视图：view，是一种有结构（有行有列），但没有结果（结构中不真实存放数据）的虚拟表，虚拟表的结构来源不是自己定义的，而是从对应的基表（视图的数据来源）中产生的。

创建视图

首先，给出创建视图的基本语法，

基本语法：create view + 视图名 + as + select语句;

其中，select语句可以是普通查询，也可以是连接查询、联合查询、子查询等。

此外，视图根据数据的来源，可以分为单表视图和多表视图：

单表视图：基表只有一个；

多表视图：基表至少两个。

执行如下 SQL 语句，进行测试：

-- 单表视图

create view my\_v1 as select \* from student;

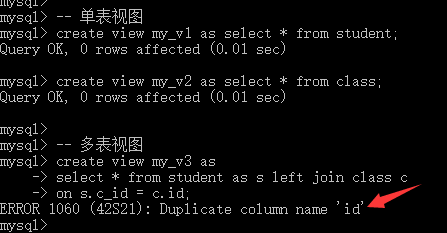
create view my\_v2 as select \* from class;

-- 多表视图

create view my\_v3 as

select \* from student as s left join class c

on s.c\_id = c.id;

[](https://camo.githubusercontent.com/469912742e80821389b16059ac4c8479bbdfb141/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730393033313733383335343834)

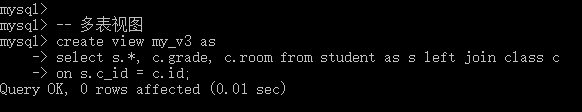
如上图所示，在我们创建多表视图的时候，由于表student和class中都含有id字段，因此导致出现错误“重复列名”。修改上述创建多表视图的 SQL 语句，继续进行测试：

-- 多表视图

create view my\_v3 as

select s.\*, c.grade, c.room from student as s left join class c

on s.c\_id = c.id;

[](https://camo.githubusercontent.com/dbee2bd3f4b747c4087a6a8fd7e5cf52d106af5a/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730393033313734323332373735)

如上图所示，当我们去掉表class中的id字段之后，成功创建多表视图。这是因为，当视图的基表有多张的时候，字段名不能重复。

查询视图

在这里，查询视图是指查看视图的结构，而不是查看视图的结果。

由于视图是一张虚拟表，因此表的所用查询语句，都适用于视图，例如：

desc + 视图名;

show tables + 视图名;

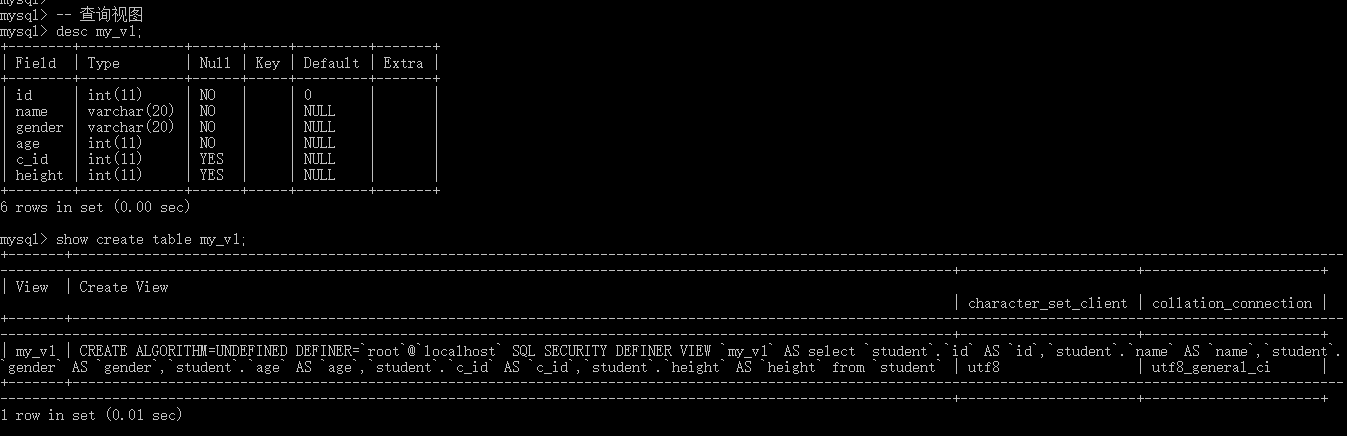
show create table + 视图名;

执行如下 SQL 语句，进行测试：

-- 查询视图

desc my\_v1;

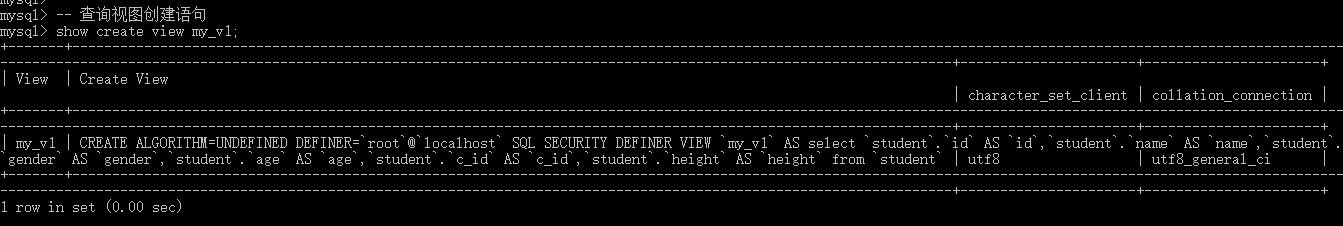
show create table my\_v1;

[](https://camo.githubusercontent.com/7b827016b4006f97901fddd176e82cd87ff19499/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730393033313735393132333136)

虽然视图是虚拟表，但它和真正的表，至少在关键字上还是有区别的，因此在查询视图创建语句的时候，可以使用如下 SQL 语句：

-- 查询视图创建语句

show create view my\_v1;

[](https://camo.githubusercontent.com/33e49e2ba358bf74299796d7b808361f66e9b577/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730393033313830323337323637)

此外，视图一旦创建，系统就会在视图对应的数据库文件夹下创建一个对应的frm结构文件，以保证结构的完整性。

使用视图

在操作数据库表的过程中，使用视图，主要就是为了查询，因此将视图当做表一样查询即可。

在这里需要注意的是，虽然我们说视图是一个虚拟表，它不保存数据，但是它却可以获取数据。

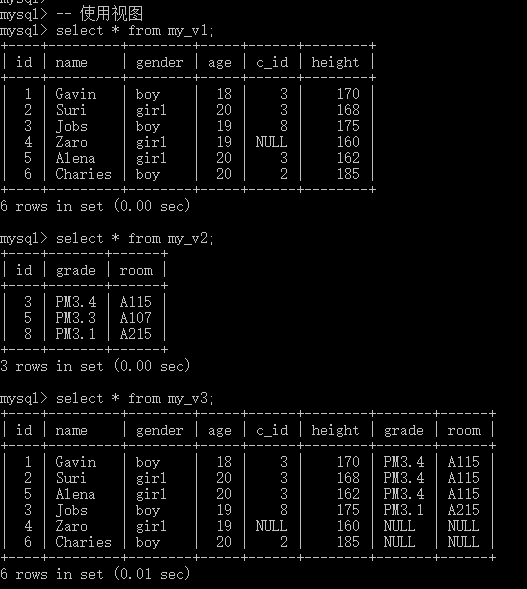
执行如下 SQL 语句，进行测试：

-- 使用视图

select \* from my\_v1;

select \* from my\_v2;

select \* from my\_v3;

[](https://camo.githubusercontent.com/104beb23a84e58a9f13ddc7f559819a2dcbf0590/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730393033313831333239313836)

如上图所示，我们查询视图的结果和查询创建视图时as后面连接的select语句的结果完全相同。

因此，我们也可以认为：创建视图，就是给一条select语句起别名，或者说是封装select语句。

修改视图

视图本身不可修改，但是视图的来源（select）语句是可以修改的。因此，修改视图，就是修改视图的来源（select）语句。

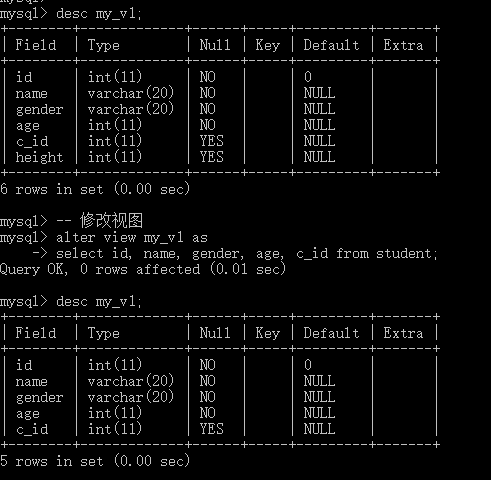
基本语法：alter view + 视图名 + as + 新的select语句;

执行如下 SQL 语句，进行测试：

-- 修改视图

alter view my\_v1 as

select id, name, gender, age, c\_id from student;

[](https://camo.githubusercontent.com/483014967ab847e5ffbb57089a0cf8daf11d09f7/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730393033313832383332333134)

删除视图

与视图的其他操作相比，删除视图比较简单，

基本语法：drop view + 视图名;

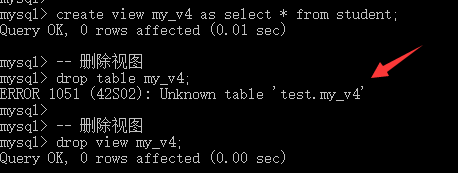
执行如下 SQL 语句，进行测试：

-- 删除视图

drop table my\_v4;

-- 删除视图

drop view my\_v4;

[](https://camo.githubusercontent.com/57f908f22e6bf9afb112c7a4947028ffacc7eafb/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313730393033313835353136323738)

如上图所示，我们不能用

drop table + 视图名;

来删除视图，因为table包含真实的数据，而view说到底就是封装的select语句，并不包含真实的数据。虽然删除视图并不会影响数据，但在实际工作中，建议还是不要乱删别人建立的视图，因为视图封装的select语句很有可能包含复杂的业务逻辑。

视图意义

视图可以节省 SQL 语句，将一条复杂的查询语句用视图来进行封装，以后可以直接对视图进行操作；

数据安全，视图操作主要是针对查询的，如果对视图结构进行处理，例如删除，并不会影响基表的数据；

视图往往在大型项目中使用，而且是多系统使用，可以对外提供有用的数据，但是隐藏关键（或无用）的数据；

视图是对外提供友好型的，不同的视图提供不同的数据，就如专门对外设计的一样；

视图可以更好（或者说，容易）的进行权限控制。