openGauss 安全体系创新

实践课



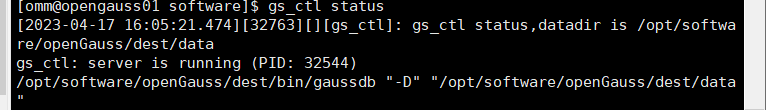
华为技术有限公司

# 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：

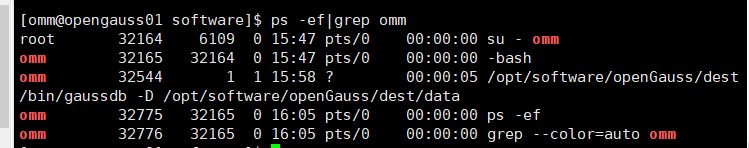
任务一：数据库状态验证

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？

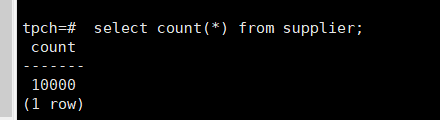
源码编译和安装数据库可以定制化设置、优化性能、版本控制、提高安全性，以及有助于学习和掌握数据库相关技术。

# 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：

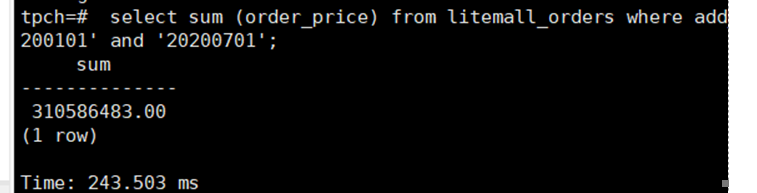
select count(\*) from supplier;;



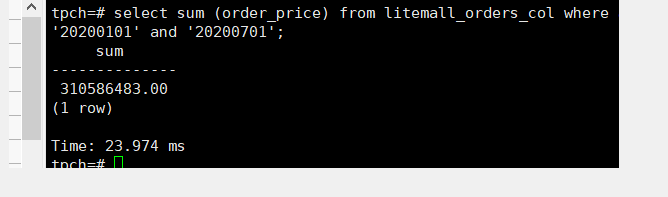
任务二：行存表与列存表执行效率对比

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';

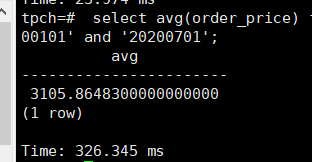


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

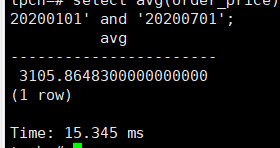


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

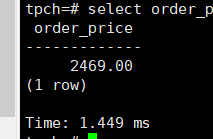


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

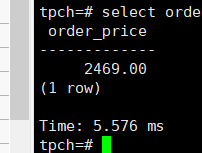


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

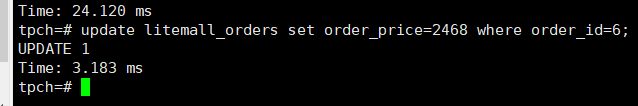


select order\_price from litemall\_orders\_col where order\_id=6;

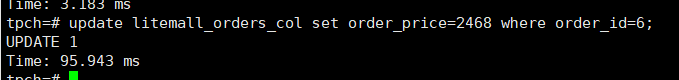


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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？

因为他们数据的组织方式和存储方式不同，所以可能会导致执行时间的差异。行存表是按行存储数据的，每一行数据包含了表中所有的字段信息。当执行查询语句时，需要读取整行数据，其中可能包含不需要的字段信息，导致数据的冗余和浪费，同时会增加磁盘 I/O 和 CPU 的负担。列存表是按列存储数据的，每一列数据被存储在一起。当执行查询语句时，只需要读取需要的列数据，减少了数据冗余和浪费，同时也减少了磁盘 I/O 和 CPU 的负担。由于在列存表中每个列的数据类型是相同的，所以在进行查询时可以使用更加高效的压缩算法，从而节约了存储空间。

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？

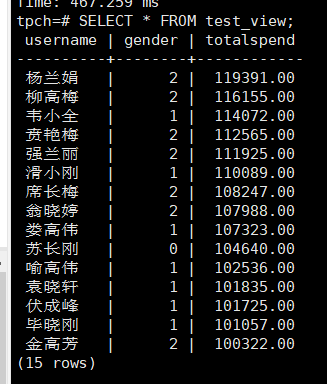
行存表在执行单行查询、数据的增删改操作时效率更高，而列存表在执行数据聚合查询、需要查询部分列数据的查询和针对大型数据集的复杂查询时效率更高。

# 关卡三：openGauss物化视图应用

任务一：物化视图的使用

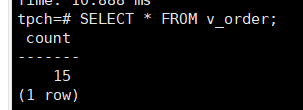
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



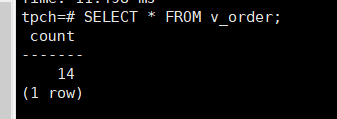
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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？

全量物化视图保存所有数据，需要定期完全刷新，而增量物化视图只保存部分数据，可以进行增量刷新，以提高刷新效率。选择使用哪种物化视图取决于具体的业务需求和数据处理情况。思考题2：物化视图适用那些使用场景？

物化视图通常适用于以下场景：

复杂的查询：当查询需要联结多个表、进行聚合计算或者使用复杂的逻辑时，查询的性能往往较低。此时，可以使用物化视图将查询的结果预先计算出来，并将结果存储到物化视图中，这样可以大幅提高查询的性能。

数据仓库和大数据分析：物化视图常常被用于数据仓库和大数据分析系统中，这些系统需要快速、高效地处理大量数据。通过使用物化视图，可以将数据预先计算出来，并存储在物化视图中，这样可以避免重复计算，提高数据处理的效率。

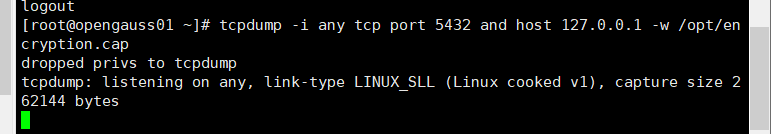
实时报表和数据可视化：当需要生成实时报表或者进行数据可视化时，物化视图可以提供高效的数据查询和统计功能。通过使用物化视图，可以将需要查询的数据预先计算出来，并存储在物化视图中，这样可以在生成报表或者进行数据可视化时大幅提高性能。

数据同步和数据复制：物化视图还可以用于实现数据同步和数据复制功能。通过使用物化视图，可以将源数据库中的数据同步到目标数据库中，从而实现数据复制和数据同步的功能。

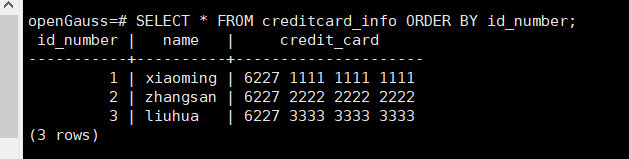
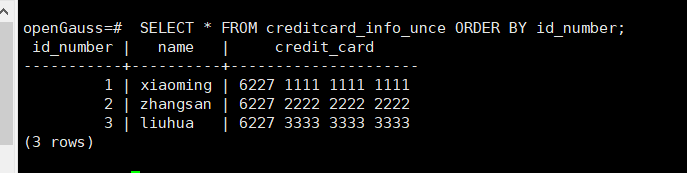
# 关卡四：openGauss密态数据库特性应用

任务一：物化视图的使用

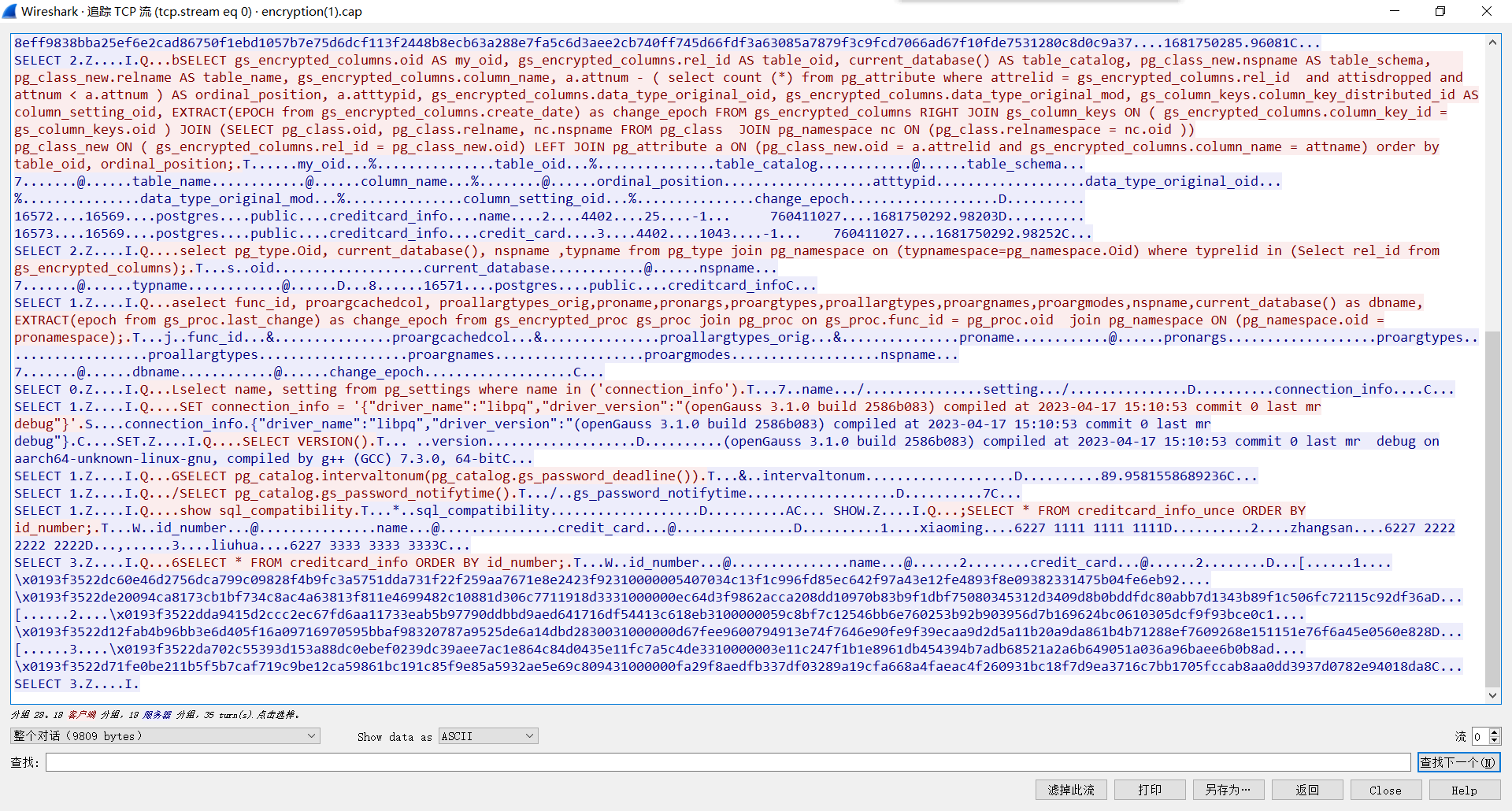
1. 通过tcpdump抓取数据流，此putty窗口暂时保持不动，将执行结果截图：



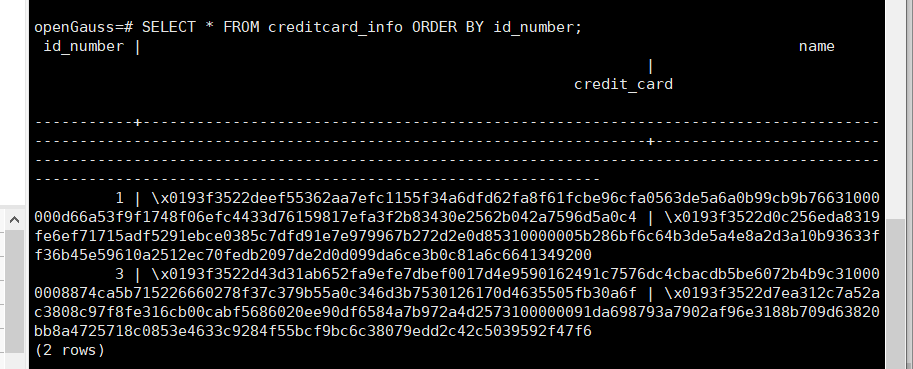
1. 将加密表和非加密表查询结果截图：



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通常是明文形式存储的，因为数据库系统可能需要随时对数据进行读写操作，如果数据存储为密文，那么每次读写都要增加加解密的时间，这会影响数据库的响应时间等性能。但是在一些安全要求较高的场所，数据可能会被存为密文形式。

数据的加解密通常在服务端完成，以保证数据的安全性和一致性。为了增强数据库系统的安全性，数据库系统通常提供数据加密、身份认证、访问控制等功能。