

DBS-实验3 SQL数据完整性

ZHANG Yichi

实验目的:

1. 熟悉通过SQL进行数据完整性控制的方法。

实验平台:

1. 数据库管理系统: MySQL

实验内容和要求:

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

- 定义表时使用的sql语句如下

```
use test;
create table employee(
    employee_name varchar(20),
    street varchar(20),
    city varchar(20),
    primary key(employee_name)
);

create table company(
    company_name varchar(20),
    city varchar(20),
    primary key(company_name)
);

create table works(
    employee_name varchar(20),
    company_name varchar(20),
    salary double,
    primary key(employee_name),
    foreign key (employee_name)
        references employee(employee_name)
        on update cascade on delete restrict,
    foreign key (company_name)
        references company(company_name)
        on update cascade on delete restrict,
    check(salary>=2000)
);
```

2. 让表中插入数据, 考察primary key如何控制实体完整性。

```
insert into employee values('zhangyichi','ZJU','Hangzhou');
insert into employee values('zhangyichi','ZJU2','Hangzhou');
```

- 在进行上述插入操作时, 第一条插入语句成功执行, 第二条插入语句执行失败, 说明primary key控制了该数据表中这一字段的数据的值不能重复
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。

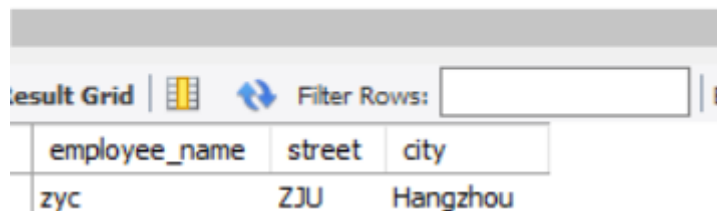
```
delete from employee
where employee_name='zhangyichi';
```

- 此时删除失败，显示Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (test.works, CONSTRAINT works_ibfk_1 FOREIGN KEY (employee_name) REFERENCES employee (employee_name) ON DELETE RESTRICT ON UPDATE CASCADE)

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。

- 对employee表中的内容进行修改，显示已经修改成功

```
1 • update employee
2   set employee_name='zyc'
3   where employee_name='zhangyichi';
4 • select * from employee;
```



employee_name	street	city
zyc	ZJU	Hangzhou

5. 修改或插入表中数据，考察check子句如何控制校验完整性。

- 发现插入时工资小于2000的时候不能正常执行insert语句，说明on delete语句起作用

```
insert into works values('zyc', 'ZJU', 1999); //插入失败
insert into works values('zyc', 'ZJU', 2000); //插入成功
```

6. 定义一个assertion, 并通过修改表中数据考察断言如何控制数据完整性。

- 插入如下所示的assertion，显示语法错误，经检查没有语法错误，后经查明，**Mysql不支持assertion操作**，相关的保留字assertion也没有形成语法高亮

```
create assertion test check(
  not exists(
    select * from works
    where salary < 2000
  )
);
```

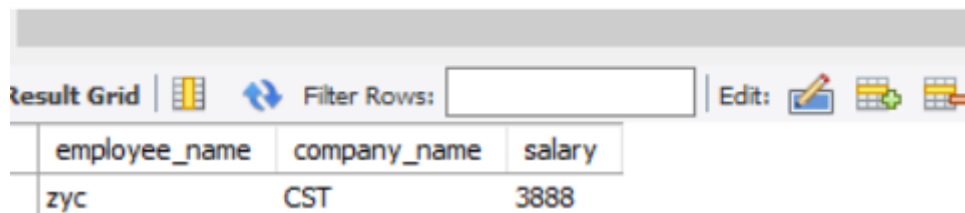
7. 定义一个trigger, 并通过修改表中数据考察触发器如何起作用。

- 创建一个如下的触发器，保证每个人工资高于2000元

```
use test;
create trigger salary_check
before update on works
for each row
set new.salary=new.salary+2000;
```

- 检验的结果如下，发现更新之后工资自动上涨2000元

```
1 • update works
2   set salary=1888
3   where employee_name='zyc';
4 • select * from works;
```



The screenshot shows a database interface with a 'Result Grid' tab. The grid contains a table with three columns: 'employee_name', 'company_name', and 'salary'. The first row of data shows 'zyc' for the employee name, 'CST' for the company name, and '3888' for the salary. The interface also includes a 'Filter Rows' search bar and an 'Edit' button with a pencil icon.

employee_name	company_name	salary
zyc	CST	3888

8. 完成实验报告。