数据库权限和用户管理

对于任何一种数据库而言，安全性在实际应用中非常重要。如果安全性得不到保证，那么数据库将面临各种各样的威胁，可能导致数据丢失，严重时会直接导致系统瘫痪。 `MySQL` 提供了完善的用户管理和权限控制机制以支持限定对不同数据库对象的访问和操作。

一、访问控制和权限表

考虑以下数据库需求：

- 多数用户只需要对表进行**读**和**写**，但少数用户需要能创建和删除表；
- 某些用户需要读表，但可能不需要更新表；
- 可能想允许用户**添加**数据，但不允许他们删除数据；
- 某些用户（管理员）可能需要处理用户账号的权限，但多数用户不需要；
- 可能想让用户通过**存储过程**访问数据，但不允许他们直接访问数据；
- 可能想根据用户**登录的地点** `ip` 限制对某些功能的访问。

`MySQL` 服务器通过 `MySQL` 权限来控制用户对数据库的访问，安装 `MySQL` 数据库成功后，会自动安装多个数据库。 `MySQL` **权限表**存放在名称为 `MySQL` 的数据库里。常用到的表有 `user、db、table_priv、columns_priv、column_priv` 和 `procs_priv` 。

`MySQL` 中的权限是按照 `user` 表、 `db` 表、 `tables_priv` 表和 `columns_priv` 表的顺序进行分配的。数据库系统先判断 `user` 表中对应某一权限的值是否为 `'Y'` ，如果 `user` 表中的值是 `'Y'` ，就不需要检查后面的表了；如果 `user` 表中的值为 `'N'` ，则依次检查 `db` 表、 `tables_priv` 表和 `columns_priv` 表。

表1 `mysql` 数据库中的权限相关表

数据表	描述
`user`	保存用户被授予的全局权限
`db`	保存用户被授予的数据库权限
`tables_priv`	保存用户被授予的表权限
`columns_priv`	保存用户被授予的列权限
`procs_priv`	保存用户被授予的存储过程权限
`proxies_priv`	保存用户被授予的代理权限

1. `user` 表

```
1   DESC mysql.user;
```

`user` 表是 `MySQL` 中最终的一个**权限表**。可以使用 `desc` 语句来查看 `user` 的基本结构。 `user` 列主要分为4个部分：**用户列、权限列、安全列和资源控制列**。其中，用户列和权限列使用最为频繁。权限可以进一步分为：

- 普通权限：数据库操作。
- 管理权限：数据库**管理**操作。

当用户进行连接时，权限表的存取过程有以下两个阶段：

- 先从 `user` 表中的 `host、user` 和 `authentication_string` 这3个字段中判断连接的 `ip` 、用户名称和密码是否存在于表中，如果存在，则通过身份验证，否则拒绝连接。
- 如果通过身份验证，按照以下权限的顺序得到数据库权限： `user、db、table_priv、colums_priv` 。

这几个表的权限依次递减，**全局权限覆盖局部权限**。

`user` 表主要包含四类字段：

- 用户字段：`user` 表中的 `host、user` 和 `authentication_string` 字段都属于用户字段。
- 权限字段：`user` 表中包含几十个与权限有关以 `_priv` 结尾的字段，这些权限字段决定了用户的权限，这些权限包括基本权限、修改和添加权限、关闭服务器权限、超级权限和加载权限等。
- 安全字段：安全列只有6个字段，2个是 `ssl` 相关的：`ssl_type` 和 `ssl_cipher`，2个是 `x509` 相关的：`x509_issuer` 和 `x509_subject`，另外2个是授权插件相关的。`ssl` 用于**加密**；`x509` 标准可用于**标识用户**；`plugin` 字段标识可以用于**验证用户身份**的插件，如果该字段为空，服务器使用内建授权验证机制验证用户身份。
- 资源控制字段：资源控制列的字段用来**限制用户使用的资源**，这些字段的默认值为 `0`，表示没有限制，包含4个字段：
  - `max_questions`：用户每小时允许执行的查询操作次数。
  - `max_updates`：用户每小时允许执行的更新操作次数。
  - `max_connections`：用户每小时允许执行的连接操作次数。
  - `max_user_connections`：单个用户可以同时具有的连接次数。

2. `db` 表

```
1   DESC mysql.db;
```

`db` 表是 `mysql` 数据库中非常重要的权限表。`db` 表中存储了用户对某个数据库的操作权限，决定用户能从哪个主机存取哪个数据库。该表不受 `grant` 和 `revoke` 语句的影响。`db` 表的字段大致可以分为用户列和权限列：

- 用户列
  - `db` 表的用户列有3个字段：`host、db` 和 `user`。这3个字段分别表示主机名、数据库名和用户名；`host` 表的用户列有两个字段：`host` 和 `db`。这两个字段分别表示主机名和数据库名。
- 权限列
  - `db` 表的权限列大致相同，表中 `create_routine_priv` 和 `alter_routine_priv` 这两个字段表明用户是否有创建和修改存储过程的权限。

`user` 表中的权限是针对所有数据库的。如果 `user` 表中的 `select_priv` 字段取值为 `'Y'`，那么该用户可以查询所有数据库中的表；如果为某个用户只设置了查询 `test` 数据库的权限，那么 `user` 表的 `select_priv` 字段的取值为 `'N'`。而这个 `select` 权限则记录在 `db` 表中。`db` 表中 `select_priv` 字段的取值将会是 `'Y'`。由此可知，用户先根据 `user` 表的内容获取权限，然后再根据 `db` 表的内容获取权限。

3. `tables_priv` 表

```
1   DESC mysql.tables_priv;
```

`tables_priv` 表可以对单个表进行权限设置 ,用来指定表级权限。这里指定的权限适用于一个表的所有列。用户可以用 `desc` 语句查看表结构。 `tables_priv` 表有8个字段： `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` 等4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` 字段表示修改该记录的时间。
- `table_priv` 字段表示对表进行操作的权限，这些权限包括 `select、insert、update、delete、create、drop、grant、references、index和alter` 。
- `column_priv` 字段表示对表中的列进行操作的权限，这些权限包括 `select、insert、update和references` 。

4. `columns_priv` 表

```
1    DESC columns_priv;
```

`columns_priv` 表可以对表中的某一列进行权限设置。 `columns_priv` 表只有7个字段，分别是 `host、db、user、table_name、column_name、timestamp` 和 `column_priv` 。其中， `column_name` 用来指定对哪些数据列具有操作权限。

5. `procs_priv` 表

```
1    DESC procs_priv;
```

`procs_priv` 表可以对存储过程和存储函数进行权限设置。可以使用 `desc` 语句来查看 `procs_priv` 表的基本结构。 `procs_priv` 表包含8个字段： `host、db、user、routine_name、routine_type、grantor、proc_priv` 和 `timestamp` 等。各个字段的说明如下：

- `host、db` 和 `user` 字段分别表示主机名、数据库名和用户名。
- `routine_name` 字段表示存储过程或存储函数的名称。
- `routine_type` 字段表示存储过程或存储函数的类型。该字段有两个值，分别是 `function` 和 `procedure` 。 `function` 表示是一个存储函数； `procedure` 表示是一个存储过程。
- `grantor` 字段存储插入或修改该记录的用户。
- `proc_priv` 字段表示拥有的权限，包括 `execute、alter routine、grant` 等3种。
- `timestamp` 字段存储记录更新的时间。

二、用户管理

`MySQL` 用户账号和信息存储在名称 `MySQL` 的数据库中。这个数据库里有一个名为 `user` 的数据表，包含了所有用户账号，并且它用一个名为 `user` 的列存储用户的登录名。一般不需要直接访问 `MySQL` 数据库和表，但有时需要直接访问。在需要获得所有用户账号列表时，可使用以下代码实现：

```
1    use mysql;
2    select `user` from user;
```

`MySQL` 大致将用户分为两类：

- 超级管理员用户：具有所有权限，如创建用户、删除用户、管理用户等
- 普通用户：只拥有被赋予的某些权限

作为一个新安装的 `MySQL` 数据库系统，当前只有一个名为**root**的用户。这个用户是在成功安装 `MySQL` 服务器后，由系统创建的，并且被赋予了操作和管理 `MySQL` 的所有权限。root用户有对整个 `MySQL` 服务器**完全控制**的权限。

1. 创建用户

a. 使用 `create user` 创建用户账号

> 推荐使用创建账户方法

```
create user <user1> identified by 'password1',
<user2> identified by 'password2',
...;
```

- 要使用 `create user` 语句，必须拥有 `mysql` 中 `mysql` 数据库的 `insert` 权限或全局 `create user` 权限。使用 `create user` 语句创建一个用户账号后，会在系统自身的 `mysql` 数据库的 `user` 表添加一条新记录。如果创建的账户已经存在，则语句执行会出现错误。
- `create user` 可以一次性创建多个用户，多个用户之间使用逗号分隔。使用 `create user` 语句，如果没有为用户指定口令，那么 `mysql` 允许该用户可以不使用口令登录系统，然而从安全的角度而言，不推荐这种做法。
- `user_name`:指定创建用户账号，其格式为 `'user_name'@'host name'` 。 `user_name` 是用户名， `host_name` 为主机名，即用户连接 `mysql` 时所在主机的名字。如果在创建的过程中，只给出了账中的用户名，而没指定主机名，则主机名会默认为是 `'%'` ，表示一组主机。用户名的不能超过32个字符，区分大小写，主机名不区分大小写。
- `identified by` 子句：用于指定用户账号对应的口令，若该用户账号无口令，则可省略此子句。此处密码为明文密码，即登录时输入的密码。在数据表 `mysql.user` 对应存储了和明文密码对应的加密密码。如果两个用户具有相同的用户名但有不同的主机名， `mysql` 会将他们视为不同的用户，并允许为这两个用户分配不同的权限集合。
- 新创建的用户拥有的权限很少。他们可以登录到 `mysql` ，只允许进行不需要权限的操作，因此不能使用 `use` 语句来让其他用户已经创建的任何数据库成为当前数据库，因而无法访问相关数据库的表。

示例1：创建本机上的用户 `'ali'@'localhost'` ,然后通过 `ali` 登录本地的 `MySQL` ， 试试能否查看数据库中的对象。

```
CREATE USER 'ali'@'localhost' IDENTIFIED BY '11111';
CREATE USER 'ali' IDENTIFIED BY '11111';
SELECT `user`, `host`, `plugin`, `authentication_string` FROM mysql.user;
```

> - 在创建用户时，若不指定主机地址、密码以及相关的用户选项，则表示此用户在访问 `MySQL` 服务器时，不限定客户端、不需要密码并且没有任何限制
> - `host` 的值为 `"%"` 表示任何主机，其值为 `localhost` 时，表本地主机，其值为空字符串 `''` 时，表示所有客户端
> - `plugin` 字段用于指定用户的验证插件名称

示例2: 创建一个锁定状态的用户

```
CREATE USER 'ali'@'localhost' IDENTIFIED BY '11111' ACCOUNT LOCK;
SELECT `user`, `host`, `plugin`, `authentication_string`, `account_locked` FROM mysql.user;
```

创建用户时，可以添加 `WITH` 直接为用户指定可操作的资源范围，如登录的用户在一小时内可以查询数据的次数等。

表2 可操作资源范围

选项	描述
`MAX_QUERIES_PER_HOUR`	在任何一个小时内，允许此用户执行多少次查询
`MAX_UPDATES_PER_HOUR`	在任何一个小时内，允许此用户执行多少次更新
`MAX_CONNECTIONS_PER_HOUR`	在任何一个小时内，允许此用户执行多少次服务器连接
`MAX_USER_CONNECTIONS`	限制用户同时连接到服务器的最大数量

示例3：创建用户，并为用户指定可操作的资源范围

```
-- 限制其每小时最多可以更新10次
CREATE USER 'bob'@'localhost' IDENTIFIED BY '11111'
WITH MAX_UPDATES_PER_HOUR 10;
-- 查看user表的max_updates字段
SELECT max_updates
FROM mysql.user
WHERE `user`='bob' AND host='localhost';
```

在创建用户时，不仅可以为用户设置密码，还可以为密码设置有效时间。

表3 密码有效时间选项

选项	描述
`PASSWORD EXPIRE`	将密码标记为过期
`PASSWORD EXPIRE DEFAULT`	根据default_password_lifetime系统变量的指定设 置密码的有效性
`PASSWORD EXPIRE NEVER`	密码永不过期
`PASSWORD EXPIRE INTERVAL n DAY`	将帐户密码生存期设置为 n天

示例4： 创建用户，并为用户指定密码设置有效时间

```
-- 设置用户密码每180天更改一次
CREATE USER 'cindy'@'localhost' IDENTIFIED BY '11111'
PASSWORD EXPIRE INTERVAL 180 DAY;
```

> 为了确保 MySQL 客户端用户本身的安全，通常情况下推荐每3~6个月变更一次数据库用户密码。

b. 使用 grant 语句通过赋权创建用户账户

> 不推荐使用
>
> 早期版本(5.7 之前)有效，最新 8.0 版本已移除

```
-- 赋予某个账户权限，如果账户不存在，则新建该账户
GRANT privileges ON database.table
TO 'username'@'hostname'
[IDENTIFIED BY [PASSWORD]'password']
[,'username'@'hostname
[IDENTIFIED BY [PASSWORD]'password']] …
```

示例5：创建本机上的用户 'dog'@'localhost' ，并赋予其 purchase 数据库中 product 上的权限，然后通过 dog 登录 MySQL ，尝试相应权限功能。

```
-- mysql5.7运行成功，mysql8运行不成功
GRANT SELECT ON purchase.product
TO 'dog'@'localhost'
IDENTIFIED BY '11111';

-- 等价于
-- 创建用户
CREATE USER 'dog'@'localhost'
IDENTIFIED BY '11111';
-- 对该用户赋权
GRANT SELECT ON purchase.product
TO 'dog'@'localhost';
```

c. 使用 INSERT INTO mysql.user 直接往用户表 user 中插入一条用户记录

> 不推荐

可以使用 insert 语句直接将用户信息添加到 mysql.user 表中，但需要由对 user 表的插入权限。由于 user 表中的字段非常对，在插入数据时，要保证没有默认值的字段一定要给出值，所以插入数据时，至少要插入以下6个字段的值，
即 host,user,password,ss_cipher,x09_issuer,x509_subject 。

```
INSERT INTO mysql.user(Host, User, Authentication_string, ssl_cipher, x509_issuer,
x509_subject)
VALUES(<host_name>, <user_name>, <password>, '', '', '');

FLUSH PRIVILEGES;
```

2. 修改用户密码

- 第1种语法（推荐）

```
ALTER USER 账户名 IDENTIFIED BY '明文密码';
```

- 第2种语法:

```
SET PASSWORD [FOR账户名] = '明文密码';
```

> 可能会被记录到服务器的日志或客户端的历史文件中，会有密码泄露的风险，因此建议用户尽量少的使用此方式设置密码

示例6：为指定用户设置密码，将 `'ali'@'%'` 的密码修改为 `'22222'`

```
1   ALTER USER 'ali'@'%' IDENTIFIED BY '22222';
2   -- 或者
3   SET PASSWORD FOR 'ali'@'%' = '22222';
```

- **`mysqladmin` 修改用户密码**

> 在 `MySQL` 的安装目录 `bin` 下还有一个 `mysqladmin.exe` 应用程序，它通常用于执行一些管理性的工作，以及显示服务器状态等。同时，在 `MySQL` 中也可以使用该命令修改用户的密码。
>
> ```
> 1   mysqladmin -u 用户名 [-h 主机地址] -p password 新密码
> ```
>
> - -u：用于指定待要修改的用户名，通常情况下指的是 `root` 用户。
> - -h：用于指定对应的主机地址，省略时默认值为 `localhost` 。
> - -p：后面的 `password` 为关键字，而不是待修改用户的原密码。
>
> 通过 `mysqladmin` 完成用户密码的设置时，会有两个安全警告提示。因此，为了确保密码安全应以安全连接方式连接 `MySQL` 服务器。在开发中一般不推荐使用此种方式修改用户密码，此处将其作为知识扩展中需了解的内容即可。

- **`root` 密码丢失找回**

> `root` 用户是超级管理员，具有很多的权限，因此该用户的密码一旦丢失，就会造成很大的麻烦。针对这种情况， `MySQL` 提供了对应的处理机制，可以通过特殊方法登录到 `MySQL` 服务器，然后重新为 `root` 用户设置密码。 但是此种方式存在非常大的安全风险。因此， **建议读者要慎重使用**。
>
> 1. 停止 `MySQL` 服务。
>
> - `windows` 系统的资源管理器中，右击“计算机”，分别选择“管理/服务和应用程序/服务”
> - 停止 `MySQL` 服务
>
> 2. 在 `MySQL` 的配置文件 `my.ini` 或者 `my.cnf` 中添加 `skip-grant-tables` 选项。
>
> - 如果是 `Windows` 系统，请编辑 `C:\ProgramData\MySQL\MySQL Server 5.7\my.ini` ；如果是 `MACOS` 或者 `Linux` 系统，则 `vim /etc/my.cnf` 。具体 `vim` 的操作方式请参阅网络资料。

> - 在 `[mysqld]` 下，添加 `skip-grant-tables` ，以跳过权限的审核。
>
> 3. 重启 `MySQL` 服务。
> 4. 利用 `root` 用户登录，跳过密码的输入直接登录 `MySQL` 服务。
>
> - 登录 `MySQL` 服务器，不必指定用户名，主机名和密码
>
> 5. 为 `root` 用户设置密码。
>
> - 使用 `ALTER USER` 语句设置 `root` 用户密码
> - 使用新密码重新登录。
>
> 至此，密码修改完毕，请返回执行第1、2步，将 `[mysqld]` 下的 `skip-grant-tables` 去掉，再启动服务器。

3. 为用户重命名

在利用 `ALTER USER` 修改用户时只能修改指定账户的相关选项，如**密码、验证插件、资源控制选项**等，而不能够为用户重新命名，此时可以使用 `MySQL` 专门提供的 `RENAME USER` 语句实现。

```
RENAME USER
旧用户名1 TO 新用户名1 [, 旧用户名2 TO 新用户名2] …
```

- `RENAME USER` 在为用户重命名时，旧用户名与新用户名之间使用 `TO` 关键字连接。
- 同时为多个用户重命名时使用逗号 `,` 进行分割。

示例7：为指定用户设置密码

```
RENAME USER 'ali'@'localhost' TO 'alice'@'localhost';
```

4. 删除用户

在 `MySQL` 中经常会创建多个普通用户管理数据库，但如果发现某些用户是没有必要的，就可以将其删除，通常**删除用户**的方式采用 `MySQL` 提供的专门 `SQL` 语句。

```
DROP USER 'username'@'hostname'[,'username'@'hostname'];
```

- `MySQL5.6` 版本之后 `DROP USER` 语句可以同时删除一个或多个 `MySQL` 中的指定用户，并会同时从授权表中删除账户对应的权限行。
- `MySQL5.6` 之前的版本中，在删除用户前必须先回收用户的权限。

示例8：删除用户

```
DROP USER 'alice'@'localhost';
```

三、权限管理

权限信息根据其作用范围，分别存储在 `MySQL` 数据库中的**不同数据表**中。当 `MySQL` 启动时会自动加载这些权限信息，并将这些权限信息读取到内存中。根据**权限的操作内容**可将权限大致分为**数据权限**、**结构权限**以及**管理权限**。

权限级别指的就是特权可以被应用在哪些数据库的内容中。例如， `SELECT` 特权可以被授予到全局（任意数据库下的任意内容）、数据库（指定数据库下的任意内容）、表（指定数据库下的指定数据表）、列（指定数据库下的指定数据表中的指定字段）。

表4 权限的操作内容

权限类别	特权	权限级别	描述
数据权限	SELECT	全局、数据库、表、列	SELECT
数据权限	UPDATE	全局、数据库、表、列	UPDATE
数据权限	DELETE	全局、数据库、表	DELETE
数据权限	INSERT	全局、数据库、表、列	INSERT
数据权限	SHOW DATABASES	全局	SHOW DATABASES
数据权限	SHOW VIEW	全局、数据库、表	SHOW CREATE VIEW
数据权限	PROCESS	全局	SHOW PROCESSLIST

结构权限	DROP	全局、数据库、表	允许删除数据库、表和视图
结构权限	CREATE	全局、数据库、表	创建数据库、表
结构权限	CREATE ROUTINE	全局、数据库	创建存储过程
结构权限	CREATE TABLESPACE	全局	允许创建、修改或删除表空间和日志文件组
结构权限	CREATE TEMPORARY TABLES	全局、数据库	CREATE TEMPORARY TABLE
结构权限	CREATE VIEW	全局、数据库、表	允许创建或修改视图
结构权限	ALTER	全局、数据库、表	ALTER TABLE
结构权限	ALTER ROUTINE	全局、数据库、存储过程	允许删除或修改存储过程
结构权限	INDEX	全局、数据库、表	允许创建或删除索引
结构权限	TRIGGER	全局、数据库、表	允许触发器的所有操作

结构权限	REFERENCES	全局、数据库、表、列	允许创建外键
管理权限	SUPER	全局	允许使用其他管理操作，如CHANGE MASTER TO等
管理权限	CREATE USER	全局	CREATE USER、DROP USER、 RENAME USER 和REVOKEALL PRIVILEGES
管理权限	GRANT OPTION	全局、数据库、表、存储过程、代理	允许授予或删除用户特权
管理权限	RELOAD	全局	FLUSH操作
管理权限	PROXY		与代理的用户权限相同
管理权限	REPLICATION CLIENT	全局	允许用户访问主服务器或从服务器
管理权限	REPLICATION SLAVE	全局	允许复制从服务器读取的主服务器二进制日志事件
管理权限	SHUTDOWN	全局	允许使用mysqladmin shutdown
管理权限	LOCK TABLES	全局、数据库	允许在有SELECT表权限上使用LOCK TABLES

1. 授予用户权限

授予权限的语法：

```
GRANT 权限类型 [字段列表][, 权限类型 [字段列表]] ...
ON [目标类型] 权限级别
TO 账户名 [用户身份验证选项] [, 账户名 [用户身份验证选项]] ...
[WITH {GRANT OPTION | 资源控制选项}]
```

- **权限类型**：指的就是 `SELECT、DROP、CREATE` 等特权。
- **字段列表**：用于设置列权限。
- **目标类型**：默认为 `TABLE` ，表示将全局、数据库、表或列中的某些权限授予给指定的用户。其他值为 `FUNCTION` （函数）或 `PROCEDURE` （存储过程）。
- **权限级别**：用于定义全局权限、数据库权限和表权限。
- 添加 `WITH GRANT OPTION` ：表示当前账户可以为其他账户进行授权。
- **其余各参数**均与 `CREATE USER` 中的用户选项相同，这里不再赘述。

示例9：创建用户 `'ali'@'localhost'` ，并向其授权 `product` 表上的 `insert` 和 `select` 权限，并允许它将这些权限赋予其它用户。

```
CREATE USER 'ali'@'localhost' IDENTIFIED BY '11111';

GRANT insert, select ON purchase.product
TO 'ali'@'localhost'
WITH GRANT OPTION;
```

退出 `mysql` ，然后以 `'ali'@'localhost'` 重新登录

```
mysql -h localhost -u ali -p
```

验证 `ali` 具有的权限

```
show databases;
use purchase;

select * from product limit 5;

insert into product(product_id, product_name, sort_id, subsort_id)
values (6666, '晨光牌复印机', 11, 1101);
```

2. 查看用户权限

查看权限的语法：

```
SHOW GRANTS FOR [用户名];
```

示例10：查看用户 `'ali'@'localhost'` 的权限

```
show grants for 'ali'@'localhost';
```

3. 回收用户权限

在 MySQL 中，为了保证数据库的安全性，需要将用户不必要的**权限回收**。例如，数据管理员发现某个用户不应该具有 DELETE 权限，就应该及时将其收回。为此， MySQL 专门提供了一个 REVOKE 语句用于回收指定用户的权限。回收权限的语法如下：

```
-- 回收用户指定权限
REVOKE 权限类型 [(字段列表)] [, 权限类型[(字段列表)]] …
ON [目标类型] 权限级别 FROM 账户名 [, 账户名] …

-- 回收用户所有权限
REVOKE ALL [PRIVILEGES], GRANT OPTION FROM 账户名 [, 账户名] …
```

示例11：回收用户 'ali'@'localhost' 的对 product 表的 insert 权限

```
REVOKE INSERT ON purchase.product FROM ali@localhost;
```

示例12：回收用户 'ali'@'localhost' 的所有权限

```
REVOKE ALL PRIVILEGES, GRANT OPTIONI FROM ali@localhost;
```

注意，在对用户进行权限操作之后，需要对权限表进行刷新，即从系统数据库 mysql 中的权限表中重新加载用户的特权。 GRANT、CREATE USER 等操作会将服务器的缓存信息保存到内存中，而 REVOKE、DROP USER 操作并不会同步到内存中，因此可能会造成服务器内存的消耗，所以在 REVOKE、DROP USER 后推荐读者使用 MySQL 提供的 FLUSH PRIVILEGES 重新加载用户的特权。