实验报告	
题目	实验五 安全性和完整性实验

实验环境（计算机配置，操作系统，RDBMS 版本等）

CPU：12th Gen Intel(R) Core(TM) i7-12700H，2700Mhz，14 个内核，20 个逻辑处理器

内存：32G（DDR5-4800）

操作系统：Windows 11

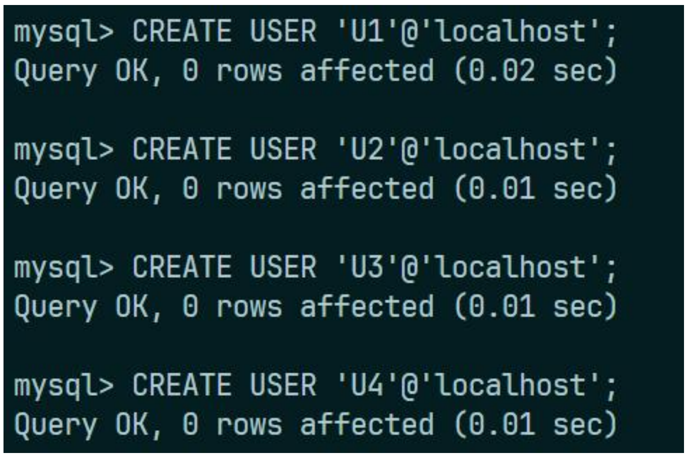
RDBMS：MySQL 8.0.39 for Win64 on x86_64

实验步骤（SQL 语句）和运行效果截图

1. 安全性实验

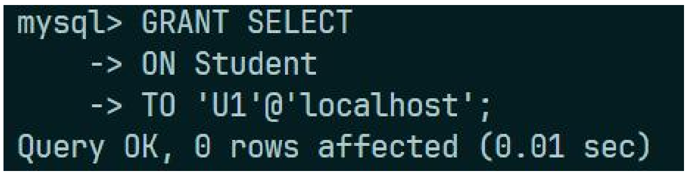
1.1 在 DBMS 上尝试教科书第四章例子。建立多个用户，在权限发生变化时，尝试登录，观察结果。

- 首先，创建 4 个实验用户 U1-U4



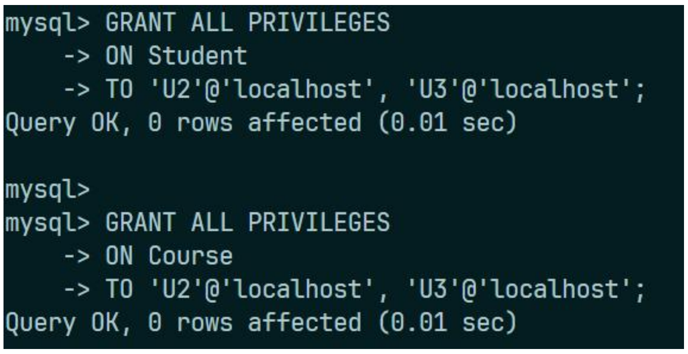
```
mysql> CREATE USER 'U1'@'localhost';
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE USER 'U2'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE USER 'U3'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE USER 'U4'@'localhost';
Query OK, 0 rows affected (0.01 sec)
```

- （教材例 4.1）把查询 Student 表的权限授给用户 U1



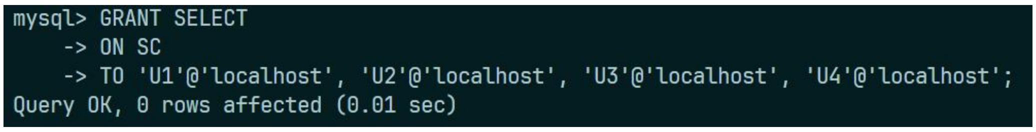
```
mysql> GRANT SELECT
    -> ON Student
    -> TO 'U1'@'localhost';
Query OK, 0 rows affected (0.01 sec)
```

- （教材例 4.2）把对 Student 表和 Course 表的全部操作权限授予用户 U2 和 U3



```
mysql> GRANT ALL PRIVILEGES
    -> ON Student
    -> TO 'U2'@'localhost', 'U3'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> GRANT ALL PRIVILEGES
    -> ON Course
    -> TO 'U2'@'localhost', 'U3'@'localhost';
Query OK, 0 rows affected (0.01 sec)
```

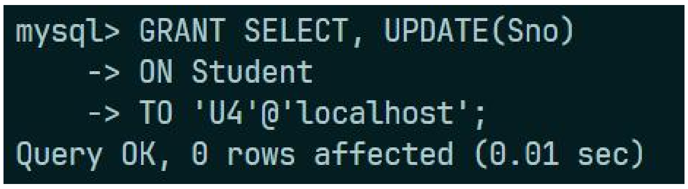
- （教材例 4.3）把对表 SC 的查询权限授予所有用户



```
mysql> GRANT SELECT
    -> ON SC
    -> TO 'U1'@'localhost', 'U2'@'localhost', 'U3'@'localhost', 'U4'@'localhost';
Query OK, 0 rows affected (0.01 sec)
```

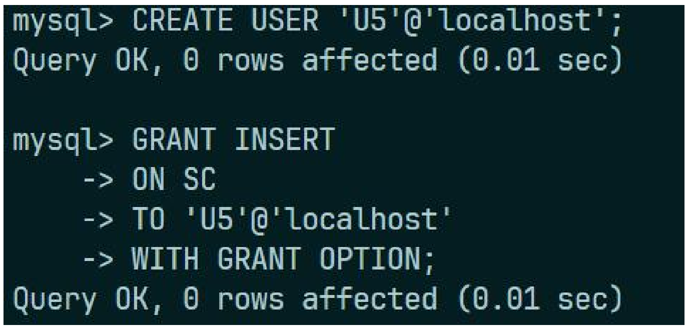
> MySQL 不支持 PUBLIC 关键字，所以此处手动列出所有用户来授权。

- （教材例 4.4）把查询 Student 表和修改学生学号的权限授予用户 U4



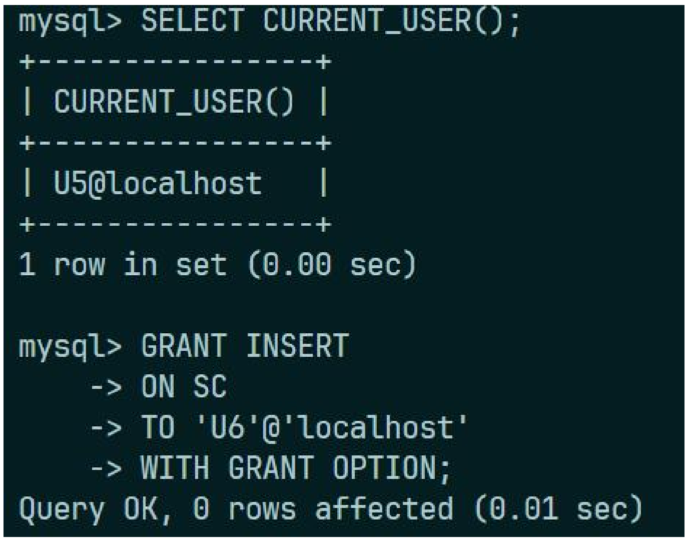
```
mysql> GRANT SELECT, UPDATE(Sno)
    -> ON Student
    -> TO 'U4'@'localhost';
Query OK, 0 rows affected (0.01 sec)
```

- （教材例 4.5）把对表 SC 的 INSERT 权限授予 U5 用户，并允许将此授权再授予其他用户



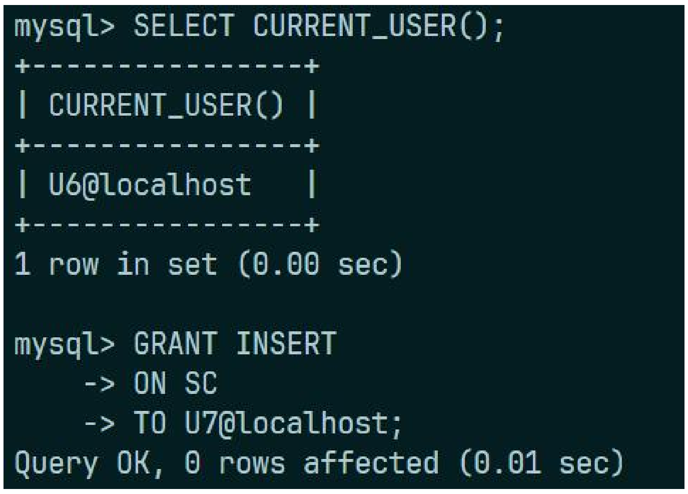
```
mysql> CREATE USER 'U5'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT INSERT
    -> ON SC
    -> TO 'U5'@'localhost'
    -> WITH GRANT OPTION;
Query OK, 0 rows affected (0.01 sec)
```

- （教材例 4.6）用户 U5 将对表 SC 的 INSERT 权限授予用户 U6（允许再授权）



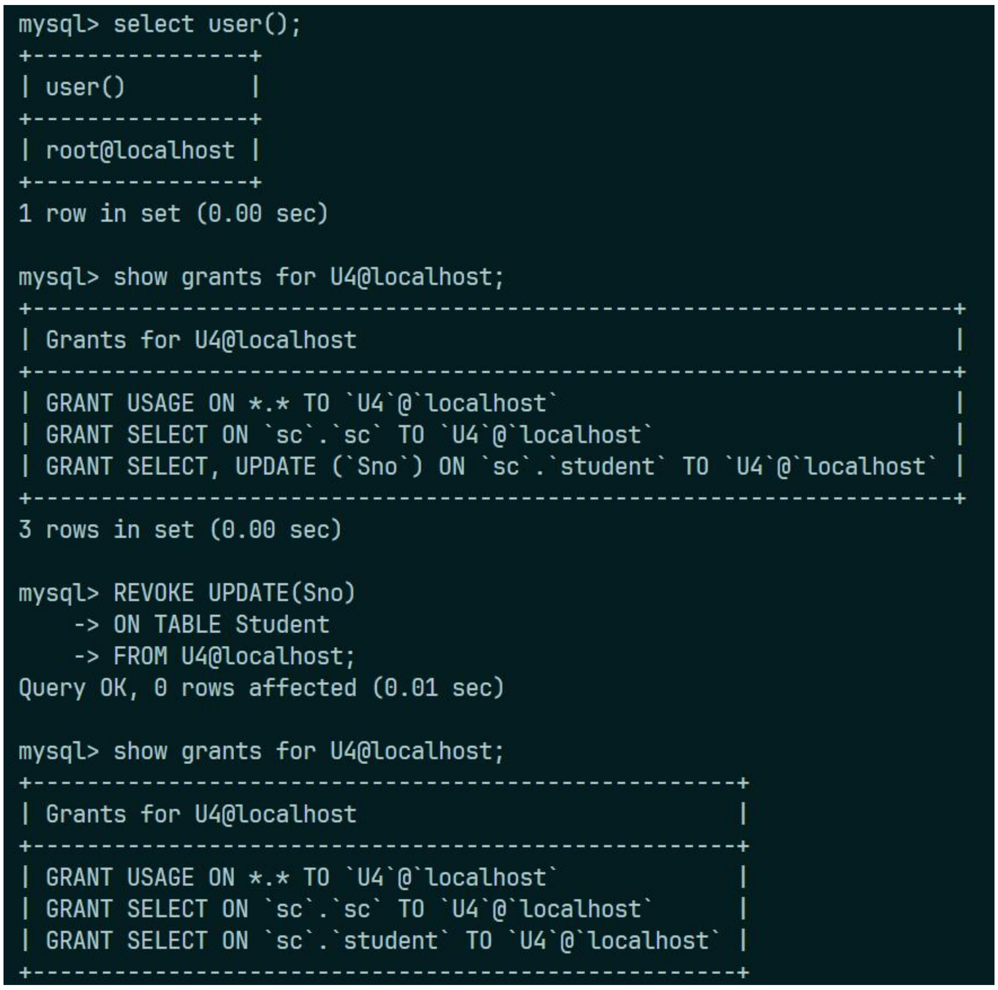
```
mysql> SELECT CURRENT_USER();
+----------------+
| CURRENT_USER() |
+----------------+
| U5@localhost   |
+----------------+
1 row in set (0.00 sec)

mysql> GRANT INSERT
    -> ON SC
    -> TO 'U6'@'localhost'
    -> WITH GRANT OPTION;
Query OK, 0 rows affected (0.01 sec)
```

同样，U6 还可以将此权限继续授予 U7



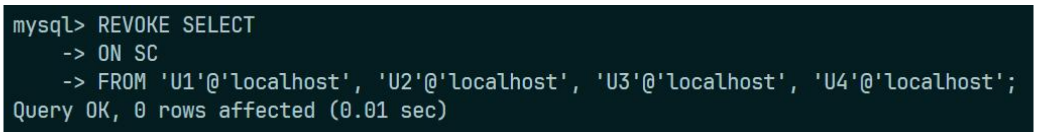
```
mysql> SELECT CURRENT_USER();
+----------------+
| CURRENT_USER() |
+----------------+
| U6@localhost   |
+----------------+
1 row in set (0.00 sec)

mysql> GRANT INSERT
    -> ON SC
    -> TO U7@localhost;
Query OK, 0 rows affected (0.01 sec)
```

- （教材例 4.8）把用户 U4 修改学生学号的权限收回



```
mysql> select user();
+----------------+
| user()         |
+----------------+
| root@localhost |
+----------------+
1 row in set (0.00 sec)

mysql> show grants for U4@localhost;
+----------------------------------------------------------------------+
| Grants for U4@localhost                                              |
+----------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `U4`@`localhost`                               |
| GRANT SELECT ON `sc`.`sc` TO `U4`@`localhost`                        |
| GRANT SELECT, UPDATE (`Sno`) ON `sc`.`student` TO `U4`@`localhost`  |
+----------------------------------------------------------------------+
3 rows in set (0.00 sec)

mysql> REVOKE UPDATE(Sno)
    -> ON TABLE Student
    -> FROM U4@localhost;
Query OK, 0 rows affected (0.01 sec)

mysql> show grants for U4@localhost;
+----------------------------------------------------+
| Grants for U4@localhost                            |
+----------------------------------------------------+
| GRANT USAGE ON *.* TO `U4`@`localhost`             |
| GRANT SELECT ON `sc`.`sc` TO `U4`@`localhost`      |
| GRANT SELECT ON `sc`.`student` TO `U4`@`localhost` |
+----------------------------------------------------+
```

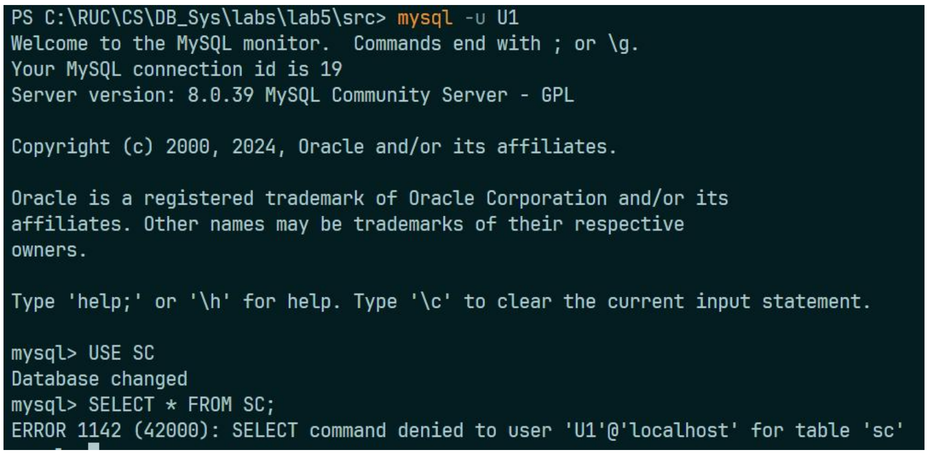
- （教材例 4.9）收回所有用户对表 SC 的查询权限



```
mysql> REVOKE SELECT
    -> ON SC
    -> FROM 'U1'@'localhost', 'U2'@'localhost', 'U3'@'localhost', 'U4'@'localhost';
Query OK, 0 rows affected (0.01 sec)
```

> MySQL 不支持 PUBLIC 关键字，所以此处手动列出所有有查询权限的用户收回。

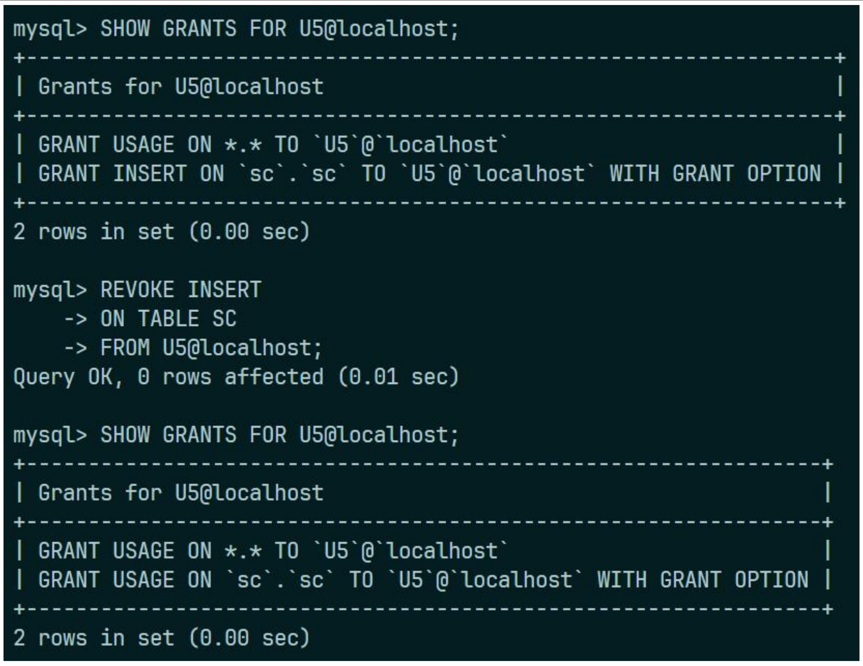
收回权限后，登录 U1，尝试查询表 SC：



```
PS C:\RUC\CS\DB_Sys\labs\lab5\src> mysql -u U1
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 8.0.39 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE SC
Database changed
mysql> SELECT * FROM SC;
ERROR 1142 (42000): SELECT command denied to user 'U1'@'localhost' for table 'sc'
```

可以看到，查询被拒绝了。

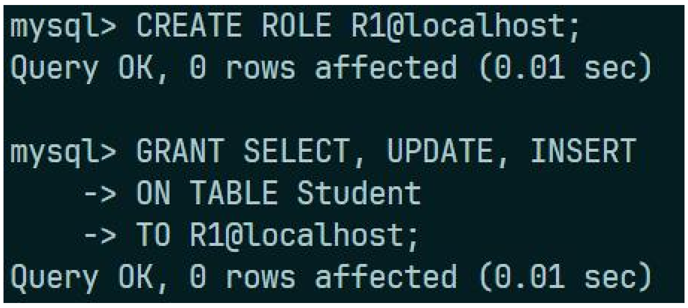
- （教材例 4.10）把用户 U5 对 SC 表的 INSERT 权限收回



```
mysql> SHOW GRANTS FOR U5@localhost;
+-----------------------------------------------------------------+
| Grants for U5@localhost                                         |
+-----------------------------------------------------------------+
| GRANT USAGE ON *.* TO `U5`@`localhost`                          |
| GRANT INSERT ON `sc`.`sc` TO `U5`@`localhost` WITH GRANT OPTION |
+-----------------------------------------------------------------+
2 rows in set (0.00 sec)

mysql> REVOKE INSERT
    -> ON TABLE SC
    -> FROM U5@localhost;
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW GRANTS FOR U5@localhost;
+----------------------------------------------------------------+
| Grants for U5@localhost                                        |
+----------------------------------------------------------------+
| GRANT USAGE ON *.* TO `U5`@`localhost`                         |
| GRANT USAGE ON `sc`.`sc` TO `U5`@`localhost` WITH GRANT OPTION |
+----------------------------------------------------------------+
2 rows in set (0.00 sec)
```

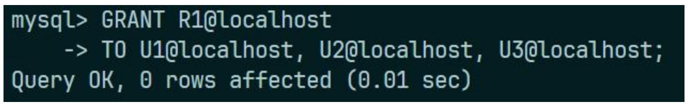
- （教材例 4.14）通过角色来实现将一组权限授予用户

  创建角色 R1，使 R1 拥有 Student 表的 SELECT, UPDATE, INSERT 权限



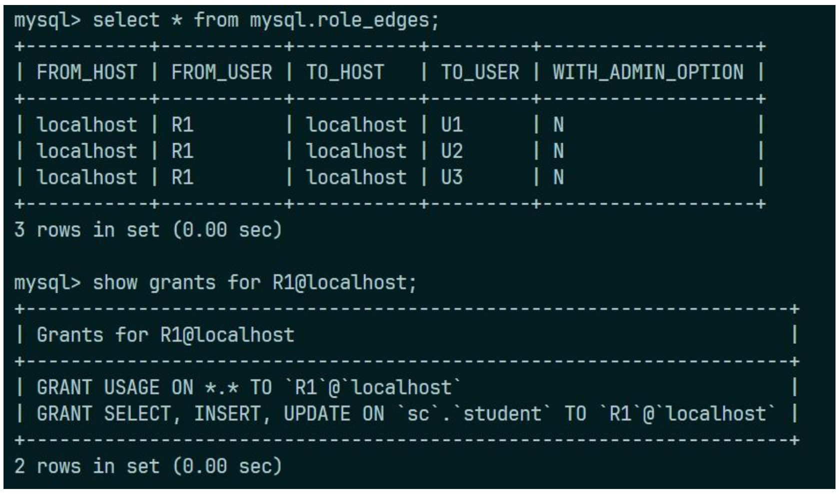
```
mysql> CREATE ROLE R1@localhost;
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT SELECT, UPDATE, INSERT
    -> ON TABLE Student
    -> TO R1@localhost;
Query OK, 0 rows affected (0.01 sec)
```

  将角色 R1 授予 U1、U2、U3



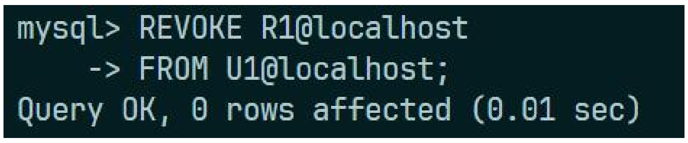
```
mysql> GRANT R1@localhost
    -> TO U1@localhost, U2@localhost, U3@localhost;
Query OK, 0 rows affected (0.01 sec)
```

  检查授权结果



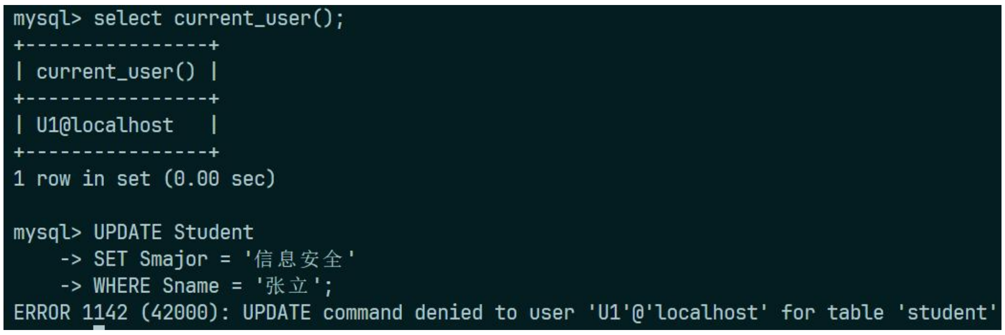
```
mysql> select * from mysql.role_edges;
+-----------+-----------+-----------+---------+-------------------+
| FROM_HOST | FROM_USER | TO_HOST   | TO_USER | WITH_ADMIN_OPTION |
+-----------+-----------+-----------+---------+-------------------+
| localhost | R1        | localhost | U1      | N                 |
| localhost | R1        | localhost | U2      | N                 |
| localhost | R1        | localhost | U3      | N                 |
+-----------+-----------+-----------+---------+-------------------+
3 rows in set (0.00 sec)

mysql> show grants for R1@localhost;
+-------------------------------------------------------------------+
| Grants for R1@localhost                                           |
+-------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `R1`@`localhost`                            |
| GRANT SELECT, INSERT, UPDATE ON `sc`.`student` TO `R1`@`localhost` |
+-------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

  一次性通过 R1 来收回 U1 的这三个权限：



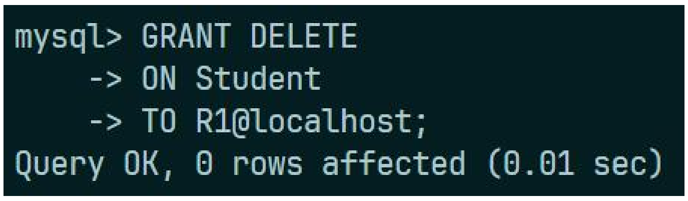
```
mysql> REVOKE R1@localhost
    -> FROM U1@localhost;
Query OK, 0 rows affected (0.01 sec)
```

登录 U1，发现确实不能修改 Student 表了：



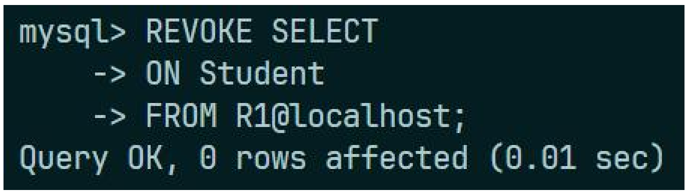
```
mysql> select current_user();
+----------------+
| current_user() |
+----------------+
| U1@localhost   |
+----------------+
1 row in set (0.00 sec)

mysql> UPDATE Student
    -> SET Smajor = '信息安全'
    -> WHERE Sname = '张立';
ERROR 1142 (42000): UPDATE command denied to user 'U1'@'localhost' for table 'student'
```

- （教材例 4.15）给角色 R1 添加新的权限



```
mysql> GRANT DELETE
    -> ON Student
    -> TO R1@localhost;
Query OK, 0 rows affected (0.01 sec)
```

- （教材例 4.16）减少角色 R1 的权限

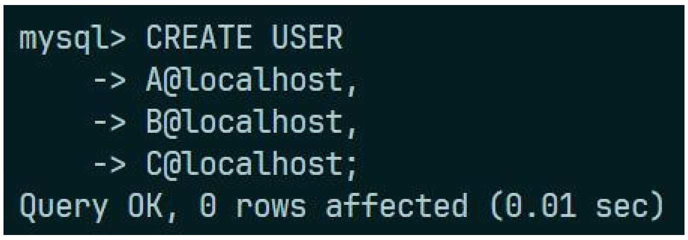


```
mysql> REVOKE SELECT
    -> ON Student
    -> FROM R1@localhost;
Query OK, 0 rows affected (0.01 sec)
```

1.2 在 DBMS 上尝试下面的实验，并分析原因：

A、B、C 为用户，x 为在 Student 表上的 SELECT 权限

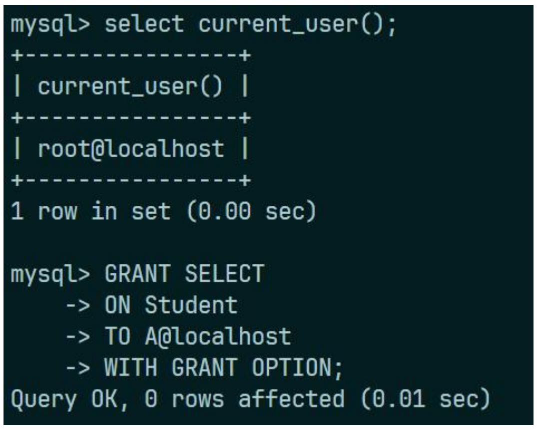
- 创建用户如下：



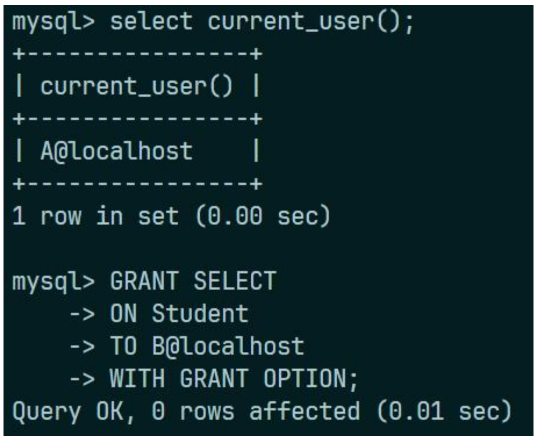
```
mysql> CREATE USER
    -> A@localhost,
    -> B@localhost,
    -> C@localhost;
Query OK, 0 rows affected (0.01 sec)
```

（1） *A--x-->B--x-->C，收回 B 的 x，C 是否还具有 x？ cascade 是否有效？*

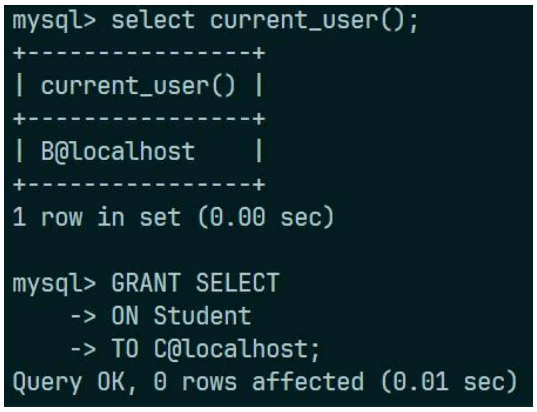
- 将权限 x 授予 A、B、C（root--x-->A--x-->B--x-->C)：



```
mysql> select current_user();
+----------------+
| current_user() |
+----------------+
| root@localhost |
+----------------+
1 row in set (0.00 sec)

mysql> GRANT SELECT
    -> ON Student
    -> TO A@localhost
    -> WITH GRANT OPTION;
Query OK, 0 rows affected (0.01 sec)
```



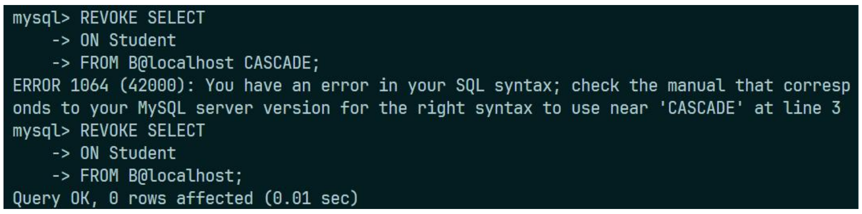
```
mysql> select current_user();
+----------------+
| current_user() |
+----------------+
| A@localhost    |
+----------------+
1 row in set (0.00 sec)

mysql> GRANT SELECT
    -> ON Student
    -> TO B@localhost
    -> WITH GRANT OPTION;
Query OK, 0 rows affected (0.01 sec)
```



```
mysql> select current_user();
+----------------+
| current_user() |
+----------------+
| B@localhost    |
+----------------+
1 row in set (0.00 sec)

mysql> GRANT SELECT
    -> ON Student
    -> TO C@localhost;
Query OK, 0 rows affected (0.01 sec)
```

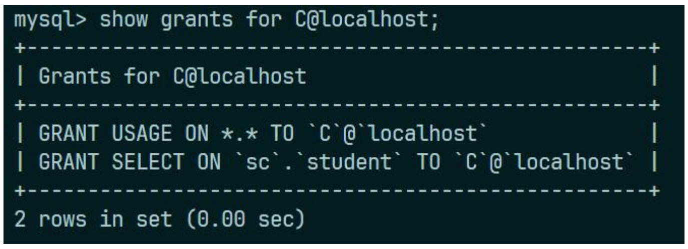
- 收回 B 的 x 权限（登录用户为 A）

由于 MySQL 不支持 CASCADE 关键字，只能单独收回 B 的 x 权限：



```
mysql> REVOKE SELECT
    -> ON Student
    -> FROM B@localhost CASCADE;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresp
onds to your MySQL server version for the right syntax to use near 'CASCADE' at line 3
mysql> REVOKE SELECT
    -> ON Student
    -> FROM B@localhost;
Query OK, 0 rows affected (0.01 sec)
```

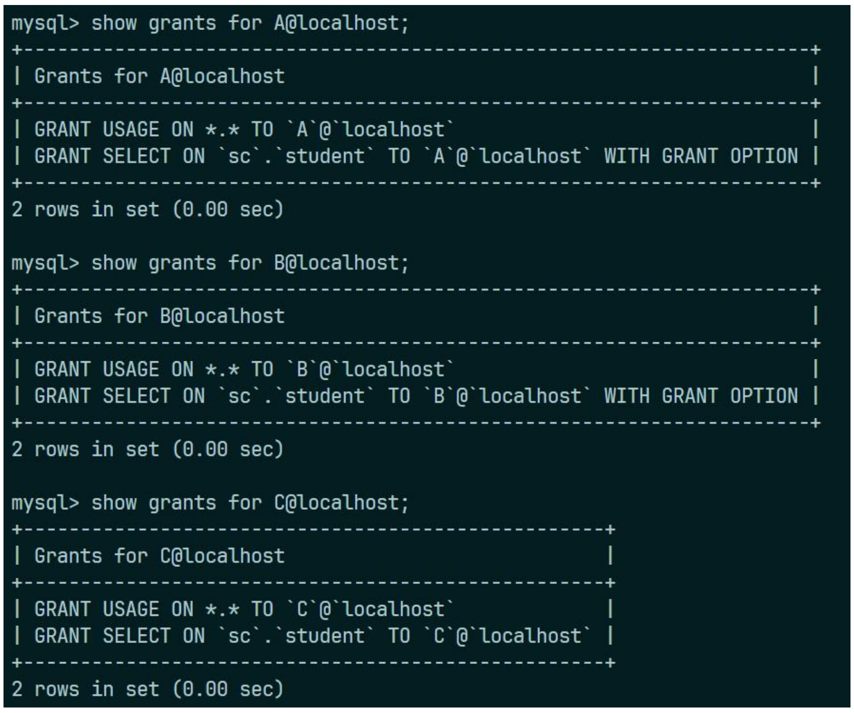
此时，C 的 x 权限仍然存在：



```
mysql> show grants for C@localhost;
+---------------------------------------------------+
| Grants for C@localhost                            |
+---------------------------------------------------+
| GRANT USAGE ON *.* TO `C`@`localhost`             |
| GRANT SELECT ON `sc`.`student` TO `C`@`localhost` |
+---------------------------------------------------+
2 rows in set (0.00 sec)
```

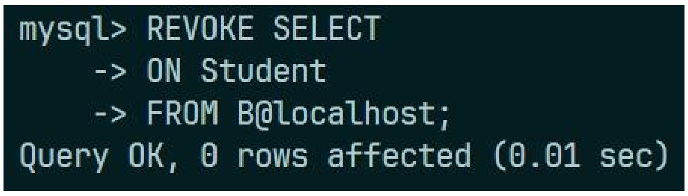
（2） *A--x-->B--x-->C，A--x-->C，收回 B 的 x，C 是否还有 x？*

- 同上，完成权限分配，结果如下：



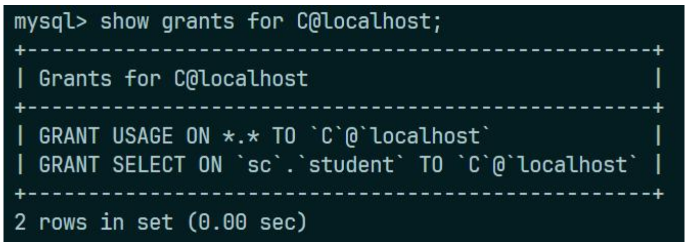
```
mysql> show grants for A@localhost;
+---------------------------------------------------------------------+
| Grants for A@localhost                                              |
+---------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `A`@`localhost`                               |
| GRANT SELECT ON `sc`.`student` TO `A`@`localhost` WITH GRANT OPTION |
+---------------------------------------------------------------------+
2 rows in set (0.00 sec)

mysql> show grants for B@localhost;
+---------------------------------------------------------------------+
| Grants for B@localhost                                              |
+---------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `B`@`localhost`                               |
| GRANT SELECT ON `sc`.`student` TO `B`@`localhost` WITH GRANT OPTION |
+---------------------------------------------------------------------+
2 rows in set (0.00 sec)

mysql> show grants for C@localhost;
+---------------------------------------------------+
| Grants for C@localhost                            |
+---------------------------------------------------+
| GRANT USAGE ON *.* TO `C`@`localhost`             |
| GRANT SELECT ON `sc`.`student` TO `C`@`localhost` |
+---------------------------------------------------+
2 rows in set (0.00 sec)
```

收回 B 的 x：



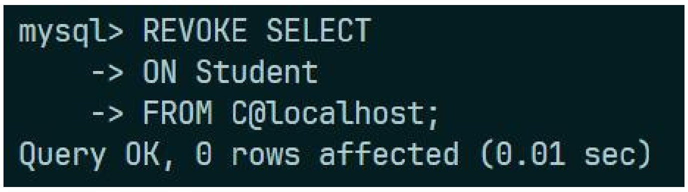
```
mysql> REVOKE SELECT
    -> ON Student
    -> FROM B@localhost;
Query OK, 0 rows affected (0.01 sec)
```

可以看到，C 仍然有权限 x：



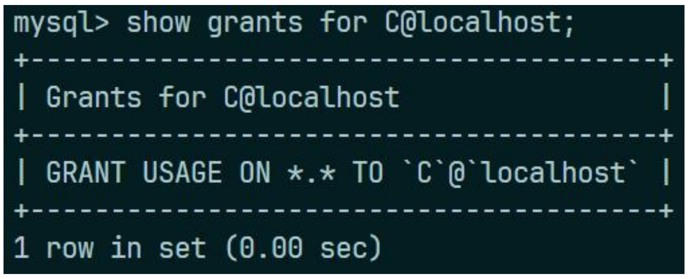
```
mysql> show grants for C@localhost;
+-----------------------------------------------+
| Grants for C@localhost                        |
+-----------------------------------------------+
| GRANT USAGE ON *.* TO `C`@`localhost`         |
| GRANT SELECT ON `sc`.`student` TO `C`@`localhost` |
+-----------------------------------------------+
2 rows in set (0.00 sec)
```

收回 C 的 x（MySQL 中收回权限时，不支持指定权限来源）：



```
mysql> REVOKE SELECT
    -> ON Student
    -> FROM C@localhost;
Query OK, 0 rows affected (0.01 sec)
```

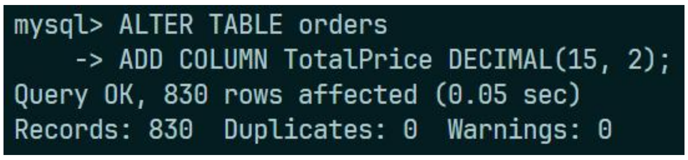
可以看到，虽然 A 和 B 都给 C 授予过权限 x，但是被一并收回了：



```
mysql> show grants for C@localhost;
+---------------------------------------+
| Grants for C@localhost                |
+---------------------------------------+
| GRANT USAGE ON *.* TO `C`@`localhost` |
+---------------------------------------+
1 row in set (0.00 sec)
```

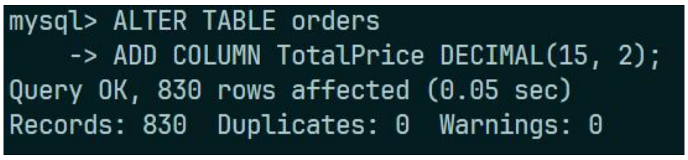
2. 触发器实验

首先，在 orders 表中，插入一列，TotalPrice，含义为该订单的总价。

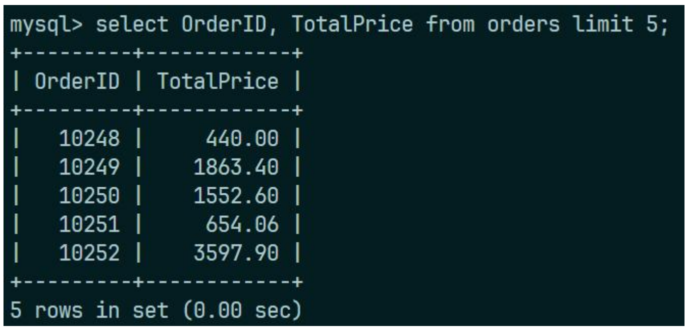


```
mysql> ALTER TABLE orders
    -> ADD COLUMN TotalPrice DECIMAL(15, 2);
Query OK, 830 rows affected (0.05 sec)
Records: 830  Duplicates: 0  Warnings: 0
```

使用 Update 语句为每个订单填入总价。

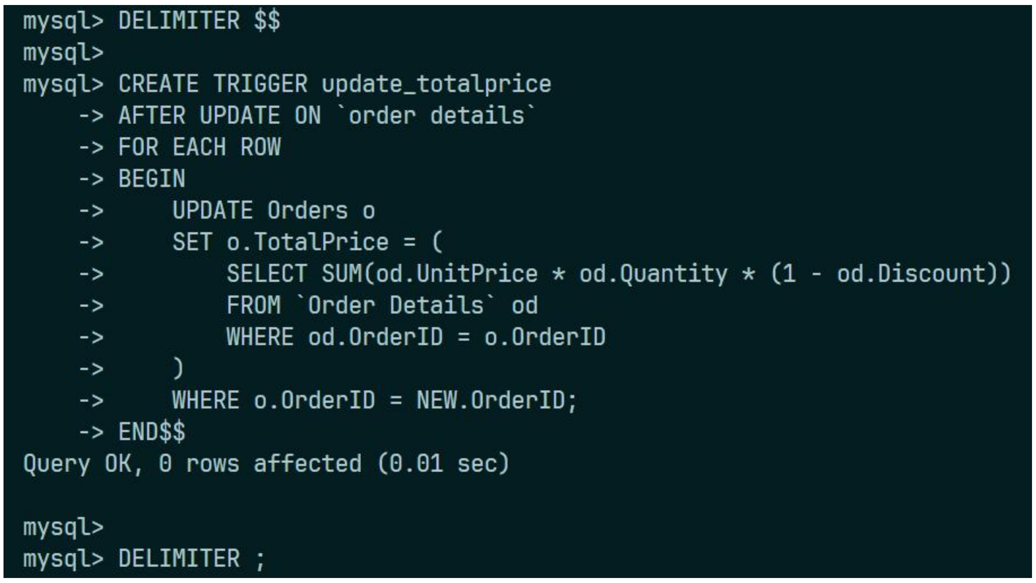


```
mysql> ALTER TABLE orders
    -> ADD COLUMN TotalPrice DECIMAL(15, 2);
Query OK, 830 rows affected (0.05 sec)
Records: 830  Duplicates: 0  Warnings: 0
```



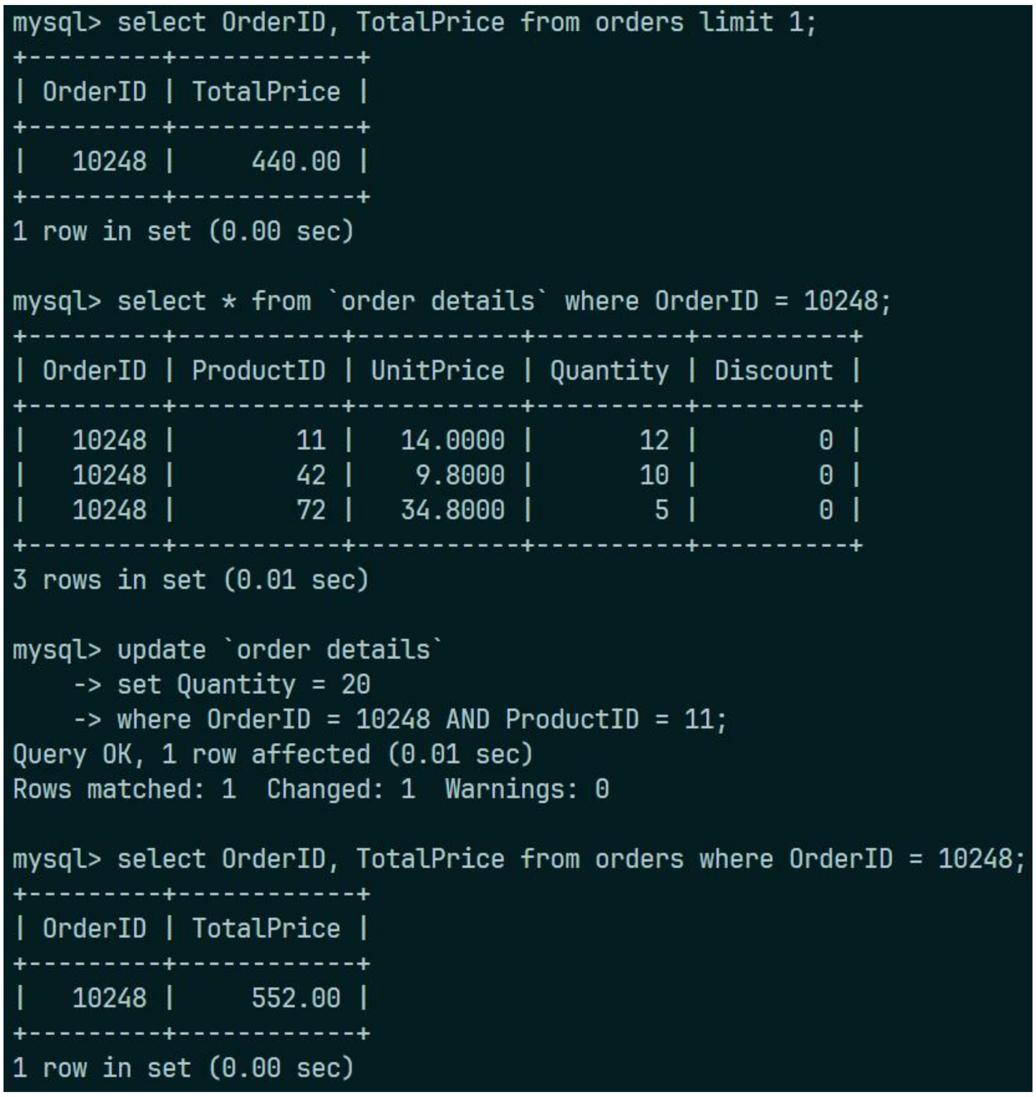
```
mysql> select OrderID, TotalPrice from orders limit 5;
+---------+------------+
| OrderID | TotalPrice |
+---------+------------+
|   10248 |     440.00 |
|   10249 |    1863.40 |
|   10250 |    1552.60 |
|   10251 |     654.06 |
|   10252 |    3597.90 |
+---------+------------+
5 rows in set (0.00 sec)
```

2.1 在 order_details 表上定义一个 UPDATE 触发器，当修改订单明细（quantity, discount）时，自动修改订单 Orders 的 TotalPrice，以保持数据一致性。



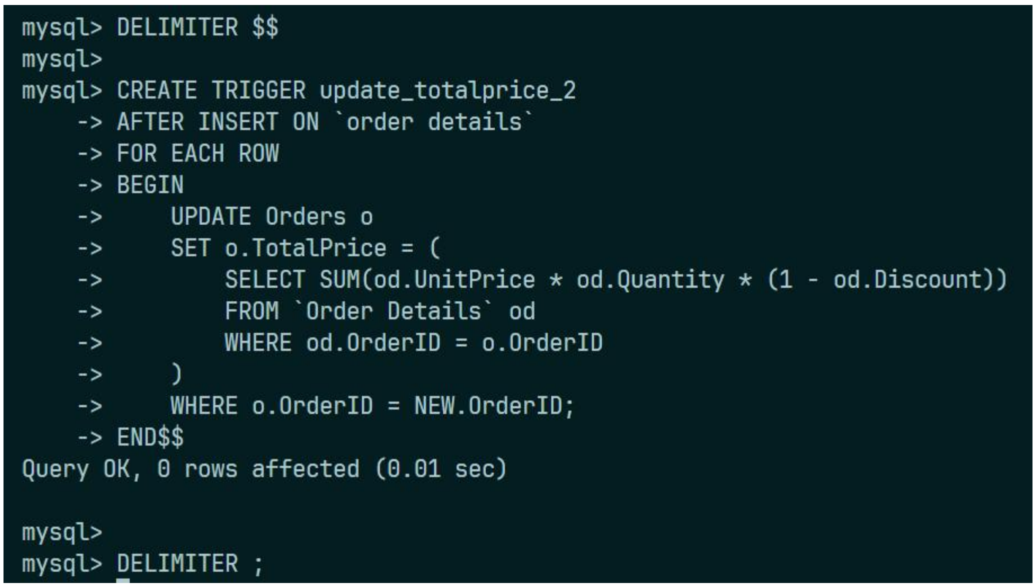
```
mysql> DELIMITER $$
mysql>
mysql> CREATE TRIGGER update_totalprice
    -> AFTER UPDATE ON `order details`
    -> FOR EACH ROW
    -> BEGIN
    ->     UPDATE Orders o
    ->     SET o.TotalPrice = (
    ->         SELECT SUM(od.UnitPrice * od.Quantity * (1 - od.Discount))
    ->         FROM `Order Details` od
    ->         WHERE od.OrderID = o.OrderID
    ->     )
    ->     WHERE o.OrderID = NEW.OrderID;
    -> END$$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
```

尝试更新 order details 表，可以看到，orders 表中相应订单的 TotalPrice 也自动更新了：



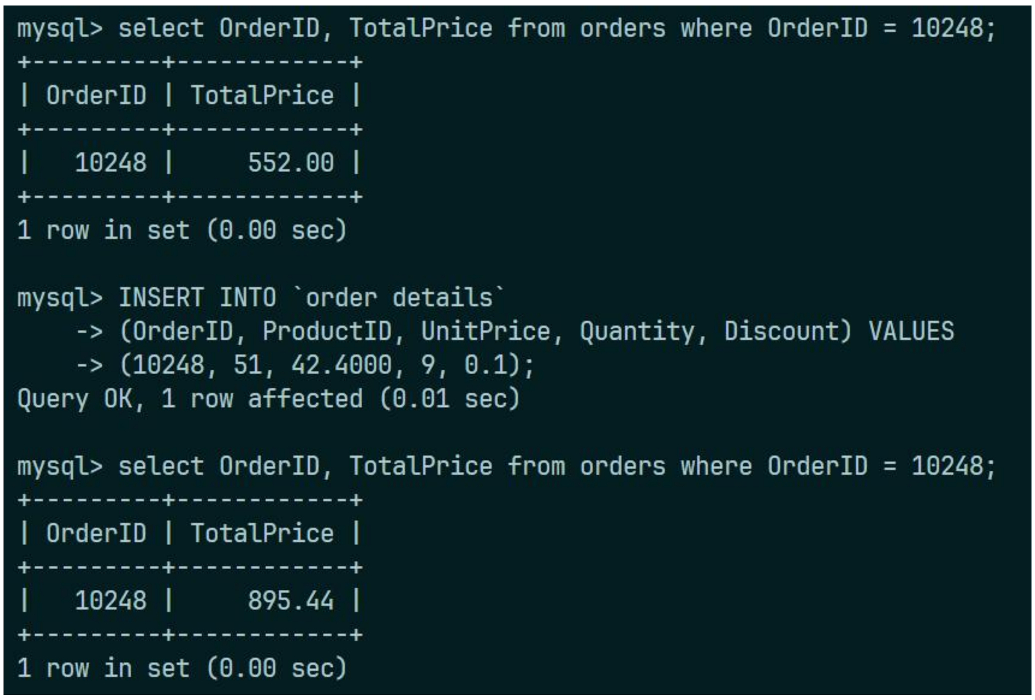
```
mysql> select OrderID, TotalPrice from orders limit 1;
+---------+------------+
| OrderID | TotalPrice |
+---------+------------+
|   10248 |     440.00 |
+---------+------------+
1 row in set (0.00 sec)

mysql> select * from `order details` where OrderID = 10248;
+---------+-----------+-----------+----------+----------+
| OrderID | ProductID | UnitPrice | Quantity | Discount |
+---------+-----------+-----------+----------+----------+
|   10248 |        11 |   14.0000 |       12 |        0 |
|   10248 |        42 |    9.8000 |       10 |        0 |
|   10248 |        72 |   34.8000 |        5 |        0 |
+---------+-----------+-----------+----------+----------+
3 rows in set (0.01 sec)

mysql> update `order details`
    -> set Quantity = 20
    -> where OrderID = 10248 AND ProductID = 11;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select OrderID, TotalPrice from orders where OrderID = 10248;
+---------+------------+
| OrderID | TotalPrice |
+---------+------------+
|   10248 |     552.00 |
+---------+------------+
1 row in set (0.00 sec)
```

2.2 在 order_details 表上定义一个 INSERT 触发器，当增加一项订单明细时，自动修改订单 Orders 的 TotalPrice，以保持数据一致性。假设增加订单明细项时，对应订单已经存在。



```
mysql> DELIMITER $$
mysql>
mysql> CREATE TRIGGER update_totalprice_2
    -> AFTER INSERT ON `order details`
    -> FOR EACH ROW
    -> BEGIN
    ->     UPDATE Orders o
    ->     SET o.TotalPrice = (
    ->         SELECT SUM(od.UnitPrice * od.Quantity * (1 - od.Discount))
    ->         FROM `Order Details` od
    ->         WHERE od.OrderID = o.OrderID
    ->     )
    ->     WHERE o.OrderID = NEW.OrderID;
    -> END$$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
```

尝试增加一项订单明细，可以看到，orders 表中相应订单的 TotalPrice 也自动更新了：



```
mysql> select OrderID, TotalPrice from orders where OrderID = 10248;
+---------+------------+
| OrderID | TotalPrice |
+---------+------------+
|   10248 |     552.00 |
+---------+------------+
1 row in set (0.00 sec)

mysql> INSERT INTO `order details`
    -> (OrderID, ProductID, UnitPrice, Quantity, Discount) VALUES
    -> (10248, 51, 42.4000, 9, 0.1);
Query OK, 1 row affected (0.01 sec)

mysql> select OrderID, TotalPrice from orders where OrderID = 10248;
+---------+------------+
| OrderID | TotalPrice |
+---------+------------+
|   10248 |     895.44 |
+---------+------------+
1 row in set (0.00 sec)
```

实验总结

本次实验通过权限授予和收回、触发器创建和使用的实践，加深了我对数据库安全性和完整性的理解和认识，很好地锻炼了我这方面在 DBMS 上的实操水平。