
浙江大学

数据库系统实验报告

作业名称: SQL 数据定义和操作

姓 名: 刘韬

学 号: 3220103422

电子邮箱: xliutao17@163.com

联系电话: 13695835699

指导老师: 孙建伶

2024 年 3 月 19 日

实验名称

一、实验目的

1. 掌握关系数据库语言**SQL**的使用。
2. 使所有的 **SQL** 作业都能上机通过。

二、实验环境

操作系统： windows 11

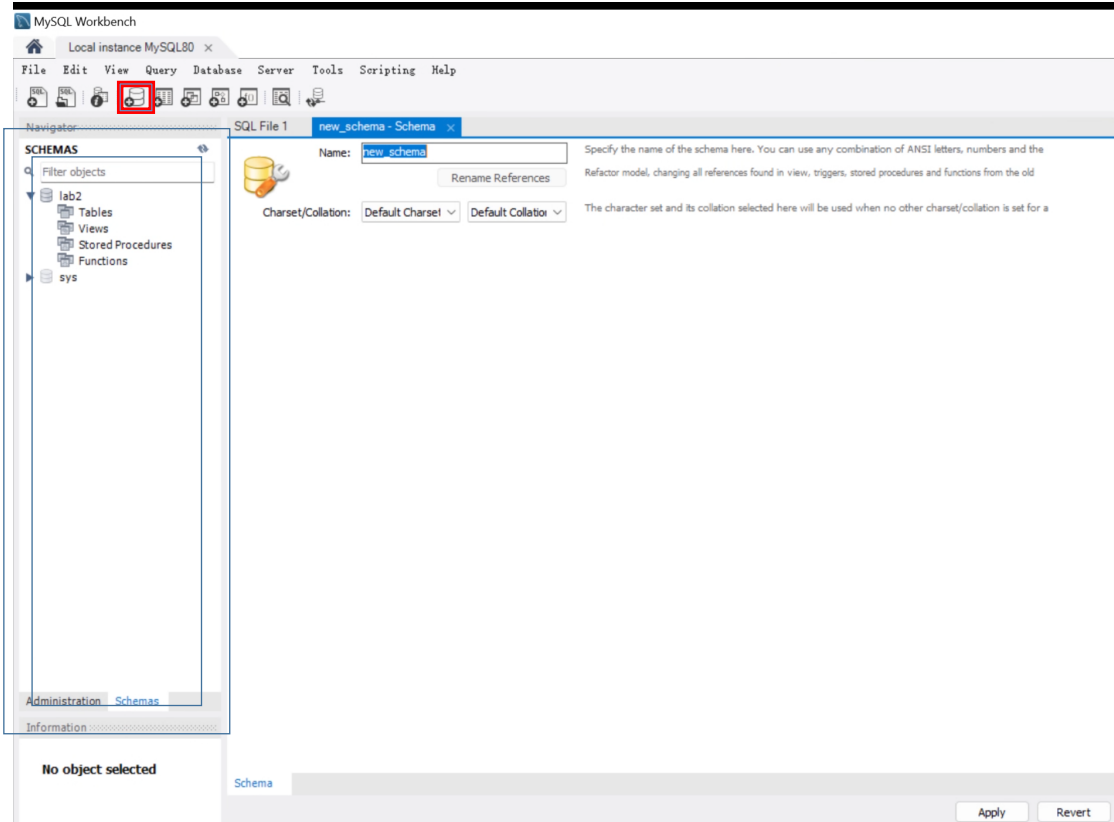
数据库管理系统： MySQL 8.0.36

图形化工具： MySQL Workbench

三、实验流程

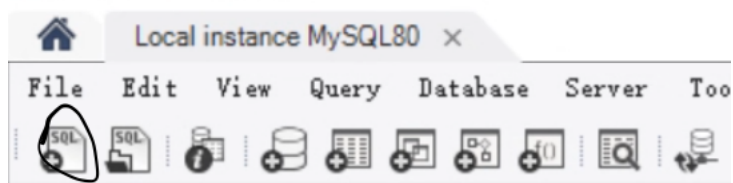
3.1 建立数据库

点击圈出的按钮，出现如下界面，自定义 Name 为 lab2，然后一直 apply 直到完成创建 schema，建立新架构即建立一个数据库。在下图的框内已经出现了新建的数据库 lab2.

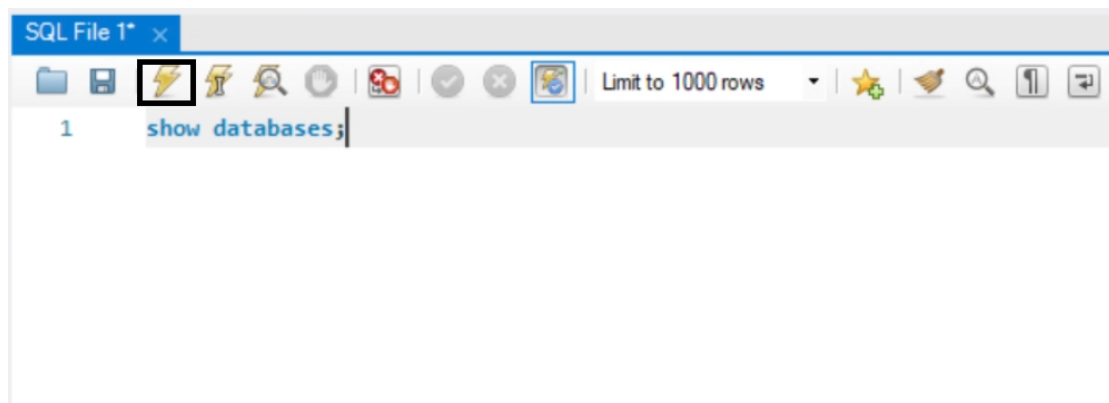


3.2 数据定义：表的建立/删除/修改；索引的建立/删除；视图的建立/删除

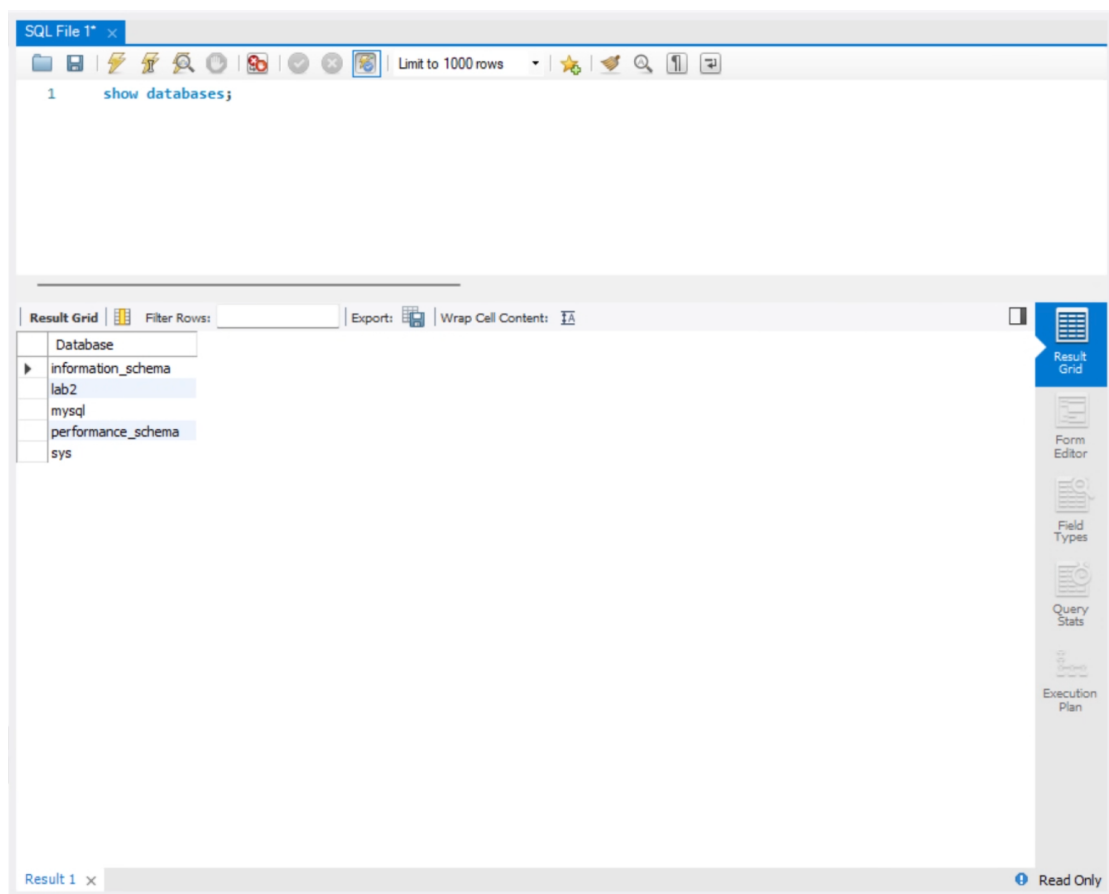
MySQL Workbench



点击圈出的 SQL+按钮，进入如下编辑界面

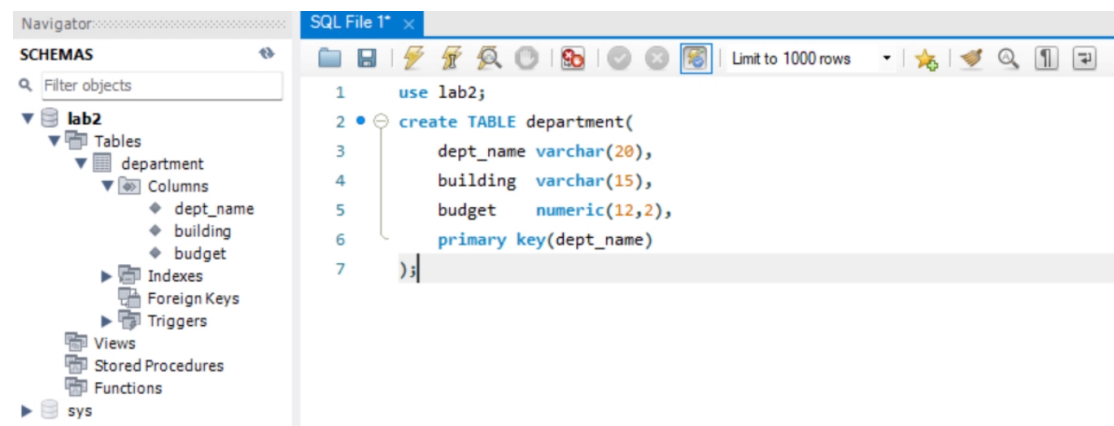


输入 SQL 语句，点击上方闪电按钮，刷新得到下面的显示：



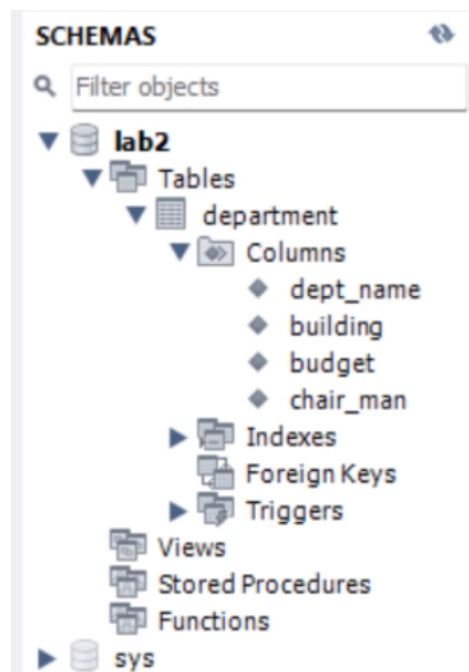
3.2.1 表的建立

输入下面的语句，点击执行，在刷新后可以看到多了一张表 department



3.2.2 表的修改

运用 Alter 语句对表进行修改，输入 alter table department add chair_man varchar(20);执行语句后点击刷新就可以看到表内多了一个名为 chair_man 的属性



接下来执行：alter table department drop column chair_man;就可以把刚才添加的属性删除。

3.2.3 表的删除

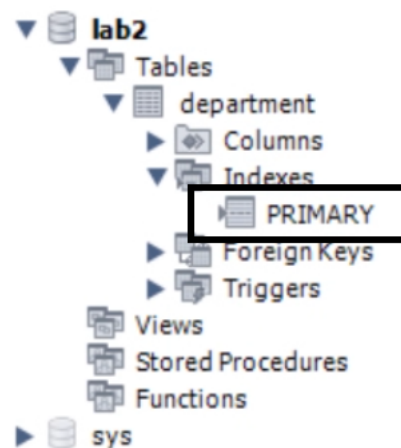
执行语句：drop table department;就可以把刚才建立的表删除。

以下是前述指令的执行情况，均执行成功。

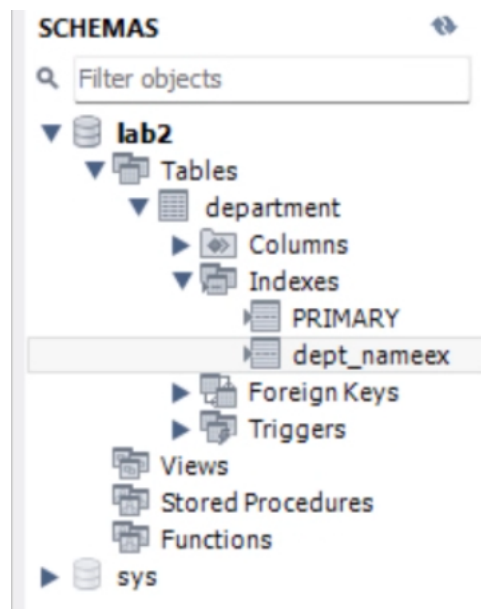
Output		
Action Output		
#	Time	Action
✓ 5	11:08:38	create TABLE department(dept_name varchar(20), building varchar(15), budget numeric(12,2), primary key(dept_name))
✓ 6	11:12:54	use lab2
✓ 7	11:14:27	alter table department add chair_man varchar(20)
✓ 8	11:17:05	alter table department drop column chair_man
✓ 9	11:18:36	drop table department

3.2.4 索引的建立

重新建立表以后, 未手动建立索引时, 只有系统自动为 primary key 建立的索引。



我们使用指令: `create index dept_nameex on department(dept_name);`



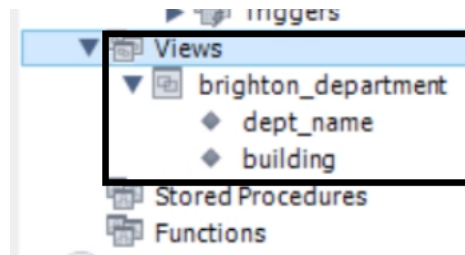
3.2.5 索引的删除

使用指令: `drop index dept_nameex on department;`即可将刚刚建立的索引删除。

3.2.6 视图的建立

使用下面的指令建立视图:

```
create view Brighton_department
as
select dept_name, building
from department
where dept_name = 'Brighton'
;
```



3.2.7 视图的删除

执行指令：drop view Brighton_department;即可删除视图

3.3 数据更新用 insert/delete/update 命令插入/删除/修改表数据。

3.3.1 数据插入：

Insert

```
30 • insert into department value('cs','A-1','100000');
31 • insert into department value('physics','A-2','10000');
32 • insert into department value('math','A-3','20000');
33 • insert into department value('biology','A-4','30000');
34 • insert into department value('chinese','B-1','1000');
```

接着我们执行 select * from department;

```
36 • select * from department;
```

	dept_name	building	budget
▶	biology	A-4	30000.00
	chinese	B-1	1000.00
	cs	A-1	100000.00
	math	A-3	20000.00
	physics	A-2	10000.00
•	NULL	NULL	NULL

可以看到我们刚刚到五条插入都完成了。

3.3.2 数据修改

我们可以使用 update 指令修改数据

```
38 • update department
39     set building = 'C-1'
40     where dept_name = 'cs';
41
42 • select * from department;
```

使用 select * from department 来查询

	dept_name	building	budget
▶	biology	A-4	30000.00
	chinese	B-1	1000.00
	cs	C-1	100000.00
	math	A-3	20000.00
	physics	A-2	10000.00
*	NULL	NULL	NULL

可以看到 cs 的 building 得到了修改由'A-1'变成了'C-1'

3.3.3 删除数据

执行指令：delete from department where budget = 1000

	dept_name	building	budget
▶	biology	A-4	30000.00
	chinese	B-1	1000.00
	cs	C-1	100000.00
	math	A-3	20000.00
	physics	A-2	10000.00
*	NULL	NULL	NULL

结果如下：主键名为'chinese'的元组被删除了；

	dept_name	building	budget
▶	biology	A-4	30000.00
	cs	C-1	100000.00
	math	A-3	20000.00
	physics	A-2	10000.00
*	NULL	NULL	NULL

在删除的时候遇到问题在章节四中写出。

4 数据查询

4.1 单表查询

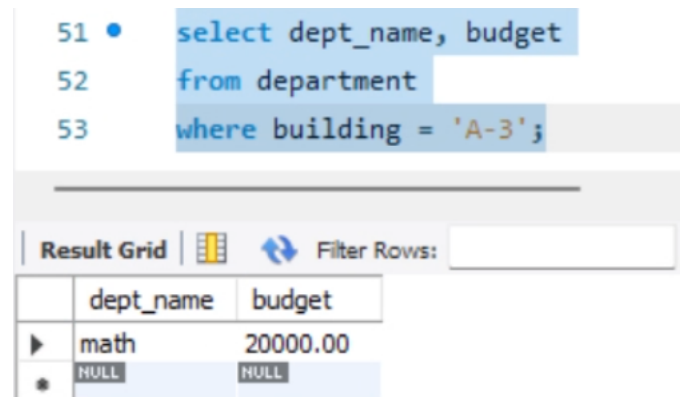
运行如下的查询语句：

```
select dept_name, budget
```

```
from department
```

```
where building = 'A-3';
```

可以看到结果显示了 building = 'A-3'的数据条目



The screenshot shows a SQL query editor with the following query:

```
51 • select dept_name, budget
52   from department
53  where building = 'A-3';
```

Below the query editor, there is a "Result Grid" tab. The results are displayed in a table with two columns: dept_name and budget.

	dept_name	budget
▶	math	20000.00
•	NULL	NULL

4.2 多表查询

多表查询，我们再建立几个表

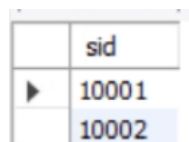
```
56 • create table student(
57     dept_name varchar(20),
58     sid varchar(10),
59     primary key (dept_name, sid)
60 );
61
62 • insert into student value('cs', '10001');
63 • insert into student value('cs', '10002');
64 • insert into student value('physics', '10003');
65 • insert into student value('math', '20001');
66 • insert into student value('chinese', '30001');
```

然后执行多表查询：

```
select sid
```

```
from student, department
```

```
where student.dept_name = department.dept_name AND department.budget =
100000;
```



The screenshot shows a table with one column named 'sid'. It contains two rows of data: 10001 and 10002.

sid
10001
10002

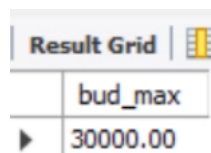
结果与插入相符合，budget 为 100000 的 department 为 cs，查询到两个结果。

4.3 嵌套子查询:

嵌套子查询实际上就是把内部的查询结果作为外层的查询条件，执行下面语句

```
select max(budget) as bud_max  
from (  
    select budget, dept_name  
    from department as d  
    where d.dept_name <> 'cs'  
) as no_cs;
```

我们找到了除了 cs 以外的具有最大 budget 的部门的 budget。



	bud_max
▶	30000.00

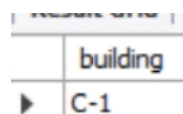
5. 视图操作：通过视图的数据查询和数据修改 重新建立视图

```
create view cs_department  
as  
    select dept_name, building  
    from department  
    where dept_name = 'cs';
```

输入下面的语句

```
select building  
from cs_department;
```

这条命令是在视图 cs_department 中查询 building，结果如下



	building
▶	C-1

数据修改:

```
update cs_department  
SET building = 'B-3'  
where dept_name = 'cs';
```

查询结果

```
select * from cs_department;
```

从 view 的角度看，属性得到修改了

Result Grid	Filter Rows:
dept_name	building
cs	B-3

`select * from department;`

	dept_name	building	budget
▶	biology	A-4	30000.00
	cs	B-3	100000.00
	math	A-3	20000.00
	physics	A-2	10000.00
•	NULL	NULL	NULL

从另一个表中也可以看到 cs.building 值也发生了改变。

四、遇到的问题及解决方法

[error 1146]在执行索引删除的指令时出现报错，原因是指令最后的分号误输入为中文分号，导致错误。检查输入以后完成指令。

[error code: 1175]在执行数据删除的时候出现报错，原因是 mysql 运行在 safe-update 模式下，导致非主键条件下无法执行删除操作；随后我使用指令 `SET SQL_SAFE_UPDATES = 0;` 关闭了 safe-update 模式，再次执行指令就可以通过了。

[Error Code: 1248]. Every derived table must have its own alias。在使用嵌套子查询的过程中出现错误，开始查询时并未添加 `as no_cs`，这导致了这个查询的表没有自己的名字。在多级查询中要求给每个表都有一个别名才能完成搜索。

五、总结

本次实验中，我们完成了新建数据库、数据定义、数据更新、数据查询、以及视图操作，基本完成了表的建立/删除/修改；索引的建立/删除；视图的建立/删除；能用 `insert/delete/update` 命令插入/删除/修改数据；以及单表/多表/嵌套子查询。圆满地完成了实验内容和操作，也在一些操作中发现了一些规范性的操作问题，在实践中得到了学习。