外键

外键：foreign key，外面的键，即不在自己表中的键。如果一张表中有一个非主键的字段指向另外一张表的主键，那么将该字段称之为外键。每张表中，可以有多个外键。

新增外键

外键既可以在创建表的时候增加，也可以在创建表之后增加（但是要考虑数据的问题）。

第 1 种：在创建表的时候，增加外键

基本语法：foreign key(外键字段) + references + 外部表名(主键字段);

执行如下 SQL 语句，进行测试：

-- 创建外键

create table my\_foreign1(

id int primary key auto\_increment,

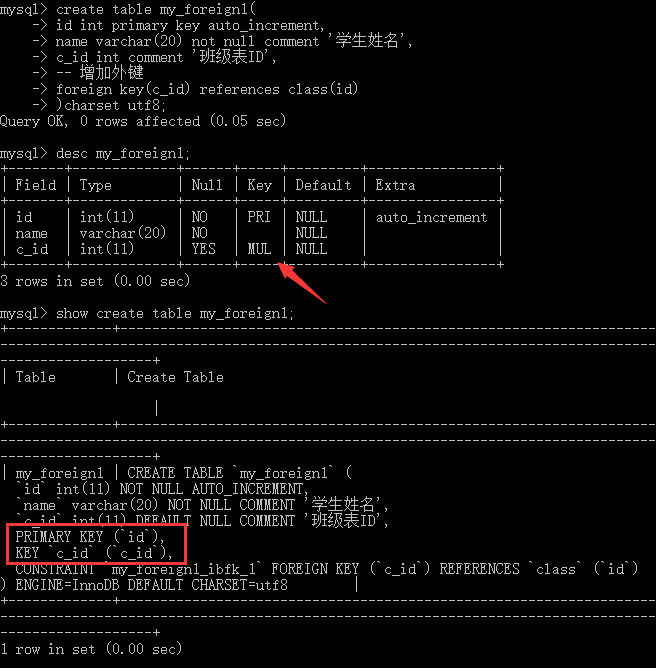
name varchar(20) not null comment '学生姓名',

c\_id int comment '班级表ID',

-- 增加外键

foreign key(c\_id) references class(id)

)charset utf8;

[](https://camo.githubusercontent.com/b319276c820fd11ac1236f048d11a49fea71cead/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730383139323234393032323231)

观察上图可知，字段c\_id的key显示为MUL，表示多个键的意思。这是因为外键要求字段本身是一个索引（普通索引）如果字段本身没有索引，外键就会先创建一个索引，然后才创建外键本身。此外，CONSTRAINT后面的my\_foreign\_ibfk\_1表示外键的名字。

第 2 种：在创建表之后，增加外键

基本语法：alter table + 表名 + add[constraint + 外键名字] + foreign key(外键字段) + references + 外部表名(主键字段);

执行如下 SQL 语句，进行测试：

-- 创建外键

create table my\_foreign2(

id int primary key auto\_increment,

name varchar(20) not null comment '学生姓名',

c\_id int comment '班级表ID'

)charset utf8;

-- 增加外键

alter table my\_foreign2 add

-- 指定外键名

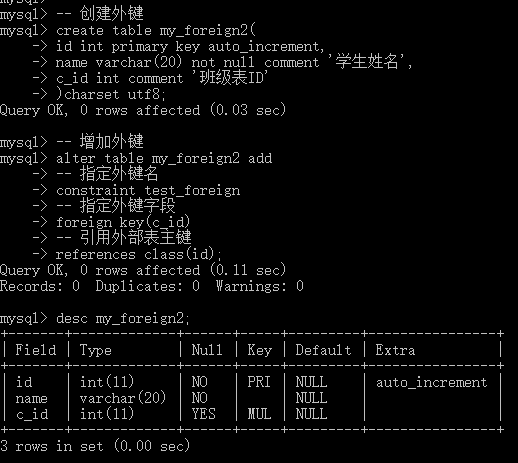
constraint test\_foreign

-- 指定外键字段

foreign key(c\_id)

-- 引用外部表主键

references class(id);

[](https://camo.githubusercontent.com/fadb12bf75b22e65551df9bb3d6772c9d6084628/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730383139323330303431363239)

如上图所示，显然咱们已经增加外键成功啦！

修改外键 & 删除外键

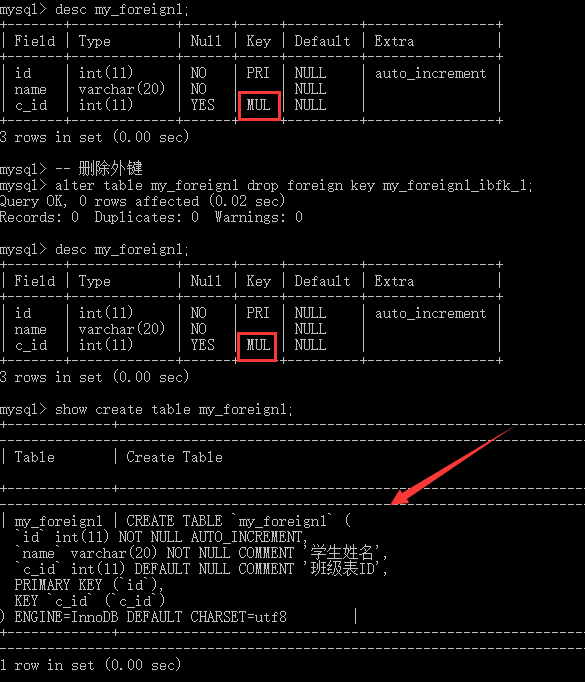
外键不能修改，只能先删除后增加。

基本语法：alter table + 表名 + drop foreign key + 外键名字;

执行如下 SQL 语句，进行测试：

-- 删除外键

alter table my\_foreign1 drop foreign key my\_foreign1\_ibfk\_1;

[](https://camo.githubusercontent.com/b6fc7e191353005b9259ad7eca16cd67b9af3e49/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730383139323331303134373735)

观察上图可知，删除外键不能通过查看表结构来体现，而是应该通过创建表的语句来查看。

外键作用

首先，给出父表和子表的定义：

父表，指外键所指向的表；

子表，指相对于父表，拥有外键的表。

外键默认的作用有两个，分别对子表和父表进行约束。

第 1 种：约束子表

在子表进行数据的写操作（增和改）的时候，如果对应的外键字段在父表找不到对应的匹配，那么操作就会失败。

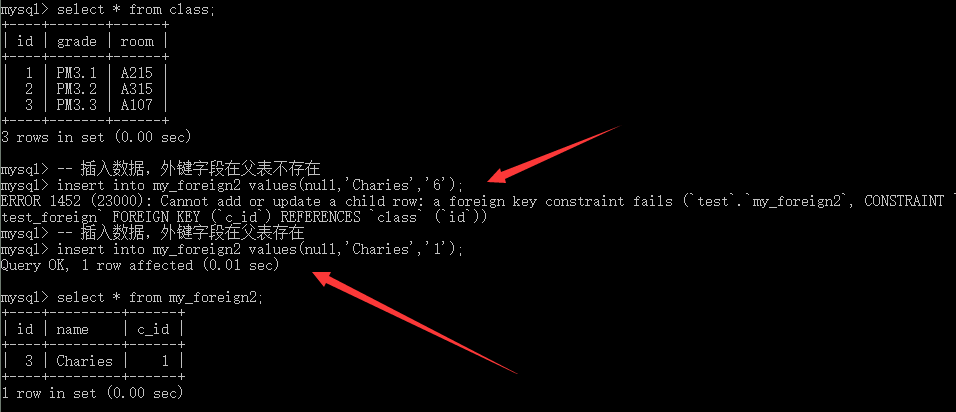
执行如下 SQL 语句，进行测试：

-- 插入数据，外键字段在父表不存在

insert into my\_foreign2 values(null,'Charies','6');

-- 插入数据，外键字段在父表存在

insert into my\_foreign2 values(null,'Charies','1');

[](https://camo.githubusercontent.com/7a214a3870cae0acf4aa777e4d1ce88106cf6c37/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730383230313035313234363437)

如上图所示，在我们向子表my\_foreign2插入外键字段为6的时候，提示插入失败，原因就是在父表class中，没有ID为6的记录。而在我们向子表my\_foreign2插入外键字段为1的时候，提示成功，原因就是在父表class中，有ID为1的记录。

第 2 种：约束父表

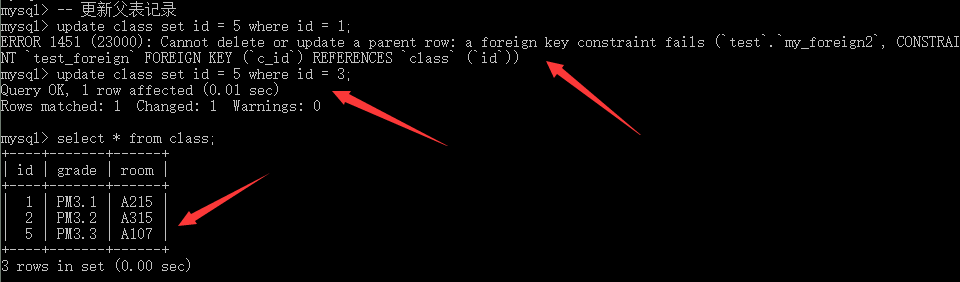
在父表进行数据的写操作（删和改，且涉及主键）的时候，如果对应的主键字段在子表已经被数据引用，那么操作就会失败。

执行如下 SQL 语句，进行测试：

-- 更新父表记录

update class set id = 5 where id = 1;

update class set id = 5 where id = 3;

[](https://camo.githubusercontent.com/2cdcaccdac64bbdaabe93f9559f5f80ec3be74cc/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730383230313130313330383335)

如上图所示，在我们修改父表class中ID为1的时候，提示修改失败，原因就是在子表my\_foreign2中已经引用了该值的主键字段。而在我们修改父表class中ID为3的时候，提示修改成功，原因就是在子表my\_foreign2中并没有引用该值的主键字段。

外键条件

在我们使用外键的时候，应该遵循如下条件：

外键要存在，首先必须保证表的引擎是 InnoDB（默认的存储引擎），如果不是 InnoDB 存储引擎，那么外键可以创建成功，但没有约束作用；

外键字段的字段类型（列类型），必须与父表的主键类型完全一致；

每张表中的外键名称不能重复；

增加外键的字段，如果数据已经存在，那么要保证数据与父表中的主键对应。

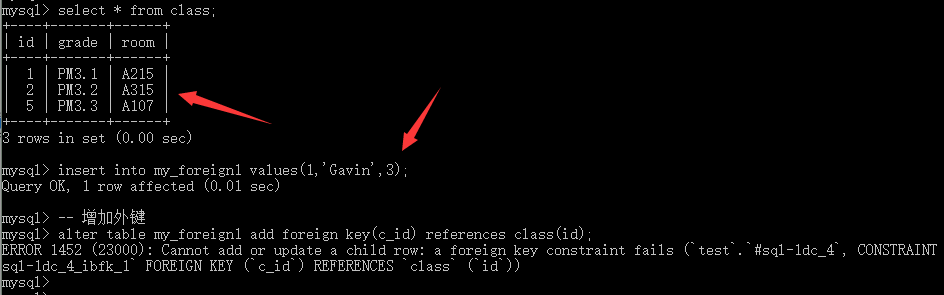
下面以最后一个条件为例，执行如下 SQL 语句，进行测试：

-- 新增数据

insert into my\_foreign1 valuse(1,'Gavin',3);

-- 增加外键

alter table my\_foreign1 add foreign key(c\_id) references class(id);

[](https://camo.githubusercontent.com/21128e94ebdb7a8c41df8c6f75fd55b33063c1b6/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730383230313132383139313739)

如上图所示，在新增外键的时候，如果子表中（想要新增外键的字段）的数据已经存在，而父表中又没有与子表中（想要新增外键的字段）的数据相匹配的主键的话，那么操作就会失败；反之，则会成功。

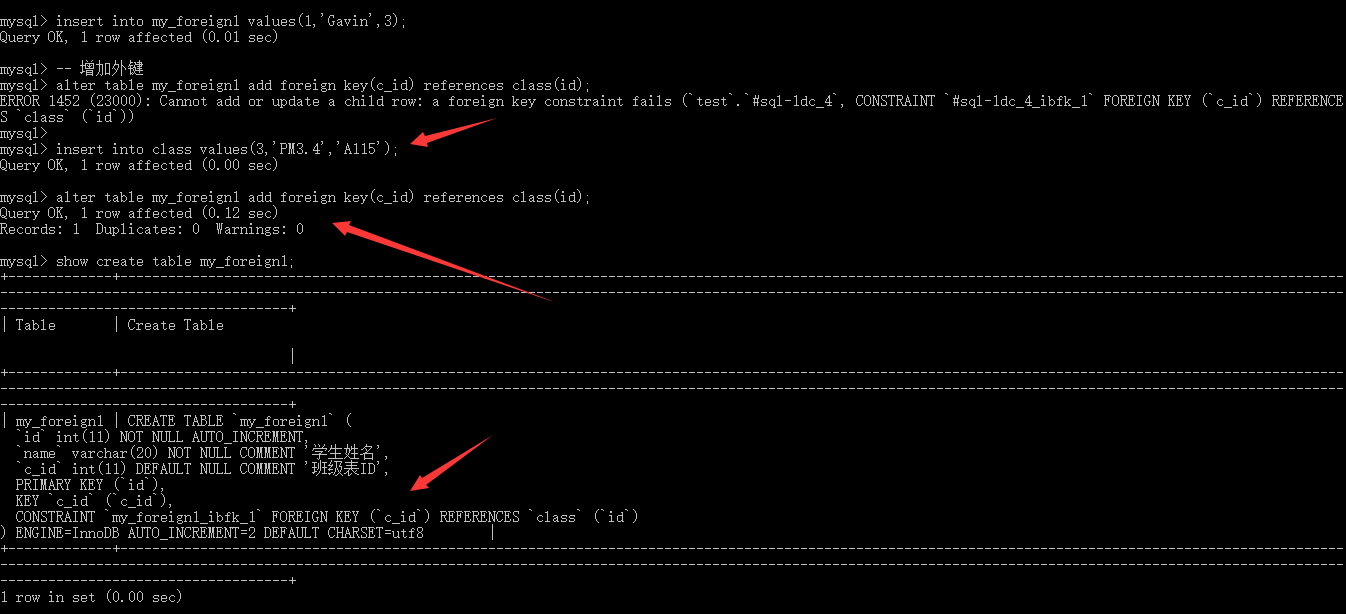
执行如下 SQL 语句，进行测试：

-- 新增数据

insert into class valuse(3,'PM3.4','A115');

-- 增加外键

alter table my\_foreign1 add foreign key(c\_id) references class(id);

[](https://camo.githubusercontent.com/30cc878e49b079d0b32c92c2dfa37ca260e80cc6/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730383230313133363338343138)

如上图所示，显然当父表中存在与子表中（想要新增外键的字段）的数据相匹配的主键的话，增加主键的操作就会成功。

外键约束

所谓外键约束，就是指外键的作用。之前所讲的外键的作用都是默认的作用，实际上，可以通过对外键的需求，进行定制操作。

外键约束有三种模式，分别为：

district：严格模式（默认），父表不能删除或更新一个已经被子表数据引用的记录；

cascade：级联模式，父表的操作，对应子表关联的数据也跟着被删除；

set null：置空模式，父表的操作之后，子表对应的数据（外键字段）被置空。

在此需要注意：以上三种模式，都是对父表的约束。

基本语法：foreign key(外键字段) + references + 父表(主键字段) + [on delete + 模式 + on update + 模式];

通常一个合理的做法（约束模式）是：删除的时候， 子表被置空；更新的时候，子表进行级联操作。

执行如下 SQL 语句，进行测试：

-- 创建外键，指定模式：删除置空，更新级联

create table my\_foreign3(

id int primary key auto\_increment,

name varchar(20) not null,

c\_id int,

-- 增加外键

foreign key(c\_id)

-- 引用父表

references class(id)

-- 指定删除模式

on delete set null

-- 指定更新模式

on update cascade

)charset utf8;

[](https://camo.githubusercontent.com/a30d51552f28e2a7d156097041b3ef4aab15a488/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730383230313232373132373334)

如上图所示，在我们指定外键的约束模式之后，通过查看表的创建语句，可以看到具体的约束语句。

接下来，执行如下 SQL 语句，继续进行测试：

-- 插入数据

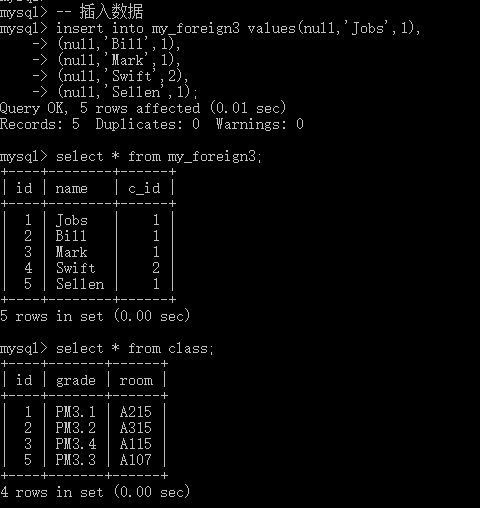
insert into my\_foreign3 values(null,'Jobs',1),

(null,'Bill',1),

(null,'Mark',1),

(null,'Swift',2),

(null,'Sellen',1);

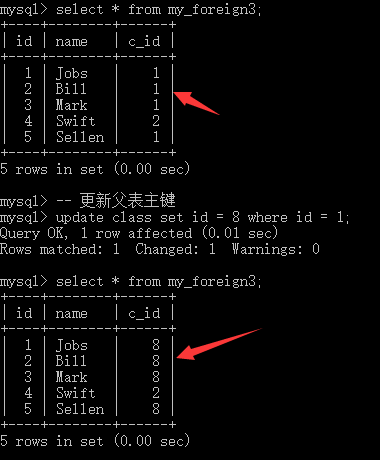
[](https://camo.githubusercontent.com/80760dbff7fb8431b7e716cc6d861d19b8350499/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730383230313233343231353137)

如上图所示，我们向表my\_foreign3中插入了 5 条记录。接下来，我们就可以测试外键的级联模式和置空模式啦！呃，对啦，前提是我们需要把与父表class相关联的除my\_foreign3之外的其他子表，也就是my\_foreign1和my\_foreign2的外键删除掉，否则的话，由于这两个子表的外键使用了严格模式，会干扰我们接下来的测试。

在我们删除表my\_foreign1和my\_foreign2的外键之后，执行如下 SQL 语句，测试级联模式：

-- 更新父表主键

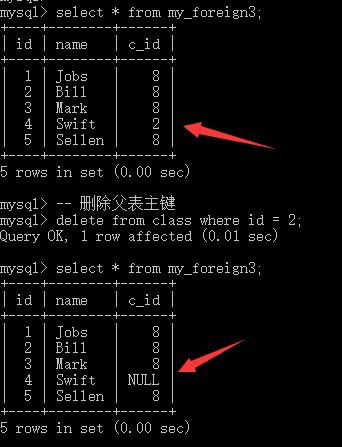
update class set id = 8 where id = 1;

[](https://camo.githubusercontent.com/2bd34053bb608637de99d6db519a7c0343e78bfe/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730383230313234363033363131)

执行如下 SQL 语句，测试置空模式：

-- 删除父表主键

delete from class where id = 2;

[](https://camo.githubusercontent.com/f953835bf16e7174d057291c8a12beaf27404f45/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730383230313234393134333230)

通过以上测试，我们已经验证了级联模式和置空模式的效果。其实，在我们进行删除置空操作的时候，有一个前提，那就是：子表的外键字段必须允许为空，否则的话，操作是无法成功的。

至此，我们已经把外键的相关操作都演示了一遍。在这里，我们会发现外键的功能非常强大，能够进行各种的约束，也正是由于外键这种约束的强大，其降低了开发语言对数据的可控性，因此在实际的开发中，很少使用外键来处理数据。