openGauss 安全体系创新

实践课



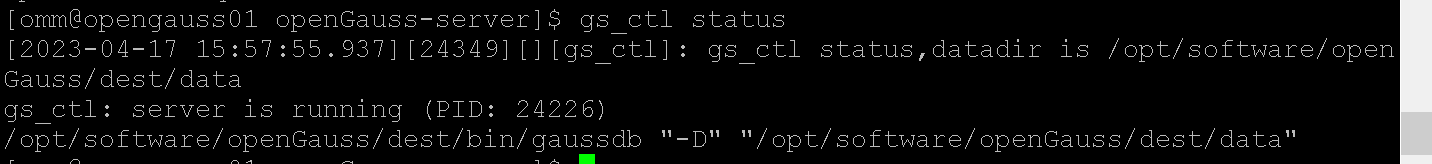
华为技术有限公司

# 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？

通过源码编译可以定制化OpenGauss数据库的安装，根据实际需求选择需要的功能，同时也能够适应不同的操作系统和硬件平台。另外，在某些情况下，由于软件或硬件限制，使用二进制版本可能无法满足要求，而通过源码编译则可以解决这个问题。

另一方面，通过源码编译还能够带来更好的性能和稳定性。源码编译可以根据当前的硬件环境和操作系统优化生成可执行文件，从而使得OpenGauss数据库在特定硬件平台和操作系统上运行更加高效和稳定。此外，通过源码编译还可以方便地进行调试和问题排查，因为可以直接访问源代码并进行改动和调试。

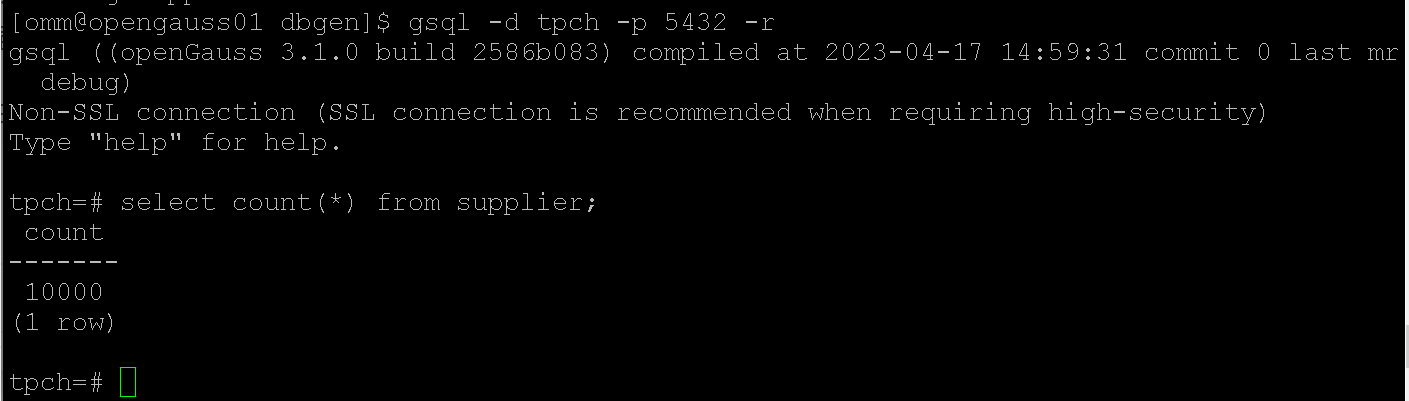
综上所述，通过源码编译和安装OpenGauss数据库可以更好地满足用户需求，提高性能和稳定性，并方便问题排查和调试。

# 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：

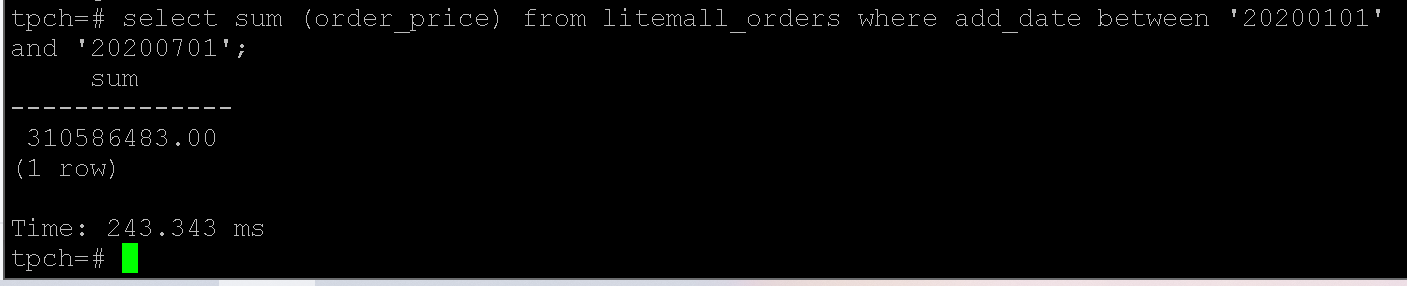
select count(\*) from supplier;;



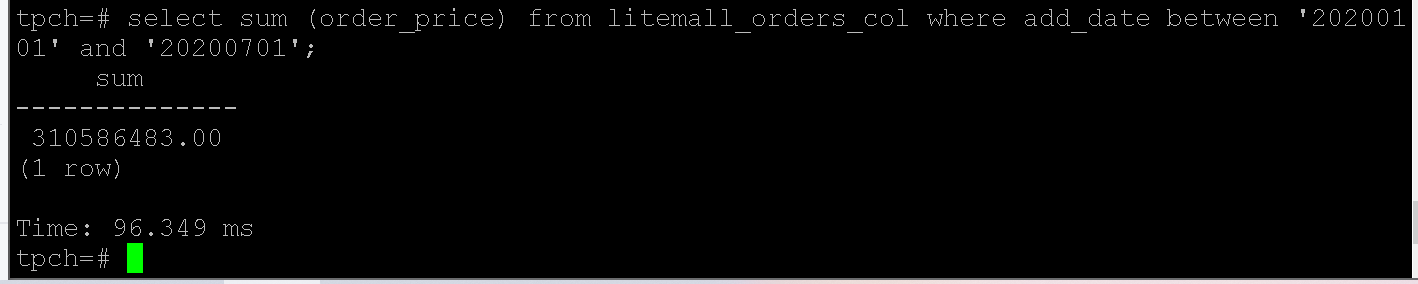
任务二：行存表与列存表执行效率对比

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';

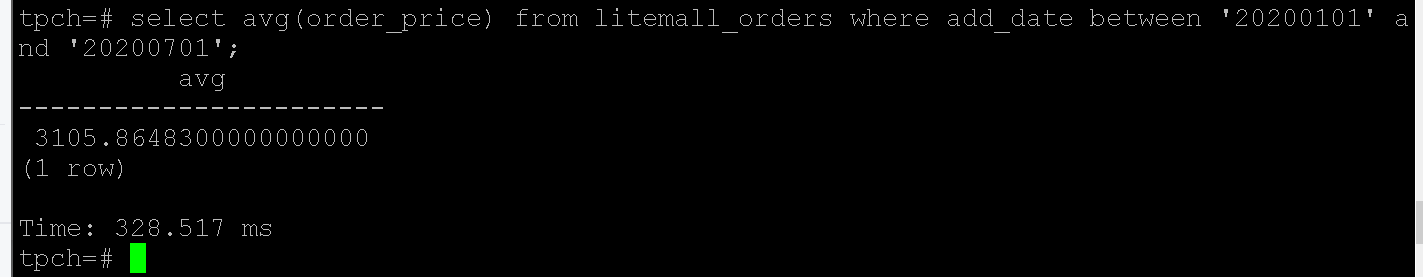


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

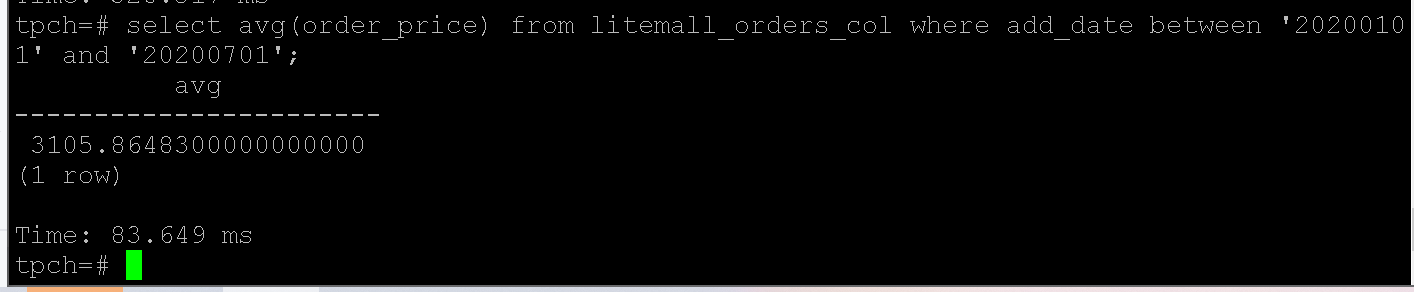


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

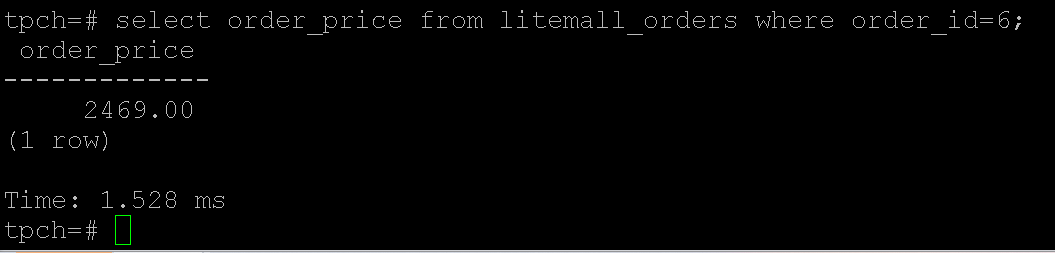


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

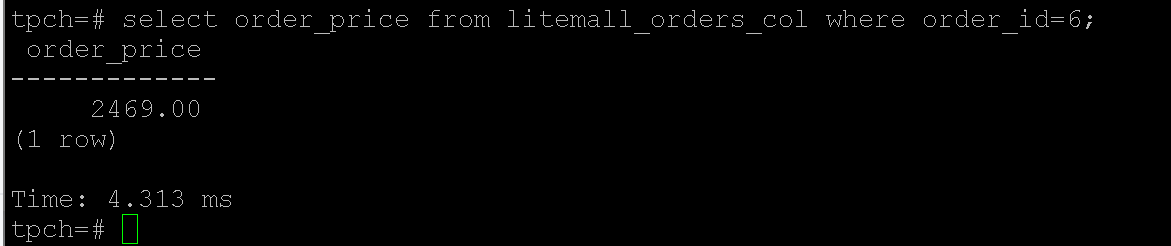


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

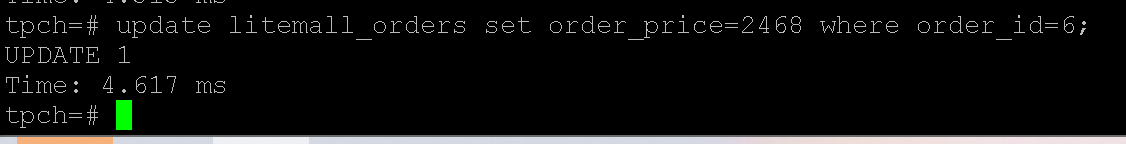


select order\_price from litemall\_orders\_col where order\_id=6;

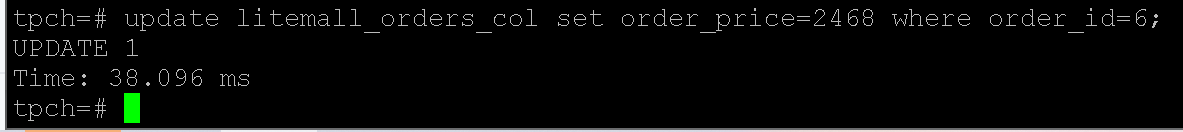


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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？

行存表和列存表是两种不同的数据库表存储方式，它们在执行相同的SQL语句时表现不同的主要原因是因为数据存储和访问方式不同。

行存表（Row-oriented table）将每一行数据作为一个整体存储，每个字段都存储在同一行内，这样的设计方便进行读取整行数据的操作，但当只需要读取部分字段时就会存在冗余读取的情况，造成了I/O不必要的浪费。

而列存表（Column-oriented table）则将每一个列独立存储，相同列数据存储在一起，这样的设计可以最大化地利用数据压缩算法，减少冗余存储，提高数据的压缩比。对于需要查询单个或少量字段的场景，列存表可以大幅度减少物理I/O以及CPU资源的消耗，提高查询效率。

总体来说，当查询涉及到大量字段时，行存表可能会更快，因为它可以更快地检索整行数据。对于只查询特定列的查询，列存表可能会更快，因为它可以更快地检索其所需的列，并且避免了读取冗余数据。因此，在选择使用行存表还是列存表时，需要根据实际的业务场景、数据量大小等因素进行综合考虑，根据查询和分析的需求合理选择表的存储方式。

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？

在执行哪些类型SQL时，行存表效率更高？

对整行进行读取查询时，行存表的效率更高。因为行存表将每一行数据作为一个整体存储，可以更快速地读取整行数据。

更新单行操作时，行存表通常比列存表效率更高。因为在行存表中更新单行比较容易，而在列存表中进行单行更新需要读取大量的数据块，并且数据块的压缩和解压缩也会导致额外的开销。执行聚合函数时，行存表通常更高效，因为大多数聚合函数都需要读取整个行的数据。

在执行哪些类型SQL时，列存表效率更高？

当需要查询特定列或少量的列时，列存表的效率更高，因为列存表将每一个列独立存储，相同列数据存储在一起，可以更快地检索其所需的列，并且避免了读取冗余数据，从而大幅度减少物理I/O以及CPU资源的消耗。

用于对大型数据集的扫描和分析时，列存表的效率更高，因为在列存表中进行这些操作可以只读取所需的列，减少了I/O操作的数量，从而提高查询速度。此外，列存表还能够利用列之间的相似性和数据压缩技术来减少存储空间和I/O消耗。

总之，行存表和列存表各有优劣。在实际应用中，需要根据具体的查询需求、数据量、硬件条件等综合因素进行选择。

# 关卡三：openGauss物化视图应用

任务一：物化视图的使用

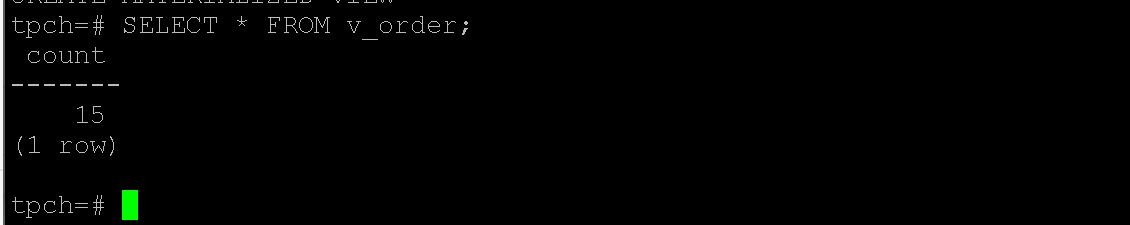
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



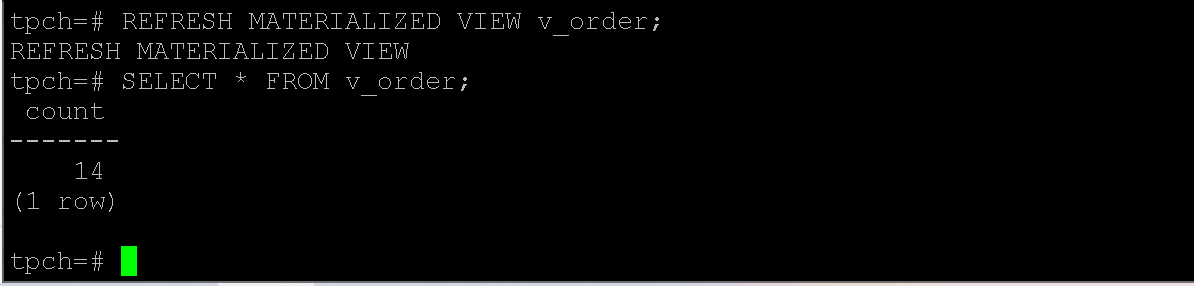
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



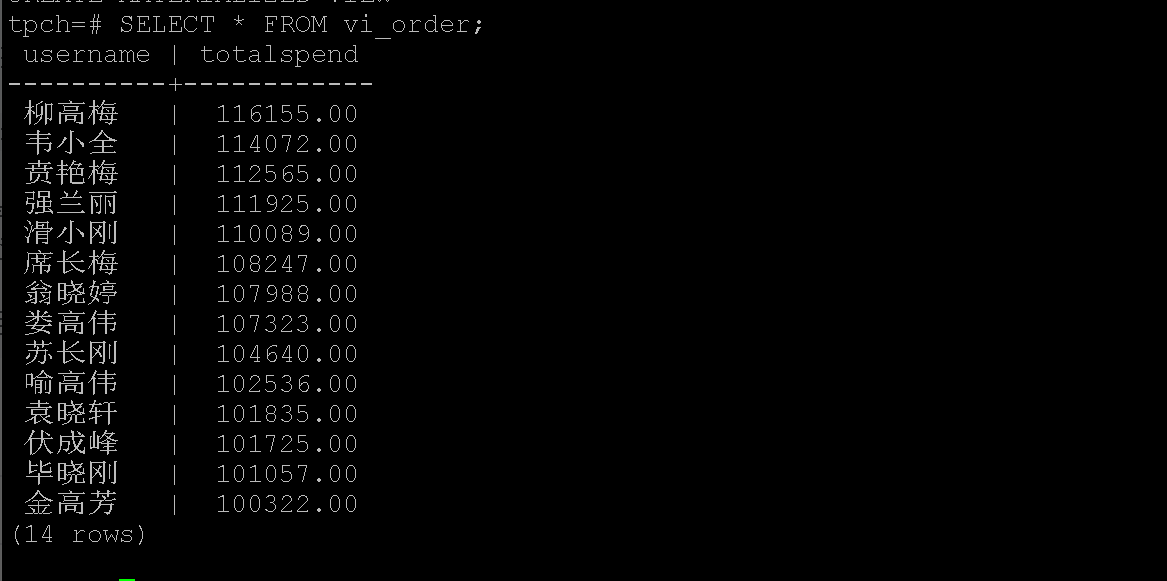
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



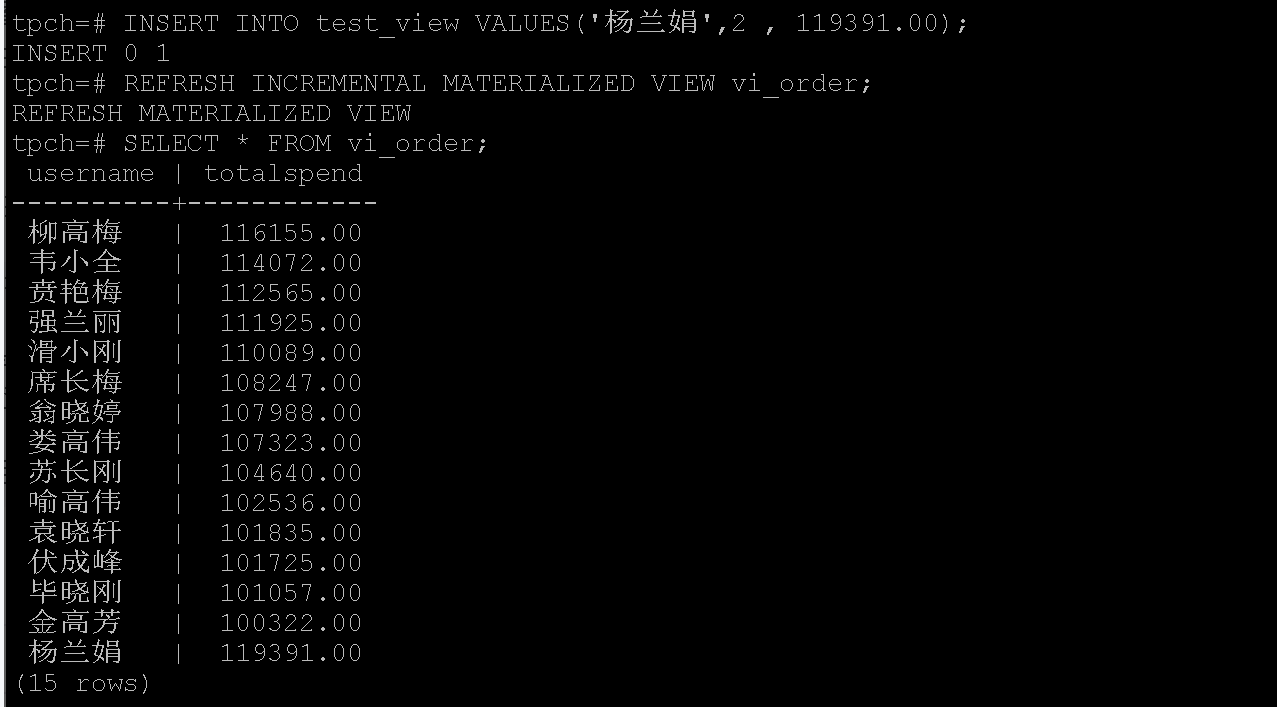
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

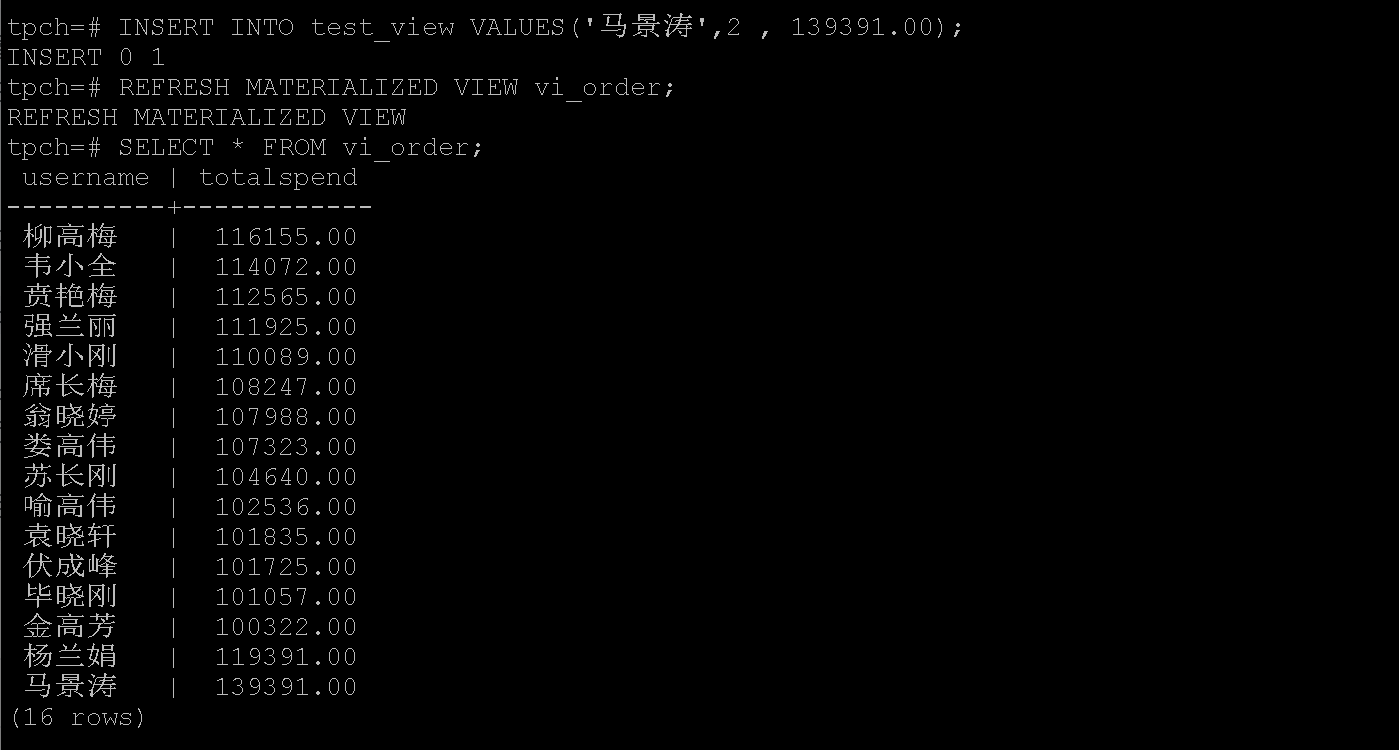
SELECT \* FROM vi\_order;



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的物化视图。

全量物化视图是完全重新计算整个查询结果，并将结果存储在物化视图中。当物化视图被刷新时，数据库会执行完整的查询语句，并将结果保存到物化视图中。在查询物化视图时，数据库引擎直接返回预先计算好的结果。全量物化视图适用于需要完整结果的场景，但如果数据量较大，性能可能会受到影响。

与全量物化视图相比，增量物化视图仅更新发生变化的数据行或者所依赖表的数据行，在新增、修改或删除数据时，只需更新部分数据，而不是全部重新计算。增量物化视图需要在创建时定义日志表，用于保存变更记录，以支持增量更新。增量物化视图适用于对实时查询要求较高、数据变化频繁的场景。

总之，全量物化视图和增量物化视图各有优缺点，在具体应用场景中需要根据数据变化的频率、数据量大小、查询效率等因素，选择合适的物化视图类型。

思考题2：物化视图适用那些使用场景？

物化视图是一种预先计算好的查询结果集，可以缓存查询结果并提高查询性能，降低查询开销。物化视图适用于以下场景：

复杂的查询：当需要多个表进行关联查询或使用复杂的聚合函数时，查询的性能会受到很大的影响。使用物化视图来缓存查询结果，可以大幅度提高查询效率。

数据仓库：物化视图常用于数据仓库中，可以预先计算复杂的指标（例如时间维度的聚合指标）或汇总数据，并将结果保存到物化视图中，以加速常见的查询操作。特别是在面对大数据量和频繁查询的情况下，物化视图可以有效减少I/O操作和提升查询性能。

频繁变更的数据：当查询的数据经常发生变化时，使用物化视图可以有效减少再次执行查询语句的开销，在数据库更新数据时只需刷新部分数据即可，从而能够提高查询性能并减少资源的浪费。

总之，物化视图可以显著提高复杂查询和数据仓库等场景下的查询性能，适用于需要频繁查询的应用场景，但也需要考虑对数据的实时性要求和物化视图本身的维护成本等因素。

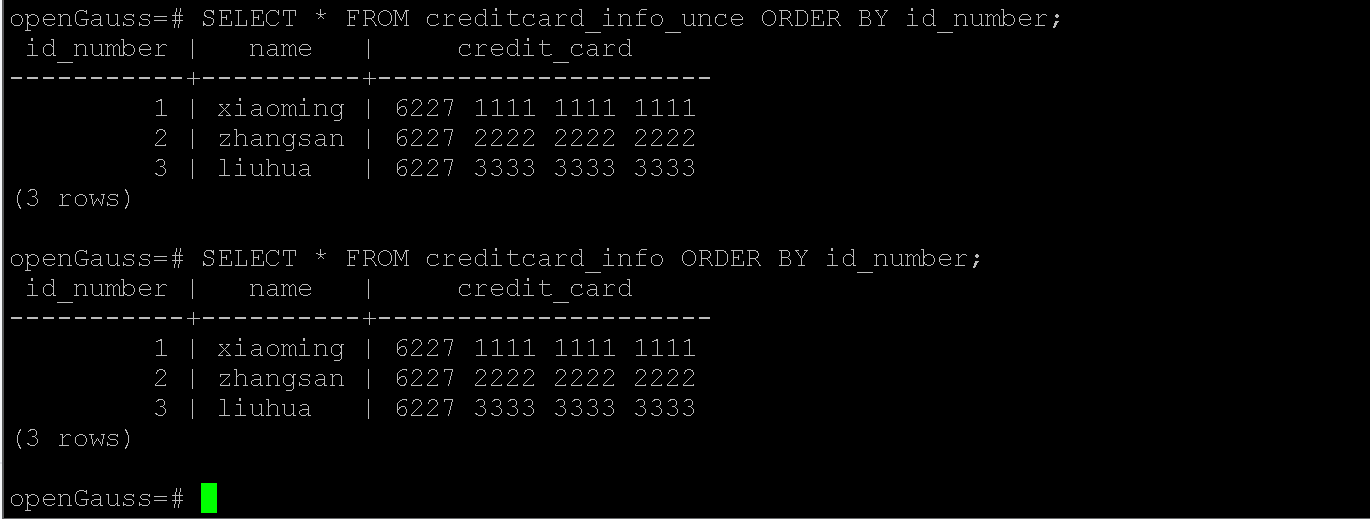
# 关卡四：openGauss密态数据库特性应用

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：



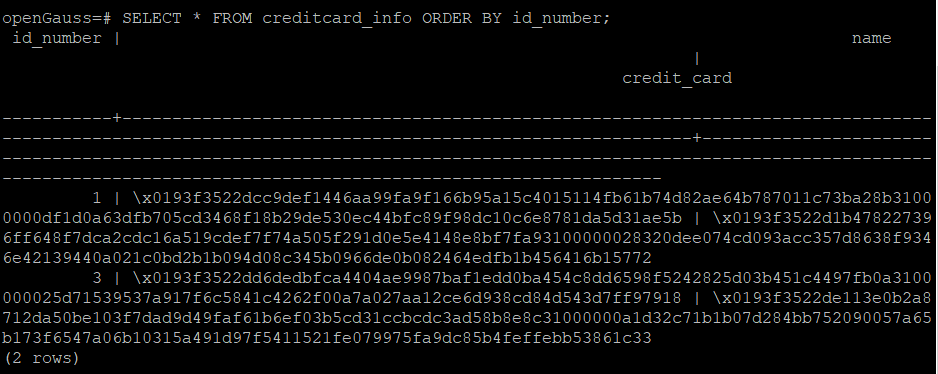
1. 将加密表和非加密表查询结果截图：



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，一般情况下是以密文的方式进行存储。在数据传输过程中，如果采用了加密机制，那么数据在传输过程中也是以密文的形式进行传输的。

加解密的动作可以在客户端或服务端完成，这取决于具体的加密方案和实现方式。如果采用对称加密，那么加解密的动作通常是在客户端和服务端各自完成的，因为对称加密使用的密钥是相同的。而如果采用非对称加密，加密和解密的动作可以分别由客户端和服务端完成，因为非对称加密使用的公钥和私钥是不同的。

总之，在数据存储和传输过程中，采用加密机制可以保障数据的安全性，具体的加密方式和实现取决于具体的应用场景和需求。