

MySQL（七）

1. 用户权限管理

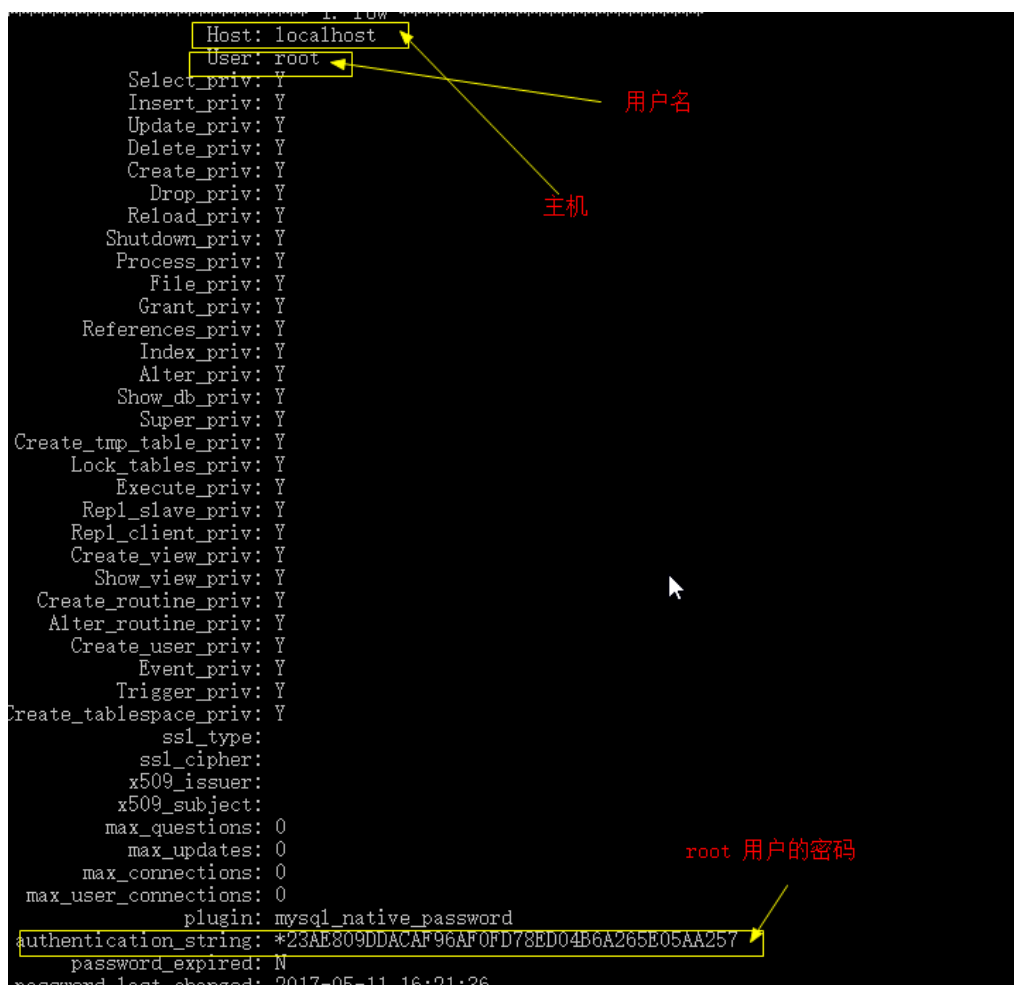
用户权限管理：在不同的项目中给不同的角色（开发者）不同的操作权限，为了保证数据库数据的安全。

通常，一个用户的密码不会长期不变，所以需要经常性的变更数据库用户密码来确保用户本身安全（mysql 客户端用户）

1.1 用户管理

MySQL 需要客户端进行连接认证才能进行服务器操作：需要用户信息。MySQL 中所有的用户信息都是保存在 mysql 数据库下的 user 表中。

```
select * from mysql.user; 可以查看当前数据库的用户
```



```
Host: localhost
User: root
Select_priv: Y
Insert_priv: Y
Update_priv: Y
Delete_priv: Y
Create_priv: Y
Drop_priv: Y
Reload_priv: Y
Shutdown_priv: Y
Process_priv: Y
File_priv: Y
Grant_priv: Y
References_priv: Y
Index_priv: Y
Alter_priv: Y
Show_db_priv: Y
Super_priv: Y
Create_tmp_table_priv: Y
Lock_tables_priv: Y
Execute_priv: Y
Repl_slave_priv: Y
Repl_client_priv: Y
Create_view_priv: Y
Show_view_priv: Y
Create_routine_priv: Y
Alter_routine_priv: Y
Create_user_priv: Y
Event_priv: Y
Trigger_priv: Y
Create_tablespace_priv: Y
ssl_type:
ssl_cipher:
x509_issuer:
x509_subject:
max_questions: 0
max_updates: 0
max_connections: 0
max_user_connections: 0
plugin: mysql_native_password
authentication_string: *23AE809DDACAF96AF0FD78ED04B6A265E05AA257
password_expired: N
password_last_changed: 2017-05-11 16:21:36
```

默认的，在安装 MySQL 的时候，如果不选择创建匿名用户，那么意味着所有的用户只有一个：

root 超级用户

在 mysql 中，对用的用户管理中，是由对应的 Host 和 User 共同组成主键来区分用户。

```
mysql> desc mysql.user;
```

Field	Type	Null	Key	Default	Extra
Host	char(60)	NO	PRI		
User	char(32)	NO	PRI		
Select_priv	enum('N','Y')	NO		N	
Insert_priv	enum('N','Y')	NO		N	
Update_priv	enum('N','Y')	NO		N	
Delete_priv	enum('N','Y')	NO		N	
Create_priv	enum('N','Y')	NO		N	
Drop_priv	enum('N','Y')	NO		N	
Reload_priv	enum('N','Y')	NO		N	
Shutdown_priv	enum('N','Y')	NO		N	
Process_priv	enum('N','Y')	NO		N	
File_priv	enum('N','Y')	NO		N	
Grant_priv	enum('N','Y')	NO		N	
References_priv	enum('N','Y')	NO		N	

复合主键

User: 代表用户的用户名

Host: 代表本质是允许访问的客户端（IP 或者主机地址）。如果 host 使用%代表所有的用户（客户端）都可以访问

1.1.1 创建用户

理论上讲可以采用两种方式创建用户：

- 直接使用 root 用户在 mysql.user 表中插入记录（不推荐）
- 专门创建用户的 SQL 指令

➢ 基本语法：create user 用户名 identified by '明文密码';

用户名：用户名@主机地址

主机地址：'' 或者 '%'、或者 ip 地址

如：create user 'stu1'@'%' identified by '123';

创建成功！

```
mysql> create user 'stu1'@'%' identified by '123';
Query OK, 0 rows affected (0.01 sec)
mysql>
```

查看 mysql.user 表中是否存在新增的用户



```

account_locked: N
***** 3. row *****
      Host: %
      User: stu1
      Select_priv: N
      Insert_priv: N
      Update_priv: N
      Delete_priv: N
      Create_priv: N
      Drop_priv: N
      Reload_priv: N
      Shutdown_priv: N
      Process_priv: N
      File_priv: N
      Grant_priv: N
      References_priv: N
      Index_priv: N
      Alter_priv: N
      Show_db_priv: N
      Super_priv: N
      Create_tmp_table_priv: N
      Lock_tables_priv: N
      Execute_priv: N
      Repl_slave_priv: N
      Repl_client_priv: N
      Create_view_priv: N
      Show_view_priv: N
      Create_routine_priv: N
      Alter_routine_priv: N
      Create_user_priv: N
      Event_priv: N
      Trigger_priv: N
      Create_tablespace_priv: N
      ssl_type:
      ssl_cipher:
      x509_issuer:
      x509_subject:
      max_questions: 0
      max_updates: 0
      max_connections: 0
      max_user_connections: 0
      plugin: mysql_native_password
      authentication_string: *23AE809DDACAF96AF0FD78ED04B6A265E05AA257
      password_expired: N
      password_last_changed: 2017-08-05 17:14:28
      password_lifetime: NULL
      account_locked: N

```

➤ 简化版创建用户（谁都可以访问，不需要密码）

```

mysql> create user user2;
Query OK, 0 rows affected (0.00 sec)

```

1、不限定客户端IP
2、没有密码的用户



```
account_locked: N
***** 4. row *****
Host: %
User: stu2
Select_priv: N
Insert_priv: N
Update_priv: N
Delete_priv: N
Create_priv: N
Drop_priv: N
Reload_priv: N
Shutdown_priv: N
Process_priv: N
File_priv: N
Grant_priv: N
References_priv: N
Index_priv: N
Alter_priv: N
Show_db_priv: N
Super_priv: N
Create_tmp_table_priv: N
Lock_tables_priv: N
Execute_priv: N
Repl_slave_priv: N
Repl_client_priv: N
Create_view_priv: N
Show_view_priv: N
Create_routine_priv: N
Alter_routine_priv: N
Create_user_priv: N
Event_priv: N
Trigger_priv: N
Create_tablespace_priv: N
ssl_type:
ssl_cipher:
x509_issuer:
x509_subject:
max_questions: 0
max_updates: 0
max_connections: 0
max_user_connections: 0
plugin: mysql_native_password
authentication_string:
password_expired: N
password last changed: 2017-08-05 17:18:55
password lifetime: NULL
```

任意位置登录

用户名称

密码为空

- 当用户创建完成之后，用户是否可以使用？

```
C:\Users\liwei>C:\wamp64\bin\mysql\mysql5.7.14\bin\mysql.exe -ustul -p
Enter password: ***
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.14 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```



```
C:\Users\liwei>C:\wamp64\bin\mysql\mysql5.7.14\bin\mysql.exe -ustu2
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.14 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

没有密码，直接登录成功！

1.1.2 删除用户

注意：mysql 中 user 是带着 host 本身的（具有唯一性）

基本语法：drop user 用户名;

```
mysql> drop user stu2;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

1.1.3 修改用户密码

MySQL 中提供了多种修改的方式：基本上都必须使用对应提供的一个系统函数：password()，需要靠该函数对密码进行加密处理。

- 使用专门的修改密码的指令

基本语法：set password for 用户 = password('新的明文密码');

```
mysql> -- 修改用户密码
mysql> set password for 'user1'@'%' = password('654321');
Query OK, 0 rows affected (0.00 sec)

mysql>
```

修改后的数据测试

```
C:\Users\liwei>C:\wamp64\bin\mysql\mysql5.7.14\bin\mysql.exe -ustul -p123456
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 5.7.14 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

使用新密码登录成功!

● 使用更新语句 update 来修改表

基本语法: update mysql.user set authentication_string= password('新的明文密码') where user = " " and host= " ";

1.2 权限管理

在 mysql 中将权限管理分为三类:

- 1、数据权限: 增删改查 (select\update\delete\insert)
- 2、结构权限: 结构操作 (create\drop)
- 3、管理权限: 权限管理 (create user\grant\revoke): 通常只给管理员如此权限

注意: MySQL 权限名称会以英文显示

数据	结构	管理
<input checked="" type="checkbox"/> SELECT <input type="checkbox"/> INSERT <input type="checkbox"/> UPDATE <input type="checkbox"/> DELETE <input type="checkbox"/> FILE	<input type="checkbox"/> CREATE <input type="checkbox"/> ALTER <input type="checkbox"/> INDEX <input type="checkbox"/> DROP <input type="checkbox"/> CREATE TEMPORARY TABLES <input type="checkbox"/> SHOW VIEW <input type="checkbox"/> CREATE ROUTINE <input type="checkbox"/> ALTER ROUTINE <input type="checkbox"/> EXECUTE <input type="checkbox"/> CREATE VIEW <input type="checkbox"/> EVENT <input type="checkbox"/> TRIGGER	<input type="checkbox"/> GRANT <input type="checkbox"/> SUPER <input type="checkbox"/> PROCESS <input type="checkbox"/> RELOAD <input type="checkbox"/> SHUTDOWN <input type="checkbox"/> SHOW DATABASES <input type="checkbox"/> LOCK TABLES <input type="checkbox"/> REFERENCES <input type="checkbox"/> REPLICATION CLIENT <input type="checkbox"/> REPLICATION SLAVE <input type="checkbox"/> CREATE USER

5个 12个 11个

1.2.1 授予权限：grant

将权限分配给指定的用户

基本语法：grant 权限列表 on 数据库[*].表名[*] to 用户;

权限列表：使用逗号分隔，但是可以使用 all privileges 代表全部权限

数据库.表名：可以是单表（数据库名字.表名），可以是具体某个数据库（数据库.*），也可以整库（ *.* ）

给 stu1 用户授权 czxy 数据库的 students 表

```
grant select,update,insert,delete on czxy.students to 'stu1'@'%';
```

```
mysql> grant select,update,insert,delete on czxy.students to 'stu1'@'%';  
Query OK, 0 rows affected (0.00 sec)
```

```
mysql>
```

给 stu1 用户授权 mydb 数据库的所有表

```
grant all privileges on mydb.* to 'stu1'@'%';
```

```
mysql> grant all privileges on mydb.* to 'stu1'@'%';  
Query OK, 0 rows affected (0.00 sec)
```

```
mysql>
```

用户被分配权限以后不需要退出就可以看到效果

```
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| czxy |  
| mydb |  
+-----+  
3 rows in set (0.00 sec)
```



```

mysql> use czxy;
Database changed
mysql> show tables;
+-----+
| Tables_in_czxy |
+-----+
| students        |
+-----+
1 row in set (0.00 sec)

mysql>

mysql> use czxy;
Database changed
mysql> show tables;
+-----+
| Tables_in_czxy |
+-----+
| children        |
| course          |
| mother          |
| person          |
| person_detail   |
| student_course  |
| students        |
+-----+
7 rows in set (0.00 sec)

mysql>

```

因为我们只给stul 授权了czxy中的students的权限
所以，czxy即便有很多表，但是stul仍然只能看到students这个表

具体权限查看：单表权限只能看到数据库中的一张表

```

mysql> use czxy;
Database changed
mysql> show tables;
+-----+
| Tables_in_czxy |
+-----+
| students        |
+-----+
1 row in set (0.00 sec)

mysql> select * from students;
+----+-----+-----+-----+
| id | name   | gender | tel      |
+----+-----+-----+-----+
| 1  | 八戒   | 男     | 13912345555 |
| 2  | 大师兄 | 男     | 13912346666 |
| 3  | 金角大王 | 男     | 13922222222 |
+----+-----+-----+-----+
3 rows in set (0.01 sec)

```

```

mysql> use mydb;
Database changed
mysql> show tables;
+-----+
| Tables_in_mydb |
+-----+
| student         |
| test_int        |
+-----+
2 rows in set (0.00 sec)

```




1.2.2 取消权限：revoke

权限回收：将权限从用户手中收回

基本语法：revoke 权限列表/all privileges on 数据库/*.表/* from 用户;

```
revoke all privileges on czxy.students from 'stu1'@'%';
```

```
mysql> revoke all privileges on czxy.students from 'stu1'@'%';  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> _
```

权限回收，同样不需要刷新，用户马上就会感受到

```
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| mydb |  
+-----+  
2 rows in set (0.00 sec)  
  
mysql>
```

1.2.3 刷新权限：flush

flush：刷新，将当前对用户的权限操作，进行一个刷新，将操作的具体内容同步到对应的表中。

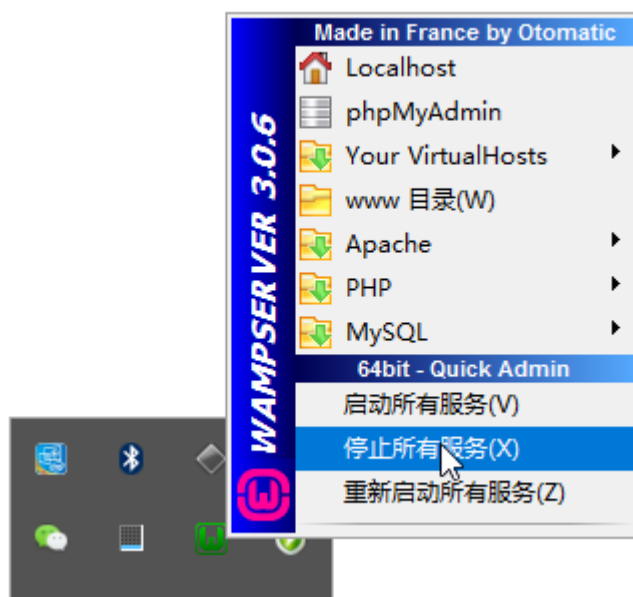
基本语法：flush privileges;

```
mysql> flush privileges;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> _
```

1.3 密码丢失的解决方案

如果忘记了 root 用户密码，就需要去找回或者重置 root 用户密码

1、 停止服务



- 2、重新启动服务：`mysqld.exe --skip-grant-tables` //启动服务器但是跳过权限

```
C:\Users\liwei>C:\wamp64\bin\mysql\mysql5.7.14\bin\mysqld.exe --skip-grant-tables
```

- 3、当前启动的服务器没有权限概念：非常危险，任何客户端，不需要任何用户信息都可以直接登录，而且是 root 权限：新开客户端，使用 `mysql.exe` 登录即可

```
C:\Users\liwei>C:\wamp64\bin\mysql\mysql5.7.14\bin\mysql.exe
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.14 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```



```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| baijia |
| czxy |
| czxy_2017_db |
| czxy_new |
| mydatabase1 |
| mydb |
| mydb3 |
| mysql |
| performance_schema |
| sys |
+-----+
11 rows in set (0.00 sec)
```

此时已经拥有最高的权限

4、修改 root 用户的密码：指定 用户名@host

```
mysql> set password for 'root'@'localhost' = password('123');
ERROR 1290 (HY000): The MySQL server is running with the --skip-grant-tables option so it cannot execute this statement
mysql> update mysql.user set authentication_string = password(123) where user='root' and host='localhost';
Query OK, 0 rows affected, 1 warning (0.01 sec)
Rows matched: 1 Changed: 0 Warnings: 1
mysql>
```

使用update语句可以进行密码更新!

在这种情况下, 不能使用 set password for ...

5、赶紧关闭服务器, 重启服务



登录成功!



```
c:\wamp64\bin\mysql\mysql5.7.14\bin\mysql.exe
Enter password: ***
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.14 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

2. 变量

MySQL 本质是一种编程语言，需要很多变量来保存数据。MySQL 中很多的属性控制都是通过 mysql 中固有的变量来实现的。

2.1 系统变量

系统内部定义的变量，系统变量针对所有用户（MySQL 客户端）有效。

- 显示系统所有变量：show variables [like 'pattern'];

```
| warning_count | 0
+-----+
504 rows in set, 1 warning (0.00 sec)
mysql>
```

show variables like 'auto%'; -- 这是模糊查找

Variable_name	Value
auto_increment_increment	1
auto_increment_offset	1
autocommit	OFF
automatic_sp_privileges	ON

4 rows in set, 1 warning (0.00 sec)

事务自动提交

- select 查询变量的数据值（系统变量）

基本语法：select @@变量名;

```
mysql> -- 查看系统变量
mysql> select @@autocommit;
```

@@autocommit
1

1 row in set (0.00 sec)

修改系统变量：分为两种修改方式

2.1.1 局部修改（会话级别）

只针对当前自己客户端当次连接有效

基本语法：set @@变量名 = 新值;

客户端2

```
mysql> select @@autocommit;
```

@@autocommit
1

1 row in set (0.00 sec)

客户端1

```
mysql> set autocommit = 0;
Query OK, 0 rows affected (0.00 sec)

mysql> select @@autocommit;
```

@@autocommit
0

1 row in set (0.00 sec)

2.1.2 全局修改

针对所有的客户端，“所有时刻”都有效

基本语法：set global 变量名 = 值; 或者 set @@global.变量名 = 值;

```
mysql> -- 全局修改系统变量
mysql> set global autocommit = 0;
Query OK, 0 rows affected (0.00 sec)

mysql> set @@global.auto_increment_increment = 2;
Query OK, 0 rows affected (0.00 sec)

mysql> show variables like 'auto_increment%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| auto_increment_increment | 1 |
| auto_increment_offset | 1 |
+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

```
mysql> select @@autocommit;
+-----+
| @@autocommit |
+-----+
| 1 |
+-----+
1 row in set (0.00 sec)

mysql> select @@autocommit, @@auto_increment_increment;
+-----+-----+
| @@autocommit | @@auto_increment_increment |
+-----+-----+
| 1 | 1 |
+-----+-----+
1 row in set (0.00 sec)

mysql>
```

全局修改之后：所有连接的客户端并没发现改变？全局修改只针对新客户端生效（正在连着的无效）

```
mysql> -- 全局修改系统变量
mysql> set global autocommit = 0;
Query OK, 0 rows affected (0.00 sec)

mysql> set @@global.auto_increment_increment = 2;
Query OK, 0 rows affected (0.00 sec)

mysql> show variables like 'auto_increment%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| auto_increment_increment | 1 |
| auto_increment_offset | 1 |
+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

```
mysql> select @@autocommit;
+-----+
| @@autocommit |
+-----+
| 1 |
+-----+
1 row in set (0.00 sec)

mysql> select @@autocommit, @@auto_increment_increment;
+-----+-----+
| @@autocommit | @@auto_increment_increment |
+-----+-----+
| 1 | 1 |
+-----+-----+
1 row in set (0.00 sec)

mysql>
```

注意：如果想要本次连接对应的变量修改有效，那么不能使用全局修改，只能使用会话级别修改（set 变量名 = 值）；

```
mysql> set global autocommit = 1;
Query OK, 0 rows affected (0.00 sec)

mysql> select @@autocommit;
+-----+
| @@autocommit |
+-----+
| 0 |
+-----+
1 row in set (0.00 sec)
```

2.2 会话变量

会话变量也称之为用户变量，会话变量跟 mysql 客户端是绑定的，设置的变量，只对当前用户使用的客户端生效。

2.2.1 定义用户变量：set @变量名 = 值;

```
set @name="helloworld!";
```



```
mysql> set @name="helloworld!";
Query OK, 0 rows affected (0.00 sec)

mysql> select @name;
+-----+
| @name |
+-----+
| helloworld! |
+-----+
1 row in set (0.00 sec)
```

在 mysql 中因为没有比较符号==, 所以是用=代替比较符号: 有时候在赋值的时候, 会报错:
mysql 为了避免系统分不清是赋值还是比较: 特定增加一个变量的赋值符号: :=

2.2.2 Set @变量名 := 值;

```
mysql> -- 使用专用赋值符号
mysql> set @age := 50;
Query OK, 0 rows affected (0.00 sec)

mysql> select @age;
+-----+
| @age |
+-----+
| 50 |
+-----+
1 row in set (0.00 sec)
```

MySQL 是专门存储数据的: 允许将数据从表中取出存储到变量中: 查询得到的数据必须只能是一行数据 (一个变量对应一个字段值): MySQL 没有数组。

2.2.3 使用 select 赋值且查看赋值过程

语法: select @变量 1:= 字段 1, @变量 2:= 字段 2 from 数据表 where 条件;

```
select @name:=name,@telnum:=telnum from person limit 1;
```



```
mysql> select @name:=name, @telnum:=telnum from person limit 1;
+-----+-----+
| @name:=name | @telnum:=telnum |
+-----+-----+
| 张三       | 13800138000     |
+-----+-----+
1 row in set (0.00 sec)

mysql> select @name, @telnum;
+-----+-----+
| @name | @telnum |
+-----+-----+
| 张三  | 13800138000 |
+-----+-----+
1 row in set (0.00 sec)

mysql>
```

:= 是真正的赋值操作

错误语法：使用“=”当做赋值了，系统会当做比较符号（MySQL 中“=”是关系运算符，where xxx='xx'）来处理

```
mysql> select @name=name, @telnum=telnum from person limit 1;
+-----+-----+
| @name=name | @telnum=telnum |
+-----+-----+
| NULL      | NULL           |
+-----+-----+
1 row in set (0.00 sec)
```

当做关系运算符“=”处理了

2.2.4 使用 select 只赋值，不看过程

语法格式：

select 字段 1, 字段 2... from 数据源 where 条件 into @变量 1, @变量 2...

```
select name, telnum from person limit 1 into @myName, @myTelnum;
```




```
mysql> select name,telnum from person limit 1;
+-----+-----+
| name | telnum |
+-----+-----+
| 张三 | 13800138000 |
+-----+-----+
1 row in set (0.00 sec)

mysql> select name,telnum from person limit 1 into @myName,@myTelnum;
Query OK, 1 row affected (0.00 sec)

mysql> select @myName,@myTelnum;
+-----+-----+
| @myName | @myTelnum |
+-----+-----+
| 张三 | 13800138000 |
+-----+-----+
1 row in set (0.00 sec)
```

客户端2

```
c:\wamp64\bin\mysql\mysql5.7.14\bin\mysql.exe
mysql> select @myName,@myTelnum;
+-----+-----+
| @myName | @myTelnum |
+-----+-----+
| NULL | NULL |
+-----+-----+
1 row in set (0.00 sec)

mysql> 客户端2中无法使用客户端1的会话变量
```

客户端1

```
c:\wamp64\bin\mysql\mysql5.7.14\bin\mysql.exe
mysql> select name,telnum from person limit 1;
+-----+-----+
| name | telnum |
+-----+-----+
| 张三 | 13800138000 |
+-----+-----+
1 row in set (0.00 sec)

mysql> select name,telnum from person limit 1 into @myName,@myTelnum;
Query OK, 1 row affected (0.00 sec)

mysql> select @myName,@myTelnum;
+-----+-----+
| @myName | @myTelnum |
+-----+-----+
| 张三 | 13800138000 |
+-----+-----+
1 row in set (0.00 sec)

mysql>
```

客户端1定义的会话变量，只能在客户端1中使用

2.3 局部变量

作用范围在 **begin** 到 **end** 语句块之间。在该语句块里设置的变量，**declare** 语句专门用于定义局部变量。

- 1、局部变量是使用 **declare** 关键字声明
- 2、局部变量 **declare** 语句出现的位置一定是在 **begin** 和 **end** 之间（**begin/end** 是在大型语句块中使用：函数/存储过程/触发器）
- 3、声明语法：**declare** 变量名 数据类型 **default** 值;

3. 流程结构

流程结构：代码的执行顺序

3.1 if 分支

3.1.1 基本语法

if 在 MySQL 中有两种基本用法

- 用在 select 查询当中，当做一种条件来进行判断

基本语法：if(条件,为真结果,为假结果)

```
-- 未起别名
select *,if(age>20,'符合','不符合') from students;

-- 起别名 fuhe
select *,if(age>20,'符合','不符合') fuhe from students;
```

```
mysql> select *,if(age>20,'符合','不符合') from students;
+----+-----+-----+-----+-----+-----+
| id | name  | gender | tel      | age | if(age>20,'符合','不符合') |
+----+-----+-----+-----+-----+-----+
| 1  | 八戒 | 男     | 13912345555 | 18 | 不符合                       |
| 2  | 大师兄 | 男     | 13912346666 | 23 | 符合                         |
| 3  | 金角大王 | 男     | 13922222222 | 88 | 符合                         |
+----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
mysql> select *,if(age>20,'符合','不符合') fuhe from students;
+----+-----+-----+-----+-----+-----+
| id | name  | gender | tel      | age | fuhe |
+----+-----+-----+-----+-----+-----+
| 1  | 八戒 | 男     | 13912345555 | 18 | 不符合 |
| 2  | 大师兄 | 男     | 13912346666 | 23 | 符合   |
| 3  | 金角大王 | 男     | 13922222222 | 88 | 符合   |
+----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

- 用在复杂的语句块中（函数/存储过程/触发器）

基本语法

mysql 中的 if

```
if 条件表达式 then
    满足条件要执行的语句;
end if;
```

php、javascript

```
if( 条件 ) {
    满足条件要执行的语句;
}
```

T

3.1.2 复合语法

复合语法：代码的判断存在两面性，两面都有对应的代码执行。

基本语法：

```
if 条件表达式 then
    满足条件要执行的语句;
else
    不满足条件要执行的语句;
    //如果还有其他分支（细分），可以在里面再使用 if
        if 条件表达式 then
            //满足要执行的语句
        end if;
end if;
```

```
BEGIN
    declare result tinyint;
    set result=0;
    if period=0 then
        if date=periodData then
            set result=1;
        end if;
    elseif period=1 then
        set result=1;
    elseif period=2 then
        if WEEKDAY(date)+1=periodData then
            set result=1;
        end if;
    elseif period=3 then
        if DAYOFMONTH(date)=periodData then
            set result=1;
        end if;
    else
        if DAYOFMONTH(date)=periodData then
            set result=1;
        end if;
    end if;
    RETURN result;
END
```

4. While 循环

4.1 基本语法

循环体都是需要在大型代码块中使用
基本语法

mysql 中

```
while 条件 do
    要循环执行的代码;
end while;
```

```
while(条件) {
    循环体;
}
```

while

```
1 delimiter $$
2 create procedure test_while()
3 begin
4 declare sum int default 0;
5 declare t int default 5;
6 while t>0 do
7 set sum=sum+1;
8 set t=t-1;
9 end while;
10 select sum;
11 end $$
12 delimiter ;
```

4.2 结构标识符(别名)

结构标识符：为某些特定的结构进行命名，然后为的是在某些地方使用名字

基本语法

标识名字:while 条件 do

循环体

end while [标识名字];



标识符的存在主要是为了循环体中使用循环控制。在 mysql 中没有 continue 和 break，有自己的关键字替代：

iterate: 迭代，就是以下的代码不执行，重新开始循环（continue）

leave: 离开，整个循环终止（break）

标识名字:while 条件 do

if 条件判断 then

循环控制;

iterate/leave 标识名字;

end if;

循环体

end while [标识名字];

```
-- 别名（结构化表示符）
-- w1:while i<=100 do
-- 循环体
-- end while w1
```

w1 是 while 循环的一个别名

别名用途：比如在while 循环中，要跳出while循环，就要使用别名。

5. 函数

password('明文的密码') -> 生成加密的密码

在 mysql 中，函数分为两类：系统函数（内置函数）和自定义函数

不管是内置函数还是用户自定义函数，都是使用 select 函数名(参数列表);

5.1 内置函数

5.1.1 字符串函数

- char_length(): 判断字符串的字符数
- length(): 判断字符串的字节数（与字符集）



```
mysql>
mysql> select char_length('你好中国'),length('你好中国');
+-----+-----+
| char_length('你好中国') | length('你好中国') |
+-----+-----+
| 4 | 8 |
+-----+-----+
1 row in set (0.00 sec)
```

- concat(): 连接字符串
- instr(): 判断字符在目标字符串中是否存在，存在返回其位置，不存在返回 0

```
mysql> select concat('你好','中国'),instr('你好中国','中'),instr('你好中国','万');
+-----+-----+-----+
| concat('你好','中国') | instr('你好中国','中') | instr('你好中国','万') |
+-----+-----+-----+
| 你好中国 | 3 | 0 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

- lcase(): 全部小写
- left(): 从左侧开始截取，直到指定长度的位置（位置如果超过长度，截取所有）
- right(): 从右侧开始截取，直到指定长度的位置（位置如果超过长度，截取所有）

```
mysql> select lcase('aBcD'),left('你好中国',2);
+-----+-----+
| lcase('aBcD') | left('你好中国',2) |
+-----+-----+
| abcd | 你好 |
+-----+-----+
1 row in set (0.00 sec)
```

使用 left 截取时间

```
mysql> select left('2017-08-08 10:10:10',10);
+-----+
| left('2017-08-08 10:10:10',10) |
+-----+
| 2017-08-08 |
+-----+
1 row in set (0.00 sec)
```

使用 right() 从右侧截取

```
mysql> select right('2017-08-08 10:10:10',8);
+-----+
| right('2017-08-08 10:10:10',8) |
+-----+
| 10:10:10 |
+-----+
1 row in set (0.00 sec)
```

- substring() 函数

substring (str, pos)

substring (str, pos, length)

说明: substring (被截取字段, 从第几位开始截取)

substring (被截取字段, 从第几位开始截取, 截取长度)

例: select substring (content,5) as abstract from my_content_t

select substring (content,5,200) as abstract from my_content_t

(注: 如果位数是负数 如-5 则是从后倒数位数, 到字符串结束或截取的长度)

```
mysql> select substring("传智专修学院", 3);
+-----+
| substring("传智专修学院", 3) |
+-----+
| 专修学院 |
+-----+
1 row in set (0.00 sec)

mysql> select substring("传智专修学院", 3, 2);
+-----+
| substring("传智专修学院", 3, 2) |
+-----+
| 专修 |
+-----+
1 row in set (0.00 sec)

mysql>
```

- ltrim(): 消除左边对应的空格

```
mysql> select ltrim(" 哈哈 沐阳 我来了! ");
+-----+
| ltrim(" 哈哈 沐阳 我来了! ") |
+-----+
| 哈哈 沐阳 我来了! |
+-----+
1 row in set (0.00 sec)
```

5.1.2 时间函数

now(): 返回当前时间, 日期 时间

curdate(): 返回当前日期

curtime(): 返回当前时间

```
mysql> select now(), curdate(), curtime();
+-----+-----+-----+
| now() | curdate() | curtime() |
+-----+-----+-----+
| 2017-03-04 00:53:06 | 2017-03-04 | 00:53:06 |
+-----+-----+-----+
```

datediff(): 判断两个日期之间的天数差距, 参数日期必须使用字符串格式 (用引号)

```
mysql> select datediff('2017-08-10','2017-08-08');
+-----+
| datediff('2017-08-10','2017-08-08') |
+-----+
| 2 |
+-----+
1 row in set (0.00 sec)

mysql> _
```

此函数经常用于判断文章是否最新发布, 并配合下面图标进行显示



Unix_timestamp(): 获取时间戳

```
mysql> select unix_timestamp();
+-----+
| unix_timestamp() |
+-----+
| 1488560184 |
+-----+
1 row in set (0.00 sec)

mysql> select unix_timestamp();
+-----+
| unix_timestamp() |
+-----+
| 1488560189 |
+-----+
1 row in set (0.00 sec)
```

From_unixtime(): 将指定时间戳转换成对应的日期时间格式

```
mysql> select from_unixtime(1234567890);
+-----+
| from_unixtime(1234567890) |
+-----+
| 2009-02-14 07:31:30 |
+-----+
1 row in set (0.00 sec)
```

5.1.3 数学函数

abs(): 绝对值



ceil(): 向上取整

floor(): 向下取整

pow(): 求指数, 谁的多少次方

rand(): 获取一个随机数 (0-1 之间)

round(): 四舍五入函数

```
mysql> select abs(-1),ceiling(1.1),floor(1.1),pow(2,4),rand(),round(1.5);
+-----+-----+-----+-----+-----+-----+
| abs(-1) | ceiling(1.1) | floor(1.1) | pow(2,4) | rand() | round(1.5) |
+-----+-----+-----+-----+-----+-----+
| 1 | 2 | 1 | 16 | 0.8089020352893528 | 2 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

5.1.4 其他函数

md5(): 对数据进行 md5 加密 (mysql 中的 md5 与其他任何地方的 md5 加密出来的内容是完全相同的)

version(): 获取版本号

database(): 显示当前所在数据库

uuid(): 生成一个唯一标识符 (自增长): 自增长是单表唯一, UUID 是整库 (数据唯一同时空间唯一)

```
mysql> select md5('a'),version(),database(),uuid();
+-----+-----+-----+-----+
| md5('a') | version() | database() | uuid() |
+-----+-----+-----+-----+
| 0cc175b9c0f1b6a831c399e269772661 | 5.5.28 | mydb | 412f8d9b-0033-11e7-beb5-080027bf2eaa |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

6. 自定义函数

自定义函数: 用户自己定义的函数

函数: 实现某种功能的语句块 (由多条语句组成)

- 1、函数内部的每条指令都是一个独立的个体: 需要符合语句定义规范: 需要语句结束符分号;
- 2、函数是一个整体, 而且函数是在调用的时候才会被执行, 那么当设计函数的时候, 意味着整体不能被中断;



3、MySQL 一旦见到语句结束符分号，就会自动开始执行

解决方案：在定义函数之前，尝试修改临时的语句结束符

基本语法：delimiter

修改临时语句结束符：delimiter 新符号[可以使用系统非内置即可\$\$]

中间为正常 SQL 指令：使用分号结束（系统不会执行：不认识分号）

使用新符号结束

修改回语句结束符：delimiter ;

6.1 创建函数

自定义函数包含几个要素：function 关键字，函数名，参数（形参和实参[可选]），确认函数返回值类型，函数体，返回值语句

6.1.1 函数定义基本语法

```
create function 函数名(形参) returns 返回值类型
begin
    //函数体
    return 返回值数据;    //数据必须与结构中定义的返回值类型一致
end
```

6.1.2 创建无参函数

```
-- 创建一个无参函数，返回值是int
create function myfun1() returns int
begin
    return 10;
end
```

```
-- 创建一个无参函数，返回值是 int
create function myfun1() returns int
begin
    return 10;
end
```

但执行 sql 指令时报错



```
mysql> create function myfun1() returns int
-> begin
-> return 10;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version
for the right syntax to use near ';' at line 3
mysql> end
```

错误原因为:

mysql 以“;”作为语句结束符,但是当执行到 `return 10;` 时认为指令结束,实际上函数并未结束。

解决办法:临时设置 sql 指令的结束符为其他符号,只要不是“;”和系统特殊含义的符号(`_`,`%`)就行

通常设置为: `$$`

具体操作:使用 `delimiter` 符号;可以修改 sql 指令结束符

6.1.3 修改语句结束符

```
mysql> delimiter $$
mysql> create function myfun1() returns int
-> begin
-> return 10;
-> end
-- $$ 表示语句结束
mysql> $$
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
mysql>
```

6.1.4 函数体只有一条指令

并不是所有的函数都需要 `begin` 和 `end`: 如果函数体本身只有一条指令 (`return`),那么可以省略 `begin` 和 `end`

```
mysql> create function myfun2() returns int
-> return 100;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

因为“;”恰巧是函数语句结束,所以此类型函数不需要修改 sql 语句结束符。

6.1.5 创建有参函数

形参：在 mysql 中需要为函数的形参指定数据类型（形参本身可以有多个）

基本语法：变量名 字段类型

```
mysql> create function myfun3(a int,b int) returns int
-> return a+b;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

6.2 查看函数

- 可以通过查看 function 状态，查看所有的函数

Show function status [like 'pattern'];

```
mysql> show function status like 'my%' \G;
***** 1. row *****
      Db: czxy
      Name: myfun1
      Type: FUNCTION
      Definer: root@localhost
      Modified: 2017-08-05 23:23:48
      Created: 2017-08-05 23:23:48
      Security_type: DEFINER
      Comment:
      character_set_client: gbk
      collation_connection: gbk_chinese_ci
      Database Collation: utf8_general_ci
***** 2. row *****
      Db: czxy
      Name: myfun2
      Type: FUNCTION
      Definer: root@localhost
      Modified: 2017-08-05 23:26:17
      Created: 2017-08-05 23:26:17
      Security_type: DEFINER
      Comment:
      character_set_client: gbk
      collation_connection: gbk_chinese_ci
      Database Collation: utf8_general_ci
***** 3. row *****
      Db: czxy
```

- show 查看函数的创建语句：show create function 函数名字;



```
mysql> show create function myfun1 \G;
***** 1. row *****
Function: myfun1
sql_mode: STRICT_ALL_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER
Create Function: CREATE DEFINER=`root`@`localhost` FUNCTION `myfun1`() RETURNS int(11)
begin
return 10;
end
character_set_client: gbk
collation_connection: gbk_chinese_ci
Database Collation: utf8_general_ci
1 row in set (0.00 sec)

ERROR:
No query specified
```

6.3 调用函数

自定义函数的调用与内置函数的调用是一样的：select 函数名(实参列表);
调用格式：

```
mysql> select myfun1(),myfun2(),myfun3(10,34);
+-----+-----+-----+
| myfun1() | myfun2() | myfun3(10,34) |
+-----+-----+-----+
|      10 |      100 |             44 |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> _
```

6.4 删除函数

删除函数：drop function 函数名;



```
mysql> drop function myfun1;
Query OK, 0 rows affected (0.00 sec)

mysql> show function status like 'my%' \G;
***** 1. row *****
      Db: czxy
      Name: myfun2
      Type: FUNCTION
      Definer: root@localhost
      Modified: 2017-08-05 23:26:17
      Created: 2017-08-05 23:26:17
      Security_type: DEFINER
      Comment:
      character_set_client: gbk
      collation_connection: gbk_chinese_ci
      Database Collation: utf8_general_ci
***** 2. row *****
      Db: czxy
      Name: myfun3
      Type: FUNCTION
      Definer: root@localhost
      Modified: 2017-08-05 23:28:43
      Created: 2017-08-05 23:28:43
      Security_type: DEFINER
      Comment:
      character_set_client: gbk
      collation_connection: gbk_chinese_ci
      Database Collation: utf8_general_ci
```

删除函数 myfun1()

6.5 注意事项

- 1、 自定义函数是属于用户级别的：只有当前客户端对应的数据库中可以使用
- 2、 可以在不同的数据库下看到对应的函数，但是不可以调用

```
mysql> use mydb;
Database changed
mysql> select myfun2();
ERROR 1305 (42000): FUNCTION mydb.myfun2 does not exist
mysql>
```

myfun2()函数在czxy库中有效

- 3、 自定义函数：通常是为了将多行代码集合到一起解决一个重复性的问题
- 4、 函数因为必须规范返回值：那么在函数内部不能使用 select 指令：select 一旦执行就会得到一个结果（result set）

例外的情况：select 字段 into @变量;

7. 函数流程结构案例

需求：从 1 开始，直到用户传入的对应的值为止，自动求和：凡是 5 的倍数都不要。



8

$1+2+3+4+5+6+7+8 = 31$

设计:

- 1、 创建函数
- 2、 需要一个形参: 确定要累加到什么位置
- 3、 需要定义一个变量来保存对应的结果: set @变量名;
使用局部变量来操作: 此结果是在函数内部使用
Declare 变量名 类型 [= 默认值];
- 4、 内部需要一个循环来实现迭代累加
- 5、 循环内部需要进行条件判断控制: 5 的倍数
- 6、 函数必须有返回值

函数指令代码:

```
-- 修改 sql 结束符
delimiter $$

-- 1、 创建函数
create function my_sum(end_value int) returns int
begin
-- 2、 需要一个形参: 确定要累加到什么位置
-- 3、 需要定义一个变量来保存对应的结果: set @变量名;
    declare res int default 0;
    declare i int default 1;
-- 使用局部变量来操作: 此结果是在函数内部使用
-- Declare 变量名 类型 [= 默认值];
-- 4、 内部需要一个循环来实现迭代累加
    mywhile:while i <= end_value do
-- 5、 循环内部需要进行条件判断控制: 5 的倍数
        if i % 5 = 0 then
            set i = i + 1;
            iterate mywhile;
        end if;
        -- 累加操作
        set res = res + i;
        set i = i + 1;
    end while mywhile;
-- 6、 函数必须有返回值
    return res;
end
$$
delimiter ;
```

运行结果:



```
mysql> delimiter $$
mysql> 1、创建函数
mysql> create function my_sum(end_value int) returns int
-> begin
-> -- 2、需要一个形参：确定要累加到什么位置
-> -- 3、需要定义一个变量来保存对应的结果：set @变量名；
-> declare res int default 0;
-> declare i int default 1;
-> -- 使用局部变量来操作：此结果是在函数内部使用
-> -- Declare 变量名 类型 [= 默认值];
-> -- 4、内部需要一个循环来实现迭代累加
-> mywhile:while i <= end_value do
-> -- 5、循环内部需要进行条件判断控制：5的倍数
-> if i % 5 = 0 then
-> set i = i + 1;
-> iterate mywhile;
-> end if;
-> -- 累加操作
-> set res = res + i;
-> set i = i + 1;
-> end while mywhile;
-> -- 6、函数必须有返回值
-> return res;
-> end
-> $$
Query OK, 0 rows affected (0.00 sec)
mysql> delimiter ;
mysql>
```

函数的创建过程

调用函数：select 函数名(实参);

```
mysql> delimiter ;
mysql> select my_sum(8);
+-----+
| my_sum(8) |
+-----+
|          31 |
+-----+
1 row in set (0.00 sec)
```

8. 变量作用域

变量作用域：变量能够使用的区域范围

8.1 局部作用域

使用 declare 关键字声明（在结构体内：函数/存储过程/触发器），而且只能在结构体内部使用

1、declare 关键字声明的变量没有任何符号修饰，就是普通字符串，如果在外部访问该变量，系统会自动认为是字段

8.2 会话作用域

用户定义的，使用@符号定义的变量，使用 set 关键字

会话作用域：在当前用户当次连接有效，只要在本连接之中，任何地方都可以使用（可以在结构内部，也可以跨库）

会话变量可以在函数内部使用

```
mysql> set @name = "张三";
Query OK, 0 rows affected (0.00 sec)

mysql> create function myfun4() returns char(4)
-> return @name;
Query OK, 0 rows affected (0.00 sec)

mysql> select myfun4();
+-----+
| myfun4() |
+-----+
| 张三     |
+-----+
1 row in set (0.00 sec)

mysql> _
```

此处定义的是一个会话级变量

定义一个函数，返回值是@name 会话变量值

说明会话级变量可以再函数内部使用

会话变量可以跨库

```
mysql> use mydb3;
Database changed
mysql> select @name;
+-----+
| @name |
+-----+
| 张三  |
+-----+
1 row in set (0.00 sec)
```

切换库

依然可以使用会话级变量

8.3 全局作用域

所有的客户端所有的连接都有效：需要使用全局符号来定义

Set global 变量名 = 值;

Set @@global.变量名 = 值;

通常，在 SQL 编程的时候，不会使用自定义变量来控制全局。一般都是定义会话变量或者在结构中使用局部变量来解决问题。



9. 今日总结

