变量

在 MySQL 数据库中，变量有两种，分别为：系统变量和自定义变量。

根据变量的作用范围，又分为：

会话级别变量：仅对当前客户端当次连接有效；

全局级别变量：对所有客户端的任一次连接都有效。

系统变量

系统变量，顾名思义，是系统设置好的变量（皆为全局级别变量），也是用来控制服务器表现的，如autocommit、wait\_timeout等。

大多数的时候，我们并不需要使用系统变量，但我们仍然需要了解有这么回事，在必须要的时候，它可以帮助我们完成特殊的需求。

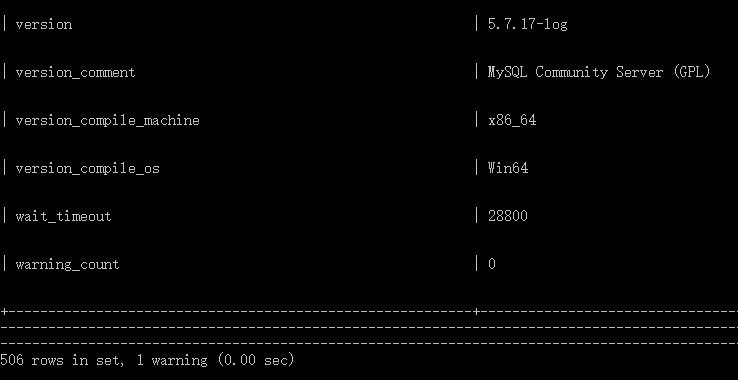
首先，查看系统变量，语法为：

基本语法：show variables;

执行如下 SQL 语句，进行测试：

-- 查看系统变量

show variables;

[](https://camo.githubusercontent.com/ea015e44c613bfecc2cfd45c918de71360db0e7e/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731323033313933353535343137)

如上图所示，显示了 MySQL 在本服务器上共含有 506 个系统变量。更近一步，我们可以查看具体的系统变量的值，语法为：

基本语法：select + @@变量名 + [, @@变量名, ... , @@变量名];

执行如下 SQL 语句，进行测试：

-- 查看具体的系统变量的值

select @@autocommit,@@version,@@version\_compile\_os,@@wait\_timeout;

[](https://camo.githubusercontent.com/1334af749fe25e2cde09616dc9a1337842016dda/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731323033313934323137383035)

如上图所示，我们查到了具体的变量的值。此外，任何一个有内容返回的查询操作都是用select来完成的。

接下来，我们尝试修改系统变量，先修改会话级别变量，再修改全局级别变量。

对于修改会话级别变量，有两种方法，语法分别为：

基本语法 1：set 变量名 = 值;

基本语法 2：set @@变量名 = 值;

执行如下 SQL 语句，进行测试：

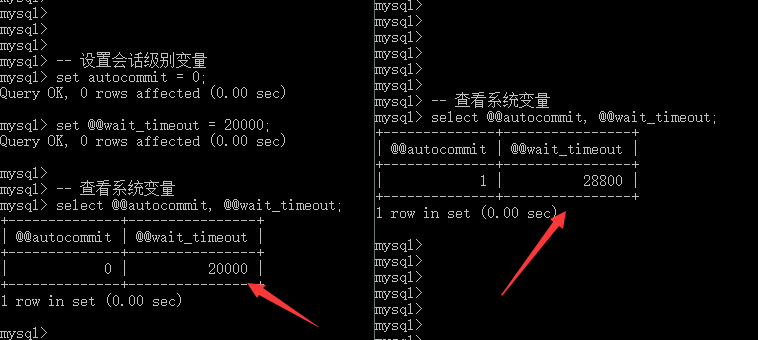
-- 设置会话级别变量

set autocommit = 0;

set @@wait\_timeout = 20000;

-- 查看系统变量

select @@autocommit, @@wait\_timeout;

[](https://camo.githubusercontent.com/2c7a3d7807ec43aeb898d217c39137988df9bbfc/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731323033323030323033323834)

如上图所示，我们修改了autocommit和wait\_timeout的值，但仅作用于会话级别，即只有当前当次连接有效，当再次打开一个新窗口的时候，我们会发现所有的变量值都恢复如初。

对于修改全局级别变量，语法为：

基本语法：set global 变量名 = 值;

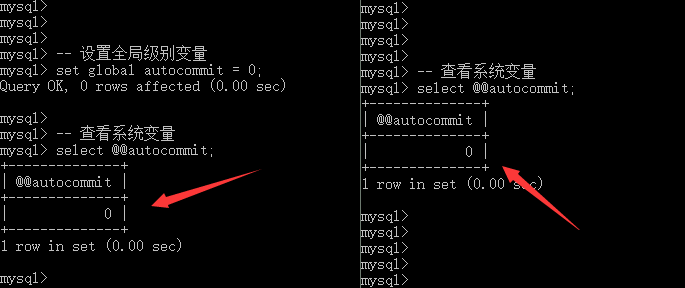
执行如下 SQL 语句，进行测试：

-- 设置全局级别变量

set global autocommit = 0;

-- 查看系统变量

select @@autocommit;

[](https://camo.githubusercontent.com/b2c93e21ec221de1c7f070a02dd9f7e3aa6b40c6/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731323033323031313339383138)

如上图所示，当我们修改全局变量的时候，其效果对所有客户端的任一次连接都有效。But，如果某一个客户端在我们修改全局变量之前已经连上了服务器并且没有退出的话，那么我们的修改对其当前当次连接无效，需要重新登录才能生效。

自定义变量

自定义变量，顾名思义，是用户自己定义的变量，并且都是会话级别的变量。

系统为了区别系统变量与自定义变量，规定用户自定义的变量必须使用一个@符号。设置自定义变量的语法为：

基本语法：set @变量名 = 值;

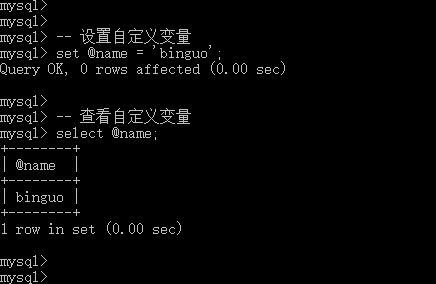
执行如下 SQL 语句，进行测试：

-- 设置自定义变量

set @name = 'binguo';

-- 查看自定义变量

select @name;

[](https://camo.githubusercontent.com/c251b10fa928a386391ffbcccd587725046cf1d9/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731323033323032363331373936)

观察上图，我们会发现查看自定义变量和系统变量有些细微的区别，那就是：查看系统变量时，select后面是跟着@@的，而查看自定义变量时，select后面是跟着@的。在这里，我们需要注意：在 MySQL 中，很多地方会默认将=处理为比较符号，因此 MySQL 还提供了另外一种赋值符号:=，即冒号与等号拼接而成的符号。

此外，MySQL 允许我们从数据表中获取数据，然后直接赋值给变量，共有两种方式，分别为：

第 1 种：边赋值，边查看结果。语法为

基本语法：select @变量名 := 字段名 from 表名;

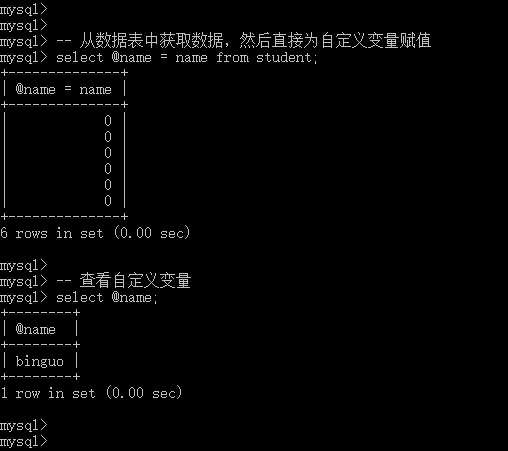
执行如下 SQL 语句，进行测试：

-- 从数据表中获取数据，然后直接为自定义变量赋值

select @name = name from student;

-- 查看自定义变量

select @name;

[](https://camo.githubusercontent.com/bfada9d687e07541956dbc3687919f485eacae0c/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731323033323033383333363334)

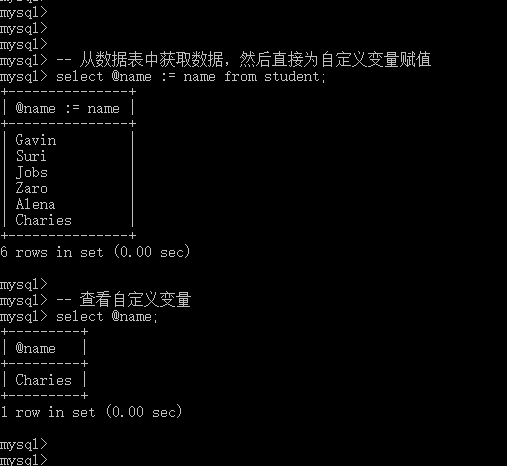
如上图所示，呃，这是什么鬼？好吧，细心的同学估计已经发现了，在上面的select语句中，我们误将:=写为=啦，然后 MySQL 将=处理为比较符号，并且在student表中没有发现与binguo匹配的名字，因此显示的结果皆为0，如果匹配成功，则会显示1。下面，我们修改赋值符号，重新进行测试：

-- 从数据表中获取数据，然后直接为自定义变量赋值

select @name := name from student;

-- 查看自定义变量

select @name;

[](https://camo.githubusercontent.com/26ac0df82ef6629d0770dcad16559ae4733780a4/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731323033323034343130313833)

如上图所示，我们会发现上述select语句的作用为：从student表读取数据，然后依次赋值给自定义变量@name，并且先赋的值会被覆盖，仅保留最后一个赋值结果。

第 2 种：只赋值，不查看结果。语法为

基本语法：select + 字段列表 + from + 表名 + into + 变量列表;

执行如下 SQL 语句，进行测试：

-- 从数据表中获取数据，然后直接为自定义变量赋值

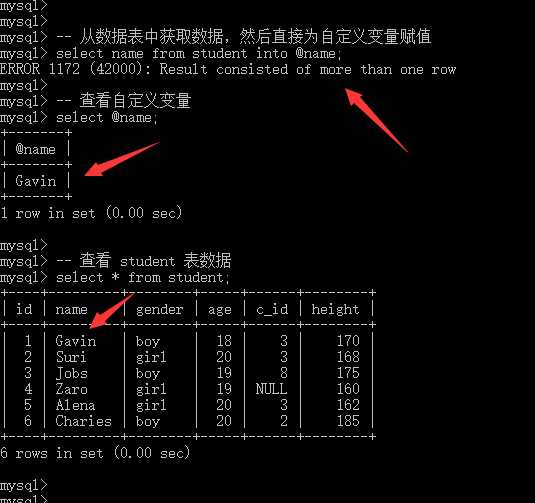
select name from student into @name;

-- 查看自定义变量

select @name;

-- 查看 student 表数据

select \* from student;

[](https://camo.githubusercontent.com/f4081f9de62f0d9ae8ad01f249a887d7eda8fe49/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731323033323131313038363838)

如上图所示，显然EEROR，内容为：返回结果包含的内容超过一列。实际上，在未加限制条件的情况下，我们直接从表中捞取数据，是捞取全部数据，因此忽略上述 SQL 语句中的into @name，其返回的结果为表中的全部name值，自然是超过一个了。在这种情况下，系统会报错，却将捞取数据的第一个值赋值给了@name，也就是说，在捞取数据超过一条记录的时候，系统会默认将第一个值赋值给自定义变量。

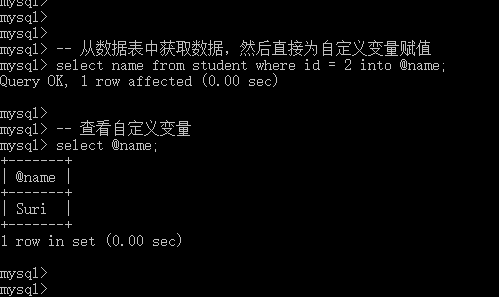
虽然上述 SQL 语句修改了@name的值，但却是一种错误的赋值方式，也是不可控的，其结果往往并不是我们想要的。对于上述的赋值方式，MySQL 的要求比较严格，规定每次只能获取一条记录。因此正确的做法是，加上一个where条件，将查询的结果限制为一条，例如

-- 从数据表中获取数据，然后直接为自定义变量赋值

select name from student where id = 2 into @name;

-- 查看自定义变量

select @name;

[](https://camo.githubusercontent.com/16cf750ee7407a2c30788f05bc589fbddf14fa91/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313731323033323035393339363439)

如上图所示，我们获取数据并赋值成功。

最后，在强调一点：自定义变量都是会话级别，只要是当前用户当次连接，都会受到影响，不区分数据库。