《数据库系统及应用实践》课程实验报告

实验 2：SQL 练习

姓　名:	李鹏达	学　号:	10225101460	完成日期:	2024 年 3 月 21 日

1 实验目标

1. 学习和掌握 MySQL 数据库管理系统中 SQL 的基本语法
2. 能够编写 SQL 语句完成指定的查询

2 实验过程记录

2.1 创建数据表并导入数据

首先下载实验 2 中的两个 SQL 脚本文件 DDL+drop.sql 和 largeRelationsInsertFile.sql 然后运行在实验一中创建的数据库实例，并启动其中的 bash shell

```
sudo docker run --name dbcourse -v ./datadir:/var/lib/mysql -d -p 53306:3306
    dbcourse:v1
sudo docker exec -it dbcourse bash
```

图 1: 运行数据库实例并启动 bash shell

接着创建一个 dbcourse 文件夹，并 cd 到该文件夹下

```
mkdir dbcourse
cd dbcourse
pwd
```

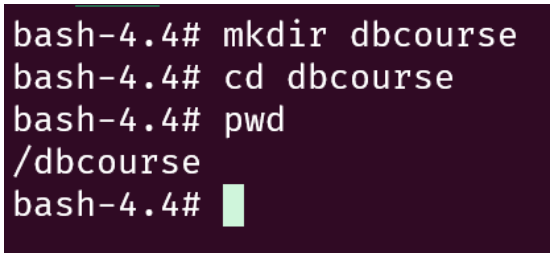


图 2: 创建并进入文件夹

接着，在另一个终端中，将下载的两个 SQL 文件复制到容器内的 dbcourse 文件夹下

```
sudo docker cp ./DDL+drop.sql dbcourse:/dbcourse/
sudo docker cp ./largeRelationsInsertFile.sql.sql dbcourse:/dbcourse/
```

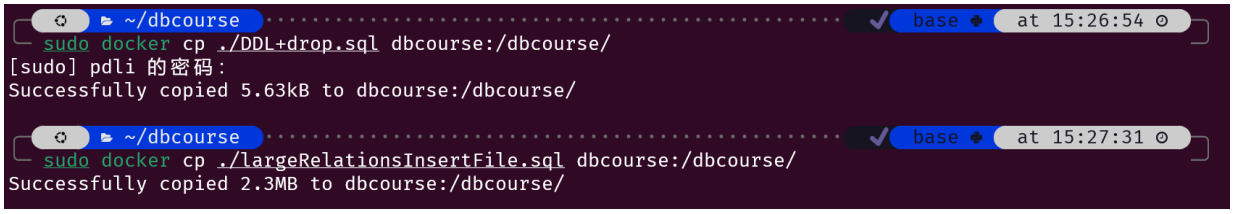


图 3: 复制 SQL 文件到容器内

确认文件已经复制到容器内后，进入容器内的 MySQL 数据库

```
ls -l
mysql -u root -p -D dbcourse
```

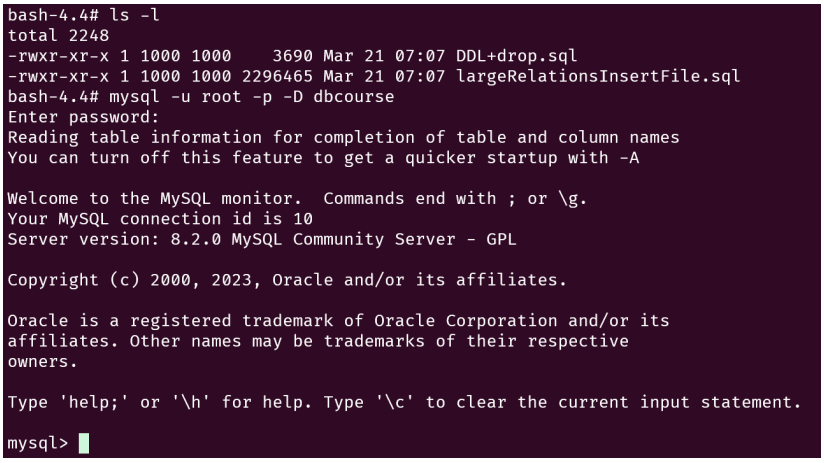


图 4: 进入 MySQL 数据库

运行 DDL+drop.sql 文件，创建数据表，然后运行 largeRelationsInsertFile.sql 文件，导入数据

```
source DDL+drop.sql;
source largeRelationsInsertFile.sql;
```

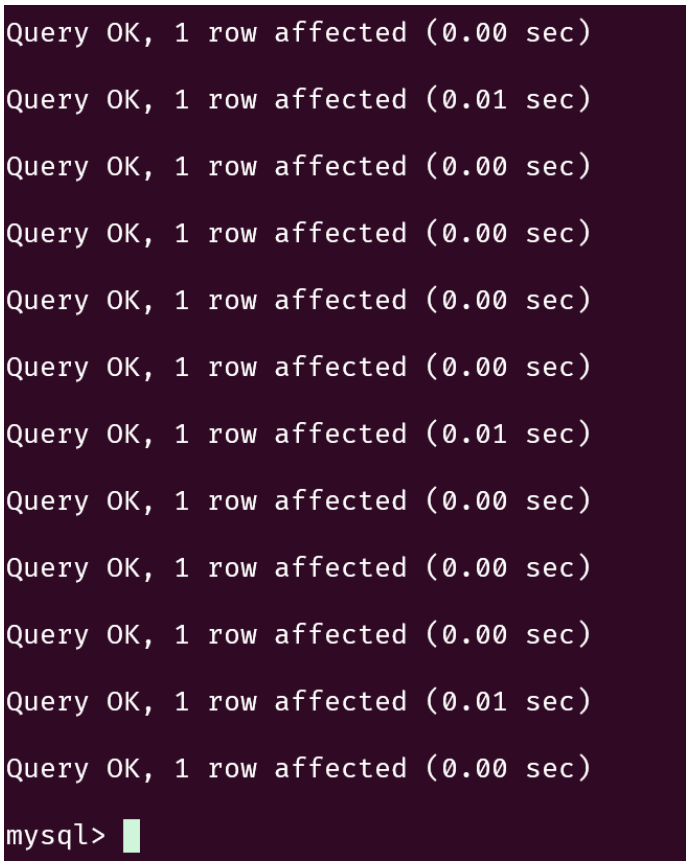


图 5: 创建数据表并导入数据

执行下列 SQL 语句，确认每张数据表中的记录数正确

```
select count(*) from advisor;
select count(*) from classroom;
select count(*) from course;
select count(*) from department;
select count(*) from instructor;
select count(*) from prereq;
select count(*) from section;
select count(*) from student;
select count(*) from takes;
select count(*) from teaches;
```

```
select count(*) from time_slot;
```

结果为 2000，30，200，20，50，100，118，2000，30000，116，20，分别对应每张数据表中的记录数，说明数据表创建和数据导入成功。

使用 exit 命令退出 MySQL 数据库

```
exit;
```

执行下列命令,通过 MySQL 的命令行客户端执行单条 SQL 语句,并将查询结果保存到文件 department.csv 中

```
mysql -u root -p -D dbcourse -e "select * from department;" > department.csv
cat department.csv
```

```
bash-4.4# mysql -u root -p -D dbcourse -e "select * from department" > department.csv
Enter password:
bash-4.4# cat department.csv
dept_name       building        budget
Accounting      Saucon  441840.92
Astronomy       Taylor  617253.94
Athletics       Bronfman        734550.70
Biology Candlestick     647610.55
Civil Eng.      Chandler        255041.46
Comp. Sci.      Lamberton       106378.69
Cybernetics     Mercer  794541.46
Elec. Eng.      Main    276527.61
English Palmer  611042.66
Finance Candlestick     866831.75
Geology Palmer  406557.93
History Taylor  699140.86
Languages       Linderman       601283.60
Marketing       Lambeau 210627.58
Math    Brodhead        777605.11
Mech. Eng.      Rauch   520350.65
Physics Wrigley 942162.76
Pol. Sci.       Whitman 573745.09
Psychology      Thompson        848175.04
Statistics      Taylor  395051.74
```

图 6: 执行 SQL 语句并将结果保存到文件

2.2 SQL 练习

1. Find the names of all the instructors from Biology department.

```
SELECT `name` FROM `instructor` WHERE `dept_name` = 'Biology';
```

结果如下：

```
mysql> SELECT `name` FROM `instructor` WHERE `dept_name` = 'Biology';
+----------+
| name     |
+----------+
| Queiroz  |
| Valtchev |
+----------+
2 rows in set (0.00 sec)
```

图 7: 查询结果

2. Find the names of courses in Computer Science department which have 3 credits.

```
SELECT `title` FROM `course` WHERE `dept_name` = 'Comp. Sci.' AND `credits` =
    3;
```

结果如下：

```
mysql> SELECT `title` FROM `course` WHERE `dept_name` = 'Comp. Sci.' AND `credits` = 3;
+----------------------+
| title                |
+----------------------+
| International Finance |
| Computability Theory  |
| Japanese             |
+----------------------+
3 rows in set (0.03 sec)
```

图 8: 查询结果

3. For the student with ID 13403 (or any other value), show all course_id and title of all courses taken by the student.

```
SELECT `course_id`, `title` FROM `takes` NATURAL JOIN `course` WHERE `ID` =
    13403;
```

结果如下：

```
mysql> SELECT `course_id`, `title` FROM `takes` NATURAL JOIN `course` WHERE `ID` = 13403;
+-----------+-------------------------------+
| course_id | title                         |
+-----------+-------------------------------+
| 158       | Elastic Structures            |
| 192       | Drama                         |
| 258       | Colloid and Surface Chemistry |
| 319       | World History                 |
| 338       | Graph Theory                  |
| 349       | Networking                    |
| 352       | Compiler Design               |
| 366       | Computational Biology         |
| 400       | Visual BASIC                  |
| 400       | Visual BASIC                  |
| 443       | Journalism                    |
| 486       | Accounting                    |
| 489       | Journalism                    |
| 493       | Music of the 50s              |
| 696       | Heat Transfer                 |
| 748       | Tort Law                      |
| 795       | Death and Taxes               |
+-----------+-------------------------------+
17 rows in set (0.01 sec)
```

图 9: 查询结果

4. As above, but show the total number of credits for such courses (taken by that student).

```
SELECT `course_id`, `title`, `credits` FROM `takes` NATURAL JOIN `course` WHERE
    `ID` = 13403;
```

结果如下：

```
mysql> SELECT `course_id`, `title`, `credits` FROM `takes` NATURAL JOIN `course` WHERE `ID` = 13403;
+-----------+-------------------------------+---------+
| course_id | title                         | credits |
+-----------+-------------------------------+---------+
| 158       | Elastic Structures            |       3 |
| 192       | Drama                         |       4 |
| 258       | Colloid and Surface Chemistry |       3 |
| 319       | World History                 |       4 |
| 338       | Graph Theory                  |       3 |
| 349       | Networking                    |       4 |
| 352       | Compiler Design               |       4 |
| 366       | Computational Biology         |       3 |
| 400       | Visual BASIC                  |       4 |
| 400       | Visual BASIC                  |       4 |
| 443       | Journalism                    |       4 |
| 486       | Accounting                    |       3 |
| 489       | Journalism                    |       4 |
| 493       | Music of the 50s              |       3 |
| 696       | Heat Transfer                 |       4 |
| 748       | Tort Law                      |       4 |
| 795       | Death and Taxes               |       3 |
+-----------+-------------------------------+---------+
17 rows in set (0.01 sec)
```

图 10: 查询结果

5. Display the total credits for each of the students, along with the ID of the student.

```sql
SELECT `ID`, SUM(`credits`) AS `total_credits` FROM `takes` NATURAL JOIN `course` GROUP BY `ID`;
```

结果如下：

```
| 9933  |              45 |
| 99348 |              74 |
| 99369 |              45 |
| 99399 |              61 |
| 99422 |              55 |
| 99451 |              49 |
| 99463 |              41 |
| 9947  |              56 |
| 9953  |              66 |
| 99553 |              45 |
| 99611 |              48 |
| 99647 |              40 |
| 99660 |              53 |
| 99694 |              64 |
| 99710 |              46 |
| 99711 |              36 |
| 99719 |              71 |
| 99730 |              57 |
| 99754 |              56 |
| 99760 |              70 |
| 99764 |              53 |
| 99775 |              29 |
| 99780 |              53 |
| 9993  |              69 |
| 99949 |              44 |
| 99977 |              61 |
+-------+-----------------+
2000 rows in set (0.04 sec)
```

图 11: 查询结果

6. Find the names of all students who have taken any Comp. Sci. course ever (there should be no duplicate names).

```sql
SELECT DISTINCT `name` FROM `student` NATURAL JOIN `takes` NATURAL JOIN `course` WHERE `dept_name` = 'Comp. Sci.';
```

结果如下：

```
mysql> SELECT DISTINCT `name` FROM `student` NATURAL JOIN `takes` NATURAL JOIN `course` WHERE `dept_name` = 'Comp. Sci.';
+-----------+
| name      |
+-----------+
| Oswald    |
| Miao      |
| Haigh     |
| Singhal   |
| Kuwadak   |
| Pampal    |
| Cai       |
| Sahani    |
| Brookh    |
| Jessup    |
| Hunter    |
| Crick     |
| Bondi     |
| Kawakami  |
| Nives     |
| Maglioni  |
| Ploski    |
| Rioult    |
| Arinb     |
| Gilmour   |
| Tleu      |
| Albinal   |
| Wakamiya  |
| Macias    |
| Lagendijk |
| Carvey    |
| Du        |
| Pelletier |
| Guyer     |
| Enokib    |
| Saad      |
| Goldman   |
| Chenu     |
| Senn      |
| Konno     |
| Thoreson  |
| Jawad     |
| Morton    |
| Kurt      |
| Paddock   |
| Varghese  |
| Grant     |
| Caporali  |
| Wunderli  |
| Sakanushi |
| Zuo       |
| Zubai     |
| Schill    |
| Schrefl   |
| Olin      |
| Rammer    |
| Salzman   |
| Mathias   |
| Frolova   |
| Bartels   |
+-----------+
55 rows in set (0.09 sec)
```

图 12: 查询结果

7. Display the IDs of all instructors who have never taught a couse (interpret "taught" as "taught or is scheduled to teach").

```sql
SELECT `ID` FROM `instructor` NATURAL LEFT JOIN `teaches` WHERE `teaches`.`ID`
    IS NULL;
```

结果如下:

```
mysql> SELECT `ID` FROM `instructor` NATURAL LEFT JOIN `teaches` WHERE `teaches`.`ID` IS NULL;
+-------+
| ID    |
+-------+
| 31955 |
| 57180 |
| 16807 |
| 40341 |
| 63395 |
| 79653 |
| 72553 |
| 50885 |
| 59795 |
| 58558 |
| 74426 |
| 96896 |
| 97302 |
| 52647 |
| 35579 |
| 37687 |
| 64871 |
| 78699 |
| 95030 |
+-------+
19 rows in set (0.00 sec)
```

图 13: 查询结果

8. As above, but display the names of the instructors also, not just the IDs.

```sql
SELECT `ID`, `name` FROM `instructor` NATURAL LEFT JOIN `teaches` WHERE `teaches`.`ID` IS NULL;
```

结果如下:

```
mysql> SELECT `ID`, `name` FROM `instructor` NATURAL LEFT JOIN `teaches` WHERE `teaches`.`ID` IS NULL;
+-------+--------------------+
| ID    | name               |
+-------+--------------------+
| 16807 | Yazdi              |
| 31955 | Moreira            |
| 35579 | Soisalon-Soininen  |
| 37687 | Arias              |
| 40341 | Murata             |
| 50885 | Konstantinides     |
| 52647 | Bancilhon          |
| 57180 | Hau                |
| 58558 | Dusserre           |
| 59795 | Desyl              |
| 63395 | McKinnon           |
| 64871 | Gutierrez          |
| 72553 | Yin                |
| 74426 | Kenje              |
| 78699 | Pingr              |
| 79653 | Levine             |
| 95030 | Arinb              |
| 96896 | Mird               |
| 97302 | Bertolino          |
+-------+--------------------+
19 rows in set (0.00 sec)
```

图 14: 查询结果

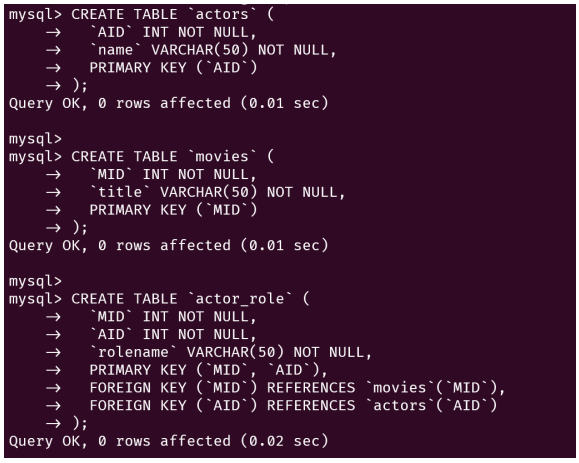
9. You need to create a movie database. Create three tables, one for actors(AID, name), one for movies(MID, title) and one for actor_role(MID, AID, rolename). Use appropriate data types for each of the attributes, and add appropriate primary/foreign key constraints.

```sql
CREATE DATABASE `movie`;

CREATE TABLE `actors` (
  `AID` INT NOT NULL,
  `name` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`AID`)
);

CREATE TABLE `movies` (
  `MID` INT NOT NULL,
  `title` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`MID`)
);

CREATE TABLE `actor_role` (
  `MID` INT NOT NULL,
  `AID` INT NOT NULL,
  `rolename` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`MID`, `AID`),
  FOREIGN KEY (`MID`) REFERENCES `movies`(`MID`),
  FOREIGN KEY (`AID`) REFERENCES `actors`(`AID`)
);
```



```
mysql> CREATE TABLE `actors` (
    -> `AID` INT NOT NULL,
    -> `name` VARCHAR(50) NOT NULL,
    -> PRIMARY KEY (`AID`)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> CREATE TABLE `movies` (
    -> `MID` INT NOT NULL,
    -> `title` VARCHAR(50) NOT NULL,
    -> PRIMARY KEY (`MID`)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> CREATE TABLE `actor_role` (
    -> `MID` INT NOT NULL,
    -> `AID` INT NOT NULL,
    -> `rolename` VARCHAR(50) NOT NULL,
    -> PRIMARY KEY (`MID`, `AID`),
    -> FOREIGN KEY (`MID`) REFERENCES `movies`(`MID`),
    -> FOREIGN KEY (`AID`) REFERENCES `actors`(`AID`)
    -> );
Query OK, 0 rows affected (0.02 sec)
```

图 15: 创建数据表

10. Insert data to the above tables (approx 3 to 6 rows in each table), including data for actor "Charlie Chaplin", and for yourself (using your student number as ID).

```sql
INSERT INTO `actors` (`AID`, `name`) VALUES (1, 'Charlie Chaplin'), (2, 'John Doe'), (460, 'Pengda Li');

INSERT INTO `movies` (`MID`, `title`) VALUES (1, 'The Great Dictator'), (2, 'Modern Times'), (3, 'City Lights');

INSERT INTO `actor_role` (`MID`, `AID`, `rolename`) VALUES (1, 1, 'Adenoid Hynkel'), (2, 1, 'The Tramp'), (3, 1, 'A Tramp'), (1, 2, 'Extra'), (2, 2, 'Extra'), (3, 460, 'Director');
```

```
mysql> INSERT INTO `actors` (`AID`, `name`) VALUES (1, 'Charlie Chaplin'), (2, 'John Doe'), (460, 'Pengda
 Li');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
mysql> INSERT INTO `movies` (`MID`, `title`) VALUES (1, 'The Great Dictator'), (2, 'Modern Times'), (3, '
City Lights');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
mysql> INSERT INTO `actor_role` (`MID`, `AID`, `rolename`) VALUES (1, 1, 'Adenoid Hynkel'), (2, 1, 'The T
ramp'), (3, 1, 'A Tramp'), (1, 2, 'Extra'), (2, 2, 'Extra'), (3, 460, 'Director');
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

图 16: 插入数据

11. Write a query to list all movies in which actor "Charlie Chaplin" has acted, along with the number of roles he had in that movie.

```sql
SELECT `title`, COUNT(`rolename`) AS `roles` FROM `movies` NATURAL JOIN `actor_role` NATURAL JOIN `actors` WHERE `name` = 'Charlie Chaplin' GROUP BY `MID`;
```

结果如下:

```
mysql> SELECT `title`, COUNT(`rolename`) AS `roles` FROM `movies` NATURAL JOIN `actor_role` NATURAL JOIN
`actors` WHERE `name` = 'Charlie Chaplin' GROUP BY `MID`;
+---------------------+-------+
| title               | roles |
+---------------------+-------+
| The Great Dictator |     1 |
| Modern Times        |     1 |
| City Lights         |     1 |
+---------------------+-------+
3 rows in set (0.01 sec)
```

图 17: 查询结果

12. Write a query to list the names of actors who have not acted in any movie.

```
SELECT `name` FROM `actors` NATURAL LEFT JOIN `actor_role` WHERE `actor_role`.`MID` IS NULL;
```

结果如下:

```
mysql> SELECT `name` FROM `actors` NATURAL LEFT JOIN `actor_role` WHERE `actor_role`.`MID` IS NULL;
+-----------+
| name      |
+-----------+
| San Zhang |
+-----------+
1 row in set (0.00 sec)
```

图 18: 查询结果

13. List names of actors, along with titles of movies they have acted in. If they have not acted in any movie, show the movie title as null. (Do not use SQL outerjoin syntax here, write it from scratch.)

```
SELECT `name`, `title` FROM `actors` NATURAL LEFT JOIN `actor_role` NATURAL LEFT JOIN `movies`;
```

结果如下:

```
mysql> SELECT `name`, `title` FROM `actors` NATURAL LEFT JOIN `actor_role` NATURAL LEFT JOIN `movies`;
+----------------+---------------------+
| name           | title               |
+----------------+---------------------+
| Charlie Chaplin | The Great Dictator |
| Charlie Chaplin | Modern Times        |
| Charlie Chaplin | City Lights         |
| John Doe       | The Great Dictator |
| John Doe       | Modern Times        |
| Pengda Li      | City Lights         |
| San Zhang      | NULL                |
+----------------+---------------------+
7 rows in set (0.00 sec)
```

图 19: 查询结果

14. Find the maximum and minimum enrollment across all sections, considering only sections that had some enrollment, don’ t worry about those that had no students taking that section.

```
WITH `enrollment` AS (
  SELECT `sec_id`, COUNT(`ID`) AS `enrollment` FROM `takes` GROUP BY `sec_id`
) SELECT MAX(`enrollment`) AS `max_enrollment`, MIN(`enrollment`) AS `
    min_enrollment` FROM `enrollment`;
```

结果如下：

```
mysql> WITH `enrollment` AS (
CT `sec_id`,     →   SELECT `sec_id`, COUNT(`ID`) AS `enrollment` FROM `takes` GROUP BY `sec_id`
    → ) SELECT MAX(`enrollment`) AS `max_enrollment`, MIN(`enrollment`) AS `min_enrollment` FROM `enroll
ment`;
+----------------+----------------+
| max_enrollment | min_enrollment |
+----------------+----------------+
|          25512 |            322 |
+----------------+----------------+
1 row in set (0.03 sec)
```

图 20: 查询结果

15. Find all sections that had the maximum enrollment (along with the enrollment).

```
WITH `enrollment` AS (
  SELECT `sec_id`, COUNT(`ID`) AS `enrollment` FROM `takes` GROUP BY `sec_id`
) SELECT `sec_id`, `enrollment` FROM `enrollment` WHERE `enrollment` = (SELECT
    MAX(`enrollment`) FROM `enrollment`);
```

16. As in step 14, but now it also includes sections with no students taking them; the enrollment for such sections should be treated as 0. Do this in two different ways.

- Using a scalar subquery;

```
SELECT MAX(`enrollment`) AS `max_enrollment`, MIN(`enrollment`) AS `
    min_enrollment` FROM (SELECT `sec_id`, COALESCE((SELECT COUNT(`ID`) FROM
     `takes` WHERE `sec_id` = `section`.`sec_id`), 0) AS `enrollment` FROM `
    section`) AS `enrollments`;
```

结果如下：

```
mysql> WITH `enrollment` AS (
    -> SELECT `sec_id`, COUNT(`ID`) AS `enrollment` FROM `takes` GROUP BY `sec_id`
    -> ) SELECT `sec_id`, `enrollment` FROM `enrollment` WHERE `enrollment` = (SELECT MAX(`enrollment`) F
ROM `enrollment`);
+--------+------------+
| sec_id | enrollment |
+--------+------------+
| 1      |      25512 |
+--------+------------+
1 row in set (0.10 sec)
```

图 21: 查询结果

- Using aggregation on a left outer join (use the SQL natural left outer join syntax);

```sql
SELECT MAX(`enrollment`) AS `max_enrollment`, MIN(`enrollment`) AS `min_enrollment` FROM (SELECT `sec_id`, COUNT(`ID`) AS `enrollment` FROM `section` NATURAL LEFT JOIN `takes` GROUP BY `sec_id`) AS `enrollments`;
```

结果如下：

```
mysql> SELECT MAX(`enrollment`) AS `max_enrollment`, MIN(`enrollment`) AS `min_enrollment` FROM (SELECT `
sec_id`, COALESCE((SELECT COUNT(`ID`) FROM `takes` WHERE `sec_id` = `section`.`sec_id`), 0) AS `enrollmen
t` FROM `section`) AS `enrollments`;
+----------------+----------------+
| max_enrollment | min_enrollment |
+----------------+----------------+
|          25512 |              0 |
+----------------+----------------+
1 row in set (0.64 sec)
```

图 22: 查询结果

17. Find all courses whose title starts with the string “Comp”.

```sql
SELECT `title` FROM `course` WHERE `title` LIKE 'Comp%';
```

结果如下：

```
mysql> SELECT MAX(`enrollment`) AS `max_enrollment`, MIN(`enrollment`) AS `min_enrollment` FROM (SELECT `
sec_id`, COUNT(`ID`) AS `enrollment` FROM `section` NATURAL LEFT JOIN `takes` GROUP BY `sec_id`) AS `enro
llments`;
+----------------+----------------+
| max_enrollment | min_enrollment |
+----------------+----------------+
|          25512 |              0 |
+----------------+----------------+
1 row in set (0.03 sec)
```

图 23: 查询结果

18. Find instructors who have taught all courses in in Comp. Sci. Department.

- Using the “not exists ···except ···” structure;

```sql
SELECT `ID`, `name` FROM `instructor` WHERE NOT EXISTS (SELECT `course_id`
    FROM `course` WHERE `dept_name` = 'Comp. Sci.' EXCEPT SELECT `course_id`
     FROM `teaches` WHERE `teaches`.`ID` = `instructor`.`ID`);
```

结果如下：

```
mysql> SELECT `ID`, `name` FROM `instructor` WHERE NOT EXISTS (SELECT `course_id` FROM `course` WHERE `de
pt_name` = 'Comp. Sci.' EXCEPT SELECT `course_id` FROM `teaches` WHERE `teaches`.`ID` = `instructor`.`ID`
);
+-------+-------+
| ID    | name  |
+-------+-------+
| 34175 | Bondi |
+-------+-------+
1 row in set (0.02 sec)
```

图 24: 查询结果

- Using matching of counts (Don’ t forget the distinct clause!).

```sql
SELECT `ID`, `name` FROM `instructor` WHERE (SELECT COUNT(DISTINCT `
    course_id`) FROM `course` WHERE `dept_name` = 'Comp. Sci.') = (SELECT
    COUNT(DISTINCT `course_id`) FROM `teaches` NATURAL JOIN `course` WHERE `
    teaches`.`ID` = `instructor`.`ID` AND `dept_name` = 'Comp. Sci.');
```

结果如下：

```
mysql> SELECT `ID`, `name` FROM `instructor` WHERE (SELECT COUNT(DISTINCT `course_id`) FROM `course` WHER
E `dept_name` = 'Comp. Sci.') = (SELECT COUNT(DISTINCT `course_id`) FROM `teaches` NATURAL JOIN `course`
WHERE `teaches`.`ID` = `instructor`.`ID` AND `dept_name` = 'Comp. Sci.');
+-------+-------+
| ID    | name  |
+-------+-------+
| 34175 | Bondi |
+-------+-------+
1 row in set (0.03 sec)
```

图 25: 查询结果

19. Insert each instructor as a student, with tot_cred = 0, in the same department.

```sql
INSERT INTO `student` (`ID`, `name`, `dept_name`, `tot_cred`) SELECT `ID`, `
    name`, `dept_name`, 0 FROM `instructor`;
```

结果如下：

```
mysql> INSERT INTO `student` (`ID`, `name`, `dept_name`, `tot_cred`) SELECT `ID`, `name`, `dept_name`, 0
FROM `instructor`;
Query OK, 50 rows affected (0.03 sec)
Records: 50  Duplicates: 0  Warnings: 0
```

图 26: 插入数据

20. Now delete all the newly added "students" above.

```
DELETE FROM `student` WHERE `ID` IN (SELECT `ID` FROM `instructor`);
```

结果如下：

```
mysql> DELETE FROM `student` WHERE `ID` IN (SELECT `ID` FROM `instructor`);
Query OK, 50 rows affected (0.04 sec)
```

图 27: 删除数据

21. Some of you may have noticed that the tot_creds value for students did not match the credits from courses they have taken. Write and execute query to update tot_creds based on the credits passed, to bring the database back to consistency.

```
UPDATE `student` SET `tot_cred` = (SELECT SUM(`credits`) FROM `takes` NATURAL
    JOIN `course` WHERE `takes`.`ID` = `student`.`ID`);
```

结果如下：

```
mysql> UPDATE `student` SET `tot_cred` = (SELECT SUM(`credits`) FROM `takes` NATURAL JOIN `course` WHERE
`takes`.`ID` = `student`.`ID`);
Query OK, 1981 rows affected (0.12 sec)
Rows matched: 2000  Changed: 1981  Warnings: 0
```

图 28: 更新数据

22. Increase the salaries of instructors by 1000 times the number of course sections they have taught.

```
UPDATE `instructor` SET `salary` = `salary` + 1000 * (SELECT COUNT(`sec_id`)
    FROM `teaches` WHERE `teaches`.`ID` = `instructor`.`ID`);
```

结果如下：

```
mysql> UPDATE `instructor` SET `salary` = `salary` + 1000 * (SELECT COUNT(`sec_id`) FROM `teaches` WHERE
`teaches`.`ID` = `instructor`.`ID`);
Query OK, 31 rows affected (0.02 sec)
Rows matched: 50  Changed: 31  Warnings: 0
```

图 29: 更新数据

23. The university rules allow an F grade to be overridden by any pass grade (A, A-, B+, B, B-, C+, C, C-, D, D-). Now, create a view that lists information about all fail grades that have not been overridden (the view should contain all attributes from the takes relation).

```sql
CREATE VIEW `fail_grades` AS SELECT * FROM `takes` WHERE `grade` = 'F' AND NOT
    EXISTS (SELECT * FROM `takes` AS `override` WHERE `takes`.`ID` = `override
    `.`ID` AND `takes`.`course_id` = `override`.`course_id` AND `override`.`
    grade` IN ('A', 'A-', 'B+', 'B', 'B-', 'C+', 'C', 'C-', 'D', 'D-'));
```

结果如下：

```
mysql> CREATE VIEW `fail_grades` AS SELECT * FROM `takes` WHERE `grade` = 'F' AND NOT EXISTS (SELECT * FR
OM `takes` AS `override` WHERE `takes`.`ID` = `override`.`ID` AND `takes`.`course_id` = `override`.`cours
e_id` AND `override`.`grade` IN ('A', 'A-', 'B+', 'B', 'B-', 'C+', 'C', 'C-', 'D', 'D-'));
Query OK, 0 rows affected (0.01 sec)
```

图 30: 创建视图

24. Find all students who have 2 or more non-overridden F grades, and list them along with the F grades.

```sql
SELECT `ID`, `course_id`, `grade` FROM `fail_grades` WHERE `ID` IN (SELECT `ID`
    FROM `fail_grades` GROUP BY `ID` HAVING COUNT(`grade`) >= 2);
```

结果如下：

```
| 95850 | 559        | F     |
| 96085 | 192        | F     |
| 96085 | 319        | F     |
| 96085 | 445        | F     |
| 96085 | 457        | F     |
| 97658 | 629        | F     |
| 97658 | 694        | F     |
| 98019 | 169        | F     |
| 98019 | 974        | F     |
| 98315 | 169        | F     |
| 98315 | 192        | F     |
| 98359 | 192        | F     |
| 98359 | 401        | F     |
| 993   | 400        | F     |
| 993   | 457        | F     |
| 9933  | 443        | F     |
| 9933  | 457        | F     |
| 99660 | 629        | F     |
| 99660 | 802        | F     |
| 99710 | 169        | F     |
| 99710 | 192        | F     |
| 99949 | 319        | F     |
| 99949 | 559        | F     |
| 99949 | 808        | F     |
+-------+------------+-------+
379 rows in set (0.04 sec)
```

图 31: 查询结果

25. Grades are mapped to a grade point as follows:

Grade	A+	A	A-	B+	B	B-	C+	C	C-	D	D-	F
GP	4.0	4.0	3.7	3.3	3.0	2.7	2.3	2.0	1.7	1.3	1.0	0.0

表 1: Grade to GP conversion

Create a table to store these mappings, and write a query to find the GPA (Grade Point Average) of each student, using this table. Make sure students who have not got a non-null grade in any course are displayed with a GPA of null.

```sql
CREATE TABLE `grade_point` (
  `grade` CHAR(2) NOT NULL,
  `gp` DECIMAL(3, 1) NOT NULL,
  PRIMARY KEY (`grade`)
);

INSERT INTO `grade_point` (`grade`, `gp`) VALUES ('A+', 4.0), ('A', 4.0), ('A-', 3.7), ('B+', 3.3), ('B', 3.0), ('B-', 2.7), ('C+', 2.3), ('C', 2.0), ('C-', 1.5), ('D', 1.3), ('D-', 1.0), ('F', 0.0);

SELECT `ID`, SUM(`gp` * `credits`) / SUM(`credits`) AS `GPA` FROM `takes` NATURAL JOIN `grade_point` NATURAL JOIN `course` GROUP BY `ID`;
```

结果如下：

```
mysql> CREATE TABLE `grade_point` (
 NOT NULL    →   `grade` CHAR(2) NOT NULL,
    →   `gp` DECIMAL(3, 1) NOT NULL,
    →   PRIMARY KEY (`grade`)
;    → );
Query OK, 0 rows affected (0.06 sec)

mysql> INSERT INTO `grade_point` (`grade`, `gp`) VALUES ('A+', 4.0), ('A', 4.0), ('A-', 3.7), ('B+', 3.3)
, ('B', 3.0), ('B-', 2.7), ('C+', 2.3), ('C', 2.0), ('C-', 1.5), ('D', 1.3), ('D-', 1.0), ('F', 0.0);
Query OK, 12 rows affected (0.03 sec)
Records: 12  Duplicates: 0  Warnings: 0

mysql> SELECT `ID`, SUM(`gp` * `credits`) / SUM(`credits`) AS `GPA` FROM `takes` NATURAL JOIN `grade_poin
t` NATURAL JOIN `course` GROUP BY `ID`;
+-------+----------+
| ID    | GPA      |
+-------+----------+
| 1000  | 2.84286 |
| 10033 | 2.49250 |
| 10076 | 3.06486 |
| 1018  | 2.34444 |
| 10204 | 3.13043 |
| 10267 | 2.54828 |
| 10269 | 2.37241 |
| 10454 | 2.62051 |
| 10481 | 2.25938 |
| 10527 | 2.79091 |
| 10556 | 2.96400 |
```

图 32: 查询结果

26. Find all classrooms that have been assigned to more than one section at the same time. Display the rooms along with the assigned sections; Using a with clause or a view to simplify this query.

```
WITH `conflict` AS (
  SELECT `room_number`, `time_slot_id` FROM `section` GROUP BY `room_number`, `time_slot_id` HAVING COUNT(`sec_id`) > 1
) SELECT `room_number`, `sec_id` FROM `section` WHERE `room_number` IN (SELECT `room_number` FROM `conflict`) AND `time_slot_id` IN (SELECT `time_slot_id` FROM `conflict`);
```

结果如下：

```
mysql> WITH `conflict` AS (
    -> SELECT `room_number`, `time_slot_id` FROM `section` GROUP BY `room_number`, `time_slot_id` HAVIN
G COUNT(`sec_id`) > 1
LECT    -> ) SELECT `room_number`, `sec_id` FROM `section` WHERE `room_number` IN (SELECT `room_number` F
ROM `conflict`) AND `time_slot_id` IN (SELECT `time_slot_id` FROM `conflict`);
+-------------+--------+
| room_number | sec_id |
+-------------+--------+
| 183         | 2      |
| 434         | 1      |
| 180         | 1      |
| 717         | 1      |
| 145         | 2      |
| 183         | 1      |
| 145         | 1      |
| 717         | 1      |
| 183         | 1      |
| 183         | 1      |
| 143         | 1      |
| 348         | 1      |
| 183         | 1      |
| 348         | 1      |
```

图 33: 查询结果

27. Create a view faculty showing only the ID, name, and department of instructors.

```
CREATE VIEW `faculty` AS SELECT `ID`, `name`, `dept_name` FROM `instructor`;
```

结果如下：

```
mysql> CREATE VIEW `faculty` AS SELECT `ID`, `name`, `dept_name` FROM `instructor`;
Query OK, 0 rows affected (0.02 sec)
```

图 34: 创建视图

28. Create a view CSinstructors, showing all information about instructors from the Comp. Sci. department.

```sql
CREATE VIEW `CSinstructors` AS SELECT * FROM `instructor` WHERE `dept_name` = 'Comp. Sci.';
```

结果如下：

```
mysql> CREATE VIEW `CSinstructors` AS SELECT * FROM `instructor` WHERE `dept_name` = 'Comp. Sci.';
Query OK, 0 rows affected (0.01 sec)
```

图 35: 创建视图

29. Insert appropriate tuple into each of the views faculty and CSinstructors, to see what updates your database allows on views; explain what happens.

```sql
INSERT INTO `faculty` (`ID`, `name`, `dept_name`) VALUES (10101, 'John Doe', 'Comp. Sci.');
INSERT INTO `CSinstructors` (`ID`, `name`, `dept_name`, `salary`) VALUES (10102, 'John Smith', 'Comp. Sci.', 100000);
```

结果如下：

```
mysql> INSERT INTO `faculty` (`ID`, `name`, `dept_name`) VALUES (10101, 'John Doe', 'Comp. Sci.');
NSERT INTO `CSinstructors` (`ID`, `name`, `dept_name`, `salary`) VALUES (10102, 'John Smith', 'Comp. Sci.
', 100000);
SELECT * FROM `instructor` LIMIT 5;Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO `CSinstructors` (`ID`, `name`, `dept_name`, `salary`) VALUES (10102, 'John Smith', 'Co
mp. Sci.', 100000);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM `instructor` LIMIT 5;
+-------+-------------+-------------+------------+
| ID    | name        | dept_name   | salary     |
+-------+-------------+-------------+------------+
| 10101 | John Doe    | Comp. Sci.  |       NULL |
| 10102 | John Smith  | Comp. Sci.  | 100000.00  |
| 14365 | Lembr       | Accounting  |  34241.56  |
| 15347 | Bawa        | Athletics   |  73140.88  |
| 16807 | Yazdi       | Athletics   |  98333.65  |
+-------+-------------+-------------+------------+
5 rows in set (0.02 sec)
```

图 36: 插入数据

可以发现原表中的数据也被更新了，说明视图是基于原表的，视图的更新会影响原表。

30. Create a new user and grant permission to the user to view all data in your student relation.

```sql
create user newuser identified by 'Password@123';
grant select on student to newuser;
-- 切换至newuser用户登录，验证对数据表的权限；
exit;
```

```sql
mysql -unewuser -p -Ddbcourse
select * from advisor;
-- ERROR 1142 (42000): SELECT command denied to user 'newuser'@'localhost' for table 'advisor'
update student set tot_cred = tot_cred + 1;
-- ERROR 1142 (42000): UPDATE command denied to user 'newuser'@'localhost' for table 'student'
select * from student;
-- 切换至root用户登录
exit;
mysql -uroot -p -Ddbcourse
```

结果如下:

```
mysql> create user newuser identified by 'Password@123';
Query OK, 0 rows affected (0.10 sec)

mysql> grant select on student to newuser;
Query OK, 0 rows affected (0.01 sec)

mysql> exit;
Bye
bash-4.4# mysql -unewuser -p -Ddbcourse
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 20
Server version: 8.2.0 MySQL Community Server - GPL

Copyright (c) 2000, 2023, 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>
```

图 37: 创建用户并授权

```
mysql> select * from advisor;
ERROR 1142 (42000): SELECT command denied to user 'newuser'@'localhost' for table 'advisor'
mysql> update student set tot_cred = tot_cred + 1;
ERROR 1142 (42000): UPDATE command denied to user 'newuser'@'localhost' for table 'student'
mysql> select * from student;
+-------+--------------------+-------------+----------+
| ID    | name               | dept_name   | tot_cred |
+-------+--------------------+-------------+----------+
| 1000  | Manber             | Civil Eng.  |       47 |
| 10033 | Zelty              | Mech. Eng.  |       71 |
| 10076 | Duan               | Civil Eng.  |       46 |
| 1018  | Colin              | Civil Eng.  |       82 |
| 10204 | Mediratta          | Geology     |       40 |
| 10267 | Rzecz              | Comp. Sci.  |       38 |
| 10269 | Hilberg            | Psychology  |       36 |
| 10454 | Ugarte             | Pol. Sci.   |       61 |
| 10481 | Grosch             | Astronomy   |       62 |
| 10527 | Kieras             | Physics     |       73 |
| 10556 | Reed               | English     |       38 |
| 10663 | Okaf               | Geology     |       63 |
| 10693 | Zabary             | Statistics  |       46 |
| 107   | Shabuno            | Math        |       86 |
| 10705 | Terauchi           | Physics     |       59 |
| 10727 | Allard             | Physics     |       55 |
| 10736 | Veselovsky         | Elec. Eng.  |       41 |
| 108   | Dhav               | Biology     |       56 |
```

图 38: 验证用户权限

```
| 99775 | Epley              | Athletics   |       29 |
| 99780 | Bravo              | English     |       53 |
| 9993  | Won                | Math        |       69 |
| 99949 | Samo               | Astronomy   |       44 |
| 99977 | Englund            | Psychology  |       61 |
+-------+--------------------+-------------+----------+
2000 rows in set (0.02 sec)

mysql> exit;
Bye
bash-4.4# mysql -uroot -p -Ddbcourse
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 21
Server version: 8.2.0 MySQL Community Server - GPL

Copyright (c) 2000, 2023, 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>
```

图 39: 切换用户

31. Now grant permission to all users to see all data in your faculty view.

```
create role all_users;
grant all_users to root, newuser;
grant select on faculty to all_users;
```

```sql
-- 切换至newuser用户登录，验证权限
exit;
mysql -unewuser -p -Ddbcourse
set role all_users;
select * from instructor;
select * from faculty;
```

结果如下：

```
mysql> create role all_users;
Query OK, 0 rows affected (0.00 sec)

mysql> grant all_users to root, newuser;
Query OK, 0 rows affected (0.02 sec)

mysql> grant select on faculty to all_users;
Query OK, 0 rows affected (0.01 sec)

mysql> exit;
Bye
bash-4.4# mysql -unewuser -p -Ddbcourse
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 8.2.0 MySQL Community Server - GPL

Copyright (c) 2000, 2023, 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.
```

图 40: 授权用户查看视图

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 8.2.0 MySQL Community Server - GPL

Copyright (c) 2000, 2023, 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> set role all_users;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from instructor;
ERROR 1142 (42000): SELECT command denied to user 'newuser'@'localhost' for table 'instructor'
mysql> select * from faculty;
+-------+--------------------+-------------+
| ID    | name               | dept_name   |
+-------+--------------------+-------------+
| 10101 | John Doe           | Comp. Sci.  |
| 10102 | John Smith         | Comp. Sci.  |
| 14365 | Lembr              | Accounting  |
| 15347 | Bawa               | Athletics   |
```

图 41: 验证用户权限

3 存在的问题及解决方案

1. 对 COALESCE 函数的使用不熟悉，导致在查询中出现错误，通过查阅文档解决了问题。
2. 更新数据时出现错误，导致数据库内数据被破坏，通过备份数据恢复了数据库。

4 实验小结

通过本次实验，我学习了 MySQL 数据库管理系统中 SQL 的基本语法，掌握了 SQL 语句的编写方法，能够编写 SQL 语句完成指定的查询。同时，我还学习了如何创建数据表、插入数据、创建视图、创建用户、授权用户等操作，对数据库的管理有了更深入的了解。