第03章_用户与权限管理

1. 用户管理

1.1 登录MySQL服务器

启动MySQL服务后，可以通过mysql命令来登录MySQL服务器，命令如下：

```
mysql -h hostname|hostIP -P port -u username -p DatabaseName -e "SQL语句"
```


下面详细介绍命令中的参数：

- `-h参数` 后面接主机名或者主机IP，hostname为主机，hostIP为主机IP。
- `-P参数` 后面接MySQL服务的端口，通过该参数连接到指定的端口。MySQL服务的默认端口是3306，不使用该参数时自动连接到3306端口，port为连接的端口号。
- `-u参数` 后面接用户名，username为用户名。
- `-p参数` 会提示输入密码。
- `DatabaseName参数` 指明登录到哪一个数据库中。如果没有该参数，就会直接登录到MySQL数据库中，然后可以使用USE命令来选择数据库。
- `-e参数` 后面可以直接加SQL语句。登录MySQL服务器以后即可执行这个SQL语句，然后退出MySQL服务器。

举例：

```
mysql -uroot -p -hlocalhost -P3306 mysql -e "select host,user from user"
```

1.2 创建用户

CREATE USER语句的基本语法形式如下：

```
CREATE USER 用户名 [IDENTIFIED BY '密码'][,用户名 [IDENTIFIED BY '密码']];
```

- 用户名参数表示新建用户的账户，由 `用户（User）` 和 `主机名（Host）` 构成；
- “[]”表示可选，也就是说，可以指定用户登录时需要密码验证，也可以不指定密码验证，这样用户可以直接登录。不过，不指定密码的方式不安全，不推荐使用。如果指定密码值，这里需要使用IDENTIFIED BY指定明文密码值。
- CREATE USER语句可以同时创建多个用户。

举例：

```
CREATE USER zhang3 IDENTIFIED BY '123123';  # 默认host是 %
```

```
CREATE USER 'kangshifu'@'localhost' IDENTIFIED BY '123456';
```

1.3 修改用户

修改用户名：

```
UPDATE mysql.user SET USER='li4' WHERE USER='wang5';

FLUSH PRIVILEGES;
```

```
mysql> update mysql.user set user='li4' where user='wang5';
Query OK, 1 row affected (0.27 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

1.4 删除用户

方式1：使用DROP方式删除（推荐）

使用DROP USER语句来删除用户时，必须用于DROP USER权限。DROP USER语句的基本语法形式如下：

```
DROP USER user[,user]…;
```

举例：

```
DROP USER li4 ;  # 默认删除host为%的用户
```

```
DROP USER 'kangshifu'@'localhost';
```

```
mysql> drop user li4;
Query OK, 0 rows affected (0.00 sec)

mysql> select host,user,password,select_priv,insert_priv,drop_priv from mysql.user;
+--------------+------+-------------------------------------------+-------------+-------------+-----------+
| host         | user | password                                  | select_priv | insert_priv | drop_priv |
+--------------+------+-------------------------------------------+-------------+-------------+-----------+
| %            | root | *E56A114692FE0DE073F9A1DD68A00EEB9703F3F1 | Y           | Y           | Y         |
| jack.atguigu | root |                                           | Y           | Y           | Y         |
| 127.0.0.1    | root |                                           | Y           | Y           | Y         |
| ::1          | root |                                           | Y           | Y           | Y         |
| localhost    |      |                                           | N           | N           | N         |
| jack.atguigu |      |                                           | N           | N           | N         |
| localhost    | root | *E56A114692FE0DE073F9A1DD68A00EEB9703F3F1 | Y           | Y           | Y         |
+--------------+------+-------------------------------------------+-------------+-------------+-----------+
7 rows in set (0.00 sec)
```

方式2：使用DELETE方式删除

```
DELETE FROM mysql.user WHERE Host='hostname' AND User='username';
```

执行完DELETE命令后要使用FLUSH命令来使用户生效，命令如下：

```
FLUSH PRIVILEGES;
```

举例：

```
DELETE FROM mysql.user WHERE Host='localhost' AND User='Emily';

FLUSH PRIVILEGES;
```

> 注意：不推荐通过 `DELETE FROM USER u WHERE USER='li4'` 进行删除，系统会有残留信息保留。而drop user命令会删除用户以及对应的权限，执行命令后你会发现mysql.user表和mysql.db表的相应记录都消失了。

1.5 设置当前用户密码

旧的写法如下：

```
# 修改当前用户的密码：（MySQL5.7测试有效）
SET PASSWORD = PASSWORD('123456');
```

这里介绍推荐的写法：

1. 使用ALTER USER命令来修改当前用户密码 用户可以使用ALTER命令来修改自身密码，如下语句代表修改当前登录用户的密码。基本语法如下：

```
ALTER USER USER() IDENTIFIED BY 'new_password';
```

2. 使用SET语句来修改当前用户密码 使用root用户登录MySQL后，可以使用SET语句来修改密码，具体SQL语句如下：

```
SET PASSWORD='new_password';
```

该语句会自动将密码加密后再赋给当前用户。

1.6 修改其它用户密码

1. 使用ALTER语句来修改普通用户的密码 可以使用ALTER USER语句来修改普通用户的密码。基本语法形式如下：

```
ALTER USER user [IDENTIFIED BY '新密码']
[,user[IDENTIFIED BY '新密码']]…;
```

2. 使用SET命令来修改普通用户的密码 使用root用户登录到MySQL服务器后，可以使用SET语句来修改普通用户的密码。SET语句的代码如下：

```
SET PASSWORD FOR 'username'@'hostname'='new_password';
```

3. 使用UPDATE语句修改普通用户的密码（不推荐）

```
UPDATE MySQL.user SET authentication_string=PASSWORD("123456")
WHERE User = "username" AND Host = "hostname";
```

1.7 MySQL8密码管理(了解)

1. 密码过期策略

- 在MySQL中，数据库管理员可以手动设置账号密码过期，也可以建立一个自动密码过期策略。
- 过期策略可以是全局的，也可以为每个账号设置单独的过期策略。

```
ALTER USER user PASSWORD EXPIRE;
```

练习：

```
ALTER USER 'kangshifu'@'localhost' PASSWORD EXPIRE;
```

- **方式①：使用SQL语句更改该变量的值并持久化**

```
SET PERSIST default_password_lifetime = 180; # 建立全局策略，设置密码每隔180天过期
```

- **方式②：配置文件my.cnf中进行维护**

```
[mysqld]
default_password_lifetime=180  #建立全局策略，设置密码每隔180天过期
```

手动设置指定时间过期方式2：单独设置

每个账号既可延用全局密码过期策略，也可单独设置策略。在 `CREATE USER` 和 `ALTER USER` 语句上加入 `PASSWORD EXPIRE` 选项可实现单独设置策略。下面是一些语句示例。

```
#设置kangshifu账号密码每90天过期：
CREATE USER 'kangshifu'@'localhost' PASSWORD EXPIRE INTERVAL 90 DAY;
ALTER USER 'kangshifu'@'localhost' PASSWORD EXPIRE INTERVAL 90 DAY;

#设置密码永不过期：
CREATE USER 'kangshifu'@'localhost' PASSWORD EXPIRE NEVER;
ALTER USER 'kangshifu'@'localhost' PASSWORD EXPIRE NEVER;

#延用全局密码过期策略：
CREATE USER 'kangshifu'@'localhost' PASSWORD EXPIRE DEFAULT;
ALTER USER 'kangshifu'@'localhost' PASSWORD EXPIRE DEFAULT;
```

2. 密码重用策略

- **手动设置密码重用方式1：全局**
  - **方式①：使用SQL**

```
SET PERSIST password_history = 6; #设置不能选择最近使用过的6个密码

SET PERSIST password_reuse_interval = 365; #设置不能选择最近一年内的密码
```

```
mysql> SET PERSIST password_history = 6;
Query OK, 0 rows affected (0.00 sec)

mysql> SET PERSIST password_reuse_interval = 365;
Query OK, 0 rows affected (0.00 sec)
```

  - **方式②：my.cnf配置文件**

```
[mysqld]
password_history=6
password_reuse_interval=365
```

- **手动设置密码重用方式2：单独设置**

```
#不能使用最近5个密码：
CREATE USER 'kangshifu'@'localhost' PASSWORD HISTORY 5;
ALTER USER 'kangshifu'@'localhost' PASSWORD HISTORY 5;

#不能使用最近365天内的密码：
CREATE USER 'kangshifu'@'localhost' PASSWORD REUSE INTERVAL 365 DAY;
ALTER USER 'kangshifu'@'localhost' PASSWORD REUSE INTERVAL 365 DAY;

#既不能使用最近5个密码，也不能使用365天内的密码
CREATE USER 'kangshifu'@'localhost'
PASSWORD HISTORY 5
PASSWORD REUSE INTERVAL 365 DAY;

ALTER USER 'kangshifu'@'localhost'
```

```
PASSWORD HISTORY 5
PASSWORD REUSE INTERVAL 365 DAY;
```

2. 权限管理

2.1 权限列表

MySQL到底都有哪些权限呢?

```
mysql> show privileges;
```

（1）`CREATE和DROP权限`，可以创建新的数据库和表，或删除（移掉）已有的数据库和表。如果将MySQL数据库中的DROP权限授予某用户，用户就可以删除MySQL访问权限保存的数据库。（2）`SELECT、INSERT、UPDATE和DELETE权限`允许在一个数据库现有的表上实施操作。（3）`SELECT权限`只有在它们真正从一个表中检索行时才被用到。（4）`INDEX权限`允许创建或删除索引，INDEX适用于已有的表。如果具有某个表的CREATE权限，就可以在CREATE TABLE语句中包括索引定义。（5）`ALTER权限`可以使用ALTER TABLE来更改表的结构和重新命名表。（6）`CREATE ROUTINE权限`用来创建保存的程序（函数和程序），ALTER ROUTINE权限用来更改和删除保存的程序，`EXECUTE权限`用来执行保存的程序。（7）`GRANT权限`允许授权给其他用户，可用于数据库、表和保存的程序。（8）`FILE权限`使用户可以使用LOAD DATA INFILE和SELECT … INTO OUTFILE语句读或写服务器上的文件，任何被授予FILE权限的用户都能读或写MySQL服务器上的任何文件（说明用户可以读任何数据库目录下的文件，因为服务器可以访问这些文件）。

2.2 授予权限的原则

权限控制主要是出于安全因素，因此需要遵循以下几个`经验原则`：

1、只授予能`满足需要的最小权限`，防止用户干坏事。比如用户只是需要查询，那就只给select权限就可以了，不要给用户赋予update、insert或者delete权限。

2、创建用户的时候`限制用户的登录主机`，一般是限制成指定IP或者内网IP段。

3、为每个用户`设置满足密码复杂度的密码`。

4、`定期清理不需要的用户`，回收权限或者删除用户。

2.3 授予权限

给用户授权的方式有 2 种，分别是通过把`角色赋予用户给用户授权`和`直接给用户授权`。用户是数据库的使用者，我们可以通过给用户授予访问数据库中资源的权限，来控制使用者对数据库的访问，消除安全隐患。

授权命令：

```
GRANT 权限1,权限2,…权限n ON 数据库名称.表名称 TO 用户名@用户地址 [IDENTIFIED BY '密码口令'];
```

- 该权限如果发现没有该用户，则会直接新建一个用户。

比如：

- 给li4用户用本地命令行方式，授予atguigudb这个库下的所有表的插删改查的权限。

  ```
  GRANT SELECT,INSERT,DELETE,UPDATE ON atguigudb.* TO li4@localhost ;
  ```

- 授予通过网络方式登录的joe用户 ，对所有库所有表的全部权限，密码设为123。注意这里唯独不包括grant的权限

```
GRANT ALL PRIVILEGES ON *.* TO joe@'%' IDENTIFIED BY '123';
```

> 我们在开发应用的时候，经常会遇到一种需求，就是要根据用户的不同，对数据进行横向和纵向的分组。
>
> - 所谓横向的分组，就是指用户可以接触到的数据的范围，比如可以看到哪些表的数据；
> - 所谓纵向的分组，就是指用户对接触到的数据能访问到什么程度，比如能看、能改，甚至是删除。

2.4 查看权限

- 查看当前用户权限

```
SHOW GRANTS;
# 或
SHOW GRANTS FOR CURRENT_USER;
# 或
SHOW GRANTS FOR CURRENT_USER();
```

- 查看某用户的全局权限

```
SHOW GRANTS FOR 'user'@'主机地址' ;
```

2.5 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可以在一定程度上保证系统的安全性。**MySQL中使用`REVOKE语句`取消用户的某些权限。使用REVOKE收回权限之后，用户账户的记录将从db、host、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（删除user表中的账户记录使用DROP USER语句）。

注意：在将用户账户从user表删除之前，应该收回相应用户的所有权限。

- 收回权限命令

```
REVOKE 权限1,权限2,…权限n ON 数据库名称.表名称 FROM 用户名@用户地址;
```

- 举例

```
#收回全库全表的所有权限
REVOKE ALL PRIVILEGES ON *.* FROM joe@'%';

#收回mysql库下的所有表的插删改查权限
REVOKE SELECT,INSERT,UPDATE,DELETE ON mysql.* FROM joe@localhost;
```

- 注意：`须用户重新登录后才能生效`

3. 权限表

3.1 user表

user表是MySQL中最重要的一个权限表，记录用户账号和权限信息，有49个字段。如下图：

字段名	数据类型	默认值
Host	char(60)	
User	char(16)	
authentication_string	text	
Select_priv	enum('N','Y')	N
Insert_priv	enum('N','Y')	N
Update_priv	enum('N','Y')	N
Delete_priv	enum('N','Y')	N
Create_priv	enum('N','Y')	N
Drop_priv	enum('N','Y')	N
Reload_priv	enum('N','Y')	N
Shutdown_priv	enum('N','Y')	N
Process_priv	enum('N','Y')	N
File_priv	enum('N','Y')	N
Grant_priv	enum('N','Y')	N
References_priv	enum('N','Y')	N
Index_priv	enum('N','Y')	N
Alter_priv	enum('N','Y')	N
Show_db_priv	enum('N','Y')	N
Super_priv	enum('N','Y')	N
Create_tmp_table_priv	enum('N','Y')	N
Lock_tables_priv	enum('N','Y')	N
Execute_priv	enum('N','Y')	N
Repl_slave_priv	enum('N','Y')	N
Repl_client_priv	enum('N','Y')	N
Create_view_priv	enum('N','Y')	N
Show_view_priv	enum('N','Y')	N
Create_routine_priv	enum('N','Y')	N
Alter_routine_priv	enum('N','Y')	N
Create_user_priv	enum('N','Y')	N
Event_priv	enum('N','Y')	N
Trigger_priv	enum('N','Y')	N
Create_tablespace_priv	enum('N','Y')	N
ssl_type	enum('','ANY','X509','SPECIFIED')	
ssl_cipher	blob	NULL
x509_issuer	blob	NULL
x509_subject	blob	NULL
max_questions	int(11) unsigned	0
max_updates	int(11) unsigned	0
max_connections	int(11) unsigned	0
max_user_connections	int(11) unsigned	0
plugin	char(64)	
authentication_string	text	NULL

这些字段可以分成4类，分别是范围列（或用户列）、权限列、安全列和资源控制列。

1.范围列（或用户列）

- host： 表示连接类型
  - `%` 表示所有远程通过 TCP方式的连接
  - `IP 地址` 如 (192.168.1.2、127.0.0.1) 通过制定ip地址进行的TCP方式的连接
  - `机器名` 通过制定网络中的机器名进行的TCP方式的连接
  - `::1` IPv6的本地ip地址，等同于IPv4的 127.0.0.1
  - `localhost` 本地方式通过命令行方式的连接 ，比如mysql -u xxx -p xxx 方式的连接。
- user： 表示用户名，同一用户通过不同方式链接的权限是不一样的。
- password： 密码
  - 所有密码串通过 password(明文字符串) 生成的密文字符串。MySQL 8.0 在用户管理方面增加了角色管理，默认的密码加密方式也做了调整，由之前的 `SHA1` 改为了 `SHA2` ，不可逆 。同时加上 MySQL 5.7 的禁用用户和用户过期的功能，MySQL 在用户管理方面的功能和安全性都较之前版本大大的增强了。
  - mysql 5.7 及之后版本的密码保存到 `authentication_string` 字段中不再使用password 字段。

2. 权限列

- Grant_priv字段
  - 表示是否拥有GRANT权限
- Shutdown_priv字段
  - 表示是否拥有停止MySQL服务的权限
- Super_priv字段
  - 表示是否拥有超级权限
- Execute_priv字段
  - 表示是否拥有EXECUTE权限。拥有EXECUTE权限，可以执行存储过程和函数。
- Select_priv , Insert_priv等
  - 为该用户所拥有的权限。

3. 安全列 安全列只有6个字段，其中两个是ssl相关的（ssl_type、ssl_cipher），用于 `加密` ；两个是x509相关的（x509_issuer、x509_subject），用于 `标识用户` ；另外两个Plugin字段用于 `验证用户身份` 的插件，该字段不能为空。如果该字段为空，服务器就使用内建授权验证机制验证用户身份。

4. 资源控制列 资源控制列的字段用来 `限制用户使用的资源` ，包含4个字段，分别为：

①max_questions，用户每小时允许执行的查询操作次数； ②max_updates，用户每小时允许执行的更新操作次数； ③max_connections，用户每小时允许执行的连接操作次数； ④max_user_connections，用户允许同时建立的连接次数。

查看字段：

```
DESC mysql.user;
```

查看用户, 以列的方式显示数据：

```
SELECT * FROM mysql.user \G;
```

查询特定字段：

```
SELECT host,user,authentication_string,select_priv,insert_priv,drop_priv
FROM mysql.user;
```

```
mysql> select host,user,authentication_string,Select_priv from user;
+-----------+-----------+-------------------------------------------+-------------+
| host      | user      | authentication_string                     | Select_priv |
+-----------+-----------+-------------------------------------------+-------------+
| localhost | root      | *E56A114692FE0DE073F9A1DD68A00EEB9703F3F1 | Y           |
| localhost | mysql.sys | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | N           |
| %         | zhang3    | *E56A114692FE0DE073F9A1DD68A00EEB9703F3F1 | N           |
| %         | root      | *E56A114692FE0DE073F9A1DD68A00EEB9703F3F1 | Y           |
+-----------+-----------+-------------------------------------------+-------------+
4 rows in set (0.01 sec)
```

3.2 db表

使用DESCRIBE查看db表的基本结构：

```
DESCRIBE mysql.db;
```

1. 用户列 db表用户列有3个字段，分别是Host、User、Db。这3个字段分别表示主机名、用户名和数据库名。表示从某个主机连接某个用户对某个数据库的操作权限，这3个字段的组合构成了db表的主键。

2. 权限列

Create_routine_priv和Alter_routine_priv这两个字段决定用户是否具有创建和修改存储过程的权限。

3.3 tables_priv表和columns_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`。tables_priv表和columns_priv表的结构分别如图：

```
desc mysql.tables_priv;
```

tables_priv表有8个字段，分别是Host、Db、User、Table_name、Grantor、Timestamp、Table_priv和Column_priv，各个字段说明如下：

- `Host`、`Db`、`User`和`Table_name`四个字段分别表示主机名、数据库名、用户名和表名。
- Grantor表示修改该记录的用户。
- Timestamp表示修改该记录的时间。
- `Table_priv`表示对象的操作权限。包括Select、Insert、Update、Delete、Create、Drop、Grant、References、Index和Alter。
- Column_priv字段表示对表中的列的操作权限，包括Select、Insert、Update和References。

```
desc mysql.columns_priv;
```

3.4 procs_priv表

procs_priv表可以对`存储过程和存储函数设置操作权限`，表结构如图：

```
desc mysql.procs_priv;
```

字段名	数据类型	默认值
Host	char(60)	
Db	char(64)	
User	char(16)	
Routine_name	char(64)	
Routine_type	enum('FUNCTION','PROCEDURE')	NULL
Grantor	char(77)	
Proc_priv	set('Execute','Alter Routine','Grant')	
Timestamp	timestamp	CURRENT_TIMESTAMP

4. 访问控制(了解)

4.1 连接核实阶段

当用户试图连接MySQL服务器时，服务器基于用户的身份以及用户是否能提供正确的密码验证身份来确定接受或者拒绝连接。即客户端用户会在连接请求中提供用户名、主机地址、用户密码，MySQL服务器接收到用户请求后，会**使用user表中的host、user和authentication_string这3个字段匹配客户端提供信息**。

服务器只有在user表记录的Host和User字段匹配客户端主机名和用户名，并且提供正确的密码时才接受连接。**如果连接核实没有通过，服务器就完全拒绝访问**；**否则，服务器接受连接，然后进入阶段2等待用户请求。**

4.2 请求核实阶段

一旦建立了连接，服务器就进入了访问控制的阶段2，也就是请求核实阶段。对此连接上进来的每个请求，服务器检查该请求要执行什么操作、是否有足够的权限来执行它，这正是需要授权表中的权限列发挥作用的地方。这些权限可以来自user、db、table_priv和column_priv表。

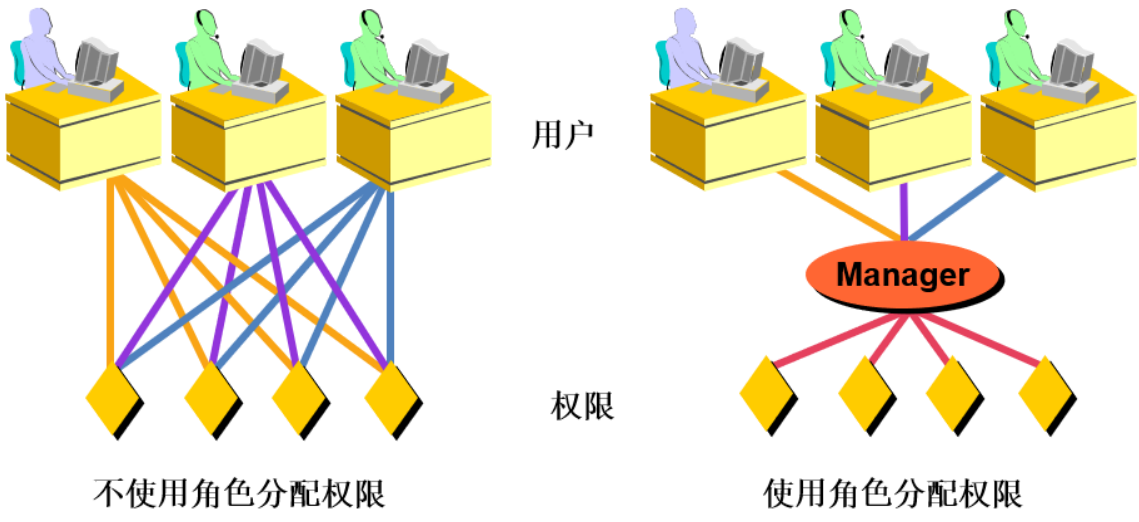
确认权限时，MySQL首先`检查user表`，如果指定的权限没有在user表中被授予，那么MySQL就会继续`检查db表`，db表是下一安全层级，其中的权限限定于数据库层级，在该层级的SELECT权限允许用户查看指定数据库的所有表中的数据；如果在该层级没有找到限定的权限，则MySQL继续`检查tables_priv表`以及`columns_priv表`，如果所有权限表都检查完毕，但还是没有找到允许的权限操作，MySQL将`返回错误信息`，用户请求的操作不能执行，操作失败。

> 提示： MySQL通过向下层级的顺序（从user表到columns_priv表）检查权限表，但并不是所有的权限都要执行该过程。例如，一个用户登录到MySQL服务器之后只执行对MySQL的管理操作，此时只涉及管理权限，因此MySQL只检查user表。另外，如果请求的权限操作不被允许，MySQL也不会继续检查下一层级的表。

5. 角色管理

5.1 角色的理解

引入角色的目的是`方便管理拥有相同权限的用户`。**恰当的权限设定，可以确保数据的安全性，这是至关重要的。**



5.2 创建角色

创建角色使用`CREATE ROLE`语句，语法如下：

```
CREATE ROLE 'role_name'[@'host_name'] [,'role_name'[@'host_name']]...
```

角色名称的命名规则和用户名类似。如果`host_name省略，默认为%`，`role_name不可省略`，不可为空。

练习：我们现在需要创建一个经理的角色，就可以用下面的代码：

```
CREATE ROLE 'manager'@'localhost';
```

5.3 给角色赋予权限

创建角色之后，默认这个角色是没有任何权限的，我们需要给角色授权。给角色授权的语法结构是：

```
GRANT privileges ON table_name TO 'role_name'[@'host_name'];
```

上述语句中privileges代表权限的名称，多个权限以逗号隔开。可使用SHOW语句查询权限名称，图11-43列出了部分权限列表。

```
SHOW PRIVILEGES\G;
```

```
mysql> show privileges\G
*************************** 1. row ***************************
Privilege: Alter
  Context: Tables
  Comment: To alter the table
*************************** 2. row ***************************
Privilege: Alter routine
  Context: Functions,Procedures
  Comment: To alter or drop stored functions/procedures
*************************** 3. row ***************************
Privilege: Create
  Context: Databases,Tables,Indexes
  Comment: To create new databases and tables
*************************** 4. row ***************************
```

```
*************************** 60. row ***************************
Privilege: INNODB_REDO_LOG_ENABLE
  Context: Server Admin
  Comment:
*************************** 61. row ***************************
Privilege: INNODB_REDO_LOG_ARCHIVE
  Context: Server Admin
  Comment:
*************************** 62. row ***************************
Privilege: REPLICATION_APPLIER
  Context: Server Admin
  Comment:
62 rows in set (0.00 sec)
```

练习1：我们现在想给经理角色授予商品信息表、盘点表和应付账款表的只读权限，就可以用下面的代码来实现：

```
GRANT SELECT ON demo.settlement TO 'manager';

GRANT SELECT ON demo.goodsmaster TO 'manager';

GRANT SELECT ON demo.invcount TO 'manager';
```

5.4 查看角色的权限

赋予角色权限之后，我们可以通过 SHOW GRANTS 语句，来查看权限是否创建成功了：

```
mysql> SHOW GRANTS FOR 'manager';
+-----------------------------------------------------------+
| Grants for manager@% |
+-----------------------------------------------------------+
| GRANT USAGE ON *.* TO `manager`@`%` |
| GRANT SELECT ON `demo`.`goodsmaster` TO `manager`@`%` |
| GRANT SELECT ON `demo`.`invcount` TO `manager`@`%` |
| GRANT SELECT ON `demo`.`settlement` TO `manager`@`%` |
+-----------------------------------------------------------+
```

只要你创建了一个角色，系统就会自动给你一个“ USAGE ”权限，意思是 连接登录数据库的权限 。代码的最后三行代表了我们给角色“manager”赋予的权限，也就是对商品信息表、盘点表和应付账款表的只读权限。

结果显示，库管角色拥有商品信息表的只读权限和盘点表的增删改查权限。

5.5 回收角色的权限

角色授权后，可以对角色的权限进行维护，对权限进行添加或撤销。添加权限使用GRANT语句，与角色授权相同。撤销角色或角色权限使用REVOKE语句。

修改了角色的权限，会影响拥有该角色的账户的权限。

撤销角色权限的SQL语法如下：

```
REVOKE privileges ON tablename FROM 'rolename';
```

练习1：撤销school_write角色的权限。 （1）使用如下语句撤销school_write角色的权限。

```
REVOKE INSERT, UPDATE, DELETE ON school.* FROM 'school_write';
```

（2）撤销后使用SHOW语句查看school_write对应的权限，语句如下。

```
SHOW GRANTS FOR 'school_write';
```

5.6 删除角色

当我们需要对业务重新整合的时候，可能就需要对之前创建的角色进行清理，删除一些不会再使用的角色。删除角色的操作很简单，你只要掌握语法结构就行了。

```
DROP ROLE role [,role2]...
```

注意， 如果你删除了角色，那么用户也就失去了通过这个角色所获得的所有权限 。

练习：执行如下SQL删除角色school_read。

```
DROP ROLE 'school_read';
```

5.7 给用户赋予角色

角色创建并授权后，要赋给用户并处于`激活状态`才能发挥作用。给用户添加角色可使用GRANT语句，语法形式如下：

```
GRANT role [,role2,...] TO user [,user2,...];
```

在上述语句中，role代表角色，user代表用户。可将多个角色同时赋予多个用户，用逗号隔开即可。

练习：给kangshifu用户添加角色school_read权限。 （1）使用GRANT语句给kangshifu添加school_read权限，SQL语句如下。

```
GRANT 'school_read' TO 'kangshifu'@'localhost';
```

（2）添加完成后使用SHOW语句查看是否添加成功，SQL语句如下。

```
SHOW GRANTS FOR 'kangshifu'@'localhost';
```

（3）使用kangshifu用户登录，然后查询当前角色，如果角色未激活，结果将显示NONE。SQL语句如下。

```
SELECT CURRENT_ROLE();
```

```
mysql> select current_role();
+----------------+
| current_role() |
+----------------+
| NONE           |
+----------------+
1 row in set (0.00 sec)
```

5.8 激活角色

方式1：使用set default role 命令激活角色

举例：

```
SET DEFAULT ROLE ALL TO 'kangshifu'@'localhost';
```

举例：使用`SET DEFAULT ROLE`为下面4个用户默认激活所有已拥有的角色如下：

```
SET DEFAULT ROLE ALL TO
  'dev1'@'localhost',
  'read_user1'@'localhost',
  'read_user2'@'localhost',
  'rw_user1'@'localhost';
```

方式2：将activate_all_roles_on_login设置为ON

- 默认情况：

```
mysql> show variables like 'activate_all_roles_on_login';
+------------------------------+-------+
| Variable_name                | Value |
+------------------------------+-------+
| activate_all_roles_on_login | OFF   |
+------------------------------+-------+
1 row in set (0.00 sec)
```

- 设置：

```
SET GLOBAL activate_all_roles_on_login=ON;
```

这条 SQL 语句的意思是，对`所有角色永久激活`。运行这条语句之后，用户才真正拥有了赋予角色的所有权限。

5.9 撤销用户的角色

撤销用户角色的SQL语法如下：

```
REVOKE role FROM user;
```

练习：撤销kangshifu用户的school_read角色。 （1）撤销的SQL语句如下

```
REVOKE 'school_read' FROM 'kangshifu'@'localhost';
```

（2）撤销后，执行如下查询语句，查看kangshifu用户的角色信息

```
SHOW GRANTS FOR 'kangshifu'@'localhost';
```

执行发现，用户kangshifu之前的school_read角色已被撤销。

5.10 设置强制角色(mandatory role)

方式1：服务启动前设置

```
[mysqld]
mandatory_roles='role1,role2@localhost,r3@%.atguigu.com'
```

方式2：运行时设置

```
SET PERSIST mandatory_roles = 'role1,role2@localhost,r3@%.example.com';  #系统重启后仍然有效
SET GLOBAL mandatory_roles = 'role1,role2@localhost,r3@%.example.com';  #系统重启后失效
```