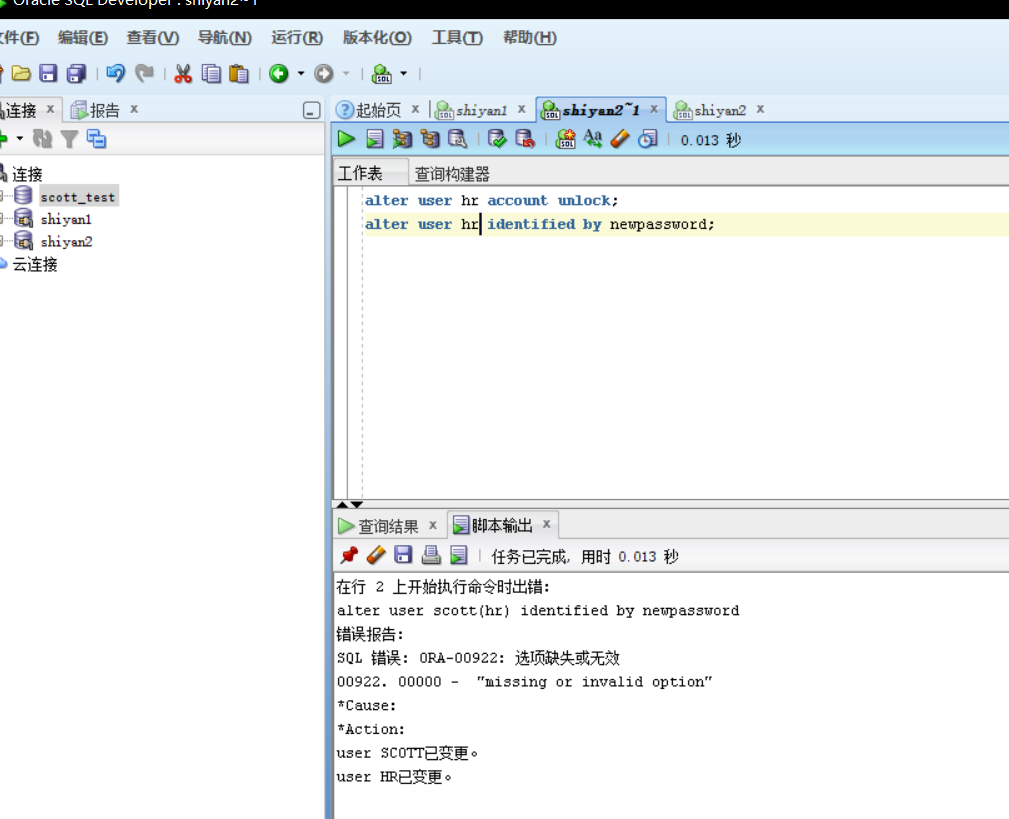
6) 如果 SCOTT 和 HR 用户为锁定和密码过期状态（LOCKED

&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;

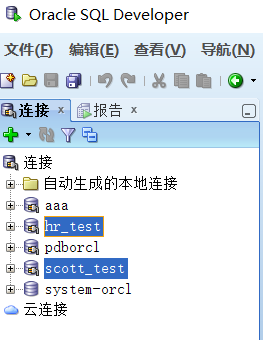
alter user scott(hr) identified by *newpassword*;

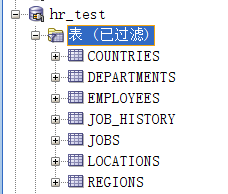


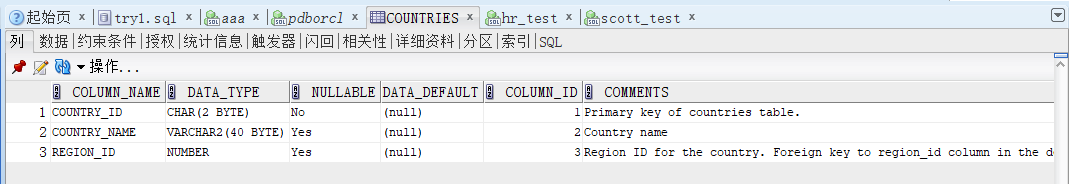
7) 使用 SQL Developer，新建两个连接，分别以 SCOTT 和 HR 用户连

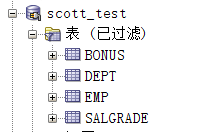
接默认的 PDB（*PDBORCL*），查看 SCOTT 模式下和 HR 模式下的

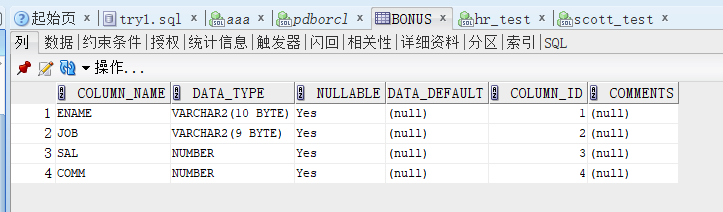
数据表，了解每个表的字段含义和数据类型，大致查看数据内容。











|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称 | 大型数据库技术应用 | 班级 | 信1901-3 |
| 姓名 | 崔金泽 | 学号 | 20192163 |
| 实验日期 | 2021-11-5 | 评分 |  |

# 实验二: 数据库编程

一、预习任务

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。

SCOTT : Oracle 便于用户学习所提供的内置用户。 HR ：一个示例用户，是在安装Oracle，创建数据库时选中“示例数据 库”后产生的，实际就是模拟一个人力资源部的数据库。 SYS：所有 Oracle 的数据字典的基表和视图都存放在 sys 用户中，这些 基表和视图对于 Oracle 的运行至关重要，由数据库自己维护，任何用 户都不能手动更改。sys用户拥有 dba，sysdba，sysoper 等角色或权限， 是 Oracle 权限最高的用户。SYSTEM：用于存放次一级的内部数据，如 Oracle 的一些特性或工具 的管理信息。system 用户拥有普通 dba 角色权限。

匿名块是未命名的块，在运行时被执行，匿名块不存储在数据库中，如果要再次执行相同的块，则必须重写该块，并在执行时重新进行编译。由于块是匿名的，每次执行后该块就不再存在，开发者不能调用以前写过的匿名块。

过程和函数都称作子程序，它们是匿名块的有益补充，是存储在数据库中被命名的PL/SQL块。由于子程序已经命名，且存储在数据库中，开发者可以随时调用它们，可以将子程序声明为过程或函数，通常使用过程执行一些操作，而使用函数进行计算并返回一个值。

函数和过程类似，不同的是函数必须有返回值，而过程没有。

标识符用于定义PL/SQL变量、常量、异常、游标名称、游标变量、参 数、子程序名称和其他的程序单元名称等。在PL/SQL程序中，标识符是以字母开头的，后边可以跟字母、数字、 美元符号（$）、井号（#）或下划线（\_），其最大长度为30个字符。标识符中不能包含减号“ - ”和空格，不能使SQL保留字例如，X，v\_empno，v\_$等都是有效的标识符，而X-y，\_temp则 是非法的标识符。

Oracle的触发器有三类:DML触发器、INSTEAD OF触发器和系统触发器。

DML触发器:在DML操作前或操作后进行触发，并且可以在每个行或语句操作上进行触发。DML触发器可分为INSERT触发器、UPDATE触发器和DELETE触发器三类。

替代触发器:在ORACLE里，不能直接对由两个以上的表建立的视图进行操作。所以给出了替代触发器，为进行视图操作提供处理方法。

系统触发器:系统触发器由数据定义语言(DDL)事件(如CREATE语句、ALTER语句、DROP语句)、数据库系统事件(如系统启动或退出、异常操作)、用户事件(如用户登录或退出数据库)触发。

二、实验步骤

（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：

1) 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）

create table hr.jobs\_temp(

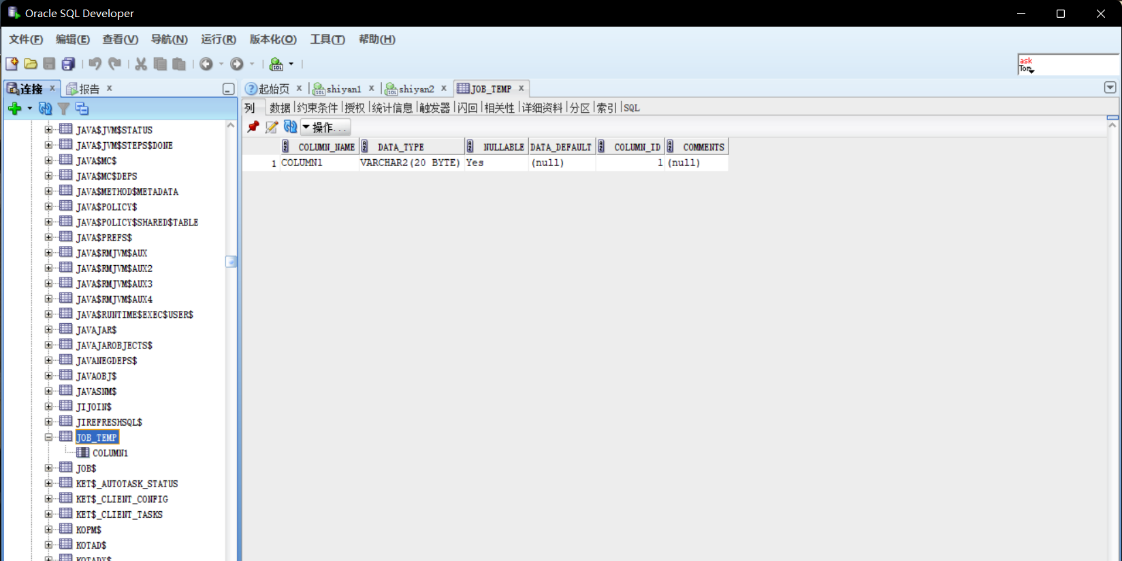
job\_id varchar2(10) not null primary key,

job\_title varchar2(35) not null,

min\_salary number(6),

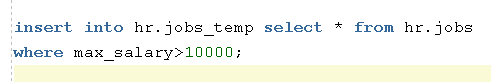
max\_salary number(6)

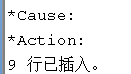
);



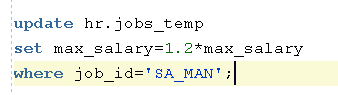
insert into hr.jobs\_temp select \* from hr.jobs

where max\_salary>10000;





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

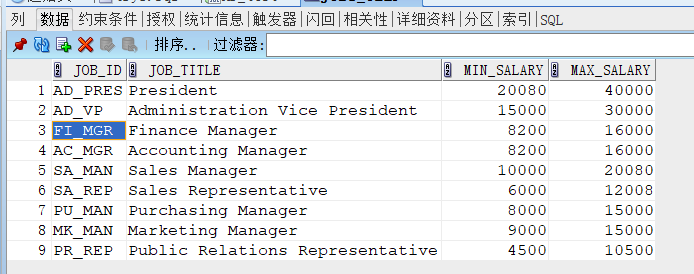


update hr.jobs\_temp

set max\_salary=1.2\*max\_salary

where job\_id='SA\_MAN';

修改前：

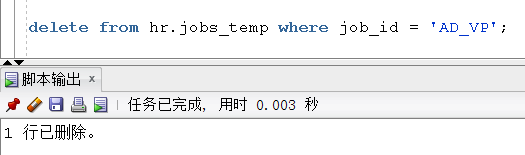


修改后：



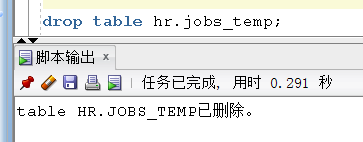
3) 删除表 jobs\_temp 中 job\_id 为 AD\_VP 的记录。

delete from hr.jobs\_temp where job\_id = 'AD\_VP';



4) 删除表 jobs\_temp。

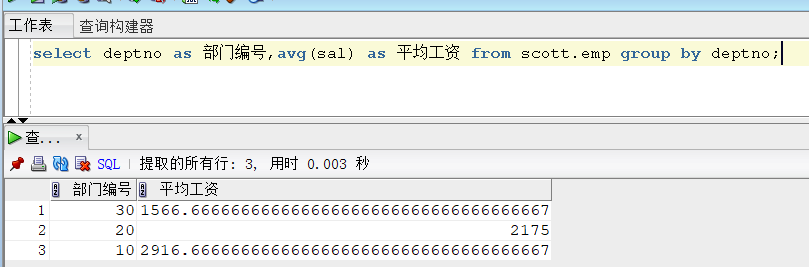
drop table hr.jobs\_temp;



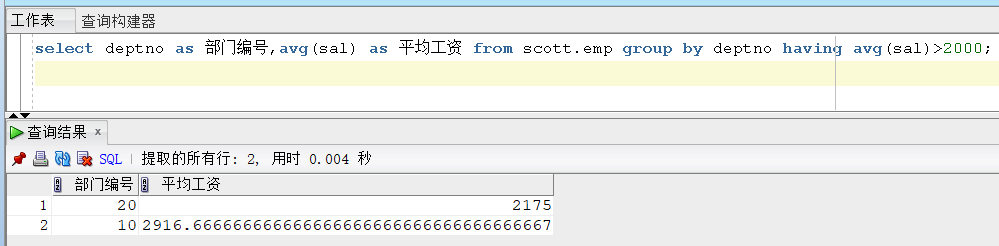
2. 在 SCOTT 模式，下用使用 SQL 语句完成下列操作

1) 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。

select deptno as 部门编号,avg(sal) as 平均工资 from scott.emp group by deptno;

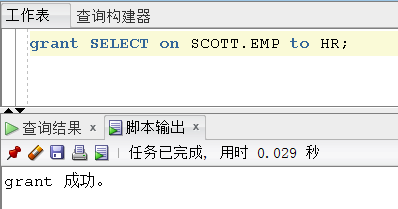


select deptno as 部门编号,avg(sal) as 平均工资 from scott.emp group by deptno having avg(sal)>2000;

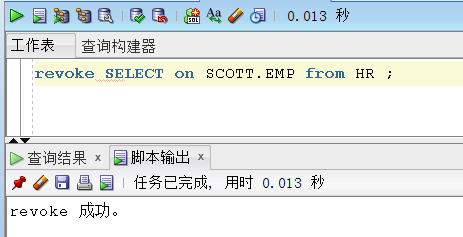


2) 把 EMP 表的查询权限授给用户 HR，验证授权成功，之后收回权限。

grant SELECT on SCOTT.EMP to HR;



revoke SELECT on SCOTT.EMP from HR ;



（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：

--SET serveroutput ON

DECLARE

type dept\_type IS record --声明record类型dept\_type

(

var\_DEPTNO NUMBER, --定义字段/成员变量

var\_DNAME VARCHAR2(20),

var\_LOC VARCHAR2(20)

);

var\_dept dept\_type; --定义变量

rowVar\_dept dept%rowtype;

BEGIN

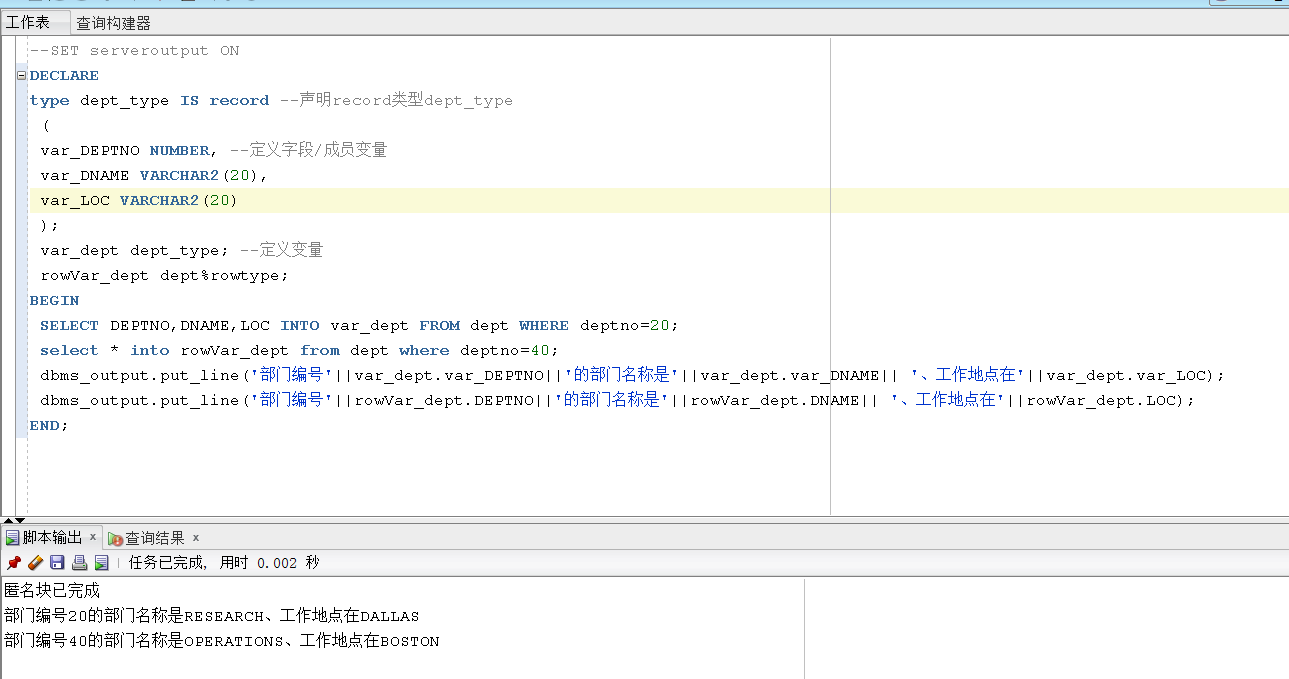
SELECT DEPTNO,DNAME,LOC INTO var\_dept FROM dept WHERE deptno=20;

select \* into rowVar\_dept from dept where deptno=40;

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);

END;



2. 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：

DECLARE

CURSOR cur\_emp IS SELECT \* FROM emp; -- 步骤1: 声明游标

v\_emp cur\_emp%ROWTYPE; -- 定义一个变量存放游标指示的内容

BEGIN

OPEN cur\_emp; -- 步骤2: 打开游标

LOOP

FETCH cur\_emp INTO v\_emp; -- 步骤3: 提取数据

EXIT WHEN cur\_emp%NOTFOUND;

if(v\_emp.SAL>2000)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; -- 步骤4: 关闭游标

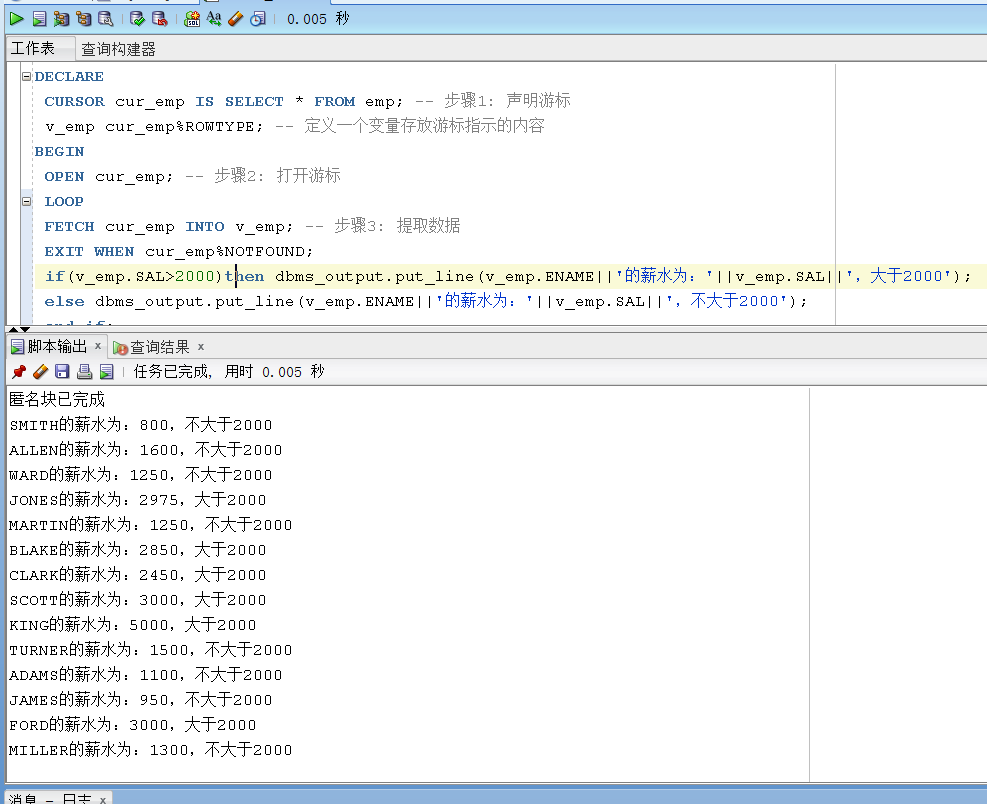
EXCEPTION

WHEN OTHERS THEN

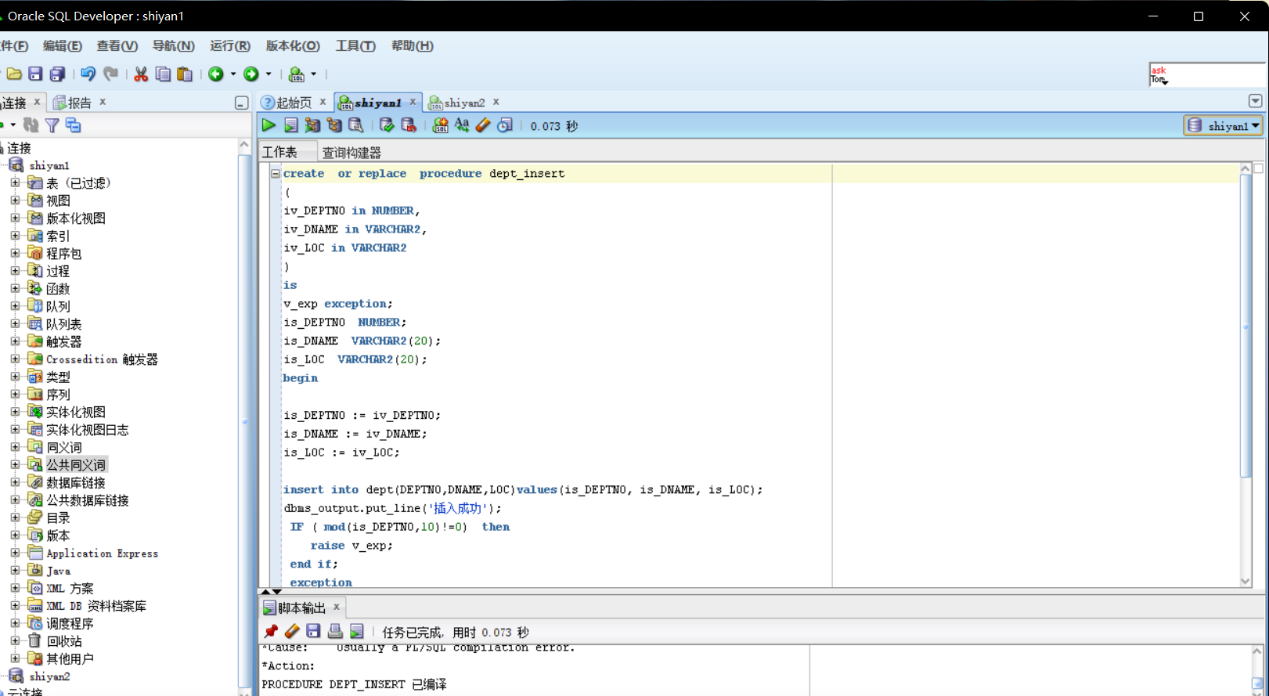
dbms\_output.put\_line(SQLCODE || ' : ' || SQLERRM);

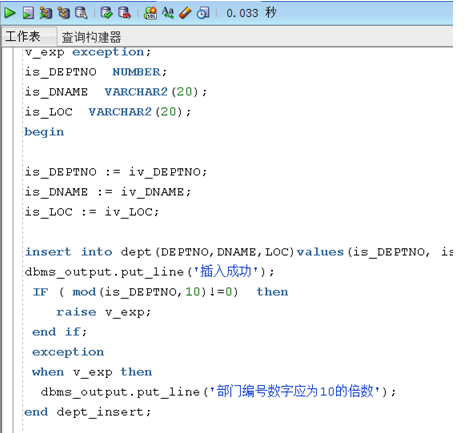
dbms\_output.put\_line(dbms\_utility.format\_error\_backtrace);

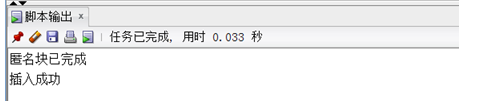
END;

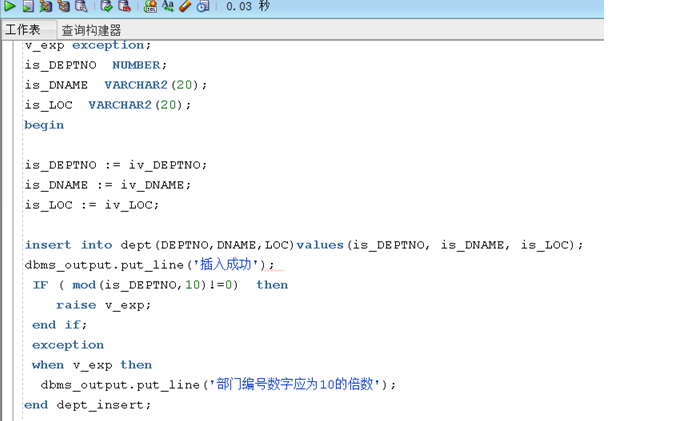


3. 在SCOTT 模式中，创建一个带参数的存储过程 dept\_insert 实现向dept 表中插入数据， 插入时 deptno 不为 10 的整数倍视为异常，给出提示。 通过调用该存储过程插入，验证其有效性。











create or replace procedure dept\_insert

(

iv\_DEPTNO in NUMBER,

iv\_DNAME in VARCHAR2,

iv\_LOC in VARCHAR2

)

is

v\_exp exception;

is\_DEPTNO NUMBER;

is\_DNAME VARCHAR2(20);

is\_LOC VARCHAR2(20);

begin

is\_DEPTNO := iv\_DEPTNO;

is\_DNAME := iv\_DNAME;

is\_LOC := iv\_LOC;

insert into dept(DEPTNO,DNAME,LOC)values(is\_DEPTNO, is\_DNAME, is\_LOC);

dbms\_output.put\_line('插入成功');

IF ( mod(is\_DEPTNO,10)!=0) then

raise v\_exp;

end if;

exception

when v\_exp then

dbms\_output.put\_line('部门编号数字应为10的倍数');

end dept\_insert;

4.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）

--创建函数

CREATE OR REPLACE FUNCTION get\_avg\_sal

(

empname IN VARCHAR2

)

RETURN NUMBER

IS

Result NUMBER; --声明变量

BEGIN

SELECT sal INTO Result FROM emp WHERE ename=empname;

RETURN(Result);

END get\_sal;

--调用函数

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Scott的薪水为：'||get\_sal('SCOTT'));

END;

CREATE OR REPLACE FUNCTION get\_avg\_sal(eno emp.deptno%TYPE)

RETURN NUMBER

IS

avgsal NUMBER;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(avgsal);

SELECT AVG(sal) INTO avgsal FROM emp WHERE empno=eno;

IF((avgsal) = (avgsal))THEN RETURN avgsal;

else

DBMS\_OUTPUT.PUT\_LINE('该部门编号不存在');

return 0;

end if;

RETURN avgsal;

END;

BEGIN

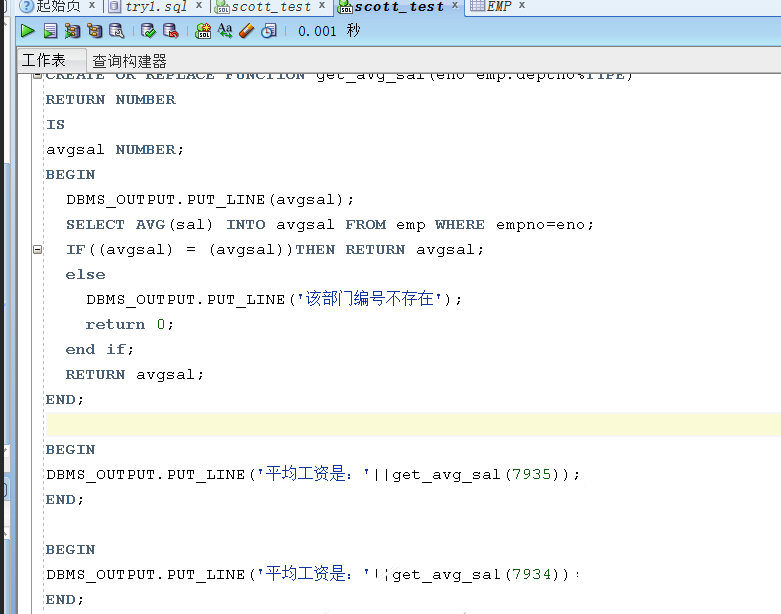
DBMS\_OUTPUT.PUT\_LINE('平均工资是：'||get\_avg\_sal(7935));

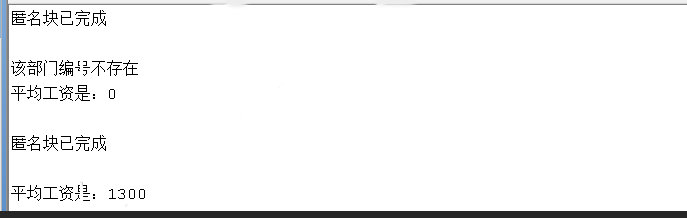
END;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('平均工资是：'||get\_avg\_sal(7934));

END;





|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称 | 大型数据库技术应用 | 班级 | 信1901-3 |
| 姓名 | 崔金泽 | 学号 | 20192163 |
| 实验日期 | 2021-11-05 | 评分 |  |

# 实验三: Oracle 数据库控制

一、预习任务

事务（Trasanction）是由一系列语句构成的逻辑工作单元，通常是为了完成一定的业务逻辑而将一条或多条语句“封装”起来，是它们与其它语句之间存在一个逻辑上的边界。事务中的语句要么全部执行，完成整个工作单元的操作。要么全部不执行。

事务的提出主要是为了解决并发情况下保持数据一致性的问题。每个事务从开始到结束的整个生命周期可以分为以下几种状态，DBMS记录每个事务的生命周期状态，以便恢复时进行不同的操作处理。活动状态：事务在执行时的状态。部分提交状态：事务中最后一条语句被执行后的状态。事务虽然已经完成，但由于实际输出可能在内存中，未被写回，可能还会发生硬件故障而失败，并进入中止状态。提交状态：事务在部分提交后，将往磁盘中写入数据，最后一条信息写入后的状态称为提交状态，进入提交状态的事务就成功完成了。失败状态：事务不能正常执行的状态。导致失败状态的发生可能原因有硬件问题或逻辑错误，这时事务必须回滚，进入中止状态。中止状态：事务回滚，并且数据库已经恢复到事务开始执行前的状态。

事务管理的命令包括：COMMIT，语句提交事务。ROLLBACK语句回退全部事务。设置保存点 回退部分事务。多用户数据库必须保证数据的并发性和数据的一致性。并发性是指允许多用户访问同一数据，一致性是指多用户数据库中每个用户看到的数据都是一致的，用户对数据进行修改和提交事务所产生的的变化对于其他用户也是可见的。

为了应对上述数据不一致问题，事务隔离级别的概念就产生了。隔离级别越高，上述问题就会越少，但是性能消耗也会越大。所以在实际生产过程中，要根据需求去确定隔离级别。

在 SQL 92 标准中从低到高定义了 4 个事务的隔离级别：Read uncommitted （读未提交）

Read committed（读已提交）Repeatable read（可重复读）Serializable（串行读）。

锁（Lock）用来防止多个共同访问共享数据的事务之间的破坏性交互，包括不正确地更新数据或不正确地更改基础数据结构。锁在多用户的环境下可以保证数据库的完整性和一致性。

事务内各语句获得的锁在事务执行期内有效，以防止事务间破坏性的相互干扰。如果某个事务中的 SQL 语句对数据进行了修改，只有在此事务提交后开始的事务才能看到前者修改的结果。当用户提交或回滚一个事务后，Oracle 将释放此事务内各个 SQL 语句获得的锁。

二、实验步骤

**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。

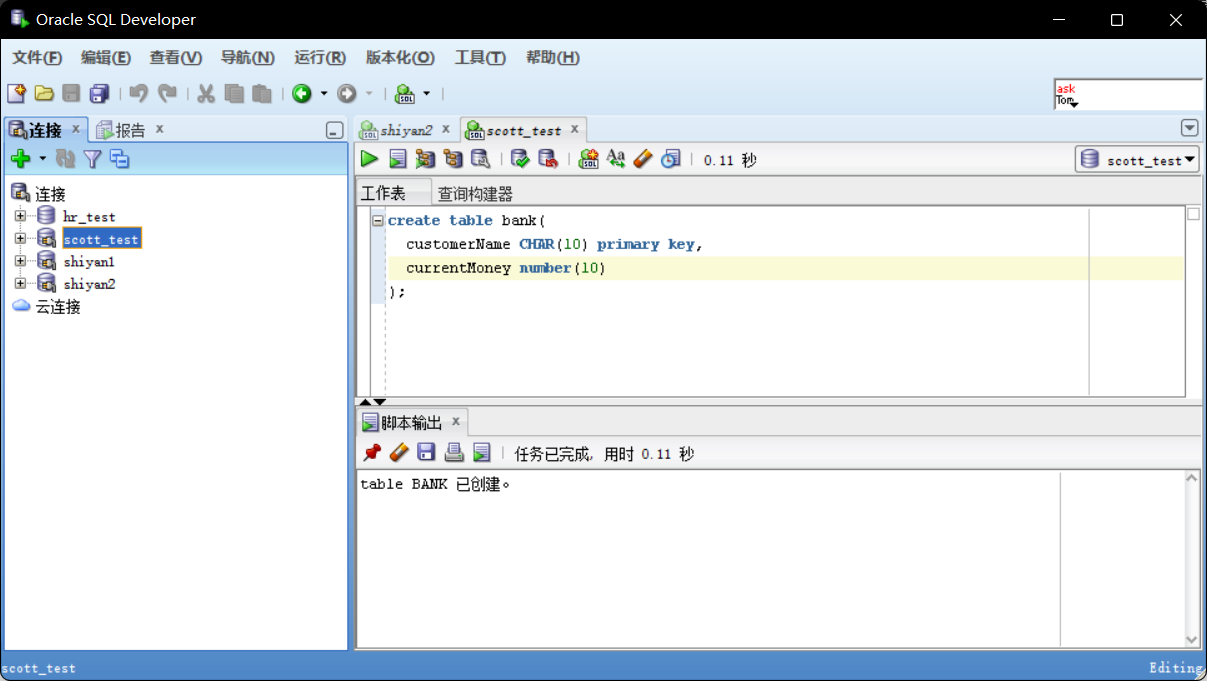
顾客姓名 customerName CHAR(10)和当前余额 currentMoney number(10)

create table bank(

customerName CHAR(10) primary key,

currentMoney number(10)

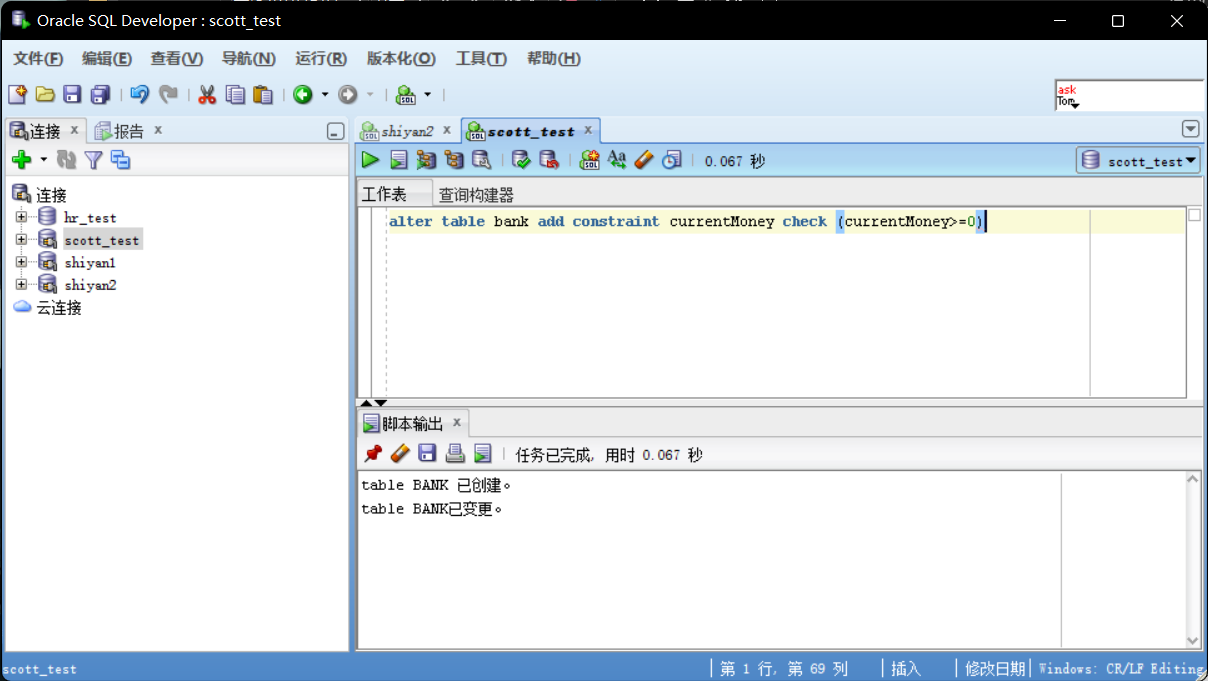
);



2)添加余额不能小于 0 的约束。

提示：使用 ALTER TABLE … ADD CONSTRAIN

alter table bank add constraint currentMoney check (currentMoney>=0)

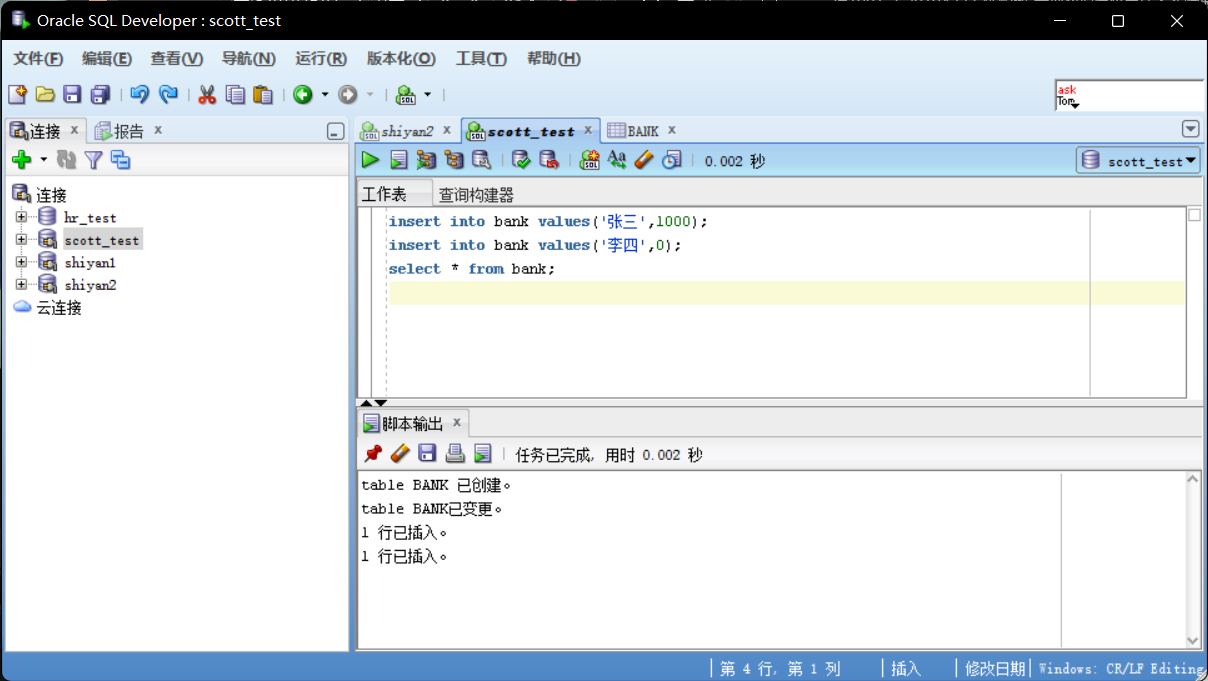


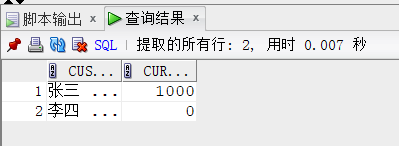
1. 插入数据：张三余额 1000，李四余额 0，提交，查看表中数据。

insert into bank values('张三',1000);

insert into bank values('李四',0);

select \* from bank;

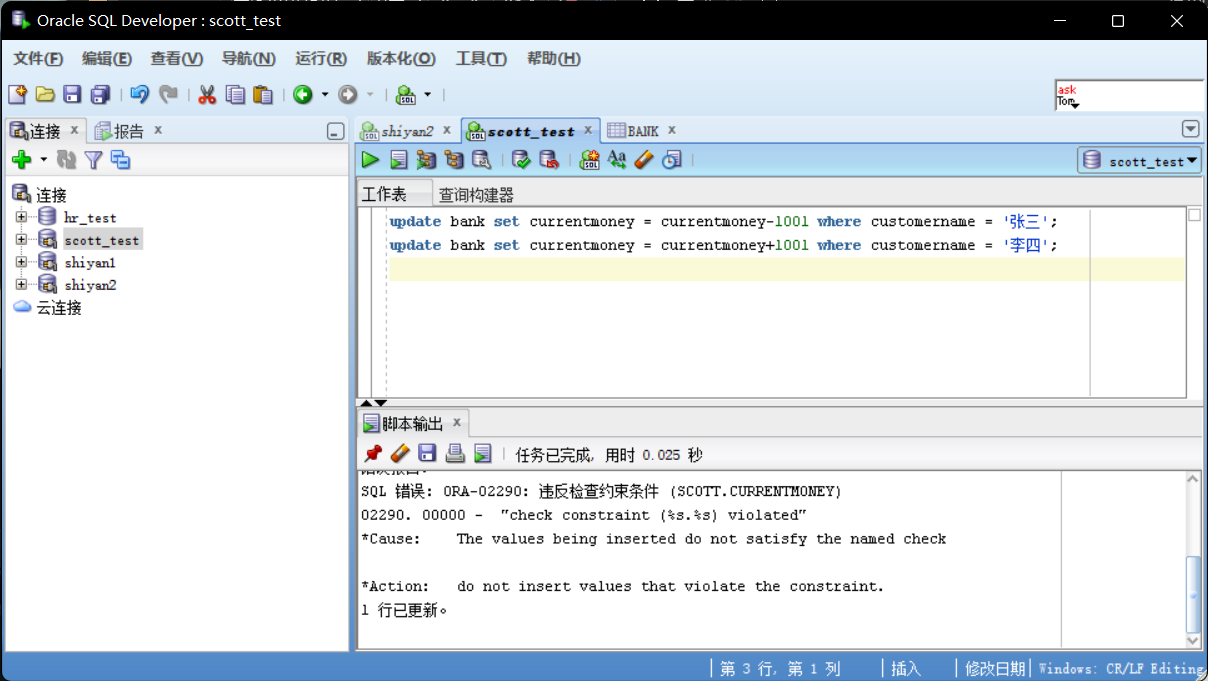




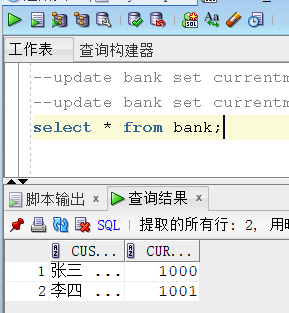
1. 张三给李四转 1001 元。

update bank set currentmoney = currentmoney-1001 where customername = '张三';

update bank set currentmoney = currentmoney+1001 where customername = '李四';



5)查看表中数据，发现什么情况？为什么出现，如何解决？



情况：张三的余额没有变更，而李四的余额增加成功，钱变多了

出现原因：余额列有check约束

解决办法：增加一个异常处理部分，当更改之后余额为负数时不执行更改，并给出提示

6) 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+1000 where customerName='李四';

update bank set currentMoney=currentMoney-1000 where customerName='张三';

exception

when others then***请补充***

end;

完善后：

--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

--给出提示

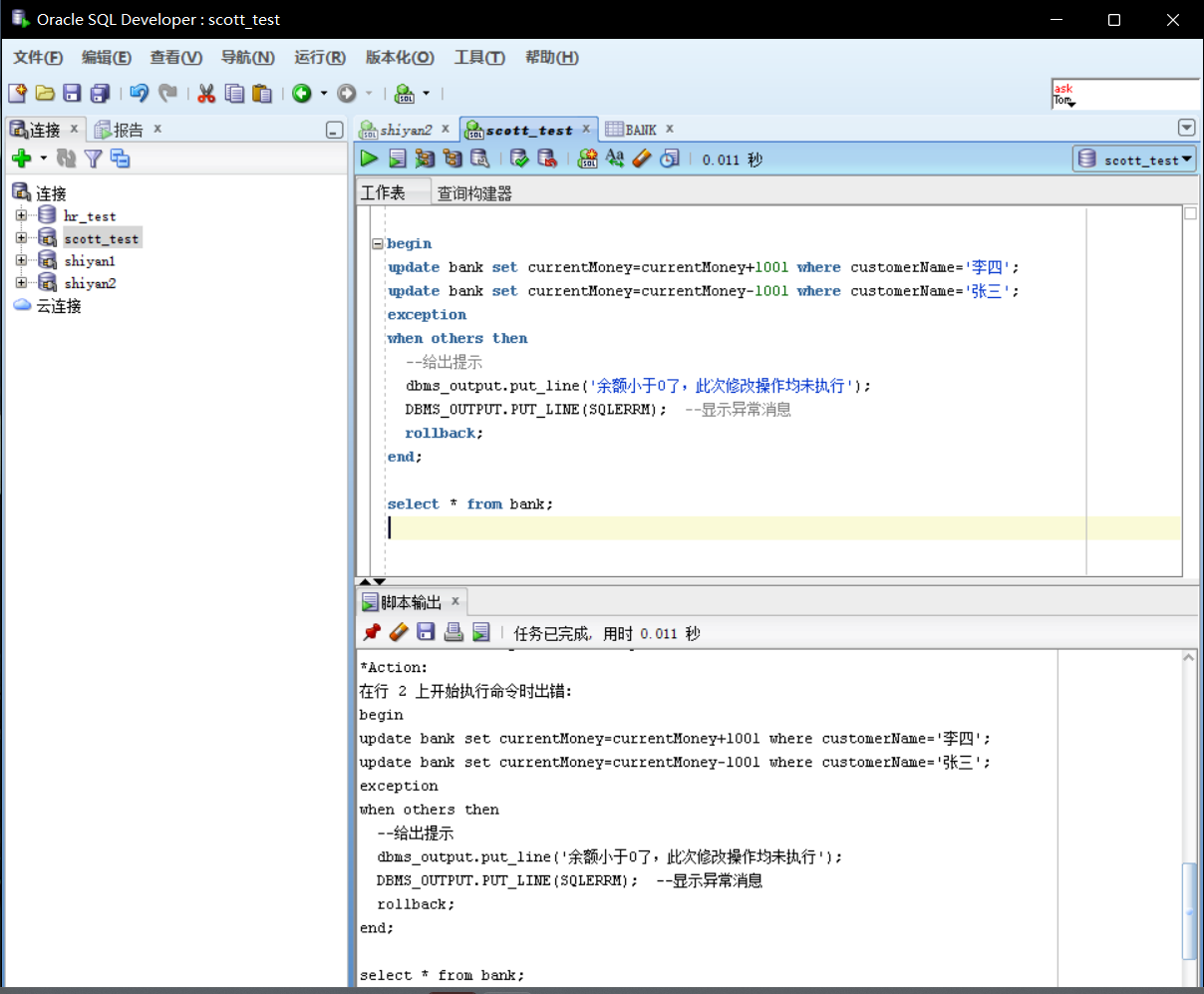
dbms\_output.put\_line('余额小于0了，此次修改操作均未执行');

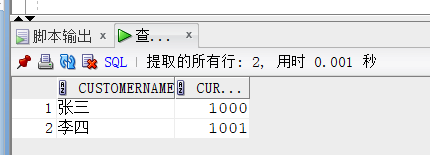
DBMS\_OUTPUT.PUT\_LINE(SQLERRM); --显示异常消息

rollback;

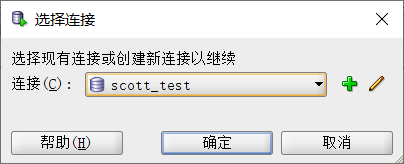
end;

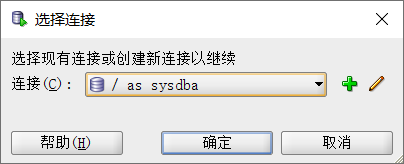
select \* from bank;



修改后再次查询：  


1. **演示 Oracle 锁机制**
2. 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。





1. 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

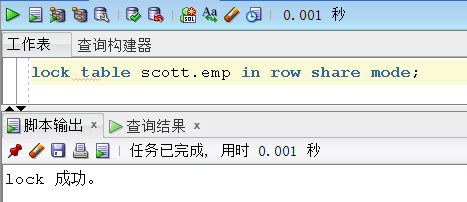
产生情况：会话1可以给表加上行共享表锁，而会话2不能给表加上排他锁。

解释：当给表加上共享锁之后，仅允许其他用户查询表中的行，禁止其他用户插入、更新和删除行，也就排斥了其他事件给表加排他锁。

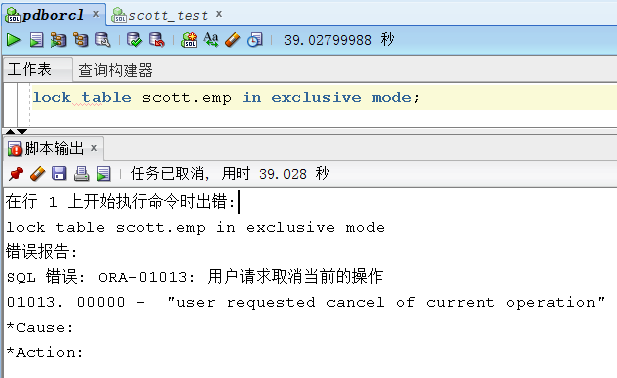
--使用LOCK TABLE语句为scott的emp表加行共享表锁

lock table emp in row share mode;

lock table scott.emp in exclusive mode



lock table scott.emp in exclusive mode



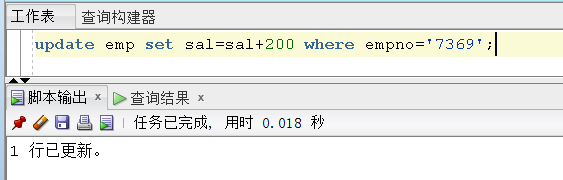
1. 会话 1 为 7369 号员工增加 200 薪水，会话 2 中尝试删除 7369 号员工对应的记录，将产生什么情况，如何解释。

产生情况：会话1操作成功，会话2失败。

解释：会话1给表加上了共享锁，禁止其他用户插入、更新和删除行，会话2没有权限删除会画1数据。

--为7369号员工增加200薪水

update emp set sal=sal+200 where empno='7369';



delete from scott.emp where empno='7369';

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称 | 大型数据库技术应用 | 班级 | 信1901-3 |
| 姓名 | 崔金泽 | 学号 | 20192163 |
| 实验日期 | 2021-11-13 | 评分 |  |

# 实验四: Oracle 安全管理、备份与恢复

一、预习任务

当前对数据库的主要安全威胁有物理威胁和逻辑威胁。物理威胁主要是像各种外力，如恐怖袭击，火灾等造成的数据库服务器故障或数据库中存储介质的损坏造成的数据丢失；逻辑威胁主要是指对信息的未授权存取，如恶意用户侵入某银行数据库系统窃取信用卡数据信息。对数据库安全物理威胁的主要解决方案主要是数据备份与恢复等技术；对逻辑威胁的主要解决方案主要是用户管理、权限管理、角色管理、概要文件管理等技术。从广义上讲，数据库系统的安全框架可以划分为三个层次：系统安全性。在系统级别控制数据库的存取和使用机制，包括有效的用户名与口令、是否可以连接数据库、可以进行哪些系统级操作等。

数据安全性。在模式对象级别控制数据库的存取和使用机制。用户要对某个模式对象进行操作，必须要有操作的权限。网络安全性。Oracle 通过分发 Wallet、数字证书、SSL 安全套接字和数据密钥等办法来保证数据库的网络传输安全性。

Oracle 数据库从以下几个方面进行安全管理：用户账户和认证方式管理。权限和角色管理：限制用户对数据库的访问和操作。数据加密管理：保证网络传输的安全性。表空间设置和配额：控制用户对数据库存储空间的使用。用户资源限制：通过概要文件设置，可以限制用户对数据库资源的使用。数据库审计：监视和记录数据库中的活动。管理员用户SYS：数据库中拥有最高权限的管理员，可以启动、关闭、修改数据库，拥有数据字典。SYSTEM：一个辅助的数据库管理员，不能启动和关闭数据库，但是可以进行一些管理工作，如创建和删除用户。SYSMAN：OEM 的管理员，可以对 OEM 进行配置和管理。DBSNMP：OEM 代理，用来监视数据库。

在 Oracle 数据库中，用户权限主要分为系统权限与对象权限两类。系统权限是指在数据库基本执行某些操作的权限，或针对某一类对象进行操作的权限。对象权限是指对某个特定模式对象的操作权限。数据库模式对象所有者拥有该对象的所有对象权限，对象权限的管理实际上是对象所有者对其它用户操作该对象的权限管理。有的对象并没有对象权限，只能通过系统权限进行管理，如簇、索引、触发器、数据库链接等。

备份(Backup)和恢复(Recovery)是两个互相联系的概念，备份就是将数据库中的数据保存起来，而恢复就是当意外发生或者其他某种需要时，将已备份的数据还原到数据库中。

Oracle数据库12c提供了多种备份方法，每种方法都有自己的特点，需要根据具体的应用状况来选择合适的备份方法。Oracle设计备份策略的指导思想是：以最小的代价恢复数据。由于备份与恢复是密切联系的，备份策略应与恢复结合起来考虑。备份从不同的角度有不同的分类。 Oracle 运行的时候至少需要两组联机日志，每当一组日志写满后会发生日志切换，继续向下一组联机日志写入。归档模式：触发 ARCn 进程，把写满的重做日志文件复制到归档日志文件。非归档模式：重做日志直接被覆盖。非归档模式只能进行脱机备份，备份过程中数据库不能使用；必须备份整个数据库，不能备份部分数据库，只能部分恢复，如果数据文件丢失，只能恢复最后一次的完全备份，而之后的所有数据库改变将全部丢失。

二、实验步骤

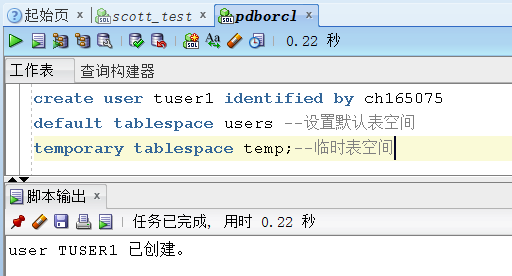
1. **完成下列用安全管理相关操作。**

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 ch165075

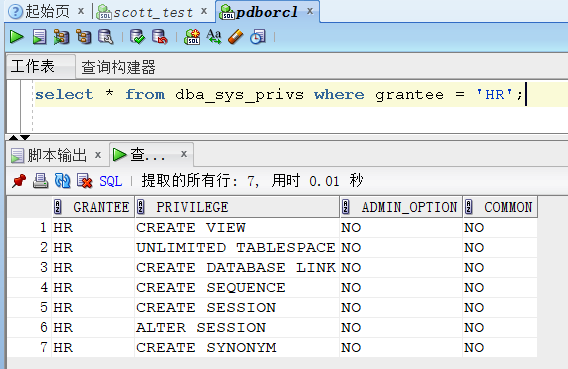
default tablespace users --设置默认表空间

temporary tablespace temp;--临时表空间

2) 查看 tuser1 拥有的系统权限和角色，与 HR 用户做对比。

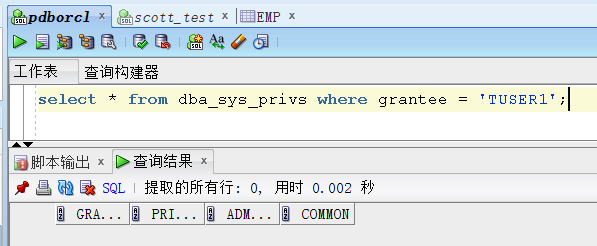
--查看HR所拥有的系统权限

select \* from dba\_sys\_privs where grantee = 'HR';

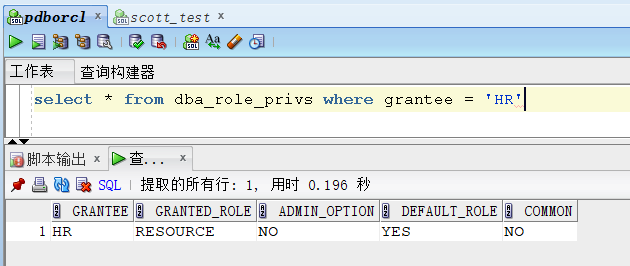


--查看tuser1所拥有的系统权限

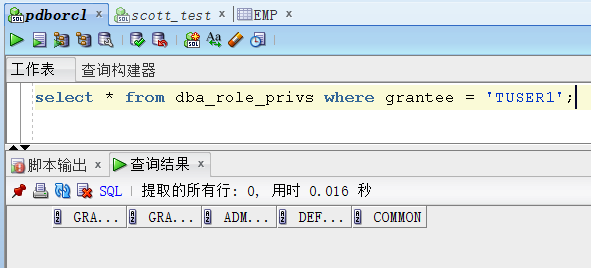
select \* from dba\_sys\_privs where grantee = 'TUSER1';



select \* from dba\_role\_privs where grantee = 'HR';



select \* from dba\_role\_privs where grantee = 'TUSER1'

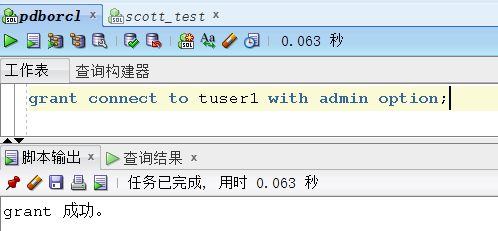


3) 尝试使用 tuser1 用户连接 PDBORCL 是否能成功？如不成功请予以

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。





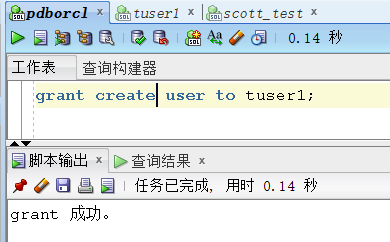


4) 为 tuser1 用户赋予创建用户的系统权限，使用 tuser1 用户创建 tuser11

用户，使 tuser11 可以连接 PDBORCL。

为tuser1用户赋予创建用户的系统权限:

grant creat user to tuser1;

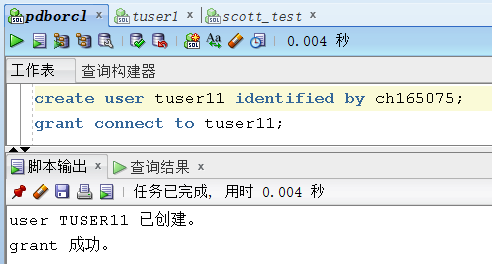


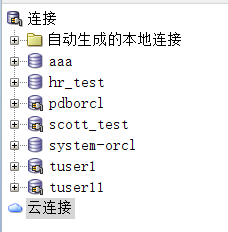
--使用tuser1用户创建tuser11用户

create user tuser11 identified by 001027;

--授予权限，使tuser11可以连接PDBORCL

grant connect to tuser11;

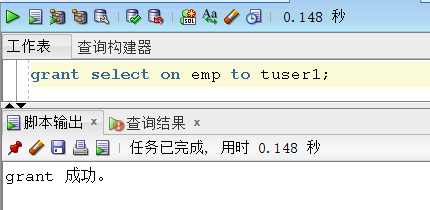




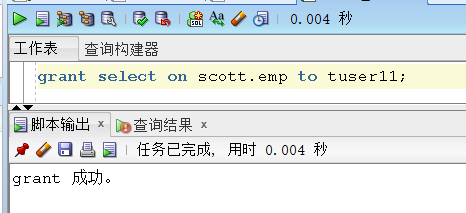
5) SCOTT 用户为 tuser1 用户赋予查看 emp 表的权限，tuser1 将此权限

传递给 tuser11，验证传递成功。

grant select on emp to tuser1;

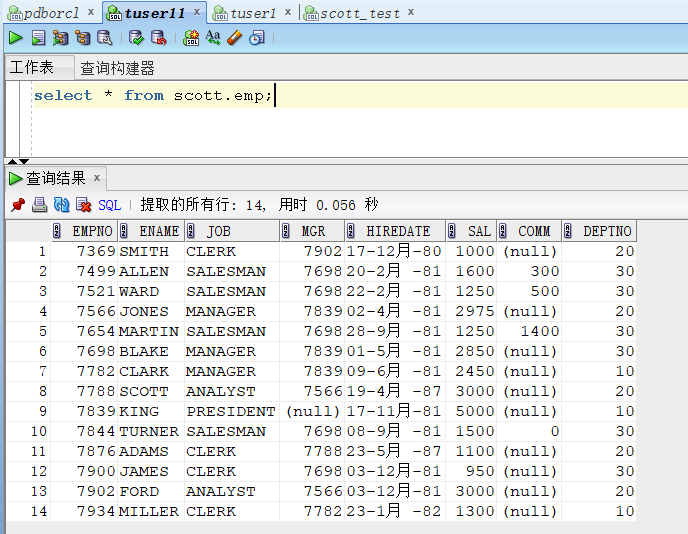


grant select on scott.emp to tuser11;



验证权限传递成功:

select \* from scott.emp;

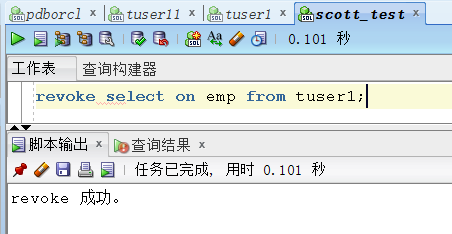


6) SCOTT 用户收回 tuser1 用户查看 emp 表的权限，验证 tuser11 用户

的权限是否被收回。

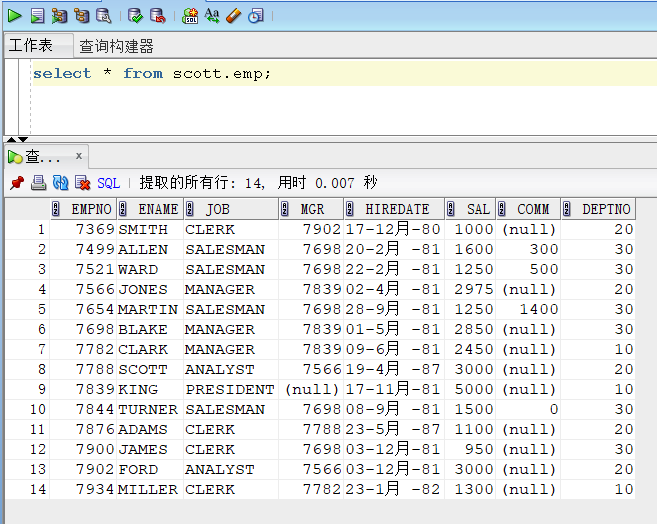
收回tuser1用户查看emp表的权限:

revoke select on emp from tuser1;



验证权限未被收回:

select \* from scott.emp;



**2. 使用 RMAN 完成备份与恢复操作。**

1) 切换数据库为归档模式。

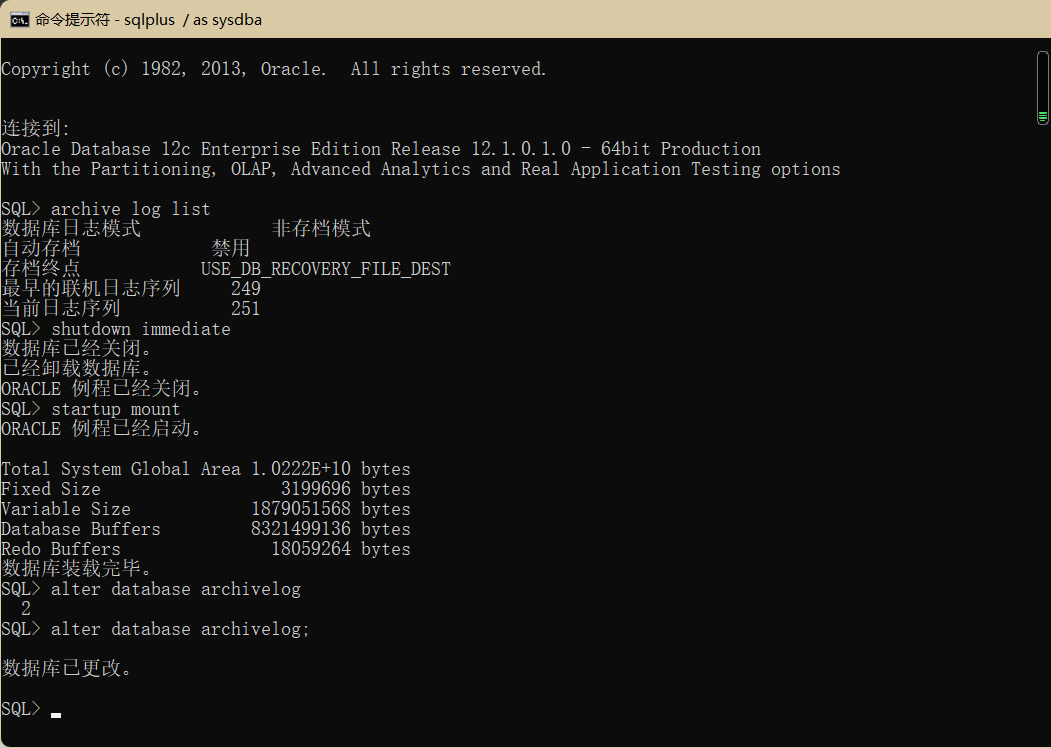
sqlplus / as sysdba --sys用户登录数据库

archive log list; --查看当前模式，默认为非归档模式

shutdown immediate; --关闭数据库

startup mount; --启动数据库到mount模式

alter database archivelog; --修改数据库为归档模式



三、心得体会

orcl相比上个学期学习的sqlserver区别不大，可能是数据量不够大的原因，但是orcl有很多优势，比如SQL Server 只能在windows 上运行，没有丝毫的开放性，操作系统的系统的稳定对数据库是十分重要的。Windows10系列产品是偏重于桌面应用，NT server只适合中小型企业。而且windows平台的可靠性，安全性和伸缩性是非常有限的。它不像unix那样久经考验，尤其是在处理大数据量的关键业务时。

Oracle 能在所有主流平台上运行（包括 windows）。完全支持所有的工业标准。采用完全开放策略。可以使客户选择最适合的解决方案。对开发商全力支持。SQL Server 多用户时性能不佳 Oracle 性能高， 保持windowsNT下的TPC-D和TPC-C的世界记录。

SQL Server C/S结构，只支持windows客户，可以用ADO,DAO,OLEDB ,ODBC连接。 Oracle 多层次网络计算，支持多种工业标准，可以用ODBC, JDBC,OCI等网络客户连接。

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称 | 大型数据库技术应用 | 班级 | 信1901-3 |
| 姓名 | 崔金泽 | 学号 | 20192163 |
| 实验日期 | 2021-11-29 | 评分 |  |

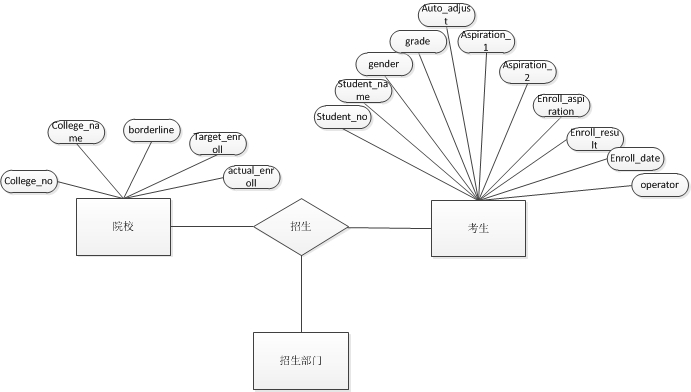
# 实验五 高校招生系统说明书

## 一 需求分析：系统开发背景分析

### 背景

高等院校的招生录取工作，是一个复杂工作。涉及到大量考生和院校的数据，人工进行录取工作量大、主观性强、容易滋生腐败行为。通过开发“招生录取系统”，实现录取过程的自动化等工作。大大节约录取时间、使录取更加公正、客观。系统分析与设计：重点为数据库表结构设计，各表的描述等。

### 实体关系



### 表结构设计

高校(高校代码,高校名称,录取分数线,预计录取数)

学生(准考证号,学生姓名,性别,高考成绩)

志愿(学生准考证号,志愿优先级,院校代码)

调剂意愿(学生准考证号,是否调剂)

-- 0未录取,-1调剂录取,1第一志愿录取,2第二志愿录取

录取结果(学生准考证号,录取结果)

-- -1退档,非-1院校代码

调剂结果(学生准考证号,调剂结果)

## 二 系统实现

主要包含数据库用户的创建、创建表的脚本、存储过程及触发器实现脚本等。

### 2.1创建用户并授权

|  |
| --- |
| CREATE USER yjy\_homework IDENTIFIED BY 111111 default tablespace MY\_TABLESPACE; |
| GRANT CONNECT,RESOURCE TO yjy\_homework;  GRANT CREATE TABLE,CREATE ANY INDEX,CREATE VIEW,CREATE SEQUENCE,CREATE SYNONYM TO yjy\_homework; |
| GRANT UNLIMITED TABLESPACE TO yjy\_homework; |

### 2.2表、序列、索引

#### 表

|  |  |
| --- | --- |
| 高校表 | CREATE TABLE college (  college\_number NUMBER(4) PRIMARY KEY,  college\_name VARCHAR2(30) NOT NULL,  borderline NUMBER(3) CHECK (borderline BETWEEN 300 AND 700),  targetEnroll NUMBER(3) CHECK (targetEnroll <= 100),  actualEnroll NUMBER(3) DEFAULT 0  ) |
| 学生表 | CREATE TABLE STUDENT (  student\_number NUMBER(5) PRIMARY KEY,  student\_name VARCHAR2(15) NOT NULL,  gender NCHAR(1) CHECK (gender IN ('男', '女')),  grade NUMBER(3) CHECK (grade BETWEEN 0 AND 700)  ) |
| 学生志愿表 | CREATE TABLE student\_will(  student\_number NUMBER(5) REFERENCES STUDENT,  priority NUMBER(2),  aspiration NUMBER(4) REFERENCES COLLEGE,  PRIMARY KEY (student\_number,priority) |
| 调剂意愿表 | CREATE TABLE auto\_adjust\_will (  student\_number NUMBER(5) REFERENCES STUDENT,  auto\_adjust NUMBER(1) DEFAULT 0  ) |
| 录取结果表 | CREATE TABLE student\_enroll\_result(  student\_number NUMBER(5) PRIMARY KEY REFERENCES STUDENT,  enroll\_result NUMBER(1) -- 0未录取,-1调剂录取,1第一志愿录取,2第二志愿录取  "ENROLL\_OPERATOR" NVARCHAR2(15) DEFAULT NULL  ) |
| 调剂录取结果表 | CREATE TABLE auto\_adjust\_result(  student\_number NUMBER(5) PRIMARY KEY REFERENCES STUDENT,  college\_number NUMBER(5) REFERENCES COLLEGE  ) |

#### 序列

|  |  |
| --- | --- |
| 学生表id生成序列 | CREATE SEQUENCE student\_sequence START WITH 1 |
| 高校表id生成序列 | CREATE SEQUENCE college\_sequence START WITH 1 |

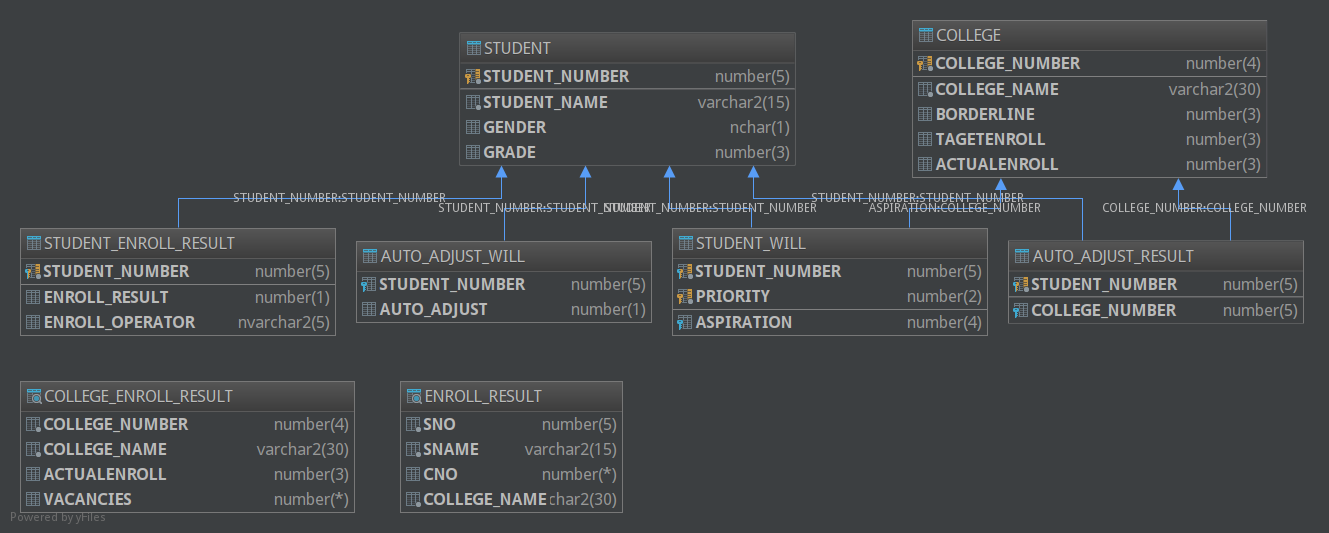
#### 索引

暂不需要。

### 2.3视图

|  |  |
| --- | --- |
| 考生录取结果(考生编号，姓名，院校代码，院校名称)  （包括志愿录取、调剂录取和退档的所有考生） | CREATE OR REPLACE VIEW ENROLL\_RESULT AS  SELECT  sno,  sname,  cno,  COLLEGE\_NAME  FROM (  SELECT  STUDENT\_NUMBER sno,  STUDENT\_NAME sname,  FIND\_ENROLL\_COLLEGE(STUDENT\_NUMBER) cno  FROM STUDENT  ) my\_table, COLLEGE  WHERE my\_table.cno = COLLEGE\_NUMBER; |
| 录取情况(院校编号，院校名称，招生人数，缺额) | CREATE OR REPLACE VIEW COLLEGE\_ENROLL\_RESULT AS  (  SELECT  COLLEGE\_NUMBER,  COLLEGE\_NAME,  ACTUALENROLL,  TARGETENROLL - ACTUALENROLL Vacancies  FROM (SELECT  COLLEGE\_NUMBER,  COLLEGE\_NAME,  TARGETENROLL,  COUNT\_ACTUAL\_ENROLL(COLLEGE\_NUMBER) ACTUALENROLL  FROM COLLEGE)  ) |
| 志愿视图(学生编号,志愿1,志愿1学校名称,志愿2,志愿2学校名称,是否调剂) | CREATE OR REPLACE VIEW ALL\_WILL AS  SELECT  STUDENT\_NUMBER,  ASPIRATION1,  FIND\_COLLEGE\_NAME(ASPIRATION1) CNAME1,  ASPIRATION2,  FIND\_COLLEGE\_NAME(ASPIRATION2) CNAME2,  AUTO\_ADJUST  FROM (  SELECT  STUDENT\_NUMBER,  GET\_ASPIRATION(STUDENT\_NUMBER, 1) ASPIRATION1,  GET\_ASPIRATION(STUDENT\_NUMBER, 2) ASPIRATION2,  AUTO\_ADJUST  FROM AUTO\_ADJUST\_WILL  ) |

表、视图间关系如图：



### 2.4初始化数据

数据已导出为.sql文件，放置在src/main/doc/testdata目录下。

### 2.5函数、存储过程、触发器、pl/sql

#### （1）函数

|  |  |
| --- | --- |
| 根据学生编号找出考入的高校编号，退档则为null | create FUNCTION find\_Enroll\_College(sno NUMBER) RETURN NUMBER  IS  college\_number NUMBER(5);  v\_enroll\_result NUMBER(1);  BEGIN  SELECT ENROLL\_RESULT INTO v\_enroll\_result FROM STUDENT\_ENROLL\_RESULT WHERE STUDENT\_NUMBER=sno;  IF (v\_enroll\_result=0) THEN -- 退档  SELECT NULL INTO college\_number FROM dual;  ELSIF (v\_enroll\_result=-1) THEN -- 调剂录取  SELECT COLLEGE\_NUMBER INTO college\_number FROM AUTO\_ADJUST\_RESULT WHERE AUTO\_ADJUST\_RESULT.STUDENT\_NUMBER=sno;  ELSIF (v\_enroll\_result=1) THEN -- 第一志愿录取  SELECT ASPIRATION INTO college\_number FROM STUDENT\_WILL WHERE STUDENT\_WILL.STUDENT\_NUMBER=sno AND STUDENT\_WILL.PRIORITY=1;  ELSIF (v\_enroll\_result=2) THEN -- 第二志愿录取  SELECT ASPIRATION INTO college\_number FROM STUDENT\_WILL WHERE STUDENT\_WILL.STUDENT\_NUMBER=sno AND STUDENT\_WILL.PRIORITY=2;  ELSE -- 异常  SELECT NULL INTO college\_number FROM dual;  END IF;  RETURN college\_number;  END; |
| 计算高校的实际招生人数 | CREATE FUNCTION count\_actual\_enroll(p\_college\_number NUMBER)  RETURN NUMBER  IS  enroll\_count NUMBER;  BEGIN  SELECT count(\*)  INTO enroll\_count  FROM ENROLL\_RESULT  WHERE ENROLL\_RESULT.CNO = p\_college\_number;  RETURN enroll\_count;  END; |
| 根据学生编号和志愿号得出志愿高校代码 | CREATE OR REPLACE FUNCTION GET\_ASPIRATION(sno IN NUMBER,ano IN NUMBER) RETURN NUMBER  AS  college NUMBER;  BEGIN  SELECT ASPIRATION INTO college FROM STUDENT\_WILL WHERE STUDENT\_NUMBER=sno AND PRIORITY=ano;  RETURN college;  END; |
| 根据高校编号选出高校名称 | CREATE OR REPLACE FUNCTION FIND\_COLLEGE\_NAME(cno IN NUMBER) RETURN NVARCHAR2  AS  cname NVARCHAR2;  BEGIN  SELECT COLLEGE\_NAME INTO cname FROM COLLEGE WHERE COLLEGE\_NUMBER=cno;  RETURN cname;  END; |
| 判断某个学生是否已被录取 | create OR REPLACE FUNCTION IS\_STUDENT\_ENROLLED(sno IN NUMBER)  RETURN NUMBER  AS  is\_enrolled NUMBER(1);  BEGIN  SELECT count(\*)  INTO is\_enrolled  FROM STUDENT\_ENROLL\_RESULT  WHERE STUDENT\_NUMBER = sno AND ENROLL\_RESULT IS NOT NULL AND ENROLL\_RESULT != 0;  RETURN is\_enrolled;  END; |

#### （2）存储过程

|  |  |
| --- | --- |
| 清空录取结果 | create OR REPLACE PROCEDURE clear\_status  AS  BEGIN  DELETE STUDENT\_ENROLL\_RESULT;  DELETE AUTO\_ADJUST\_RESULT;  COMMIT ;  END; |
| 投档(高校编号,志愿编号) | CREATE OR REPLACE PROCEDURE ENROLL\_PROC(P\_YXBH IN NUMBER, WILL\_NUMBER IN NUMBER)  AS  V\_ZSRS NUMBER(3); -- 招生人数  V\_LQFSX NUMBER(3); -- 录取分数线  V\_YXMC VARCHAR2(30); -- 院校名称  V\_COUNT NUMBER(3);  BEGIN  SELECT  --取院校信息  TARGETENROLL,  BORDERLINE,  COLLEGE\_NAME  INTO V\_ZSRS, V\_LQFSX, V\_YXMC  FROM COLLEGE  WHERE COLLEGE\_NUMBER = P\_YXBH;  V\_COUNT := 0;  -- 志愿为高校P\_YXBH,且成绩高于分数线,且尚未被录取的学生  FOR STU\_REC IN (SELECT STUDENT.STUDENT\_NUMBER  FROM STUDENT, STUDENT\_WILL  WHERE IS\_STUDENT\_ENROLLED(STUDENT.STUDENT\_NUMBER)=0  AND PRIORITY = WILL\_NUMBER  AND ASPIRATION = P\_YXBH  AND STUDENT.STUDENT\_NUMBER = STUDENT\_WILL.STUDENT\_NUMBER  AND GRADE >= V\_LQFSX  ORDER BY GRADE DESC) LOOP  EXIT WHEN V\_COUNT >= V\_ZSRS;  -- 插入录取结果  INSERT INTO STUDENT\_ENROLL\_RESULT VALUES  (STU\_REC.STUDENT\_NUMBER, WILL\_NUMBER, USER);  V\_COUNT := V\_COUNT + 1;  END LOOP;  COMMIT;  END; |
| 一键自动投档 | CREATE OR REPLACE PROCEDURE AUTO\_PROC  AS  v\_max\_priority NUMBER(2);  v\_min\_priority NUMBER(2);  BEGIN  FOR COLLEGE\_REC IN (SELECT COLLEGE\_NUMBER  FROM COLLEGE) LOOP  SELECT  max(PRIORITY),  min(PRIORITY)  INTO v\_max\_priority, v\_min\_priority  FROM STUDENT\_WILL;  -- 第n志愿录取  FOR n IN v\_min\_priority..v\_max\_priority LOOP  ENROLL\_PROC(COLLEGE\_REC.COLLEGE\_NUMBER, n);  END LOOP;  END LOOP;  END; |
|  | -- 尚未被录取，且同意调剂的学生  CURSOR adjustable\_student IS  SELECT STUDENT\_NUMBER  FROM AUTO\_ADJUST\_WILL  GROUP BY STUDENT\_NUMBER  HAVING IS\_STUDENT\_ENROLLED(STUDENT\_NUMBER) = 0;  -- 调剂录取  FOR v\_student IN adjustable\_student LOOP  SELECT \*  INTO v\_college\_number  FROM COLLEGE, COLLEGE\_ENROLL\_RESULT, STUDENT  WHERE VACANCIES > 0 AND COLLEGE.COLLEGE\_NUMBER = COLLEGE\_ENROLL\_RESULT.COLLEGE\_NUMBER AND  STUDENT\_NUMBER = v\_student.STUDENT\_NUMBER AND GRADE > BORDERLINE;  END LOOP;  V\_COUNT := V\_COUNT + 1;  COMMIT; |

#### （3）触发器

|  |  |  |
| --- | --- | --- |
| 已删除，采用视图提取高校实际招生人数，而不直接将数据保存在基表。因为oracle没有计算列，触发器又因为读锁而无法计算。 | 变动志愿录取结果表时自动更新学校实际录取人数（也可以将学校实际录取人数做成计算列） | CREATE OR REPLACE TRIGGER UPDATE\_COLLEGE\_ENROLL  AFTER INSERT OR UPDATE OR DELETE  ON STUDENT\_ENROLL\_RESULT  BEGIN  UPDATE COLLEGE  SET ACTUALENROLL = COUNT\_ACTUAL\_ENROLL(COLLEGE\_NUMBER);  END; |
| 变动调剂录取结果表时自动更新学校实际录取人数（也可以将学校实际录取人数做成计算列） | CREATE OR REPLACE TRIGGER UPDATE\_COLLEGE\_ENROLL\_ADJUST  AFTER INSERT OR UPDATE OR DELETE  ON AUTO\_ADJUST\_RESULT  BEGIN  UPDATE COLLEGE  SET ACTUALENROLL = COUNT\_ACTUAL\_ENROLL(COLLEGE\_NUMBER);  END; |

#### pl/sql

|  |  |
| --- | --- |
| 创建测试数据 | DECLARE  sno NUMBER(5);  gender NVARCHAR2(1);  BEGIN  sno := STUDENT\_SEQUENCE.nextval;  IF (mod(sno,2) = 1)  THEN  gender := '男';  ELSE  gender := '女';  END IF;  INSERT INTO STUDENT VALUES (sno, '学生' || sno, gender, 500 + sno);  Commit；  END; |

### 2.6权限与安全

暂无。

### 2.7性能分析

|  |  |
| --- | --- |
| 查询 | 视图的定义、以及所有的查询语句，其中用到的查询连接都是基于主键的连接。按照当前的表结构设计，即使不建索引查询速度也不会慢多少。 |
| 并发 | 学生和高校信息录入功能暂不开放，因此不考虑这两者相关的并发情况；一键录取功能是基于ENROLL\_PROC存储过程实现的，经过测试，该存储过程能完全保障数据一致性。 |

## 三 心得体会

①dml触发器中不能调用读该表的函数，因为此时表发生了变化, 触发器/函数不能读它。

②数据库设计时不要考虑过多扩展性，这会导致编码难度大大增加（增删改都要专门编写一个存储过程）。按照ORM的思路建表最合适，最终结果大概符合2NF或3NF，但是存取数据的sql可以很简单地编写，也和ORM框架有很好的契合。

运行结果：

