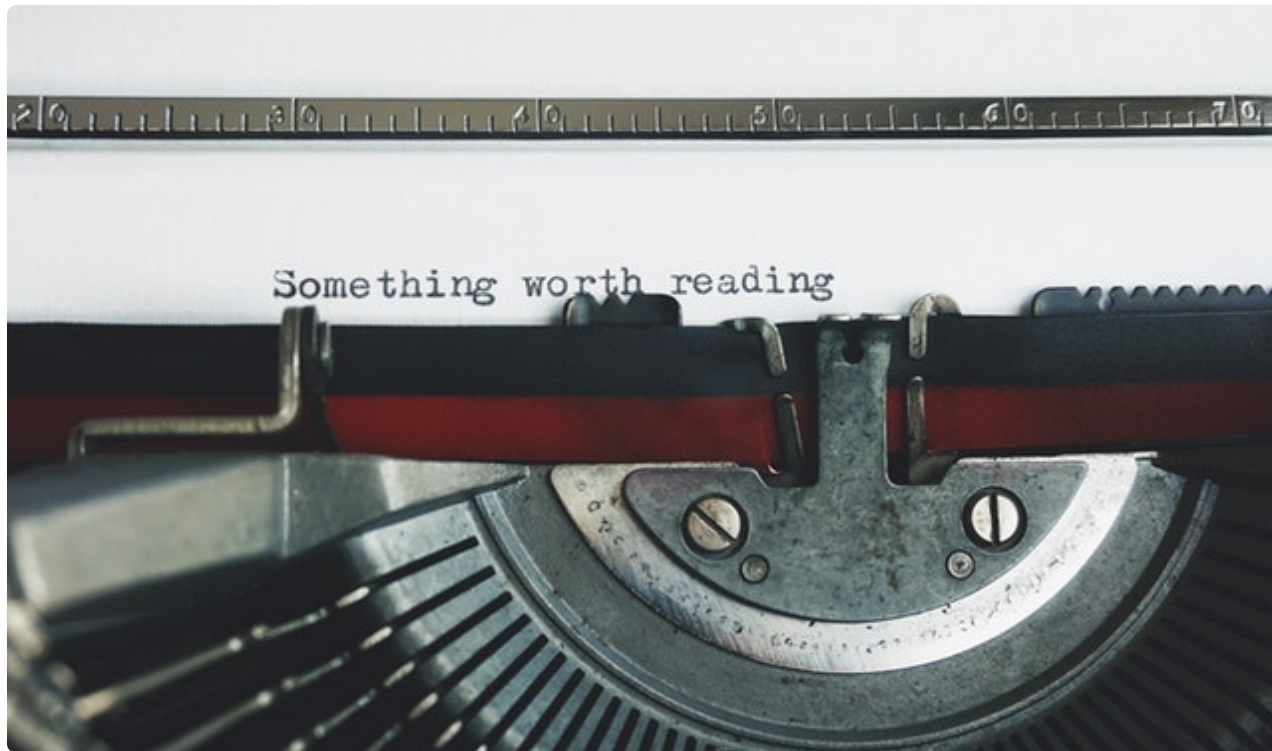


06 常常被忽略的用户与权限

更新时间：2020-03-13 09:42:11



“ 什么是路？就是从没路的地方践踏出来的，从只有荆棘的地方开辟出来的。 —— 鲁迅 ”

相信大家对于“用户与权限”这个名词一定不会陌生，我们的应用系统、Redis 集群、Hadoop 集群等等都会有各自对应的用户与权限体系。

对于 MySQL 这种多用户数据库来说，拥有强大的访问控制体系自然是理所应当的，而访问控制过程也正是用户与权限所负责的。但是，这个概念常常被忽略，因为大多数的时间，我们都是询问 DBA 获取一个可用的用户名和密码，而它背后的逻辑与思想也就自然没有去深究了。

这一节里，我们需要围绕三个问题去看待 MySQL 的用户与权限：

- MySQL 是怎样存储权限的，又对权限做了怎样的分类？
- MySQL 是如何控制用户行为的？
- 怎样在 MySQL 中去操作用户与权限？

下面，我们就带着这三个问题来系统的学习下常常被忽略的用户与权限吧。

1 权限的存储与分类

既然要匹配哪些用户拥有哪些权限，那么这些用户及权限的信息一定得有地方存储才行。我们在安装完 MySQL 之后，可以看到有一个系统库叫做 `mysql`，用户及权限信息就存储在这个库中。同时，由于 MySQL 中的权限太多（超过 20 个），做好分类可以简化学习过程，也方便记忆。

1.1 权限的存储

用户与权限信息存储在 `mysql` 系统库中，一共用 5 张表去表达，分别是：`user`、`db`、`tables_priv`、`columns_priv` 和 `procs_priv`。同时，这 5 张表也代表了 5 个层级。下面，我们来详细的说一说这几张表的含义（只需要知道这几张表，暂时不用看它们的结构，我后面会讲到）。

- **user 表**：它属于全局层级，存储用户账户信息以及全局级别的权限，这些全局权限适用于系统中所有的数据库
- **db 表**：它属于数据库层级，适用于一个给定数据库中的所有目标
- **tables_priv 表**：它属于表层级，适用于一个给定表中的所有列
- **columns_priv 表**：它属于列层级，适用于一个给定表中的单一列
- **procs_priv 表**：它属于子程序层级，存放存储过程和函数级别的权限

从层级关系中可以看出，除 `procs_priv` 表之外，各层级的权限范围依次递减，即用户（代表所有）、库、表和列。

1.2 权限的分类

搞清楚了权限存储的位置之后，我们接下来看一看 `MySQL` 支持哪些权限，以及怎样对这些权限进行分类。下面，我先用一张表来说明 `MySQL` 中常见的权限。

| 权限 | 权限层级 | 权限解释 |
|----------------|--------|------------------------------|
| CREATE | 库、表和索引 | 创建库、表和索引的权限 |
| DROP | 库、表 | 删除库和表的权限 |
| ALTER | 表 | 更改表的权限，例如：添加字段、修改字段类型 |
| DELETE | 表 | 删除数据的权限 |
| INDEX | 表 | 索引权限 |
| INSERT | 表 | 插入权限 |
| SELECT | 表 | 查询权限 |
| UPDATE | 表 | 更新权限 |
| CREATE VIEW | 视图 | 创建视图权限 |
| SHOW VIEW | 视图 | 查看视图权限 |
| CREATE ROUTINE | 存储过程 | 创建存储过程的权限 |
| ALTER ROUTINE | 存储过程 | 更改存储过程的权限 |
| EXECUTE | 存储过程 | 执行存储过程的权限 |
| PROCESS | 服务器 | 查看进程的权限 |
| SHOW DATABASES | 服务器 | 查看数据库的权限 |
| SHUTDOWN | 服务器 | 关闭数据库的权限 |
| SUPER | 服务器 | 执行 <code>KILL</code> 用户线程的权限 |

其实，根据上表中对权限层级的描述，也就能够大概知道怎样对权限进行分类了。这里我们大致可以把权限简单的分为三类：

- 数据：对应到库、表、索引和列层级的权限都归为数据权限
- 管理：对应到服务器层级的权限，即管理员所操作的都归为管理权限
- 程序：对应到存储过程层级的权限都归为程序权限

当然，这些权限的分类是“自定义的”，也就是说并不是 `MySQL` 标准中定义的，你也完全可以按照你的理解去把它们进行分类。

2 解读 `MySQL` 中的权限表

到这里，我们已经知道了 MySQL 使用 5 张表去存储用户与权限信息，但是并没有细致的讲解这其中各个表中的字段代表什么样的含义。下面，我们来依次解读下这 5 张表。

2.1 mysql.user 表

user 表的权限是用户层级（全局），是基于服务器范围的所有权限。这个表包含的列非常多，其中大多数都以 **priv** 结尾，也就是权限标识的意思。在这里我以其中的一列 **Select_priv** 来说明含义：如果某个用户拥有所有数据库的 **SELECT** 权限，那么，这个用户对应的记录 **Select_priv** 列值就为 **Y**，否则为 **N**（枚举类型定义）。下面，我用一张表解释下 user 表各个字段的含义（并不对字段类型、默认值做过多说明，可以使用 **desc** 自行查看）。

| 字段名 | 字段解释 |
|------------------------|---|
| Host | 主机名，其中： localhost 代表本地； % 代表不受限制；单独的 ip 代表某一台机器的访问权限 |
| User | 用户名 |
| Select_priv | SELECT 权限 |
| Insert_priv | INSERT 权限 |
| Update_priv | UPDATE 权限 |
| Delete_priv | DELETE 权限 |
| Create_priv | CREATE 权限 |
| Drop_priv | DROP 权限 |
| Reload_priv | 刷新和重新加载 MySQL 内部缓存的权限 |
| Shutdown_priv | 关闭 MySQL 服务器的权限，只能是 root 用户 |
| Process_priv | SHOW PROCESSLIST 命令权限 |
| File_priv | SELECT INTO OUTFILE 和 LOAD DATA INFILE 命令权限 |
| Grant_priv | 权限授予其他用户的权限 |
| References_priv | 参照表权限 |
| Index_priv | 索引操作（创建、删除）权限 |
| Alter_priv | ALTER 权限 |
| Show_db_priv | 查看数据库权限 |
| Super_priv | Super 权限 |
| Create_tmp_table_priv | 创建临时表权限 |
| Lock_tables_priv | 执行 lock table 权限 |
| Execute_priv | 执行存储过程的权限 |
| Repl_slave_priv | 复制权限 |
| Repl_client_priv | 复制权限 |
| Create_view_priv | 创建视图权限 |
| Show_view_priv | 查看视图权限 |
| Create_routine_priv | 创建存储过程、函数的权限 |
| Alter_routine_priv | 修改、删除存储过程、函数的权限 |
| Create_user_priv | 创建用户的权限 |
| Event_priv | 操作（创建、修改、删除）事件的权限 |
| Trigger_priv | 操作（创建、修改、删除）触发器的权限 |
| Create_tablespace_priv | 创建表空间的权限 |
| ssl_type | 加密 |
| ssl_cipher | 加密 |
| x509_issuer | 标识用户 |
| x509_subject | 标识用户 |
| max_questions | 每小时允许执行多少次查询，0 表示没有限制 |
| max_updates | 每小时允许多少次更新 |
| max_connections | 每小时允许建立多少连接 |
| max_user_connections | 单个用户可以同时建立的连接数 |
| plugin | 认证插件 |
| authentication_string | 登录密码 |
| password_expired | 密码是否过期 |
| password_last_changed | 密码最近一次修改时间 |

| 字段名 | 字段解释 |
|-------------------|---------|
| password_lifetime | 密码使用期限 |
| account_locked | 账户是否被锁定 |

由于 `user` 表的权限级别太高，责任重大，所以字段也就会稍微多一些。但是，仔细观察，你会发现，表字段的命名非常讲究，易于理解。

最后，需要特别提醒一下，MySQL 专门提供了命令创建用户与权限，表的修改过程也是封装在这些命令的实现中。如果你不是很清楚修改这张表会造成什么影响，一定不要尝试修改。

2.2 mysql.db 表

如果授予了某个用户单独一个数据库的权限，MySQL 就会在 `db` 表中保存一条记录。同样，下面我用一张表来解读下 `db` 表中各个字段的含义。

| 字段名 | 字段解释 |
|-----------------------|---|
| Host | 主机名，其中： <code>localhost</code> 代表本地； <code>%</code> 代表不受限制；单独的 <code>ip</code> 代表某一台机器的访问权限 |
| Db | 数据库名 |
| User | 用户名 |
| Select_priv | SELECT 权限 |
| Insert_priv | INSERT 权限 |
| Update_priv | UPDATE 权限 |
| Delete_priv | DELETE 权限 |
| Create_priv | CREATE 权限 |
| Drop_priv | DROP 权限 |
| Grant_priv | 权限授予其他用户的权限 |
| References_priv | 参照表权限 |
| Index_priv | 索引操作（创建、删除）权限 |
| Alter_priv | ALTER 权限 |
| Create_tmp_table_priv | 创建临时表权限 |
| Lock_tables_priv | 执行 <code>lock table</code> 权限 |
| Create_view_priv | 创建视图权限 |
| Show_view_priv | 查看视图权限 |
| Create_routine_priv | 创建存储过程、函数的权限 |
| Alter_routine_priv | 修改、删除存储过程、函数的权限 |
| Execute_priv | 执行存储过程的权限 |
| Event_priv | 操作（创建、修改、删除）事件的权限 |
| Trigger_priv | 操作（创建、修改、删除）触发器的权限 |

看上去似乎与 `user` 表的字段定义是相同的，这种感觉也是没问题的，只是 `db` 表中的每一条记录代表的是一个库，而 `user` 表记录代表的是所有库。

2.3 mysql.tables_priv 表

表级权限相对来说就比较简单了，根据之前对 `user` 和 `db` 的理解，我们应该可以猜到这张表的核心字段是“表名”。关于这张表的字段解读，可以参照下表所示。

| 字段名 | 字段解释 |
|------|---|
| Host | 主机名，其中： <code>localhost</code> 代表本地； <code>%</code> 代表不受限制；单独的 <code>ip</code> 代表某一台机器的访问权限 |
| Db | 数据库名 |
| User | 用户名 |

| 字段名 | 字段解释 |
|-------------|--|
| Table_name | 表名 |
| Grantor | 谁授予该用户的权限 |
| Timestamp | 授予权限的时间戳 |
| Table_priv | 授予表的哪些权限（例如：Select、Insert、Drop、Index 等等） |
| Column_priv | 对表中的字段授予的权限（Select、Insert、Update、References） |

也许你对这张表最大的疑问是 **Column_priv** 字段，毕竟 **tables_priv** 表定义的是表级别的权限，为什么需要有列级权限的声明呢？其实这是 **MySQL** 做的一种优化，为了提高权限检查时的性能。即在权限检查时，如果发现 **Column_priv** 列值为空，就不需要再去检查列级权限了，省去了一步查询操作。

2.4 mysql.columns_priv 表

列级权限表与 **tables_priv** 表的结构是高度相似的，只是多了一列 **Column_name**。下面，我们来继续看这张表的解读。

| 字段名 | 字段解释 |
|-------------|--|
| Host | 主机名，其中： localhost 代表本地；% 代表不受限制；单独的 ip 代表某一台机器的访问权限 |
| Db | 数据库名 |
| User | 用户名 |
| Table_name | 表名 |
| Column_name | 列名 |
| Timestamp | 授予权限的时间戳 |
| Column_priv | 授予列的权限（Select、Insert、Update、References） |

2.5 mysql.procs_priv 表

procs_priv 表与之前介绍的 4 张表的结构有所不同，但是有了之前的层级权限基础，理解这张表也是比较简单的，解读如下。

| 字段名 | 字段解释 |
|--------------|--|
| Host | 主机名，其中： localhost 代表本地；% 代表不受限制；单独的 ip 代表某一台机器的访问权限 |
| Db | 数据库名 |
| User | 用户名 |
| Routine_name | 程序名 |
| Routine_type | 程序类型（函数、存储过程） |
| Grantor | 谁授予该用户的权限 |
| Proc_priv | 授予的权限（Execute、Alter Routine、Grant） |
| Timestamp | 授予权限的时间戳 |

到这里，我们基本就把 **MySQL** 中的用户与权限的概念、思想和存储搞清楚了。接下来，我们再去看一看 **MySQL** 怎样利用这些存储的权限对用户的连接和查询做访问控制。

3 MySQL 的访问控制

MySQL 的访问控制分为两个阶段：用户连接阶段和执行查询阶段。抛开权限验证不说，这两个阶段其实也就是你在使用 **MySQL** 时的所有操作了。所以，可以看到，我们在使用 **MySQL** 的过程中，权限验证自始至终都伴随着，它有多重要想必你也应该有答案了。

3.1 用户连接阶段

类似的权限验证出现在我们工作中的方方面面，我们在登录任何系统的时候，都会让我们输入用户名和密码，完全匹配之后才能进行接下来的操作。但是，对于 MySQL 来说，为了保证数据的安全，除了最基本的用户名和密码之外，还会有更加严格的限制。

- 用户连接时，MySQL 服务器首先匹配 `mysql.user` 表中的主机名、用户名和密码，匹配不到则拒绝当前的连接请求
- 检查 `mysql.user` 表的 `max_connections` 和 `max_user_connections` 字段值，如果超过设置的值范围则拒绝连接请求
- 检查 `mysql.user` 表的 SSL 安全连接配置，如果有配置 SSL，则需要用户提供证书且是合法的

不论我们是通过哪一种方式与 MySQL 服务器建立连接，MySQL 都会依次执行以上的 3 个权限校验工作。只有当所有的检查都通过后，服务器才会与客户端建立连接。当连接建立之后，用户使用 SQL 语句执行增删改查时，MySQL 服务器又需要做执行查询阶段的验证工作。

3.2 执行查询阶段

查询阶段与连接阶段的检查工作是类似的，也是会对照之前介绍的各个存储表进行比对工作，也同样会有几个步骤的操作。下面，一起来看看吧：

- 检查 `mysql.user` 表的 `max_questions` 和 `max_updates` 字段值，如果超过上限值，则拒绝执行 SQL 语句
- 检查 `mysql.user` 表，如果拥有全局性权限，直接执行；否则，继续下一步检查
- 检查 `mysql.db` 表，如果拥有数据库级别的权限，则执行；否则，继续下一步检查
- 检查 `mysql.tables_priv`, `mysql.columns_priv`, `mysql.procs_priv` 表，如果拥有相应对象的权限，则执行；否则，上报权限不足错误

从以上的检查过程可以看出，第一个检查是对属性规定的要求检查，后三个才是对权限的检查。同时，也可以看出，MySQL 的权限检查是一个比较复杂的过程。所以，为了提高性能，MySQL 在启动时会把 5 张权限表加载到内存中，是典型的空间换时间的思想。

4 用户与权限管理实践

虽然我们在使用 MySQL 的过程中，几乎不会涉及到用户与权限的问题，因为这些都被 DBA 处理了。但是，如果你想完全掌控自己的 MySQL 或者是在没有 DBA 的情况下，学会用户与权限的实践操作就显得很有必要了。下面，我将主要围绕创建用户、分配权限来对这个话题进行讲解。

4.1 用户与权限管理的操作

想要给用户授予或者是删除权限，首先得有个用户。我们在 MySQL 中可以通过 `CREATE USER` 或 `GRANT` 语句来创建用户，它们的区别是 `GRANT` 可以在创建用户的同时授予权限。下面，我们先通过 `CREATE USER` 来创建用户 `imooc-1`：

```
-- 语法：CREATE USER 'USERNAME'@'HOST' IDENTIFIED BY 'PASSWORD';
-- 其中 HOST 可以是 ip 地址、主机名、网段地址或通配符
mysql> CREATE USER 'imooc-1'@'localhost' IDENTIFIED BY 'imooc';
Query OK, 0 rows affected (0.04 sec)
```

就这样，我们创建完成了用户 `imooc-1`，是不是非常的简单（此时，机智的你是不是去看一看那几张权限表发生了哪些变化呢？）。接下来，我们再去通过 `GRANT` 语句去创建用户 `imooc-2`：


```
-- 语法: GRANT PRIVILEGES ON DB_NAME.TABLE_NAME TO 'USERNAME'@'HOST' IDENTIFIED BY 'PASSWORD';
-- 其中 PRIVILEGES 代表想要授予的权限, ALL PRIVILEGES 代表所有的权限; * 是通配符, 代表所有的库和表
mysql> GRANT ALL PRIVILEGES ON *.* TO 'imoooc-2'@'localhost' IDENTIFIED BY 'imoooc';
Query OK, 0 rows affected, 1 warning (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.02 sec)
```

GRANT 语句的语法也是相当简单的, 需要注意的是紧接着执行了一个 **FLUSH PRIVILEGES** 操作, 这是因为 **MySQL** 的权限数据在缓存中, 我们更改了权限表, 需要刷新缓存。下面, 我将给出 **GRANT** 语句的常见授权方法 (结合注释)。掌握了创建用户并给用户授权, 也就基本掌握了用户与权限管理的操作。

```
-- 授予所有的权限
GRANT ALL PRIVILEGES ON *.* TO 'USERNAME'@'HOST';

-- 授予某个库所有表的所有权限
GRANT ALL PRIVILEGES ON DB_NAME.* TO 'USERNAME'@'HOST';

-- 授予某个库所有表的 SELECT、UPDATE 权限
GRANT SELECT, UPDATE ON DB_NAME.* TO 'USERNAME'@'HOST';

-- 授予某个库下某个表的 INSERT 权限
GRANT INSERT ON DB_NAME.TABLE_NAME TO 'USERNAME'@'HOST';

-- 授予某个库下某个表的某个列的 UPDATE 权限
GRANT UPDATE(COLUMN_NAME) ON DB_NAME.TABLE_NAME TO 'USERNAME'@'HOST';

-- 授予某个库下创建、修改、删除表结构的权限
GRANT CREATE ON DB_NAME.* TO 'USERNAME'@'HOST';
GRANT ALTER ON DB_NAME.* TO 'USERNAME'@'HOST';
GRANT DROP ON DB_NAME.* TO 'USERNAME'@'HOST';
```

以上基本上就是 **GRANT** 语句的使用方法, 更多的权限授予语句无非是 **MySQL** 提供的权限的组合。需要记住, 每次执行授权语句之后, 都需要去刷新权限缓存, 即执行 **FLUSH PRIVILEGES** 操作。想一想, 我如果授予了错误的用户权限, 想要收回怎么办呢? 别急, **MySQL** 提供了 **REVOKE** 语句, 它的使用方法如下所示。

```
-- 撤销某个库下某个表的 DROP 权限
REVOKE DROP ON DB_NAME.TABLE_NAME FROM 'USERNAME'@'HOST';

-- 撤销某个库下某个表的所有权限
REVOKE ALL ON DB_NAME.TABLE_NAME FROM 'USERNAME'@'HOST';
```

同样, 执行撤销授权语句之后, 还需要再次执行 **FLUSH PRIVILEGES** 操作刷新缓存。但同时, 需要注意, 撤销的权限必须是用户已经被授予的权限。否则, **MySQL** 将会返回错误。那么, 如果我忘记了某个用户是否拥有某个权限, 又该怎么办呢? **MySQL** 当然想到了这一点, 继续往下看吧:

```
mysql> SHOW GRANTS FOR 'imoooc-1'@'localhost';
+-----+
| Grants for imoooc-1@localhost |
+-----+
| GRANT USAGE ON *.* TO 'imoooc-1'@'localhost' |
| GRANT SELECT, UPDATE, CREATE, DROP, ALTER ON `imoooc_mysql`.* TO 'imoooc-1'@'localhost' |
| GRANT INSERT, UPDATE (type) ON `imoooc_mysql`.`worker` TO 'imoooc-1'@'localhost' |
+-----+
```

可以看到, 使用 **SHOW GRANTS FOR** 可以查看到格式非常清晰的已经授予的权限。不得不说, **MySQL** 考虑的是非常全面了。

4.2 用户与权限管理的建议

由于用户与权限在任何系统中都是比较敏感的存在，对于 MySQL 这种管理数据的就更不用多说。所以，一定要谨慎使用。下面，我将总结一些经验之谈，给出用户与权限管理的建议。

- 使用 `CREATE USER` 或 `GRANT` 语句去创建用户，不要直接去修改系统表，这是非常危险的操作
- 授予用户能满足使用的最小权限，避免越权访问，最常见的就是只读账号
- 授予权限时一定要指定 `ip` 地址或 `ip` 段，避免外部撞库攻击
- 不允许创建弱密码的用户，密码最好包含：字母（大小写都有）、数字、特殊字符
- 定期删除无效用户，或者是回收不再需要的权限

关于以上这些建议只是一些通用场景，大家在实际使用时只作为参考就好，并不是一种限制或者约束。另外，多去尝试应用，记录并分享你在工作中的经验与建议。

5 总结

MySQL 中的用户与权限是很容易被大家忽略的话题，也确实是因为在日常的工作中，它出现的频率太低。但是，这并不意味着它就不重要，相反，它是使用 MySQL 的基础，也是保证数据安全的必备技能。大家并不一定要记住它的方方面面，理解其核心思想，在使用到的时候再回过头去看、去查就是非常好的学习方式。

6 问题

试一试在你的本地 MySQL 上创建新用户并分配一些权限

从 MySQL 的用户与权限设计，你能得到什么样的启发？

7 参考资料

《高性能 MySQL（第三版）》

[MySQL 官方文档: Access Control and Account Management](#)

[MySQL 官方文档: Security Plugins](#)

}



05 很有用的条件判断函数与系统函数

07 掌握数据备份与恢复是很有必要的

