

一对多关联

- 创建两个表：

```
create table t_stu(  
    s_id int not null auto_increment primary key comment '学生编号',  
    s_name varchar(50) not null comment '学生姓名',  
    s_sex int not null default 1 comment '性别。1-男, 0-女',  
    s_age int not null default 0 comment '学生年龄',  
    c_id int not null comment '班级编号'  
);  
  
create table t_class(  
    c_id int not null auto_increment primary key comment '班级编号',  
    c_name varchar(50) not null comment '班级名称'  
);
```

在多方表上添加一个关联字段，这个字段的内容是一方的主键值，所以关联字段名一般跟一方的主键字段同名

如下图：

学生表有一个c_id的关联字段，它与班级表的主键字段c_id同名

s_id	s_name	s_sex	s_age	c_id
1	小李	1	22	1
2	小王	1	21	1

c_id	c_name
1	计科1班

- 外键

在多方表上添加外键如下：

保存	添加外键	删除外键				
字段	索引	外键	触发器	选项	注释	SQL 预览
名	字段	参考模式	参考表	参考字段	删除时	更新时
fk_t_stu_t_calss	c_id	java201102	t_class	c_id	RESTRICT	RESTRICT
多方表的关联字段		一方表名		一方主键字段		

外键的约束原则：

- 在多方表插入数据时，关联字段的值必须在一方表里存在的

- 2. 删除一方表的记录时，这个主键值在多方表不能被使用

注意：外键只是起数据约束关系，两个表之间有没有关联关系不是由外键决定的。

- 关联查询语句

```
select s.s_name, c.c_name from t_stu s, t_class c where s.c_id=c.c_id;
select s.*, c.c_name from t_stu s, t_class c where s.c_id = c.c_id and
s.s_age>21; -- 关联查询可以使用其它查询条件
select s.*, c.c_name from t_stu s, t_class c where s.c_id = c.c_id order by
s.s_age desc; -- 关联查询可以排序
select count(*) total, c.c_name from t_stu s, t_class c where s.c_id = c.c_id
GROUP BY c.c_id; -- 关联查询使用分组
select count(*) total, c.c_name from t_stu s, t_class c where s.c_id = c.c_id
GROUP BY c.c_id HAVING total>1;
```

- 内联，外联，交叉连接

```
select s.*, c.c_name from t_stu s inner join t_class c on s.c_id=c.c_id; -- 内联查
询
SELECT s.*, c.c_name from t_stu s left join t_class c on s.c_id=c.c_id; -- 左联
select s.*, c.c_name from t_stu s right join t_class c on s.c_id=c.c_id; -- 右联
select s.*, c.c_name from t_stu s cross join t_class c; -- 交叉连接，笛卡尔
```

- 子查询

```
select s.*, (select c.c_name from t_class c where c.c_id=s.c_id) c_name from
t_stu s; -- 子查询作为字段
select tmp.* from (select sin.* from t_stu sin where sin.s_sex=1) tmp where
tmp.s_age>22; -- 子查询作为临时表
select s.* from t_stu s where s.c_id in (select c.c_id from t_class c where
c.c_name like '%计科%'); -- 子查询作为where条件
```

- 子查询的应用

```
insert into student(stu_name,stu_age,stu_sex,stu_birthday)
(select stu_name,stu_age,stu_sex,stu_birthday from student); -- 插入语句
的values用子查询

create table new_student (select * from student); -- 用子查询来创建一个新表

-- 利用not in方式实现一个表存在，另一给表不存在
select c.* from t_class c where c.c_id not in(select DISTINCT s.c_id from
t_stu s);
-- 利用外联方式通过判断null 实现一个表存在，另一个表不存在
```

```

select s.*,c.c_name from t_stu s right join t_class c on s.c_id=c.c_id where
s.s_name is null;
-- 利用exists(只要子查询返回记录, 结果就为真, 当前这条记录就会被查出来)实现一个表存在, 另
一个表不存在
select c.c_name from t_class c where not EXISTS(select s.* from t_stu s
where s.c_id=c.c_id);

-- 查询表里重复的记录
select s.* from t_stu s where s.s_name in(select tmp.s_name from ( select
count(*) total,s.s_name from t_stu s GROUP BY s.s_name HAVING total>1) tmp);

```

多对多

通过中间表来设置对应关系

- 创建表和插入数据

```

/*
创建多对多关系表: 产品表product, 功能表func, 中间表product_func
*/
create table product(
    p_id int not null auto_increment PRIMARY key comment '产品编号',
    p_name varchar(50) not null comment '产品名称'
);

create table func(
    f_id int not null auto_increment primary key comment '功能编号',
    f_name varchar(50) not null comment '功能名称'
);

create table product_func(
    p_id int not null comment '产品编号',
    f_id int not null comment '功能编号'
);

insert into product(p_name) values('华为P40');
insert into product(p_name) values('华为M40');
insert into product(p_name) values('小米10');
insert into product(p_name) values('VIVO X10');
insert into product(p_name) values('OPPO R10');
insert into product(p_name) values('苹果12');

insert into func(f_name) values('智能手机');
insert into func(f_name) values('大屏手机');
insert into func(f_name) values('拍照手机');
insert into func(f_name) values('5G手机');
insert into func(f_name) values('音乐手机');
insert into func(f_name) values('超长待机');

insert into product_func(p_id,f_id) values(1,1);
insert into product_func(p_id,f_id) values(1,3);
insert into product_func(p_id,f_id) values(1,4);
insert into product_func(p_id,f_id) values(3,1);

```

```
insert into product_func(p_id,f_id) values(3,6);
insert into product_func(p_id,f_id) values(4,3);
insert into product_func(p_id,f_id) values(4,5);
```

- 多对多关系的关联查询

```
select p.*,f.*
  from product p, func f, product_func pf
   where p.p_id = pf.p_id and f.f_id = pf.f_id;
```

字段的约束

- 主键

主键不允许为Null，不能有重复，自动排序

一个表只有一个主键，用能够唯一识别记录这种字段做主键，如果没有类似字段，建议新建一个数字的编号或随机字符串的字段来作为主键

- 外键

一对多关联时，在多方的表上创建外键。外键用来约束两个表数据的一致性：

1. 插入多方表的时候，关联字段的值一定要在一方表里存在
2. 删除一方表记录时，一定要保证多方表没有使用这个id

- 不为null

插入记录时字段必须被赋值

- 默认值

当插入记录时，没有给该字段赋值就用默认值来填充

- 唯一索引

给字段创建一个unique类型的索引，就能保证该字段不会出现重复内容

字段	索引	外键	触发器	选项	注释	SQL 预览	
名	字段						
▶ emp_name	`emp_name`					UNIQUE	BTREE

索引

为了提高查询效率，给表添加索引（排序文件）

创建原则：

- 1.数据量大（百万），影响查询效率
- 2.针对查询语句的where条件字段添加索引，这些字段不能有大量重复内容，多个字段可以用联合索引
- 3.要使用索引有效，查询条件的字段不要使用函数
- 4.索引会影响插入，删除，修改的效率，所以索引不是越多越好

创建索引：

保存							添加索引		删除索引		
字段	索引	外键	触发器	选项	注释	SQL 预览					
名		字段				索引类型	索引方法	注释			
emp_name		`emp_name`				UNIQUE	BTREE				
emp_salary		`emp_salary`				NORMAL	BTREE				
emp_job		`emp_job`, `emp_salary`				NORMAL	BTREE				