

# 《数据库系统及应用实践》课程实验报告

## 实验 1: 在 Docker 环境下使用 MySQL

姓 名: 李鹏达 学 号: 10225101460 完成日期: 2024 年 3 月 7 日

## 1 实验目标

1. 学习 Docker 环境的原理和基本操作
2. 能够通过 Docker 容器启动 MySQL 数据库实例
3. 掌握连接和操作 MySQL 数据库的基本命令

## 2 实验过程记录

### 2.1 安装 Docker

我的实验环境是 Ubuntu 22.04 LTS on Windows 10 x86\_64, 在这里, 我选择按照官网文档 (<https://docs.docker.com/engine/install/ubuntu/>) 安装 Docker。

首先, 添加 Docker 的 apt 源:

```
1 # Add Docker's official GPG key:
2 sudo apt-get update
3 sudo apt-get install ca-certificates curl
4 sudo install -m 0755 -d /etc/apt/keyrings
5 sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/
   docker.asc
6 sudo chmod a+r /etc/apt/keyrings/docker.asc
7
8 # Add the repository to Apt sources:
9 echo \
10     "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
   https://download.docker.com/linux/ubuntu \
11     $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
12     sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
13 sudo apt-get update
```

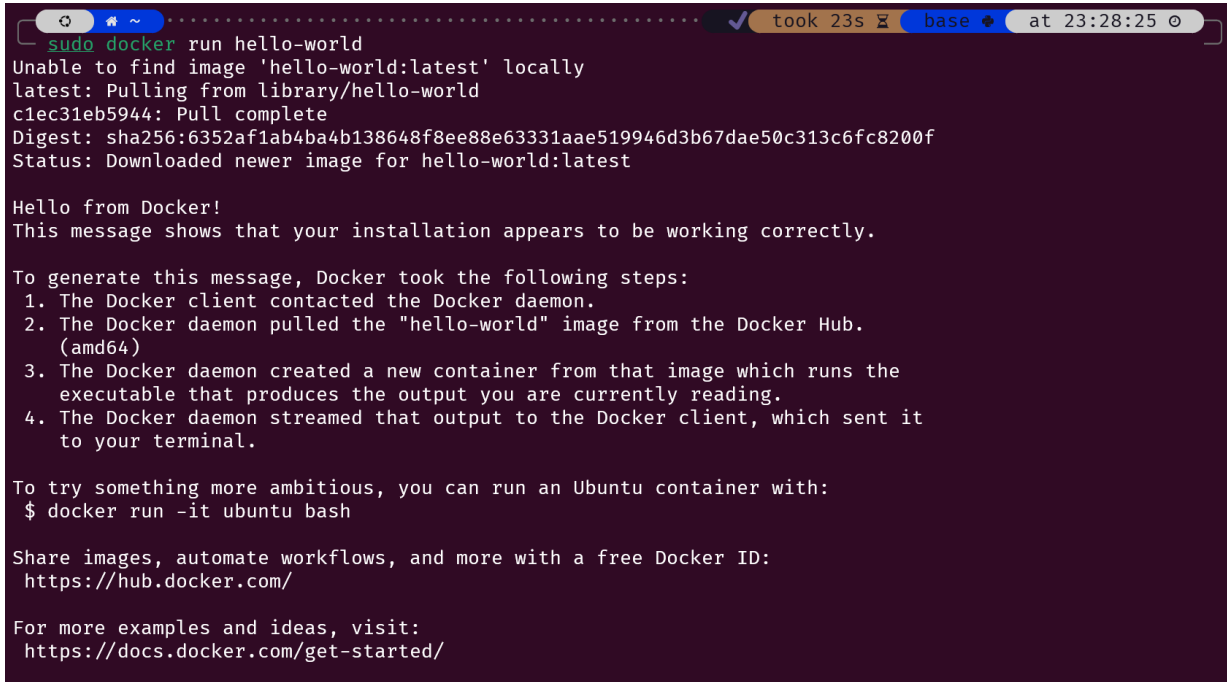
然后使用 apt 安装 Docker:

```
1 sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
   docker-compose-plugin
```

待安装完成后，运行命令以检查 Docker 是否安装成功：

```
1 sudo docker run hello-world
```

输出结果如下图所示，说明 Docker 安装成功。

A terminal window with a dark background. The command 'sudo docker run hello-world' has been executed. The output shows that Docker pulled the 'hello-world:latest' image from the Docker Hub, created a new container, and ran the executable inside it. The output message says 'Hello from Docker!' and 'This message shows that your installation appears to be working correctly.' It also lists the steps Docker took to generate this message: 1. The Docker client contacted the Docker daemon. 2. The Docker daemon pulled the "hello-world" image from the Docker Hub. (amd64) 3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading. 4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal. At the bottom, it suggests trying something more ambitious by running an Ubuntu container with the command '\$ docker run -it ubuntu bash' and provides links to Docker Hub and Docker documentation.

```
✓ took 23s base at 23:28:25
$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:6352af1ab4ba4b138648f8ee88e63331aae519946d3b67dae50c313c6fc8200f
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

图 1: 检查 Docker 是否安装成功

## 2.2 在容器中启动 MySQL 示例

首先，创建一个 dbcourse 文件夹，并在其中创建一个 datadir 文件夹：

```
1 mkdir dbcourse
2 cd dbcourse
3 mkdir datadir
```

然后拉取 MySQL 镜像：

```
1 sudo docker pull mysql:8.2.0
```

结果如下图所示：

```
base at 00:06:51
sudo docker pull mysql:8.2.0
8.2.0: Pulling from library/mysql
558b7d69a2e5: Pull complete
599b67b0dd6a: Pull complete
50314d46ce2b: Pull complete
494babc92263: Pull complete
02548e6f2dbf: Pull complete
a9e5e2637e0d: Pull complete
657b198fe6b7: Pull complete
215a2b0eabff: Pull complete
377a4c7a89c5: Pull complete
4bfe599fe218: Pull complete
Digest: sha256:212fe73edca5df6ff14826d5eb975c914bfb91f82a2e923f9050568f99525da1
Status: Downloaded newer image for mysql:8.2.0
docker.io/library/mysql:8.2.0
```

图 2: 拉取 MySQL 镜像

接下来, 使用如下命令启动一个 MySQL 容器:

```
1 sudo docker run --name dbcourse -v ./datadir:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=
  password -d -p 53306:3306 mysql:8.2.0
```

这个命令将会创建一个名为 `dbcourse` 的容器, 并将容器内的 `/var/lib/mysql` 文件夹映射到宿主机的 `./datadir` 文件夹, 设置 root 用户的密码为 `password`, 将容器的 3306 端口映射到宿主机的 53306 端口。

结果如下图所示:

```
base at 00:15:51
sudo docker run --name dbcourse -v ./datadir:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=password -d -p 5330
6:3306 mysql:8.2.0
1da884ed16dcf7066eb9c7a93f7829ea4360f6fd9bce6ddb99275ecb70458cb8
```

图 3: 创建容器

使用 `docker ps` 命令可以看到这个容器正在运行:

```
base at 00:18:50
sudo docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
1da884ed16dc   mysql:8.2.0 "docker-entrypoint.s..." 2 minutes ago  Up 2 minutes  33060/tcp,
0.0.0.0:53306->3306/tcp, :::53306->3306/tcp
dbcourse
```

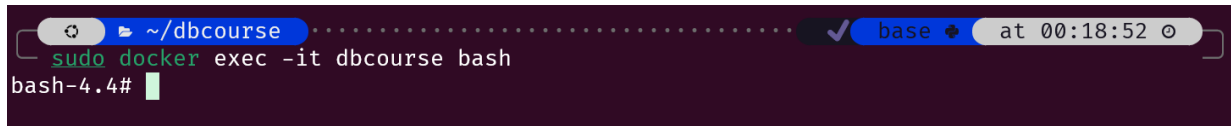
图 4: 查看运行中的容器

## 2.3 对 MySQL 数据库进行操作

执行如下命令, 启动容器内的一个 `bash` 终端:

```
1 sudo docker exec -it dbcourse bash
```

结果如下：

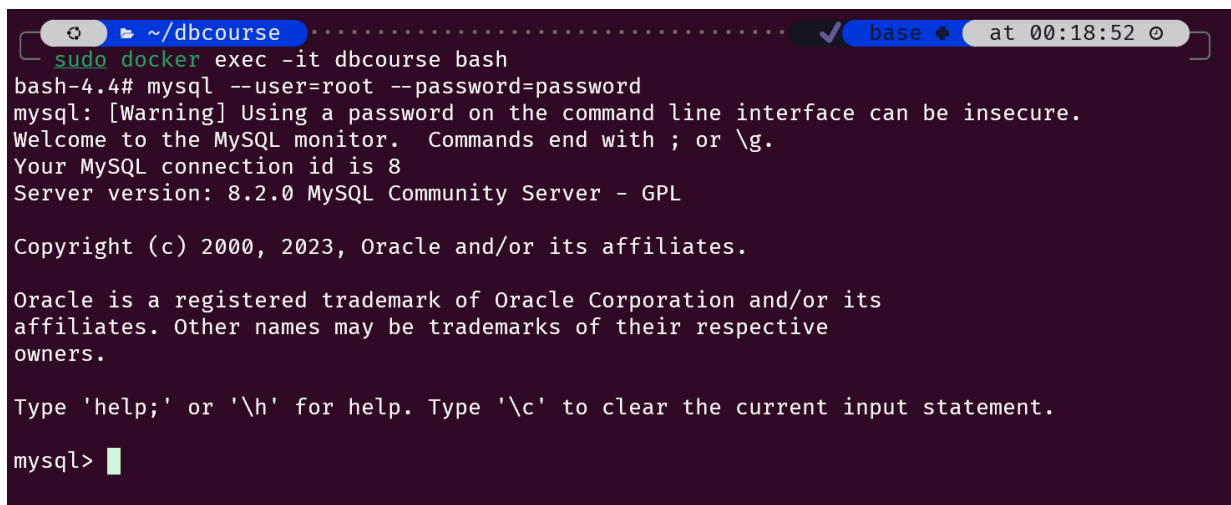


```
~/dbcourse ... base at 00:18:52  
sudo docker exec -it dbcourse bash  
bash-4.4#
```

图 5: 启动容器内的 bash 终端

使用命令登录 MySQL 数据库：

```
1 mysql --user=root --password=password
```



```
~/dbcourse ... base at 00:18:52  
sudo docker exec -it dbcourse bash  
bash-4.4# mysql --user=root --password=password  
mysql: [Warning] Using a password on the command line interface can be insecure.  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 8  
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 6: 登录 MySQL

执行 help 命令，可以看到 MySQL 的帮助信息：

```
1 help;
```

```
edit      (\e) Edit command with $EDITOR.
ego       (\G) Send command to mysql server, display result vertically.
exit      (\q) Exit mysql. Same as quit.
go        (\g) Send command to mysql server.
help      (\h) Display this help.
nopager   (\n) Disable pager, print to stdout.
notee     (\t) Don't write into outfile.
pager     (\P) Set PAGER [to_pager]. Print the query results via PAGER.
print     (\p) Print current command.
prompt    (\R) Change your mysql prompt.
quit      (\q) Quit mysql.
rehash    (\#) Rebuild completion hash.
source    (\.) Execute an SQL script file. Takes a file name as an argument.
status    (\s) Get status information from the server.
system    (\!) Execute a system shell command.
tee        (\T) Set outfile [to_outfile]. Append everything into given outfile.
use       (\u) Use another database. Takes database name as argument.
charset   (\C) Switch to another charset. Might be needed for processing binlog with multi-byte charsets.
warnings  (\W) Show warnings after every statement.
nowarning (\w) Don't show warnings after every statement.
resetconnection(\x) Clean session context.
query_attributes Sets string parameters (name1 value1 name2 value2 ...) for the next query to pick up.
ssl_session_data_print Serializes the current SSL session data to stdout or file

For server side help, type 'help contents'

mysql> █
```

图 7: help 命令

创建一个名为 dbcourse 的数据库对象

```
1 create database dbcourse;
```

```
mysql> create database dbcourse;
Query OK, 1 row affected (0.01 sec)
```

图 8: 创建数据库对象

查看系统中所有的数据库对象

```
1 show databases;
```

```
mysql> show databases;
+-----+
| Database |
+-----+
| dbcourse |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.00 sec)
```

图 9: 查看所有的数据库对象

将当前使用的数据库设置为 dbcourse

```
1 use dbcourse;
```

```
mysql> use dbcourse;
Database changed
```

图 10: 设置当前使用的数据库

接下来，执行下列 SQL 语句，在 dbcourse 数据库中创建数据表

```
1 create table classroom(
2     building varchar(15),
3     room_number varchar(7),
4     capacity numeric(4,0),
5     primary key (building, room_number)
6 );
7 create table department(
8     dept_name varchar(20),
9     building varchar(15),
10    budget numeric(12,2) check (budget > 0),
11    primary key (dept_name)
12 );
13 create table course(
14     course_id varchar(8),
```

```
15     title varchar(50),
16     dept_name varchar(20),
17     credits numeric(2,0) check (credits > 0),
18     primary key (course_id),
19     foreign key (dept_name) references department (dept_name)
20     on delete set null
21 );
22 create table instructor(
23     ID varchar(5),
24     name varchar(20) not null,
25     dept_name varchar(20),
26     salary numeric(8,2) check (salary > 29000),
27     primary key (ID),
28     foreign key (dept_name) references department (dept_name)
29     on delete set null
30 );
31 create table section(
32     course_id varchar(8),
33     sec_id varchar(8),
34     semester varchar(6) check (semester in
35     ('Fall', 'Winter', 'Spring', 'Summer')),
36     year numeric(4,0) check (year > 1701 and year < 2100),
37     building varchar(15),
38     room_number varchar(7),
39     time_slot_id varchar(4),
40     primary key (course_id, sec_id, semester, year),
41     foreign key (course_id) references course (course_id) on delete cascade,
42     foreign key (building, room_number)
43     references classroom (building, room_number) on delete set null
44 );
45 create table teaches(
46     ID varchar(5),
47     course_id varchar(8),
48     sec_id varchar(8),
49     semester varchar(6),
50     year numeric(4,0),
51     primary key (ID, course_id, sec_id, semester, year),
52     foreign key (course_id, sec_id, semester, year)
53     references section (course_id, sec_id, semester, year)
54     on delete cascade,
55     foreign key (ID) references instructor (ID) on delete cascade
56 );
57 create table student(
58     ID varchar(5),
```

```
59     name varchar(20) not null,
60     dept_name varchar(20),
61     tot_cred numeric(3,0) check (tot_cred >= 0),
62     primary key (ID),
63     foreign key (dept_name) references department (dept_name)
64     on delete set null
65 );
66 create table takes(
67     ID varchar(5),
68     course_id varchar(8),
69     sec_id varchar(8),
70     semester varchar(6),
71     year numeric(4,0),
72     grade varchar(2),
73     primary key (ID, course_id, sec_id, semester, year),
74     foreign key (course_id, sec_id, semester, year)
75     references section (course_id, sec_id, semester, year)
76     on delete cascade,
77     foreign key (ID) references student (ID) on delete cascade
78 );
79 create table advisor(
80     s_ID varchar(5),
81     i_ID varchar(5),
82     primary key (s_ID),
83     foreign key (i_ID) references instructor (ID) on delete set null,
84     foreign key (s_ID) references student (ID) on delete cascade
85 );
86 create table time_slot(
87     time_slot_id varchar(4),
88     day varchar(1),
89     start_hr numeric(2) check (start_hr >= 0 and start_hr < 24),
90     start_min numeric(2) check (start_min >= 0 and start_min < 60),
91     end_hr numeric(2) check (end_hr >= 0 and end_hr < 24),
92     end_min numeric(2) check (end_min >= 0 and end_min < 60),
93     primary key (time_slot_id, day, start_hr, start_min)
94 );
95 create table prereq(
96     course_id varchar(8),
97     prereq_id varchar(8),
98     primary key (course_id, prereq_id),
99     foreign key (course_id) references course (course_id) on delete cascade,
100    foreign key (prereq_id) references course (course_id)
101 );
```



然后查看 dbcourse 数据库中所有的数据表

```
1 show tables;
```

```
mysql> show tables;
+-----+
| Tables_in_dbcourse |
+-----+
| advisor             |
| classroom            |
| course              |
| department           |
| instructor           |
| prereq              |
| section             |
| student             |
| takes               |
| teaches             |
| time_slot           |
+-----+
11 rows in set (0.00 sec)
```

图 11: 查看所有的表

接下来, 执行下列 SQL 语句, 向数据表中插入数据:

```
1 insert into classroom values ('Packard', '101', '500');
2 insert into classroom values ('Painter', '514', '10');
3 insert into classroom values ('Taylor', '3128', '70');
4 insert into classroom values ('Watson', '100', '30');
5 insert into classroom values ('Watson', '120', '50');
6 insert into department values ('Biology', 'Watson', '90000');
7 insert into department values ('Comp. Sci.', 'Taylor', '100000');
8 insert into department values ('Elec. Eng.', 'Taylor', '85000');
9 insert into department values ('Finance', 'Painter', '120000');
10 insert into department values ('History', 'Painter', '50000');
11 insert into department values ('Music', 'Packard', '80000');
12 insert into department values ('Physics', 'Watson', '70000');
13 insert into course values ('BIO-101', 'Intro. to Biology', 'Biology', '4');
14 insert into course values ('BIO-301', 'Genetics', 'Biology', '4');
15 insert into course values ('BIO-399', 'Computational Biology', 'Biology', '3');
16 insert into course values ('CS-101', 'Intro. to Computer Science', 'Comp. Sci.', '4'
    );
17 insert into course values ('CS-190', 'Game Design', 'Comp. Sci.', '4');
18 insert into course values ('CS-315', 'Robotics', 'Comp. Sci.', '3');
```

```
19 insert into course values ('CS-319', 'Image Processing', 'Comp. Sci.', '3');
20 insert into course values ('CS-347', 'Database System Concepts', 'Comp. Sci.', '3');
21 insert into course values ('EE-181', 'Intro. to Digital Systems', 'Elec. Eng.', '3')
    ;
22 insert into course values ('FIN-201', 'Investment Banking', 'Finance', '3');
23 insert into course values ('HIS-351', 'World History', 'History', '3');
24 insert into course values ('MU-199', 'Music Video Production', 'Music', '3');
25 insert into course values ('PHY-101', 'Physical Principles', 'Physics', '4');
26 insert into instructor values ('10101', 'Srinivasan', 'Comp. Sci.', '65000');
27 insert into instructor values ('12121', 'Wu', 'Finance', '90000');
28 insert into instructor values ('15151', 'Mozart', 'Music', '40000');
29 insert into instructor values ('22222', 'Einstein', 'Physics', '95000');
30 insert into instructor values ('32343', 'El Said', 'History', '60000');
31 insert into instructor values ('33456', 'Gold', 'Physics', '87000');
32 insert into instructor values ('45565', 'Katz', 'Comp. Sci.', '75000');
33 insert into instructor values ('58583', 'Califieri', 'History', '62000');
34 insert into instructor values ('76543', 'Singh', 'Finance', '80000');
35 insert into instructor values ('76766', 'Crick', 'Biology', '72000');
36 insert into instructor values ('83821', 'Brandt', 'Comp. Sci.', '92000');
37 insert into instructor values ('98345', 'Kim', 'Elec. Eng.', '80000');
38 insert into section values ('BIO-101', '1', 'Summer', '2017', 'Painter', '514', 'B')
    ;
39 insert into section values ('BIO-301', '1', 'Summer', '2018', 'Painter', '514', 'A')
    ;
40 insert into section values ('CS-101', '1', 'Fall', '2017', 'Packard', '101', 'H');
41 insert into section values ('CS-101', '1', 'Spring', '2018', 'Packard', '101', 'F');
42 insert into section values ('CS-190', '1', 'Spring', '2017', 'Taylor', '3128', 'E');
43 insert into section values ('CS-190', '2', 'Spring', '2017', 'Taylor', '3128', 'A');
44 insert into section values ('CS-315', '1', 'Spring', '2018', 'Watson', '120', 'D');
45 insert into section values ('CS-319', '1', 'Spring', '2018', 'Watson', '100', 'B');
46 insert into section values ('CS-319', '2', 'Spring', '2018', 'Taylor', '3128', 'C');
47 insert into section values ('CS-347', '1', 'Fall', '2017', 'Taylor', '3128', 'A');
48 insert into section values ('EE-181', '1', 'Spring', '2017', 'Taylor', '3128', 'C');
49 insert into section values ('FIN-201', '1', 'Spring', '2018', 'Packard', '101', 'B')
    ;
50 insert into section values ('HIS-351', '1', 'Spring', '2018', 'Painter', '514', 'C')
    ;
51 insert into section values ('MU-199', '1', 'Spring', '2018', 'Packard', '101', 'D');
52 insert into section values ('PHY-101', '1', 'Fall', '2017', 'Watson', '100', 'A');
53 insert into teaches values ('10101', 'CS-101', '1', 'Fall', '2017');
54 insert into teaches values ('10101', 'CS-315', '1', 'Spring', '2018');
55 insert into teaches values ('10101', 'CS-347', '1', 'Fall', '2017');
56 insert into teaches values ('12121', 'FIN-201', '1', 'Spring', '2018');
57 insert into teaches values ('15151', 'MU-199', '1', 'Spring', '2018');
```

```
58 insert into teaches values ('22222', 'PHY-101', '1', 'Fall', '2017');
59 insert into teaches values ('32343', 'HIS-351', '1', 'Spring', '2018');
60 insert into teaches values ('45565', 'CS-101', '1', 'Spring', '2018');
61 insert into teaches values ('45565', 'CS-319', '1', 'Spring', '2018');
62 insert into teaches values ('76766', 'BIO-101', '1', 'Summer', '2017');
63 insert into teaches values ('76766', 'BIO-301', '1', 'Summer', '2018');
64 insert into teaches values ('83821', 'CS-190', '1', 'Spring', '2017');
65 insert into teaches values ('83821', 'CS-190', '2', 'Spring', '2017');
66 insert into teaches values ('83821', 'CS-319', '2', 'Spring', '2018');
67 insert into teaches values ('98345', 'EE-181', '1', 'Spring', '2017');
68 insert into student values ('00128', 'Zhang', 'Comp. Sci.', '102');
69 insert into student values ('12345', 'Shankar', 'Comp. Sci.', '32');
70 insert into student values ('19991', 'Brandt', 'History', '80');
71 insert into student values ('23121', 'Chavez', 'Finance', '110');
72 insert into student values ('44553', 'Peltier', 'Physics', '56');
73 insert into student values ('45678', 'Levy', 'Physics', '46');
74 insert into student values ('54321', 'Williams', 'Comp. Sci.', '54');
75 insert into student values ('55739', 'Sanchez', 'Music', '38');
76 insert into student values ('70557', 'Snow', 'Physics', '0');
77 insert into student values ('76543', 'Brown', 'Comp. Sci.', '58');
78 insert into student values ('76653', 'Aoi', 'Elec. Eng.', '60');
79 insert into student values ('98765', 'Bourikas', 'Elec. Eng.', '98');
80 insert into student values ('98988', 'Tanaka', 'Biology', '120');
81 insert into takes values ('00128', 'CS-101', '1', 'Fall', '2017', 'A');
82 insert into takes values ('00128', 'CS-347', '1', 'Fall', '2017', 'A-');
83 insert into takes values ('12345', 'CS-101', '1', 'Fall', '2017', 'C');
84 insert into takes values ('12345', 'CS-190', '2', 'Spring', '2017', 'A');
85 insert into takes values ('12345', 'CS-315', '1', 'Spring', '2018', 'A');
86 insert into takes values ('12345', 'CS-347', '1', 'Fall', '2017', 'A');
87 insert into takes values ('19991', 'HIS-351', '1', 'Spring', '2018', 'B');
88 insert into takes values ('23121', 'FIN-201', '1', 'Spring', '2018', 'C+');
89 insert into takes values ('44553', 'PHY-101', '1', 'Fall', '2017', 'B-');
90 insert into takes values ('45678', 'CS-101', '1', 'Fall', '2017', 'F');
91 insert into takes values ('45678', 'CS-101', '1', 'Spring', '2018', 'B+');
92 insert into takes values ('45678', 'CS-319', '1', 'Spring', '2018', 'B');
93 insert into takes values ('54321', 'CS-101', '1', 'Fall', '2017', 'A-');
94 insert into takes values ('54321', 'CS-190', '2', 'Spring', '2017', 'B+');
95 insert into takes values ('55739', 'MU-199', '1', 'Spring', '2018', 'A-');
96 insert into takes values ('76543', 'CS-101', '1', 'Fall', '2017', 'A');
97 insert into takes values ('76543', 'CS-319', '2', 'Spring', '2018', 'A');
98 insert into takes values ('76653', 'EE-181', '1', 'Spring', '2017', 'C');
99 insert into takes values ('98765', 'CS-101', '1', 'Fall', '2017', 'C-');
100 insert into takes values ('98765', 'CS-315', '1', 'Spring', '2018', 'B');
101 insert into takes values ('98988', 'BIO-101', '1', 'Summer', '2017', 'A');
```

```
102 insert into takes values ('98988', 'BIO-301', '1', 'Summer', '2018', null);
103 insert into advisor values ('00128', '45565');
104 insert into advisor values ('12345', '10101');
105 insert into advisor values ('23121', '76543');
106 insert into advisor values ('44553', '22222');
107 insert into advisor values ('45678', '22222');
108 insert into advisor values ('76543', '45565');
109 insert into advisor values ('76653', '98345');
110 insert into advisor values ('98765', '98345');
111 insert into advisor values ('98988', '76766');
112 insert into time_slot values ('A', 'M', '8', '0', '8', '50');
113 insert into time_slot values ('A', 'W', '8', '0', '8', '50');
114 insert into time_slot values ('A', 'F', '8', '0', '8', '50');
115 insert into time_slot values ('B', 'M', '9', '0', '9', '50');
116 insert into time_slot values ('B', 'W', '9', '0', '9', '50');
117 insert into time_slot values ('B', 'F', '9', '0', '9', '50');
118 insert into time_slot values ('C', 'M', '11', '0', '11', '50');
119 insert into time_slot values ('C', 'W', '11', '0', '11', '50');
120 insert into time_slot values ('C', 'F', '11', '0', '11', '50');
121 insert into time_slot values ('D', 'M', '13', '0', '13', '50');
122 insert into time_slot values ('D', 'W', '13', '0', '13', '50');
123 insert into time_slot values ('D', 'F', '13', '0', '13', '50');
124 insert into time_slot values ('E', 'T', '10', '30', '11', '45 ');
125 insert into time_slot values ('E', 'R', '10', '30', '11', '45 ');
126 insert into time_slot values ('F', 'T', '14', '30', '15', '45 ');
127 insert into time_slot values ('F', 'R', '14', '30', '15', '45 ');
128 insert into time_slot values ('G', 'M', '16', '0', '16', '50');
129 insert into time_slot values ('G', 'W', '16', '0', '16', '50');
130 insert into time_slot values ('G', 'F', '16', '0', '16', '50');
131 insert into time_slot values ('H', 'W', '10', '0', '12', '30');
132 insert into prereq values ('BIO-301', 'BIO-101');
133 insert into prereq values ('BIO-399', 'BIO-101');
134 insert into prereq values ('CS-190', 'CS-101');
135 insert into prereq values ('CS-315', 'CS-101');
136 insert into prereq values ('CS-319', 'CS-101');
137 insert into prereq values ('CS-347', 'CS-101');
138 insert into prereq values ('EE-181', 'PHY-101');
```

执行下列 SQL 语句，查询每张数据表中的数据：

```
1 select * from advisor;
2 select * from classroom;
3 select * from course;
4 select * from department;
```

```

5 select * from instructor;
6 select * from prereq;
7 select * from section;
8 select * from student;
9 select * from takes;
10 select * from teaches;
11 select * from time_slot;

```

```

mysql> select * from time_slot;
+-----+-----+-----+-----+-----+-----+
| time_slot_id | day | start_hr | start_min | end_hr | end_min |
+-----+-----+-----+-----+-----+-----+
| A | F | 8 | 0 | 8 | 50 |
| A | M | 8 | 0 | 8 | 50 |
| A | W | 8 | 0 | 8 | 50 |
| B | F | 9 | 0 | 9 | 50 |
| B | M | 9 | 0 | 9 | 50 |
| B | W | 9 | 0 | 9 | 50 |
| C | F | 11 | 0 | 11 | 50 |
| C | M | 11 | 0 | 11 | 50 |
| C | W | 11 | 0 | 11 | 50 |
| D | F | 13 | 0 | 13 | 50 |
| D | M | 13 | 0 | 13 | 50 |
| D | W | 13 | 0 | 13 | 50 |
| E | R | 10 | 30 | 11 | 45 |
| E | T | 10 | 30 | 11 | 45 |
| F | R | 14 | 30 | 15 | 45 |
| F | T | 14 | 30 | 15 | 45 |
| G | F | 16 | 0 | 16 | 50 |
| G | M | 16 | 0 | 16 | 50 |
| G | W | 16 | 0 | 16 | 50 |
| H | W | 10 | 0 | 12 | 30 |
+-----+-----+-----+-----+-----+-----+
20 rows in set (0.00 sec)

```

图 12: 查询表中的数据

使用 `describe` 命令查看数据表的结构:

```

1 describe advisor;

```

```

mysql> describe advisor;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| s_ID  | varchar(5)    | NO   | PRI | NULL    |       |
| i_ID  | varchar(5)    | YES  | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.05 sec)

```

图 13: 查看表的结构

使用 `show index from` 语句查看数据表的索引信息

```
1 show index from advisor;
```

```
mysql> show index from advisor;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
advisor	0	PRIMARY	1	s_ID	A	9	NULL	NULL	YES	BTREE			YES	NULL
advisor	1	i_ID	1	i_ID	A	6	NULL	NULL	YES	BTREE			YES	NULL

2 rows in set (0.02 sec)

图 14: 查看表的索引

将当前使用的数据库设置为 `mysql`

```
1 use mysql;
```

查看 `mysql` 数据库中的所有数据表对象

```
1 show tables;
```

```
mysql> show tables;
```

Tables_in_mysql
columns_priv
component
db
default_roles
engine_cost
func
general_log
global_grants
gtid_executed
help_category
help_keyword
help_relation
help_topic
innodb_index_stats
innodb_table_stats
ndb_binlog_index
password_history
plugin
procs_priv
proxies_priv
replication_asynchronous_connection_failover
replication_asynchronous_connection_failover_managed
replication_group_configuration_version
replication_group_member_actions
role_edges
server_cost
servers
slave_master_info
slave_relay_log_info
slave_worker_info
slow_log
tables_priv
time_zone
time_zone_leap_second
time_zone_name
time_zone_transition
time_zone_transition_type
user

38 rows in set (0.00 sec)

图 15: 查看 `mysql` 数据库中的所有数据表对象

查看数据库服务的状态

```
1 show status;
```

```
| Table_open_cache_overflows | 0 |
| Tc_log_max_pages_used     | 0 |
| Tc_log_page_size          | 0 |
| Tc_log_page_waits         | 0 |
| Telemetry_metrics_supported | ON |
| Telemetry_traces_supported | ON |
| Threads_cached            | 0 |
| Threads_connected         | 1 |
| Threads_created           | 1 |
| Threads_running           | 2 |
| Tls_library_version        | OpenSSL 1.1.1k FIPS 25 Mar 2021 |
| Tls_sni_server_name        | |
| Uptime                    | 3691 |
| Uptime_since_flush_status  | 3691 |
+-----+-----+
503 rows in set (0.00 sec)
```

图 16: 查看数据库服务的状态

命令查询系统变量的值

```
1 show variables;
```

updatable_views_with_limit	YES
use_secondary_engine	ON
version	8.2.0
version_comment	MySQL Community Server - GPL
version_compile_machine	x86_64
version_compile_os	Linux
version_compile_zlib	1.2.13
wait_timeout	28800
warning_count	0
windowing_use_high_precision	ON
xa_detach_on_prepare	ON
+-----+-----+	
658 rows in set (0.04 sec)	

图 17: 查询系统变量的值

查询数据库服务的启动时间

```
1 show global status like 'uptime';
```

```
mysql> show global status like 'uptime';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Uptime       | 4166  |
+-----+-----+
1 row in set (0.01 sec)
```

图 18: 查询数据库服务的启动时间



查询系统中的当前时间戳

```
1 select unix_timestamp();
```

```
mysql> select unix_timestamp();
+-----+
| unix_timestamp() |
+-----+
|          1710955749 |
+-----+
1 row in set (0.00 sec)
```

图 19: 查询系统中的当前时间戳

查询 MySQL 数据库服务的版本、当前日期、当前时间、当前用户和所使用的数据库

```
1 select version(), curdate(), curtime(), current_user(), database();
```

```
mysql> select version(), curdate(), curtime(), current_user(), database();
+-----+-----+-----+-----+-----+
| version() | curdate() | curtime() | current_user() | database() |
+-----+-----+-----+-----+-----+
| 8.2.0     | 2024-03-20 | 17:34:02 | root@localhost | mysql      |
+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

图 20: 查询相关内容

退出 MySQL

```
1 exit;
```

退出 bash

```
1 exit
```

## 2.4 通过图形工具 Navicat 连接并操作 MySQL 数据库

在 Navicat 中新建一个连接，连接到 MySQL 数据库：



图 21: 新建连接

连接成功后，通过 Navicat 的图形界面查看数据表 `instructor` 的结构和内容

对象		instructor @dbcourse (dbcourse) - 表						
保存		添加字段	插入字段	删除字段	主键	上移	下移	
字段	索引	外键	检查	触发器	选项	注释	SQL 预览	
名	类型	长度	小数点	不是 null	虚拟	键	注释	
ID	varchar	5		<input checked="" type="checkbox"/>	<input type="checkbox"/>	1		
name	varchar	20		<input checked="" type="checkbox"/>	<input type="checkbox"/>			
dept_name	varchar	20		<input type="checkbox"/>	<input type="checkbox"/>			
salary	decimal	8	2	<input type="checkbox"/>	<input type="checkbox"/>			

图 22: 结构

对象

instructor @dbcourse (dbcourse) - 表

instructor @dbcourse (dbcourse) - 表

开始事务

文本

筛选

排序

列

导入

导出

数据生成

创建图表

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

图 23: 内容

通过 Navicat 的图形界面创建一个名为 `dbtest` 的数据库，在其中创建与 `dbcourse` 数据库相同的数据表，并导入相同的数据

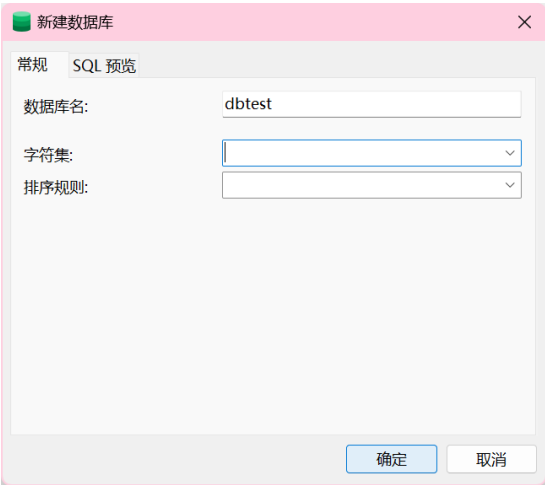


图 24: 创建数据库

保存 查询创建工具 美化 SQL 代码段 创建图表		
dbcourse dbtest 运行 停止 解释		
<pre>1 create table classroom( 2     building varchar(15), 3     room_number varchar(7), 4     capacity numeric(4,0), 5     primary key (building, room_number) 6 ); 7 create table department( 8     dept_name varchar(20), 9     building varchar(15), 10    budget numeric(12,2) check (budget &gt; 0), 11    primary key (dept_name) 12 ); 13 create table course( 14     course_id varchar(8), 15     title varchar(50),</pre>		
消息	摘要	剖析 状态
查询	消息	查询时间
insert into time_slot values ('C', 'F', '11', '0', '11', '50')	Affected rows: 1	0.002s
insert into time_slot values ('D', 'M', '13', '0', '13', '50')	Affected rows: 1	0.002s
insert into time_slot values ('D', 'W', '13', '0', '13', '50')	Affected rows: 1	0.002s
insert into time_slot values ('D', 'F', '13', '0', '13', '50')	Affected rows: 1	0.002s
insert into time_slot values ('E', 'T', '10', '30', '11', '45')	Affected rows: 1	0.002s
insert into time_slot values ('E', 'R', '10', '30', '11', '45')	Affected rows: 1	0.002s
insert into time_slot values ('F', 'T', '14', '30', '15', '45')	Affected rows: 1	0.002s
insert into time_slot values ('F', 'R', '14', '30', '15', '45')	Affected rows: 1	0.003s
insert into time_slot values ('G', 'M', '16', '0', '16', '50')	Affected rows: 1	0.002s
insert into time_slot values ('G', 'W', '16', '0', '16', '50')	Affected rows: 1	0.002s
insert into time_slot values ('G', 'F', '16', '0', '16', '50')	Affected rows: 1	0.003s
insert into time_slot values ('H', 'W', '10', '0', '12', '30')	Affected rows: 1	0.002s
insert into prereq values ('BIO-301', 'BIO-101')	Affected rows: 1	0.003s
insert into prereq values ('BIO-399', 'BIO-101')	Affected rows: 1	0.003s
insert into prereq values ('CS-190', 'CS-101')	Affected rows: 1	0.002s
insert into prereq values ('CS-315', 'CS-101')	Affected rows: 1	0.003s
insert into prereq values ('CS-319', 'CS-101')	Affected rows: 1	0.002s
insert into prereq values ('CS-347', 'CS-101')	Affected rows: 1	0.003s
insert into prereq values ('EE-181', 'PHY-101')	Affected rows: 1	0.002s

图 25: 执行相关 SQL 语句

2.5 Docker 常用操作

查看系统中所有的容器信息

```
1 sudo docker ps -a
```

base at 01:51:04							
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES	
1da884ed16dc	mysql:8.2.0	"docker-entrypoint.s..."	2 hours ago	Up 2 hours	33060/tcp, 0.0.0.0:53306→3306/tcp, :::53306→3306/tcp	dbcourse	
a2484b0159fc	hello-world	"/hello"	2 hours ago	Exited (0) 2 hours ago		festive_g	
anguly							

图 26: 查看所有的容器信息

将容器打包为镜像

```
1 sudo docker commit dbcourse dbcourse:v1
```

```
~/dbcourse
sudo docker commit dbcourse dbcourse:v1
sha256:572e5591ca7ebd577bc2b945fd080a58a9073f7484075b266d2cc3e8944c6b6d
```

图 27: 将容器打包为镜像

查看本地仓库中的镜像

```
1 sudo docker images
```

```
~/dbcourse
sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
dbcourse      v1        572e5591ca7e   49 seconds ago 619MB
mysql         8.2.0     bc861cf238f2   3 months ago  619MB
hello-world   latest    d2c94e258dcb   10 months ago 13.3kB
```

图 28: 本地仓库中的镜像

将镜像 dbcourse 保存为文件

```
1 sudo docker save -o dbcourse.tar dbcourse
```

```
~/dbcourse
sudo docker save -o dbcourse.tar dbcourse

~/dbcourse
ls
datadir dbcourse.tar
```

图 29: 将镜像 dbcourse 保存为文件

将本地仓库中的镜像 dbcourse 删除

```
1 sudo docker image rm dbcourse:v1
```

```
~/dbcourse
sudo docker image rm dbcourse:v1
Untagged: dbcourse:v1
Deleted: sha256:572e5591ca7ebd577bc2b945fd080a58a9073f7484075b266d2cc3e8944c6b6d
Deleted: sha256:f46ca4b7cc225dec255634c2ee129f8acef35d946dd7f6d195f4adb82957b4ee
```

图 30: 删除镜像

停止容器 dbcourse 的运行

```
1 sudo docker stop dbcourse
```

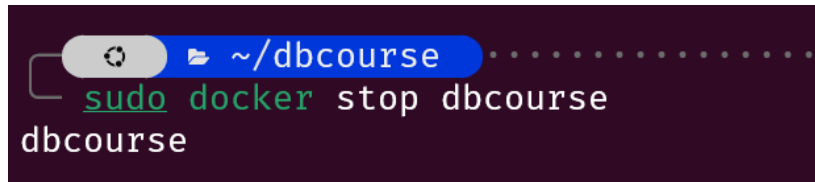


图 31: 停止容器的运行

删除容器 dbcourse

```
1 sudo docker rm dbcourse
```

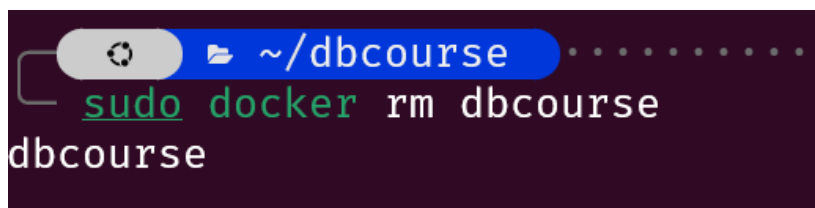


图 32: 删除容器

将镜像文件加载到本地仓库

```
1 sudo docker load --input dbcourse.tar
```



图 33: 加载镜像文件

### 3 存在的问题及解决方案

在实验中，运行插入数据的 SQL 语句时，发生错误。经检查，是由于在复制 SQL 语句时，发生了在字符串中央的意外换行，导致 MySQL 无法识别。解决方案是将 SQL 语句复制到文本编辑器中，检查并删除意外的换行。

## 4 实验小结

通过本次实验，我学习了 Docker 的基本操作，掌握了通过 Docker 容器启动 MySQL 数据库实例，并连接和操作 MySQL 数据库的基本命令。同时，我还学会了使用 Navicat 连接并操作 MySQL 数据库，以及 Docker 的其他常用操作。这些知识对我今后的学习和工作都有很大的帮助。