



华中科技大学

数据库系统概论实验报告

姓 名：袁 也
学 院：网络空间安全学院
专 业：网络空间安全专业
班 级：网安 1902 班
学 号：U201911808
指导教师：路松峰

分数	
教师	

2021 年 12 月 24 日

目 录

1	课程任务概述	1
1.1	课程概述.....	1
1.2	任务简介.....	1
2	数据库定义与基本操作	2
2.1	任务要求.....	2
2.2	完成过程.....	2
2.3	任务总结.....	13
3	SQL 的复杂操作	15
3.1	任务要求.....	15
3.2	完成过程.....	15
3.3	任务总结.....	20
4	SQL 的高级实验	21
4.1	任务要求.....	21
4.2	完成过程.....	21
4.3	任务总结.....	31
5	数据库设计	32
5.1	任务要求.....	32
5.2	完成过程.....	32
5.3	任务总结.....	44
6	课程总结	45
A	附录.....	46
A.1	数据库管理系统源码.....	46

1 课程任务概述

1.1 课程概述

本实验是数据库系统概论的实验课，数据库系统是一个偏向应用的学科，在学习这门课的过程中，最重要的就是讲究理论与实操相结合。在课上的理论学习之后，更需要掌握对数据库的使用，以及数据库 SQL 语言的使用的各项功能。只有经过了实验的考核，才能说是掌握了数据库。也是对理论知识更加全面的了解和掌握。

1.2 任务简介

本课程共有 16 个课时，分为四次实验：

1. 数据库定义与基本操作：

掌握 DBMS 的数据定义功能：掌握 SQL 语言的数据定义语句：掌握 DBMS 的数据单表查询功能：掌握 SQL 语言的数据单表查询语句：

具体内容：创建数据库：创建、删除表：查看、修改表的定义：理解索引的特点：创建和删除索引：SELECT 语句的基本用法：使用 WHERE 子句进行有条件的查询：使用 IN，NOT IN，BETWEEN AND 等谓词查询：利用 LIKE 子句实现模糊查询：使用 ORDER BY 子句为结果排序：使用 SQL Server/MySQL 的聚集函数进行统计计算：使用 GROUP BY 子句实现分组查询的方法。

2. SQL 的复杂操作：

掌握 SQL 语言的数据多表查询语句和更新操作；

具体内容：等值连接查询（含自然连接查询）与非等值连接查询：自身连接查询：外连接查询；复合条件连接查询；嵌套查询（带有 IN 谓词的子查询；嵌套查询（带有比较运算符的子查询）；嵌套查询（带有 ANY 或 ALL 谓词的子查询）；嵌套查询（带有 EXISTS 谓词的子查询）；集合查询：update 语句用于对表进行更新：delete 语句用于对表进行删除：insert 语句用于对表进行插入。

3. SQL 的高级实验：

掌握 SQL 语言的视图、触发器、存储过程、安全等功能：

具体内容：创建表的视图：利用视图完成表的查询：删除表的视图：创建触发器：创建存储过程：对用户进行授权和查询：理解用户定义完整性。

4. 数据库设计：

通过一个数据库具体设计实例，掌握数据库设计的方法。

2 数据库定义与基本操作

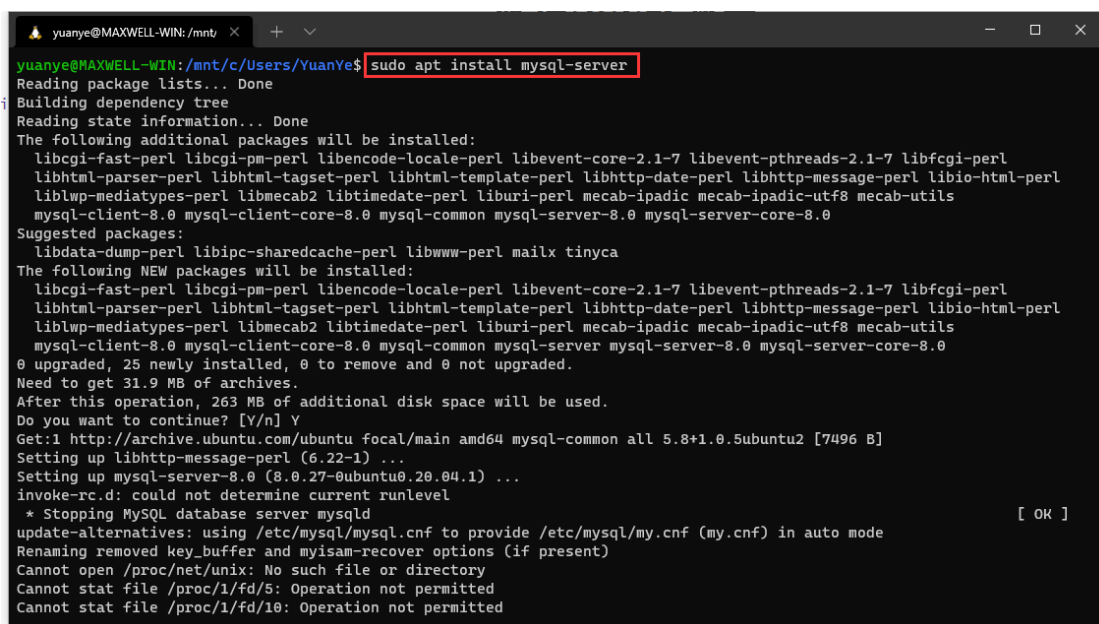
2.1 任务要求

- (1) 熟练掌握SQL的数据定义语句CREATE、ALTER、DROP、Select
- (2) 写出实验报告

2.2 完成过程

2.2.1 安装数据库

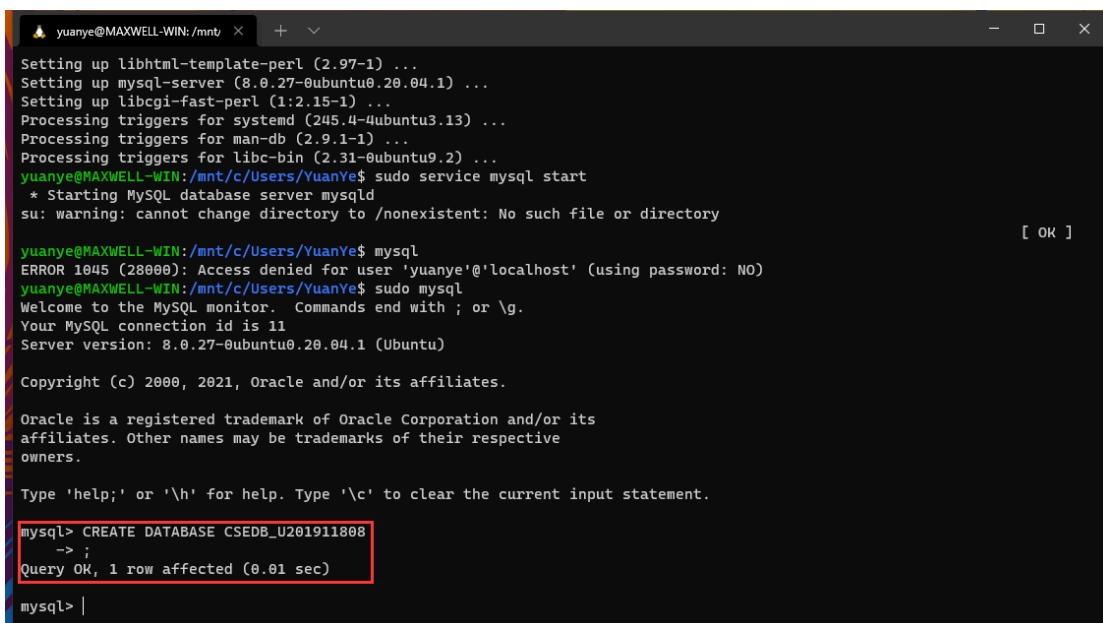
- 1) 安装并启动数据库，如图 2.1 所示：



```
yuanye@MAXWELL-WIN: /mnt/ x + v
yuanye@MAXWELL-WIN:/mnt/c/Users/YuanYe$ sudo apt install mysql-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libbgi-fast-perl libbgi-pm-perl libencode-locale-perl libevent-core-2.1-7 libevent-pthreads-2.1-7 libfcgi-perl
  libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl
  liblwp-mediatypes-perl libmecab2 libtimedate-perl liburi-perl mecab-ipadic mecab-ipadic-utf8 mecab-utils
  mysql-client-8.0 mysql-client-core-8.0 mysql-common mysql-server-8.0 mysql-server-core-8.0
Suggested packages:
  libdata-dump-perl libipc-sharedcache-perl libwww-perl mailx tinyca
The following NEW packages will be installed:
  libbgi-fast-perl libbgi-pm-perl libencode-locale-perl libevent-core-2.1-7 libevent-pthreads-2.1-7 libfcgi-perl
  libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl
  liblwp-mediatypes-perl libmecab2 libtimedate-perl liburi-perl mecab-ipadic mecab-ipadic-utf8 mecab-utils
  mysql-client-8.0 mysql-client-core-8.0 mysql-common mysql-server mysql-server-8.0 mysql-server-core-8.0
0 upgraded, 25 newly installed, 0 to remove and 0 not upgraded.
Need to get 31.9 MB of archives.
After this operation, 263 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 mysql-common all 5.8+1.0.5ubuntu2 [7496 B]
Setting up libhttp-message-perl (6.22-1) ...
Setting up mysql-server-8.0 (8.0.27-0ubuntu0.20.04.1) ...
invoke-rc.d: could not determine current runlevel
* Stopping MySQL database server mysqld
update-alternatives: using /etc/mysql/mysql.cnf to provide /etc/mysql/my.cnf (my.cnf) in auto mode
Renaming removed key_buffer and myisam-recover options (if present)
Cannot open /proc/net/unix: No such file or directory
Cannot stat file /proc/1/fd/5: Operation not permitted
Cannot stat file /proc/1/fd/10: Operation not permitted
[ OK ]
```

图 2.1 安装并启动数据库

- 2) 创建名为 CSEDB_U201911808 的数据库，如图 2.2 所示：



```
yuanye@MAXWELL-WIN: /mnt/ x + v
Setting up libhtml-template-perl (2.97-1) ...
Setting up mysql-server (8.0.27-0ubuntu0.20.04.1) ...
Setting up libbgi-fast-perl (1:2.15-1) ...
Processing triggers for systemd (245.4-4ubuntu3.13) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
yuanye@MAXWELL-WIN:/mnt/c/Users/YuanYe$ sudo service mysql start
* Starting MySQL database server mysqld
su: warning: cannot change directory to /nonexistent: No such file or directory
[ OK ]
yuanye@MAXWELL-WIN:/mnt/c/Users/YuanYe$ mysql
ERROR 1045 (28000): Access denied for user 'yuanye'@'localhost' (using password: NO)
yuanye@MAXWELL-WIN:/mnt/c/Users/YuanYe$ sudo mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.27-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE CSEDB_U201911808
-> ;
Query OK, 1 row affected (0.01 sec)

mysql>
```

图 2.2 创建数据库

2.2.2 基本表操作

- 3) 在数据库中按照要求创建三个表，如图 2.3 所示：

```
mysql> use CSEDB_U201911808
Database changed
mysql> CREATE TABLE Student
-> (Sno CHAR(5) NOT NULL UNIQUE,
-> Sname CHAR(20) UNIQUE,
-> Ssex CHAR(1),
-> Sage INT,
-> Sdept CHAR(15),
-> Scholarship CHAR(2));
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Course
-> (Cno CHAR(5) NOT NULL UNIQUE,
-> Cname CHAR(30) UNIQUE,
-> Cpno CHAR(20),
-> Ccredit FLOAT
-> );
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE SC
-> (Sno CHAR(5),
-> Cno CHAR(5),
-> Grade INT,
-> PRIMARY KEY(Sno, Cno));
Query OK, 0 rows affected (0.02 sec)

mysql> show tables;
+-----+
| Tables_in_CSEDB_U201911808 |
+-----+
| Course                      |
| SC                          |
| Student                     |
+-----+
3 rows in set (0.00 sec)
```

图 2.3 创建三个表

- 4) 使用 DROP 语句删除表，如图 2.4 所示：

```
mysql> DROP TABLE Student;
Query OK, 0 rows affected (0.01 sec)
```

图 2.4 删除表

- 5) 练习创建和删除索引操作，如图 2.5 所示：

```
mysql> CREATE UNIQUE INDEX CourseNo ON Course(Cno);
Query OK, 0 rows affected, 1 warning (0.02 sec)
Records: 0 Duplicates: 0 Warnings: 1

mysql> DROP INDEX CourseNo;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1
```

图 2.5 创建和删除索引

- 6) 使用 ALTER 指令，在表格中增加一列，如图 2.6 所示：

```
mysql> ALTER TABLE Course ADD Cstart DATETIME;
Query OK, 0 rows affected (0.02 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

图 2.6 表格增加一列

2.2.3 删除数据库

- 7) 使用 DROP 语句删除数据库，如图 2.7 所示:

```
mysql> DROP DATABASE CSEDB_U201911808
-> ;
Query OK, 1 row affected (0.01 sec)

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0.00 sec)

mysql>
```

图 2.7 删除数据库

完成上述联系后，重新创建新的数据库，为接下来的实验做准备。

2.2.4 创建示例数据库

- 1) 创建新的数据库 S_T_U201911808，并在其中定义三个基本表，Student, Course, SC. 如图 2.8 所示:

```
mysql> CREATE DATABASE S_T_U201911808;
Query OK, 1 row affected (0.01 sec)

mysql> USE S_T_U201911808;
Database changed
mysql> CREATE TABLE Student
-> (Sno CHAR(9) PRIMARY KEY,
-> Sname CHAR(20) UNIQUE,
-> Ssex CHAR(2),
-> Sage SMALLINT,
-> Sdept CHAR(20),
-> Scholarship CHAR(2));
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Course
-> (Cno CHAR(4) PRIMARY KEY,
-> Cname CHAR(40),
-> Cpno CHAR(4),
-> Ccredit SMALLINT,
-> FOREIGN KEY (Cpno) REFERENCES Course(Cno));
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE SC
-> (Sno CHAR(9),
-> Cno CHAR(4),
-> Grade SMALLINT,
-> PRIMARY KEY (Sno, Cno),
-> FOREIGN KEY (Sno) REFERENCES Student(Sno),
-> FOREIGN KEY (Cno) REFERENCES Course(Cno));
Query OK, 0 rows affected (0.02 sec)
```

图 2.8 创建数据库并定义三个基本表

2.2.5 创建基本表并添加数据

2) 使用可视化数据库管理软件——Navicat 向刚刚定义的三个基本表中添加数据，如图 2.9 所示：

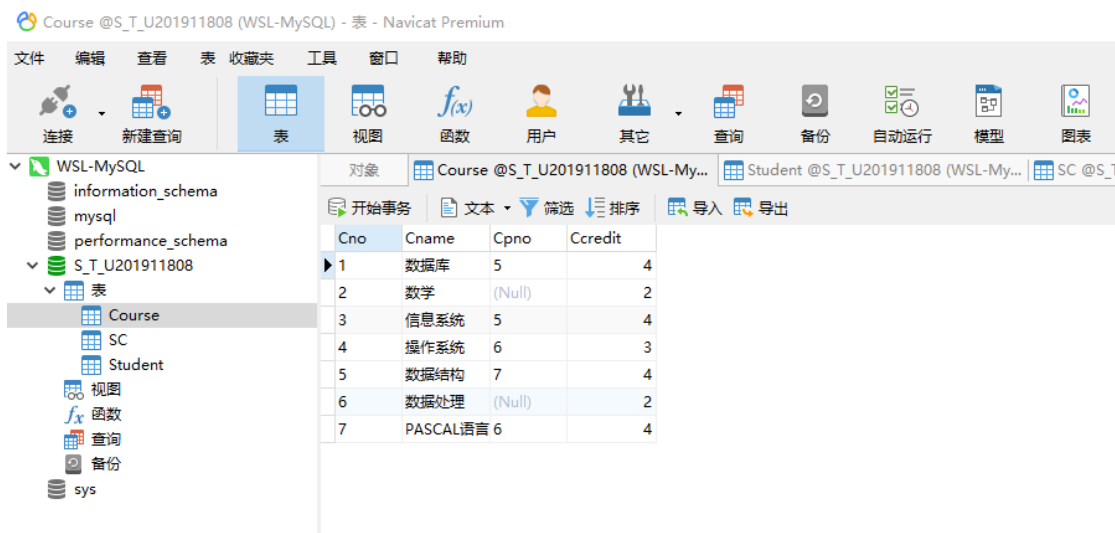


图 2.9 使用可视化软件添加数据

3) 回到命令行界面，使用指令查看并当前表格中现有的数据，如图 2.10 所示：

```
mysql> show tables;
+-----+
| Tables_in_S_T_U201911808 |
+-----+
| Course                     |
| SC                         |
| Student                    |
+-----+
3 rows in set (0.01 sec)

mysql> SELECT * FROM Course;
+-----+-----+-----+-----+
| Cno | Cname      | Cpno | Ccredit |
+-----+-----+-----+-----+
| 1   | 数据库    | 5    | 4       |
| 2   | 数学      | NULL | 2       |
| 3   | 信息系统  | 5    | 4       |
| 4   | 操作系统  | 6    | 3       |
| 5   | 数据结构  | 7    | 4       |
| 6   | 数据处理  | NULL | 2       |
| 7   | PASCAL语言 | 6    | 4       |
+-----+-----+-----+-----+
7 rows in set (0.00 sec)

mysql> SELECT * FROM SC;
+-----+-----+-----+
| Sno      | Cno | Grade |
+-----+-----+-----+
| 200215121 | 1   | 92    |
| 200215121 | 2   | 85    |
| 200215121 | 3   | 88    |
| 200215122 | 2   | 90    |
| 200215122 | 3   | 80    |
+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> SELECT * FROM Student;
+-----+-----+-----+-----+-----+-----+
| Sno      | Sname | Ssex | Sage | Sdept | Scholarship |
+-----+-----+-----+-----+-----+-----+
| 200215121 | 李勇  | 男   | 20   | CS    | 否          |
| 200215122 | 刘晨  | 女   | 19   | CS    | 否          |
| 200215123 | 王敏  | 女   | 18   | MA    | 否          |
| 200215125 | 张立  | 男   | 19   | IS    | 否          |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

图 2.10 查看当前表格中全部数据

2.2.6 对基本表进行查询

4) 练习使用 SELECT 语句查询，如图 2.11 所示的是查询 student 表格中的全部学生信息，如图 2.11 所示：


```
yuanye@MAXWELL-WIN: /mnt/

mysql> show tables;
+-----+
| Tables_in_S_T_U201911808 |
+-----+
| Course                    |
| SC                        |
| Student                   |
+-----+
3 rows in set (0.00 sec)

mysql> SELECT Sno,Sname,Ssex,Sage,Sdept FROM Student;
+-----+-----+-----+-----+-----+
| Sno      | Sname | Ssex | Sage | Sdept |
+-----+-----+-----+-----+-----+
| 200215121 | 李勇  | 男   | 20   | CS    |
| 200215122 | 刘晨  | 女   | 19   | CS    |
| 200215123 | 王敏  | 女   | 18   | MA    |
| 200215125 | 张立  | 男   | 19   | IS    |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> |
```

图 2.11 查询 srudent 表信息

5) 接下来，使用可视化界面查询选修 2 号课程并且成绩在 90 分以上的全部同学学号、姓名，查询结果为空，如图 2.12 所示：

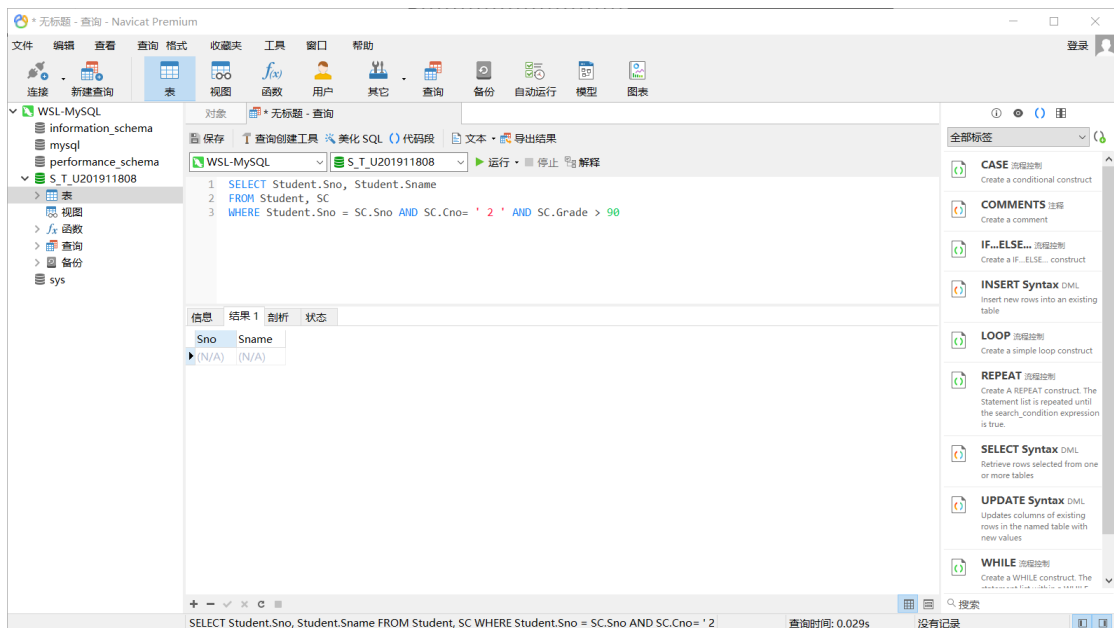


图 2.12 使用可视化管理软件查询信息

6) 接下来，依然利用可视化软件练习谓词查询操作，如图 2.13 所

示：

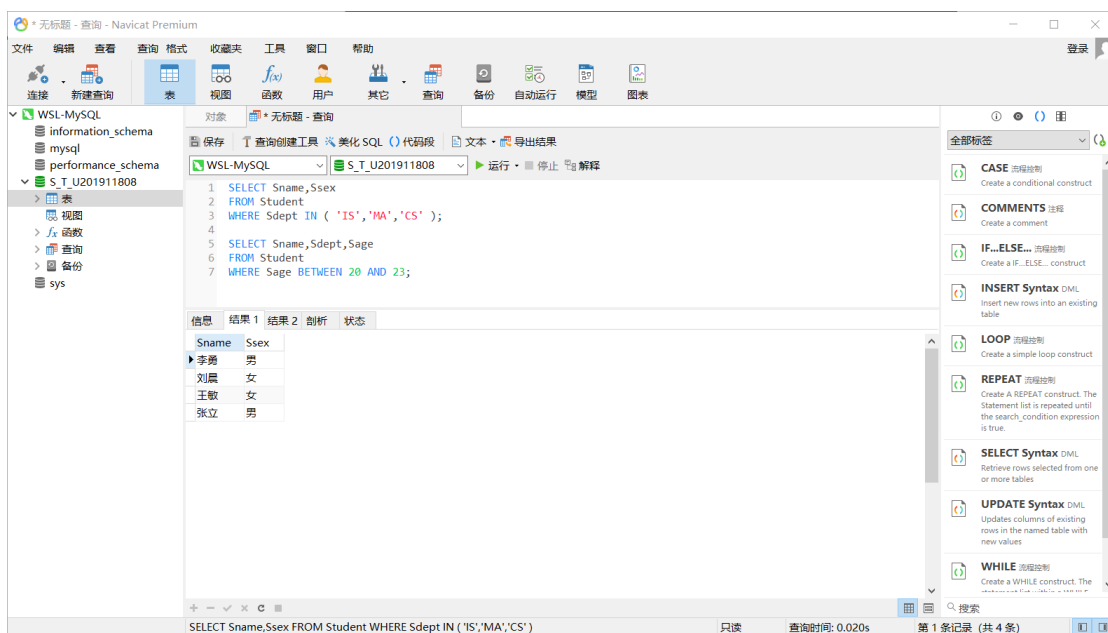


图 2.13 利用可视化软件进行谓词查询

7) 练习模糊查询，利用 LIKE 子句实现模糊查询。例如：查询所有姓刘学生的姓名、学号和性别，其结果如图 2.14 所示：

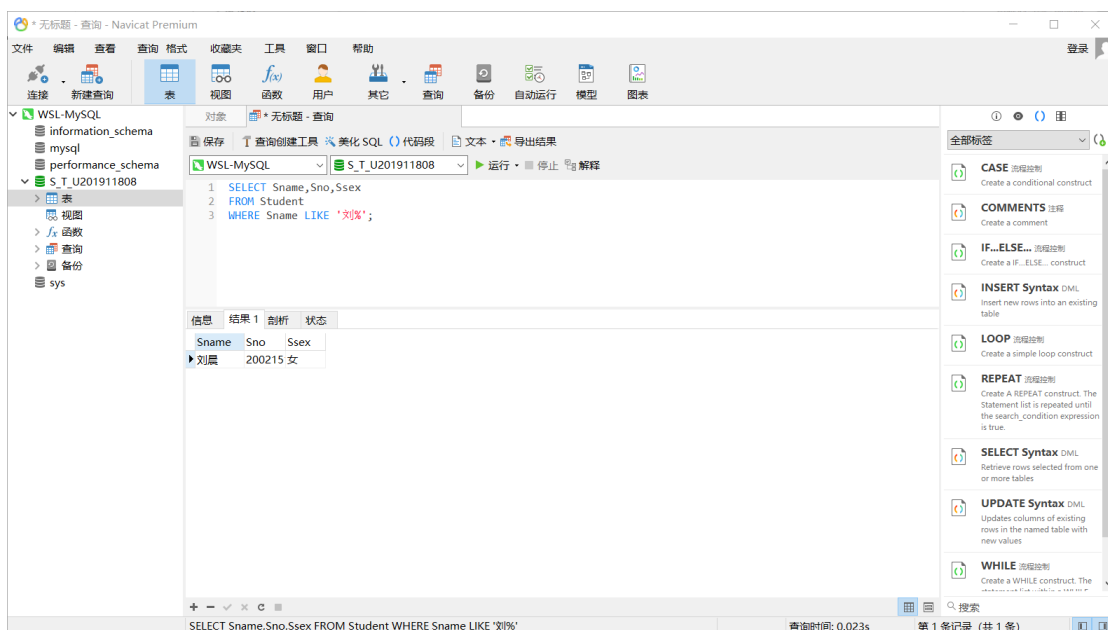


图 2.14 模糊查询练习

8) 利用 ORDER 子句为结果排序，例如：查询选修了 3 号课程的学生学号及其成绩，查询结果按分数降序排列，结果如图 2.15 所示：

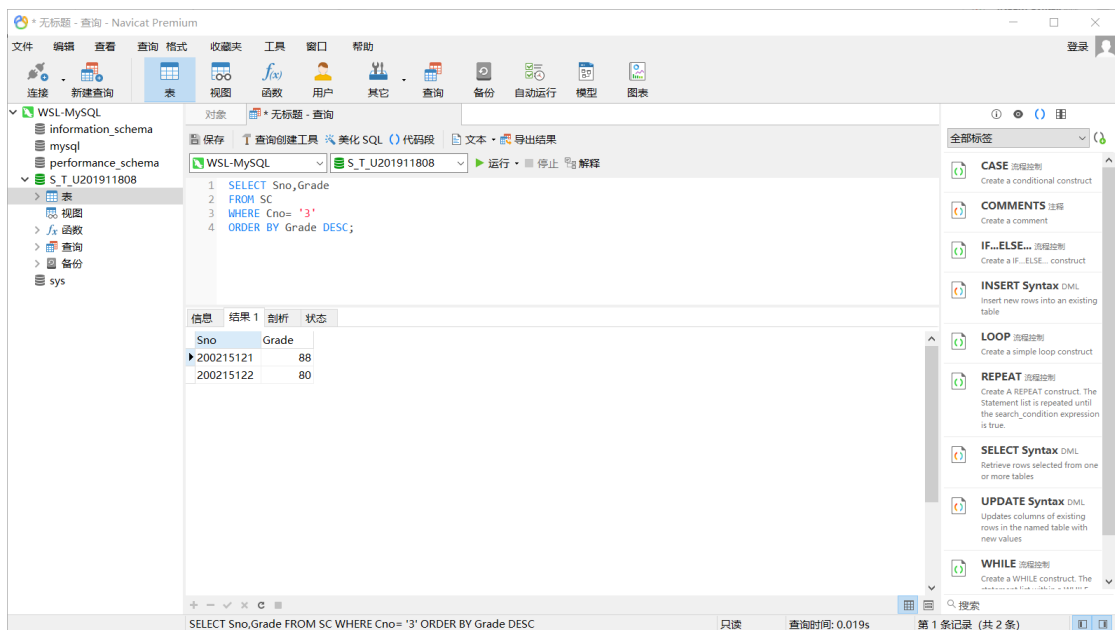


图 2.15 排序查询练习

9) 用 SQL Server 的统计函数进行统计计算，例如：计算 1 号课程的学生平均成绩，其结果如图 2.16 所示：

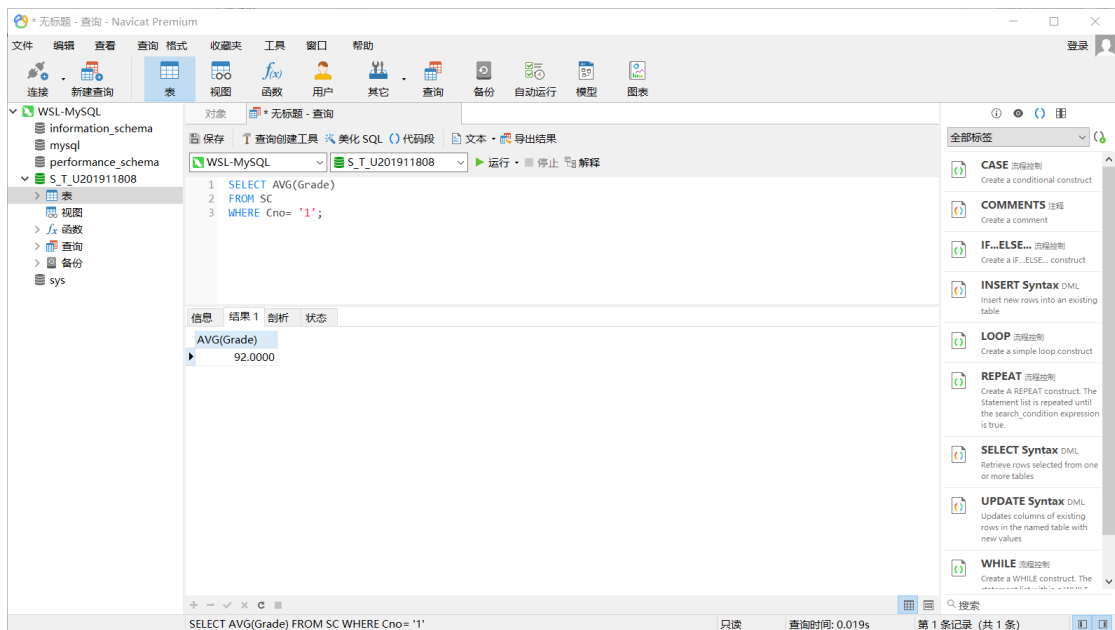


图 2.16 统计计算函数

10) 使用 Group By 子句查询选修了 3 门课以上课程的学生学号，查询结果为空，如图 2.17 所示：

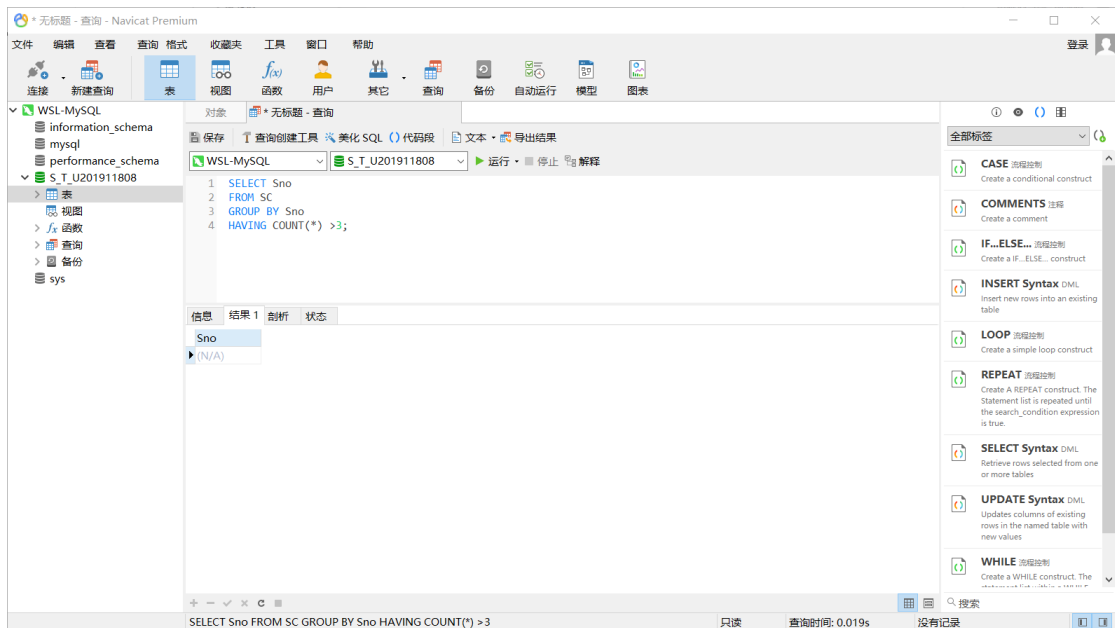


图 2.17 分组查询练习

2.2.7 扩展练习

(1) 查询全体学生的学号、姓名和年龄，如图 2.18 所示：

```
1 SELECT Sno, Sname, Sage
2 FROM Student;
3
```

Sno	Sname	Sage
200215121	李勇	20
200215122	刘晨	19
200215123	王敏	18
200215125	张立	19

图 2.18 查询信息

(2) 查询所有计算机系学生的详细记录，如图 2.19 所示：

```
1 SELECT *
2 FROM Student
3 WHERE Sdept = 'CS';
```

Sno	Sname	Ssex	Sage	Sdept	Scholarship
200215121	李勇	男	20	CS	否
200215122	刘晨	女	19	CS	否

图 2.19 查询计算机系信息

(3) 找出考试成绩为优秀（90 分及以上）或不及格的学生的学号、课程号及成绩，如图 2.20 所示：

1	SELECT Sno, Cno, Grade
2	FROM SC
3	WHERE Grade >= 90 OR Grade < 60;
4	

信息	结果 1	剖析	状态
Sno	Cno	Grade	
▶ 200215121	1	92	
200215122	2	90	

图 2.20 查询优秀信息

(4) 查询年龄不在 19~20 岁之间的学生姓名、性别和年龄，如图 2.21 所示：

1	SELECT Sname, Ssex, Sage
2	FROM Student
3	WHERE Sage NOT BETWEEN 19 AND 20;
4	

信息	结果 1	剖析	状态
Sname	Ssex	Sage	
▶ 王敏	女	18	

图 2.21 年龄筛选查询

(5) 查询数学系（MA）、信息系（IS）的学生的姓名和所在系，如图 2.22 所示：

1	SELECT Sname, Sdept
2	FROM Student
3	WHERE Sdept IN ('MA', 'IS');

信息	结果 1	剖析	状态
Sname	Sdept		
▶ 王敏	MA		
张立	IS		

图 2.22 按系别筛选查询

(6) 查询名称中包含“数据”的所有课程的课程号、课程名及其学分，如图 2.23 所示：

1	SELECT Cno, Cname, Ccredit
2	FROM Course
3	WHERE Cname LIKE '%数据%';

信息	结果 1	剖析	状态
	Cno	Cname	Ccredit
▶	1	数据库	4
	5	数据结构	4
	6	数据处理	2

图 2.23 模糊查询实现

(7) 找出所有没有选修课成绩的学生学号和课程号，如图 2.24 所示：

1	SELECT Sno, Cno
2	FROM SC
3	WHERE Grade IS NULL;
4	

信息	结果 1	剖析	状态
	Sno	Cno	
▶	(N/A)	(N/A)	

图 2.24 筛选空置的信息

(8) 查询学生 200215121 选修课的最高分、最低分以及平均成绩，如图 2.25 所示：

1	SELECT MAX(Grade), MIN(Grade), AVG(Grade)
2	FROM SC
3	WHERE Sno = '200215121';

信息	结果 1	剖析	状态
	MAX(Grade)	MIN(Grade)	AVG(Grade)
▶	92	85	88.3333

图 2.25 按照学号筛选查询

(9) 查询选修了 2 号课程的学生学号及其成绩，查询结果按成绩升序排列，如图 2.26 所示：

1	SELECT Sno, Grade
2	FROM SC
3	WHERE Cno = '2'
4	ORDER BY Grade;

信息	结果 1	剖析	状态
	Sno	Grade	
▶	200215121	85	
	200215122	90	

图 2.26 查询特定课程学生信息并排序

(10) 查询每个系名及其学生的平均年龄，如图 2.27 所示：

1	SELECT Sdept, AVG(Sage)
2	FROM Student
3	GROUP BY Sdept;

信息	结果 1	剖析	状态
	Sdept	AVG(Sage)	
▶	CS	19.5000	
	MA	18.0000	
	IS	19.0000	

图 2.27 查询综合函数

(思考：如何查询学生平均年龄在 19 岁以下（含 19 岁）的系别及其学生的平均年龄？），如图 2.28 所示：

1	SELECT Sdept, AVG(Sage)
2	FROM Student
3	GROUP BY Sdept
4	HAVING AVG(Sage) <= 19;
5	

信息	结果 1	剖析	状态
	Sdept	AVG(Sage)	
▶	MA	18.0000	
	IS	19.0000	

图 2.28 思考题

2.3 任务总结

a) 问题描述：在使用 wsl 安装 mysql 数据库中发生报错异常；

解决方案：这是因为以前在本机中安装过 MySQL，卸载的时候没有卸载干净导致的，因为有的文件夹隐藏的很深。1、卸载 MySQL 相关组件；2、删除 MySQL 的安装目录；然后重新安装即可。

b) 问题描述：再次安装过程中出现错误 1067

解决方案：这是配置文件修改错误，确认一下配置文件是否正确即可。

c) 问题描述:NaviCat 连接 WSL 中的 MySQL 数据库时发生 1130-host ... is not allowed to connect to this MySql server 报错；

解决方案：遇到这个问题首先到 mysql 所在的服务器上用连接进行处理

1、连接服务器: mysql -u root -p。2、看当前所有数据库: show databases;。3、进入 mysql 数据库: use mysql;。4、查看 mysql 数据库中所有的表: show tables; 重新授予相应用户权限即可解决问题。具体如图 2.29 所示：

```
mysql> use mysql;
Database changed
mysql> select USER from user;
+-----+
| USER |
+-----+
| root |
| ucp  |
| mysql.session |
| mysql.sys |
| root |
+-----+
5 rows in set (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO root@'%' IDENTIFIED BY '123456' WITH GRANT OPTION;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

图 2.29 授予用户权限

3 SQL 的复杂操作

3.1 任务要求

- (1) 熟练掌握 SQL 的连接查询语句
- (2) 熟练掌握 SQL 的嵌套查询语句
- (3) 掌握表名前缀、别名前缀的用法
- (4) 掌握不相关子查询和相关子查询的区别和用法
- (5) 掌握不同查询之间的等价替换方法（一题多解）及限制
- (6) 熟练掌握 SQL 的数据更新语句 INSERT、UPDATE、DELETE
- (7) 记录实验结果，认真完成实验报告

3.2 完成过程

3.2.1 扩展练习

(1) 查询每门课程及其被选情况（输出所有课程中每门课的课程号、课程名称、选修该课程的学生学号及成绩--如果没有学生选择该课，则相应的学生学号及成绩为空值）。

```
1 SELECT Course.Cno, Cname, Sno, Grade
2 FROM SC RIGHT JOIN Course ON (Course.Cno = SC.Cno);
```

信息	结果 1	剖析	状态
Cno	Cname	Sno	Grade
1	数据库	200215121	92
2	数学	200215122	90
2	数学	200215121	85
3	信息系统	200215122	80
3	信息系统	200215121	88
4	操作系统	(Null)	(Null)
5	数据结构	(Null)	(Null)
6	数据处理	(Null)	(Null)
7	PASCAL语言	(Null)	(Null)

图 3.1 查询 1

(2) 查询与“张立”同岁的学生的学号、姓名和年龄。（要求使用至少 3 种方法求解）

方法 1：嵌套搜索

```

1  SELECT Sno, Sname, Sage
2  FROM Student
3  WHERE Sage = (SELECT Sage FROM Student WHERE Sname='张立');

```

信息	结果 1	剖析	状态
	Sno	Sname	Sage
▶	200215122	刘晨	19
	200215125	张立	19

图 3.2 嵌套搜索

方法 2: 自连接

```

1  SELECT S1.Sno, S1.Sname, S1.Sage
2  FROM Student S1, Student S2
3  WHERE S2.Sname = '张立' AND S1.Sage = S2.Sage;

```

信息	结果 1	剖析	状态
	Sno	Sname	Sage
▶	200215122	刘晨	19
	200215125	张立	19

图 3.3 自连接

方法 3: EXIST 子句

```

1  SELECT Sno, Sname, Sage
2  FROM Student S1
3  WHERE EXISTS ( SELECT * FROM Student S2 WHERE S2.Sage = S1.Sage AND S2.Sname = '张立' );

```

信息	结果 1	剖析	状态
	Sno	Sname	Sage
▶	200215122	刘晨	19
	200215125	张立	19

图 3.4 EXIST 子句

(3) 查询选修了 3 号课程而且成绩为良好 (80~89 分) 的所有学生的学号和姓名。

```

1  SELECT Student.Sno, Sname
2  FROM Student, SC
3  WHERE Student.Sno = SC.Sno AND Cno = 3 AND Grade BETWEEN 80 AND 89;

```

信息	结果 1	剖析	状态
	Sno	Sname	
▶	200215121	李勇	
	200215122	刘晨	

图 3.5 查询 2

(4) 查询学生 200215122 选修的课程号、课程名

```

1  SELECT Course.Cno, Cname
2  FROM SC, Course
3  WHERE SC.Cno = Course.Cno AND Sno = '200215122';
4

```

信息	结果 1	剖析	状态
	Cno	Cname	
▶	2	数学	
	3	信息系统	

图 3.6 查询 3

(思考：如何查询学生 200215122 选修的课程号、课程名及成绩？)

```

1  SELECT Course.Cno, Cname, Grade
2  FROM SC, Course
3  WHERE SC.Cno = Course.Cno AND Sno = '200215122';
4

```

信息	结果 1	剖析	状态
	Cno	Cname	Grade
▶	2	数学	90
	3	信息系统	80

图 3.7 查询 4

(5) 找出每个学生低于他所选修课程平均成绩 5 分以上的课程号。(输出学号和课程号)

```

1  SELECT Sno, Cno
2  FROM SC SC1
3  WHERE Grade - ( SELECT AVG(Grade) FROM SC SC2 GROUP BY Sno HAVING SC1.Sno = SC2.Sno ) < - 5;
4

```

信息	结果 1	剖析	状态
	Sno	Cno	
▶	(N/A)	(N/A)	

图 3.8 查询 5

(6) 查询比所有男生年龄都小的女生的学号、姓名和年龄。

```

1  SELECT Sno, Sname, Sage
2  FROM Student
3  WHERE Ssex = '女' AND Sage < ALL ( SELECT Sage FROM Student WHERE Ssex = '男' );

```

信息	结果 1	剖析	状态
	Sno	Sname	Sage
▶	200215123	王敏	18

图 3.9 查询 ALL

(7) 查询所有选修了 2 号课程的学生姓名及所在系。

```

1  SELECT Sname, Sdept
2  FROM Student, SC
3  WHERE Student.Sno = SC.Sno AND Cno = 2;

```

信息	结果 1	剖析	状态
	Sname	Sdept	
▶	李勇	CS	
	刘晨	CS	

图 3.10 查询 AND

(8) 使用 update 语句把成绩为良的学生的年龄增加 2 岁，并查询出来。

```

1  UPDATE Student
2  SET Sage = Sage + 2
3  WHERE
4  Sno IN (
5  SELECT
6  temp.Sno
7  FROM
8  ( SELECT DISTINCT Student.Sno FROM SC, Student WHERE Student.Sno = SC.Sno AND Grade BETWEEN 80 AND 89 ) temp
9  );
10
11 SELECT *
12 FROM Student;

```

信息	结果 1	剖析	状态		
Sno	Sname	Ssex	Sage	Sdept	Scholarship
▶ 200215121	李勇	男	22	CS	否
200215122	刘晨	女	21	CS	否
200215123	王敏	女	18	MA	否
200215125	张立	男	19	IS	否

图 3.11 查询嵌套 IN

(9) 使用 insert 语句增加两门课程：C 语言和人工智能，并查询出来

1	INSERT
2	INTO Course
3	VALUES
4	(8, 'C语言', NULL, 4),
5	(9, '人工智能', 2, 2);
6	
7	SELECT *
8	FROM Course;

信息	结果 1	剖析	状态
Cno	Cname	Cpno	Ccredit
1	数据库	5	4
2	数学	(Null)	2
3	信息系统	5	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理	(Null)	2
7	PASCAL语言	6	4
8	C语言	(Null)	4
9	人工智能	2	2

图 3.12 使用 INSERT 插入课程

(10) 使用 delete 语句把人工智能课程删除，并查询出来。

1	DELETE
2	FROM Course
3	WHERE Cname = '人工智能';

信息	剖析	状态
DELETE FROM Course WHERE Cname = '人工智能' > Affected rows: 1 > 时间: 0.004s		

图 3.13 首先执行删除操作

Cno	Cname	Cpno	Ccredit
1	数据库	5	4
2	数学	(Null)	2
3	信息系统	5	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理	(Null)	2
7	PASCAL语言	6	4
8	C语言	(Null)	4

图 3.14 然后再次查询表，发现已经被删除

3.3 任务总结

本次任务主要复习并训练了 σ 连接、自然连接、外连接以及符合条件连接查询进行多表之间的连接操作，复合查询。此外，还练习了嵌套查询，能够较好地发挥 SQL 语言的优化特性，较高地提升查询效率。在进行嵌套查询时，往往先对查询过程进行分析，然后自内向外地逐步实现逐层查询。在这次实验中，我也发现了一些进行嵌套查询的技巧和方法，比如首先要对查询描述进行语义分割，这一步的目的是分析出查询分为几步，每一步的查询目标是什么，要怎么查，有什么约束条件。然后找到语义分割结果中的最内层查询，使用正常的 SELECT-FROM WHERE 子句描述，之后类似地实现外层的其他查询。最后，要讲这些子查询拼接起来，这时就要使用到 IN ANY ALL EXIST 这些谓词进行嵌套的语义描述。

最后，还练习了集合查询，可以将不便于在一次查询中完成的查询整合起来，有利于提升查询的清晰程度。

当然，从本实验开始，我逐步认识到了一个复杂的查询往往由多种方式，都可以达到查询的名誉表，但是他们的效率是有差异的，因此我认识到了点对资深数据库管理员的要求，要懂得怎样查询才能最高效，有时候我们不能完全依赖 DBMS 自身对于查询的优化处理，我们自己也需要懂得怎么对查询进行优化吗，从而在源头上达到最高效的查询。

4 SQL 的高级实验

4.1 任务要求

- (1) 掌握视图的定义与操作
- (2) 掌握对触发器的定义
- (3) 掌握对存储过程的定义
- (4) 掌握如何对用户进行授权和收回权限
- (5) 掌握用户定义完整性的方法
- (6) 写出实验报告

4.2 完成过程

4.2.1 扩展练习

- (1) 创建 CS 系的视图 CS_View

```
1 CREATE VIEW CS_View AS SELECT
2 *
3 FROM
4 Student
5 WHERE
6 Sdept = 'CS';
7
8 SELECT *
9 FROM CS_View;
```

图 4.1 创建视图操作

	Sno	Sname	Ssex	Sage	Sdept	Scholarship
▶	200215121	李勇	男	22	CS	否
	200215122	刘晨	女	21	CS	否

图 4.2 查询视图查看创建结果

- (2) 在视图 CS_View 上查询 CS 系选修了 1 号课程的学生

```
1 SELECT CS_View.Sno, Sname, Ssex, Sage, Sdept, Scholarship
2 FROM CS_View, SC
3 WHERE CS_View.Sno = SC.Sno AND Cno = 1;
4
```

信息	结果 1	剖析	状态		
Sno	Sname	Ssex	Sage	Sdept	Scholarship
▶ 200215121	李勇	男	22	CS	否

图 4.3 在视图上进行查询

- (3) 创建 IS 系成绩大于 80 的学生的视图 IS_View

```

1 CREATE VIEW IS_View ( Sno, Sname, Ssex, Sage, Sdept, Scholarship ) AS SELECT
2 Student.Sno,
3 Sname,
4 Ssex,
5 Sage,
6 Sdept,
7 Scholarship
8 FROM
9 Student,
10 SC
11 WHERE
12 Student.Sno = SC.Sno
13 AND Sdept = 'IS'
14 AND Grade > 80;

```

信息	剖析	状态
Ssex, Sage, Sdept, Scholarship FROM Student, SC WHERE Student.Sno = SC.Sno AND Sdept = 'IS' AND Grade > 80 > OK > 时间: 0.007s		

图 4.4 新建关于 IS 系得视图

(4) 在视图 IS_View 查询 IS 系成绩大于 80 的学生

```

1 SELECT *
2 FROM IS_View;
3

```

信息	结果 1	剖析	状态												
	<table border="1"> <thead> <tr> <th>Sno</th> <th>Sname</th> <th>Ssex</th> <th>Sage</th> <th>Sdept</th> <th>Scholarship</th> </tr> </thead> <tbody> <tr> <td>(N/A)</td> <td>(N/A)</td> <td>(N/A)</td> <td>(N/A)</td> <td>(N/A)</td> <td>(N/A)</td> </tr> </tbody> </table>	Sno	Sname	Ssex	Sage	Sdept	Scholarship	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)		
Sno	Sname	Ssex	Sage	Sdept	Scholarship										
(N/A)	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)										

图 4.5 查询视图，无结果

(5) 删除视图 IS_View

```

1 DROP
2 VIEW IS_View;

```

信息	剖析	状态
DROP VIEW IS_View > OK > 时间: 0.007s		

图 4.6 删除视图

(6) 利用可视化窗口创建 2 个不同的用户 U1 和 U2,利用系统管理员给 U1 授

予 Student 表的查询和更新的权限，给 U2 对 SC 表授予插入的权限。

```
mysql> show grants for 'U1'@'localhost';
+-----+
| Grants for U1@localhost |
+-----+
| GRANT USAGE ON *.* TO 'U1'@'localhost' |
| GRANT SELECT, UPDATE, ALTER ON 'S_T_U201911808'.'Student' TO 'U1'@'localhost' |
+-----+
2 rows in set (0.01 sec)

mysql> show grants for 'U2'@'localhost';
+-----+
| Grants for U2@localhost |
+-----+
| GRANT USAGE ON *.* TO 'U2'@'localhost' |
| GRANT INSERT ON 'S_T_U201911808'.'SC' TO 'U2'@'localhost' |
+-----+
2 rows in set (0.00 sec)
```

图 4.7 在命令行中检查权限授予情况

然后用 U1 登录，分别

1) 查询学生表的信息；

```
1 SELECT *
2 FROM Student;
```

信息	结果 1	剖析	状态		
Sno	Sname	Ssex	Sage	Sdept	Scholarship
▶ 200215121	李勇	男	22	CS	否
200215122	刘晨	女	21	CS	否
200215123	王敏	女	18	MA	否
200215125	张立	男	19	IS	否

图 4.8 使用 U1 查询 Student 表

2) 把所有学生的年龄增加 1 岁，然后查询；

```
1 UPDATE Student
2 SET Sage = Sage + 1;
3
4 SELECT *
5 FROM Student;
```

信息	结果 1	剖析	状态		
Sno	Sname	Ssex	Sage	Sdept	Scholarship
200215121	李勇	男	23	CS	否
200215122	刘晨	女	22	CS	否
200215123	王敏	女	19	MA	否
200215125	张立	男	20	IS	否

图 4.9 修改年龄并查询

3) 删除 IS 系的学生；

```
1 DELETE
2 FROM Student
3 WHERE Sdept = 'IS';
4
```

信息	状态
DELETE FROM Student WHERE Sdept = 'IS' > 1142 - DELETE command denied to user 'U1'@'localhost' for table 'Student' > 时间: 0s	

图 4.10 无权限删除

4) 查询 CS 系的选课信息。

```
1 SELECT Student.Sno, Sname, Cno, Cname
2 FROM Student, SC
3 WHERE Student.Sno = SC.Sno AND Student.Sdept = 'CS';
4
```

信息	状态
SELECT Student.Sno, Sname, Cno, Cname FROM Student, SC WHERE Student.Sno = SC.Sno AND Student.Sdept = 'CS' > 1142 - SELECT command denied to user 'U1'@'localhost' for table 'SC' > 时间: 0s	

图 4.11 无权查询 CS 系选课信息

用 U2 登录，分别

1) 在 SC 表中插入 1 条记录（'200215122'，'1'，75）；

```
1 INSERT INTO SC
2 VALUES ('200215122', '1', 75);
3
```

信息	剖析	状态
INSERT INTO SC VALUES ('200215122', '1', 75) > Affected rows: 1 > 时间: 0.002s		

图 4.12 U2 可以向 SC 中插入数据

2) 查询 SC 表的信息，

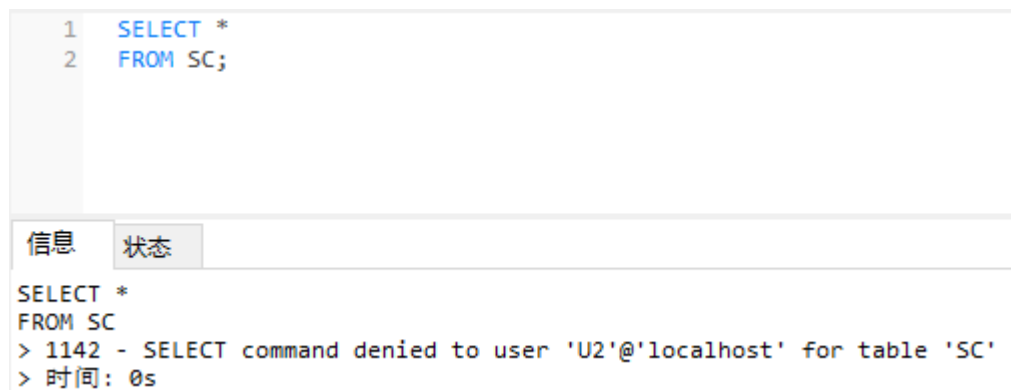


图 4.13 U2 无权查询 SC 表得信息

3) 查询视图 CS_View 的信息。

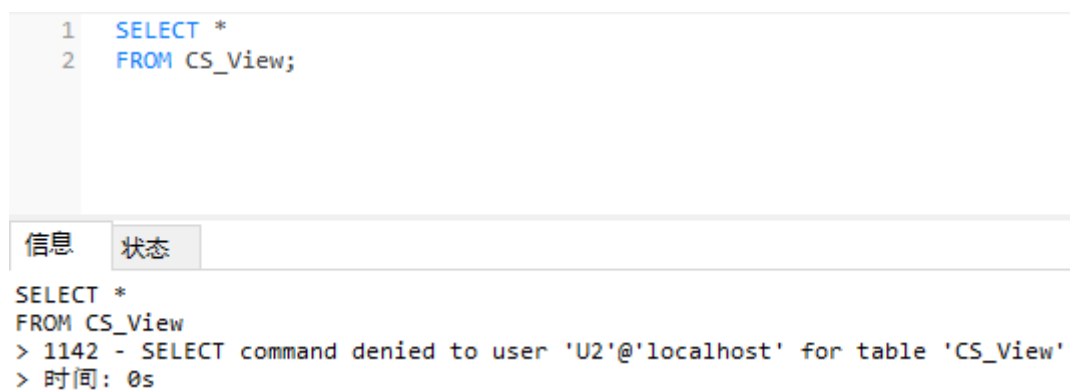


图 4.14 U2 无权查询 CS 视图

(7) 用系统管理员登录，收回 U1 的所有权限

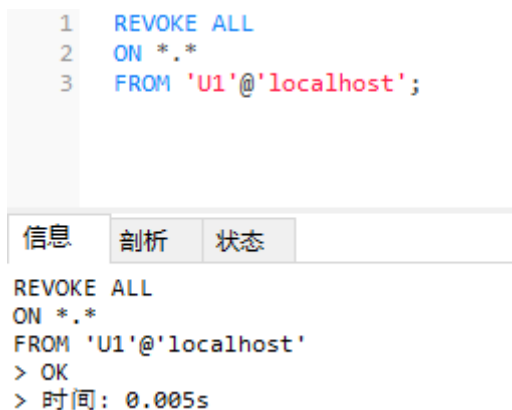


图 4.15 撤销权限

(8) 用 U1 登录，查询学生表的信息

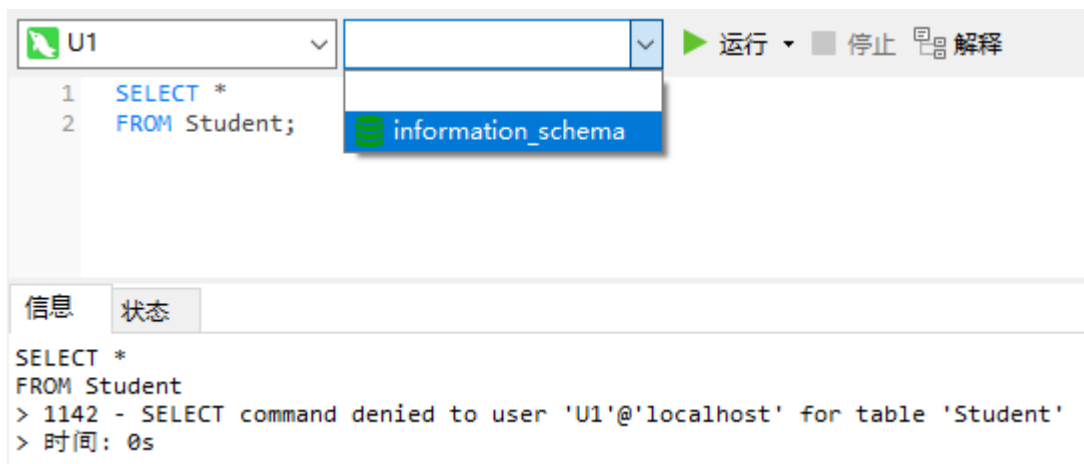


图 4.16 U1 已经无权限

(9) 用系统管理员登录

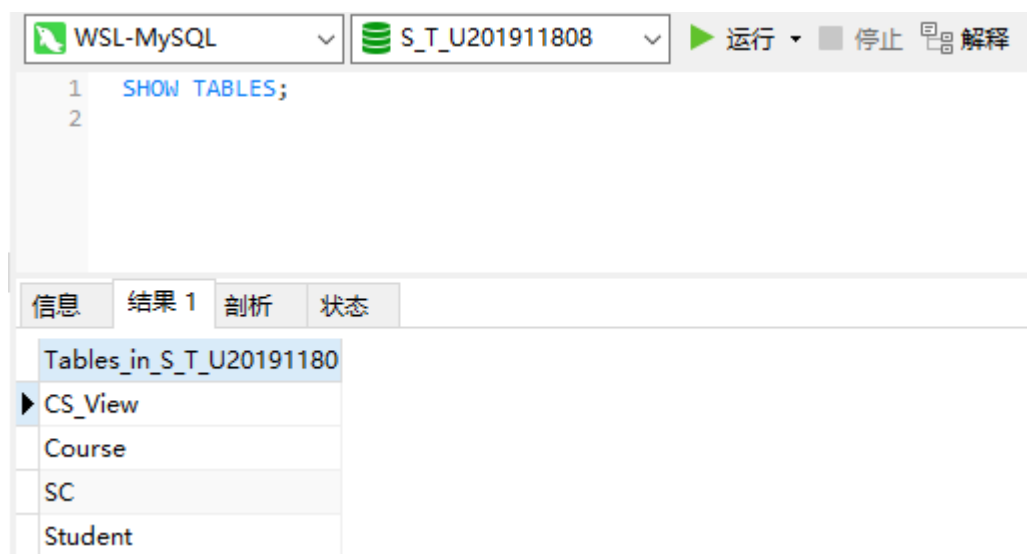


图 4.17 重新用系统管理员登录

(10) 对 SC 表建立一个更新触发器，当更新了 SC 表的成绩时，如果更新后的成绩大于等于 95，则检查该成绩的学生是否有奖学金，如果奖学金是“否”，则修改为“是”。如果修改后的成绩小于 95，则检查该学生的其他成绩是不是有大于 95 的，如果都没有，且修改前的成绩是大于 95 时，则把其奖学金修改为“否”。

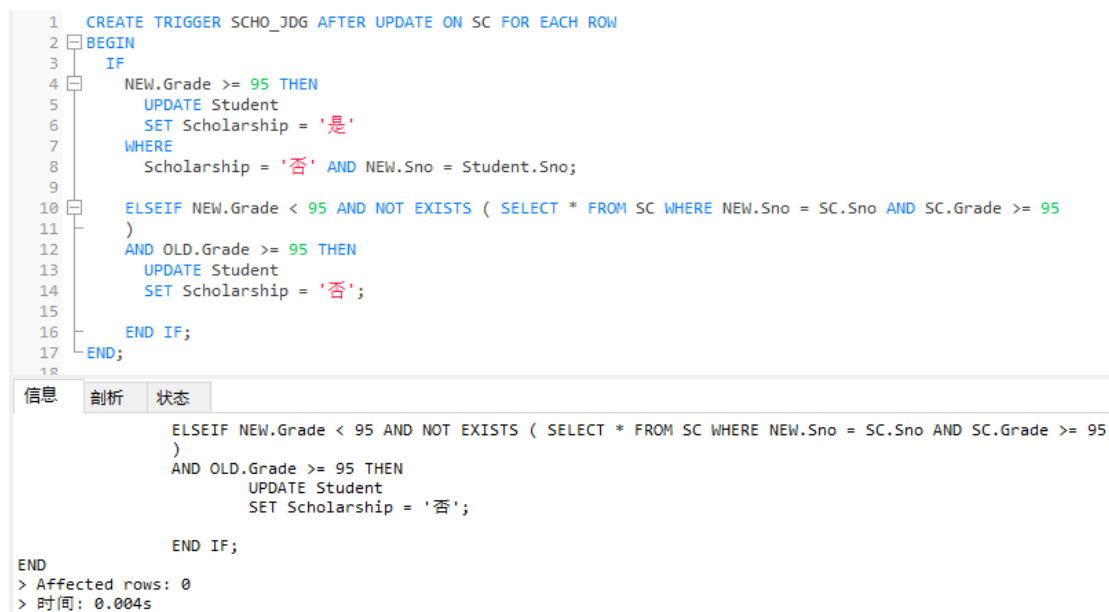


图 4.18 定义触发器

然后进行成绩修改，并进行验证是否触发器正确执行。

- 1) 首先把某个学生成绩修改为 98，查询其奖学金。

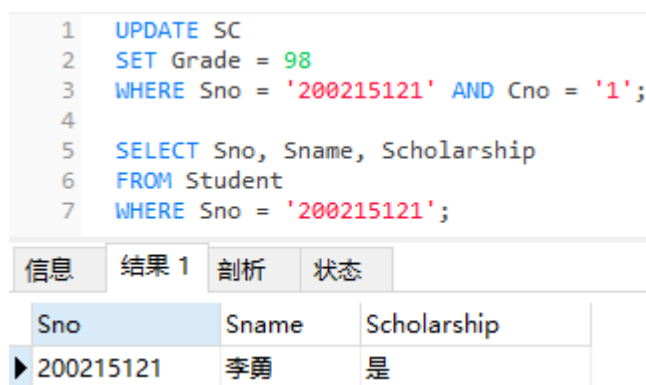


图 4.19 查看触发器效果

- 2) 再把刚才的成绩修改为 80，再查询其奖学金。

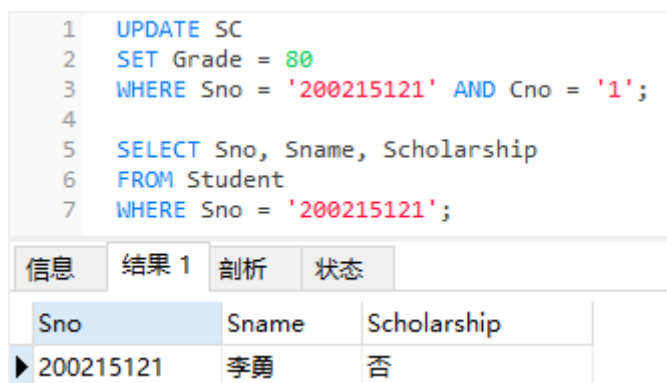


图 4.20 查看触发器效果

- (11) 删除刚定义的触发器



图 4.21 删除触发器

(12) 定义一个存储过程计算 CS 系的课程的平均成绩和最高成绩，在查询分析器或查询编辑器中执行存储过程，查看结果。

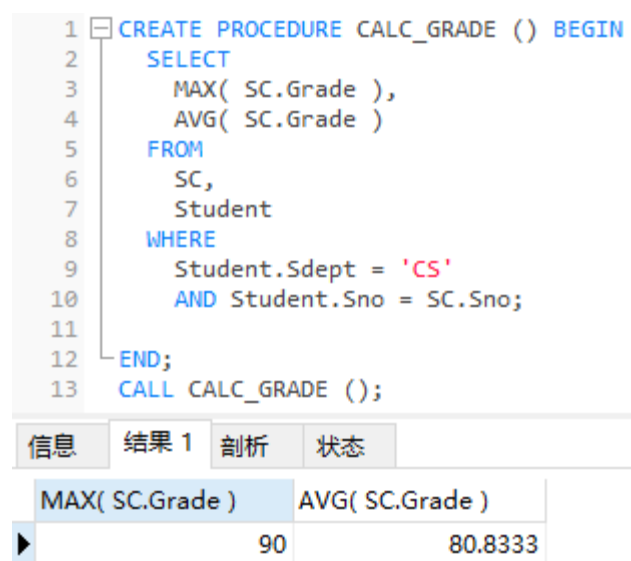


图 4.22 定义存储过程

(13) 定义一个带学号为参数的查看某个学号的所有课程的成绩，查询结果要包含学生姓名。进行验证。

```

1  CREATE PROCEDURE SNO_GRADE (
2      IN SID CHAR ( 9 )) BEGIN
3      SELECT
4          Student.Sno,
5          Student.Sname,
6          Course.Cname,
7          SC.Grade
8      FROM
9          Student,
10         SC,
11         Course
12     WHERE
13         Student.Sno = SID
14         AND Student.Sno = SC.Sno
15         AND SC.Cno = Course.Cno;
16
17 END;
18 CALL SNO_GRADE ( '200215121' );

```

信息	结果 1	剖析	状态	
	Sno	Sname	Cname	Grade
▶	200215121	李勇	数据库	80
	200215121	李勇	数学	80
	200215121	李勇	信息系统	80

图 4.23 利用存储过程进行查询

(14) 把上一题改成函数。再进行验证。

本题在 MySQL 中无法实现

(15) 在 SC 表上定义一个完整性约束，要求成绩再 0-100 之间。定义约束前，先把某个学生的成绩修改成 120，进行查询，再修改回来。定义约束后，再把该学生成绩修改为 120，然后进行查询。

在定义完整性约束前，先修改学号为 200215122 同学 1 号课程的成绩为 120 分，操作过程及修改后的结果如下：

1	UPDATE SC
2	SET Grade = 120
3	WHERE
4	Sno = '200215122'
5	AND Cno = '1';
6	SELECT
7	*
8	FROM
9	SC;

信息	结果 1	剖析	状态
	Sno	Cno	Grade
▶	200215121	1	80
	200215121	2	80
	200215121	3	80
	200215122	1	120
	200215122	2	90
	200215122	3	80

图 4.24 添加完整性约束

新增约束条件，如下图所示：

1	ALTER TABLE SC
2	ADD CONSTRAINT CON_GRADE
3	CHECK (Grade BETWEEN 0 AND 100);

信息	剖析	状态
ALTER TABLE SC ADD CONSTRAINT CON_GRADE CHECK (Grade BETWEEN 0 AND 100) > OK > 时间: 0.085s		

图 4.25 向 SC 表中新增完整性约束条件

之后，再次尝试修改学生成绩为 120 分，报错，如下图所示：

1	UPDATE SC
2	SET Grade = 120
3	WHERE
4	Sno = '200215122'
5	AND Cno = '1';
6	SELECT
7	*
8	FROM
9	SC;

信息	状态
UPDATE SC SET Grade = 120 WHERE Sno = '200215122' AND Cno = '1' > 3819 - Check constraint 'CON_GRADE' is violated. > 时间: 0s	

4.3 任务总结

本次实验主要练习了 SQL 语言的视图、触发器、存储过程、安全等功能。

视图是虚表，是由基本表或其他视图导出的表。视图可以简化用户操作，适当利用视图可以更清晰地表达查询，视图可以为数据库重构提供一定程度上的逻辑独立性，视图可以使用户从多个角度看待同一数据，视图也能够对机密数据提供安全保护。

触发器是用户定义在关系表上的一类由事件驱动的特殊过程。当满足了触发事件的条件后进行相应的处理操作，例如在数据库表中增加或者删除、修改了某条记录后，输出消息来告知该操作。这样就可以在这个表上设置一个触发器，触发条件为增加，删除或者修改了记录，触发的时间就是进行消息通知的输出。

存储过程是在大型数据库系统中，一组为了完成特定功能的 SQL 语句集，它存储在数据库中，一次编译后永久有效，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。存储过程是数据库中的一个重要对象。存储过程的优点包括：1. 存储过程只在创建时进行编译，以后每次执行存储过程都不需再重新编译，而一般 SQL 语句每执行一次就编译一次,所以使用存储过程可提高数据库执行速度。2.当对数据库进行复杂操作时(如对多个表进行 Update, Insert, Query, Delete 时)，可将此复杂操作用存储过程封装起来与数据库提供的事务处理结合一起使用。3.存储过程可以重复使用,可减少数据库开发人员的工作量。4.安全性高,可设定只有某此用户才具有对指定存储过程的使用权。缺点有如下两点：1.如果更改范围大到需要对输入存储过程的参数进行更改，或者要更改由其返回的数据，则您仍需要更新程序集中的代码以添加参数、更新 GetValue() 调用，等等，这时候估计比较繁琐了。2.可移植性差，由于存储过程将应用程序绑定到 SQL Server，因此使用存储过程封装业务逻辑将限制应用程序的可移植性。

5 数据库设计

5.1 任务要求

熟练掌握使用 SQL 语句设计数据库的方法，实现满足下面的系统功能的学生管理系统，完成实验报告。

系统功能要求：

- (1) 新生入学信息增加，学生信息修改。
- (2) 课程信息维护（增加新课程，修改课程信息，删除没有选课的课程信息）。
- (3) 录入学生成绩，修改学生成绩。
- (4) 按系统统计学生的平均成绩、最好成绩、最差成绩、优秀率、不及格人数。
- (5) 按系对学生成绩进行排名，同时显示出学生、课程和成绩信息。
- (6) 输入学号，显示该学生的基本信息和选课信息。

5.2 完成过程

5.2.1 需求分析

在任务文档中已经对系统的需求有非常具体的要求和说明。在实现以上的功能的同时，为了便于演示和测试时实时地检查数据库的状态，额外增加了一些辅助的功能，具体如下：

- (1) 连接到本机上的 mysql 数据库。
- (2) 对 Student 表进行更新操作，增加新入学学生的学生信息，对已有的学生信息进行修改，查看当前的学生表。
- (3) 对 Course 表进行更新维护，包括增加新课程的课程信息，修改已有的课程信息，删除未被选课的课程信息，查看当前的课程表。
- (4) 对 SC 表进行更新维护，包括录入学生成绩，修改学生成绩，查看当前的成绩表。
- (5) 统计功能。按系统统计学生的平均成绩、最好成绩、最差成绩、优秀率、不及格人数。
- (6) 统计功能。按系对学生成绩进行排名（同时显示出学生、课程和成绩信息）。
- (7) 自定义的查询：输入学号能够查询出相应学生的基本信息和选课信息。
- (8) 显示学生表、课程表、选课表的信息。
- (9) 异常处理。
- (10) 退出程序。

系统需要达到的目标主要有以下四点：

- 1) 操作简单方便、界面简洁美观；

- 2) 查看学生、课程、成绩信息时，可以对其属性添加、修改、删除；
- 3) 按照指定的条件对学生进行统计和查找；
- 4) 系统运行稳定、安全可靠。

5.2.2 系统设计

依据需求分析的结果可以将学生管理系统分成下面几个主体功能：基本查询功能，基本表（学生表、选课表、课程表）更新功能，统计功能和指定学号的查询功能。根据具体的功能再进行细分。学生管理系统的功能结构如图 5.1 所示：

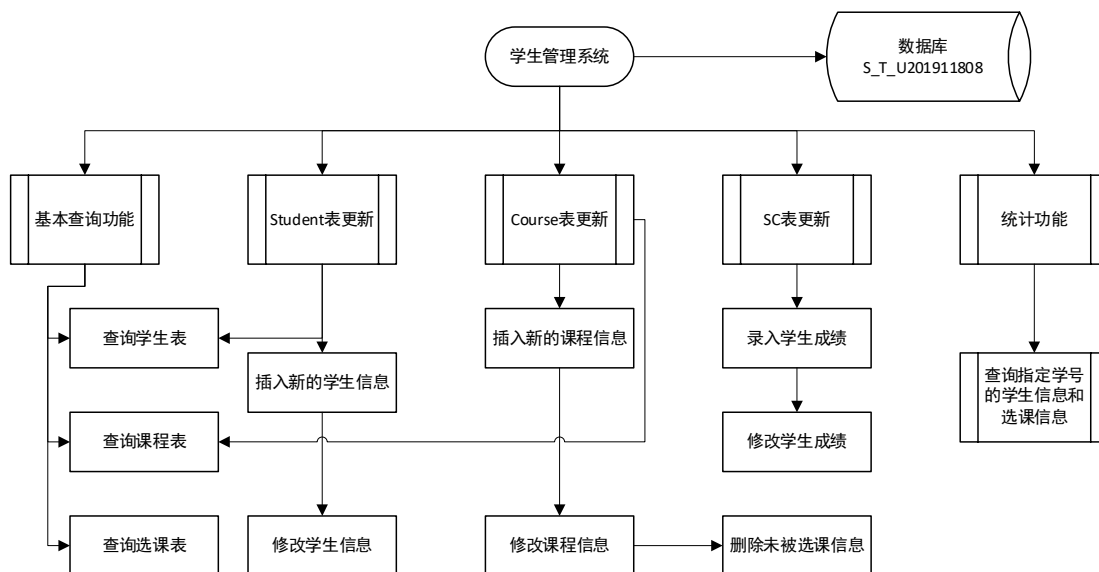


图 5.1 系统总体设计框图

5.2.3 基础数据库

本次实验在之前几次实验的基础上进行，所使用的数据即为前几次实验中所使用的 S_T_U201911808。数据库系统中包含两个实体：学生和课程，他们之间存在选课的多对多的关系。基础的三张基本表如图 5.2~5.4 所示。

	Cno	Cname	Cpno	Ccredit
▶ 0		入学考试	(Null)	5
1		数据库	5	4
2		数学	(Null)	2
3		信息系统	5	4
4		操作系统	6	3
5		数据结构	7	4
6		数据处理	(Null)	2
7		PASCAL语言	6	4
8		C语言	(Null)	4
9		Py语言	0	2

图 5.2 Course 表

Sno	Cno	Grade
200215121	1	80
200215121	2	80
200215121	3	80
200215122	1	80
200215122	2	90
200215122	3	80
200215122	5	96
200215123	9	98

图 5.3 SC 表

Sno	Sname	Ssex	Sage	Sdept	Scholarship
200215121	李勇	男	23	CS	否
200215122	刘晨	男	20	CS	是
200215123	王敏	女	19	MA	否
200215125	张立	男	20	IS	否

图 5.4 Student 表

5.2.4 概念模式设计

绘制学生数据库系统概念模式 E-R 图，如图 5.4 所示：

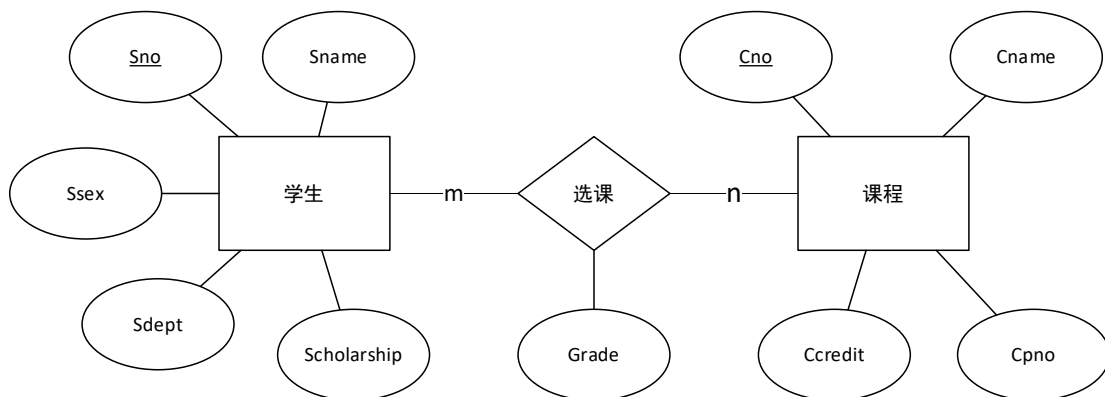


图 5.5 概念模式 E-R 图

5.2.5 逻辑结构设计

由 E-R 图导出的关系模式如下：

学生表： **Student**(Sno, Sname, Ssex, Sage, Sdept, Scholarship)

课程表： **Course**(Cno, Cname, Cpno, Ccredit)

学生选课表： **SC**(Sno, Cno, Grade)

5.2.6 物理结构设计

在物理结构设计方面，数据库自动对主码进行索引。

5.2.7 应用程序实现

(1) 环境描述

本应用程序基于 python 3.8 实现，需要手动安装 mysql 依赖的必要库：conda

install pymysql。

程序引用了以下的库:

```
import os
import pymysql
import pandas as pd
from tabulate import tabulate
```

(2) 菜单显示模块

由于 python 中没有 switch case 语句，因此包括系统主菜单及各级子菜单都是利用 if else 来判断流程跳转的，根据用户的输入字符来调用相应的控制功能函数或者返回上一层实现。

(3) 连接数据库

连接数据库采用的是 dbmysql 库中自带的函数 pymysql.connect，其中需要的参数解释如下：

- a) host=ip_addr, 填写目标数据库的 IP 地址。由于本次实验中数据库就在一本机上实现，因此可以默认填写 localhost 或者 127.0.0.1.
- b) user=usr, 填写登录数据库的用户名，需要特别注意的是，由于本应用程序需要支持更高权限的操作，因此需要登录的用户具有数据库的最高权限，即 DBS 权限模式。
- c) password=passwd, 需要传入与用户名相匹配的密码参数，需要注意的是，在程序实现的过程中，密码得登录应当尽可能采用隐士实现，防止可能的任意泄露数据的可能性。
- d) database=select_db, 这里需要输入被操作的数据库名称。由于之前的实验中定义的数据库为 S_T_U201911808，因此之后的登陆中应当填写这个字段，不能填写其他字段。

(4) 向学生表中插入学生数据

这里实现的思路是，首先通过 python 应用程序交互式地引导用户输入需要被插入学生的全部信息，然后将这些数据保存在 python 中临时定义的字典数据结构后当中。之后调用该字典中的字段内容，填充到 SQL 语句的空缺当中，形成最终的完整 SQL 查询语句。之后通过 pymysql 库中自带的封装好的函数，将这个 SQL 查询语句发送给 ODBC，通过 ODBC 的转译和激活，将特定的指令序列发送给运行在 WSL 虚拟机中的 MySQL 服务端程序进行查询。此时位于 python 主程序中的游标正在期待着从 mysql 返回的数据，当游标接收到全部的数据后，游标会将信息传递给位于 python 中的主变量，当然，此时的传输内容意味着刚刚插入操作的状态。如果为 1 的话，代表刚刚插入操作执行成功，Student 表中已经可以查询到刚刚插入的同学信息了。如果为 0 的话，代表插入操作失败，还需要用户重新采取插入操作。这时，python 主程序会尽可能地告知用户插入失败的具体原因，从而引导用户修改插入的内容，正确地执行插入操作。

（5）更新 Student 表中的学生信息

这里的操作和（4）非常类似，只是由于我们更新的是已经存在在表中的数据，因此在输入新的数据之前应当先让数据库管理系统在表格中查找并定位需要修改的学生信息。这时，python 主程序首先会引导用户搜索相应的同学信息，然后将用户的输入保存在字典数据结构中，采取和（4）类似的操作，构造相应的 SQL 语句执行预查询操作，预查询操作是必不可缺的，因为当用户输入一个非法的或者数据库中并不存在的学生信息时，应用程序应当及时报错，避免无效的修改操作，以浪费系统资源和用户时间。假如预操作返回了成功的信息，那么 python 主程序接下来将会提示用户选择一个需要修改的属性名称，并输入修改后的新值。

同样地，当用户输入完这些数据后，python 主程序会将这些数据集合在一个数据字典中，然后利用字典中的数据以及刚刚定位到的学生信息构造新的 SQL 修改执行语句，执行操作后，python 主程序会告知用户刚刚的修改操作是否成功，为了更加直观地展示修改结果，python 主程序还会将修改后的表格 i 相应数据项打印给用户，供用户查看与分析。

（6）插入课程信息

课程信息的插入和（4）学生信息插入非常类似，只是用户需要提供的属性分量有一定差异，其余操作都可以按照（4）的操作进行，因此在此不再赘述。

（7）修改课程信息

课程信息的修改和（5）学生信息修改非常类似，只是用户需要提供的属性分量有一定差异，其余操作都可以按照（5）的操作进行，因此在此不再赘述。

（8）删除课程信息

删除课程信息的操作与（7）修改课程信息操作类似，同样需要用户先对需要删除的课程信息进行检索，当发现了该课程的信息后，将信息打印在屏幕上并呈现给用户，询问用户是否确认要删除这个课程的信息。当用户输入了确认的信号时，python 主程序生成删除数据项执行 SQL 语句，并发送给 MYSQL 数据库执行该操作。

（9）登记学生成绩

等级学生成绩类似于（4）插入学生信息以及（6）插入课程信息，唯一有差异的地方是，等级学生信息是将新的信息插入到 SC 表中，SC 表的特殊之处在于，它的主键(Sno, Cno)拥有两个分量，因此在检索数据项的过程种风格，python 主程序需要引导用户输入两个信息，才能够正确查找到相应的数据项进行插入。

（10）修改学生成绩

修改学生成绩与（7）的操作类似，在此不再赘述。

（11）按系统计

这个函数的操作氛围三个主要步骤：查询所有的系名、按照系便利数据库，查询每个系的全部数据信息、展示信息。

第一步，查询所有得系名。这一步的目的是为接下来的具体数据查询做准备工作。Python 主程序向 ODBC 发送 `SELECT DISTINCT Sdept FROM Student ORDER BY Sdept` 的查询语句，游标接受返回的数据，并依次将他们存储在列表当中。

第二步，依次查询每个系的全部数据信息。这一步的目的是获取展示给用户的数据。`"SELECT AVG(SC.Grade), MAX(SC.Grade), MIN(SC.Grade) FROM Student, SC WHERE Student.Sdept = '%s' AND \"Student.Sno = SC.Sno\" % dept`，按照这样的语句查询每个系的各项数据，并存储在字典中等待展示给用户。

第三步，展示信息。这一步主要是将刚刚从数据库中检索到的全部信息整合起来并统一展示给用户。在展示信息是，应当注意信息显示的美观程度，因此这里采用了 python 中的优美表格展示库，可以将数据表中的信息以表格的形式展示给用户查看，类似交互式 mysql 命令行中表格显示的那种样子。

（12）按系对学生成绩进行排名

这一步的操作与（11）类似，需要分为三个主要步骤进行。由于具体的过程在（11）中已经描述的很清楚了，因此再次不咋赘述。感兴趣的读者可以参考本实验报告后面附带的相关代码思考与理解。

（13）按学号显示学生信息

这一步其实就是数据查询操作，通过用户输入的学号，作为主键输入到 mysql 数据库中检索数据，并展示相关的数据。

5.2.8 系统测试

测试环境：操作系统：Windows 10；数据库版本：8.0.27-0ubuntu0.20.04.1

1. 应用程序启动与连接数据库测试

首先按回车采取默认值，当然如果需要连接其他数据库，也可以输入相应的值，如图 5.6 所示。

```
D:\Anaconda\python.exe C:/Users/YuanYe/Desktop/数据库/expr4.py
Input the IP Address (default: localhost):
Username (default: yuanye):
Password (default: yuanye):
Select one database to operate (default: S_T_U201911808): |
```

图 5.6

然后会显示登录信息、数据库版本信息以及初始主菜单界面，如图 5.7 所示。

```

... Connecting to the Database ...
----- Student Management System -----
Author: 网安1902班 袁也 U201911808
MySQL Version: 8.0.27-0ubuntu0.20.04.1
User: yuanye@localhost Database: S_T_U201911808

***** Menu *****
1. Insert New Student Info          2. Modify Existing Student Info
3. Insert New Course Info          4. Modify Existing Course Info
5. Delete Existing Course Info     6. Record Student Grade
7. Modify Student Grade            8. Print Analytical Info by Dept
9. Print Grade Rank by Dept        10. Print Student Info

                                0. Exit

*****

Choose what you want (0~10):

```

2. 测试插入学生信息

选择操作 1，并输入相应的插入数据，如图 5.8 所示。

```

Choose what you want (0~10): 1
===== Mode 1: Insert New Student Info =====
Please input the following information!
Student ID: 200215129
Name: 张三
Gender: 男
Age: 21
Student's Department: EE
Have Scholarship? (y/n)n
Re-check the student's information below:
+---+-----+-----+-----+-----+-----+-----+
|  |      sno | sname  | sex   | age  | dept  | scholarship |
|---+-----+-----+-----+-----+-----+-----|
| 0 | 200215129 | 张三   | 男    | 21   | EE    | 否          |
+---+-----+-----+-----+-----+-----+-----+
ARE YOU SURE TO INSERT THE NEW INFORMATION? (y/n)y
Successfully Inserted!

```

在 navicat 中查询 student 表的信息，如图 5.9 所示，可以看到已经能够查询到相应的学生信息。

Sno	Sname	Ssex	Sage	Sdept	Scholarship
200215121	李勇	男	23	CS	否
200215122	刘晨	男	20	CS	是
200215123	王敏	女	19	MA	否
200215125	张立	男	20	IS	否
200215129	张三	男	21	EE	否

3. 测试修改学生信息

选择操作 2，进行修改操作，如图 5.10 所示。


```

Choose what you want (0~10): 2
===== Mode 2: Modify Existing Student Info =====
Please input the Student ID to search the student!
Student ID: 200215129
Student founded! Here is the information of this person:
+---+-----+-----+-----+-----+-----+-----+
|   |      sno | sname  | sex   | age  | dept  | scholarship |
|---+-----+-----+-----+-----+-----+-----|
| 0 | 200215129 | 张三   | 男    | 21   | EE    | 否          |
+---+-----+-----+-----+-----+-----+-----+
Select column 1~6 to modify, select 0 to exit: 4
Input the new data: 24
Successfully Updated!
Select column 1~6 to modify, select 0 to exit: 6
Input the new data: 是
Successfully Updated!
Select column 1~6 to modify, select 0 to exit: 0

```

Student 表如图 5.11 所示:

Sno	Sname	Ssex	Sage	Sdept	Scholarship
200215121	李勇	男	23	CS	否
200215122	刘晨	男	20	CS	是
200215123	王敏	女	19	MA	否
200215125	张立	男	20	IS	否
200215129	张三	男	24	EE	是

4. 插入课程信息

选择模式 3，输入相应的数据，如图 5.12 所示:

```

Choose what you want (0~10): 3
===== Mode 3: Insert New Course Info =====
Please input the following information!
Course ID: 10
Course Name: 大学物理
Previous Course ID: 2
Credits: 5
Re-check the course's information below:
+---+-----+-----+-----+-----+
|   |   cno | cname  |   cpno |   ccredit |
|---+-----+-----+-----+-----|
| 0 |    10 | 大学物理 |    2   |    5   |
+---+-----+-----+-----+-----+
ARE YOU SURE TO INSERT THE NEW INFORMATION? (y/n)y
Successfully Inserted!

```

查询 course 表，如图 5.13 所示。

Cno	Cname	Cpno	Ccredit
0	入学考试	(Null)	5
1	数据库	5	4
10	大学物理	2	5
2	数学	(Null)	2
3	信息系统	5	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理	(Null)	2
7	PASCAL语言	6	4
8	C语言	(Null)	4
9	Py语言	0	2

5. 修改课程信息

选择模式 4，输入修改的信息，如图 5.14 所示。

```

Choose what you want (0~10): 4
===== Mode 4: Modify Existing Course Info =====
Please input the Course ID to search the course!
Course ID: 10
Course founded! Here is the information of this course:
+---+-----+-----+-----+-----+
|   | cno | cname   | cpno | ccredit |
+---+-----+-----+-----+-----+
| 0 | 10 | 大学物理 | 2    | 5    |
+---+-----+-----+-----+-----+
Select column 1~4 to modify, select 0 to exit: 4
Input the new data: 3
Successfully Updated!
Select column 1~4 to modify, select 0 to exit: 0

```

修改后的 course 表如图 5.15 所示：

Cno	Cname	Cpno	Ccredit
0	入学考试	(Null)	5
1	数据库	5	4
10	大学物理	2	3
2	数学	(Null)	2
3	信息系统	5	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理	(Null)	2
7	PASCAL语言	6	4
8	C语言	(Null)	4
9	Py语言	0	2

6. 删除课程信息

选择模式 5，删除大学物理课程信息，如图 5.16 所示。

```
Choose what you want (0~10): 5
===== Mode 5: Delete Existing Course Info =====
Please input the Course ID to search the course!
Course ID: 10
Course founded! Here is the information of this course:
+---+-----+-----+-----+
|   | cno | cname | cpno | ccredit |
+---+-----+-----+-----+
| 0 | 10 | 大学物理 | 2 | 3 |
+---+-----+-----+-----+
ARE YOU SURE TO DELETE THIS COURSE? (y/n): y
Successfully Deleted!
```

删除后的 course 表如图 5.17 所示。

	Cno	Cname	Cpno	Ccredit
▶	0	入学考试	(Null)	5
	1	数据库	5	4
	2	数学	(Null)	2
	3	信息系统	5	4
	4	操作系统	6	3
	5	数据结构	7	4
	6	数据处理	(Null)	2
	7	PASCAL语言	6	4
	8	C语言	(Null)	4
	9	Py语言	0	2

7. 登记学生成绩

选择模式 6，输入相应的数据，如图 5.18 所示。

```
Choose what you want (0~10): 6
===== Mode 6: Record Student's Grade =====
Please input the following information!
Student ID: 200215129
Course ID: 9
Student's Grade: 76
Re-check the information below:
+---+-----+-----+-----+
|   |      sno |   cno |   grade |
+---+-----+-----+-----+
| 0 | 200215129 | 9 | 76 |
+---+-----+-----+-----+
ARE YOU SURE TO INSERT THE NEW INFORMATION? (y/n)y
Successfully Inserted!
```

添加后的 SC 表如图 5.19 所示。

Sno	Cno	Grade
200215121	1	80
200215121	2	80
200215121	3	80
200215122	1	80
200215122	2	90
200215122	3	80
200215122	5	96
200215123	9	98
200215129	9	76

8. 修改学生成绩

选择模式 7，输入相应的信息查询并修改学生成绩，如图 5.20 所示。

```

Choose what you want (0~10): 7
===== Mode 7: Modify Student's Grade =====
Please input the Student ID and Course ID to search the grade record!
Student ID: 200215129
Course ID: 9
Course founded! Here is the information of this course:
+---+-----+-----+-----+
|  |      sno |   cno |  grade |
+---+-----+-----+-----+
| 0 | 200215129 |     9 |     76 |
+---+-----+-----+-----+
Are you sure to modify this score? (y/n): y
Input the new grade: 87
Successfully Updated!

```

修改后的 SC 表如图 5.21 所示。

Sno	Cno	Grade
200215121	1	80
200215121	2	80
200215121	3	80
200215122	1	80
200215122	2	90
200215122	3	80
200215122	5	96
200215123	9	98
200215129	9	87

9. 按系打印统计信息

选择模式 8，可以看到统计信息如图 5.22 所示。

```

Choose what you want (0~10): 8
===== Mode 8: Print Analytical Info by Dept =====
Department List: CS,EE,IS,MA
Here is the information for your inquire:
+---+-----+-----+-----+-----+-----+-----+
|   | dept |   avg |   max |   min |   a_rate |   f_num |
+---+-----+-----+-----+-----+-----+-----+
| 0 | CS   | 83.7143 |   96 |   80 | 0.285714 |   0 |
| 1 | EE   | 87      |   87 |   87 | 0        |   0 |
| 2 | IS   |         |  nan |  nan | 0        |   0 |
| 3 | MA   | 98      |   98 |   98 | 1        |   0 |
+---+-----+-----+-----+-----+-----+-----+
Press Enter to continue...

```

10. 学生成绩排名

选择模式 9，输入项查询的院系信息，查询结果如图 5.23 所示。

```

Choose what you want (0~10): 9
===== Mode 9: Print Department Student Ranking =====
Department List: CS,EE,IS,MA
Please select one department to inquire the student's ranking: CS
Here is the ranking of CS department:
+---+-----+-----+-----+-----+-----+
|   |      sno | sname |   avg |   c_num |   credits |
+---+-----+-----+-----+-----+-----+
| 0 | 200215122 | 刘晨  | 86.5 |   4 |   14 |
| 1 | 200215121 | 李勇  | 80   |   3 |   10 |
+---+-----+-----+-----+-----+-----+
Do you want to search other department's rank? (y/n)y
Department List: CS,EE,IS,MA
Please select one department to inquire the student's ranking: IS
Here is the ranking of IS department:
+---+-----+-----+-----+-----+-----+
| sno | sname | avg | c_num | credits |
+---+-----+-----+-----+-----+-----+

```

11. 查询学生信息

选择模式 10，输入需要查询的学生学号，查询结果如图 5.24 所示。

```

Choose what you want (0~10): 10
===== Mode 10: Lookup Student Basic Info =====
Please input the Student ID to search the student!
Student ID: 200215129
Student founded! Here is the information of this person:
+---+-----+-----+-----+-----+-----+-----+
|   |      sno | sname  | sex   | age  | dept  | scholarship |
|---+-----+-----+-----+-----+-----+-----|
| 0 | 200215129 | 张三   | 男    | 24   | EE    | 是          |
+---+-----+-----+-----+-----+-----+-----+
Student's Course Info:
+---+-----+-----+-----+-----+-----+
|   |   cno | cname  | grade | credits | cpno |
|---+-----+-----+-----+-----+-----|
| 0 |     9 | Py语言 | 87    | 2       | 0     |
+---+-----+-----+-----+-----+-----+
Press Enter to continue ...

```

5.3 任务总结

本次实验不仅仅是对前几次实验的一个总的综合、复习和运用，而且学习了使用 pymysql 数据库扩展 python 库提供的函数 API，用 python 实现建立和操作数据库的方法，我们整体地实现了一个数据库管理系统，对数据库原理理论课中学习到的数据库设计思路应用到实际中来。通过初期对数据库功能的设计，加深了对关系模式、E-R 图、范式、数据流图等概念的理解，确实地体会到了这些工程化的设计方法对实际工程设计起到的重要作用。

在编写源代码的过程中，不仅又加强锻炼了对 SQL 语法的使用，各种复杂查询和操作语句的写法，还学会了如何利用 MySQL 提供的 API 来操作数据库中的数据。有了 API 说明手册，就只需要考虑用 python 的方式去完成相应的功能，而不需要考虑数据库具体的物理数据模式，这样大大简化了设计过程。

在初始编写好系统进行测试时，我还没有编写相应功能的提示和输出排版，因此在命令行窗口上测试时，有时会分不清各功能的作用，输出的结果也不容易审查排错，后来在每一个功能的界面和步骤中都加入了充足的提示信息，对输出也进行了规范的排版，测试功能时便方便了很多，对输出也更好辨识。用户在使用系统时也需要一个便宜的 GUI 从而使得操作的难度降低，所以这一部分在设计中也非常重要。

6 课程总结

本次的数据库原理实验在前三次实验中熟悉了对数据库各种操作语句的使用，在此基础上，最后一次实验利用这些知识实现了一个学生信息数据库管理系统。整个实验课程圆满结束，在这个过程中也有许多值得记录的困难和收获。

第一次实验印象最深刻的就是使用数据库可视化工具，第一次使用这种工具感觉非常便捷，理论课上学习了各种冗长复杂的 SQL 语句，在可视化工具中有时只需简单的键鼠操作就可以完成，在创建基本表和插入数据的过程中起到了非常高效的作用。不过后面的实验中也有一些操作使用可视化工具也没有做到很好的简化，例如定义存储过程和函数等，因此要使用这个数据库的高级功能仍然需要学习掌握他们的 SQL 具体语法。

在对数据库进行查询的过程中，实践了各种功能的查询方法，查询语句的关键就在于如何描述出查询的条件，利用不同的谓词、集合函数、子查询等组合可以构造出各种不同的查询条件，重点在于如何去理清其中的逻辑，对于复杂的查询，写出来的查询语句可能非常复杂，其中可能有许多细节需要特别注意，在编写时要非常细心。对于同一个查询条件，可以用不同的 SQL 语句来实现，不同的数据库在语法和规范上有细微的差别，因此在对不同的数据库操作的时候必须先了解相关的规范才能避免出错。

在对数据库中的数据操作的过程中，如果违背了完整性约束就会操作失败，完整性约束保护了数据完整性，使的数据之间的逻辑关系不会遭到破坏，在设计数据库的时候也要注意制定完全的完整性约束。

最后一次的编写数据库管理系统的实验不仅运用到了前几次实验中使用到的各种语句，还学习了使用 pymysql 编写嵌入式 SQL 程序的方法，是一次从学到用的实践体验，在掌握了 pymysql 库中函数的用法后，结合前几次课的基础练习，要求的数据库管理系统并不难完成，不过由于类似的功能在三个表上都有重复，所以仍然耗费了一些时间。

在完成实验的过程中也存在一些问题没有办法自己解决（例如利用函数完成存储过程的功能的问题），原因有时并不是难度非常大，而是自己可能陷入了一个死胡同，没有办法打开新的思路，这个时候向老师同学们请教询问就有很好的效果，在整个过程中有很多问题都是大家都没有很好的思路，但是互相讨论交流后慢慢地将每个人的想法进行比较融合最后得到正确的答案，因此能够顺利地完成任务整个实验也非常感谢老师和同学们。

非常高兴顺利地完成了实验，数据库在当下的应用可以说是无处不在，在以后这门课上学到的知识应该也会有很大的作用，我也会继续努力学习更多的知识。

A 附录

A.1 数据库管理系统源码

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

import os
import pymysql
import pandas as pd
from tabulate import tabulate

def print_menu():
    print(''''
    ***** Menu
    *****
    1. Insert New Student Info          2. Modify Existing Student
Info
    3. Insert New Course Info          4. Modify Existing Course
Info
    5. Delete Existing Course Info      6. Record Student Grade
    7. Modify Student Grade             8. Print Analytical Info by
Dept
    9. Print Grade Rank by Dept         10. Print Student Info

                                0. Exit

    *****
    ''')
    return

def insert_student(db):
    cursor = db.cursor()
    dic_student = {'sno': [], 'sname': [], 'sex': [], 'age': [], 'dept':
[], 'scholarship': []}
    print("===== Mode 1: Insert New Student Info
=====")
    print("Please input the following information! ")

    dic_student['sno'].append(input("Student ID: "))
    dic_student['sname'].append(input("Name: "))
    dic_student['sex'].append(input("Gender: "))
    dic_student['age'].append(input("Age: "))
    dic_student['dept'].append(input("Student's Department: "))
    dic_student['scholarship'].append("是" if input("Have Scholarship?
(y/n)") == 'y' else "否")

    df = pd.DataFrame(dic_student)

    print("Re-check the student's information below: ")
    print(tabulate(df, headers='keys', tablefmt='psql'))

    op = input("ARE YOU SURE TO INSERT THE NEW INFORMATION? (y/n)")
    if op == 'n':
        print("rollback successfully! ")
        return
```



```

else:
    pass

sql = "INSERT INTO Student " \
      "VALUES('%s', '%s', '%s', %s, '%s', '%s')" % (
          dic_student['sno'][0], dic_student['sname'][0],
dic_student['sex'][0], dic_student['age'][0],
          dic_student['dept'][0],
          dic_student['scholarship'][0])

try:
    cursor.execute(sql)
    db.commit()
    print("Successfully Inserted! ")
except:
    print("An unexpected error occurred and we rollback all the changes.
")
    db.rollback()

return

def update_student(db):
    cursor = db.cursor()
    dic_student = {'sno': [], 'sname': [], 'sex': [], 'age': [], 'dept':
[], 'scholarship': []}
    print("===== Mode 2: Modify Existing Student Info
=====")
    print("Please input the Student ID to search the student! ")

    dic_student['sno'].append(input("Student ID: "))

    sql = "SELECT * FROM Student WHERE Sno = '%s'" % dic_student['sno'][0]

    try:
        cursor.execute(sql)
        results = cursor.fetchall()
        for index, row in enumerate(results):
            # dic_student['sno'].append(row[0])
            dic_student['sname'].append(row[1])
            dic_student['sex'].append(row[2])
            dic_student['age'].append(row[3])
            dic_student['dept'].append(row[4])
            dic_student['scholarship'].append(row[5])

        print("Student founded! Here is the information of this person:
")
        df = pd.DataFrame(dic_student)
        print(tabulate(df, headers='keys', tablefmt='psql'))
    except:
        print("Error: unable to fetch data")
        return

    header_list = ['', 'Sno', 'Sname', 'Ssex', 'Sage', 'Sdept',
'Scholarship']

    while True:
        modify_op = input("Select column 1~6 to modify, select 0 to exit:
")
        if modify_op == '0':
            break

```

```

        else:
            pass
        new_data = input("Input the new data: ")

        sql = "UPDATE Student SET %s = " % header_list[int(modify_op)]
        if modify_op != 4:
            sql += "'"
        sql += new_data
        if modify_op != 4:
            sql += "'"
        sql += "WHERE Sno = '%s'" % dic_student['sno'][0]

        try:
            print("Successfully Updated! ")
            cursor.execute(sql)
            db.commit()
        except:
            print("update error! ")
            db.rollback()
            return

    return

def insert_course(db):
    cursor = db.cursor()
    dic_course = {'cno': [], 'cname': [], 'cpno': [], 'ccredit': []}
    print("===== Mode 3: Insert New Course Info =====")
    print("Please input the following information! ")

    dic_course['cno'].append(input("Course ID: "))
    dic_course['cname'].append(input("Course Name: "))
    dic_course['cpno'].append(input("Previous Course ID: "))
    dic_course['ccredit'].append(input("Credits: "))

    df = pd.DataFrame(dic_course)

    print("Re-check the course's information below: ")
    print(tabulate(df, headers='keys', tablefmt='psql'))

    op = input("ARE YOU SURE TO INSERT THE NEW INFORMATION? (y/n)")
    if op == 'n':
        print("rollback successfully! ")
        return
    else:
        pass

    sql = "INSERT INTO Course " \
          "VALUES('%s', '%s', '%s', %s)" % (
              dic_course['cno'][0], dic_course['cname'][0],
              dic_course['cpno'][0], dic_course['ccredit'][0])

    try:
        cursor.execute(sql)
        db.commit()
        print("Successfully Inserted! ")
    except:
        print("An unexpected error occurred and we rollback all the changes. ")
        db.rollback()

```

```

return

def modify_course(db):
    cursor = db.cursor()
    dic_course = {'cno': [], 'cname': [], 'cpno': [], 'ccredit': []}
    print("===== Mode 4: Modify Existing Course Info =====")
    print("Please input the Course ID to search the course! ")

    dic_course['cno'].append(input("Course ID: "))

    sql = "SELECT * FROM Course WHERE Cno = '%s'" % dic_course['cno'][0]

    try:
        cursor.execute(sql)
        results = cursor.fetchall()
        for index, row in enumerate(results):
            # dic_course['cno'].append(row[0])
            dic_course['cname'].append(row[1])
            dic_course['cpno'].append(row[2])
            dic_course['ccredit'].append(row[3])

        print("Course founded! Here is the information of this course: ")
        df = pd.DataFrame(dic_course)
        print(tabulate(df, headers='keys', tablefmt='psql'))
    except:
        print("Error: unable to fetch data")
        return

    header_list = ['', 'Cno', 'Cname', 'Cpno', 'Ccredit']

    while True:
        modify_op = input("Select column 1~4 to modify, select 0 to exit: ")

        if modify_op == '0':
            break
        else:
            pass
        new_data = input("Input the new data: ")

        sql = "UPDATE Course SET %s = " % header_list[int(modify_op)]
        if modify_op != 4:
            sql += "'"
            sql += new_data
        if modify_op != 4:
            sql += "'"
        sql += "WHERE Cno = '%s'" % dic_course['cno'][0]

        try:
            print("Successfully Updated! ")
            cursor.execute(sql)
            db.commit()
        except:
            print("update error! ")
            db.rollback()
            return

    return

```

```

def delete_course(db):
    cursor = db.cursor()
    dic_course = {'cno': [], 'cname': [], 'cpno': [], 'ccredit': []}
    print("===== Mode 5: Delete Existing Course Info =====")
    print("Please input the Course ID to search the course! ")

    dic_course['cno'].append(input("Course ID: "))

    sql = "SELECT * FROM Course WHERE Cno = '%s'" % dic_course['cno'][0]

    try:
        cursor.execute(sql)
        results = cursor.fetchall()
        for index, row in enumerate(results):
            # dic_course['cno'].append(row[0])
            dic_course['cname'].append(row[1])
            dic_course['cpno'].append(row[2])
            dic_course['ccredit'].append(row[3])

        print("Course founded! Here is the information of this course: ")
        df = pd.DataFrame(dic_course)
        print(tabulate(df, headers='keys', tablefmt='psql'))
    except:
        print("Error: unable to fetch data")
        return

    header_list = ['', 'Cno', 'Cname', 'Cpno', 'Ccredit']

    confirm_op = input("ARE YOU SURE TO DELETE THIS COURSE? (y/n): ")
    if confirm_op == 'n':
        return
    else:
        pass

    sql = "DELETE FROM Course WHERE Cno = '%s'" % dic_course['cno'][0]

    try:
        print("Successfully Deleted! ")
        cursor.execute(sql)
        db.commit()
    except:
        print("delete error! ")
        db.rollback()
        return

    return

def record_grade(db):
    cursor = db.cursor()
    dic_sc = {'sno': [], 'cno': [], 'grade': []}
    print("===== Mode 6: Record Student's Grade =====")
    print("Please input the following information! ")

    dic_sc['sno'].append(input("Student ID: "))
    dic_sc['cno'].append(input("Course ID: "))
    dic_sc['grade'].append(input("Student's Grade: "))

```

```

df = pd.DataFrame(dic_sc)

print("Re-check the information below: ")
print(tabulate(df, headers='keys', tablefmt='psql'))

op = input("ARE YOU SURE TO INSERT THE NEW INFORMATION? (y/n)")
if op == 'n':
    print("rollback successfully! ")
    return
else:
    pass

sql = "INSERT INTO SC " \
      "VALUES('%s', '%s', %s)" % (
          dic_sc['sno'][0], dic_sc['cno'][0], dic_sc['grade'][0])

try:
    cursor.execute(sql)
    db.commit()
    print("Successfully Inserted! ")
except:
    print("An unexpected error occurred and we rollback all the changes.
")
    db.rollback()

return

def modify_grade(db):
    cursor = db.cursor()
    dic_sc = {'sno': [], 'cno': [], 'grade': []}
    print("===== Mode 7: Modify Student's Grade
=====")
    print("Please input the Student ID and Course ID to search the grade
record! ")

    dic_sc['sno'].append(input("Student ID: "))
    dic_sc['cno'].append(input("Course ID: "))

    sql = "SELECT * FROM SC WHERE Sno = '%s' AND Cno = '%s'" %
(dic_sc['sno'][0], dic_sc['cno'][0])

    try:
        cursor.execute(sql)
        results = cursor.fetchall()
        for index, row in enumerate(results):
            # dic_sc['sno'].append(row[0])
            # dic_sc['cno'].append(row[1])
            dic_sc['grade'].append(row[2])

        print("Course founded! Here is the information of this course: ")
        df = pd.DataFrame(dic_sc)
        print(tabulate(df, headers='keys', tablefmt='psql'))
    except:
        print("Error: unable to fetch data")
        return

header_list = ['', 'Sno', 'Cno', 'Grade']

modify_op = input("Are you sure to modify this score? (y/n): ")
if modify_op == 'n':

```

```

        return
    else:
        pass
    new_data = input("Input the new grade: ")

    sql = "UPDATE SC SET Grade = %s WHERE Sno = '%s' AND Cno = '%s'" %
(new_data, dic_sc['sno'][0], dic_sc['cno'][0])

    try:
        print("Successfully Updated! ")
        cursor.execute(sql)
        db.commit()
    except:
        print("update error! ")
        db.rollback()
        return

    return

def analyze_grade(db):
    cursor = db.cursor()
    dic_sc = {'dept': [], 'avg': [], 'max': [], 'min': [], 'a_rate': [],
'f_num': []}
    l_dept = []
    print("===== Mode 8: Print Analytical Info by Dept
=====")
    # print("Please input the Student ID and Course ID to search the grade
record! ")

    # Step 1. acquire all of the dept names
    sql = "SELECT DISTINCT Sdept FROM Student ORDER BY Sdept"
    try:
        cursor.execute(sql)
        results = cursor.fetchall()
        for row in results:
            l_dept.append(row[0])
    except:
        print("Error: unable to fetch department data")
        return

    print("Department List: " + ",".join(str(x) for x in l_dept))

    # Step 2. get the data of each dept
    for dept in l_dept:
        dic_sc['dept'].append(dept)

        # fetch the avg, max, min number
        sql = "SELECT AVG(SC.Grade), MAX(SC.Grade), MIN(SC.Grade) FROM
Student, SC WHERE Student.Sdept = '%s' AND " \
            "Student.Sno = SC.Sno" % dept
        try:
            cursor.execute(sql)
            results = cursor.fetchall()
            for row in results:
                dic_sc['avg'].append(row[0])
                dic_sc['max'].append(row[1])
                dic_sc['min'].append(row[2])
        except:
            print("Error: unable to fetch avg/max/min data")
            return

```

```

a_number = 0
total_number = 0

# fetch the A number
sql = "SELECT COUNT(*) FROM Student, SC WHERE Student.Sdept = '%s'
AND Student.Sno = SC.Sno AND SC.Grade >= 90" % dept
try:
    cursor.execute(sql)
    results = cursor.fetchall()
    for row in results:
        a_number = row[0]
except:
    print("Error: unable to fetch avg/max/min data")
    return

# fetch all dept student number
sql = "SELECT COUNT(*) FROM Student, SC WHERE Student.Sdept = '%s'
AND Student.Sno = SC.Sno" % dept
try:
    cursor.execute(sql)
    results = cursor.fetchall()
    for row in results:
        total_number = row[0]
except:
    print("Error: unable to fetch total student number data")
    return

if total_number != 0:
    a_rate = a_number / total_number
else:
    a_rate = 0.0
dic_sc['a_rate'].append(a_rate)

fail_num = 0

# fetch the failed students number
sql = "SELECT COUNT(*) FROM Student, SC WHERE Student.Sdept = '%s'
AND Student.Sno = SC.Sno AND SC.Grade < 60" % dept
try:
    cursor.execute(sql)
    results = cursor.fetchall()
    for row in results:
        fail_num = row[0]
except:
    print("Error: unable to fetch avg/max/min data")
    return

dic_sc['f_num'].append(fail_num)

# Step 3. display the information
print("Here is the information for your inquire: ")
df = pd.DataFrame(dic_sc)
print(tabulate(df, headers='keys', tablefmt='psql'))
input("Press Enter to continue...")
return

def rank_student(db):
    cursor = db.cursor()
    l_dept = []

```

```

print("===== Mode 9: Print Department Student Ranking
=====")
# print("Please input the Student ID and Course ID to search the grade
record! ")

# Step 1. acquire all of the dept names
sql = "SELECT DISTINCT Sdept FROM Student ORDER BY Sdept"
try:
    cursor.execute(sql)
    results = cursor.fetchall()
    for row in results:
        l_dept.append(row[0])
except:
    print("Error: unable to fetch department data")
    return

while True:
    dic_sc = {'sno': [], 'sname': [], 'avg': [], 'c_num': [], 'credits':
[]}]
    print("Department List: " + ",".join(str(x) for x in l_dept))
    while True:
        sel_dept = input("Please select one department to inquire the
student's ranking: ")
        if sel_dept in l_dept:
            break
        else:
            print('Department %s not found! ' % sel_dept)

    # dic_sc = {'sno': [], 'sname': [], 'avg': [], 'c_num': [], 'credits':
[]}]

    # Step 2. get the data of the chosen dept
    sql = "SELECT Student.Sno, Student.Sname, AVG(SC.Grade),
COUNT(SC.Grade), SUM(Course.Ccredit) FROM Student, SC, " \
        "Course WHERE Student.Sdept = '%s' AND SC.Sno = Student.Sno
AND Course.Cno = SC.Cno GROUP BY Student.Sno " \
        "ORDER BY AVG(SC.Grade) DESC" % sel_dept
    try:
        cursor.execute(sql)
        results = cursor.fetchall()
        for row in results:
            dic_sc['sno'].append(row[0])
            dic_sc['sname'].append(row[1])
            dic_sc['avg'].append(row[2])
            dic_sc['c_num'].append(row[3])
            dic_sc['credits'].append(row[4])
    except:
        print("Error: unable to fetch student's data")
        return

    # Step 3. display the information
    print("Here is the ranking of %s department: " % sel_dept)
    df = pd.DataFrame(dic_sc)
    print(tabulate(df, headers='keys', tablefmt='psql'))
    sel_op = input("Do you want to search other department's rank?
(y/n) ")
    if sel_op == 'n':
        break
    else:
        pass

return

```



```

def show_stu_info(db):
    cursor = db.cursor()
    dic_student = {'sno': [], 'sname': [], 'sex': [], 'age': [], 'dept':
[], 'scholarship': []}
    dic_course = {'cno': [], 'cname': [], 'grade': [], 'credits': [],
'cpno': []}
    print("===== Mode 10: Lookup Student Basic Info
=====")
    print("Please input the Student ID to search the student! ")

    dic_student['sno'].append(input("Student ID: "))

    sql = "SELECT * FROM Student WHERE Sno = '%s'" % dic_student['sno'][0]

    try:
        cursor.execute(sql)
        results = cursor.fetchall()
        for index, row in enumerate(results):
            # dic_student['sno'].append(row[0])
            dic_student['sname'].append(row[1])
            dic_student['sex'].append(row[2])
            dic_student['age'].append(row[3])
            dic_student['dept'].append(row[4])
            dic_student['scholarship'].append(row[5])

        print("Student founded! Here is the information of this person:
")

        df = pd.DataFrame(dic_student)
        print(tabulate(df, headers='keys', tablefmt='psql'))
    except:
        print("Error: unable to fetch data")
        return

    # definition: dic_course = {'cno': [], 'cname': [], 'grade': [],
'credits': [], 'cpno': []}
    sql = "SELECT SC.Cno, Course.Cname, SC.Grade, Course.Ccredit,
Course.Cpno FROM SC, Course WHERE SC.Sno = '%s' AND " \
        "SC.Cno = Course.Cno" % dic_student['sno'][0]

    try:
        cursor.execute(sql)
        results = cursor.fetchall()
        for index, row in enumerate(results):
            dic_course['cno'].append(row[0])
            dic_course['cname'].append(row[1])
            dic_course['grade'].append(row[2])
            dic_course['credits'].append(row[3])
            dic_course['cpno'].append(row[4])

        print("Student's Course Info: ")
        df = pd.DataFrame(dic_course)
        print(tabulate(df, headers='keys', tablefmt='psql'))
        input("Press Enter to continue ... ")
    except:
        print("Error: unable to fetch course data")
        return

    return

```

```

def main():
    ip_addr = input("Input the IP Address (default: localhost): ") or "localhost"
    usr = input("Username (default: yuanye): ") or "yuanye"
    passwd = input("Password (default: yuanye): ") or "yuanye"
    select_db = input("Select one database to operate (default: S_T_U201911808): ") or "S_T_U201911808"

    print(" ... Connecting to the Database ... ")

    db = pymysql.connect(
        host=ip_addr,
        user=usr,
        password=passwd,
        database=select_db
    )

    cursor = db.cursor()

    cursor.execute("SELECT VERSION()")

    db_version = cursor.fetchone()

    print("----- Student Management System -----")
    print("          Author: 网安1902班 袁也  U201911808")
    print("          MySQL Version: {}".format(db_version[0]))
    print("   User: {}@{}   Database: {}".format(usr, ip_addr, select_db))

    while True:
        print_menu()
        op = input("Choose what you want (0~10): ")
        if op == '0':
            break
        elif op == '1':
            insert_student(db)
        elif op == '2':
            update_student(db)
        elif op == '3':
            insert_course(db)
        elif op == '4':
            modify_course(db)
        elif op == '5':
            delete_course(db)
        elif op == '6':
            record_grade(db)
        elif op == '7':
            modify_grade(db)
        elif op == '8':
            analyze_grade(db)
        elif op == '9':
            rank_student(db)
        elif op == '10':
            show_stu_info(db)
        else:
            print("Wrong Input! ")

    db.close()
    print("Thanks for using this database! Bye!")

```

```
if __name__ == '__main__':  
    main()
```