**openGauss与PostgreSQL的对比分析**

#### 一． ****实验目标****

本次实验旨在从性能，可靠性和安全性等方面，通过实验验证比较与评估openGauss和PostgreSQL数据库，推测两个数据库在实验结果中产生差别的可能原因，进而分析openGauss的优缺点。

1. ****评估数据库的指标****

我们可以根据以下几个标准来评估数据库的质量：

1. **性能**：可以通过数据库处理sql命令的时间分析吞吐量、并发处理能力等。
2. **可靠性**：验证数据库的ACID特性，检验数据库的稳定性和错误恢复能力。
3. **安全性**：数据库的数据保护能力。

#### ****实验准备****

1. **测试环境**

本次实验使用了 PostgreSQL 12.1 版本，通过拉取 Docker 镜像部署了 openGauss 3.0.0 版本，使用 DataGrip 作为客户端进行连接和操作。

1. 实验数据

本次实验中使用：

1. dim\_product表，包含32226条关于商品信息的数据
2. fact\_sales\_apr1表，包含7,563,475条关于商品售卖信息的数据

数据来自Kaggle网站下载。

1. **实验设计**
2. **性能测试**
3. 导入数据

在导入较大的表格fact\_sales\_apr1时

在导入openGauss数据库时显示

fact\_sales\_apr1.csv imported to fact\_sales\_apr1: 7,563,475 rows (2 min, 45 sec, 722 ms, 4.43 MB/s)

在导入postgreSQL数据库时显示

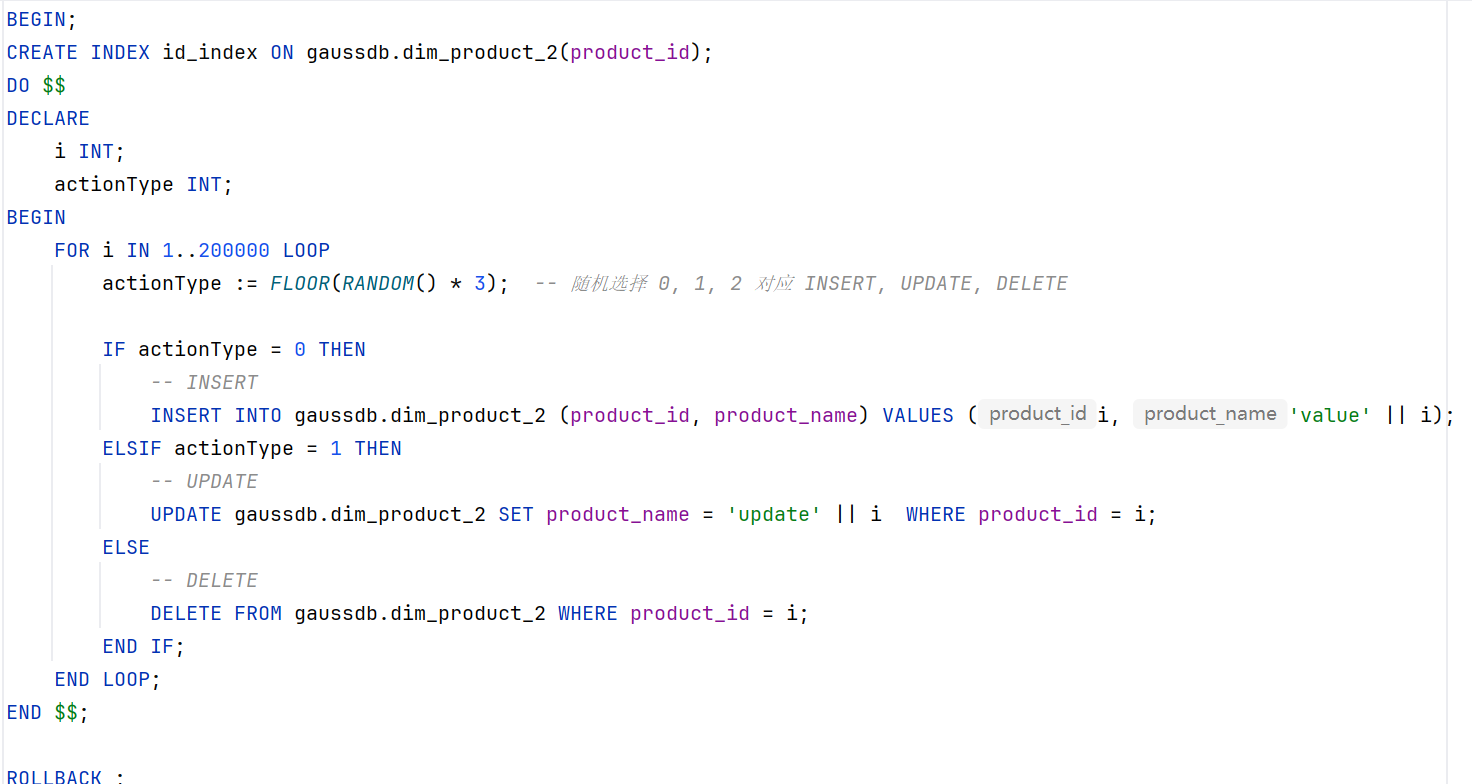
fact\_sales\_apr1.csv imported to fact\_sales\_apr1: 7,563,475 rows (3 min, 17 sec, 794 ms, 3.71 MB/s)

**可以看到，openGauss** 在导入速度上明显优于 **PostgreSQL**，这可能是由于 openGauss 对数据导入进行了更高效的并行优化，或者在 I/O 操作和磁盘配置上做了更好的优化。这个差异在大数据量的处理过程中尤为明显。

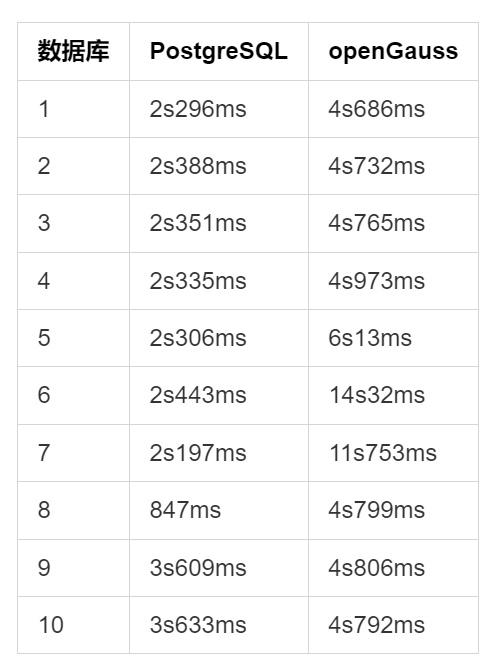
1. 简单sql命令测试(simple.sql)

这次测试首先通过重复200000条简单的sql命令，包括更新操作，插入操作，删除操作，得到两个数据库的执行时间，进行比较。

执行代码为（以openGauss为例，postgreSQL中基本一致）：

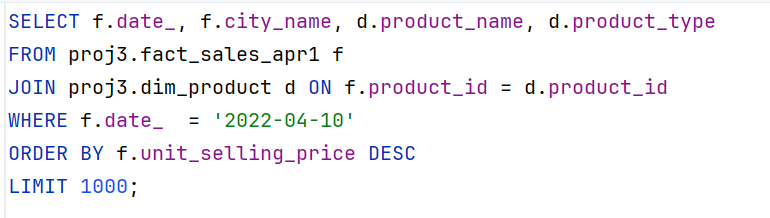


重复进行10次，得到的执行时间为：



1. 复杂sql命令测试（complex.sql）

（1）查询1：

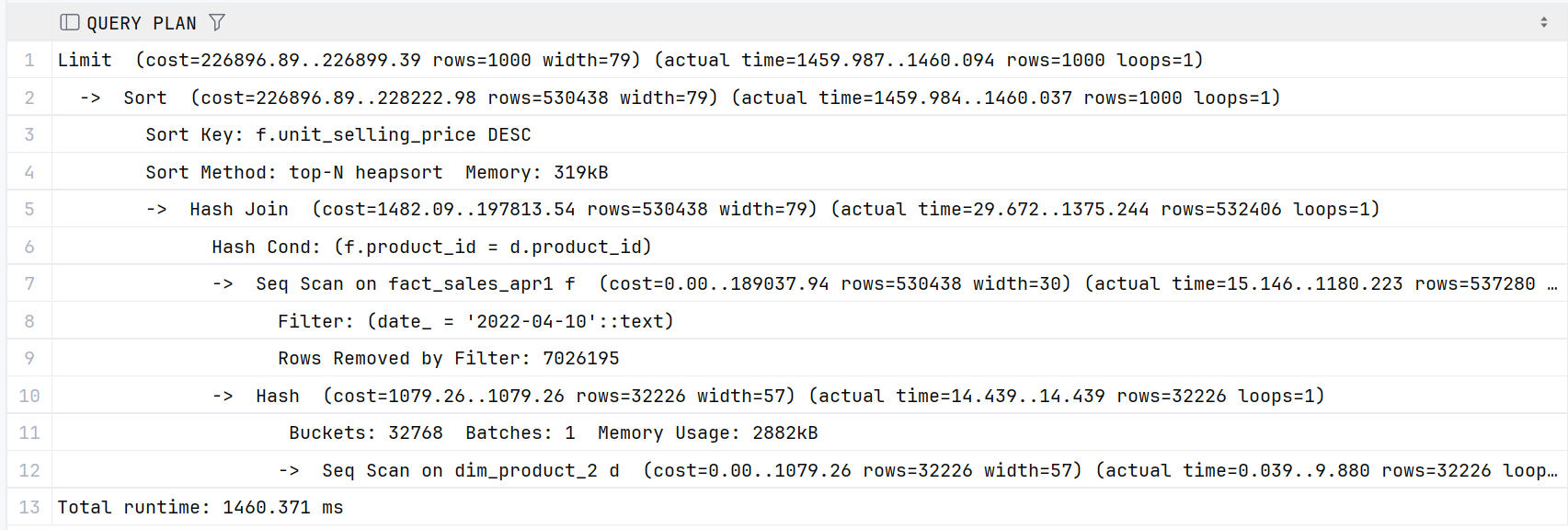


在这次查询中，openGauss执行时间为16s206ms，postgreSQL执行时间为1s158ms。

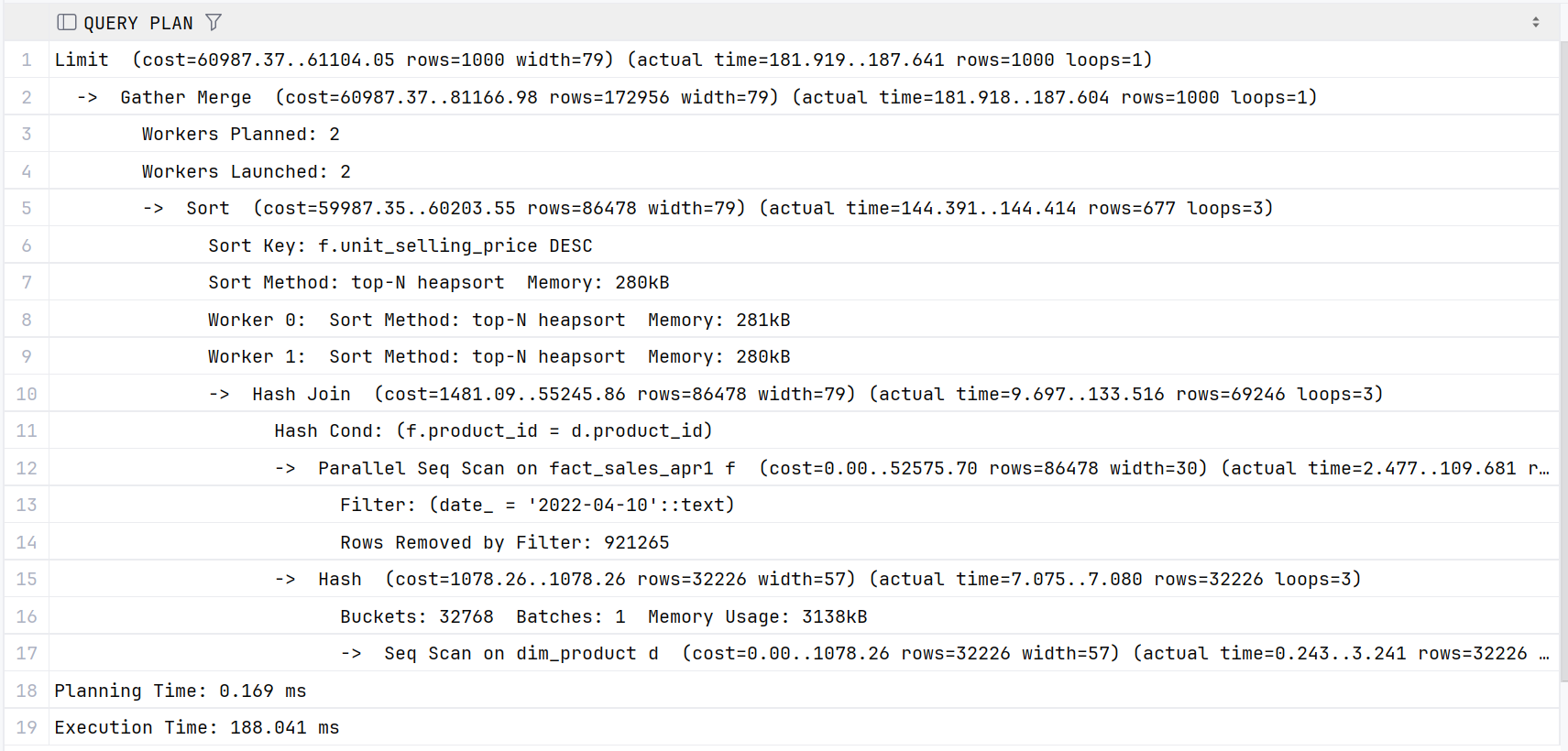
起初我怀疑这个查询的时间的差异如此大是与config里面的参数设置有关，于是我更改了postgreSQL 的config文件中决定了数据库缓存区域的大小的shared\_buffers和每个排序操作的内存限制work\_mem，将其分别改为32Mb和64Mb，与openGauss的config文件设置一致，在重新启动数据库之后再次执行相同的操作，postgreSQL的执行时间为1s229ms，仍然明显优于openGauss的第一次查询的时间。

当再次在openGauss中执行相同命令的时候，执行时间为1s460ms，而再次在postgreSQL中执行相同语句时执行时间达到了188ms。

openGauss：

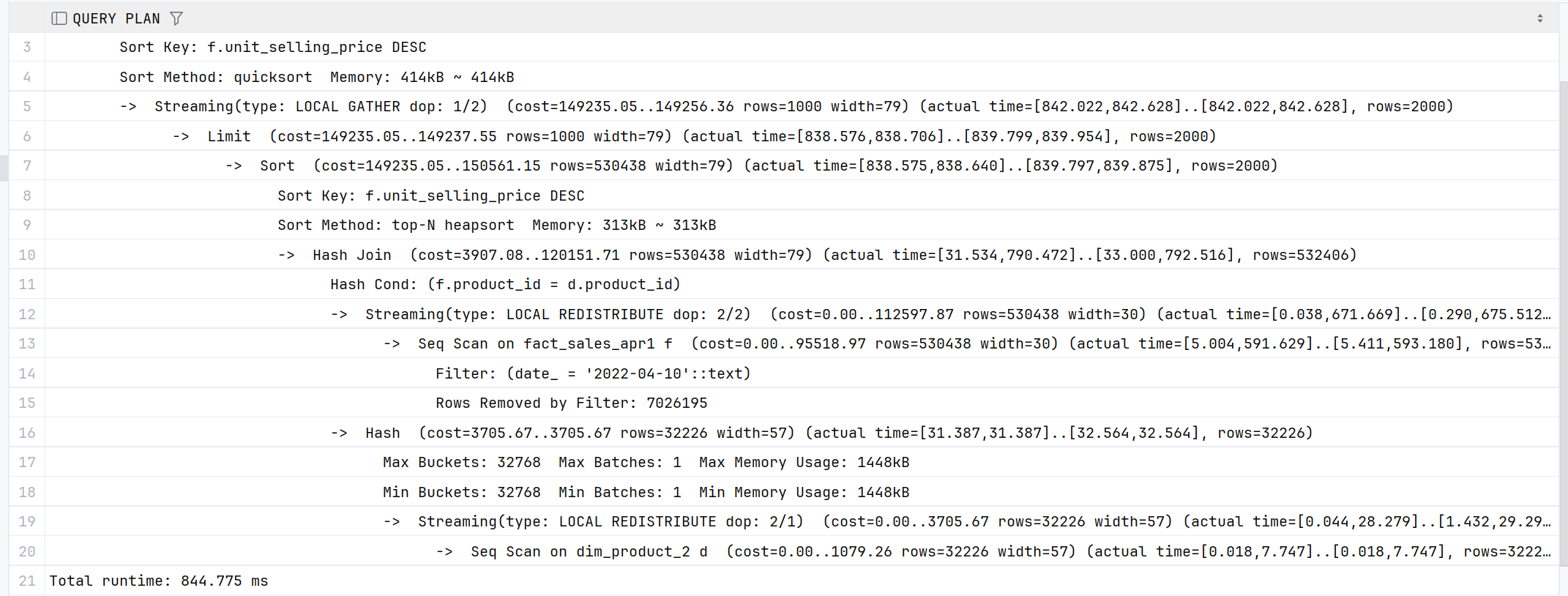


postgreSQL：

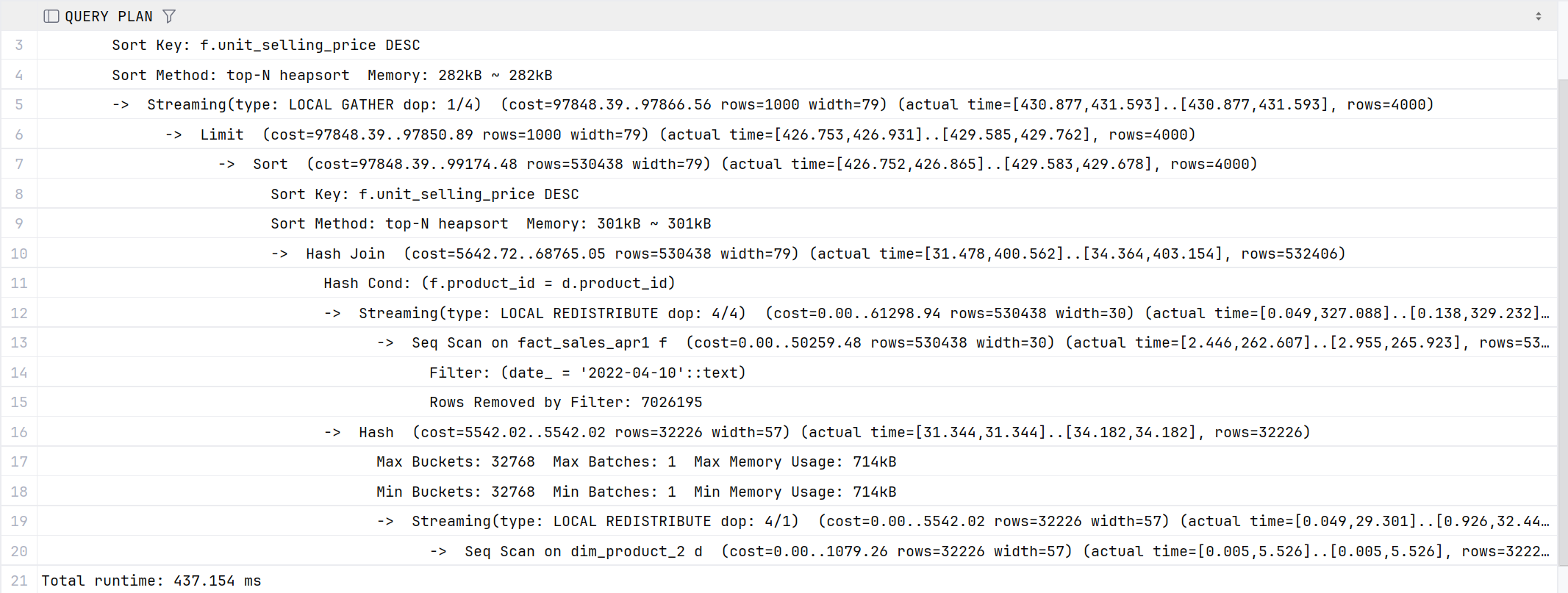


在查询计划里显示，postgreSQL使用了并行查询优化，而openGauss并没有使用并行查询优化，进而我怀疑是否是由于这个原因导致open Gauss的查询时间过长，于是我根据[官方文档](https://docs.opengauss.org/zh/docs/3.0.0/docs/BriefTutorial/%E5%B9%B6%E8%A1%8C%E6%9F%A5%E8%AF%A2.html)显示，使用SET query\_dop = 4;强制选取dop，在设置dop为2的时候，查询时间减少为844ms，在设置dop为4的时候;查询时间减少为437ms，再增加dop的数目时，查询时间基本不会再减少。

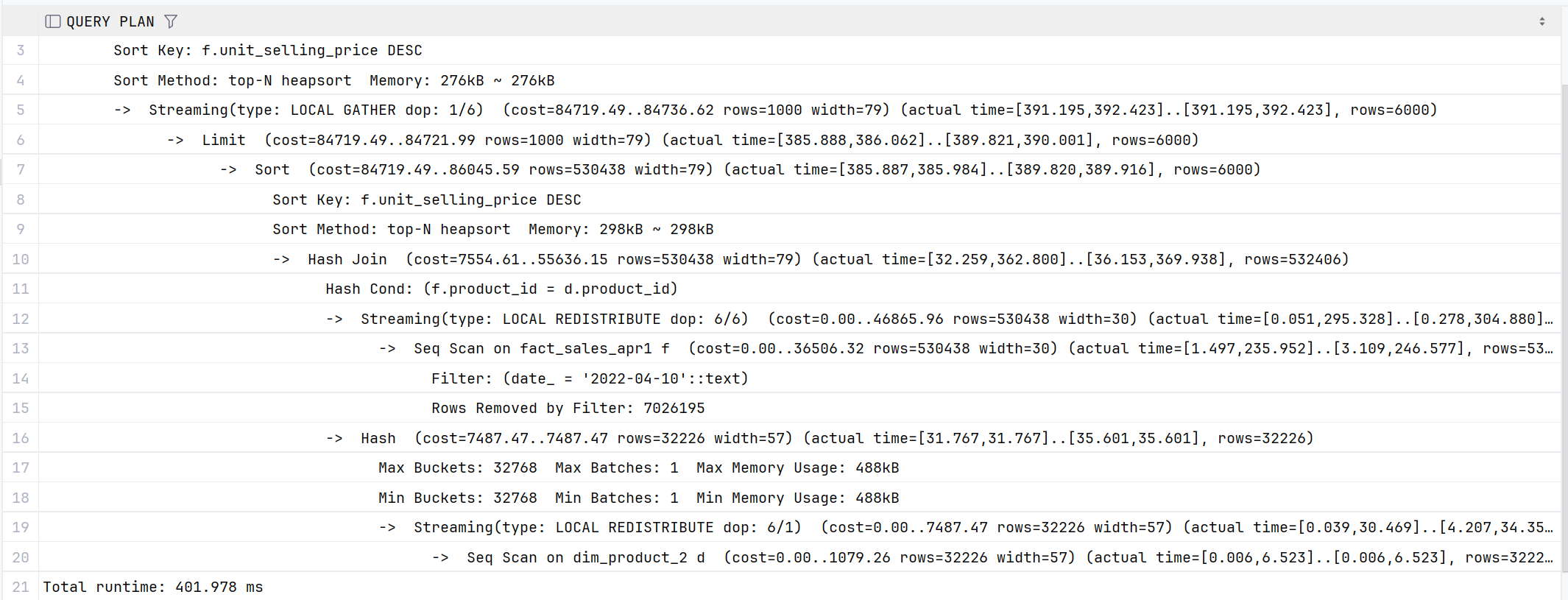
Dop = 2



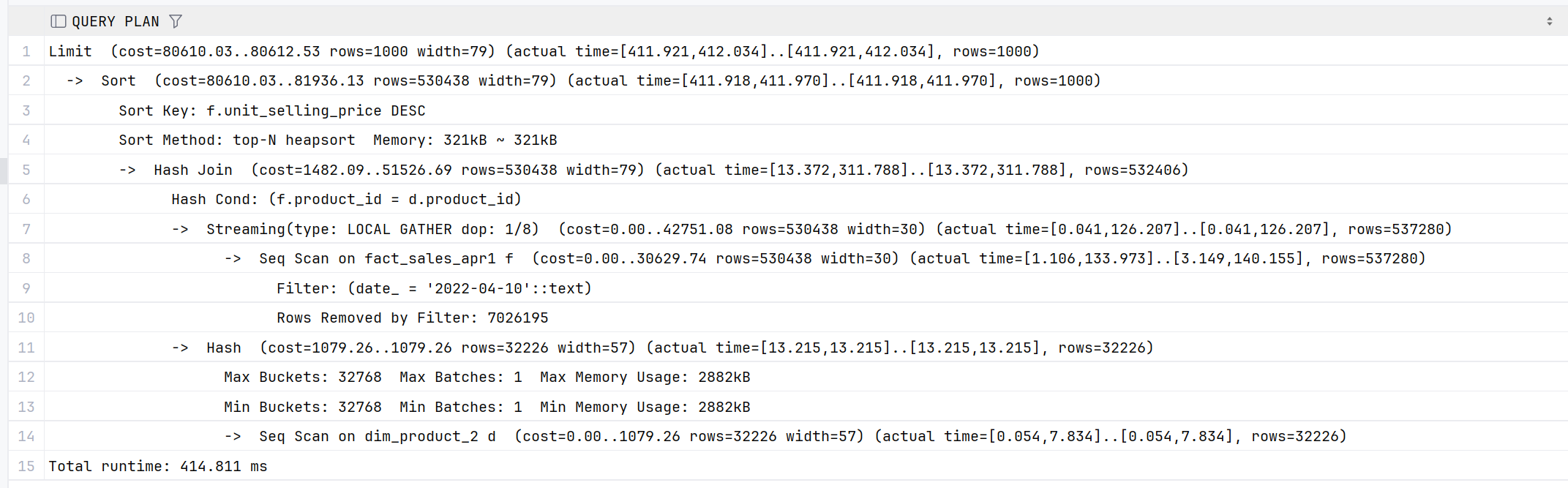
Dop = 4



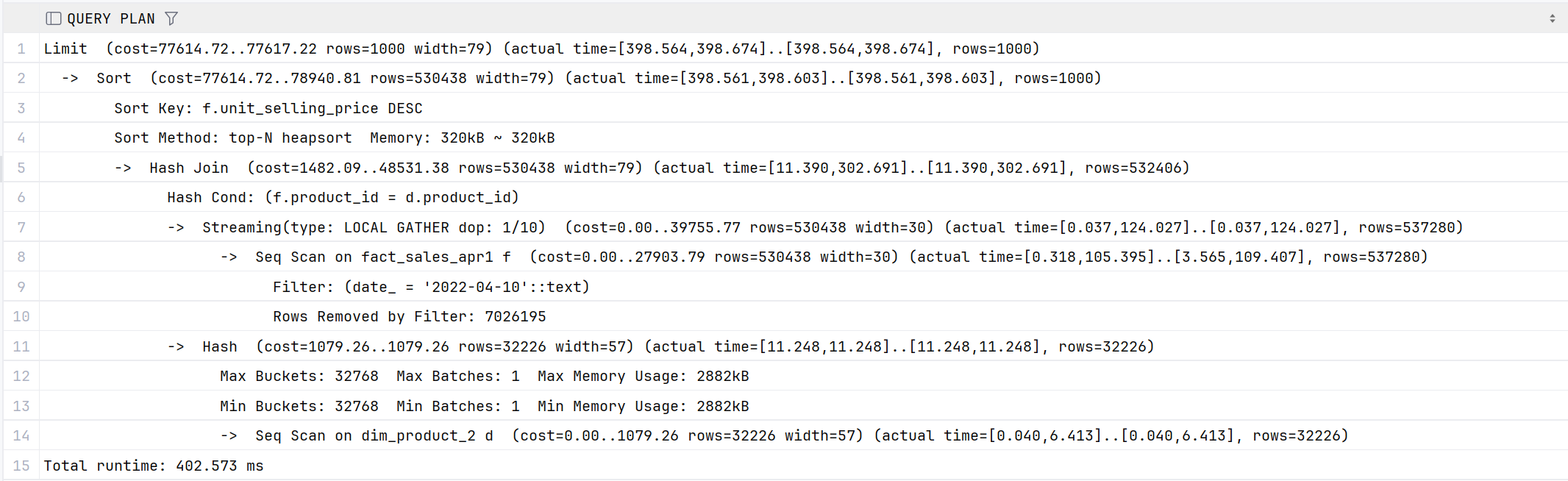
Dop = 6



Dop = 8

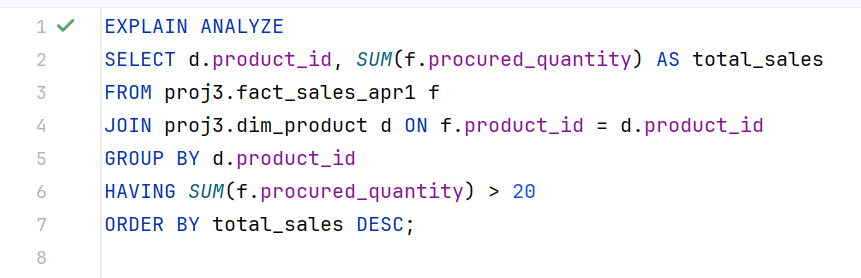


Dop = 10

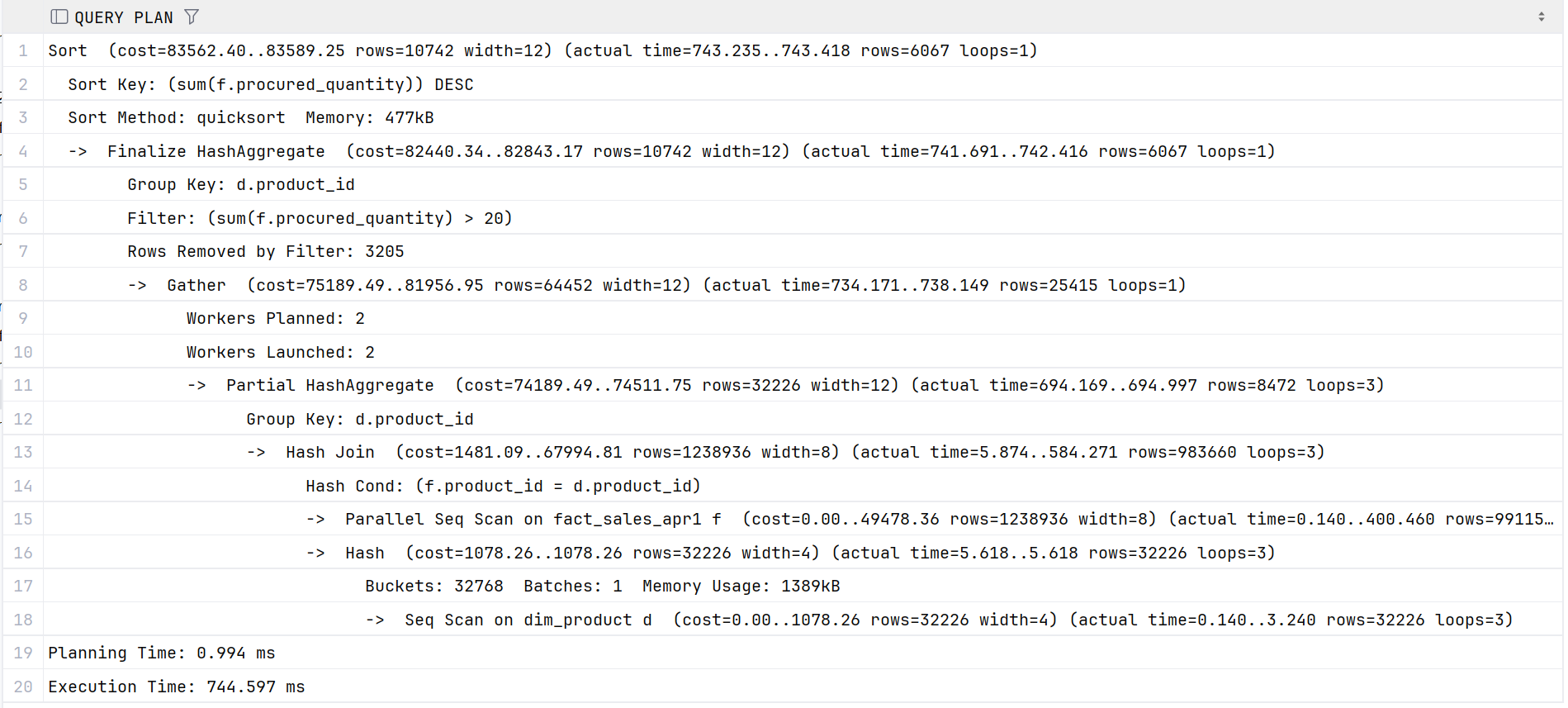


所以，在这次查询中openGauss在查询中似乎并没有体现出能够主动进行并行查询优化的特点。

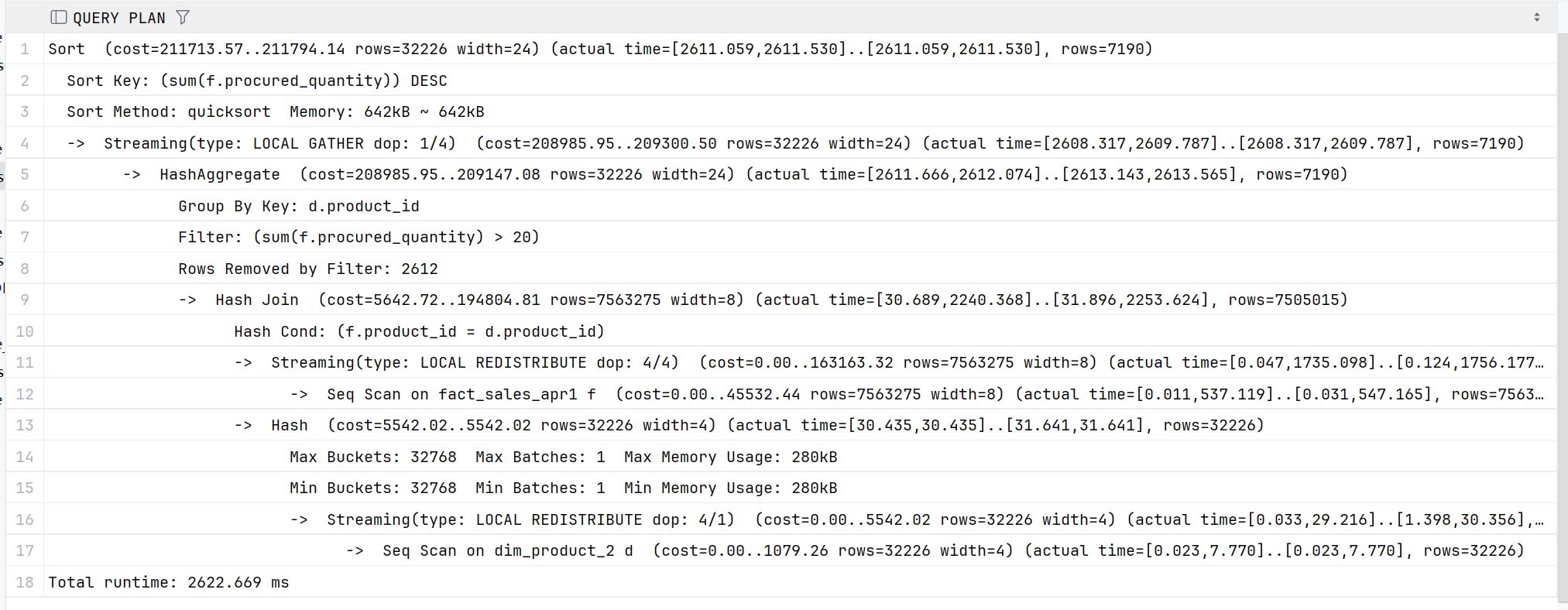
（2）查询2：



postgreSQL：



openGauss：



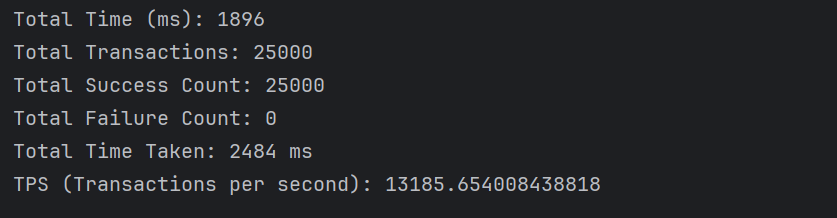
同样的，这次查询中openGauss依旧比postgreSQL要慢许多。

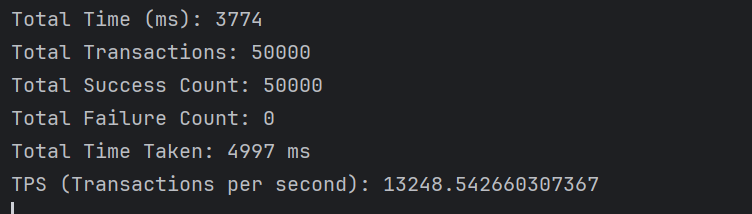
**总结：在大量简单查询和复杂查询中，openGauss的表现都不如postgreSQL，通过改变postgreSQL的配置文件中的shared\_buffers和work\_mem参数使之与openGauss一致，以及强制调整dop并不能改变postgreSQL与openGauss的表现差距。**

用java程序模拟多线程并发测试openGauss和postgreSQL的吞吐量（DatabaseThroughputTest.java）

postgreSQL：

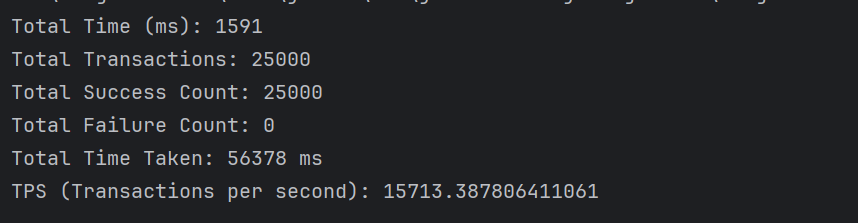
线程数为50/100，每个线程执行事务数为500

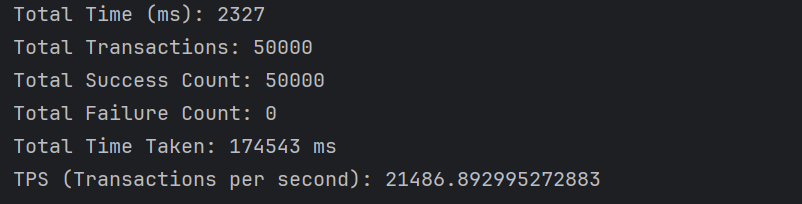




openGauss：

线程数为50/100，每个线程执行事务数为500





**总结：经过本次测试可以看到openGauss在高并发时的吞吐量确实要大于postgreSQL，而且当把线程数由50改为100之后，openGauss的吞吐量增加要比postgreSQL明显的多，说明openGauss更适合高并发查询环境。**

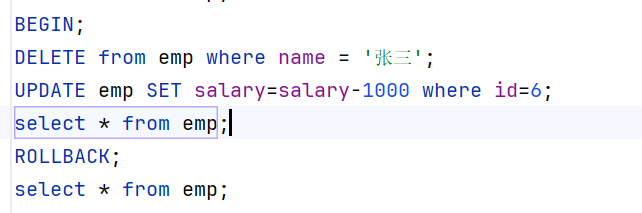
1. **可靠性测试**

在这部分的测试中，我将测试openGauss满足原子性，一致性，隔离性和持久性的特点，以说明其是一个能够正确保持数据存储状态的数据库。

在这部分的测试中，我参考了之前lab课上对postgreSQLACID性质的测试时使用的sql语句。创建了一个emp表，并向其中插入数据。（acid.sql）

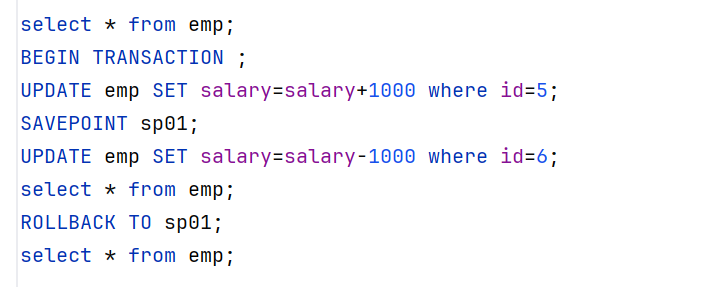


1. 原子性：事务所包含的操作要么全部完成，要么什么也没做。



在事务回滚后，两条语句都没有执行。

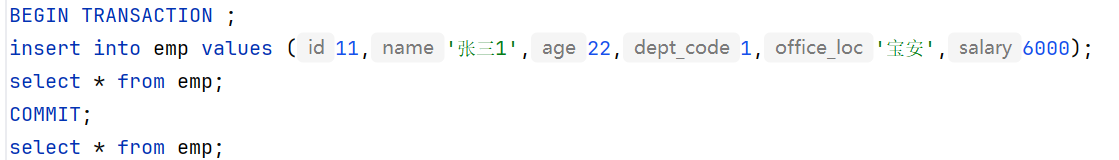
1. 一致性：在一致性数据库上执行事务后，数据库仍需保持为一致性的状态。



数据完全执行每一条命令。

1. 隔离性：没有结束的事务在提交之前不允许将其结果暴露给其它事务。

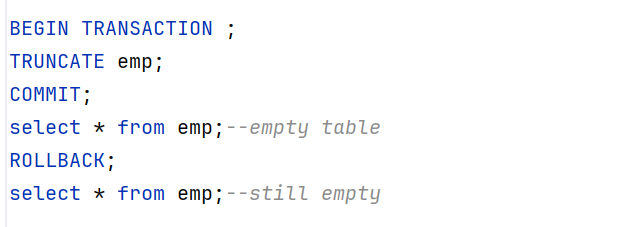
在console1中执行：



在console2中执行：select *\** from emp;

当console1中没有执行commit的时候，console2中不能查询到console1中刚刚插入的新信息，只有console1中执行完commit，console2才会查询到刚刚console1中执行的插入值。

1. 持久性：当一个事务的结果提交后，系统保证该结果不会因以后的故障而丢失。



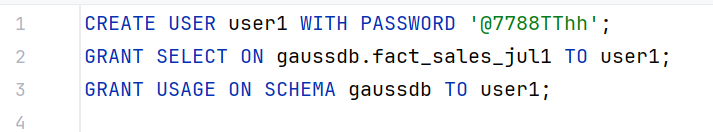
在执行commit语句之后，数据将不会丢失。

**总结：openGauss能够良好保持ACID特性。**

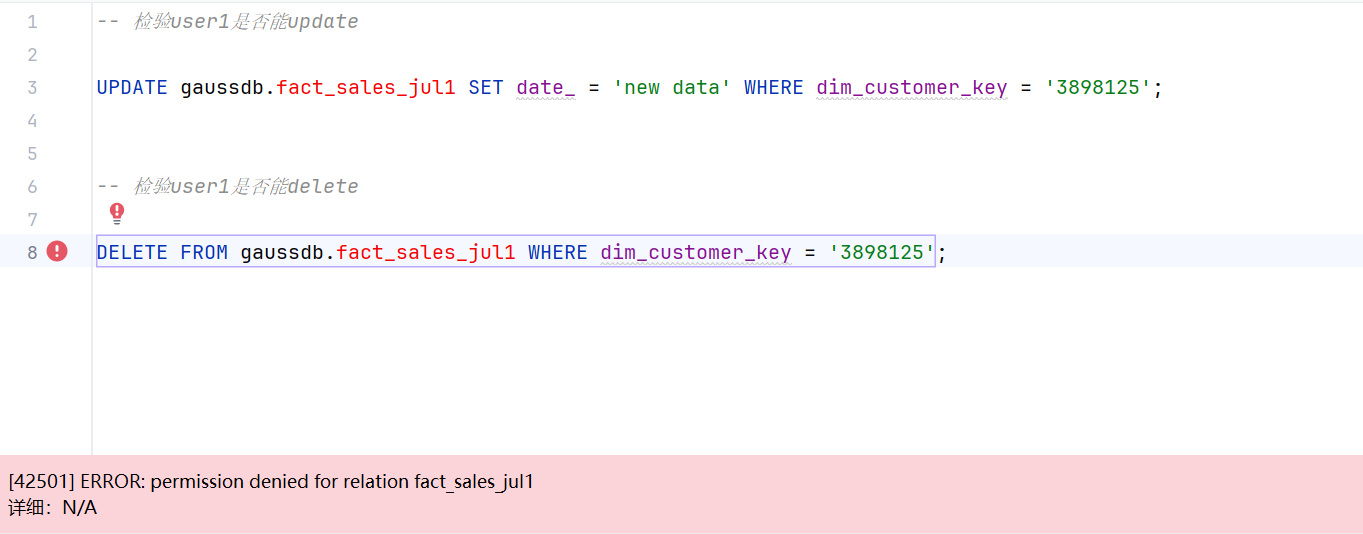
1. **安全性测试**

在这次测试中，实现了对于opengauss的安全性的简单测试，测试了openGauss数据库能够正确赋予用户一定的权限。（user.sql）

首先在gaussdb中赋予user1对于表fact\_sales\_jul1的select权限



在使用user1连接openGauss数据库的时候，执行update和delete操作的时候都会正确显示权限受限，可见openGauss数据库能够正常赋予并检验用户的权限。



1. **总结**
2. 实验结果分析

综上所述，通过进行对于openGauss的ACID性质的检验和能够正确赋予并检查用户权限的安全性检验可以得出openGauss具有一定的可靠性和安全性；通过对于openGauss和postgreSQL的从简单多次执行sql语句，执行复杂查询语句以及模拟多线程高并发查询的任务的方面进行性能测试和对比分析，可以初步得出，在数据量比较小和单线程执行查询任务的时候，openGauss不能主动进行并行查询优化，而postgreSQL可以使用并行查询，这也导致在这样的查询任务中postgreSQL数据库所用时间更短。而在模拟多线程的查询任务的时候，openGauss的吞吐量要明显优于postgreSQL，而且如果增加线程数的话，openGauss的吞吐量会有比较大的增长。

1. 对于openGauss的优缺点的初步总结

openGauss的优点：

**（1）**高吞吐量与并发性能**：** 在多线程查询任务中，openGauss表现出更高的吞吐量，并且随着线程数的增加，吞吐量显著提升。这表明openGauss在高并发场景下具有较强的性能优势，尤其适用于多线程并发查询。

**（2）**ACID属性的实现**：** openGauss在ACID（原子性、一致性、隔离性、持久性）特性上的实现较好，保证了数据的一致性与可靠性。

**（3）**用户权限控制与安全性**：** openGauss能够正确赋予并检查用户权限，具备较好的安全性管理机制。

### **openGauss的缺点：**

### ****查询优化**：** 在数据量较小和单线程查询时，openGauss未能主动进行并行查询优化，而是较为依赖传统的查询方式。与PostgreSQL相比，openGauss在这些场景下的查询性能较低，特别是在不能启用并行查询的情况下