

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: 秦嘉俊

学 号: 3210106182

电子邮箱: hobbitqia@zju.edu.cn

联系电话: 18084011903

指导老师: 孙建伶

2023 年 3 月 13 日

SQL 数据定义和操作

一、实验目的

1. 掌握关系数据库语言 SQL 的使用。
2. 面向某个应用定义数据模式和操作数据。

二、实验环境

操作系统: Windows 10

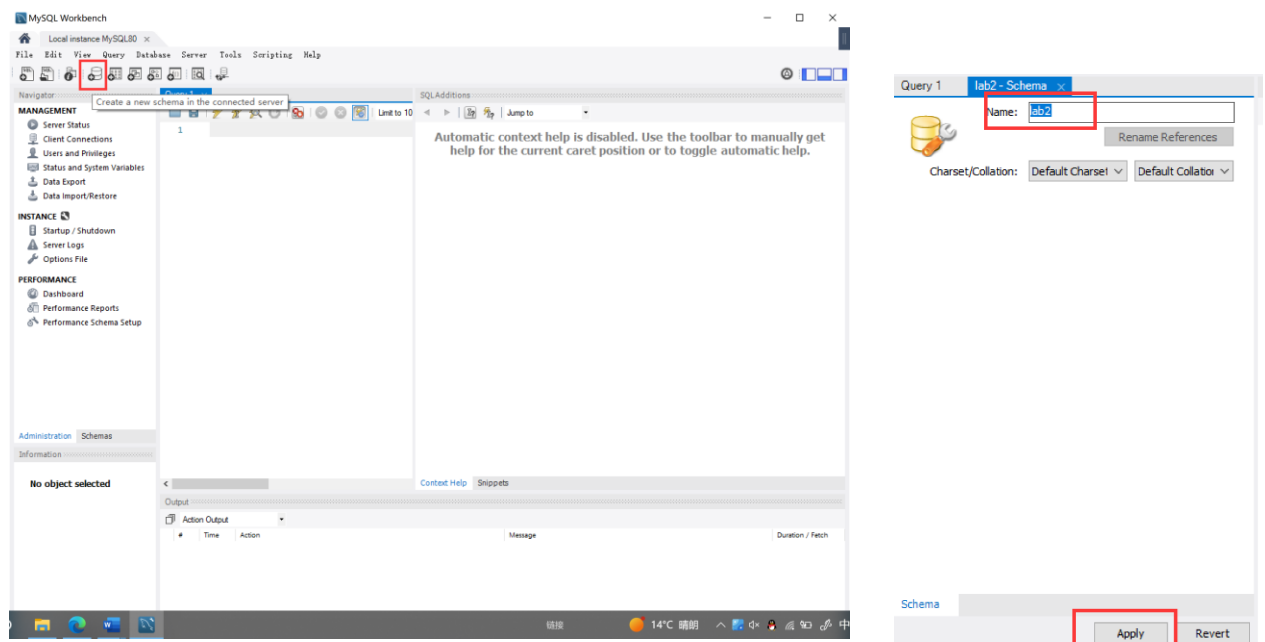
数据库管理系统: MySQL 8.0.32.0

工具: MySQL Workbench

三、实验流程

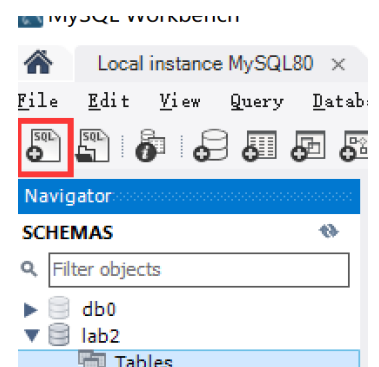
3.1 建立数据库

首先进入 MySQL Workbench 并且本地登录。点击所示的图标，代表在连接的服务器中创建新架构。即新建一个数据库。相当于 create database 指令。填写数据库名称 lab2，点击 Apply 确定。新建的数据库可以在 Schemas 一栏中显示出来，如下图所示。

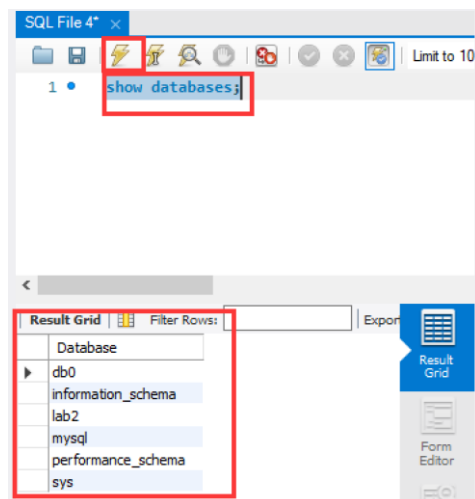


3.2 数据定义

点击 SQL+图表，在下方的文本栏中可以编辑 SQL 语句。

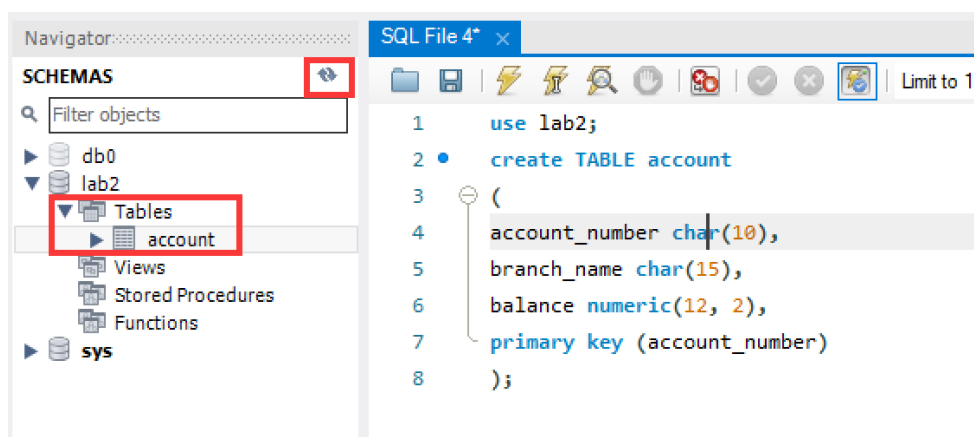


在 SQL file 中，编辑的 SQL 语句不会立刻执行，而是选中后选择“闪电”图标才能执行。如图所示。结果会在 Result Grid 一栏中显示。



3.2.1 表的建立

用 use lab2;语句指明你想在哪个数据库上执行操作，再执行如下建表语句：



可以看到多出了一张 account 表。

注意每次执行 SQL 语句操作后，点击刷新，才能看到更新。

3.2.2 表的修改

运用 Alter 语句对表进行修改，例如增加一个名为“branch_city”属性

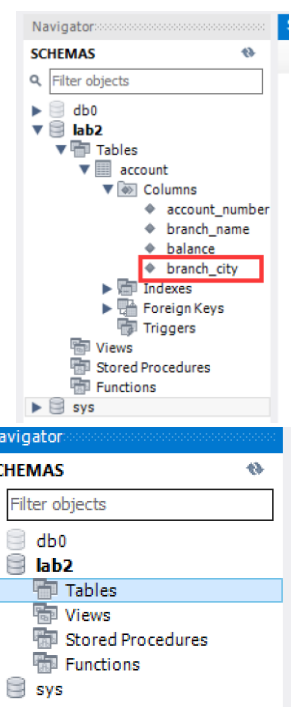
输入语句为 ALTER TABLE account ADD branch_city char(30); 得到右图的结果

ALTER TABLE account DROP COLUMN branch_city; 这样即可删除刚刚增加的属性

3.2.3 表的删除

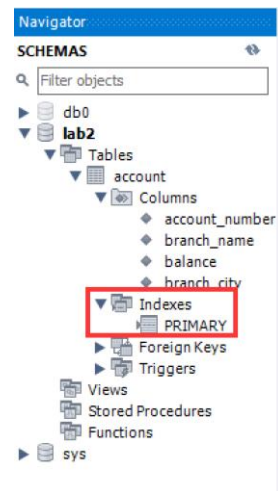
运行 DROP TABLE account; 即可删除整张表

（如右图）（为确保后续实验进行，测试了删除功能后我们又按之前的步骤重新搭建了这张表）

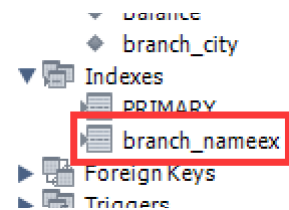


3.2.4 索引的建立

索引是为了更加快速高效地查询数据。未手动建立索引时，只有系统自动为 primary key 建立的索引。



我们可以用语句 `CREATE INDEX branch_nameex ON account (branch_name);` 增加一个 `branch_name` 的索引



3.2.5 索引的删除

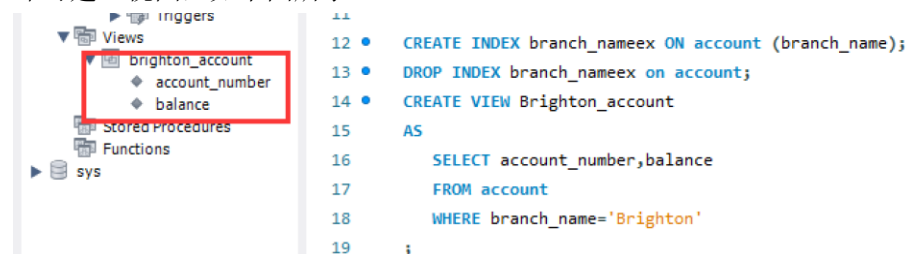
执行语句 `DROP INDEX branch_nameex on account;` 即可删除刚刚建立的索引

3.2.6 视图的建立

执行如下语句

```
CREATE VIEW Brighton_account
AS
    SELECT account_number, balance
    FROM account
    WHERE branch_name='Brighton';
```

即可建立视图，如下图所示



3.2.7 视图的删除

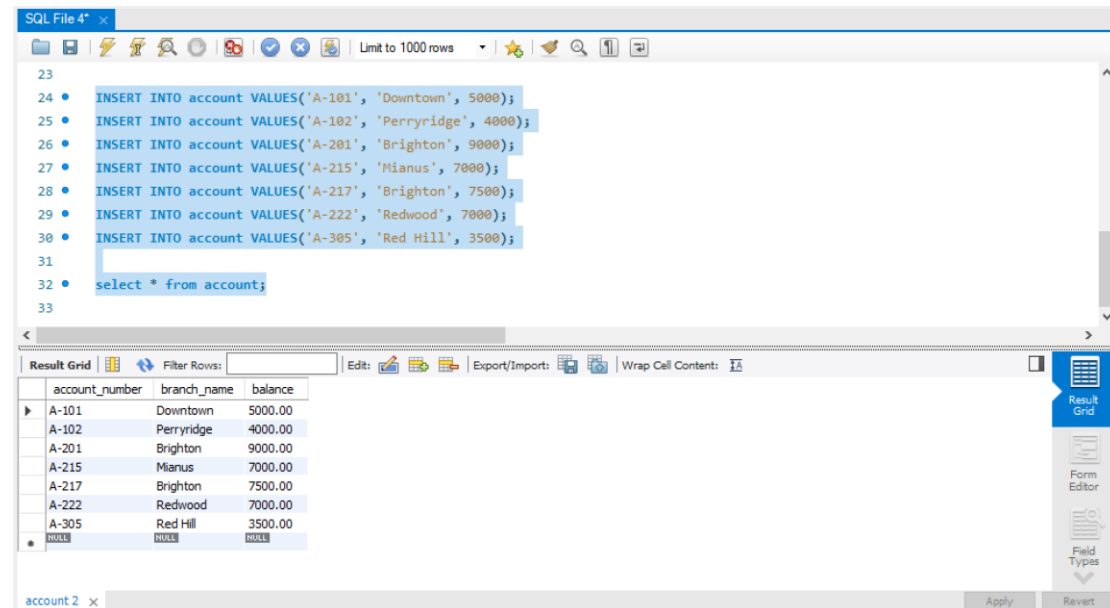
执行 `DROP VIEW Brighton_account;` 即可成功删除视图

3.3 数据更新

3.3.1 insert

我们可以用如下指令对数据库进行数据插入。

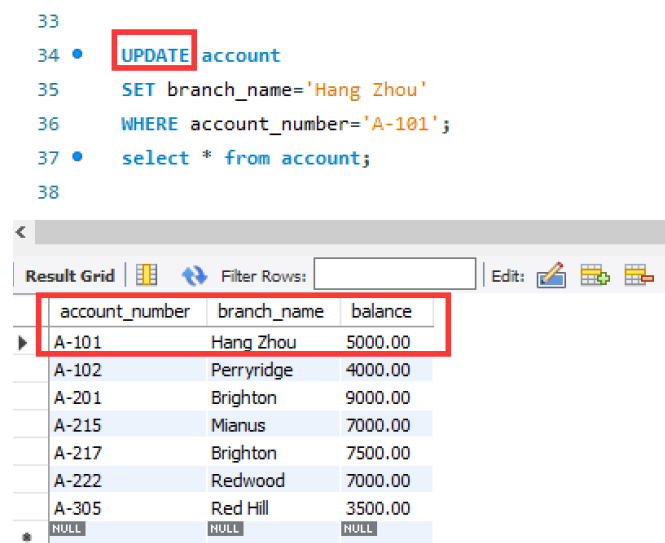
最后用 `select * from account;` 查询插入后的数据库。可以看到我们刚刚的七条插入都在其中显示。



3.3.2 update

我们可以用如下指令对数据库进行数据更新。

最后用 `select * from account;` 查询更新后的数据库。可以看到第一条数据的 `branch_name` 变为了 `Hang Zhou`.



3.3.3 delete

我们可以用右图中指令对数据库进行数据删除。

最后用 `select * from account;` 查询删除后的数据库。

可以看到 `account_number` 为 A-101 的数据已经不再出现，说明我们删除成功。

```
39 • DELETE FROM account
40   WHERE account_number='A-101';
41 • select * from account;
42
```

account_number	branch_name	balance
A-102	Perryridge	4000.00
A-201	Brighton	9000.00
A-215	Mianus	7000.00
A-217	Brighton	7500.00
A-222	Redwood	7000.00
A-305	Red Hill	3500.00
NULL	NULL	NULL

3.4 数据查询

3.4.1 单表查询

我们运行如下的查询语句

```
SELECT account_number,balance
```

```
FROM account
```

```
WHERE branch_name='Mianus';
```

可以看到，结果显示了 `branch_name` 为 Mianus 的数据条目。

```
43 • SELECT account_number,balance
44   FROM account
45   WHERE branch_name='Mianus';
```

account_number	balance
A-215	7000.00
NULL	NULL

3.4.2 多表查询

为了执行多表查询，我们再新建一个表 `depositor`

```
• create TABLE depositor
  (
    customer_name char(20) not null,
    account_number char(15) not null,
    primary key (customer_name, account_number)
  );

• INSERT INTO depositor VALUES('Qin Jiajun', 'A-101');
• INSERT INTO depositor VALUES('Zhang Rui', 'A-215');
• INSERT INTO depositor VALUES('Du Jialu', 'A-102');
• INSERT INTO depositor VALUES('Yao Xin', 'A-305');
• INSERT INTO depositor VALUES('Yuan Quanze', 'A-201');
• INSERT INTO depositor VALUES('Luo Xinyue', 'A-217');
• INSERT INTO depositor VALUES('Sun Jianling', 'A-222');

• select * from depositor;
```

经过多次插入后，这个表中的数据目前的数据可见右图

customer_name	account_number
Du Jialu	A-102
Luo Xinyue	A-217
Qin Jiajun	A-101
Sun Jianling	A-222
Yao Xin	A-305
Yuan Quanze	A-201
Zhang Rui	A-215
NULL	NULL

随后就可以开始我们的多表查询

运行如下查询指令

```
SELECT customer_name, balance
FROM account, depositor
WHERE account.account_number=depositor.account_number;
```

得到 customer_name 与 balance 对应的结果

```
64 • SELECT customer_name, balance
65 FROM account, depositor
66 WHERE account.account_number=depositor.account_number;
67
```

< Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	customer_name	balance
▶	Du Jialu	4000.00
	Luo Xinyue	7500.00
	Sun Jianling	7000.00
	Yao Xin	3500.00
	Yuan Quanze	9000.00
	Zhang Rui	7000.00

3.4.3 嵌套子查询

嵌套子查询实质把内层的查询结果作为外层的查询条件。

通过运行如下的语句，我们可以得到存储在 Brighton 的最大余额。

```
71
72 • SELECT MAX(balance) as Bri_max
73 FROM (SELECT customer_name, branch_name, balance
74 FROM account, depositor
75 WHERE account.account_number = depositor.account_number)
76 AS acc_dep
77 WHERE branch_name = 'Brighton';
```

< Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	Bri_max
▶	9000.00

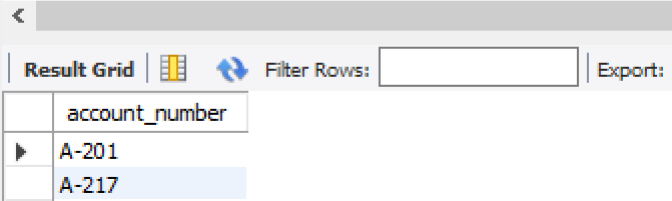
3.5 视图操作

在 SQL 中，视图是基于 SQL 语句的结果集的可视化的表。

3.5.1 视图数据查询

在之前的部分中，我们新建了视图而又将其删去。所以这里我们要重新建立一次视图。随后进行我们的视图数据查询。

```
79 • CREATE VIEW Brighton_account
80 AS
81     SELECT account_number,balance
82     FROM account
83     WHERE branch_name='Brighton'
84 ;
85
86 • SELECT account_number
87     FROM Brighton_account;
88
```

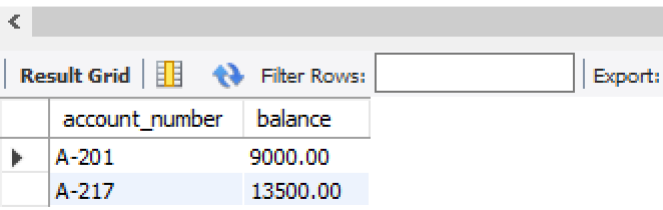


account_number
A-201
A-217

3.5.2 视图数据修改

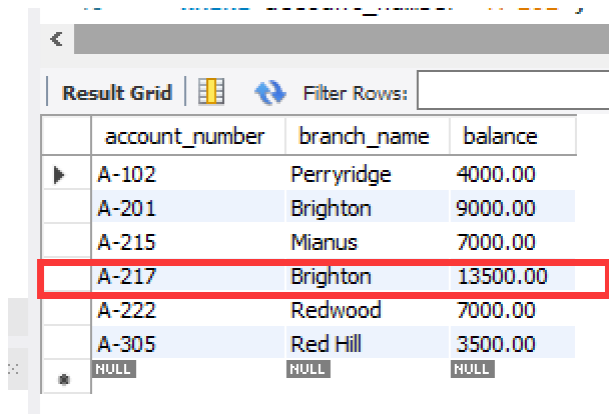
我们通过视图修改其中的数据，这里我们对 A-217 这个账户进行余额修改，于是他的余额从 7500 变为了 13500，而另一个账户的余额保持在 9000.

```
88
89 • UPDATE Brighton_account
90     SET balance=balance*1.5
91     WHERE account_number='A-217';
92
93 • SELECT *
94     FROM Brighton_account;
95
```



account_number	balance
A-201	9000.00
A-217	13500.00

随后我们运行 `select * from account;` 观察另一张表（下图）可以看到 A-217 这个账户的余额也显示已经改变。



Result Grid | Filter Rows:

	account_number	branch_name	balance
▶	A-102	Perryridge	4000.00
	A-201	Brighton	9000.00
	A-215	Mianus	7000.00
	A-217	Brighton	13500.00
	A-222	Redwood	7000.00
	A-305	Red Hill	3500.00
✱	NULL	NULL	NULL

四、遇到的问题及解决方法

本次实验较为顺利，没有遇到问题。

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。