**《数据库系统原理》实验报告**

### **实验题目：**实验11 数据库的安全性设计

**姓名：**

**实验日期：**

**实验内容及完成情况：**（可续页）

### 实验11 数据库的安全性设计

**实验学时：2学时**

**实验类型：验证**

**实验要求：必做**

**一、实验目的**

加深对数据安全性的理解，并掌握Navicat中有关用户，角色及操作权限的管理方法。掌握数据库审计的设置和管理方法，以便监控数据库操作，维护数据库安全。

**二、实验内容**

1、在Navicat中，管理建立用户和分配用户权限。

在这里，我们可以发现root是MySQL最高级别权限的用户，它拥有查看、修改和删除MySQL软件中所有数据库的权限。当需要有多个数据库，并且分配给不同的用户使用，多个用户之间只有查看自己对应数据库的权限，不相互干扰, 需要建立多个数据库和用户，给用户设置管理指定数据库的权限。

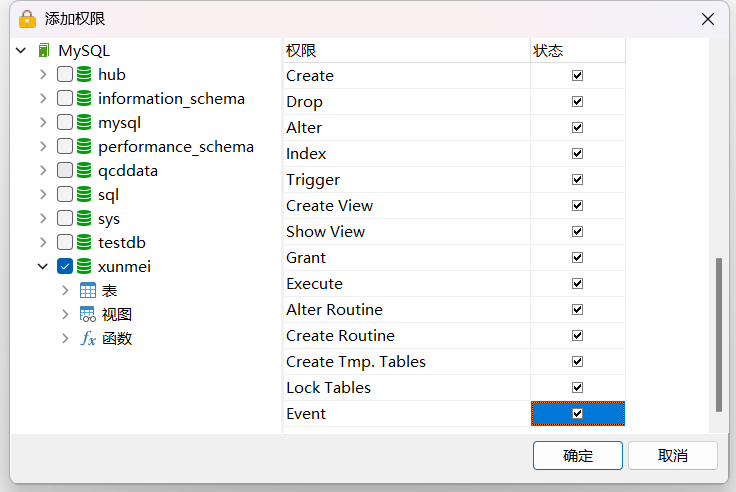
操作方法如下：

①右键点击“MySQL” 连接，选择“连接属性”，将“保存密码”取消。

②鼠标右键点击“MySQL” 连接，选择“创建数据库”，创建名为xunmei的数据库，字符集选择utf8mb4排序规则选择utf8mb4\_general\_ci。

③打开数据库xunmei后，点击界面上方的“用户”—“新建用户”—输入用户名“xmuser”、主机“localhost”、密码“123456”—保存（服务器权限自行定义）。

④点选新建的用户xmuser—编辑用户—权限—添加权限--选择数据库xunmei，添加相应权限（可不全选），如图所示：



⑤设置到此完成。为了验证我们的设置是否正确，我们可以用刚才建立的用户xmuser和密码123456来连接localhost主机地址（方法：关闭连接MySQl，该连接属于最高权限用户root。我们用新用户新建一个到localhost的连接来模拟不同用户使用数据库的过程）。连上后，可使用用户xmuser对数据库xunmei或其他数据库进行操作，看是否与自己设置的权限相吻合。（可只添加添加少量权限，然后尝试在xunmei里进行操作，看是否能成功）。

2、打开数据库审计开关，并查看存储审计信息的存储文件所在位置。

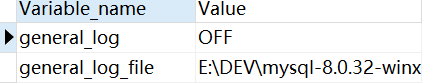
①显示当前审计开关状态。

其中log\_timestamps表示记录审计日志的时间是从那里获取，general\_log表示审计功能是否开启，general\_log\_file表示审计信息的存储文件。

show global variables like 'log\_timestamps';

show global variables like '%general%';

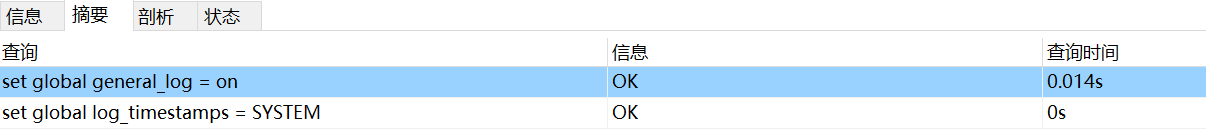




②打开审计开关

set global general\_log = on;

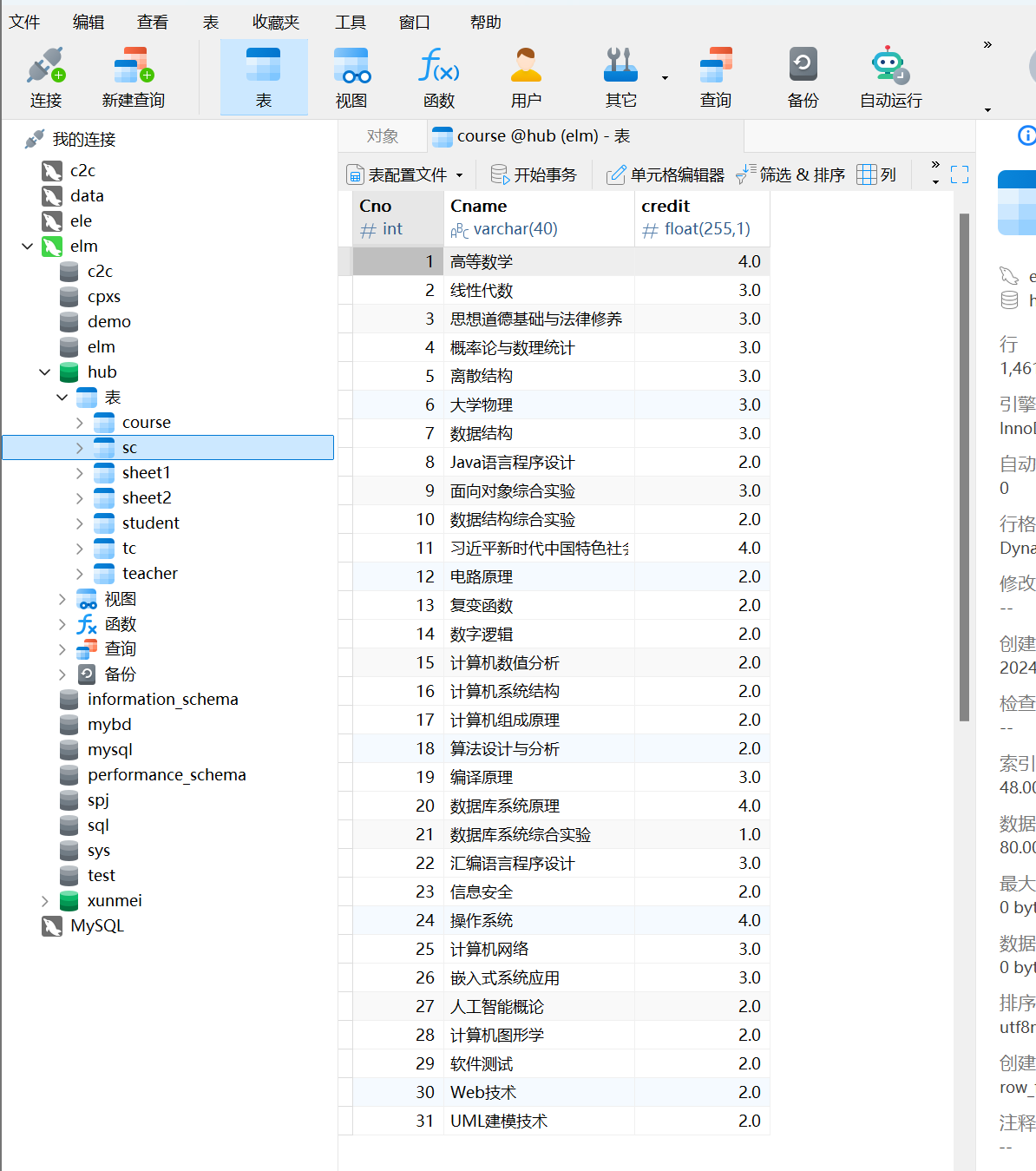
set global log\_timestamps = SYSTEM;



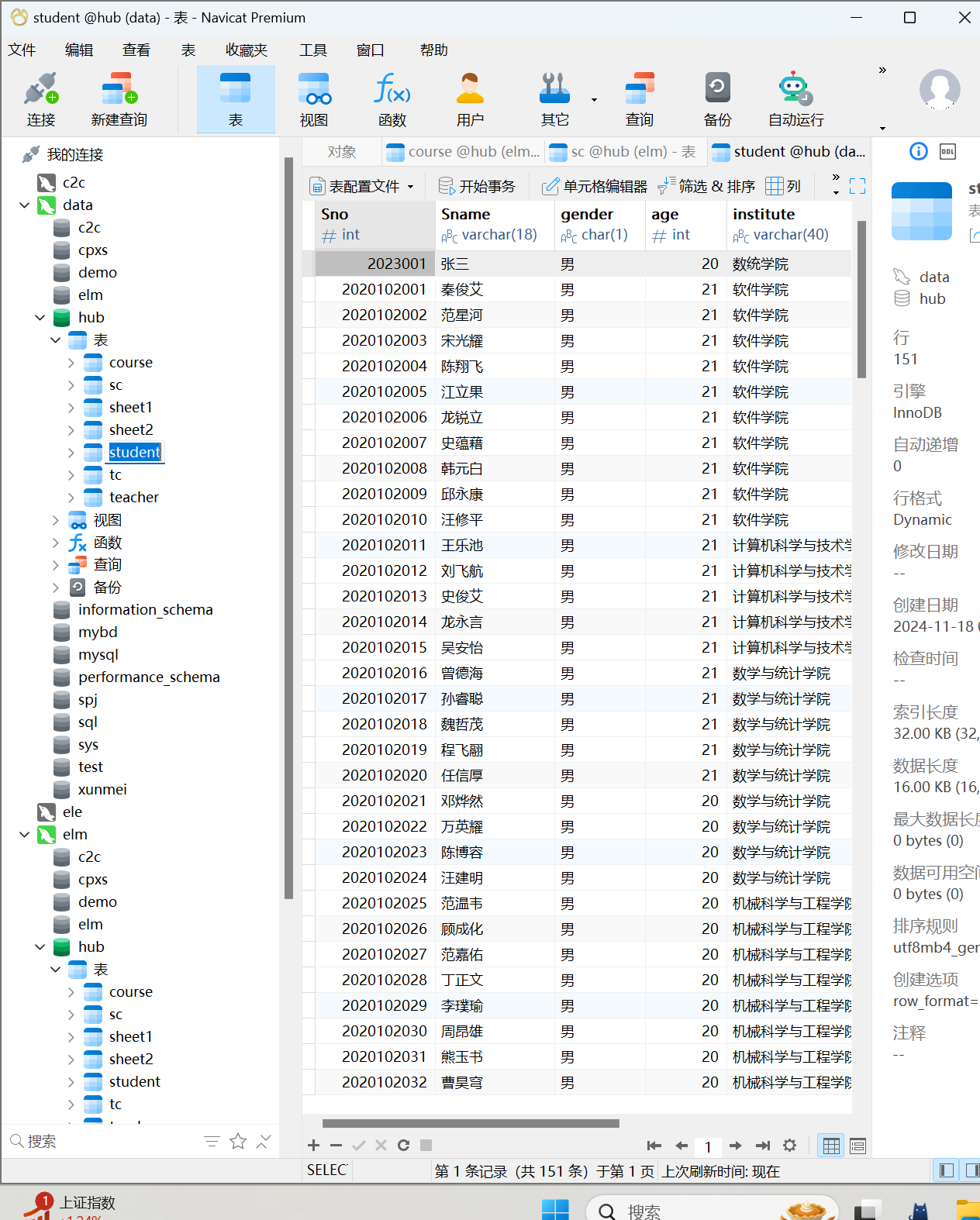
**三、课后练习题**

1、使用两个不同的用户通过Navicat和查询分析器查看hub数据库中student、course两个表的所有数据。

用户elm查看hub数据库中student,course表：



用户data查看hub数据库中student,course表:



2、删除用户xmuser，尝试新建另一个用户来管理数据库、表等：

添加新用户对数据库hub的权限，比较在有无相关权限的情况之下进行操作的区别：

1. Create。尝试新建表
2. Update。尝试修改表数据或结构
3. ……..（自行设计，总共尝试5组，进行对比）

**1.删除用户** xmuser：

DROP USER 'xmuser'@'localhost';

1. 创建新用户并赋权:

新建用户：

CREATE USER 'dbmanager'@'localhost' IDENTIFIED BY 'password123';

给用户赋予 hub 数据库的权限：

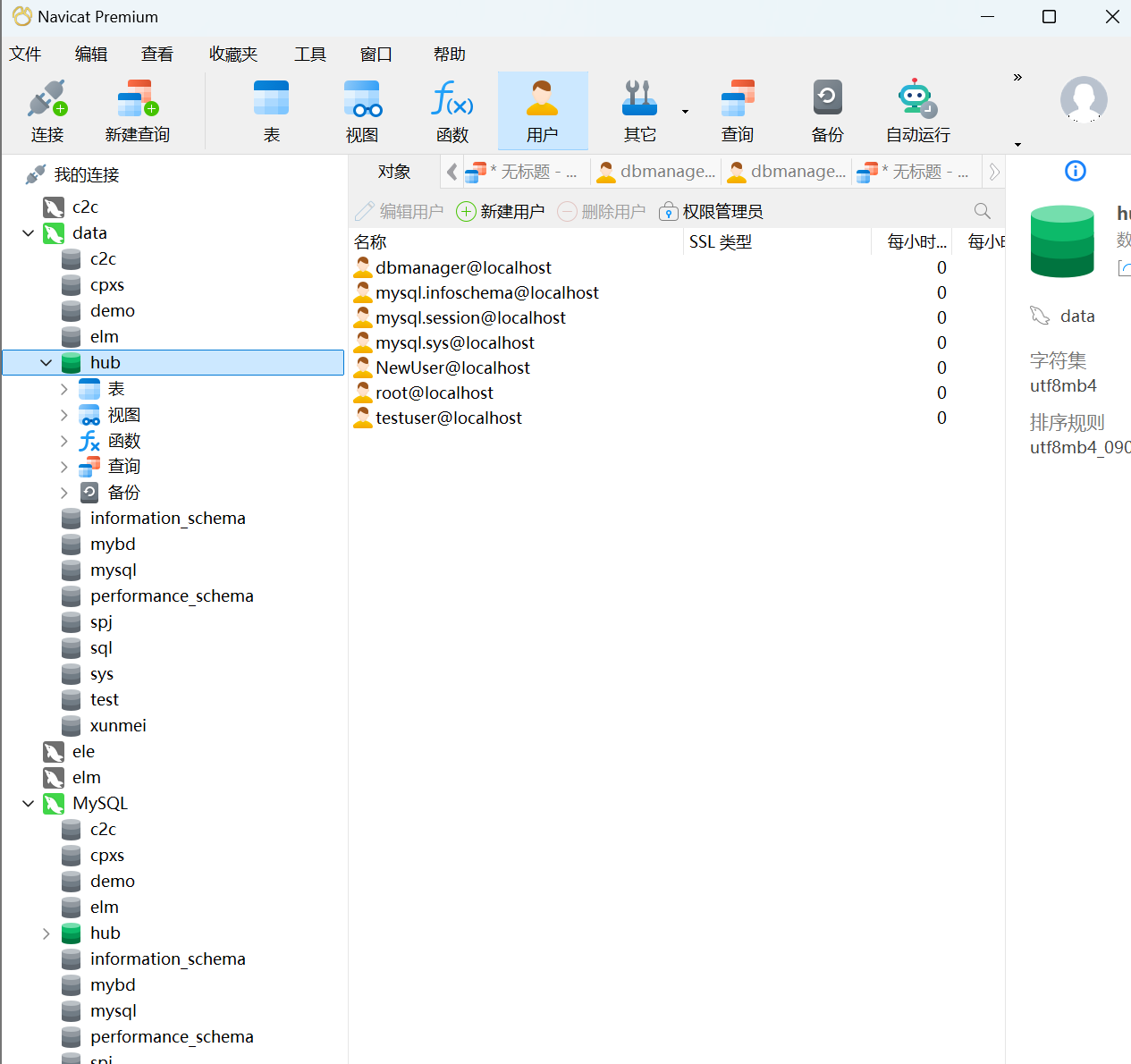
GRANT ALL PRIVILEGES ON hub.\* TO 'dbmanager'@'localhost';

FLUSH PRIVILEGES;

1. **不赋权的情况下操作**：

创建一个新用户，无权限操作:

CREATE USER 'testuser'@'localhost' IDENTIFIED BY 'password456';



**4. **设计5组操作对比**：**

#### ① Create：尝试新建表

#### ****有权限****：

-- 使用 dbmanager 用户

USE hub;

CREATE TABLE test\_table (

id INT PRIMARY KEY,

name VARCHAR(50)

);

**无权限**：

-- 使用 testuser 用户

USE hub;

CREATE TABLE test\_table (

id INT PRIMARY KEY,

name VARCHAR(50)

);-- 预期错误：ERROR 1044 (42000): Access denied for user 'testuser'@'localhost'

#### ② Update：修改表数据或结构

**有权限**：

-- 修改表数据

INSERT INTO test\_table (id, name)VALUES (1, 'Alice');

UPDATE test\_table SET name = 'Bob' WHERE id = 1;

-- 修改表结构

ALTER TABLE test\_table ADD COLUMN age INT;

**无权限**：

-- 修改表数据

UPDATE test\_table SET name = 'Charlie' WHERE id = 1;

-- 修改表结构

ALTER TABLE test\_table DROP COLUMN age;

-- 预期错误：Access denied

#### 

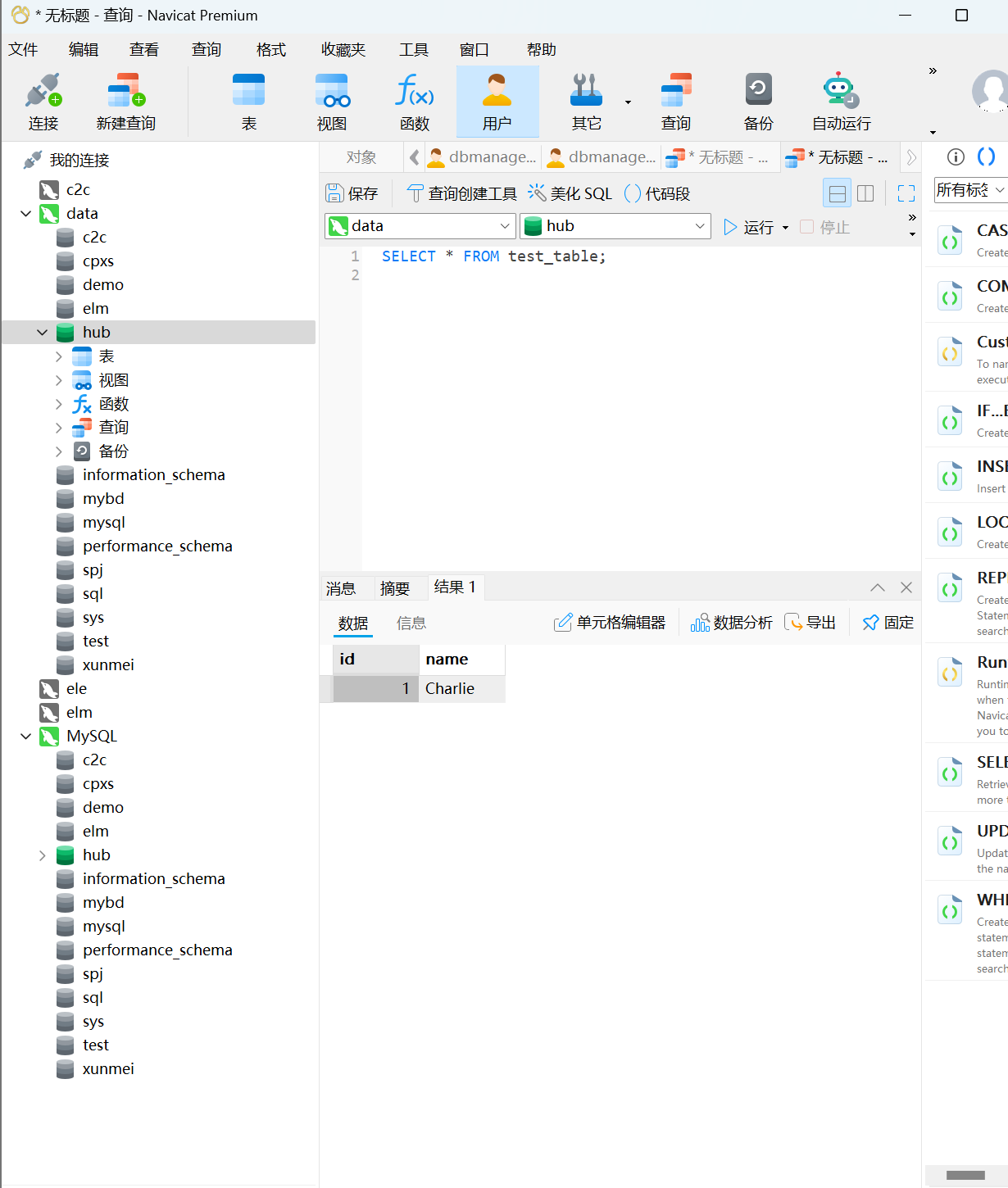
#### ③ Select：查询表数据

**有权限**：

SELECT \* FROM test\_table;

**无权限**：

SELECT \* FROM test\_table;-- 预期错误：Access denied



#### ④ Drop：删除表

**有权限**：

DROP TABLE test\_table;

**无权限**：

DROP TABLE test\_table;-- 预期错误：Access denied

#### 

#### ⑤ Grant：赋予权限

**有权限**（需有 GRANT OPTION 权限）：

GRANT SELECT ON hub.test\_table TO 'anotheruser'@'localhost';

**无权限**：

GRANT SELECT ON hub.test\_table TO 'anotheruser'@'localhost';-- 预期错误：Access denied

**实验结果与对比**：

**有权限的用户**可以顺利完成表的创建、更新、查询、删除等操作。

**无权限的用户**在操作时均会收到 Access denied 错误，无法对数据库或表进行任何修改。

权限管理有效地保障了数据库的安全性和操作的可控性。

**四、思考题**

1、解释10个数据库操作权限的含义和影响。

**1.SELECT**

* **含义**：允许用户读取表或视图中的数据。
* **影响**：只提供数据查询能力，适合需要分析或查看数据但不进行修改的用户，确保数据完整性。

**2.INSERT**

* **含义**：允许用户向表中插入新记录。
* **影响**：用户可以添加数据，但无法修改或删除已有数据，适合日志记录等场景。

**3.UPDATE**

* **含义**：允许用户修改表中已有的数据。
* **影响**：用户可以更改数据内容，适合需要实时更新数据的场景，但可能带来数据污染风险。

**4.DELETE**

* **含义**：允许用户删除表中的记录。
* **影响**：用户可移除不需要的数据，但需防止误删数据导致信息丢失。

**5.CREATE**

* **含义**：允许用户创建新的数据库对象，如表、视图等。
* **影响**：用户能扩展数据库功能，但可能因创建冗余对象导致数据库管理复杂性增加。

**6.DROP**

* **含义**：允许用户删除数据库对象，如表、视图、数据库等。
* **影响**：删除操作是不可逆的，需谨慎分配此权限以避免误删重要数据。

**7.GRANT OPTION**

* **含义**：允许用户将其拥有的权限授予其他用户。
* **影响**：适合管理员角色，但需注意避免权限滥用。

**8.REVOKE**

* **含义**：允许用户收回已授予的权限。
* **影响**：控制权限范围，提升数据库安全性。

**9.ALTER**

* **含义**：允许用户修改数据库对象的结构，如表的列或索引。
* **影响**：用户可调整数据库设计以满足需求，但需控制修改权限以防止错误操作。

**10.EXECUTE**

* **含义**：允许用户运行存储过程或函数。
* **影响**：提高了操作自动化水平，但需确保存储过程和函数的安全性。

2、试着设计一个例子，分析数据库审计对数据库性能的影响情况。

**场景设计**  
一个电商平台的 orders 表存储了大量订单数据，需要审计所有数据操作（如 INSERT、UPDATE、DELETE），以满足数据合规性要求。

**表结构**：

CREATE TABLE orders (

order\_id INT PRIMARY KEY,

customer\_id INT,

product\_id INT,

order\_date DATETIME,

amount DECIMAL(10, 2)

);

**启用审计功能**：

配置审计以记录所有对 orders 表的修改操作：

CREATE TRIGGER audit\_orders

AFTER INSERT OR UPDATE OR DELETE ON orders

FOR EACH ROW

BEGIN

INSERT INTO audit\_log (action, order\_id, timestamp, user)

VALUES (CASE

WHEN INSERTING THEN 'INSERT'

WHEN UPDATING THEN 'UPDATE'

WHEN DELETING THEN 'DELETE'

END,

NEW.order\_id,

NOW(),

USER());

END;

audit\_log 表：

CREATE TABLE audit\_log (

action VARCHAR(10),

order\_id INT,

timestamp DATETIME,

user VARCHAR(50)

);

实验步骤：

**未启用审计**：进行一组数据插入、更新、删除操作并记录执行时间。

-- 插入10000条记录

INSERT INTO orders (order\_id, customer\_id, product\_id, order\_date, amount)

VALUES (1, 101, 201, NOW(), 100.00), ..., (10000, 102, 202, NOW(), 150.00);

-- 测试更新和删除操作

UPDATE orders SET amount = amount \* 1.1 WHERE order\_id <= 5000;

DELETE FROM orders WHERE order\_id > 8000;

**启用审计**：重复相同的操作，记录执行时间：

-- 插入10000条记录

INSERT INTO orders (order\_id, customer\_id, product\_id, order\_date, amount)

VALUES (1, 101, 201, NOW(), 100.00), ..., (10000, 102, 202, NOW(), 150.00);

-- 测试更新和删除操作

UPDATE orders SET amount = amount \* 1.1 WHERE order\_id <= 5000;

DELETE FROM orders WHERE order\_id > 8000;

**分析结果**：

**性能影响**：

* 审计功能会增加额外的插入操作，用于记录日志，导致操作时间明显增加。
* 对高频数据操作的表（如 orders）审计可能显著增加数据库负载。

**优化建议**：

* 审计只针对关键字段或高风险操作。
* 使用批量写入审计日志，减少每次操作的事务开销。
* 定期清理或归档审计日志，避免表过大影响性能。

**实验总结：**

通过本次实验，我们深入理解了数据库安全性的核心概念，并在 Navicat 工具中实践了用户管理、角色配置以及权限分配等操作，具体收获如下：

**用户与角色管理**：

·学会了如何创建、修改和删除用户，分配不同权限，以及使用角色来简化权限管理。

·理解了权限的粒度控制，如对单个表或整个数据库的操作权限分配，提高了权限管理效率。

**操作权限的影响**：

·对比了不同权限（如 SELECT、INSERT、UPDATE、DROP）对数据库操作的限制，进一步认识到权限控制在保护数据完整性和安全性方面的重要性

**数据库审计设置**：

·学会了配置和管理数据库审计功能，记录关键操作日志，从而实现对数据库操作的监控。

·通过实验验证了审计对性能的影响，并总结出优化建议，如定期清理日志和审计关键操作。

**安全性提升的意义**：

·理解了合理分配权限和启用审计对防止误操作、非法访问和数据泄露的重作用，为实际项目中维护数据库安全奠定了基础。

本实验提升了我们在数据库安全管理和性能优化方面的能力，强化了理论与实践的结合，为后续更加复杂的数据库管理提供了宝贵经验。

**教师评语及成绩**：