

实验4 SQL安全性

3200105787 张云策

一、实验目的

1. 熟悉通过SQL进行数据完整性控制的方法

实验内容和要求：

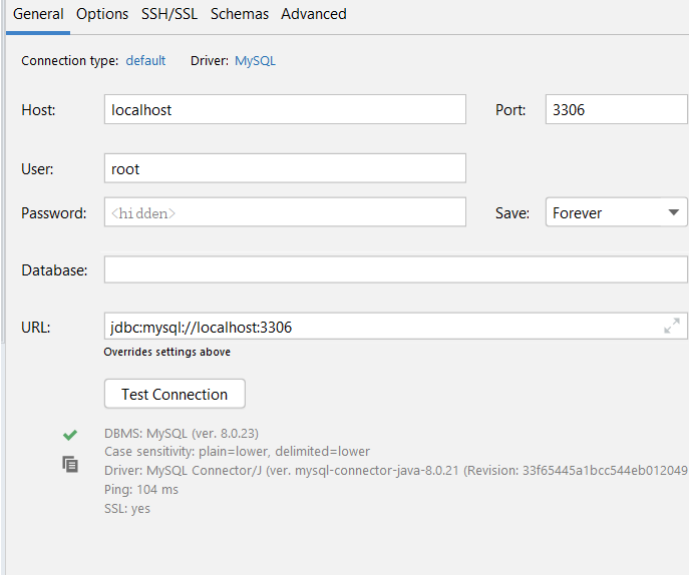
1. 建立表，考察表的生成者拥有该表的哪些权限。
2. 使用SQL的grant 和revoke命令对其他用户进行授权和权力回收，考察相应的作用。
3. 建立视图，并把该视图的查询权限授予其他用户，考察通过视图进行权限控制的作用。
4. 完成实验报告。

二、实验环境

数据库管理系统：MySQL

三、实验流程

首先用root用户登录MySQL



The screenshot shows the 'General' tab of the MySQL Connector/J configuration window. The 'Connection type' is set to 'default' and the 'Driver' is 'MySQL'. The 'Host' is 'localhost', 'Port' is '3306', 'User' is 'root', and 'Password' is masked as '<hidden>'. The 'Save' option is set to 'Forever'. The 'Database' field is empty. The 'URL' field contains 'jdbc:mysql://localhost:3306'. Below the URL, it states 'Overrides settings above'. A 'Test Connection' button is present. At the bottom, a green checkmark indicates a successful connection with the following details: 'DBMS: MySQL (ver. 8.0.23)', 'Case sensitivity: plain=lower, delimited=lower', 'Driver: MySQL Connector/J (ver. mysql-connector-java-8.0.21 (Revision: 33f65445a1bcc544eb012049))', 'Ping: 104 ms', and 'SSL: yes'.

root用户即超级管理员用户，拥有数据库的全部权限，而普通用户，由root创建，普通用户只拥有root所分配的权限。

新建用户的语法如下

```
1 | CREATE USER <user_name>@<host_name> identified BY <password>;
```

```
1 | create user 'zyc'@'localhost' IDENTIFIED BY 'zyc666';
```

```
[2022-03-29 16:42:13] Connected
```

```
lab4> use lab4
```

```
[2022-03-29 16:42:13] completed in 4 ms
```

```
lab4> create user 'zyc'@'localhost' IDENTIFIED BY 'zyc666'
```

现在我们使用新创建的用户进行登录

之后我们尝试查询由zyc用户创建的lab3数据库中的某表

```
1 | select * from lab3.TicketWorkers;
```

```
information_schema> select * from lab4.TicketWorkers
```

```
2022-03-29 16:52:39] [42000][1142] SELECT command denied to user 'zyc'@'localhost' for table 'ticketworkers'
```

- **建立表，考察表的生成者拥有该表的哪些权限**

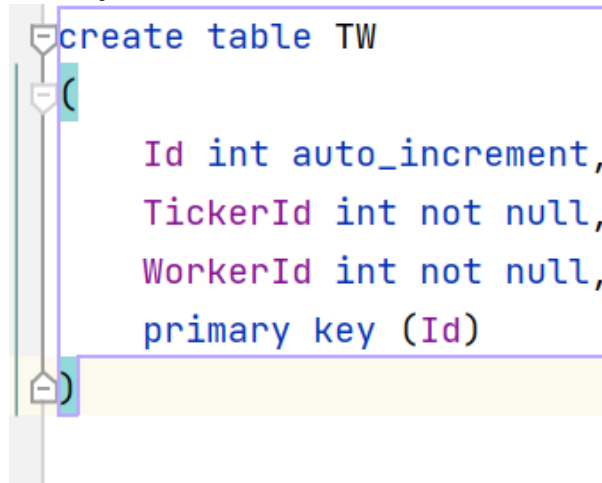
首先使用root用户授予zyc用户建立表的权限

```
1 | GRANT create ON * TO 'zyc'@'localhost';
```

```
[2022-03-29 17:03:32] completed in 3 ms
```

```
information_schema> GRANT create ON * TO 'zyc'@'localhost'
```

我们用zyc用户建立表



```
create table TW
(
    Id int auto_increment,
    TickerId int not null,
    WorkerId int not null,
    primary key (Id)
)
```

```
[2022-03-29 18:51:08] completed in 4 ms
```

```
information_schema> create table TW
(
    Id int auto_increment,
    TickerId int not null,
    WorkerId int not null,
    primary key (Id)
)
```

可以看到zyc001用户在tw表上有create权限，没有其他的select、update、delete权限，之后我们也可以通过基本相同授权命令利用root用户授予zyc用户其他权限。

- **使用SQL的grant 和revoke命令对其他用户进行授权和权力回收，考察相应的作用。**

我们回到root用户，并且为zyc用户进行授权

我们使用show grants命令查看该用户

```
1 | show grants for 'zyc'@'localhost';
```

Grants for zyc@localhost	
1	GRANT USAGE ON *.* TO `zyc`@`localhost`

可以看到目前是没有任何权限的

授权的命令为

```
1 | grant [权限] on [数据库名].[表名] to '[用户名]'@'[作用域]' identified by '[密码]';
```

其中权限为ALL PRIVILEGES或ALL是所有权限，还有单个权限select、update、insert、delete等，单个权限之间用逗号隔开，详细可以查看下mysql.user表的表结构。

之后我们授予zyc用户lab3.TicketWorkers表的select权限

1	GRANT select ON lab3.TicketWorkers TO 'zyc'@'localhost';
---	--

Output	Result 4
information_schema>	GRANT select ON lab3.TicketWorkers TO 'zyc'@'localhost'
	[2022-03-29 19:29:41] completed in 7 ms

现在我们看下zyc拥有的权限

1	show grants for 'zyc'@'localhost';
---	------------------------------------

1	show grants for 'zyc'@'localhost';
---	------------------------------------

Output	Result 8
	Grants for zyc@localhost
	1 GRANT USAGE ON *.* TO `zyc`@`localhost`
	2 GRANT SELECT ON `lab3`.`ticketworkers` TO `zyc`@`localhost`

查看一下哪些用户有哪些在表TicketWorkers上的权限：

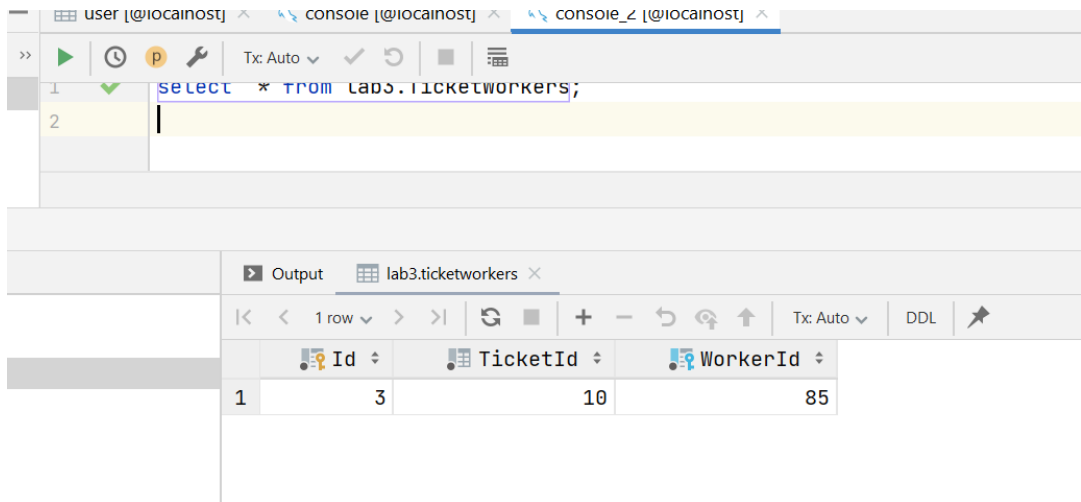
1	select * from mysql.tables_priv where table_name='ticketworkers';
---	---

1	select * from mysql.tables_priv where table_name='ticketworkers';
---	---

Host	Db	User	Table_name	Grantor	Timestamp	Table_priv
localhost	lab3	zyc	ticketworkers	root@localhost	0000-00-00 00:00:00	Select

回到zyc用户，查询下刚才被授予select权限的表

```
1 | select * from lab3.TicketWorkers;
```



收回权限

```
1 | REVOKE priv_type ON [object_type] FROM user
```

回到root身份后，将表的select权限收回

```
localhost on table 'TicketWorkers'
lab3> REVOKE select ON TicketWorkers FROM 'zyc'@'localhost'
[2022-03-29 19:33:32] completed in 6 ms
```

之后用zyc查询发现访问被拒，证明授权、回收权力方法正确

- 建立视图，并把该视图的查询权限授予其他用户，考察通过视图进行权限控制的作用。

我们首先对users表建立一个视图

```
1 | create view user_view
2 | as select Id, StudentId, Name, Role
3 | from users
4 | where Role>0;
```

```
lab3> create view user_view
      as select Id, StudentId, Name, Role
      from user
      where Role>0
```

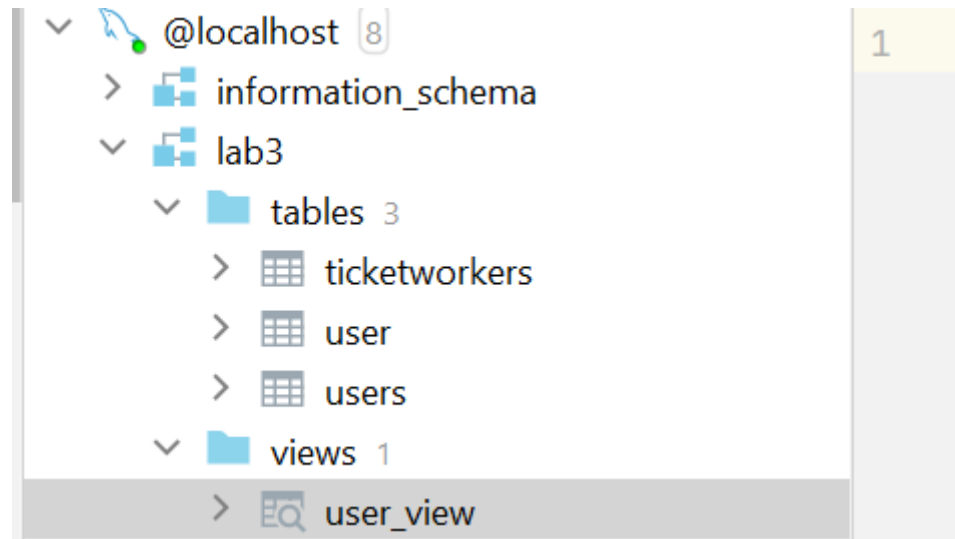
	Id	StudentId	Name	Secret	Role	Department	ComputerFixedCount	Ap
1	85	3200105787	zyc	6ACFEA919AFD1918C1D23	1	2	110	
2	43232	3200105847	zwq	7D774C7907EA58579D423	1	2	110	

可以看到已经生成了视图，现在我们把视图的查询权限交给zyc

```
1 | GRANT select ON user_view TO 'zyc'@'localhost';
```

```
lab3> GRANT select ON user_view TO 'zyc'@'localhost'
[2022-03-29 19:37:38] completed in 9 ms
```

我们切换到zyc用户进行查询



在左边我们看到view视图模式中可以看到存在users_view视图，我们尝试对该视图进行查询操作

```
1 | select * from user_view;
```

Output lab3.user_view				
	Id	StudentId	Name	Role
1	85	3200105787	zyc	1
2	43232	3200105847	zwq	1

我们尝试使用update方法进行更新

发现我们并没有进行update的权限，因此更新失败。

我们切换到root用户并赋予update的权限再次进行尝试

```
1 | update user_view
2 | set Name = 'skl'
3 | where Name='zwq'
```

Output lab3.user_view				
	Id	StudentId	Name	Role
1	85	3200105787	zyc	1
2	43232	3200105847	skl	1

可以看到查询成功，证明视图来进行权限控制有效！

四、遇到的问题及解决方法

- 本次实验中基本没有遇到较大的问题，主要是一些授权的语法使用上还不算完全清楚，在搞明白授权语法后所进行实验的过程中还算很顺利，因此本部分此次内容较少

五、总结

在本次报告中，我亲自实践了利用SQL进行数据完整性控制的步骤，也将理论中的各种高级权限控制用法在mysql上亲自实现出来，在之前的日常使用中基本没有过多的了解到mysql比较精细的权限控制，这次也大大拓展了我的眼界，切实的提高了自己对sql以及DBMS的理解和应用水平。