

- Report

- 操作过程

- 1.完整性约束

- (1). 创建PRIMARY KEY约束（主键约束），将Users表的“用户名”列（UserName）设置为主键
 - 请先删除原有主键，再重新新建主键，以上操作请通过SQL完成
 - (2). 创建FOREIGN KEY（外键约束），将Songs表的AlbumID列设置为外键FK_Songs_Album，该外键参照Album表中的主键AlbumID，且违约时采用“级联更新”和“级联删除”的策略。
 - 建立好外键约束后，请通过删除数据验证级联更新、级联删除等约束是否有效
 - (3). 创建UNIQUE约束（唯一性约束），为Songs表的“歌曲名”列（SongTitle）创建唯一性约束IX_SongTitle。
 - 请先删除原有约束，再建立约束
 - (4). 创建CHECK约束（检查约束），为Album表的“专辑语言”列（AlbumLanguage）创建一个检查约束CK_Language，使得“专辑语言”的取值范围为“汉语普通话、粤语、英语、日语、韩语、多国、其他”之一。
 - 在Mysql上实验一下到底有没有起作用。

- 2.触发器

- (1). 创建Before触发器
 - (2)创建after触发器

- 附完整代码

Report

操作过程

1.完整性约束

(1). 创建PRIMARY KEY约束（主键约束），将Users表的“用户名”列（UserName）设置为主键

请先删除原有主键，再重新新建主键，以上操作请通过SQL完成

```
alter table users
drop primary key ;

alter table users
add primary key (UserName);
```

(2). 创建**FOREIGN KEY**（外键约束），将**Songs**表的**AlbumID**列设置为外键**FK_Songs_Album**，该外键参照**Album**表中的主键**AlbumID**，且违约时采用“级联更新”和“级联删除”的策略。

建立好外键约束后，请通过删除数据验证级联更新、级联删除等约束是否有效

```
-- 创建外键约束
alter table songs
add foreign key FK_Songs_Album (AlbumID) references albums(AlbumID)
    ON DELETE RESTRICT
    ON UPDATE CASCADE ;
-- 删除Albums表中的一条数据
delete from albums where AlbumID = 1;
-- 更新Albums表中的一条数据
update albums set AlbumID = 2
where AlbumID = 1 ;
```

(3). 创建**UNIQUE**约束（唯一性约束），为**Songs**表的“歌曲名”列（**SongTitle**）创建唯一性约束**IX_SongTitle**。

请先删除原有约束，再建立约束

```
alter table songs
add constraint IX_SongTitle unique (SongTitle);
```

(4). 创建**CHECK**约束（检查约束），为**Album**表的“专辑语言”列（**AlbumLanguage**）创建一个检查约束**CK_Language**，使得“专辑语言”的取值范围为“汉语普通话、粤语、英语、日语、韩语、多国、其他”之一。

在**Mysql**上实验一下到底有没有起作用。

```
alter table albums
add constraint CK_Language check(AlbumLanguage in ('汉语普通话','粤语','英语','日语','韩语','多国','其他'));
```

```
update albums set AlbumLanguage = '法语'
where AlbumID = 1 ;
-- MySQL实际上并不支持CHECK约束，上述代码在MySQL中并不会产生预期的效果
```

• ORDER BY

	AlbumSinger	AlbumLanguage	AlbumMarketPrice	AL
Eye Fever_古巨基	古巨基	法语	100.00	

可见check约束并未起作用

2.触发器

(1). 创建Before触发器

```
-- 创建教室表
create table Teacher(
    tno int auto_increment primary key,
    t_name varchar(20),
    job varchar(10),
    salary double
);

delimiter &&
create trigger before_Sal_insert before insert on Teacher FOR EACH ROW
BEGIN
    if new.Job = '教授' and new.Salary < 4000
    THEN
        set new.Salary = 4000;
    end if;
END&&
delimiter ;

-- test
insert into teacher(t_name, job, salary) VALUES ('Test Professor', '教授', 3000);
```

console.sql	teacher [@localhost]	albums [@localhost]	report.md
<div> <div>1行</div> <div> <div>刷新</div> <div>回滚</div> <div>重做</div> <div>撤销</div> <div>重做</div> </div> <div>Tx: 自动</div> <div>DDL</div> <div>Q</div> <div>📄</div> </div>			
WHERE		ORDER BY	
tno	t_name	job	salary
1	Test Professor	教授	4000

```

delimiter $$
create trigger before_Sal_update before update on Teacher for each row
begin
    if new.job = 'joker' and new.salary > 0
    then
        set new.salary = 0;
    end if;
end $$
delimiter ;

-- test
update teacher
set job = 'joker'
where tno = 1;

```

WHERE					ORDER BY	
	tno	t_name	job	salary		
1	1	Test Professor	joker	0		

(2)创建after触发器

```

-- 创建表sal_log
create table sal_log(
    id int auto_increment primary key ,
    tno int ,
    salary double ,
    updateDate datetime,
    foreign key fk(tno) references teacher(tno)
);

delimiter &&
create trigger after_Sal_insert after insert on teacher for each row
begin
    insert into sal_log(tno, salary, updateDate) VALUES (new.tno, new.salary,
current_time);
end &&
delimiter ;

```

```

insert into teacher(t_name, job, salary) VALUES ('Test', '教授', 3000);

delimiter &&
create trigger after_Sal_update after update on teacher for each row
begin
    if (new.salary <> OLD.salary)
    then insert into sal_log(tno, salary, updateDate) VALUES (new.tno, new.salary,
current_time);
    end if;
end &&
delimiter ;

update teacher
set salary = 10000
where tno = 2;

```

两次执行结果如下：

<div> <div> <div>1行</div> <div> <div>刷新</div> <div>历史</div> <div>SQL</div> </div> <div> <div>+</div> <div>-</div> <div>↶</div> <div>↷</div> <div>↕</div> </div> <div> <div>Tx: 自动</div> <div>DDL</div> <div>🔍</div> <div>📄</div> </div> </div> </div>				
WHERE			ORDER BY	
id	tno	salary	updateDate	
1	1	2	4000	2023-12-08 09:28:26

WHERE			ORDER BY	
id	tno	salary	updateDate	
1	1	2	4000	2023-12-08 09:28:26
2	2	2	10000	2023-12-08 09:30:13

附完整代码

```

alter table users
drop primary key ;

alter table users
add primary key (UserName);

```

-- 创建FOREIGN KEY（外键约束），将Songs表的AlbumID列设置为外键FK_Songs_Album，该外键参照Album表中的主键AlbumID，且违约时采用“级联更新”和“级联删除”的策略。
-- 建立好外键约束后，请通过删除数据验证级联更新、级联删除等约束是否有效

```

-- 创建外键约束
alter table songs
add foreign key FK_Songs_Album (AlbumID) references albums(AlbumID)
    ON DELETE RESTRICT
    ON UPDATE CASCADE ;
-- 删除Albums表中的一条数据
delete from albums where AlbumID = 1;
-- 更新Albums表中的一条数据
update albums set AlbumID = 2
where AlbumID = 1 ;

alter table songs
add constraint IX_SongTitle unique (SongTitle);

alter table albums
add constraint CK_Language check(AlbumLanguage in ('汉语普通话','粤语','英语','日语','韩语','多国','其他'));

update albums set AlbumLanguage = '法语'
where AlbumID = 1 ;
-- MySQL实际上并不支持CHECK约束, 上述代码在MySQL中并不会产生预期的效果

-- 创建教室表
create table Teacher(
    tno int auto_increment primary key,
    t_name varchar(20),
    job varchar(10),
    salary double
);

delimiter &&
create trigger before_Sal_insert before insert on Teacher FOR EACH ROW
BEGIN
    if new.Job = '教授' and new.Salary < 4000
    THEN
        set new.Salary = 4000;
    end if;
END&&
delimiter ;

-- test
insert into teacher(t_name, job, salary) VALUES ('Test', '教授', 3000);

delimiter $$
create trigger before_Sal_update before update on Teacher for each row
begin
    if new.job = 'joker' and new.salary > 0
    then
        set new.salary = 0;
    end if;
end $$
delimiter ;

```

```

-- test
update teacher
set job = 'joker'
where tno = 1;

-- 创建表sal_log
create table sal_log(
    id int auto_increment primary key ,
    tno int ,
    salary double ,
    updateDate datetime,
    foreign key fk(tno) references teacher(tno)
);

delimiter &&
create trigger after_Sal_insert after insert on teacher for each row
begin
    insert into sal_log(tno, salary, updateDate) VALUES (new.tno, new.salary,
current_time);
end &&
delimiter ;

insert into teacher(t_name, job, salary) VALUES ('Test', '教授', 3000);

delimiter &&
create trigger after_Sal_update after update on teacher for each row
begin
    if (new.salary <> OLD.salary)
        then insert into sal_log(tno, salary, updateDate) VALUES (new.tno,
new.salary, current_time);
    end if;
end &&
delimiter ;

update teacher
set salary = 10000
where tno = 2;

```