

实验3 SQL 数据完整性

学号：3200102324

姓名：王晨雨

实验目的：

1. 掌握 MySQL 数据库的完整性约束的定义方法；
2. 掌握 MySQL 数据库的用户的创建方法。
3. 掌握 MySQL 数据库的权限设置方法。

实验平台：

1. 数据库管理系统：Mysql

实验内容和要求：

1. 定义三张关系表如下：

S (SID,SNAME,AGE,SEX)

中文语义：学生（学号，姓名，年龄，性别）

SC (SID,CID,GRADE)

中文语义：学习（学号，课程号，成绩）

C (CID,CNAME,TEACHER)

中文语义：课程（课程号，课程名，任课教师）

```
create table S(  
SID varchar(20) primary key,  
SNAME varchar(20),  
AGE integer,  
SEX enum('m', 'f'),  
check(AGE > 0 and AGE < 200)  
);  
create table SC(  
SID varchar(20),  
CID varchar(20),  
GRADE integer,  
primary key (SID, CID),  
foreign key (SID) references S(SID),  
foreign key (CID) references C(CID)  
);  
create table C(  

```

```
CID varchar(20) primary key,
CNAME varchar(20),
TEACHER varchar(20)
);
```

2. 完整性约束设置:

- a) 为三张表创建主键约束:

```
primary key (SID, CID)
primary key (CID)
primary key (SID)
```

- b) 为 SC 表添加外键, 引用 S 的 SID 和 C 的 CID, 通过删除和修改 SC 表的数据测试 on delete 和 on update 的完整性;

```
foreign key (SID) references S(SID)
```

```
foreign key (CID) references C(CID)
```

先在 S 中插入三组数据, C 中插入三组数据

	SID	SNAME	AGE	SEX		CID	CNAME	TEACHER
▶	S1	Andy	20	m	▶	C1	Physics	Newton
	S2	Lucy	20	f		C2	Music	Morzar
	S3	Jack	20	m		C3	DataSet	Chenyue
*	NULL	NULL	NULL	NULL	*	NULL	NULL	NULL

考察 foreign key 中 on delete 子句如何控制参照完整性

首先在 SC 表中插入一条记录, 引用表 S 中的主键 SID, C 中的主键 CID

```
insert into SC values ("S1", "C1", 80)
```

	SID	CID	GRADE
▶	S1	C1	80
*	NULL	NULL	NULL

然后删除 SC 表中记录引用的 S 表中的学生信息

```
Delete from S where SID = 'S1'
```

报错信息如下:

```
Error Code: 1451. Cannot delete or update a parent row: a
foreign key constraint fails (`world`.`sc`, CONSTRAINT
`sc_ibfk_1` FOREIGN KEY (`SID`) REFERENCES `s` (`SID`))
```

这个报错信息才体现了 foreign key 中的 on delete 子句是如何控制参照完整性的: 除非我们先把 SC 中的记录删除, 否则无法直接删除 S 中被引用的记录。(删除表 C 中的信息也同理)

考察 foreign key 中 on update 子句如何控制参照完整性

更新 S 表中的信息:

```
update S set SID = 'S4' where SNAME = 'Andy'
```

报错信息如下:

```
Error Code: 1451. Cannot delete or update a parent row: a
foreign key constraint fails (`world`.`sc`, CONSTRAINT
`sc_ibfk_1` FOREIGN KEY (`SID`) REFERENCES `s` (`SID`))
```

这个报错信息才体现了 foreign key 中的 on update 子句是如何控制参照完整性的：除非我们先把 SC 中的记录删除，否则无法直接更新 S 中被引用的记录。（更新表 C 中的信息也同理）

- c) 通过 enum 保证 S 表中的 sex 只能是 f 或者 m，并测试：
清空 S 表

添加约束：

SEX enum('m', 'f')

测试：

插入数据：insert into S values ("S2", "Lucy", 20, "F")

此时 S 表中：

	SID	SNAME	AGE	SEX
▶	S2	Lucy	20	f
★	NULL	NULL	NULL	NULL

插入数据：insert into S values ("S1", "Andy", 20, "S")

此时 S 表中仍只有：

	SID	SNAME	AGE	SEX
▶	S2	Lucy	20	f
★	NULL	NULL	NULL	NULL

说明添加约束后，sex 只能为 f 或者 m。

- d) 通过 check 保证 S 表中的 age 不能小于 0，不能大于 200，并测试：

添加约束：check(AGE > 0 and AGE < 200)

测试：

我们尝试插入一个学生，年龄为负数。

insert S values ('S4', 'wcy', -7, 'F')

报错信息：

Error Code: 3819. Check constraint 's_chk_1' is violated.

违反了约束条件。

- e) 定义一个 trigger 实现 d) 中的约束条件，并测试
先把之前的 check 约束删除：DROP CONSTRAINT S_CHK_1 ;
定义一个 trigger：

```

Delimiter $$
CREATE TRIGGER trg_tb_student_insert_check BEFORE INSERT
ON S FOR EACH ROW
BEGIN
    DECLARE msg varchar(100);

    IF NEW.age <= 0 OR NEW.age >= 200
    THEN
        SET msg = CONCAT('您输入的年龄值：',NEW.age,' 为无效的年龄，请输入 0
到 100 以内的有效数字。');
        SIGNAL SQLSTATE 'HY000' SET MESSAGE_TEXT = msg;
    END IF;
END; $$

```

Delimiter ;

测试:

我们尝试插入一个学生, 年龄为负数。

```
insert S values ('S4', 'wcy', -7, 'F')
```

报错信息:

✖	102	20:02:52	insert S values ('S4', 'wcy', -7, 'F')	Error Code: 1644. 您输入的年龄值: -7 为无效的...	0.015 sec
---	-----	----------	--	---------------------------------------	-----------

违反了 trigger 中定义的约束条件。

3. MySQL 上的用户和权限的设置:

- a) 以 root 身份登录 MySQL, 创建两个用户 test1 和 test2; 并将 C 表的 UPDATE 和 INSERT 权限给 test1 用户, 并且保证他能够将这些权限授权给其他用户;

创建两个用户 test1 和 test2:

```
create user `test1`@`*` identified by '123456'
```

```
create user `test2`@`*` identified by '123456'
```

将 C 表的 UPDATE 和 INSERT 权限给 test1 用户, 并且保证他能够将这些权限授权给其他用户:

```
grant update, insert on world.C to `test1`@`*` WITH GRANT OPTION
```

Grants for test1@%	
▶	GRANT USAGE ON *.* TO `test1`@`%`
	GRANT INSERT, UPDATE ON `world`.`c` TO `t...

- b) 以 test1 用户登录 MySQL, 使用查询语句, 观察执行效果; 将 C 表的 UPDATE 权限给 test2 用户;

查询语句:

```
select * from C;
```

执行效果:

报错: Error Code: 1142. SELECT command denied to user 'test1'@'localhost' for table 'c'

将 C 表的 UPDATE 权限给 test2 用户:

```
GRANT UPDATE ON world.C TO 'test2'@`%` WITH GRANT OPTION;
```

- c) 以 root 身份登录 MySQL, 撤销 test1 用户在 C 表的 UPDATE 的权限, 观察 SHOW GRANTS 观察 test2 用户的权限变化情况。

撤销 test1 用户在 C 表的 UPDATE 的权限:

```
REVOKE UPDATE ON world.C FROM 'test1'@`%`;
```

观察 test1 用户的权限:

Grants for test1@%	
	GRANT INSERT ON `world`.`c` TO `test1`@`...
▶	GRANT USAGE ON *.* TO `test1`@`%`

观察 test2 用户的权限:

Grants for test2@%	
▶	GRANT USAGE ON *.* TO `test2`@`%`
	GRANT UPDATE ON `world`.`c` TO `test2`@`...

- d) 以 root 身份登录 MySQL，删除用户 test2，再注销以 test2 用户名登录 Mysql，观察效果。

删除用户 test2:

```
DROP USER 'test2'@'%';
```

注销 root，以 test2 进入数据库，此时无法登陆进入数据库。

