**数据库系统课程上机实验手册**

**华为OpenGauss版（2024）**

学号：20222241379

姓名：林洋

教学班：03班

行政班：软2211班

大连理工大学

Dalian University of Technology

目录

[前言 实验介绍 3](#_Toc165661868)

[0.1关于本实验 3](#_Toc165661869)

[0.2实验目的 3](#_Toc165661870)

[0.3实验内容 3](#_Toc165661871)

[0.4 实验环境配置 3](#_Toc165661872)

[第一章 DDL 4](#_Toc165661873)

[1.1 预备知识 4](#_Toc165661874)

[1.2 实验任务 4](#_Toc165661875)

[第二章 DML-增删改 8](#_Toc165661876)

[2.1 预备知识 8](#_Toc165661877)

[第三章 DML-查询 10](#_Toc165661878)

[3.1 预备知识 10](#_Toc165661879)

[3.2 单表查询 11](#_Toc165661880)

[3.3 聚合查询 13](#_Toc165661881)

[3.4 多表查询 14](#_Toc165661882)

[3.5 子查询（必须使用子查询实现） 16](#_Toc165661883)

[3.6 集合查询 19](#_Toc165661884)

[第四章 数据库设计 21](#_Toc165661885)

[4.1 预备知识 21](#_Toc165661886)

[4.2 实验内容 22](#_Toc165661887)

[第五章 索引操作 24](#_Toc165661888)

[5.1 预备知识 24](#_Toc165661889)

[5.2实验任务 25](#_Toc165661890)

[第六章 事务的并发控制 26](#_Toc165661891)

[6.1预备知识 26](#_Toc165661892)

[6.2实验任务 26](#_Toc165661893)

[第七章 用户权限设置及回收 27](#_Toc165661894)

[7.1 预备知识 27](#_Toc165661895)

[7.2 将数据库对象授权给角色或用户 27](#_Toc165661896)

[7.3 将用户或者角色的权限授权给其他用户或角色 28](#_Toc165661897)

[7.4 权限回收 29](#_Toc165661898)

[第八章 自我总结 30](#_Toc165661899)

# 前言 实验介绍

## 0.1关于本实验

《数据库系统》实验内容包括六个部分，从实验平台准备开始，掌握数据库的基本管理方法以及SQL命令（包括数据查询、数据更新、数据定义和数据控制）。

## 0.2实验目的

熟练掌握OpenGauss数据库的常规操作，尤其是基于SQL语言的数据访问,通过实践理解OpenGauss系统对索引和事务并发控制的实现。

## 0.3实验内容

第一章 数据定义实验。介绍了DDL的类型、语法格式和使用场景，掌握如何用数据定义语言定义或修改数据库中的对象。

第二章 数据更新实验。通过对DML语言基本语法和使用，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。

第三章 数据查询实验。通过基本的SQL语言使用，掌握从一个或多个表查询数据的操作。

第四章 数据库设计。使用数据库设计工具，完成数据库概念结构设计及逻辑结构设计，并根据设计结果生成实际可操作的数据库模式。

第五章 索引操作。通过基本的索引使用，掌握索引的创建和使用。

第六章 事务并发控制。通过对事务并发场景的设计，理解OpenGauss数据库关系对并发的支持情况。

第七章 用户权限设置及回收，掌握用户和权限管理的基本操作。

## 0.4 实验环境配置

参照《使用docker安装opengauss》教程，完成数据库实验环境安装。注：Mac系统无法使用Data Studio，可以选择使用DataGrip、Navicat、DBeaver配合openGuass的JDBC驱动程序包访问openGauss数据库。

# 第一章 DDL

## 1.1 预备知识

DDL全称是Data Definition Language，即数据定义语言，定义语言就是定义关系模式、删除关系、修改关系模式以及创建数据库中的各种对象，比如表、聚簇、索引、视图、函数、存储过程和触发器等等。

## 1.2 实验任务

#### 1.2.1 创建数据库和用户

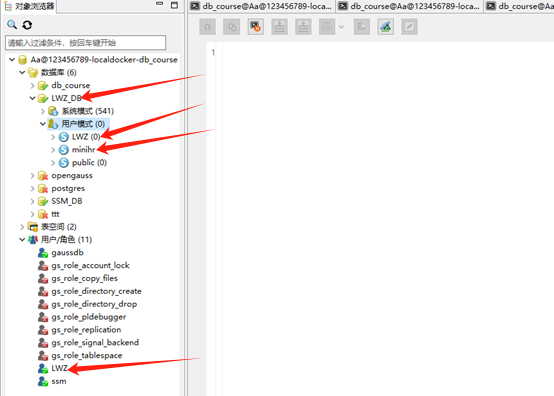
参考教程完成OpenGauss安装，并利用DataStudio连接到数据库。按照教程操作，默认会自动已经存在名称为Postgres的数据库，同时有用户名为gaussdb的用户可用于连接到数据库。本节的实验任务是：

* 创建一个新的数据库（用自己姓名的首字母加\_DB命名，例如用户是龙文章，就命名为LWZ\_DB）；
* 创建一个新的用户（用自己姓名的首字母命名）。注意，创建一个新的用户就会在当前连接的数据库上创建一个新的模式；
* 创建一个新的模式（XXX\_minihr），并授予自定义用户（比如LWZ）以及gaussdb数据库对于此模式的所有权限。
* 创建一个新的连接，用新创建的用户（例如，LWZ）连接新创建的数据库（例如，LWZ\_DB）。

最后需要通过截图证明完成上述操作。

可以在DataStudio上用界面操作，也可以在通过写SQL语言完成。

示例截图（用自己的图替换示例图）**：**

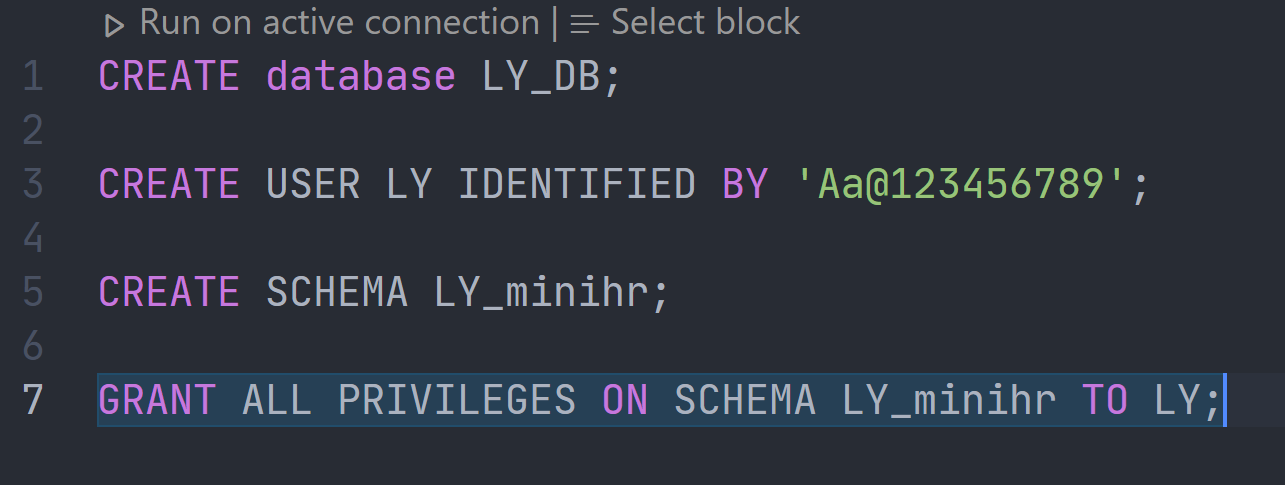


**待完成内容（见下页）**

1. **按照上面要求完成结果的截图：**



1. **给出创建上面要求数据库、模式及用户的SQL命令**



#### 1.2.2 创建数据表

创建两个数据表：员工（XXX\_emp）和部门（XXX\_dept）。

员工表中包括如下属性：员工编号（empno），员工姓名（ename），员工职位（empJOB）,员工直属领导编号（MGR），入职时间（hiredate），月工资（sal）,补助（comm）, 员工所在部门编号（deptno）。其中员工编号为主码（empno）。

部门表中包含如下属性：部门编号（deptno），部门名称（dname），部门所在地名称（loc）。

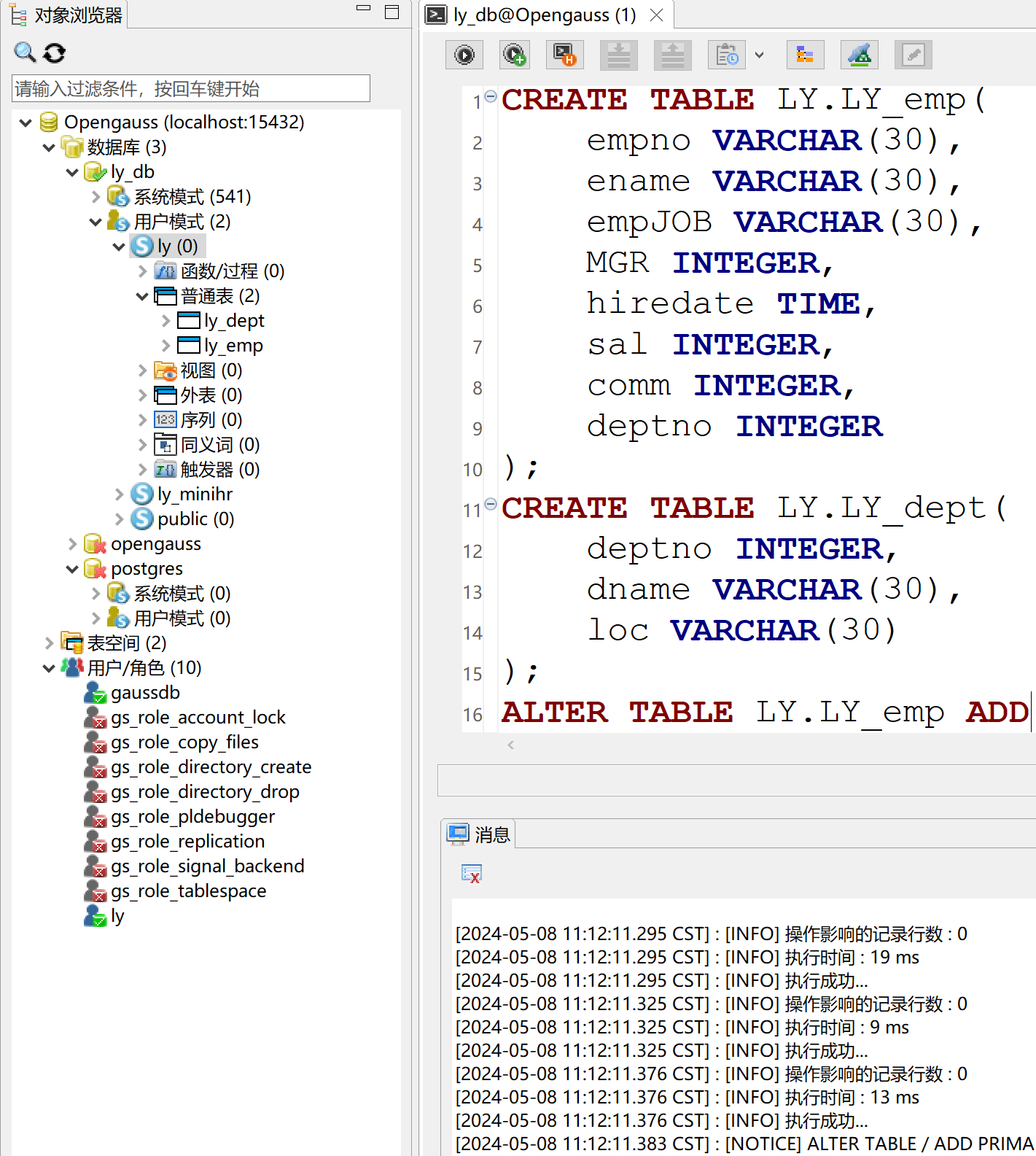
要求：

1. 每个属性的域根据实际情况进行选择。
2. 用SQL语言在上一步创建的数据库（XXX\_DB）的自定义模式（XXX）中创建上述两个表。
3. 员工表中主码的名称设置为：XXX\_PK\_EMP。

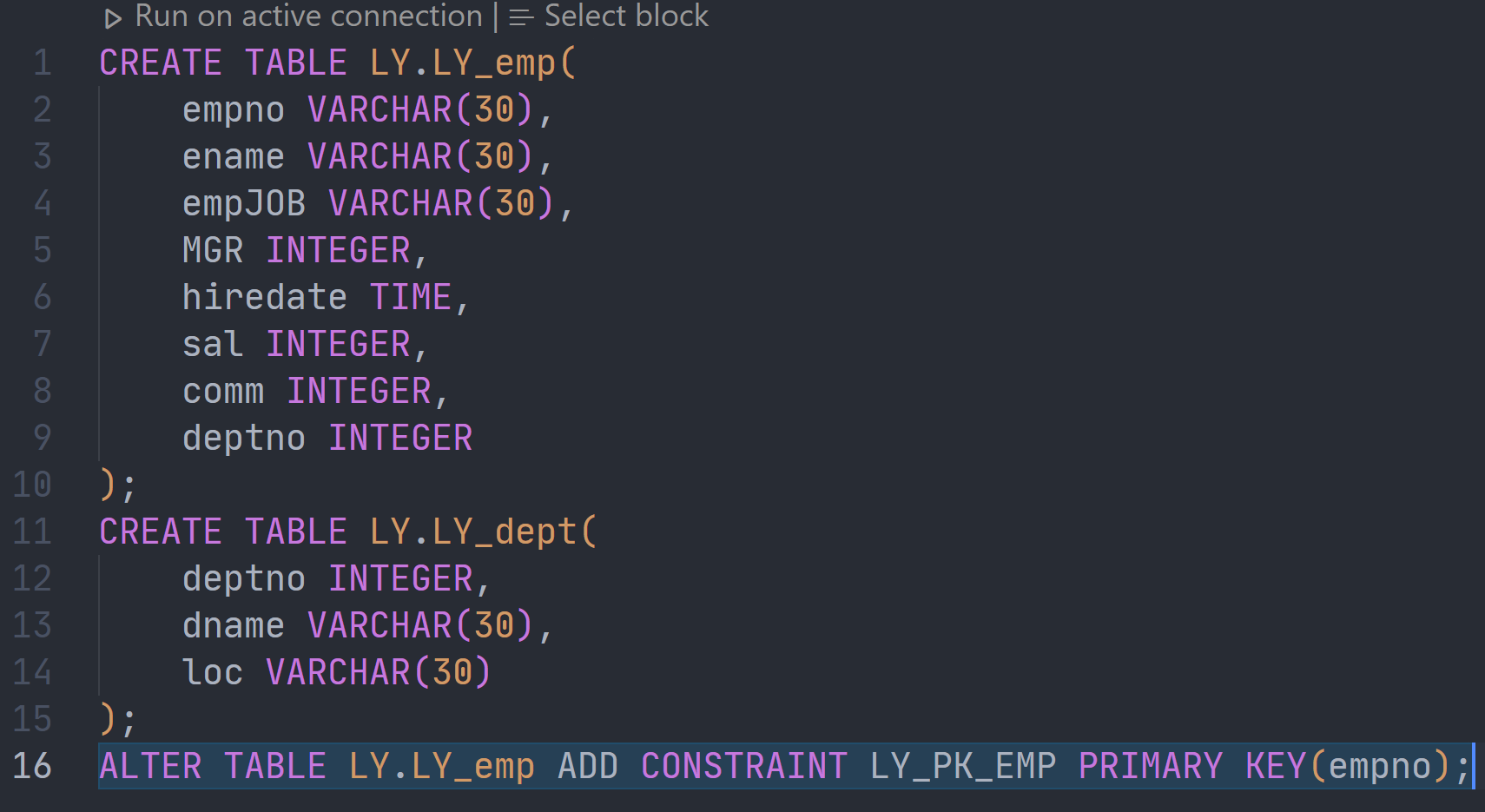
**需要给出创建成功的截图和对应创建表的SQL代码。**

**注：XXX为自己姓名的首字母**

截图（用自己的图替换示例图）：



代码：



#### 1.2.3 修改约束

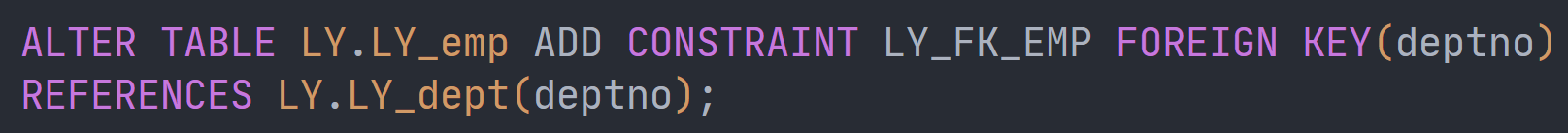
（1）在不删除和重建XXX\_dept表的情况下，为XXX\_dept表添加主码，将deptno属性设为主码属性，主码约束名称为XXX\_pk\_dept。

给出执行代码：



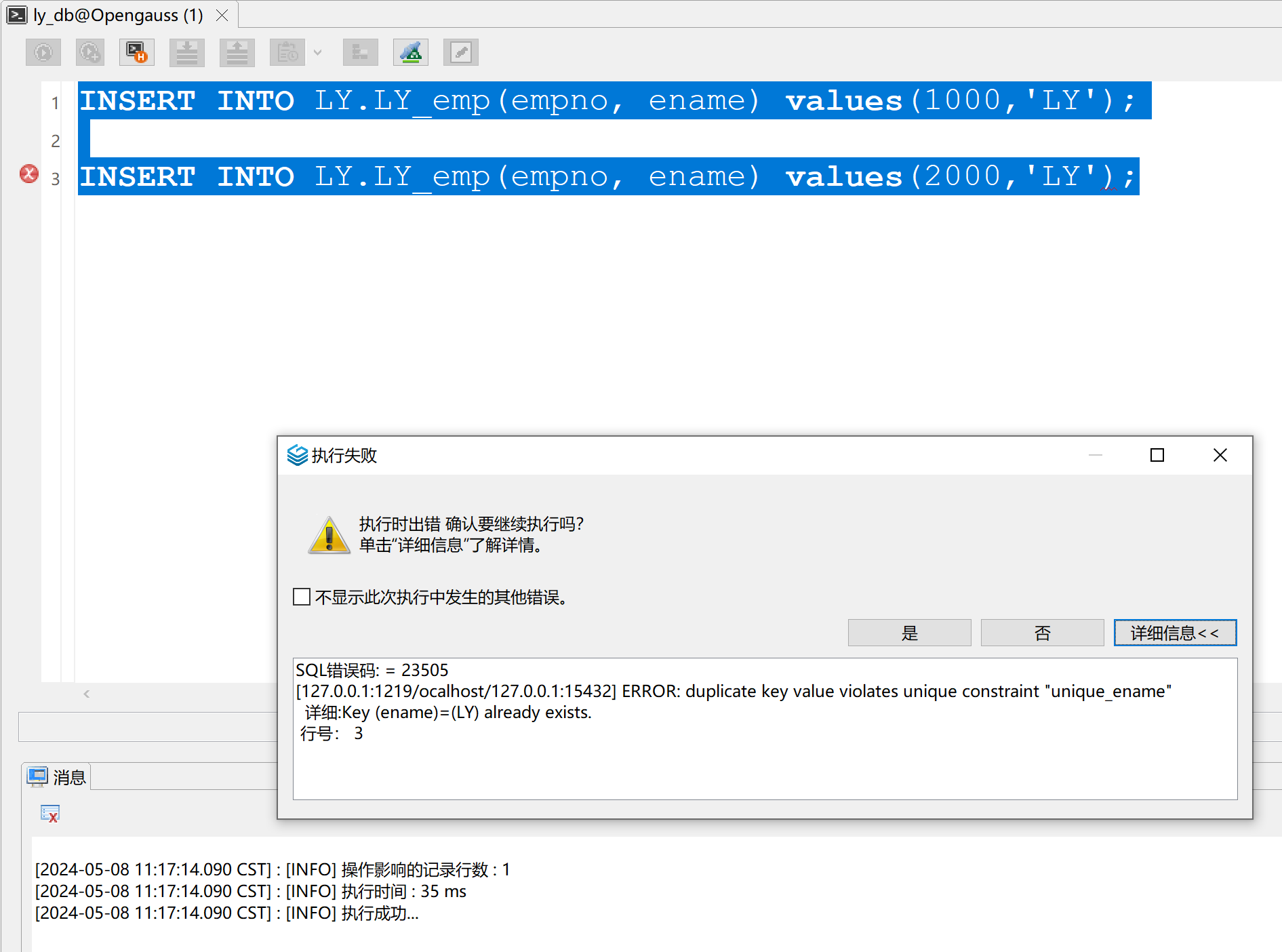
（2）在不删除和重建XXX\_emp表的情况下，为XXX\_emp表添加外码，将deptno属性设为外码，被参照表为XXX\_dept表。

给出执行代码：

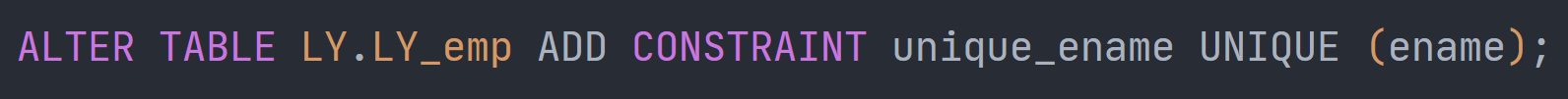


（3）给员工（XXX\_emp）模式中的员工姓名（ename）属性添加unique约束，之后通过插入同名数据进行验证，最后，再将添加的unique约束删除。

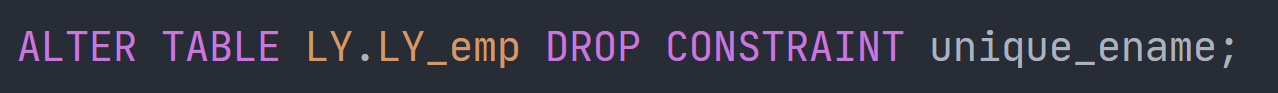
要求：给出执行代码



添加代码：



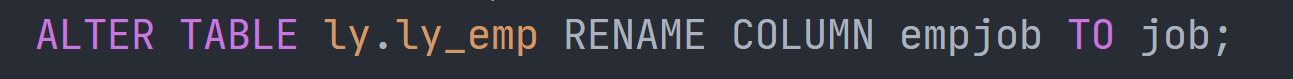
删除代码：



#### 1.2.4 修改模式结构

（1）将员工职位属性更名为job。

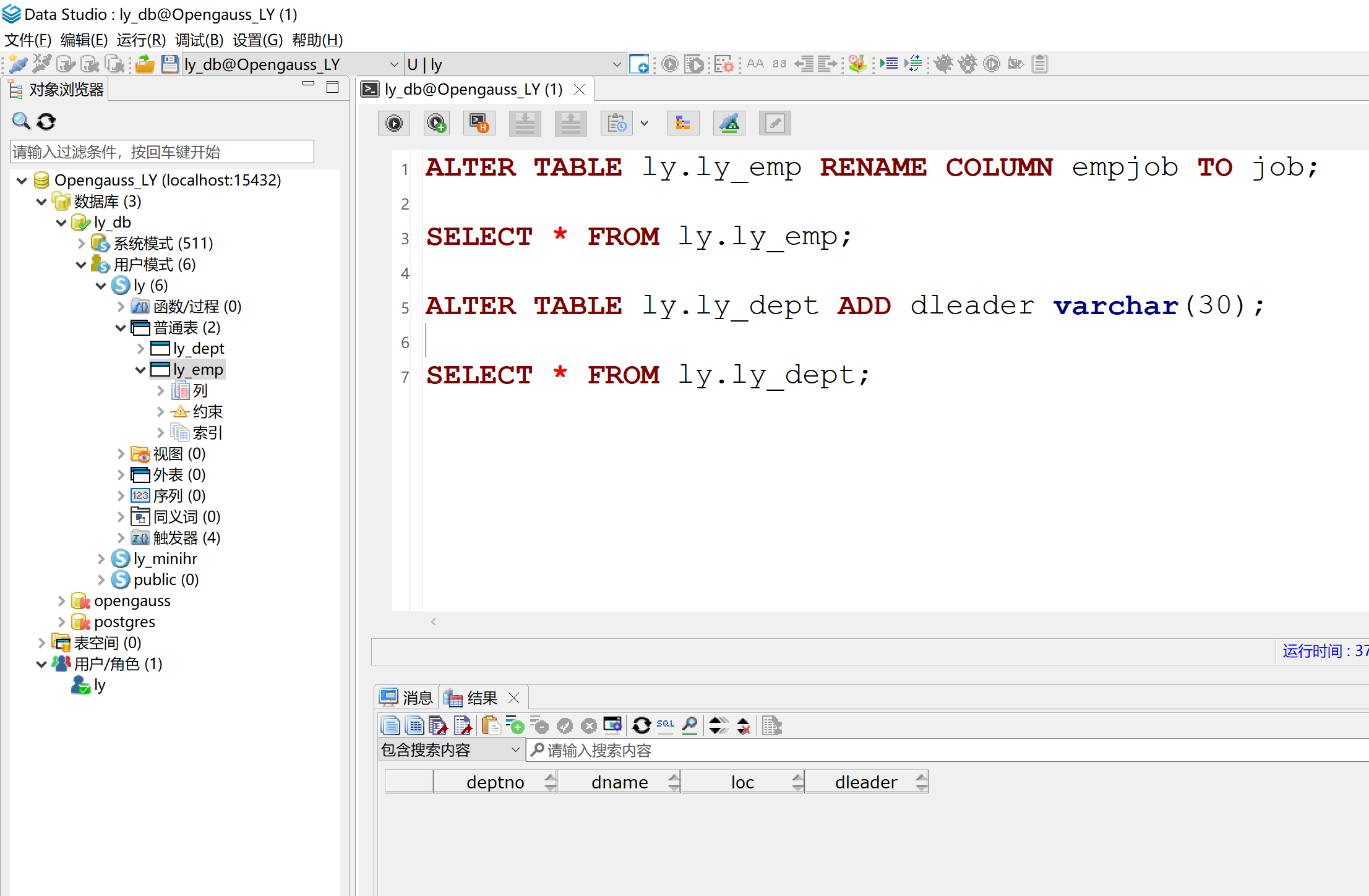
给出执行代码



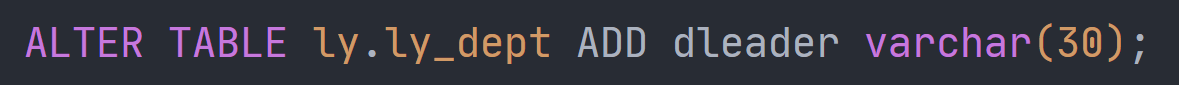
（2）给部门（XXX\_dept）模式中添加一列——部门领导（dleader），添加完成后再进行删除。

要求：给出执行代码和对应添加成功后的截图

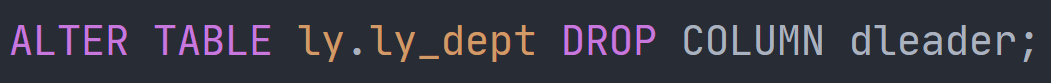
添加列成功的截图：



添加代码：



删除代码：



# 第二章 DML-增删改

## 2.1 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。

#### 2.2.1 添加数据

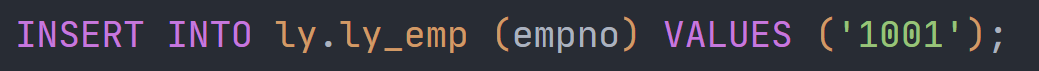
在员工表中添加一条员工信息，具体属性需要根据自己真实情况进行发挥。

要求：给出三种insert语句实现方案，并给出最终的插入成功后截图。可以用select \* from XXX\_emp语句查询出XXX\_emp表中所有的数据。

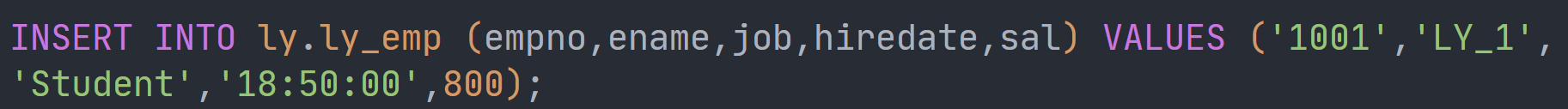
代码1：（不指定数据表属性列表方式）



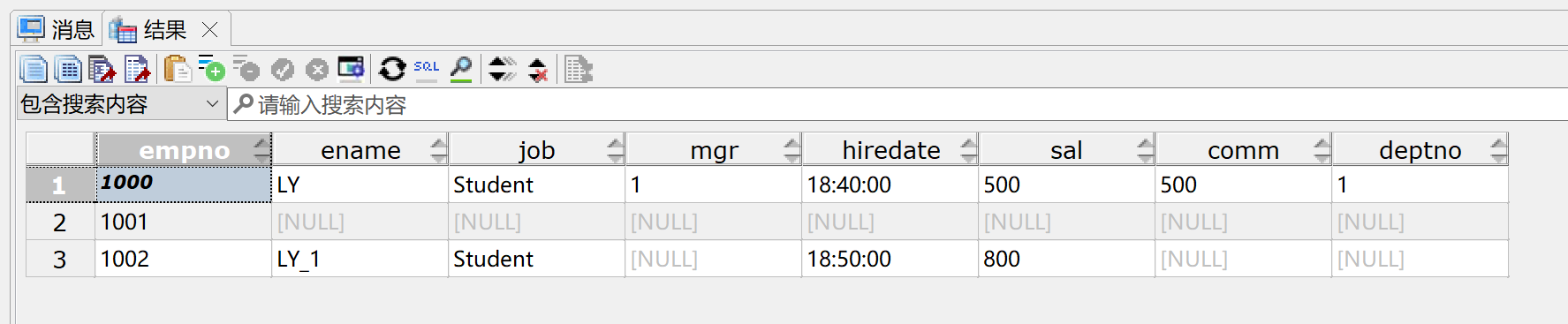
代码2：（指定数据表属性列表方式）



代码3：（仅仅指定empno,ename,job,hiredate,sal属性）



截图（用自己的截图代替示例截图）：

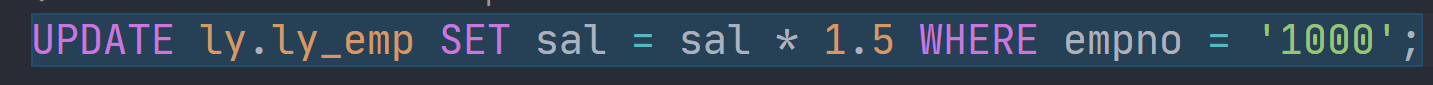


#### 2.2.2 修改数据

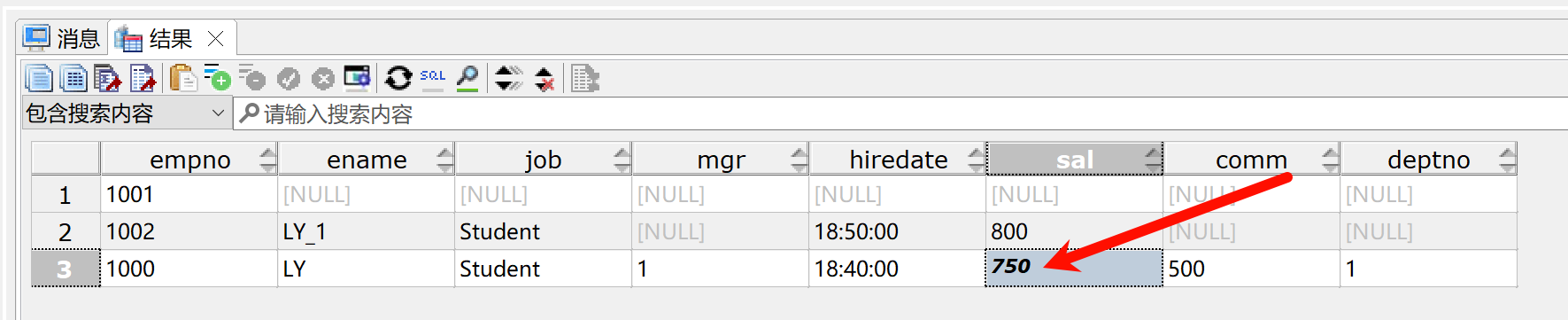
对于上一题中添加的数据，修改其工资为原来的1.5倍。

要求：给出执行的代码，并给出成功后截图。可以用select \* from XXX\_emp语句查询出XXX\_emp表中所有的数据。

代码：



截图（用自己的截图代替示例截图）：

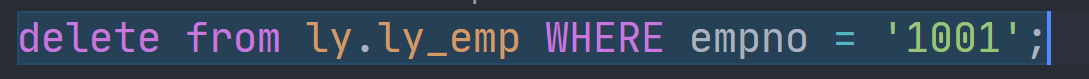


#### 2.2.3 删除数据

删除在2.2.1题中添加的那条数据。

要求：给出执行的代码。

代码：



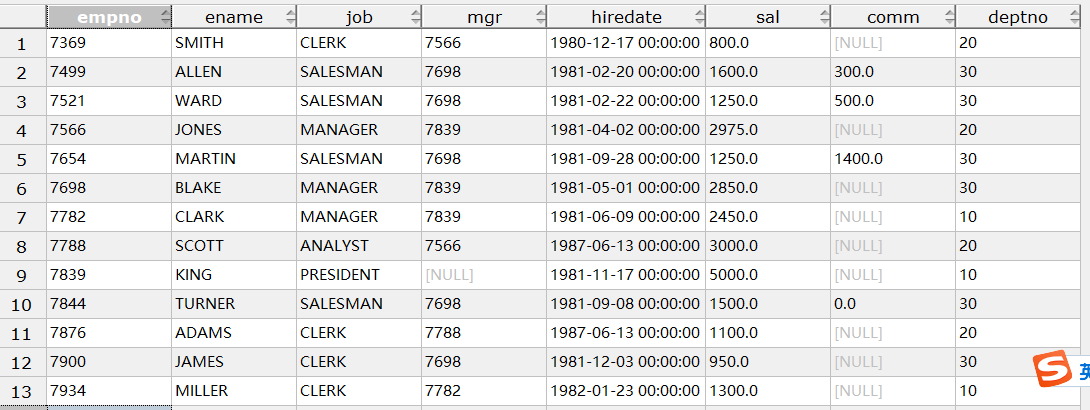
# 第三章 DML-查询

## 3.1 预备知识

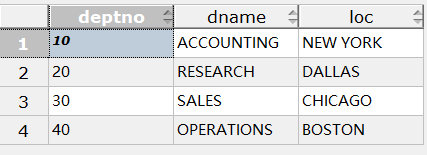
DML是Data Manipulation Language的缩写，意思是数据操纵语言，是指在SQL语言中，负责对数据库对象运行数据访问工作的指令集，除了INSERT、UPDATE、DELETE三种指令为之外，更加常用的是SELECT。本章的题目主要围绕SELECT指令高效查询数据库。

在进行具体的查询任务之前，需要根据提供的脚本（**scott-opengauss-init.sql**）创建相关数据表，并导入供查询的数据。可以自行与下面给出的截图进行对比，保证对应的模式和数据已经创建和导入完成，以保证后续题目顺利执行。（注意：直接执行脚本中命令默认是在所选择数据库的public模式下创建数据表。）

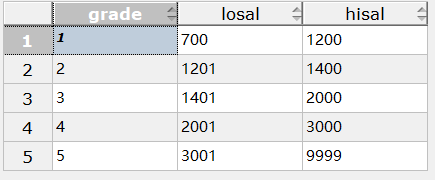
Emp表中包含的数据：



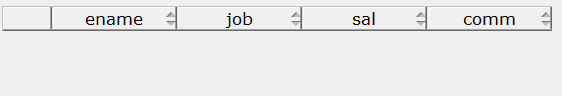
Dept表中包含的数据：



Salgrade表中包含的数据：



Bonus表中没有数据：



## 3.2 单表查询

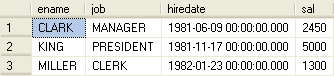
单表查询是最简单的查询方式。所有要查询的信息，都集中在一张表中。也就是说，SQL语句中的FROM子句中只有一个表。

要求：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。

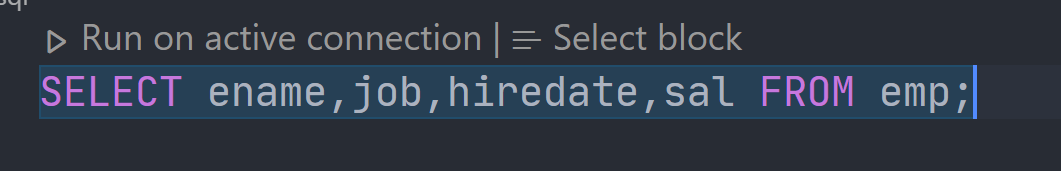
#### 3.2.1 实践题目1

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：



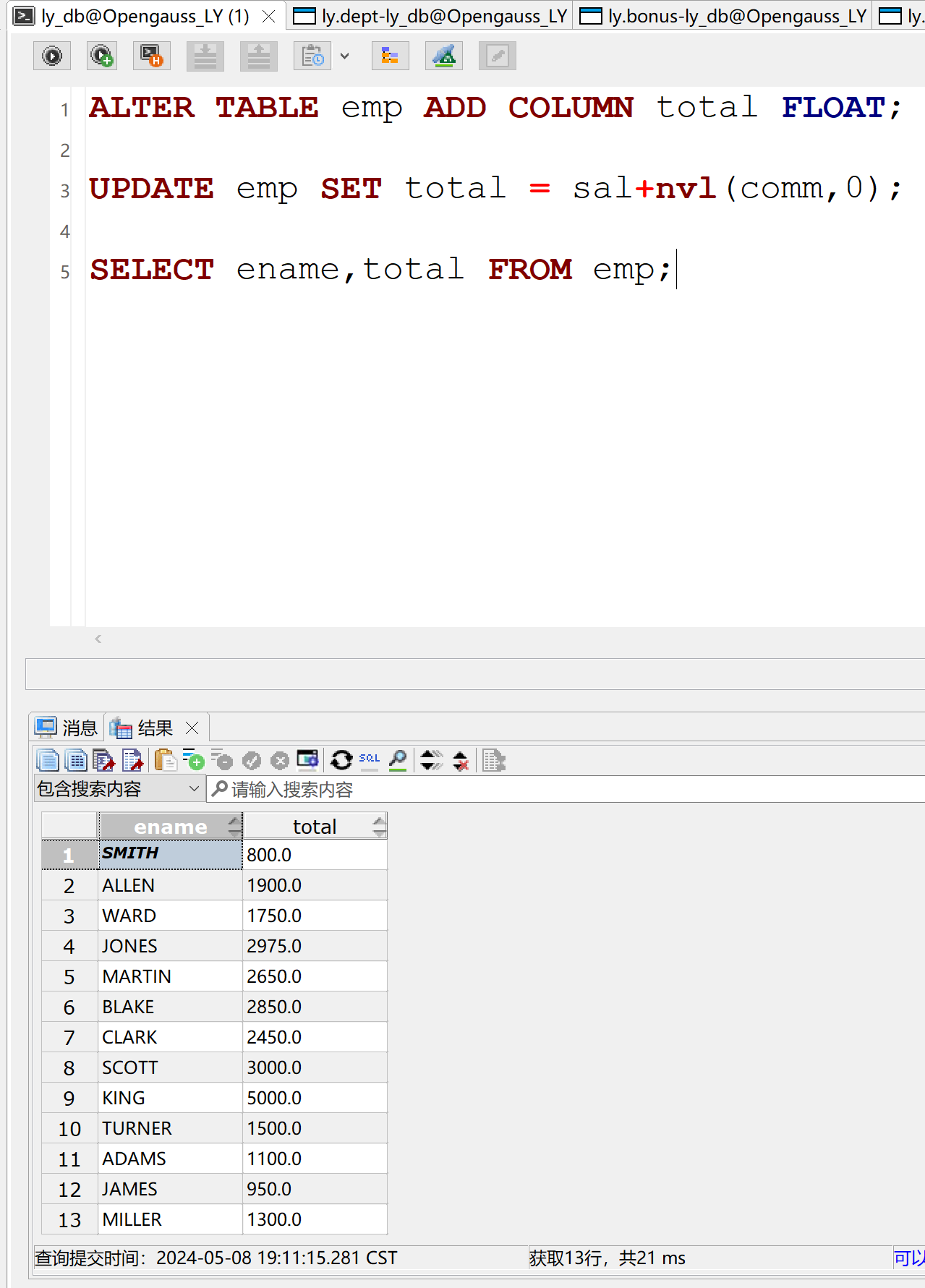
代码：



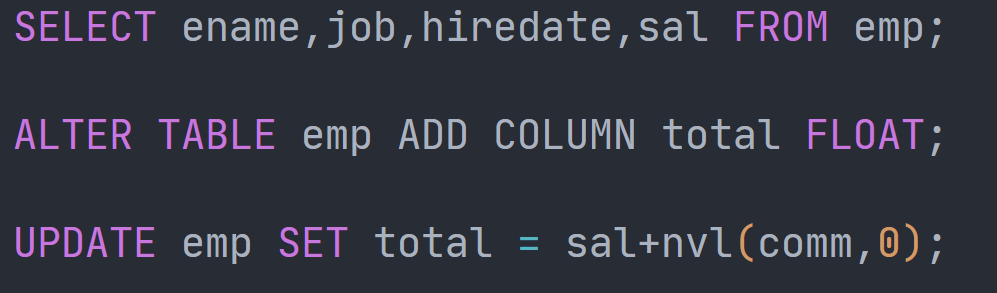
#### 3.2.2 实践题目2

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：



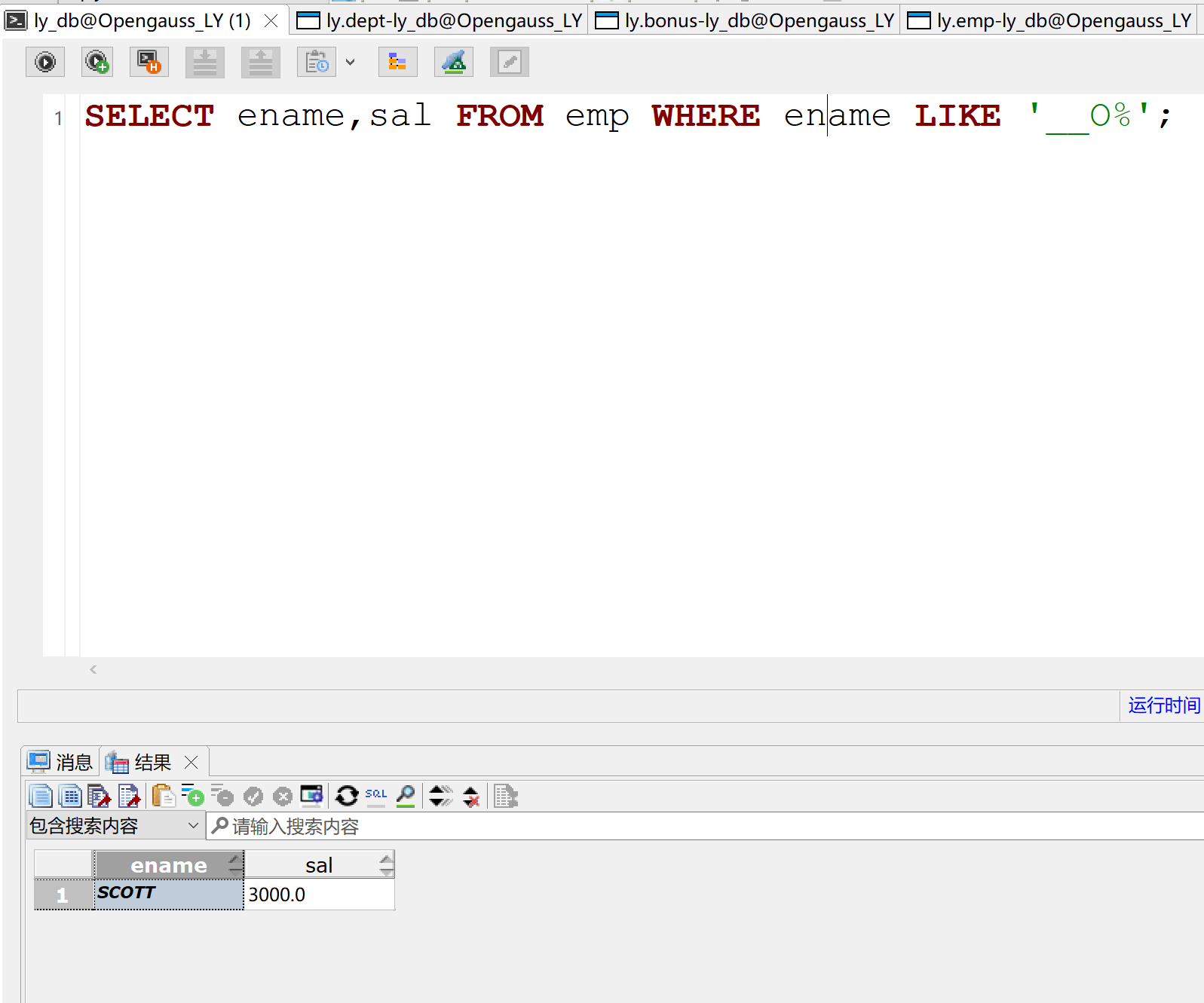
代码：



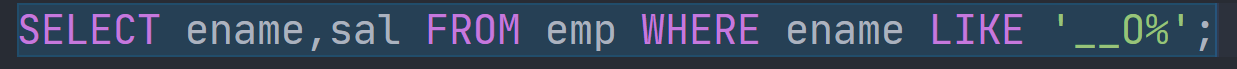
#### 3.2.3 实践题目3

显示第3个字符为大写O的所有员工的姓名及工资。

结果：



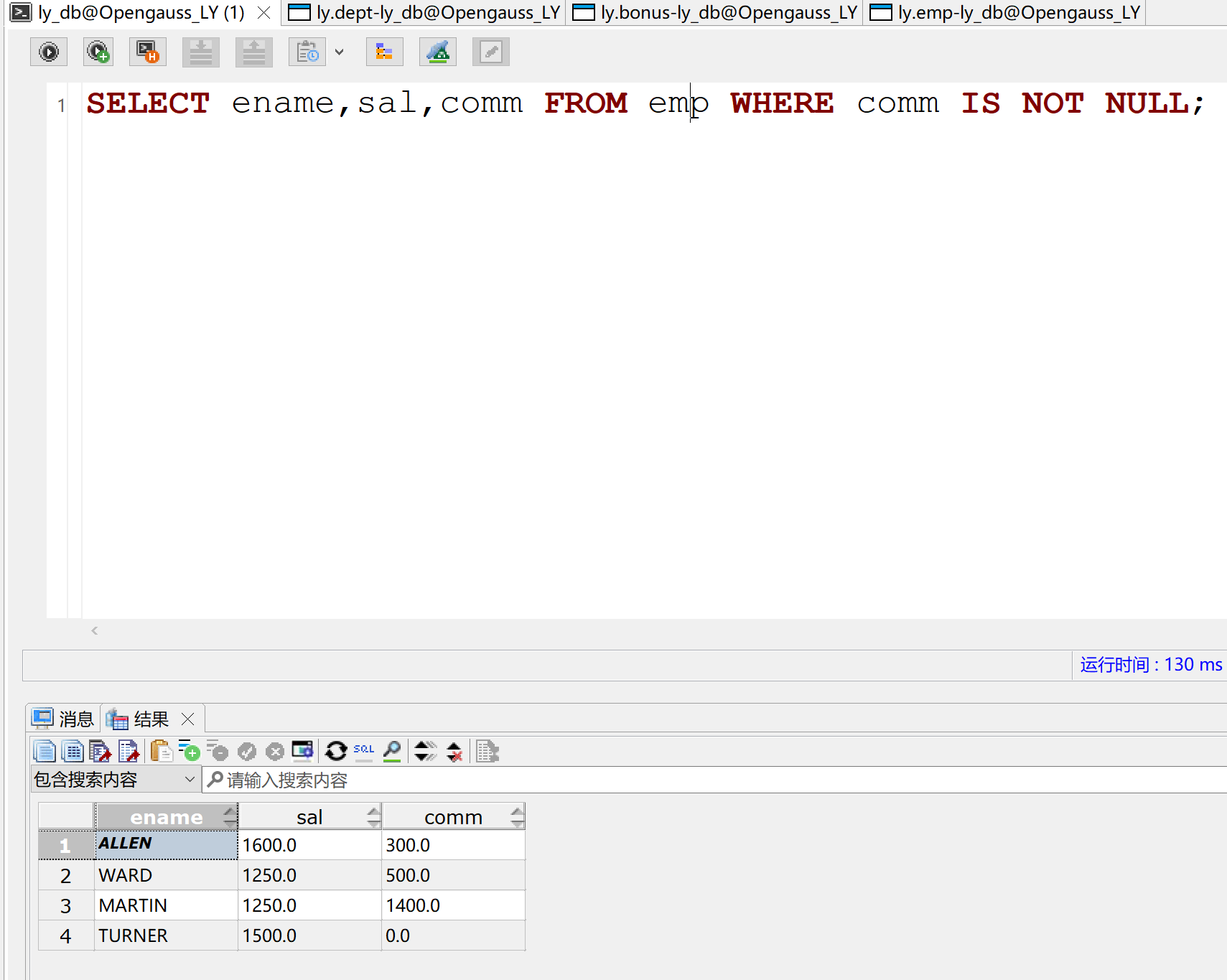
代码：



#### 3.2.4 实践题目4

显示有补助的员工的姓名，工资，补助。

结果：



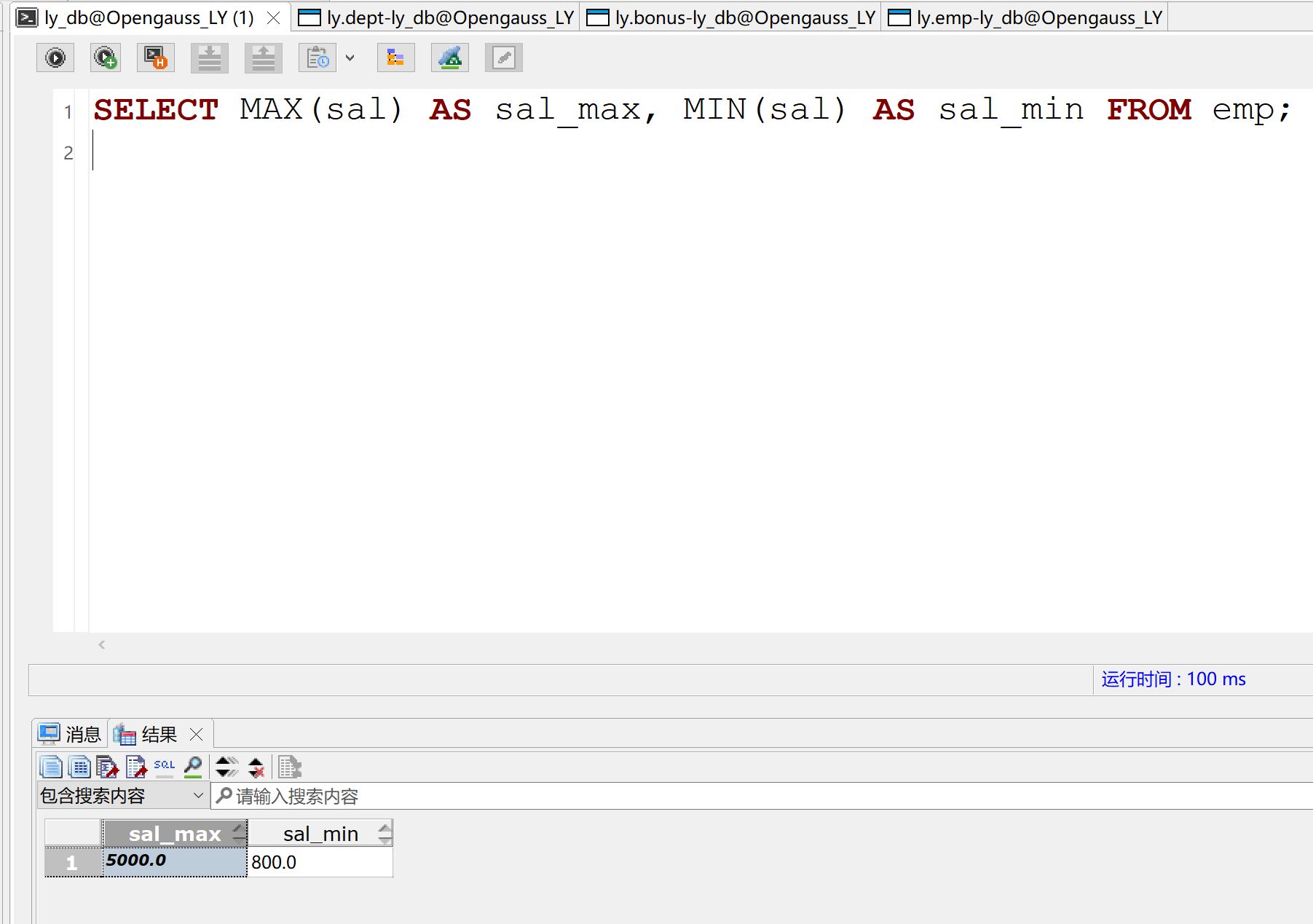
代码：



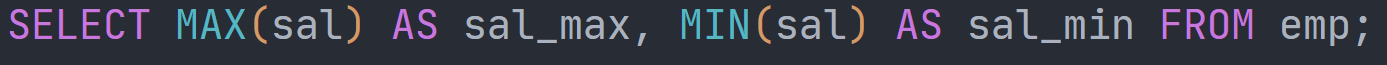
#### 3.2.5 实践题目5

显示员工的最高工资和最低工资。

结果：



代码：



## 3.3 聚合查询

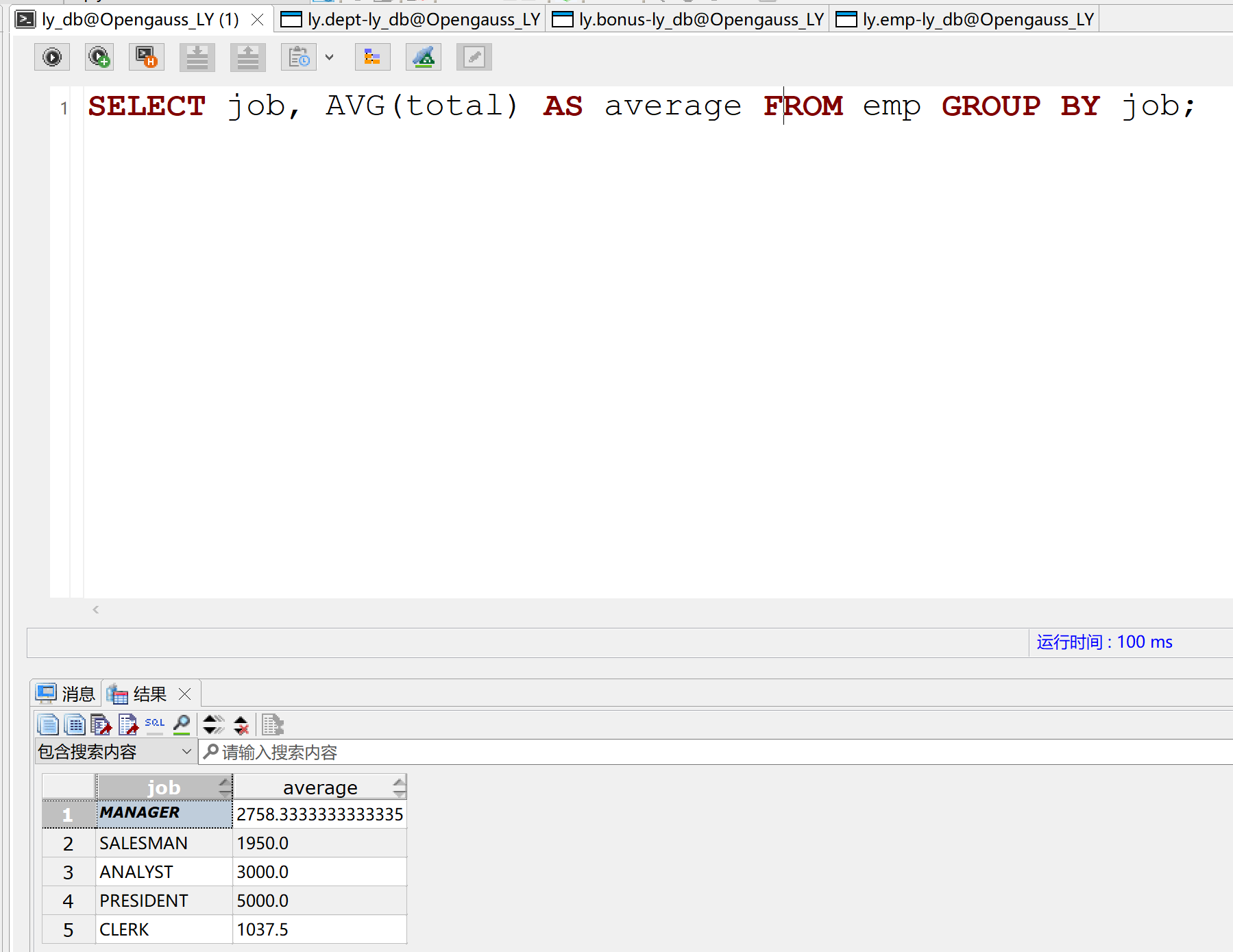
在查询中，我们经常会遇到这样的问题：求平均值、求最值等等。我们需要使用一些函数如AVG(), MAX()等来进行计算，也需要通过GROUP BY子句来聚合属性。

要求：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。

#### 3.3.1 实践题目1

显示每种职业的平均工资。

结果：



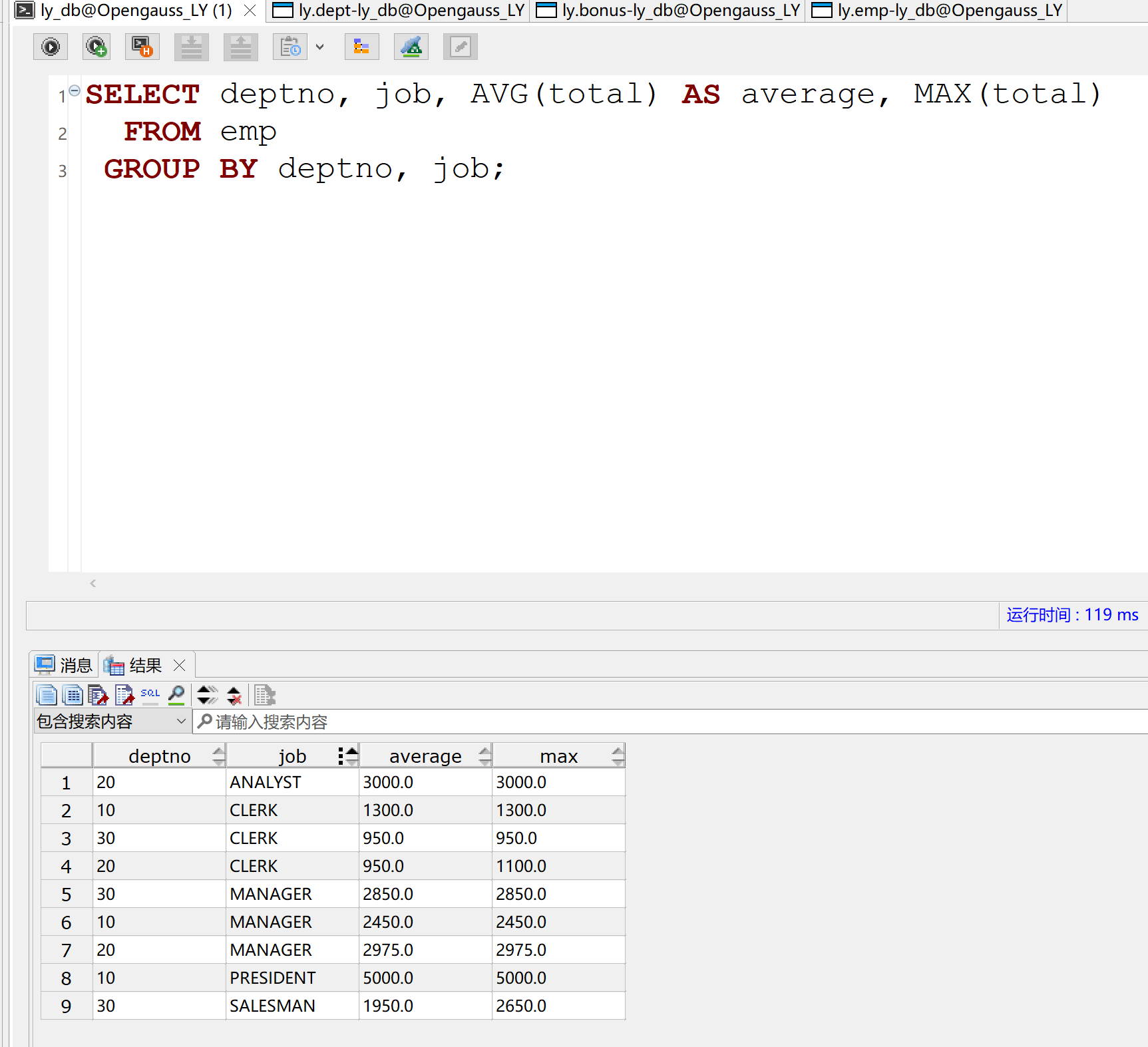
代码：



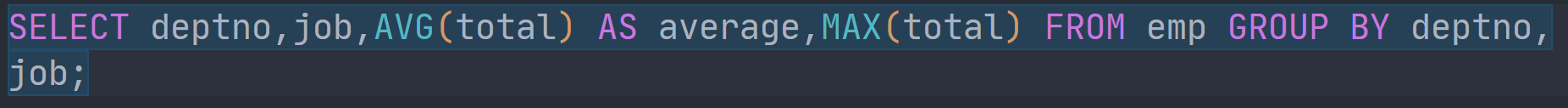
#### 3.3.2 实践题目2

显示每个部门每种岗位的平均工资和最高工资。

结果：



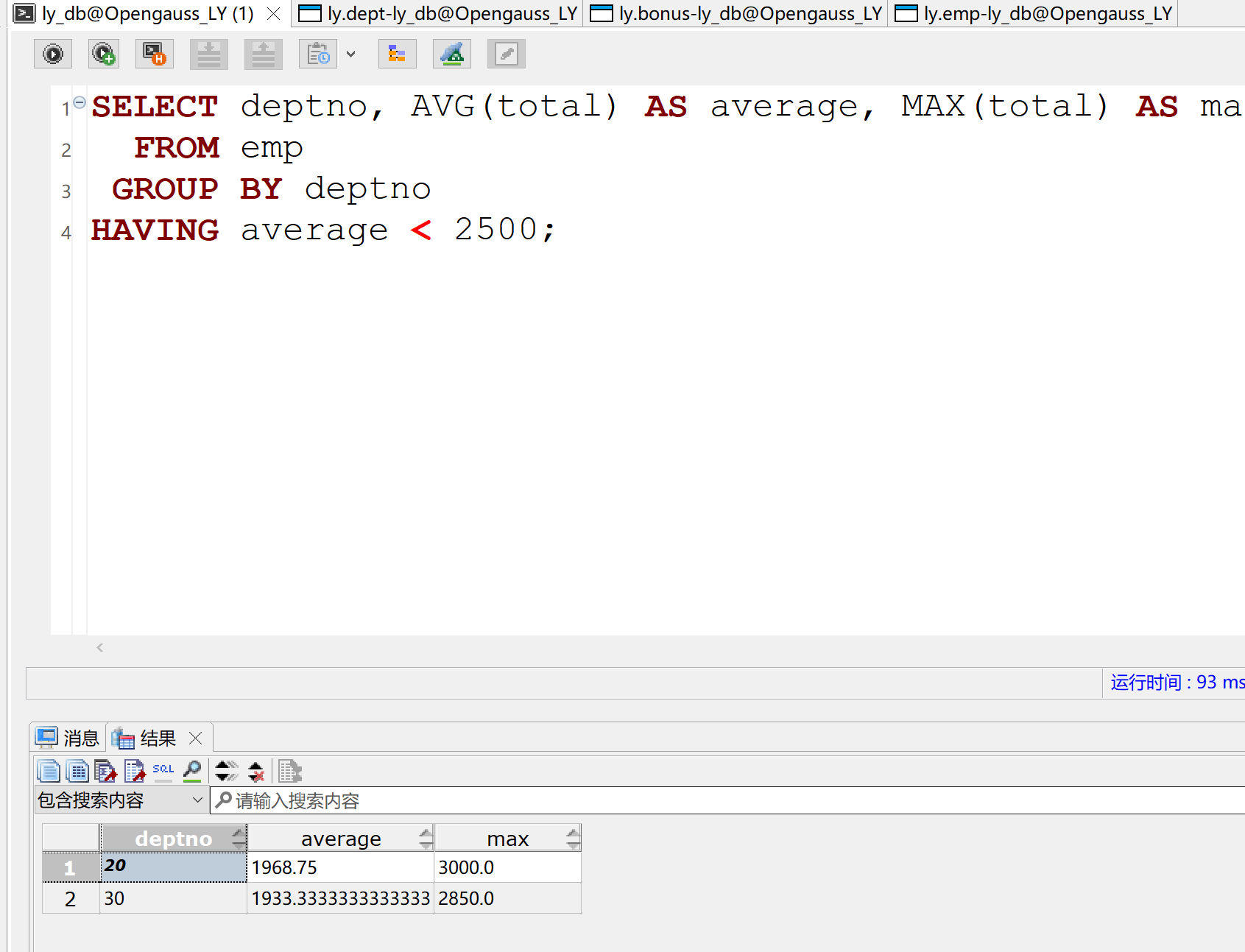
代码：



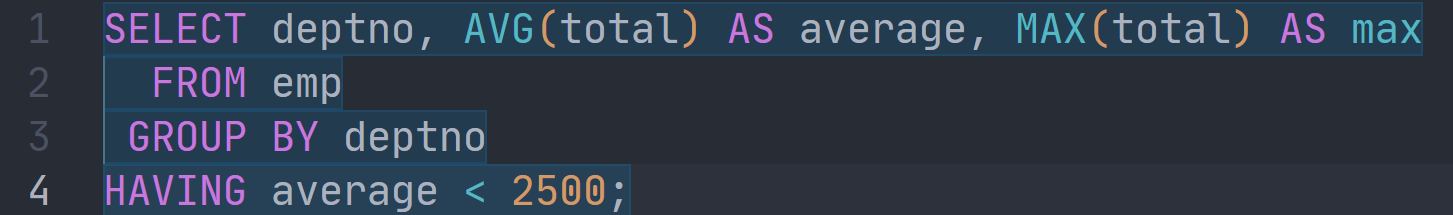
#### 3.3.3 实践题目3

显示平均工资低于2500的部门号，平均工资及最高工资。

结果：



代码：



## 3.4 多表查询

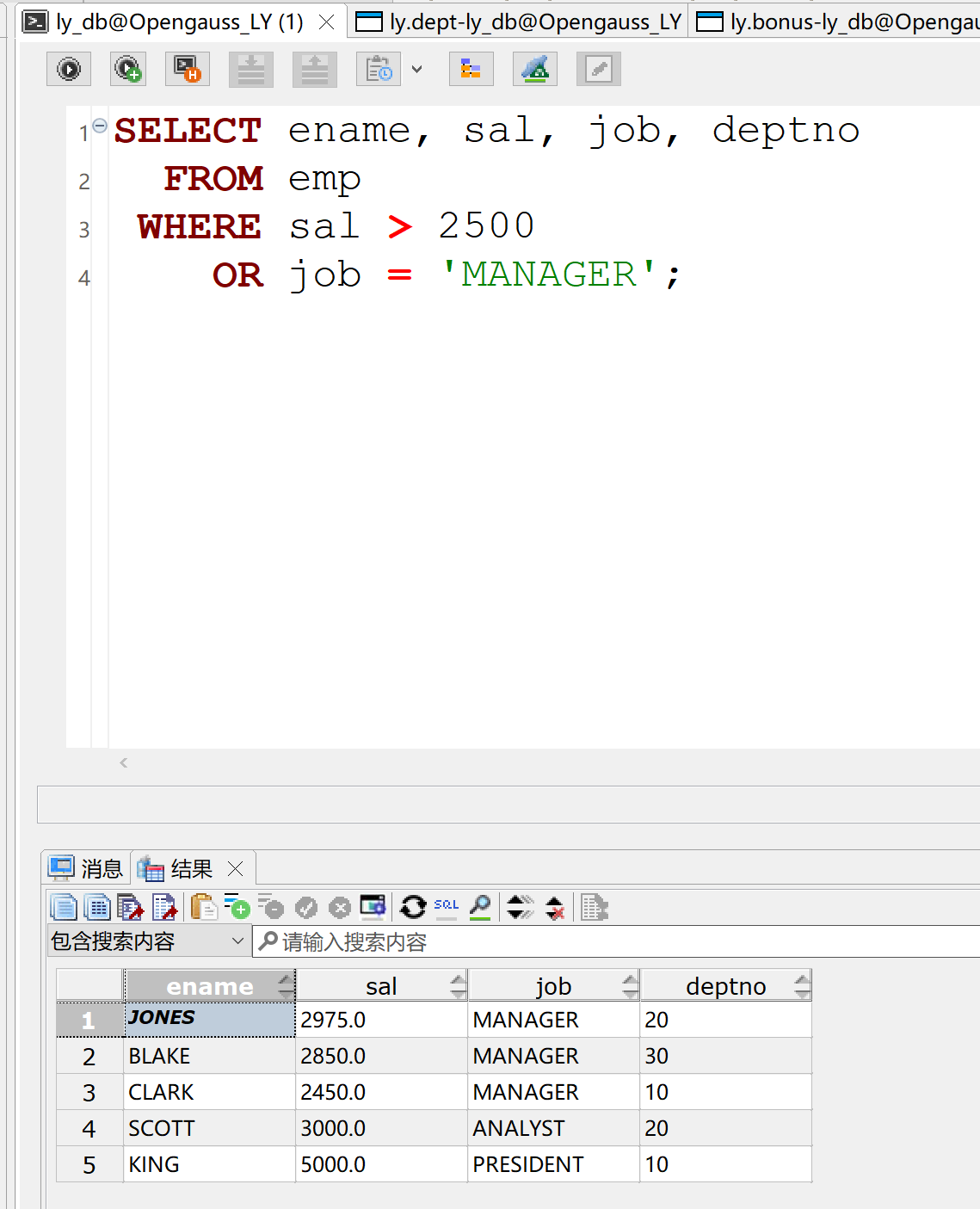
在大部分情况下，我们所需要的信息并不仅仅包含在一张表中。我们首先需要使用join连接多个表，然后再进行查询。

要求：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。

#### 3.4.1 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：



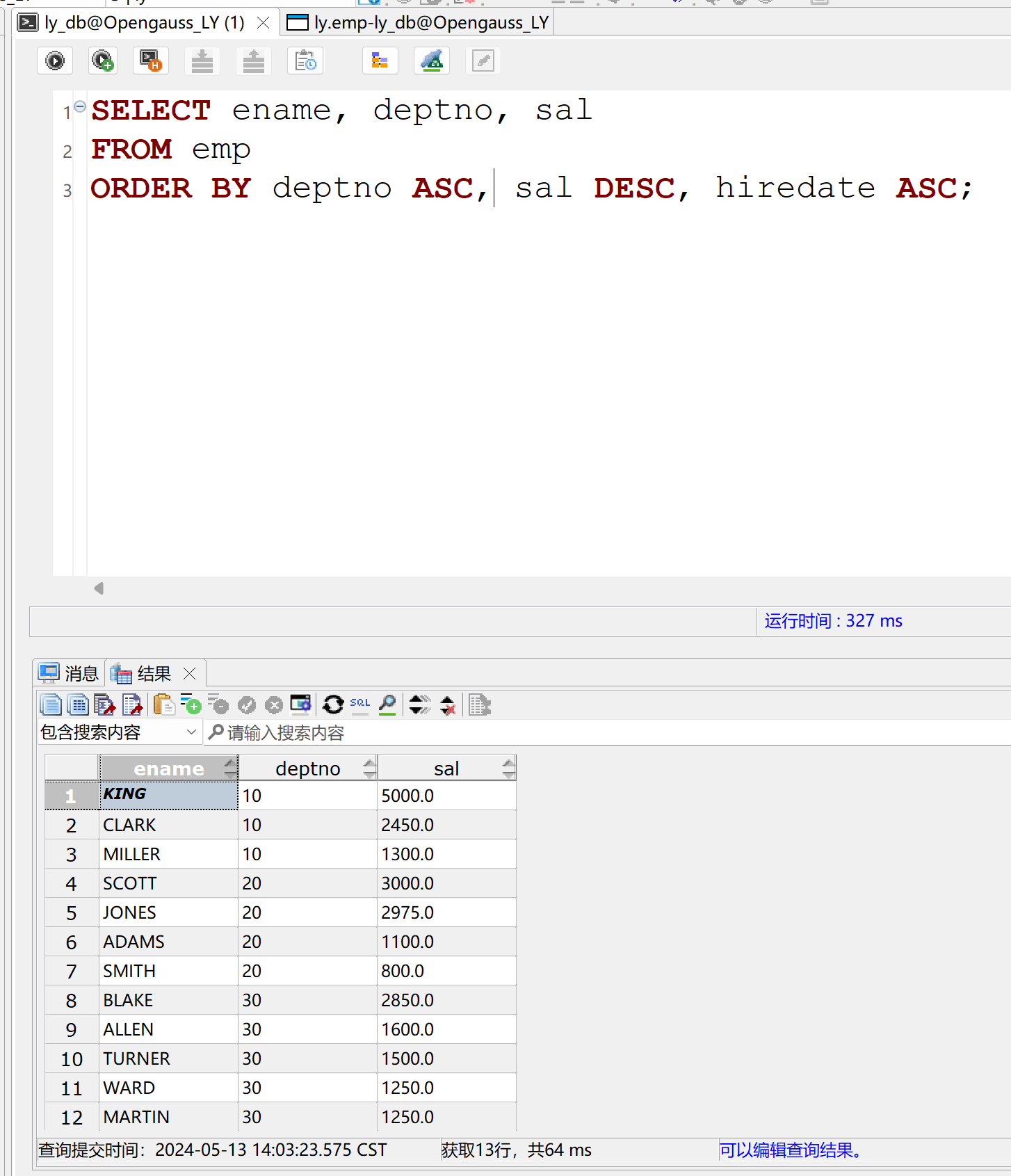
代码：



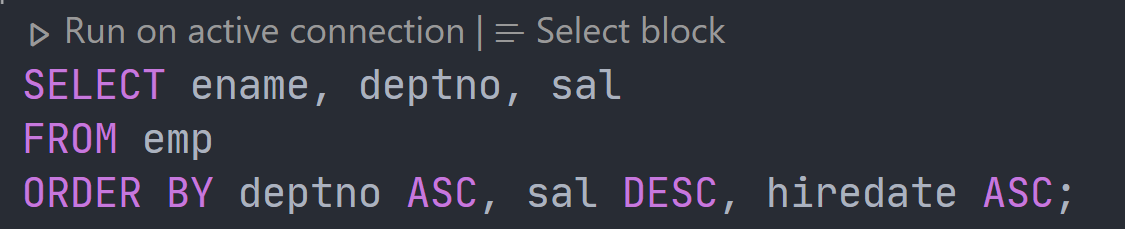
#### 3.4.2 实践题目2

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：



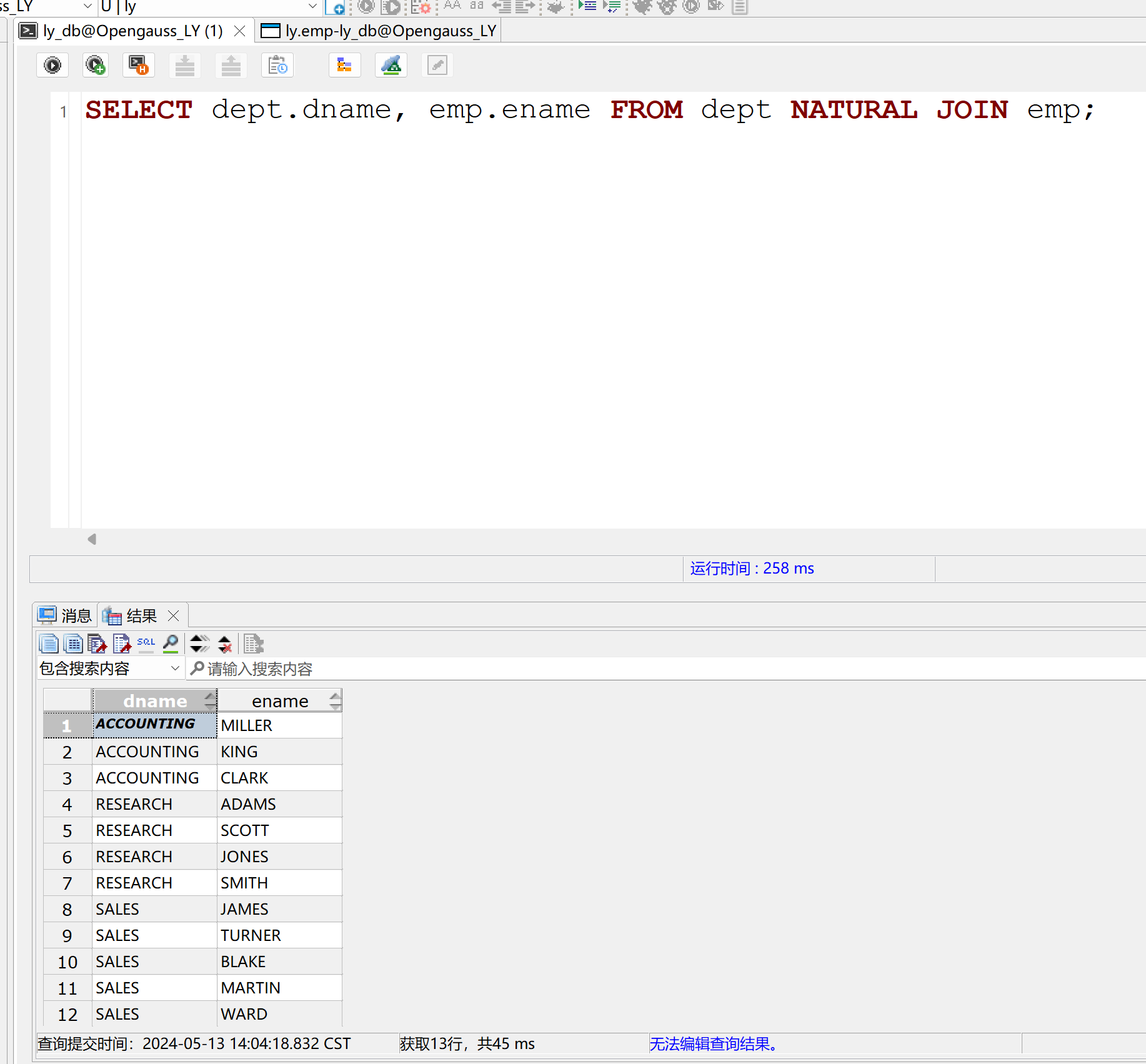
代码：



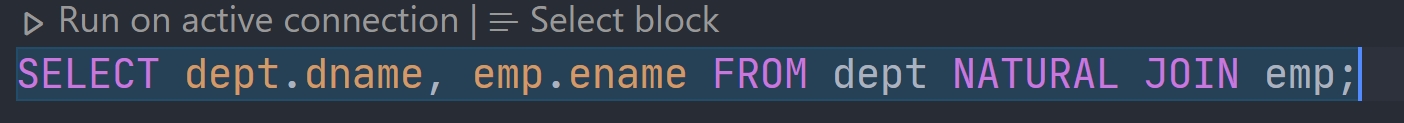
#### 3.4.3 实践题目3

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：



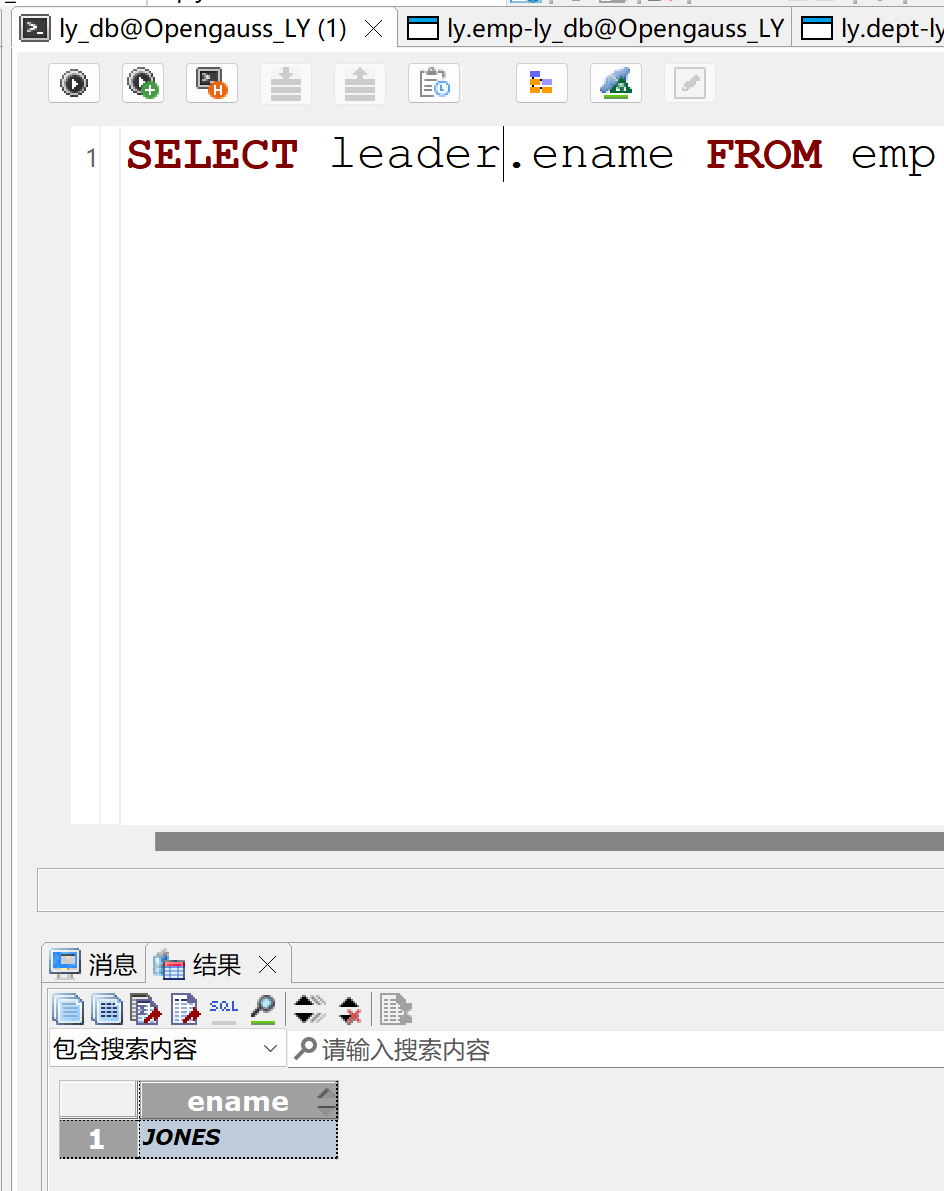
代码：



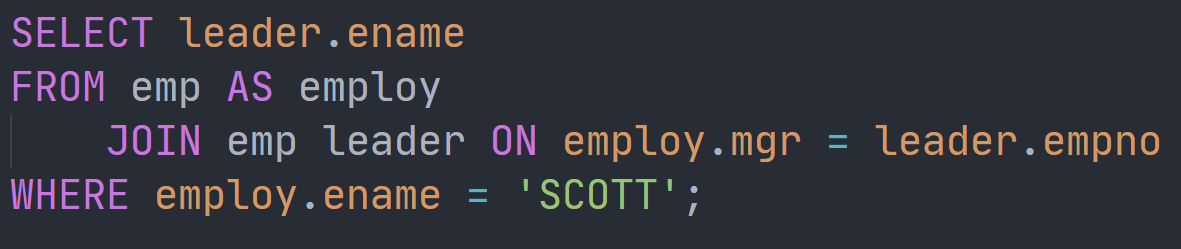
#### 3.4.4 实践题目4

查询SCOTT的上级领导的姓名。

结果：



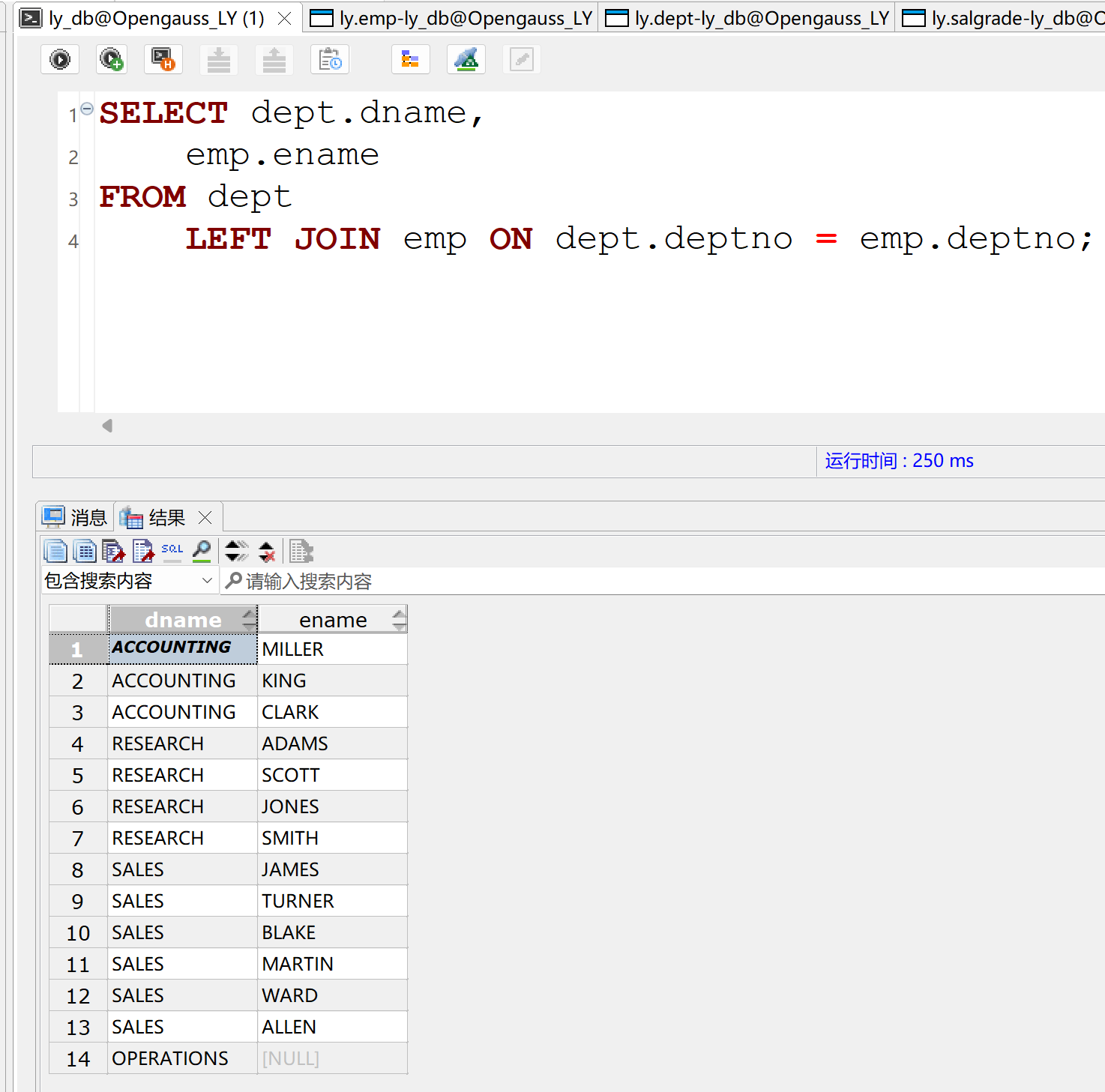
代码：



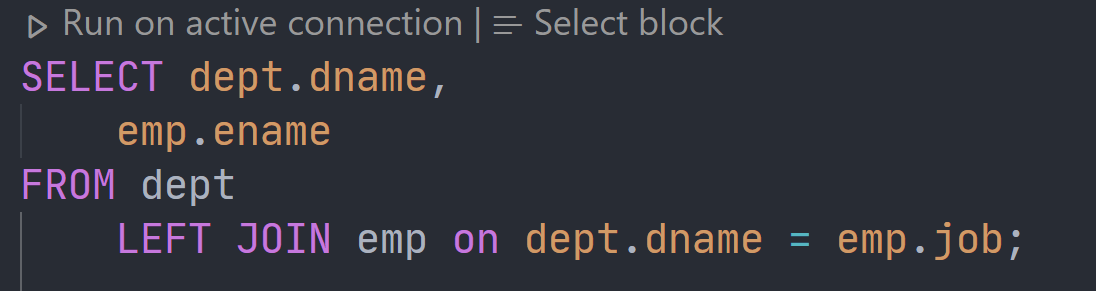
#### 3.4.5 实践题目5

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：



代码：



## 3.5 子查询（必须使用子查询实现）

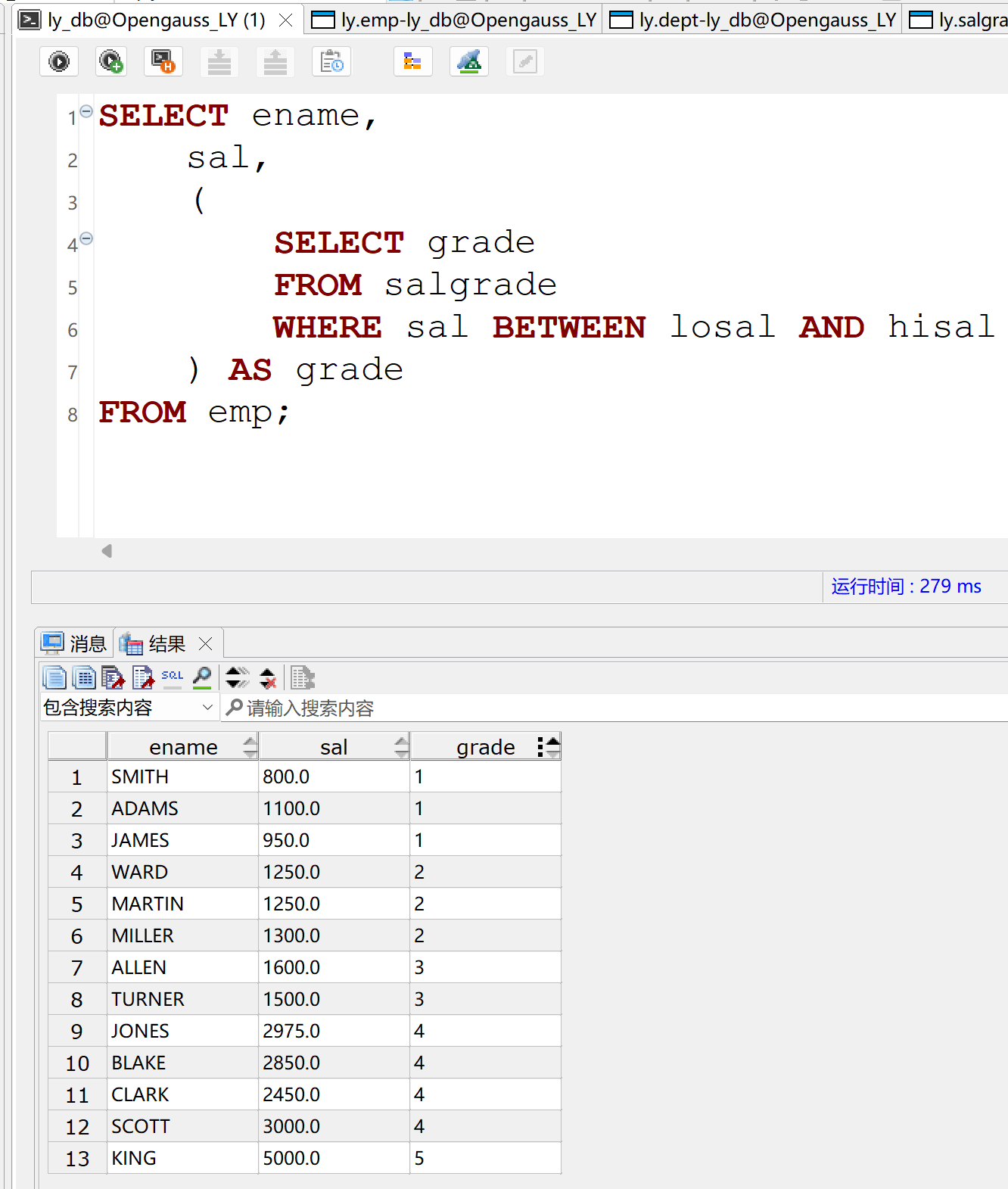
子查询就是指的在一个完整的查询语句之中，嵌套若干个不同功能的小查询，从而一起完成复杂查询的一种编写形式。

要求：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。

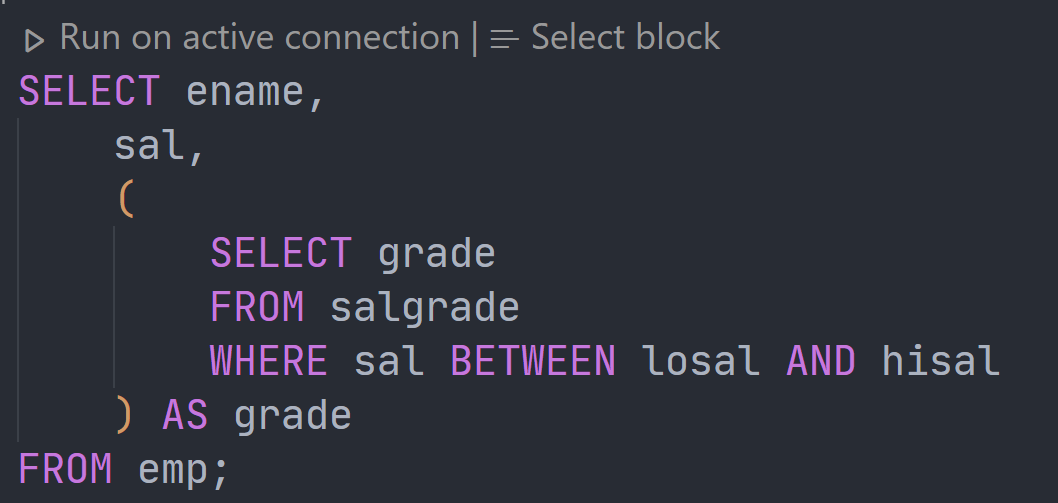
#### 3.5.1 实践题目1

显示所有员工的名称、工资以及工资级别。

结果：



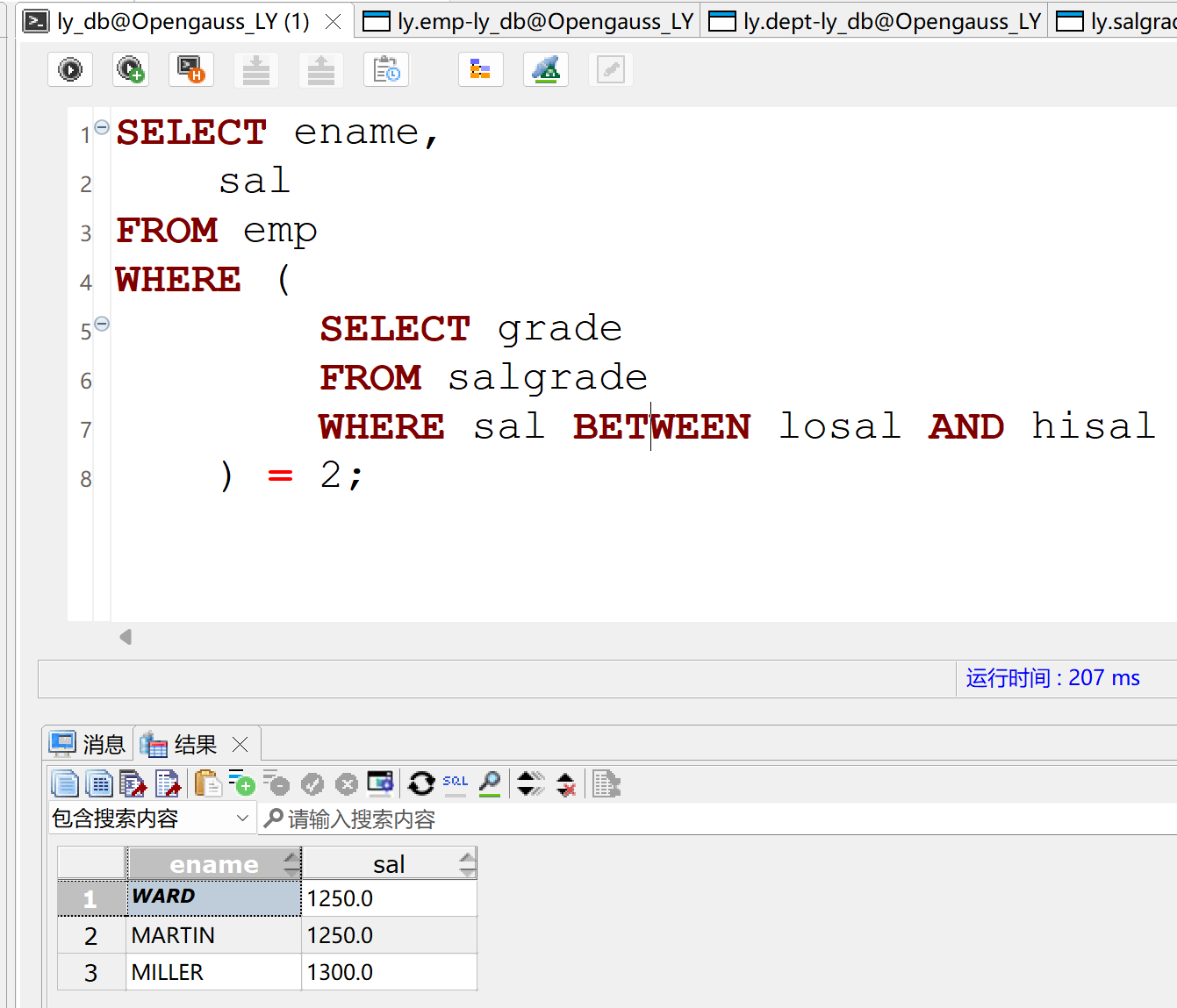
代码：



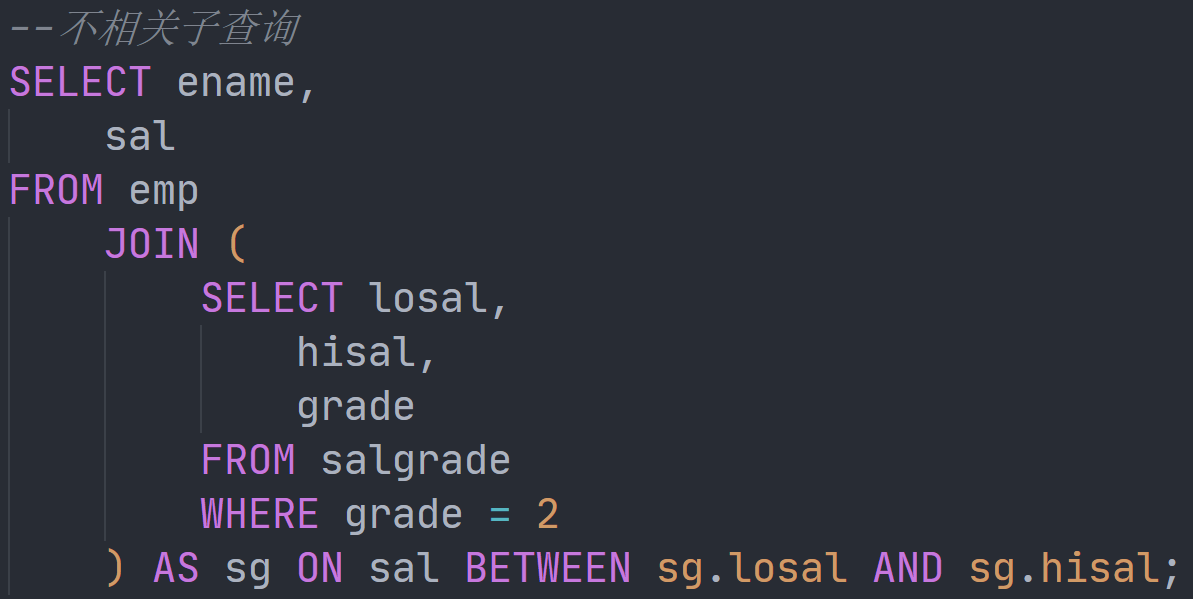
#### 3.5.2 实践题目2

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）

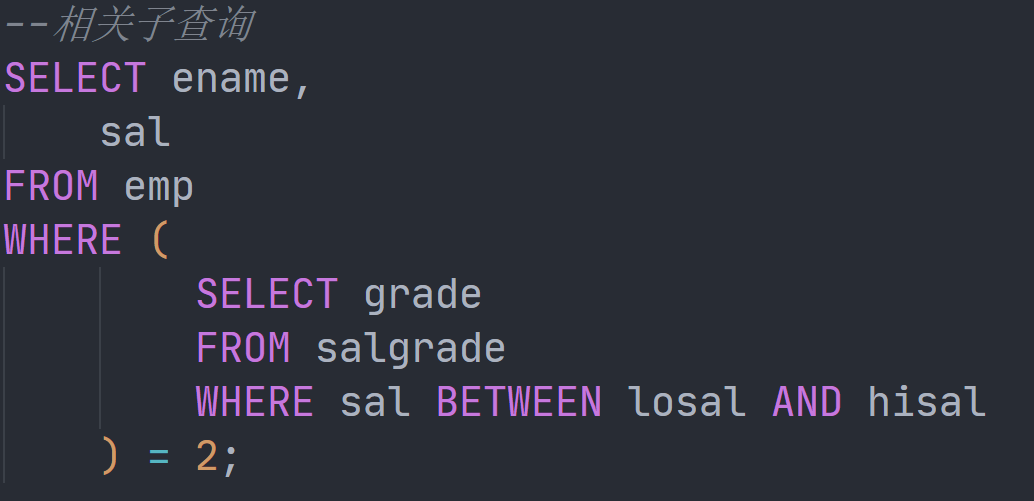
结果：



不相关子查询代码：



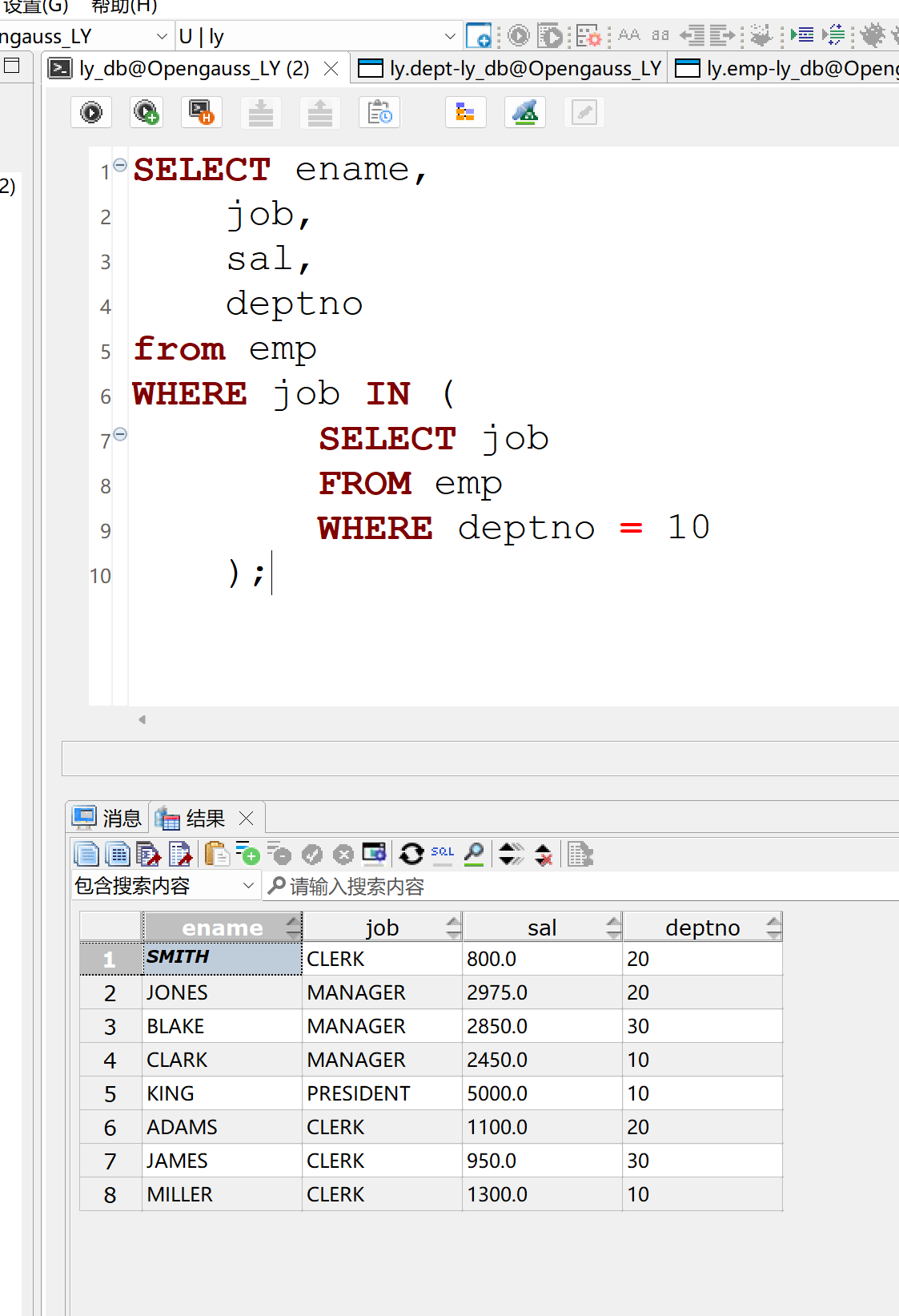
相关子查询代码：



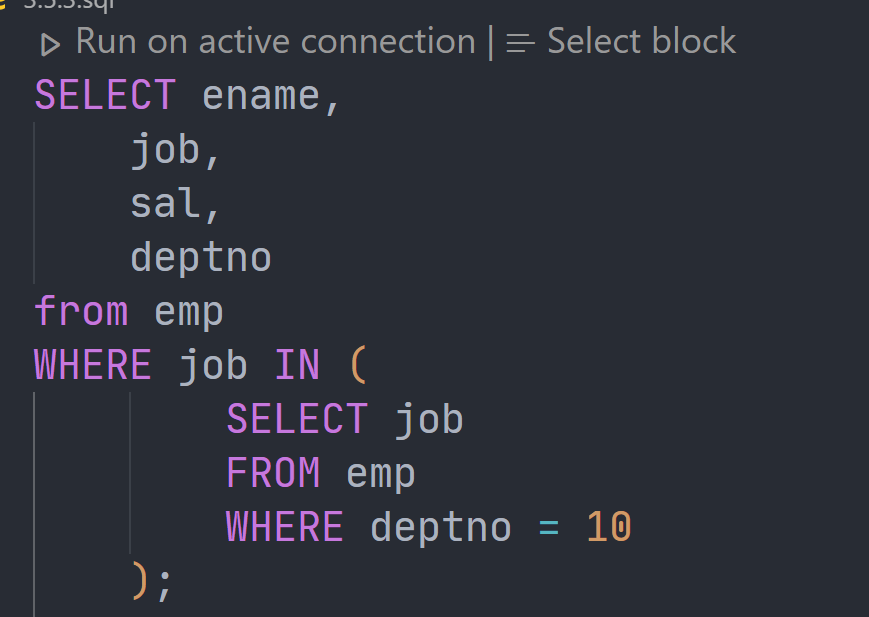
#### 3.5.3 实践题目3

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：



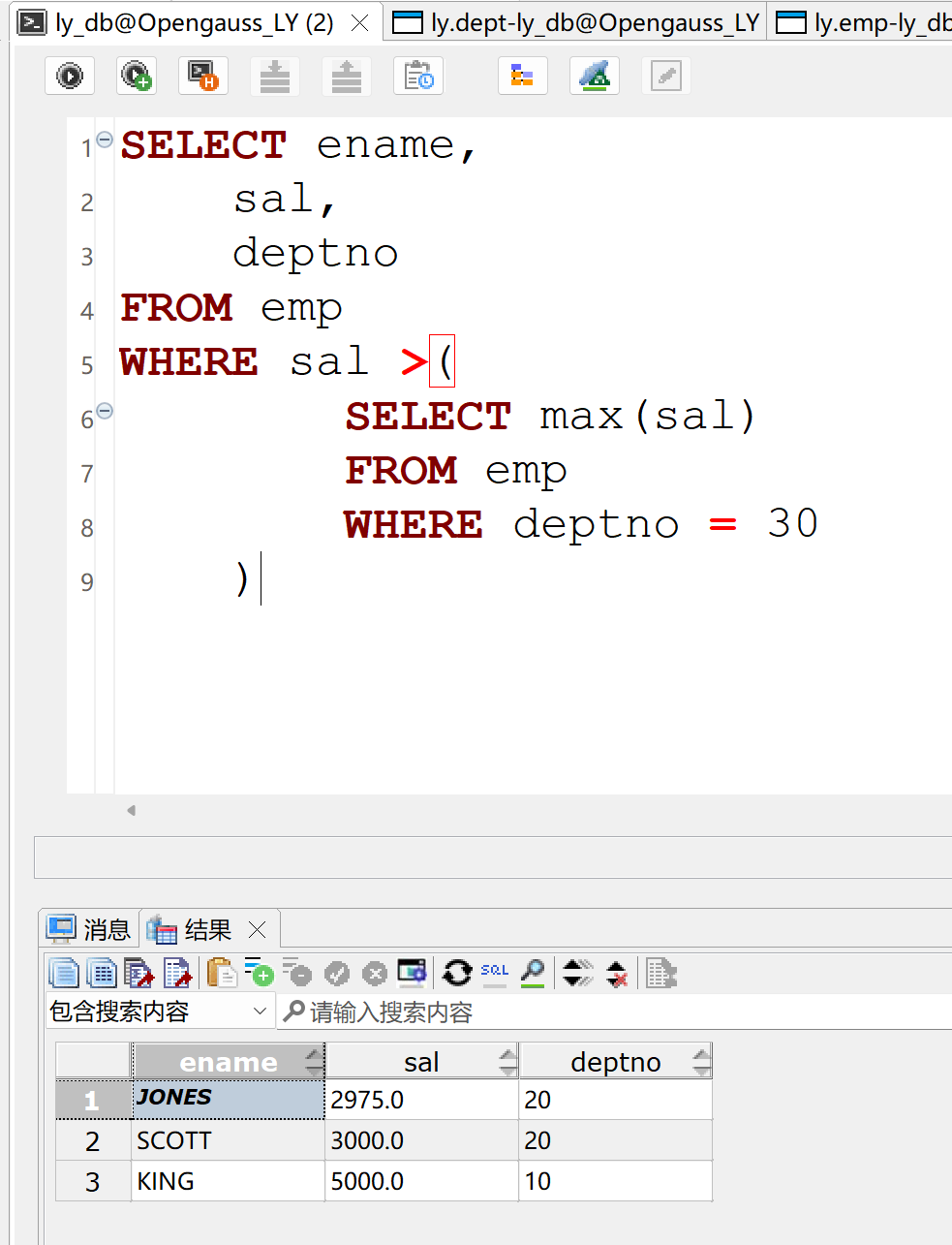
代码：



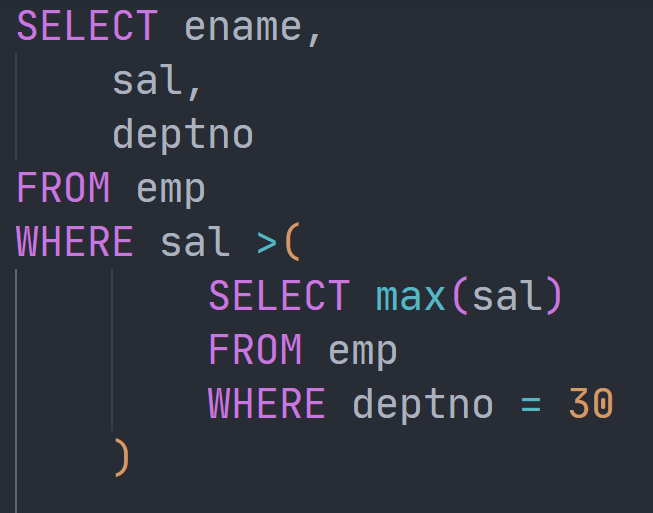
#### 3.5.4 实践题目4

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：



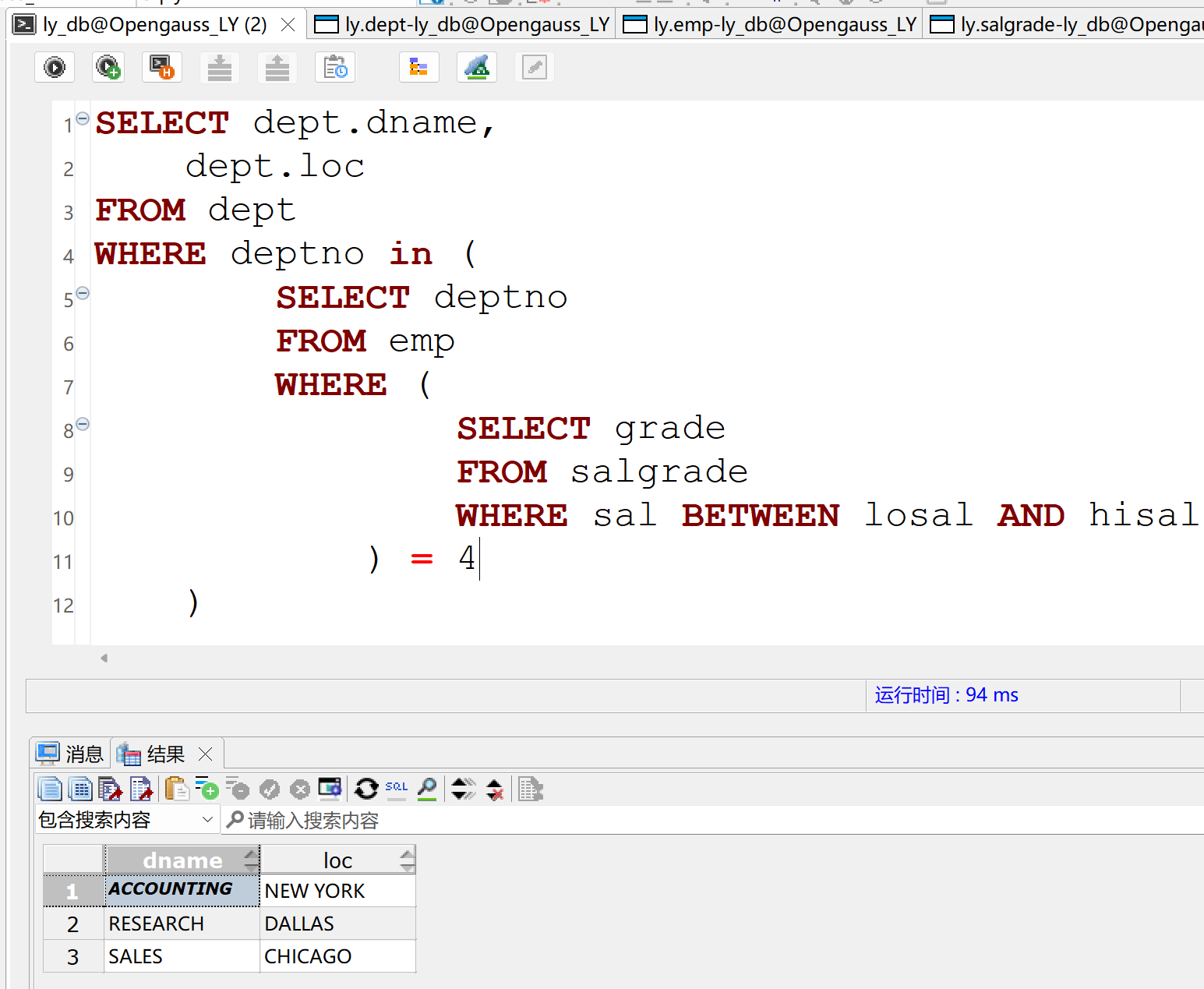
代码：



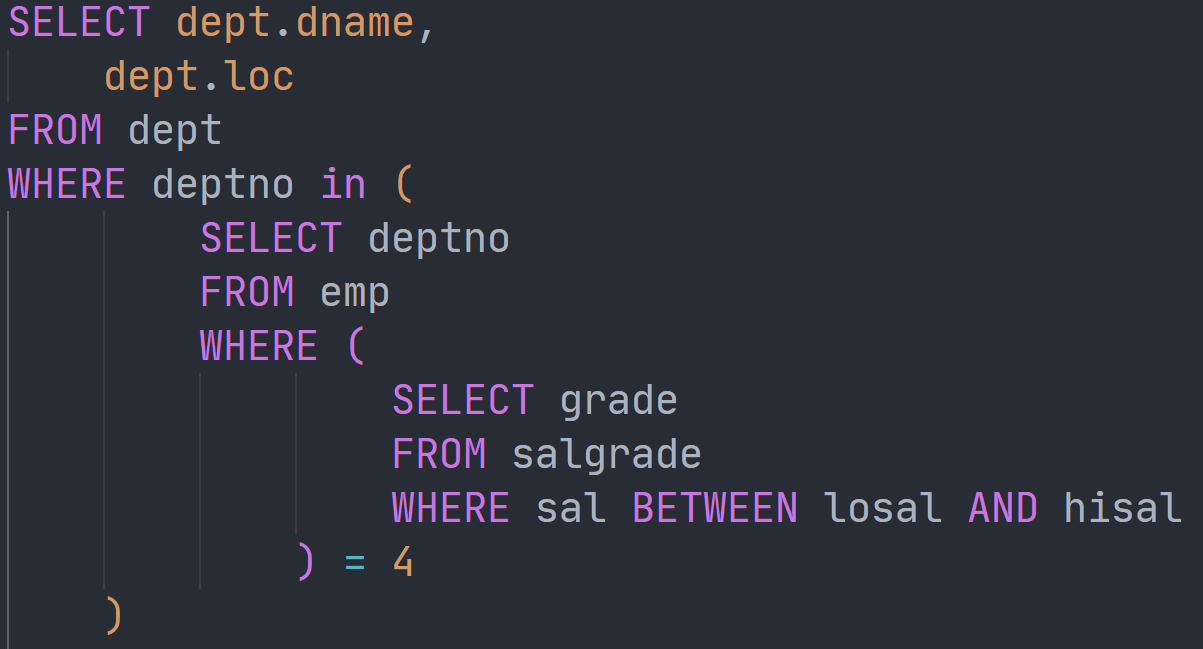
#### 3.5.5 实践题目5

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：



代码：



## 3.6 集合查询

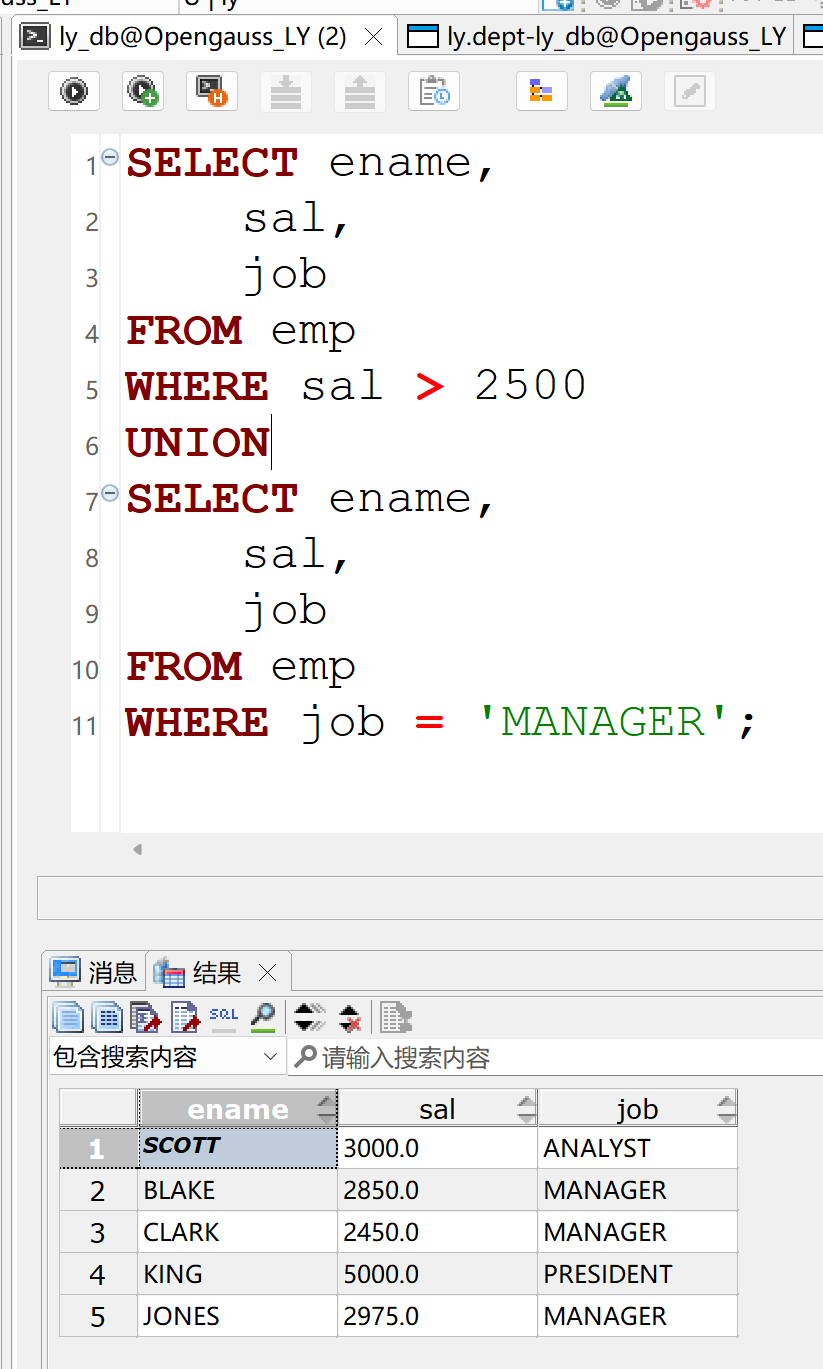
当两张表的属性相同时，我们可以对它们做一些集合运算，如并集、交集等。

要求：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。

#### 3.6.1 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：



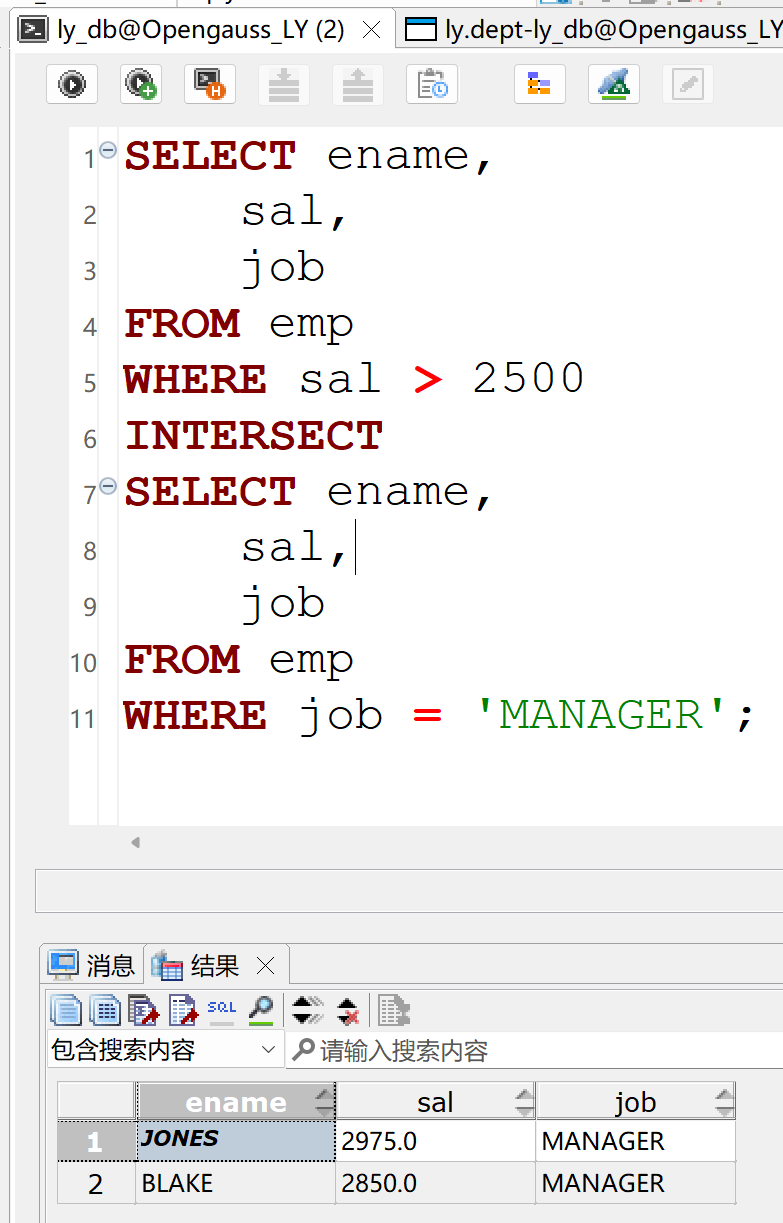
代码：



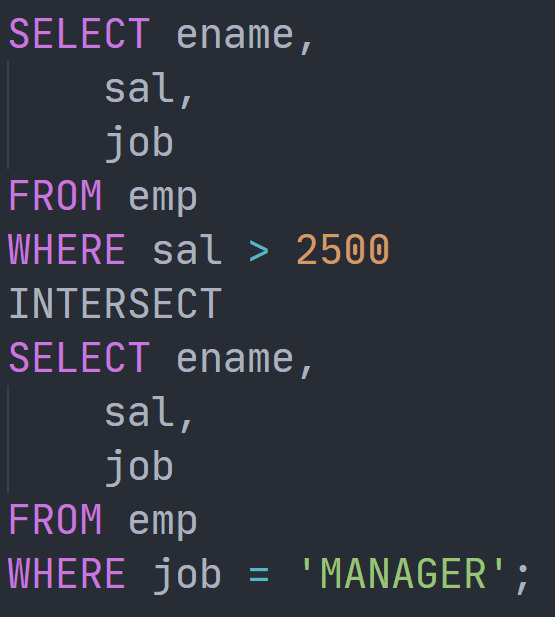
#### 3.6.2 实践题目2

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：



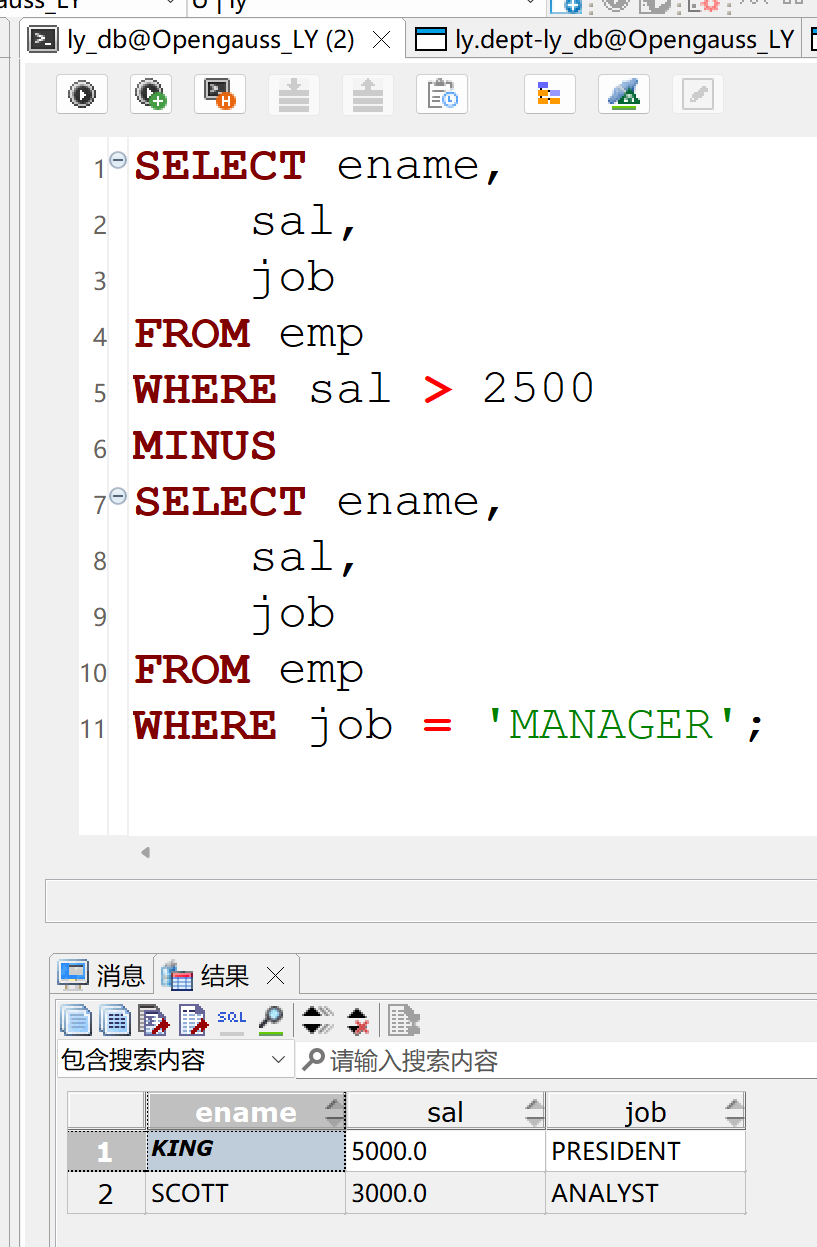
代码：



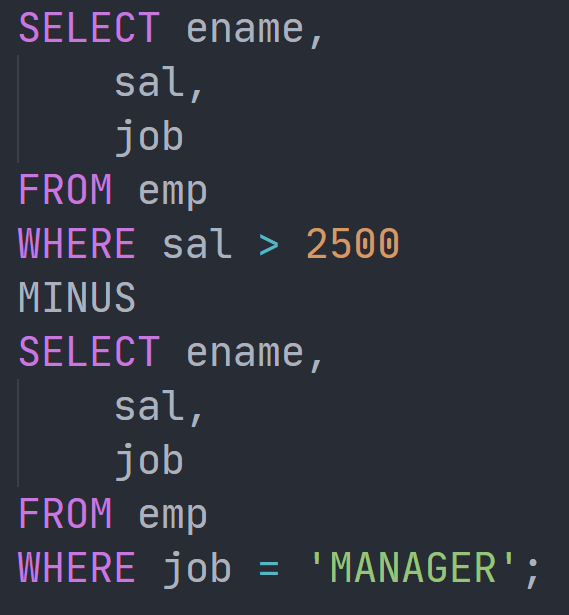
#### 3.6.3 实践题目3

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：



代码：



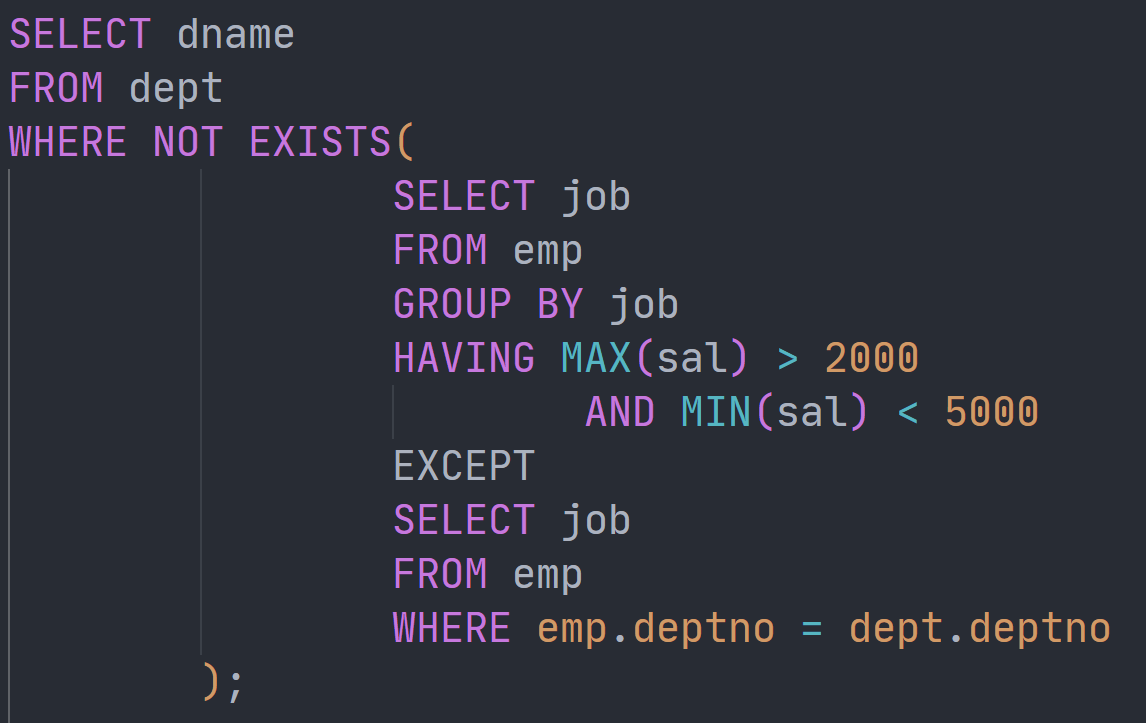
#### 3.6.4 实践题目4

显示提供了工资在2000~5000之间的所有职位的部门名称

结果：



代码：



# 第四章 数据库设计

## 4.1 预备知识

创建数据库应用程序是一项复杂的任务，它涉及数据库架构设计、访问和更新数据的程序设计以及控制数据访问的安全架构设计。用户的需求在设计过程中起着核心作用。对于小型应用程序来说，数据库设计人员如果理解应用程序需求，可以直接决定要创建的关系、它们的属性以及关系的约束。然而，这种直接的设计过程对于现实世界的应用程序来说是困难的，因为它们通常非常复杂。通常情况下，没有人能完全理解应用程序的全部数据需求。数据库设计人员必须与应用程序的用户进行交互，以理解应用程序的需求，并以用户可以理解的高层次方式表示这些需求，然后将这些需求转化为更低层次的设计。一个高级数据模型通过提供一个概念框架，以一种系统的方式来规范数据库用户的需求，以及一个满足这些需求的数据库结构，从而为数据库设计人员提供服务。

数据库设计的第一阶段是全面描述潜在数据库用户的需求。为了完成这项任务，数据库设计人员需要与领域专家和用户进行大量交互。该阶段的成果是对用户需求的规范说明。

接下来，设计人员会选择一个数据模型，并通过应用所选数据模型的概念，将这些需求转化为数据库的概念模式。在**概念设计阶段**开发的模式提供了对整个企业的详细概述。实体关系模型（我们在本章稍后部分将要学习）通常用于表示概念设计。用实体关系模型术语来说，概念模式会指定数据库中表示的实体、实体的属性、实体之间的关系以及实体和关系的约束。通常，概念设计阶段会产生一个实体关系图（ER图），该图提供模式的图形表示。设计人员会审查模式，以确认所有数据需求都确实得到满足并且互不冲突。设计人员还可以检查设计以删除任何冗余的特性。此时，实体关系图的重点在于描述数据及其关系，而不是指定物理存储细节。

完全开发的概念模式还会指明企业的的功能需求。在功能需求的规范说明中，用户会描述将要对数据执行的操作（或事务）类型。示例操作包括修改或更新数据、搜索和检索特定数据以及删除数据。在概念设计阶段的这一步，设计人员可以审查模式以确保它满足功能需求。

将抽象数据模型转换为数据库实现的过程分为最后两个设计阶段：

**逻辑设计阶段**：设计人员将高级的概念模式映射到将要使用的数据库系统的实现数据模型上。实现数据模型通常是关系数据模型，此步骤通常包括将使用实体关系模型定义的概念模式映射到关系模式。

**物理设计阶段**：最后，设计人员会在后续的物理设计阶段中使用生成的特定于系统的数据库模式，其中指定了数据库的物理特性。这些特性包括文件组织形式和索引结构的选择。

与构建应用程序之后相比，数据库的物理模式可以相对容易地进行更改，逻辑模式的更改通常更难进行，因为它们可能会影响散布在应用程序代码中的许多查询和更新。因此，在构建数据库应用程序的其他部分之前，仔细进行数据库设计阶段非常重要。

## 4.2 实验内容

针对以下客户需求设计相应的数据库模式方案，**使用powerdesigner工具完成概念结构设计、逻辑结构设计、物理结构设计三个阶段，并将三个阶段的模式图截图放在实验手册中。**

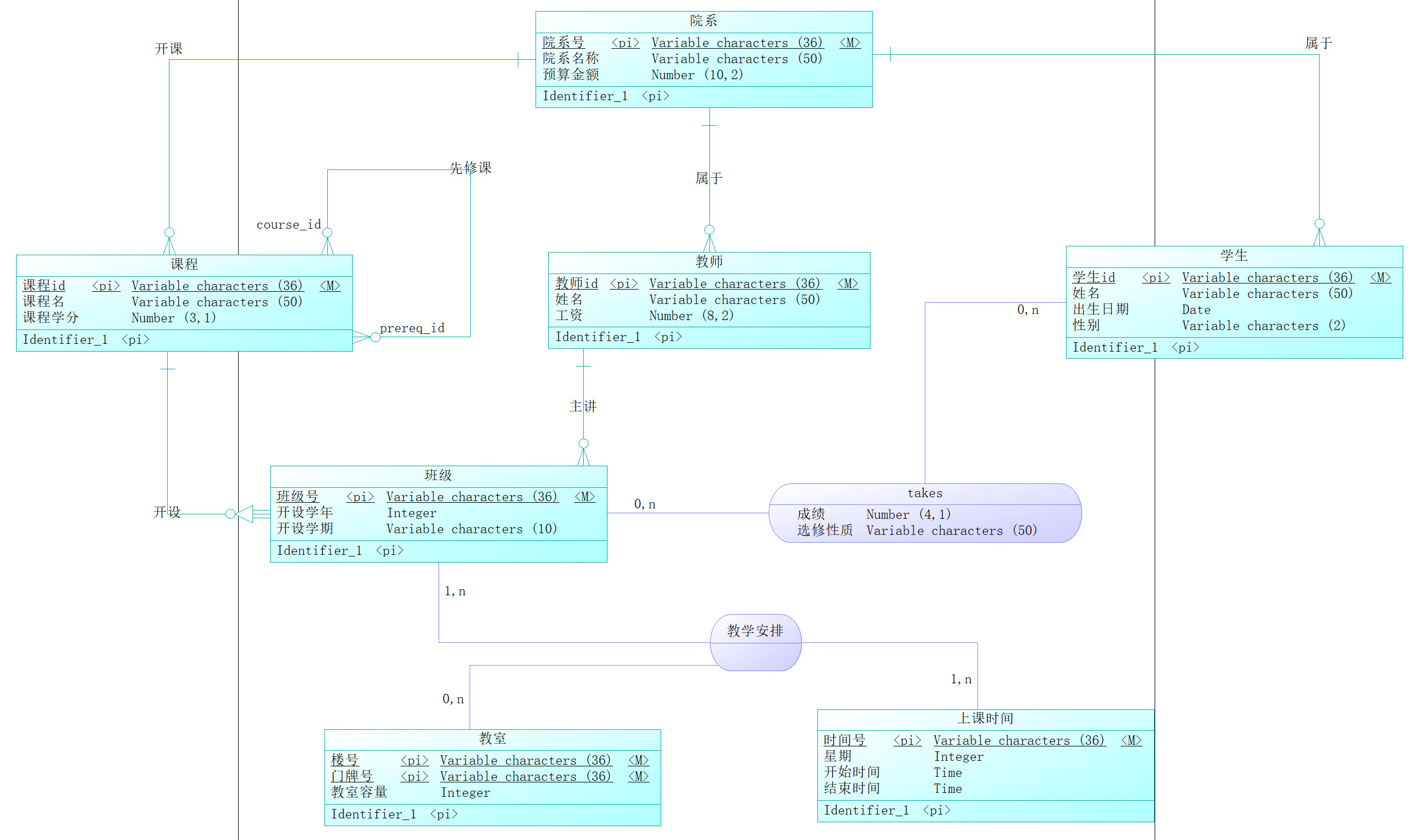
需求描述：

当前需要开发用于管理一个大学的教学信息的系统，此系统需要至少能够管理院系、教师、学生、课程、教学班、教室这些核心实体的基本信息。

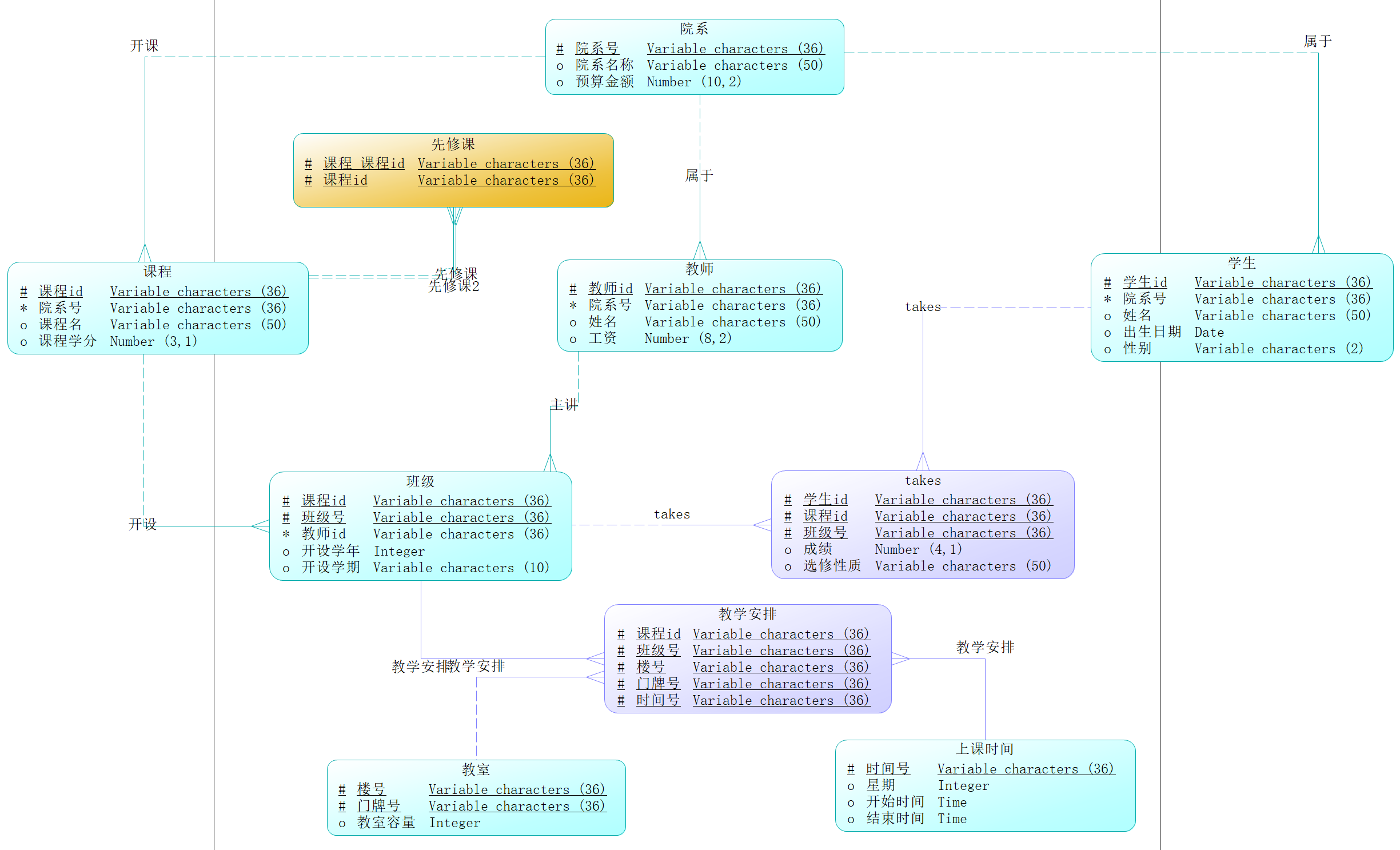
需要能够管理如下核心实体间的联系：

* 在人事关系方面，教师归属于哪个院系；
* 在学籍关系方面，学生属于哪个院系；
* 各个院系负责开设什么课程；
* 每个学期，各个课程开设了哪些教学班；
* 课程之间具有什么先修支撑联系；
* 哪个教师在哪个学期主讲了哪个教学班；
* 哪个学生在哪个学期选修了哪个教学班的课；
* 哪个学生在那个学期选修哪个课程的成绩是多少，选修的性质是正常还是重修；
* 在教学安排方面，每个教学班使用的时间安排及教室安排；

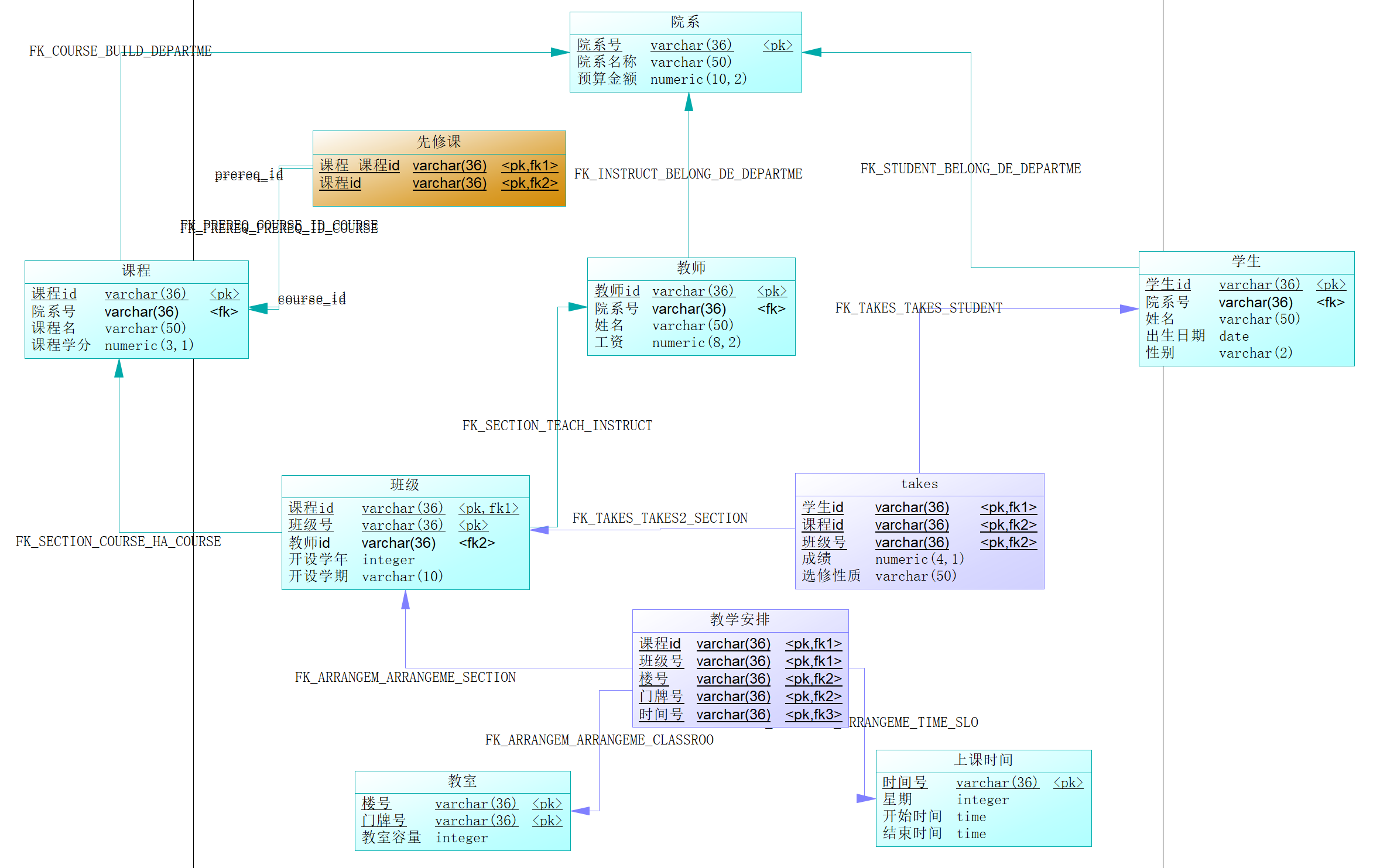
**概念结构设计结果：**



**逻辑结构设计结果：**



**物理结构设计结果：**



# 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。



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**;

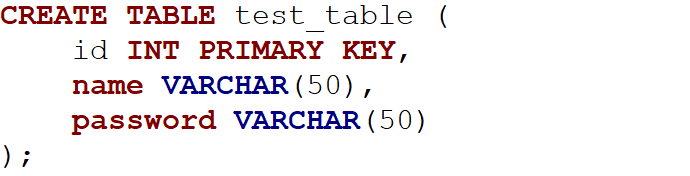
## 5.2实验任务

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**：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。

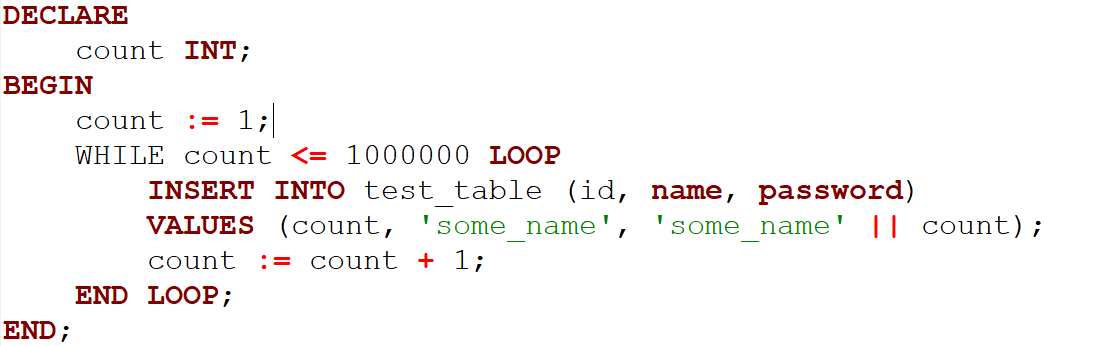
**1. 创建一个表**

首先，创建一个包含三个字段的表 test\_table：



**2. 向表中循环插入大量数据**

使用 PL/SQL 代码向表中插入 1,000,000 条数据：



**3. 对某个属性建立索引**

这里我选择在 name 属性上建立索引：



**4. 对比实验**

执行两次相同的查询任务，一次不使用索引，一次使用索引，并对比查询时间。

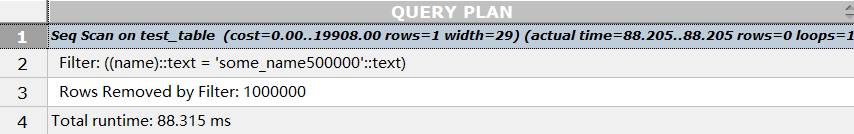
**查询不使用索引（禁用索引）**

首先，禁用索引：



然后执行查询并记录时间：





时间：

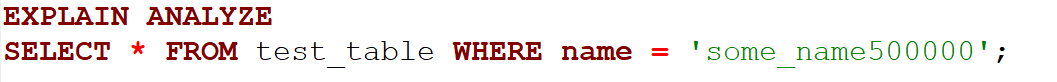


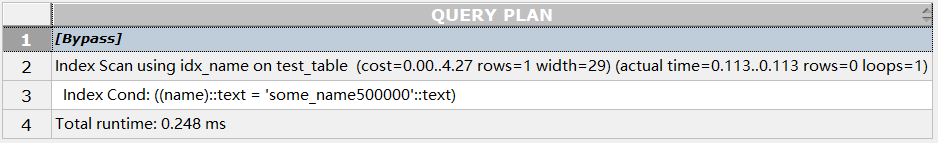
**查询使用索引（启用索引）**

启用索引：



然后执行相同的查询并记录时间：





时间：

**总结**

通过对比明显可以看出索引在提高查询效率方面的效果

# 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。

1. non-repeatable read(不可重复读)，同一事务中，前后读取的数据不一致。
2. phantom read(幻读)，类似不可重复读，但针对插入/删除操作。

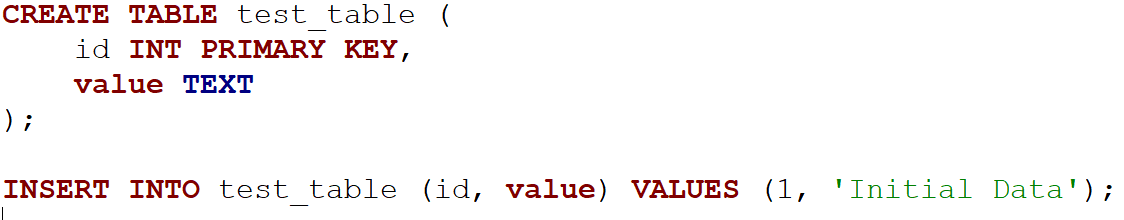
美国国家标准协会在SQL标准中，对于满足事务的隔离性的程度划分为以下四个级别。未提交读、已提交读、可重复读和序列化。不同的隔离级别能够解决的上述问题也不同。

## 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。

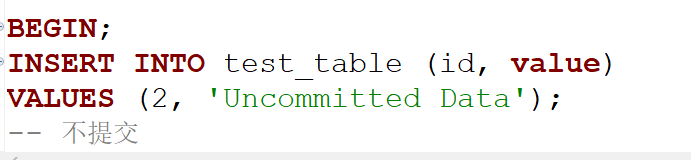
**注**：设计两个事务的并发操作场景，判断是否存在对应问题。需要详细的实验设计过程描述和实验结果的截图。

**1.创建测试表并插入初始数据**

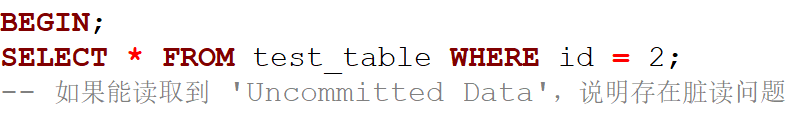


**2.验证脏读**

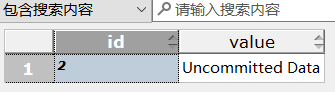
**事务A**：插入新记录但不提交



**事务B**：读取新记录



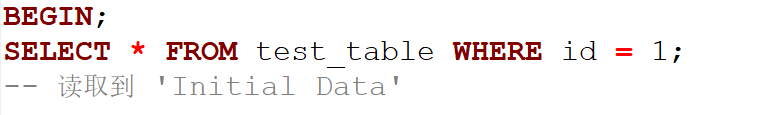
**实验结果截图：**



可见存在脏读问题

**3.验证不可重复读**

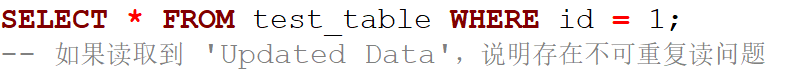
**事务A**：第一次读取记录



**事务B**：更新记录并提交



**事务A**：第二次读取记录



**实验结果截图：**



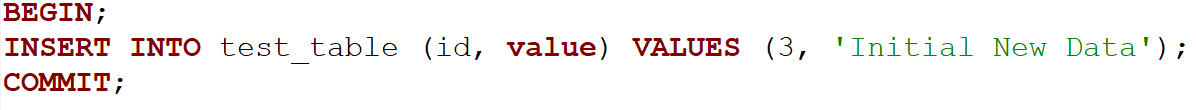
说明存在不可重复读问题

**4.验证幻读**

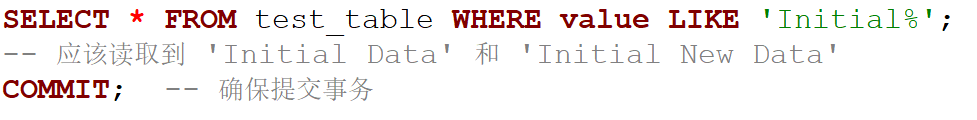
**事务A**：第一次读取满足条件的记录



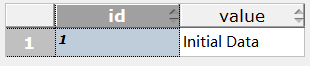
**事务B**：插入满足条件的新记录并提交

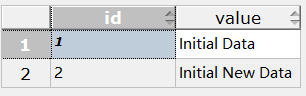


**事务A**：第二次读取满足条件的记录



**实验结果截图：**





可以看出存在幻读现象

# 第七章 用户权限设置及回收

## 7.1 预备知识

数据库发展早期,访问控制通常可以分为自主访问控制（Discretionary AccessControl， DAC）和强制访问控制（Mandatory Access Control, MAC）。在自主访问控制模式下，用户是数据对象的控制者，用户依据自身的意愿决定是否将自己的对象访问权或部分访问权授予其他用户。而在强制访问控制模式下，对特定用户指定授权,用户不能将权限转交给他人。在实际应用中，DAC模式太弱，MAC又太强，且两者工作量较大，不便于管理。基于角色的访问控制机制（Role-Based Access Control, RBAC）是一种更加灵活的机制，可以作为传统访问控制机制（DAC、MAC）的代替，也是较为有效的管理方法。

openGauss 继承了业界目前通用的权限管理模型，实现了基于角色的访问控制机制。整个机制中的核心概念是“角色”，其更深层次的含义是角色组，即角色所拥有的权限实际上对应着这个角色组中所有成员的权限。管理员只需要将管理所希望的权限赋给角色，用户再从角色继承相应的权限即可，而无须对用户进行单一管理。当管理员需要增加和删减相关的权限时，角色组内的用户成员也会自动继承权限变更。

基于角色管理模型,用户可具备对对象的访问操作权限,并基于此完成数据管理而这些用户所具备的权限是会经常发生变化的，为了有效地防止诸如权限提升、利用权限漏洞进行恶意操作等行为，必须进行权限的合理管控，即对象权限管理。更重要的是，需要在对象被访问操作时对当前用户的合法权限进行有效性检查，即对象权限检查。

**在openGauss内核中，用户和角色是基本相同的两个对象，区别在于：1）创建角色时默认没有登录权限，而创建用户时包含了登录权限；2）创建用户时，系统会默认创建个与之同名的schema,用于该用户进行对象管理。因此在权限管理实践中，建议通过角色进行权限的管理，通过用户进行数据的管理。**

用户权限管理的场景主要包含：1）将系统权限授权给角色或用户；2）将数据库对象授权给角色或用户；3）将角色或用户的权限授权给其他角色或用户。在本次实验开始阶段已经完成用户的创建，并赋予了所有权限。因此，本章节将对其他两个场景进行实践。

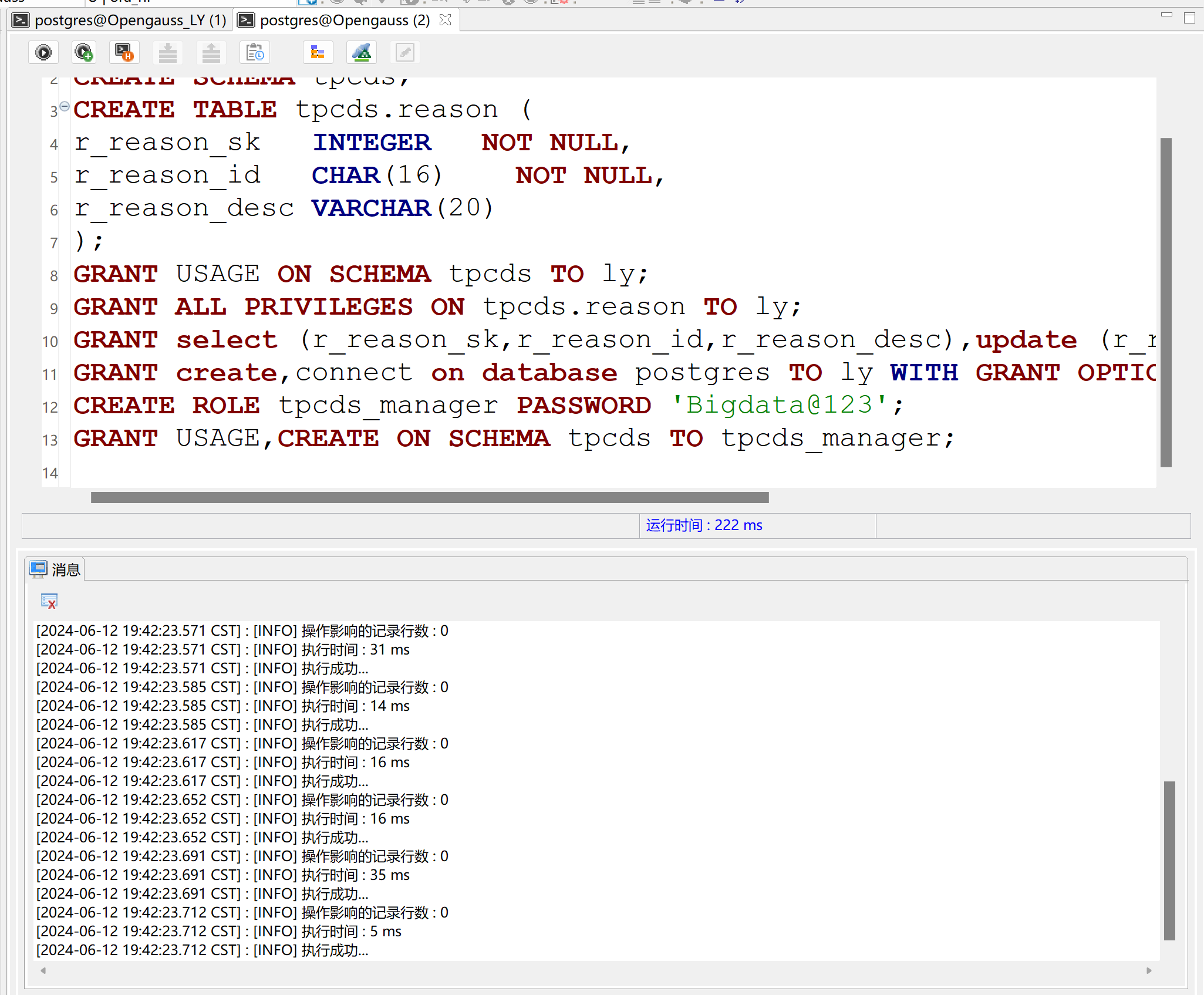
实验准备：

1. 需要切换回系统的管理员用户（gaussdb）连接到数据库。
2. 以在第一章创建的用户为后续实验的操作对象。下面用LWZ作为示例说明。

## 7.2 将数据库对象授权给角色或用户

任务：完成下述操作，并将操作过程截屏。

截图：



步骤1，撤销LWZ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。

回收权限：

REVOKE ALL PRIVILEGES FROM LWZ;

创建模式：

CREATE SCHEMA tpcds;

创建表：

CREATE TABLE tpcds.reason (

r\_reason\_sk INTEGER NOT NULL,

r\_reason\_id CHAR(16) NOT NULL,

r\_reason\_desc VARCHAR(20)

);

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LWZ。

给用户授权：

GRANT USAGE ON SCHEMA tpcds TO LWZ;

给用户授权：

GRANT ALL PRIVILEGES ON tpcds.reason TO LWZ;

授权成功后，LWZ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LWZ。

给用户授权：

GRANT select (r\_reason\_sk,r\_reason\_id,r\_reason\_desc),update (r\_reason\_desc) ON tpcds.reason TO LWZ;

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

给用户授权：

GRANT create,connect on database postgres TO LWZ WITH GRANT OPTION;

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。

创建角色：

CREATE ROLE tpcds\_manager PASSWORD 'Bigdata@123';

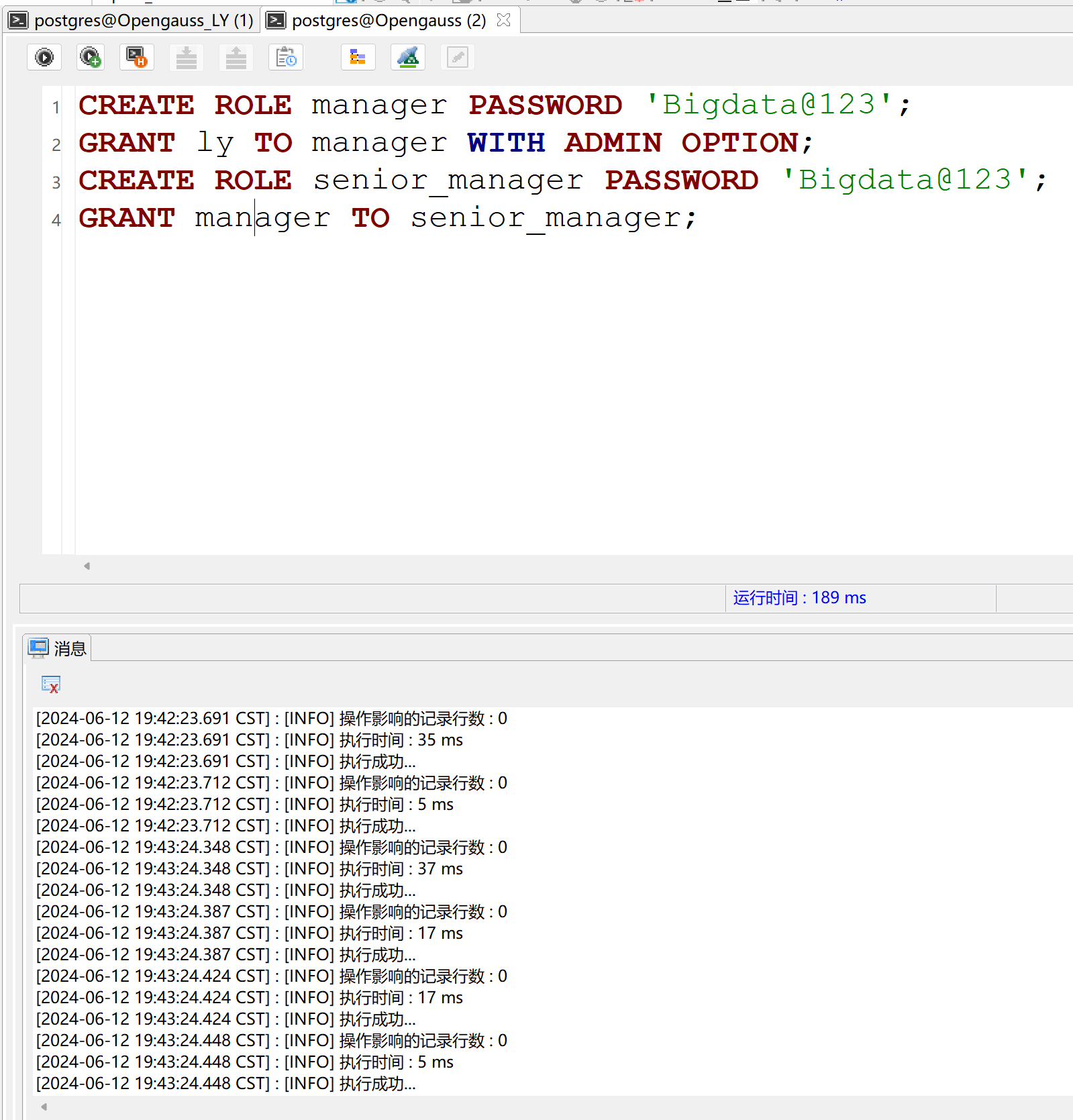
给角色授权：

GRANT USAGE,CREATE ON SCHEMA tpcds TO tpcds\_manager;

## 7.3 将用户或者角色的权限授权给其他用户或角色

任务：完成下述操作，并将操作过程截屏。

截图：



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

创建角色：

CREATE ROLE manager PASSWORD 'Bigdata@123';

给角色授权：

GRANT LWZ TO manager WITH ADMIN OPTION;

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

创建角色：

CREATE ROLE senior\_manager PASSWORD 'Bigdata@123';

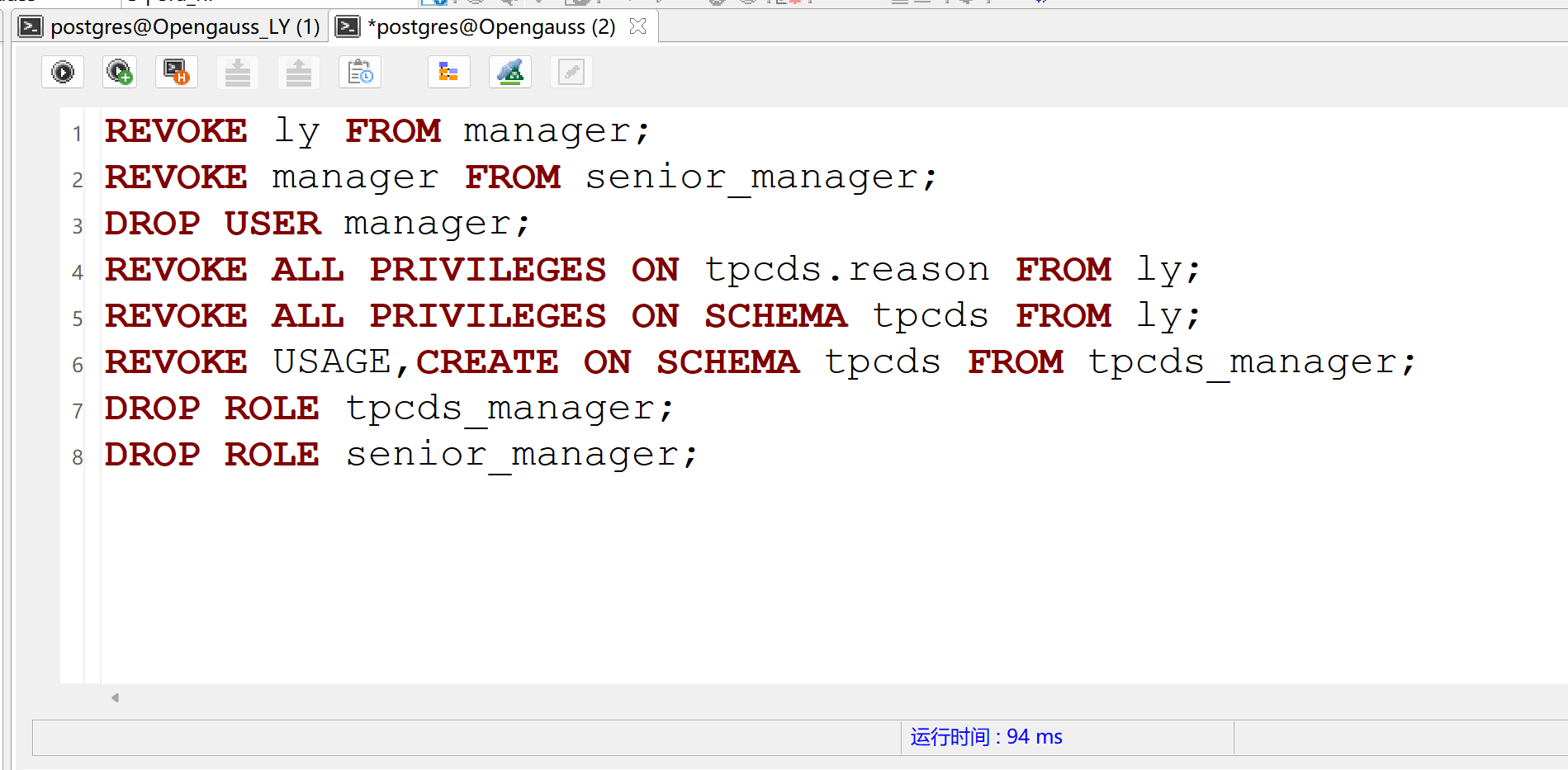
给角色授权：

GRANT manager TO senior\_manager;

## 7.4 权限回收

任务：完成下述操作，并将操作过程截屏。

截图：



步骤1，撤销权限，并清理用户。

撤销用户/角色权限：

REVOKE LWZ FROM manager;

REVOKE manager FROM senior\_manager;

删除用户/角色：

DROP USER manager;

撤销用户/角色权限：

REVOKE ALL PRIVILEGES ON tpcds.reason FROM LWZ;

REVOKE ALL PRIVILEGES ON SCHEMA tpcds FROM LWZ;

REVOKE USAGE,CREATE ON SCHEMA tpcds FROM tpcds\_manager;

删除用户/角色：

DROP ROLE tpcds\_manager;

DROP ROLE senior\_manager;

DROP USER LWZ CASCADE;

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

在完成《数据库系统》课程的上机实验后，我对数据库的理论知识和实际操作有了更深入的理解和体会

在完成实验手册的过程中，我积极利用各种学习资源，包括教材、学习通课程视频、在线 Sql 语句手册等，这些资源帮助我更好地理解实验内容和操作步骤。通过参考 OpenGauss 官方文档，我对 OpenGauss 数据库的高级特性有了更深入的了解

在进行复杂查询尤其是多表连接和嵌套子查询时，我遇到了不少困难。理解不同类型的连接（如 INNER JOIN、LEFT JOIN 等）及其适用场景，编写复杂的查询语句需要花费较多时间和精力。通过反复练习和调试，我逐渐掌握了这些技巧，但仍需在实践中不断积累经验。

通过完成该 Word 文档，我掌握了 OpenGauss 数据库的基本操作和使用 Data Studio 数据库管理工具来管理数据库的方法，还培养了实际操作能力和问题解决能力。尽管在学习过程中遇到了一些困难，但通过不断的努力和实践，我逐渐克服了这些难点，并取得了显著的进步。希望在未来的学习中，能够进一步深入学习数据库系统的高级特性，提升自己的专业能力