实验二 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. 使所有的SQL作业都能上机通过。

二、实验平台：

操作系统：Windows10

CPU Name: Intel CoreTM i5-9300H CPU @ 2.40GHz

Property	Value
Base Frequency	2.4 GHz
Max Turbo Frequency	4.10 GHz
Cache	4 MB Intel® Smart Cache
Cores Number	4
Threads	8
TDP	45W

RAM: 16GB DDR4 2666MHz with Two channel memory

SSD: SAMSUNG MZVLB1T0HBLR-000L2

数据库管理系统： MySQL-8.0.19

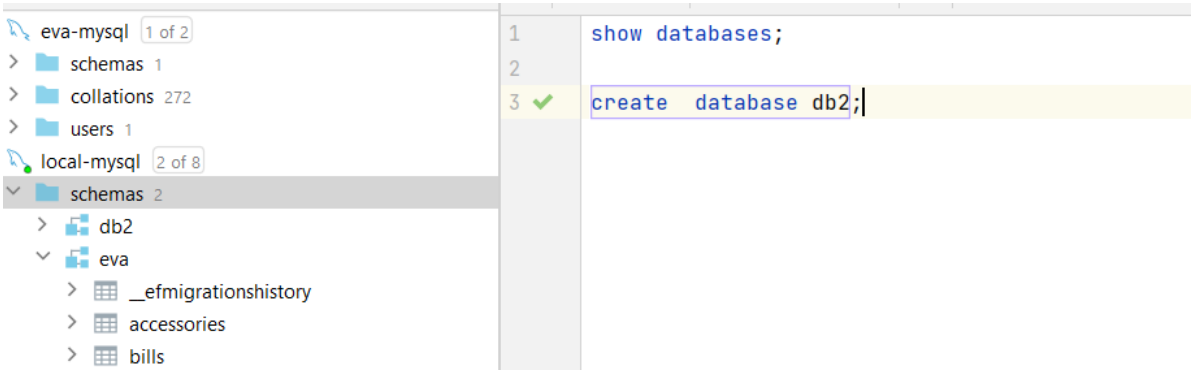
三、实验流程

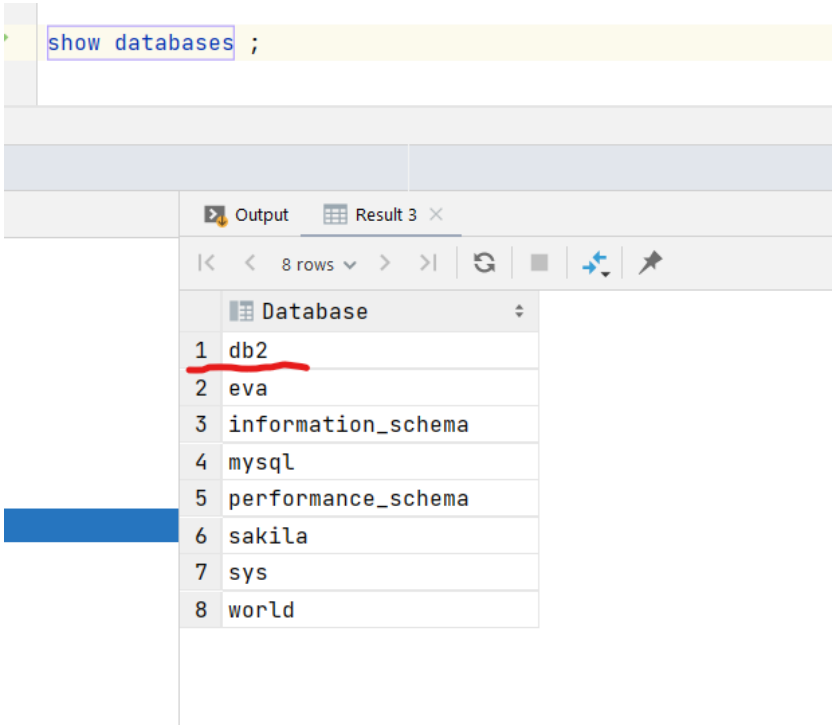
这里我选择使用了datagrip数据管理工具，并未使用mysql workbench

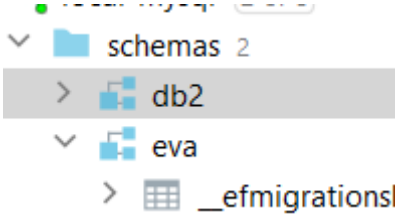
创建数据库

这里我们使用指令进行创建

```
create database name
```



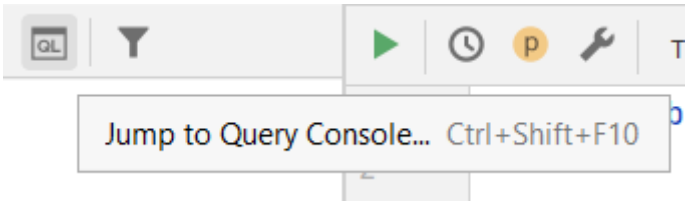


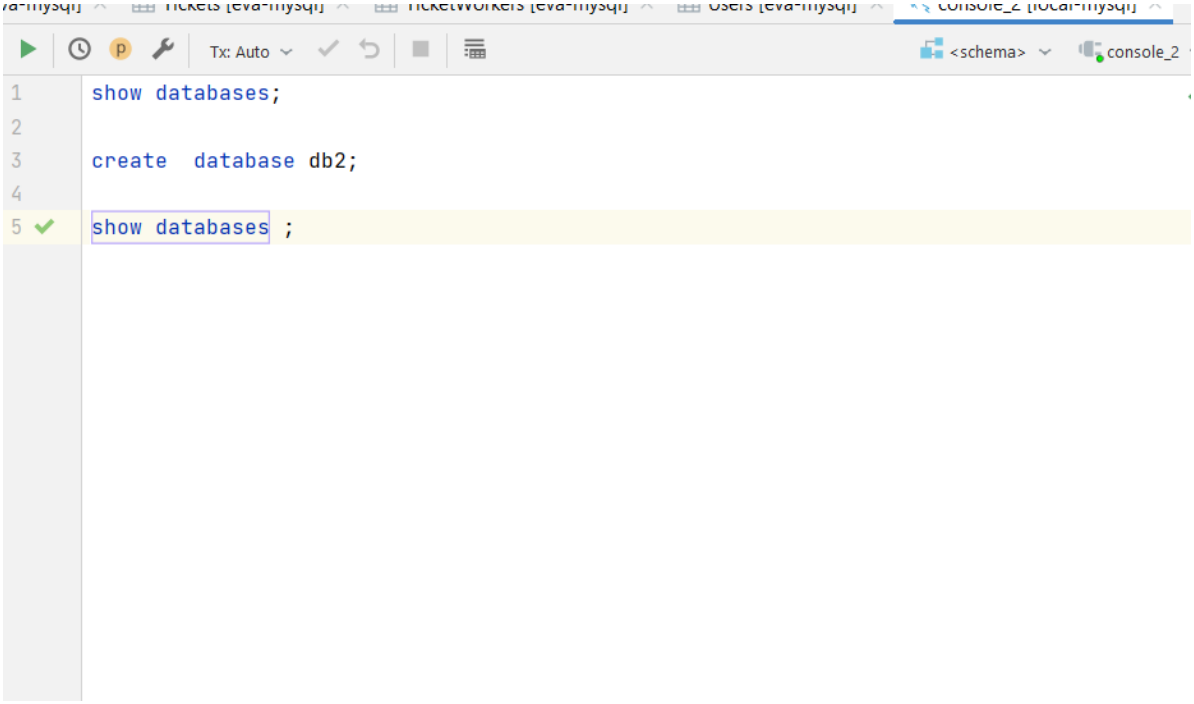


可以看到已经成功创建db2数据库

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

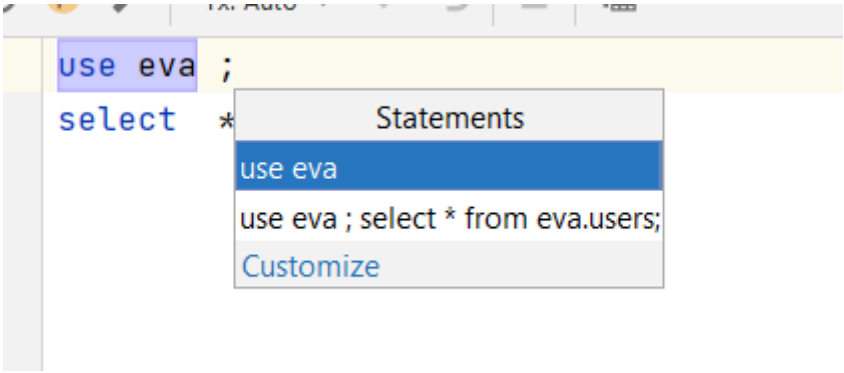
点击console，打开一个可以写SQL语句的文本。





我们直接将指针移动到相应的行，ctrl+空格键直接执行命令即可

![image-20210314223735286](https://pic.raynor.top/images/2021/03/14/image-20210314223735286.png



创建数据表

更改数据表

首先我们可以看到db2数据库下没有任何表，此时新生成的数据库里还没有表

首先用

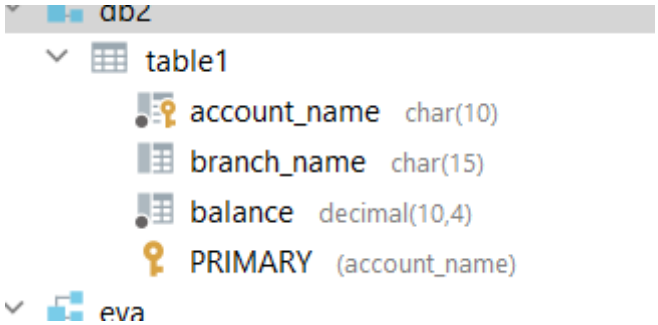
```
use db2;
```

指明数据库操作对象

之后使用

```
create table table1
(
    account_name char(10) not null,
    branch_name char(15) null,
    balance numeric(10,4) not null,
    constraint table1_pk
        primary key (account_name)
);
```

可以看到创建成功

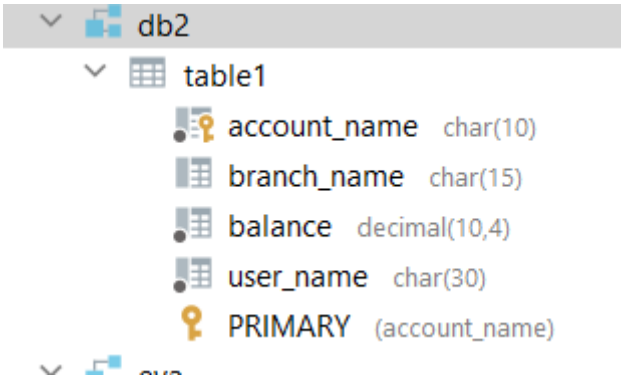


此时新生成的数据库里还没有表。

进行表的修改

```
alter table table1 add user_name char(30) not null ;

```

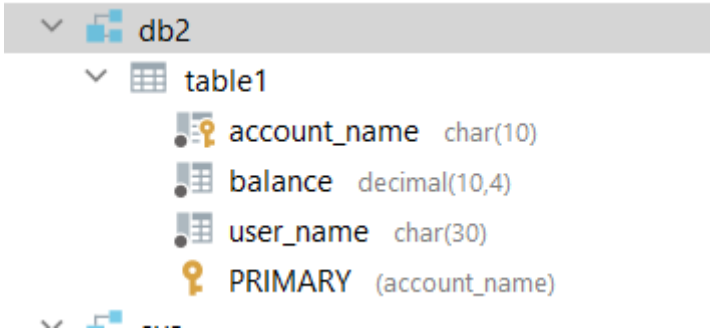
这里我们添加一个名为user_name的列



可以看到添加成功

```
alter alter  table table1 drop  branch_name;

```

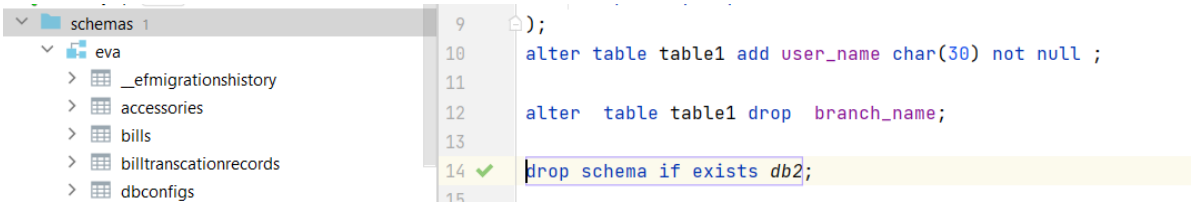
这里我们删除名为branch_name的列



可以看到删除成功

表的删除

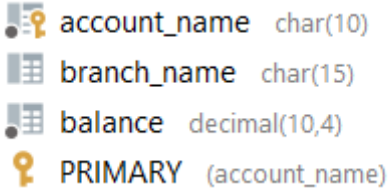
```
drop schema if exists db2;

```



可以看到表已经被删除了

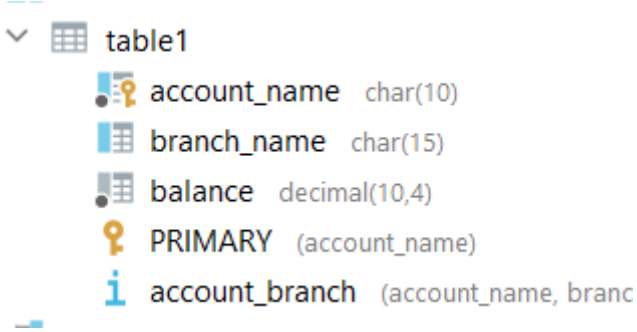
建立索引

重新使用上述建立语句建好表。当我们未使用手工建立索引时，只有系统为主键建立的索引



这里我们创建一个组合索引

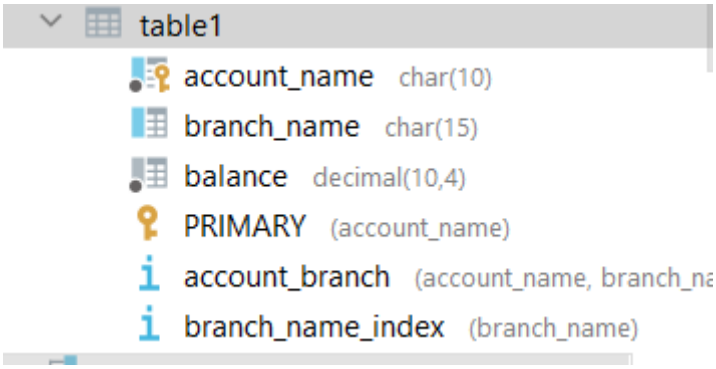
```
alter  table table1 add  index   account_branch (account_name, branch_name)
```



可以看到索引已经成功建立。

这里我们再建立一个普通索引

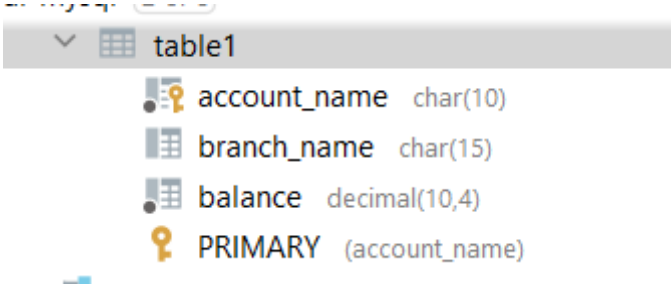
```
alter  table table1 add  index  branch_name_index (branch_name)
```



可以看到已经生成了索引

删除索引

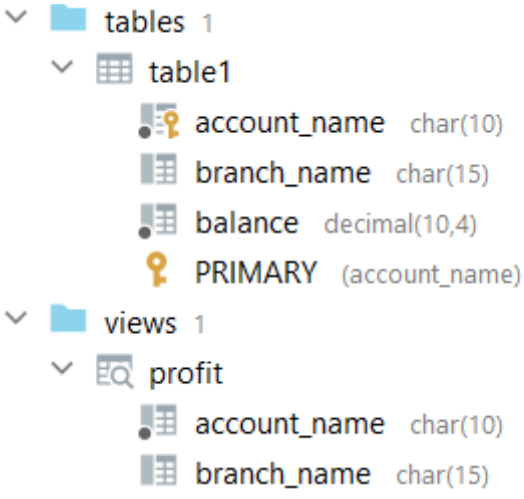
```
drop index branch_name_index on table1;
drop index account_branch on table1;
```



可以看到索引已经删除

建立视图

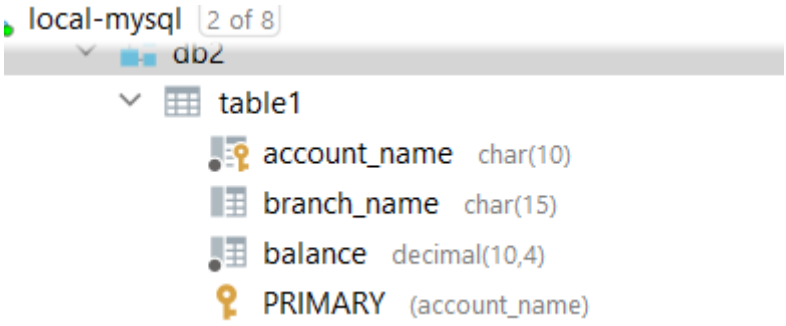
```
create view profit
as  select account_name,branch_name
from table1
where  balance>0;
```



可以看到已经生成了视图

删除视图

```
drop view profit;
```



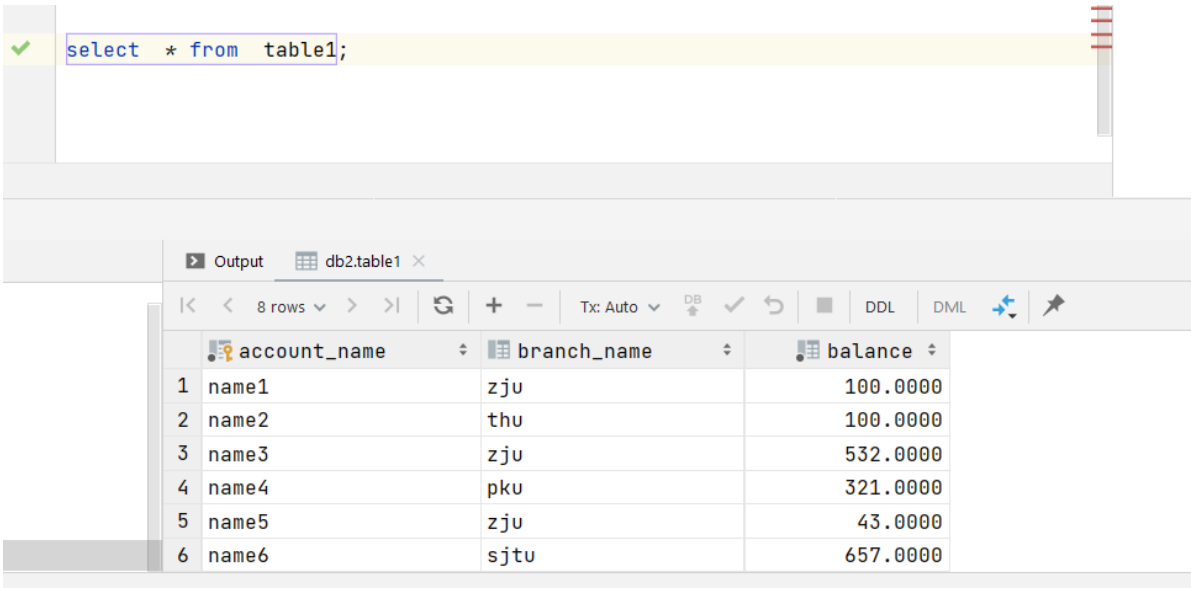
可以看到已经成功删除

插入数据

```
insert into table1 values ("name1","zju",100);
insert into table1 values ("name2","thu",100);
insert into table1 values ("name3","zju",532);
insert into table1 values ("name4","pku",321);
insert into table1 values ("name5","zju",43);
insert into table1 values ("name6","sjtu",657);
insert into table1 values ("name7","thu",234);
insert into table1 values ("name8","zju",134);
```

查看数据

```
select  * from  table1;
```



	account_name	branch_name	balance
1	name1	zju	100.0000
2	name2	thu	100.0000
3	name3	zju	532.0000
4	name4	pku	321.0000
5	name5	zju	43.0000
6	name6	sjtu	657.0000

更新数据

```
update table1
 set balance=balance+100
where branch_name='thu'
```

可以看到branch_name为**thu**的行都增加了100

	account_name	branch_name	balance
1	name1	zju	100.0000
2	name2	thu	200.0000
3	name3	zju	532.0000
4	name4	pku	321.0000
5	name5	zju	43.0000
6	name6	sjtu	657.0000
7	name7	thu	334.0000
8	name8	zju	134.0000

删除数据

```
delete from table1
where branch_name='sjtu'
```

	accour	Change page size	branch_name	balance
1	name1		zju	100.0000
2	name2		thu	200.0000
3	name3		zju	532.0000
4	name4		pku	321.0000
5	name5		zju	43.0000
6	name7		thu	334.0000
7	name8		zju	134.0000

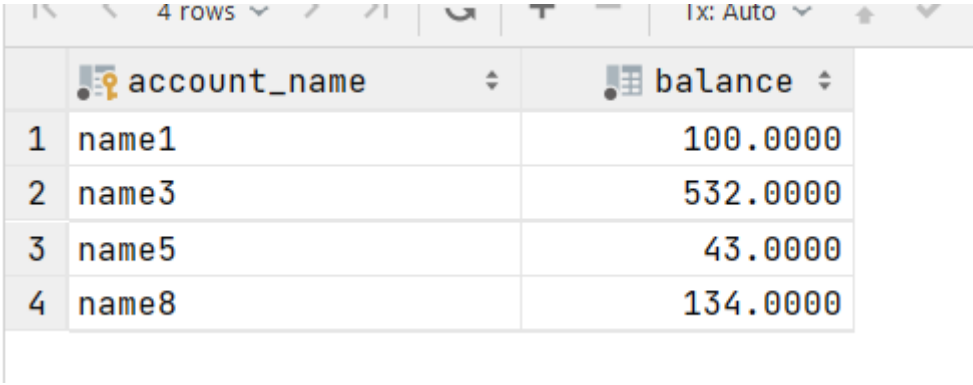
可以看到branch-name为**sjtu**的行已经被删除

查询语句

单表查询

```
select account_name,balance
from table1
where  branch_name='zju';
```

这里我们查询branch_name为zju的人的名字和收益

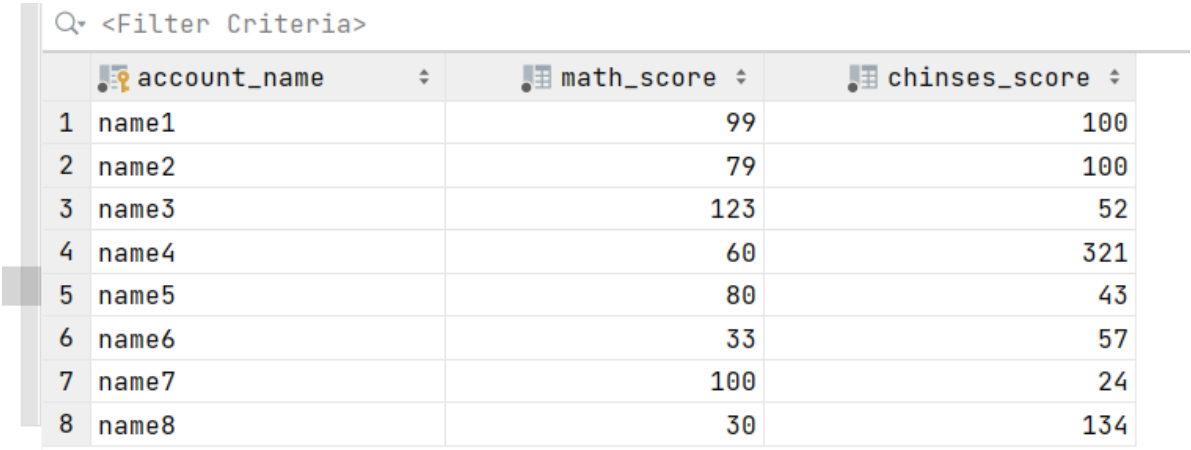


	account_name	balance
1	name1	100.0000
2	name3	532.0000
3	name5	43.0000
4	name8	134.0000

通过和上表的比对可以得出此次查询正确

多表查询

```
create table stu_score
(
   account_name char(10) not null,
   math_score int not null,
   chinses_score int not null,
   constraint stu_score_pk
      primary key (account_name)
);
insert into stu_score values ("name1",99,100);
insert into stu_score values ("name2",79,100);
insert into stu_score values ("name3",123,52);
insert into stu_score values ("name4",60,321);
insert into stu_score values ("name5",80,43);
insert into stu_score values ("name6",33,57);
insert into stu_score values ("name7",100,24);
insert into stu_score values ("name8",30,134);

```



	account_name	math_score	chinses_score
1	name1	99	100
2	name2	79	100
3	name3	123	52
4	name4	60	321
5	name5	80	43
6	name6	33	57
7	name7	100	24
8	name8	30	134

这里我们又定义了一张表，其中的数据结构如图所示

之后我们进行多表查询

```
select  table1.account_name,math_score,chinses_score,balance
from stu_score,table1
where stu_score.account_name=table1.account_name
```

结果如下

	account_name	math_score	chinses_score	balance
1	name1	99	100	100.0000
2	name2	79	100	200.0000
3	name3	123	52	532.0000
4	name4	60	321	321.0000
5	name5	80	43	43.0000
6	name7	100	24	334.0000
7	name8	30	134	134.0000

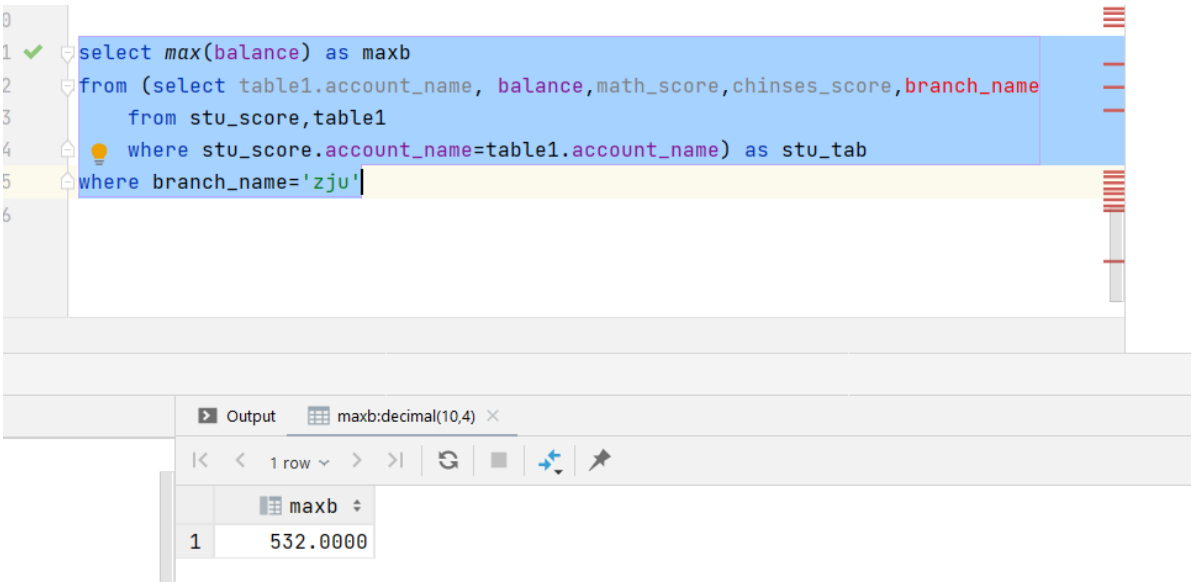
可以看到查询正确

嵌套子查询

```
select max(balance) as maxb
from (select table1.account_name,
balance,math_score,chinses_score,branch_name
    from stu_score,table1
    where stu_score.account_name=table1.account_name) as stu_tab
where branch_name='zju'
```

这里我们查询branch_name为zju的最大的收益

可以看到结果如下所示



证明我们查询正确

视图操作

```
select account_name
from db2.profit;
```

这里我们查询profit视图下的所有账户名字

```
select account_name
from db2.profit;
```

Output db2.profit

7 rows Tx: Auto

	account_name
1	name1
2	name2
3	name3
4	name4
5	name5
6	name7
7	name8

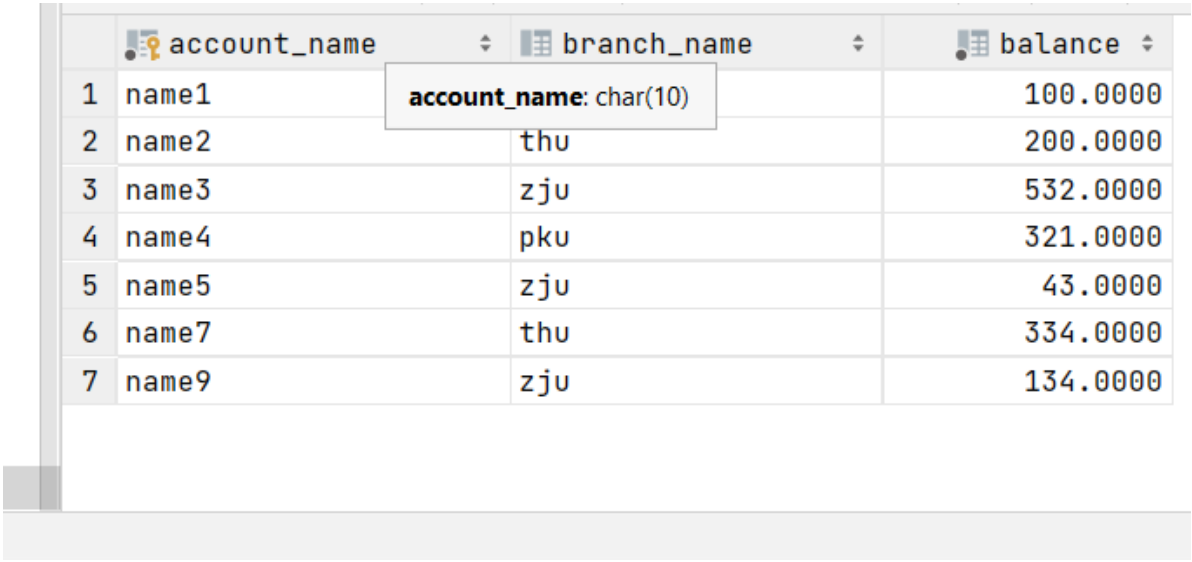
可以看到满足我们的要求

视图数据修改

```
update db2.profit
set account_name='name9'
where account_name='name8'
```

这里我们把视图中名字为name8的改名为name9

	account_name	branch_name
1	name1	zju
2	name2	thu
3	name3	zju
4	name4	pku
5	name5	zju
6	name7	thu
7	name9	zju



	account_name	branch_name	balance
1	name1		100.0000
2	name2	thu	200.0000
3	name3	zju	532.0000
4	name4	pku	321.0000
5	name5	zju	43.0000
6	name7	thu	334.0000
7	name9	zju	134.0000

可以看到数据表和视图都进行了更改

以上就是本次实验用到的数据库操作sql语句