openGauss 安全体系创新

实践课



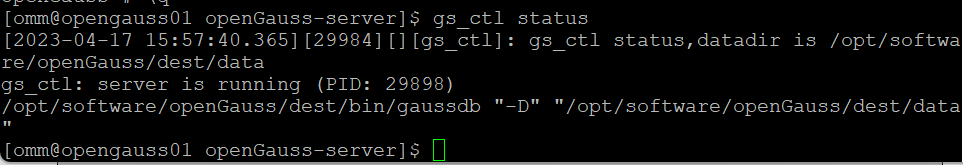
华为技术有限公司

# 关卡一、openGauss数据安装及基本操作

openGauss数据安装及基本操作, 作业提交任务如下：

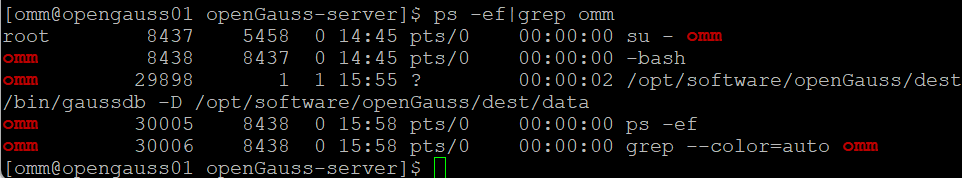
任务一：数据库状态验证

1. 查询数据库状态成功截图



任务二：数据库服务进程验证

1. 查看数据库服务进程截图（包含数据库服务器的主机名）



任务三：实践思考题

思考题1：为什么需要通过源码编译，安装数据库？

编译安装数据库可以带来以下几个好处：

定制化：通过编译安装可以选择需要的功能模块，而不是使用预编译好的通用版本。这样可以避免安装一些不需要的模块，减小软件包的体积，也可以提高系统的性能。

最新版：通过源代码编译安装可以获得最新的软件版本，而不必等待官方发行预编译的软件包。

灵活性：编译安装可以让用户更灵活地控制软件的配置。例如，可以设置和优化编译参数，或者安装所需的插件和扩展，从而满足特定的需求。

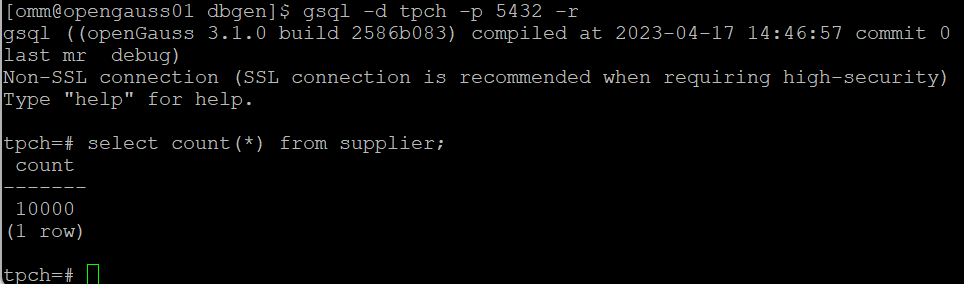
安全性：编译安装可以减少安全漏洞。因为编译过程中，可以加入更多的安全措施，避免一些安全问题。

# 关卡二、openGauss数据导入及行存列存

任务一：数据初始化验证

1. 查询supplier表的行数，并将结果进行图：

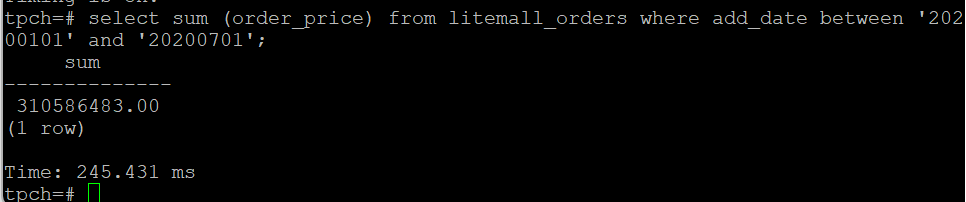
select count(\*) from supplier;;



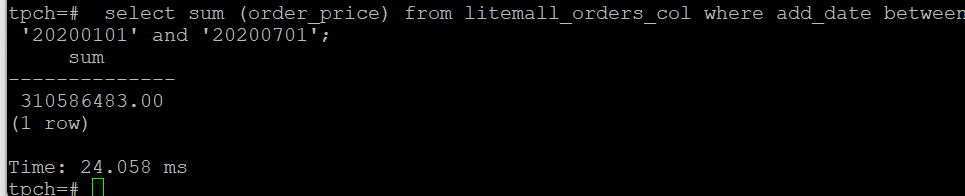
任务二：行存表与列存表执行效率对比

1. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的总和查询，并对比执行效率截图

select sum (order\_price) from litemall\_orders where add\_date between '20200101' and '20200701';

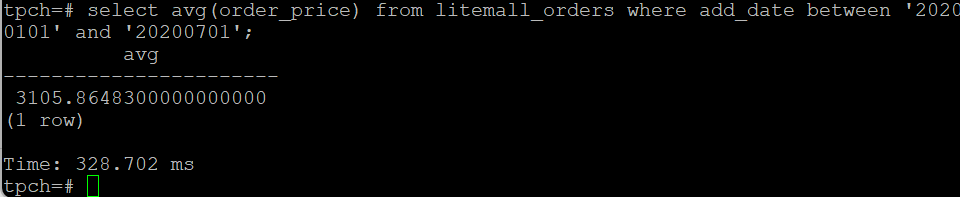


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

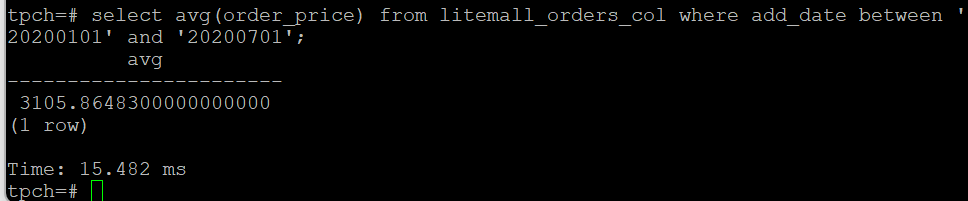


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

select avg (order\_price) from litemall\_orders where add\_date between '20200101' and '20200701';

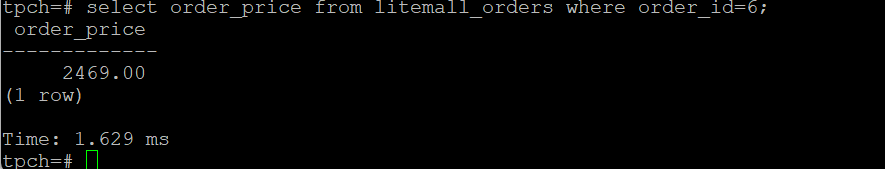


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

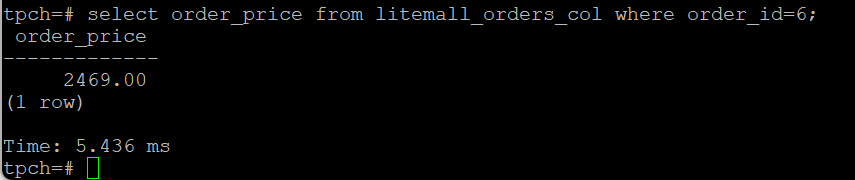


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

select order\_price from litemall\_orders where order\_id=6;

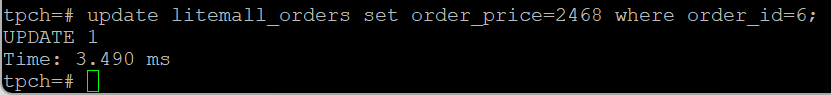


select order\_price from litemall\_orders\_col where order\_id=6;

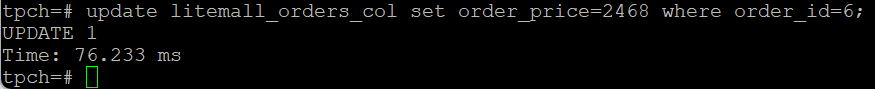


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

update litemall\_orders set order\_price=2468 where order\_id=6;



update litemall\_orders\_col set order\_price=2468 where order\_id=6;



任务三：实践思考题

思考题1：

行存表与列存表在执行相同的SQL语句时，为何执行的时间不同？

行存表将数据按照行的方式存储，即每一行都按照固定的数据类型和长度进行存储；而列存表则是将同一列的数据放在一起存储，即同一列的数据类型和长度是相同的。

当执行相同的SQL语句时，行存表需要在磁盘上读取整张表的数据，然后再对符合条件的行进行筛选，这样会涉及到大量的I/O操作，因此执行速度较慢。而列存表只需读取符合条件的列的数据，这样可以减少I/O操作的数量，从而执行速度较快。

另外，列存表还具有更好的压缩性能，可以大大减少存储空间的占用。但是，列存表也存在着一些局限性，例如更新操作的效率较低，适用场景相对较少。

思考题2：

在执行哪些类型SQL时，行存表效率更高？在执行哪些类型SQL时，列存表效率更高？

行存表适用于：

（1）经常需要修改、更新、删除数据。（2）需要获取整条记录。（3）需要进行复杂的SQL查询操作。

列存表适用于：

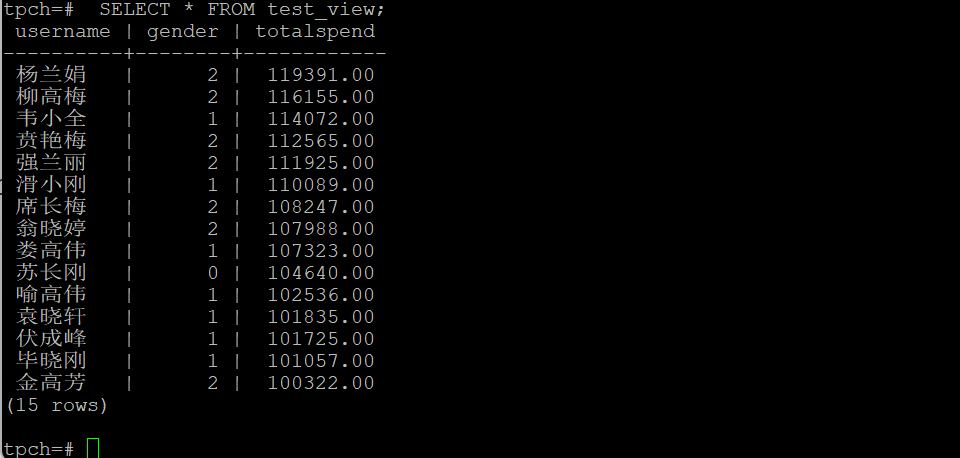
（1）需要大量读取单个或少数列数据。（2）存储超大型数据。（3）数据压缩比较高。

# 关卡三：openGauss物化视图应用

任务一：物化视图的使用

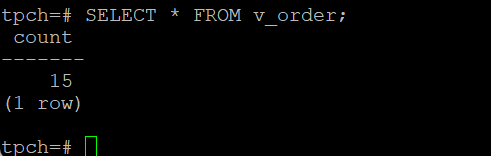
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



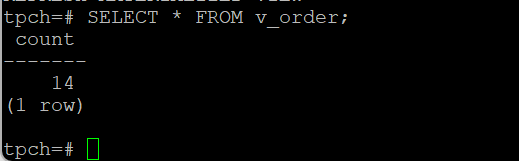
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



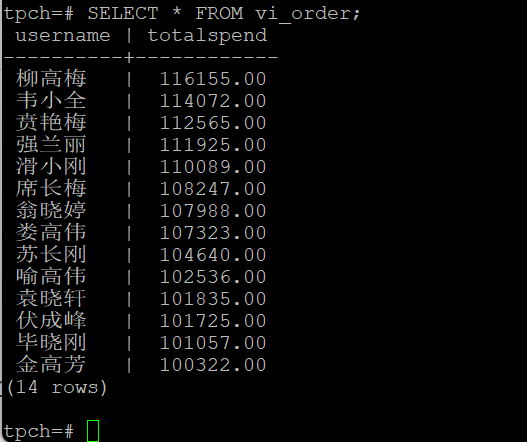
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



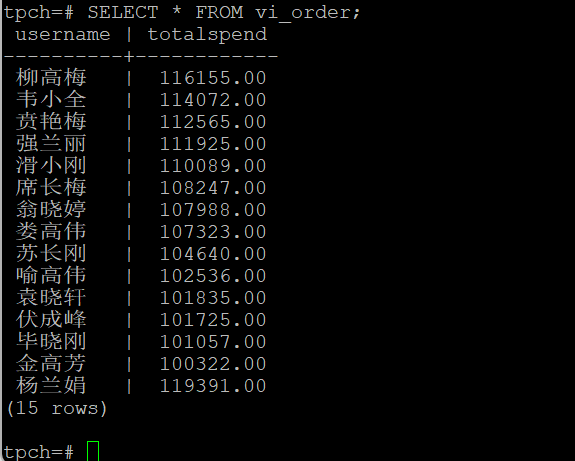
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM vi\_order;



5. 对表进行操作后，刷新增量物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM vi\_order;



任务二：实践思考题

思考题1：全量物化视图与增量物化视图有哪些差别？

全物化视图是将一个查询的结果集物化到一个表中，而增量物化视图是将查询的结果集物化到一个临时表中，当这个临时表被填满时，结果集会写入到一个物化表中。增量物化视图可以自动进行增量更新，因为它只需更新结果集而不是整个物化表。

思考题2：物化视图适用那些使用场景？

频繁查询的结果集比较稳定，且数据量比较大，可以通过物化视图来减少查询时间和资源消耗。

有多条SQL语句需要联合查询多个表，并且这些查询语句的结果集都较为相似，此时可以将这些查询语句的结果集合并为一个物化视图，以减少重复查询和提高性能。

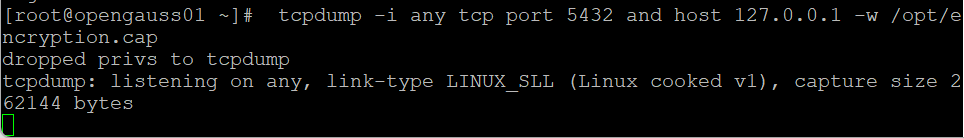
系统中需要经常使用复杂查询语句，但是每次查询都需要消耗大量的时间和计算资源，此时可以通过物化视图来缓存查询结果，提高查询效率。

对于数据分析、报表生成等业务场景，由于这些业务场景需要频繁查询大量的数据，因此可以通过物化视图来优化性能，提高查询速度。

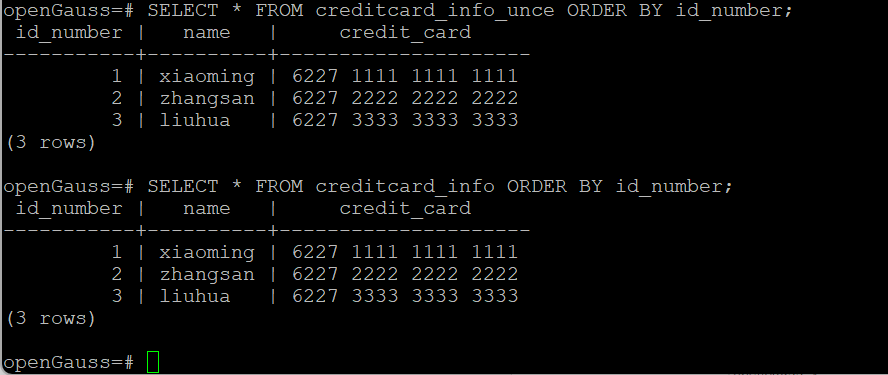
# 关卡四：openGauss密态数据库特性应用

任务一：物化视图的使用

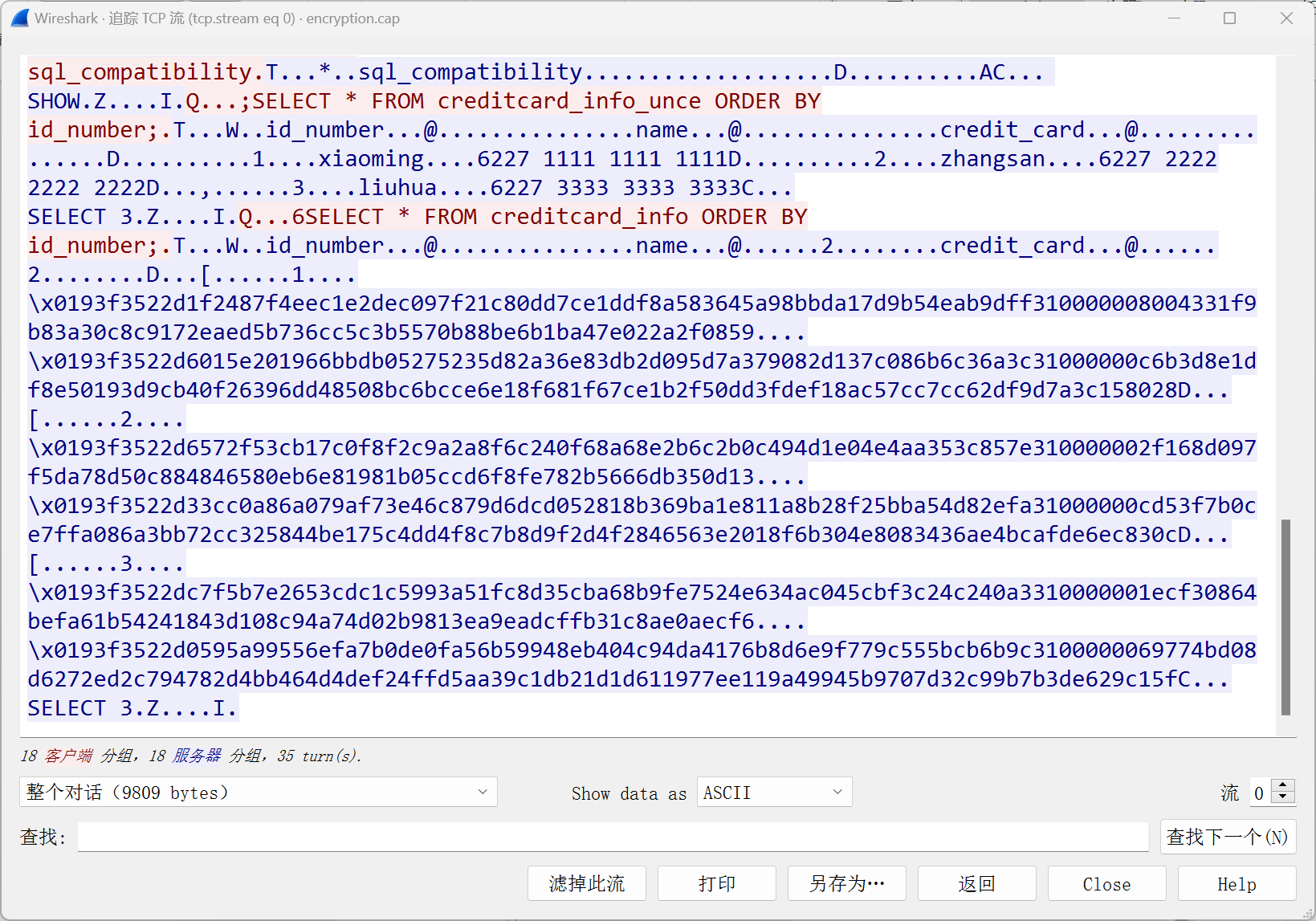
1. 通过tcpdump抓取数据流，此putty窗口暂时保持不动，将执行结果截图：



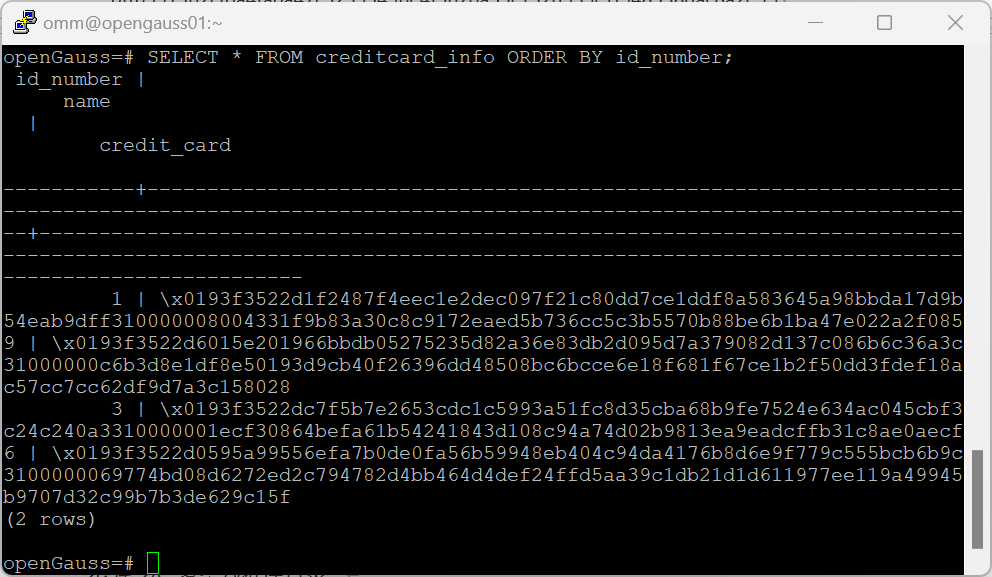
1. 将加密表和非加密表查询结果截图：



1. 用wireshark解析加密表和非加密表的差异时，非加密表name列和credit\_card列是明文，加密表name列和credit\_card列均是密文，将执行结果截图：



1. 查询加密表，查询到的结果为密文，将执行结果截图：



任务二：实践思考题

思考题1：

数据实际存储在物理磁盘上的时候是明文还是密文？数据的加解密的动作是在客户端完成的还是服务端完成的？

明文，即未加密的原始数据。

数据的加解密的动作可以在客户端或服务端完成，具体取决于数据传输安全的需求和实现方式。

如果数据传输过程中需要保证数据的机密性和完整性，那么可以在客户端进行加密，将加密后的数据发送到服务端，服务端再进行解密。这样可以有效地避免在网络传输过程中被第三方篡改或窃听。

但是，如果客户端的计算资源比较有限，或者需要考虑多个客户端的数据一致性，那么可以在服务端进行加解密。这样可以保证数据的安全性和一致性，但同时也会增加服务端的计算负担。