外键约束

MySQL的外键约束用来在两个表数据之间建立链接，其中一张表的一个字段被另一张表中对应的字段约束。也就是说，设置外键约束至少要有两种表，被约束的表叫做从表（子表），另一张叫做主表（父表），属于主从关系。

其中主表中的关联字段叫做主键，外表中的关联字段叫做外键。

外键约束主要作用就是能够让表与表之间的数据建立关联，使数据更加完整，关联性更强。

关于完整性，可以通过一个例子来说明

假如有两种表，一张用户账户表（用于存储用户账户），一张是账户信息表（用于存储账户中的信息）。

1）我不小心将用户账户表中的某个用户删除了，那么账户信息表中与这个用户有关的数据就变成无源数据了，找不到其属于哪个用户账户，导致用户信息不完整。

2）我在账户信息表中随便添加了一条数据，而其在用户账户表中没有对应的用户，这样用户信息也是不完整的。

如果有了外键约束，将用户账户表设为主表，账户信息表设为从表，那么就无法直接往账户信息表中插入数据，在用户账户表中删除用户，账户信息表中的用户信息也会被删除。外键约束的方式可以使两张紧密的联系在一起，保障数据完整性和一致性的同时，日常维护也更加方便

create table product(

 pid int primary key auto\_increment, -- 自增长

 pname varchar(40),

 price double,

 num int

);

insert into product values(null,'苹果电脑',18000.0,10);

insert into product values(null,'华为5G手机',30000,20);

insert into product values(null,'小米手机',1800,30);

insert into product values(null,'iPhonex',8000,10);

insert into product values(null,'iPhone7',6000,200);

insert into product values(null,'iPhone6s',4000,1000);

insert into product values(null,'iPhone6',3500,100);

insert into product values(null,'iPhone5s',3000,100);

SELECT \* from product;

-- DQL 语句，mysql中最重要的，必须掌握

SELECT pid,pname,num FROM product;-- 指定某张表的 特定列来查询

-- 去重，只能去重一样的内容（工作中不用）

select DISTINCT num from product

select pid as '序号',pname as '名字',num as '数量' from product

-- 运算符（了解，工作中用不到） +,-,\*,/,%不会影响表中实际结果，作用不大

select pid,pname,price+99,num from product;

-- 条件查询 where 关键字(重要)

# 大于：>

select \* from product where price > 3000;

# 小于：<

select \* from product where price < 3000;

# 大于等于：>=

select \* from product where price >= 3000;

# 小于等于：<=

select \* from product where price <= 3000;

# 等于：= 不能用于null判断,null不能作为判断条件

select \* from product where pid = 1;

# 不等于：!= 或 <>

select \* from product where pid <> 1;

# 安全等于: <=> 可以用于null值判断

select \* from product where pid <=> 1;

-- 逻辑运算符（建议用单词，可读性来说）（也算重要）

# 逻辑与：&& 或 and

select \* from product where price > 3000 and num > 20;-- 价格大于3000和数量大于20的

# 逻辑或：|| 或 or

select \* from product where price > 3000 or num > 20;-- 价格大于3000且数量大于20的

# 逻辑非：! 或 not

select \* from product where not price = 3000;-- 价格不等于3000的

-- 范围（不太重要，但面试中可能会考）

-- 集合范围 in

-- 区间范围 BETWEEN

-- 不在这个区间 not between

SELECT \* from product where pid between 1 and 3;

SELECT \* from product where pid not between 1 and 3;

SELECT \* from product where pid in(6,8);

-- 模糊查询 关键字 like（工作用的不多，算重要，面试可能会问）

SELECT \* from product where pname like 'i%'-- 以字母i开头的pname

SELECT \* from product where pname like '%s'-- 以字母s结尾的pname

SELECT \* from product where pname like '%one%'-- pname中包含one

-- 特殊值null处理 is（重要）

SELECT \* from product where num is null

SELECT \* from product where num is not null

SELECT \* from product where num <=> null

-- 排序（重要）

-- 排序 ORDER BY 字段 asc/desc asc（升序）/desc（降序）

SELECT \* from product ORDER BY PID DESC

SELECT \* from product ORDER BY PRICE ASC

-- 多列排序（组合排序）在第一个排序的数据基础上，再进行一次排序

-- ORDER BY 字段1 asc/desc，字段2 asc/desc

SELECT \* from product ORDER BY PRICE ASC,NUM DESC

-- mysql的聚合函数

select version()-- 查看当前版本号

select now()-- 查看当前时间

select 10-9

-- 常用的聚合函数（以下函数重要）

-- max()求最大值

-- min()求最小值

-- avg()求这一列的平均值

-- sum()求和

-- count()统计这一列有多少条数据

SELECT max(num) from product

SELECT max(num) from product

SELECT avg(price) from product -- 9287.5

SELECT sum(price) from product -- 74300

SELECT count(price) from product-- 8

-- 分页 limit 起始数据，查多少页

SELECT \* from product limit 0,3;-- 第一个数字代表角标，角标时从零开始的。从第一列开始往后取3条数据

-- GROUP BY 分组查询，一般与聚合函数联合使用

SELECT count(\*) from product

SELECT num,count(num) from product GROUP BY num

# 创建学生表(有sid,学生姓名,学生性别,学生年龄,分数列,其中sid为主键自动增长)

CREATE TABLE student(

sid INT PRIMARY KEY auto\_increment,

sname VARCHAR(40),

sex VARCHAR(10),

age INT,

score DOUBLE,

classid int -- 班级编号

);

INSERT INTO student VALUES(null,'zs','男',18,98.5,1);

INSERT INTO student VALUES(null,'ls','女',18,96.5,1);

INSERT INTO student VALUES(null,'ww','男',15,50.5,1);

INSERT INTO student VALUES(null,'zl','女',20,98.5,2);

INSERT INTO student VALUES(null,'tq','男',18,60.5,2);

INSERT INTO student VALUES(null,'wb','男',38,98.5,1);

INSERT INTO student VALUES(null,'小丽','男',18,100,1);

INSERT INTO student VALUES(null,'小红','女',28,28,2);

INSERT INTO student VALUES(null,'小强','男',21,95,3);

select \* from student;

-- 根据性别分组，统计每一组有多少人

select sex,count(\*) from student GROUP BY sex

select count(sex) from student

-- 根据性别分组，统计每组学生的平均分

select sex,avg(score) from student GROUP BY sex

-- 根据性别分组，统计每组学生的人数和总分

select sex,count(sex),sum(score) from student GROUP BY sex

-- 分组后进行筛选 having

-- 根据性别分组，统计每一组有多少人，并且筛选出大于5的数据

select sex,count(\*) from student GROUP BY sex having count(\*)>5