openGauss 安全体系创新

实践课



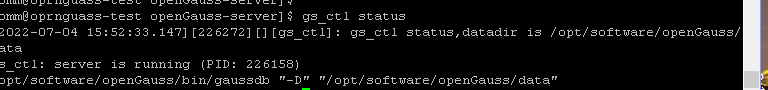
华为技术有限公司

# 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：

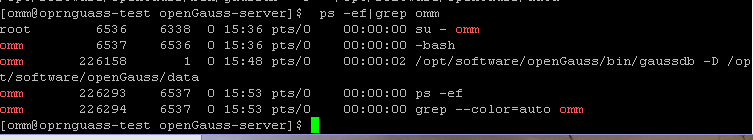
任务一：数据库状态验证

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？

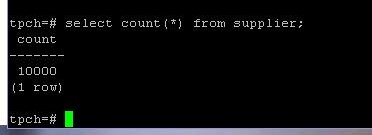
可以满足不同的运行平台，编译过程可以进行参数设定，方便快捷，不用考虑包依赖，安装的版本，可以自己选择，灵活性比较大。

# 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：

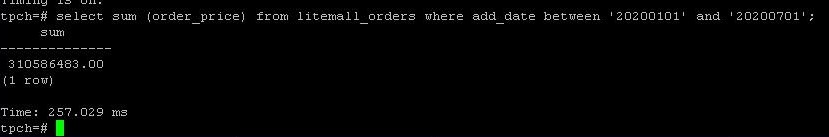
select count(\*) from supplier;;



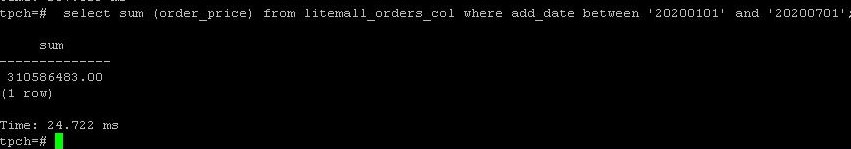
任务二：行存表与列存表执行效率对比

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';

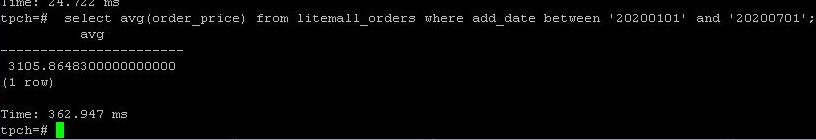


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

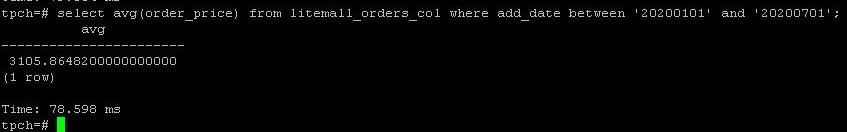


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

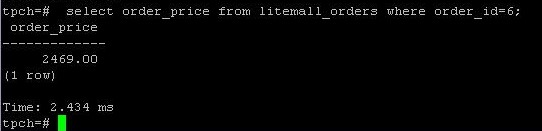


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

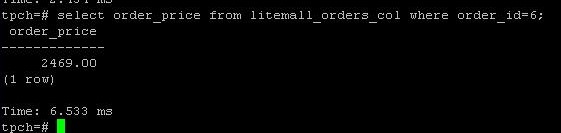


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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



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

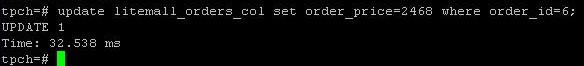


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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？

行存储通常将一行数据完全读出，如果只需要其中几列数据，就会存在冗余列，一行记录中保存了多种类型的数据，数据解析消耗CPU。

列存储每次读取的数据是集合中的一段或者全部。

二者执行时进行的工作有差别，执行的时间也不同。

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？

当执行查询或修改命令只需要对部分数据，如order\_id=6进行操作时，行存表的效率更高。当查询某一属性在全部数据中的内容时，列存表的效率更高。

# 关卡三：openGauss物化视图应用

任务一：物化视图的使用

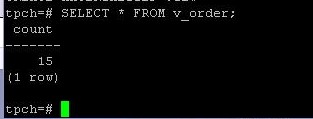
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



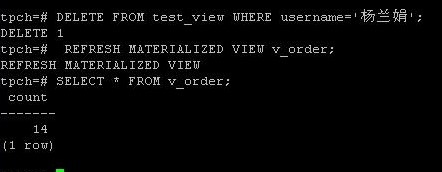
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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;



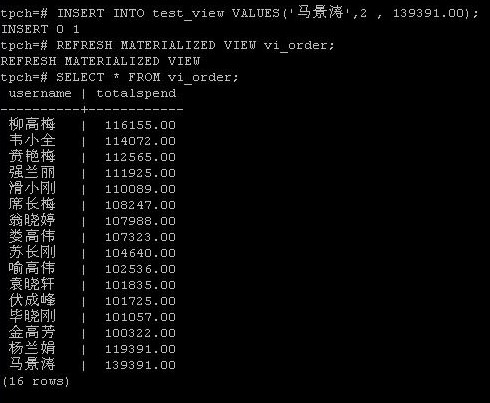
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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？

全量物化视图可以反应总体数据的大小情况，增量物化视图可以反应数据的增减变化情况。

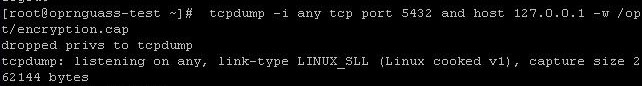
思考题2：物化视图适用那些使用场景？

在需要性能优化的场景，高级复制的时候，物化视图具有更高的效率，所以此时可以使用物化视图。

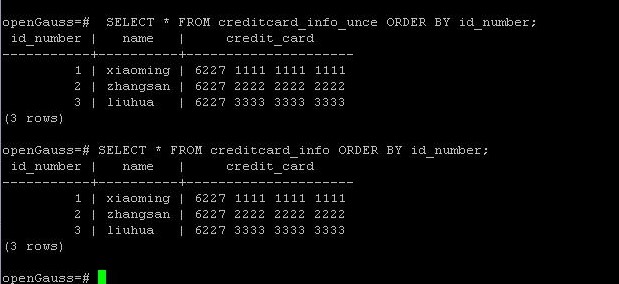
# 关卡四：openGauss密态数据库特性应用

任务一：物化视图的使用

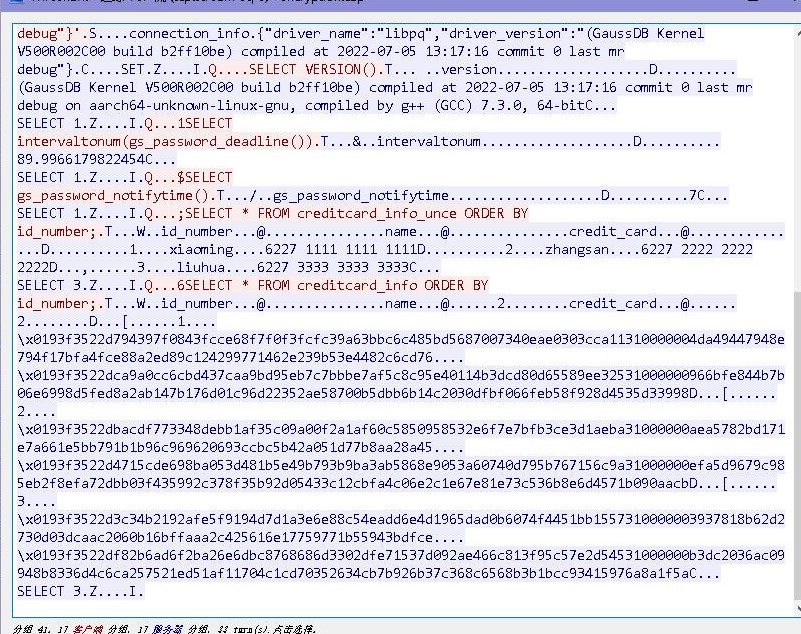
1. 通过tcpdump抓取数据流，此putty窗口暂时保持不动，将执行结果截图：



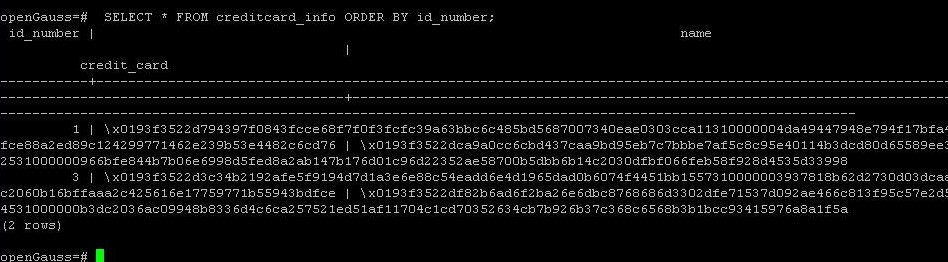
1. 将加密表和非加密表查询结果截图：



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？

实际存储的时候是以明文的方式进行存储，可以使用grep 命令，在数据目录下查询到对明文

信息。

数据在客户端完成加密，以密文形式发送到 GaussDB Kernel 数据库服务侧，即需要在客户端构建加解密模块。在数据库服务侧，加密列的数据始终以密文形态存在，整个查询也在密文形态下实现。用户根据业务需要对数据定义加密属性信息(被加密的列被称之为加密列)，对于不需要加密的数据则按照原有明文格式发送至服务端。当查询任务发起后，客户端需要对当前的 Query 进行解析，如果查询语句中涉及加密列，则对对应的列参数(加密列关联参数)也要进行加密(这里说的加密均需要为确定性加密，否则无法支持对应的查询），如果查询语句中不涉及加密列，则直接发送至服务端，无需额外的操作。