openGauss 安全体系创新

实践课



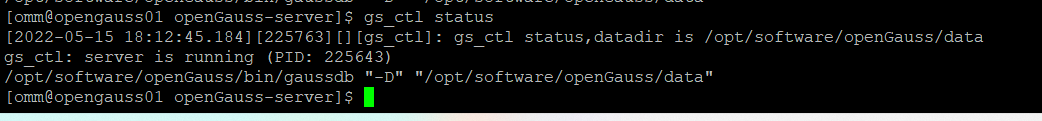
华为技术有限公司

# 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：

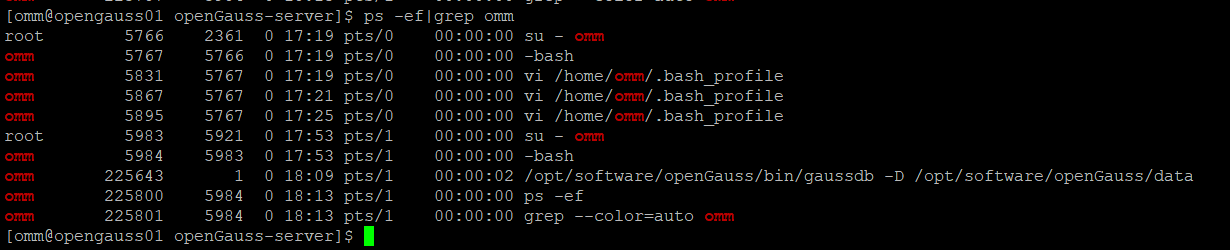
任务一：数据库状态验证

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、源码安装的优点，编译安装过程，可以设定参数，按照需求，进行安装，并且安装的版本，可以自己选择，灵活性比较大。

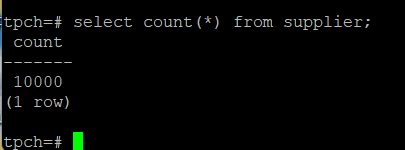
2、源码编译安装可以快速更新，不需要等待打包。

# 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：

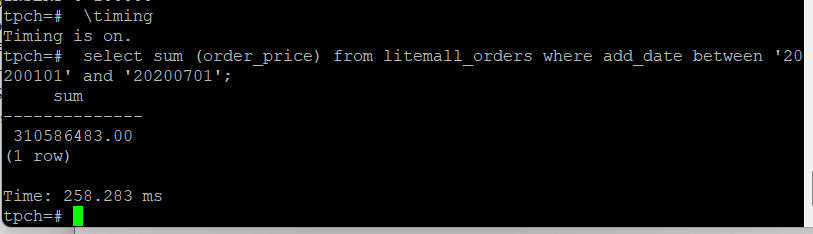
select count(\*) from supplier;;



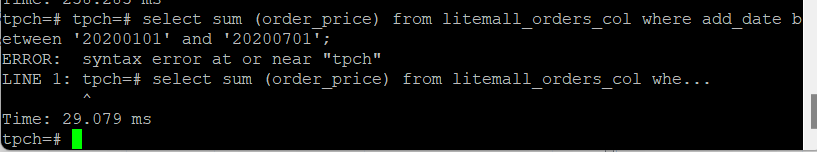
任务二：行存表与列存表执行效率对比

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';

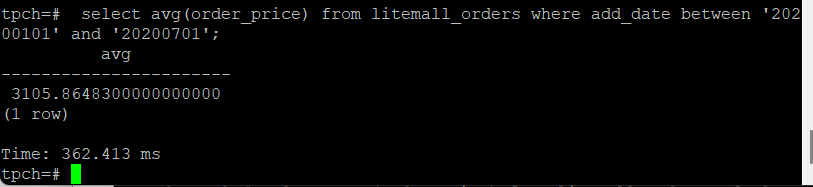


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

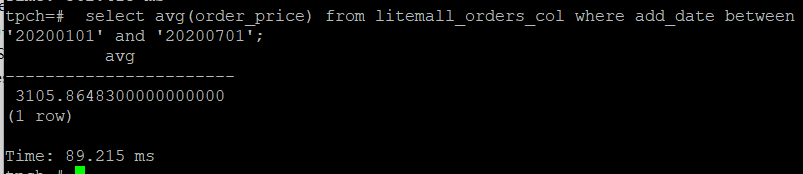


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

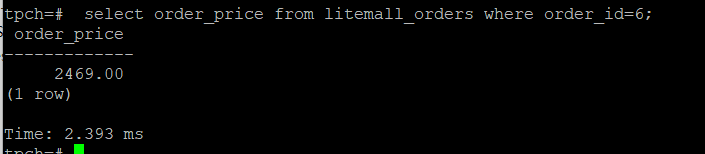


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

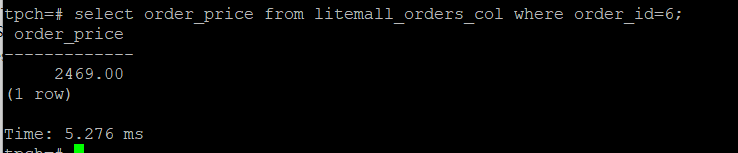


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

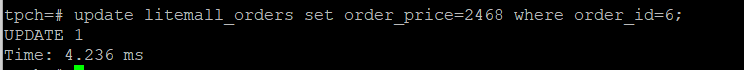


select order\_price from litemall\_orders\_col where order\_id=6;

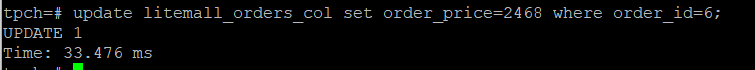


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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？

数据写入（修改）：

1、行存储的写入是一次完成。如果这种写入建立在操作系统的文件系统上，可以保证写入过程的成功或者失败，数据的完整性因此可以确定。

2、列存储由于需要把一行记录拆分成单列保存，写入次数明显比行存储多（意味着磁头调度次数多，而磁头调度是需要时间的，一般在1ms~10ms)，再加上磁头需要在盘片上移动和定位花费的时间，实际时间消耗会更大。所以，行存储在写入上占有很大的优势。

3、数据修改,这实际也是一次写入过程。不同的是，数据修改是对磁盘上的记录做删除标记。行存储是在指定位置写入一次，列存储是将磁盘定位到多个列上分别写入，这个过程仍是行存储的列数倍。所以，数据修改也是以行存储占优。

数据查询：

1、数据读取时，行存储通常将一行数据完全读出，如果只需要其中几列数据的情况，就会存在冗余列，出于缩短处理时间的考量，消除冗余列的过程通常是在内存中进行的。

2、列存储每次读取的数据是集合的一段或者全部，不存在冗余性问题。

列式数据库数据类型一致,数据特征相似,可以高效压缩。比如有增量压缩、前缀压缩算法都是基于列存储的类型定制的,所以可以大幅度提高压缩比,有利于存储和网络输出数据带宽的消耗。

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？

行存表效率更高：

1、随机的增删改查操作。

2、需要在行中选取所有属性的查询操作。

3、需要频繁插入或更新的操作。

列存表效率更高：

1、对列进行操作（比如求均值、求和等等）。

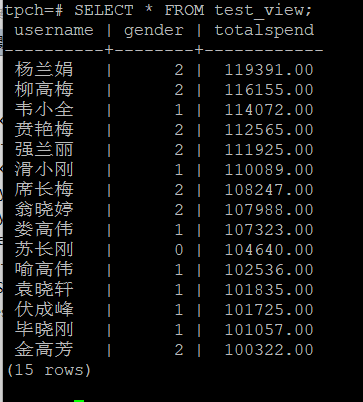
2、在对大量数据进行查询地过程中，如果只关心某些数据列。

# 关卡三：openGauss物化视图应用

任务一：物化视图的使用

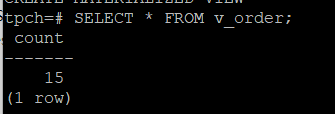
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



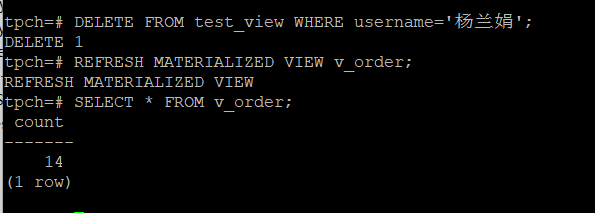
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



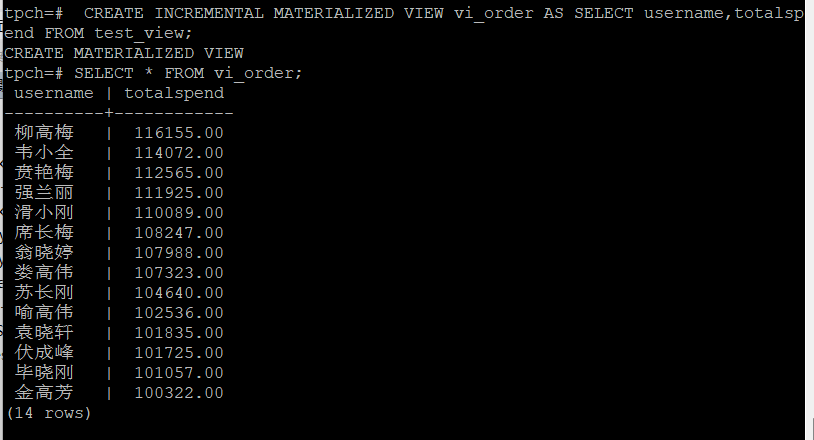
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



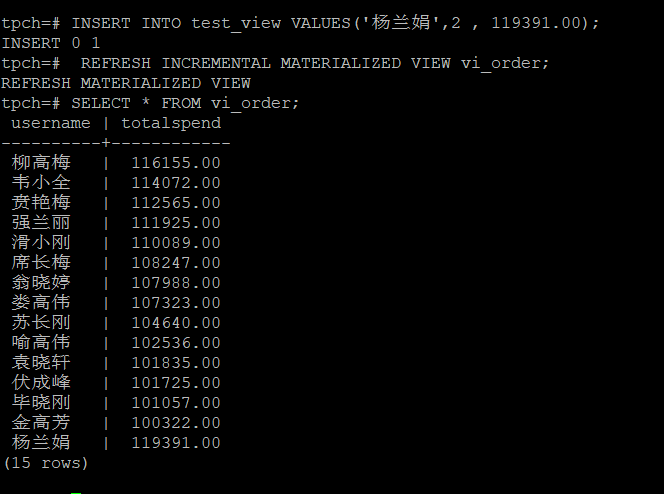
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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

1、全量物化视图所支持的查询范围与CREATE TABLE AS语句一致，适用范围广。

2、创建全量物化视图可以指定分布列。

3、全量物化视图不支持NodeGroup。

4、创建全量物化视图所使用的基表必须在所有DN上有定义，基表所属nodegroup必须为installation group。

5、全量物化视图不支持增量更新。

增量物化视图

1、仅支持单表查询语句和多个单表查询的UNION ALL。

2、增量物化视图会继承基表NodeGroup创建（检查各个基表是否在同一个NodeGroup，并基于这个NodeGroup进行创建）。

3、创建物化视图不可指定物化视图分布列。

4、仅支持行存表，不支持列存表。

5、支持增量更新。

思考题2：物化视图适用那些使用场景？

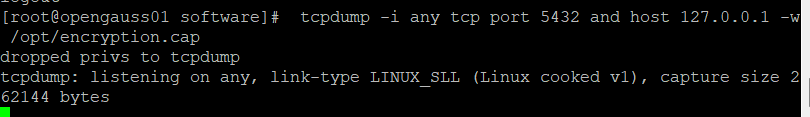
物化视图是有一个与之对应的容器表的。容器表是一个跟物化视图同名的“规则”的表，用于存储查询返回的结果集。这是物化视图与普通视图的根本区别，它是有储存结果集的“物理存在”的，而普通视图则没有这个物理存在，只是一个虚表，每访问一次，查询就要执行一次基表访问。

物化视图的应用场景有两种：1、用于查询优化 2、用于高级复制。

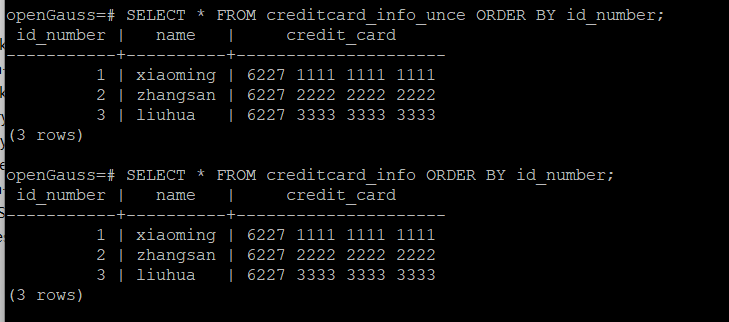
# 关卡四：openGauss密态数据库特性应用

任务一：物化视图的使用

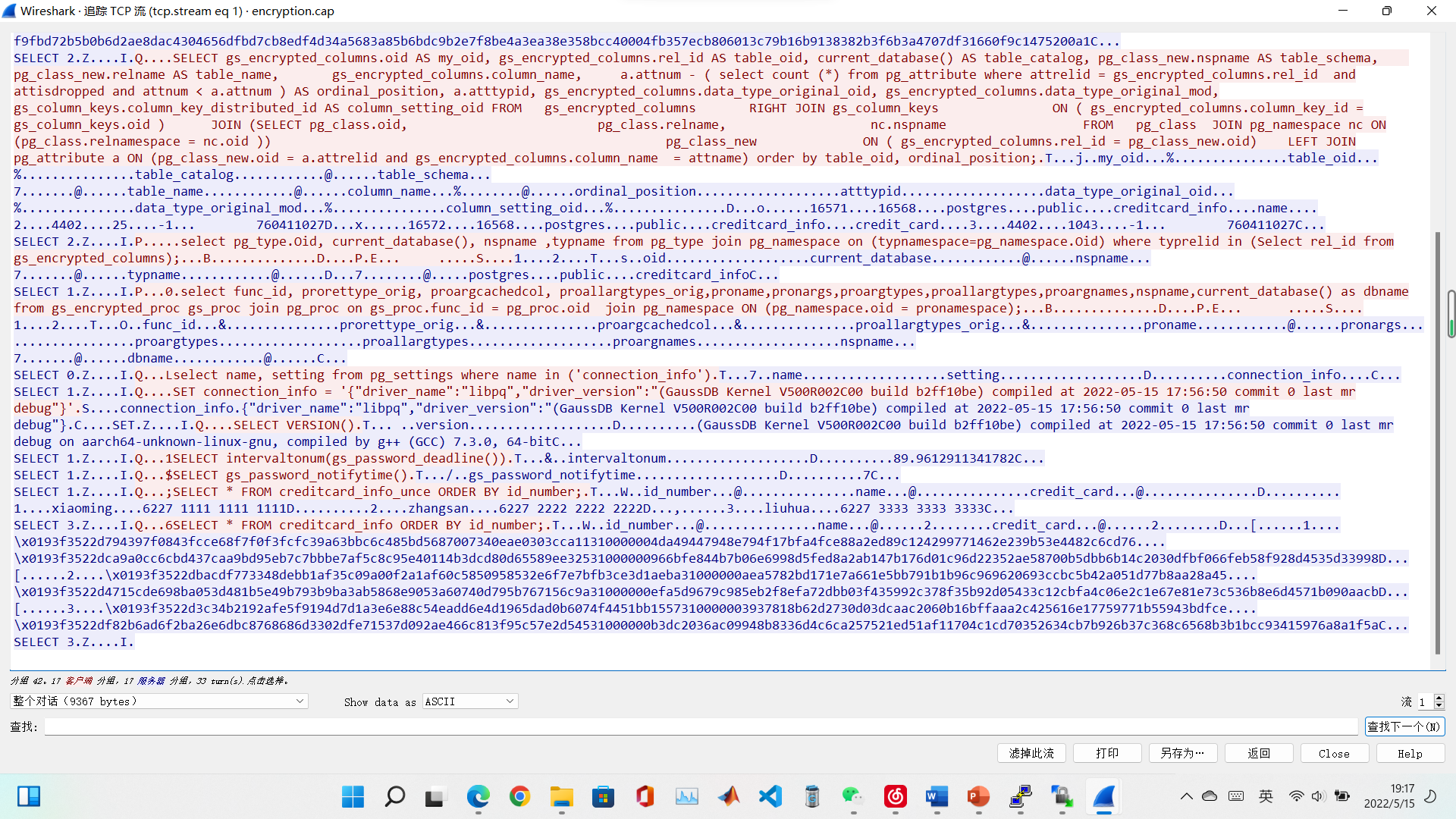
1. 通过tcpdump抓取数据流，此putty窗口暂时保持不动，将执行结果截图：



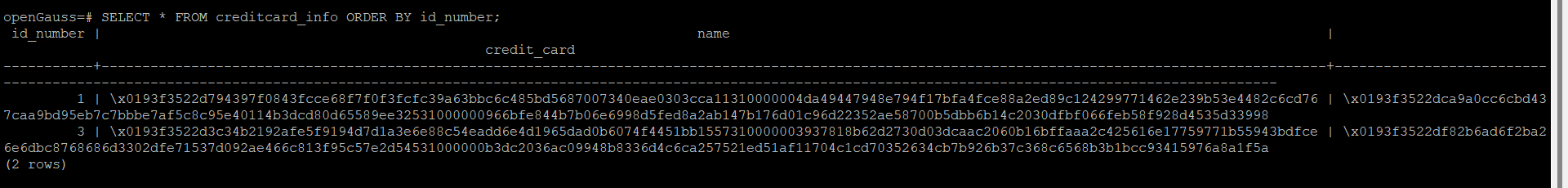
1. 将加密表和非加密表查询结果截图：



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密文。防止数据库数据泄露。

加解密的动作均在客户端完成。即使数据在传输过程中被截取也不会泄露信息，因为加密规则只有客户端知道。