**数据库实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **实验人：** | **凌国明** | | **学号：** | **21307077** | **日 期：** | **2023年10月13日** |
| **院（系）：** | **计算机学院** | | | **专业（班级）：** | **计算机科学与技术** | |
| **联系方式：** |  | | |  |  | |
| **实验题目：** | | **2.1 自主存取控制实验** | | | | |

* + - 1. **实验目的**

掌握自主存取控制权限的定义和维护方法

* + - 1. **实验内容和要求**

定义用户、角色，分配权限给用户、角色，回收权限，以相应的用户名登录数据库验证权限分配是否为正确。选择一个应用场景，使用自主存取控制机制设计权限分配。可以采用两种方案。

方案一：采用SYSTEM超级用户登录数据库，完成所有权限分配工作，然后用相应用户名登录数据库以验证权限分配正确性;

方案二：采用SYSTEM用户登录数据库创建三个部门经理用户，并分配相应的权限，然后分别用三个经理用户名登录数据库，创建相应部门的USER、ROLE，并分配相应权限。

下面的实验报告示例采用了实验方案一。验证权限分配之前，请备份好数据库;针对不同用户所具有的权限，分别设计相应的SQL语句加以验证。

* + - 1. **实验重点和难点**

实验重点：定义角色，分配权限和回收权限。

实验难点：实验方案二实现权限的再分配和回收。

* + - 1. **实验工具**

MySQL、SQL Server、Navicat

* + - 1. **实验过程**

设有一个企业，包括采购、销售和客户管理等三个部门，

采购部门经理user\_pm， 采购员user\_pe;

销售部门经理user\_sm， 销售员user\_se;

客户管理部门经理user\_cm， 职员user\_ce。

该企业一个信息系统覆盖采购、销售和客户管理等三个部门的业务，其数据库为TPC-H。针对此应用场景，使用自主存取控制机制设计一个具体的权限分配方案。

* 注：在 MySQL 环境中，以root用户登录，运行以下语句（参阅<http://c.biancheng.net/view/7490.html> ）。

(1)创建用户

①为采购、销售和客户管理等三个部门的经理创建用户标识，要求具有创建用户或角色的权利。

CREATE USER user\_pm IDENTIFIED BY '123456';

GRANT CREATE USER on \*.\* to user\_pm;

CREATE USER user\_sm IDENTIFIED BY '123456';

GRANT CREATE USER on \*.\* to user\_sm;

CREATE USER user\_cm IDENTIFIED BY '123456';

GRANT CREATE USER on \*.\* to user\_cm;

②为采购、销售和客户管理等三个部门的职员创建用户标识和用户口令。

CREATE USER user\_pe IDENTIFIED BY '123456';

CREATE USER user\_se IDENTIFIED BY '123456';

CREATE USER user\_ce IDENTIFIED BY '123456';

(2)创建角色并分配权限

* 在 MySQL 环境下中要激活角色，即执行以下语句（参阅：<https://blog.csdn.net/qq_39746820/article/details/123710158> ）：

SET GLOBAL activate\_all\_roles\_on\_login=ON;

①为各个部门分别创建一个查询角色，并分配相应的查询权限。

CREATE ROLE PurchaseQueryRole;

GRANT SELECT ON TABLE Part TO PurchaseQueryRole;

GRANT SELECT ON TABLE Supplier TO PurchaseQueryRole;

GRANT SELECT ON TABLE PartSupp TO PurchaseQueryRole;

CREATE ROLE SaleQueryRole;

GRANT SELECT ON TABLE Orders TO SaleQueryRole;

GRANT SELECT ON TABLE LineItem TO SaleQueryRole;

CREATE ROLE CustomerQueryRole;

GRANT SELECT ON TABLE Customer TO CustomerQueryRole;

GRANT SELECT ON TABLE Nation TO CustomerQueryRole;

GRANT SELECT ON TABLE Region TO CustomerQueryRole;

②为各个部门分别创建一个职员角色，对本部门信息具有查看、插入权限。

CREATE ROLE PurchaseEmployeeRole;

GRANT SELECT, INSERT ON TABLE Part TO PurchaseEmployeeRole;

GRANT SELECT, INSERT ON TABLE Supplier TO PurchaseEmployeeRole;

GRANT SELECT, INSERT ON TABLE PartSupp TO PurchaseEmployeeRole;

CREATE ROLE SaleEmployeeRole;

GRANT SELECT, INSERT ON TABLE Orders TO SaleEmployeeRole;

GRANT SELECT, INSERT ON TABLE LineItem TO SaleEmployeeRole;

CREATE ROLE CustomerEmployeeRole;

GRANT SELECT, INSERT ON TABLE Customer TO CustomerEmployeeRole;

GRANT SELECT, INSERT ON TABLE Nation TO CustomerEmployeeRole;

GRANT SELECT, INSERT ON TABLE Region TO CustomerEmployeeRole;

③为各部门创建一个经理角色，相应角色对本部门的信息具有完全控制权限，对其他部门的信息具有查询权。经理有权给本部门职员分配权限。

CREATE ROLE PurchaseManagerRole;

GRANT CREATE ROLE on \*.\* to PurchaseManagerRole;

GRANT ALL ON TABLE Part TO PurchaseManagerRole;

GRANT ALL ON TABLE Supplier TO PurchaseManagerRole;

GRANT ALL ON TABLE PartSupp TO PurchaseManagerRole;

GRANT SaleQueryRole TO PurchaseManagerRole;

GRANT CustomerQueryRole TO PurchaseManagerRole;

CREATE ROLE SaleManagerRole;

GRANT CREATE ROLE on \*.\* to SaleManagerRole;

GRANT ALL ON TABLE Orders TO SaleManagerRole;

GRANT ALL ON TABLE LineItem TO SaleManagerRole;

GRANT PurchaseQueryRole TO SaleManagerRole;

GRANT CustomerQueryRole TO SaleManagerRole;

CREATE ROLE CustomerManagerRole;

GRANT CREATE ROLE on \*.\* to CustomerManagerRole;

GRANT ALL ON TABLE Customer TO CustomerManagerRole;

GRANT ALL ON TABLE Nation TO CustomerManagerRole;

GRANT ALL ON TABLE Region TO CustomerManagerRole;

GRANT PurchaseQueryRole TO CustomerManagerRole;

GRANT SaleQueryRole TO CustomerManagerRole;

(3)给用户分配权限

①给各部门经理分配权限。

GRANT PurchaseManagerRole TO user\_pm;

GRANT SaleManagerRole TO user\_sm;

GRANT CustomerManagerRole TO user\_cm;

②给各部门职员分配权限。

GRANT PurchaseEmployeeRole TO user\_pe;

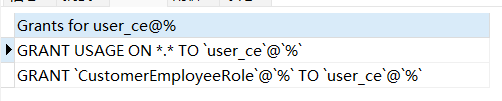
GRANT SaleEmployeeRole TO user\_se;

GRANT CustomerEmployeeRole TO user\_ce;

(4)回收角色或用户权限

①收回客户经理角色的销售信息查看权限。

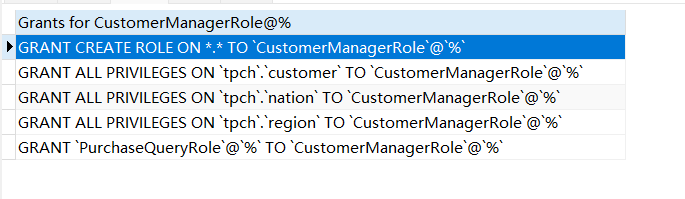
第1步执行“SHOW GRANTS FOR CustomerManagerRole;”，可显示：



第2步执行：

REVOKE SaleQueryRole FROM CustomerManagerRole;

第3步再执行“SHOW GRANTS FOR CustomerManagerRole;”，可显示：



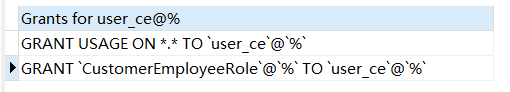
结论：

执行 REVOKE SaleQueryRole FROM CustomerManagerRole;命令后，

CustomerManagerRole 失去 SaleQueryRole 的权限。

②回收user\_ce的客户部门职员权限

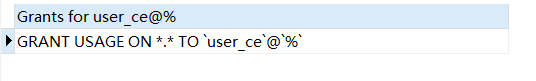
第1步执行“SHOW GRANTS FOR user\_ce;”，可显示：



第2步执行：

REVOKE CustomerEmployeeRole FROM user\_ce;

第3步再执行“SHOW GRANTS FOR user\_ce;”，可显示：



结论：

执行 REVOKE CustomerEmployeeRole FROM user\_ce;命令后，

user\_ce 失去 CustomerEmployeeRole 的权限。

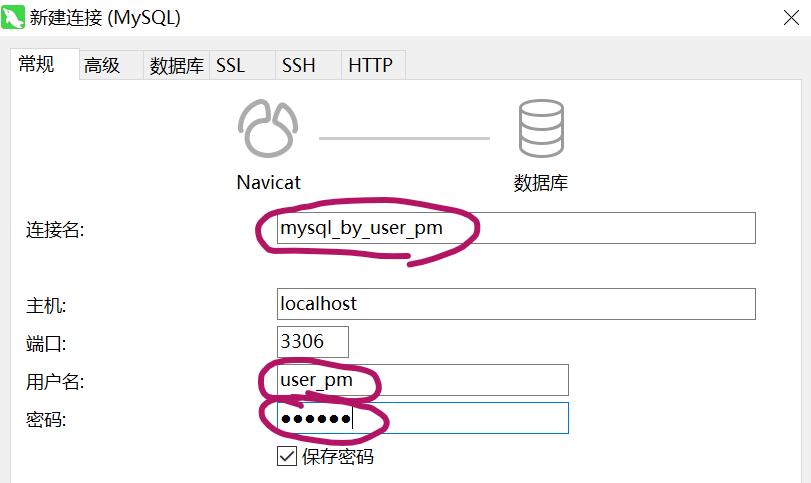
(5)验证权限分配正确性

①以user\_pm用户名登录数据库，通过执行以下两条命令验证采购部门经理的权限。

SELECT \* FROM Part;

DELETE FROM orders;

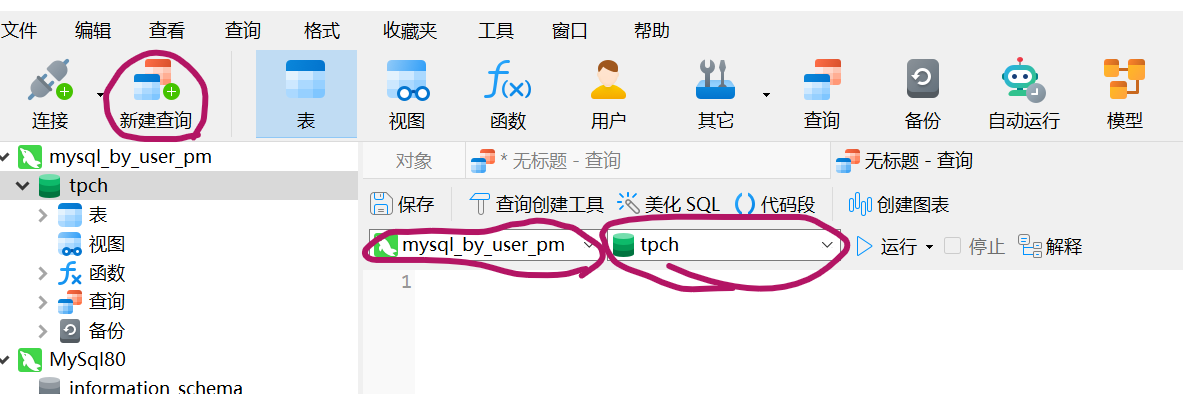
第1步在Navicat中以user\_pm用户名登录数据库，如图：

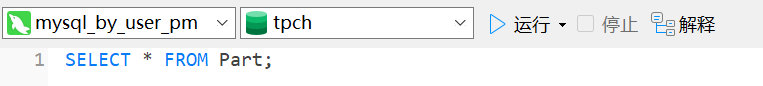
“确定”后，可见：



第2步“新建查询”，确保当前数据库是连接 mysql\_by\_user\_pm中的数据库 tpch.



第3步执行查询语句，结果如图；

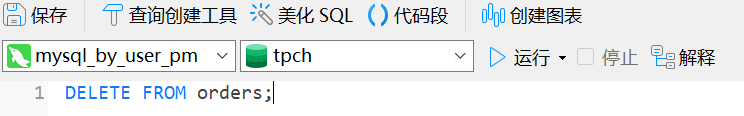


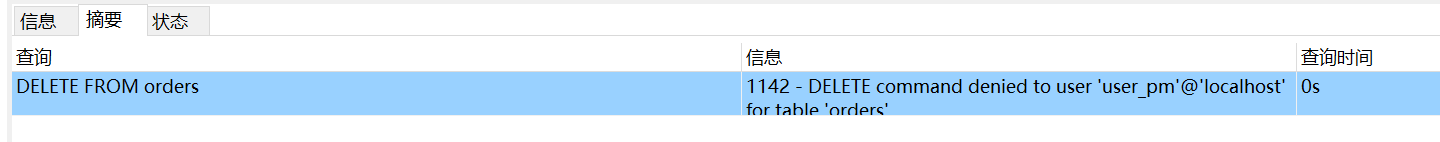


结论：

user\_pm在localhost上，有查询表Part的权限

第4步执行删除语句，结果如图；





结论：

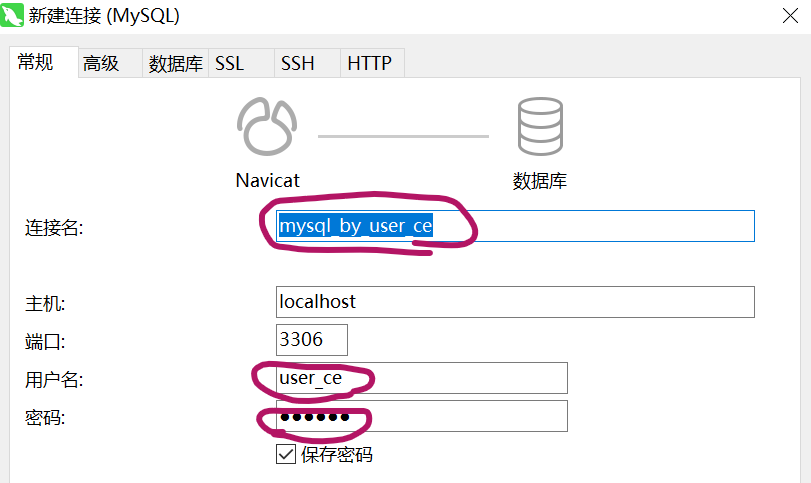
在“localhost”上，用户“user\_pm”没有删除“orders”表中数据的权限。因此删除操作失败。用户需要有相应删除权限才能执行这个操作

②以user\_ce用户名登录数据库，通过执行以下两条命令验证该用户的权限（注：前面的操作已经回收了user\_ce的客户部门职员权限）

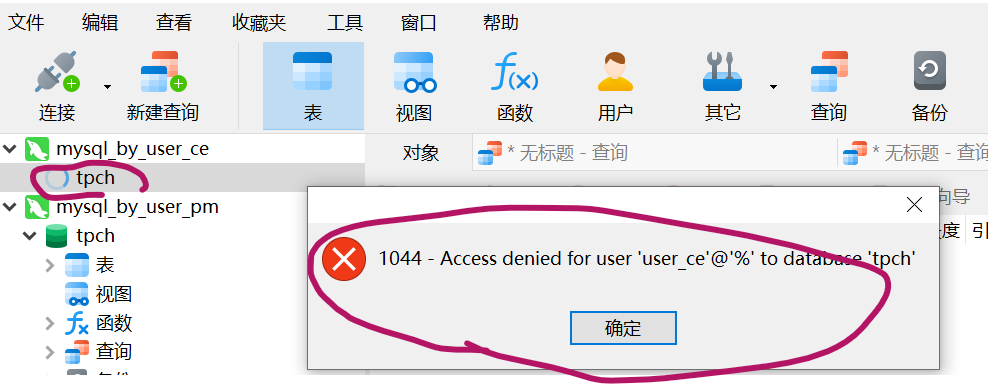
SELECT \* FROM Customer;

SELECT \* FROM Part;

第1步在Navicat中以user\_ce用户名登录数据库，如图：

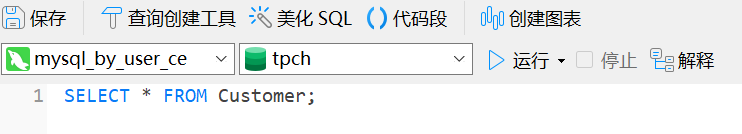
“确定”后，可见(注：出现的提示是指该用户不能访问该数据库中的任何表)：



第2步“新建查询”，确保当前数据库是连接 mysql\_by\_user\_ce中的数据库 tpch.



第3步执行以下查询语句，结果如图；

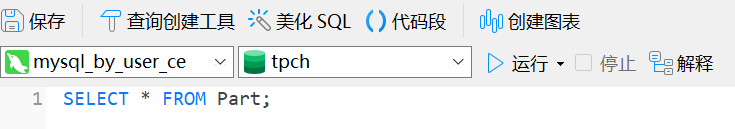




结论：

用户“user\_ce”没有权限访问“tpch”数据库，因此访问遭到拒绝

第4步再执行以下查询语句，结果如图；





结论：

用户“user\_ce”没有权限访问“tpch”数据库，因此对part表的查询遭到拒绝

* + - 1. **与实验结果相关的文件**

无

* + - 1. **实验总结**

通过本次实验，我成功掌握了自主存取控制权限的定义和维护方法。在实验中，我定义了用户和角色，分配了权限，并验证了权限分配的正确性，这有助于我更好地理解数据库权限管理的重要性。在实验中，我发现定义角色和进行权限的再分配和回收是实验的重点和难点。尤其是在方案二中，需要更多的操作步骤来确保权限的准确性。通过这次实验，我增强了对数据库权限管理的认识，这对于未来的数据库管理工作将非常有帮助。