openGauss 安全体系创新

实践课



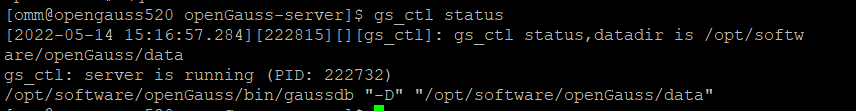
华为技术有限公司

# 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：

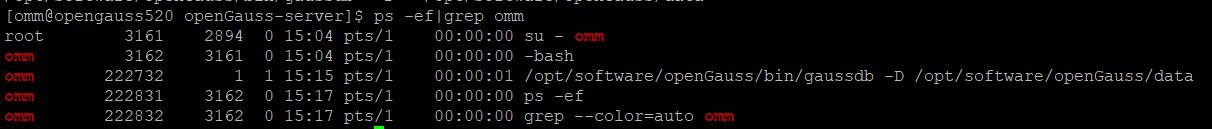
任务一：数据库状态验证

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？

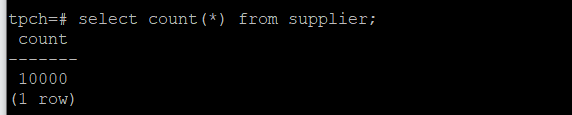
1. 源码编译安装可以进行定制化编译参数设定，能更确切的满足需求，包管理器安装不呢自定义参数安装，不能完全符合需求；
2. 源码编译安装一般可以安装最新的软件版本，在选择安装版本时，有更多的选择，而包管理器安装则只能选择被封装的版本。
3. 源码编译安装可以自定义部署路径，进行统一管理，而包管理器安装不能自定义部署路径，后期维护工作量会更大；
4. 如果通过源码编译安装到用户目录，则安装时不需要有管理员权限，也不必考虑会对系统造成的污染；但包管理器安装则必须要求管理员权限，如果没有管理员权限将无法进行安装。

# 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：

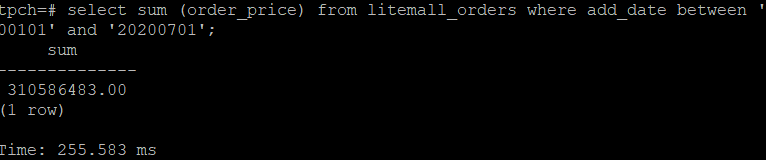
select count(\*) from supplier;;



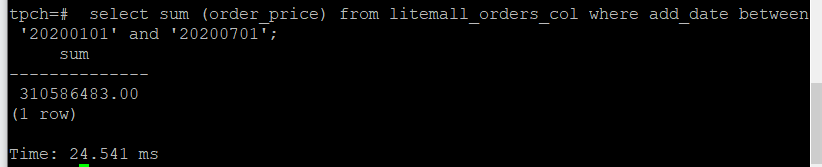
任务二：行存表与列存表执行效率对比

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';

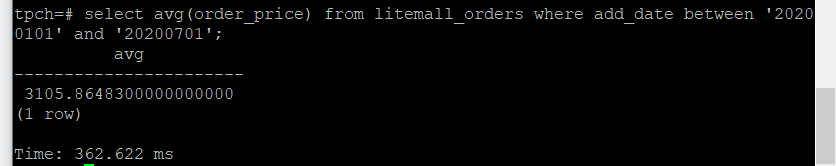


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

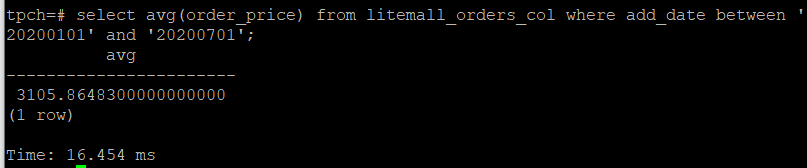


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

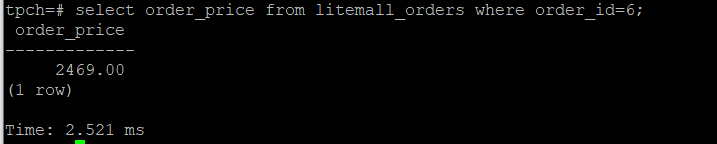


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

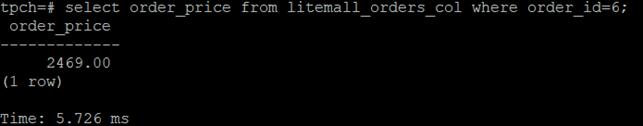


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

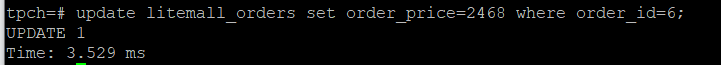


select order\_price from litemall\_orders\_col where order\_id=6;

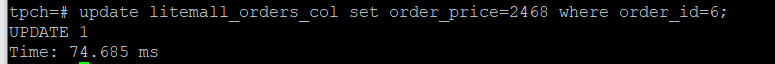


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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？

对于数据读取而言：

行存表通常将一行数据完全读出，若只需要其中部分列，则会存在冗余列；而列存表每次读取的数据是一列的一段或者全部，不存在冗余列。行存表的一行数据中包含不同类型的数据，在数据解析时需要在多种数据类型之间进行转换，而列存表中的每一列数据类型都是相同的，数据解析更容。行存表若没有建立索引，查询时要耗费大量的I/O，但建立索引又要花费大量的时间与资源；而列存表中的数据本身即是索引，大大降低了I/O。

对于数据写入而言：

行存表的写入是一次性完成一行的写，而列存表需要把一行记录拆分为不同列保存，相比之下要花费更多的写入次数。行存表在数据修改时对磁盘上指定位置做删除与写入一次；而列存表需要将磁盘定位到多个列上分别删除与写入。

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？

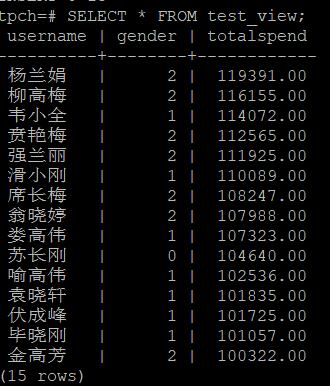
综上，在执行较多数据读取的SQL时，列存表的效率更高，在执行较多数据写入的SQL时，行存表的效率更高

# 关卡三：openGauss物化视图应用

任务一：物化视图的使用

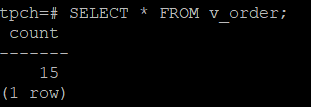
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



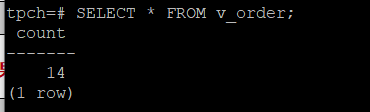
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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？

1. 全量物化视图即仅支持对创建好的物化视图做全量更新，而不支持做增量更新；而增量物化视图可以对物化视图进行增量更新，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。增量更新建立在全量更新的基础之上；
2. 目前而言全量物化视图所支持的功能与场景较广泛，而增量物化视图所支持场景较小；
3. 全量更新需要一次性将全部数据进行同步，因此需要占用大量时间和资源；而增量更新只抓取某个检查点之后的数据进行更新，占用时间与资源相对较少，但需要更新端记录或可以查询更新日志确定更新部分的数据。

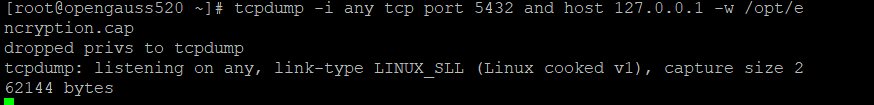
思考题2：物化视图适用那些使用场景？

物化视图的应用场景有两种：1、用于查询优化 2、用于高级复制

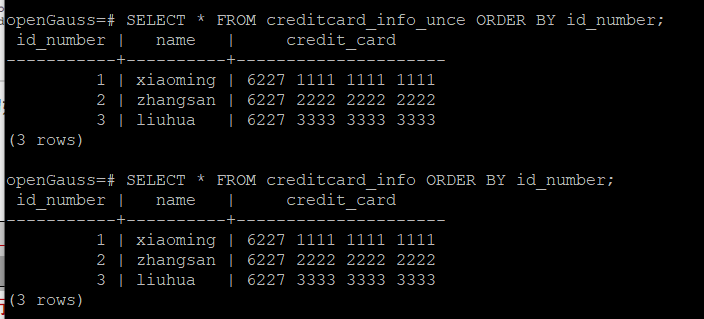
# 关卡四：openGauss密态数据库特性应用

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：



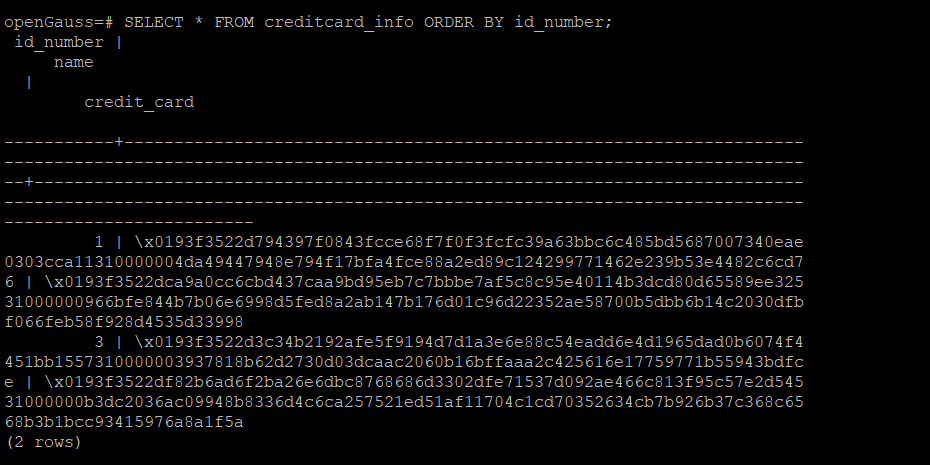
1. 将加密表和非加密表查询结果截图：



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？

数据存储在物理磁盘上是密文，加密动作在客户端完成。