Neuedu

MySQL数据库开发技术

—— 用户和权限管理

V1.0

本章内容

节	知识点	掌握程度	难易程度
权限表	权限表	了解	
	权限表的结构和作用	了解	
	tables_priv表、columns_priv表和procs_priv表	了解	
权限系统的工作原理	权限系统的工作原理	了解	难
	连接核实阶段	了解	
	请求核实阶段	了解	难
账户管理	创建新用户	掌握	
	删除用户	掌握	
	修改用户名和密码	掌握	
授权	授权	掌握	
收回权限	收回权限	掌握	
查看权限	查看权限	了解	

权限表

- 通过网络连接服务器的客户对MySQL数据库的访问由权限表内容来控制。这些表位于MySQL数据库中，并在第1次安装MySQL的过程中初始化。权限表共有6个表：user、db、host、tables_priv、 columns_priv和procs_priv。
  - 当MySQL服务启动时，会首先读取MySQL中的权限表，并将表中的数据装入内存。当用户进行存取操作时，MySQL会根据这些表中的数据做相应的权限控制。

权限表

- **权限表user、db和host的结构和作用**
  - user表是MySQL中最重要的一个权限表，记录允许连接到服务器的账号信息。user表列出可以连接服务器的用户及其口令，并且指定他们有哪种全局（超级用户）权限。在user表启用的任何权限均是全局权限，并适用于所有数据库。例如，如果用户启用了DELETE权限，则该用户可以从任何表中删除记录。

权限表

- **权限表user、db和host的结构和作用**
  - db表和host表也是MySQL数据库中非常重要的权限表。db表中存储了用户对某个数据库的操作权限，决定用户能从哪个主机存取哪个数据库。
  - host表中存储了某个主机对数据库的操作权限，配合db权限表对给定主机上数据库级操作权限做更细致的控制。这个权限表不受GRANT和REVOKE语句的影响。db表比较常用，host表一般很少使用，新版本中已经没有host表了

权限表

- **tables_priv表、columns_priv表和procs_priv表**
  - tables_priv表用来对表设置操作权限
  - columns_priv表用来对表的某一列设置权限
  - procs_priv表可以对存储过程和存储函数设置操作权限

MySQL权限系统的工作原理

- 为了确保数据库的安全性与完整性，系统并不希望每个用户可以执行所有的数据库操作。当MySQL允许一个用户执行各种操作时，它将首先核实用户向MySQL服务器发送的连接请求，然后确认用户的操作请求是否被允许。
- MySQL的访问控制分为两个阶段：
  - 连接核实阶段
  - 请求核实阶段

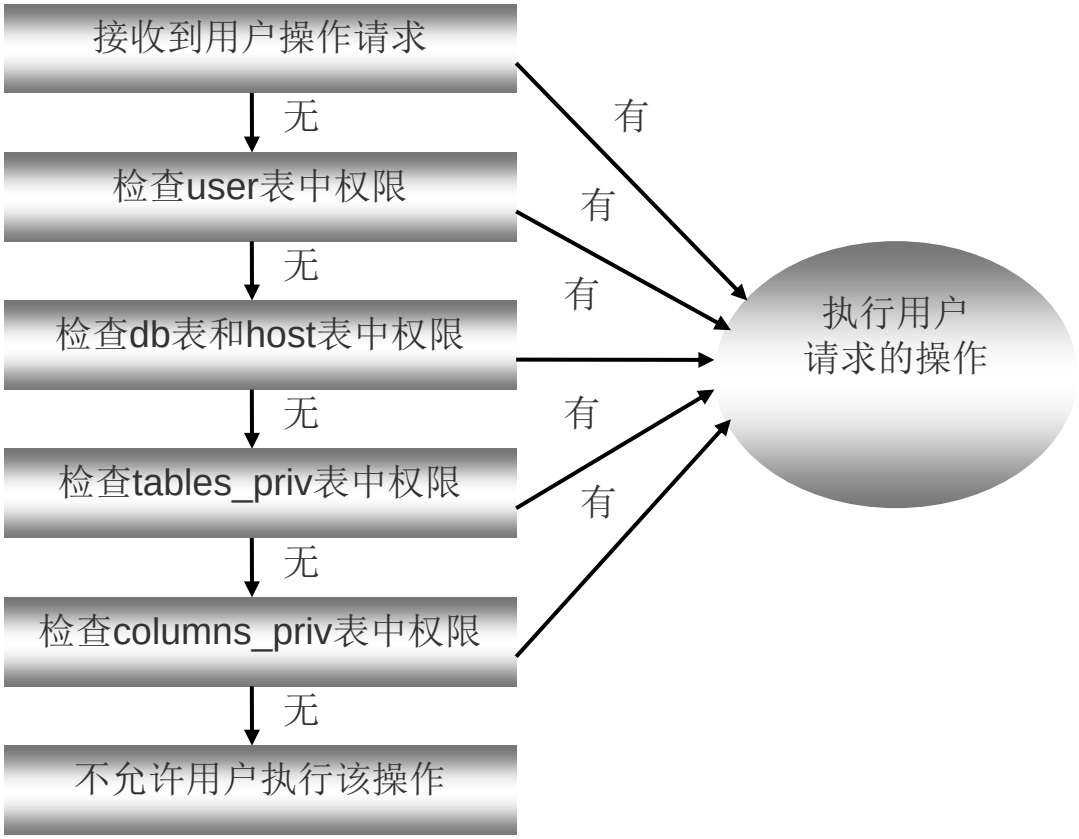
MySQL权限系统的工作原理

- **连接核实阶段**
  - 当用户试图连接MySQL服务器时，服务器基于用户提供的信息来验证用户身份，如果不能通过身份验证，服务器会完全拒绝该用户的访问。如果能够通过身份验证，则服务器接受连接，然后进入第2个阶段等待用户请求。
  - MySQL使用user表中的3个字段（Host、User和Password）进行身份检查，服务器只有在用户提供主机名、用户名和密码并与user表中对应的字段值完全匹配时才接受连接。新版本中Password字段名改为authentication_string。

MySQL权限系统的工作原理

- **请求核实阶段**
  - 一旦连接得到许可，服务器进入请求核实阶段。在这一阶段，MySQL服务器对当前用户的每个操作都进行权限检查，判断用户是否有足够的权限来执行它。用户的权限保存在user、db、host、tables_priv或columns_priv权限表中。
  - 在MySQL权限表的结构中，user表在最顶层，是全局级的。下面是db表和host表，它们是数据库层级的。最后才是tables_priv表和columns_priv表，它们是表级和列级的。低等级的表只能从高等级的表得到必要的范围或权限。
  - 确认权限时，MySQL首先检查user表，如果指定的权限没有在user表中被授权，MySQL服务器将检查db表和host表，在该层级的SELECT权限允许用户查看指定数据库的所有表的数据。如果在该层级没有找到限定的权限，则MySQL继续检查tables_priv表以及columns_priv表。如果所有权限表都检查完毕，依旧没有找到允许的权限操作，MySQL服务器将返回错误信息，用户操作不能执行，操作失败。

MySQL权限系统的工作原理



账户管理

- MySQL的账户管理包括登录和退出MySQL服务器、创建用户、删除用户、密码管理和权限管理等内容。通过账户管理，可以保证MySQL数据库的安全性。

账户管理

- 创建新用户
  - 使用CREATE USER语句创建新用户。
- 执行CREATE USER或GRANT语句时，服务器会有相应的用户权限表，添加或修改用户及其权限。
- CREATE USER语句的基本语法格式如下。

```
CREATE USER user[IDENTIFIED BY [PASSWORD]'password']
      [,user[IDENTIFIED BY [PASSWORD]'password']][,…];
```

账户管理

- 创建两个用户

```
mysql> create user 'ttcuser'@'localhost' identified
  by '123456';
mysql> select password('neusoft');
mysql> create user 'ttcuser2'@'localhost' identified
  by '*39F58E32475715731C339136692B4BCF344268E7';
```

```
mysql> create user 'ttcuser'@'localhost' identified by '123456';
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select password('neusoft');
+-------------------------------------------+
| password('neusoft')                       |
+-------------------------------------------+
| *39F58E32475715731C339136692B4BCF344268E7 |
+-------------------------------------------+
1 row in set, 1 warning (0.00 sec)

mysql> create user 'ttcuser2'@'localhost' identified by password '*3
9F58E32475715731C339136692B4BCF344268E7';
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

账户管理

- 创建新用户
  - 使用GRANT语句创建新用户。
- GRANT语句不仅可创建新用户，还可以在创建的同时对用户授权。GRANT语句还可以指定用户的其他特点，如安全连接、限制使用服务器资源等。使用GRANT语句创建新用户时必须有GRANT权限。
- 其基本语法格式如下。

```
GRANT  priv_type  ON database.table
    TO user [IDENTIFIED BY [PASSWORD] 'password']
        [, user [IDENTIFIED BY [PASSWORD] 'password']]
  [,…]
        [WITH GRANT OPTION];
```

账户管理

- 使用GRANT语句创建一个新用户test1，主机名为localhost，密码为123456，并授予所有数据表的SELECT和UPDATE权限。

```
mysql> grant select ,update on *.* to 'test1'@localhot
identified by '123456';
```

- MySQL建用户的时候会指定一个host，默认是127.0.0.1/localhost，那么这个用户就只能本机访问，其它机器用这个用户帐号访问会提示没有权限，host改为%，表示允许所有机器访问。

```
mysql> grant select ,update on *.* to 'test1'@%
  identified by '123456';
```

```
mysql> grant select ,update on *.* to 'test1'@localhot identified by '123456';
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

练习1

- 1.自己尝试创建一个用户user1,并授予所有数据表的SELECT和UPDATE权限

账户管理

- 删除账户
  - 使用DROP USER语句删除用户。
- DROP USER的语法格式如下。
  - DROP USER user_name[, user_name] [,…];
- DROP USER语句用于删除一个或多个MySQL账户，并取消其权限。要使用DROP USER，必须拥有MySQL数据库的全局CREATE USER权限或DELETE权限。
- 删除用户

```
mysql>DROP USER ttcuser2@localhost;
```

账户管理

- 修改用户名
  - 使用RENAME USER语句修改用户。
- 基本语法格式如下。

```
RENAME USER old_user TO new_user,
                    [, old_user TO new_user] [,…];
```

- 将用户haha的名字分别修改为haha2。

```
mysql> rename user haha@localhost to haha2@localhost;
```

```
mysql> rename user haha@localhost to haha2@localhost;
Query OK, 0 rows affected (0.00 sec)
```

账户管理

- 修改密码
  - 使用ALTER USER语句修改密码。
- 基本语法格式如下。

```
ALTER USER userIDENTIFIED BY 'password';
```

```
mysql> alter user haha2@localhost identified by '123456';
Query OK, 0 rows affected (0.00 sec)
```

账户管理

- 修改密码
  - 使用SET语句修改密码。
- 基本语法格式如下。
  - SET PASSWORD [FOR user]= PASSWORD('newpassword');
- 将用户king的密码修改为queen1。

  SET PASSWORD FOR ‘king’ @ ‘localhost’
  =PASSWORD(‘queen1');
- 将root用户的密码修改为123456。

  SET PASSWORD =PASSWORD('123456');

```
mysql> set password for haha2@localhost=password('123456');
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> set password =password('123456');
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

账户管理

- root用户修改自己的密码
  - 使用UPDATE语句修改MySQL数据库中的user表。因为所有账户信息都保存在user表中，因此可以通过直接修改user表来改变root用户的密码。
  - root用户登录到MySQL服务器后，使用UPDATE语句修改MySQL数据库中的user表的authentication_string字段值，从而修改用户密码。使用UPDATE语句修改root用户密码的语句如下。

```
update mysql.user set
  authentication_string=password('123qwe') where
  user='root'
```

```
mysql>  update mysql.user set authentication_string=password('123qwe') where user='root';
Query OK, 1 row affected, 1 warning (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 1
```

权限管理

- MySQL的权限类型
  - MySQL数据库中有多种类型的权限，这些权限都存储在MySQL数据库的权限表中。在MySQL启动时，服务器将这些数据库中的权限信息读入内存。

授权

- 授权就是为某个用户授予权限。在MySQL中，可以使用GRANT语句为用户授予权限。
- 权限的级别。授予的权限可以分为多层级别。
  - 全局层级。全局权限作用于一个给定服务器上的所有数据库。这些权限存储在mysql.user表中。可以通过使用GRANT ALL ON *.*语法设置全局权限。
  - 数据库层级。数据库权限作用于一个给定数据库的所有表。这些权限存储在mysql.db表中。读者可以通过使用GRANT ON db_name.*语法设置数据库权限。
  - 子程序层级。CREATE ROUTINE、ALTER ROUTINE、EXECUTE和GRANT权限适用于已存储的子程序（存储过程或函数）。这些权限可以被授予全局层级和数据库级别。而且，除了CREATE ROUTINE外，这些权限可以被授予子程序层级，并存储在mysql.procs_priv表中。

授权

- 权限的级别。授予的权限可以分为多层级别。
  - 表层级。表权限作用于一个给定表的所有列。这些权限存储在mysql.tables_priv表中。可以通过GRANT ON table_name为具体的表名设置权限。
  - 列层级。列权限作用于在一个给定表的单个列。这些权限存储在mysql.columns_priv表中。读者可以通过指定一个columns子句将权限授予特定的列，同时要在ON子句中指定具体的表。

授权

- 在MySQL中，必须是拥有GRANT权限的用户才可以执行GRANT语句。
- GRANT语句的基本语法格式如下。

```
GRANT priv_type [(column_list)] [,priv_type
  [(column_list)]] [,…n]
       ON
  {table_name|*|*.*|database_name.*|database_name.ta
  ble_name}
       TO user[IDENTIFIED BY [PASSWORD] 'password']
      [,user[IDENTIFIED BY [PASSWORD] 'password']]
  [,…n]
      [WITH GRANT OPTION];
```

授权

- 使用GRANT语句创建一个新用户grantUser，密码为“grantpwd”。用户grantUser对所有的数据有查询、插入权限，并授予GRANT权限。

```
mysql> grant select ,insert on *.* to
    grantUser@localhost identified by 'grantpwd' with
    grant option;
```

- 如果是远程用户，使用%

```
mysql> grant select ,insert on *.* to grantUser@%
    identified by 'grantpwd' with grant option;
```

```
mysql> grant select ,insert on *.* to grantUser@localhost identified by 'grantpwd' with grant option;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

授权

- 使用GRANT语句将mysqldb数据库中Employees表的DELETE权限授予用户grantUser。
- mysql> grant delete on mysqldb.Employees to grantUser@localhost;

- 使用GRANT语句将mysqldb数据库中Departments表的DepId列的UPDATE权限授予用户grantUser。
  mysql> grant update(DepId) on mysqldb.Departments to grantUser@localhost;

```
mysql> grant delete on mysqldb.Employees to grantUser@localhost;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> grant update(DepId) on mysqldb.Departments to grantUser@localhost;
Query OK, 0 rows affected (0.00 sec)
```

收回权限

- 收回权限就是取消已经赋予用户的某些权限。收回用户不必要的权限在一定程度上可以保证数据的安全性。权限收回后，用户账户的记录将从db、tables_priv和columns_priv表中删除，但是用户账户记录仍然在user表中保存。收回权限利用REVOKE语句来实现，语法格式有两种，一种是收回用户的所有权限，另一种是收回用户指定的权限。

收回权限

- 收回所有权限。其基本语法如下。

```
REVOKE ALL PRIVILEGES,GRANT OPTION
FROM
   'username'@'hostname'[,'username'@'hostname'][,…n
   ];
```

- 使用REVOKE语句收回grantUser用户的的所有权限，包括GRANT权限。

```
mysql> revoke all privileges,grant option from
grantUser@localhost;
```

```
mysql> revoke all privileges,grant option from grantUser@localhost;
Query OK, 0 rows affected (0.00 sec)
```

收回权限

- 收回指定权限，其基本语法如下。

```
REVOKE priv_type [(column_list)] [,priv_type
  [(column_list)]] [,…n]
      ON
  {table_name|*|*.*|database_name.*|database_name.ta
  ble_name}
      FROM
  'username'@'hostname'[,'username'@'hostname'][,…n
  ];
```

- 收回用户grantUser 对mysqldb数据库中Departments表的DepId列的UPDATE权限。

```
mysql> revoke update(DepId) on mysqldb.Departments
from grantUser@localhost;
```

```
mysql> revoke update(DepId) on mysqldb.Departments from grantUser@localhost;
Query OK, 0 rows affected (0.00 sec)
```

查看权限

- SHOW GRANTS语句可以显示指定用户的权限信息，使用SHOW GRANTS查看账户权限信息的基本语法格式如下。
- SHOW GRANTS FOR 'username'@'hostname';

- 使用SHOW GRANTS语句查看root用户的权限信息。
  mysql> show grants for root@localhost;

```
mysql> show grants for root@localhost;
+-----------------------------------------------------------------------------------------+
| Grants for root@localhost                                                               |
+-----------------------------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION                     |
| GRANT SELECT (DepId), UPDATE (DepId) ON `mysqldb`.`Departments` TO 'root'@'localhost' |
| GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH GRANT OPTION                            |
+-----------------------------------------------------------------------------------------+
3 rows in set (0.00 sec)
```

小结

- 掌握MySQL用户和权限的用法
  - 理解MySQL的安全机制
  - MySQL的用户管理
  - MySQL权限操作

Neuedu

课后作业

- 1. 建立新用户neu
- 2. 给用户neu授权
- 3. 回收用户neu的权限