openGauss 安全体系创新

实践课



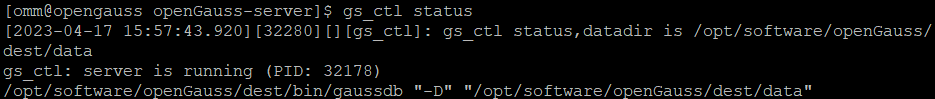
华为技术有限公司

# 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：

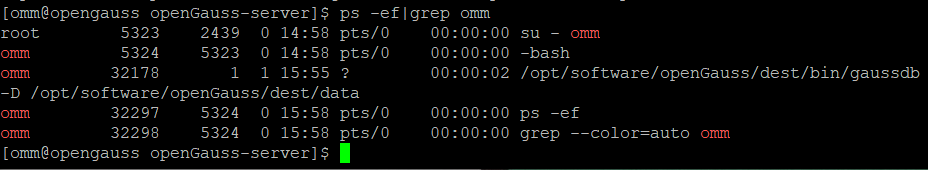
任务一：数据库状态验证

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？

通过源码编译安装数据库是因为源代码可以根据当前的硬件和运行环境进行编译，以提高性能和功能。另外，通过源码编译还可以灵活地选择需要安装的组件和配置参数。

安装数据库的步骤如下：

1、下载数据库源代码或安装包，并解压至指定目录。

2、安装编译工具及其他依赖库，例如GCC、make等。

3、进入解压后的目录，运行configure命令进行配置，指定安装路径、编译选项等。

4、运行make命令进行编译，此过程可能需要较长时间，具体时间因硬件配置和代码规模而异。

5、运行make install命令进行安装，此命令会将编译后的二进制文件和相关组件安装至指定路径。

6、配置数据库的环境变量、启动脚本等。

7、初始化数据库，创建数据表、用户、角色等。

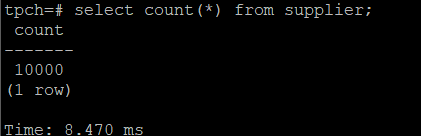
8、启动数据库，测试是否正常工作。

# 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：

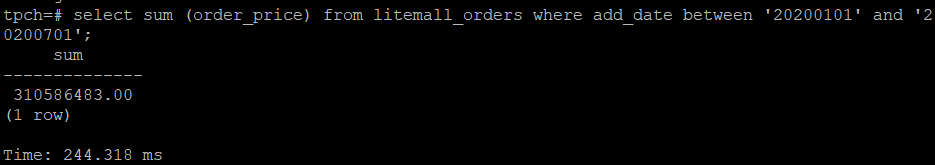
select count(\*) from supplier;;



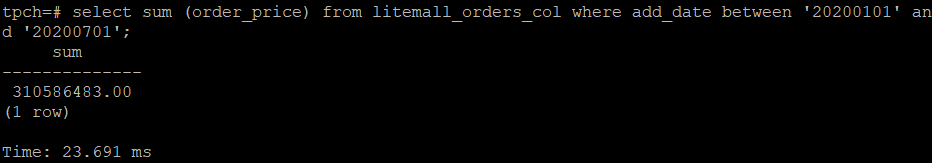
任务二：行存表与列存表执行效率对比

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';

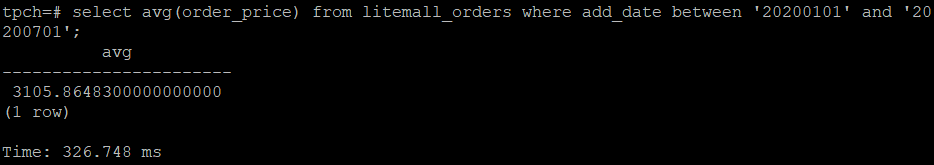


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

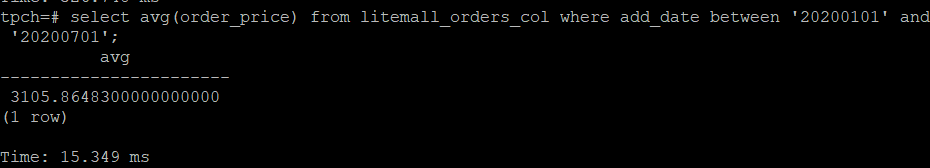


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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？

是它们在数据存储方式上有所不同。行存表以行为单位存储数据，即每行数据存储在一起，而列存表则按照列为单位存储数据，即将同一列的数据存储在一起。

对于一些特定的SQL语句（如聚合函数、分组查询等），列存表可以通过减少I/O操作和利用向量化运算等技术，在查询大量数据时具有明显的优势。这是因为列存表可以只读取所需的列而不必读取整行数据，从而减少了I/O操作和内存使用，提高了查询效率。而行存表则需要读取整行数据，无法利用向量化运算，从而在查询大量数据时效率较低。

但是对于一些随机访问或更新操作，由于列存表需要访问多个列才能完成一次操作，因此其效率会比行存表低。同时，由于列存表需要维护每个列的缩影(index)，因此在写入数据时需要花费更多的时间。

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？

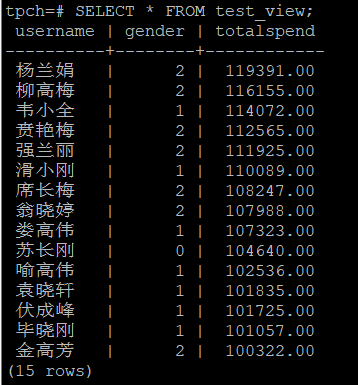
在执行单条记录的增删改查（CRUD）操作时，行存表效率较高。在执行单条记录的增删改查（CRUD）操作时，行存表效率较高。

# 关卡三：openGauss物化视图应用

任务一：物化视图的使用

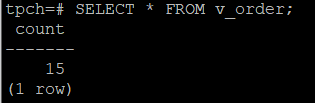
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



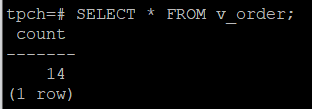
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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？

全量物化视图和增量物化视图的主要差别在于数据更新的方式和更新效率上。

全量物化视图的数据是一次性全部预先计算并存储在物化视图中，当源表的数据发生变化时，需要重新计算整个物化视图的数据。这种方式的优点是查询效率较高，因为物化视图中的数据已经是最终结果，可以直接从中查询。但是缺点是由于需要全量更新，当源表数据量大或者更新频繁时，更新操作将消耗大量的时间和资源。

而增量物化视图则是只计算和存储源表发生变化的部分数据，根据变化的情况更新物化视图，从而提高更新效率。增量物化视图通常需要结合触发器或日志等机制来实现，以识别源表发生的变化，并计算增量变化部分的数据。虽然增量物化视图的更新效率较高，但是对于复杂查询需要动态地计算物化视图，可能会降低查询效率。

因此，在选择全量物化视图或增量物化视图时，需要根据实际应用场景进行综合考虑，从而选择最适合的物化视图类型。

思考题2：物化视图适用那些使用场景？

物化视图适用于需要频繁查询、数据量大或数据结构复杂的应用场景

1、数据仓库查询优化：物化视图通常用于数据仓库等需要处理大量历史数据的场景，通过预先计算和存储查询结果，加快查询速度。

2、数据复杂性高：当源表的数据比较复杂，包含多表关联、分组统计等操作，而查询频率较高时，物化视图可以将数据处理结果缓存起来，减少每次查询时的计算量。

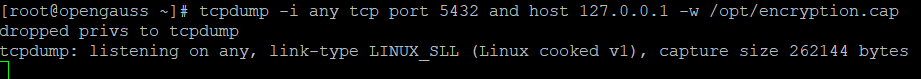
3、聚合查询需求多：当需要对大量数据进行聚合操作，如求和、求平均值等，物化视图可以提前计算并存储这些聚合结果，从而提高查询效率。

4、频繁的OLAP操作：物化视图可以在批处理过程中生成，使得系统可以在业务低峰期进行数据处理，并且物化视图可以作为一种缓存机制，减少对源表的频繁访问，提高性能

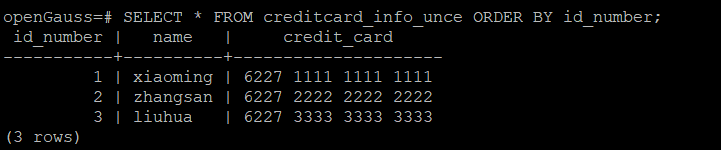
# 关卡四：openGauss密态数据库特性应用

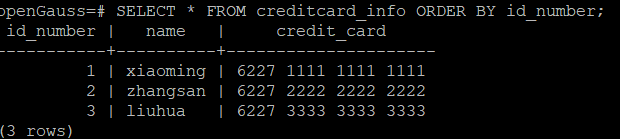
任务一：物化视图的使用

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：

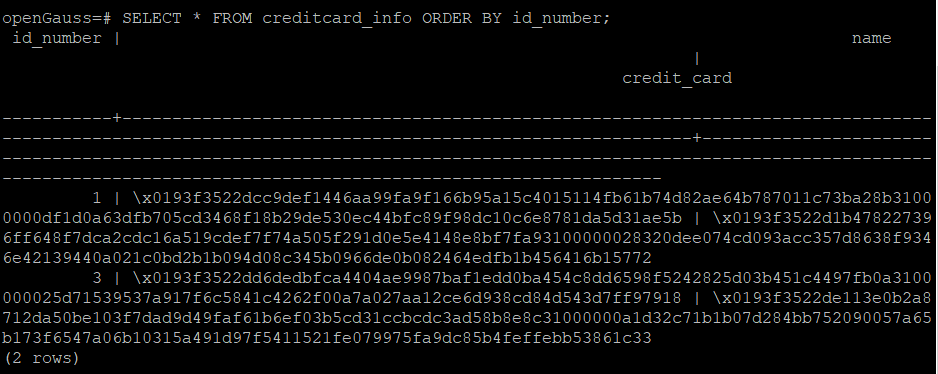




1. 用wireshark解析加密表和非加密表的差异时，非加密表name列和credit\_card列是明文，加密表name列和credit\_card列均是密文，将执行结果截图：



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？

密文。可能在客户端完成，也可能在服务端完成。通常情况下，HTTPS协议的传输过程中，加密过程是在客户端和服务端之间进行的，即客户端利用公钥加密信息，服务端利用私钥解密信息，并使用私钥对返回的数据进行数字签名，客户端再利用公钥解密和验证该数字签名。