openGauss 安全体系创新

实践课



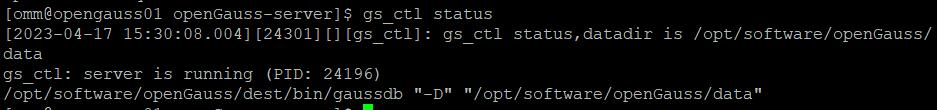
华为技术有限公司

# 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：

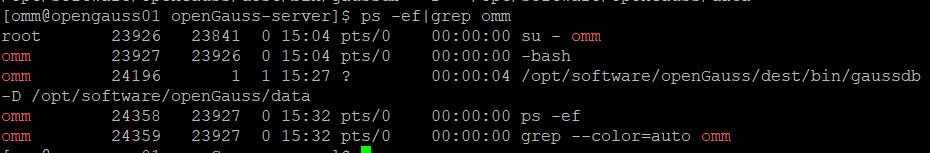
任务一：数据库状态验证

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？

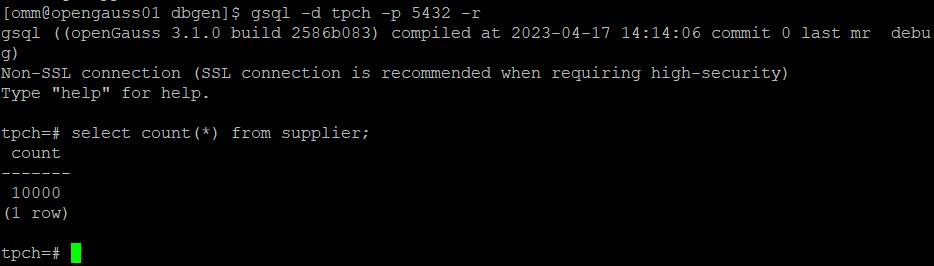
通过源码编译安装数据库可以让用户更好地控制数据库的安装过程，以及更好地适应自己的需求。此外，还可以让用户更好地了解数据库的内部结构和工作原理。相比于其他安装方式，通过源码编译安装数据库需要用户具备一定的Linux基础知识和编译经验。

# 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：

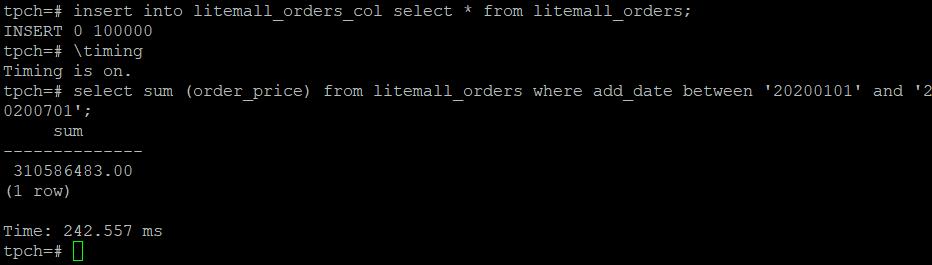
select count(\*) from supplier;;



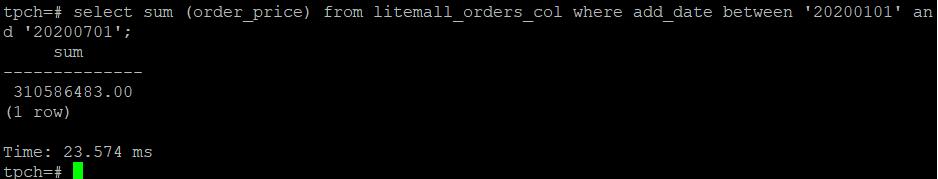
任务二：行存表与列存表执行效率对比

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';

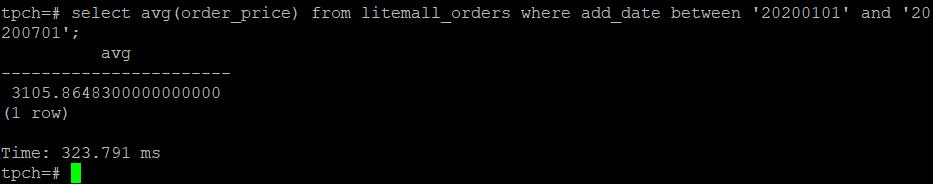


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

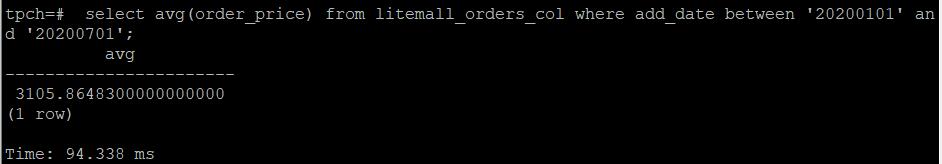


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

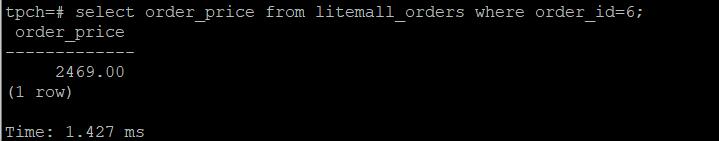


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

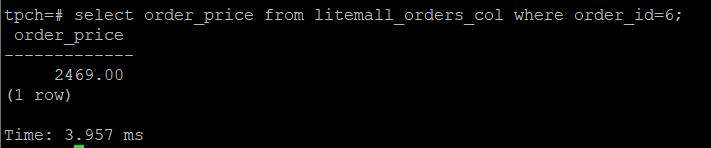


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

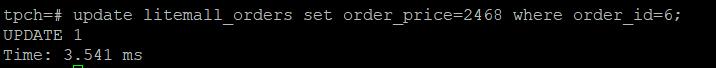


select order\_price from litemall\_orders\_col where order\_id=6;

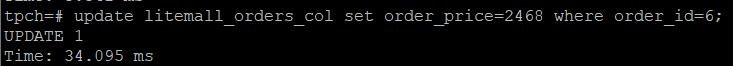


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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？

在执行相同的SQL语句时，它们的执行时间不同，主要是因为它们的数据存储方式不同。

行存表是将一行数据作为一个记录来存储的，每个字段都是一个列。当执行查询语句时，需要读取整行数据，包括不需要的字段，这就会导致查询速度变慢。

列存表是将每个字段作为一个列来存储的，每个记录都是一个行。当执行查询语句时，只需要读取需要的字段，这样就可以大大提高查询速度

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？

行存表适合处理大量数据，例如批量插入、更新、删除等操作。因为行存表将一行数据作为一个记录来存储，每个字段都是一个列。当执行这些操作时，需要读取整行数据，包括不需要的字段，这就会导致查询速度变慢。

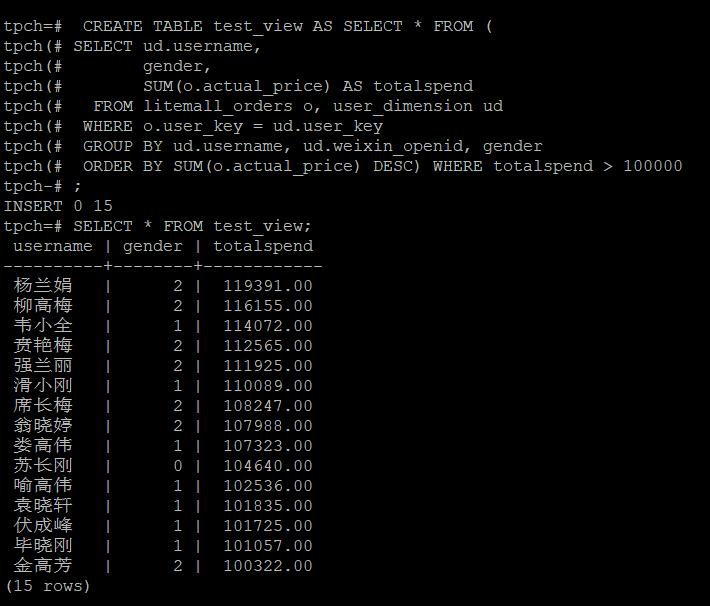
列存表适合处理少量数据，例如查询操作。因为列存表将每个字段作为一个列来存储，每个记录都是一个行。当执行查询语句时，只需要读取需要的字段，这样就可以大大提高查询速度。

# 关卡三：openGauss物化视图应用

任务一：物化视图的使用

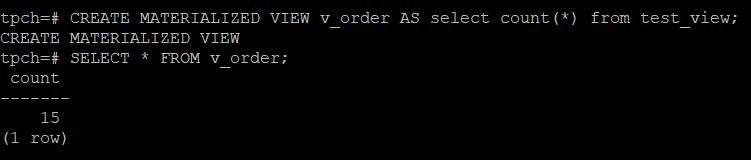
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



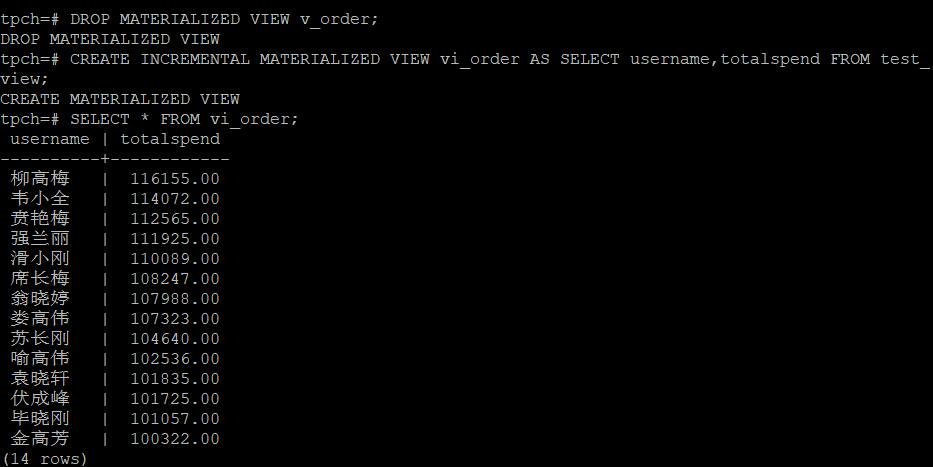
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



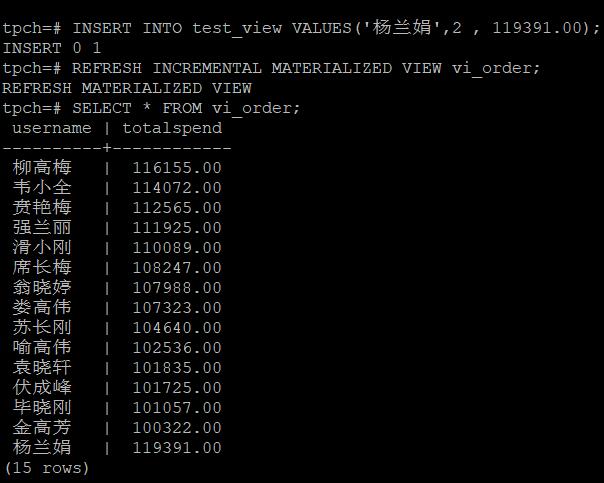
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



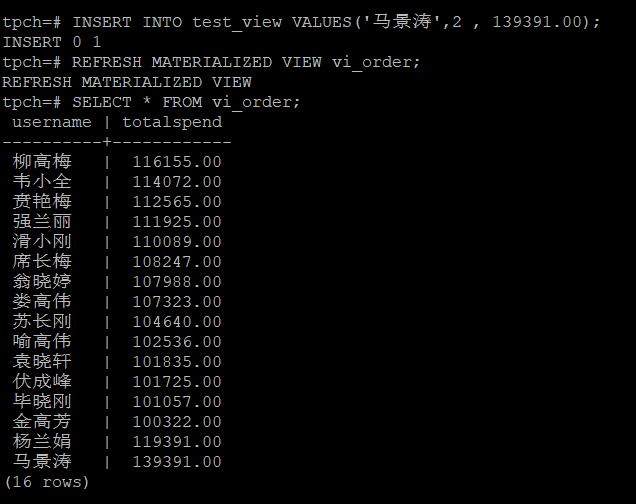
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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区别在于刷新方式不同。

全量物化视图是一种特殊的物理表，它的数据会持久化。当查询全量物化视图时，会直接从物化视图里读取数据，而不是访问基表。全量物化视图需要定期刷新以保证数据的准确性。

增量物化视图是一种可以对物化视图增量刷新的物化视图，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？

[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。](https://blog.csdn.net/weixin_34310785/article/details/91565355" \t "https://www.bing.com/_blank)

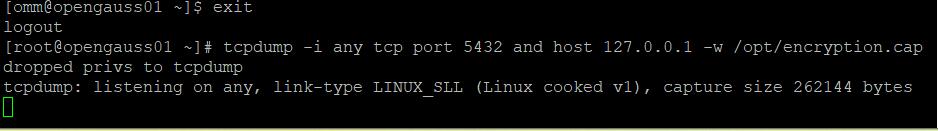
[在查询优化方面，物化视图可以提高查询性能，减少查询时间，降低系统负载。当查询全量物化视图时，会直接从物化视图里读取数据，而不是访问基表。全量物化视图需要定期刷新以保证数据的准确性。增量物化视图是一种可以对物化视图增量刷新的物化视图，需要用户手动执行语句完成对物化视图在一段时间内的增量数据进行刷新。与全量创建物化视图不同在于目前增量物化视图所支持场景较小，目前物化视图创建语句仅支持基表扫描语句或者UNION ALL语句。](https://zhuanlan.zhihu.com/p/486051815" \t "https://www.bing.com/_blank)

[在高级复制方面，物化视图可以作为数据同步的一种方式。当基表数据发生变更时，可以通过触发器等方式将变更同步到物化视图中，然后再将变更同步到其他数据库中。](https://blog.csdn.net/weixin_34310785/article/details/91565355" \t "https://www.bing.com/_blank)

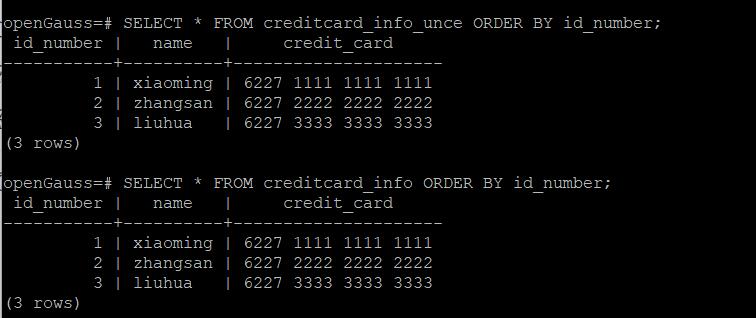
# 关卡四：openGauss密态数据库特性应用

任务一：物化视图的使用

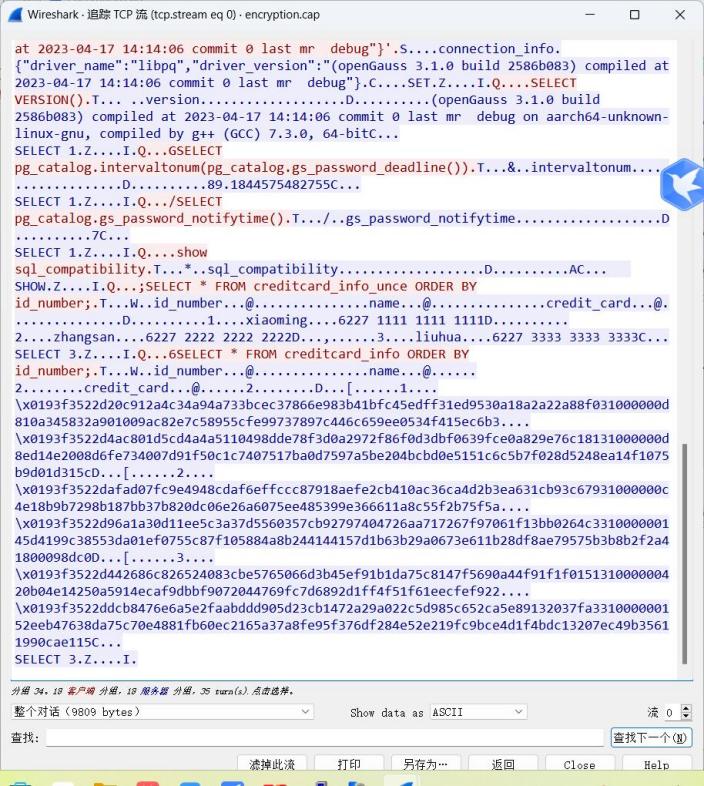
1. 通过tcpdump抓取数据流，此putty窗口暂时保持不动，将执行结果截图：



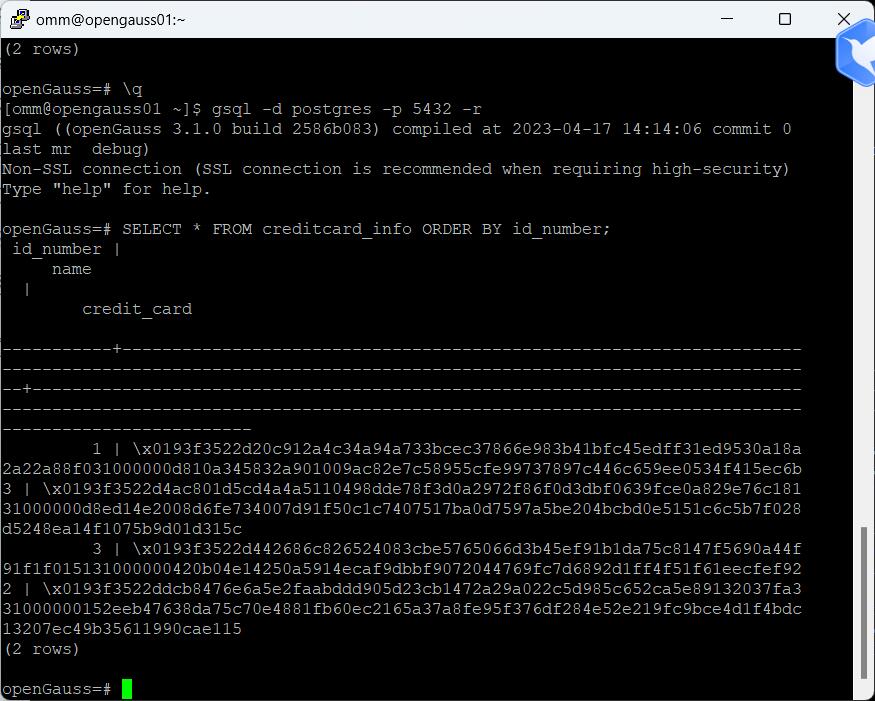
1. 将加密表和非加密表查询结果截图：



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？

[数据实际存储在物理磁盘上时，通常是明文的。加解密的动作可以在客户端完成，也可以在服务端完成。](https://zhuanlan.zhihu.com/p/363126880" \t "https://www.bing.com/_blank)

在客户端完成加解密的动作，可以保证数据在传输过程中不被窃取，但是一旦攻击者获得了客户端的密钥，就可以轻松地解密数据。因此，这种方式适用于对安全性要求不高的场景。

在服务端完成加解密的动作，可以保证数据在传输过程中不被窃取，并且攻击者无法获得密钥。因此，这种方式适用于对安全性要求较高的场景。