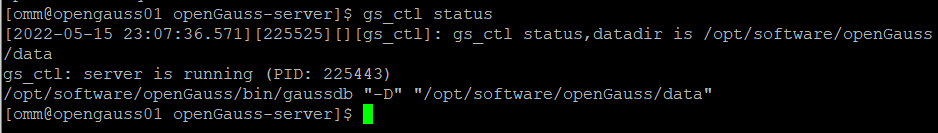
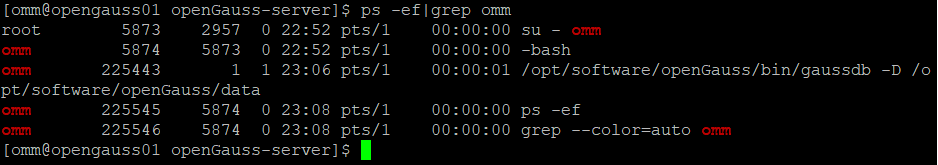
## 关卡一

### 2.2.3关卡验证

步骤 1此处需要通过截图，来完成关卡的验证，首先需要对数据库状态进行验证。



步骤 2其次，对数据库进程进行截图验证，需包含数据库服务器的主机名。



简单描述关卡一中，**安装数据库所需要的步骤**。完成以上信息，表明关卡一完成。

1. 在服务器上创建openGauss数据库安装用户omm和其属组dbgrp；
2. 创建openGauss源码存放及openGauss安装路径；
3. 下载使用第三方编译库并解压安装；
4. 下载openGauss源码，并将源码包上传到服务器指定文件夹中；
5. 对源码进行解压后，安装依赖包和替换某些软件版本；
6. 修改路径用户所属组以及权限；
7. 在创建的omm用户下设置添加环境变量；
8. 进入openGauss源码中生成配置文件后，使用make命令进行编译和安装；
9. 初始化数据库，设置数据库密码后可尝试启动和登录数据库。

### 2.2.4 思考题

**为什么需要通过源码编译，完成数据库的安装？**

数据要在Linux上执行，必须是二进制文件，因此当我们拿到源码后，需要将它编译成二进制文件才能在Linux上运行。使用源码包的好处除了可以自定义修改源代码外还可以定制相关的功能，因为源码包在编译的时候是可以附加额外的选项的。

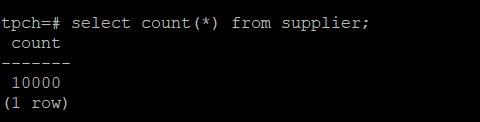
**安装数据库所需要的步骤：**

1. 在服务器上创建openGauss数据库安装用户omm和其属组dbgrp；
2. 创建openGauss源码存放及openGauss安装路径；
3. 下载使用第三方编译库并解压安装；
4. 下载openGauss源码，并将源码包上传到服务器指定文件夹中；
5. 对源码进行解压后，安装依赖包和替换某些软件版本；
6. 修改路径用户所属组以及权限；
7. 在创建的omm用户下设置添加环境变量；
8. 进入openGauss源码中生成配置文件后，使用make命令进行编译和安装；
9. 初始化数据库，设置数据库密码后可尝试启动和登录数据库。

## 关卡二

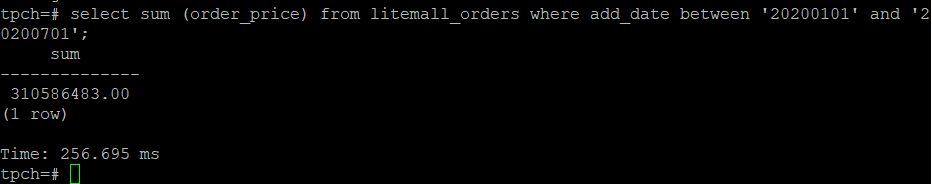
### 3.2.1 数据初始化

步骤12查询结果截图

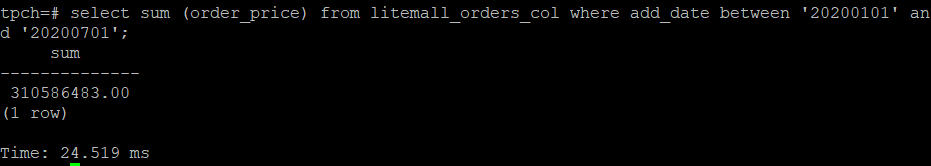


### 3.2.2 列存表的创建与使用

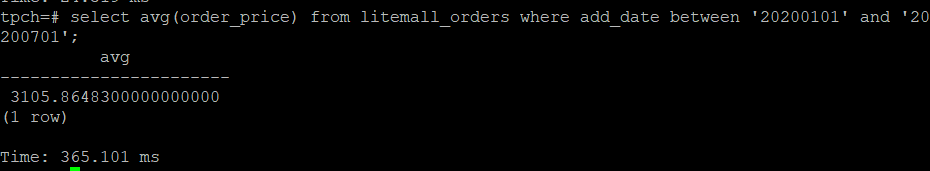
步骤 4使用SQL语句查询，2020年上半年litemall\_orders行存表中的order\_price的总和，并对结果和耗时截图。



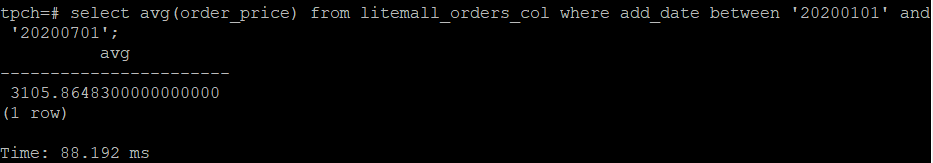
步骤 5使用SQL语句查询，2020年上半年litemall\_orders\_col列存表中的order\_price的总和，并对结果和耗时截图。



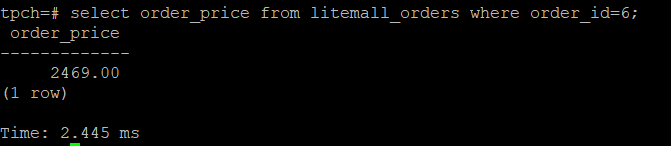
步骤 6使用SQL语句查询，2020年上半年litemall\_orders行存表中的order\_price的平均值，并对结果和耗时截图。



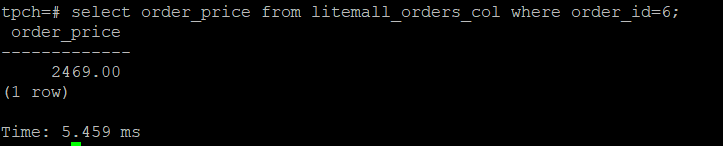
步骤 7使用SQL语句查询，2020年上半年litemall\_orders\_col列存表中的order\_price的平均值，并对结果和耗时截图。



步骤 8使用SQL语句查询，litemall\_orders行存表中order\_id为6的order\_price的值，并对结果和耗时截图。



步骤 9使用SQL语句查询，litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对结果和耗时截图。



步骤 10使用SQL语句，将litemall\_orders行存表中order\_id为6的order\_price修改为2468，并对结果和耗时截图。



步骤 11使用SQL语句，将litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对结果和耗时截图。



**对比在行存表与列存表中的操作，根据每种SQL执行的时间，得出结论。**

### 3.2.3 关卡验证

**本关卡需要将行存与列存执行时间对比进行截图，同时通过对比，得出使用两种不同的存储所适用的场景。**

1. **对比行存表与列存表每种SQL执行的时间，得出结论：**

对于修改单行语句，行存表会比列存表要快。

对于查询语句，如果是查询某行的具体数据，则行存表会比列存表要快；如果是使用子查询访问查询多行数据或者使用了聚集函数，则列存表会比行存表更快。

1. **两种不同的存储的适用场景：**
   1. 行存表适用于查询、修改数据表中的单行数据；
   2. 列存表适用于需要访问、查询多行数据时，包括使用子查询和聚集函数的情况。

### 3.2.4 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？**

在写入时，行存表是一次将所有数据写入指定位置；列存表是将磁盘定位到多个列上分别写入。

在读取出数据时，行存表通常是将一行数据完全读出，如果需要其中的几列数据，就会存在数据冗余的情况；列存表每次读取的数据是列集合中的一段或者全部。

因此，在访问某行数据时，行存表会读取速度更快。

由于列存表每列存储的数据同质的，因此在对多行中的同一列数据中进行解析的时更加容易；而对于列存表中每行存储了多种类型的数据，数据解析需要在多种数据类型之间频繁转换，效率降低。

这就决定了列存储在执行聚集函数和返回多列数据时效率更高，执行时间更短。

1. **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. SELECT FROM XX WHERE “属性”=XXX；#访问具体单行数据的语句；
2. UPDATE/DROP 修改或删除某行数据的语句；
3. INSERT INTO 向表中插入行数据的语句。

**列存表效率更高：**

1. 集合运算、涉及到需要返回多行、多列表格的SQL查询语句、使用聚集函数的语句；
2. ORDER BY、GROUP BY、HAVING子句等需要对多行、多组中某列数据进行比较、组织的语句；
3. 多表连接、创建视图的语句；

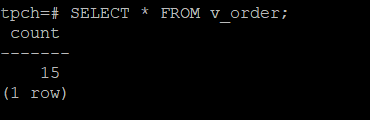
## 关卡三

### 4.1.3.1 全量物化视图

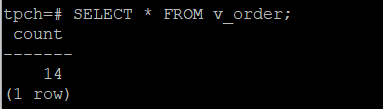
步骤 4对表内容进行查询，将执行结果截图。



步骤 6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。



步骤 9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。



### 4.1.3.2 增量物化视图

步骤 2查询物化视图结果，将执行结果截图。



步骤 5查询物化视图结果，将执行结果截图。



步骤 8查询物化视图结果，将执行结果截图。



### 4.1.4 关卡验证

**得出全量物化视图与增量物化视图的区别。**

全量物化视图仅支持对已创建的物化视图进行全量更新，而不支持进行增量更新；

增量物化视图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刷新。

### 4.1.5 思考题

1. **全量物化视图与增量物化视图有哪些差别？**

全量物化视图仅支持对已创建的物化视图进行全量更新，而不支持进行增量更新；

增量物化视图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刷新。

1. **物化视图适用那些使用场景？**

相比较于普通视图，物化视图不是保存查询语句，而是将查询结果存储到了磁盘或者内存中。

其优点为查询速度快，不用在频繁访问视图时重复执行视图的查询语句。因此物化视图适用于对某些查询结果需要频繁访问并且对查询速度有一定要求的场景。

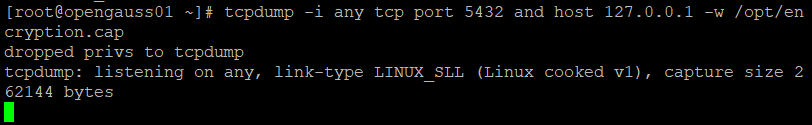
但由于物化视图的本质是一个流式数据的使用场景，也就是累加式的技术。如果一张表上累加了许多物化视图，那么在写表的时候需要消耗机器的大量资源。并且对于需要频繁更新的表，在使用物化视图进行增量更新的时候，也需要频繁执行刷新语句。在大量使用物化视图时，也要考虑到机器的数据带宽和存储容量。

## 关卡四

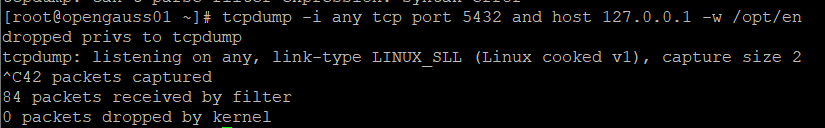
### 5.2.2.1 密态数据库

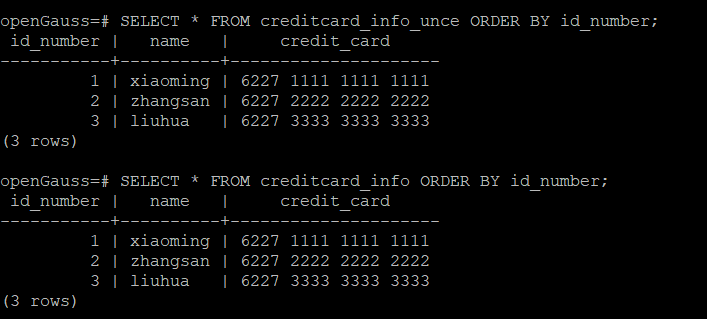
步骤 9通过tcpdump抓取数据流。5432是数据库端口号，w指把数据包数据写入指定的文件。

此putty窗口暂时保持不动，具体如下，将执行结果截图：

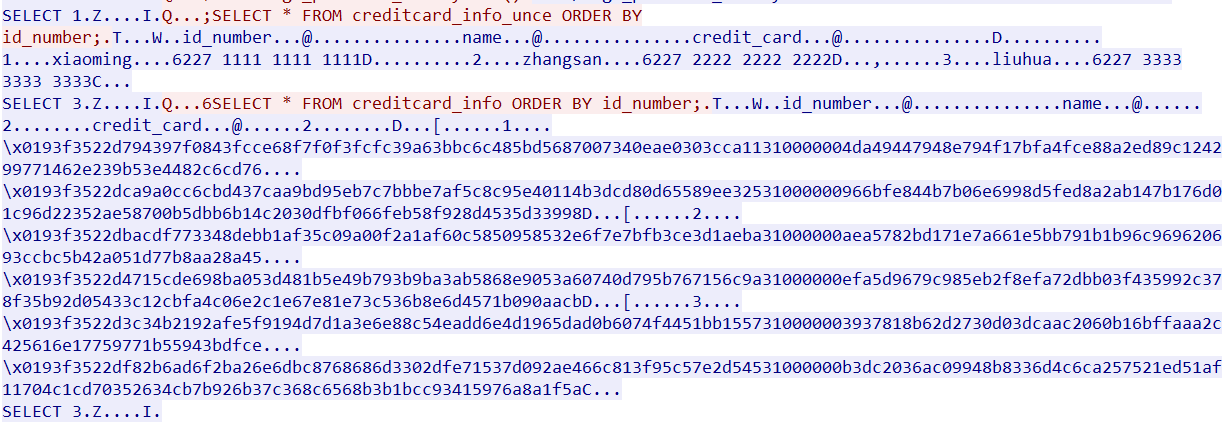


步骤10加密表和非加密表查询结果如下，将执行结果截图：

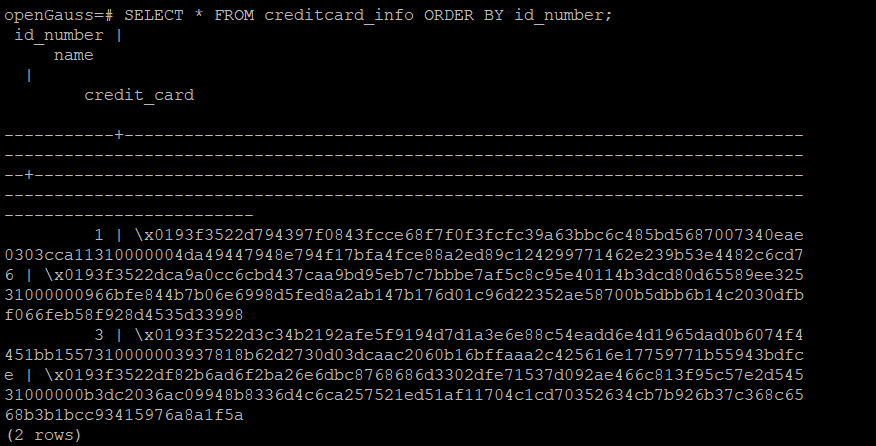




步骤15把右边的滚动条拉到最底端，可以发现虽然步骤10的gsql窗口查出的结果都是明文，但实际的数据流却不一样，非加密表name列和credit\_card列是明文，加密表name列和credit\_card列均是密文，结果如下图所示，将执行结果截图：



步骤 22查询加密表，查询到的结果为密文，将执行结果截图。



### 5.2.3 关卡验证

**完成密态数据库的使用步骤并截图，对比密态数据库场景下和非密态数据库场景下对加密表做增删改查的区别。**

在密态数据库场景下对加密表做增删改查与在非非密态数据库场景下对非加密表进行增删改查的操作相同；而如果在非密态数据库场景下对加密表做增删改操作是无法成功的，对于加密表进行查询的结果返回也是密文。

### 5.2.4 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？**

数据实际存储在物理磁盘上时是密文。数据在经过加密后以密文形式存放在指定的目录下。加密的意义在于,通过一系列复杂的迭代计算,将原本的明文转换为随机的没有任何具体含义的字符串,即密文。当所使用的加密算法足够安全时,攻击者在有限的计算资源下将很难根据密文获取到明文信息。

在密态数据库场景下对数据进行加密时，openGauss会将与加密相关的元数据存储在系统表中，当处理敏感数据时，客户端会自动检索加密相关元数据并对数据进行加解密。这保证了敏感数据明文不在除客户端驱动外的地方存在。

1. **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？**

数据在客户端就被加密，从客户端传输到数据库内核，到在内核中完成查询运算，到返回结果给客户端，数据始终处于加密状态，最后再由客户端进行解密。