openGauss 安全体系创新

实践课



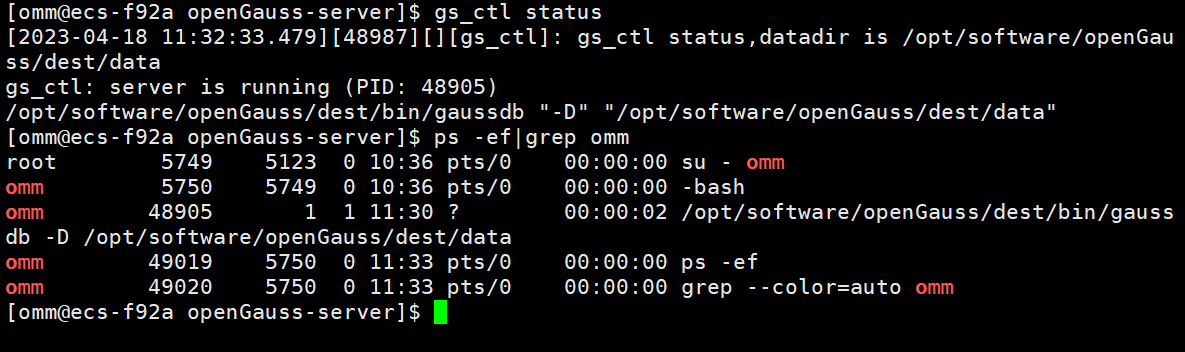
华为技术有限公司

# 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：

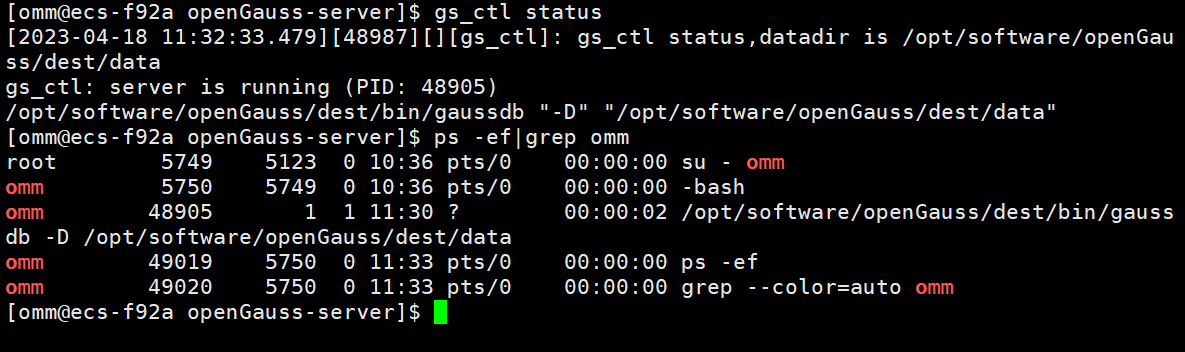
任务一：数据库状态验证

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？

版本问题：发行版提供的数据库软件版本可能不是最新的，或者不是需要的特定版本。在这种情况下，编译源代码是获得所需版本的唯一方法。

自定义选项：使用发行版提供的软件包时，通常无法使用自定义编译选项，例如启用或禁用某些特性，或者将数据库与特定的外部库链接。通过编译源代码，可以使用各种自定义选项来满足特定需求。

发行版不支持：有时，可能使用的Linux发行版不支持所需的数据库软件，或者没有提供相应的软件包。在这种情况下，编译源代码是安装所需软件的唯一方法。

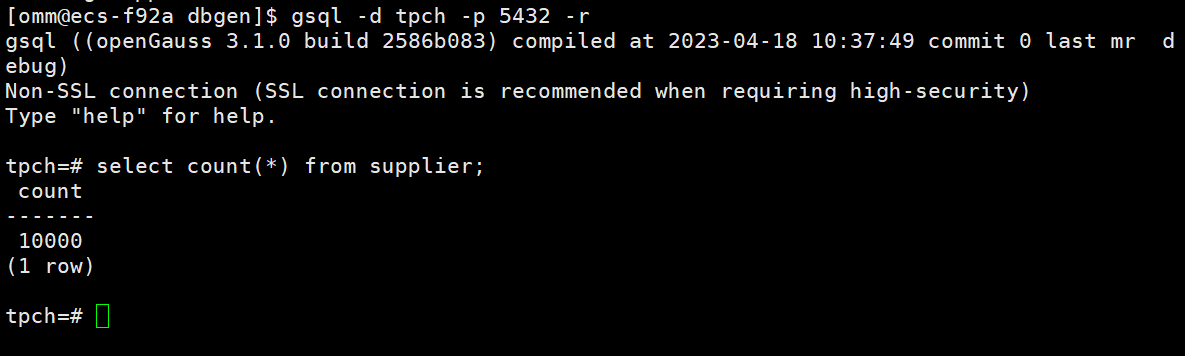
通过源代码编译可以为数据库软件提供更大的灵活性和自定义选项，从而更好地满足特定需求。

# 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：

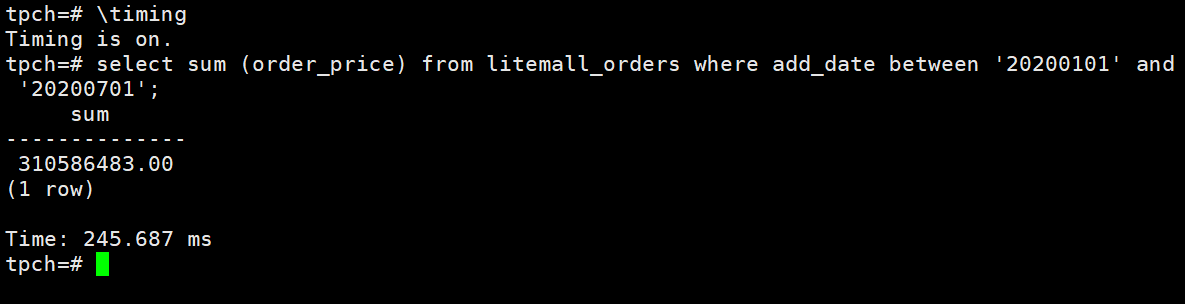
select count(\*) from supplier;;



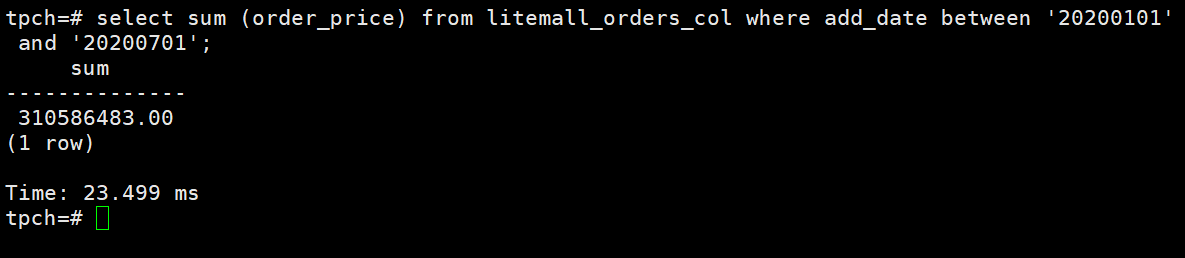
任务二：行存表与列存表执行效率对比

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';

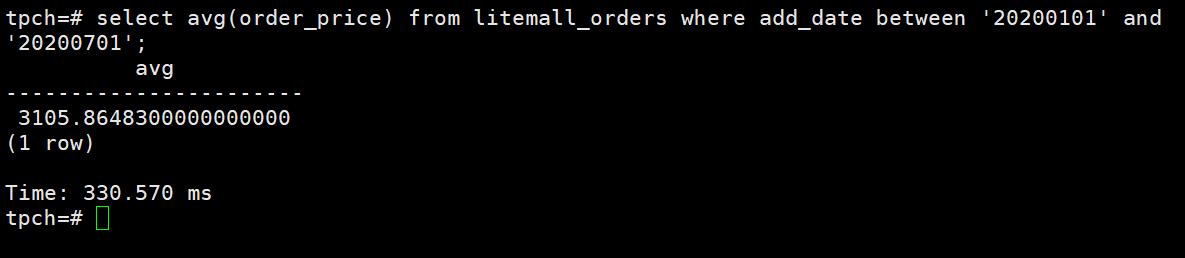


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

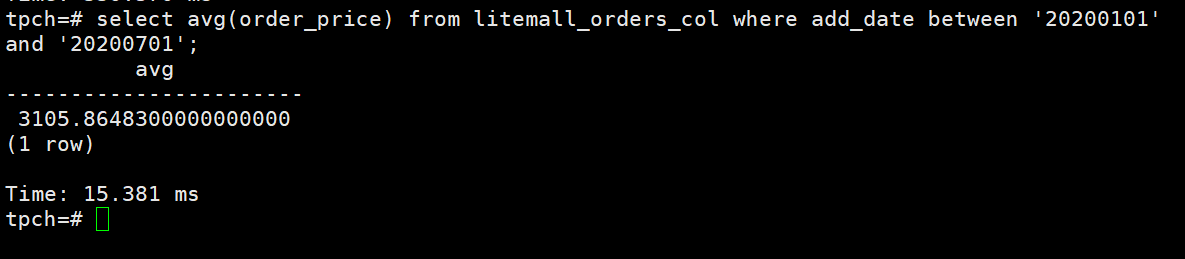


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

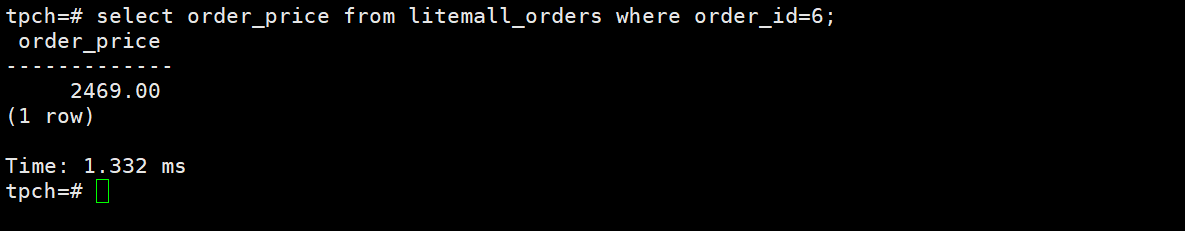


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

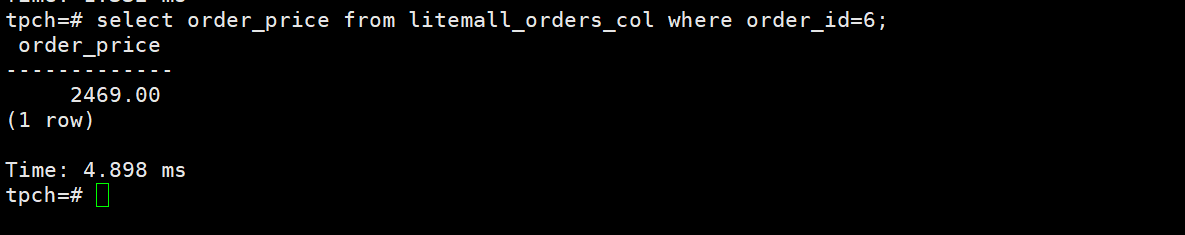


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

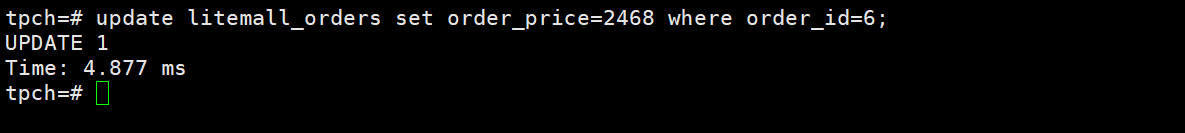


select order\_price from litemall\_orders\_col where order\_id=6;

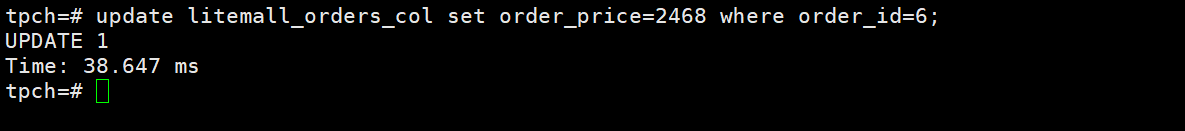


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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？

行存表存储方式将每一行数据作为一个记录存储在磁盘上，每一行数据包含表中所有列的信息。当执行查询语句时，需要读取每个记录的所有列信息，因此可能需要读取大量的数据，导致查询速度变慢。

而列存表存储方式将每一列数据存储在一起，这意味着查询只需要读取需要的列数据，而不必读取整行记录，从而减少了磁盘I/O的数量。此外，列存表通常使用一些高级压缩算法，可以在不丧失查询性能的情况下减少存储空间，从而进一步提高查询性能。

因此，当执行相同的SQL查询语句时，列存表的查询速度通常比行存表更快。不过，在某些情况下，行存表可能更适合某些查询模式，例如需要查询完整记录或者进行行级别的更新操作时。需要根据具体的应用场景来选择存储方式。

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？

行存表和列存表在处理不同类型的SQL查询时，可能会表现出不同的效率优势。

对于行存表，由于每个记录包含所有列信息，因此对于需要查询完整记录或者进行行级别的更新操作的SQL查询，行存表的效率通常比列存表更高。以下是几个行存表效率更高的SQL查询类型：

基于主键或唯一索引的查询：行存表可以通过索引快速定位到满足条件的记录，查询效率较高。

查询部分列：当查询只需要部分列的数据时，行存表可以直接从记录中读取所需列的数据，效率较高。

更新操作：当进行行级别的更新操作时，行存表可以直接修改记录中的数据，效率较高。

对于列存表，由于每个列的数据都存储在一起，因此在进行需要查询多个列的SQL查询时，列存表的效率通常比行存表更高。以下是几个列存表效率更高的SQL查询类型：

聚合查询：对于需要进行聚合操作的SQL查询（如SUM、AVG、COUNT等），列存表可以只读取需要的列数据进行计算，效率较高。

查询部分行：当查询只需要某些行的数据时，列存表可以通过跳过不需要的列数据，减少磁盘I/O次数，从而提高查询效率。

高效压缩：列存表通常使用高级压缩算法，在不降低查询性能的前提下减少存储空间，从而提高查询效率。

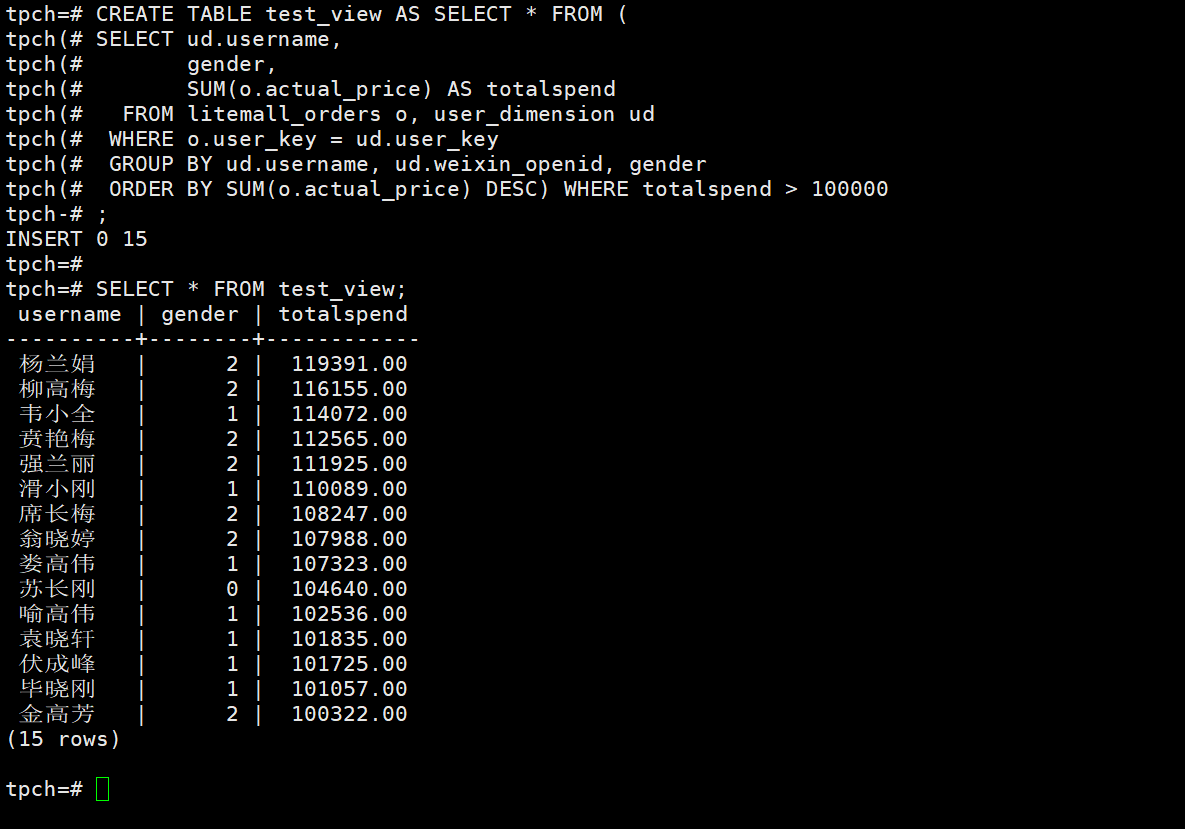
需要注意的是，这些只是一些常见的SQL查询类型，实际应用中具体的查询类型会因数据模式、查询条件等因素而有所不同。在选择行存表或列存表存储方式时，需要根据具体的应用场景和查询需求综合考虑。

# 关卡三：openGauss物化视图应用

任务一：物化视图的使用

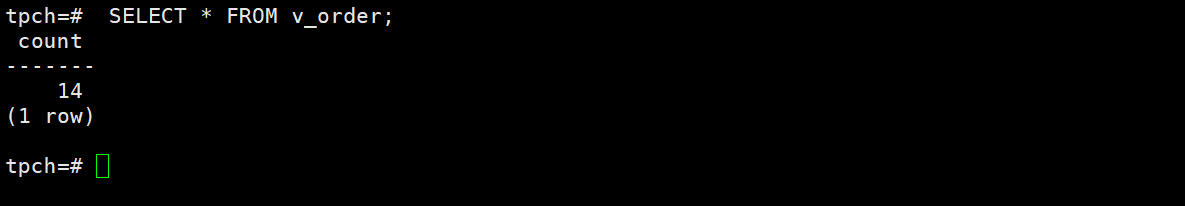
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



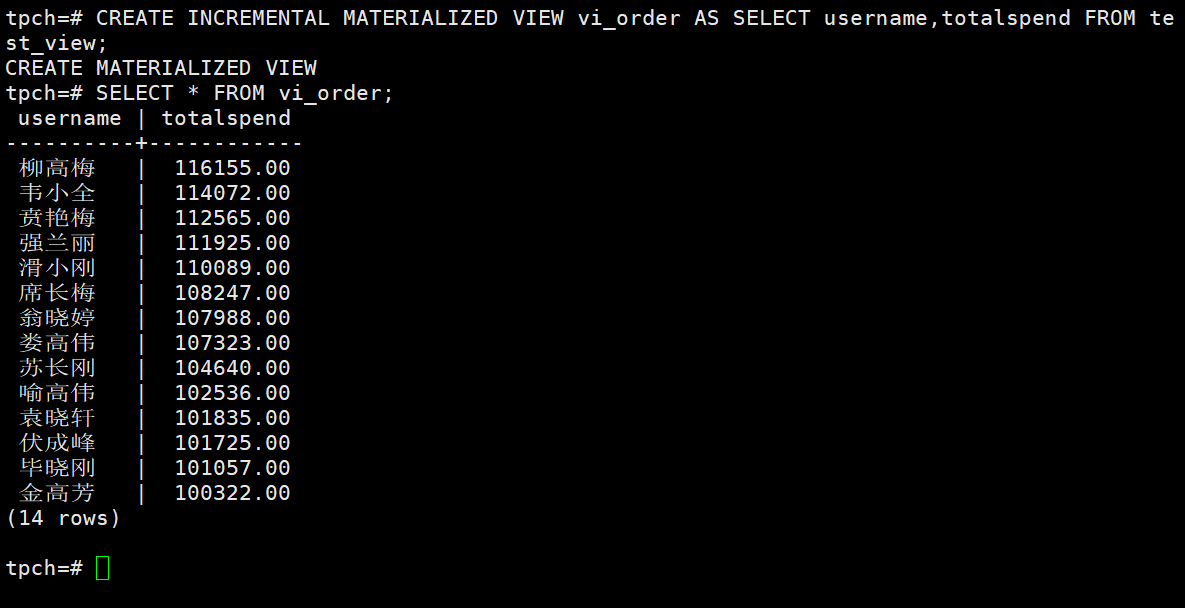
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



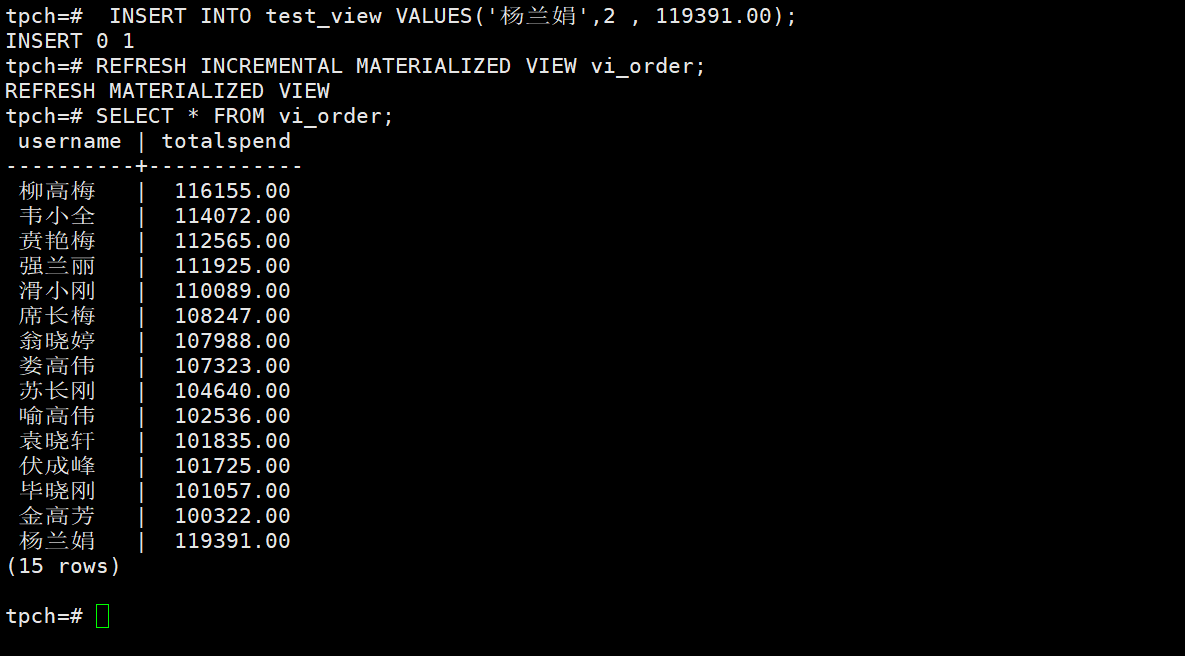
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



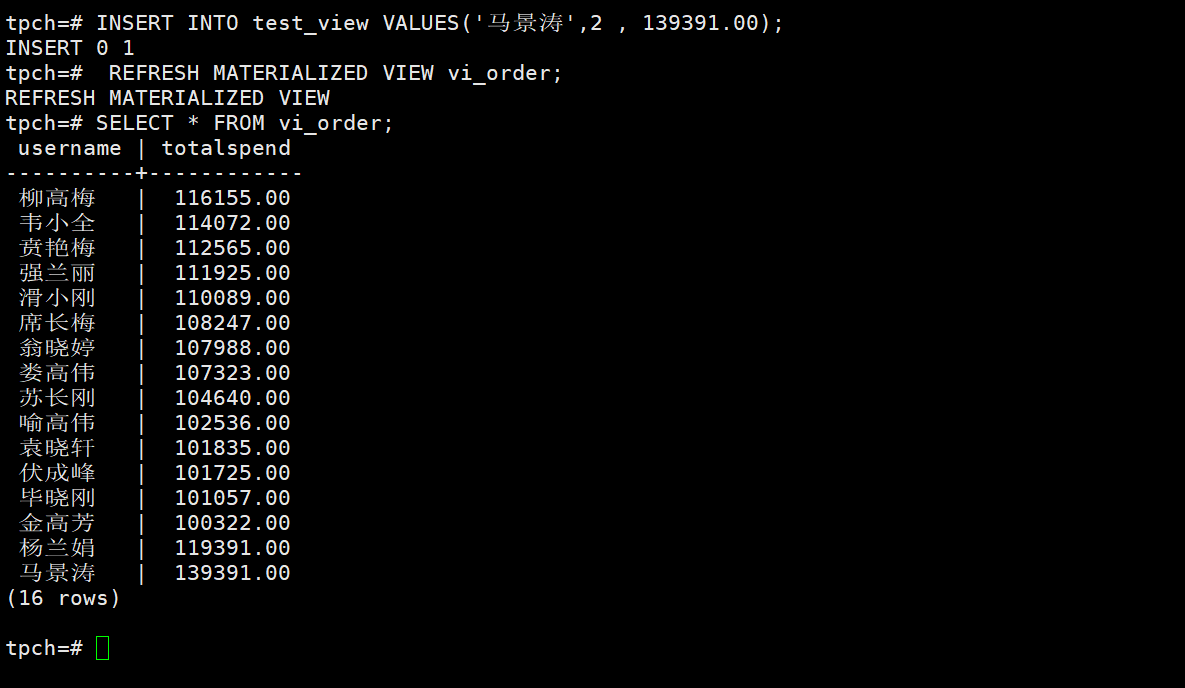
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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？

全量物化视图和增量物化视图是两种不同类型的物化视图，它们在维护和使用上有一些差别。

全量物化视图是指完全基于源表数据生成的物化视图。在创建全量物化视图时，系统会根据视图定义和查询语句从源表中提取数据并将其存储在物化视图中。全量物化视图通常适用于源表数据更新较少的情况下使用，因为每次更新都需要重新生成整个物化视图，这可能会带来较大的开销。

增量物化视图是指在全量物化视图的基础上，利用触发器或日志等机制，对源表的更新进行增量同步的物化视图。增量物化视图只对源表进行增量更新，避免了重复计算和重复存储已有的数据，因此相比于全量物化视图，它的维护和查询效率更高。增量物化视图通常适用于源表数据更新较为频繁的情况下使用，因为它只需要更新增量部分，可以避免全量物化视图每次都重新生成的开销。

全量物化视图和增量物化视图的差别在于是否基于触发器或日志等机制进行增量同步。全量物化视图适用于源表数据更新较少的情况下使用，而增量物化视图则适用于源表数据更新较为频繁的情况下使用。在实际使用时，需要根据具体的业务需求和数据特点进行选择。

思考题2：物化视图适用那些使用场景？

物化视图适用于一些需要经常查询、数据量较大、查询结果较为固定的场景。通过预先计算、缓存视图结果，可以提高查询效率，减少重复计算和存储开销。

具体来说，物化视图适用于以下场景：

频繁查询的复杂计算。当需要频繁查询某个复杂计算的结果时，可以使用物化视图将计算结果预先计算并缓存，以提高查询效率。

数据汇总和聚合。当需要对大量数据进行汇总和聚合时，可以使用物化视图将汇总和聚合结果预先计算并缓存，以提高查询效率。

查询结果相对稳定。当查询结果相对稳定，即数据更新频率较低时，可以使用物化视图将结果预先计算并缓存，以避免每次查询都需要重新计算的开销。

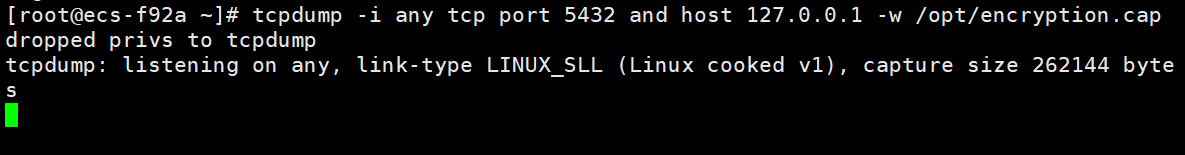
数据分析和报表。当需要对大量数据进行数据分析和生成报表时，可以使用物化视图将数据预先计算并缓存，以提高数据处理和报表生成的效率。

物化视图适用于一些需要频繁查询的复杂计算、数据汇总和聚合、查询结果相对稳定以及数据分析和报表等场景。在使用物化视图时，需要根据具体的业务需求和数据特点进行选择，并注意物化视图的维护和更新。

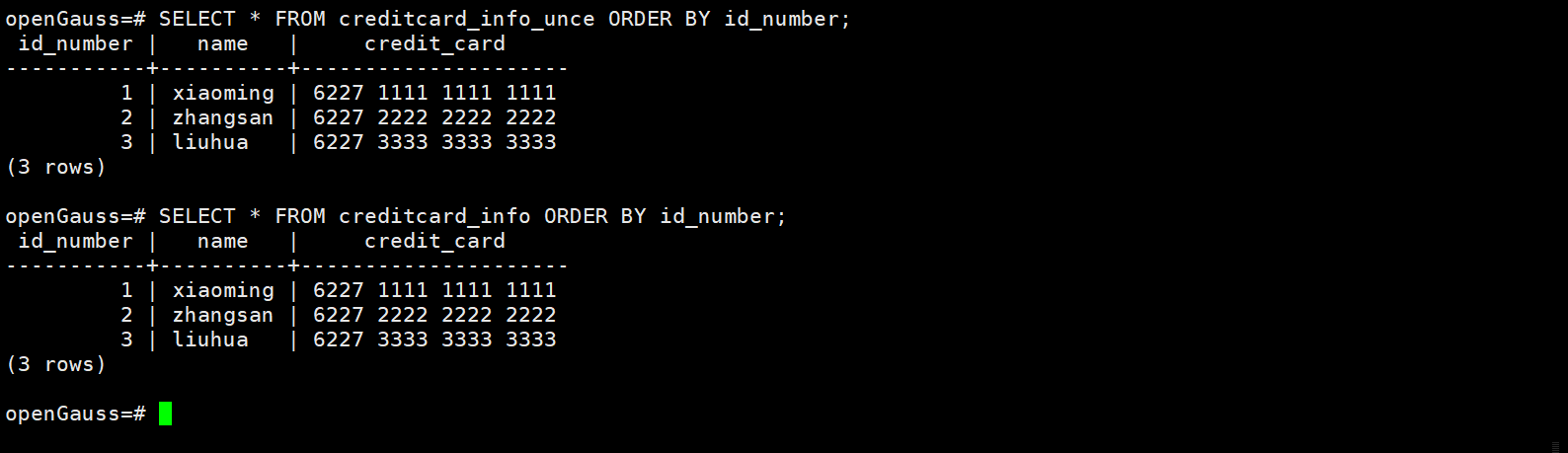
# 关卡四：openGauss密态数据库特性应用

任务一：物化视图的使用

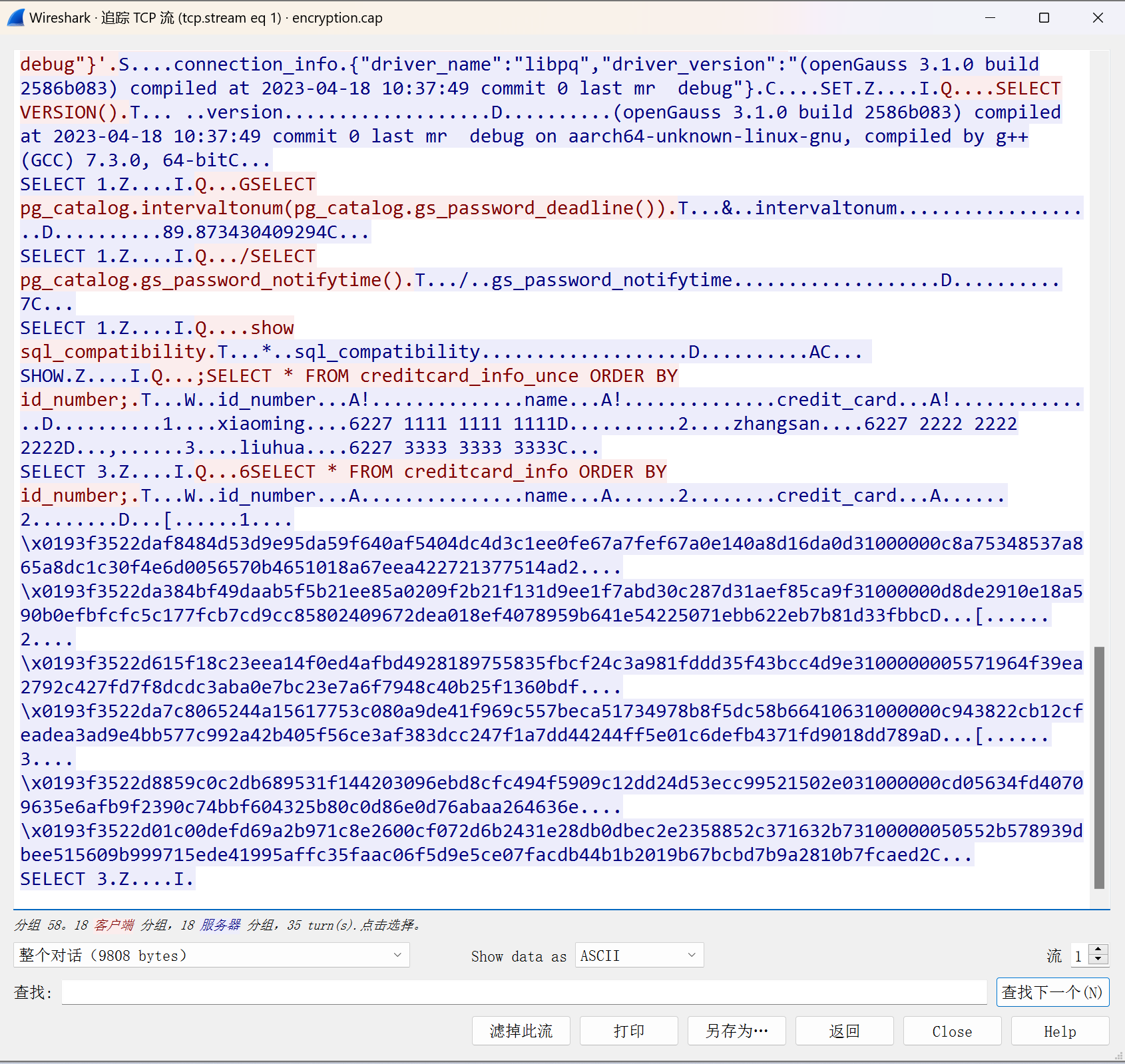
1. 通过tcpdump抓取数据流，此putty窗口暂时保持不动，将执行结果截图：



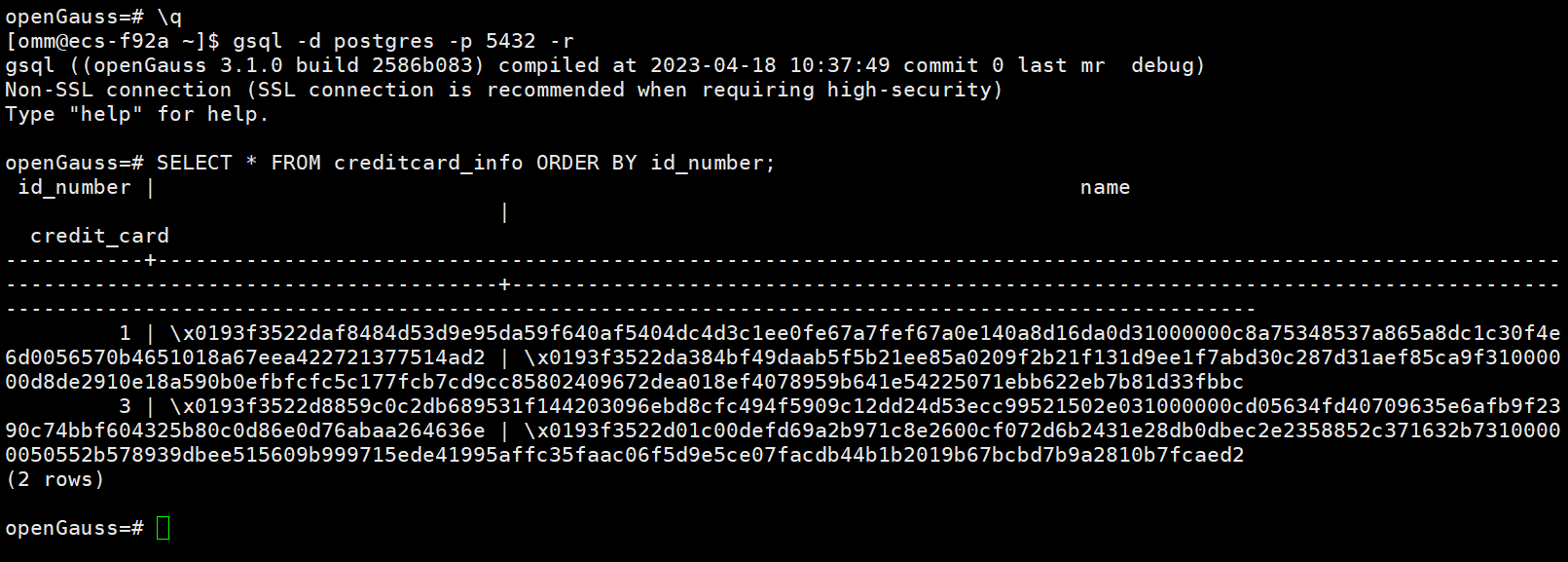
1. 将加密表和非加密表查询结果截图：



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时通常是密文形式，而不是明文形式，为了保护数据的安全性，通常会采用加密算法对数据进行加密存储。

数据的加解密动作通常是在服务端完成的。具体来说，当客户端向服务端发送加密数据时，服务端会接收数据并进行解密操作，然后再将解密后的数据存储在物理磁盘上。当客户端需要获取数据时，服务端会从物理磁盘上读取加密数据，并进行加密操作后将数据返回给客户端。

在一些特殊情况下，比如需要在客户端进行数据加解密的场景，可以通过在客户端使用加密库或者加密算法来实现。但通常情况下，为了保护数据的安全性，数据的加解密操作都应该由服务端完成。