openGauss 安全体系创新

实践课



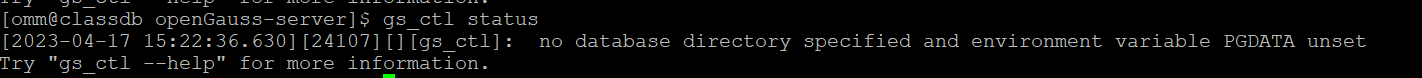
华为技术有限公司

# 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：

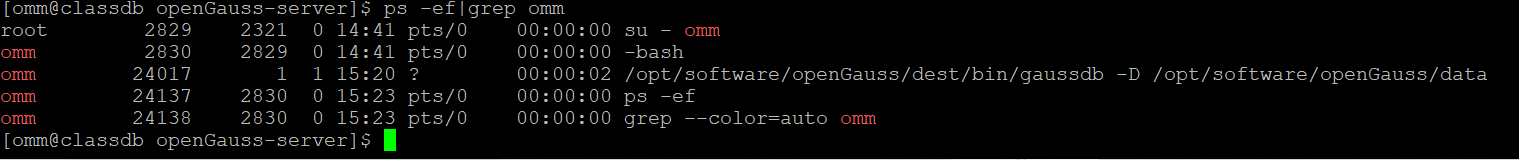
任务一：数据库状态验证

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？

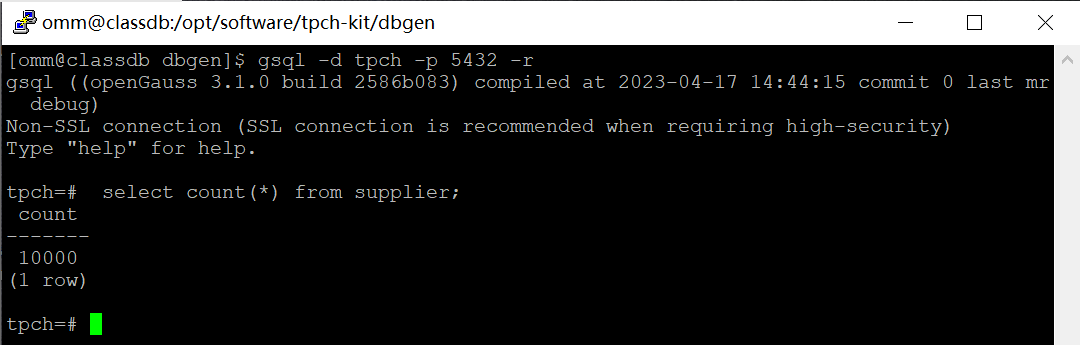
1. 定制化：通过源码编译安装，可以根据自己的需要进行定制化配置，包括指定特定的编译选项、安装目录、模块等。  
2. 最新版本：源码编译安装可以确保你安装的是最新的版本，而不是依赖于操作系统提供的版本。这样可以获得最新的功能和修复的 bug。  
3. 最大性能：源码编译可以根据硬件环境和需要进行优化，以获得最大性能。而预编译的二进制版本则可能无法完全利用硬件性能。  
4. 安全性：通过源码编译安装数据库，可以避免使用可能存在漏洞或被恶意修改的预编译版本，提高安全性。  
5. 学习机会：通过源码编译安装，可以深入了解数据库的结构和运作原理，从而提高对数据库的理解和掌握能力。

# 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：

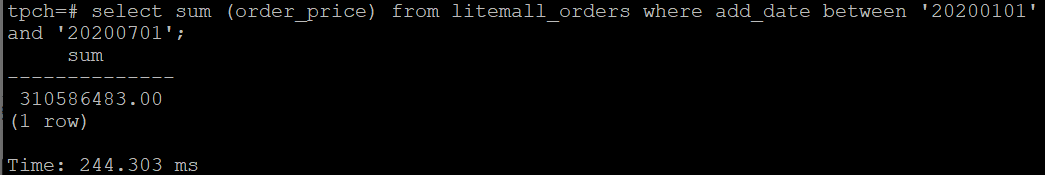
select count(\*) from supplier;;



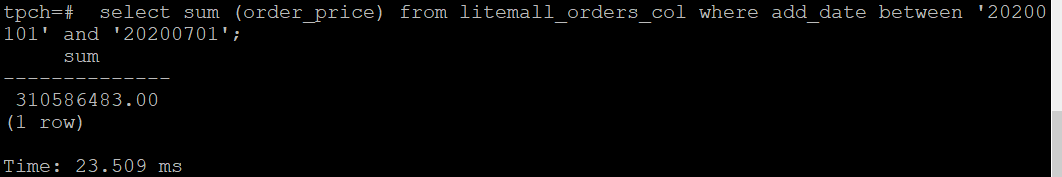
任务二：行存表与列存表执行效率对比

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';

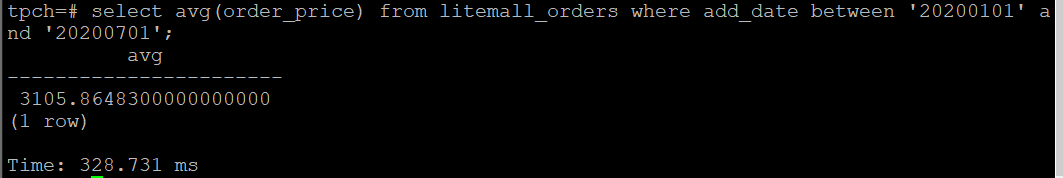


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

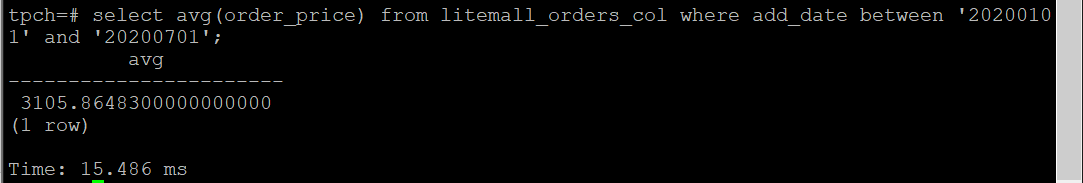


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

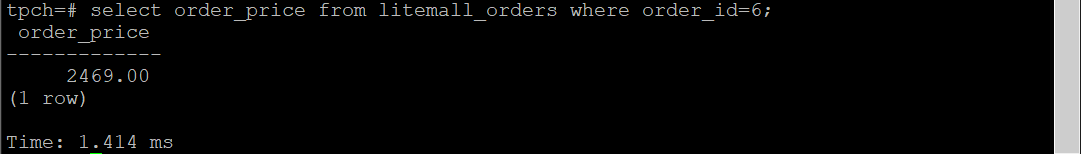


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

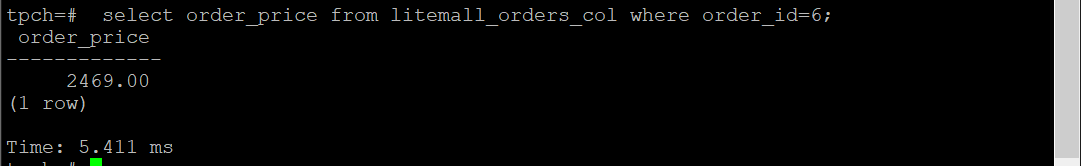


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

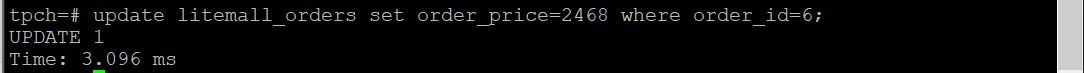


select order\_price from litemall\_orders\_col where order\_id=6;

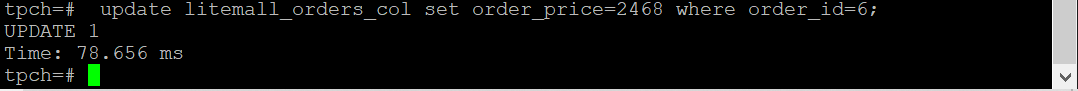


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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？

行式存储法，在基于行式存储的数据库中， 数据是按照行数据为基础逻辑存储单元进行存储的， 一行中的数据在存储介质中以连续存储形式存在。

列式存储是相对于行式存储来说的。在基于列式存储的数据库中， 数据是按照列为基础的逻辑存储单元进行存储的，一列中的数据在存储介质中以连续存储形式存在。

因此两种表在执行相同的SQL语句时，各自的执行时间不同。

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？

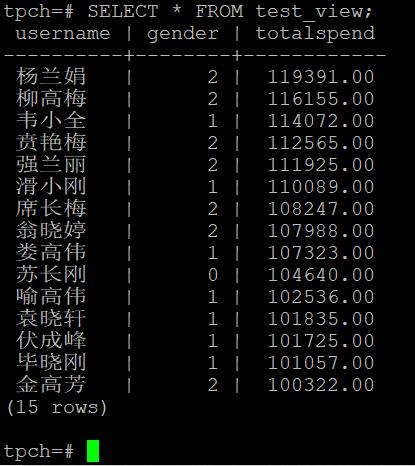
在执行增删查改类型SQL的时候，行存表的效率更高。在执行查询单列所有数据类型SQL时候，列存表效率更高。

# 关卡三：openGauss物化视图应用

任务一：物化视图的使用

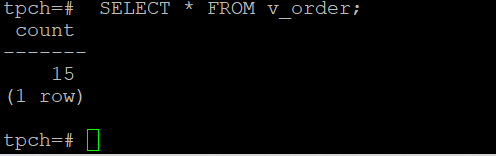
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



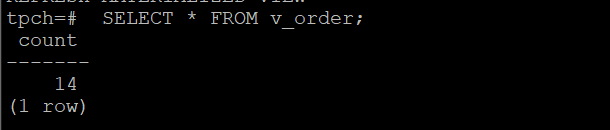
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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;



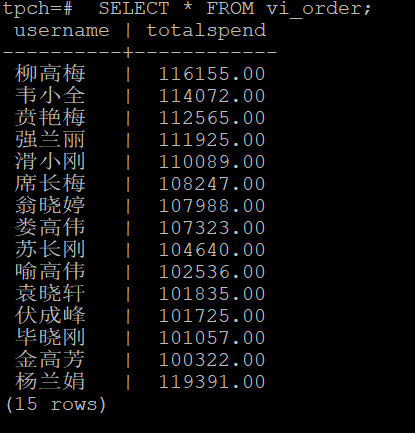
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

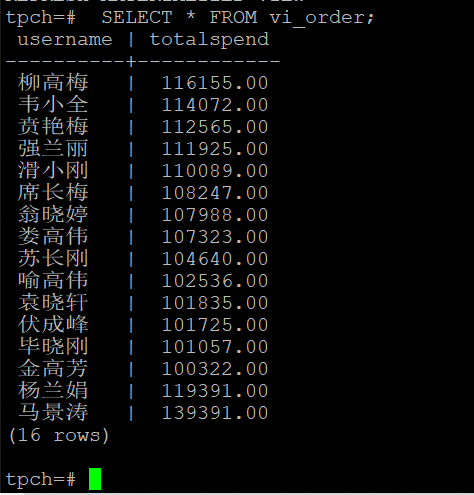
SELECT \* FROM vi\_order;



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：每次更新时把 MV 清空，再查询并把结果全部插入到 MV 中。

增量物化：不用每次把 MV 清空再重新插入，而是获取对基表的增量修改，再计算并应用到 MV 上。

全量物化视图能根据用户每次的查询，更新时候刷新数据。

增量物化视图顾名思义就是可以对物化视图增量刷新，需要用户手动执行语句完成对物化视图在一段时间内的增量数据进行刷新。

思考题2：物化视图适用那些使用场景？

物化视图的应用场景有两种：

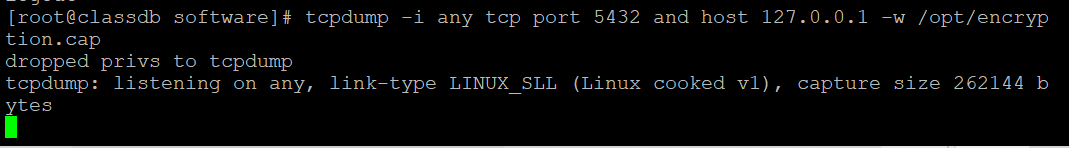
1、用于查询优化 ：可以将多个表的属性组合到物化视图表中完成查询，降低查询压力。

2、用于高级复制：很多业务场景不需要对整个数据库进行同步，只需要对部分表的部分字段进行同步。物化视图在此方面应用。

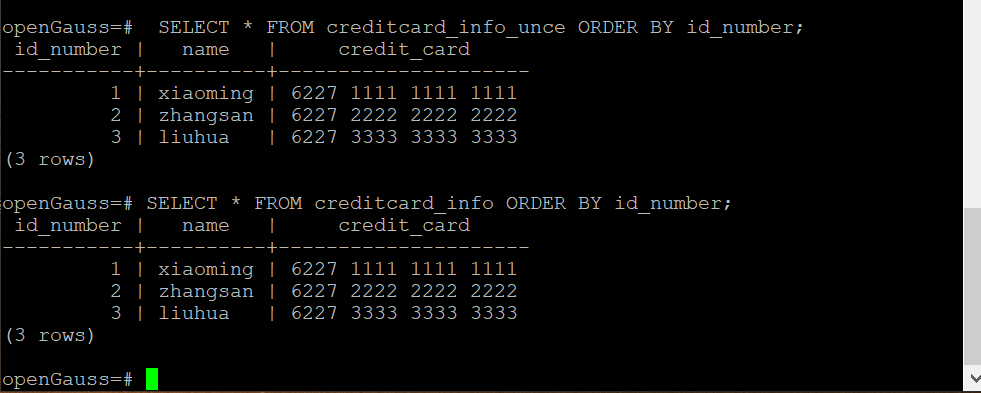
# 关卡四：openGauss密态数据库特性应用

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：



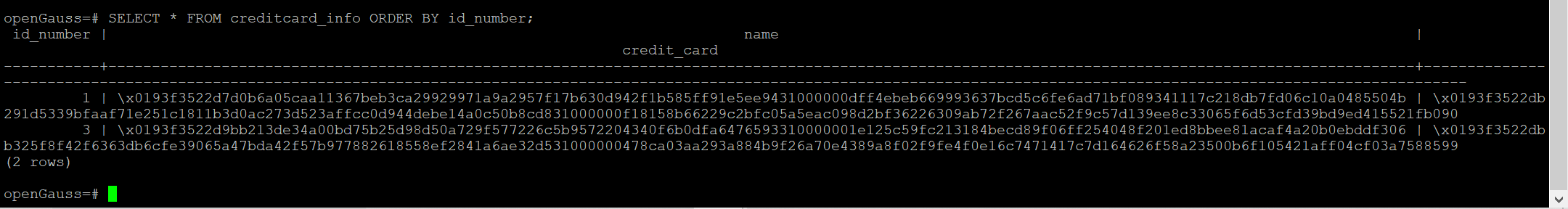
1. 将加密表和非加密表查询结果截图：



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明文，数据的加解密的动作是在客户端完成的。