数据库系统实验报告（二）

课程名称：数据库系统原理 实验项目名称：SQL 数据定义和操作

学生姓名：刘轩铭 专业：软件工程 学号：3180106071

指导老师：周波 实验日期：2020 年 3 月 13 日

一、实验目的和要求

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、实验内容和要求

1. 建立数据库。
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
3. 数据更新： 用 insert/delete/update 命令插入/删除/修改表数据。
4. 数据查询： 单表查询，多表查询， 嵌套子查询等。
5. 视图操作：通过视图的数据查询和数据修改
6. 所有的 SQL 作业都上机通过。
7. 完成实验报告。

三、主要仪器设备

1. 操作系统： Windows
2. 数据库管理系统：SQL Server 或 MySQL （本次实验选用 MySQL）

四、操作方法与实验步骤

4.1 建立作业 3.8 中的银行数据库

输入下面的代码，可以建立并选择我们需要的银行数据库

```
create table bank;
use bank;
```

命令行返回结果说明我们已经建立并转到了这个数据库，结果见第五部分（下同）

4.2 建立银行数据库中的六个表，以及相应的索引和视图

以第一个表 branch 为例，输入以下的代码，可以建立该表：

```
create table if not exists branch(
    branch_name varchar(40),
    branch_city varchar(40),
    asset double,
    primary key (branch_name)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

用同样的方法可以建立其余的五个表。

给表 branch 添加一个字段 customer_cnt,然后删除它：

接下来，输入如下的代码，可以给这六个表创建索引（以 branch 为例）：

```
ALTER table branch ADD INDEX brh(branch_name);
```

然后可以删除该索引：

```
ALTER table branch drop INDEX brh;
```

以 branch 表为基础建立视图 branch_view，包括字段 branch_name, branch_city:

```
CREATE VIEW branch_biew AS SELECT branch_name, branch_city from branch;
```

4.3 用 insert/delete/update 命令插入/删除/修改表数据。

批量执行如下语句插入如下数据供后面实验使用（以 branch 为例）：

```
insert into branch values('bank_a','Hangzhou',1500000);
insert into branch values('bank_b','Changsha',18000);
insert into branch values('bank_c','Beijing',1500000);
insert into branch values('bank_d','Changsha',17000);
insert into branch values('bank_e','Chengdu',1900000);
insert into branch values('bank_f','Changsha',2000000);
insert into branch values('bank_g','Hangzhou',1900000);
```

这样我们完成了插入指令。

分别对第一条记录进行删除和修改操作：

```
delete from branch where branch_name = 'bank_a';
insert into branch values('bank_a','Hangzhou',1500000);
update branch set asset = 17000 where branch_name = 'bank_a';
```

对其余的六张表也进行这样的插入操作，为后续的查询做准备。

4.4 单表查询，多表查询， 嵌套子查询等。

输入下列代码，分别完成作业 3.8 的三道题目：

```
select customer_name
from customer
where customer_name not in (select customer_name from borrower);

select C1.customer_name
from customer as C1, customer as C2
where C2.customer_name = 'Smith' and C1.customer_street = C2.customer_street and
C1.customer_city = C2.customer_city;

select branch_name
from depositor natural join account natural join customer
where customer_name = 'Harrison';
```

这样我们完成了单表查询和多表查询，嵌套子查询的操作在之后的题目里也进行了验证，这里就不赘述了。

我对作业的每一个结果都进行了验证，结果都说明我们的查询是可以通过的。

4.5 通过视图的数据查询和数据修改

通过 branch_view 视图 列出所有的银行分行字段信息

```
SELECT branch_name, branch_city from branch_view;
```

通过 branch_view 视图更新'bank_g'的位置改为' Shanghai' （注意观察 branch 表中的数据也相应发生了变化，我理解视图与数据表的区别与联系）

```
UPDATE branch_view set branch_city = 'Shanghai' where branch_name = 'bank_g';
```

五、实验结果与分析

5.1 建立数据库如图

```
mysql> create database bank;
Query OK, 1 row affected (0.10 sec)

mysql> use bank;
Database changed
```

返回结果说明已经建立了该数据库。

5.2 对表/索引/试图的操作结果展示

```
mysql> create table if not exists branch(
    ->     branch_name varchar(40),
    ->     branch_city varchar(40),
    ->     asset double,
    ->     primary key (branch_name)
    -> )ENGINE=InnoDB DEFAULT CHARSET=utf8;
Query OK, 0 rows affected, 1 warning (1.65 sec)
```

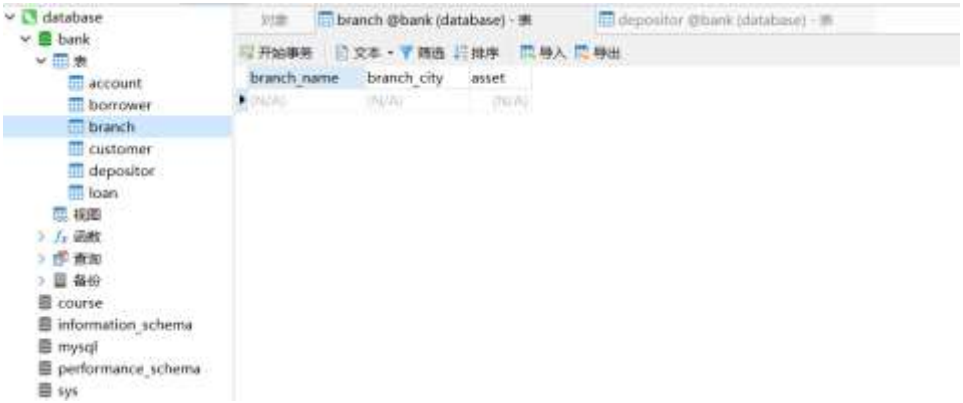
如图所示，我们建立了 branch 表，并指定了 branch_name 属性作为主键。

对表进行修改操作，结果如下：

```
mysql> ALTER TABLE branch ADD cunstomer_cnt int;
Query OK, 0 rows affected (0.52 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE branch drop cunstomer_cnt;
Query OK, 0 rows affected (1.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

根据图形界面我们看出，我们建立了这个数据库中的六个表。



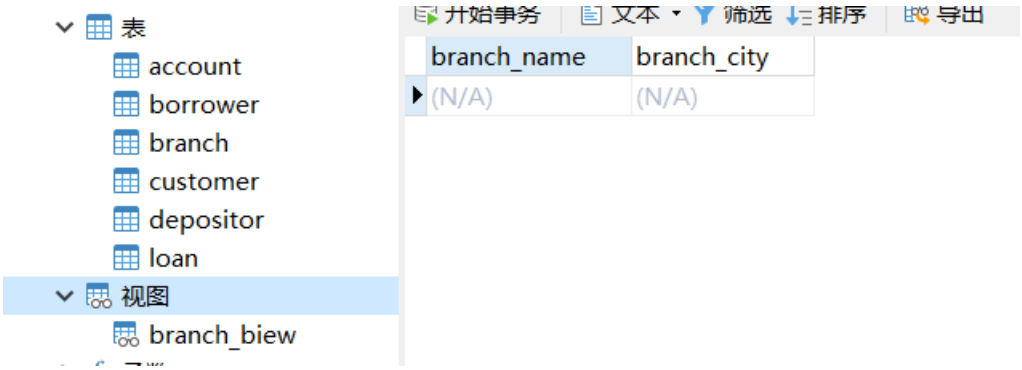
接下来为表建立索引，并删除索引，结果如图：

```
mysql> ALTER table branch ADD INDEX brh(branch_name);
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER table branch drop INDEX brh;
Query OK, 0 rows affected (0.17 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

建立 branch 为基础的视图，如果如下：

```
mysql> CREATE VIEW branch_biew AS SELECT branch_name, branch_city from branch;
Query OK, 0 rows affected (0.14 sec)
```



5.3 插入，删除，更新表中数据

以 branch 为例，如图为插入的结果：

```
mysql> insert into branch values('bank_a','Hangzhou',1500000);
Query OK, 1 row affected (0.28 sec)

mysql> insert into branch values('bank_b','Changsha',18000);
Query OK, 1 row affected (0.10 sec)

mysql> insert into branch values('bank_c','Beijing',1500000);
Query OK, 1 row affected (0.07 sec)

mysql> insert into branch values('bank_d','Changsha',17000);
Query OK, 1 row affected (0.08 sec)

mysql> insert into branch values('bank_e','Chengdu',1900000);
Query OK, 1 row affected (0.06 sec)

mysql> insert into branch values('bank_f','Changsha',2000000);
Query OK, 1 row affected (0.09 sec)

mysql> insert into branch values('bank_g','Hangzhou',1900000);
Query OK, 1 row affected (0.09 sec)
```

branch_name	branch_city	asset
bank_a	Hangzhou	1500000
bank_b	Changsha	18000
bank_c	Beijing	1500000
bank_d	Changsha	17000
bank_e	Chengdu	1900000
bank_f	Changsha	2000000
bank_g	Hangzhou	1900000

之后一次性进行删除和更新操作：

```
mysql> delete from branch where branch_name = 'bank_a';
Query OK, 1 row affected (0.09 sec)

mysql> insert into branch values('bank_a','Hangzhou',1500000);
Query OK, 1 row affected (0.12 sec)

mysql> update branch set asset = 17000 where branch_name = 'bank_a';
Query OK, 1 row affected (0.07 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

5.4 例 3.8 的查找结果

首先给出的设定数据图形化展示：

account_number	branch_name	balance
10001	bank_a	100
10002	bank_b	100
10003	bank_c	100
10004	bank_a	100
10005	bank_b	100
10006	bank_c	100
10007	bank_d	100
10008	bank_e	100
10009	bank_f	100
10010	bank_a	100

customer_name	loan_number
Amy	200
David	200
Emma	200
Frank	200
Hank	200
Harrison	200

branch_name	branch_city	asset
bank_a	Hangzhou	1500000
bank_b	Changsha	18000
bank_c	Beijing	1500000
bank_d	Changsha	17000
bank_e	Chengdu	1900000
bank_f	Changsha	2000000
bank_g	Hangzhou	1900000

customer_name	account_number
Amy	10004
Amy	10008
Bob	10005
Cate	10006
David	10007
Emma	10009
Frank	10010
Harrison	10001
Harrison	10002
Harrison	10003

customer_name	customer_street	customer_city
Amy	S1	C1
Bob	S2	C2
Cate	S1	C1
David	S2	C2
Emma	S2	C2
Frank	S3	C1
Guff	S1	C2
Hank	S1	C1
Harrison	S2	C2

下面给出三道题目的查询结果：

```
mysql> select customer_name
    -> from customer
    -> where customer_name not in (select customer_name from borrower);
+---------------+
| customer_name |
+---------------+
| Bob           |
| Cate          |
| Guff          |
| Smith         |
+---------------+
4 rows in set (0.13 sec)
```

```
mysql> select C1.customer_name
    -> from customer as C1, customer as C2
    -> where C2.customer_name = 'Smith' and C1.customer_street = C2.customer_street and C1.customer_city = C2.customer_c
ity;
+---------------+
| customer_name |
+---------------+
| Bob           |
| David         |
| Emma          |
| Harrison      |
| Smith         |
+---------------+
5 rows in set (0.01 sec)
```

```
mysql> select branch_name
    -> from depositor natural join account natural join customer
    -> where customer_name = 'Harrison';
+-------------+
| branch_name |
+-------------+
| bank_a      |
| bank_b      |
| bank_c      |
+-------------+
3 rows in set (0.26 sec)
```

由自定数据表可以看出，可以看出我们的查询结果是正确的。

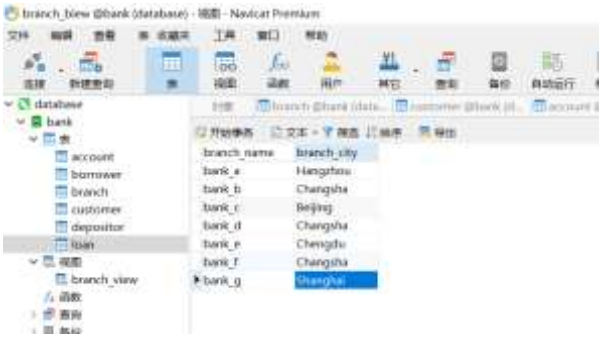
5.5 视图查询和修改的结果

通过结果我们看出，视图可以同步数据表对数据进行查询和修改

```
mysql> SELECT branch_name, branch_city from branch_view;
+-------------+-------------+
| branch_name | branch_city |
+-------------+-------------+
| bank_a      | Hangzhou    |
| bank_b      | Changsha    |
| bank_c      | Beijing     |
| bank_d      | Changsha    |
| bank_e      | Chengdu     |
| bank_f      | Changsha    |
| bank_g      | Hangzhou    |
+-------------+-------------+
7 rows in set (0.00 sec)
```

```
mysql> UPDATE branch_view set branch_city = 'Shanghai' where branch_name = 'bank_g';
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

以下是修改后的视图结果：



六、讨论与心得

此次实验是初步用 MySQL 进行一些基本的数据库操作，包括查询修改删除以及视图的一些应用于操作。在此次实验过程中，对于 SQL 语法有了进一步的熟悉和深入了解，对于复习上一节课的相关知识起到了很好的作用。

上一节课讲了很多 SQL 语句，由于没有上手操作，对于这些语句的理解并不深入，此次实验过程中， 对于 select 语句，alter 的用法，update 的使用以及笛卡尔积、自然连接这些知识点进行了很好的复习。

更重要的是在做实验的过程中，拓展了很多知识。由于一些语句不会写，需要查资料，就在网络上发现连接分为内部连接，外部连接，左连接，右连接等多种方式，还有对于同一查询的实现，也有不同的语句可以使用，比如可以使用嵌套，也可以其他方法。

唯一的不足是在此次实验中没有使用 having 语句，所以对 having 的使用还需要进一步加强练习。