mysql查询DQL&多表关系

- [ ] 能够使用SQL语句查询数据
- [ ] 能够使用SQL语句进行条件查询
- [ ] 能够使用SQL语句进行排序
- [ ] 能够使用聚合函数
- [ ] 能够使用SQL语句进行分组查询
- [ ] 能够完成数据的备份和恢复
- [ ] 能够使用可视化工具连接数据库,操作数据库
- [ ] 能够说出多表之间的关系及其建表原则
- [ ] 能够理解外键约束

第一章 SQL语句(DQL)

1.1 DQL准备工作和语法

准备工作

```
#创建商品表：
create table product(
    pid int primary key,
    pname varchar(20),
    price double,
    category_id varchar(32)
);
INSERT INTO product(pid,pname,price,category_id) VALUES(1,'联想',5000,'c001');
INSERT INTO product(pid,pname,price,category_id) VALUES(2,'海尔',3000,'c001');
INSERT INTO product(pid,pname,price,category_id) VALUES(3,'雷神',5000,'c001');
INSERT INTO product(pid,pname,price,category_id) VALUES(4,'JACK JONES',800,'c002');
INSERT INTO product(pid,pname,price,category_id) VALUES(5,'真维斯',200,'c002');
INSERT INTO product(pid,pname,price,category_id) VALUES(6,'花花公子',440,'c002');
INSERT INTO product(pid,pname,price,category_id) VALUES(7,'劲霸',2000,'c002');
INSERT INTO product(pid,pname,price,category_id) VALUES(8,'香奈儿',800,'c003');
INSERT INTO product(pid,pname,price,category_id) VALUES(9,'相宜本草',200,'c003');
INSERT INTO product(pid,pname,price,category_id) VALUES(10,'面霸',5,'c003');
INSERT INTO product(pid,pname,price,category_id) VALUES(11,'好想你枣',56,'c004');
INSERT INTO product(pid,pname,price,category_id) VALUES(12,'香飘飘奶茶',1,'c005');
INSERT INTO product(pid,pname,price,category_id) VALUES(13,'果9',1,NULL);
```

语法

```
select [distinct]
* | 列名,列名
from 表
where 条件
```

1.2 简单查询

- 练习

```
#查询所有的商品.
select * from product;
```

```
#查询商品名和商品价格.
select pname,price from product;
```

```
#别名查询.使用的关键字是as（as可以省略的）.表别名:
select * from product as p;
```

```
#别名查询.使用的关键字是as（as可以省略的）.列别名:
select pname as pn from product;
```

```
#去掉重复值.
select distinct price from product;
```

```
#查询结果是表达式（运算查询）：将所有商品的价格+10元进行显示.
select pname,price+10 from product;
```

1.3 条件查询

比较运算符	**< <= = = <>**	**大于、小于、大于(小于)等于、不等于**
	BETWEEN ...AND...	显示在某一区间的值(含头含尾)
	IN(set)	显示在in列表中的值，例：in(100,200)
	LIKE '张 pattern'	模糊查询，Like语句中，% 代表零个或多个任意字符，_ 代表一个字符， 例如：`first_name like '_a%';`
	IS NULL	判断是否为空
逻辑运行符	and	多个条件同时成立
	or	多个条件任一成立
	not	不成立，例：`where not(salary>100);`

- 练习

```
#查询商品名称为“花花公子”的商品所有信息：
SELECT * FROM product WHERE pname = '花花公子'
```

```
#查询价格为800商品
SELECT * FROM product WHERE price = 800
```

```
#查询价格不是800的所有商品
SELECT * FROM product WHERE price != 800
SELECT * FROM product WHERE price <> 800
SELECT * FROM product WHERE NOT(price = 800)
```

```
#查询商品价格大于60元的所有商品信息
SELECT * FROM product WHERE price > 60;
```

```
#查询商品价格在200到1000之间所有商品
SELECT * FROM product WHERE price >= 200 AND price <=1000;
SELECT * FROM product WHERE price BETWEEN 200 AND 1000;
```

```
#查询商品价格是200或800的所有商品
SELECT * FROM product WHERE price = 200 OR price = 800;
SELECT * FROM product WHERE price IN (200,800);
```

```
#查询含有'霸'字的所有商品
SELECT * FROM product WHERE pname LIKE '%霸%';

#查询以'香'开头的所有商品
SELECT * FROM product WHERE pname LIKE '香%';

#查询第二个字为'想'的所有商品
SELECT * FROM product WHERE pname LIKE '_想%';
```

```
#商品没有分类的商品
SELECT * FROM product WHERE category_id IS NULL

#查询有分类的商品
SELECT * FROM product WHERE category_id IS NOT NULL
```

1.4 排序查询

通过order by语句，可以将查询出的结果进行排序。暂时放置在select语句的最后。

- 格式:

```
SELECT * FROM 表名 ORDER BY 排序字段 ASC|DESC;
#ASC 升序（默认）
#DESC 降序
```

- 练习：

```
#使用价格排序(降序)
SELECT * FROM product ORDER BY price DESC;
```

```
#在价格排序(降序)的基础上，以分类排序(降序)
SELECT * FROM product ORDER BY price DESC,category_id DESC;
```

```
#显示商品的价格(去重复)，并排序(降序)
SELECT DISTINCT price FROM product ORDER BY price DESC;
```

1.5 聚合查询

之前我们做的查询都是横向查询，它们都是根据条件一行一行的进行判断，而使用聚合函数查询是纵向查询，它是对一列的值进行计算，然后返回一个单一的值；另外聚合函数会忽略空值。

今天我们学习如下五个聚合函数：

- count：统计指定列不为NULL的记录行数；
- sum：计算指定列的数值和，如果指定列类型不是数值类型，那么计算结果为0；

- max：计算指定列的最大值，如果指定列是字符串类型，那么使用字符串排序运算；
- min：计算指定列的最小值，如果指定列是字符串类型，那么使用字符串排序运算；
- avg：计算指定列的平均值，如果指定列类型不是数值类型，那么计算结果为0；

练习：

```
#查询商品的总条数
SELECT COUNT(*) FROM product;
```

```
#查询价格大于200商品的总条数
SELECT COUNT(*) FROM product WHERE price > 200;
```

```
#查询分类为'c001'的所有商品的总和
SELECT SUM(price) FROM product WHERE category_id = 'c001';
```

```
#查询分类为'c002'所有商品的平均价格
SELECT AVG(price) FROM product WHERE category_id = 'c002';
```

```
#查询商品的最大价格和最小价格
SELECT MAX(price),MIN(price) FROM product;
```

1.6 分组查询

分组查询是指使用group by字句对查询信息进行分组。

- 格式：

```
SELECT 字段1,字段2… FROM 表名 GROUP BY分组字段 HAVING 分组条件;
```

分组操作中的having子语句，是用于在分组后对数据进行过滤的，作用类似于where条件。

- having与where的区别:
  - having是在分组后对数据进行过滤.

    where是在分组前对数据进行过滤
  - having后面可以使用分组函数(统计函数)

    where后面不可以使用分组函数。

练习：

```
#统计各个分类商品的个数
SELECT category_id ,COUNT(*) FROM product GROUP BY category_id ;
```

```sql
#统计各个分类商品的个数,且只显示个数大于1的信息
SELECT category_id ,COUNT(*) FROM product GROUP BY category_id HAVING COUNT(*) > 1;
```

第二章 SQL备份与恢复

3.1 SQL备份

数据库的备份是指将数据库转换成对应的sql文件

MySQL命令备份

数据库导出sql脚本的格式：

```
mysqldump  -u用户名 -p密码 数据库名>生成的脚本文件路径
```

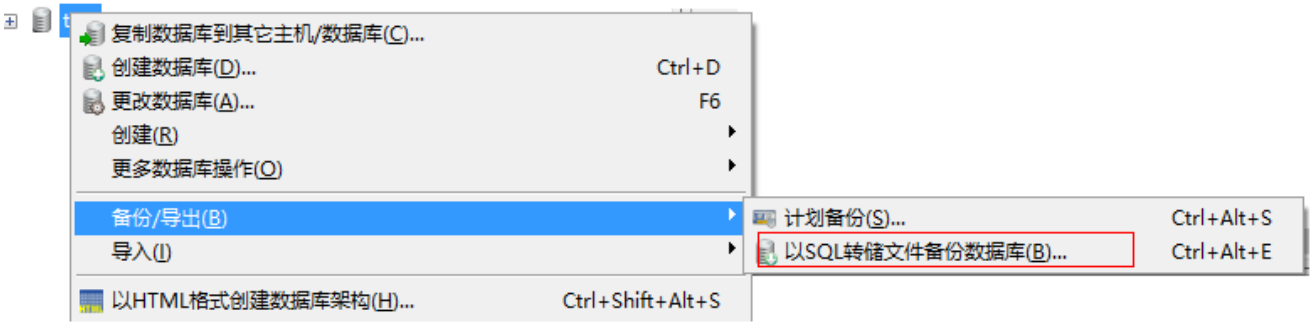
例如:

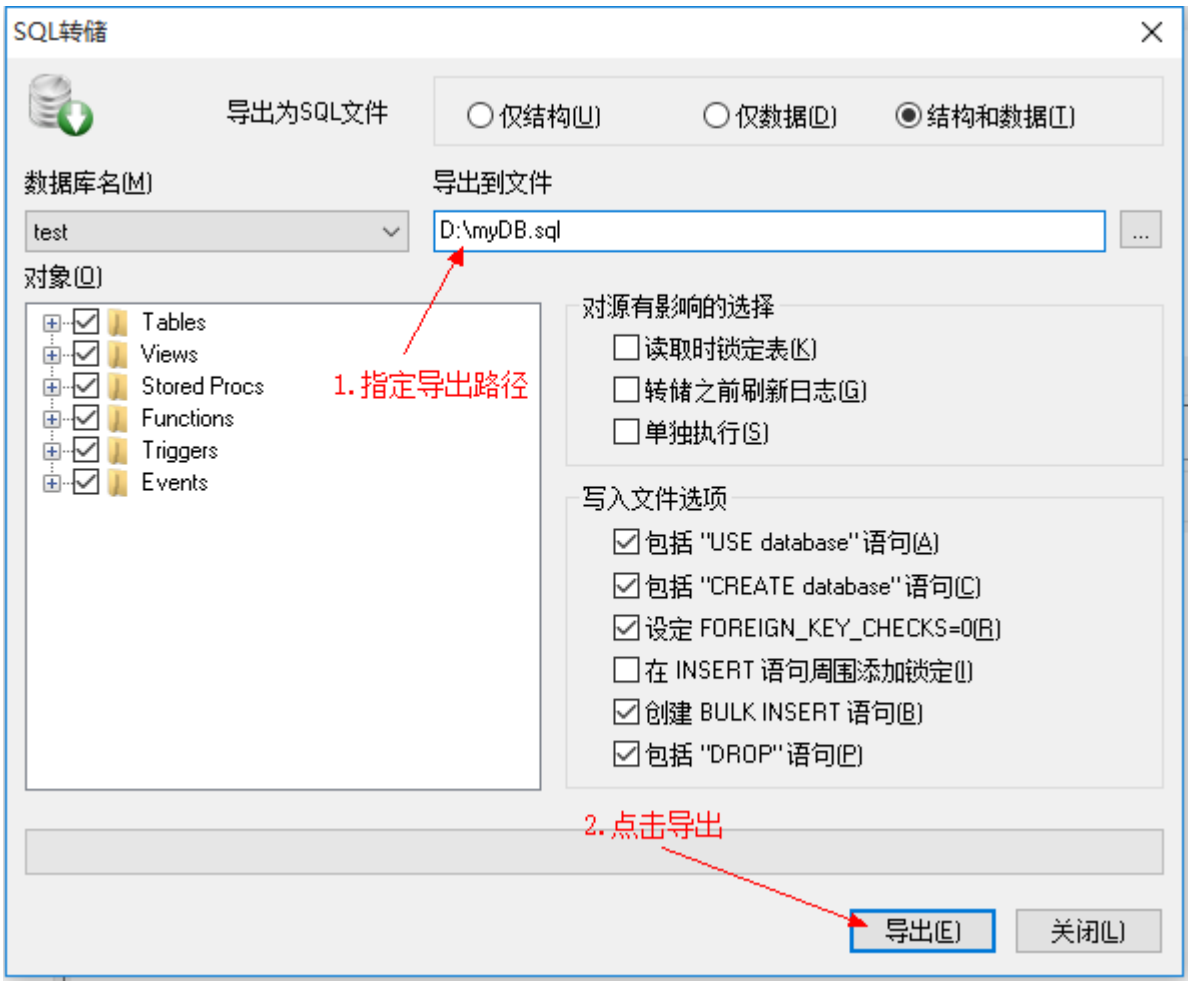
```
mysqldump  -uroot  -proot day04>d:\day03.sql
```

以上备份数据库的命令中需要用户名和密码，即表明该命令要在用户没有登录的情况下使用

可视化工具备份

选中数据库，右键 ”备份/导出” ， 指定导出路径，保存成.sql文件即可。





3.2 SQL恢复

数据库的恢复指的是使用备份产生的sql文件恢复数据库，即将sql文件中的sql语句执行就可以恢复数据库内容。

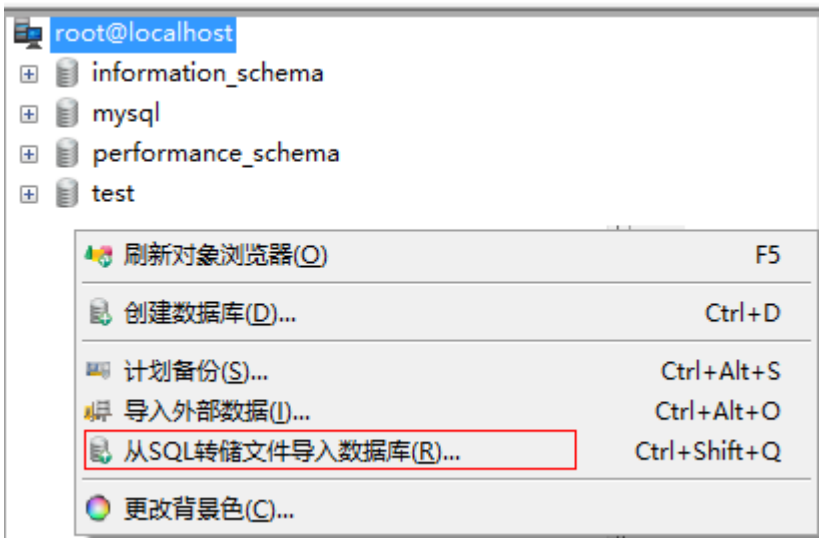
MySQL命令恢复

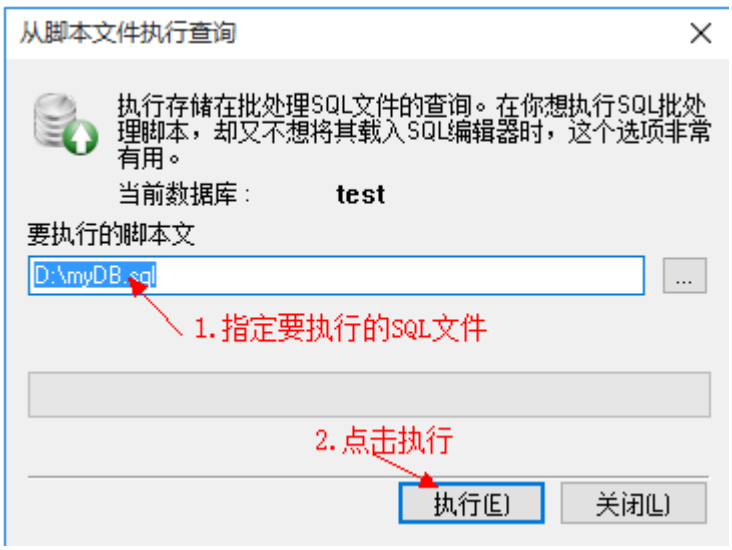
使用数据库命令备份的时候只是备份了数据库内容，产生的sql文件中没有创建数据库的sql语句，在恢复数据库之前需要自己动手创建数据库。

- 在数据库外恢复
  - **格式:** `mysql -uroot -p密码 数据库名 < 文件路径`
  - 例如: `mysql -uroot -proot day03<d:\day03.sql`
- 在数据库内恢复
  - **格式:** `source SQL脚本路径`
  - 例如: `source d:\day03.sql`
  - 注意:使用这种方式恢复数据，首先要登录数据库.

可视化工具恢复

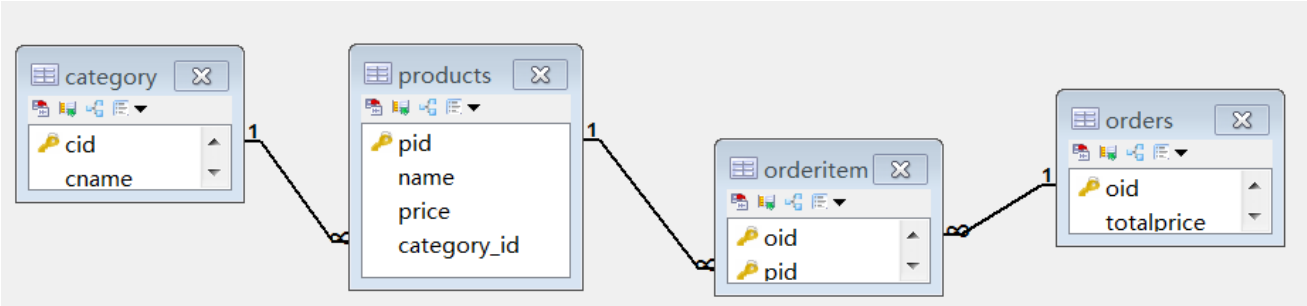
数据库列表区域右键"从SQL转储文件导入数据库"， 指定要执行的SQL文件，执行即可。





第三章 多表操作

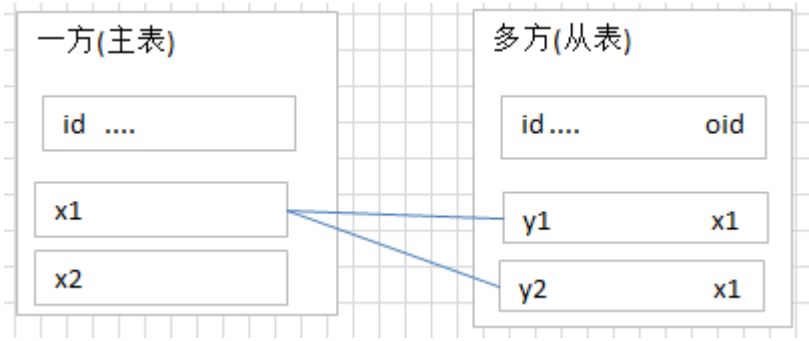
实际开发中，一个项目通常需要很多张表才能完成。例如：一个商城项目就需要分类表(category)、商品表(products)、订单表(orders)等多张表。且这些表的数据之间存在一定的关系，接下来我们将在单表的基础上，一起学习多表方面的知识。



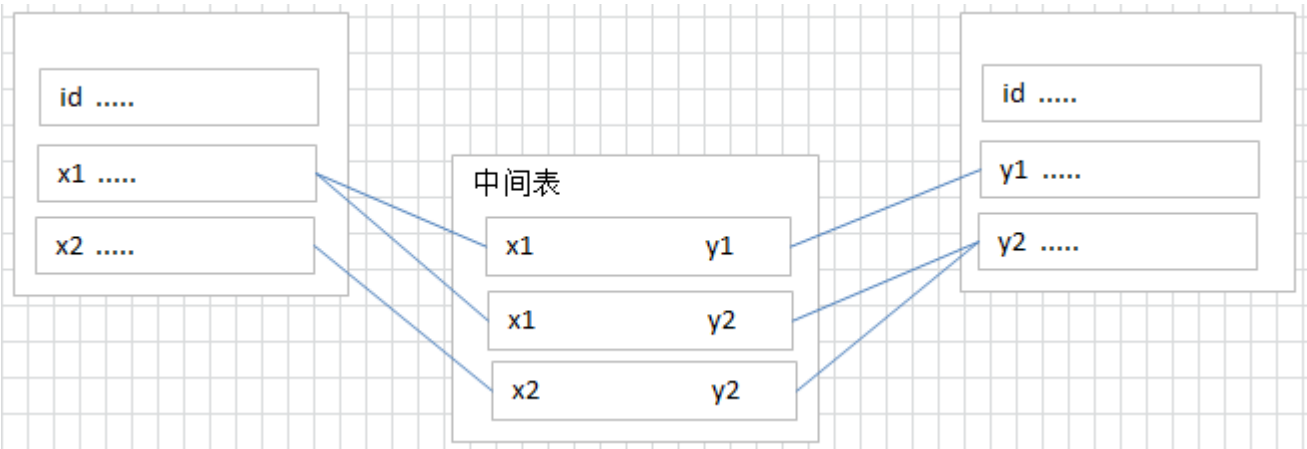
4.1 表与表之间的关系

- 一对多关系：
  - 常见实例：客户和订单，分类和商品，部门和员工.

- 一对多建表原则：在从表(多方)创建一个字段，字段作为外键指向主表(一方)的主键.



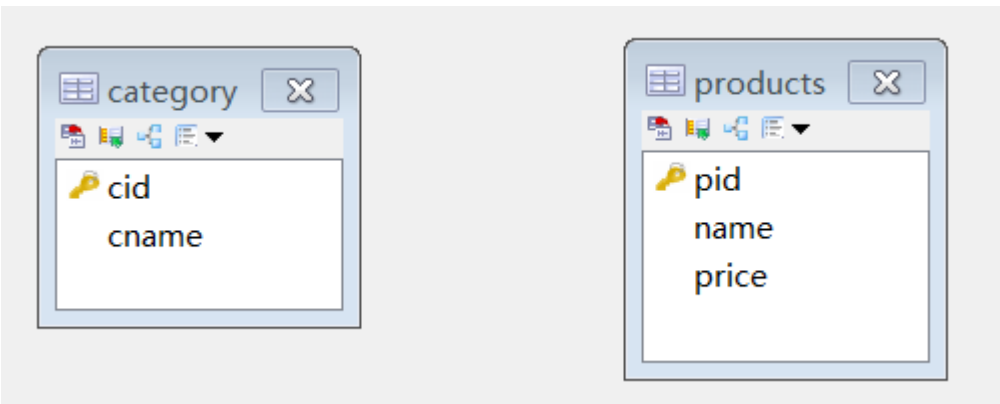
- 多对多关系：
  - 常见实例：学生和课程、用户和角色
  - 多对多关系建表原则：需要创建第三张表,中间表中至少两个字段，这两个字段分别作为外键指向各自一方的主键.

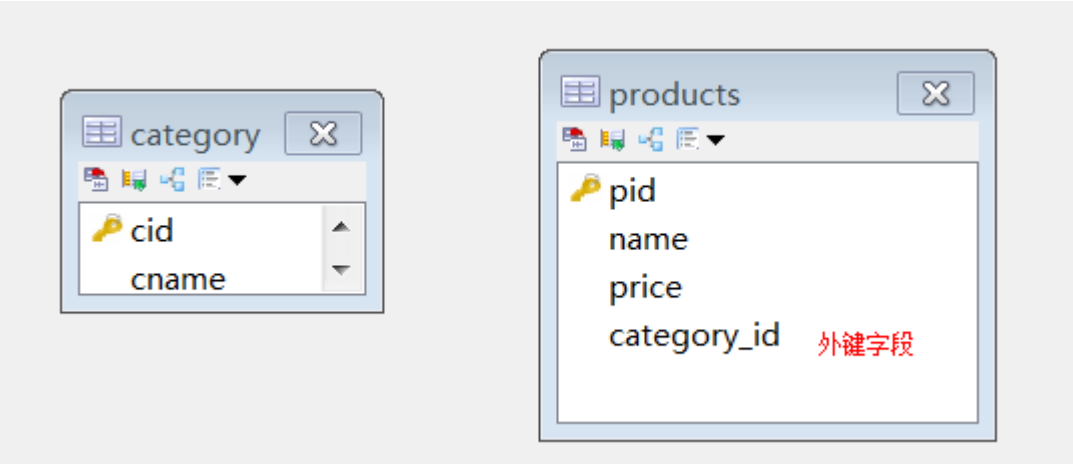


- 一对一关系：(了解)
  - 在实际的开发中应用不多.因为一对一可以创建成一张表.
  - 两种建表原则：
    - 外键唯一：主表的主键和从表的外键（唯一），形成主外键关系，外键唯一unique。
    - 外键是主键：主表的主键和从表的主键，形成主外键关系。

4.2 外键约束

现在我们有两张表"分类表"和"商品表"，为了表明商品属于哪个分类，通常情况下，我们将在商品表上添加一列，用于存放分类cid的信息，此列称为：外键





此时“分类表category”称为：主表，“cid”我们称为主键。“商品表products”称为：从表，category_id称为外键。我们通过主表的主键和从表的外键来描述主外键关系，呈现就是一对多关系。

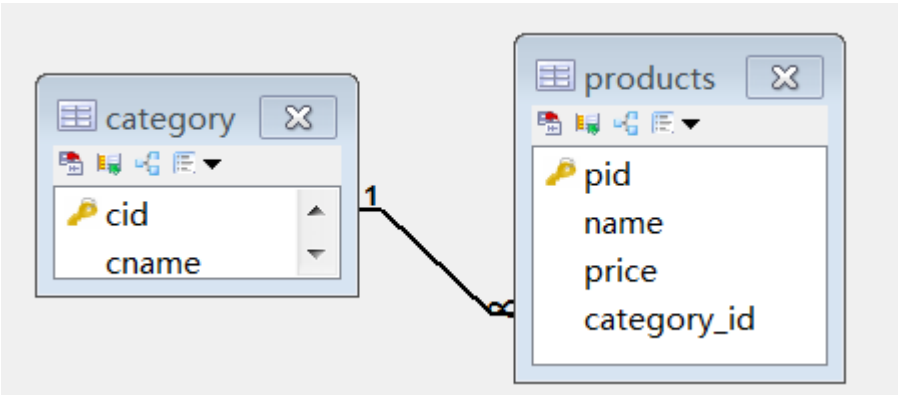
- 外键特点：
  - 从表外键的值是对主表主键的引用。
  - 从表外键类型，必须与主表主键类型一致。
- 声明外键约束

```
语法：
alter table 从表 add [constraint][外键名称] foreign key (从表外键字段名) references 主表 (主表的主键);

[外键名称]用于删除外键约束的，一般建议“_fk”结尾
alter table 从表 drop foreign key 外键名称
```

- 使用外键目的：
  - 保证数据完整性

4.3 一对多操作

分析



- category分类表，为一方，也就是主表，必须提供主键cid
- products商品表，为多方，也就是从表，必须提供外键category_id

实现：分类和商品

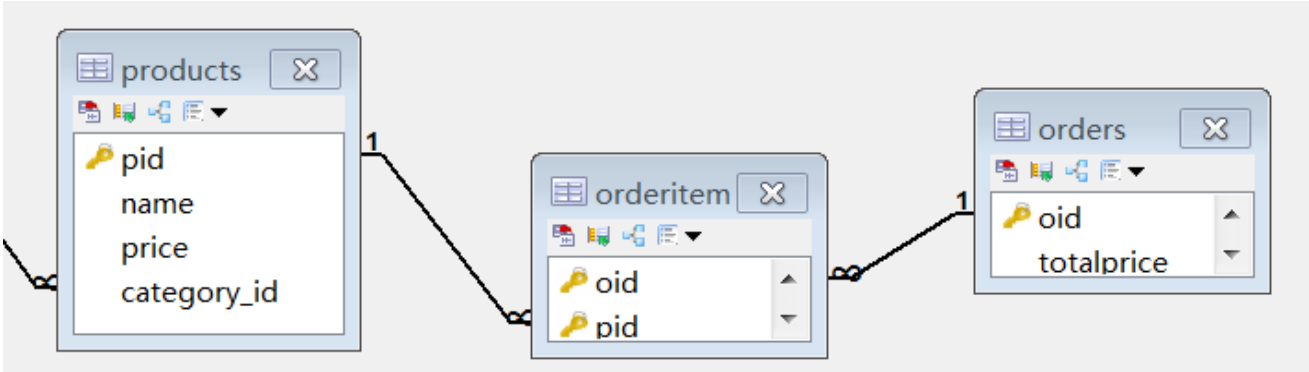
```
#创建分类表
create table category(
  cid varchar(32) PRIMARY KEY ,
  cname varchar(100) -- 分类名称
);

# 商品表
CREATE TABLE `products` (
  `pid` varchar(32) PRIMARY KEY  ,
  `name` VARCHAR(40) ,
  `price` DOUBLE
);

#添加外键字段
alter table products add column category_id varchar(32);

#添加约束
alter table products add constraint product_fk foreign key (category_id) references category
(cid);
```

操作

```
#1 向分类表中添加数据
INSERT INTO category (cid ,cname) VALUES('c001','服装');

#2 向商品表添加普通数据,没有外键数据，默认为null
INSERT INTO products (pid,pname) VALUES('p001','商品名称');

#3 向商品表添加普通数据，含有外键信息(category表中存在这条数据)
INSERT INTO products (pid ,pname ,category_id) VALUES('p002','商品名称2','c001');

#4 向商品表添加普通数据，含有外键信息(category表中不存在这条数据) -- 失败,异常
INSERT INTO products (pid ,pname ,category_id) VALUES('p003','商品名称2','c999');

#5 删除指定分类(分类被商品使用) -- 执行异常
DELETE FROM category WHERE cid = 'c001';
```

4.4 多对多

分析



- 商品和订单多对多关系，将拆分成两个一对多。
- products商品表，为其中一个一对多的主表，需要提供主键pid
- orders 订单表，为另一个一对多的主表，需要提供主键oid
- orderitem中间表，为另外添加的第三张表，需要提供两个外键oid和pid

实现：订单和商品

```
#商品表[已存在]

#订单表
create table `orders`(
  `oid` varchar(32) PRIMARY KEY ,
  `totalprice` double   #总计
);

#订单项表
create table orderitem(
  oid varchar(50),-- 订单id
  pid varchar(50)-- 商品id
);

#订单表和订单项表的主外键关系
alter table `orderitem` add constraint orderitem_orders_fk foreign key (oid) references
orders(oid);

#商品表和订单项表的主外键关系
alter table `orderitem` add constraint orderitem_product_fk foreign key (pid) references
products(pid);

#联合主键（可省略）
alter table `orderitem` add primary key (oid,pid);
```

操作

```
#1 向商品表中添加数据
INSERT INTO products (pid,pname) VALUES('p003','商品名称');

#2 向订单表中添加数据
INSERT INTO orders (oid ,totalprice) VALUES('x001','998');
INSERT INTO orders (oid ,totalprice) VALUES('x002','100');
```

```
#3向中间表添加数据(数据存在)
INSERT INTO orderitem(pid,oid) VALUES('p001','x001');
INSERT INTO orderitem(pid,oid) VALUES('p001','x002');
INSERT INTO orderitem(pid,oid) VALUES('p002','x002');

#4删除中间表的数据
DELETE FROM orderitem WHERE pid='p002' AND oid = 'x002';

#5向中间表添加数据(数据不存在) -- 执行异常
INSERT INTO orderitem(pid,oid) VALUES('p002','x003');

#6删除商品表的数据 -- 执行异常
DELETE FROM products WHERE pid = 'p001';
```