

数据库系统课程上机实验手册

华为 OpenGauss 版

大连理工大学

Dalian University of Technology

目录

| | |
|-------------------------|----|
| 第一章 实验介绍..... | 3 |
| 1.1 关于本实验..... | 3 |
| 1.2 实验目的..... | 3 |
| 1.3 内容描述..... | 3 |
| 第二章 DDL..... | 4 |
| 2.1 预备知识..... | 4 |
| 2.2 实验任务..... | 4 |
| 2.3 SQL 代码与对应结果..... | 4 |
| 第三章 DML..... | 6 |
| 3.1 预备知识..... | 6 |
| 3.2 实验任务..... | 6 |
| 第四章 数据查询..... | 13 |
| 4.1 单表查询..... | 13 |
| 4.2 聚合查询..... | 16 |
| 4.3 多表查询..... | 19 |
| 4.4 子查询（必须使用子查询实现）..... | 24 |
| 4.5 集合查询..... | 28 |
| 第五章 索引操作..... | 32 |
| 5.1 预备知识..... | 32 |
| 5.2 实验任务..... | 33 |
| 第六章 事务的并发控制..... | 37 |
| 6.1 预备知识..... | 37 |
| 6.2 实验任务..... | 37 |
| 第七章 自我总结..... | 45 |

第一章 实验介绍

1.1 关于本实验

本实验共包含 5 个子实验，从数据准备开始，逐一介绍了 scoot 数据库中的 SQL 语法，包括数据查询、数据更新、数据定义和数据控制。

1.2 实验目的

学习 SQL 相关操作, 了解 OpenGauss 系统对索引和事务并发控制的实现。

1.3 内容描述

子实验 2 为数据定义实验，介绍了 DDL 的类型、语法格式和使用场景，帮助读者熟练掌握如何用数据定义语言定义或修改数据库中的对象。

子实验 3 为数据更新实验，通过对 DML 语言基本语法和使用，帮助读者掌握如何对数据库表中数据进行更新操作，包括数据插入、数据修改和数据删除。

子实验 4 为数据查询实验，通过基本的 DQL 语言使用，帮助读者掌握从一个或多个表查询数据的操作。

子实验 5 为索引操作，通过基本的索引使用，帮助读者掌握索引的创建和使用。

子实验 6 为事务并发控制，通过对事务并发场景的设计，帮助读者了解 OpenGauss 数据库关系对并发的支持情况。

第二章 DDL

2.1 预备知识

数据库模式定义语言 DDL (Data Definition Language)，是用于描述数据库中要存储的现实世界实体的语言，用来创建数据库中的各种对象——表、视图、索引、同义词、聚簇等

```
create table 表名(  
    字段名 1 类型[(宽度) 约束条件],  
    字段名 2 类型[(宽度) 约束条件],  
    字段名 3 类型[(宽度) 约束条件]  
);  
#类型：使用限制字段必须以什么样的数据类型传值  
#约束条件：约束条件是在类型之外添加一种额外的限制
```

2.2 实验任务

创建 DEPT、BONUS、SALGRADE、EMP 表，关系模式为：

```
DEPT (DEPTNO INT, DNAME VARCHAR(14), LOC VARCHAR(13));  
EMP (EMPNO INT, ENAME VARCHAR(10), JOB VARCHAR(9), MGR INT, HIREDATE DATE, SAL  
    FLOAT, COMM FLOAT, DEPTNO INT);  
BONUS (ENAME VARCHAR(10), JOB VARCHAR(9), SAL INT, COMM INT);  
SALGRADE ( GRADE INT, LOSAL INT, HISAL INT);  
注意：EMP 表中的 DEPTNO 属性为外键，对应 DEPT 表中的 DEPTNO
```

2.3 SQL 代码与对应结果

请输入如下代码，验证执行结果是否与答案相符。

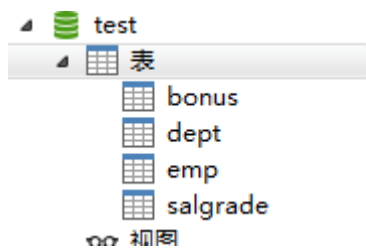
```
CREATE TABLE DEPT (  
    DEPTNO      INT,  
    DNAME       VARCHAR(14),  
    LOC         VARCHAR(13),  
    CONSTRAINT PK_DEPT PRIMARY KEY (DEPTNO)  
);
```

```
CREATE TABLE BONUS (  
  ENAME      VARCHAR(10),  
  JOB        VARCHAR(9),  
  SAL        INT,  
  COMM       INT  
);
```

```
CREATE TABLE SALGRADE (  
  GRADE      INT,  
  LOSAL      INT,  
  HISAL      INT  
);
```

```
CREATE TABLE EMP (  
  EMPNO      INT,  
  ENAME      VARCHAR(10),  
  JOB        VARCHAR(9),  
  MGR        INT,  
  HIREDATE   DATETIME,  
  SAL        FLOAT,  
  COMM       FLOAT,  
  DEPTNO     INT,  
  CONSTRAINT PK_EMP PRIMARY KEY (EMPNO),  
  CONSTRAINT FK_DEPTNO FOREIGN KEY (DEPTNO) REFERENCES DEPT(DEPTNO)  
);
```

答案:



第三章 DML

3.1 预备知识

DML 是 Data Manipulation Language 的缩写，意思是数据操纵语言，是指在 SQL 语言中，负责对数据库对象运行数据访问工作的指令集，以 INSERT、UPDATE、DELETE 三种指令为核心，分别代表插入、更新与删除，是开发以数据为中心的应用程序必定会使用到的指令。

1 增

```
INSERT INTO tb_name (col_name1, col_name2) VALUES ('val1', 'val1'), ('val2', 'val2');
```

2 删

```
DELETE FROM table_name [WHERE];
```

3 改

```
UPDATE tb_name SET col_name1 = 'new_val1', col_name2 = 'new_val2' [WHERE];
```

3.2 实验任务

3.2.1 实践题目 1

在 DEPT 表中插入数据 10, 'ACCOUNTING', 'NEW YORK'。

请写出满足查询结果的查询代码。

结果：

| DEPTNO | DNAME | LOC |
|--------|------------|----------|
| 10 | ACCOUNTING | NEW YORK |

代码：

```
insert into dept values(10, 'ACCOUNTING', 'NEW YORK')
select * from dept
```

| Contains | | Enter a search term here. | |
|----------|--------|---------------------------|----------|
| | deptno | dname | loc |
| 1 | 10 | ACCOUNTING | NEW YORK |

3.2.2 实践题目 2

在 DEPT 表中根据 DEPTNO 修改 LOC 为 BEIJING。

请写出满足查询结果的查询代码。

结果：

| DEPTNO | DNAME | LOC |
|--------|------------|---------|
| 10 | ACCOUNTING | BEIJING |

代码：

```
UPDATE dept SET loc = 'BEIJING' where deptno = 10;  
select * from dept
```

| | deptno | dname | loc |
|---|--------|------------|---------|
| 1 | 20 | RESEARCH | DALLAS |
| 2 | 30 | SALES | CHICAGO |
| 3 | 40 | OPERATIONS | BOSTON |
| 4 | 10 | ACCOUNTING | BEIJING |

3.2.3 实践题目 3

在 DEPT 表中删除 DEPTNO 为 10 的数据,。

请写出满足查询结果的查询代码。

结果：

| DEPTNO | DNAME | LOC |
|--------|--------|--------|
| (Null) | (Null) | (Null) |

代码：

```
/DELETE FROM dept where deptno=10;  
select * from dept
```

| Contains Enter a search term here. | | | |
|------------------------------------|--------|------------|---------|
| | deptno | dname | loc |
| 1 | 20 | RESEARCH | DALLAS |
| 2 | 30 | SALES | CHICAGO |
| 3 | 40 | OPERATIONS | BOSTON |

3.2.4 实践题目 4

在 DEPT 表中插入两条数据 10, 'ACCOUNTING', 'NEW YORK' 和 20, 'RESEARCH', 'DALLAS'。

请写出满足查询结果的查询代码。

结果：

| DEPTNO | DNAME | LOC |
|--------|------------|----------|
| 10 | ACCOUNTING | NEW YORK |
| 20 | RESEARCH | DALLAS |

代码：

```
INSERT INTO DEPT VALUES (20, 'RESEARCH', 'DELLAS');  
insert into dept values (10, 'ACCOUNTING', 'NEW YORK');
```

```
3  
4 INSERT INTO DEPT VALUES (20, 'RESEARCH', 'DELLAS');  
5 insert into dept values (10, 'ACCOUNTING', 'NEW YORK');  
6 select * from dept
```

| Messages | | Result | |
|------------------------------------|--------|------------|----------|
| Contains Enter a search term here. | | | |
| | deptno | dname | loc |
| 1 | 30 | SALES | CHICAGO |
| 2 | 40 | OPERATIONS | BOSTON |
| 3 | 20 | RESEARCH | DELLAS |
| 4 | 10 | ACCOUNTING | NEW YORK |

3.2.5 实践题目 5

删除 DEPT 表中所有数据。

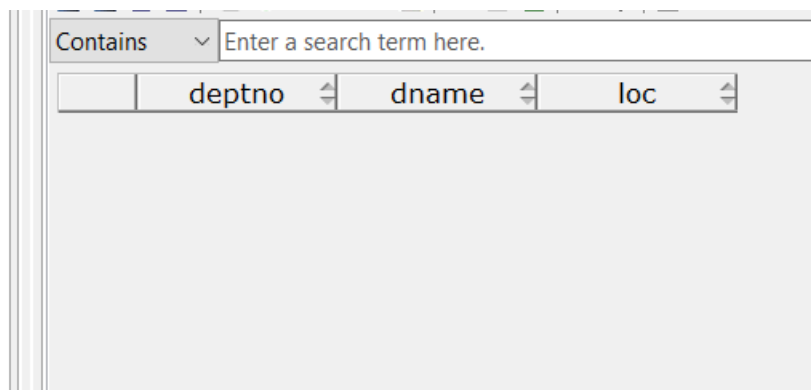
请写出满足查询结果的查询代码。

结果：

| DEPTNO | DNAME | LOC |
|--------|--------|--------|
| (Null) | (Null) | (Null) |

代码：

```
delete from dept;
```



3.2.6 实践题目 6

执行如下代码，并验证代码执行结果是否与图片给出的结果相同。

代码：

```
Delete from DEPT;  
INSERT INTO DEPT VALUES  
(10, 'ACCOUNTING', 'NEW YORK');  
INSERT INTO DEPT VALUES  
(20, 'RESEARCH', 'DALLAS');  
INSERT INTO DEPT VALUES  
(30, 'SALES', 'CHICAGO');  
INSERT INTO DEPT VALUES  
(40, 'OPERATIONS', 'BOSTON');
```

```
Delete from EMP;  
INSERT INTO EMP VALUES  
(7369, 'SMITH', 'CLERK', 7566, '1980-12-17', 800, NULL, 20);  
INSERT INTO EMP VALUES
```

```

(7499, 'ALLEN', 'SALESMAN', 7698, '1981-02-20', 1600, 300, 30);
INSERT INTO EMP VALUES
(7521, 'WARD', 'SALESMAN', 7698, '1981-02-22', 1250, 500, 30);
INSERT INTO EMP VALUES
(7566, 'JONES', 'MANAGER', 7839, '1981-04-02', 2975, NULL, 20);
INSERT INTO EMP VALUES
(7654, 'MARTIN', 'SALESMAN', 7698, '1981-09-28', 1250, 1400, 30);
INSERT INTO EMP VALUES
(7698, 'BLAKE', 'MANAGER', 7839, '1981-05-01', 2850, NULL, 30);
INSERT INTO EMP VALUES
(7782, 'CLARK', 'MANAGER', 7839, '1981-06-09', 2450, NULL, 10);
INSERT INTO EMP VALUES
(7788, 'SCOTT', 'ANALYST', 7566, '1987-06-13', 3000, NULL, 20);
INSERT INTO EMP VALUES
(7839, 'KING', 'PRESIDENT', NULL, '1981-11-17', 5000, NULL, 10);
INSERT INTO EMP VALUES
(7844, 'TURN...', 'SALESMAN', 7698, '1981-09-08', 1500, 0, 30);
INSERT INTO EMP VALUES
(7876, 'ADAMS', 'CLERK', 7788, '1987-06-13', 1100, NULL, 20);
INSERT INTO EMP VALUES
(7900, 'JAMES', 'CLERK', 7698, '1981-12-03', 950, NULL, 30);
INSERT INTO EMP VALUES
(7934, 'MILLER', 'CLERK', 7782, '1982-01-23', 1300, NULL, 10);

```

```

Delete from SALGRADE;
INSERT INTO SALGRADE VALUES
(1, 700, 1200);
INSERT INTO SALGRADE VALUES
(2, 1201, 1400);
INSERT INTO SALGRADE VALUES
(3, 1401, 2000);
INSERT INTO SALGRADE VALUES
(4, 2001, 3000);
INSERT INTO SALGRADE VALUES
(5, 3001, 9999);

```

结果:

DEPT表:

| DEPTNO | DNAME | LOC |
|--------|------------|----------|
| 10 | ACCOUNTING | NEW YORK |
| 20 | RESEARCH | DALLAS |
| 30 | SALES | CHICAGO |
| 40 | OPERATIONS | BOSTON |

```

50
51 select * from dept;
52 select * from emp;
53 select * from salgrade;
54

```

Messages Result

Contains Enter a search term here.

| | deptno | dname | loc |
|---|--------|------------|----------|
| 1 | 10 | ACCOUNTING | NEW YORK |
| 2 | 20 | RESEARCH | DALLAS |
| 3 | 30 | SALES | CHICAGO |
| 4 | 40 | OPERATIONS | BOSTON |

EMP表:

| | EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|----|-------|---------|-----------|------|-------------------------|------|------|--------|
| 1 | 7369 | SMITH | CLERK | 7566 | 1980-12-17 00:00:00.000 | 800 | NULL | 20 |
| 2 | 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 00:00:00.000 | 1600 | 300 | 30 |
| 3 | 7521 | WARD | SALESMAN | 7698 | 1981-02-22 00:00:00.000 | 1250 | 500 | 30 |
| 4 | 7566 | JONES | MANAGER | 7839 | 1981-04-02 00:00:00.000 | 2975 | NULL | 20 |
| 5 | 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 00:00:00.000 | 1250 | 1400 | 30 |
| 6 | 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 00:00:00.000 | 2850 | NULL | 30 |
| 7 | 7782 | CLARK | MANAGER | 7839 | 1981-06-09 00:00:00.000 | 2450 | NULL | 10 |
| 8 | 7788 | SCOTT | ANALYST | 7566 | 1987-06-13 00:00:00.000 | 3000 | NULL | 20 |
| 9 | 7839 | KING | PRESIDENT | NULL | 1981-11-17 00:00:00.000 | 5000 | NULL | 10 |
| 10 | 7844 | TURN... | SALESMAN | 7698 | 1981-09-08 00:00:00.000 | 1500 | 0 | 30 |
| 11 | 7876 | ADAMS | CLERK | 7788 | 1987-06-13 00:00:00.000 | 1100 | NULL | 20 |
| 12 | 7900 | JAMES | CLERK | 7698 | 1981-12-03 00:00:00.000 | 950 | NULL | 30 |
| 13 | 7934 | MILLER | CLERK | 7782 | 1982-01-23 00:00:00.000 | 1300 | NULL | 10 |

```

52 select * from emp;
53 select * from salgrade;
54

```

Elapsed Time : 528 ms

Messages Result

Contains Enter a search term here.

| | empno | ename | job | mgr | hiredate | sal | comm | deptno |
|---|-------|--------|-----------|--------|---------------------|--------|--------|--------|
| 1 | 7369 | SMITH | CLERK | 7566 | 1980-12-17 00:00:00 | 800.0 | [NULL] | 20 |
| 2 | 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 00:00:00 | 1600.0 | 300.0 | 30 |
| 3 | 7521 | WARD | SALESMAN | 7698 | 1981-02-22 00:00:00 | 1250.0 | 500.0 | 30 |
| 4 | 7566 | JONES | MANAGER | 7839 | 1981-04-02 00:00:00 | 2975.0 | [NULL] | 20 |
| 5 | 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 00:00:00 | 1250.0 | 1400.0 | 30 |
| 6 | 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 00:00:00 | 2850.0 | [NULL] | 30 |
| 7 | 7782 | CLARK | MANAGER | 7839 | 1981-06-09 00:00:00 | 2450.0 | [NULL] | 10 |
| 8 | 7788 | SCOTT | ANALYST | 7566 | 1987-06-13 00:00:00 | 3000.0 | [NULL] | 20 |
| 9 | 7839 | KING | PRESIDENT | [NULL] | 1981-11-17 00:00:00 | 5000.0 | [NULL] | 10 |

Query Submit Time: 2021-06-18 14:47:42.449 CST|13 rows fetched in 226 ms

Query Results can be edited.

SALGRADE表:

| GRADE | LOSAL | HISAL |
|-------|-------|-------|
| 1 | 700 | 1200 |
| 2 | 1201 | 1400 |
| 3 | 1401 | 2000 |
| 4 | 2001 | 3000 |
| 5 | 3001 | 9999 |

```
52 select * from emp;
53 select * from salgrade;
54
```

Messages Result

Contains Enter a search term here.

| | grade | losal | hisal |
|---|-------|-------|-------|
| 1 | 1 | 700 | 1200 |
| 2 | 2 | 1201 | 1400 |
| 3 | 3 | 1401 | 2000 |
| 4 | 4 | 2001 | 3000 |
| 5 | 5 | 3001 | 9999 |

BONUS表: 无数据

```
52 select * from emp;
53 select * from salgrade;
54 select * from bonus;
55
```

Messages Result

Contains Enter a search term here.

| | ename | job | sal | comm |
|--|-------|-----|-----|------|
|--|-------|-----|-----|------|

第四章 数据查询

对于每个查询需求，给出对应的查询语句，并验证是否与给出的结果相同。将查询语句写在“代码：”下面，具体行数可以根据语句行数自行调节。

4.1 单表查询

4.1.1 预备知识

单表查询是最简单的查询方式。所有要查询的信息，都集中在一张表中。也就是说，SQL 语句中的 FROM 子句中只有一个表。我们可以通过以下的几道实践题目来巩固这个知识点。

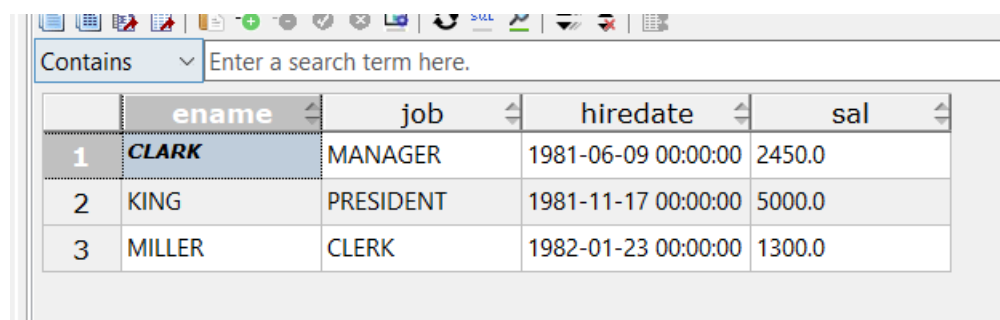
4.1.2 实践题目 1

查看 EMP 表中部门号为 10 的员工的姓名，职位，参加工作时间，工资。
结果：

| | ename | job | hiredate | sal |
|---|--------|-----------|-------------------------|------|
| 1 | CLARK | MANAGER | 1981-06-09 00:00:00.000 | 2450 |
| 2 | KING | PRESIDENT | 1981-11-17 00:00:00.000 | 5000 |
| 3 | MILLER | CLERK | 1982-01-23 00:00:00.000 | 1300 |

代码：

```
select ename ,job,hiredate,sal
from emp
where deptno = 10
```



| | ename | job | hiredate | sal |
|---|--------|-----------|---------------------|--------|
| 1 | CLARK | MANAGER | 1981-06-09 00:00:00 | 2450.0 |
| 2 | KING | PRESIDENT | 1981-11-17 00:00:00 | 5000.0 |
| 3 | MILLER | CLERK | 1982-01-23 00:00:00 | 1300.0 |

4.1.3 实践题目 2

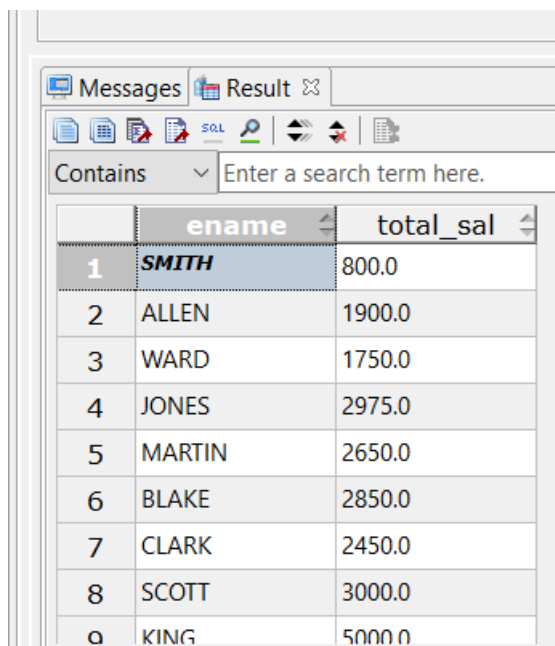
查询每个员工每个月拿到的总金额(emp.sal 为工资,emp.comm 为补助)。(提示:gaussdb

中，nvl (ex1,ex2) 表示如果 ex1 为空则返回 ex2)
结果：

| | ename | total |
|----|--------|-------|
| 1 | SMITH | 800 |
| 2 | ALLEN | 1900 |
| 3 | WARD | 1750 |
| 4 | JONES | 2975 |
| 5 | MARTIN | 2650 |
| 6 | BLAKE | 2850 |
| 7 | CLARK | 2450 |
| 8 | SCOTT | 3000 |
| 9 | KING | 5000 |
| 10 | TURNER | 1500 |
| 11 | ADAMS | 1100 |
| 12 | JAMES | 950 |
| 13 | MILLER | 1300 |

代码：

```
select  ename, nvl(sal,0)+nvl(comm,0) as total_sal
from emp
```



| | ename | total_sal |
|---|--------|-----------|
| 1 | SMITH | 800.0 |
| 2 | ALLEN | 1900.0 |
| 3 | WARD | 1750.0 |
| 4 | JONES | 2975.0 |
| 5 | MARTIN | 2650.0 |
| 6 | BLAKE | 2850.0 |
| 7 | CLARK | 2450.0 |
| 8 | SCOTT | 3000.0 |
| 9 | KING | 5000.0 |

4.1.4 实践题目 3

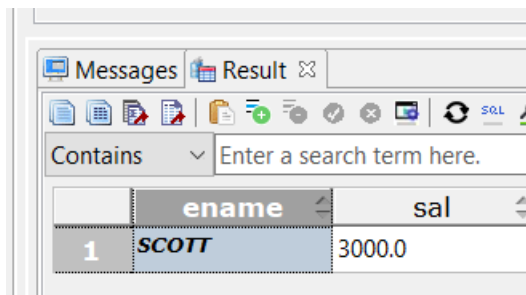
显示第 3 个字符为大写 O 的所有员工的姓名及工资。

结果：

| | ename | sal |
|---|-------|------|
| 1 | SCOTT | 3000 |

代码：

```
select ename,sal
from emp
where ename like '__O%'
```



| | ename | sal |
|---|-------|--------|
| 1 | SCOTT | 3000.0 |

4.1.5 实践题目 4

显示有补助的员工的姓名，工资，补助。

结果：

| | ename | sal | comm |
|---|--------|------|------|
| 1 | ALLEN | 1600 | 300 |
| 2 | WARD | 1250 | 500 |
| 3 | MARTIN | 1250 | 1400 |
| 4 | TURNER | 1500 | 0 |

代码：

```
select ename,sal,comm
from emp
where comm is not null
```

| | ename | sal | comm |
|---|---------|--------|--------|
| 1 | ALLEN | 1600.0 | 300.0 |
| 2 | WARD | 1250.0 | 500.0 |
| 3 | MARTIN | 1250.0 | 1400.0 |
| 4 | TURN... | 1500.0 | 0.0 |

4.1.6 实践题目 5

显示员工的最高工资和最低工资。

结果：

| | 最高工资 | 最低工资 |
|---|------|------|
| 1 | 5000 | 800 |

代码：

```
select max(sal) as 最高工资 ,min(sal) as 最低工资
from emp
```

| | 最高工资 | 最低工资 |
|---|--------|-------|
| 1 | 5000.0 | 800.0 |

4.2 聚合查询

4.2.1 预备知识

在查询中，我们经常会遇到这样的问题：求平均值、求最值等等。我们需要使用一些函数如 AVG(), MAX() 等进行计算，也需要通过 GROUP BY 子句来聚合属性。

4.2.2 实践题目 1

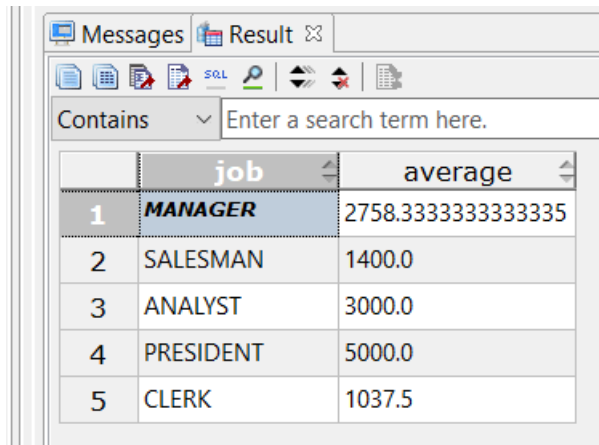
显示每种职业的平均工资。

结果：

| | job | average |
|---|-----------|-------------------|
| 1 | ANALYST | 3000 |
| 2 | CLERK | 1037.5 |
| 3 | MANAGER | 2758.333333333333 |
| 4 | PRESIDENT | 5000 |
| 5 | SALESMAN | 1400 |

代码：

```
select job, avg(sal) as average
from emp
group by job
```



The screenshot shows a database query result window with a toolbar and a search bar. The result is a table with two columns: 'job' and 'average'. The data is as follows:

| | job | average |
|---|-----------|--------------------|
| 1 | MANAGER | 2758.3333333333335 |
| 2 | SALESMAN | 1400.0 |
| 3 | ANALYST | 3000.0 |
| 4 | PRESIDENT | 5000.0 |
| 5 | CLERK | 1037.5 |

4.2.3 实践题目 2

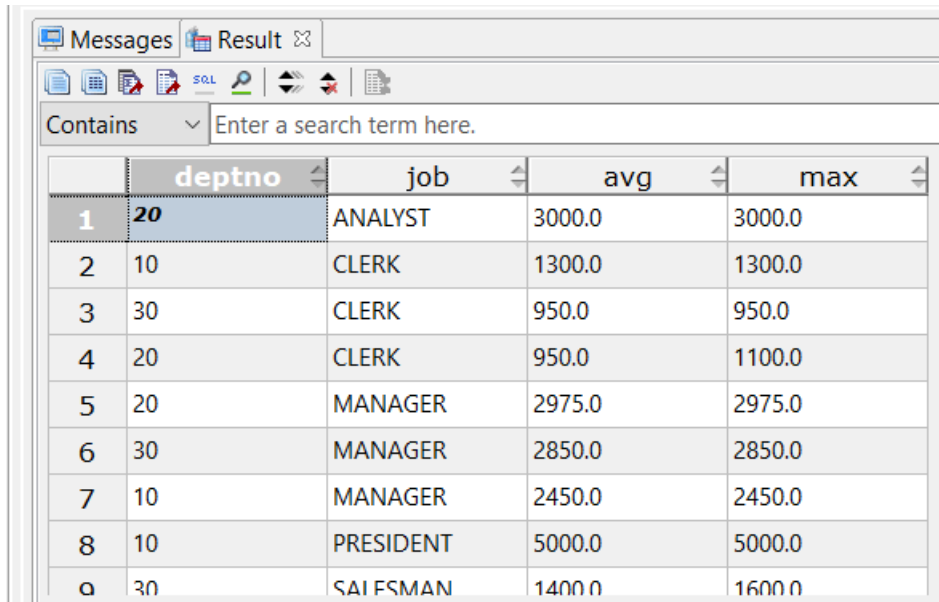
显示每个部门每种岗位的平均工资和最高工资。

结果：

| | deptno | job | average | max |
|---|--------|-----------|---------|------|
| 1 | 20 | ANALYST | 3000 | 3000 |
| 2 | 10 | CLERK | 1300 | 1300 |
| 3 | 20 | CLERK | 950 | 1100 |
| 4 | 30 | CLERK | 950 | 950 |
| 5 | 10 | MANAGER | 2450 | 2450 |
| 6 | 20 | MANAGER | 2975 | 2975 |
| 7 | 30 | MANAGER | 2850 | 2850 |
| 8 | 10 | PRESIDENT | 5000 | 5000 |
| 9 | 30 | SALESMAN | 1400 | 1600 |

代码:

```
select deptno,job,avg(sal) as avg ,max(sal)
from emp
group by job,deptno
order by job asc ,avg desc
```



| | deptno | job | avg | max |
|---|--------|-----------|--------|--------|
| 1 | 20 | ANALYST | 3000.0 | 3000.0 |
| 2 | 10 | CLERK | 1300.0 | 1300.0 |
| 3 | 30 | CLERK | 950.0 | 950.0 |
| 4 | 20 | CLERK | 950.0 | 1100.0 |
| 5 | 20 | MANAGER | 2975.0 | 2975.0 |
| 6 | 30 | MANAGER | 2850.0 | 2850.0 |
| 7 | 10 | MANAGER | 2450.0 | 2450.0 |
| 8 | 10 | PRESIDENT | 5000.0 | 5000.0 |
| 9 | 30 | SALESMAN | 1400.0 | 1600.0 |

4.2.4 实践题目 3

显示平均工资低于 2500 的部门号，平均工资及最高工资。

结果:

| | deptno | average | max |
|---|--------|--------------------|------|
| 1 | 20 | 1968.75 | 3000 |
| 2 | 30 | 1566.6666666666667 | 2850 |

代码:

```
select deptno,avg(sal) as avg,max(sal)
from emp
group by deptno
having avg<2500
```

| Messages Result | | | |
|------------------------------------|--------|--------------------|--------|
| Contains Enter a search term here. | | | |
| | deptno | avg | max |
| 1 | 20 | 1968.75 | 3000.0 |
| 2 | 30 | 1566.6666666666667 | 2850.0 |

4.3 多表查询

4.3.1 预备知识

在大部分情况下，我们所需要的信息并不仅仅包含在一张表中。我们首先需要使用 join 连接多个表，然后再进行查询

4.3.2 实践题目 1

显示工资高于 2500 或岗位为 MANAGER 的所有员工的姓名，工资，职位，和部门号。
结果：

| | ename | sal | job | deptno |
|---|-------|------|-----------|--------|
| 1 | JONES | 2975 | MANAGER | 20 |
| 2 | BLAKE | 2850 | MANAGER | 30 |
| 3 | CLARK | 2450 | MANAGER | 10 |
| 4 | SCOTT | 3000 | ANALYST | 20 |
| 5 | KING | 5000 | PRESIDENT | 10 |

代码：

```
select ename,sal,job,emp.deptno
from emp join dept on dept.deptno = emp.deptno
where sal>2500 or job = 'MANAGER'
```

| Messages Result | | | | |
|------------------------------------|-------|--------|-----------|--------|
| Contains Enter a search term here. | | | | |
| | ename | sal | job | deptno |
| 1 | JONES | 2975.0 | MANAGER | 20 |
| 2 | BLAKE | 2850.0 | MANAGER | 30 |
| 3 | CLARK | 2450.0 | MANAGER | 10 |
| 4 | SCOTT | 3000.0 | ANALYST | 20 |
| 5 | KING | 5000.0 | PRESIDENT | 10 |

4.3.3 实践题目 2

排序显示所有员工的姓名，部门号，工资（以部门号升序，工资降序，雇用日期升序显示）。

结果：

| | ename | deptno | sal |
|----|--------|--------|------|
| 1 | KING | 10 | 5000 |
| 2 | CLARK | 10 | 2450 |
| 3 | MILLER | 10 | 1300 |
| 4 | SCOTT | 20 | 3000 |
| 5 | JONES | 20 | 2975 |
| 6 | ADAMS | 20 | 1100 |
| 7 | SMITH | 20 | 800 |
| 8 | BLAKE | 30 | 2850 |
| 9 | ALLEN | 30 | 1600 |
| 10 | TURNER | 30 | 1500 |
| 11 | WARD | 30 | 1250 |
| 12 | MARTIN | 30 | 1250 |
| 13 | JAMES | 30 | 950 |

代码：

```
select ename,emp.deptno,sal
from dept join emp on dept.deptno = emp.deptno
order by dept.deptno asc,sal desc,hiredate asc
```

| | ename | deptno | sal |
|---|--------|--------|--------|
| 1 | KING | 10 | 5000.0 |
| 2 | CLARK | 10 | 2450.0 |
| 3 | MILLER | 10 | 1300.0 |
| 4 | SCOTT | 20 | 3000.0 |
| 5 | JONES | 20 | 2975.0 |
| 6 | ADAMS | 20 | 1100.0 |
| 7 | SMITH | 20 | 800.0 |
| 8 | BLAKE | 30 | 2850.0 |
| 9 | ALLEN | 30 | 1600.0 |

4.3.4 实践题目 3

采用自然连接的原理显示部门名以及相应的员工姓名。（Sql server 不支持 NATURAL JOIN 语法。）

结果：

| | dname | ename |
|----|------------|--------|
| 1 | RESEARCH | SMITH |
| 2 | SALES | ALLEN |
| 3 | SALES | WARD |
| 4 | RESEARCH | JONES |
| 5 | SALES | MARTIN |
| 6 | SALES | BLAKE |
| 7 | ACCOUNTING | CLARK |
| 8 | RESEARCH | SCOTT |
| 9 | ACCOUNTING | KING |
| 10 | SALES | TURNER |
| 11 | RESEARCH | ADAMS |
| 12 | SALES | JAMES |
| 13 | ACCOUNTING | MILLER |

代码：

Part1: 使用 natural join

```
select dept.dname, emp.ename
from dept natural join emp
```

| Messages Result | | |
|------------------------------------|------------|--------|
| Contains Enter a search term here. | | |
| | dname | ename |
| 1 | RESEARCH | SMITH |
| 2 | SALES | ALLEN |
| 3 | SALES | WARD |
| 4 | RESEARCH | JONES |
| 5 | SALES | MARTIN |
| 6 | SALES | BLAKE |
| 7 | ACCOUNTING | CLARK |
| 8 | RESEARCH | SCOTT |
| 9 | ACCOUNTING | KING |

Query Submit Time: 2021-06-18 16:34:38.689 CST

Part2: 不用 natural, 只用原理

```
select dept.dname, emp.ename
from dept join emp using(deptno)
```

| Messages Result | | |
|------------------------------------|------------|--------|
| Contains Enter a search term here. | | |
| | dname | ename |
| 1 | RESEARCH | SMITH |
| 2 | SALES | ALLEN |
| 3 | SALES | WARD |
| 4 | RESEARCH | JONES |
| 5 | SALES | MARTIN |
| 6 | SALES | BLAKE |
| 7 | ACCOUNTING | CLARK |
| 8 | RESEARCH | SCOTT |
| 9 | ACCOUNTING | KING |

Query Submit Time: 2021-06-18 16:36:01.967 CST

4.3.5 实践题目 4

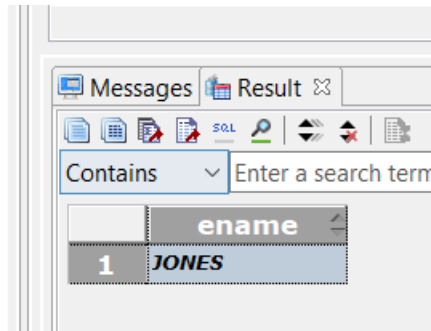
查询 SCOTT 的上级领导的姓名。

结果：

| | ename |
|---|-------|
| 1 | JONES |

代码:

```
select m.ename
from emp e join emp m on e.mgr = m.empno
where e.ename = 'SCOTT'
```



4.3.6 实践题目 5

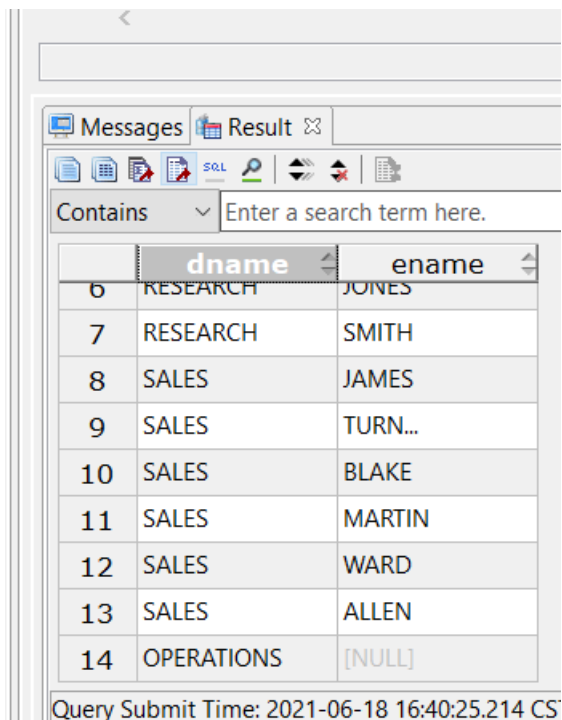
显示部门的部门名称，员工名即使部门没有员工也显示部门名称。

结果:

| | dname | ename |
|----|------------|--------|
| 1 | ACCOUNTING | CLARK |
| 2 | ACCOUNTING | KING |
| 3 | ACCOUNTING | MILLER |
| 4 | RESEARCH | SMITH |
| 5 | RESEARCH | JONES |
| 6 | RESEARCH | SCOTT |
| 7 | RESEARCH | ADAMS |
| 8 | SALES | ALLEN |
| 9 | SALES | WARD |
| 10 | SALES | MARTIN |
| 11 | SALES | BLAKE |
| 12 | SALES | TURNER |
| 13 | SALES | JAMES |
| 14 | OPERATIONS | NULL |

代码:

```
select dept.dname,ename
from dept left join emp on dept.deptno = emp.deptno
```



| | dname | ename |
|----|------------|---------|
| 6 | RESEARCH | JONES |
| 7 | RESEARCH | SMITH |
| 8 | SALES | JAMES |
| 9 | SALES | TURN... |
| 10 | SALES | BLAKE |
| 11 | SALES | MARTIN |
| 12 | SALES | WARD |
| 13 | SALES | ALLEN |
| 14 | OPERATIONS | [NULL] |

Query Submit Time: 2021-06-18 16:40:25.214 CS

4.4 子查询（必须使用子查询实现）

4.4.1 预备知识

子查询就是指的在一个完整的查询语句之中，嵌套若干个不同功能的小查询，从而一起完成复杂查询的一种编写形式。

4.4.2 实践题目 1

显示所有员工的名称、工资以及工资级别。

结果：

| | ename | sal | grade |
|----|--------|------|-------|
| 1 | SMITH | 800 | 1 |
| 2 | ADAMS | 1100 | 1 |
| 3 | JAMES | 950 | 1 |
| 4 | WARD | 1250 | 2 |
| 5 | MARTIN | 1250 | 2 |
| 6 | MILLER | 1300 | 2 |
| 7 | ALLEN | 1600 | 3 |
| 8 | TURNER | 1500 | 3 |
| 9 | JONES | 2975 | 4 |
| 10 | BLAKE | 2850 | 4 |
| 11 | CLARK | 2450 | 4 |
| 12 | SCOTT | 3000 | 4 |
| 13 | KING | 5000 | 5 |

代码：

```
select ename, sal, grade
from (select * from salgrade) join emp on sal >= losal and sal <= hisal
```

| | ename | sal | grade |
|---|---------|--------|-------|
| 1 | SMITH | 800.0 | 1 |
| 2 | ADAMS | 1100.0 | 1 |
| 3 | JAMES | 950.0 | 1 |
| 4 | WARD | 1250.0 | 2 |
| 5 | MARTIN | 1250.0 | 2 |
| 6 | MILLER | 1300.0 | 2 |
| 7 | ALLEN | 1600.0 | 3 |
| 8 | TURN... | 1500.0 | 3 |
| 9 | JONES | 2975.0 | 4 |

4.4.3 实践题目 2

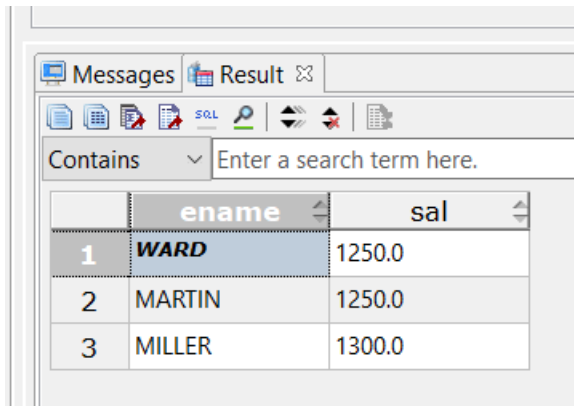
显示所有工资等级为 2 的员工的姓名，工资。（分别给出相关子查询和不相关子查询的查询语句）

结果：

| | ENAME | SAL |
|---|--------|------|
| 1 | WARD | 1250 |
| 2 | MARTIN | 1250 |
| 3 | MILLER | 1300 |

代码:

```
select ename, sal
--from salgrade join emp on sal>=losal and sal <= hisal
from (select ename,sal,grade from salgrade join emp on sal>=losal and
sal <= hisal)
where grade = 2
```



| | ename | sal |
|---|--------|--------|
| 1 | WARD | 1250.0 |
| 2 | MARTIN | 1250.0 |
| 3 | MILLER | 1300.0 |

4.4.4 实践题目 3

显示职位属于 10 号部门所提供职位范围的员工的姓名，职位，工资，部门号。

结果:

| | ename | job | sal | deptno |
|---|--------|-----------|------|--------|
| 1 | SMITH | CLERK | 800 | 20 |
| 2 | JONES | MANAGER | 2975 | 20 |
| 3 | BLAKE | MANAGER | 2850 | 30 |
| 4 | CLARK | MANAGER | 2450 | 10 |
| 5 | KING | PRESIDENT | 5000 | 10 |
| 6 | ADAMS | CLERK | 1100 | 20 |
| 7 | JAMES | CLERK | 950 | 30 |
| 8 | MILLER | CLERK | 1300 | 10 |

代码:

```
select ename,job,sal,deptno
from (select job from emp where deptno = 10) natural join emp
```

| Messages Result | | | | |
|------------------------------------|--------|-----------|--------|--------|
| Contains Enter a search term here. | | | | |
| | ename | job | sal | deptno |
| 1 | SMITH | CLERK | 800.0 | 20 |
| 2 | JONES | MANAGER | 2975.0 | 20 |
| 3 | BLAKE | MANAGER | 2850.0 | 30 |
| 4 | CLARK | MANAGER | 2450.0 | 10 |
| 5 | KING | PRESIDENT | 5000.0 | 10 |
| 6 | ADAMS | CLERK | 1100.0 | 20 |
| 7 | JAMES | CLERK | 950.0 | 30 |
| 8 | MILLER | CLERK | 1300.0 | 10 |

4.4.5 实践题目 4

显示工资比 30 号部门中所有员工的工资都高的员工的姓名，工资和部门号。

结果：

| | ename | sal | deptno |
|---|-------|------|--------|
| 1 | JONES | 2975 | 20 |
| 2 | SCOTT | 3000 | 20 |
| 3 | KING | 5000 | 10 |

代码：

```
select ename,sal,deptno
```

```
FROM emp
```

```
where sal > (select max(sal) from emp where deptno = 30)
```

| Messages Result | | | |
|------------------------------------|-------|--------|--------|
| Contains Enter a search term here. | | | |
| | ename | sal | deptno |
| 1 | JONES | 2975.0 | 20 |
| 2 | SCOTT | 3000.0 | 20 |
| 3 | KING | 5000.0 | 10 |

4.4.6 实践题目 5

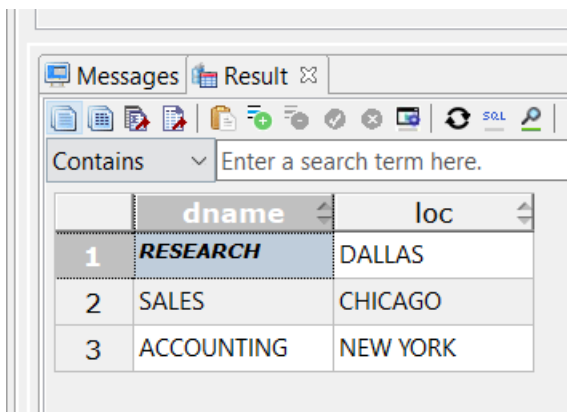
显示包含工资等级为 4 的员工部门信息，具体包括部门名称，部门所在地（使用双层嵌套子查询实现）

结果：

| | DNAME | LOC |
|---|------------|----------|
| 1 | SALES | CHICAGO |
| 2 | ACCOUNTING | NEW YORK |
| 3 | RESEARCH | DALLAS |

代码：

```
select dname, loc
FROM dept
where deptno in
(select deptno
from (select grade, deptno from emp join salgrade on sal >= losal and
sal <= hisal)
where grade = 4)
```



| | dname | loc |
|---|------------|----------|
| 1 | RESEARCH | DALLAS |
| 2 | SALES | CHICAGO |
| 3 | ACCOUNTING | NEW YORK |

4.5 集合查询

4.5.1 预备知识

当两张表的属性相同时，我们可以对它们做一些集合运算，如并集、交集等。

4.5.2 实践题目 1

显示工资高于 2500 或职位为 MANAGER 的员工的姓名，工资和职位（采用 UNION 语法实现）。

结果：

| | ename | sal | job |
|---|-------|------|-----------|
| 1 | BLAKE | 2850 | MANAGER |
| 2 | CLARK | 2450 | MANAGER |
| 3 | JONES | 2975 | MANAGER |
| 4 | KING | 5000 | PRESIDENT |
| 5 | SCOTT | 3000 | ANALYST |

代码:

```
(select ename,sal,job from emp where sal>2500)
```

UNION

```
(select ename,sal,job from emp where job = 'MANAGER')
```

| | ename | sal | job |
|---|-------|--------|-----------|
| 1 | CLARK | 2450.0 | MANAGER |
| 2 | KING | 5000.0 | PRESIDENT |
| 3 | SCOTT | 3000.0 | ANALYST |
| 4 | JONES | 2975.0 | MANAGER |
| 5 | BLAKE | 2850.0 | MANAGER |

4.5.3 实践题目 2

显示工资高于 2500 且职位为 MANAGER 的员工的姓名，工资和职位（采用 INTERSECT 语
法实现）。

结果:

| | ename | sal | job |
|---|-------|------|---------|
| 1 | BLAKE | 2850 | MANAGER |
| 2 | JONES | 2975 | MANAGER |

代码:

```
(select ename,sal,job from emp where sal>2500)
```

intersect

```
(select ename,sal,job from emp where job = 'MANAGER')
```

| | ename | sal | job |
|---|-------|--------|---------|
| 1 | JONES | 2975.0 | MANAGER |
| 2 | BLAKE | 2850.0 | MANAGER |

4.5.4 实践题目 3

显示工资高于 2500 但职位不是 MANAGER 的员工的姓名，工资和职位（采用 MINUS 语法实现）。

结果：

| | ename | sal | job |
|---|-------|------|-----------|
| 1 | KING | 5000 | PRESIDENT |
| 2 | SCOTT | 3000 | ANALYST |

代码：

```
(select ename,sal,job from emp where sal>2500)
minus
(select ename,sal,job from emp where job = 'MANAGER')
```

| | ename | sal | job |
|---|-------|--------|-----------|
| 1 | KING | 5000.0 | PRESIDENT |
| 2 | SCOTT | 3000.0 | ANALYST |

4.5.5 实践题目 4

显示提供了工资在 2000~5000 之间的所有职位的部门名称

结果：

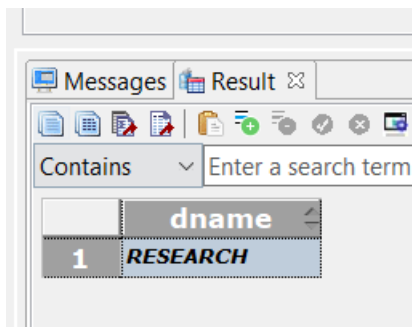
| | DNAME |
|---|----------|
| 1 | RESEARCH |

代码:

```

select dname
from dept
where not exists (
select job from emp where sal>2000 and sal<5000
MINUS
SELECT job FROM emp WHERE emp.deptno = dept.deptno)

```



| Messages | Result | | | | |
|----------|---|--|-------|---|----------|
| | <table border="1"> <thead> <tr> <th></th> <th>dname</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>RESEARCH</td> </tr> </tbody> </table> | | dname | 1 | RESEARCH |
| | dname | | | | |
| 1 | RESEARCH | | | | |

第五章 索引操作

5.1 预备知识

索引可以提高数据的访问速度，但同时也增加了插入、更新和删除表的处理时间。所以是否要为表增加索引，索引建立在哪些字段上，是创建索引前必须要考虑的问题。需要分析应用程序的业务处理、数据使用、经常被用作查询条件或者被要求排序的字段来确定是否建立索引。索引相关的 DDL 包括创建索引、删除索引属性和删除索引。

利用索引检索，需要在大数据量的情况下才能够体现出效率差距。因此，本实验需要利用 PLSQL（过程化 SQL）知识自动构建一个含有大量数据的表作为实验对象。PLSQL 相关知识请参考“HCIP-GaussDB-OLTP V1.0 培训教材”第 236 页

PLSQL

- PLSQL是一种高级数据库程序设计语言。
- 全称：
 - 过程化SQL语言 (Procedure Language & Structured Query Language)。
- 定义：
 - 一组为了完成特定功能的SQL语句的集合。

第236页 版权所有 © 2019 华为技术有限公司



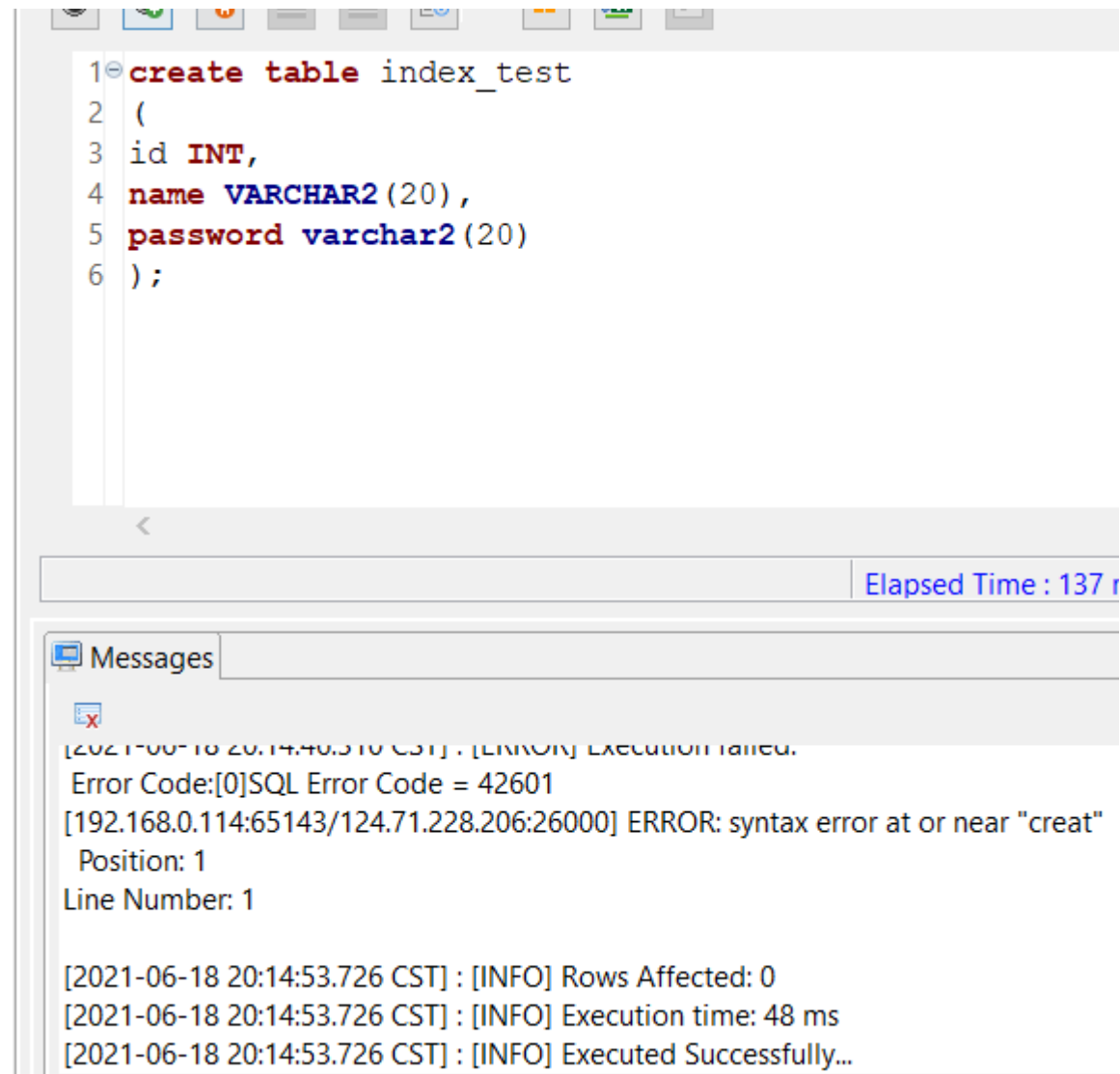
下面给出实验中可能用到的 PLSQL 代码示例，该代码用于向某个表循环插入 100 个数据。其中 count 是一个变量

```
declare
count int;
begin
count := 1;
while count<=100 loop
insert into table_name(id, name, password)
values(count,'some_name','some_name' || count);
count := count + 1;
end loop;
end;
```


5.2 实验任务

创建一个表，然后向该表中循环插入大量数据（1000000 条以上）。然后，对于该表中某个属性建立索引，并进行对比实验，对比利用索引和不利用索引的情况下，完成相同查询任务花费的时间，验证利用索引查询是否真的可以提升查询效率。

注：可以参考 5.4 节的视频。给出每个步骤的 SQL 语句，并对语句执行结果进行截图说明，尤其是其查询时间的区别截图。



```
1 create table index_test
2 (
3 id INT,
4 name VARCHAR2(20),
5 password varchar2(20)
6 );
```

Elapsed Time : 137 r

Messages

[2021-06-18 20:14:40.510 CST] : [ERROR] Execution failed.
Error Code:[0]SQL Error Code = 42601
[192.168.0.114:65143/124.71.228.206:26000] ERROR: syntax error at or near "creat"
Position: 1
Line Number: 1

[2021-06-18 20:14:53.726 CST] : [INFO] Rows Affected: 0
[2021-06-18 20:14:53.726 CST] : [INFO] Execution time: 48 ms
[2021-06-18 20:14:53.726 CST] : [INFO] Executed Successfully...

db_learn@kzmllearn (1) opengauss@kzmikoto (2)

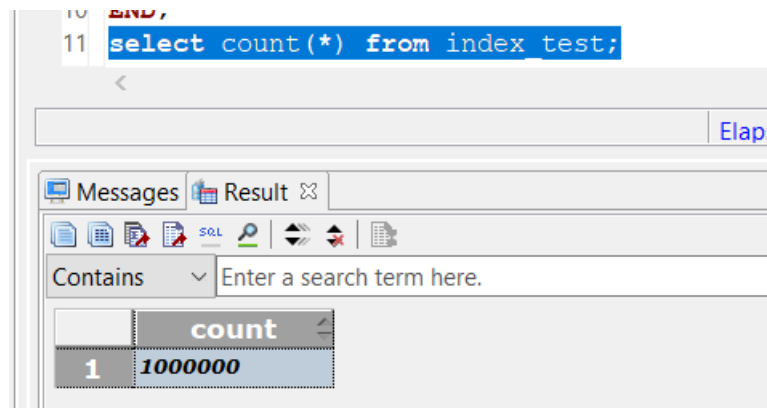
```
1 DECLARE
2 count INT;
3 BEGIN
4 count := 1;
5 while count <= 1000000 LOOP
6 INSERT INTO index_test(id, name, password)
7 VALUES (count, 'some_name', 'some_name' || count);
8 count := count + 1;
9 END LOOP;
10 END;
```

Elapsed Time : 28 sec

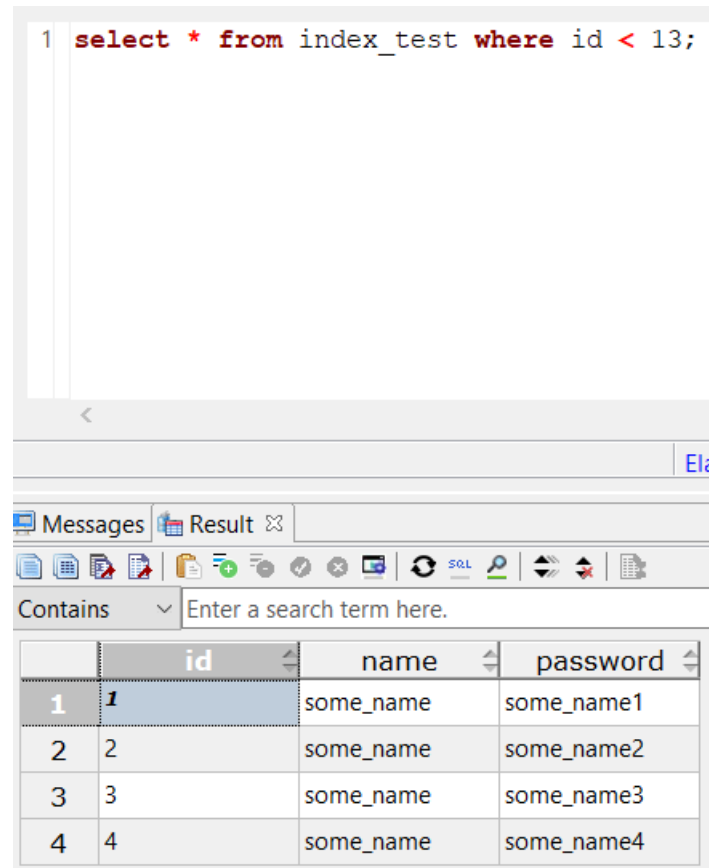
Messages

Error Code:[0]SQL Error Code = 42601
[192.168.0.114:65143/124.71.228.206:26000] ERROR: subprogram body is not ended correctly at end of input
Position: 196
Line Number: 10

[2021-06-18 20:18:29.691 CST] : [INFO] Rows Affected: 0
[2021-06-18 20:18:29.691 CST] : [INFO] Execution time: 28 sec
[2021-06-18 20:18:29.691 CST] : [INFO] Executed Successfully...



插入结果如图：



未建立索引时，查询时间为 842ms：

8.2

1 `select * from index_test where password = 'some_name123456';`

Elapsed Time : 842 ms

Messages Result

Contains Enter a search term here.

| | id | name | password |
|---|--------|-----------|-----------------|
| 1 | 123456 | some_name | some_name123456 |

建立索引后，查询时间为 432ms：

3 `select * from index_test where password = 'some_name123456';`

Elapsed Time : 432 ms

Messages Result

Contains Enter a search term here.

| | id | name | password |
|---|--------|-----------|-----------------|
| 1 | 123456 | some_name | some_name123456 |

第六章 事务的并发控制

6.1 预备知识

事务并发可能带来的 3 大类问题：

- 1) dirty read(脏读)，一个事务读取了另外一个事务未提交的数据。
- 2) non-repeatable read(不可重复读)，同一事务中，前后读取的数据不一致。
- 3) phantom read(幻读)，类似不可重复读，但针对插入/删除操作。

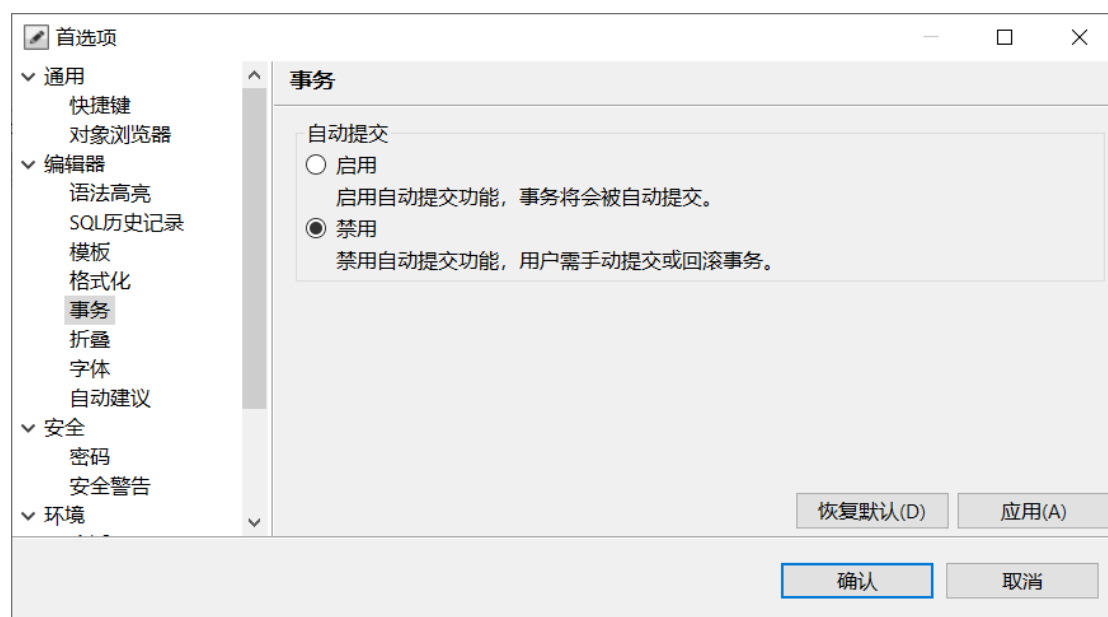
美国国家标准协会在 SQL 标准中，对于满足事务的隔离性的程度划分为以下四个级别。未提交读、已提交读、可重复读和序列化。不同的隔离级别能够解决的上述问题也不同（参考课程第一节）

6.2 实验任务

通过并发实验设计，验证 OpenGauss 是否存在上述三类问题。根据验证结果给出 OpenGauss 系统默认的事务隔离级别。

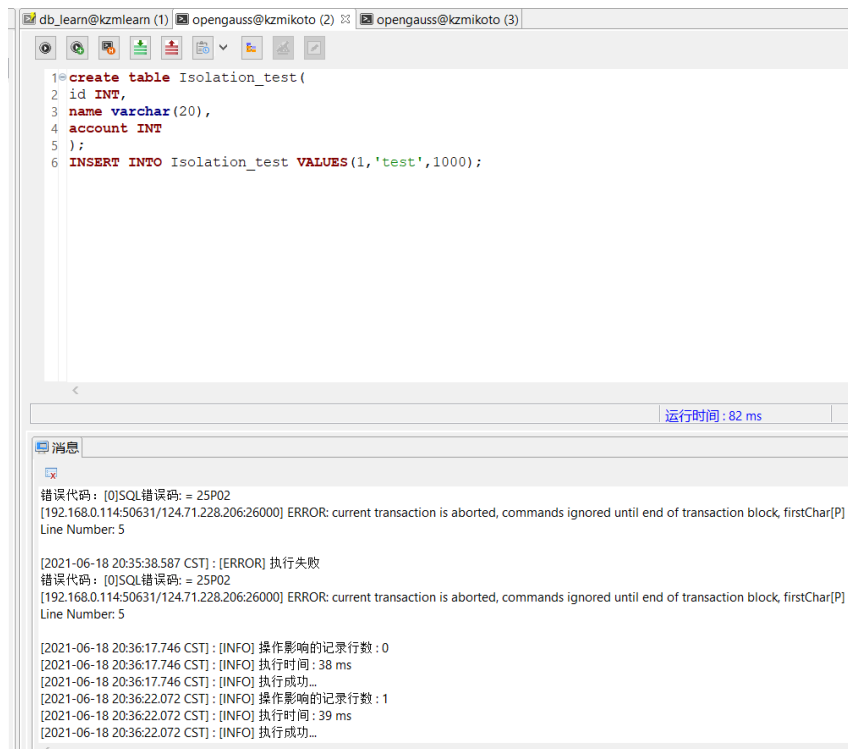
注：实验设计可以模仿 10.6 节中的操作，设计两个事务的并发操作场景，判断是否存在对应问题。需要详细的实验设计过程描述和实验结果的截图。

0. 禁用自动提交

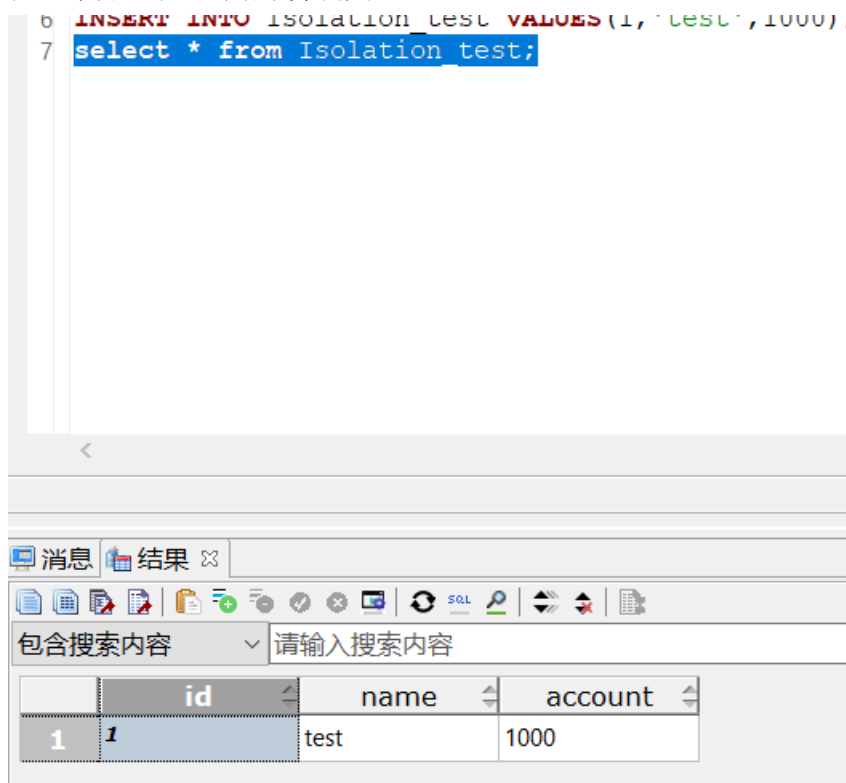


1、脏读

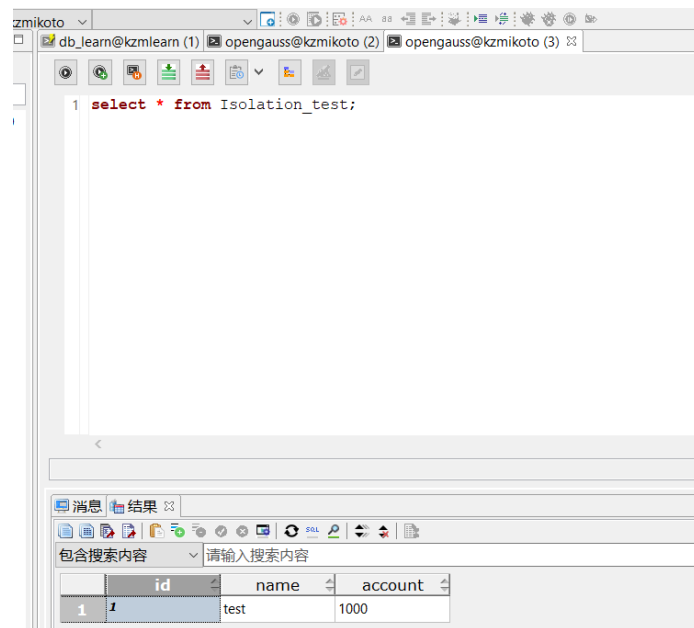
(1) 创建隔离测试表，具有 id,name,account 三个属性，插入一条数据。



在 2 号窗口中查询结果并提交

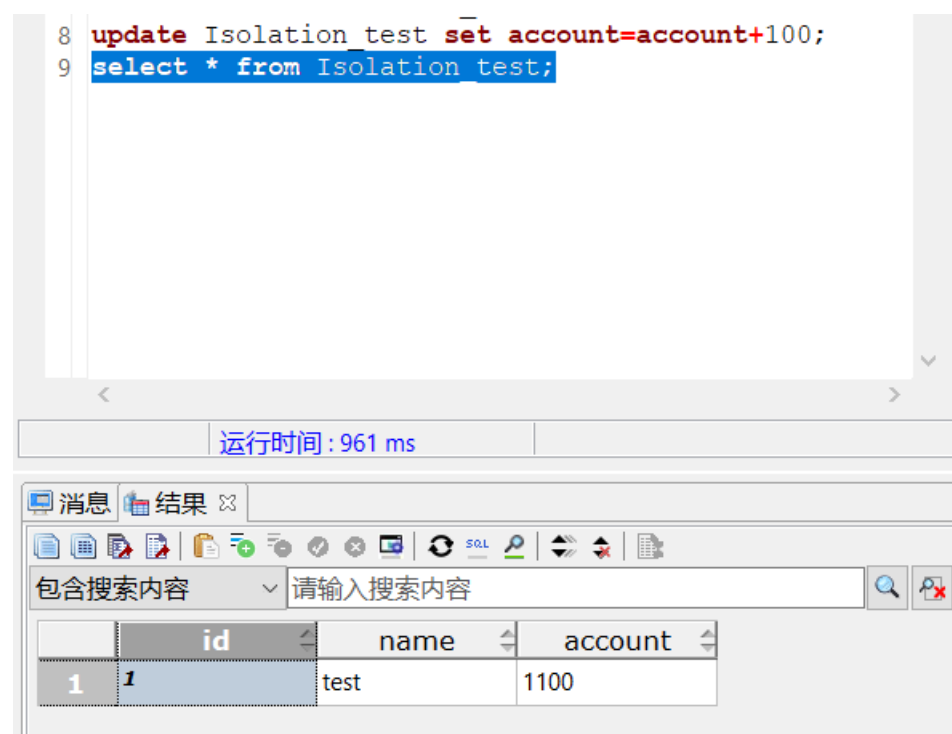


在 3 号窗口查询（由于 1 号窗口被占用，所以用的是 2, 3 窗口）

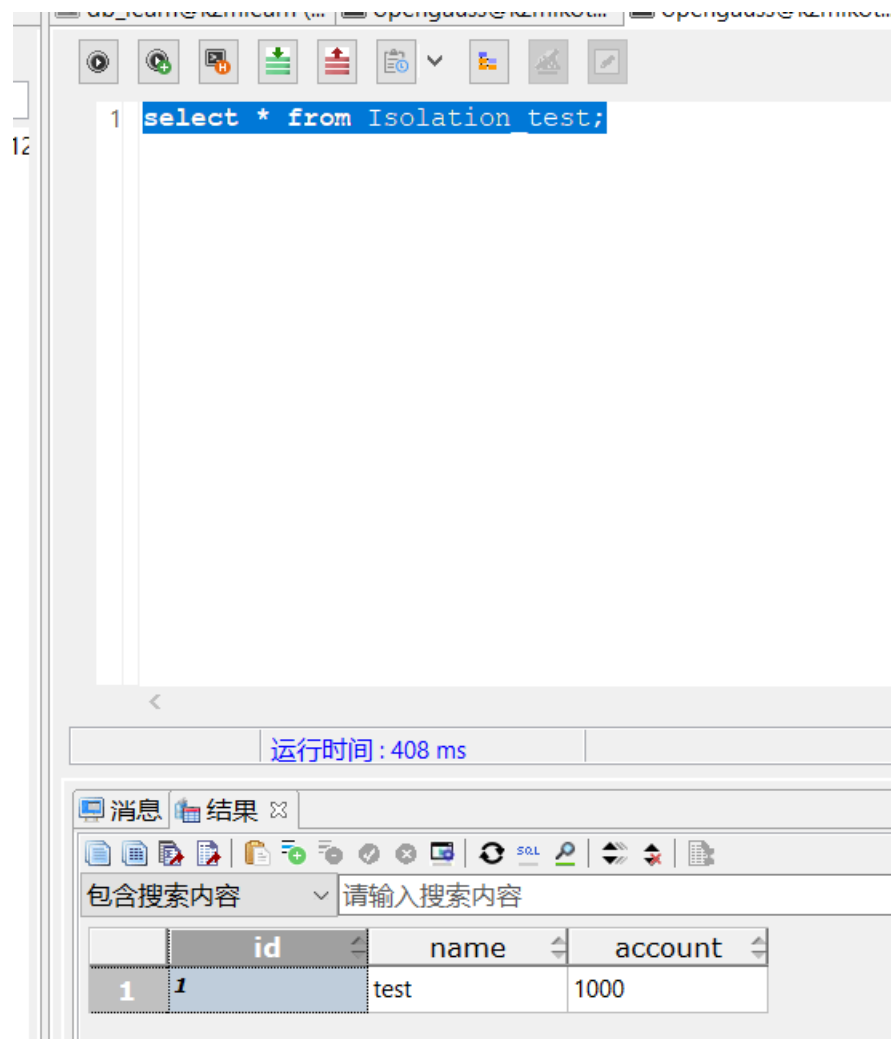


(2) 在窗口 2 中更新数据但不提交

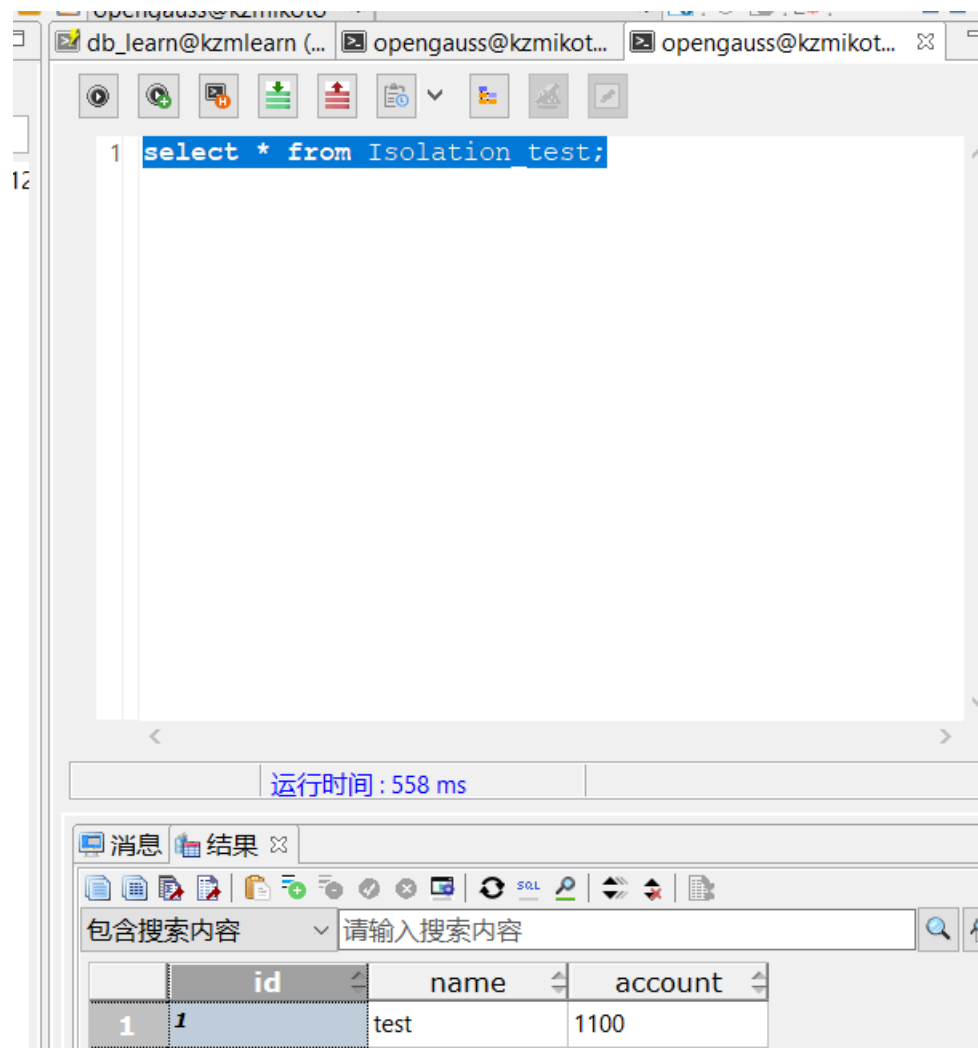
窗口 2 中查询结果:



提交前，窗口 3 中查询结果为:



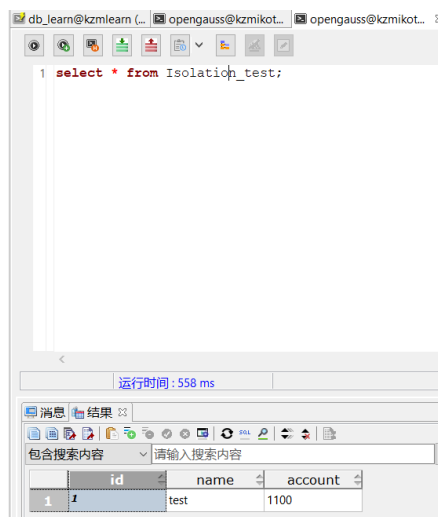
提交后，窗口 3 查询结果为：



结论: gaussdb 系统默认的事务隔离级别能够放置脏读。

2、不可重复读

(1) 窗口 2 更新前, 窗口 3 中读取数据



(2) 窗口 2 更新并提交

The screenshot shows a database client window with the following SQL code executed:

```
1 create table Isolation_test(  
2 id INT,  
3 name varchar(20),  
4 account INT  
5 );  
6 INSERT INTO Isolation_test VALUES(1, 'test', 1000);  
7 select * from Isolation_test;  
8 update Isolation_test set account=account+1000;  
9 select * from Isolation_test;
```

A dialog box titled "提交当前事务?" (Commit current transaction?) is displayed, asking for confirmation. It includes a checkbox for "不再显示" (Do not show again) and buttons for "是" (Yes) and "否" (No).

Below the dialog, a table view shows the data in the 'Isolation_test' table:

| | id | name | account |
|---|----|------|---------|
| 1 | 1 | test | 2100 |

(3) 窗口 3 中读取

The screenshot shows a database client window with the following SQL code executed:

```
1 select * from Isolation_test;
```

The execution time is displayed as "运行时间: 1 sec".

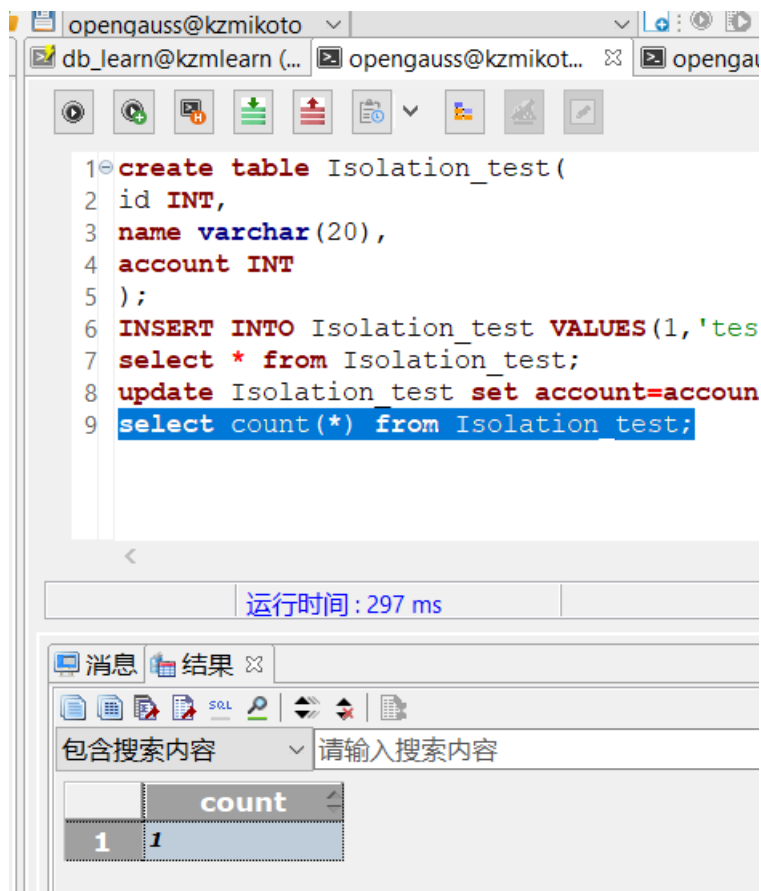
The results are shown in a table view:

| | id | name | account |
|---|----|------|---------|
| 1 | 1 | test | 2100 |

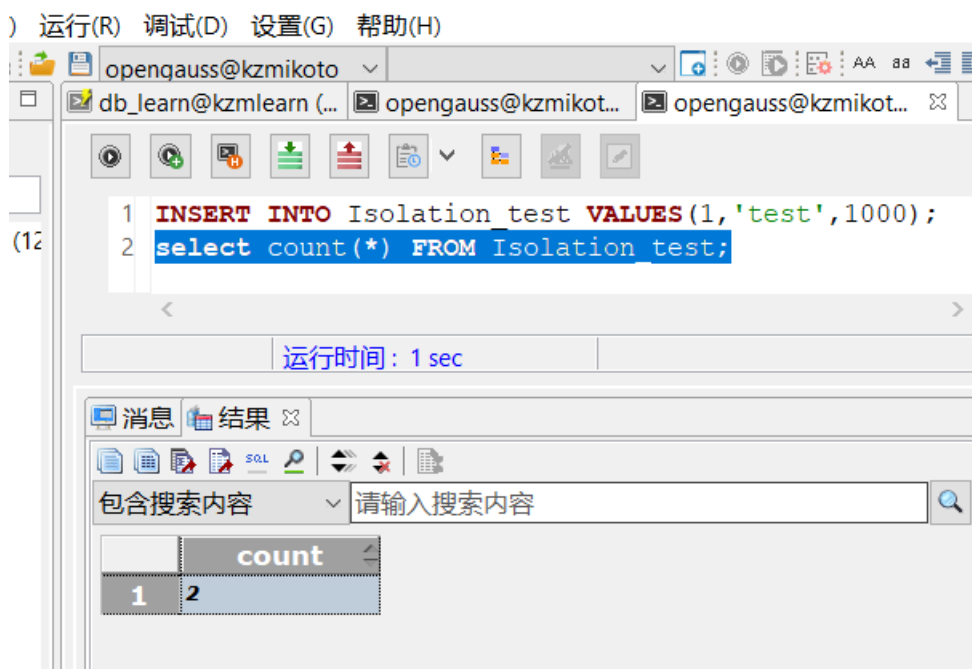
结论：在 gaussdb 系统默认的事务隔离级别下，不可重复读，即，在同一事务中，前后读取数据不一致。

3、幻读

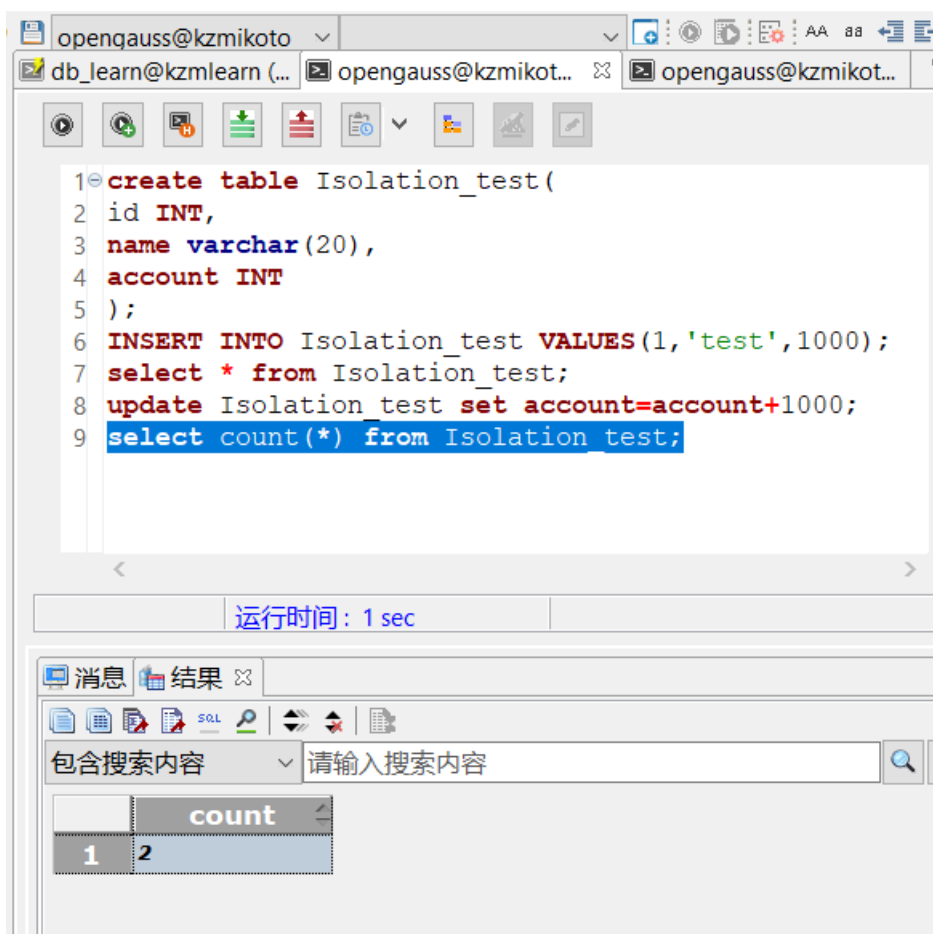
(1) 事务 1 初始数据条数



(2) 在 事 务 2 中 插 入 一 条 数 据



(3) 事务 1 中查询的数据条数



The screenshot shows a SQL client interface with a script editor and a results pane. The script in the editor is as follows:

```
1 create table Isolation_test(  
2 id INT,  
3 name varchar(20),  
4 account INT  
5 );  
6 INSERT INTO Isolation_test VALUES(1,'test',1000);  
7 select * from Isolation_test;  
8 update Isolation_test set account=account+1000;  
9 select count(*) from Isolation_test;
```

Below the script editor, the execution status is shown as "运行时间: 1 sec". The results pane at the bottom displays the output of the final query:

| | count |
|---|-------|
| 1 | 2 |

结论: gaussdb 系统默认的事务隔离级别, 会造成幻读。

综上, gaussdb 的默认隔离级别为 **READ COMMITTED** 读已提交

第七章 用户权限设置及回收

使用 GRANT 命令进行用户授权包括以下三种场景：

- 将系统权限授权给角色或用户。
- 将数据库对象授权给角色或用户。
- 将角色或用户的权限授权给其他角色或用户。

7.1 将系统权限授权给用户或者角色

任务：用自己的姓名全拼代替“joe”完成下述操作，并将操作过程截屏截图：

```
[omm@kzmikoto ~]$ gsql postgres -p 26000 -r
gsql ((openGauss 2.0.0 build 78689da9) compiled at 2021-03-31 21:03:52 commi
t 0 last mr )
Non-SSL connection (SSL connection is recommended when requiring high-securi
ty)
Type "help" for help.
```

```
postgres=# create user kangzheming password 'Shijiedeshi,Renmindemin';
CREATE ROLE
postgres=# grant all privileges to kangzheming
postgres=# ;
ALTER ROLE
```

步骤 1，启动服务器，再使用 gsql 客户端以管理员用户身份连接 postgres 数据库，假设端口号为 26000。

```
gsql -d postgres -p 26000 -r
```

步骤 2，创建名为 joe 的用户，并将 sysadmin 权限授权给 joe。

```
postgres=# CREATE USER joe PASSWORD 'Bigdata@123';
```

```
CREATE ROLE
```

```
postgres=# GRANT ALL PRIVILEGES TO joe;
```

```
ALTER ROLE
```

7.2 将数据库对象授权给角色或用户

任务：用自己的姓名全拼代替“joe”完成下述操作，并将操作过程截屏截图：

步骤 1：

```
postgres=# revoke all privileges from kangzheming
postgres=# ;
ALTER ROLE
postgres=# create schema tpcds;
CREATE SCHEMA
postgres=# █
```

```
postgres=# CREATE TABLE tpcds.reason
(
  r_reason_sk INTEGER NOT NULL,
  r_reason_id CHAR(16) NOT NULL,
  r_reason_desc VARCHAR(20)
);
CREATE TABLE
postgres=#
```

步骤 2:

```
postgres=# GRANT USAGE ON SCHEMA tpcds TO kangzheming;
GRANT
postgres=# GRANT ALL PRIVILEGES ON tpcds.reason TO kangzheming;
GRANT
postgres=#
```

步骤 3:

```
postgres=# GRANT select (r_reason_sk,r_reason_id,r_reason_desc),update (r_re
ason_desc) ON tpcds.reason TO kangzheming
postgres=# ;
GRANT
```

步骤 4:

```
postgres=# GRANT create,connect on database postgres TO kangzheming WITH GRA
NT OPTION;
GRANT
postgres=#
```

步骤 5:

```
postgres=# CREATE ROLE tpcds_manager PASSWORD 'Bigdata@123';
CREATE ROLE
postgres=# GRANT USAGE,CREATE ON SCHEMA tpcds TO tpcds_manager;
GRANT
```

步骤 1, 撤销 joe 用户的 sysadmin 权限, 然后创建 tpcds 模式, 并给 tpcds 模式下创建一张 reason 表。

```
postgres=# REVOKE ALL PRIVILEGES FROM joe;
ALTER ROLE
postgres=# CREATE SCHEMA tpcds;
CREATE SCHEMA
postgres=# CREATE TABLE tpcds.reason
(
  r_reason_sk      INTEGER      NOT NULL,
  r_reason_id      CHAR(16)     NOT NULL,
  r_reason_desc    VARCHAR(20)
);
CREATE TABLE
```

步骤 2, 将模式 tpcds 的使用权限和表 tpcds.reason 的所有权限授权给用户 joe。

```
postgres=# GRANT USAGE ON SCHEMA tpcds TO joe;
GRANT
```

```
postgres=# GRANT ALL PRIVILEGES ON tpcds.reason TO joe;
GRANT
```

授权成功后，joe 用户就拥有了 tpcds.reason 表的所有权限，包括增删改查等权限。

步骤 3，将 tpcds.reason 表中 r_reason_sk、r_reason_id、r_reason_desc 列的查询权限，r_reason_desc 的更新权限授权给 joe。

```
postgres=# GRANT select (r_reason_sk,r_reason_id,r_reason_desc),update (r_reason_desc)
ON tpcds.reason TO joe;
GRANT
```

步骤 4，将数据库 postgres 的连接权限授权给用户 joe，并给予其在 postgres 中创建 schema 的权限，而且允许 joe 将此权限授权给其他用户。

```
postgres=# GRANT create,connect on database postgres TO joe WITH GRANT OPTION;
GRANT
```

步骤 5，创建角色 tpcds_manager，将模式 tpcds 的访问权限授权给角色 tpcds_manager，并授予该角色在 tpcds 下创建对象的权限，不允许该角色中的用户将权限授权给其人。

```
postgres=# CREATE ROLE tpcds_manager PASSWORD 'Bigdata@123';
CREATE ROLE
postgres=# GRANT USAGE,CREATE ON SCHEMA tpcds TO tpcds_manager;
GRANT
```

7.3 将用户或者角色的权限授权给其他用户或角色

任务：用自己的姓名全拼代替“joe”完成下述操作，并将操作过程截屏截图：

```
postgres=# CREATE ROLE manager PASSWORD 'Bigdata@123';
CREATE ROLE
postgres=# GRANT kangzheming TO manager WITH ADMIN OPTION;
GRANT ROLE
```

```
postgres=# CREATE ROLE senior_manager PASSWORD 'Bigdata@123';
CREATE ROLE
postgres=# GRANT manager TO senior_manager;
GRANT ROLE
```

步骤 1，创建角色 manager，将 joe 的权限授权给 manager，并允许该角色将权限授权给其他人。

```
postgres=# CREATE ROLE manager PASSWORD 'Bigdata@123';
CREATE ROLE
postgres=# GRANT joe TO manager WITH ADMIN OPTION;
GRANT ROLE
```

步骤 2，创建用户 senior_manager，将用户 manager 的权限授权给该用户。

```
postgres=# CREATE ROLE senior_manager PASSWORD 'Bigdata@123';
CREATE ROLE
postgres=# GRANT manager TO senior_manager;
GRANT ROLE
```

7.4 权限回收

任务：用自己的姓名全拼代替“joe”完成下述操作，并将操作过程截屏截图：

```
postgres=# REVOKE kangzheming FROM manager;
REVOKE ROLE
postgres=# REVOKE manager FROM senior_manager;
REVOKE ROLE
postgres=# DROP USER manager;
DROP ROLE
postgres=# REVOKE ALL PRIVILEGES ON tpceds.reason FROM kangzheming
postgres=# ;
REVOKE
postgres=# REVOKE ALL PRIVILEGES ON SCHEMA tpceds FROM kangzheming;
REVOKE
postgres=# REVOKE USAGE,CREATE ON SCHEMA tpceds FROM tpceds_manager;
REVOKE
postgres=# DROP ROLE tpceds_manager;
DROP ROLE
postgres=# DROP ROLE senior_manager;
DROP ROLE
postgres=# DROP USER kangzheming CASCADE;
DROP ROLE
```

步骤 1，撤销权限，并清理用户。

```
postgres=# REVOKE joe FROM manager;
REVOKE ROLE
postgres=# REVOKE manager FROM senior_manager;
REVOKE ROLE
postgres=# DROP USER manager;
DROP ROLE
postgres=# REVOKE ALL PRIVILEGES ON tpceds.reason FROM joe;
REVOKE
postgres=# REVOKE ALL PRIVILEGES ON SCHEMA tpceds FROM joe;
REVOKE
postgres=# REVOKE USAGE,CREATE ON SCHEMA tpceds FROM tpceds_manager;
REVOKE
postgres=# DROP ROLE tpceds_manager;
DROP ROLE
postgres=# DROP ROLE senior_manager;
DROP ROLE
postgres=# DROP USER joe CASCADE;
DROP ROLE
```

注意：实验完成后请尽量清理本实验的对象，以免影响与其它实验产生冲突。

第八章 自我总结

（数据库系统理论及上机学习的心得体会，学习理解的难点和对于课程的改进建议）

在数据库上机的过程中，每次都能收获到不同的新体验。受益于华为云的大力支持，我在本学期进行的上机学习能够跳过繁琐的装机阶段，直接来到系统中进行操作。我认为这是详略得当，重点突出的好事，应该大力发扬，争取继承下去。

在数据库理论学习方面，网课和线下课的教学效果都很好，希望网课能够给后面的同学留着，复习的时候确实有用。

在学习理解的难点方面，数据库设计对我来说仍然是难点，希望能够这几天再回顾一下。

课程改进建议：希望能够前八周下发网课内容，后八周进行线下上课。这样如果前八周有同学有兴趣看也可以去看，没有兴趣去看也可以正常教学，能够提高效率，或许能够让教学质量提升。

以上为本学期我的心得体会，感谢老师和助教对本门课程的付出。