openGauss 安全体系创新

实践课



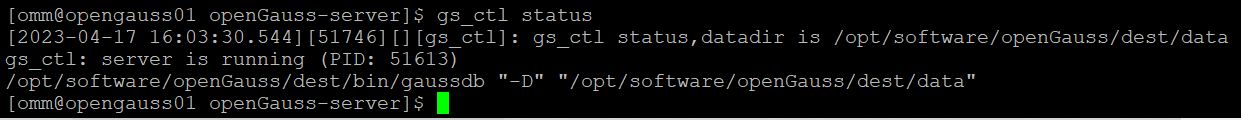
华为技术有限公司

# 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：

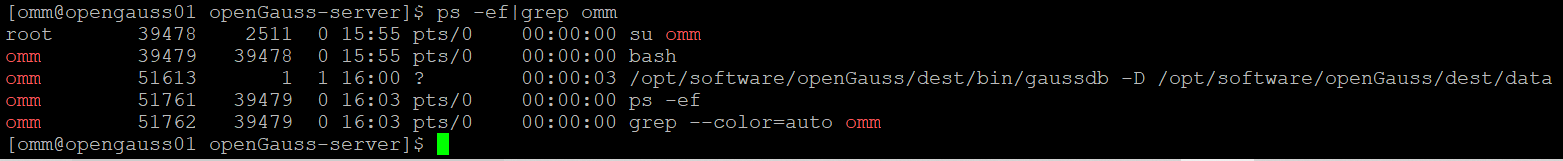
任务一：数据库状态验证

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？简单描述安装数据库所需要的步骤。

答：源码编译虽然安装过程较复杂、编译时间长，但源码编译与平台无关，可按需定制编译，最灵活，同时相较于RPM包和二进制包性能最好；同时可以自己选择安装参数、安装到自己想要安装的目录、单台服务器可以安装多个版本的软件。

安装步骤简述：

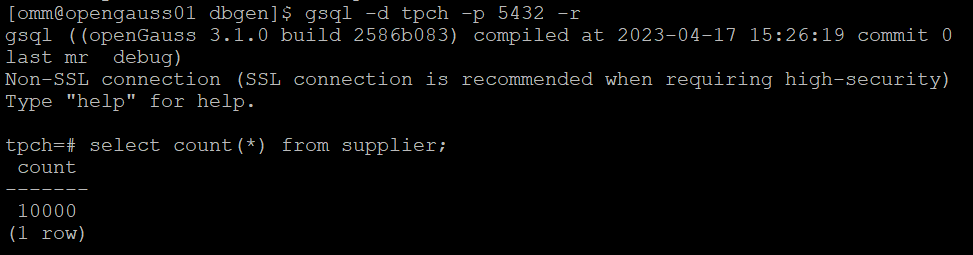
1. 创建omm用户，设置各项路径，下载第三方编译库、openGauss源码及各项安装依赖包（注意替换python版本）；
2. 切换omm用户，添加、重启环境变量，进入openGauss源码下，生成配置文件并编译、安装，初始化并启动数据库；
3. 登陆数据库查询数据库状态等以验证openGauss安装完成。

# 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：

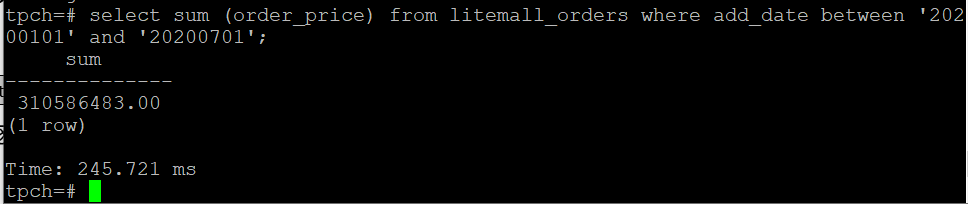
select count(\*) from supplier;;



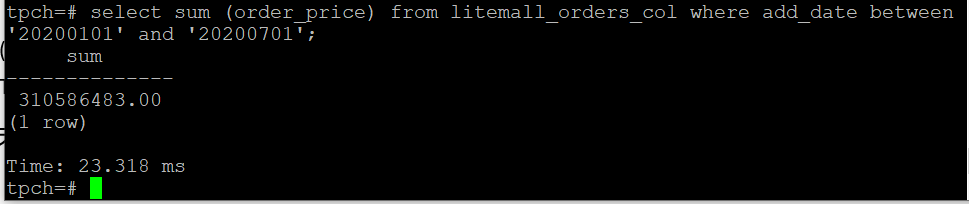
任务二：行存表与列存表执行效率对比

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';

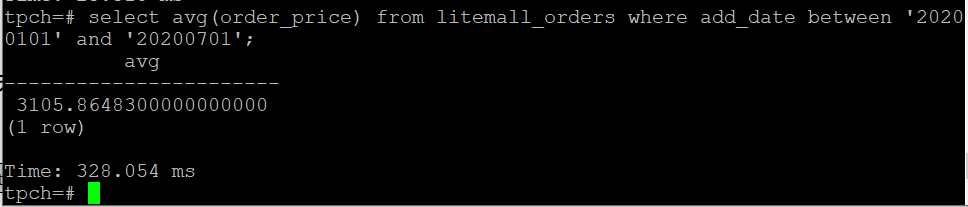


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

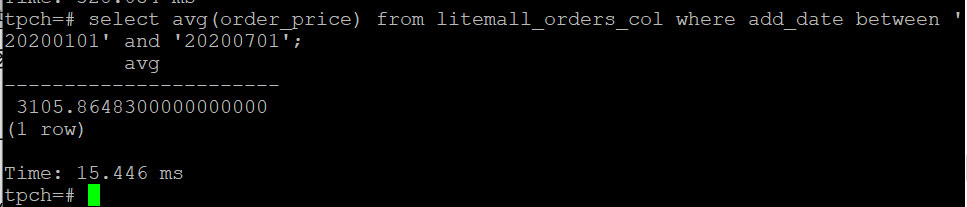


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

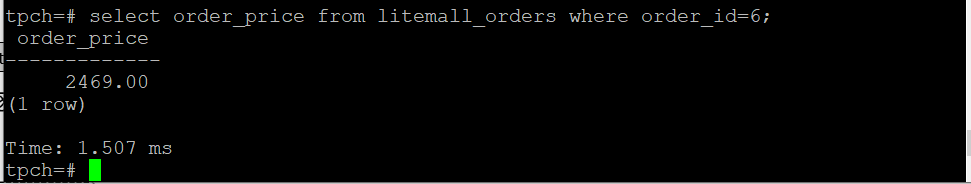


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

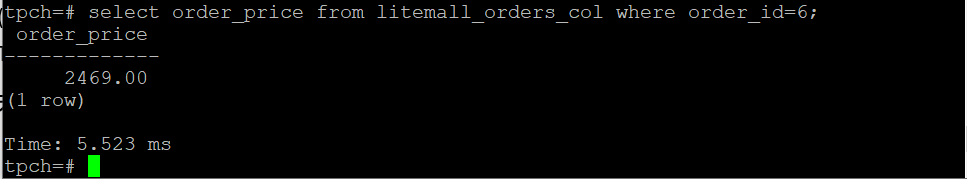


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

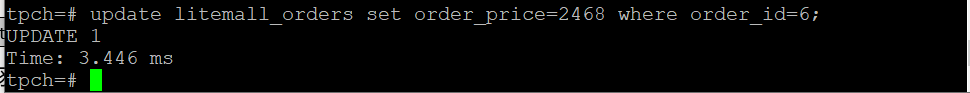


select order\_price from litemall\_orders\_col where order\_id=6;

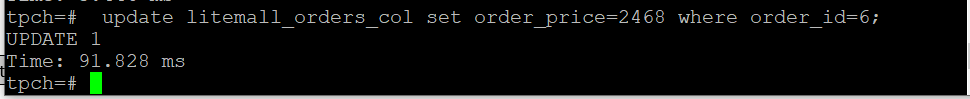


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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？

行式数据库是按照行存储的，行式数据库擅长随机读操作不适合用于大数据；列式数据库把一列中的数据值串在一起存储起来，然后再存储下一列的数据，以此类推。

查找数据时，行存表在数据列中高效查找数据，无需维护索引(任何列都能作为索引)，查询过程中能够尽量减少无关IO，避免全表扫描，因而性能优于列存表；而在需要在行中选取所有属性的查询操作、随机增删改查等操作上行存表优于列存表。

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？

行式存储（Row-based）的适用场景包括：

1、适合随机的增删改查操作;

2、需要在行中选取所有属性的查询操作;

3、需要频繁插入或更新的操作，其操作与索引和行的大小更为相关。

列式存储引擎的适用场景包括：

1、查询过程中，可针对各列的运算并发执行(SMP)，在内存中聚合完整记录集，可降低查询响应时间;

2、可在数据列中高效查找数据，无需维护索引(任何列都能作为索引)，查询过程中能够尽量减少无关IO，避免全表扫描;

3、因为各列独立存储，且数据类型已知，可以针对该列的数据类型、数据量大小等因素动态选择压缩算法，以提高物理存储利用率;如果某一行的某一列没有数据，那在列存储时，就可以不存储该列的值，这将比行式存储更节省空间。

# 关卡三：openGauss物化视图应用

任务一：物化视图的使用

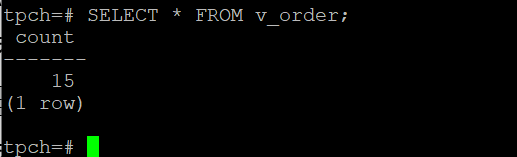
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



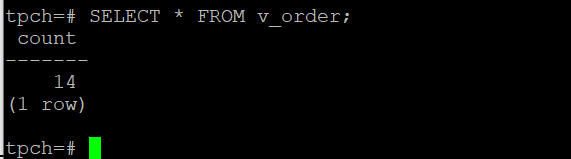
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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;



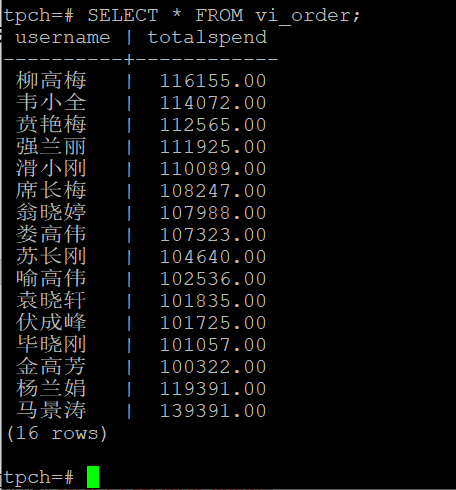
5. 对表进行操作后，刷新增量物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM vi\_order;



6. 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？

答：全量物化视图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。创建全量物化视图语法和CREATE TABLE AS语法一致，不支持对全量物化视图指定NodeGroup创建；

增量物化视图顾名思义就是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进行刷新。与全量创建物化视图不同在于目前增量物化视图所支持场景较小，目前物化视图创建语句仅支持基表扫描语句或者UNION ALL语句。

思考题2：物化视图适用那些使用场景？

适用场景：

分析需求覆盖明细数据查询以及固定维度查询两方面。

查询仅涉及表中的很小一部分列或行。

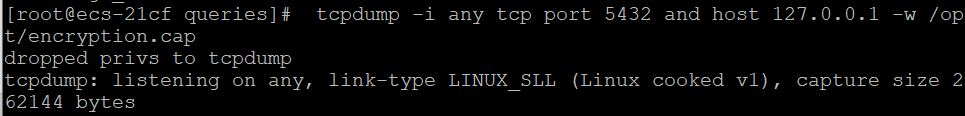
查询包含一些耗时处理操作，比如：时间很久的聚合操作等。

查询需要匹配不同前缀索引。

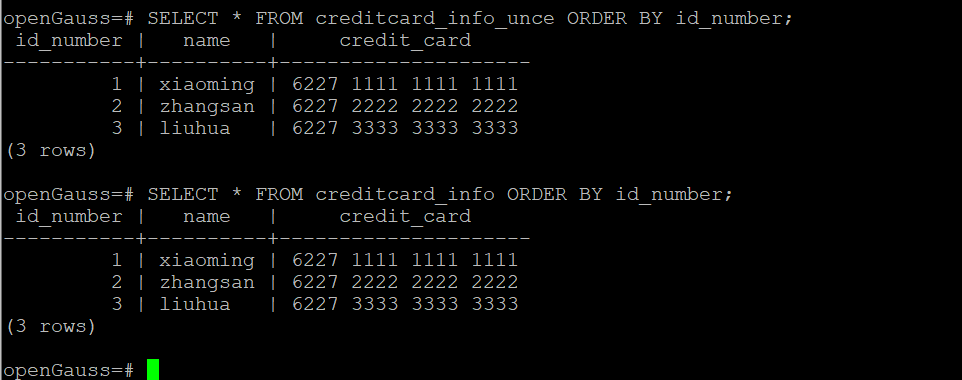
# 关卡四：openGauss密态数据库特性应用

任务一：物化视图的使用

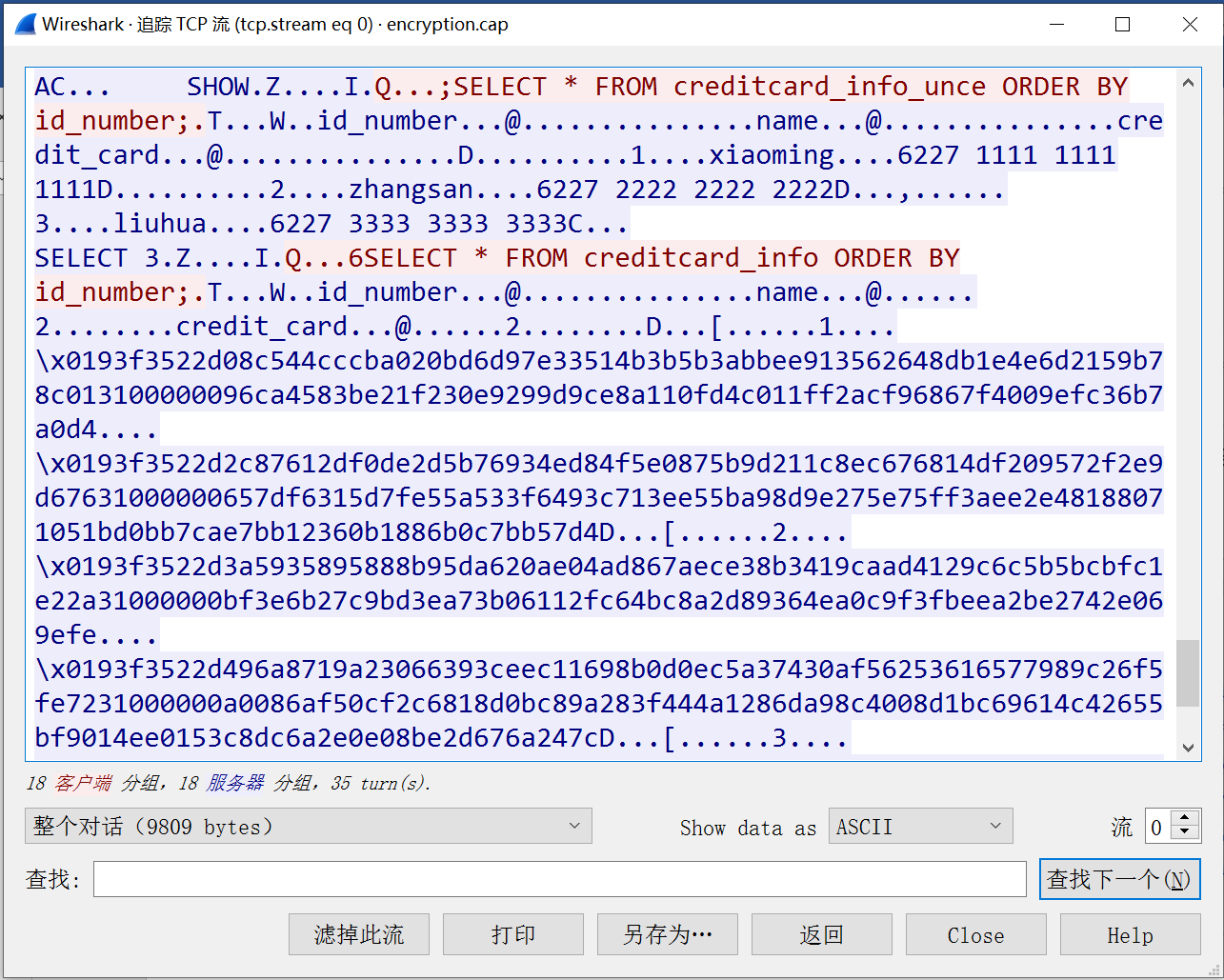
1. 通过tcpdump抓取数据流，此putty窗口暂时保持不动，将执行结果截图：



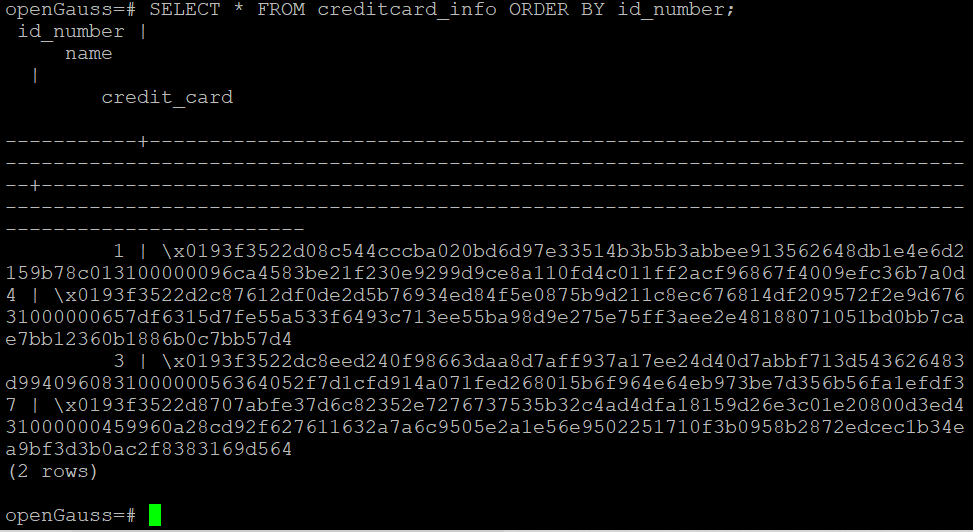
1. 将加密表和非加密表查询结果截图：



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：openGauss支持密态数据库，意在解决数据全生命周期的隐私保护问题，使得系统无论在何种业务场景和环境下，数据在传输、运算以及存储的各个环节始终都处于密文状态。

openGauss提供加密函数gs\_encrypt\_aes128()、gs\_encrypt()和解密函数gs\_decrypt\_aes128()、gs\_decrypt()接口。通过加密函数，可以对需要输入到表内的某列数据进行加密后再存储到表格内。

数据拥有者在客户端完成数据加密，参数加密完成后整个查询任务被变更成一个加密的查询任务并通过安全传输通道发到数据库服务端，由数据库服务端完成基于密文的查询检索。检索得到的结果仍然为密文，并最终返回客户端进行解密。