**《数据库系统原理》实验报告**

**实验题目：数据库的安全性设计**

**姓名： 郑德凯** **实验日期： 2024年 11月 26日**

**实验内容及完成情况：**

# 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。

# 实验内容

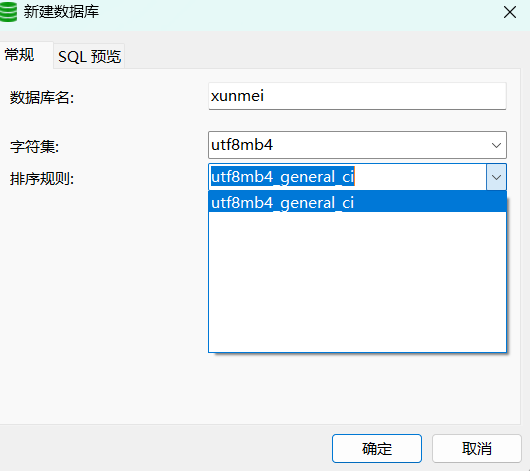
1、在Navicat中，管理建立用户和分配用户权限。

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。

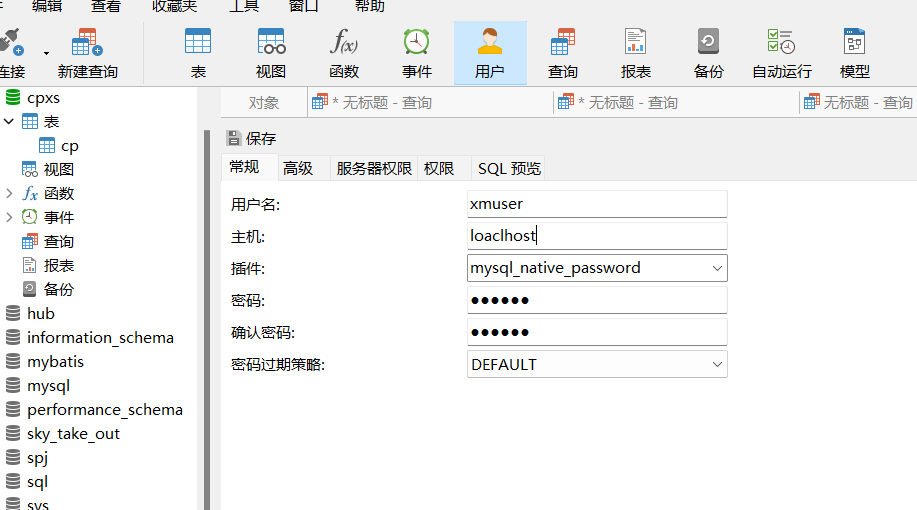
操作方法如下：

①右键点击“MySQL” 连接，选择“连接属性”，将“保存密码”取消。

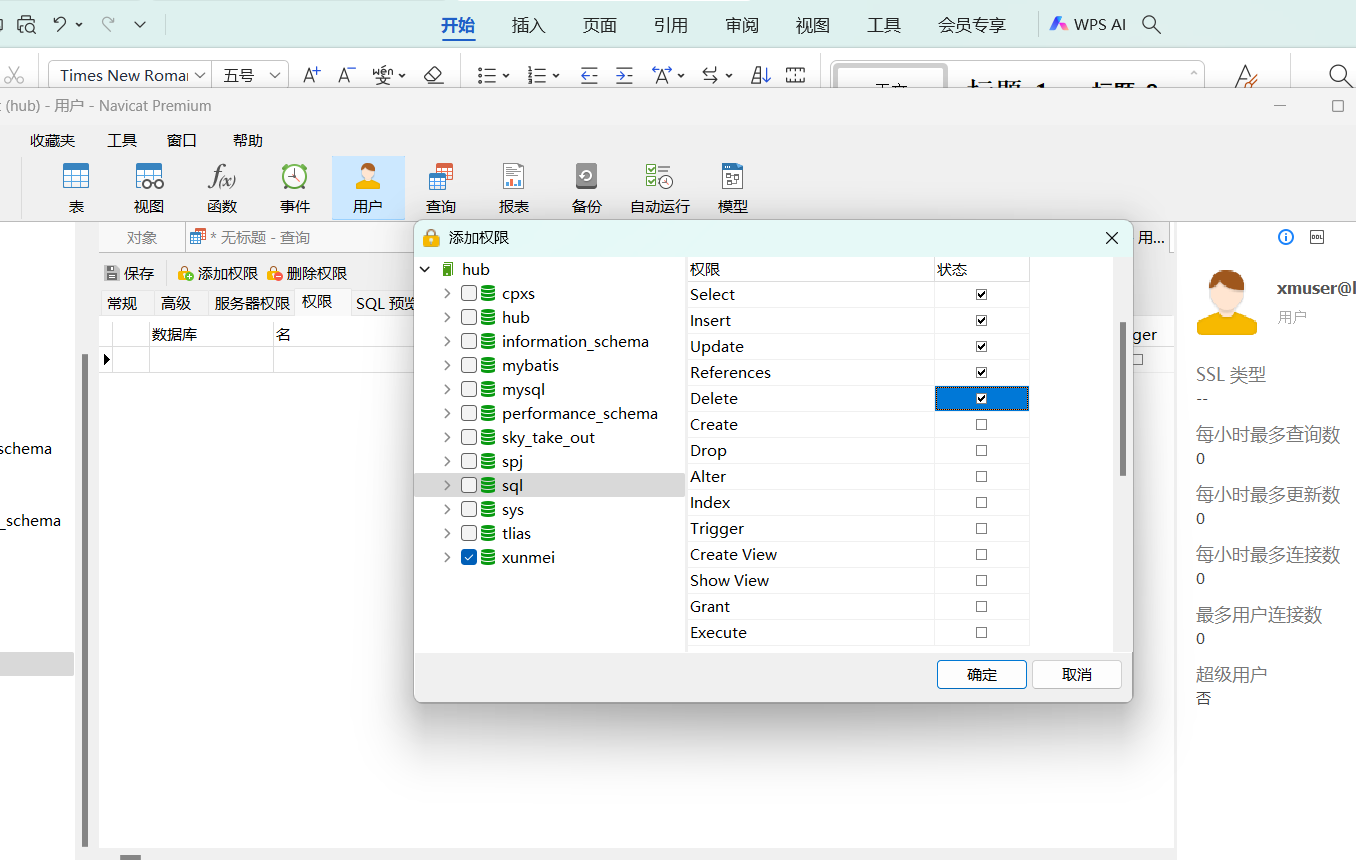
②鼠标右键点击“MySQL” 连接，选择“创建数据库”，创建名为xunmei的数据库，字符集选择utf8mb4排序规则选择utf8mb4\_general\_ci。



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）。

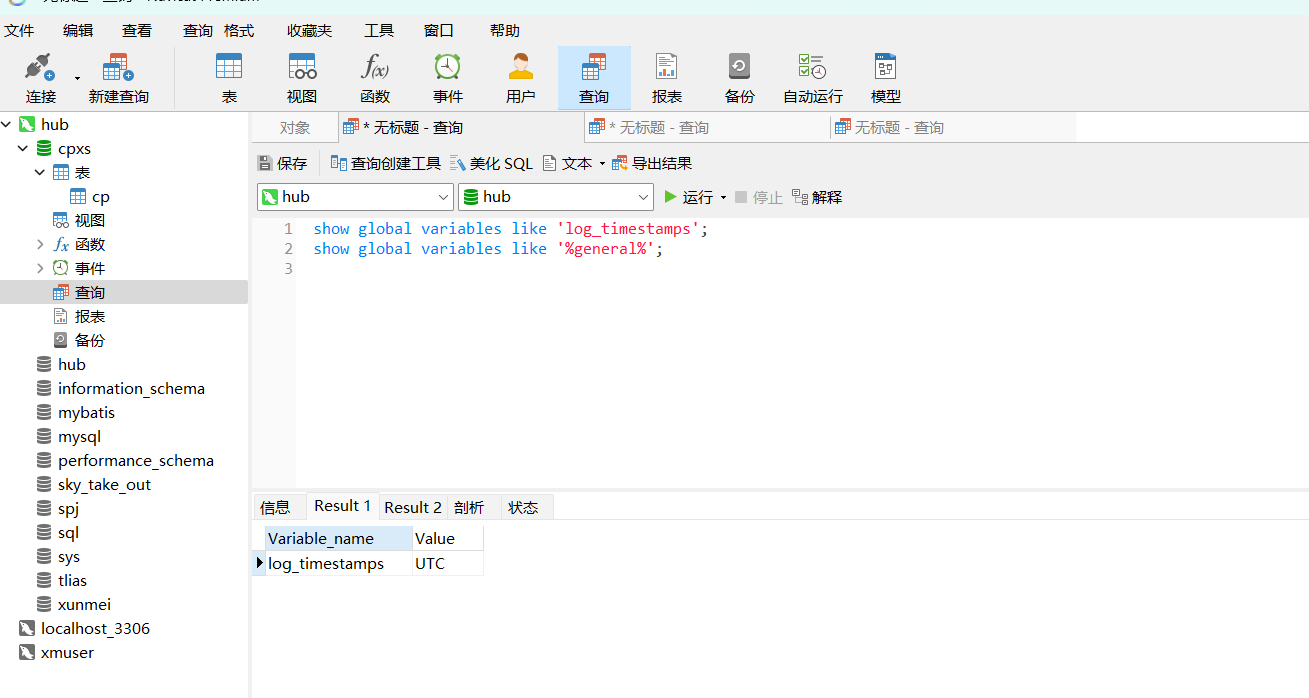
2、打开数据库审计开关，并查看存储审计信息的存储文件所在位置。

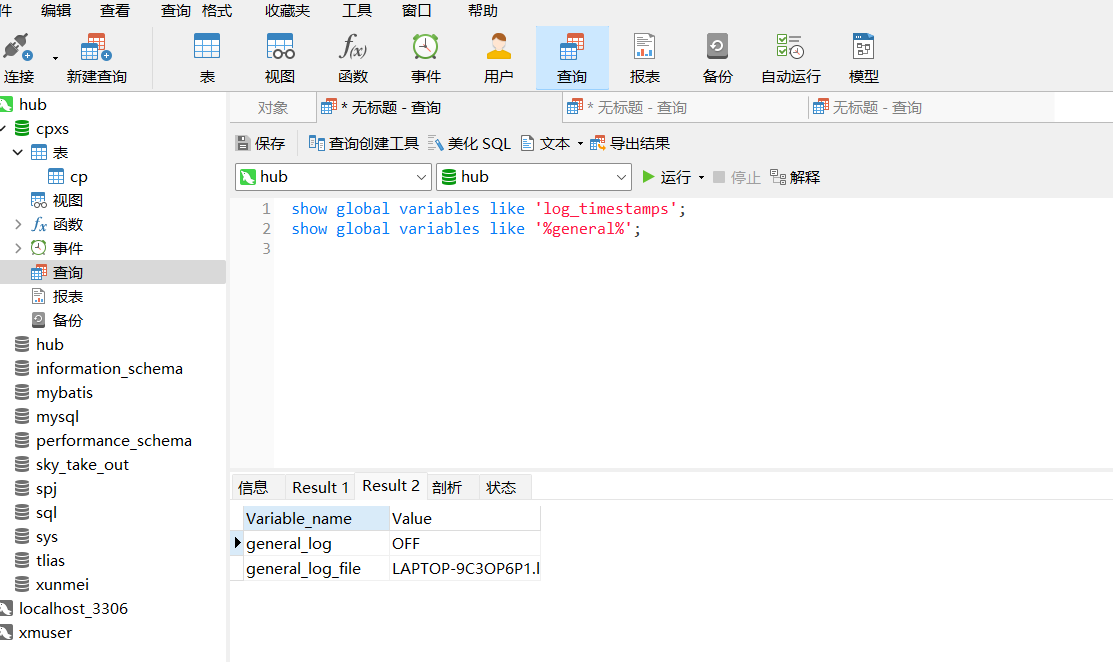
①显示当前审计开关状态。

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;

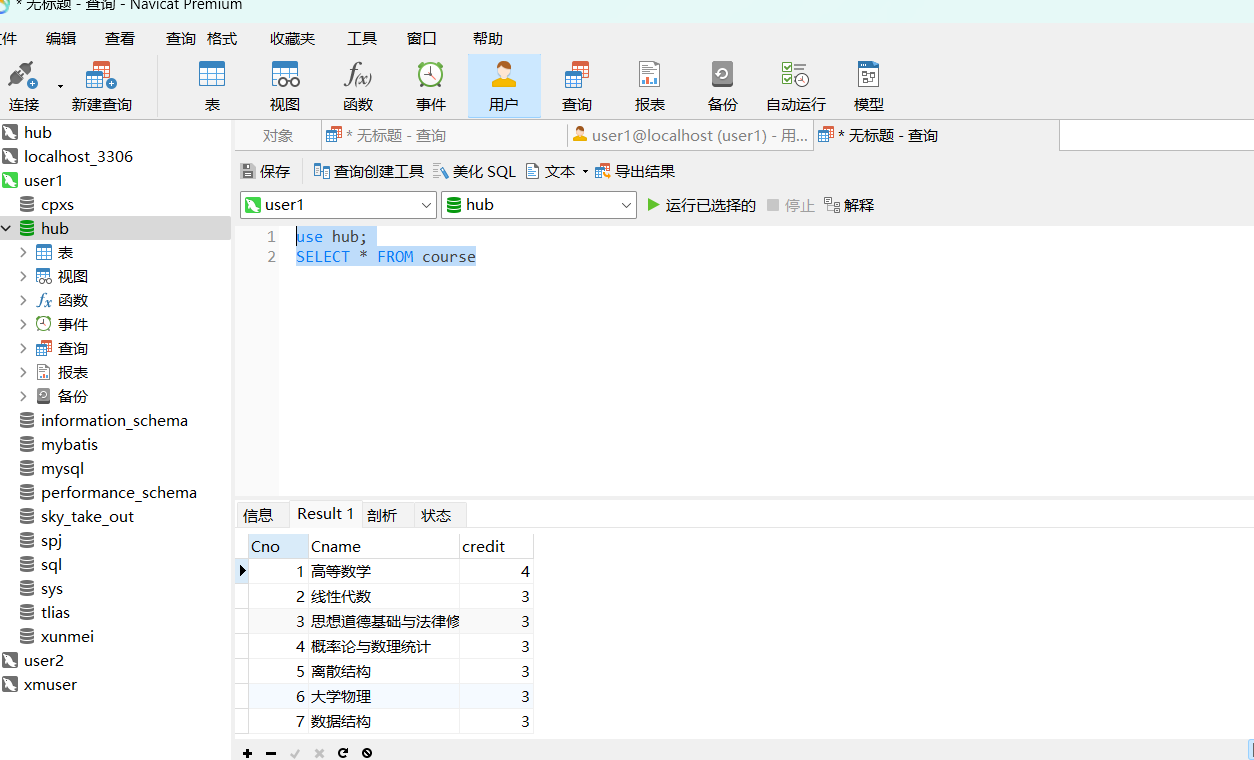


# 课后练习题

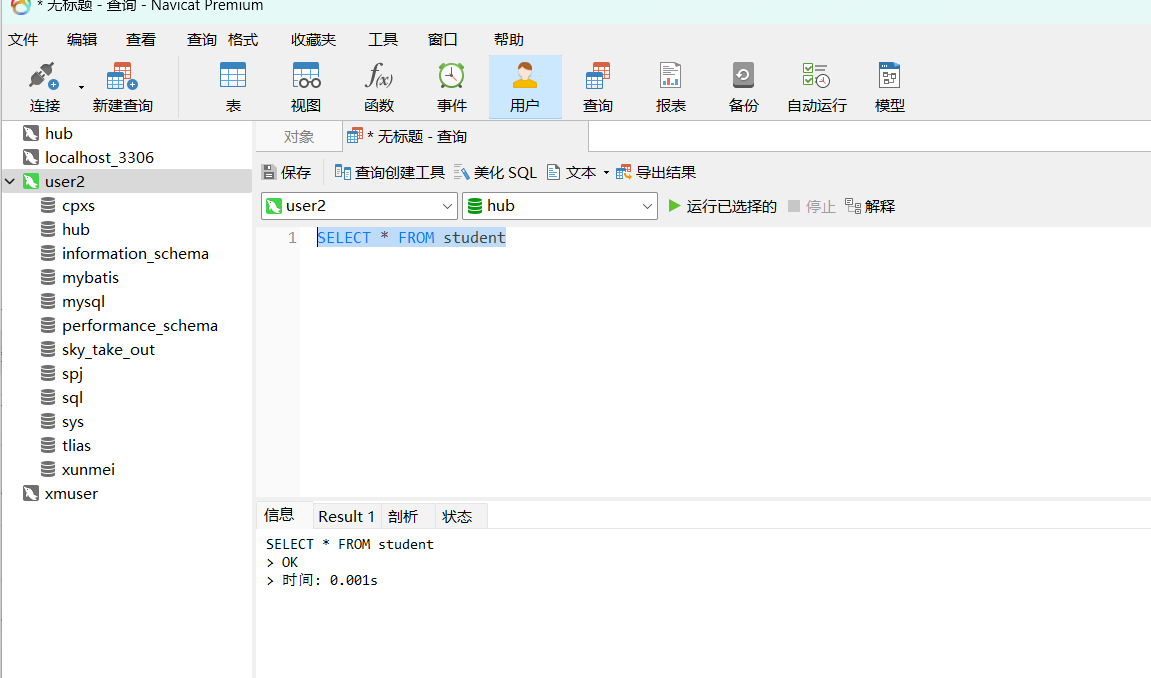
1、使用两个不同的用户通过Navicat和查询分析器查看hub数据库中student、course两个表的所有数据。

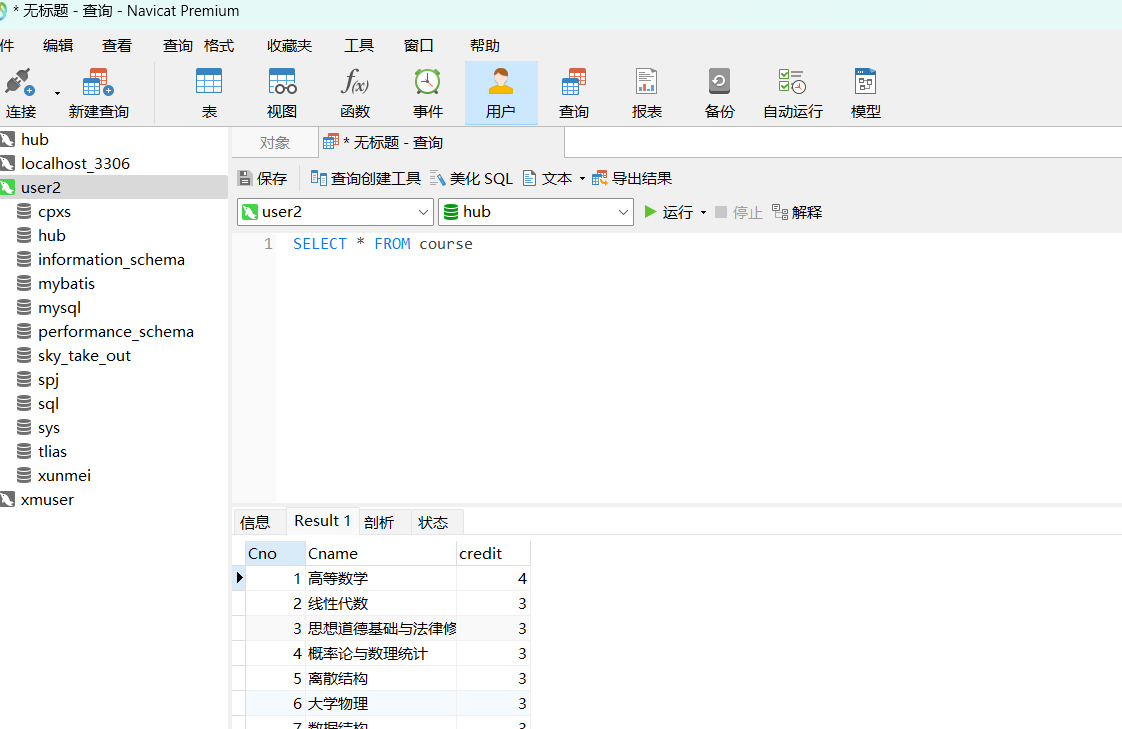
User1：





User2：

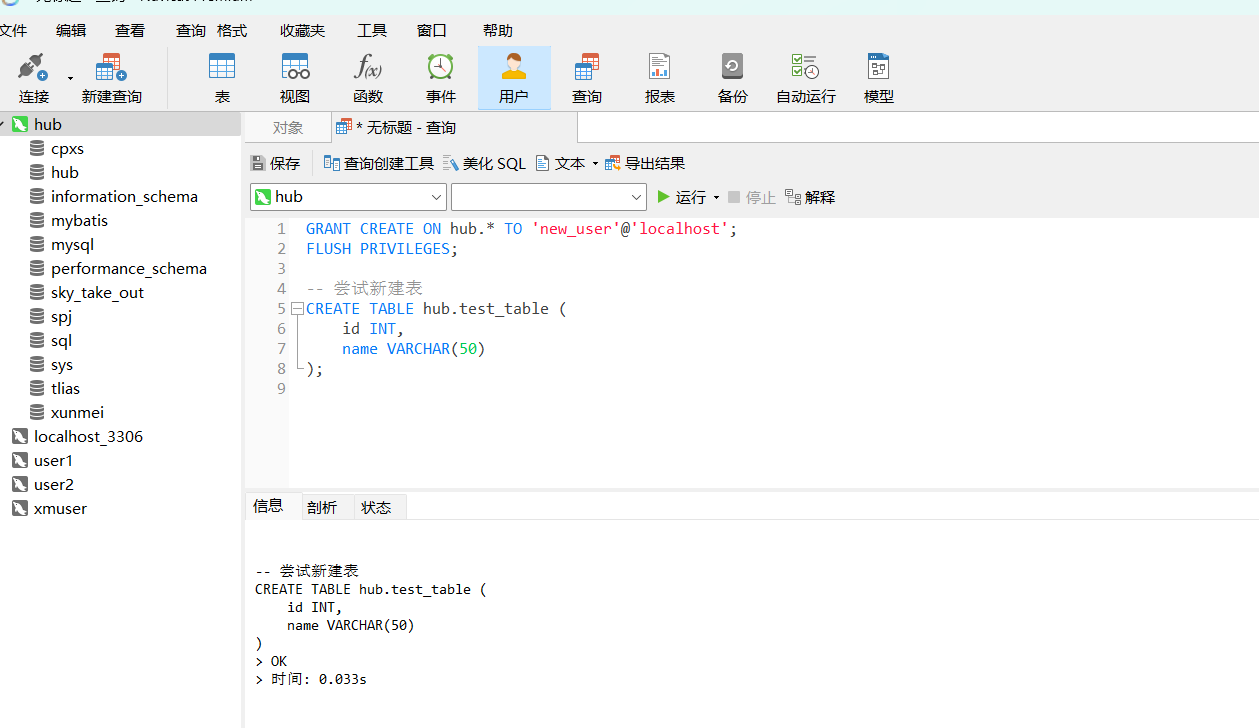




2、删除用户xmuser，尝试新建另一个用户来管理数据库、表等：

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）



# **四、思考题**

1、解释10个数据库操作权限的含义和影响。

· **SELECT**：允许查询数据，影响较小，但大规模查询可能影响性能。

· **INSERT**：允许插入数据，频繁插入可能增加I/O负担。

· **UPDATE**：允许修改数据，可能导致锁竞争，影响并发性能。

· **DELETE**：允许删除数据，可能影响数据恢复和引发触发器，增加I/O负担。

· **CREATE**：允许创建数据库对象，频繁创建可能影响性能。

· **DROP**：允许删除数据库对象，删除时可能影响数据库结构。

· **ALTER**：允许修改对象结构，可能导致锁定，影响性能。

· **INDEX**：允许创建删除索引，影响查询性能，但创建时可能造成锁定。

· **GRANT OPTION**：允许授予权限，滥用可能导致安全风险。

· **RELOAD**：允许重新加载配置，影响较小，但在数据库维护时需要使用。

1. 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。

背景： 数据库审计是监控和记录数据库活动的过程，旨在提高数据库的安全性，确保操作符合合规性要求。在现代企业中，数据库审计通常用于跟踪用户行为、修改日志、权限变更等。

例子： 假设在管理一个高并发的电子商务数据库系统，系统中存储着大量的订单数据和用户信息。为了增强安全性和合规性，启用了数据库审计功能来记录所有的 SELECT、INSERT、UPDATE 和 DELETE 操作。

审计方案设计：

审计级别： 启用所有用户的操作审计，包括读取（SELECT）、修改（UPDATE）、插入（INSERT）和删除（DELETE）数据的操作。

审计日志存储： 审计数据将记录到专门的审计日志表中，表结构包含操作类型、执行时间、操作者、被操作的表名、查询条件（对于SELECT）等信息。

审计频率： 审计操作会立即记录，且在系统日志表中存储所有相关信息。

对性能的影响分析：

I/O 负担：

审计日志记录： 每当用户进行查询或修改操作时，数据库不仅要完成主操作（如查询数据），还需要将这些操作写入审计日志表。这意味着每个操作都会增加额外的写入负担，尤其是在高并发情况下，这会显著增加磁盘 I/O 负载。

锁定与并发问题：

表锁与记录： 审计日志记录可能会导致额外的锁定，尤其是当审计日志表未进行适当分区或优化时，频繁的写入可能会阻塞其他操作，导致性能下降。在高并发的系统中，频繁的锁竞争会造成延迟，影响数据库的响应时间。

查询性能：

增加查询延迟： 启用审计会增加查询操作的延迟，尤其是当查询结果很大时。审计表的存在和数据插入操作可能会增加查询执行的时间，导致数据库响应变慢。

资源消耗：

CPU 和内存： 审计功能需要消耗额外的 CPU 和内存资源，尤其是在启用详细审计时（例如审计每个查询的详细信息）。这会影响数据库的总体性能，尤其是在资源有限的环境中。

日志表的增长：

审计日志表： 随着时间推移，审计日志表的大小会迅速增长，尤其是在大型数据库系统中。没有适当的日志轮换和清理机制，日志表可能会变得庞大，影响数据库的查询和存储性能。

数据恢复：

审计日志与数据恢复： 如果审计日志中包含重要的事务信息，当数据库发生故障时，恢复审计日志数据可能会影响恢复速度，增加恢复过程中的负担。

性能优化建议：

选择性审计： 只审计关键操作，例如高风险的 DELETE 和 UPDATE 操作，而不必记录每个查询。这样可以减少数据库的负担。

分区审计日志： 将审计日志表进行分区，按时间或操作类型分区存储，这样可以优化查询和存储性能。

异步审计： 如果可能，可以将审计记录的插入操作设计为异步操作，减少对主数据库操作的影响。

审计日志轮换： 定期清理过期的审计日志，避免日志表膨胀过大，影响数据库的存储和查询性能。

**实验总结：**

通过本次实验,我们深入了解了数据库安全的重要性以及相关的实践方法。

首先,我们学习了如何为数据库创建合理的权限管理体系,包括为不同角色分配对应的读写权限。这有助于防止未经授权的访问和操作,提高数据的机密性和完整性。

**教师评语及成绩**：