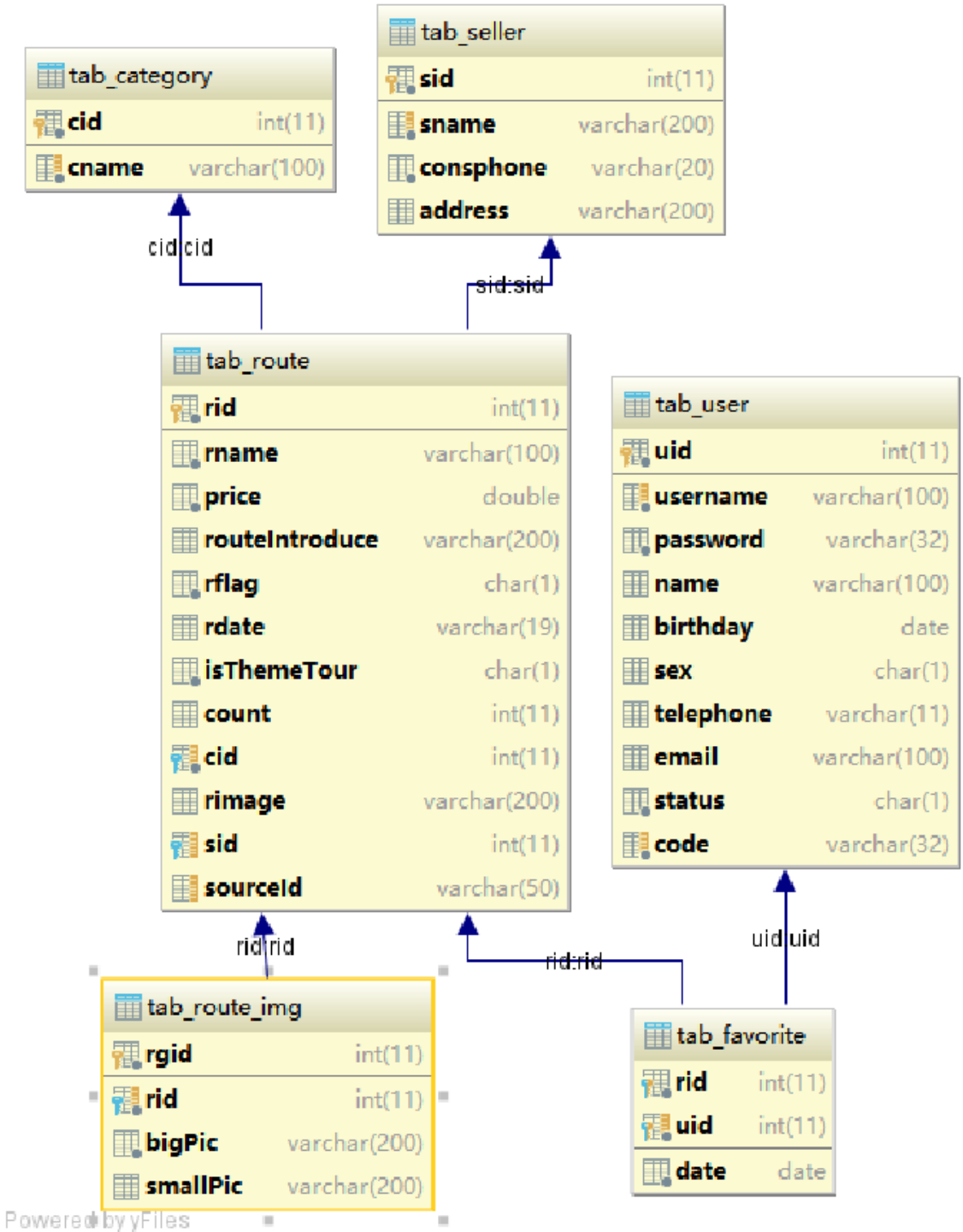
day04_多表查询

实际开发中，一个项目通常需要很多张表才能完成。

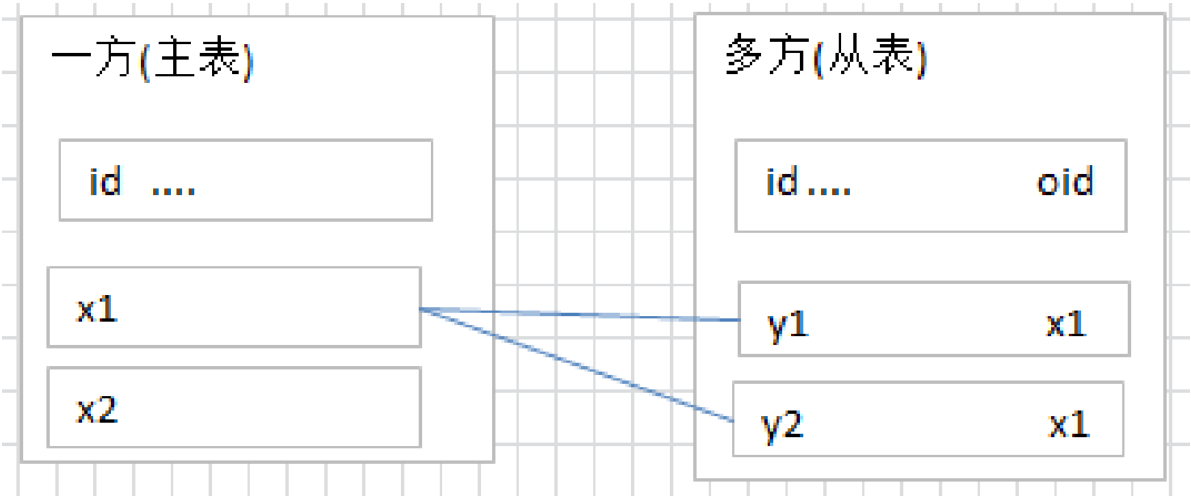
例如：一个商城项目就需要分类表(category)、商品表(products)、订单表(orders)等多张表。且这些表的数据之间存在一定的关系，接下来我们将在单表的基础上，一起学习多表方面的知识。



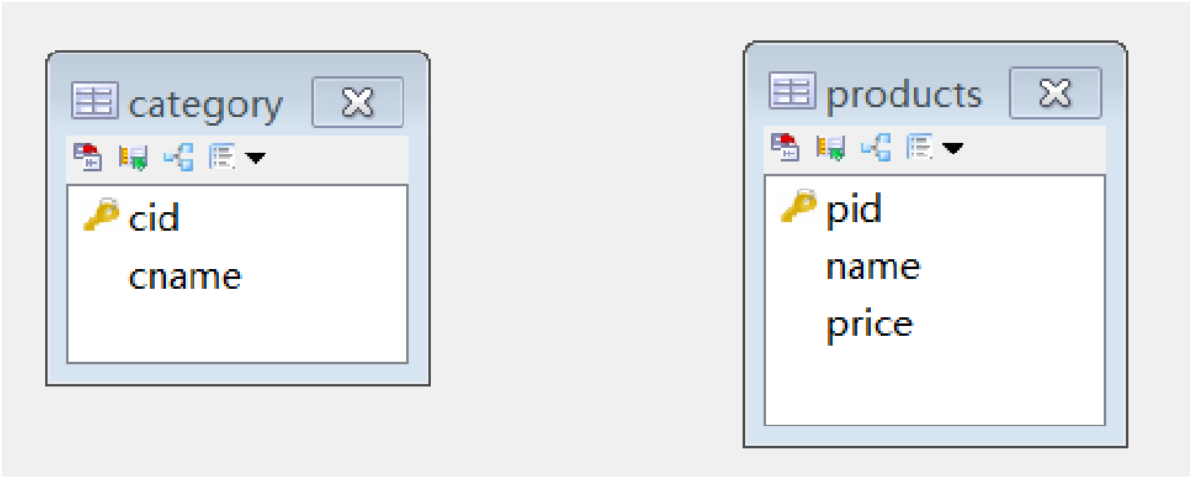
一对多关系：

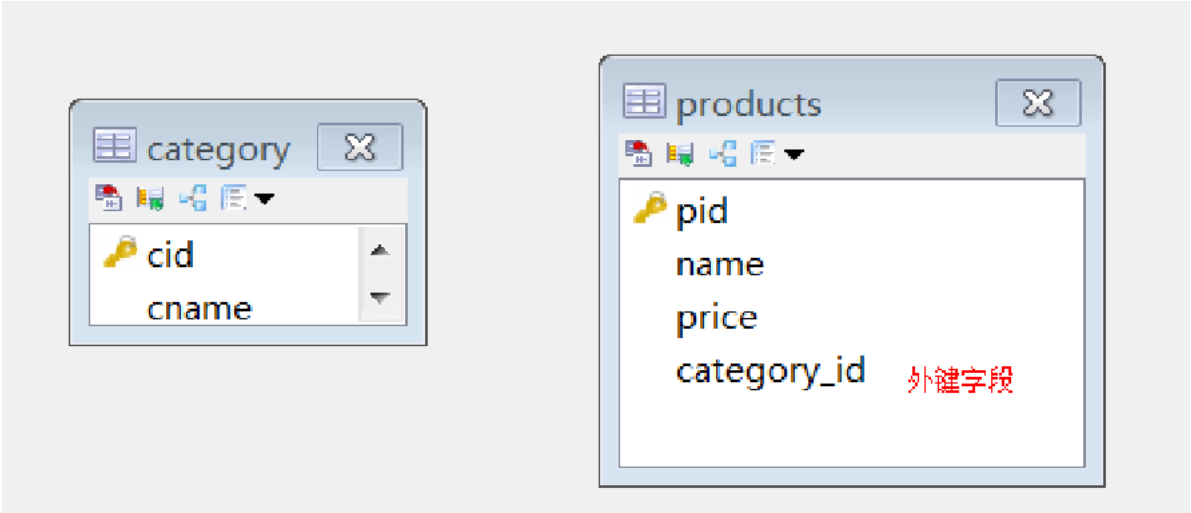
常见实例：客户和订单，分类和商品，部门和员工。

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。



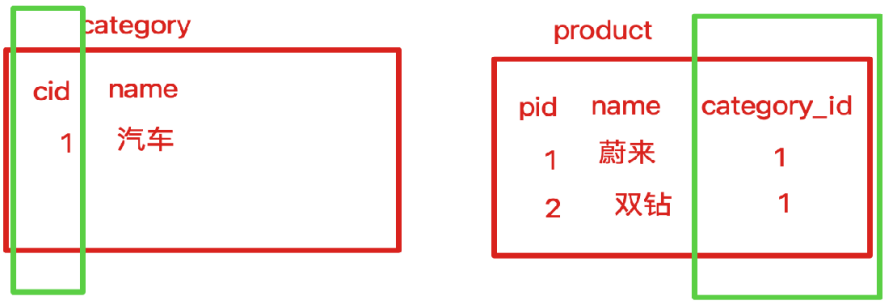
现在我们有两张表“分类表”和“商品表”，为了表明商品属于哪个分类，通常情况下，我们将在商品表上添加一列，用于存放分类cid的信息，此列称为：外键。

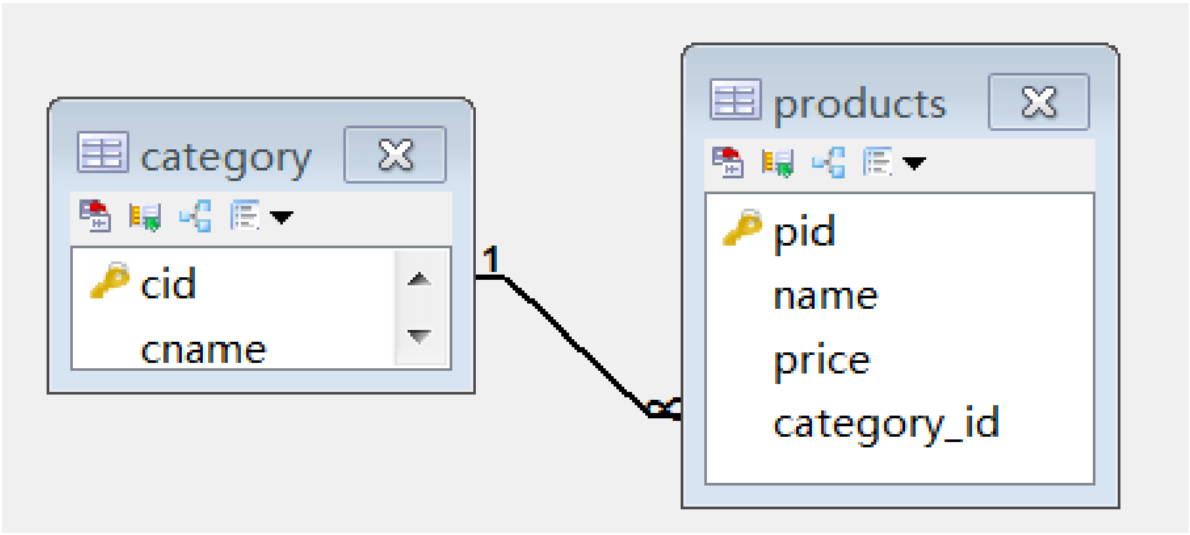




此时“分类表category”称为：主表，“cid”我们称为主键。“商品表products”称为：从表，category_id称为外键。我们通过主表的主键和从表的外键来描述主外键关系，呈现就是一对多关系。

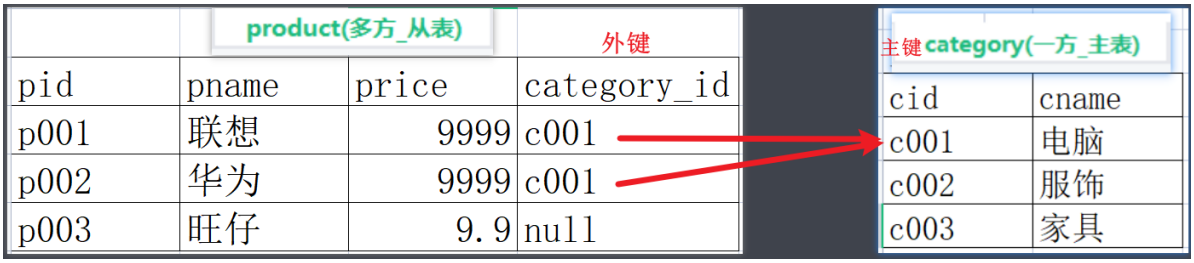
> 外键特点：
> 从表外键的值是对主表主键的引用。
> 从表外键类型，必须与主表主键类型一致。
> 外键约束优点:
> 在插入数据时，保证了数据的准确性。
> 在删除数据时，保证了数据的完整性。





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

1.外键约束简介



主从表关系：一对多关系
主表：一方
从表：多方

外键：从表中参考主表的主键额外添加了一个字段,这个字段就叫外键　　注意：外键的类型需要和主表主键类型一致

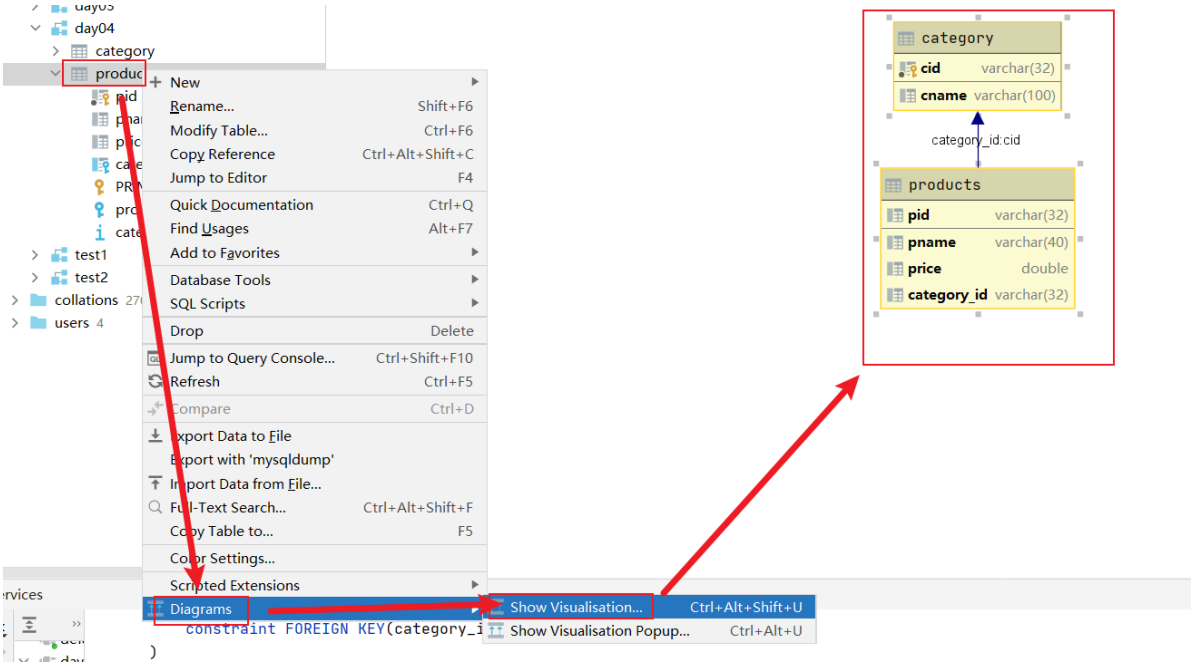
外键约束：FOREIGN KEY　从表的外键多次引用了主表的主键　　举例：product从表的category_id引用了categrory主表的cid
在从表中添加外键约束格式：　[constraint 外键名称] FOREIGN KEY(外键) references 主表名(主键)
外键约束限制主表的删除操作：保证了数据的准确性和完整性
外键约束限制从表的插入操作：保证了主从表数据的一致性

2.查看多表结构图

创建主从表添加外键

```
# 操作表的前提: 先创建库并使用它
create database day04;
use day04;
# 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,
    pname VARCHAR(40) ,
    price DOUBLE ,
    category_id varchar(32),
    constraint FOREIGN KEY(category_id) references category(cid)
);
# 注意: 想要使用外键约束,mysql底层需要配置成innodb存储引擎
```

右键从表查看结构图

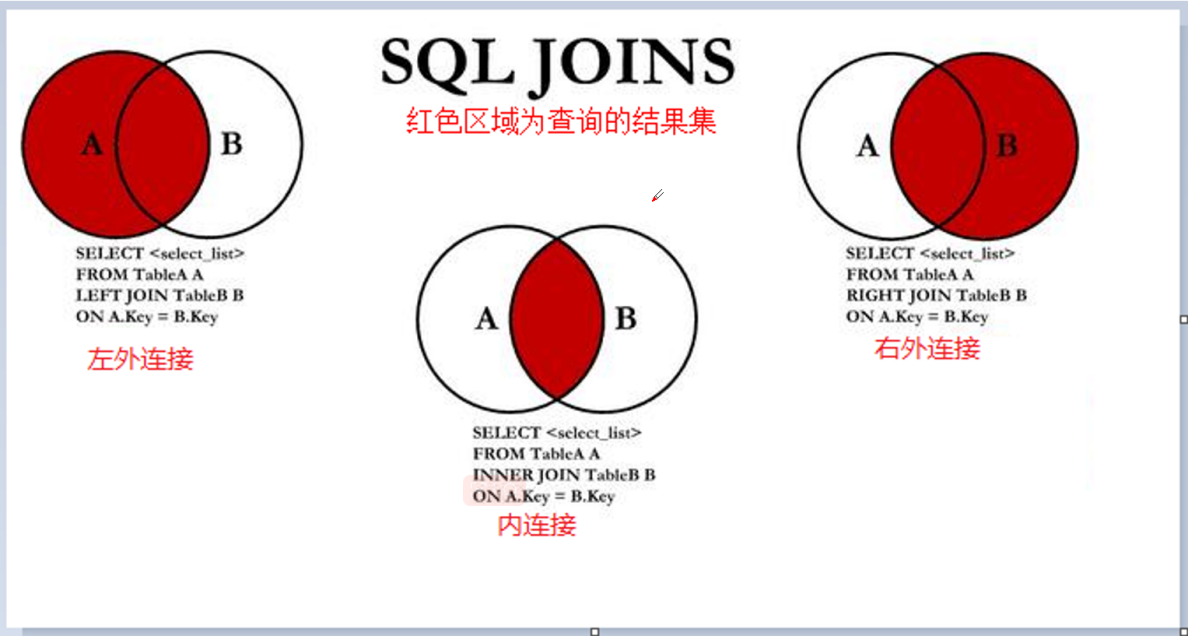


3.演示外键约束的作用

```
# 演示外键约束的作用
# 限制从表的插入操作: 保证了主从表数据的一致性
insert into products values (1,'联想',9999,'c001'); # 报错:无法添加或更新子行,因为category表中没有c001记录
# 为了演示效果,向主表中插入数据,然后从表再插入查看效果
insert into category values ('c001','电脑'); # 为了演示效果,向主表中插入数据
insert into products values (1,'联想',9999,'c001'); # 插入成功,因为category表中已经有了c001记录
```

```
# 限制主表的删除操作: 保证了数据的准确性和完整性
delete from category where cid='c001'; # 报错: 无法删除或更新父行,因为从表正在引用主表的c001记录
# 就想删除主表的c001记录,怎么办?
# 可以先把从表的引用去掉(方式1:直接干掉引用的那条记录  方式2:把引用的那条记录外键值改为null)
delete from products where category_id='c001';
# 再次在主表中删除c001记录
delete from category where cid='c001'; # 删除成功,因为c001记录没有被引用了
```

4.多表查询[重点]



准备数据

```
# 准备数据
# 创建hero表
CREATE TABLE hero
(
    hid       INT PRIMARY KEY,
    hname     VARCHAR(255),
    kongfu_id INT
);

# 创建kongfu表
CREATE TABLE kongfu
(
    kid   INT PRIMARY KEY,
    kname VARCHAR(255)
);
# 插入hero数据
INSERT INTO hero VALUES(1, '鸠摩智', 9),(3, '乔峰', 1),(4, '虚竹', 4),(5, '段誉', 12);
# 插入kongfu数据
INSERT INTO kongfu VALUES(1, '降龙十八掌'),(2, '乾坤大挪移'),(3, '猴子偷桃'),(4, '天山折梅手');
```

交叉连接

交叉连接(慎用): 就是两个表数据的乘积,所以也叫笛卡尔积

注意: 交叉连接的结果本质是一个错误(包含了很多无效数据)

交叉连接关键字: cross join

显式交叉连接格式: select 字段名 from 左表 cross join 右表;
隐式交叉连接格式: select 字段名 from 左表,右表;

注意: 左右表没有特殊含义只是位置关系,在前面的是左表,在后面的是右表

内连接

知识点:

内连接(常用): 就是保留两个表数据的交集,用on过滤其他无效数据

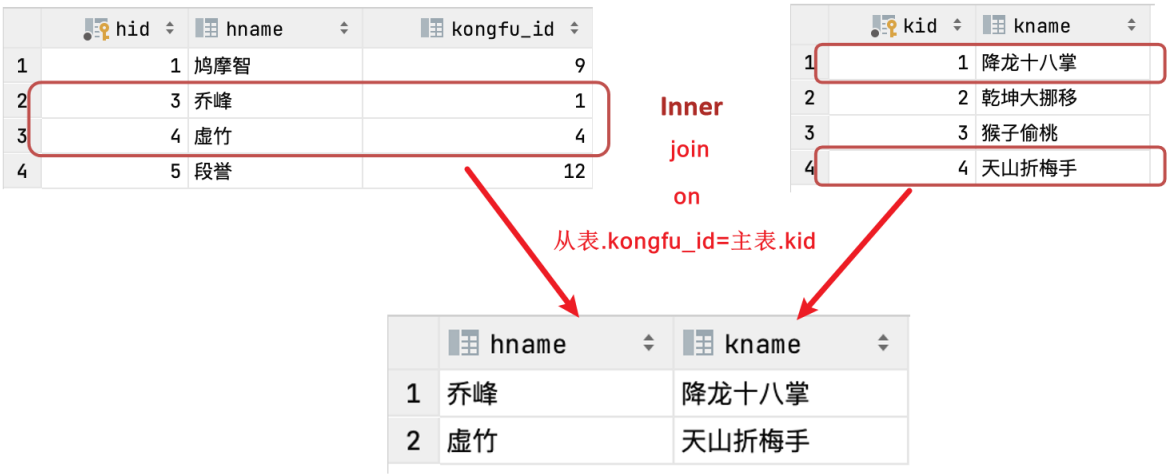
内连接关键字: inner join ... on 注意: inner 可以省略

显式内连接格式: select 字段名 from 左表 inner join 右表 on 左右表关联条件;
隐式内连接格式: select 字段名 from 左表,右表 where 左右表关联条件;

内连接特点: 只保留两个表有交集的数据,其他数据直接过滤掉

注意: 左右表没有特殊含义只是位置关系,在前面的是左表,在后面的是右表

内连接示例:




```
# 演示内连接查询
# 需求: 查询有对应功夫的英雄,要求展示英雄名和他的功夫
# 方式1:显式内连接格式: select 字段名 from 左表 inner join 右表 on 左右表关联条件;
select hname,kname from hero inner join kongfu on hero.kongfu_id=kongfu.kid;
# 当然也可以给左右表分别起别名
select hname,kname from hero h inner join kongfu k on h.kongfu_id=k.kid;

# 方式2:隐式内连接格式: select 字段名 from 左表,右表 where 左右表关联条件;
select hname,kname from hero , kongfu where hero.kongfu_id=kongfu.kid;
# 当然也可以给左右表分别起别名
select hname,kname from hero h ,kongfu k where h.kongfu_id=k.kid;
```

外连接

知识点:

```
外连接(根据实际情况定): 就是两个表数据的并集,用on过滤其他无效数据

左外连接关键字: left outer join ... on          注意: outer 可以省略
左外连接格式:  select 字段名 from 左表 left outer join 右表 on 左右表关联条件;
左外连接特点:  以左表为主,左表所有数据都保留,右表只保留交集的部分

右外连接关键字: right outer join ... on          注意: outer 可以省略
右外连接格式:  select 字段名 from 左表 right outer join 右表 on 左右表关联条件;
右外连接特点:  以右表为主,右表所有数据都保留,左表只保留交集的部分

注意: 左右表没有特殊含义只是位置关系,在前面的是左表,在后面的是右表
注意: 左外连接和右外连接记忆一种格式即可,只要左右表换下顺序就能达到对应的需求
```

左外连接示例


```
# 需求: 查询所有英雄对应的功夫,注意:没有对应功夫的用null补全
# 分析:  如果使用左外连接,左表: 英雄表
# 左外连接
select hname,kname from hero left join kongfu on hero.kongfu_id=kongfu.kid;
# 当然也可以起表名
select hname,kname from hero h left join kongfu k on h.kongfu_id=k.kid;
```

	kid	kname
1	1	降龙十八掌
2	2	乾坤大挪移
3	3	猴子偷桃
4	4	天山折梅手

	hid	hname	kongfu_id
1	1	鸠摩智	9
2	3	乔峰	1
3	4	虚竹	4
4	5	段誉	12

	hname	kname
1	乔峰	降龙十八掌
2	虚竹	天山折梅手
3	<null>	乾坤大挪移
4	<null>	猴子偷桃

```
# 需求: 查询所有功夫,并且只展示有对应功夫的英雄
# 分析:  如果使用左外连接,左表: 功夫表
# 左外连接
select hname,kname from kongfu left join hero on hero.kongfu_id=kongfu.kid;
# 当然也可以起表名
select hname,kname from kongfu k left join hero h on h.kongfu_id=k.kid;
```

右外连接示例

> 需求: 把上述左外连接的需求使用右外连接完成,实现同样效果

```
# 演示右外连接查询
# 需求: 查询所有英雄对应的功夫,注意:没有对应功夫的用null补全
# 分析:  如果使用右外连接,右表: 英雄表
select hname,kname from kongfu right join hero on hero.kongfu_id=kongfu.kid;
# 当然也可以起表名
select hname,kname from kongfu k right join hero h  on h.kongfu_id=k.kid;


# 需求: 查询所有功夫,并且只展示有对应功夫的英雄
# 分析:  如果使用右外连接,右表: 功夫表
select hname,kname from hero right join kongfu on hero.kongfu_id=kongfu.kid;
# 当然也可以起表名
select hname,kname from  hero h right join kongfu k on h.kongfu_id=k.kid;
```

子查询

知识点:

```
子查询: 一个完整的select语句作为另外一个select语句的表或者条件使用,这个完整的select语句就是子查询语句

子查询特点: 子查询语句嵌套到主查询语句中
```

示例:

```
# 需求4: 查询化妆品分类下的所有商品详情
# 分析: 先在分类表中查询化妆品分类id,再根据这个分类id去商品表中找到对应所有商品
# 方式1: 子查询作为主查询语句的条件使用
select *
from day04.products
where category_id = (select cid from day04.category where cname = '化妆品');

# 方式2: 子查询作为主查询语句的表使用
# 显示内连接
select * from products p
    inner join (select cid,cname from day04.category where cname = '化妆品') c
    on p.category_id=c.cid;

# 隐式内连接
select * from products p, (select cid,cname from day04.category where cname = '化
妆品') c
where p.category_id=c.cid;




# 需求5: 查询化妆品和家电两个分类的所有商品详情
# 分析: 先查询化妆品和家电两个分类的id,再根据分类id去商品表中查询对应的商品
# 方式1: cname使用范围查询
select *
from products
where category_id in (select cid from category where cname in ('化妆品', '家电'));

# 方式2: cname使用逻辑运算符查询
select *
from products
where category_id in (select cid from category where cname ='化妆品' or cname ='家
电');
```

自连接

知识点:

```
自连接: 作为一种特例，可以将一个表与它自身进行连接，称为自连接。

语法: 自连接语法和内外连接的语法一样，只不过换成了只在同一张表上面操作

特点: 特殊的地方就是左表和右表是同一张表,只是起了不同的别名

注意: 使用自连接时需为表指定多个别名，且对所有列的引用均要用别名区分。

应用场景: 省市县区域表 或者 员工表
```

示例

```
#  演示自连接查询
# 需求1: 查询我家乡湖南省所有的城市
# 分析: 省市区三级都在一个表中,那么就可以使用自连接
# 子查询方式: 先查询湖南省的id,再根据湖南省id查询下面所有的城市
select * from areas where pid = (select id from areas where title = '湖南省');

# 自连接方式: 给areas表起两个别名分别当成省表和市表使用
select * from areas sheng
                join areas shi on shi.pid=sheng.id
where sheng.title = '湖南省';




# 需求2: 查询我家乡株洲市下面所有的区县
# 子查询方式: 先查询株洲的id,再根据株洲id查询下面所有的区县
select * from areas where pid = (select id from areas where title = '株洲市');

# 自连接方式: 给areas表起两个别名分别当成市表和区县表使用
select * from areas shi
                join areas xian on xian.pid=shi.id
where shi.title = '株洲市';
```