openGauss 安全体系创新

实践课



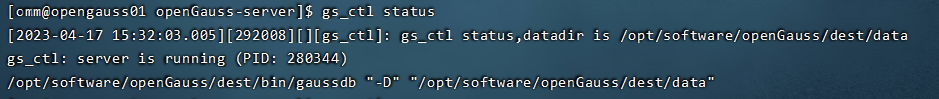
华为技术有限公司

# 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：

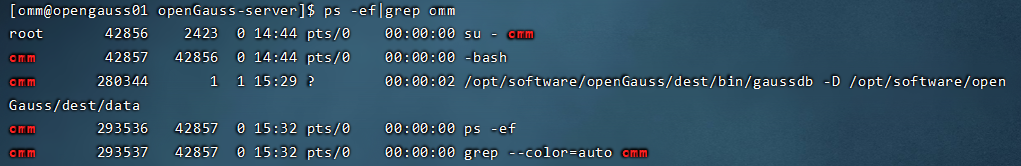
任务一：数据库状态验证

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？

答：为了满足不同操作系统和硬件平台的要求，若操作系统没有预编译好的二进制包，而且不同的操作系统和硬件平台具有不同的二进制包，则需要通过源码编译来生成适合当前环境的二进制文件。

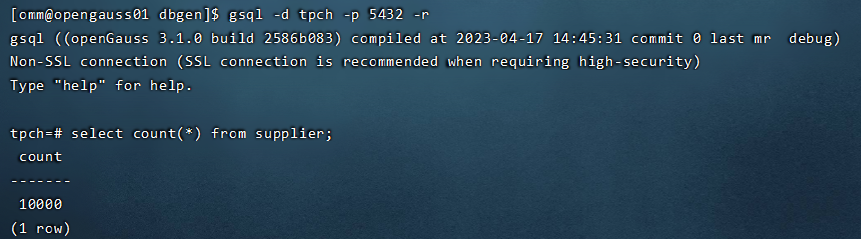
同时通过源码编译可以自定义设置安装数据库的各种功能模块；同时如果数据库程序若没有二进制安装包，则需要通过源码编译进行安装。

# 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：

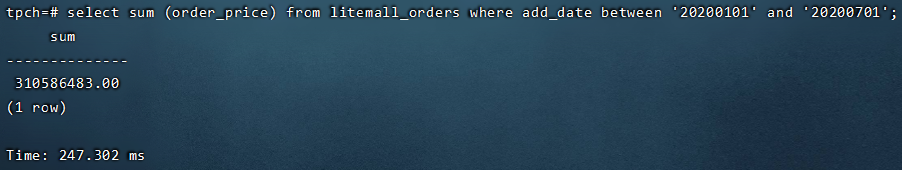
select count(\*) from supplier;;



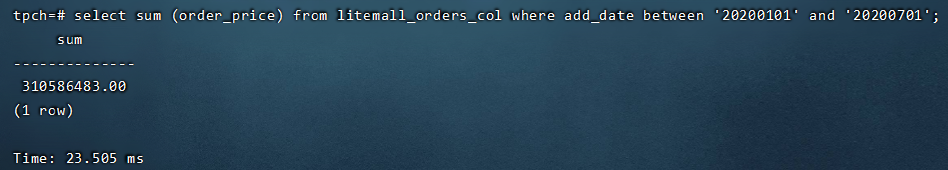
任务二：行存表与列存表执行效率对比

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';

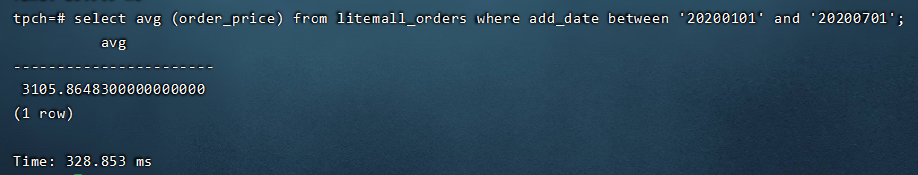


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

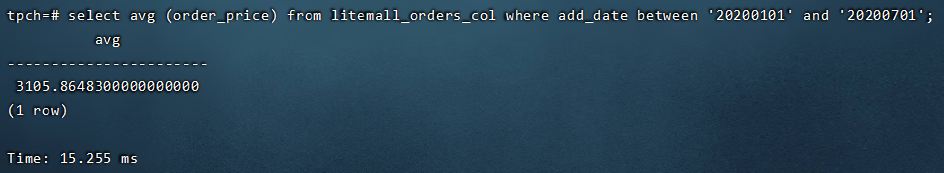


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

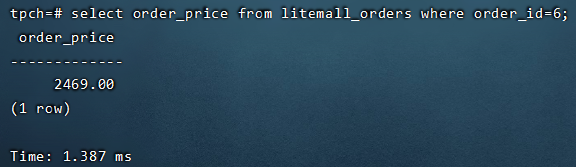


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

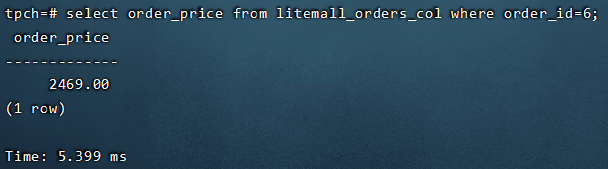


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

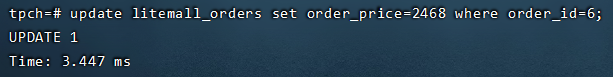


select order\_price from litemall\_orders\_col where order\_id=6;

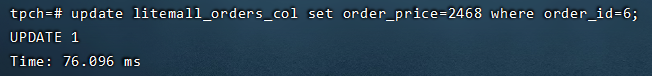


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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？

答：因为行存表与列存表在执行相同的SQL语句时的执行逻辑。行存表在执行写入逻辑的SQL语句时候，写入是一次完成的，数据修改时，在指定位置写入一次；而列存表数据修改时，在指定位置写入一次。

行存表在执行读取逻辑的SQL语句时将一列数据完全读出，如果需要其中几列的数据，就会存在冗余列，出去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；而列存表每次读取的数据是集合的一段或者全部，不存在冗余问题。

因此行存表与列存表在执行相同的SQL语句时执行的时间不同。

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？

答：执行对写入性能、写入完整性要求比较高，但是对查询性能要求不高的SQL时，行存表效率更高；

执行对写入性能、写入完整性要求不高，但是对查询性能要求很高的SQL时，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

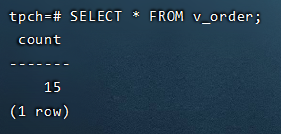
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



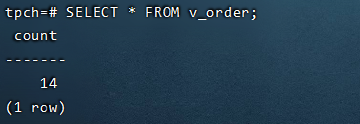
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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;



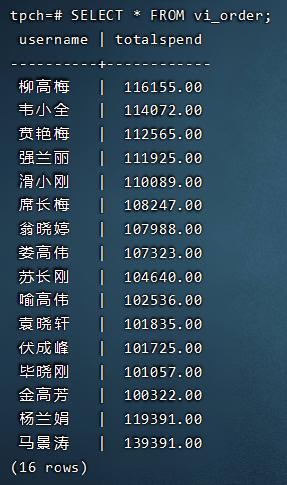
5. 对表进行操作后，刷新增量物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM vi\_order;

增量刷新物化视图：



全量刷新物化视图：



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？

答：全量物化视图无法自动更新，需要手动触发 REFRESH 命令才能更新。

增量物化视图每当有新的数据插入、更新或删除时，增量物化视图会自动更新，并保证查询结果与基表保持一致，不需要手动触发REFRESH命令。

思考题2：物化视图适用那些使用场景？

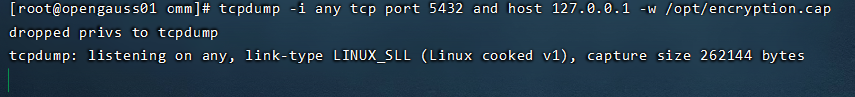
答：全量物化视图的使用场景一般是一些查询数据量比较大而且变化频率不高的情况，可以提升查询效率。

增量物化视图适用于数据变化比较频繁的场景，可以节省 REFRESH 操作的时间。

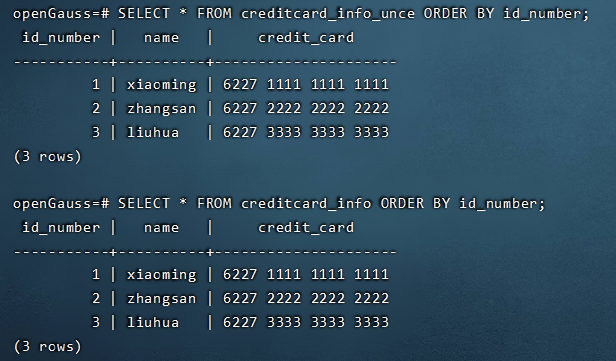
# 关卡四：openGauss密态数据库特性应用

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：



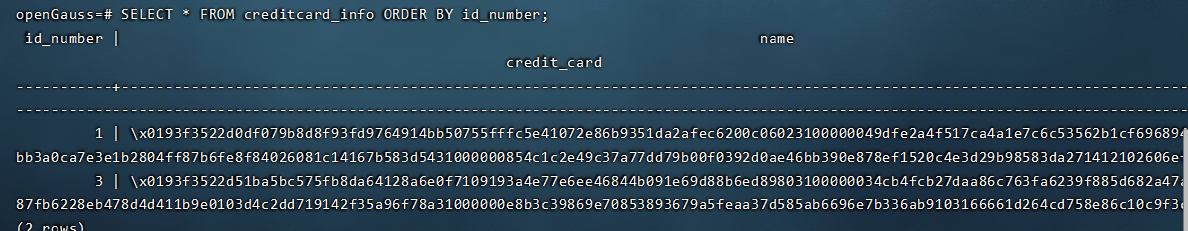
1. 将加密表和非加密表查询结果截图：



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？

答：数据实际存储在物理磁盘上的时候是密文。数据的加解密的动作是在服务端完成的。