openGauss 安全体系创新

实践课



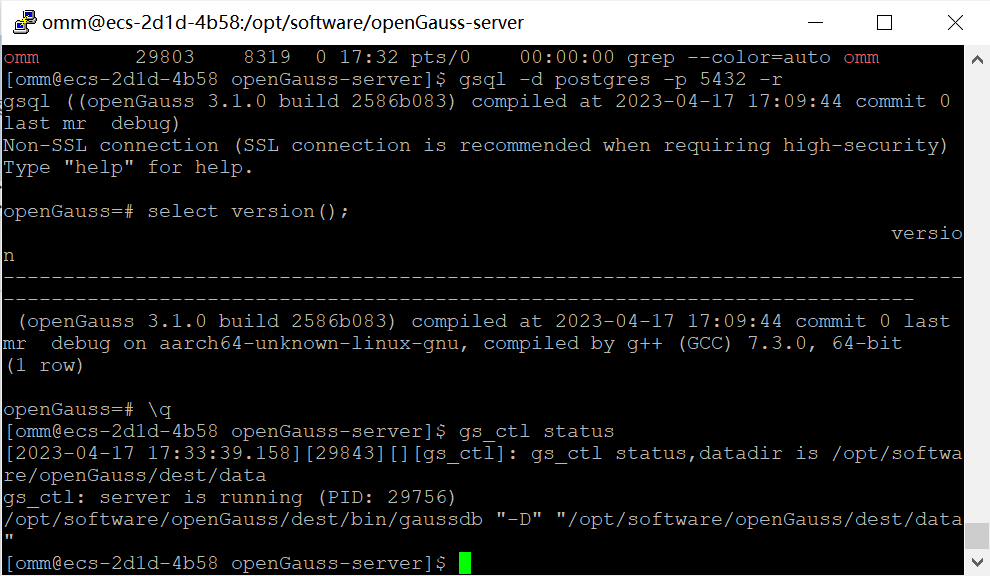
华为技术有限公司

# 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：

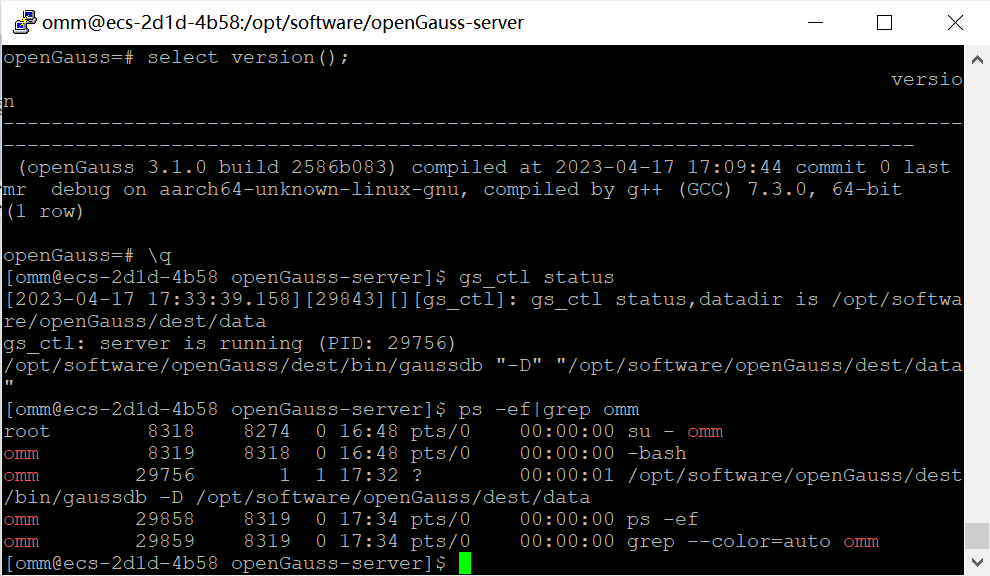
任务一：数据库状态验证

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）

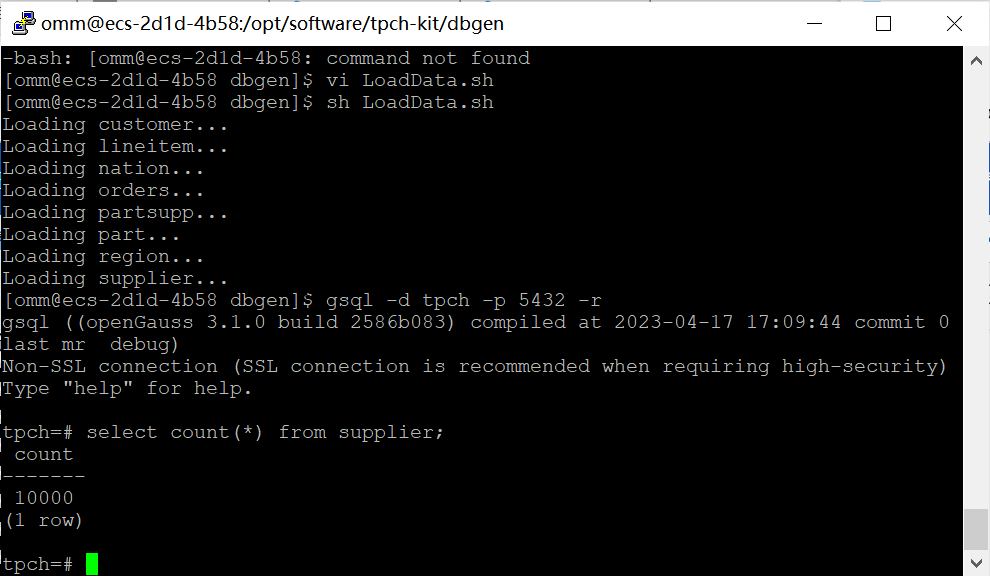


任务三：实践思考题

思考题1：为什么需要通过源码编译，安装数据库？

通过源码编译和安装数据库可以提供更高的灵活性和自定义性。通常，当我们安装预编译的二进制文件时，我们只能使用默认配置和选项，不能对其进行修改。但是，如果你通过源码编译并手动安装数据库，则可以根据你的特定需求调整数据库的配置和选项。

此外，通过源码编译还可以确保正在使用最新的软件版本，并且我们可以针对硬件和操作系统进行优化以提高性能。尽管这种方法可能需要更多的时间和技能来完成，但它通常会为我们带来更好的结果，特别是在大型生产环境中。

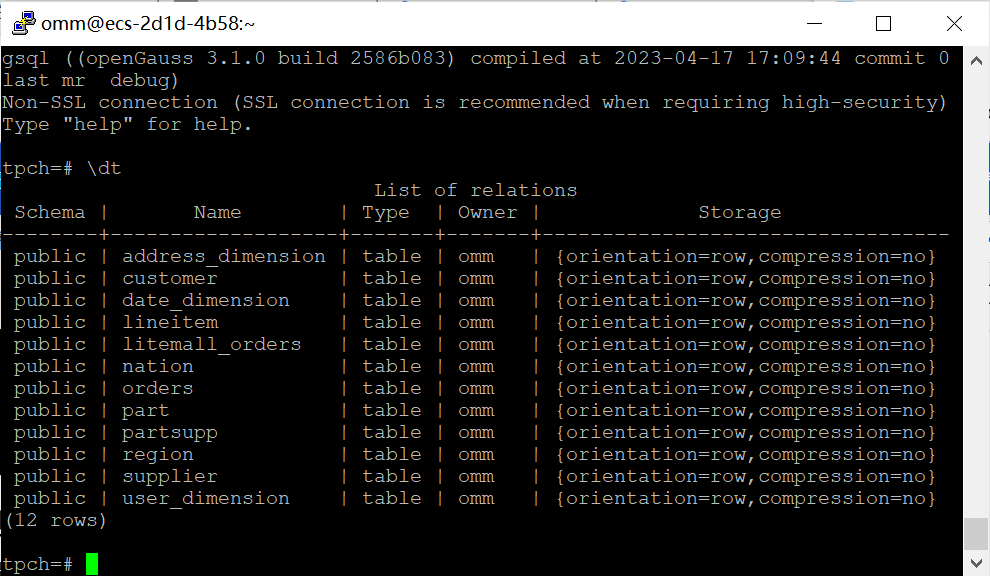


# 关卡二、openGauss数据导入及行存列存

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;;

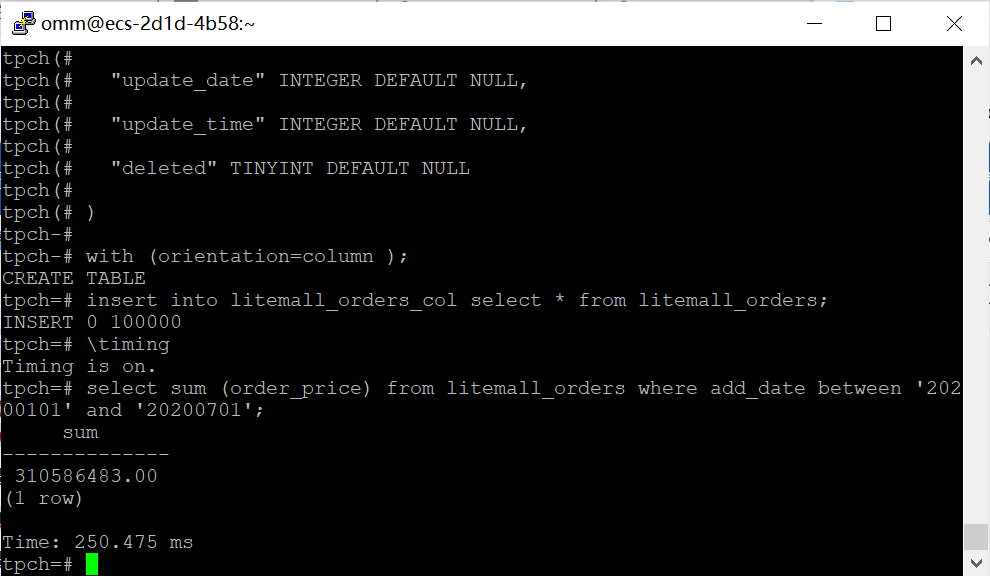


任务二：行存表与列存表执行效率对比

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';

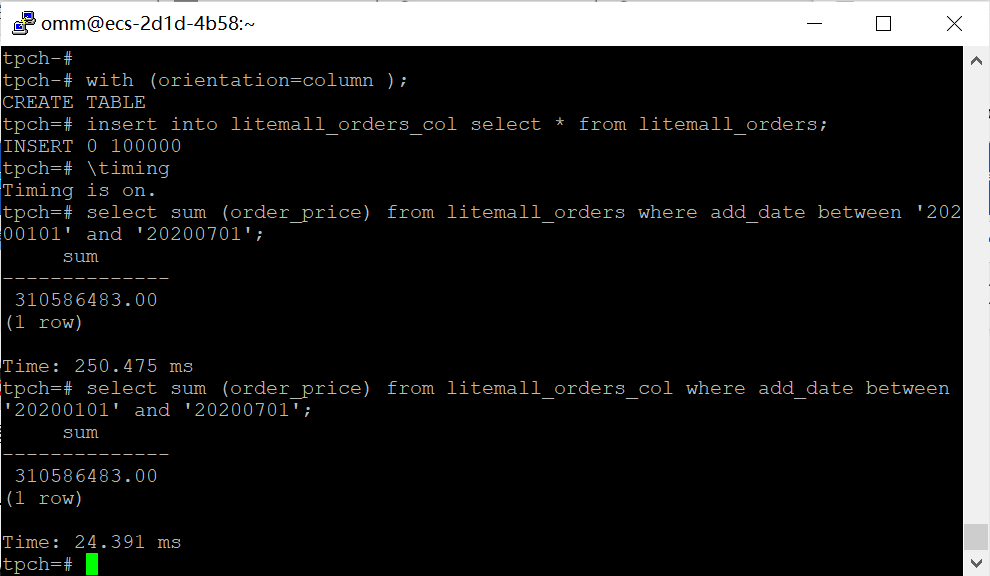
select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';



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';

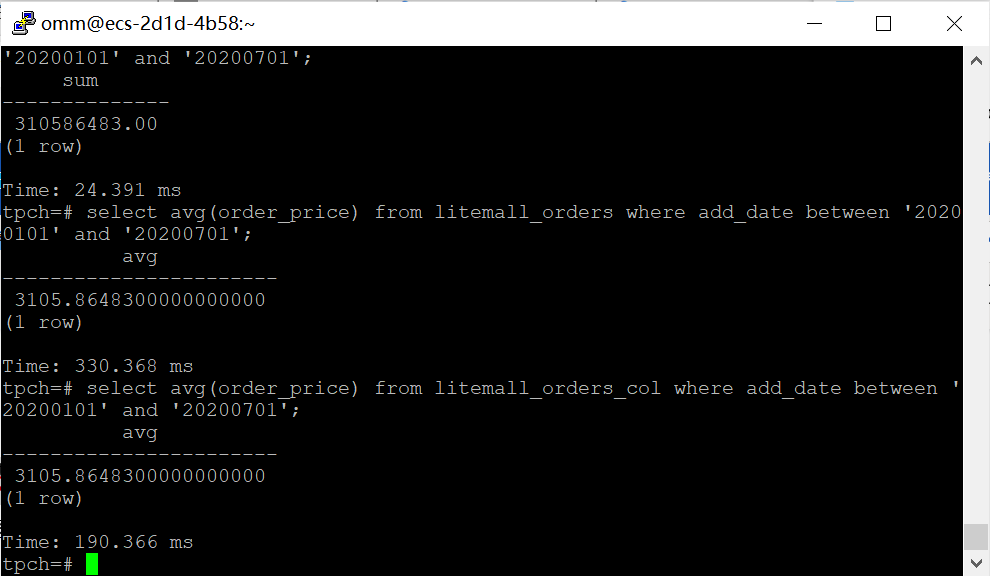
select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

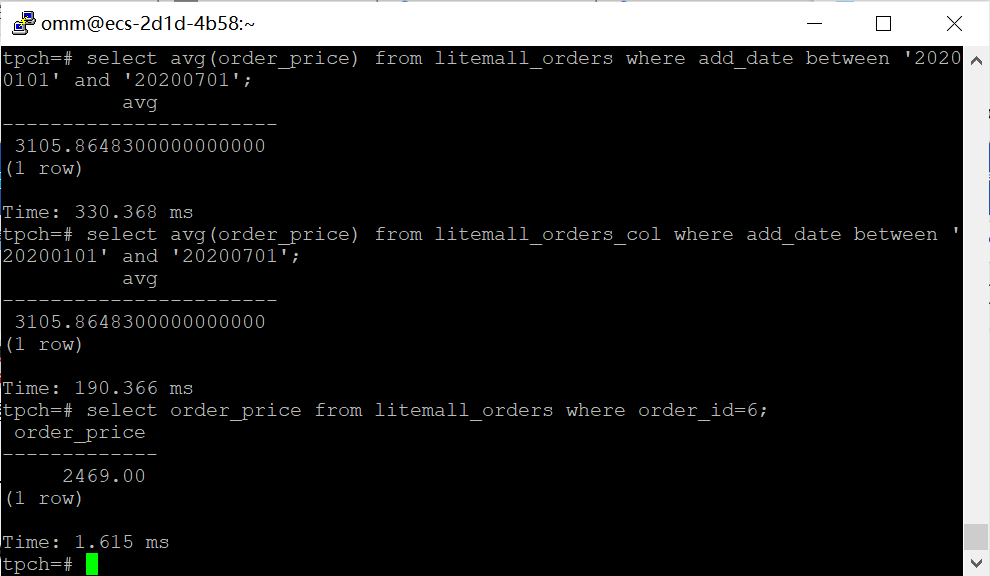


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

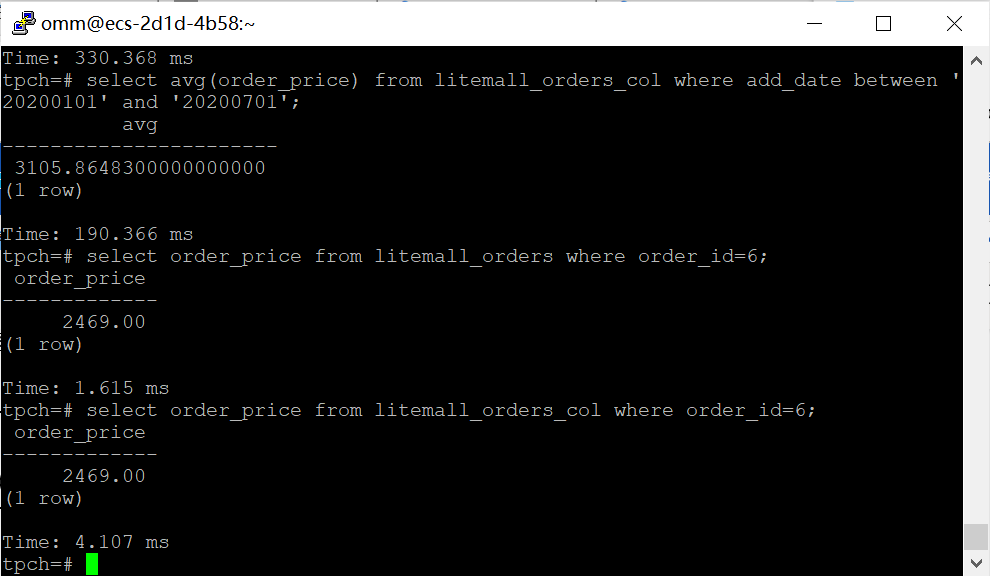
select order\_price from litemall\_orders\_col where order\_id=6;

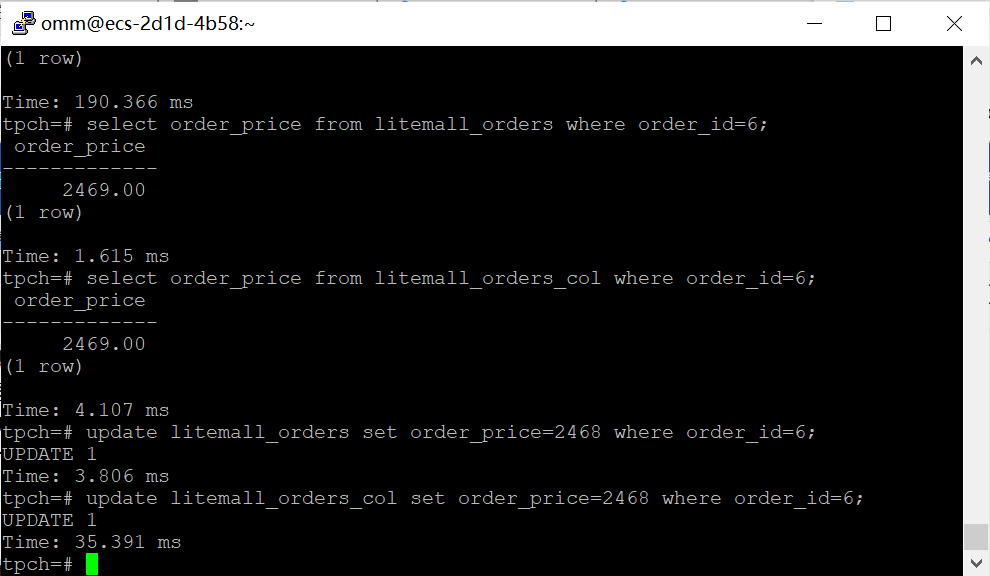




4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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？

行存表按照记录的顺序存储数据，每个记录都包含多个字段。当执行一个查询时，数据库引擎需要扫描整个表来找到匹配查询条件的行。这种扫描需要读取所有字段的值并比较它们与查询条件。如果表中的记录数很多，则这种方法可能会导致性能问题。

相反，列存表将每个字段单独存储，可以更有效地压缩数据。当执行查询时，数据库引擎只需读取包含匹配条件的字段的值，并且可以跳过其他字段的值。这种方法通常比行存表更快。

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？

行存表在执行需要检索整行数据或执行大量的更新操作时效率更高。例如，当我们需要读取完整的记录、进行聚合查询、进行遍历和修改等操作时，行存表可能会比列存表更加高效。

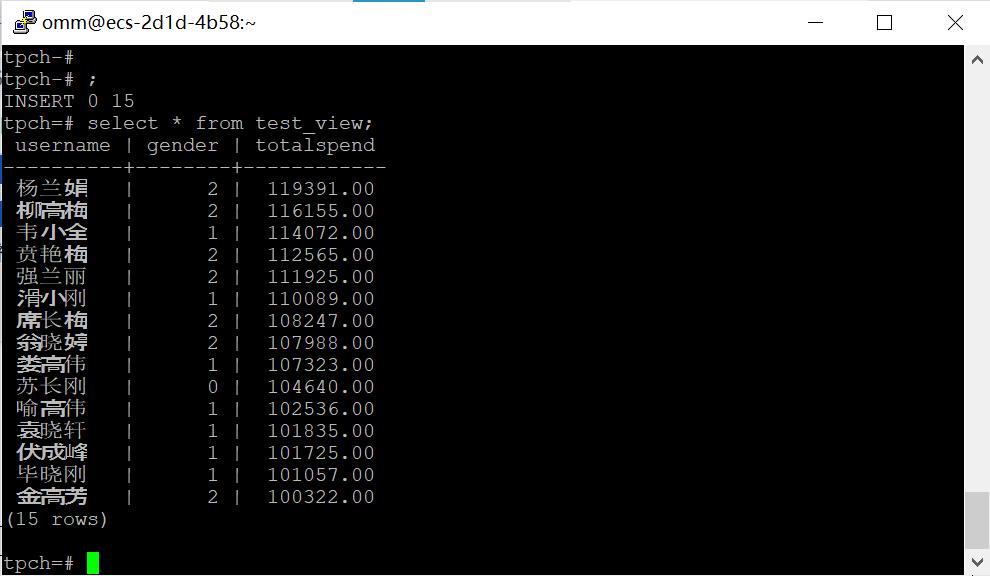
而列存表则在执行需要聚合、筛选特定列和查询多个表之间的连接时效率更高。由于列存表只需读取所需要的列，在执行这类查询时可以减少磁盘读取次数，从而提高性能。此外，列存表还可以更好地支持复杂查询，如涉及大量 join 操作的查询。

# 关卡三：openGauss物化视图应用

任务一：物化视图的使用

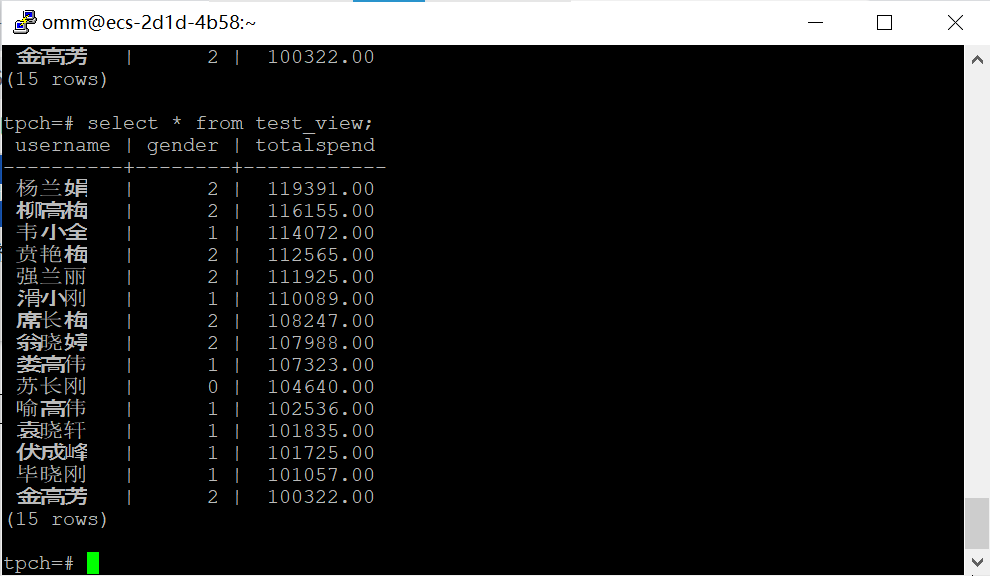
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



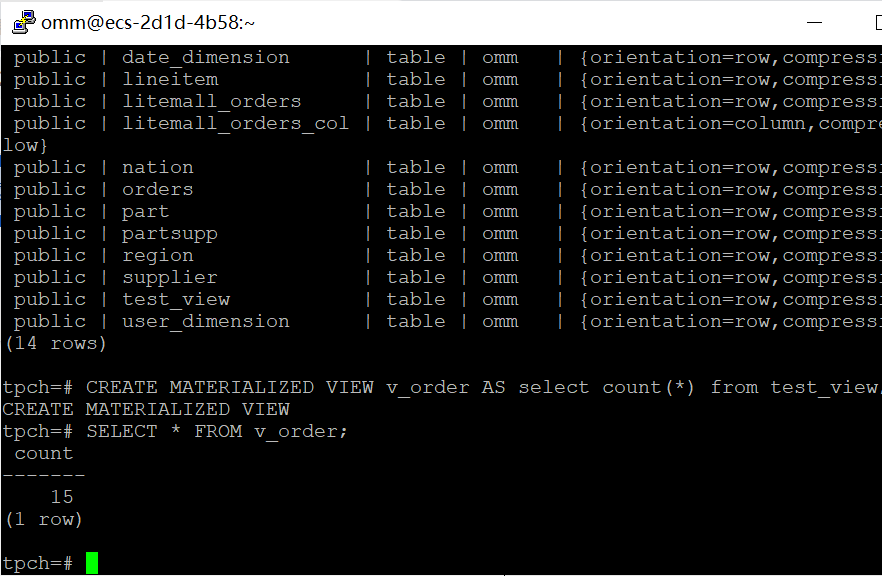
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



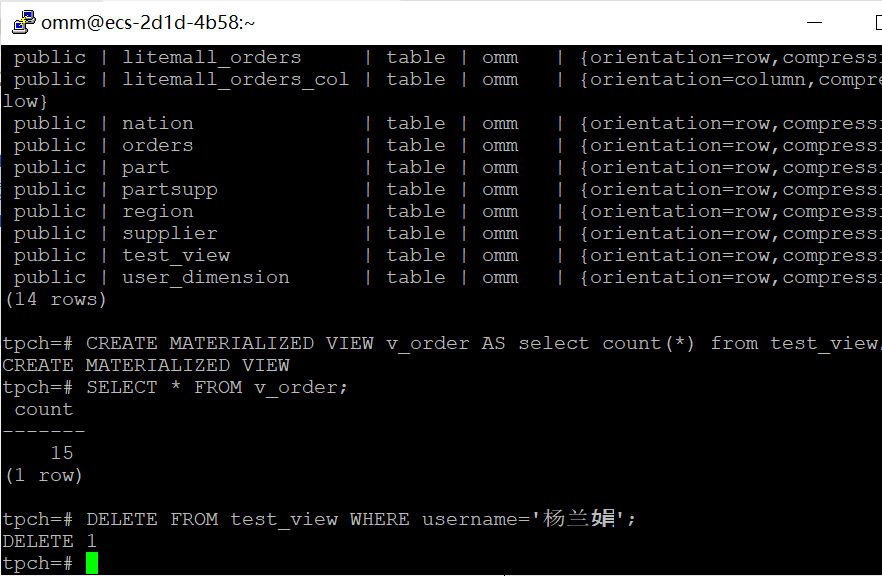
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



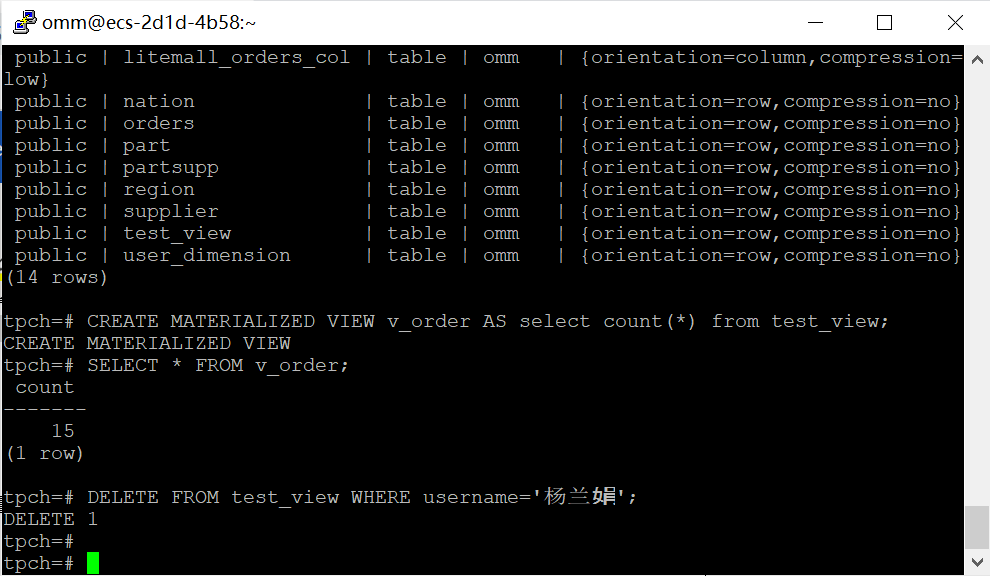
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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都是用于优化查询性能的技术，但它们有以下几点不同：

定义：全量物化视图在创建时会将整个查询结果存储在物化视图中，而增量物化视图则只存储变更的数据。

更新频率：全量物化视图的更新频率低，因为每次更新需要重新计算所有数据。而增量物化视图则可以在较短时间内快速更新，因为只需要计算变更的数据。

存储空间：全量物化视图需要占用更多的存储空间，因为存储了整个查询结果；而增量物化视图则只存储了变更的数据，因此需要的存储空间较少。

查询性能：对于频繁查询的场景，全量物化视图可能会更适合，因为它可以通过使用预先计算好的数据来快速响应查询；而对于大量变更的场景，增量物化视图可能会更适合，因为它可以更快地更新变更的数据。

总的来说，全量物化视图适用于相对静态的数据，并且查询的响应时间要非常快，而增量物化视图则适用于经常发生变更的数据集，且查询的响应时间可以稍微慢一些。

思考题2：物化视图适用那些使用场景？

物化视图适用于需要快速查询大量数据的应用场景。在传统的关系型数据库中，如果有复杂的查询操作，可能需要进行多表连接，这会导致查询效率较低。而通过使用物化视图，可以将查询结果缓存到一个物理表中，以后再次查询时直接从这个物理表中读取数据，不需要再进行计算和多表连接，从而提高查询效率。

具体来说，物化视图适用于以下情况：

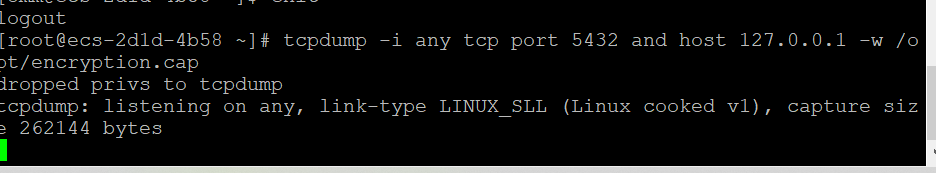
1. 经常需要查询复杂的聚合函数或多表连接的数据
2. 查询的结果数据量较大，且需要经常访问
3. 数据更新频率不高，即使更新也不会对查询产生实质性影响

需要注意的是，物化视图虽然可以提高查询效率，但是会占用额外的存储空间，并且需要定期刷新以保证数据的准确性。因此，需要根据具体的业务场景和需求来权衡是否使用物化视图。

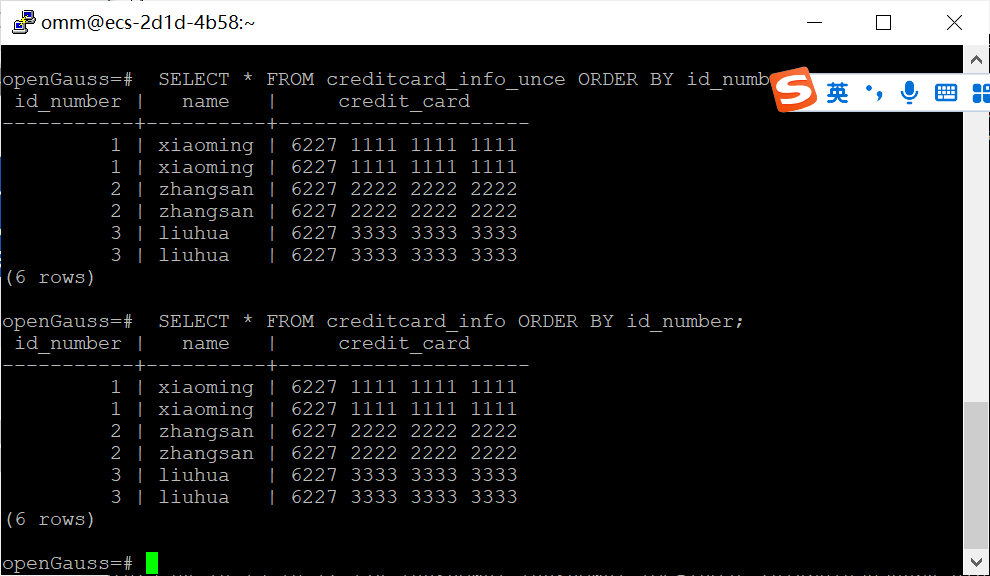
# 关卡四：openGauss密态数据库特性应用

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：



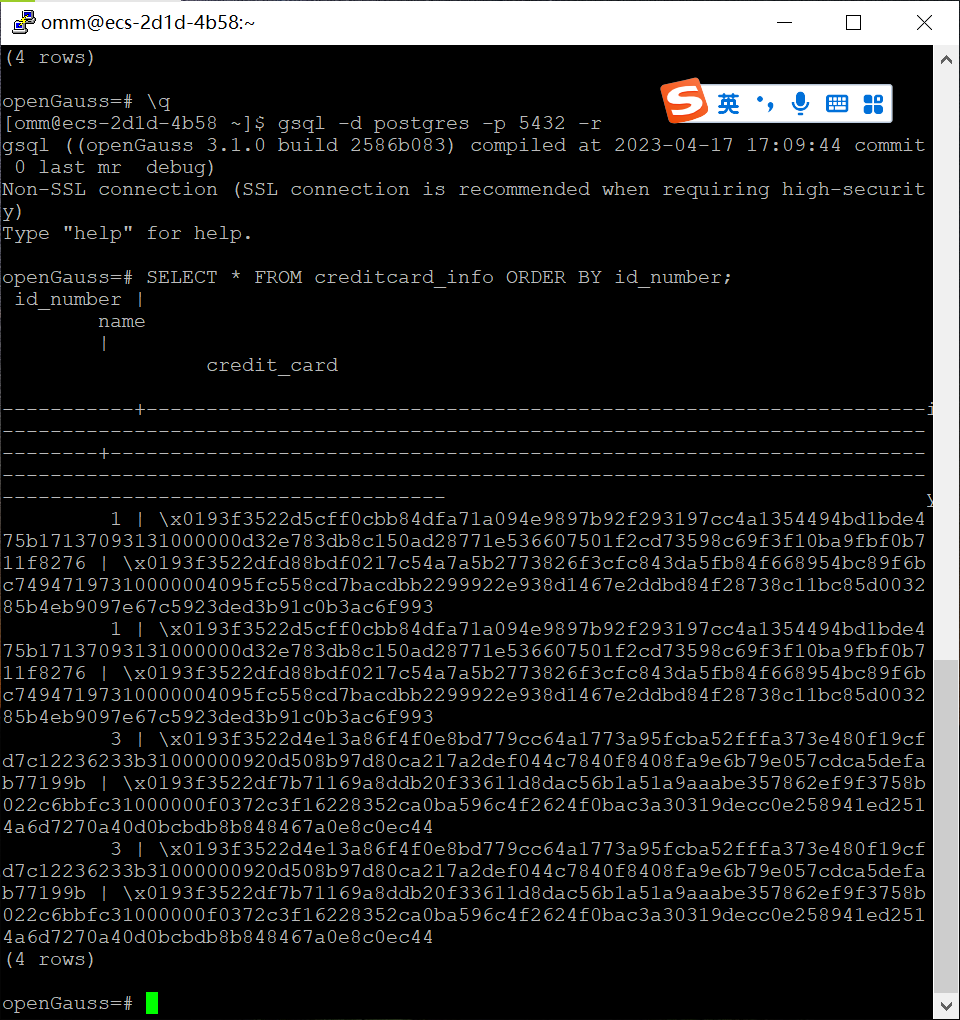
1. 将加密表和非加密表查询结果截图：



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？

数据在物理磁盘上存储时通常是以明文形式保存的。然而，在一些安全性要求更高的情况下，也可以将数据加密后再存储在磁盘上。

加密和解密的动作可以在客户端或服务端完成，具体取决于采用的加密方案和系统设计。例如，在端到端加密的场景中，加密和解密操作通常由客户端负责完成，而服务器只负责保存和转发加密后的数据。在其他场景下，可能需要服务端负责加密和解密操作，以便实现更细粒度的访问控制和审计功能。