数据库技术与应用

第五章 MYSQL权限与安全

本章主要内容

MYSQL权限系统主要用来对连接到数据库的用户进行权限的验证，以此来判断该用户是否合法，对于合法用户则赋予相应的数据库权限。

- v 5.1 MYSQL权限管理
  - * 5.1.1 MYSQL权限管理的工作原理
  - * 5.1.2 权限表的存取
  - * 5.1.3 账号管理
- v 5.2 MYSQL安全问题
  - * 5.2.1 操作系统安全问题
  - * 5.2.2 MYSQL数据库安全问题
  - * 5.2.3 MYSQL安全选项

5.1.1 MYSQL权限管理的工作原理

- MySQL存取控制包含2个阶段：
  - 用户验证：服务器检查连接用户是否合法。不合法则拒绝连接。
  - 权限验证：对通过认证的合法用户赋予相应的权限，用户在这些权限范围内对数据库做相应的操作。

注意

MYSQL用户验证是通过用户名+主机名进行联合认证，例如root@localhost。不同主机而用户名相同的被认为是不通用户，例如zzh@qq.com和zzh@zzhstudio.cn不同。

记录用户权限信息的表在MYSQL启动时加载到内存中。

5.1.2 权限表的存取

MYSQL权限存取的两个过程中，分别要用到“mysql”数据库下的系统表。

* 用户验证：使用user、db、host权限表判断用户的合法性和基本权限。
* 权限验证：在基本权限验证后，可通过tables_priv和columns_priv、procs_priv表对表、列、存储过程提供更精确的权限控制。

用户连接时，权限表的存取过程如下：

- 先通过user表的host、user、password字段判断连接的IP、用户名、密码是否正确。
- 通过身份验证后，则按照user→db→tables_priv→columns_priv顺序得到数据库的权限。权限的范围依次递减。

权限分配的说明

- user→db→tables_priv→columns_priv分配的基本步骤：

  - 1、检查全局权限user表，如果表中对应权限为Y，则该用户对所有数据库的对应权限都为Y，则不再检查其他权限表。
  - 2、如user表相应权限为N，则检查db表该用户对应的数据库。如db表中为Y，则用户对应数据库具有指定权限，不再检查其他权限表。
  - 3、如db表相应权限为N，则检查tables_priv表对应数据库下对应的具体表，如为Y，则对该表具有相应权限。
  - 4、如tables_priv表相应权限为N，则检查columns_priv中对应表的具体字段，查看其权限。

实际上user→db→tables_priv→columns_priv 顺序对应的是：

服务器→指定数据库→指定表→指定字段

权限分配的说明

- user→db→tables_priv→columns_priv分配的基本步骤：

Root拥有全局权限

```
mysql> select * from mysql.user where user='root' \G
*************************** 1. row ***************************
                  Host: localhost
                  User: root
              Password: *48413AF5843526BC0B14C3FA9C81D
           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
```

```
mysql> select * from mysql.db where user='root' \G
Empty set (0.16 sec)
```

zzh1拥有st的查询权限

```
mysql> select * from mysql.user where user='zzh1' \G
*************************** 1. row ***************************
                  Host: localhost
                  User: zzh1
              Password: *48413AF5843526BC0B14C3FA9C81
           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
```

```
mysql> select * from mysql.db where user='zzh1' \G
*************************** 1. row ***************************
                  Host: localhost
                    Db: st
                  User: zzh1
           Select_priv: Y
           Insert_priv: N
           Update_priv: N
           Delete_priv: N
           Create_priv: N
             Drop_priv: N
```

5.1.3 账号管理

* 5.1.3.1 创建账号
* 5.1.3.2 查看权限
* 5.1.3.3 权限更改
* 5.1.3.4 更改密码
* 5.1.3.5 更改用户名
* 5.1.3.6 删除账号
* 5.1.3.7 用户登录、退出MYSQL

5.1.3.1 创建账号—CREATE USER

v CREATE USER语法

CREATE USER *user* [IDENTIFIED BY [PASSWORD] '*password*']

[, *user* [IDENTIFIED BY [PASSWORD] '*password*']] ...

* 必须拥有mysql数据库的全局CREATE USER权限，或INSERT权限。
* 可同时创建多个账号
* 例如：**CREATE USER ‘zzh1’@’localhost’ IDENTIFIED BY ‘123456’, ‘zzh2’@’localhost’ IDENTIFIED BY ‘123456’;**

5.1.3.1 创建账号—添加user表记录

Host	User	Password
localhost	root	*48413AF5843526BC0B14C3FA9C81DFF89C0BAC1E
localhost	zzh1	*48413AF5843526BC0B14C3FA9C81DFF89C0BAC1E
localhost	zzhmysql	*48413AF5843526BC0B14C3FA9C81DFF89C0BAC1E
localhost	zzh2	zhihong
localhost	zzh3	*48413AF5843526BC0B14C3FA9C81DFF89C0BAC1E

v 语法

```
INSERT INTO
user(host,user,password,ssl_cipher,x509_issuer,x509_subject)
VALUES ('localhost', 'zzh1', PASSWORD ('123456'),'','','')
```

* 在直接向user表插入用户时，密码必须要用password()函数加密，MYSQL默认添加用户时密码用passwrod()函数处理。否则，即便创建了用户，该用户也不能登录。
* User表中，ssl_cipher,x509_issuer,x509_subject三个字段没有默认值，所以必须填充初始值才能插入用户。
* 直接向user表中插入用户后，需要执行FLUSH PRIVIEGES;命令使MYSQL重新装载user表中权限，方能使添加的用户生效。当然也可以重新启动MYSQL服务器。

5.1.3.1 创建账号—GRANT语句

v 基本语法

```
GRANT priv_type [(column_list)] [, priv_type [(column_list)]] ...
    ON [object_type] {tbl_name | * | *.* | db_name.*}
    TO user [IDENTIFIED BY [PASSWORD] 'password']
        [, user [IDENTIFIED BY [PASSWORD] 'password']] ...
    [WITH OPTION]

object_type = TABLE | FUNCTION | PROCEDURE
```

* GRANT语句在创建用户的同时，指定该用户的相关权限。
* 使用WITH OPTION选项使得创建的用户也可以将其获得的权限授予给其他用户。

5.1.3.1 创建账号—GRANT语句

- 授予的权限可以分为多个层级：
  - 全局层级：全局权限适用于一个给定服务器中的所有数据库。这些权限存储在mysql.user表中。GRANT ALL ON *.*和REVOKE ALL ON *.*只授予和撤销全局权限。
  - 表层级：表权限适用于一个给定表中的所有列。这些权限存储在mysql.talbes_priv表中。GRANT ALL ON db_name.tbl_name和REVOKE ALL ON db_name.tbl_name只授予和撤销表权限。
  - 列层级：列权限适用于一个给定表中的单一列。这些权限存储在mysql.columns_priv表中。当使用REVOKE时，您必须指定与被授权列相同的列。
  - 子程序层级：CREATE ROUTINE, ALTER ROUTINE, EXECUTE和GRANT权限适用于已存储的子程序。这些权限可以被授予为全局层级和数据库层级。除了CREATE ROUTINE外，这些权限可以被授予为子程序层级，并存储在mysql.procs_priv表中。

5.1.3.1 创建账号—GRANT语句

各层级权限表

表名	列名	可能的设置元素
tables_priv	Table_priv	'Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'References', 'Index', 'Alter'
tables_priv	Column_priv	'Select', 'Insert', 'Update', 'References'
columns_priv	Column_priv	'Select', 'Insert', 'Update', 'References'
procs_priv	Proc_priv	'Execute', 'Alter Routine', 'Grant'

* 对于全局、数据库、表和子程序层级，GRANT ALL只能赋予正在授权的层级中存在的权限。例如 GRANT ALL ON db_name.*，这是一个数据库层级语句。
* MySQL允许您对不存在的数据库目标授予权限。目的是允许数据库管理员为将在此后被创建的数据库目标预备用户 账户和权限。
* 删除一个表或数据库时，MySQL不会自动撤销任何权限。但是，删除一个子程序，则被赋予该子程序的所有子程序层级的权限都被撤销。

5.1.3.1 创建账号—GRANT语句

- 授予各层级权限示例：
  - GRANT ALL PRIVILEGES ON *.* TO 'zzh1'@'localhost' IDENTIFIED BY '123456' ;—全局层级
  - GRANT ALL PRIVILEGES ON test.* TO 'zzh1'@'localhost' IDENTIFIED BY '123456'; —表层级
  - GRANT SELECT,CREATE,DROP ON st.student TO 'zzh1'@'localhost' IDENTIFIED BY '123456'; —表层级
  - GRANT SELECT,UPDATE (sname)ON st.student TO 'zzh1'@'localhost' IDENTI
  - GRANT EXECUTE ON PR 'zzh1'@'localhost' IDENTI

```
mysql> select * from mysql.procs_priv where user='zzh4' \G
*************************** 1. row ***************************
        Host: localhost
          Db: st
        User: zzh4
Routine_name: getrecord
Routine_type: PROCEDURE
     Grantor: root@localhost
   Proc_priv: Execute
   Timestamp: 2011-09-11 16:03:21
```

5.1.3.1 创建账号—GRANT语句

- 权限时限制用户资源：

```
GRANT ALL ON customer.* TO francis'@'localhost'
 IDENTIFIED BY 'frank'
 WITH MAX_QUERIES_PER_HOUR 20      每小时执行查询次数
 MAX_UPDATES_PER_HOUR 10           每小时执行更新次数
 MAX_CONNECTIONS_PER_HOUR 5        每小时连接服务器次数
 MAX_USER_CONNECTIONS 2;           同时最大连接数
```

5.1.3.1 创建账号—GRANT语句

MYSQL指定用户和主机名称时，可以使用如下形式。用于创建特定的用户权限。

Host值	User值	被条目匹配的连接
'thomas.loc.gov'	'fred'	fred, 从thomas.loc.gov 连接
'thomas.loc.gov'	''	任何用户, 从thomas.loc.gov连接
'%'	'fred'	fred, 从任何主机连接
'%'	''	任何用户, 从任何主机连接
'%.loc.gov'	'fred'	fred, 从在loc.gov域的任何主机连接
'x.y.%'	'fred'	fred, 从x.y.net、x.y.com,x.y.edu等联接。（这或许无用）
'144.155.166.177'	'fred'	fred, 从有144.155.166.177 IP地址的主机连接
'144.155.166.%'	'fred'	fred, 从144.155.166 C类子网的任何主机连接

```
GRANT ALL PRIVILEGES ON *.* TO ''@'192.168.1.%' IDENTIFIED BY '123456' ;
```

5.1.3.2 查看权限

- 1、利用SHOW GRANT语句

SHOW GRANTS FOR user@host \G

```
mysql> show grants for zzh@localhost \G
*************************** 1. row ***************************
Grants for zzh@localhost: GRANT USAGE ON *.* TO 'zzh'@'localhost' IDENTIFIED BY
PASSWORD '*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9'
*************************** 2. row ***************************
Grants for zzh@localhost: GRANT SELECT, UPDATE (sname) ON `st`.`student` TO 'zzh
'@'localhost'
```

- 2、利用SELECT 语句查看权限表

select * from information_schema.schema_privileges where grantee="'zzh'@'localhost'" \G

```
mysql> select * from information_schema.schema_privileges where grantee="'zzh1'@
'localhost'"\G
*************************** 1. row ***************************
       GRANTEE: 'zzh1'@'localhost'
 TABLE_CATALOG: def
  TABLE_SCHEMA: st
PRIVILEGE_TYPE: SELECT
  IS_GRANTABLE: NO
1 row in set (0.00 sec)
```

5.1.3.3 权限更改—REVOKE语句

v 基本语法

```
REVOKE priv_type [(column_list)] [, priv_type [(column_list)]] ...
    ON [object_type] {tbl_name | * | *.* | db_name.*}
    FROM user  [, user] ...
object_type = TABLE | FUNCTION | PROCEDURE
```

* REVOKE语句可以同时收回多个用户指定的操作权限。
* REVOKE语句不能够删除MYSQL用户。
* 要撤销所有权限，需使用以下语法。

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ...
```

5.1.3.4 更改密码

- 1、修改root用户密码
  - 使用mysqladmin命令

```
C:\mysql\mysql5\bin>mysqladmin -u root -p password "zhihong"
Enter password: ********
```

Mysqladmin –u root –p password “新密码”

  - 使用SET语句

SET password=password(“新密码”);

  - 使用UPDATE修改user表

Update mysql.user set password=password(“新密码”) where User=“root” and host=“localhost”;

注意：使用UPDATE语句后应使用FLUSH PRIVILEGES语句重新加载权限表。

5.1.3.4 更改密码

- 2、root修改普通用户密码
  - 使用SET语句

    SET password for ’用户’@’主机’ =password(“新密码”);
  - 使用UPDATE修改user表

    Update mysql.user set password=password(“新密码”) where User=“用户” and host=“主机”;
- 3、普通用户修改自己的密码

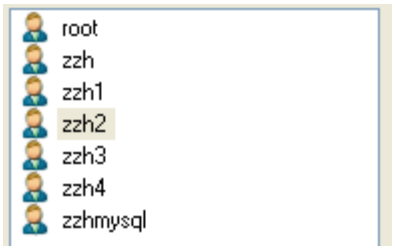
  SET password =password(“新密码”);

注意：使用UPDATE语句后应使用FLUSH PRIVILEGES语句重新加载权限表。

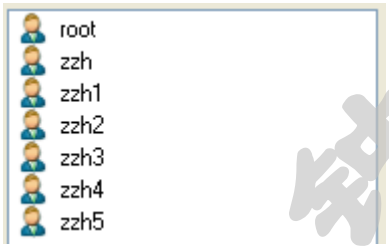
5.1.3.5 更改用户名

v 1、利用 RENAME USER 语句

RENAME USER old_user TO new_user [, old_user TO new_user] ...




```
mysql> rename user 'zzhmysql'@'localhost' to 'zzh5'@'localhost';
Query OK, 0 rows affected (0.42 sec)
```



注意：使用RENAME USER，须拥有全局CREATE USER权限或mysql数据库UPDATE权限。如果旧 账户不存在或者新账户已存在，则会出现错误。old_user和new_user值的给定方法与GRANT语句一样。

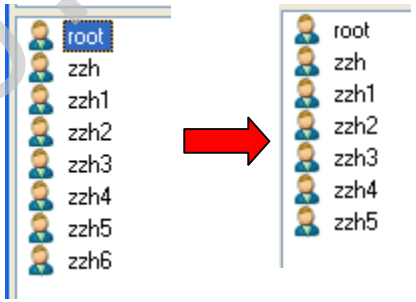
5.1.3.6 删除账号

- 1、利用DROP USER语句

DROP USER user@host ;

```
mysql> grant usage on *.* to 'zzh6'@'localhost' identified by '123456';
Query OK, 0 rows affected (0.02 sec)

mysql> drop user zzh6@localhost;
Query OK, 0 rows affected (0.00 sec)
```



- 2、利用DELETE 语句查看权限表

delete from user where user="zzh6" and host="localhost" ;

flush privileges;

5.1.3.7 用户登录、退出MYSQL

- 用户登录MYSQL基本步骤
  - 1、在DOS下进入MYSQL的安装目录BIN目录下
  - 2、执行下面登录语句

    MYSQL –h localhost –u root –p
  - 3、输入密码

```
C:\mysql\mysql5\bin>mysql -h localhost -u root -p
Enter password: *******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.5.13 MySQL Community Server (GPL)
```

- 退出MYSQL
  - 1、EXIT
  - 2、QUIT

5.2 MYSQL安全问题

任何数据库系统，安全问题最为重要。只有保障了数据的安全，用户才能够信任系统提供的信息、才能够为业务的开展提供稳定的支持。讨论安全时，强调必须完全保护整个服务器主机的安全(而不只是MySQL服务器)防范各种类型的可能的攻击：偷听、修改、重放和拒绝服务。本节将从两个方面讨论：

- 5.2.1 操作系统安全问题
- 5.2.2 MYSQL数据库安全问题
- 5.2.3 MYSQL安全选项

5.2.1 操作系统安全问题

- 1. 严格控制操作系统账号和权限
- 2. 尽量避免以root（操作系统的高级管理员）权限运行MYSQL。便于保护MYSQL安装目录。
- 3. 防止DNS欺骗。指定主机时尽量使用IP地址来识别主机。

5.2.2 MYSQL数据库安全问题

- 1. 删除匿名账号（某些版本MYSQL会创建匿名用户）。
- 2. 给root账号设置口令。
- 3. 设置安全密码。密码复杂性、使用过程的安全性。

```
C:\mysql\mysql5\bin>mysql -uroot -pzhihong
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 5.5.13 MySQL Community Server (GPL)
```

```
C:\mysql\mysql5\bin>mysql -uroot -p
Enter password: *******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 5.5.13 MySQL Community Server (GPL)
```


```
[client]
user=root
password=zhihong
port=3306
```

```
C:\mysql\mysql5\bin>mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.5.13 MySQL Community Server (GPL)
```

- 4. 只赋予普通用户必须的权限。

5.2.2 MYSQL数据库安全问题

- v5. 除root外，任何用户不能有user表的存取权限。
- v6. 不要[illegible]理员以外的账号。
- v7. LOA[illegible]
- v8. DRO[illegible]
- v9. 使用[illegible]
- v10. RE[illegible]

```
mysql> grant select,insert on st.* to zzh6@localhost identified by '123456';
Query OK, 0 rows affected (0.00 sec)

mysql> grant all privileges on *.* to zzh6@localhost;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> show grants for zzh6@localhost\G
*************************** 1. row ***************************
Grants for zzh6@localhost: GRANT ALL PRIVILEGES ON *.* TO 'zzh6'@'localhost' IDE
NTIFIED BY PASSWORD '*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9'
*************************** 2. row ***************************
Grants for zzh6@localhost: GRANT SELECT, INSERT ON `st`.* TO 'zzh6'@'localhost'
2 rows in set (0.00 sec)

mysql> revoke all privileges on *.* from zzh6@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql> show grants for zzh6@localhost\G
*************************** 1. row ***************************
Grants for zzh6@localhost: GRANT USAGE ON *.* TO 'zzh6'@'localhost' IDENTIFIED B
Y PASSWORD '*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9'
*************************** 2. row ***************************
Grants for zzh6@localhost: GRANT SELECT, INSERT ON `st`.* TO 'zzh6'@'localhost'
2 rows in set (0.00 sec)
```


5.2.3 MYSQL安全选项

- --local-infile[={0|1}]：如果用--local-infile=0启动服务器，则客户端不能使用LOCAL in LOAD DATA语句。
- --old-passwords：强制服务器为新密码生成短(pre-4.1)密码哈希。当服务器必须支持旧版本客户端程序时，为了保证兼容性这很有用。
- --safe-user-create：如果启用，用户不能用GRANT语句创建新用户，除非用户有mysql.user表的INSERT权限。如果你想让用户具有授权权限来创建新用户，你应给用户授予下面的权限：
- --secure-auth：不允许鉴定有旧(pre-4.1)密码的账户。
- --skip-grant-tables：这个选项导致服务器根本不使用权限系统。可在忘记root密码时使用,登录后用update语句修改USER表中的密码。
- --skip-name-resolve：主机名不被解析。所有在授权表的Host的列值必须是IP号或localhost。
- --skip-networking：在网络上不允许TCP/IP连接。

本章内容小结

- 5.1 MYSQL权限管理
  - 5.1.1 MYSQL权限管理的工作原理
  - 5.1.2 权限表的存取
  - 5.1.3 账号管理
- 5.2 MYSQL安全问题
  - 5.2.1 操作系统安全问题
  - 5.2.2 MYSQL数据库安全问题
  - 5.2.3 MYSQL安全选项