openGauss 安全体系创新

实践课



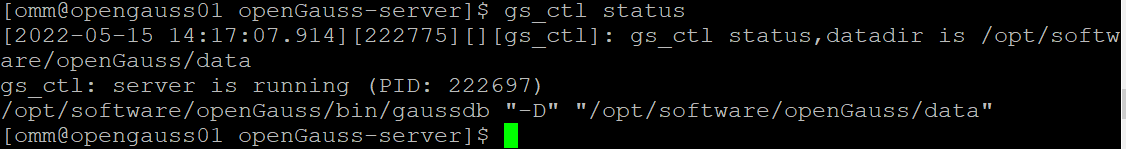
华为技术有限公司

# 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：

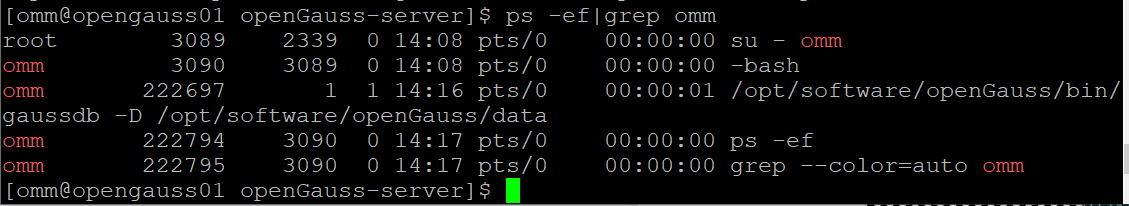
任务一：数据库状态验证

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？

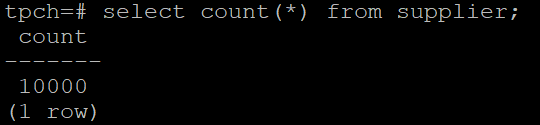
因为命令行操作中没有图形用户界面来执行安装向导程序，并且通过源码编译安装更加直接高效。

# 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：

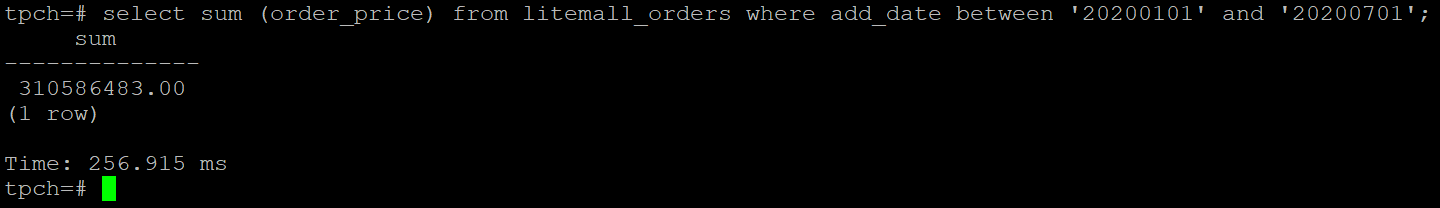
select count(\*) from supplier;;



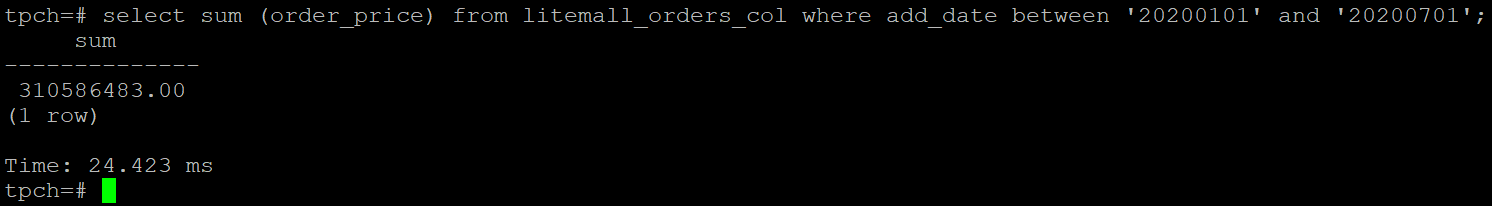
任务二：行存表与列存表执行效率对比

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';

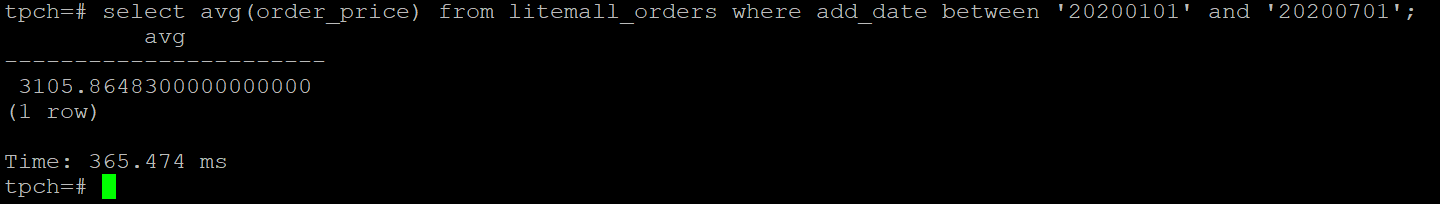


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

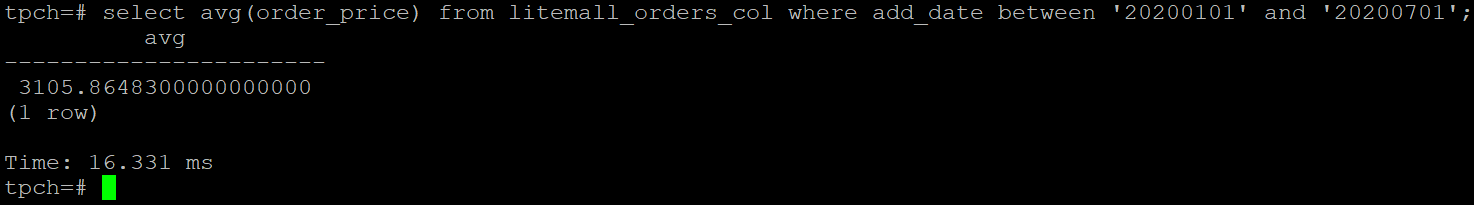


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

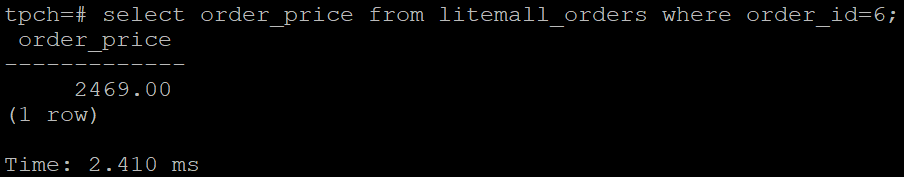


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

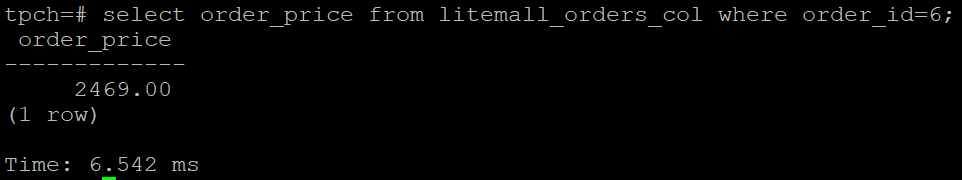


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

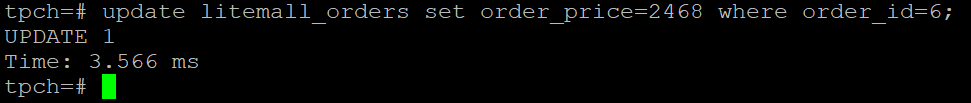


select order\_price from litemall\_orders\_col where order\_id=6;

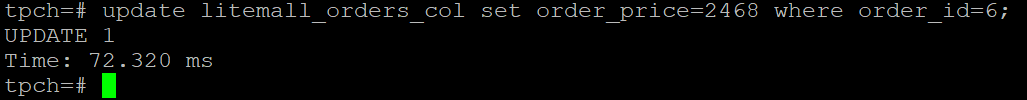


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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？

行存表在存数据的时候是先存完一个元组，再存下一个元组；列存表在存数据的时候是先存完一个列的数据，再存下一个列的数据。

因此在对某一行数据执行操作的时候，行存表的指针只需要找到这一行所在的位置并在邻近位置移动即可完成数据操作，而列存表的指针要在多个列之间跳转，跳转花费的时间更多。因此对某一行的数据执行操作的时候，行存表的执行效率更高。

对某一列的数据的数据执行操作的时候，列存表的指针只需要找到某一列的位置，然后依次移动指针即可完成操作；而行存表在对某一列执行操作的时候，指针需要在所有行之间跳转，用时会更长。因此对某一列的数据执行操作的时候，列存表的效率更高。

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？

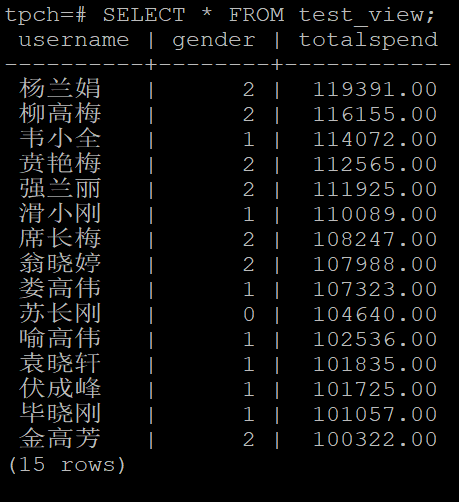
对某一列的数据进行操作的时候（比如对某列求和、求平均值），采用列存的表执行效率更高；而对某一行的数据进行操作的时候（比如查看某一行、修改某一行），采用行存的表执行效率更高。

# 关卡三：openGauss物化视图应用

任务一：物化视图的使用

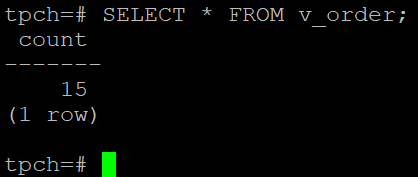
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



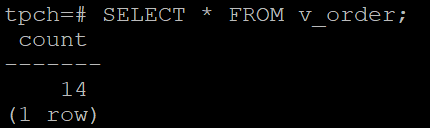
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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？

全量物化视图仅支持对已创建的物化视图进行全量更新，而不支持进行增量更新；

增量物化视图可以对物化视图增量刷新，需要用户手动执行语句完成增量数据刷新。

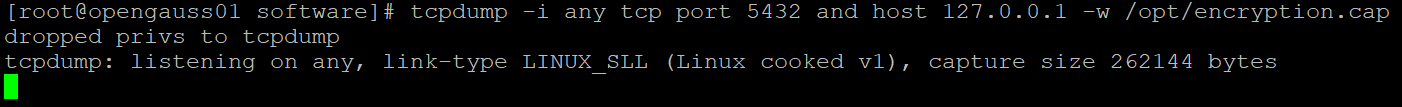
思考题2：物化视图适用那些使用场景？

物化视图适用于报表统计、大表统计等，定期固化数据快照，避免对多表重复跑相同的查询。

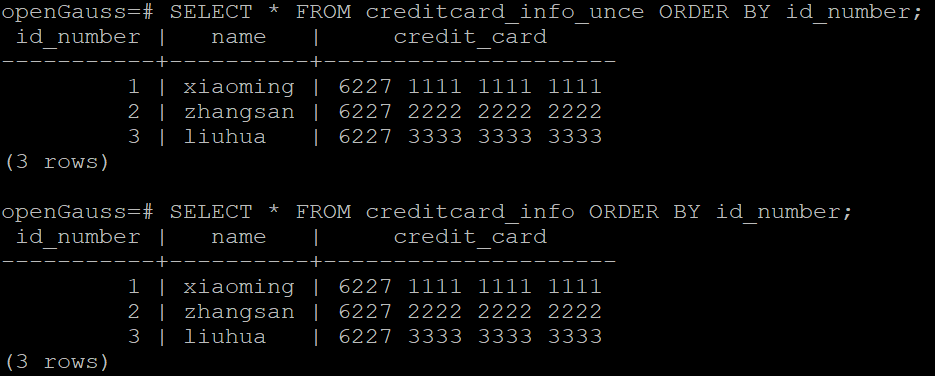
# 关卡四：openGauss密态数据库特性应用

任务一：物化视图的使用

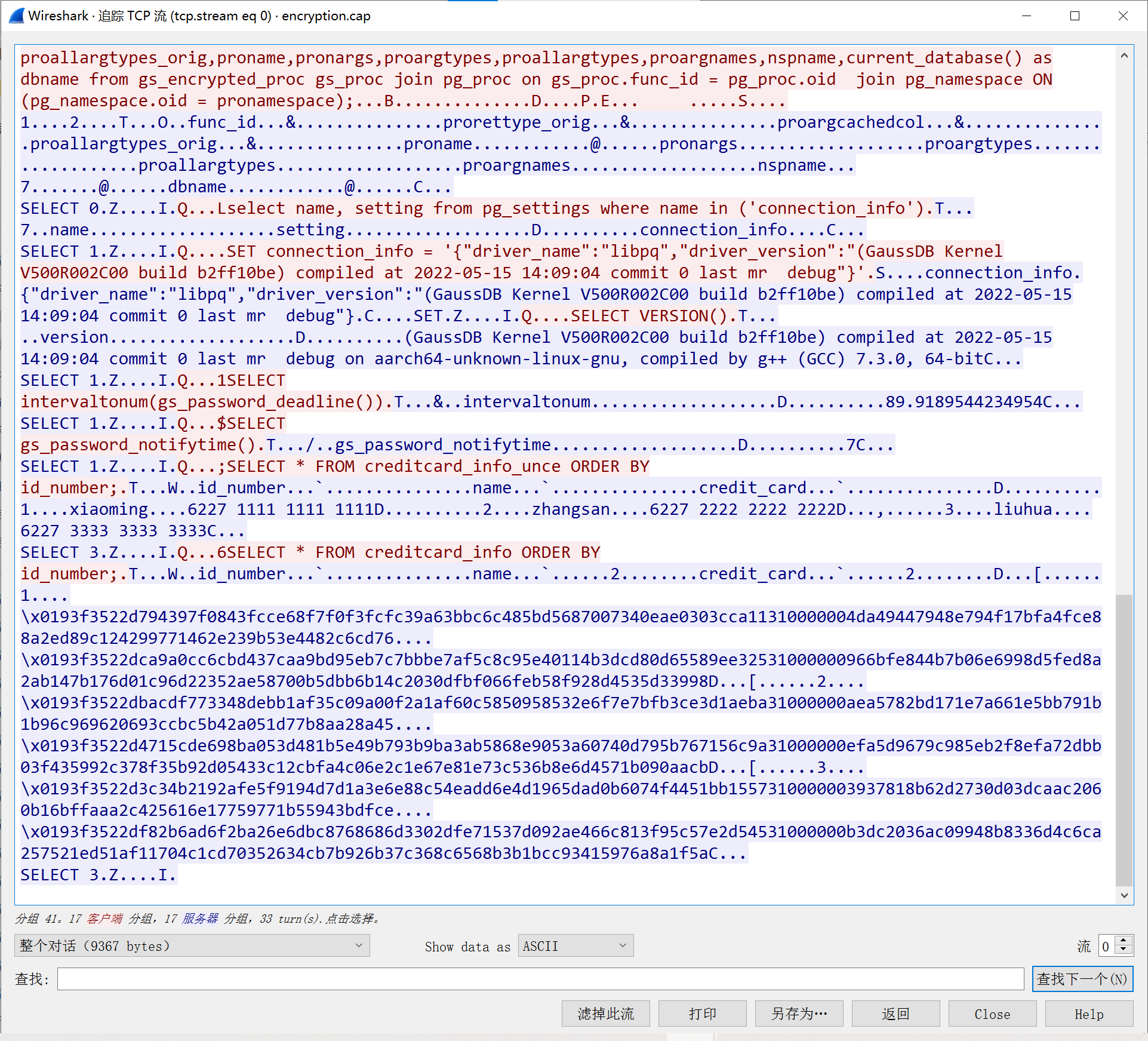
1. 通过tcpdump抓取数据流，此putty窗口暂时保持不动，将执行结果截图：



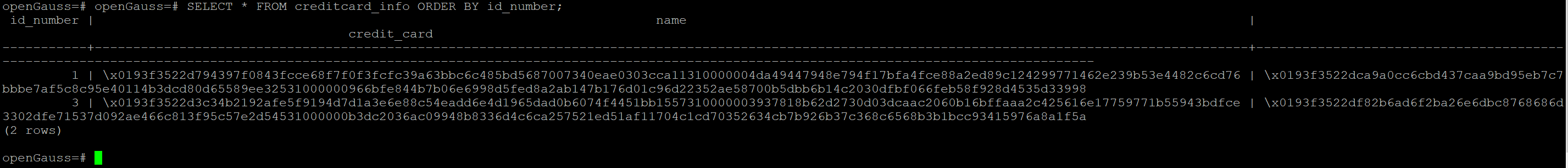
1. 将加密表和非加密表查询结果截图：



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？

数据实际存储在物理磁盘上的时候是密文。

数据的加解密动作是在客户端完成的。