外键约束与相关数据操作

一、完整的外键约束

1. 建立外键约束

```
CREATE TABLE 表名 (
    字段名1 类型(长度) 列级完整性约束,
    ...
    CONSTRAINT [外键名] FOREIGN KEY (外键字段) REFERENCES 主表(主键字段)
);

-- 追加外键约束
ALTER TABLE 表名
ADD CONSTRAINT [外键名]
FOREIGN KEY (外键字段) REFERENCES 主表(主键字段);
```

> 注意: (1)建立外键的表必须是 InnoDB 型的表，不能是临时表。因为 MySQL 中只有 InnoDB 型的表才支持外键。
> (2)定义外键名时，不能加引号。如: constraint 'FK_ID' 或 constraint "FK_ID" 都是错误的。

- 回顾：在定义表时创建外键约束: 创建 student 表，并构建与 grade 表之间的联系

```sql
CREATE DATABASE temp;
USE temp;

CREATE TABLE grade(
    id INT(4) PRIMARY KEY,
    gname VARCHAR(20) NOT NULL
);

CREATE TABLE student(
    sid INT(4) PRIMARY KEY,
    sname VARCHAR(36),
    gid INT(4) NOT NULL,
    FOREIGN KEY (gid) REFERENCES grade(id));
```

示例1: 为 `product` 表的 `sort_id` 追加外键约束，参照 `sort` 表的 `sort_id`

```sql
USE purchase;

-- 首先，被引用的主表字段应为主键，因此需为sort表的sort_id字段添加主键约束，否则建立外键时会提示1215错误。
ALTER TABLE sort
MODIFY sort_id CHAR(2) PRIMARY KEY;

-- 然后，为表product的Sort_ID字段添加外键约束
ALTER TABLE product
ADD CONSTRAINT fk_sortid FOREIGN KEY (sort_id) REFERENCES sort(sort_id);

-- 使用show create table 来查看表的构建语句
SHOW CREATE TABLE product;
```

建立外键的完整语法

```sql
ALTER TABLE 表名 ADD CONSTRAINT [外键名] FOREIGN KEY(外键字段名) REFERENCES 外表表名(主键字段名)
[ON DELETE {CASCADE | SET NULL | NO ACTION | RESTRICT}]
[ON UPDATE {CASCADE | SET NULL | NO ACTION | RESTRICT}]
```

> 注意:
>
> (1) `CASCADE`：主表中删除或更新对应的记录时，同时自动地删除或更新从表中匹配的记录。
> (2) `SET NULL`：主表中删除或更新被参照列记录时，同时将从表中的外键列设为空。
>
> (3) `RESTRICT`：拒绝删除或者更新主表被参照列记录（默认选项）。
>
> (4) `NO ACTION`：拒绝删除或者更新主表被参照列记录，标准SQL中的选项，在MYSQL中与RESTRICT等效。

2. 删除外键约束

基本语法

```
ALTER TABLE 表名
DROP FOREIGN KEY 约束名称;
```

示例2：删除product表的sort_id上的外键约束

```
ALTER TABLE product
DROP FOREIGN KEY fk_sortid;

SHOW CREATE TABLE product;
```

二、主从表中的数据插入、更新和删除

主从表之间建立外键之后，两表中的数据操作 `insert, update, delete` 会受哪些影响呢?

原则: 进行数据更新时，不违反外键约束，即更新数据过程中要保证从表的所有外键值包含在主表中或者为 `null` .

示例数据表：

- `product` 表的 `sort_id` 字段参照 `sort` 表的 `sort_id` , `subsort_id` 字段参照 `subsort` 表的 `subsort_id`
- `sort` 表的 `subsort_id` 字段参照 `subsort` 表的 `subsort_id`

1. 查询数据

可以通过从表的外键和主表的主键之间的参照关系完成跨表查询。

示例3：查询根类别为 `"办公机器设备"` 的产品

```
SELECT sort_id
FROM sort
WHERE sort_name='办公机器设备';

SELECT *
FROM product
WHERE sort_id=11;

--或者使用子查询
SELECT *
FROM product
WHERE sort_id IN(SELECT sort_id FROM sort WHERE sort_name='办公机器设备');

-- 或者使用变量保存结果
SELECT sort_id INTO @a FROM sort WHERE sort_name='办公机器设备';
SELECT * FROM product WHERE sort_id=@a;

-- 连接
SELECT p.*
FROM product p JOIN sort s ON p.sort_id = s.sort_id
```

```
21    WHERE s.sort_name = '办公机器设备';
```

2. 插入数据

往**主表**中插入数据不会影响到原表中的参照关系，因此所有主表中的插入操作并不会影响外键约束。往从表中插入数据时可能违背外键约束，如要插入的外键值并不存在于主表中。此时，有两种解决办法：

- 在主表中插入从表即将参照的外键值
- 关闭或者删除外键约束

示例4：在从表 `product` 中添加记录

```
1    -- 在从表中插入主表中没有的值99，提示1452错误，违反外键约束
2    INSERT INTO product(sort_id)
3    VALUES (99);
4
5    SELECT * FROM sort;
6    -- 为主表新添加sort_id为99的记录，然后再次往从表中插入该记录
7    INSERT INTO sort(sort_id, sort_name)
8    VALUES (99, '其它类别');
9
10   INSERT INTO product(sort_id)
11   VALUES (99);
12
13   SELECT *
14   FROM product
15   WHERE sort_id = '99';
```

3. 删除数据

从表中删除任意行并不会影响原表的参照关系，因此所有**从表的删除操作都不会违背外键约束**。

主表中数据行的主键值若被从表中的某些行参照，则将违背主键约束，从而删除失败。此时，有两种办法:

- 先删除从表中的所有相关的记录(即外键值为所需删除的主键值的记录行)
- 将从表中的所有相关的记录行中的外键值设为 `null`

以上过程可以手动完成，也可以通过在定义外键约束时的 `on delete` 选项完成。

示例5：用不同方式删除主表sort中的sort_id值为14，99和64的数据行

```
1    -- (1) 删除从表中的sort_id为14的对应记录
2    -- 直接删除，提示1451错误，因为product表中有sort_id参照sort中的sort_id
3    DELETE FROM sort
4    WHERE sort_id = 14;
5
6    SELECT *
7    FROM product
8    WHERE sort_id = 14;
9
10   -- 首先删除product表中有sort_id为14的记录
```

```sql
DELETE FROM product
WHERE sort_id = 14;
-- 然后，删除主表中的sort_id为14的记录
DELETE FROM sort
WHERE sort_id = 14;

-- (2) 设置从表中参照列对应值为null
SELECT *
FROM product
WHERE sort_id = 99;

UPDATE product
SET sort_id = NULL
WHERE sort_id = 99;

DELETE FROM sort
WHERE sort_id = 99;

-- (3) 重新设置外键的删除级联类型
ALTER TABLE product
DROP FOREIGN KEY fk_sortid1;

ALTER TABLE product
ADD CONSTRAINT fk_sortid1 FOREIGN KEY (sort_id) REFERENCES sort (sort_id) ON DELETE CASCADE;

select *
from product
where sort_id = 64;

DELETE FROM sort
WHERE sort_id = 64;

SELECT * FROM product WHERE sort_id = 64;
```

练习

在给定的 `product` 数据库中完成以下操作

- (1) 为 `product` 表添加 `Sort_ID` 的外键约束 `FK_sortid1`
- (2) 使用语句为 `subsort` 表的 `Sort_ID` 添加引用自 `sort` 表外键名为 `FK_sortid2` 的外键约束，并设置为 `on delete cascade` 。
- (3) 为 `product` 表添加 `Subsot_id` 的引用自 `subsort` 表的外键约束 `FK_subsortid` 。(注意：主表被引用的列应具有主键约束或唯一性约束。)
- (4) 删除 `product` 表中的 `Subsort_ID` 的外键约束。
- (5) 课堂练习1中已为 `subsort` 表与 `sort` 表添加关联关系（建立外键），修改该外键的参数说明均为 `RESTRICT` 。
- (6) 向 `subsort` 表添加如下表的记录。

subsort_id	sort_id
9701	97
9702	97
9801	98
9802	98
9901	99
9902	99

- (7) 按照示例5的三种方式逐次删除在 `sort` 表中的sort_id为 `97、98、99` 的记录。

附：如何将数据导入至已建立外键约束的多个表中?

```
create table department (id int primary key,
                         dept_name varchar(20));

create table employee (id int primary key,
                       emp_name varchar(20),
                       dept_id int,
                       foreign key f_dept_id (dept_id) references department(id));
```

- 方法1：明确表之间的参照关系，对于与其它表无参照关系的表可随时导入，对于参照了其它表的从表，则应先导入主表中的记录（注意，若从表中的外键列出现了主表被参照中没有的值，则外键约束不成功）。

```
insert into department(id, dept_name)
values (100, '计算机科学系'),
(200, '理论物理系');

insert into employee(id, emp_name, dept_id)
values (1, 'xiaoxia', 100), (2, 'tianxi', 200);
```

- 方法2：先删除外键约束，导入数据这时候重新创建创建（注意，若表中数据不满足外键约束的限定，则外键约束创建不成功）。

```
 delete from employee;
 delete from department;

 insert into employee(id, emp_name, dept_id)
 values (1, 'xiaoxia', 100), (2, 'tianxi', 200); -- 不成功，因为有department中没有相应的100, 200

 -- 删除外键约束
 alter table employee drop foreign key f_dept_id;

```

```
insert into employee(id, emp_name, dept_id)
values (1, 'xiaoxia', 100), (2, 'tianxi', 200); -- 成功执行

-- 添加外键约束
alter table employee
add foreign key f_dept_id (dept_id) references department(id); -- 失败

insert into department(id, dept_name)
values (100, '计算机科学系'),
(200, '理论物理系');

-- 添加外键约束
alter table employee
add foreign key f_dept_id (dept_id) references department(id);  -- 成功
```

- 方法3（慎用）：设置外键约束失效 `set foreign_key_checks=0`，导入数据之后再让其生效 `set foreign_key_checks=1` （注意，这种方式只对新导入的数据进行外键约束检查，即设置之前导入的从表的外键列的值可存在未出现在主表的值）

```
delete from employee;
delete from department;

insert into employee(id, emp_name, dept_id)
values (1, 'xiaoxia', 100),
(2, 'tianxi', 200); -- 不成功，因为department中没有相应的100，200

-- 禁止当前连接的外键约束检验
set foreign_key_checks=0;

insert into employee(id, emp_name, dept_id)
values (1, 'xiaoxia', 100),
(2, 'tianxi', 200); -- 成功执行，因为不执行外键约束检查

-- 启用当前连接的外键约束检验，
set foreign_key_checks=1;  -- 成功更新变量值， 不受之前插入的不满足外键约束的从表记录影响

insert into department(id, dept_name)
values (100, '计算机科学系'),
(200, '理论物理系');
```