浙江大学

数据库系统实验报告

作业名称：	SQL 数据完整性
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 20 日

实验名称

一、实验目的

熟悉通过 SQL 进行数据完整性控制的方法。

二、实验环境

操作系统：Windows 10

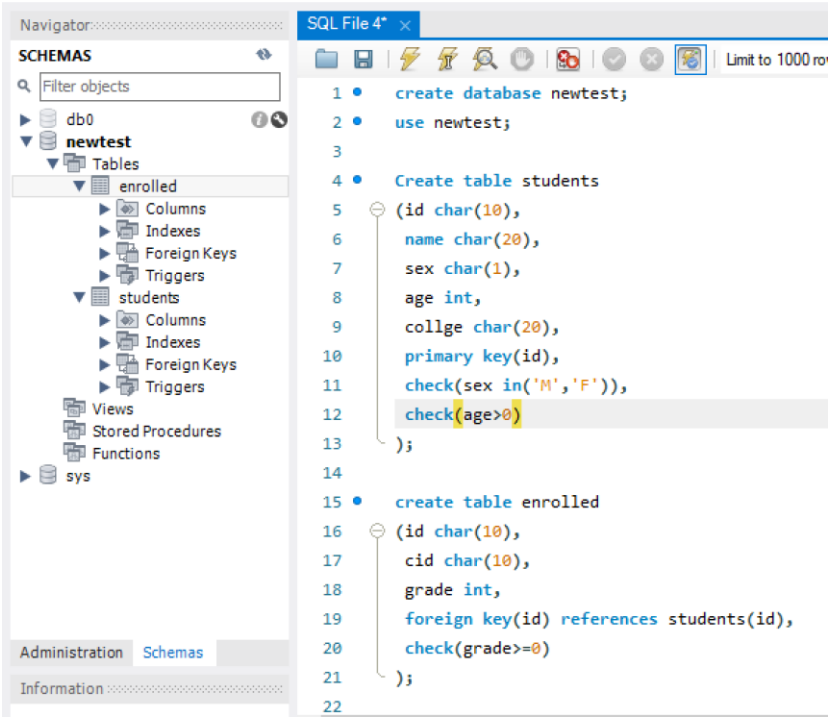
数据库管理系统：MySQL 8.0.32.0

工具：MySQL Workbench

三、实验流程

1. 定义若干表，其中包括 primary key, foreign key 和 check 的定义。

按照上周所学，定义 students, enrolled 两个表，并加上主键、外键和 check 的约束条件。



定义后，刷新左侧导航栏可以看到我们已经成功定义 students 和 enrolled 两个表。

其中，students 表包含五个属性，分别是学生 ID、姓名、性别、年龄和学院名称。enrolled 表包括三个属性，ID、cid 和成绩，使用外键约束，要求 enrolled 中的学生 ID 一定要出现在 students 表中。

同时 check(检查约束)用来检查数据表中字段值有效性。我们这里要求学生的性别只能‘M’或者‘F’，而且年龄必须大于 0.

2. 让表中插入数据，考察 primary key 如何控制实体完整性。

```
23 •  use newtest;
24 •  Insert students values('3210106182','QJJ','M',19,'CS');
25 •  Insert students values('3210100001','ZR','M',19,'CS');
26 •  Insert students values('3210100002','DJL','M',19,'AI');
27 •  Insert students values('3210100003','YQZ','M',19,'IS');
28 •  Insert students values('3210100004','LXY','F',19,'Biology');
29 •  Insert students values('3210100005','QJJ','F',19,'CS'); #duplicate name
30 •  Insert students values('3210106182','ZJJ','F',20,'History'); #duplicate ID
```

按照上次实验所学，对 students 进行插入，其中第六条插入为重复名字下的插入，第七条插入为重复 ID 下的插入。根据主键约束，最后一条插入不能执行。报错信息如下：

9	15:41:16	Insert students values('3210106182','QJJ','M',19,'CS')	1 row(s) affected	0.000 sec
10	15:41:16	Insert students values('3210100001','ZR','M',19,'CS')	1 row(s) affected	0.000 sec
11	15:41:16	Insert students values('3210100002','DJL','M',19,'AI')	1 row(s) affected	0.000 sec
12	15:41:16	Insert students values('3210100003','YQZ','M',19,'IS')	1 row(s) affected	0.000 sec
13	15:41:16	Insert students values('3210100004','LXY','F',19,'Biology')	1 row(s) affected	0.000 sec
14	15:41:16	Insert students values('3210100005','QJJ','F',19,'CS')	1 row(s) affected	0.016 sec
15	15:41:16	Insert students values('3210106182','ZJJ','F',20,'History')	Error Code: 1062. Duplicate entry '3210106182' for key 'students.PRIMARY'	0.000 sec

这说明了 primary key 唯一标识了数据库中的每条信息。

3. 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。

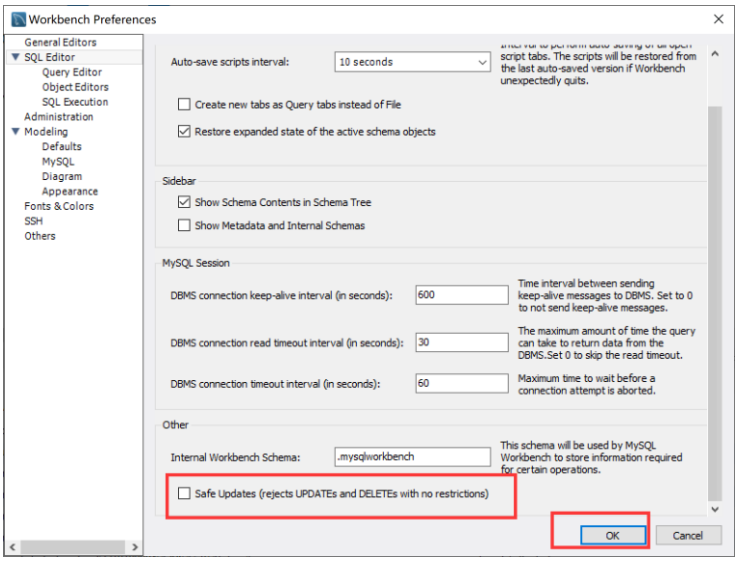
在表 enrolled 中插入一条引用表 students ID 为 3210103001 的记录,随后删除名为‘ZR’的记录，而他的 ID 恰好为 3210103001.

```
32 •  Insert enrolled values('3210100001','1',61); # ZR's ID
33 •  delete from students where name = 'ZR';
```

16	15:43:08	Insert enrolled values('3210100001','1',61)	1 row(s) affected	0.016 sec
17	15:43:08	delete from students where name = 'ZR'	Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE tha...	0.000 sec

这里报错：15:43:08 delete from students where name = ’ZR’ Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column. To disable safe mode, toggle the option in Preferences -> SQL Editor and reconnect. 0.000 sec

这是因为我们没有关闭 safe mode. 于是我们在左上角 Edit 打开 Preferences，点击 SQL Editor,将最下面 safe update 前的勾取消，随后点击 OK 重连数据库即可。



随后再次执行删除指令，得到报错：15:45:02 delete from students where name = 'ZR' Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`newtest`.`enrolled`, CONSTRAINT `enrolled_ibfk_1` FOREIGN KEY (`id`) REFERENCES `students` (`id`)) 0.000 sec

这体现了 foreign key 中 on delete 子句如何控制参照完整性。
除非先将表 enrolled 中引用‘3210103001’的记录删除，否则无法将表 students 中主键为‘3210103001’的记录删除。

4. 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。

执行下面语句，对 students 里 ‘ZR’ 的 ID 进行更新。

```
35 •    Update students
36      Set id= '3220100001'
37      Where name='ZR';     # update 'ZR' ID
```

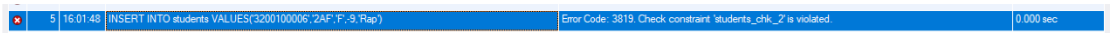
得到报错：15:47:33 Update students Set id= ’3220100001’ Where name=’ZR’ Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`newtest`.`enrolled`, CONSTRAINT `enrolled_ibfk_1` FOREIGN KEY (`id`) REFERENCES `students` (`id`)) 0.000 sec
这就是说明在外键约束下，表 students 中学生 ZR 的学号不能修改。

5. 修改或插入表中数据，考察 check 子句如何控制校验完整性。

执行下面语句，尝试往 students 里插入这样的数据:它的年龄是负数，这与我们最开始定义的 check 要求相悖。

```
39 •    INSERT INTO students VALUES('3200100006','2AF','F',-9,'Rap');
```

因此我们得到了如下报错：



表明这个插入破坏了我们之前的约束条件，故不能成功插入。

6. 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。

例如，我们设置一个断言，确保学生的年龄不大于 50 岁。

```
Create assertion age_range
check
(not exists (select * from students
Where age>50));
```

但是在 MySQL 中没有 ASSERTION 这一关键字。实现涉及一个或多个表或聚集操作的断言，应该使用触发器。

7. 定义一个 trigger,并通过修改表中数据考察触发器如何起作用。

我们通过下面的代码使用触发器，这个触发器执行的操作是，当 students 的任一行更新后，年龄低于 15 的同学成绩都会更新为 100。

```
46      Delimiter $$
47      Create trigger age_present
48      After update on students
49      For each row -- 这句话在MySQL中是固定的
50      Begin
51          Update enrolled set grade =100
52      where enrolled.id in (select id from students where age<15);
53      end; $$
54      Delimiter ;
```

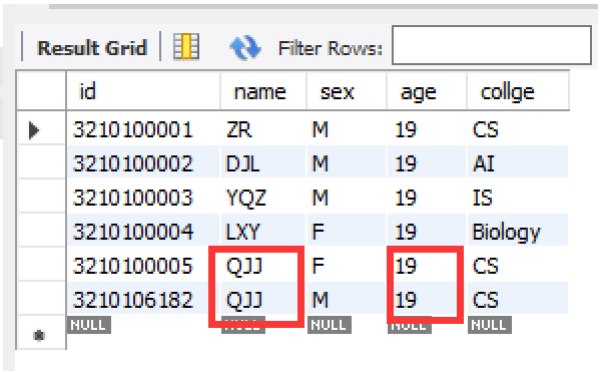
于是在定义触发器后，我们执行下面语句

Update students set age=14

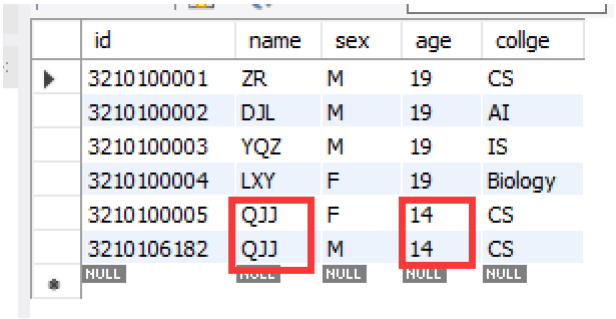
Where name='QJJ';

修改 'QJJ' 的年龄，改为 14 岁，随后观察触发器对两个表的数据的影响。

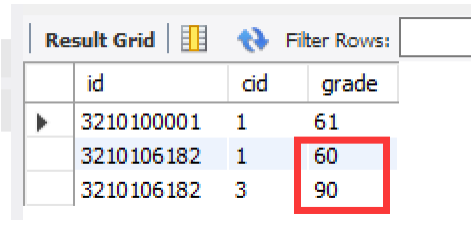
（左为修改年龄前，右为修改年龄后，第一行为 students 表，第二行为 enrolled 表）



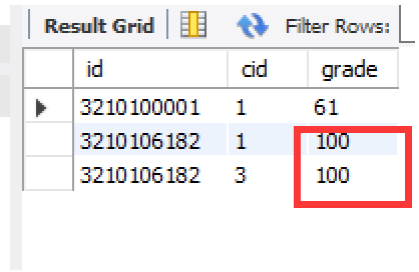
id	name	sex	age	collge
3210100001	ZR	M	19	CS
3210100002	DJL	M	19	AI
3210100003	YQZ	M	19	IS
3210100004	LXY	F	19	Biology
3210100005	QJJ	F	19	CS
3210106182	QJJ	M	19	CS



id	name	sex	age	collge
3210100001	ZR	M	19	CS
3210100002	DJL	M	19	AI
3210100003	YQZ	M	19	IS
3210100004	LXY	F	19	Biology
3210100005	QJJ	F	14	CS
3210106182	QJJ	M	14	CS



id	cid	grade
3210100001	1	61
3210106182	1	60
3210106182	3	90



id	cid	grade
3210100001	1	61
3210106182	1	100
3210106182	3	100

可以看到，我们对 QJJ 的年龄进行了修改，同时其对应 ID 在 enrolled 里的数据也因为年龄变为了 14 岁，满足触发器条件（<15）故 grade 变为 '100'，触发器功能正确。

四、遇到的问题及解决方法

本次实验较为顺利，未遇到问题。

五、总结

本次试验熟悉了通过 SQL 进行数据完整性控制的方法。包括 primary key,foreign key 和 check 的定义；让表中插入数据，考察 primary key 如何控制实体完整性；删除，修改被引用表中的行，考察 foreign key 如何控制参照完整性；改或插入表中数据，考察 check 子句如何控制校验完整性；定义一个 trigger,并通过修改表中数据考察触发器如何起作用等。