openGauss AI特性创新实践课



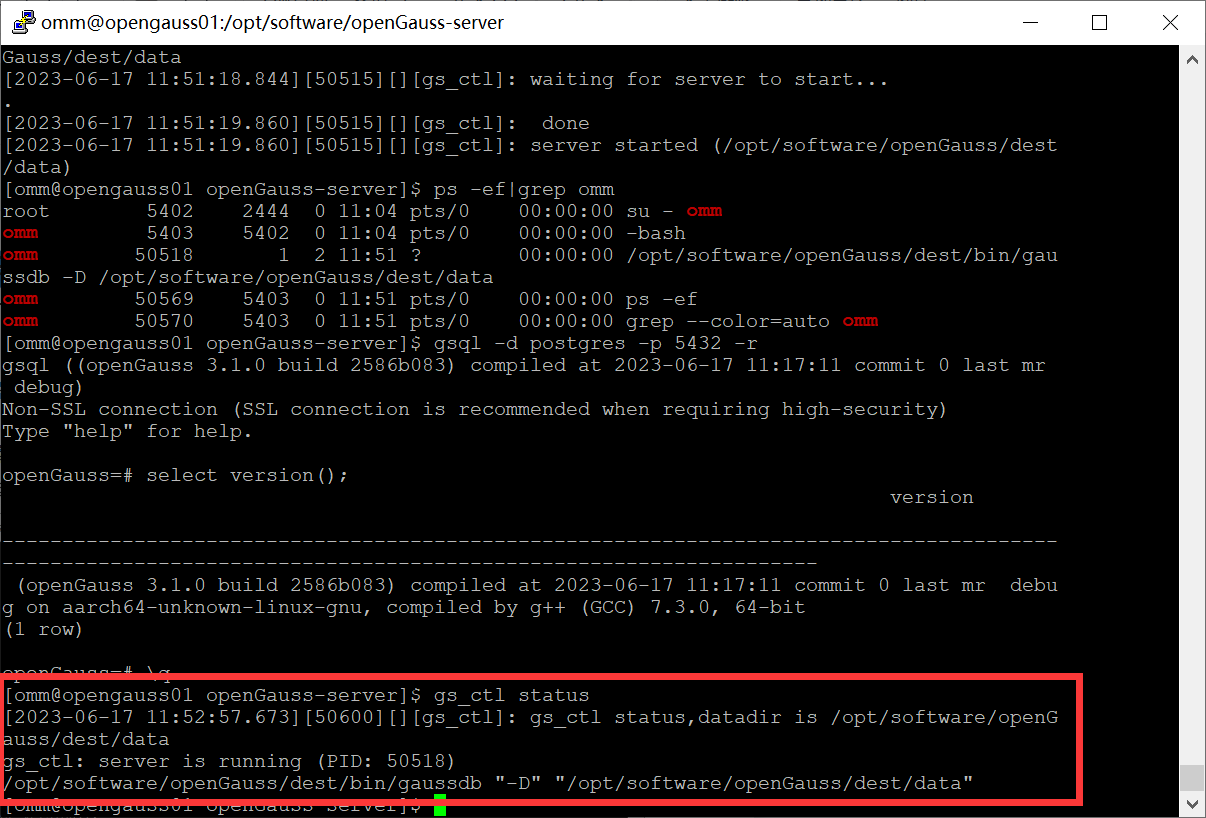
华为技术有限公司

# 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：

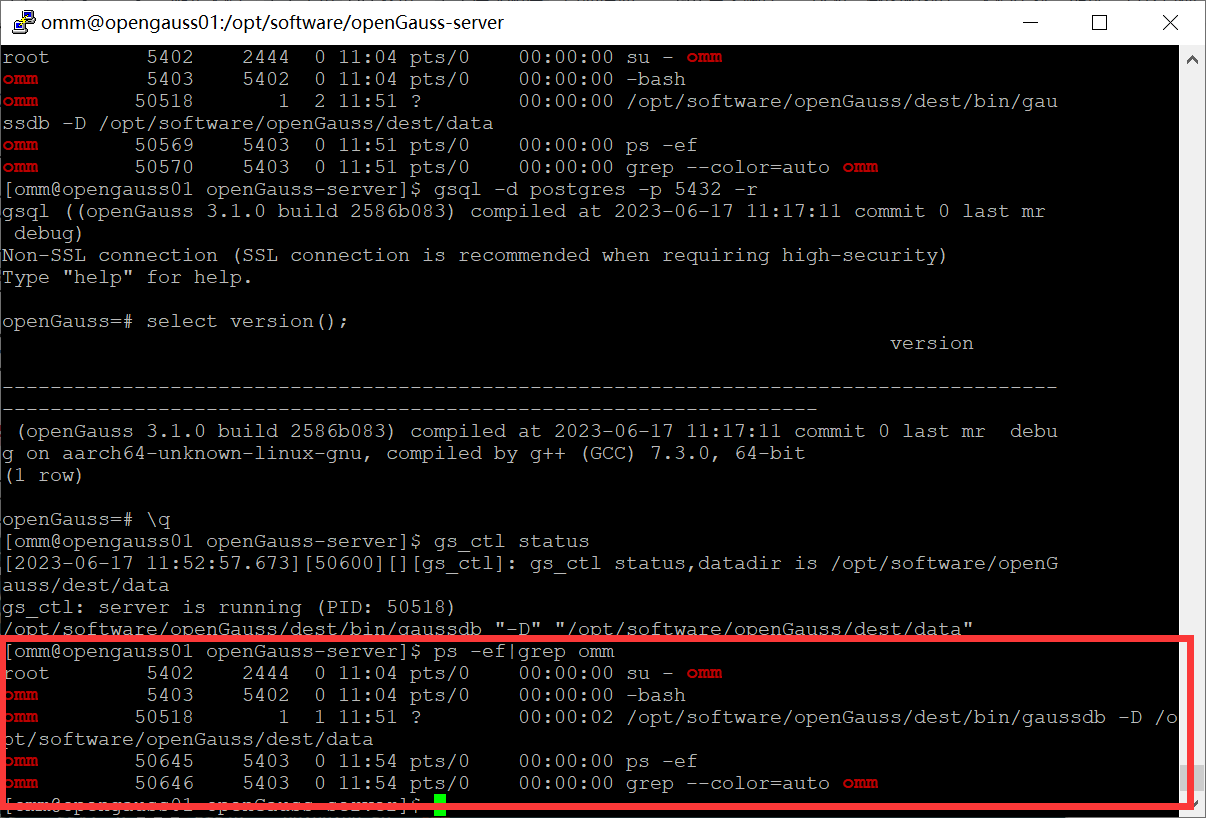
任务一：数据库状态验证

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）



安装数据库的步骤：

1. 创建好华为云ecs服务器并使用putty登录华为云ecs服务器
2. 创建 openGauss 数据库的安装用户 omm 及其属组 dbgrp、创建 openGauss 源码存放及 openGauss 安装路径
3. 下载第三方编译库openGaussthird\_party\_binarylibs.tar.gz并解压
4. 下载 openGauss 源码
5. 安装依赖包
6. 修改权限
7. 切换omm用户，设置环境变量，并使环境变量生效
8. 进入 openGauss 源码下，生成配置文件，然后编译并安装

实验思考题：为什么需要通过源码编译，安装数据库？

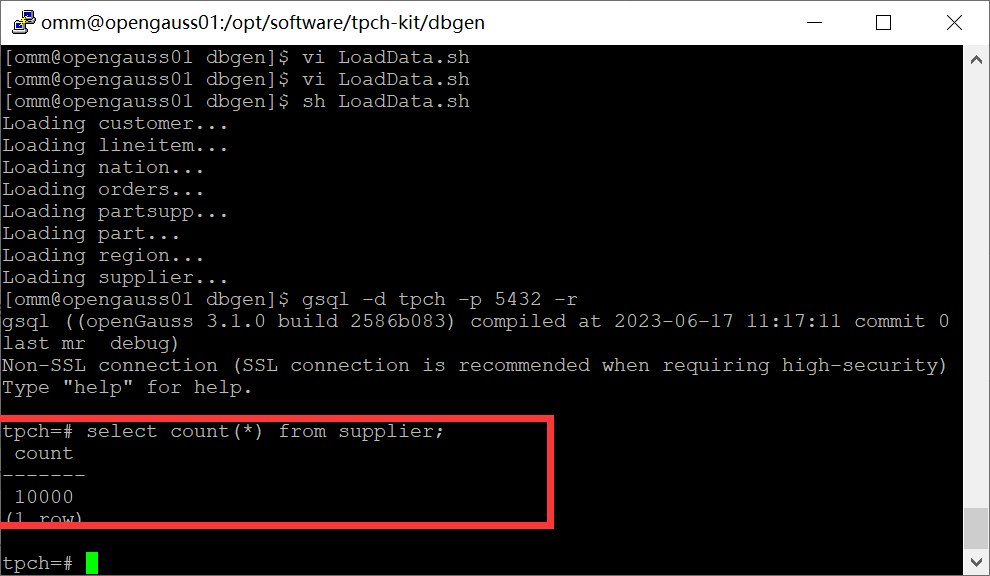
使用源码编译安装数据库可以按照自己的需求进行安装，可以指定版本，设置参数等。

# 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：

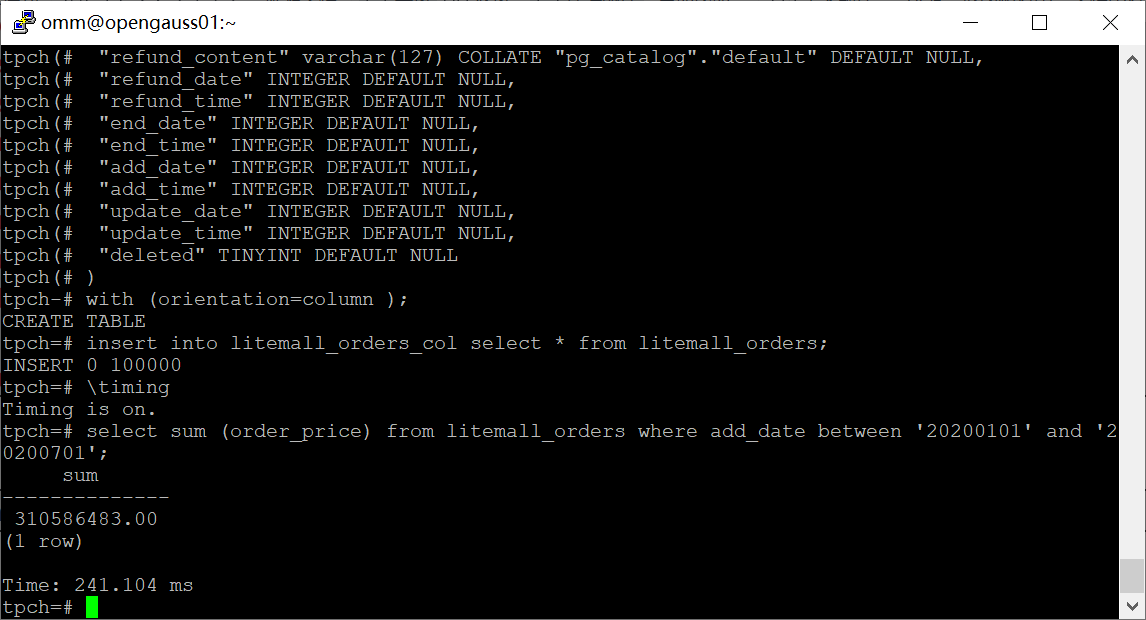
select count(\*) from supplier;;



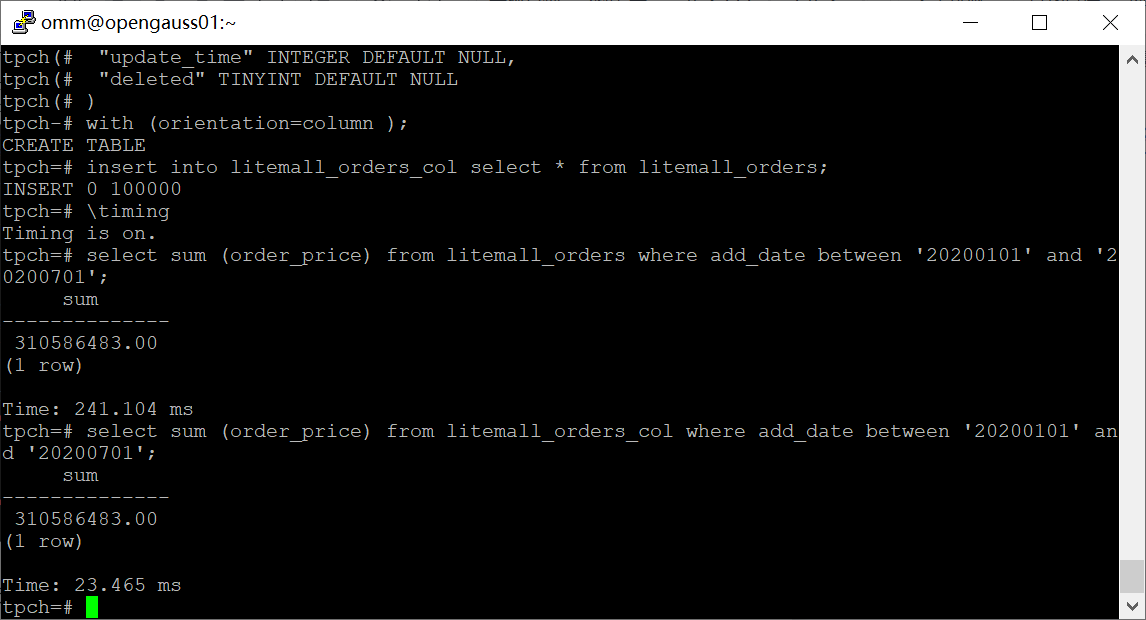
任务二：行存表与列存表执行效率对比

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';

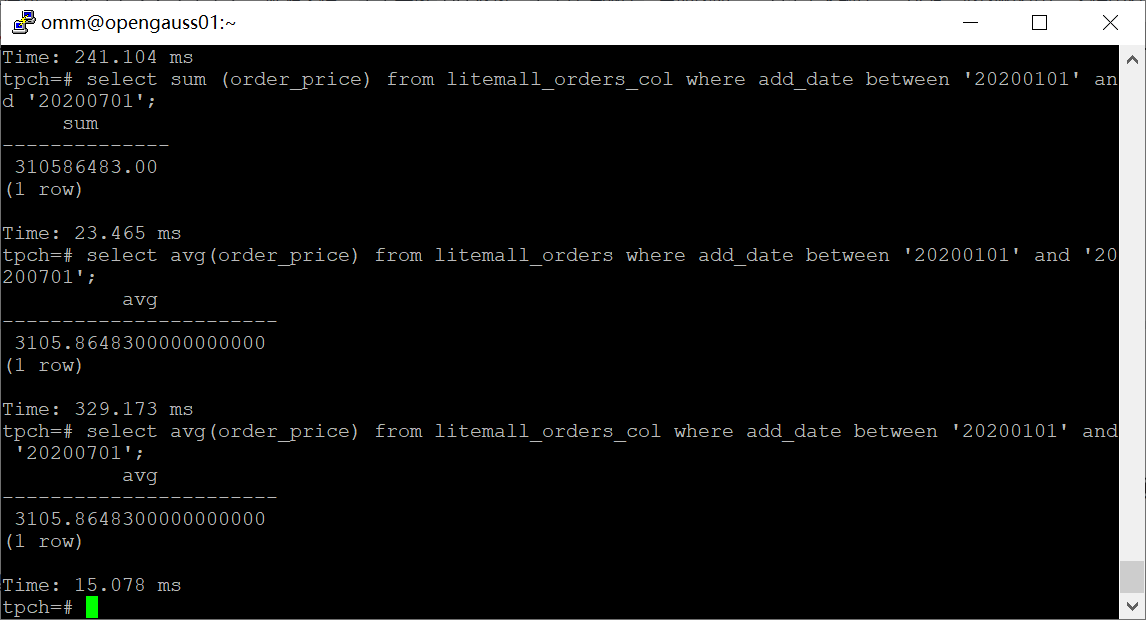


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

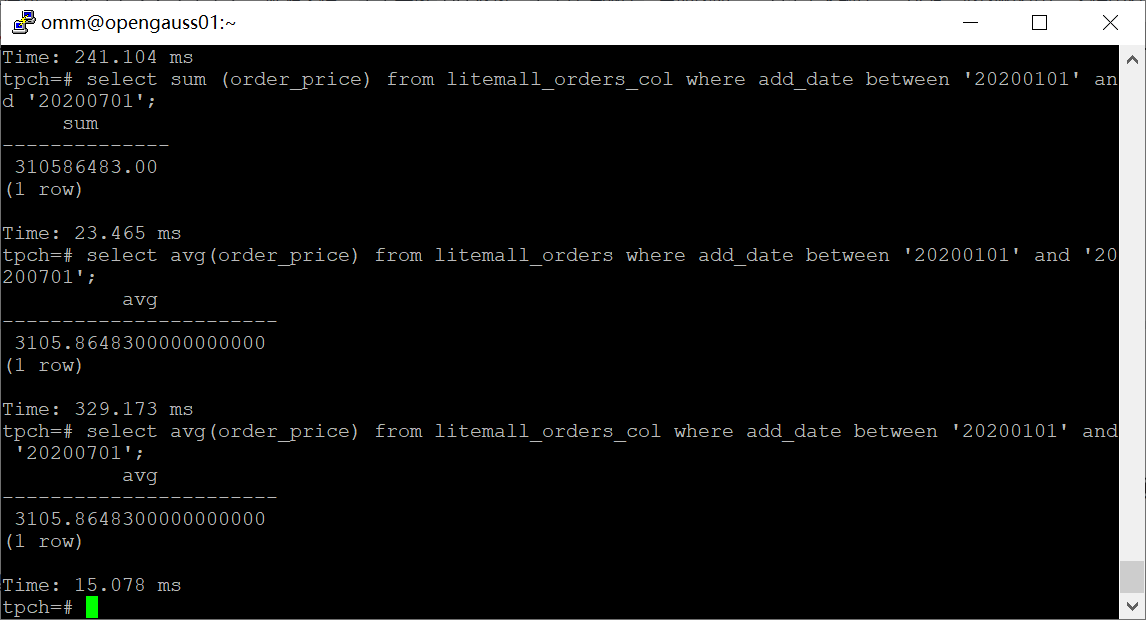


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

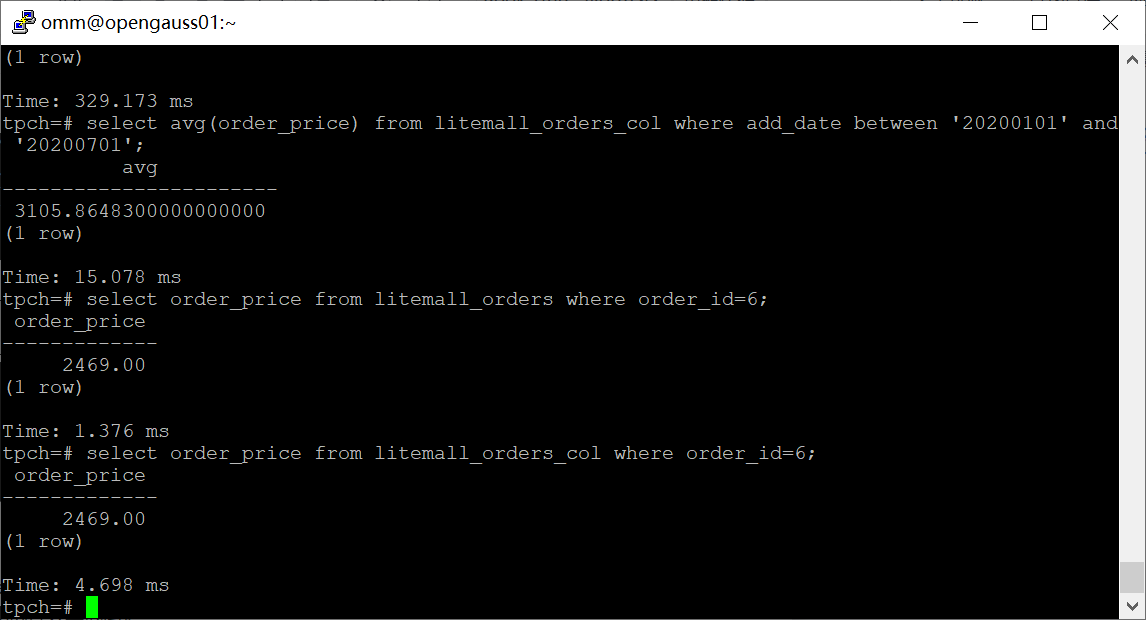


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

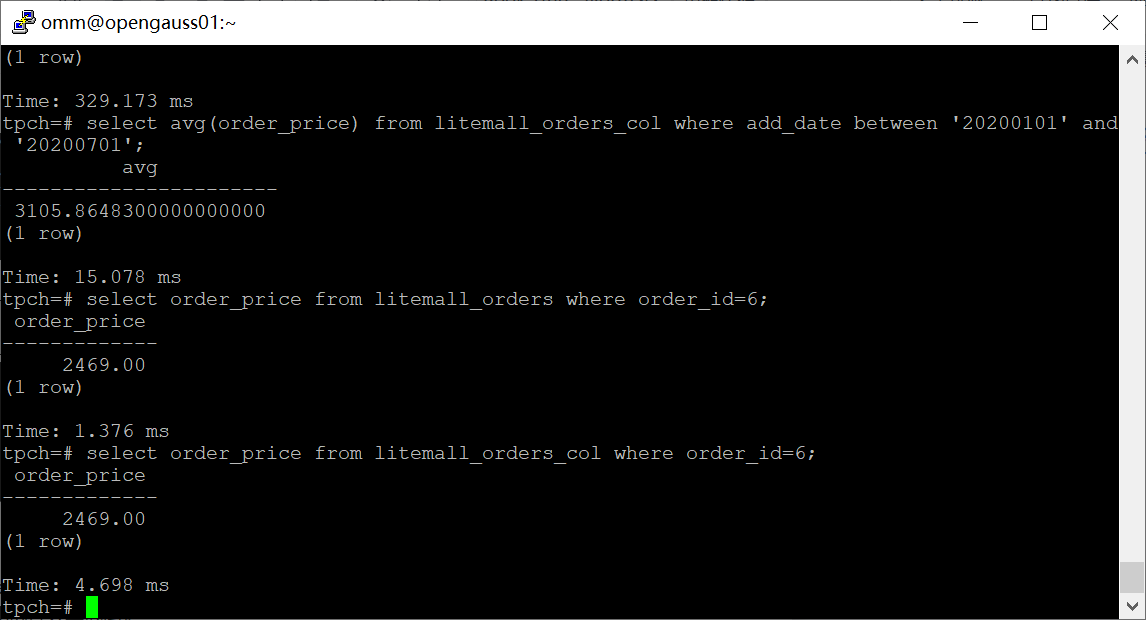


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

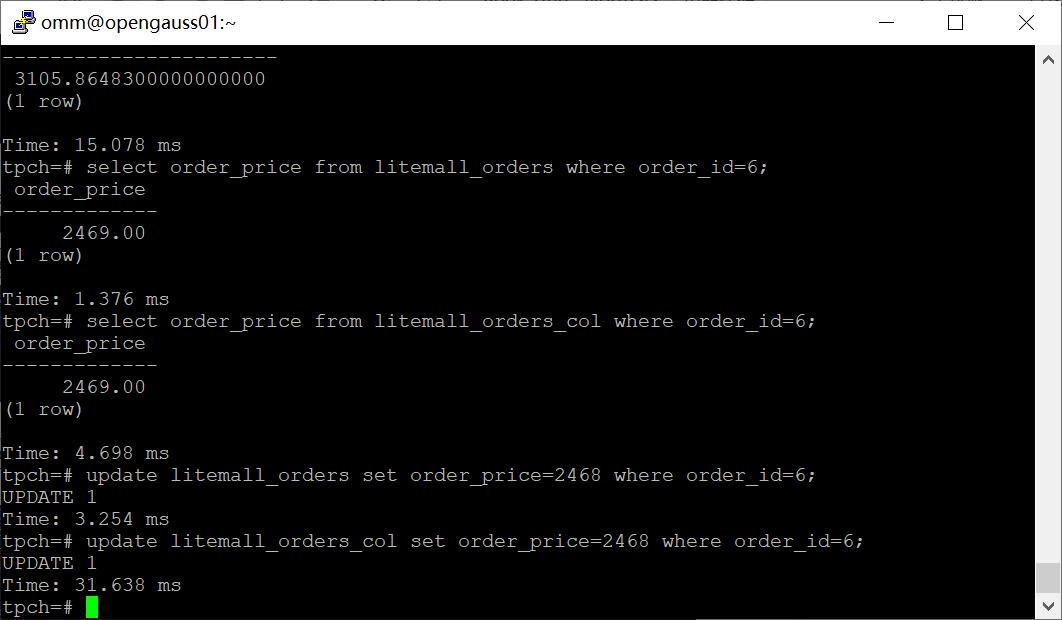


select order\_price from litemall\_orders\_col where order\_id=6;

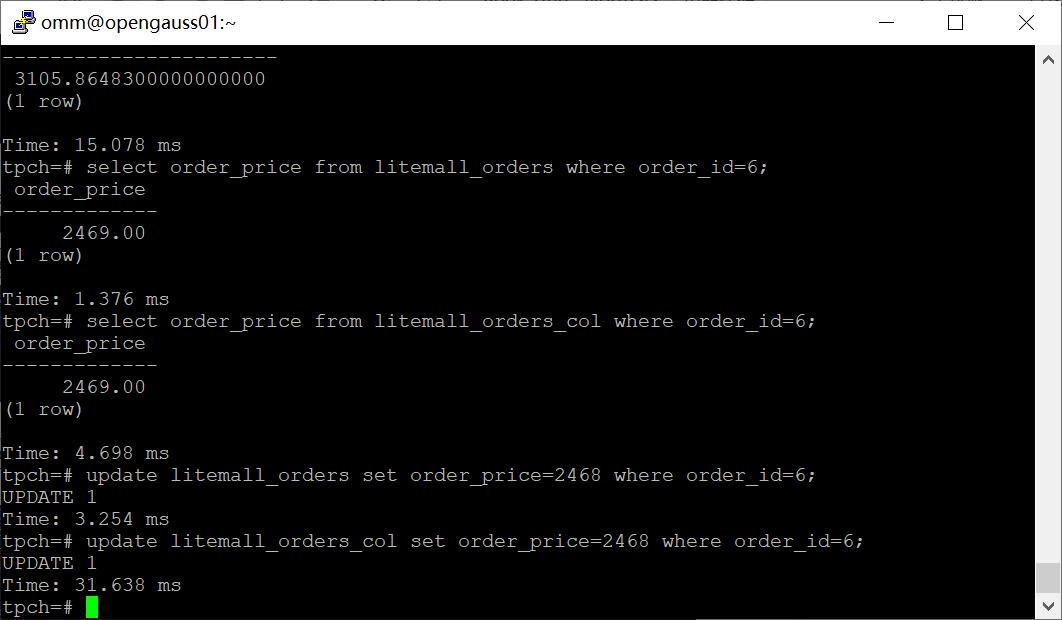


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

update litemall\_orders set order\_price=2468 where order\_id=6;



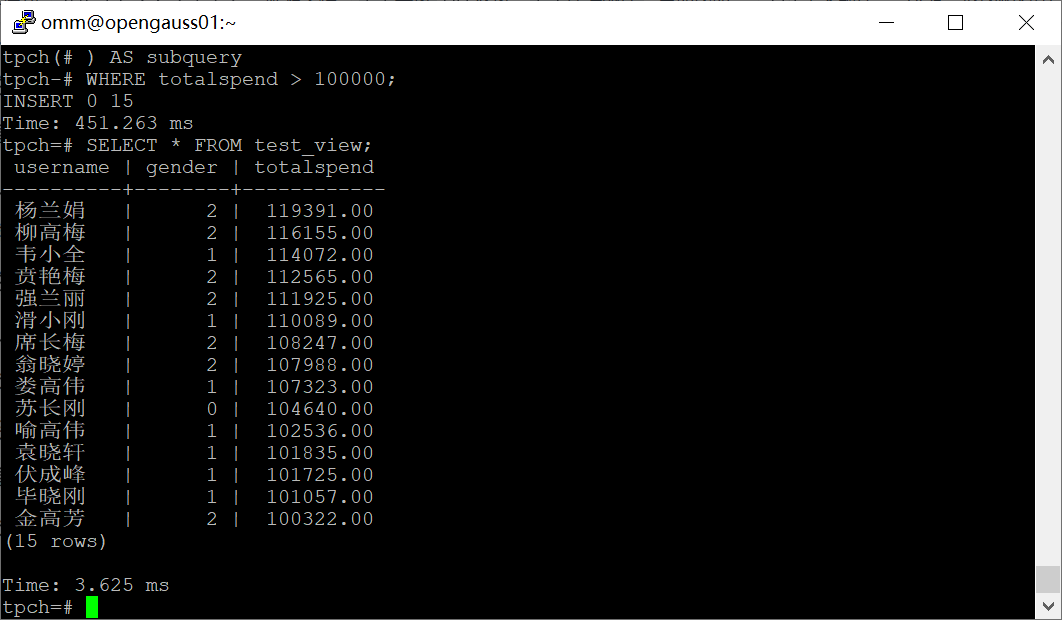
update litemall\_orders\_col set order\_price=2468 where order\_id=6;



任务三：物化视图的使用

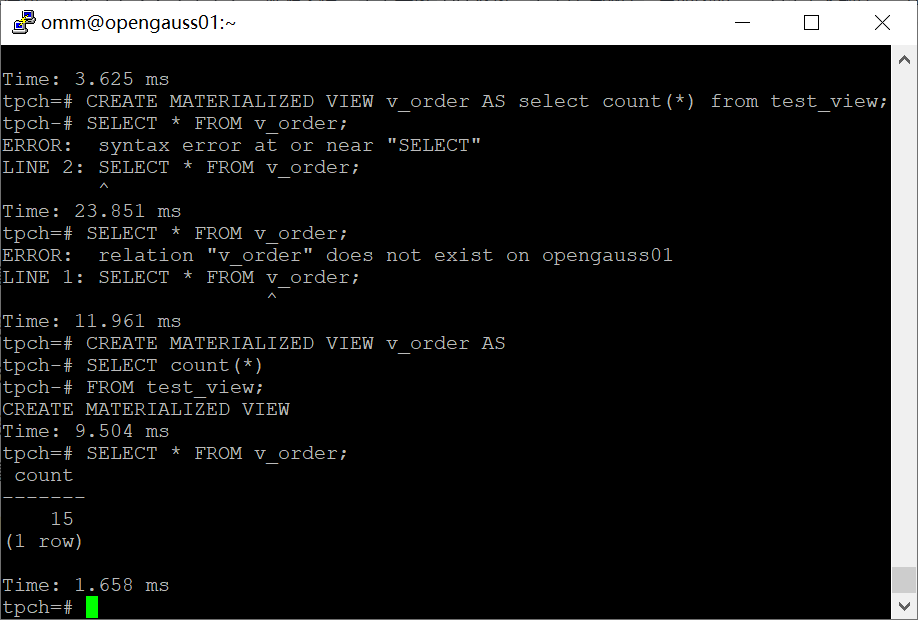
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



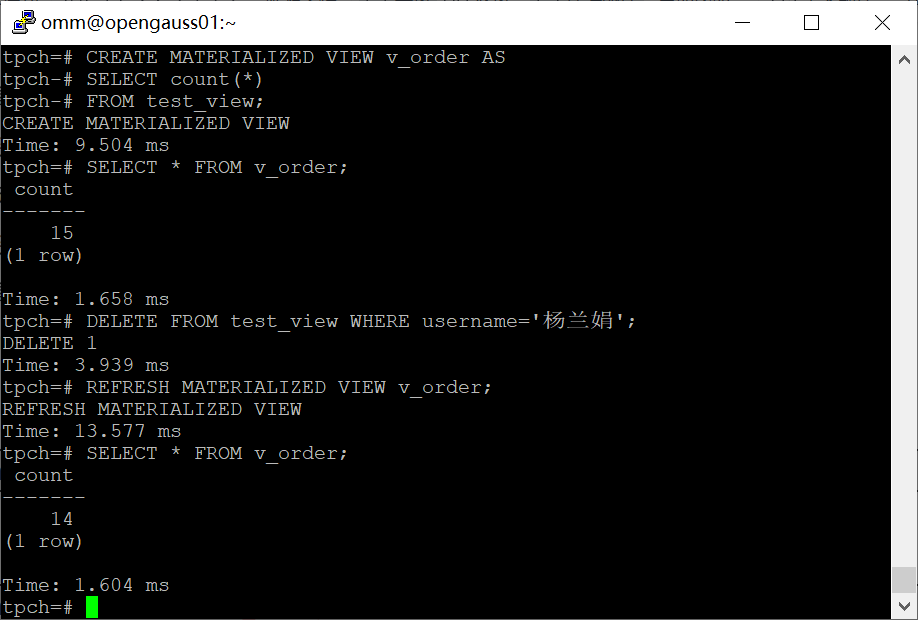
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



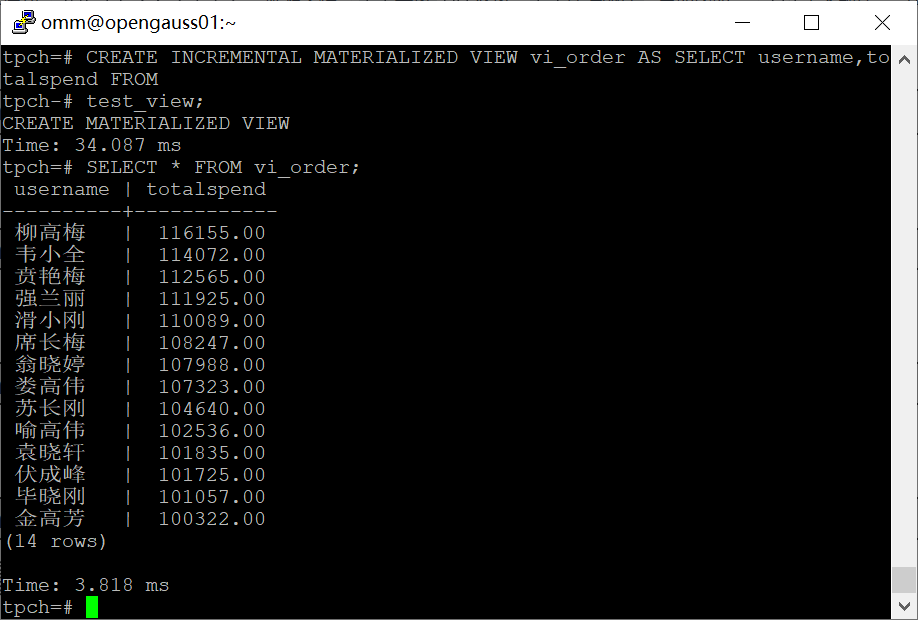
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



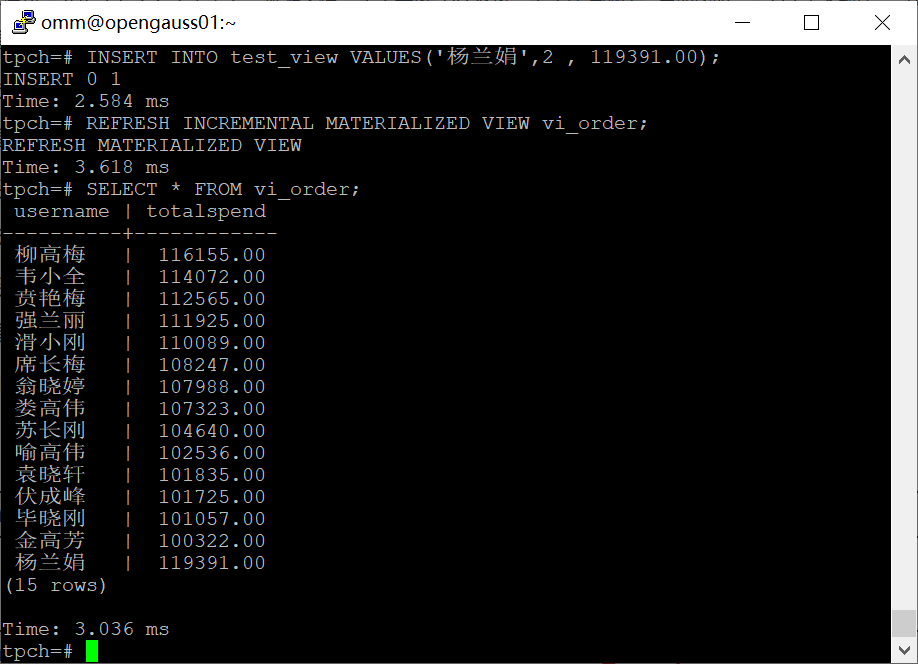
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM vi\_order;

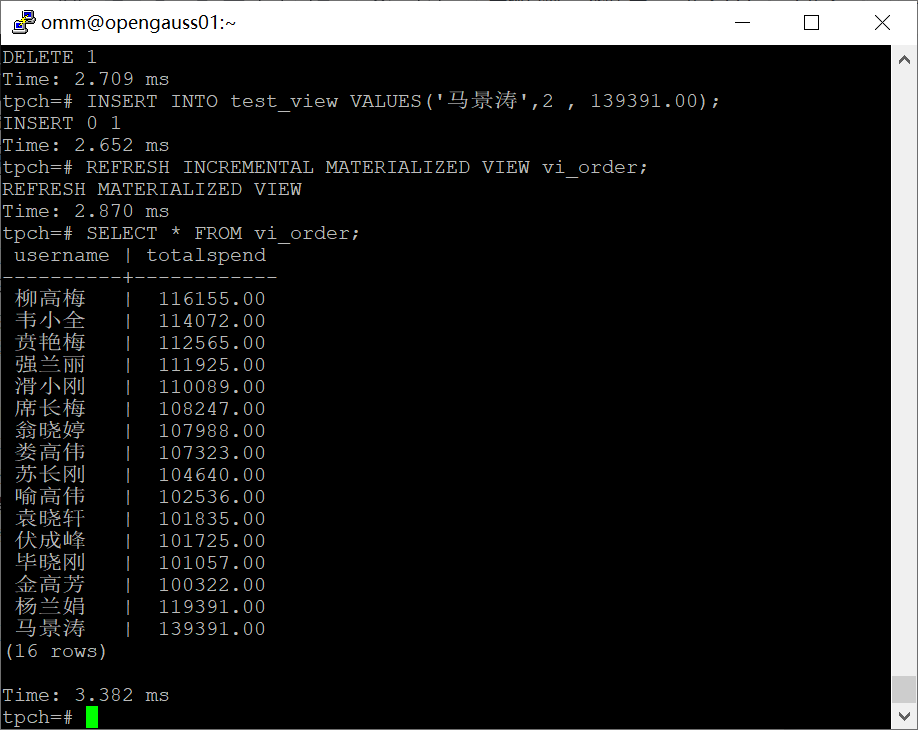


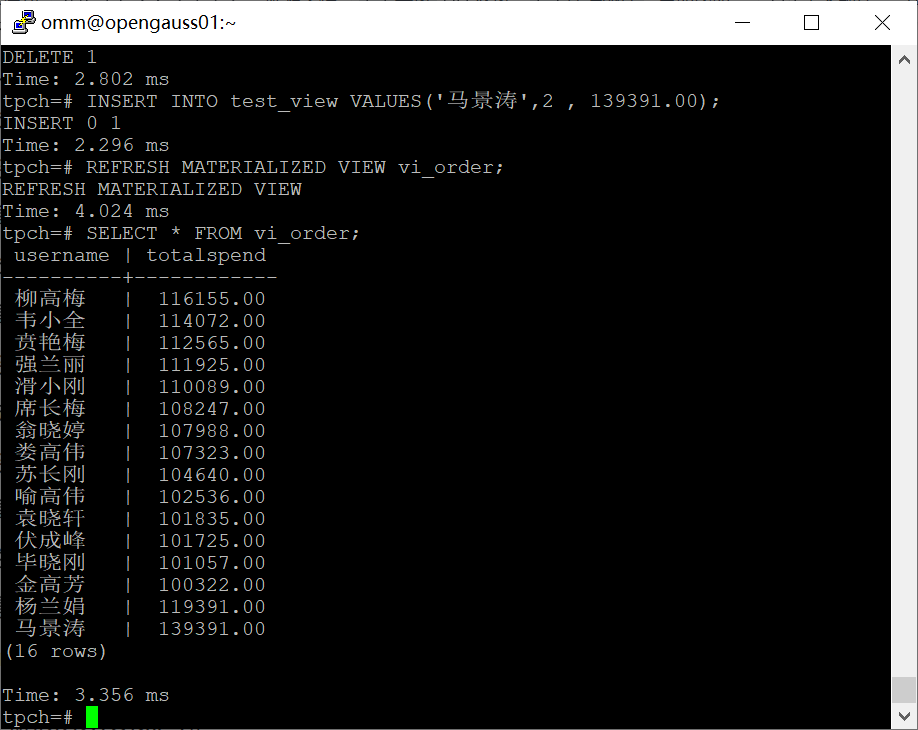
5. 对表进行操作后，刷新增量物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM vi\_order;



增量更新物化视图：



全量更新物化视图：

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？

行存表将数据按行存储在磁盘上，每一行数据包含了一条记录的所有字段，当执行查询时，行存表需要读取整行数据；列存表每一列数据都被存储在一起，当执行查询时，列存表读取所需的列数据。它们们的存储方式的不同导致了他们在执行相同sql语句时执行的时间不同。OLTP

在进行需要读取大量列操作的查询时，比如聚合等，行存表需要读取大量的数据，包括不使用的字段，但列存表不需要，它可以减少不必要的io开销，效率更高；而需要进行插入和更新整行的数据时，行存表通常较快，因为数据整行存储，直接插入即可，而列存表中由于数据被分解成列，需要在相应的列中查询并修改。Olap

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：

全量物化视图是根据基表的全部数据进行计算和存储的，仅支持对创建好的物化视图做全量更新。当基表数据发生变化时，需要重新计算整个物化视图，并将结果存储起来。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：

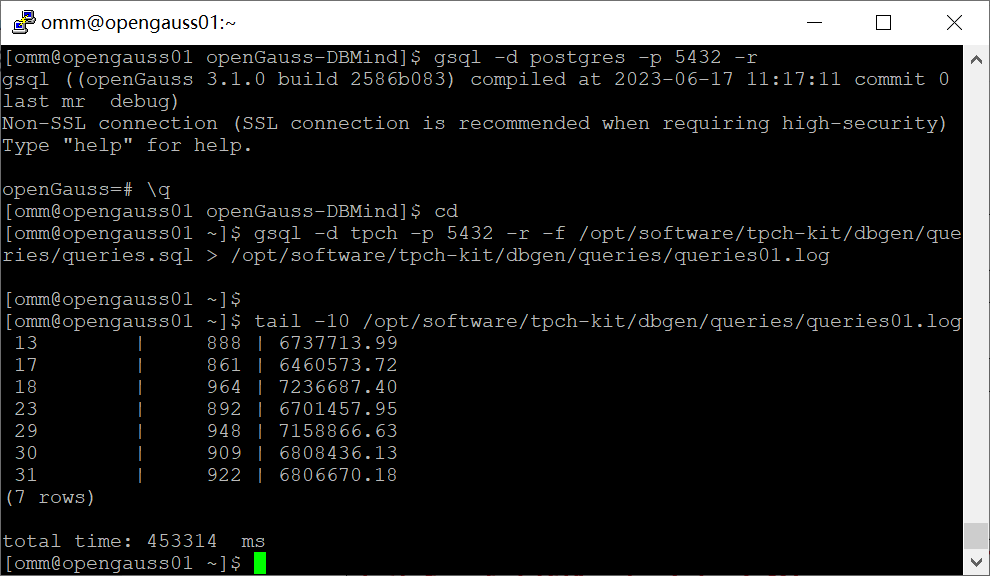
增量物化视图只针对基表的部分数据进行计算和存储，对物化视图增量刷新。当基表的指定数据发生变化时，只需计算和更新受影响的部分物化视图数据，可以减少计算和存储的开销。与全量创建物化视图不同在于目前增量物化视图所支持场景较小，目前物化视图创建语句仅支持基表扫描语句或者UNION ALL语句。。

# 关卡三、openGauss的AI4DB特性应用

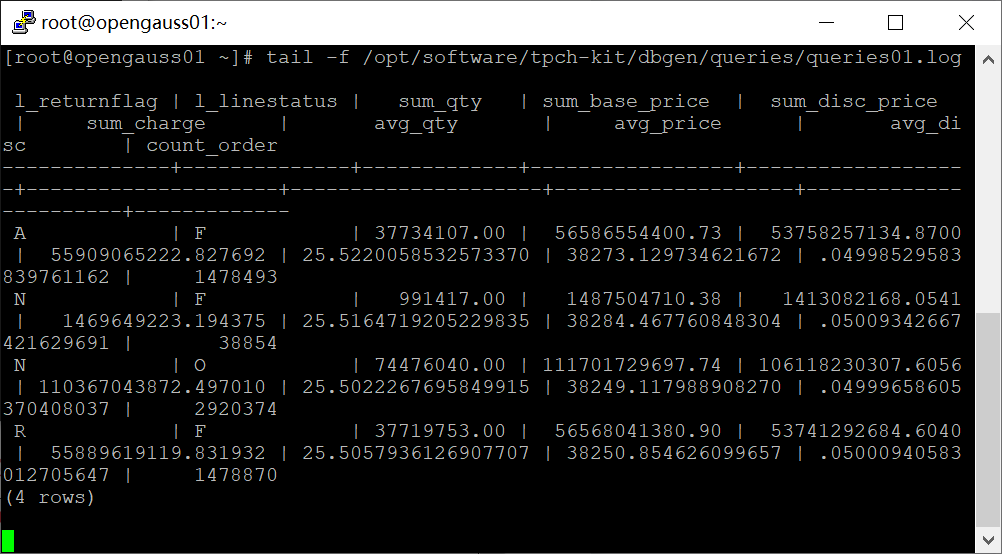
任务一：使用X-Tuner进行参数优化

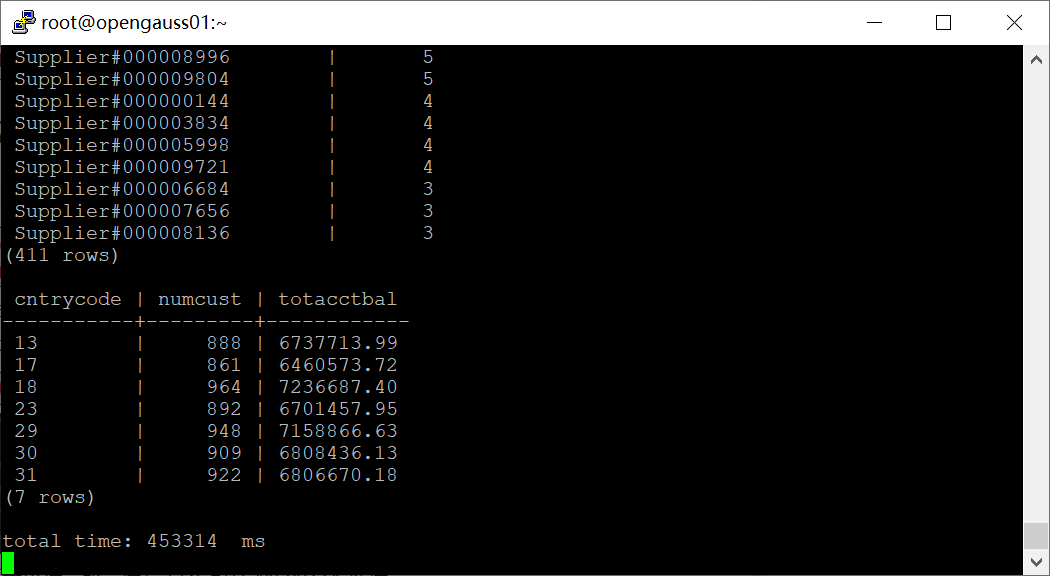
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log



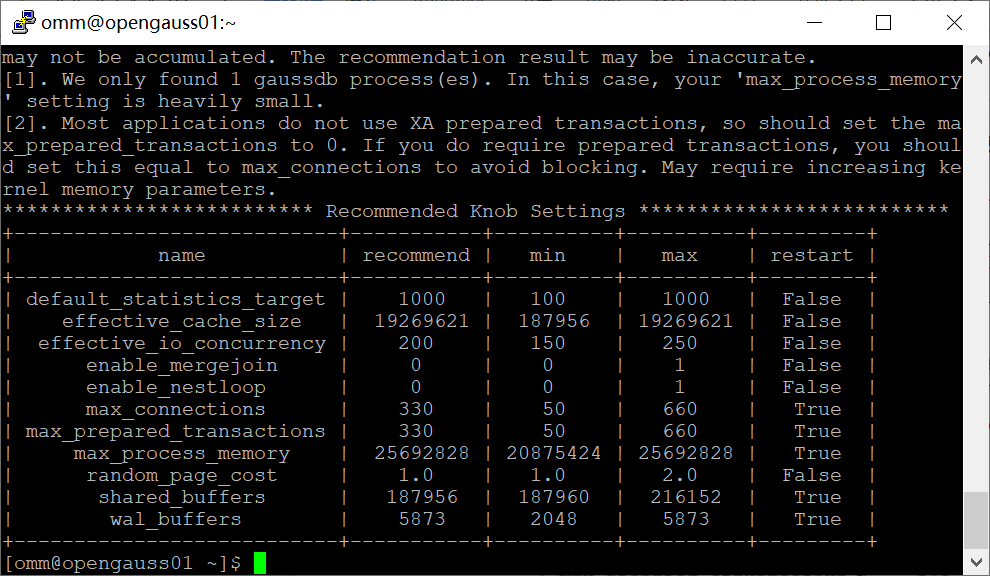
监控截图：



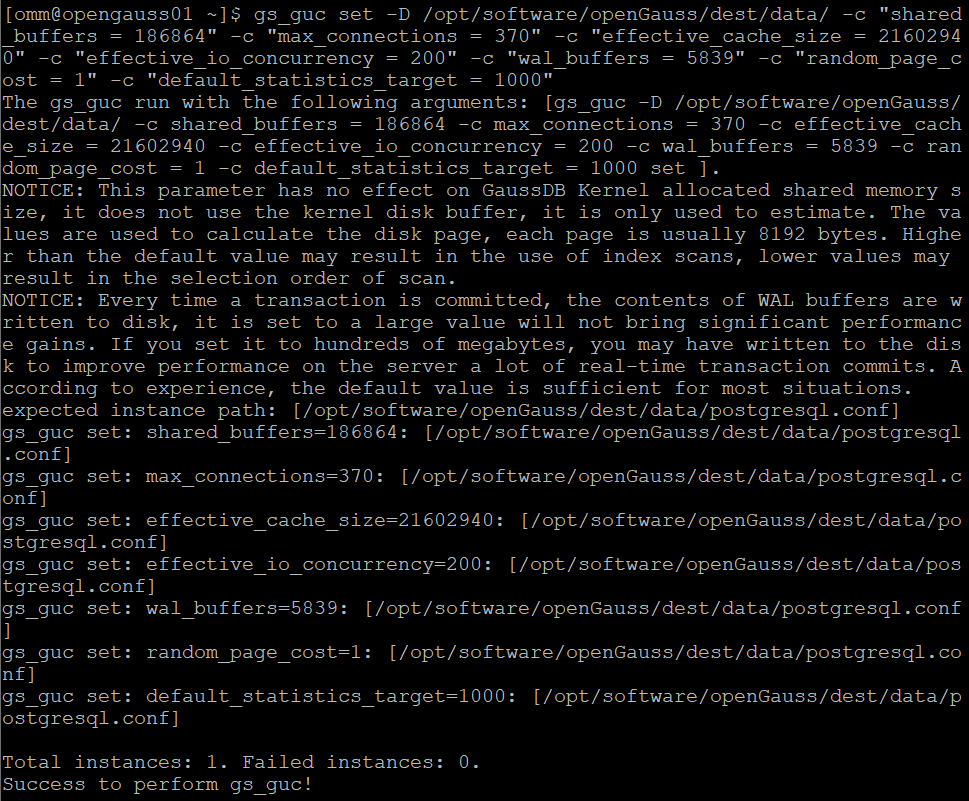


2. 使用root用户，执行X-Tuner进行参数建议优化，将执行结果截图

gs\_dbmind component xtuner recommend --database tpch --db-user omm --db-port 5432 --db-host 192.168.0.122 --host-user omm



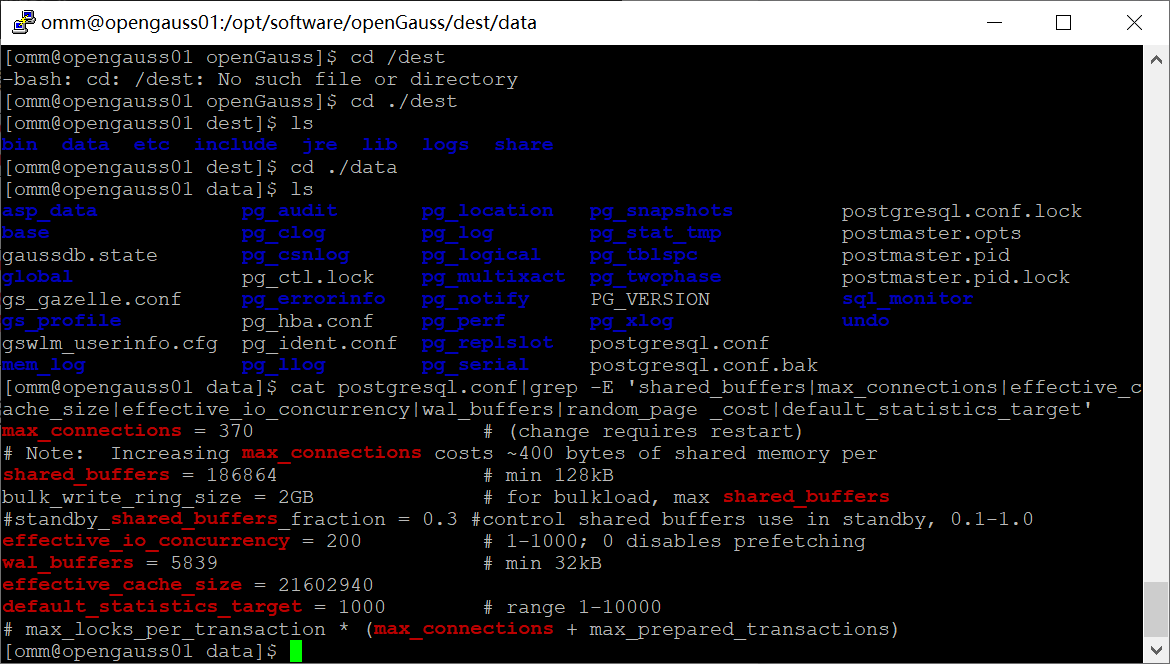
设置数据库的参数：



3.重启完成后，获取参数值：

cd /opt/software/openGauss/dest/data

cat postgresql.conf|grep -E 'shared\_buffers|max\_connections|effective\_cache\_size|effective\_io\_concurrency|wal\_buffers|random\_page\_cost|default\_statistics\_target'



任务二：使用Index-advisor对select 查询语句进行优化，并通过对比执行计划，得到优化前后的不同。

1. 使用explain，对查询2020年3月订单表收入并进行排序的SQL加以分析，将结果截图。

EXPLAIN

SELECT ad.province AS province, SUM(o.actual\_price) AS GMV

FROM litemall\_orders o,

address\_dimension ad,

date\_dimension dd

WHERE o.address\_key = ad.address\_key

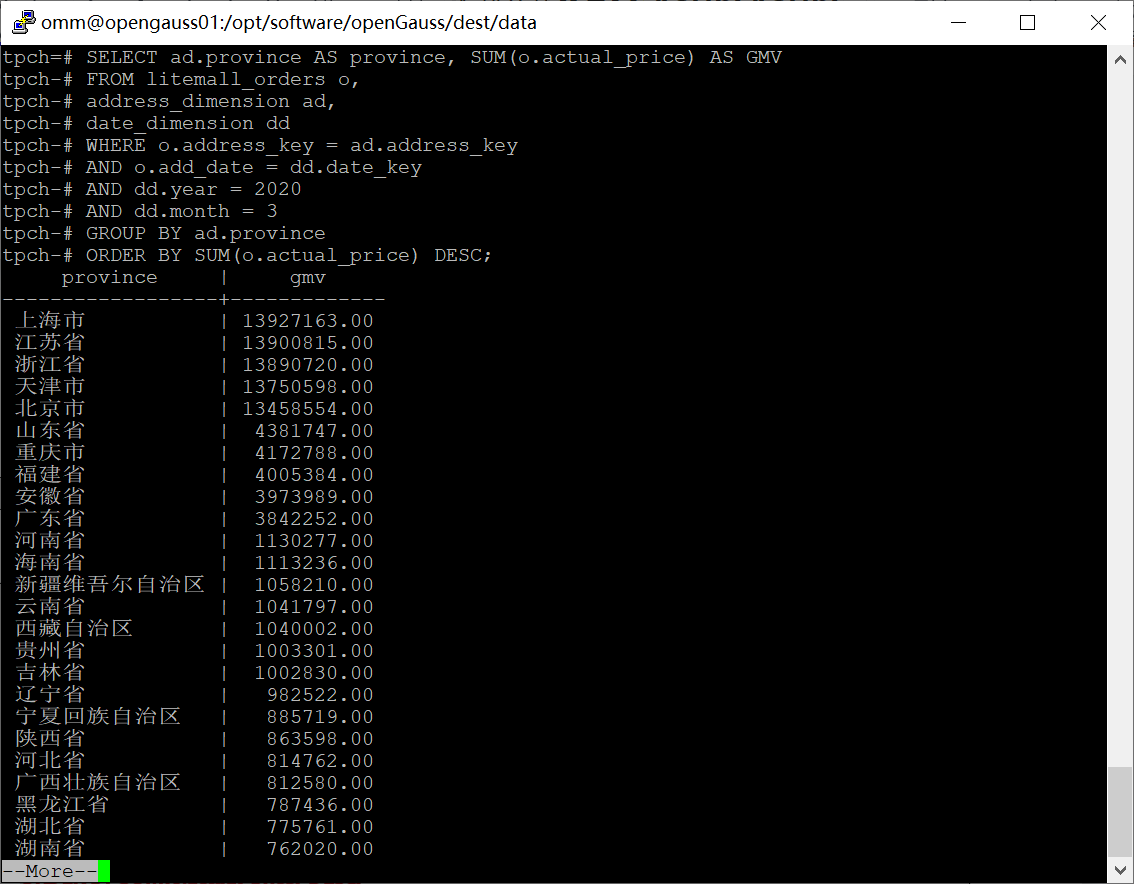
AND o.add\_date = dd.date\_key

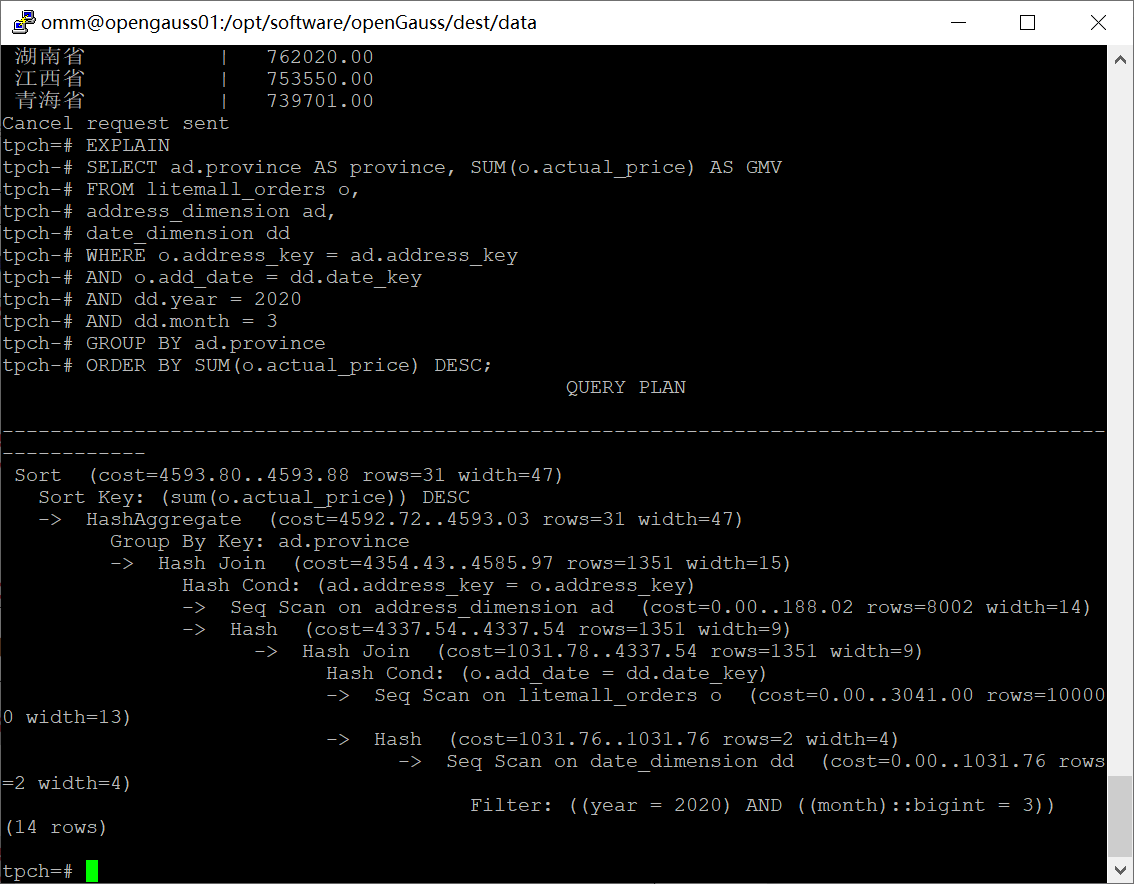
AND dd.year = 2020

AND dd.month = 3

GROUP BY ad.province

ORDER BY SUM(o.actual\_price) DESC;





2. 使用索引推荐功能，对查询语句进行推荐，将执行结果截图。

select \* from gs\_index\_advise('

SELECT ad.province AS province, SUM(o.actual\_price) AS GMV

FROM litemall\_orders o,

address\_dimension ad,

date\_dimension dd

WHERE o.address\_key = ad.address\_key

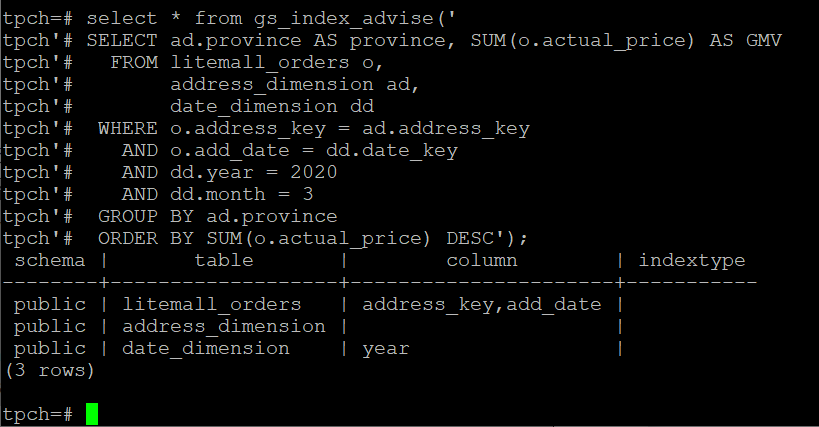
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

GROUP BY ad.province

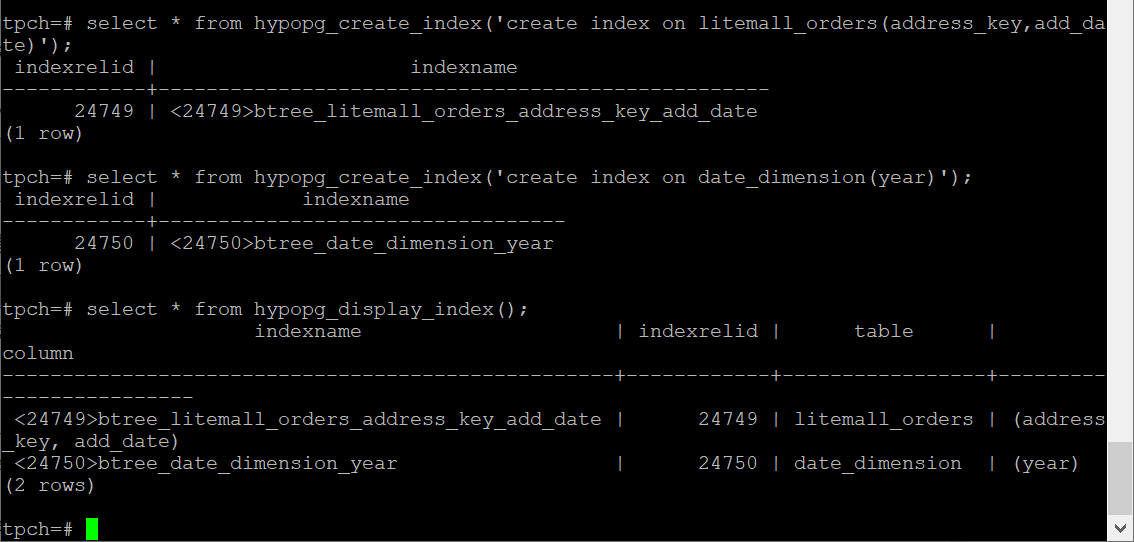
ORDER BY SUM(o.actual\_price) DESC');



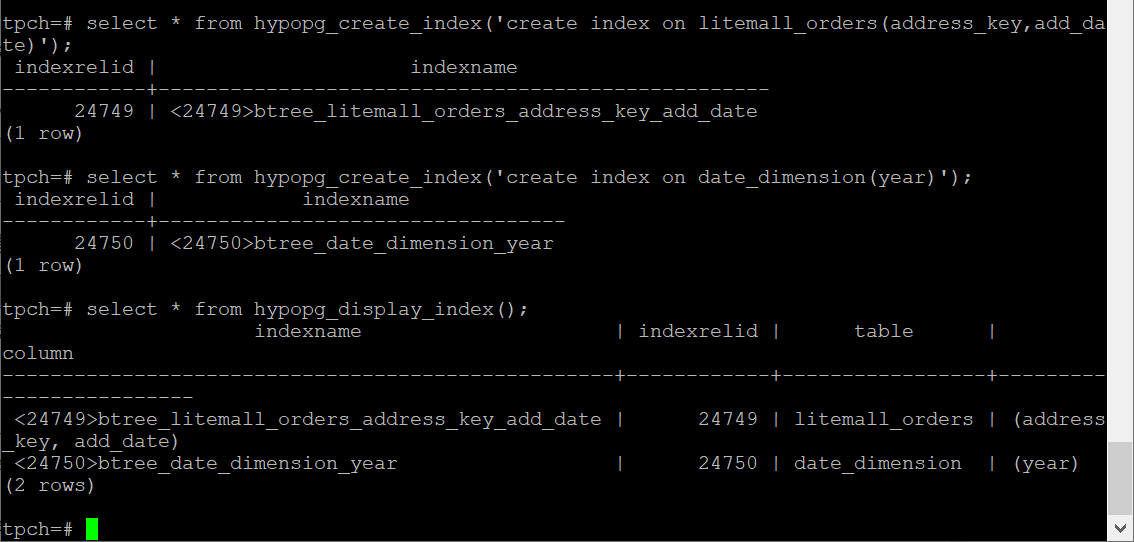
3. 查看创建的虚拟索引列，将执行结果截图。

select \* from hypopg\_display\_index();

创建虚拟索引列：



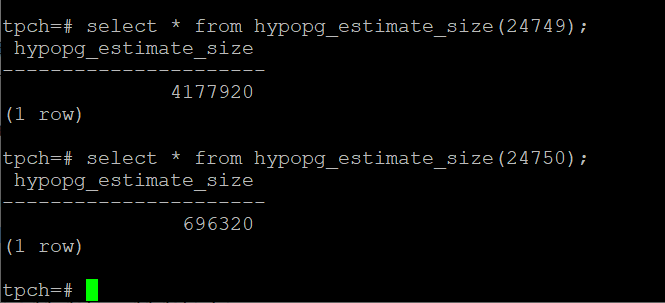
查看结果：



4. 获取索引虚拟列大小结果（单位为：字节），将执行结果截图。

select \* from hypopg\_estimate\_size(16715);

select \* from hypopg\_estimate\_size(16716);



5.再次使用explain，对该SQL加以分析，将执行结果截图。

EXPLAIN

SELECT ad.province AS province, SUM(o.actual\_price) AS GMV

FROM litemall\_orders o,

address\_dimension ad,

date\_dimension dd

WHERE o.address\_key = ad.address\_key

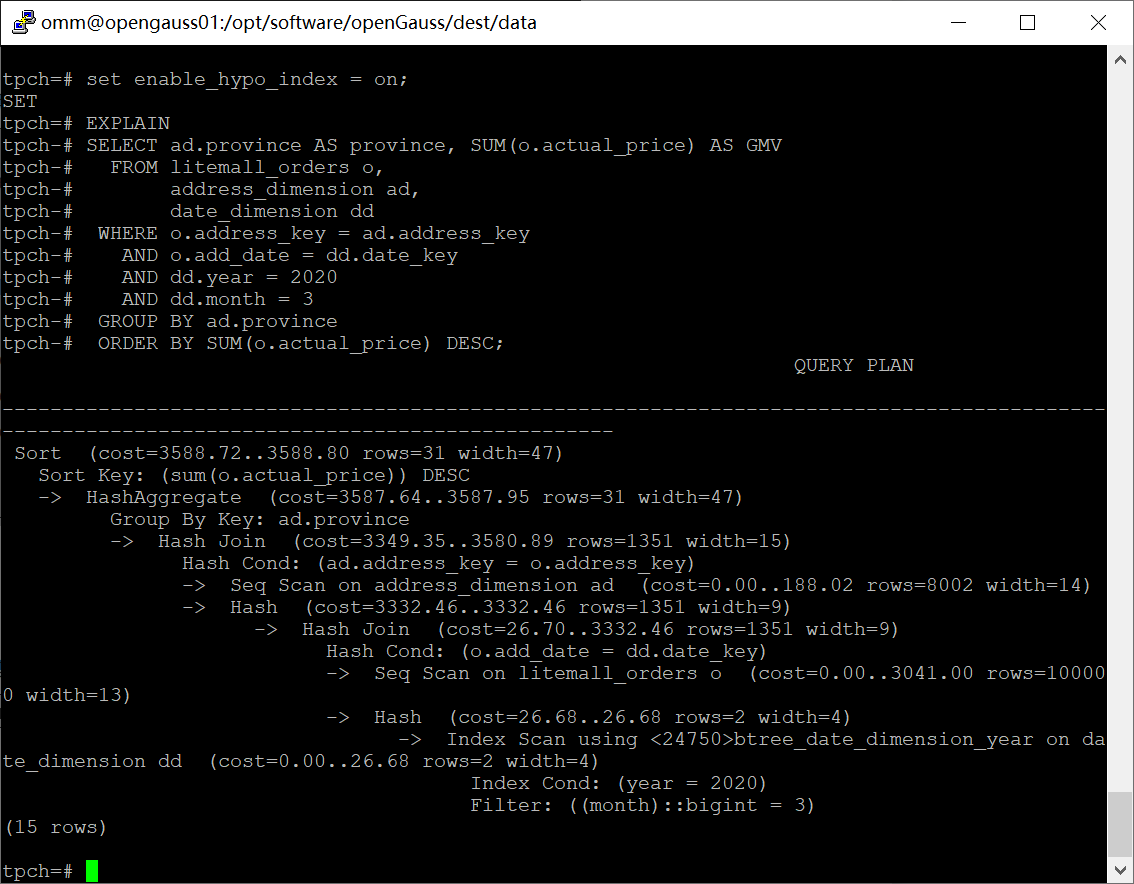
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

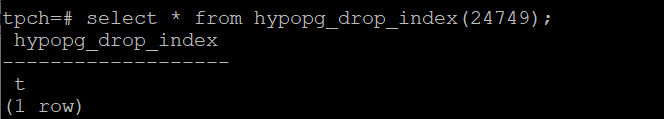
GROUP BY ad.province

ORDER BY SUM(o.actual\_price) DESC;



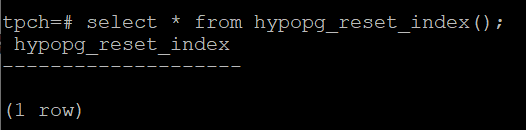
6. 删除某一个索引虚拟列，将执行结果截图。

select \* from hypopg\_drop\_index(16715);



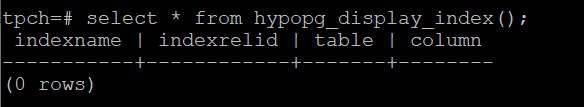
7. 删除某一个索引虚拟列，将执行结果截图。

select \* from hypopg\_reset\_index();



8. 查看索引虚拟列，将执行结果截图。

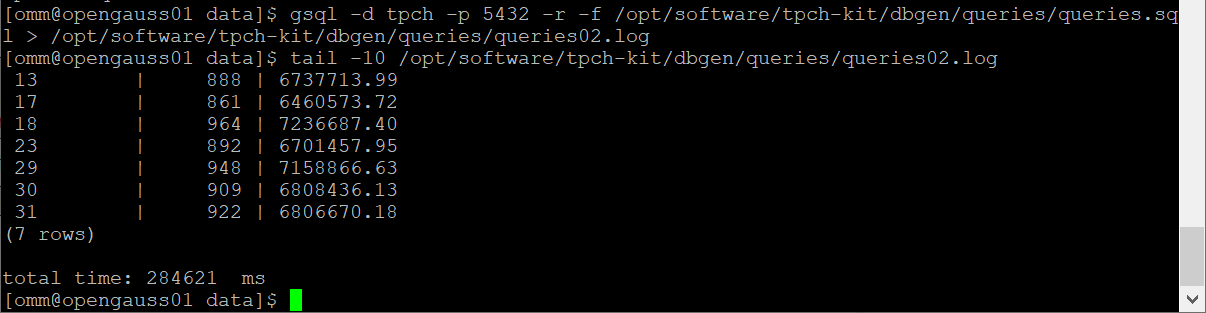
select \* from hypopg\_display\_index();



任务三：通过创建索引，对queries.sql中的SQL语句进行优化，并对比优化前后queries.sql执行的时间。

1. 重新执行queries.sql查询，将执行结果截图：

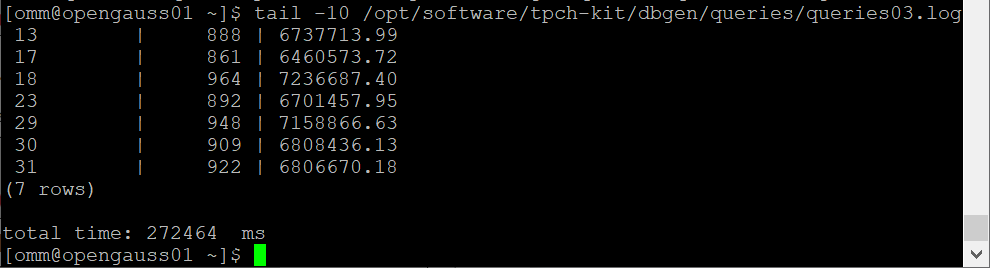
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log



挑战一：进一步优化queries.sql中的查询语句，使得前后执行时间出现倍数级的提升。

1. 重新执行queries.sql查询，将执行结果截图：

gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries03.log



实践思考题1：根据X-Tuner给出的参数优化，在哪些参数上进行了优化，为何要对这些参数进行优化？

对以下参数进行了优化：

shared\_buffers: 共享缓冲区大小设置为186864，共享缓冲区是用于在内存中缓存数据库的数据和索引的区域。通过增加共享缓冲区的大小，可以提高数据库的性能，减少磁盘IO操作的频率，从而加快查询和数据访问速度。

max\_connections: 最大连接数设置为370， 共享缓冲区是用于在内存中缓存数据库的数据和索引的区域。通过增加共享缓冲区的大小，可以提高数据库的性能，减少磁盘IO操作的频率，从而加快查询和数据访问速度。

effective\_cache\_size: 有效缓存大小设置为21602940，增加有效缓存大小可以增加数据库在内存中的可用缓存空间，从而提高查询的性能和响应时间。

effective\_io\_concurrency: 有效io并发数设置为200，效IO并发数是指数据库在执行IO操作时可以使用的并发线程数。通过增加有效IO并发数，可以提高数据库在执行IO操作时的并发性能，从而加快IO操作的完成速度。

wal\_buffers: WAL缓冲区大小设置为5839，WAL缓冲区是用于存储事务日志的缓冲区。增加WAL缓冲区的大小可以提高事务日志写入的性能，从而提高数据库的稳定性和可靠性。

random\_page\_cost: 随机页面成本设置为1，随机页面成本是用于查询优化器估算随机磁盘访问成本的参数。将随机页面成本设置为较低的值可以让查询优化器更倾向于选择索引访问而不是全表扫描，从而提高查询的性能。

default\_statistics\_target: 默认统计目标设置为 1000，随机页面成本是用于查询优化器估算随机磁盘访问成本的参数。将随机页面成本设置为较低的值可以让查询优化器更倾向于选择索引访问而不是全表扫描，从而提高查询的性能。

实践思考题2：索引的使用，对于执行SQL有什么好处？除了使用索引和参数外，还有哪些方面可以对数据库进行优化？

索引在数据库中的使用可以带来以下好处：

1. 提高查询性能

2. 加速排序和连接操作

3. 优化数据的访问路径

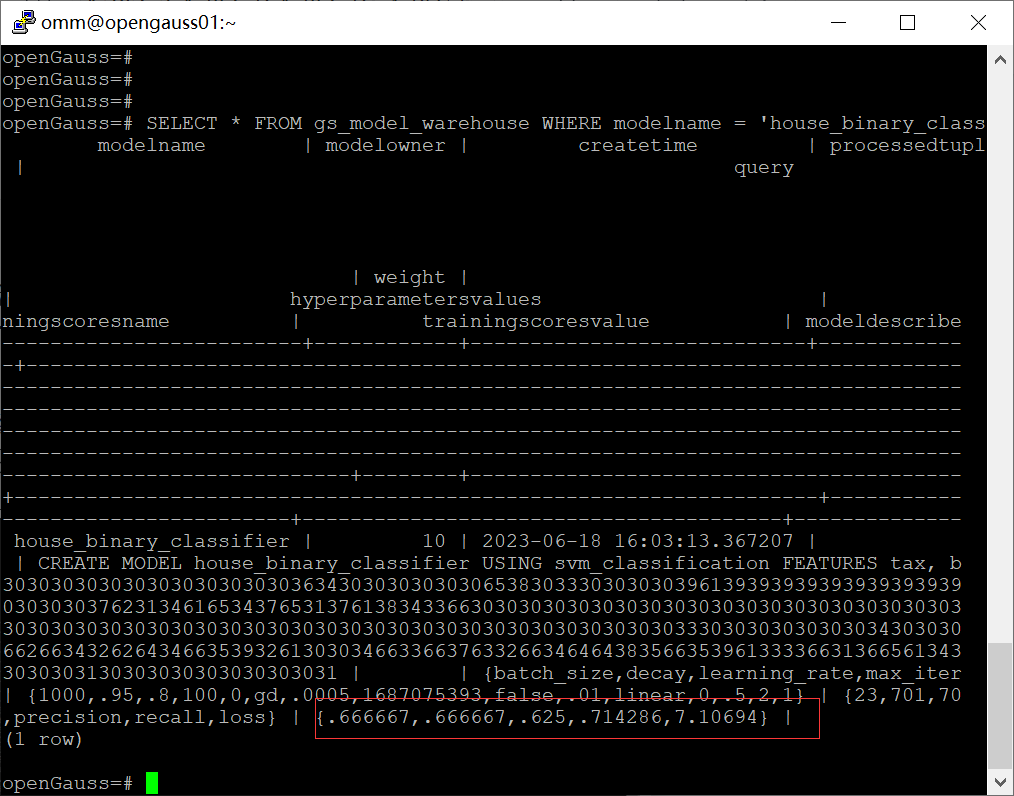
除了使用索引和参数进行优化外，还有其他方面可以对数据库进行优化，包括：

优化数据库的架构、优化查询语句、调整缓存大小、优化内存分配策略等。

# 关卡四、openGauss的DB4AI特性应用

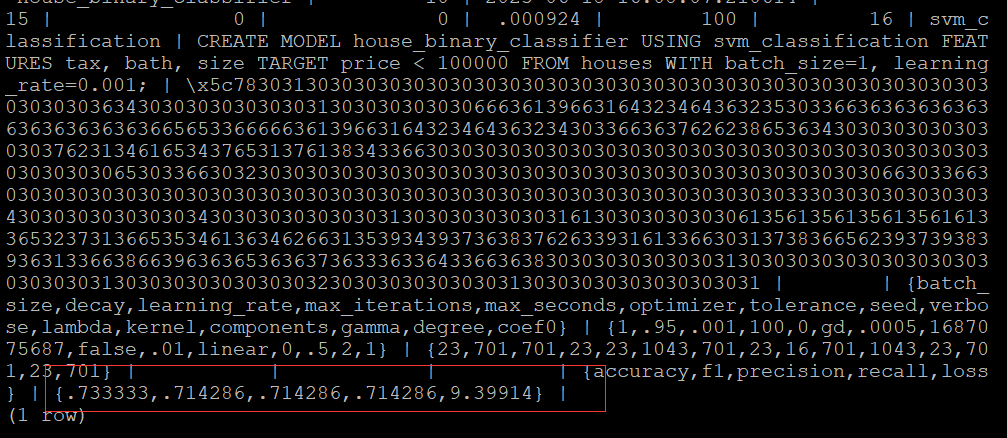
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

postgres=# SELECT \* FROM gs\_model\_warehouse WHERE modelname = 'house\_binary\_classifier';



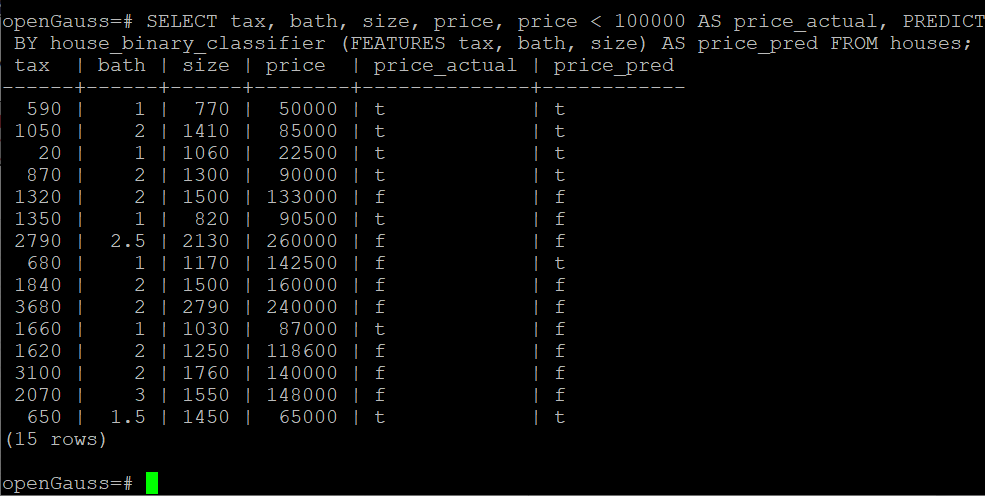
任务二：观察新模型的信息，将执行结果截图。

postgres=# SELECT \* FROM gs\_model\_warehouse WHERE modelname = 'house\_binary\_classifier';



任务三：利用训练好的逻辑回归模型预测数据，并与SVM算法进行比较，将执行结果截图。

postgres=# SELECT tax, bath, size, price, price < 100000 AS price\_actual, PREDICT BY house\_binary\_classifier (FEATURES tax, bath, size) AS price\_svm\_pred, PREDICT BY house\_logistic\_classifier (FEATURES tax, bath, size) AS price\_logistic\_pred FROM houses;



实践思考题1：分类模型与回归模型有何不同？

回归模型和分类模型的区别在于输出变量的类型：连续变量的预测叫回归，离散变量的预测是分类。

实践思考题2：什么是SVM算法？

支持向量机svm是定义在特征空间上的间隔最大的线性分类器。

具体来说就是在线性可分时，在原空间寻找两类样本的最优分类超平面。在线性不可分时，加入松弛变量并通过非线性映射将低维输入空间的样本映射到高维空间使其变为线性可分，这样就可以在该特征空间中寻找最优分类超平面。

实践思考题3：分类问题有哪些评价指标，请分别说明他们的含义？

1. 精确率：精确率是指在所有被分类为正例的样本中，真正例的比例。它衡量了分类器在预测为正例的样本中的准确性。精确率的计算公式为：精确率 = 真正例数 / (真正例数 + 假正例数)。

2. 召回率：召回率是指在所有实际为正例的样本中，被正确预测为正例的比例。它衡量了分类器对于正例样本的查全率。召回率的计算公式为：召回率 = 真正例数 / (真正例数 + 假反例数)。

3. F1分数：F1分数是精确率和召回率的调和平均值，综合考虑了分类器的准确性和查全率。F1 分数的计算公式为：F1 = 2 \* (精确率 \* 召回率) / (精确率 + 召回率)。

4. 准确率：准确率是指在所有样本中，被正确分类的比例。它衡量了分类器的整体分类准确性。准确率的计算公式为：准确率 = (真正例数 + 真反例数) / (真正例数 + 真反例数 + 假正例数 + 假反例数)。

5. Fβ分数：Fβ分数是精确率和召回率的加权调和平均值，通过调节 β 值可以给予精确率或召回率更高的权重。当 β = 1 时，即为 F1分数。Fβ分数的计算公式为：Fβ = (1 + β²) \* (精确率 \* 召回率) / (β² \* 精确率 + 召回率)。

实践思考题4：回归问题有哪些评价指标，请分别说明他们的含义？

1. 均方误差（MSE）：均方误差是预测值与真实值之间差异的平方的平均值。它衡量了模型预测结果与真实值之间的平均偏差大小，数值越小表示模型的预测越准确。MSE 的计算公式为：MSE = (1/n) \* Σ(yᵢ - ŷ)²，其中 yᵢ 是真实值，ŷ 是预测值，n 是样本数量。

2. 均方根误差（RMSE）：均方根误差是均方误差的平方根，它与原始数据的单位相同。RMSE 是一个常用的回归模型评价指标，可以直观地表示预测结果与真实值之间的平均偏差大小。RMSE 的计算公式为：RMSE = sqrt(MSE)。

3. 平均绝对误差（MAE）：平均绝对误差是预测值与真实值之间差异的绝对值的平均值。它衡量了模型预测结果与真实值之间的平均偏差大小，与均方误差相比，MAE 更加鲁棒，对异常值的敏感性较低。MAE 的计算公式为：MAE = (1/n) \* Σ|yᵢ - ŷ|。

4. 决定系数（R²）：决定系数是用来衡量回归模型拟合数据的程度，表示模型可以解释因变量方差的比例。R² 的取值范围在 0 到 1 之间，值越接近 1 表示模型对数据的拟合越好。R² 的计算公式为：R² = 1 - (SSR / SST)，其中 SSR 是回归平方和，SST 是总平方和。