启动选项和配置文件

标签： MySQL 是怎样运行的

如果你用过手机，你的手机上一定有一个设置的功能，你可以选择设置手机的来电铃声、设置音量大小、设置解锁密码等等。假如没有这些设置功能，我们的生活将置于尴尬的境地，比如在图书馆里无法把手机设置为静音，无法把流量开关关掉以节省流量，在别人得知解锁密码后无法更改密码～ MySQL的服务器程序和客户端程序也有很多设置项，比如对于MySQL服务器程序，我们可以指定诸如允许同时连入的客户端数量、客户端和服务器通信方式、表的默认存储引擎、查询缓存的大小吧啦吧啦的设置项。对于MySQL客户端程序，我们之前已经见识过了，可以指定需要连接的服务器程序所在主机的主机名或IP地址、用户名及密码等信息。

这些设置项一般都有各自的默认值，比方说服务器允许同时连入的客户端的默认数量是151，表的默认存储引擎是InnoDB，我们可以在程序启动的时候去修改这些默认值，对于这种在程序启动时指定的设置项也称之为启动选项（startup options），这些选项控制着程序启动后的行为。在MySQL安装目录下的bin目录中的各种可执行文件，不论是服务器相关的程序（比如mysqld、mysqld_safe）还是客户端相关的程序（比如mysql、mysqladmin），在启动的时候基本都可以指定启动参数。这些启动参数可以放在命令行中指定，也可以把它们放在配置文件中指定。下边我们会以mysqld为例，来详细唠叨指定启动选项的格式。需要注意的一点是，我们现在要唠叨的是设置启动选项的方式，下边出现的启动选项不论大家认不认识，先不用去纠结每个选项具体的作用是啥，之后我们会对一些重要的启动选项详细唠叨。

在命令行上使用选项

如果我们在启动客户端程序时在-h参数后边紧跟服务器的IP地址，这就意味着客户端和服务器之间需要通过TCP/IP网络进行通信。因为我的客户端程序和服务器程序都装在一台计算机上，所以在使用客户端程序连接服务器程序时指定的主机名是127.0.0.1的情况下，客户端进程和服务器进程之间会使用TCP/IP网络进行通信。如果我们在启动服务器程序的时候就禁止各客户端使用TCP/IP网络进行通信，可以在启动服务器程序的命令行里添加skip-networking启动选项，就像这样：

```
mysqld --skip-networking
```

可以看到，我们在命令行中指定启动选项时需要在选项名前加上--前缀。另外，如果选项名是由多个单词构成的，它们之间可以由短划线-连接起来，也可以使用下划线_连接起来，也就是说skip-networking和skip_networking表示的含义是相同的。所以上边的写法与下边的写法是等价的：

```
mysqld --skip_networking
```

在按照上述命令启动服务器程序后，如果我们再使用mysql来启动客户端程序时，再把服务器主机名指定为127.0.0.1（IP地址的形式）的话会显示连接失败：

```
 mysql -h127.0.0.1 -uroot -p
Enter password:

ERROR 2003 (HY000): Can't connect to MySQL server
on '127.0.0.1' (61)
```

这就意味着我们指定的启动选项skip-networking生效了！

再举一个例子，我们前边说过如果在创建表的语句中没有显式指定表的存储引擎的话，那就会默认使用InnoDB作为表的存储引擎。如果我们想改变表的默认存储引擎的话，可以这样写启动服务器的命令

行：

```
mysqld --default-storage-engine=MyISAM
```

我们现在就已经把表的默认存储引擎改为MyISAM了，在客户端程序连接到服务器程序后试着创建一个表：

```
mysql> CREATE TABLE sys_var_demo(
    ->     i INT
    -> );
Query OK, 0 rows affected (0.02 sec)
```

这个定义语句中我们并没有明确指定表的存储引擎，创建成功后再看一下这个表的结构：

```
mysql> SHOW CREATE TABLE sys_var_demo\G
*************************** 1. row
***************************
       Table: sys_var_demo
Create Table: CREATE TABLE `sys_var_demo` (
  `i` int(11) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8
1 row in set (0.01 sec)
```

可以看到该表的存储引擎已经是MyISAM了，说明启动选项default-storage-engine生效了。

所以在启动服务器程序的命令行后边指定启动选项的通用格式就是这样的：

```
--启动选项1[=值1] --启动选项2[=值2] ... --启动选项n[=值n]
```

也就是说我们可以将各个启动选项写到一行中，各个启动选项之间使用空白字符隔开，在每一个启动选项名称前边添加--。对于不需要值的启动选项，比方说skip-networking，它们就不需要指定对应的值。对于需要指定值的启动选项，比如default-storage-engine我们在指定这个设置项的时候需要显式的指定它的值，比方说InnoDB、MyISAM啦什么的～ 在命令行上指定有值的启动选项时需要注意，选项名、=、选项值之间不可以有空白字符，比如写成下边这样就是不正确的：

```
mysqld --default-storage-engine = MyISAM
```

每个MySQL程序都有许多不同的选项。大多数程序提供了一个--help选项，你可以查看该程序支持的全部启动选项以及它们的默认值。例如，使用mysql --help可以看到mysql程序支持的启动选项，mysqld_safe --help可以看到mysqld_safe程序支持的启动选项。查看mysqld支持的启动选项有些特别，需要使用mysqld --verbose --help。

选项的长形式和短形式

我们前边提到的skip-networking、default-storage-engine称之为长形式的选项（因为它们很长），设计MySQL的大叔为了我们使用的方便，对于一些常用的选项提供了短形式，我们列举一些具有短形式的启动选项来瞅瞅（MySQL支持的短形式选项太多了，全列出来会刷屏的）：

长形式	短形式	含义
--host	-h	主机名
--user	-u	用户名
--password	-p	密码
--port	-P	端口
--version	-V	版本信息

短形式的选项名只有一个字母，与使用长形式选项时需要在选项名前加两个短划线--不同的是，使用短形式选项时在选项名前只加一个短划线-前缀。有一些短形式的选项我们之前已经接触过了，比方说我们在启动服务器程序时指定监听的端口号：

```
mysqld -P3307
```

使用短形式指定启动选项时，选项名和选项值之间可以没有间隙，或者用空白字符隔开（-p选项有些特殊，-p和密码值之间不能有空白字符），也就是说上边的命令形式和下边的是等价的：

```
mysqld -P 3307
```

另外，选项名是区分大小写的，比如-p和-P选项拥有完全不同的含义，大家需要注意一下。

配置文件中使用选项

在命令行中设置启动选项只对当次启动生效，也就是说如果下一次重启程序的时候我们还想保留这些启动选项的话，还得重复把这些选项写到启动命令行中，这样真的神烦唉！于是设计MySQL的大叔们提出一种配置文件（也称为选项文件）的概念，我们把需要设置的启动选项都写在这个配置文件中，每次启动服务器的时候都从这个文件里加载相应的启动选项。由于这个配置文件可以长久的保存在计算机的硬盘里，所以只需我们配置一次，以后就都不用显式的把启动选项都写在启动命令行中了，所以我们推荐使用配置文件的方式来设置启动选项。

配置文件的路径

MySQL程序在启动时会寻找多个路径下的配置文件，这些路径有的是固定的，有的是可以在命令行指定的。根据操作系统的不同，配置文件的路径也有所不同，我们分开看一下。

Windows操作系统的配置文件

在Windows操作系统中，MySQL会按照下列路径来寻找配置文件：

路径名	备注
%WINDIR%\my.ini, %WINDIR%\my.cnf	
C:\my.ini, C:\my.cnf	
BASEDIR\my.ini, BASEDIR\my.cnf	
defaults-extra-file	命令行指定的额外配置文件路径
%APPDATA%\MySQL\.mylogin.cnf	登录路径选项（仅限客户端）

在阅读这些Windows操作系统下配置文件路径的时候需要注意一些事情：

- 在给定的前三个路径中，配置文件可以使用.ini的扩展名，也可以使用.cnf的扩展名。
- %WINDIR%指的是你机器上Windows目录的位置，通常是C:\WINDOWS，如果你不确定，可以使用这个命令来查看：

  ```
  echo %WINDIR%
  ```

- BASEDIR指的是MySQL安装目录的路径，在我的Windows机器上的BASEDIR的值是：

  ```
  C:\Program Files\MySQL\MySQL Server 5.7
  ```

- 第四个路径指的是我们在启动程序时可以通过指定defaults-extra-file参数的值来添加额外的配置文件路径，比方说我们在命令行上可以这么写：

```
mysqld --defaults-extra-
file=C:\Users\xiaohaizi\my_extra_file.txt
```

这样MySQL服务器启动时就可以额外在C:\Users\xiaohaizi\my_extra_file.txt这个路径下查找配置文件。

- %APPDATA%表示Windows应用程序数据目录的值，可以使用下列命令查看：

```
echo %APPDATA%
```

- 列表中最后一个名为.mylogin.cnf配置文件有点儿特殊，它不是一个纯文本文件（其他的配置文件都是纯文本文件），而是使用mysql_config_editor实用程序创建的加密文件。文件中只能包含一些用于启动客户端软件时连接服务器的一些选项，包括 host、user、password、port和 socket。而且它只能被客户端程序所使用。

小贴士：

mysql_config_editor实用程序其实是MySQL安装目录下的bin目录下的一个可执行文件，这个实用程序有专用的语法来生成或修改 .mylogin.cnf 文件中的内容，如何使用这个程序不是我们讨论的主题，可以到MySQL的官方文档中查看。

类Unix操作系统中的配置文件

在类UNIX操作系统中，MySQL会按照下列路径来寻找配置文件：

路径名	备注
/etc/my.cnf	
/etc/mysql/my.cnf	
SYSCONFDIR/my.cnf	

$MYSQL_HOME/my.cnf	特定于服务器的选项（仅限服务器）
defaults-extra-file	命令行指定的额外配置文件路径
~/.my.cnf	用户特定选项
~/.mylogin.cnf	用户特定的登录路径选项（仅限客户端）

在阅读这些UNIX操作系统下配置文件路径的时候需要注意一些事情：

- SYSCONFDIR表示在使用CMake构建MySQL时使用SYSCONFDIR选项指定的目录。默认情况下，这是位于编译安装目录下的etc目录。

  > 小贴士：
  >
  > 如果你不懂啥是个CMAKE，啥是个编译，那就跳过吧，对我们后续的文章没啥影响。

- MYSQL_HOME是一个环境变量，该变量的值是我们自己设置的，我们想设置就设置，不想设置就不设置。该变量的值代表一个路径，我们可以在该路径下创建一个my.cnf配置文件，那么这个配置文件中只能放置关于启动服务器程序相关的选项（言外之意就是其他的配置文件既能存放服务器相关的选项也能存放客户端相关的选项，.mylogin.cnf除外，它只能存放客户端相关的一些选项）。

> 小贴士：
>
> 如果大家使用mysqld_safe启动服务器程序，而且我们也没有主动设置这个MySQL_HOME环境变量的值，那这个环境变量的值将自动被设置为MySQL的安装目录，也就是MySQL服务器将会在安装目录下查找名为my.cnf配置文件（别忘了mysql.server会调用mysqld_safe，所以使用mysql.server启动服务器时也会在安装目录下查找配置文件）。

- 列表中的最后两个以~开头的路径是用户相关的，类UNIX系统中都有一个当前登陆用户的概念，每个用户都可以有一个用户目录，~就代表这个用户目录，大家可以查看HOME环境变量的值来确定一下当前用户的用户目录，比方说我的macOS机器上的用户目录就是/Users/xiaohaizi。之所以说列表中最后两个配置文件是用户相关的，是因为不同的类UNIX系统的用户都可以在自己的用户目录下创建.my.cnf或者.mylogin.cnf，换句话说，不同登录用户使用的.my.cnf或者.mylogin.cnf配置文件是不同的。

- defaults-extra-file的含义与Windows中的一样。

- .mylogin.cnf的含义也同Windows中的一样，再次强调一遍，它不是纯文本文件，只能使用mysql_config_editor实用程序去创建或修改，用于存放客户端登陆服务器时的相关选项。

这也就是说，在我的计算机中这几个路径中的任意一个都可以当作配置文件来使用，如果它们不存在，你可以手动创建一个，比方说我手动在~/.my.cnf这个路径下创建一个配置文件。

另外，我们在唠叨如何启动MySQL服务器程序的时候说过，使用mysqld_safe程序启动服务器时，会间接调用mysqld，所以对于传递给mysqld_safe的启动选项来说，如果mysqld_safe程序

不处理，会接着传递给mysqld程序处理。比方说skip-networking选项是由mysqld处理的，mysqld_safe并不处理，但是如果我们我们在命令行上这样执行：

```
mysqld_safe --skip-networking
```

则在mysqld_safe调用mysqld时，会把它处理不了的这个skip-networking选项交给mysqld处理。

配置文件的内容

与在命令行中指定启动选项不同的是，配置文件中的启动选项被划分为若干个组，每个组有一个组名，用中括号[]扩起来，像这样：

```
[server]
(具体的启动选项...)

[mysqld]
(具体的启动选项...)

[mysqld_safe]
(具体的启动选项...)

[client]
(具体的启动选项...)

[mysql]
(具体的启动选项...)

[mysqladmin]
(具体的启动选项...)
```

像这个配置文件里就定义了许多个组，组名分别是server、mysqld、mysqld_safe、client、mysql、mysqla
每个组下边可以定义若干个启动选项，我们以[server]组为例来看一下填写启动选项的形式（其他组中启动选项的形式是一样的）：

```
[server]
option1      #这是option1，该选项不需要选项值
option2 = value2       #这是option2，该选项需要选项值
...
```

在配置文件中指定启动选项的语法类似于命令行语法，但是配置文件中只能使用长形式的选项。在配置文件中指定的启动选项不允许加--前缀，并且每行只指定一个选项，而且=周围可以有空白字符（命令行中选项名、=、选项值之间不允许有空白字符）。另外，在配置文件中，我们可以使用#来添加注释，从#出现直到行尾的内容都属于注释内容，读取配置文件时会忽略这些注释内容。为了大家更容易对比启动选项在命令行和配置文件中指定的区别，我们再把命令行中指定option1和option2两个选项的格式写一遍看看：

```
--option1 --option2=value2
```

配置文件中不同的选项组是给不同的启动命令使用的，如果选项组名称与程序名称相同，则组中的选项将专门应用于该程序。例如，[mysqld]和[mysql]组分别应用于mysqld服务器程序和mysql客户端程序。不过有两个选项组比较特别：

- [server]组下边的启动选项将作用于所有的服务器程序。
- [client]组下边的启动选项将作用于所有的客户端程序。

需要注意的一点是，mysqld_safe和mysql.server这两个程序在启动时都会读取[mysqld]选项组中的内容。为了直观感受一下，我们挑一些启动命令来看一下它们能读取的选项组都有哪些：

启动命令	能读取的组

	类别	
mysqld	启动服务器	[mysqld]、[server]
mysqld_safe	启动服务器	[mysqld]、[server]、[mysqld_safe]
mysql.server	启动服务器	[mysqld]、[server]、[mysql.server]
mysql	启动客户端	[mysql]、[client]
mysqladmin	启动客户端	[mysqladmin]、[client]
mysqldump	启动客户端	[mysqldump]、[client]

现在我们以macOS操作系统为例，在/etc/mysql/my.cnf这个配置文件中添加一些内容（Windows系统参考上边提到的配置文件路径）：

```
[server]
skip-networking
default-storage-engine=MyISAM
```

然后直接用mysqld启动服务器程序：

```
mysqld
```

虽然在命令行没有添加启动选项，但是在程序启动的时候，就会默认的到我们上边提到的配置文件路径下查找配置文件，其中就包括/etc/mysql/my.cnf。又由于mysqld命令可以读取[server]选项组的内容，所以skip-networking和default-storage-engine=MyISAM这两个选项是生效的。你可以把这些启动选项放在[client]组里再试试用mysqld启动服务器程序，看一下里边的启动选项生效不（剧透一下，不生效）。

小贴士：

如果我们想指定mysql.server程序的启动参数，则必须将它们放在配置文件中，而不是放在命令行中。mysql.server仅支持start和stop作为命令行参数。

特定MySQL版本的专用选项组

我们可以在选项组的名称后加上特定的MySQL版本号，比如对于[mysqld]选项组来说，我们可以定义一个[mysqld-5.7]的选项组，它的含义和[mysqld]一样，只不过只有版本号为5.7的mysqld程序才能使用这个选项组中的选项。

配置文件的优先级

我们前边唠叨过MySQL将在某些固定的路径下搜索配置文件，我们也可以通过在命令行上指定defaults-extra-file启动选项来指定额外的配置文件路径。MySQL将按照我们在上表中给定的顺序依次读取各个配置文件，如果该文件不存在则忽略。值得注意的是，如果我们在多个配置文件中设置了相同的启动选项，那以最后一个配置文件中的为准。比方说/etc/my.cnf文件的内容是这样的：

```
[server]
default-storage-engine=InnoDB
```

而~/.my.cnf文件中的内容是这样的：

```
[server]
default-storage-engine=MyISAM
```

又因为~/.my.cnf比/etc/my.cnf顺序靠后，所以如果两个配置文件中出现相同的启动选项，以~/.my.cnf中的为准，所以MySQL服务器程序启动之后，default-storage-engine的值就是MyISAM。

同一个配置文件中多个组的优先级

我们说同一个命令可以访问配置文件中的多个组，比如mysqld可以访问[mysqld]、[server]组，如果在同一个配置文件中，比如~/.my.cnf，在这些组里出现了同样的配置项，比如这样：

```
[server]
default-storage-engine=InnoDB

[mysqld]
default-storage-engine=MyISAM
```

那么，将以最后一个出现的组中的启动选项为准，比方说例子中default-storage-engine既出现在[mysqld]组也出现在[server]组，因为[mysqld]组在[server]组后边，就以[mysqld]组中的配置项为准。

defaults-file的使用

如果我们不想让MySQL到默认的路径下搜索配置文件（就是上表中列出的那些），可以在命令行指定defaults-file选项，比如这样（以UNIX系统为例）：

```
mysqld --defaults-file=/tmp/myconfig.txt
```

这样，在程序启动的时候将只在/tmp/myconfig.txt路径下搜索配置文件。如果文件不存在或无法访问，则会发生错误。

> 小贴士：
>
> 注意`defaults-extra-file`和`defaults-file`的区别，使用`defaults-extra-file`可以指定额外的配置文件搜索路径（也就是说那些固定的配置文件路径也会被搜索）。

命令行和配置文件中启动选项的区别

在命令行上指定的绝大部分启动选项都可以放到配置文件中，但是有一些选项是专门为命令行设计的，比方说defaults-extra-file、defaults-file这样的选项本身就是为了指定配置文件路径的，再放在配置文件中使用就没啥意义了。剩下的一些只能用在命令行上而不能用到配置文件中的启动选项就不一一列举了，用到的时候再提哈（本书中基本用不到，有兴趣的到官方文档看哈）。

另外有一点需要特别注意，如果同一个启动选项既出现在命令行中，又出现在配置文件中，那么以命令行中的启动选项为准！比如我们在配置文件中写了：

```
[server]
default-storage-engine=InnoDB
```

而我们的启动命令是：

```
mysql.server start --default-storage-engine=MyISAM
```

那最后default-storage-engine的值就是MyISAM!

系统变量

系统变量简介

MySQL服务器程序运行过程中会用到许多影响程序行为的变量，它们被称为MySQL系统变量，比如允许同时连入的客户端数量用系统变量max_connections表示，表的默认存储引擎用系统变量default_storage_engine表示，查询缓存的大小用系统变量query_cache_size表示，MySQL服务器程序的系统变量有好几百条，我们就不一一列举了。每个系统变量都有一个默认值，我们可以使用命令行或者配置文件中的选项在启动服务器时改变一些系统变量的值。大多数的系统变量的值也可以在程序运行过程中修改，而无需停止并重新启动它。

查看系统变量

我们可以使用下列命令查看MySQL服务器程序支持的系统变量以及它们的当前值：

```
SHOW VARIABLES [LIKE 匹配的模式];
```

由于系统变量实在太多了，如果我们直接使用SHOW VARIABLES查看的话就直接刷屏了，所以通常都会带一个LIKE过滤条件来查看我们需要的系统变量的值，比方说这么写：

```
mysql> SHOW VARIABLES LIKE
'default_storage_engine';
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
1 row in set (0.01 sec)

mysql> SHOW VARIABLES like 'max_connections';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_connections | 151   |
+-----------------+-------+
1 row in set (0.00 sec)
```

可以看到，现在服务器程序使用的默认存储引擎就是InnoDB，允许同时连接的客户端数量最多为151。别忘了LIKE表达式后边可以跟通配符来进行模糊查询，也就是说我们可以这么写：

```
mysql> SHOW VARIABLES LIKE 'default%';
+--------------------------------+----------------
-------+
| Variable_name                  | Value
|
+--------------------------------+----------------
-------+
| default_authentication_plugin |
mysql_native_password |
| default_password_lifetime      | 0
|
| default_storage_engine         | InnoDB
|
| default_tmp_storage_engine     | InnoDB
|
| default_week_format            | 0
|
+--------------------------------+----------------
-------+
5 rows in set (0.01 sec)

mysql>
```

这样就查出了所有以default开头的系统变量的值。

设置系统变量

通过启动选项设置

大部分的系统变量都可以通过启动服务器时传送启动选项的方式来进行设置。如何填写启动选项我们上边已经花了大篇幅来唠叨了，就是下边两种方式：

- 通过命令行添加启动选项。

  比方说我们在启动服务器程序时用这个命令：

  ```
  mysqld --default-storage-engine=MyISAM --max-connections=10
  ```

- 通过配置文件添加启动选项。

  我们可以这样填写配置文件：

  ```
  [server]
  default-storage-engine=MyISAM
  max-connections=10
  ```

当使用上边两种方式中的任意一种启动服务器程序后，我们再来查看一下系统变量的值：

```
mysql> SHOW VARIABLES LIKE
'default_storage_engine';
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | MyISAM |
+------------------------+--------+
1 row in set (0.00 sec)

mysql> SHOW VARIABLES LIKE 'max_connections';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_connections | 10    |
+-----------------+-------+
1 row in set (0.00 sec)

mysql>
```

可以看到default_storage_engine和max_connections这两个系统变量的值已经被修改了。有一点需要注意的是，对于启动选项来说，如果启动选项名由多个单词组成，各个单词之间用短划线-或者下划线_连接起来都可以，但是对应的系统变量之间必须使用下划线_连接起来。

服务器程序运行过程中设置

系统变量比较牛逼的一点就是，对于大部分系统变量来说，它们的值可以在服务器程序运行过程中进行动态修改而无需停止并重启服务器。不过系统变量有作用范围之分，下边详细唠叨下。

设置不同作用范围的系统变量

我们前边说过，多个客户端程序可以同时连接到一个服务器程序。对于同一个系统变量，我们有时想让不同的客户端有不同的值。比方说狗哥使用客户端A，他想让当前客户端对应的默认存储引擎为InnoDB，所以他可以把系统变量default_storage_engine的值设置为InnoDB；猫爷使用客户端B，他想让当前客户端对应的默认存储引擎为MyISAM，所以他可以把系统变量default_storage_engine的值设置为MyISAM。这样可以使狗哥和猫爷的的客户端拥有不同的默认存储引擎，使用时互不影响，十分方便。但是这样各个客户端都私有一份系统变量会产生这么两个问题：

- 有一些系统变量并不是针对单个客户端的，比如允许同时连接到服务器的客户端数量max_connections，查询缓存的大小query_cache_size，这些公有的系统变量让某个客户端私有显然不合适。

- 一个新连接到服务器的客户端对应的系统变量的值该怎么设置？

为了解决这两个问题，设计MySQL的大叔提出了系统变量的作用范围的概念，具体来说作用范围分为这两种：

- GLOBAL：全局变量，影响服务器的整体操作。

- SESSION：会话变量，影响某个客户端连接的操作。（注：SESSION有个别名叫LOCAL）

在服务器启动时，会将每个全局变量初始化为其默认值（可以通过命令行或选项文件中指定的选项更改这些默认值）。然后服务器还为每个连接的客户端维护一组会话变量，客户端的会话变量在连接时使用相应全局变量的当前值初始化。

这话有点儿绕，还是以default_storage_engine举例，在服务器启动时会初始化一个名为default_storage_engine，作用范围为GLOBAL的系统变量。之后每当有一个客户端连接到该服务器

时，服务器都会单独为该客户端分配一个名为default_storage_engine，作用范围为SESSION的系统变量，该作用范围为SESSION的系统变量值按照当前作用范围为GLOBAL的同名系统变量值进行初始化。

很显然，通过启动选项设置的系统变量的作用范围都是GLOBAL的，也就是对所有客户端都有效的，因为在系统启动的时候还没有客户端程序连接进来呢。了解了系统变量的GLOBAL和SESSION作用范围之后，我们再看一下在服务器程序运行期间通过客户端程序设置系统变量的语法：

```
SET [GLOBAL|SESSION] 系统变量名 = 值;
```

或者写成这样也行：

```
SET [@@(GLOBAL|SESSION).]var_name = XXX;
```

比如我们想在服务器运行过程中把作用范围为GLOBAL的系统变量default_storage_engine的值修改为MyISAM，也就是想让之后新连接到服务器的客户端都用MyISAM作为默认的存储引擎，那我们可以选择下边两条语句中的任意一条来进行设置：

```
语句一：SET GLOBAL default_storage_engine = MyISAM;
语句二：SET @@GLOBAL.default_storage_engine = MyISAM;
```

如果只想对本客户端生效，也可以选择下边三条语句中的任意一条来进行设置：

```
语句一：SET SESSION default_storage_engine = MyISAM;
语句二：SET @@SESSION.default_storage_engine = MyISAM;
语句三：SET default_storage_engine = MyISAM;
```

从上边的语句三也可以看出，如果在设置系统变量的语句中省略了作用范围，默认的作用范围就是SESSION。也就是说SET 系统变量名 = 值和SET SESSION 系统变量名 = 值是等价的。

查看不同作用范围的系统变量

既然系统变量有作用范围之分，那我们的SHOW VARIABLES语句查看的是什么作用范围的系统变量呢?

答：默认查看的是SESSION作用范围的系统变量。

当然我们也可以在查看系统变量的语句上加上要查看哪个作用范围的系统变量，就像这样：

```
SHOW [GLOBAL|SESSION] VARIABLES [LIKE 匹配的模式];
```

下边我们演示一下完整的设置并查看系统变量的过程：

```
mysql> SHOW SESSION VARIABLES LIKE
'default_storage_engine';
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
1 row in set (0.00 sec)

mysql> SHOW GLOBAL VARIABLES LIKE
'default_storage_engine';
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
```

```
1 row in set (0.00 sec)

mysql> SET SESSION default_storage_engine =
MyISAM;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW SESSION VARIABLES LIKE
'default_storage_engine';
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | MyISAM |
+------------------------+--------+
1 row in set (0.00 sec)

mysql> SHOW GLOBAL VARIABLES LIKE
'default_storage_engine';
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
1 row in set (0.00 sec)

mysql>
```

可以看到，最初default_storage_engine的系统变量无论是在GLOBAL作用范围上还是在SESSION作用范围上的值都是InnoDB，我们在SESSION作用范围把它的值设置为MyISAM之后，可以看到GLOBAL作用范围的值并没有改变。

小贴士：

如果某个客户端改变了某个系统变量在`GLOBAL`作用范围的值，并不会影响该系统变量在当前已经连接的客户端作用范围为`SESSION`的值，只会影响后续连入的客户端在作用范围为`SESSION`的值。

注意事项

- 并不是所有系统变量都具有GLOBAL和SESSION的作用范围。
    - 有一些系统变量只具有GLOBAL作用范围，比方说max_connections，表示服务器程序支持同时最多有多少个客户端程序进行连接。
    - 有一些系统变量只具有SESSION作用范围，比如insert_id，表示在对某个包含AUTO_INCREMENT列的表进行插入时，该列初始的值。
    - 有一些系统变量的值既具有GLOBAL作用范围，也具有SESSION作用范围，比如我们前边用到的default_storage_engine，而且其实大部分的系统变量都是这样的，
- 有些系统变量是只读的，并不能设置值。

  比方说version，表示当前MySQL的版本，我们客户端是不能设置它的值的，只能在SHOW VARIABLES语句里查看。

启动选项和系统变量的区别

启动选项是在程序启动时我们程序员传递的一些参数，而系统变量是影响服务器程序运行行为的变量，它们之间的关系如下：

- 大部分的系统变量都可以被当作启动选项传入。
- 有些系统变量是在程序运行过程中自动生成的，是不可以当作启动选项来设置，比如auto_increment_offset、character_set_client啥的。
- 有些启动选项也不是系统变量，比如defaults-file。

状态变量

为了让我们更好的了解服务器程序的运行情况，MySQL服务器程序中维护了好多关于程序运行状态的变量，它们被称为状态变量。比方说Threads_connected表示当前有多少客户端与服务器建立了连接，Handler_update表示已经更新了多少行记录吧啦吧啦，像这样显示服务器程序状态信息的状态变量还有好几百个，我们就不一一唠叨了，等遇到了会详细说它们的作用的。

由于状态变量是用来显示服务器程序运行状况的，所以它们的值只能由服务器程序自己来设置，我们程序员是不能设置的。与系统变量类似，状态变量也有GLOBAL和SESSION两个作用范围的，所以查看状态变量的语句可以这么写：

```
SHOW [GLOBAL|SESSION] STATUS [LIKE 匹配的模式];
```

类似的，如果我们不写明作用范围，默认的作用范围是SESSION，比方说这样：

```
mysql> SHOW STATUS LIKE 'thread%';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Threads_cached    | 0     |
| Threads_connected | 1     |
| Threads_created   | 1     |
| Threads_running   | 1     |
+-------------------+-------+
4 rows in set (0.00 sec)

mysql>
```

所有以Thread开头的SESSION作用范围的状态变量就都被展示出来了。