42 | grant之后要跟着flush privileges吗？

2019-02-18 林晓斌

在MySQL里面，grant语句是用来给用户赋权的。不知道你有没有见过一些操作文档里面提到，grant之后要马上跟着执行一个flush privileges命令，才能使赋权语句生效。我最开始使用MySQL的时候，就是照着一个操作文档的说明按照这个顺序操作的。

那么，grant之后真的需要执行flush privileges吗？如果没有执行这个flush命令的话，赋权语句真的不能生效吗？

接下来，我就先和你介绍一下grant语句和flush privileges语句分别做了什么事情，然后再一起来分析这个问题。

为了便于说明，我先创建一个用户：

```
create user 'ua'@'%' identified by 'pa';
```

这条语句的逻辑是创建一个用户'ua'@'%'，密码是pa。注意，在MySQL里面，用户名(user)+地址(host)才表示一个用户，因此 ua@ip1 和 ua@ip2代表的是两个不同的用户。

这条命令做了两个动作：

1. 磁盘上，往mysql.user表里插入一行，由于没有指定权限，所以这行数据上所有表示权限的字段的值都是N;

2. 内存里，往数组acl_users里插入一个acl_user对象，这个对象的access字段值为0。

图1就是这个时刻用户ua在user表中的状态。

```
mysql> select * from mysql.user where user='ua'\G
*************************** 1. row ***************************
                  Host: %
                  User: ua
           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
```

图1 mysql.user 数据行

在MySQL中，用户权限是有不同的范围的。接下来，我就按照用户权限范围从大到小的顺序依次和你说明。

全局权限

全局权限，作用于整个MySQL实例，这些权限信息保存在mysql库的user表里。如果我要给用户

ua赋一个最高权限的话，语句是这么写的：

```
grant all privileges on *.* to 'ua'@'%' with grant option;
```

这个grant命令做了两个动作：

1. 磁盘上，将mysql.user表里，用户'ua'@'%'这一行的所有表示权限的字段的值都修改为'Y'；
2. 内存里，从数组acl_users中找到这个用户对应的对象，将access值（权限位）修改为二进制的“全1”。

在这个grant命令执行完成后，如果有新的客户端使用用户名ua登录成功，MySQL会为新连接维护一个线程对象，然后从acl_users数组里查到这个用户的权限，并将权限值拷贝到这个线程对象中。之后在这个连接中执行的语句，所有关于全局权限的判断，都直接使用线程对象内部保存的权限位。

基于上面的分析我们可以知道：

1. grant 命令对于全局权限，同时更新了磁盘和内存。命令完成后即时生效，接下来新创建的连接会使用新的权限。
2. 对于一个已经存在的连接，它的全局权限不受grant命令的影响。

需要说明的是，**一般在生产环境上要合理控制用户权限的范围**。我们上面用到的这个grant语句就是一个典型的错误示范。如果一个用户有所有权限，一般就不应该设置为所有IP地址都可以访问。

如果要回收上面的grant语句赋予的权限，你可以使用下面这条命令：

```
revoke all privileges on *.* from 'ua'@'%';
```

这条revoke命令的用法与grant类似，做了如下两个动作：

1. 磁盘上，将mysql.user表里，用户'ua'@'%'这一行的所有表示权限的字段的值都修改为“N”；
2. 内存里，从数组acl_users中找到这个用户对应的对象，将access的值修改为0。

db权限

除了全局权限，MySQL也支持库级别的权限定义。如果要让用户ua拥有库db1的所有权限，可以执行下面这条命令：

```
grant all privileges on db1.* to 'ua'@'%' with grant option;
```

基于库的权限记录保存在mysql.db表中，在内存里则保存在数组acl_dbs中。这条grant命令做了如下两个动作：

1. 磁盘上，往mysql.db表中插入了一行记录，所有权限位字段设置为“Y”；
2. 内存里，增加一个对象到数组acl_dbs中，这个对象的权限位为“全1”。

图2就是这个时刻用户ua在db表中的状态。

```
mysql> select * from mysql.db where user='ua'\G
*************************** 1. row ***************************
                 Host: %
                   Db: db1
                 User: ua
          Select_priv: Y
          Insert_priv: Y
          Update_priv: Y
          Delete_priv: Y
          Create_priv: Y
            Drop_priv: Y
           Grant_priv: Y
      References_priv: Y
           Index_priv: Y
           Alter_priv: Y
Create_tmp_table_priv: Y
     Lock_tables_priv: Y
     Create_view_priv: Y
       Show_view_priv: Y
  Create_routine_priv: Y
   Alter_routine_priv: Y
         Execute_priv: Y
           Event_priv: Y
         Trigger_priv: Y
1 row in set (0.00 sec)
```

图2 mysql.db 数据行

每次需要判断一个用户对一个数据库读写权限的时候，都需要遍历一次acl_dbs数组，根据user、host和db找到匹配的对象，然后根据对象的权限位来判断。

也就是说，grant修改db权限的时候，是同时对磁盘和内存生效的。

grant操作对于已经存在的连接的影响，在全局权限和基于db的权限效果是不同的。接下来，我们做一个对照试验来分别看一下。

	session A	session B	session C
T1	connect(root,root) create database db1; create user 'ua'@'%' identified by 'pa'; grant super on *.* to 'ua'@'%'; grant all privileges on db1.* to 'ua'@'%';		

T2		connect(ua,pa)
		set global sync_binlog=1;
		(Query OK)
		create table db1.t(c int);
		(Query OK)
			connect(ua,pa)
			use db1;

T3	revoke super on *.* from 'ua'@'%';		

T4		set global sync_binlog=1;
		(Query OK)
		alter table db1.t engine=innodb;
		(Query OK)
			alter table t engine=innodb;
			(Query OK)

T5	revoke all privileges on db1.* from 'ua'@'%';		

T6		set global sync_binlog=1;
		(Query OK)
		alter table db1.t engine=innodb;
		(ALTER command denied)
			alter table t engine=innodb;
			(Query OK)

图3 权限操作效果

需要说明的是，图中set global sync_binlog这个操作是需要super权限的。

可以看到，虽然用户ua的super权限在T3时刻已经通过revoke语句回收了，但是在T4时刻执行set global的时候，权限验证还是通过了。这是因为super是全局权限，这个权限信息在线程对象中，而revoke操作影响不到这个线程对象。

而在T5时刻去掉ua对db1库的所有权限后，在T6时刻session B再操作db1库的表，就会报错“权限不足”。这是因为acl_dbs是一个全局数组，所有线程判断db权限都用这个数组，这样revoke操

作马上就会影响到session B。

这里在代码实现上有一个特别的逻辑，如果当前会话已经处于某一个db里面，之前use这个库的时候拿到的库权限会保存在会话变量中。

你可以看到在T6时刻，session C和session B对表t的操作逻辑是一样的。但是session B报错，而session C可以执行成功。这是因为session C在T2 时刻执行的use db1，拿到了这个库的权限，在切换出db1库之前，session C对这个库就一直有权限。

表权限和列权限

除了db级别的权限外，MySQL支持更细粒度的表权限和列权限。其中，表权限定义存放在表mysql.tables_priv中，列权限定义存放在表mysql.columns_priv中。这两类权限，组合起来存放在内存的hash结构column_priv_hash中。

这两类权限的赋权命令如下：

```
create table db1.t1(id int, a int);


grant all privileges on db1.t1 to 'ua'@'%' with grant option;
GRANT SELECT(id), INSERT (id,a) ON mydb.mytbl TO 'ua'@'%' with grant option;
```

跟db权限类似，这两个权限每次grant的时候都会修改数据表，也会同步修改内存中的hash结构。因此，对这两类权限的操作，也会马上影响到已经存在的连接。

看到这里，你一定会问，看来grant语句都是即时生效的，那这么看应该就不需要执行flush privileges语句了呀。

答案也确实是这样的。

flush privileges命令会清空acl_users数组，然后从mysql.user表中读取数据重新加载，重新构造一个acl_users数组。也就是说，以数据表中的数据为准，会将全局权限内存数组重新加载一遍。

同样地，对于db权限、表权限和列权限，MySQL也做了这样的处理。

也就是说，如果内存的权限数据和磁盘数据表相同的话，不需要执行flush privileges。而如果我们都是用grant/revoke语句来执行的话，内存和数据表本来就是保持同步更新的。

因此，正常情况下，grant命令之后，没有必要跟着执行flush privileges命令。

flush privileges使用场景

那么，flush privileges是在什么时候使用呢？显然，当数据表中的权限数据跟内存中的权限数据不一致的时候，flush privileges语句可以用来重建内存数据，达到一致状态。

这种不一致往往是由不规范的操作导致的，比如直接用DML语句操作系统权限表。我们来看一下下面这个场景：

	client A	client B
T1	connect(root, root) create user 'ua'@'%' identified by 'pa';	
T2		connect(ua,pa) (connect ok) disconnect
T3	delete from mysql.user where user='ua';	
T4		connect(ua,pa) (connect ok) disconnect
T5	flush privileges;	
T6		connect(ua,pa) (Access Denied)

图4 使用flush privileges

可以看到，T3时刻虽然已经用delete语句删除了用户ua，但是在T4时刻，仍然可以用ua连接成功。原因就是，这时候内存中acl_users数组中还有这个用户，因此系统判断时认为用户还正常存在。

在T5时刻执行过flush命令后，内存更新，T6时刻再要用ua来登录的话，就会报错“无法访问”了。

直接操作系统表是不规范的操作，这个不一致状态也会导致一些更“诡异”的现象发生。比如，前面这个通过delete语句删除用户的例子，就会出现下面的情况：

	client A
T1	connect(root, root) create user 'ua'@'%' identified by 'pa';
T2	
T3	delete from mysql.user where user='ua';
T4	grant super on *.* to 'ua'@'%' with grant option; ERROR 1133 (42000): Can't find any matching row in the user table
T5	create user 'ua'@'%' identified by 'pa'; ERROR 1396 (HY000): Operation CREATE USER failed for 'ua'@'%'

图5 不规范权限操作导致的异常

可以看到，由于在T3时刻直接删除了数据表的记录，而内存的数据还存在。这就导致了：

1. T4时刻给用户ua赋权限失败，因为mysql.user表中找不到这行记录；
2. 而T5时刻要重新创建这个用户也不行，因为在做内存判断的时候，会认为这个用户还存在。

小结

今天这篇文章，我和你介绍了MySQL用户权限在数据表和内存中的存在形式，以及grant和revoke命令的执行逻辑。

grant语句会同时修改数据表和内存，判断权限的时候使用的是内存数据。因此，规范地使用grant和revoke语句，是不需要随后加上flush privileges语句的。

flush privileges语句本身会用数据表的数据重建一份内存权限数据，所以在权限数据可能存在不一致的情况下再使用。而这种不一致往往是由于直接用DML语句操作系统权限表导致的，所以我们尽量不要使用这类语句。

另外，在使用grant语句赋权时，你可能还会看到这样的写法：

```
grant super on *.* to 'ua'@'%' identified by 'pa';
```

这条命令加了identified by ‘密码’， 语句的逻辑里面除了赋权外，还包含了：

1. 如果用户’ua’@’%'不存在，就创建这个用户，密码是pa；
2. 如果用户ua已经存在，就将密码修改成pa。

这也是一种不建议的写法，因为这种写法很容易就会不慎把密码给改了。

“grant之后随手加flush privileges”，我自己是这么使用了两三年之后，在看代码的时候才发现其实并不需要这样做，那已经是2011年的事情了。

去年我看到一位小伙伴这么操作的时候，指出这个问题时，他也觉得很神奇。因为，他和我一样看的第一份文档就是这么写的，自己也一直是这么用的。

所以，今天的课后问题是，请你也来说一说，在使用数据库或者写代码的过程中，有没有遇到过类似的场景：误用了很长时间以后，由于一个契机发现“啊，原来我错了这么久”？

你可以把你的经历写在留言区，我会在下一篇文章的末尾选取有趣的评论和你分享。感谢你的收听，也欢迎你把这篇文章分享给更多的朋友一起阅读。

上期问题时间

上期的问题是，MySQL解析statement格式的binlog的时候，对于load data命令，解析出来为什么用的是load data local。

这样做的一个原因是，为了确保备库应用binlog正常。因为备库可能配置了secure_file_priv=null，所以如果不用local的话，可能会导入失败，造成主备同步延迟。

另一种应用场景是使用mysqlbinlog工具解析binlog文件，并应用到目标库的情况。你可以使用下面这条命令：

```
mysqlbinlog $binlog_file | mysql -h$host -P$port -u$user -p$pwd
```

把日志直接解析出来发给目标库执行。增加local，就能让这个方法支持非本地的$host。

评论区留言点赞板：

> @poppy、@库淘淘 两位同学提到了第一个场景；
> @王显伟 @lionetes 两位同学帮忙回答了 @undifined 同学的疑问，拷贝出来的文件要确保MySQL进程可以读。

精选留言

undifined 7

权限的作用范围和修改策略总结：

http://ww1.sinaimg.cn/large/d1885ed1ly1g0ab2twmjaj21gs0js78u.jpg

2019-02-18

作者回复

，优秀

2019-02-18

夜空中最亮的星（华仔） 3

通过老师的讲解 flush privileges 这回彻底懂了，高兴

2019-02-18

作者回复

2019-02-19

way 1

写个比较小的点：在命令行查询数据需要行转列的时候习惯加个\G；比如slave slave stauts \G；后来发现；是多余的。列几个常用的

\G 行转列并发送给 mysql server

\g 等同于；

\! 执行系统命令

\q exit
\c 清除当前SQL（不执行）
\s mysql status 信息
其他参考 \h

2019-02-20

作者回复

[]

我最开始使用MySQL的时候，就是不自然的在\G后面加分号
而且还看到报错，好紧张[]

2019-02-20

XD 1

老师，我刚说的是acl_db，是在db切换的时候，从acl_dbs拷贝到线程内部的？类似acl_user。

session a
drop user 'test'@'%';
create user 'test'@'%' identified by '123456';
grant SELECT,UPDATE on gt.* to 'test'@'%';

session b 使用test登录
use gt;

session a
revoke SELECT,UPDATE on gt.* from 'test'@'%';

session b
show databases; //只能看到information_schema库
use gt; // Access denied for user 'test'@'%' to database 'gt'
show tables; //可以看到gt库中所有的表
select/update //操作都正常

2019-02-18

作者回复

你说的对，我刚翻代码确认了下，确实是特别对“当前db”有一个放过的逻辑。

多谢指正。我勘误下。

2019-02-19

夹心面包 1

我在此分享一个授权库的小技巧, 如果需要授权多个库,库名还有规律,比如 db_201701 db_201702
可以采用正则匹配写一条 grant on db_____,每一个_代表一个字符.这样避免了多次授权,简化了过程。我们线上已经采用

2019-02-18

作者回复

是的，MySQL还支持 % 赋权，%表示匹配任意字符串，

比如

grant all privileges on `db%`.* to ... 表示所有以db为前缀的库。

不过。。。我比较不建议这么用

2019-02-19

萤火虫 0

坚持到最后 为老师打call

2019-02-20

作者回复

是真爱

2019-02-20

wljs 0

老师我想问个问题 我们公司一个订单表有110个字段 想拆分成两个表 第一个表放经常查的字段 第二个表放不常查的 现在程序端不想改sql，数据库端来实现 当查询字段中 第一个表不存在 就去关联第二个表查出数据 db能实现不？

2019-02-19

舜 0

老师，介绍完了order by后能不能继续介绍下group by的原理？等了好久了，一直想继续在order by基础上理解下group by，在使用过程中两者在索引利用上很相近，性能考虑也类似

2019-02-19

作者回复

37篇讲了group by的，你看下

还有问再提出来

2019-02-19

旭东 0

老师请教一个问题：MySQL 表设计时列表顺序对MySQL性能的影响大吗？对表的列顺序有什么建议吗？

2019-02-18

作者回复

没有影响

建议就是每次如果要加列都加到最后一列

2019-02-19

XD

0

老师，实际测试了下。
两个会话ab，登陆账号都为user。a中给user授予db1的select、update权限，b切换到db1，可以正常增改。然后a中回收该用户的db权限，b会话中的用户还是可以进行增改操作的。
我发现用户的db权限好像是在切换数据库的时候刷新的，只要不切换，grant操作并不会产生作用，所以acl_db是否也是维护在线程内部的呢？

以及，权限检验应该是在优化器的语义分析里进行的吧？

2019-02-18

作者回复

acl_dbs是全局数组

把你使用sql语句，和语句序列发一下哦

类似按照时间顺序

session a：

xxx

xxx

session b:

xxxx

session a:

xxxx

这样

2019-02-18

发芽的紫菜

0

老师，联合索引的数据结构是怎么样的？到底是怎么存的？看了前面索引两章，还是不太懂，留言里老师说会在后面章节会讲到，但我也没看到，所以来此问一下？老师能否画图讲解一下

2019-02-18

作者回复

联合索引就是两个字段拼起来作索引

比如一个索引如果定义为(f1,f2),

在数据上，就是f1的值之后跟着f2的值。

查找的时候，比如执行 where f1=M and f2=N, 也是把M,N拼起来，去索引树查找

2019-02-18

晨思暮语

0

丁老师,您好：
关于上一章我留言的疑问,我重新整理了下。就是第十五章中老师留的思考题。
我模拟了老师的实验,结果有点出入,请老师帮忙看看，谢谢！

基础环境:

```
mysql> select version();
+------------+
| version() |
+------------+
| 5.7.22-log |
+------------+
1 row in set (0.00 sec)

mysql> show variables like '%tx%';
+---------------+----------------+
| Variable_name | Value |
+---------------+----------------+
| tx_isolation | REPEATABLE-READ |
| tx_read_only | OFF |
+---------------+----------------+
2 rows in set (0.00 sec)
```

模拟实验:

session A:

```
mysql> begin;
mysql> select * from t;
+----+------+
| id | a |
+----+------+
| 1 | 2 |
+----+------+
1 row in set (0.00 sec)
```

session B:

```
mysql> update t set a=3 where id=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

SESSION A:

```
mysql> update t set a=3 where id=1;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1 Changed: 0 Warnings: 0
/*老师的实验显示为：1 rows affected*/
mysql> select * from t where id=1;
+----+------+
| id | a |
+----+------+
| 1 | 2 |
```

+----+------+
1 row in set (0.00 sec)
/*老师实验的查询结果为：1,3 */

2019-02-18

作者回复

这个跟binlog_format有关。

如果binlog_format=row, 那么最后session A的select查到的是2；
如果binlog_format=statement, 那么最后session A的select查到的是3；

我们在文章里面有做了说明了，这个逻辑是依赖于“MySQL在执行update语句的时候，有没有把字段c也读进来”，

2019-02-26

Sinyo 0

查一张大表，order_key字段值对应的最小createtime；
以前一直用方法一查数，后来同事说可以优化成方法二，查询效率比方法一高了几倍；
mysql特有的group by功能，没有group by的字段默认取查到的第一条记录；

方法一：

```
select distinct order_key
,createtime
from (select order_key
,min(createtime) createtime
from aaa
group by order_key) a
join aaa b
on a.order_key = b.order_key
and a.createtime = b.createtime
```

方法二：

```
select order_key
,createtime
from (select order_key
,createtime
FROM aaa
order by createtime
) a
group by order_key
```

2019-02-18

作者回复

[]

这第二个写法跟：

select order_key ,createtime FROM aaa force index(createtime) group by order_key

的逻辑语义相同吗？

2019-02-18

Leon

0

老师我使用delte删除用户，再创建用户都是失败，但是使用drop就可以了

```
mysql> create user 'ua'@'%' identified by 'L1234567890c-';
ERROR 1396 (HY000): Operation CREATE USER failed for 'ua'@'%'
mysql> drop user 'ua'@'%';
Query OK, 0 rows affected (0.00 sec)

mysql> create user 'ua'@'%' identified by 'L1234567890c-';
Query OK, 0 rows affected (0.01 sec)
```

是不是drop才会同时从内存和磁盘删除用户信息，但是delete只是从磁盘删除

2019-02-18

作者回复

对，drop是同时操作磁盘和内存，

delete就是我们说的不规范操作

2019-02-18

爸爸回来了

0

众所周知，sql是不区分大小写的。然而，涉及插件的变量却不是这样；上次在配置一个插件的参数的时候，苦思良久.....最后发现了这个问题。难受

2019-02-18

作者回复

你说的是参数的名字，还是参数的值？

2019-02-18