实验3 SQL数据完整性

3200105787 张云策

一、实验目的

1. SQL数据完整性

实验内容和要求：

1. 在实验过程中，截图所用命令和实验结果
2. 定义若干表，其中包括primary key, foreign key 和check的定义。
3. 让表中插入数据，考察primary key如何控制实体完整性。
4. 删除被引用表中的行，考察foreign key 中on delete 子句如何控制参照完整性。
5. 修改被引用表中的行的primary key，考察foreign key 中on update 子句如何控制参照完整性。
6. 修改或插入表中数据，考察check子句如何控制校验完整性。
7. 定义一个asseration,并通过修改表中数据考察断言如何控制数据完整性。
8. 定义一个trigger,并通过修改表中数据考察触发器如何起作用。
9. 完成实验报告并提交至学在浙大平台。

二、实验环境

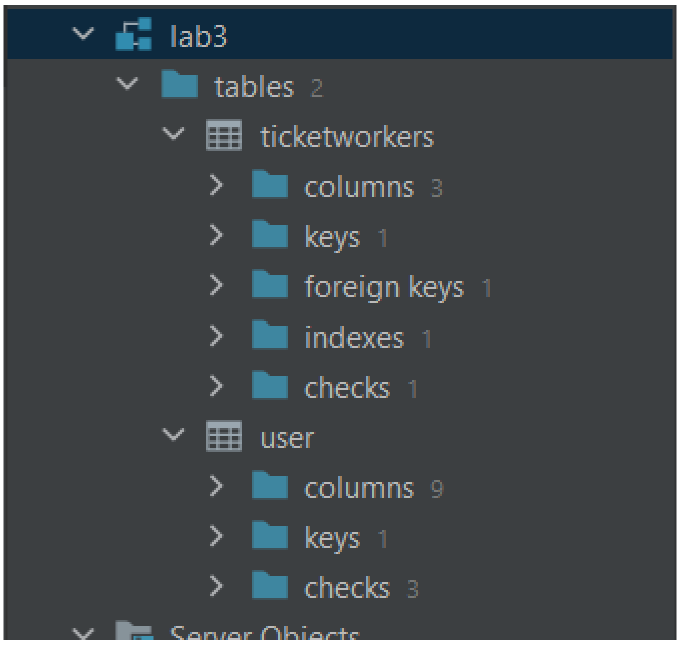
数据库管理系统： MySQL

三、实验流程

- **定义若干表，其中包括primary key, foreign key 和check的定义。**

```sql
create database lab3;
use lab3;



create table Users
(
    Id                  int auto_increment,
    StudentId           bigint       not null,
    Name                longtext     null,
    Secret              longtext     null,
    Role                int unsigned not null,
    Department          int          not null,
    ComputerFixedCount  int          not null,
    ApplianceFixedCount int          not null,
    AvatarURL           longtext     null,
    primary key (Id),
    check(Role in (0,1)),
    check ( Department>=0 and   Department<6 ),
    check(ComputerFixedCount>0 and ApplianceFixedCount>0)
);
create table TicketWorkers
(
    Id      int auto_increment,
    TicketId int not null,
    WorkerId int not null,
     primary key (Id),
    constraint foreign key (WorkerId) references Users(Id)
            on delete cascade,
    check(WorkerId>0)
);

```

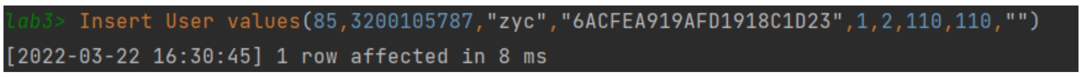
数据库如图所示



```
lab3
  tables 2
    ticketworkers
      columns 3
      keys 1
      foreign keys 1
      indexes 1
      checks 1
    user
      columns 9
      keys 1
      checks 3
Server Objects
```

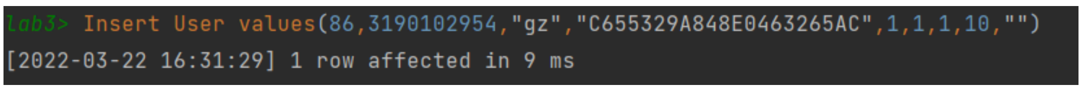
- **让表中插入数据，考察primary key如何控制实体完整性。**

```
Insert Users
values(85,3200105787,"zyc","6ACFEA919AFD1918C1D23",1,2,110,110,"");
```



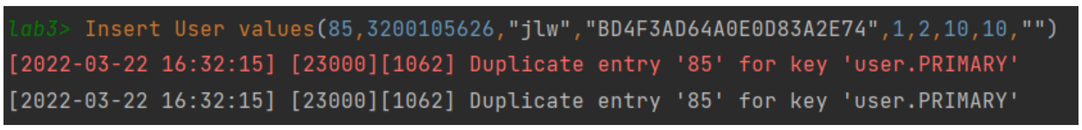
```
lab3> Insert User values(85,3200105787,"zyc","6ACFEA919AFD1918C1D23",1,2,110,110,"")
[2022-03-22 16:30:45] 1 row affected in 8 ms
```

```
Insert Users
values(86,3190102954,"gz","C655329A848E0463265AC",1,1,1,10,"");
```



```
lab3> Insert User values(86,3190102954,"gz","C655329A848E0463265AC",1,1,1,10,"")
[2022-03-22 16:31:29] 1 row affected in 9 ms
```

```
Insert Users
values(86,3190105626,"jlw","BD4F3AD64A0E0D83A2E74",1,2,10,10,"");

```



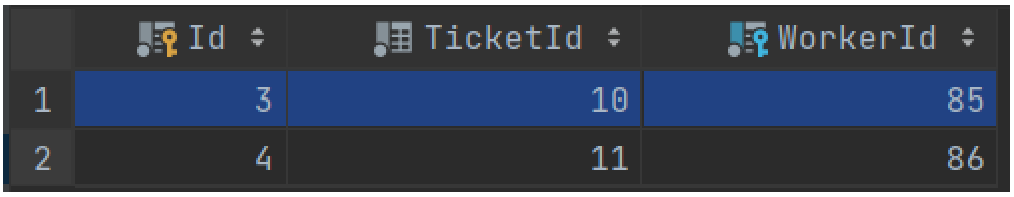
```
lab3> Insert User values(85,3200105626,"jlw","BD4F3AD64A0E0D83A2E74",1,2,10,10,"")
[2022-03-22 16:32:15] [23000][1062] Duplicate entry '85' for key 'user.PRIMARY'
[2022-03-22 16:32:15] [23000][1062] Duplicate entry '85' for key 'user.PRIMARY'
```

可以看到由于我们主键重复了，因此主键约束起了作用，我们无法插入两条Id都是86的数据

- **删除被引用表中的行，考察foreign key 中on delete 子句如何控制参照完整性。**

  我们首先插入一些数据到ticketworkers中

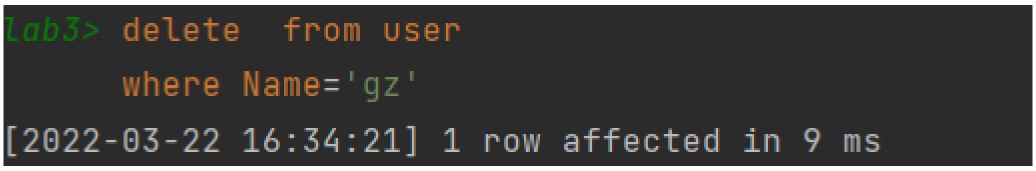
```
INSERT ticketworkers (TicketId, WorkerId) values (10,85);
INSERT ticketworkers (TicketId, WorkerId) values (11,86);
```



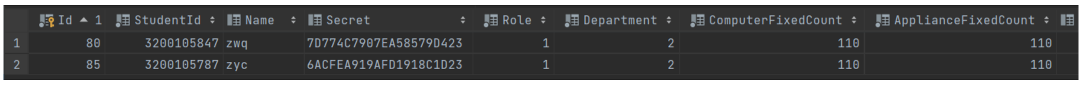
	Id	TicketId	WorkerId
1	3	10	85
2	4	11	86

之后我们删除

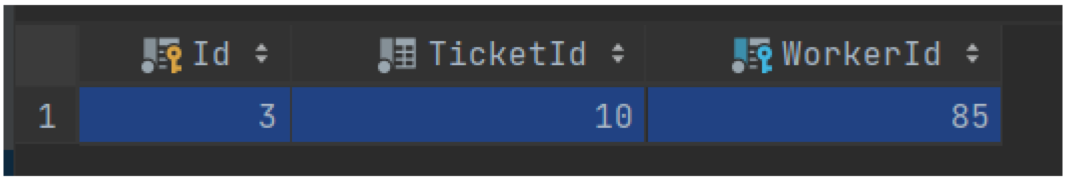
```
delete  from users
where Name='gz';
```



```
lab3> delete  from user
      where Name='gz'
[2022-03-22 16:34:21] 1 row affected in 9 ms
```



	Id ▲ 1	StudentId	Name	Secret	Role	Department	ComputerFixedCount	ApplianceFixedCount
1	80	3200105847	zwq	7D774C7907EA58579D423	1	2	110	110
2	85	3200105787	zyc	6ACFEA919AFD1918C1D23	1	2	110	110



	Id	TicketId	WorkerId
1	3	10	85

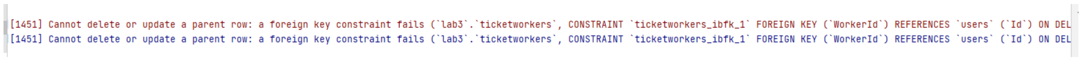
可以看到我们同时删除了两者中的id为86的记录，因为我们在建表的时候使用的

```
 on delete cascade
```

这说明在遇到引用问题时，删除时同时会删除引用的记录，上述实验也验证了这一点。

- **修改被引用表中的行的primary key，考察foreign key 中on update 子句如何控制参照完整性。**

```
update users
set Id=43232
where Name='zwq';
```



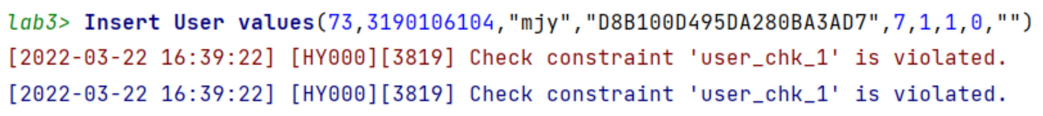
```
[1451] Cannot delete or update a parent row: a foreign key constraint fails (`lab3`.`ticketworkers`, CONSTRAINT `ticketworkers_ibfk_1` FOREIGN KEY (`WorkerId`) REFERENCES `users` (`Id`) ON DEL
[1451] Cannot delete or update a parent row: a foreign key constraint fails (`lab3`.`ticketworkers`, CONSTRAINT `ticketworkers_ibfk_1` FOREIGN KEY (`WorkerId`) REFERENCES `users` (`Id`) ON DEL
```

可以看到我们使用默认的on update阻止了本次update操作，这也验证了实验的正确。

(中途切换了datagrip主题为“Windows 10 UI”，所以颜色会有所不同)

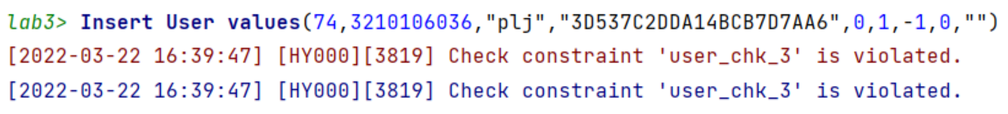
- **修改或插入表中数据，考察check子句如何控制校验完整性。**

```
Insert Users values(73,3190106104,"mjy","D8B100D495DA280BA3AD7",7,1,1,0,"");

```



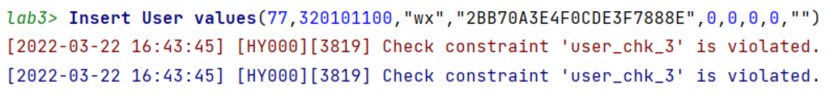
```
lab3> Insert User values(73,3190106104,"mjy","D8B100D495DA280BA3AD7",7,1,1,0,"")
[2022-03-22 16:39:22] [HY000][3819] Check constraint 'user_chk_1' is violated.
[2022-03-22 16:39:22] [HY000][3819] Check constraint 'user_chk_1' is violated.
```

```
Insert Users values(74,3210106036,"plj","3D537C2DDA14BCB7D7AA6",0,1,-1,0,"");

```



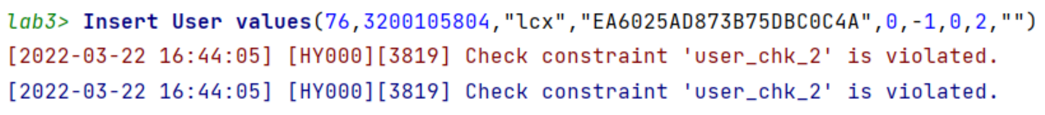
```
lab3> Insert User values(74,3210106036,"plj","3D537C2DDA14BCB7D7AA6",0,1,-1,0,"")
[2022-03-22 16:39:47] [HY000][3819] Check constraint 'user_chk_3' is violated.
[2022-03-22 16:39:47] [HY000][3819] Check constraint 'user_chk_3' is violated.
```

```
Insert Users values(77,320101100,"wx","2BB70A3E4F0CDE3F7888E",0,0,0,0,"");

```



```
lab3> Insert User values(77,320101100,"wx","2BB70A3E4F0CDE3F7888E",0,0,0,0,"")
[2022-03-22 16:43:45] [HY000][3819] Check constraint 'user_chk_3' is violated.
[2022-03-22 16:43:45] [HY000][3819] Check constraint 'user_chk_3' is violated.
```

```
Insert Users values(76,3190105804,"lcx","EA6025AD873B75DBC0C4A",0,-1,0,2,"");

```



```
lab3> Insert User values(76,3200105804,"lcx","EA6025AD873B75DBC0C4A",0,-1,0,2,"")
[2022-03-22 16:44:05] [HY000][3819] Check constraint 'user_chk_2' is violated.
[2022-03-22 16:44:05] [HY000][3819] Check constraint 'user_chk_2' is violated.
```

可以看到我们自定义的三条规则都被验证，和助教提供的实验三举例略有冲突，可能是版本或者datagrip的问题。

- **定义一个asseration，并通过修改表中数据考察断言控制数据完整性。**

```
create assertion fixed_range
check(
    not exists (
    select * from users
    Where ApplianceFixedCount>200
    )
    );
```

很神奇的是，datagrip没有assertion这个关键词，因此不用看根本执行不了。。

看了看助教的实验举例，可能mysql确实不支持这个关键词。

- **定义一个trigger, 并通过修改表中数据考察触发器如何起作用**

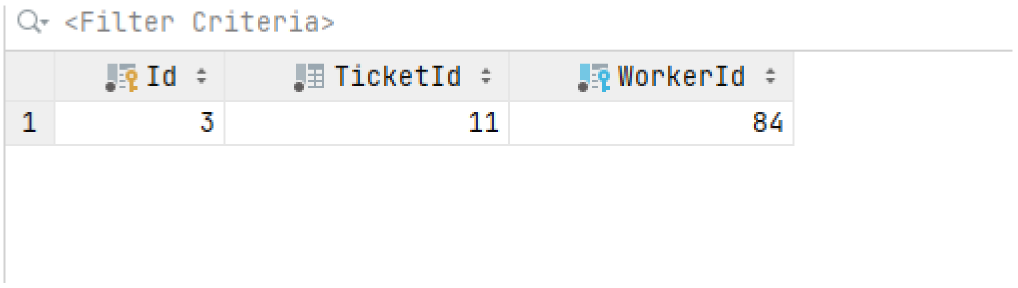
```
Delimiter $$
Create trigger Add_Count
    after update on users
    For each row
begin
    UPDATE  ticketworkers set TicketId=TicketId+1;
end; $$
Delimiter ;
```

定义触发器语句的前后加上Delimiter的目的，网上的说明是: This is so you can write “;” in your trigger definition without the MySQL client misinterpreting that as meaning you're done with it.

之后我们

```
update users
set Name='chy66'
where Name='chy';

```

可以看到



	Id	TicketId	WorkerId
1	3	11	84

TicketId已经自加1成功，证明我们触发器设计正确。

四、总结

在本次报告中，我亲自实践了各种完整性约束的实践操作，也将理论中的各种用法在mysql上亲自实现出来，在其中遇到了一些问题也能自己通过搜索引擎解决，切实的提高了自己对sql以及DBMS的理解和应用水平。