1. 用户管理

MySQL用户可以分为**普通用户**和**root**用户

root用户是超级管理员,拥有所有权限,包括创建用户、删除用户和修改用户的密码等管理权限;

普通用户只拥有被授予的各种权限

`MySQL提供了许多语句用来管理用户账号`,这些语句可以用来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

`MySQL数据库的安全性需要通过账户管理来保证`

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 -h192.168.175.135 -P3306 mysql -e "select host,user from user"
Enter password: *******

+-----------+------------------+
| host      | user             |
+-----------+------------------+
| %         | root             |
| localhost | mysql.infoschema |
| localhost | mysql.session    |
| localhost | mysql.sys        |
+-----------+------------------+
```

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：

```
-- 新建用户
CREATE USER zhang3 IDENTIFIED BY '123123'; # 默认host是 %

Query OK, 0 rows affected (0.02 sec)



-- 切换数据库
use mysql;

Database changed
```

```
-- 查看用户表
SELECT user,host FROM user;

+------------------+-----------+
| user             | host      |
+------------------+-----------+
| root             | %         |
| zhang3           | %         |
| mysql.infoschema | localhost |
| mysql.session    | localhost |
| mysql.sys        | localhost |
+------------------+-----------+
5 rows in set (0.00 sec)



-- 新建用户
CREATE USER zhang3 IDENTIFIED BY '123123'; # 默认host是 %

-- 不能重复创建
ERROR 1396 (HY000): Operation CREATE USER failed for 'zhang3'@'%'




-- 新建用户 指定host
CREATE USER 'zhang3'@'localhost' IDENTIFIED BY '123123'; # 默认host是 %

Query OK, 0 rows affected (0.00 sec)



-- 新建用户 指定host
create user 'li4'@'%' identified by '123123';

Query OK, 0 rows affected (0.01 sec)




-- 查看用户表
SELECT user,host FROM user;

+------------------+-----------+
| user             | host      |
+------------------+-----------+
| li4              | %         |
| root             | %         |
| zhang3           | %         |
```

```
| mysql.infoschema | localhost |
| mysql.session    | localhost |
| mysql.sys        | localhost |
| zhang3           | localhost |
+------------------+-----------+
7 rows in set (0.00 sec)
```

```
-- 尝试登陆
mysql -uli4 -h192.168.175.135 -p
Enter password: ******
```

1.3 修改用户

```
-- 修改用户名
UPDATE mysql.user SET USER='wang5' WHERE USER='zhang3' AND host='localhost';

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0



-- 记得刷新权限
FLUSH PRIVILEGES;

Query OK, 0 rows affected (0.01 sec)



-- 切换数据库
USE mysql;

Database changed









-- 查看用户信息
SELECT host,user from user;
```

```
+-----------+------------------+
| host      | user             |
+-----------+------------------+
| %         | li4              |
| %         | root             |
| %         | zhang3           |
| localhost | mysql.infoschema |
| localhost | mysql.session    |
| localhost | mysql.sys        |
| localhost | wang5            |
+-----------+------------------+
7 rows in set (0.00 sec)
```

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：

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

Query OK, 0 rows affected (0.01 sec)



-- 删除host为localhost 名字为wang5的用户
DROP USER 'wang5'@'localhost';

Query OK, 0 rows affected (0.00 sec)




-- 查看用户信息
SELECT host,user from user;

+-----------+------------------+
| host      | user             |
+-----------+------------------+
| %         | root             |
```

```
| %         | zhang3           |
| localhost | mysql.infoschema |
| localhost | mysql.session    |
| localhost | mysql.sys        |
+-----------+------------------+
5 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：

DML语句执行后，记得执行；DDL不用

> FLUSH PRIVILEGES;

注意：

不推荐通过 `DELETE FROM mysql.user WHERE Host='localhost' AND User='Emily';` 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

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';

举例：

```
-- 登录
```

```
mysql -uzhang3 -h192.168.175.135 -p123123



-- 修改当前用户密码
ALTER USER USER() IDENTIFIED BY 'root123';

Query OK, 0 rows affected (0.01 sec)




-- 重新登录
mysql -uzhang3 -h192.168.175.135 -proot123
```

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

在mysql中 用户密码是加密存储的

```
USE mysql;



DESC user;

+-------------------------+-----------------------------------+------+-----+-
---------------------+-------+
| Field                   | Type                              | Null | Key |
Default               | Extra |
+-------------------------+-----------------------------------+------+-----+-
---------------------+-------+
| Host                    | char(255)                         | 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                       |         |
| Index_priv                | enum('N','Y')                        | NO   |     |
N                       |         |
| Alter_priv                | enum('N','Y')                        | NO   |     |
N                       |         |
| Show_db_priv              | enum('N','Y')                        | NO   |     |
N                       |         |
| Super_priv                | enum('N','Y')                        | NO   |     |
N                       |         |
| Create_tmp_table_priv     | enum('N','Y')                        | NO   |     |
N                       |         |
| Lock_tables_priv          | enum('N','Y')                        | NO   |     |
N                       |         |
| Execute_priv              | enum('N','Y')                        | NO   |     |
N                       |         |
| Repl_slave_priv           | enum('N','Y')                        | NO   |     |
N                       |         |
| Repl_client_priv          | enum('N','Y')                        | NO   |     |
N                       |         |
| Create_view_priv          | enum('N','Y')                        | NO   |     |
N                       |         |
| Show_view_priv            | enum('N','Y')                        | NO   |     |
N                       |         |
| Create_routine_priv       | enum('N','Y')                        | NO   |     |
N                       |         |
| Alter_routine_priv        | enum('N','Y')                        | NO   |     |
N                       |         |
| Create_user_priv          | enum('N','Y')                        | NO   |     |
N                       |         |
| Event_priv                | enum('N','Y')                        | NO   |     |
N                       |         |
| Trigger_priv              | enum('N','Y')                        | NO   |     |
N                       |         |
| Create_tablespace_priv    | enum('N','Y')                        | NO   |     |
N                       |         |
| ssl_type                  | enum('','ANY','X509','SPECIFIED')    | NO   |     |
                        |         |
| ssl_cipher                | blob                                 | NO   |     |
NULL                    |         |
```

```
| x509_issuer               | blob                                | NO   |     |
NULL                    |       |
| x509_subject              | blob                                | NO   |     |
NULL                    |       |
| max_questions             | int unsigned                        | NO   |     |
0                       |       |
| max_updates               | int unsigned                        | NO   |     |
0                       |       |
| max_connections           | int unsigned                        | NO   |     |
0                       |       |
| max_user_connections      | int unsigned                        | NO   |     |
0                       |       |
| plugin                    | char(64)                            | NO   |     |
caching_sha2_password |       |  -- 密码
| authentication_string     | text                                | YES  |     |
NULL                    |       |
| password_expired          | enum('N','Y')                       | NO   |     |
N                       |       |
| password_last_changed     | timestamp                           | YES  |     |
NULL                    |       |
| password_lifetime         | smallint unsigned                   | YES  |     |
NULL                    |       |
| account_locked            | enum('N','Y')                       | NO   |     |
N                       |       |
| Create_role_priv          | enum('N','Y')                       | NO   |     |
N                       |       |
| Drop_role_priv            | enum('N','Y')                       | NO   |     |
N                       |       |
| Password_reuse_history    | smallint unsigned                   | YES  |     |
NULL                    |       |
| Password_reuse_time       | smallint unsigned                   | YES  |     |
NULL                    |       |
| Password_require_current | enum('N','Y')                       | YES  |     |
NULL                    |       |
| User_attributes           | json                                | YES  |     |
NULL                    |       |
+--------------------------+-----------------------------------+------+-----+-
----------------------+-------+
51 rows in set (0.01 sec)



-- 查看用户信息
SELECT host,user,authentication_string FROM user;

+-----------+------------------+-------------------------------------------
---------------------------+
| host      | user             | authentication_string
                          |
+-----------+------------------+-------------------------------------------
---------------------------+
```

```
| %         | root              | *FAAFFE644E901CFAFAEC7562415E5FAEC243B8B2
                          |
| %         | zhang3            | $A$005$/I•'~••<eSWY\•/OG&?
oJKkkbyg5nVJcPddyTL9d2XqJQNqlKmcUiuiJl5OmZR6 |
| localhost | mysql.infoschema  |
$A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED |
| localhost | mysql.session     |
$A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED |
| localhost | mysql.sys         |
$A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED |
+-----------+-------------------+-----------------------------------------------
--------------------------+
5 rows in set (0.00 sec)
```

举例：

```
-- 修改密码
set password='root';
Query OK, 0 rows affected (0.00 sec)




-- 重新登录
mysql -uzhang3 -h192.168.175.135 -proot
```

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']]...;

举例：

```
-- root用户

ALTER user 'zhang3'@'%' identified by 'root123';
Query OK, 0 rows affected (0.01 sec)



-- 登录
mysql -uzhang3 -h192.168.175.135 -proot123
```

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';

```
-- root用户
set password for 'zhang3'@'%'='root';
Query OK, 0 rows affected (0.01 sec)


-- 登录
mysql -uzhang3 -h192.168.175.135 -proot
```

1.7 MySQL8密码管理(了解)

MySQL中记录使用过的历史密码,目前包含如下密码管理功能:

(1)密码过期:要求定期修改密码

(2)密码重用限制:不允许使用旧密码

(3)密码强度评估:要求使用高强度的密码。(第1章已讲)

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 'zhang3'@'%' PASSWORD EXPIRE;
```

该语句将用户zhang3的密码设置为过期

zhang3用户仍然可以登录进入数据库,但无法进行查

密码过期后,只有重新设置了新密码,才能正常使用

手动设置指定时间过期方式1:全局

如果密码使用的时间大于允许的时间,服务器会自动设置为过期,不需要手动设置

MySQL使用 default_password_lifetime系统变量建立全局密码过期策略

它的默认值是0,表示禁用自动密码过期

它允许的值是正整数N,表示允许的密码生存期。密码必须每隔N天进行修改。

两种实现方式分别如图所示:

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;
```

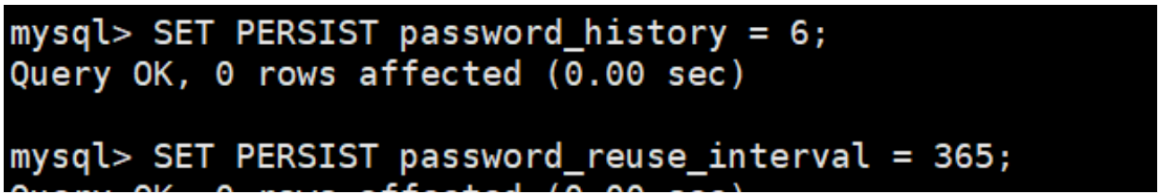
2、密码重用策略

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
```



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

关于MysQL的权限简单的理解就是MySQL允许你做你权力以内的事情,不可以越界

比如只允许你执行SELECT操作,那么你就不能执行UPDATE操作

只允许你从某台机器上连接MySQL,那么你就不能从除那台机器以外的其他机器连接MySQL

2.1 权限列表

MySQL到底都有哪些权限呢?

```
-- 查看当前用户权限
show privileges;
```

```
+--------------------------+---------------------------------------+--------
-----------------------------------------------+
| Privilege                | Context                               | Comment
                                               |
+--------------------------+---------------------------------------+--------
-----------------------------------------------+
| Alter                    | Tables                                | To
alter the table                                |
| Alter routine            | Functions,Procedures                  | To
alter or drop stored functions/procedures      |
| Create                   | Databases,Tables,Indexes              | To
create new databases and tables                |
| Create routine           | Databases                             | To use
CREATE FUNCTION/PROCEDURE                      |
| Create role              | Server Admin                          | To
create new roles                               |
| Create temporary tables  | Databases                             | To use
CREATE TEMPORARY TABLE                         |
| Create view              | Tables                                | To
create new views                               |
| Create user              | Server Admin                          | To
create new users                               |
| Delete                   | Tables                                | To
delete existing rows                           |
| Drop                     | Databases,Tables                      | To drop
databases, tables, and views                   |
| Drop role                | Server Admin                          | To drop
roles                                          |
| Event                    | Server Admin                          | To
create, alter, drop and execute events         |
| Execute                  | Functions,Procedures                  | To
execute stored routines                        |
| File                     | File access on server                 | To read
and write files on the server                  |
| Grant option             | Databases,Tables,Functions,Procedures | To give
to other users those privileges you possess    |
| Index                    | Tables                                | To
create or drop indexes                         |
| Insert                   | Tables                                | To
insert data into tables                        |
| Lock tables              | Databases                             | To use
LOCK TABLES (together with SELECT privilege)   |
| Process                  | Server Admin                          | To view
the plain text of currently executing queries  |
| Proxy                    | Server Admin                          | To make
proxy user possible                            |
| References               | Databases,Tables                      | To have
references on tables                           |
| Reload                   | Server Admin                          | To
reload or refresh tables, logs and privileges  |
| Replication client       | Server Admin                          | To ask
where the slave or master servers are          |
| Replication slave        | Server Admin                          | To read
binary log events from the master              |
```

```
| Select                    | Tables                                  | To
retrieve rows from table                                |
| Show databases            | Server Admin                            | To see
all databases with SHOW DATABASES                       |
| Show view                 | Tables                                  | To see
views with SHOW CREATE VIEW                             |
| Shutdown                  | Server Admin                            | To shut
down the server                                         |
| Super                     | Server Admin                            | To use
KILL thread, SET GLOBAL, CHANGE MASTER, etc.            |
| Trigger                   | Tables                                  | To use
triggers                                                |
| Create tablespace         | Server Admin                            | To
create/alter/drop tablespaces                           |
| Update                    | Tables                                  | To
update existing rows                                    |
| Usage                     | Server Admin                            | No
privileges - allow connect only                         |
| XA_RECOVER_ADMIN          | Server Admin                            |
                                                        |
| SHOW_ROUTINE              | Server Admin                            |
                                                        |
| SET_USER_ID               | Server Admin                            |
                                                        |
| RESOURCE_GROUP_USER       | Server Admin                            |
                                                        |
| APPLICATION_PASSWORD_ADMIN | Server Admin                           |
                                                        |
| SYSTEM_VARIABLES_ADMIN    | Server Admin                            |
                                                        |
| AUDIT_ADMIN               | Server Admin                            |
                                                        |
| SERVICE_CONNECTION_ADMIN  | Server Admin                            |
                                                        |
| CLONE_ADMIN               | Server Admin                            |
                                                        |
| PERSIST_RO_VARIABLES_ADMIN | Server Admin                           |
                                                        |
| FLUSH_USER_RESOURCES      | Server Admin                            |
                                                        |
| BINLOG_ADMIN              | Server Admin                            |
                                                        |
| ROLE_ADMIN                | Server Admin                            |
                                                        |
| SESSION_VARIABLES_ADMIN   | Server Admin                            |
                                                        |
| BINLOG_ENCRYPTION_ADMIN   | Server Admin                            |
                                                        |
| FLUSH_STATUS              | Server Admin                            |
                                                        |
| SYSTEM_USER               | Server Admin                            |
                                                        |
| ENCRYPTION_KEY_ADMIN      | Server Admin                            |
                                                        |
```

```
| REPLICATION_SLAVE_ADMIN     | Server Admin                          |
                                                                       |
| GROUP_REPLICATION_ADMIN     | Server Admin                          |
                                                                       |
| BACKUP_ADMIN                | Server Admin                          |
                                                                       |
| RESOURCE_GROUP_ADMIN        | Server Admin                          |
                                                                       |
| FLUSH_OPTIMIZER_COSTS       | Server Admin                          |
                                                                       |
| TABLE_ENCRYPTION_ADMIN      | Server Admin                          |
                                                                       |
| FLUSH_TABLES                | Server Admin                          |
                                                                       |
| CONNECTION_ADMIN            | Server Admin                          |
                                                                       |
| INNODB_REDO_LOG_ENABLE      | Server Admin                          |
                                                                       |
| INNODB_REDO_LOG_ARCHIVE     | Server Admin                          |
                                                                       |
| REPLICATION_APPLIER         | Server Admin                          |
                                                                       |
+-----------------------------+---------------------------------------+--------
-------------------------------------------+
62 rows in set (0.00 sec)
```

权限说明：

(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)

权限分布：

权限分布	可能的设置的权限
表权限	'Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'References', 'Index', 'Alter'
列权限	'Select', 'Insert', 'Update', 'References'
过程权限	'Execute', 'Alter Routine', 'Grant'

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'];

比如：

给zhang3用户授予dbtest1这个库下的所有表的改、查的权限

```
-- 在root用户中，给zhang3用户赋权
grant select,update on dbtest1.* to 'zhang3'@'%';

Query OK, 0 rows affected (0.00 sec)




-- 在zhang3用户中，查看数据库们
SHOW DATABASES;
```

```
+--------------------+
| Database           |
+--------------------+
| dbtest1            |
| information_schema |
+--------------------+
2 rows in set (0.00 sec)
```

```
-- 在zhang3用户中，查看权限
SHOW GRANTS;

+-----------------------------------------------------+
| Grants for zhang3@%                                 |
+-----------------------------------------------------+
| GRANT USAGE ON *.* TO `zhang3`@`%`                  | -- 只有登陆权限
| GRANT SELECT, UPDATE ON `dbtest1`.* TO `zhang3`@`%` | -- 对dbtest1库的 查看、修改
+-----------------------------------------------------+
2 rows in set (0.00 sec)
```

```
-- 切换数据库
USE dbtest1;
```

```
-- 查看该库下的所有表
SHOW TABLES;
+------------------+
| Tables_in_dbtest1 |
+------------------+
| emp1             |
| emp2             |
| student_myisam   |
+------------------+
3 rows in set (0.00 sec)
```

```
-- 查看数据
SELECT * FROM emp1;
```

```
+------+-------+
| id   | lname |
+------+-------+
|    1 | Tom   |
|    2 | 罗翔  |
+------+-------+
2 rows in set (0.00 sec)



-- 修改
UPDATE emp1 SET lname='法外狂徒'WHERE id=2;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0


-- 查看数据
SELECT * FROM emp1;

+------+----------+
| id   | lname    |
+------+----------+
|    1 | Tom      |
|    2 | 法外狂徒 |
+------+----------+
2 rows in set (0.00 sec)




-- 尝试删除  (没有权限)
DELETE FROM emp1 WHERE id=1;

ERROR 1142 (42000): DELETE command denied to user 'zhang3'@'192.168.175.1' for
table 'emp1'
```

```
-- 使用root用户赋权 delete
GRANT delete on dbtest1.* to 'zhang3'@'%';

Query OK, 0 rows affected (0.00 sec)



-- 使用张三用户尝试删除
DELETE FROM emp1 WHERE id=1;

ERROR 1142 (42000): DELETE command denied to user 'zhang3'@'192.168.175.1' for
table 'emp1'


-- 重新登录
```

```
-- 使用张三用户尝试删除
DELETE FROM emp1 WHERE id=1;

Query OK, 1 row affected (0.00 sec)


-- 查看数据
SELECT * FROM emp1;

+------+----------+
| id   | lname    |
+------+----------+
|    2 | 法外狂徒 |
+------+----------+
1 row in set (0.00 sec)
```

创建fgcy用户，给予所有库中所有表的权限；

```
-- 创建用户 (root)
CREATE USER 'fgcy'@'%' IDENTIFIED BY 'root123';

Query OK, 0 rows affected (0.01 sec)



-- 赋权
GRANT ALL PRIVILEGES ON *.* TO 'fgcy'@'%';

Query OK, 0 rows affected (0.00 sec)



-- 登录 fgcy用户
mysql -ufgcy -h192.168.175.135 -p



-- 查看所有数据库
SHOW DATABASES;

+--------------------+
| Database           |
+--------------------+
| dbtest1            |
| information_schema |
```

```
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)


USE dbtest1;

Database changed



-- 查看emp1表数据
SELECT * FROM emp1;

+------+----------+
| id   | lname    |
+------+----------+
|    2 | 法外狂徒 |
+------+----------+
1 row in set (0.00 sec)









-- 添加记录
INSERT INTO emp1 VALUES(3,'风格差异');

Query OK, 1 row affected (0.00 sec)


-- 修改记录
UPDATE  emp1 SET lname='fgcy' where id=3;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0




-- 查看数据
SELECT * FROM emp1;

+------+----------+
| id   | lname    |
+------+----------+
|    2 | 法外狂徒 |
|    3 | fgcy     |
+------+----------+
2 rows in set (0.00 sec)
```

```
-- 删除数据
DELETE FROM emp1 WHERE id =3;

Query OK, 1 row affected (0.00 sec)



-- 查看数据
SELECT * FROM emp1;

+------+----------+
| id   | lname    |
+------+----------+
|    2 | 法外狂徒 |
+------+----------+
1 row in set (0.00 sec)
```

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

参数解释:

`ALL PRIVILEGES`是表示所有权限,你也可以使用SELECT,UPDATE等权限。

`ON`用来指定权限针对哪些库和表中前面的*号用来指定数据库名,后面的*号用来指定表名

`*` 表示所有的

`TO` 表示将权限赋予某个用户

`'li4' @ 'localhost'` 表示li4用户,@后面接限制的主机,可以是IP、IP段、域名以及%,%表示任何地方

注意:

这里%有的版本不包括本地,以前碰到过给某个用户设置了%允许任何地方登录,但是在本地登录不了,这个和版本有关系

遇到这个问题再加一个localhost的用户就可以了

`IDENTIFIED BY` 指定用户的登录密码

注意这种方法赋权：如果发现没有该用户，则会直接新建一个用户

注意:

这种方式赋权（fgcy用户） 唯独**不包 括**grant的权限（赋予他人权限的权力）

- 如果需要赋予包括GRANIT的权限,添加参数"WITH GRANT OPTION"这个选项即可,表示该用户可以将自己拥有的权限授权给别人

  经常有人在创建操作用户的时候不指定WITH GRANT OPTION选项导致后来该用户不能使用GRANT命令创建用户或者给其它用户授权

```
-- 使用root用户赋权
GRANT ALL PRIVILEGES ON *.* TO 'fgcy'@'%' with grant option;
ERROR 1227 (42000): Access denied; you need (at least one of) the SYSTEM_USER
privilege(s) for this operation


-- 错误原因:
-- MySQL8版本中新增了一个system_user帐户类型，由于root用户没有SYSTEM_USER权限，导致错误出现。




-- 为root添加权限:
grant system_user on *.* to 'root';
Query OK, 0 rows affected (0.00 sec)


-- 使用root用户赋权
GRANT ALL PRIVILEGES ON *.* TO 'fgcy'@'%' with grant option;

Query OK, 0 rows affected (0.00 sec)
```

- 可以使用GRANT重复给用户添加权限,权限叠加

  比如你先给用户添加一个SELECT权限,然后又给用户添加一个INSERT权限,那么该用户就同时拥有了SELECT和INSERT权限

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();

```
SHOW GRANTS\G
*************************** 1. row ***************************
Grants for fgcy@%: GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD,
SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER, SHOW DATABASES, SUPER,
CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION
CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER,
EVENT, TRIGGER, CREATE TABLESPACE, CREATE ROLE, DROP ROLE ON *.* TO `fgcy`@`%`
WITH GRANT OPTION
*************************** 2. row ***************************
Grants for fgcy@%: GRANT
APPLICATION_PASSWORD_ADMIN,AUDIT_ADMIN,BACKUP_ADMIN,BINLOG_ADMIN,BINLOG_ENCRYP
TION_ADMIN,CLONE_ADMIN,CONNECTION_ADMIN,ENCRYPTION_KEY_ADMIN,FLUSH_OPTIMIZER_C
OSTS,FLUSH_STATUS,FLUSH_TABLES,FLUSH_USER_RESOURCES,GROUP_REPLICATION_ADMIN,IN
NODB_REDO_LOG_ARCHIVE,INNODB_REDO_LOG_ENABLE,PERSIST_RO_VARIABLES_ADMIN,REPLIC
ATION_APPLIER,REPLICATION_SLAVE_ADMIN,RESOURCE_GROUP_ADMIN,RESOURCE_GROUP_USER
,ROLE_ADMIN,SERVICE_CONNECTION_ADMIN,SESSION_VARIABLES_ADMIN,SET_USER_ID,SHOW_
ROUTINE,SYSTEM_USER,SYSTEM_VARIABLES_ADMIN,TABLE_ENCRYPTION_ADMIN,XA_RECOVER_A
DMIN ON *.* TO `fgcy`@`%` WITH GRANT OPTION
2 rows in set (0.00 sec)
```

查看某用户的全局权限

> SHOW GRANTS FOR 'user'@'主机地址' ;

```
-- 当前fgcy用户

SHOW GRANTS FOR 'root'@'%'\G
*************************** 1. row ***************************
Grants for root@%: GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD,
SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER, SHOW DATABASES, SUPER,
CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION
CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER,
EVENT, TRIGGER, CREATE TABLESPACE, CREATE ROLE, DROP ROLE ON *.* TO `root`@`%`
WITH GRANT OPTION
*************************** 2. row ***************************
Grants for root@%: GRANT SYSTEM_USER ON *.* TO `root`@`%`
2 rows in set (0.00 sec)
```

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：

```
-- 当前zhang3用户
 show grants;

+-------------------------------------------------------------+
| Grants for zhang3@%                                         |
+-------------------------------------------------------------+
| GRANT USAGE ON *.* TO `zhang3`@`%`                          |
| GRANT SELECT, UPDATE, DELETE ON `dbtest1`.* TO `zhang3`@`%` |
+-------------------------------------------------------------+
2 rows in set (0.00 sec)



-- 在root用户下 回收zhang3的查看权限
REVOKE SELECT ON dbtest1.* FROM 'zhang3'@'%';

Query OK, 0 rows affected (0.00 sec)




-- zhang3重新登录后
SELECT * FROM emp1;
ERROR 1142 (42000): SELECT command denied to user 'zhang3'@'192.168.175.1' for
table 'emp1'
```

注意：

须用户重新登录后才能生效

```
-- 通过 root用户 收回全库全表的所有权限
REVOKE ALL PRIVILEGES ON *.* FROM 'zhang3'@'%';
```

```
Query OK, 0 rows affected (0.00 sec)



-- zhang3用户下
show databases;

+--------------------+
| Database           |
+--------------------+
| information_schema |
+--------------------+
1 row in set (0.00 sec)


show grants;

+----------------------------------+
| Grants for zhang3@%              |
+----------------------------------+
| GRANT USAGE ON *.* TO `zhang3`@`%` |
+----------------------------------+
1 row in set (0.00 sec)



-- 收回zahng3用户 在mysql库下的所有表的插删改查权限
REVOKE SELECT,INSERT,UPDATE,DELETE ON mysql.* FROM 'zhang3'@'%';
```

总结:

有一些程序员喜欢使用Root超级用户来访问数据库,完全把权限控制放在应用层面实现。这样当然也是可以的

但建议大家,尽量使用数据库自己的角色和用户机制来控制访问权限,不要轻易用Root账号

因为Root账号密码放在代码里面不安全,一旦泄露,数据库就会完全失去保护

而且,MySQL的权限控制功能十分完善,应该尽量利用,可以提高效率,而且安全可靠。

3. 权限表

MySQL服务器**通过权限表来控制用户对数据库的访问,权限表存放在mysql数据库中**

MySQL数据库系统会**根据这些权限表的内容**`为每个用户赋予相应的权限`。这些权限表中最重要的是**user表**、db表。除此之外,还有**table_priv表**、column_priv表 和p**roc_priv表**等

在MySQL启动时,服务器将这些数据库表中权限信息的内容读入内存

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:

```
desc user;

+------------------------+-----------------------------------+------+-----+-
----------------------+-------+
| Field                  | Type                              | Null | Key |
Default               | Extra |
+------------------------+-----------------------------------+------+-----+-
----------------------+-------+
| Host                   | char(255)                         | NO   | PRI |
                      |       |
| User                   | char(32)                          | NO   | PRI |
                      |       |
| Select_priv            | enum('N','Y')                     | NO   |     |
N                     -- 这里使用的是枚举类型 Y、N
| Insert_priv            | enum('N','Y')                     | NO   |     |
N                     -- 这里的权限一旦设置为Y 则
| Update_priv            | enum('N','Y')                     | NO   |     |
N                     -- 对所有的表都有该权限
| 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                      |       |
| Index_priv             | enum('N','Y')                     | NO   |     |
N                      |       |
| Alter_priv             | enum('N','Y')                     | NO   |     |
N                      |       |
| Show_db_priv           | enum('N','Y')                     | NO   |     |
N                      |       |
| Super_priv             | enum('N','Y')                     | NO   |     |
N                      |       |
| Create_tmp_table_priv  | enum('N','Y')                     | NO   |     |
N                      |       |
```

```
| Lock_tables_priv          | enum('N','Y')                       | NO   |     |
N                       |       |
| Execute_priv              | enum('N','Y')                       | NO   |     |
N                       |       |
| Repl_slave_priv           | enum('N','Y')                       | NO   |     |
N                       |       |
| Repl_client_priv          | enum('N','Y')                       | NO   |     |
N                       |       |
| Create_view_priv          | enum('N','Y')                       | NO   |     |
N                       |       |
| Show_view_priv            | enum('N','Y')                       | NO   |     |
N                       |       |
| Create_routine_priv       | enum('N','Y')                       | NO   |     |
N                       |       |
| Alter_routine_priv        | enum('N','Y')                       | NO   |     |
N                       |       |
| Create_user_priv          | enum('N','Y')                       | NO   |     |
N                       |       |
| Event_priv                | enum('N','Y')                       | NO   |     |
N                       |       |
| Trigger_priv              | enum('N','Y')                       | NO   |     |
N                       |       |
| Create_tablespace_priv    | enum('N','Y')                       | NO   |     |
N                       |       |
| ssl_type                  | enum('','ANY','X509','SPECIFIED') | NO   |     |
                        |       |
| ssl_cipher                | blob                                | NO   |     |
NULL                    |       |
| x509_issuer               | blob                                | NO   |     |
NULL                    |       |
| x509_subject              | blob                                | NO   |     |
NULL                    |       |
| max_questions             | int unsigned                        | NO   |     |
0                       |       |
| max_updates               | int unsigned                        | NO   |     |
0                       |       |
| max_connections           | int unsigned                        | NO   |     |
0                       |       |
| max_user_connections      | int unsigned                        | NO   |     |
0                       |       |
| plugin                    | char(64)                            | NO   |     |
caching_sha2_password   |       |
| authentication_string     | text                                | YES  |     |
NULL                    -- MySQL5.7之前该字段叫password
| password_expired          | enum('N','Y')                       | NO   |     |
N                       |       |
| password_last_changed     | timestamp                           | YES  |     |
NULL                    |       |
| password_lifetime         | smallint unsigned                   | YES  |     |
NULL                    |       |
| account_locked            | enum('N','Y')                       | NO   |     |
N                       |       |
| Create_role_priv          | enum('N','Y')                       | NO   |     |
N                       |       |
```

```
| Drop_role_priv            | enum('N','Y')                       | NO   |     |
N                      |       |
| Password_reuse_history    | smallint unsigned                   | YES  |     |
NULL                   |       |
| Password_reuse_time       | smallint unsigned                   | YES  |     |
NULL                   |       |
| Password_require_current | enum('N','Y')                       | YES  |     |
NULL                   |       |
| User_attributes           | json                                | YES  |     |
NULL                   |       |
+--------------------------+-------------------------------------+------+-----+-
----------------------+-------+
51 rows in set (0.14 sec)
```

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
     1、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
     2、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。
     3、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

```
-- 查看用户，以列的方式显示数据：
SELECT * FROM mysql.user \G;

*************************** 1. row ***************************
                  Host: %
                  User: fgcy
            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: NULL
            x509_issuer: NULL
           x509_subject: NULL
          max_questions: 0
            max_updates: 0
        max_connections: 0
```

```
    max_user_connections: 0
                  plugin: caching_sha2_password
   authentication_string:
$A$005$••Kg>R[JGP•5=•_F*•=l5N6gZ1rDWfBhknyV7GLjNcWvptCx5gKUp4b1b39gUg9
        password_expired: N
   password_last_changed: 2022-07-05 23:29:36
       password_lifetime: NULL
          account_locked: N
        Create_role_priv: Y
          Drop_role_priv: Y
  Password_reuse_history: NULL
     Password_reuse_time: NULL
Password_require_current: NULL
         User_attributes: NULL
*************************** 2. row ***************************
                    Host: %
                    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: NULL
             x509_issuer: NULL
            x509_subject: NULL
           max_questions: 0
             max_updates: 0
         max_connections: 0
    max_user_connections: 0
```

```
                  plugin: mysql_native_password
   authentication_string: *FAAFFE644E901CFAFAEC7562415E5FAEC243B8B2
        password_expired: N
   password_last_changed: 2022-07-04 16:54:08
       password_lifetime: NULL
          account_locked: N
        Create_role_priv: Y
          Drop_role_priv: Y
  Password_reuse_history: NULL
     Password_reuse_time: NULL
Password_require_current: NULL
         User_attributes: NULL
*************************** 3. row ***************************
                    Host: %
                    User: zhang3
             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: NULL
             x509_issuer: NULL
            x509_subject: NULL
           max_questions: 0
             max_updates: 0
         max_connections: 0
    max_user_connections: 0
                  plugin: caching_sha2_password
```

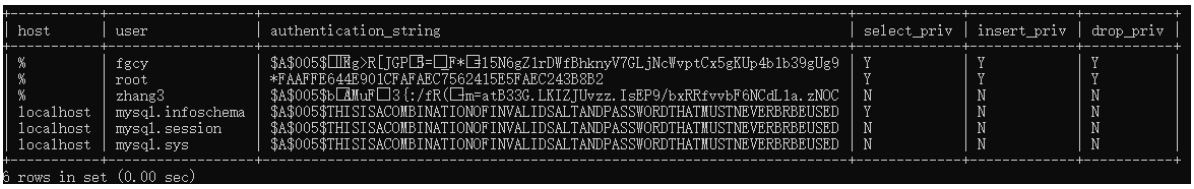
```
   authentication_string: $A$005$b•AMuF••3{:/fR(•-
m=atB33G.LKIZJUvzz.IsEP9/bxRRfvvbF6NCdL1a.zNOC
        password_expired: N
   password_last_changed: 2022-07-05 23:07:46
       password_lifetime: NULL
          account_locked: N
        Create_role_priv: N
          Drop_role_priv: N
  Password_reuse_history: NULL
     Password_reuse_time: NULL
Password_require_current: NULL
         User_attributes: NULL
*************************** 4. row ***************************
                   Host: localhost
                   User: mysql.infoschema
             Select_priv: Y
             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: NULL
             x509_issuer: NULL
            x509_subject: NULL
           max_questions: 0
             max_updates: 0
         max_connections: 0
    max_user_connections: 0
                  plugin: caching_sha2_password
```

```
   authentication_string:
$A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED
        password_expired: N
   password_last_changed: 2022-04-11 09:25:05
       password_lifetime: NULL
          account_locked: Y
        Create_role_priv: N
          Drop_role_priv: N
  Password_reuse_history: NULL
     Password_reuse_time: NULL
Password_require_current: NULL
         User_attributes: NULL
*************************** 5. row ***************************
                    Host: localhost
                    User: mysql.session
             Select_priv: N
             Insert_priv: N
             Update_priv: N
             Delete_priv: N
             Create_priv: N
               Drop_priv: N
             Reload_priv: N
           Shutdown_priv: Y
            Process_priv: N
               File_priv: N
              Grant_priv: N
         References_priv: N
              Index_priv: N
              Alter_priv: N
            Show_db_priv: N
              Super_priv: Y
   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: NULL
             x509_issuer: NULL
            x509_subject: NULL
           max_questions: 0
             max_updates: 0
         max_connections: 0
    max_user_connections: 0
                  plugin: caching_sha2_password
```

```
   authentication_string:
$A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED
        password_expired: N
   password_last_changed: 2022-04-11 09:25:05
       password_lifetime: NULL
          account_locked: Y
        Create_role_priv: N
          Drop_role_priv: N
  Password_reuse_history: NULL
     Password_reuse_time: NULL
Password_require_current: NULL
         User_attributes: NULL
*************************** 6. row ***************************
                   Host: localhost
                   User: mysql.sys
              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: NULL
             x509_issuer: NULL
            x509_subject: NULL
           max_questions: 0
             max_updates: 0
         max_connections: 0
    max_user_connections: 0
                  plugin: caching_sha2_password
```

```
   authentication_string:
$A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED
        password_expired: N
   password_last_changed: 2022-04-11 09:25:05
       password_lifetime: NULL
          account_locked: Y
        Create_role_priv: N
          Drop_role_priv: N
  Password_reuse_history: NULL
     Password_reuse_time: NULL
Password_require_current: NULL
         User_attributes: NULL
6 rows in set (0.01 sec)
```

```
-- 查询特定字段：
SELECT host,user,authentication_string,select_priv,insert_priv,drop_priv
FROM mysql.user;
```



host	user	authentication_string	select_priv	insert_priv	drop_priv
%	fgcy	A005$□□Rg>R[JGP□=□F*□15N6gZ1rDWfBhknyV7GLjNcWvptCx5gKUp4b1b39gUg9	Y	Y	Y
%	root	*FAAFFE644E901CFAFAEC7562415E5FAEC243B8B2	Y	Y	Y
%	zhang3	A005$b□AMuF□3{:/fR(□m=atB33G.LKIZ.IUvzz.IsEP9/bxRRfvvbF6NCdL1a.zNOC	N	N	N
localhost	mysql.infoschema	A005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED	Y	N	N
localhost	mysql.session	A005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED	N	N	N
localhost	mysql.sys	A005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED	N	N	N

6 rows in set (0.00 sec)

3.2 db表

使用DESCRIBE查看db表的基本结构：

```
DESCRIBE mysql.db;

+-----------------------+---------------+------+-----+---------+-------+
| Field                 | Type          | Null | Key | Default | Extra |
+-----------------------+---------------+------+-----+---------+-------+
| Host                  | char(255)     | NO   | PRI |         |       |
| Db                    | char(64)      | NO   | PRI |         |       |
| User                  | char(32)      | NO   | PRI |         |       | -- 以
上三者 构成联合主键
| Select_priv           | enum('N','Y') | NO   |     | N       |       | -- 该
数据库体现的是 某个用户是否有操作某个数据库的相关权限
| 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       |       |
| Grant_priv            | enum('N','Y') | NO   |     | N       |       |
| References_priv       | enum('N','Y') | NO   |     | N       |       |
```

```
| Index_priv             | enum('N','Y') | NO   |     | N       |         |
| Alter_priv             | enum('N','Y') | NO   |     | N       |         |
| Create_tmp_table_priv | enum('N','Y') | NO   |     | N       |         |
| Lock_tables_priv       | enum('N','Y') | NO   |     | N       |         |
| Create_view_priv       | enum('N','Y') | NO   |     | N       |         |
| Show_view_priv         | enum('N','Y') | NO   |     | N       |         |
| Create_routine_priv    | enum('N','Y') | NO   |     | N       |         |
| Alter_routine_priv     | enum('N','Y') | NO   |     | N       |         |
| Execute_priv           | enum('N','Y') | NO   |     | N       |         |
| Event_priv             | enum('N','Y') | NO   |     | N       |         |
| Trigger_priv           | enum('N','Y') | NO   |     | N       |         |
+------------------------+---------------+------+-----+---------+-------+
22 rows in set (0.05 sec)
```

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

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表的结构分别如下：

```
desc mysql.tables_priv;
```

Field	Type	Null	Key	Default	Extra
Host	char(255)	NO	PRI		
Db	char(64)	NO	PRI		
User	char(32)	NO	PRI		
Table_name	char(64)	NO	PRI		
Grantor	varchar(288)	NO	MUL		
Timestamp	timestamp	NO		CURRENT_TIMESTAMP	DEFAULT_GENERATED on update CURRENT_TIMESTAMP
Table_priv	set('Select','Insert','Update','Delete','Create','Drop','Grant','References','Index','Alter','Create View','Show view','Trigger')	NO			
Column_priv	set('Select','Insert','Update','References')	NO			

8 rows in set (0.01 sec)

描述某个用户针对某个数据库下的某张表的相关权限

tables_priv表有8个字段，分别是**Host、Db、User、Table_name**、Grantor、Timestamp、Table_priv和Column_priv

其中，前四个字段构成联合主键；

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

columns_priv

```
desc mysql.columns_priv;
```

```
+-------------+---------------------------------------+------+-----+-------------------+-----------------------------+
| Field       | Type                                  | Null | Key | Default           | Extra                       |
+-------------+---------------------------------------+------+-----+-------------------+-----------------------------+
| Host        | char(255)                             | NO   | PRI |                   |                             |
| Db          | char(64)                              | NO   | PRI |                   |                             |
| User        | char(32)                              | NO   | PRI |                   |                             |
| Table_name  | char(64)                              | NO   | PRI |                   |                             |
| Column_name | char(64)                              | NO   | PRI |                   |                             |
| Timestamp   | timestamp                             | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED on update CURRENT_TIMESTAMP |
| Column_priv | set('Select','Insert','Update','References') | NO   |     |                   |                             |
+-------------+---------------------------------------+------+-----+-------------------+-----------------------------+
7 rows in set (0.00 sec)
```

描述 `某个用户` 针对 `某个数据库下` 的 `某张表` 下的 `某个字段` 的相关权限

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)

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

4.1 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

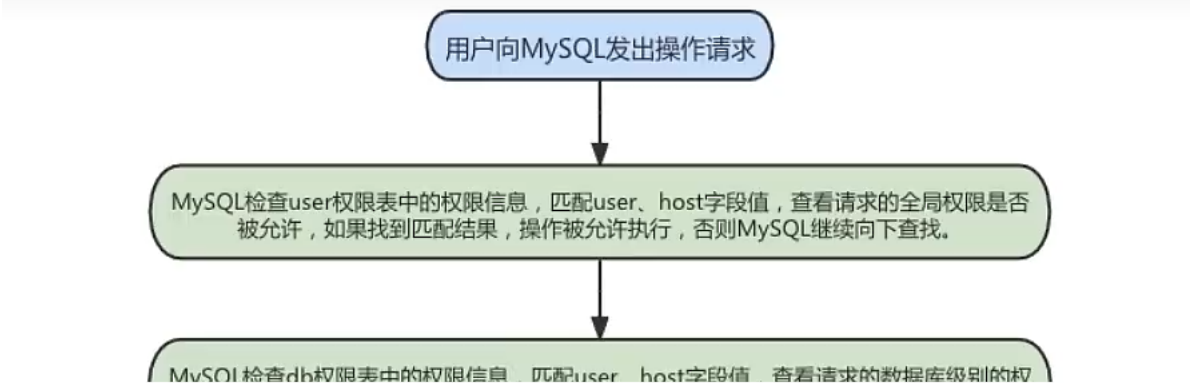
确认权限时

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

核实过程：



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

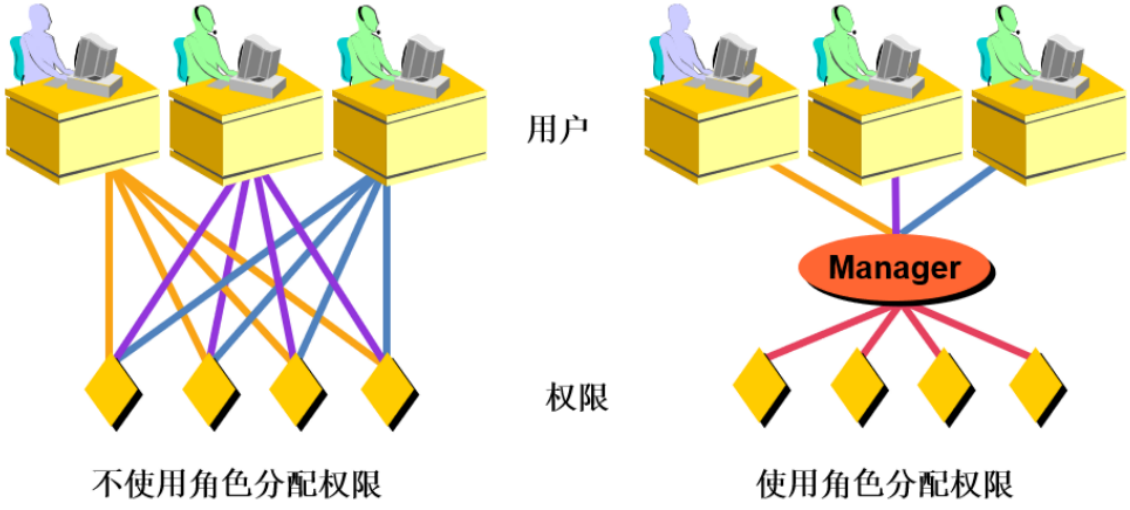
角色是在MySQL 8.0中引入的新功能。在MySQL中,**角色是权限的集合**,可以为角色添加或移除权限

用户可以被赋予角色,同时也被授予角色包含的权限

对角色进行操作需要较高的权限。并且像用户账户一样,角色可以拥有授予和撤消的权限。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

直接给用户赋权和给用户赋予角色的对比：



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

如果不写主机名,MySQL默认是通配符“%”,意思是这个账号可以从任何一台主机上登录数据库

```
-- 创建角色一
CREATE role 'manager'@'%';

Query OK, 0 rows affected (0.08 sec)



-- 创建角色二
CREATE role 'boss'@'%';

Query OK, 0 rows affected (0.00 sec)
```

还可以通过如下的指令,一次性创建3个角色:

> CREATE ROLE 'app_developer', 'app_read', 'app_write'

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

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

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'];

```
use dbtest1;

Database changed


show tables;

+------------------+
| Tables_in_dbtest1 |
+------------------+
| emp1              |
| emp2              |
| student_myisam    |
+------------------+
3 rows in set (0.03 sec)



-- 给角色赋权 (manager)
GRANT SELECT,UPDATE ON dbtest1.* TO 'manager';

Query OK, 0 rows affected (0.00 sec)




-- 给角色赋权 (boss)
GRANT ALL PRIVILEGES ON *.* TO 'boss'@'%';

Query OK, 0 rows affected (0.01 sec)
```

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：

```
-- 当前root用户下
SHOW GRANTS FOR 'manager';
+-------------------------------------------------------+
| Grants for manager@%                                  |
+-------------------------------------------------------+
| GRANT USAGE ON *.* TO `manager`@`%`                   |
| GRANT SELECT, UPDATE ON `dbtest1`.* TO `manager`@`%` |
+-------------------------------------------------------+
2 rows in set (0.00 sec)


-- 当前root用户下
SHOW GRANTS FOR 'boss'\G

*************************** 1. row ***************************
Grants for boss@%: GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD,
SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER, SHOW DATABASES, SUPER,
CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION
CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER,
EVENT, TRIGGER, CREATE TABLESPACE, CREATE ROLE, DROP ROLE ON *.* TO `boss`@`%`
*************************** 2. row ***************************
Grants for boss@%: GRANT
APPLICATION_PASSWORD_ADMIN,AUDIT_ADMIN,BACKUP_ADMIN,BINLOG_ADMIN,BINLOG_ENCRYP
TION_ADMIN,CLONE_ADMIN,CONNECTION_ADMIN,ENCRYPTION_KEY_ADMIN,FLUSH_OPTIMIZER_C
OSTS,FLUSH_STATUS,FLUSH_TABLES,FLUSH_USER_RESOURCES,GROUP_REPLICATION_ADMIN,IN
NODB_REDO_LOG_ARCHIVE,INNODB_REDO_LOG_ENABLE,PERSIST_RO_VARIABLES_ADMIN,REPLIC
ATION_APPLIER,REPLICATION_SLAVE_ADMIN,RESOURCE_GROUP_ADMIN,RESOURCE_GROUP_USER
,ROLE_ADMIN,SERVICE_CONNECTION_ADMIN,SESSION_VARIABLES_ADMIN,SET_USER_ID,SHOW_
ROUTINE,SYSTEM_USER,SYSTEM_VARIABLES_ADMIN,TABLE_ENCRYPTION_ADMIN,XA_RECOVER_A
DMIN ON *.* TO `boss`@`%`
2 rows in set (0.00 sec)
```

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';

```
-- 查看manager角色权限
SHOW GRANTS FOR 'manager';
+-------------------------------------------------------+
```

```
| Grants for manager@%                                          |
+---------------------------------------------------------------+
| GRANT USAGE ON *.* TO `manager`@`%`                           |
| GRANT SELECT, UPDATE ON `dbtest1`.* TO `manager`@`%` |
+---------------------------------------------------------------+
2 rows in set (0.00 sec)


-- 回收角色 对dbtest1的所有表的更新权限
REVOKE UPDATE ON dbtest1.* FROM 'manager'@'%';

Query OK, 0 rows affected (0.00 sec)




-- 查看manager角色权限
SHOW GRANTS FOR 'manager';

+--------------------------------------------+
| Grants for manager@%                       |
+--------------------------------------------+
| GRANT USAGE ON *.* TO `manager`@`%`        |
| GRANT SELECT ON `dbtest1`.* TO `manager`@`%` |
+--------------------------------------------+
2 rows in set (0.00 sec)
```

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

```
-- 创建用户 admin
CREATE ROLE 'admin';

Query OK, 0 rows affected (0.00 sec)



-- 查看用户权限 admin
SHOW GRANTS FOR 'admin';
```

```
+----------------------------------+
| Grants for admin@%               |
+----------------------------------+
| GRANT USAGE ON *.* TO `admin`@`%` |
+----------------------------------+
1 row in set (0.00 sec)
```

```
-- 删除角色
DROP ROLE 'admin';

Query OK, 0 rows affected (0.00 sec)
```

```
-- 查看用户权限 admin
SHOW GRANTS FOR 'admin';

ERROR 1141 (42000): There is no such grant defined for user 'admin' on host
'%'
```

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。

举例：

```
-- 切换数据库
USE mysql;

Database changed
```

```
-- 查看用户表 （发现角色也在用户表中）
SELECT user,host FROM user;

+------------------+-----------+
| user             | host      |
+------------------+-----------+
```

```
| boss             | %         |
| fgcy             | %         |
| manager          | %         |
| root             | %         |
| zhang3           | %         |
| mysql.infoschema | localhost |
| mysql.session    | localhost |
| mysql.sys        | localhost |
+------------------+-----------+
8 rows in set (0.00 sec)
```

```
-- 创建一个用户  (没有密码 直接回车进入)
CREATE USER 'wang5'@'%';

Query OK, 0 rows affected (0.00 sec)
```

```
-- 给角色赋权
GRANT 'manager'@'%' TO 'wang5'@'%';

Query OK, 0 rows affected (0.00 sec)
```

```
-- 查看wang5用户的权限
SHOW GRANTS FOR 'wang5';

+-----------------------------------+
| Grants for wang5@%                |
+-----------------------------------+
| GRANT USAGE ON *.* TO `wang5`@`%`  |
| GRANT `manager`@`%` TO `wang5`@`%` |
+-----------------------------------+
2 rows in set (0.00 sec)
```

```
-- 查看manager角色的权限
SHOW GRANTS FOR 'manager';

+-------------------------------------------+
| Grants for manager@%                      |
+-------------------------------------------+
| GRANT USAGE ON *.* TO `manager`@`%`       |
| GRANT SELECT ON `dbtest1`.* TO `manager`@`%` |
+-------------------------------------------+
2 rows in set (0.00 sec)
```

此时，使用赋予了角色的用户去登录、操作,你会发现,这个账号没有任何权限

这是因为,MySQL中创建了角色之后,默认都是没有被激活，也就是不能用,必须要手动激活,激活以后用户才能拥有角色对应的权限

使用wang5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();

+----------------+
| 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：

```
-- 激活wang5用户 （的角色）
SET DEFAULT ROLE ALL TO 'wang5'@'%';

Query OK, 0 rows affected (0.00 sec)



-- 重新登陆 （wang5用户）

-- 查看当前用户的角色状态
SELECT CURRENT_ROLE();

+----------------+
| CURRENT_ROLE() |
+----------------+
| `manager`@`%`  |
+----------------+
1 row in set (0.00 sec)



show databases;
+--------------------+
| Database           |
+--------------------+
```

```
| dbtest1            |
| information_schema |
+--------------------+
2 rows in set (0.01 sec)



-- 尝试查询
USE dbtest1;

Database changed


SELECT * FROM emp1;

+------+----------+
| id   | lname    |
+------+----------+
|    2 | 法外狂徒   |
+------+----------+
1 row in set (0.03 sec)



-- 尝试删除 (没有权限)
DELETE FROM emp1 WHERE id =2;

ERROR 1142 (42000): DELETE command denied to user 'wang5'@'192.168.175.1' for
table 'emp1'
```

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：

```
show variables like 'activate_all_roles_on_login';

+-----------------------------+-------+
| Variable_name               | Value |
+-----------------------------+-------+
| activate_all_roles_on_login | OFF   |
+-----------------------------+-------+
1 row in set (0.03 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

举例：

```
-- 全局配置 重启服务器后失效
SET GLOBAL activate_all_roles_on_login=ON;

Query OK, 0 rows affected (0.00 sec)



show variables like 'activate_all_roles_on_login';

+-----------------------------+-------+
| Variable_name               | Value |
+-----------------------------+-------+
| activate_all_roles_on_login | ON    |
+-----------------------------+-------+
1 row in set (0.01 sec)
```

查看当前会话已激活的角色

```
-- 当前用户wang5
SELECT CURRENT_ROLE();

+----------------+
| CURRENT_ROLE() |
+----------------+
| `manager`@`%`  |
+----------------+
1 row in set (0.00 sec)
```

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;

```
-- 当前root用户下
REVOKE 'manager'@'%' FROM 'wang5'@'%';

Query OK, 0 rows affected (0.02 sec)



-- 当前wang5用户下

-- 查看权限
 show grants;

+-----------------------------------+
| Grants for wang5@%                |
+-----------------------------------+
| GRANT USAGE ON *.* TO `wang5`@`%` |
+-----------------------------------+
1 row in set (0.00 sec)



-- 查看数据库
SHOW DATABASES;

+--------------------+
| Database           |
+--------------------+
| information_schema |
+--------------------+
1 row in set (0.01 sec)
```

5.10 设置强制角色(mandatory role)

了解一下

使用场景：

每一个新建的用户都有一个登录权限

强制角色是给每个创建账户的默认角色,不需要手动设置;

强制角色无法被REVOKE或者DROP

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
```

5.11小结

MySQL主要管理权限语句：

语句	作用

6.配置文件的使用

6.1配置文件格式

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`、`mysqladmin`

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
```

6.2启动命令与选项组

配置文件中不同的选项组是给不同的启动命令使用的

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`

下面是启动命令能读取的选项组都有哪些:

启动命令	类别	能读取的组

比如,在/etc/mysql/my.cnf 这个配置文件中

添加一些内容:

```
[server]

skip-networking

default-storage-engine=MyISAM
```

然后直接用mysqld启动服务器程序:

```
mysqld
```

注意:

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/my.cn

又由于mysqld命令可以读取[server]选项组的内容,所以skipnetworking 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,就不生效

6.3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

6.4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

6.5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d`efaults-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

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

7.系统变量(复习)

7.1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:`max_connections` :允许同时连入的客户端数量

`default_storage_engine`:表的默认存储引擎用系统变量

`query_cache_size`:查询缓存的大小

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

7.2查看系统变量查看

MySQL服务器程序支持的系统变量以及它们的当前值:

```
-- 查看所有全局变量
SHOW GLOBAL VARIABLES;

588 rows in set (0.05 sec)


#查看所有会话变量  -- 或SHOW VARIABLES;
SHOW SESSION VARIABLES;

610 rows in set (0.00 sec)
```

```
-- 查看满足条件的部分系统变量
SHOW GLOBAL VARIABLES LIKE '%标识符%


-- 查看满足条件的部分会话变量
SHOW SESSION VARIABLES LIKE '%标识符%';
```

由于系统变量实在太多了,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

```
SHOW VARIABLES LIKE 'default_storage_engine';

+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
1 row in set (0.00 sec)
```

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

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:

```
SHOW VARIABLES LIKE 'default%';

+-------------------------------+-----------------------+
| Variable_name                 | Value                 |
+-------------------------------+-----------------------+
| default_authentication_plugin | caching_sha2_password |
| default_collation_for_utf8mb4 | utf8mb4_0900_ai_ci    |
| default_password_lifetime     | 0                     |
| default_storage_engine        | InnoDB                |
| default_table_encryption      | OFF                   |
| default_tmp_storage_engine    | InnoDB                |
| default_week_format           | 0                     |
+-------------------------------+-----------------------+
7 rows in set (0.00 sec)
```

这样就查出了所有以default开头的系统变量的值

7.3设置系统变量

7.3.1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总结一下

主要是两种方式:

- 通过命令行添加启动选项（优先级高、临时有效）

  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 （优先级较低、长久有效）

  我们可以这样填写配置文件:

```
[server]

default-storage-engine=MyISAM

max-connections=19
```

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:

```
SHOW VARIABLES LIKE 'default_storage_engine';

+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | MyISAM |
+------------------------+--------+
1 row in set (0.00 sec)



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

7.3.2服务器程序运行过程中设置

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

但是,系统变量有作用范围之分

设置不同作用范围的系统变量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;

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`defaull_storage_engine`的值设置为MyISAM

这样两个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?

MySOL提出了系统变量作用范围的概念,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

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`并行初始化

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,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:

```
#方式一:
SET GLOBAL default_storage_engine = MyISAM;

#方式二:
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:

```
#方式一:
SET SESSION default_storage_engine = MyISAM;


#方式二:
SET @@SESSION. default_storage_engine = MyISAM;


# 方式三:
SET default_storage_engine = MyISAM;
```

从上边的方式三可以看出,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

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:

```
SHOW [GLOBAL | SESSION] VARIABLES [LIKE E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:

```
SHOW SESSION VARIABLES LIKE 'default_storage_engine';

+-----------------------+--------+
| Variable_name         | Value  |
```

```
+------------------------+--------+
| default_storage_engine | InnoDB |
+------------------------+--------+
1 row in set (0.00 sec)
```

```
 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)
```

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

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

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

每一个MySQL客户机成功连接MySQL服务器后，都会产生与之对应的会话

会话期间，MySQL服务实例会在MySQL服务器内存中生成与该会话对应的会话系统变量

这些会话系统变量的初始值是全局系统变量值的复制

如下图：

- 全局系统变量针对于所有会话（连接）有效，但`不能跨重启`
- 会话系统变量仅针对于当前会话（连接）有效。会话期间，当前会话对某个会话系统变量值的修改，不会影响其他会话同一个会话系统变量的值。
- 会话1对某个全局系统变量值的修改会导致会话2中同一个全局系统变量值的修改。

注意：

作为 MySQL 编码规范，**MySQL 中的系统变量**以`两个“@”`开头

其中`@@global`仅用于标记全局系统变量

`@@session`仅用于标记会话系统变量

@@首先标记会话系统变量，如果会话系统变量不存在，则标记全局系统变量

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。