openGauss 安全体系创新

实践课



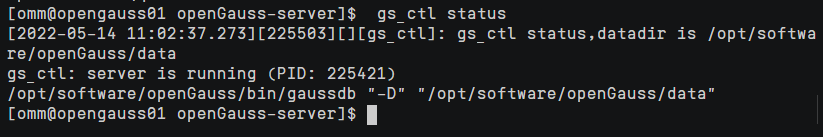
华为技术有限公司

# 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：

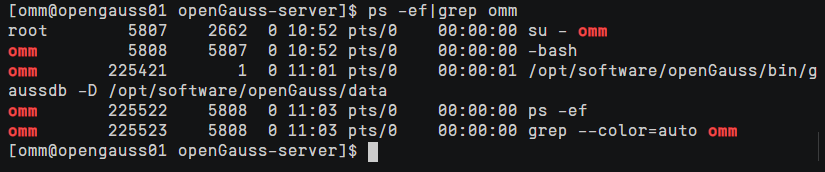
任务一：数据库状态验证

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？

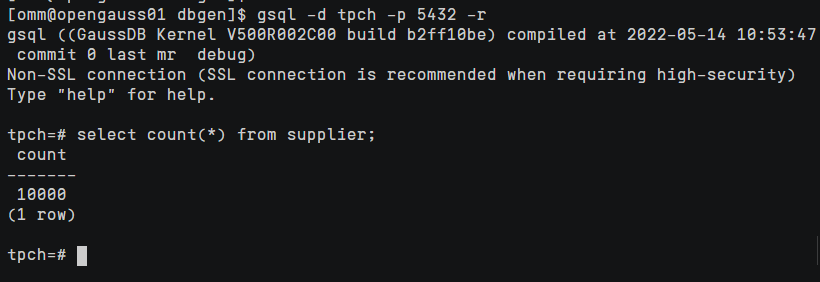
1. 可以满足不同的运行平台。因为Linux发行的版本很多，每个版本采用的软件或者内核版本不一，因此大部分软件直接提供源码。
2. 方便定制，满足用户的不同需求。大多数的二进制代码都是一键安装，自由度不高，需要定制安装内容时，需要用到源码编译安装。
3. 方便运维和开发人员的维护。

# 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：

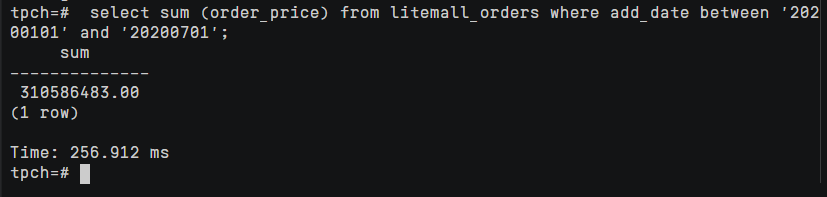
select count(\*) from supplier;;



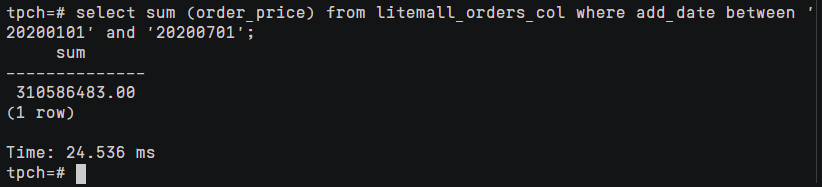
任务二：行存表与列存表执行效率对比

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';

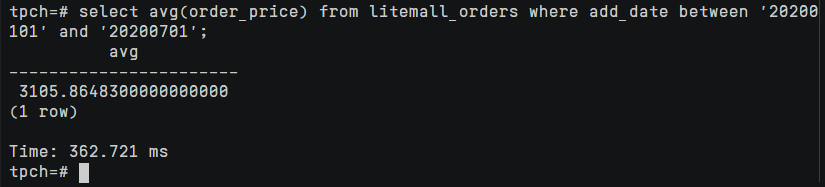


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

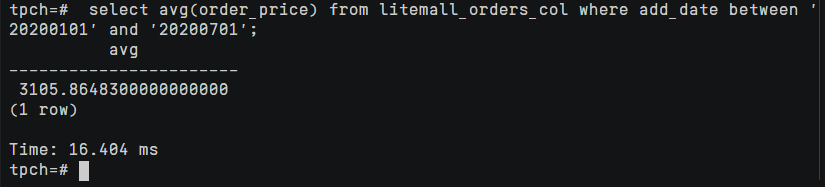


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

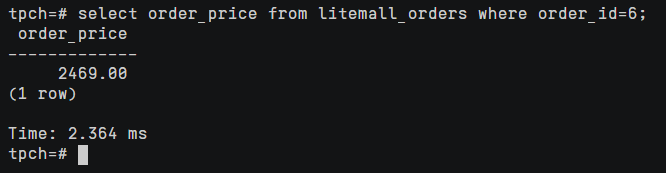


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

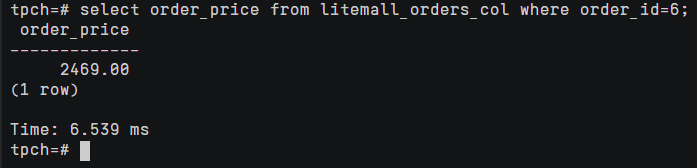


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

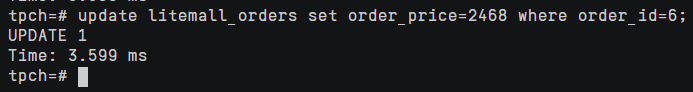


select order\_price from litemall\_orders\_col where order\_id=6;

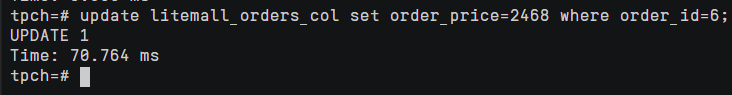


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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？

行存储是一次写入完成，而列存储由于需要把一行记录拆分成单列保存，写入次数明显多于行存储，磁头调度次数多，花费时间变长，再加上磁头需要在盘片上移动和定位花费的时间，实际消耗时间更大，因此行存表和列存表执行时间不同。

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？

行存表：

1.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.频繁插入或更新的操作，操作和索引和行的大小相关。

列存表：

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

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；

# 关卡三：openGauss物化视图应用

任务一：物化视图的使用

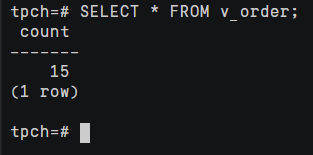
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



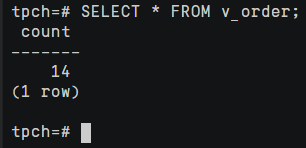
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



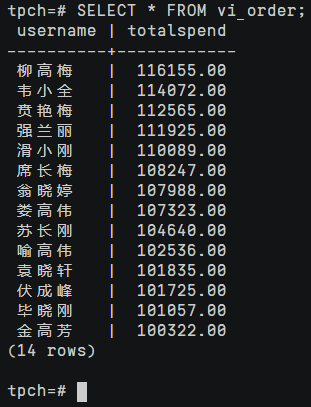
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



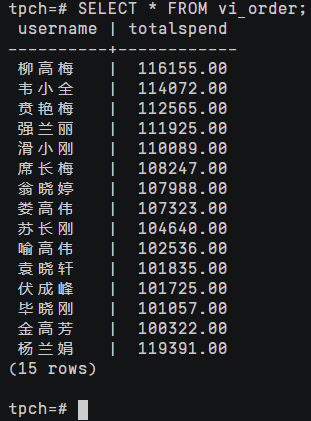
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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？

全量物化视图仅支持对创建好的物化视图做全量更新，不支持做增量更新。同全量创建物化视图相比，增量物化视图支持场景较小，目前仅支持基表扫描语句和UNION ALL语句。

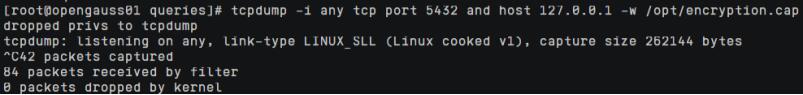
思考题2：物化视图适用那些使用场景？

1. 不同表字段聚合和信息重组：某些业务需要用到多个表查询，为加快进程启动速度，可以将这多个表的数据组合为一个物化视图；
2. 用于高级复制：不需要对整个数据库进行同步，只需要对部分表的部分字段进行同步；
3. 计算列的需要：数据库设计要求减少冗余字段，因此很多数据表没有计算列字段，因此我们可以创建一个含有计算列字段的视图来解决此问题；
4. 安全性需要：需要用到早期遗留系统数据，直接操作数据库涉及数据安全性，合理使用视图可以保证数据安全性；
5. 兼容老的数据表。

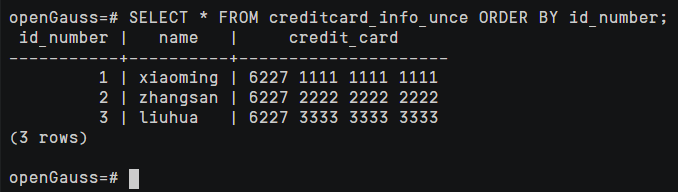
# 关卡四：openGauss密态数据库特性应用

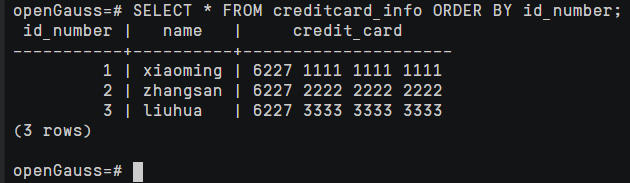
任务一：物化视图的使用

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：

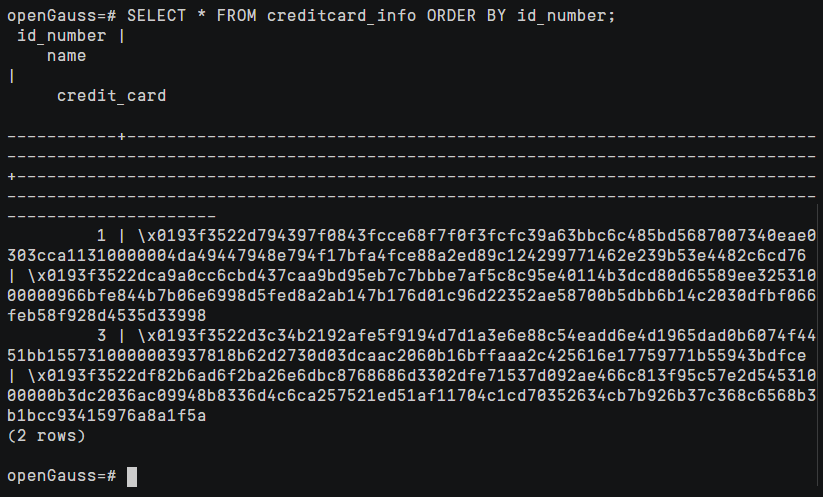




1. 用wireshark解析加密表和非加密表的差异时，非加密表name列和credit\_card列是明文，加密表name列和credit\_card列均是密文，将执行结果截图：



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？

数据库中的数据存储在磁盘上的所有数据均是明文。

客户端和服务端都采用非对称加密方法进行加解密。服务端生成私钥和公钥对，客户端预先存放服务端公钥，客户端在登陆时，使用服务端的公钥加密登陆信息。客户端生成私钥和公钥，然后将公钥发送给服务器。服务器私用客户端的公钥给客户端发送消息。