openGauss AI特性创新实践课



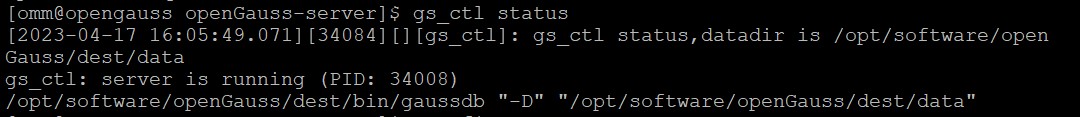
华为技术有限公司

# 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：

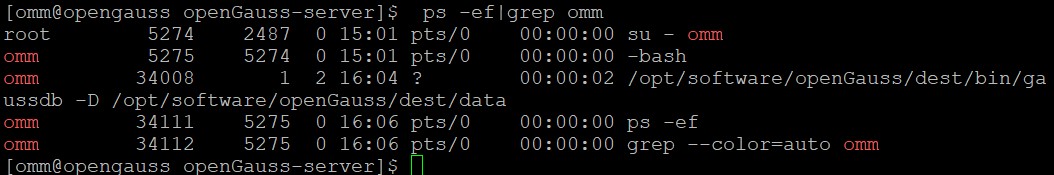
任务一：数据库状态验证

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）



实验思考题：为什么需要通过源码编译，安装数据库？

通过源码编译和安装数据库软件可以提供以下好处：

定制化：通过源码编译安装可以根据特定的需求定制化数据库软件的安装。例如，可以选择编译安装特定的模块或插件，或者调整编译参数以获得更好的性能。

最新功能：源码编译可以确保安装的数据库软件是最新的版本，包含最新的功能和修复了最新的漏洞。

更好的性能：编译过程中可以针对特定的硬件和软件环境进行调整，以获得更好的性能。此外，可以根据需要选择不同的编译选项以优化性能。

更好的安全性：通过源码编译和安装可以确保数据库软件是从可信源代码构建而成的，没有被植入任何后门或恶意代码。这可以提高数据库的安全性和可信度。

更好的可维护性：通过源码编译安装可以更好地理解数据库软件的内部结构和工作原理，以便更好地进行调试和维护。

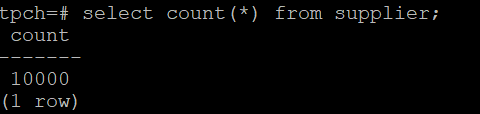
总之，通过源码编译安装数据库软件可以提供更多的灵活性、可定制性、性能和安全性，以及更好的可维护性。

# 关卡二、openGauss数据导入及基本操作

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：

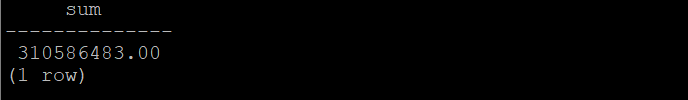
select count(\*) from supplier;;



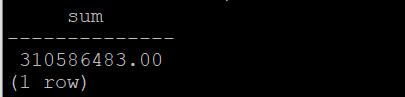
任务二：行存表与列存表执行效率对比

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';

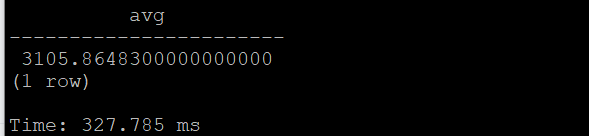


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

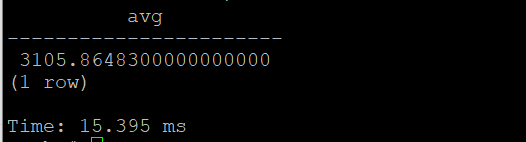


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

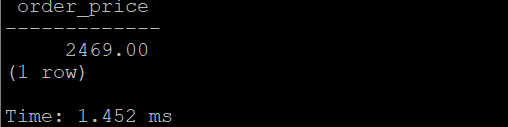


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

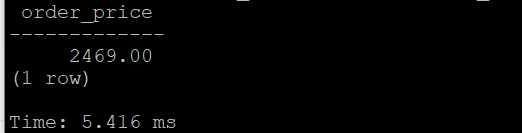


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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;



# 关卡三：openGauss物化视图应用

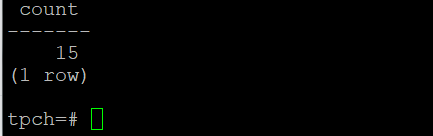
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



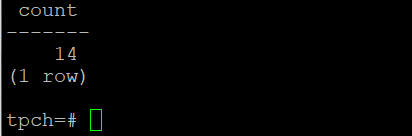
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



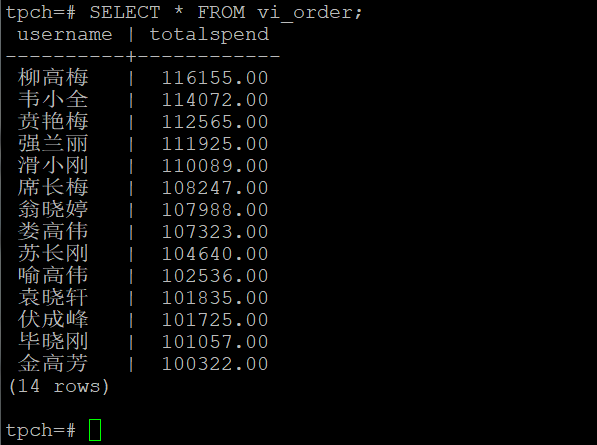
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



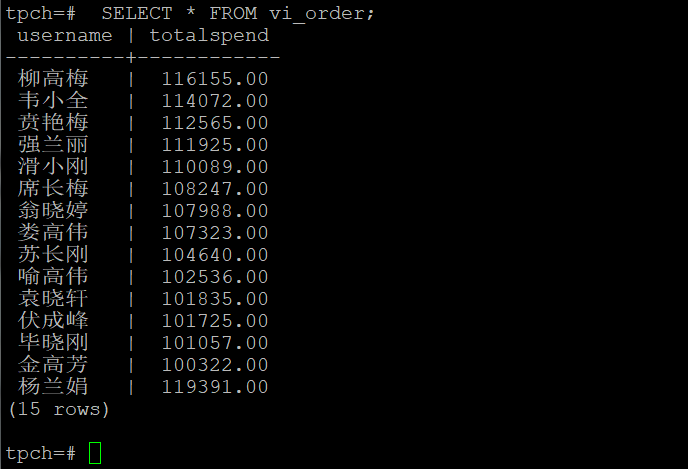
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

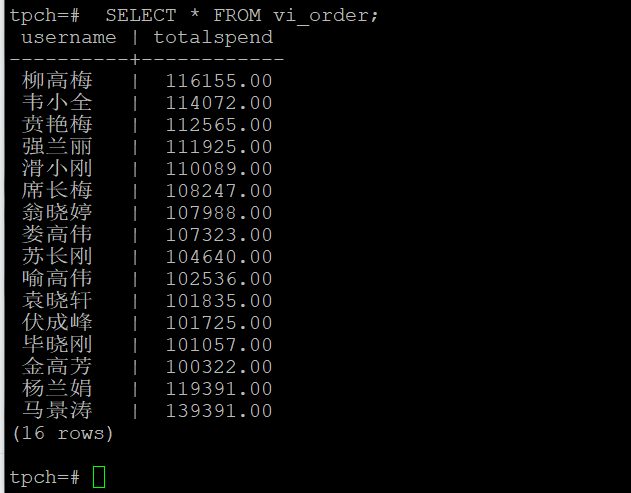
SELECT \* FROM vi\_order;



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;





实践思考题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？

行存表和列存表的物理存储方式不同，因此在执行相同的 SQL 语句时会出现执行时间不同的情况。

行存表将每一行数据存储在一起，这意味着当执行需要查询某些行的 SQL 语句时，需要将整个行读入内存中，这可能需要进行大量的磁盘 I/O 操作。对于需要读取整行数据的查询，行存表的效率可能更高。

列存表将每一列数据存储在一起，这意味着当执行需要查询某些列的 SQL 语句时，只需要将相应的列读入内存中，这可以大大减少需要进行的磁盘 I/O 操作。对于需要只查询某些列的查询，列存表的效率可能更高。

具体而言，行存表在执行需要查询整行数据的 SELECT 语句、INSERT、UPDATE 和 DELETE 语句时可能更高效；而列存表在执行需要查询某些列的 SELECT 语句时可能更高效。此外，对于需要进行聚合操作（例如 SUM、AVG 等）的查询，列存表的效率也可能更高，因为只需要读取相应的列数据并对其进行聚合，而不需要读取整个行的数据。

实践思考题2：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都是一种预先计算并存储的结果集，它们的主要区别在于更新和维护的方式。

全量物化视图：

全量物化视图是将视图的结果集缓存在数据库中，与存储表类似。当视图的基表数据发生更改时，需要重新计算整个结果集并更新物化视图中的数据。这可能会在更新较频繁的情况下导致较长的计算和等待时间，因此全量物化视图适用于基表数据更新较少的情况。

增量物化视图：

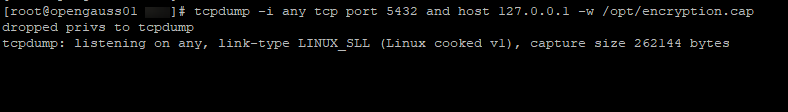
增量物化视图是在全量物化视图的基础上提供增量更新的机制。它可以在基表数据发生更改时仅计算和更新更改的部分。这可以减少更新物化视图的计算和等待时间，并且可以支持更频繁的基表更新。增量物化视图一般会使用类似于数据库日志的机制来记录基表的更改，并根据这些更改进行增量更新。

总之，全量物化视图和增量物化视图都是一种预先计算并存储的结果集，可以提高查询效率。全量物化视图在基表数据更新较少的情况下使用较为合适，而增量物化视图可以在基表数据更新较频繁的情况下提供更好的性能和响应速度。

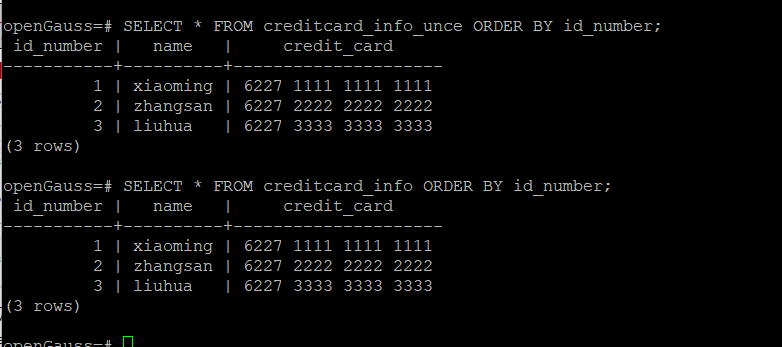
# 关卡四：openGauss密态数据库特性应用

任务一：物化视图的使用

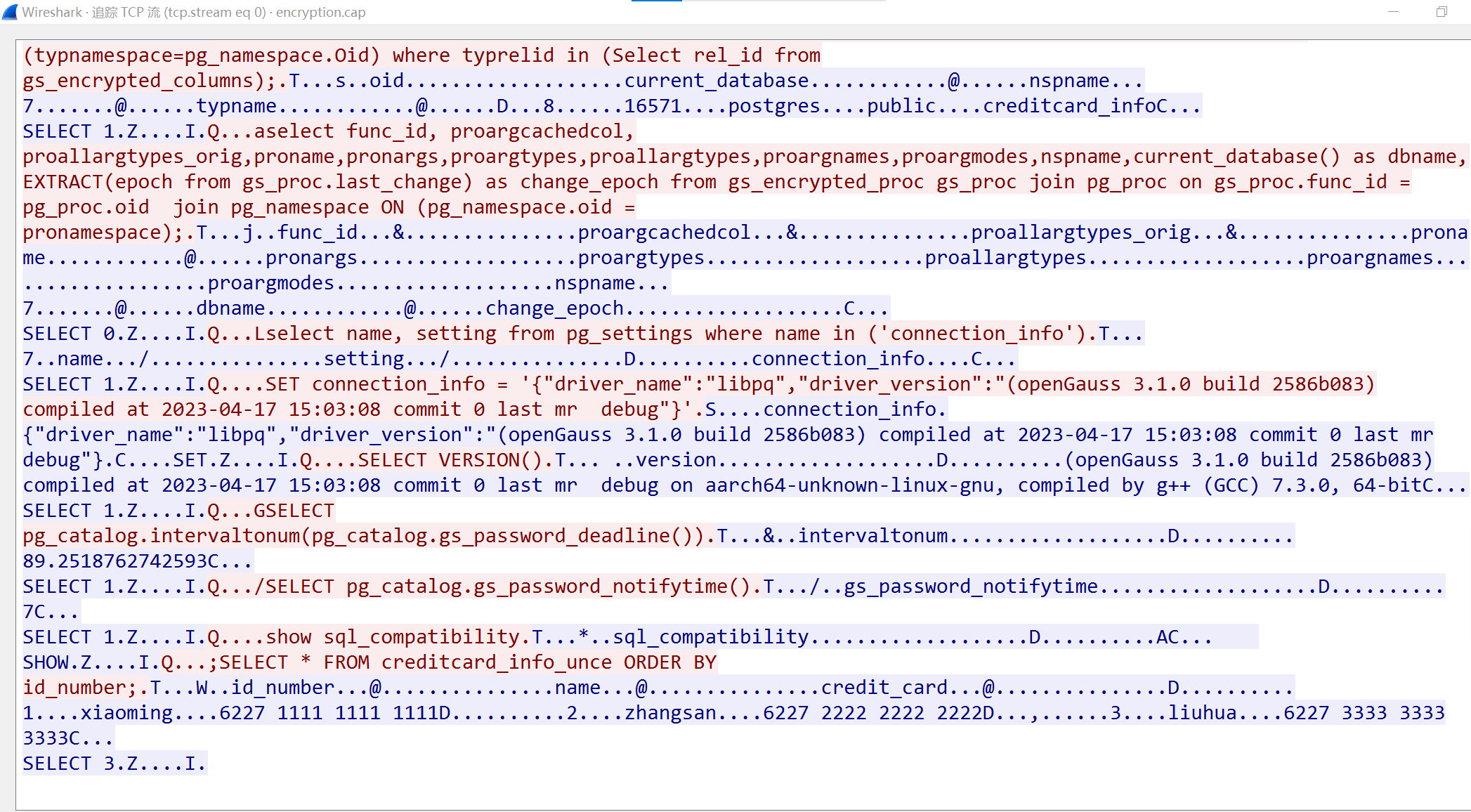
1. 通过tcpdump抓取数据流，此putty窗口暂时保持不动，将执行结果截图：



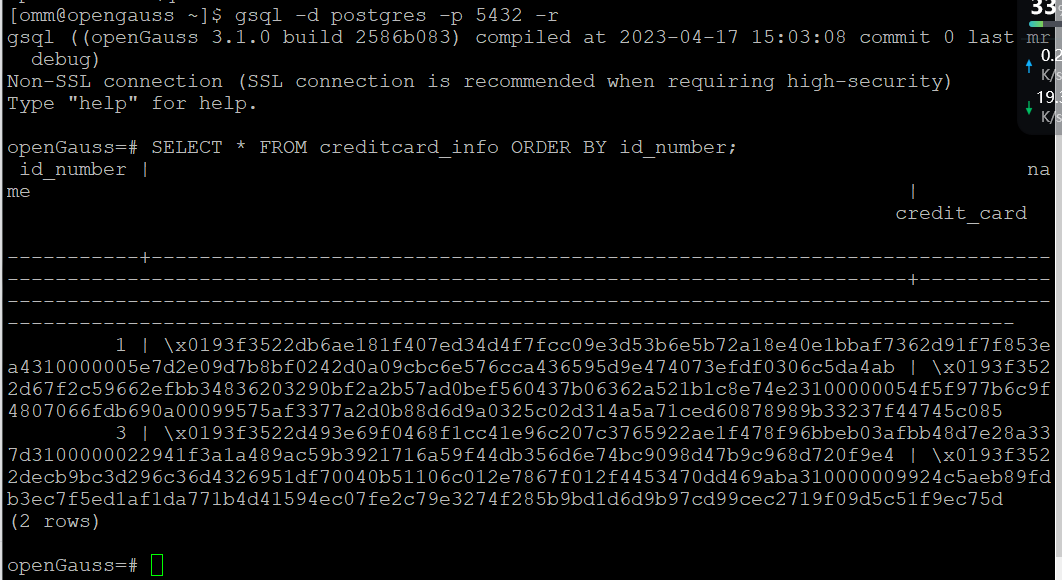
1. 将加密表和非加密表查询结果截图：



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的时候通常是密文形式存储的，以保护数据的机密性和安全性。

至于数据的加解密的动作，则可以由客户端或服务端来完成，这取决于系统的具体设计和实现方式。

在某些情况下，为了保护数据的安全，数据可能会在客户端进行加密处理，并在加密后的状态下发送到服务端进行存储。在这种情况下，数据的加解密动作都是由客户端完成的。

在其他情况下，服务端可能会负责对数据进行加密和解密。例如，如果数据需要在多个客户端之间共享，服务端可能会使用一种安全的加密方案，以确保数据的机密性和完整性。

总之，数据加解密的动作可以在客户端或服务端完成，具体取决于系统的设计和实现方式以及安全性需求。