openGauss AI特性创新实践课



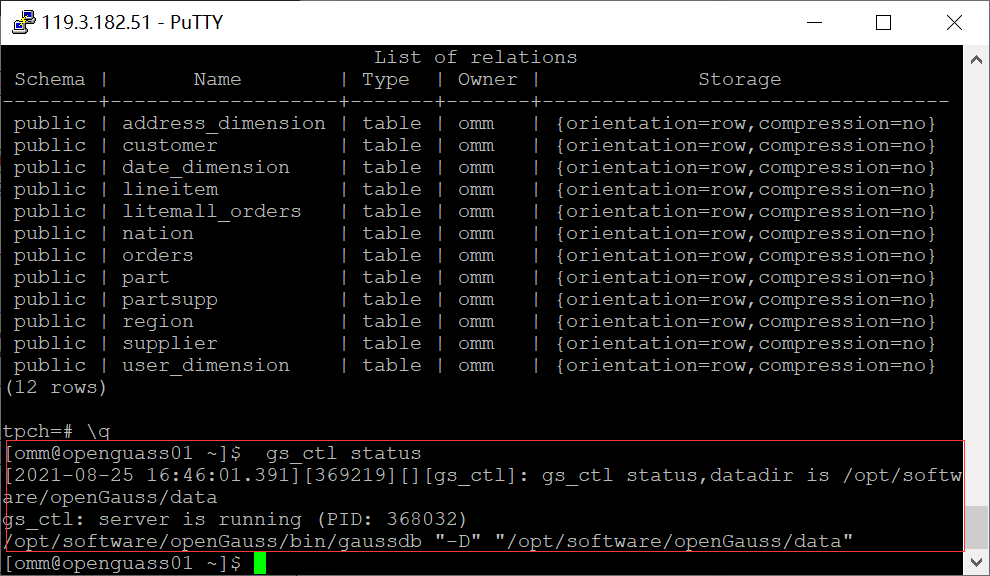
华为技术有限公司

# 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：

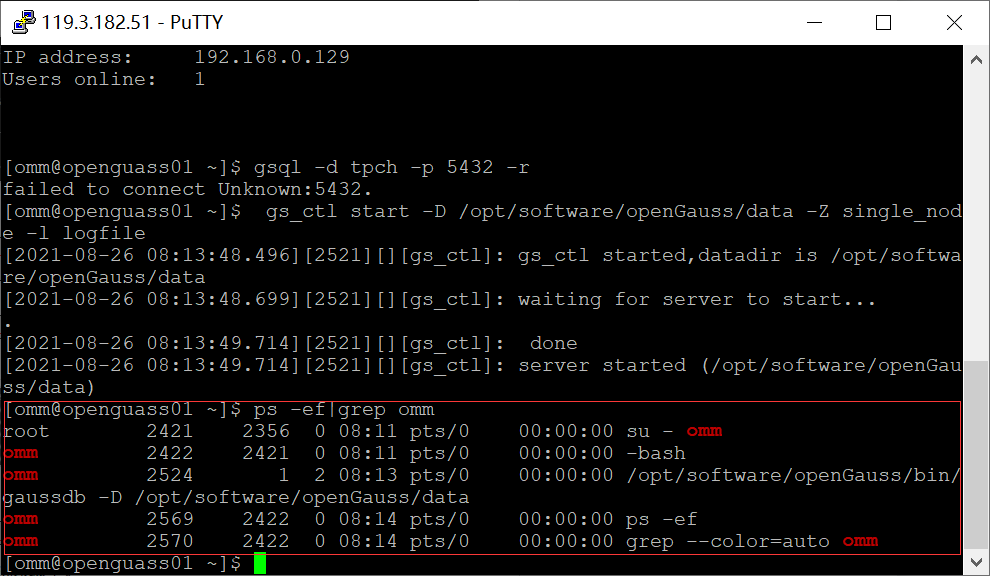
任务一：数据库状态验证

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？

① 便于适应不同的运行平台和依赖环境；

② 可以方便地调整组件或相关编译参数；

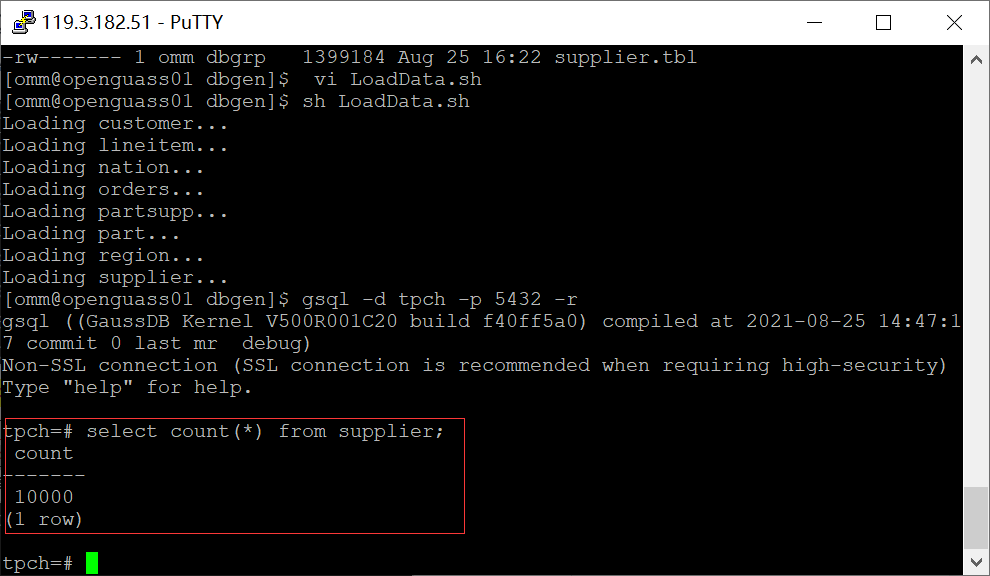
③ 开发人员对源码直接维护，方便运维。

# 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;;

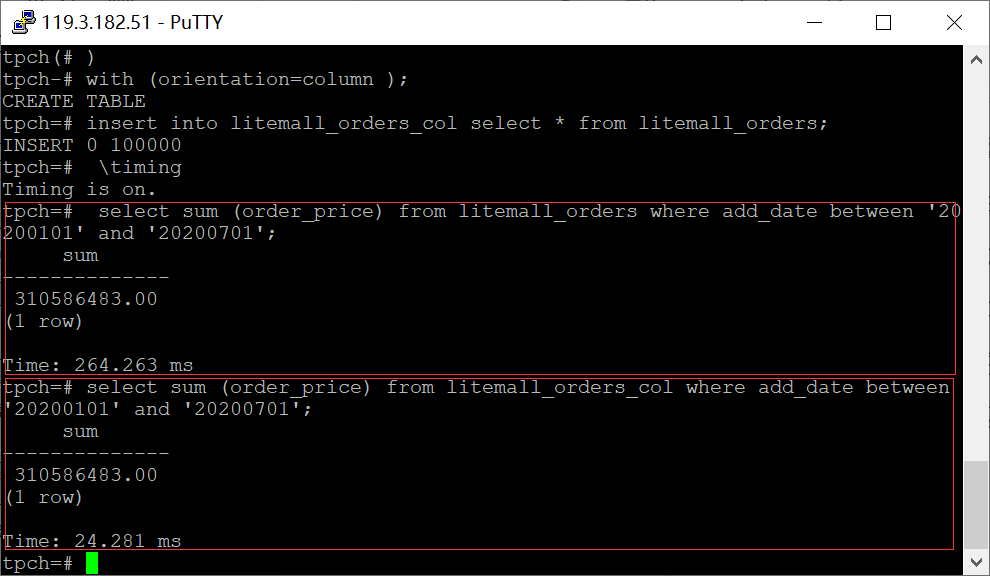


任务二：行存表与列存表执行效率对比

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';

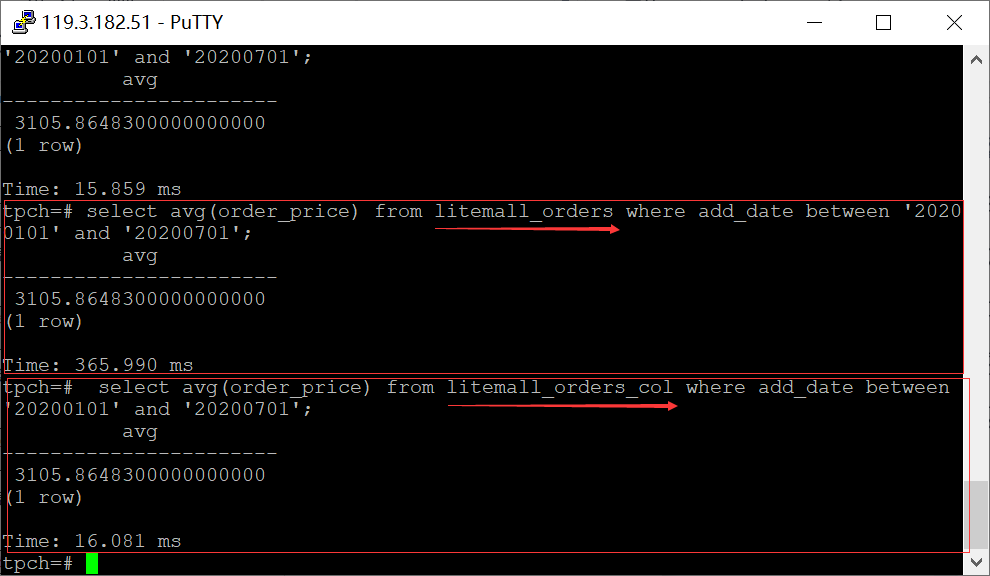
select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';



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';

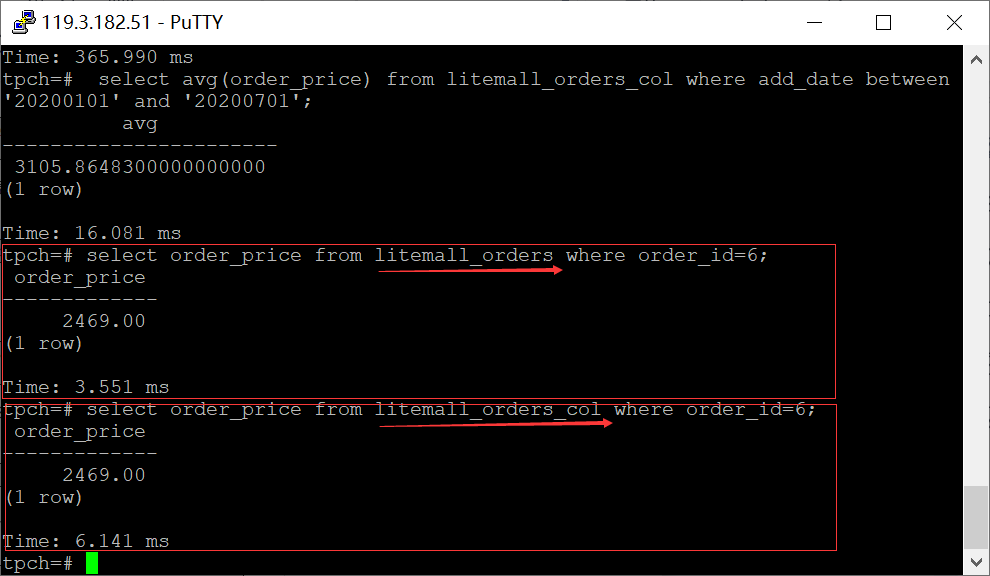
select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';



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;

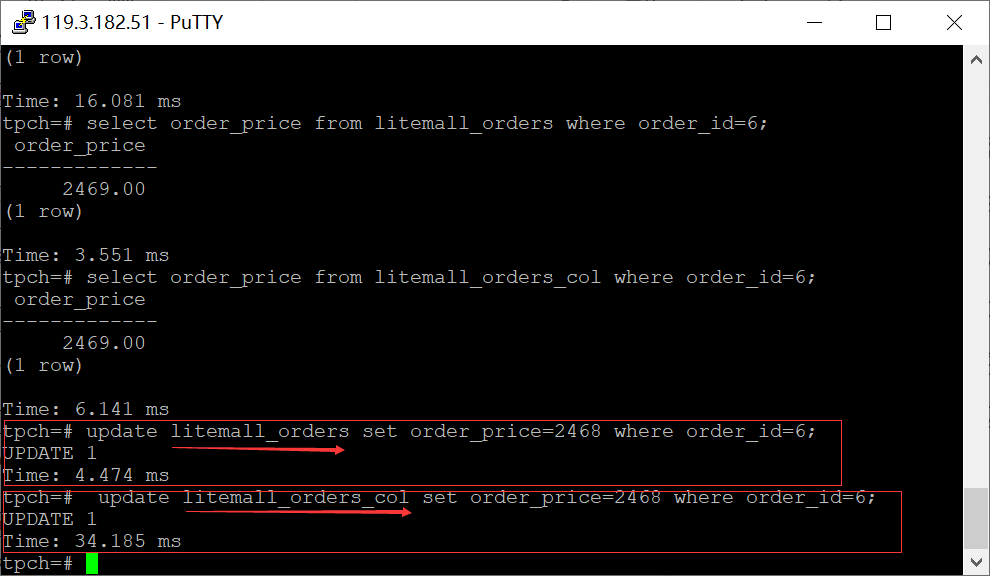
select order\_price from litemall\_orders\_col where order\_id=6;



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;

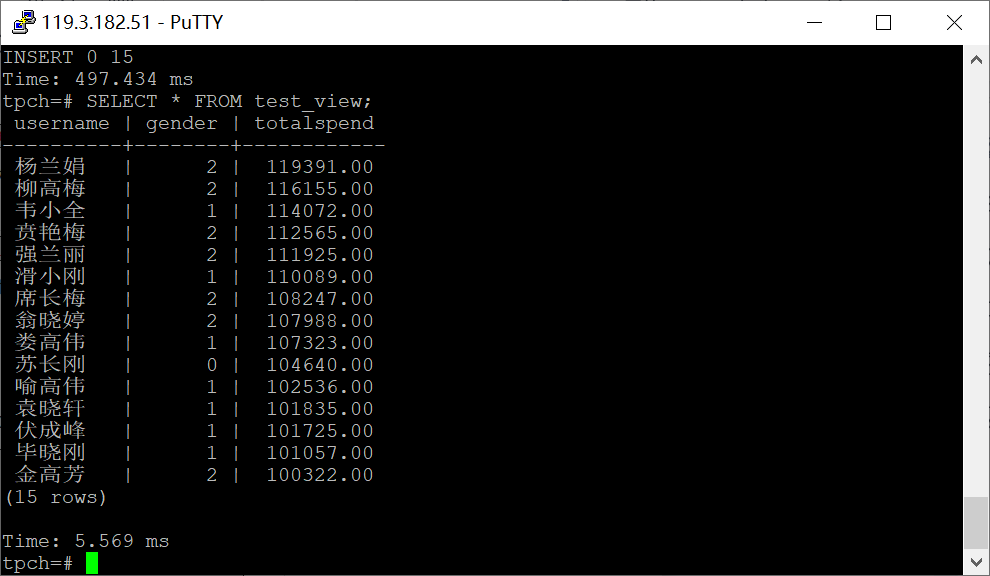
update litemall\_orders\_col set order\_price=2468 where order\_id=6;



任务三：物化视图的使用

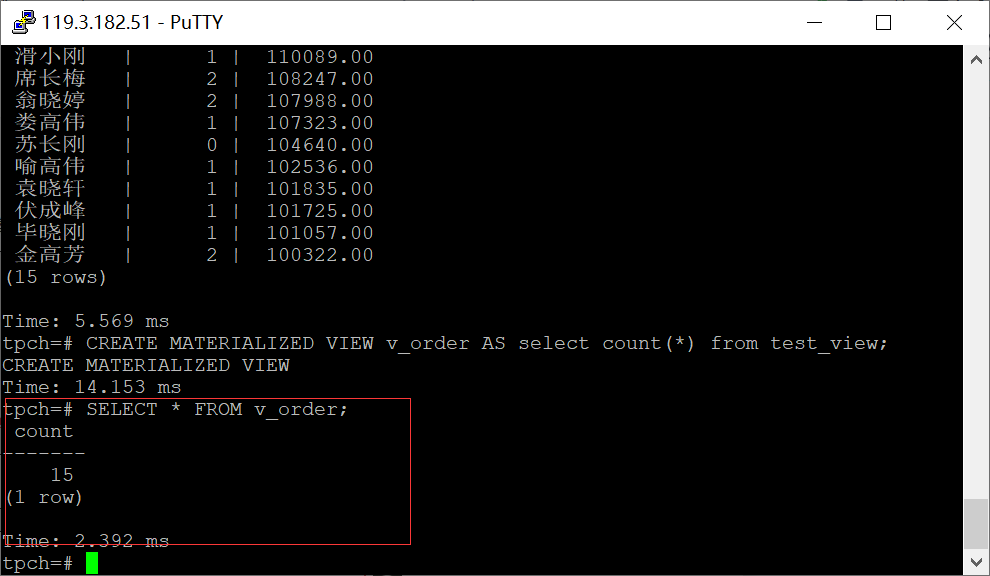
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



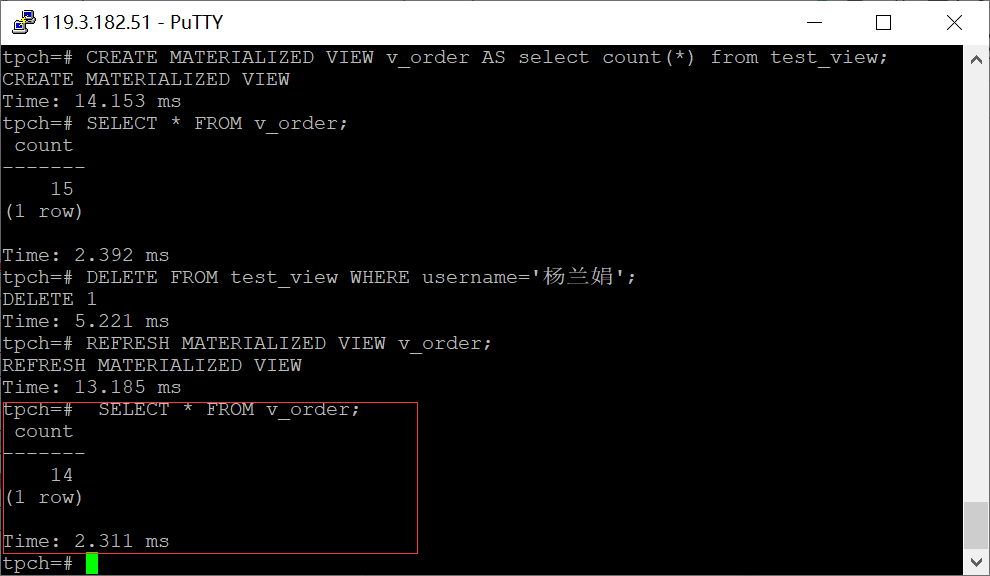
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



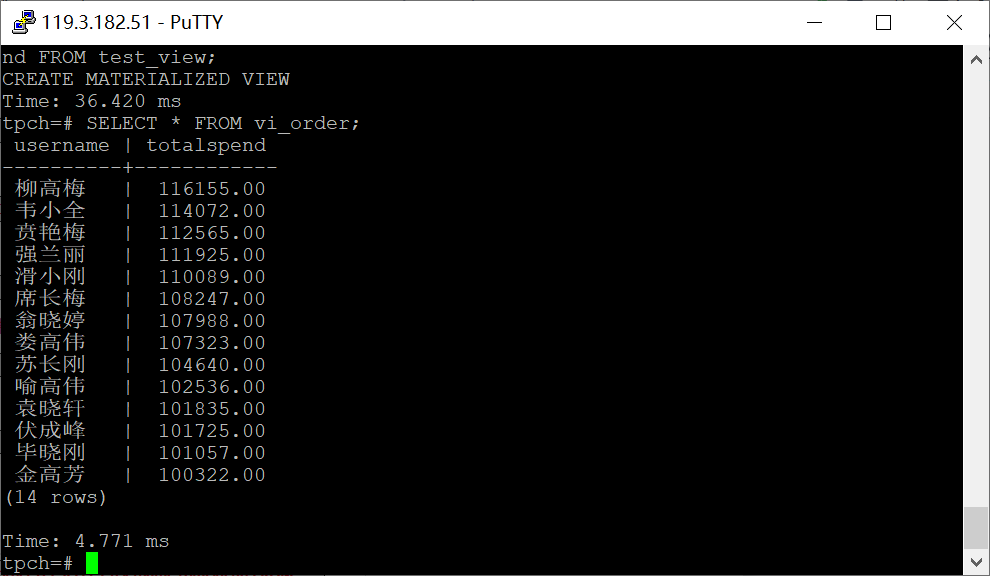
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



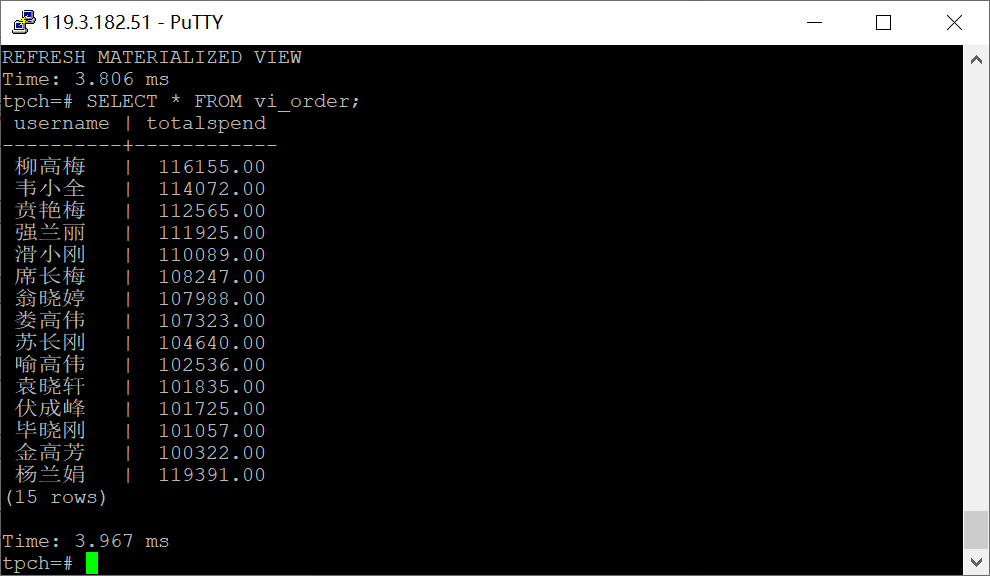
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

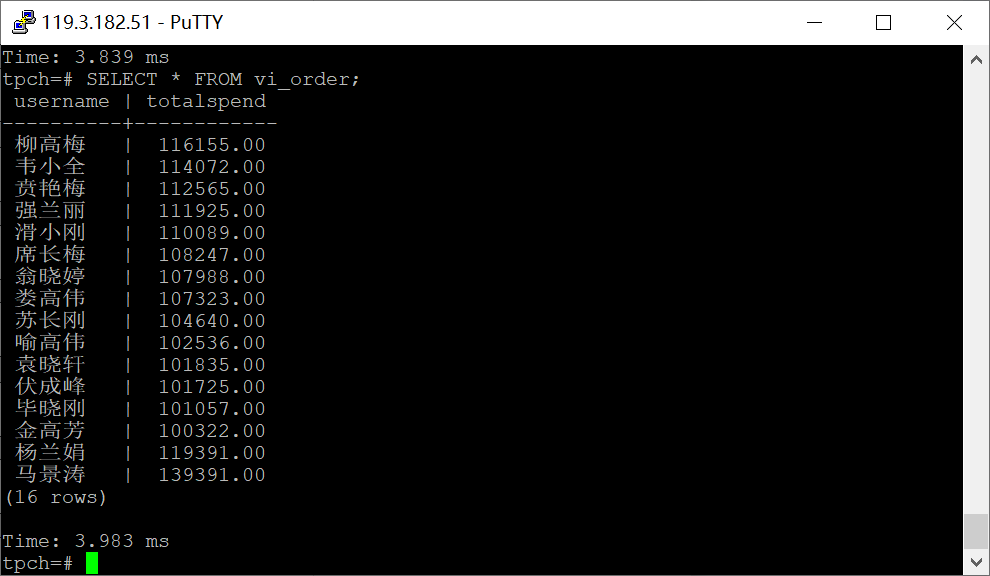
SELECT \* FROM vi\_order;



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;





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？

按行存储时，数据是按照行数据为基础逻辑存储单元进行存储的；按列存储时，数据是按照列为基础的逻辑存储单元进行存储的。

行存储适用：适合随机的增删改查操作；需要在行中选取所有属性的查询操作；需要频繁插入或更新的操作等。

列存储适用：对于整列求值或查询；有频繁聚集需要的表等。

行存储的写入是一次性完成，并且能够保证数据的完整性；而列存储的优势是在读取过程，不会产生冗余数据。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

全量物化视图仅支持对创建好的物化视图做全量更新，而不支持做增量更新。

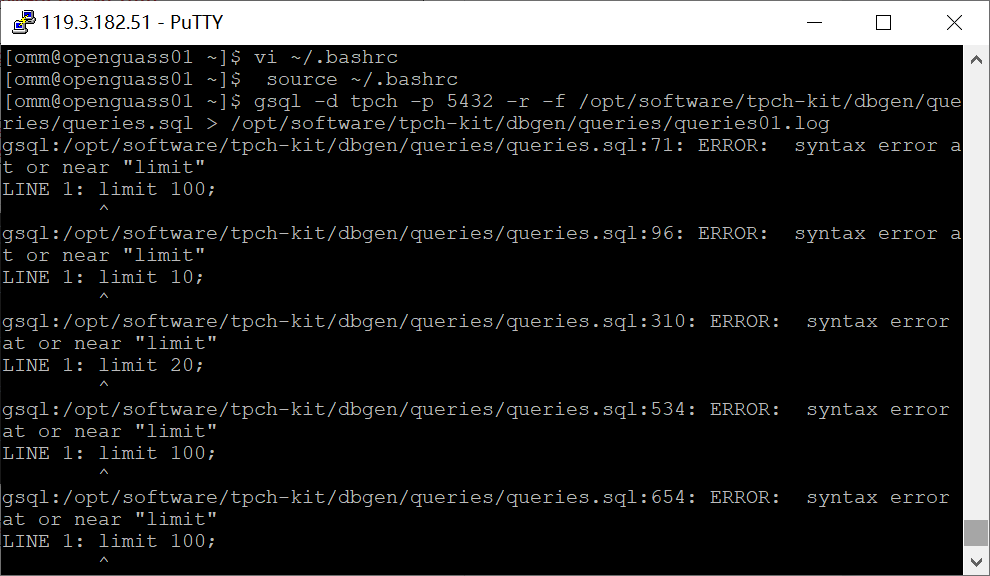
增量物化视图可以对物化视图增量刷新，用户可以修改数据并刷新显示增量后的数据，增量物化视图可以更方便地应用于数据的修改与增量刷新，目前主要支持较小的应用场景。

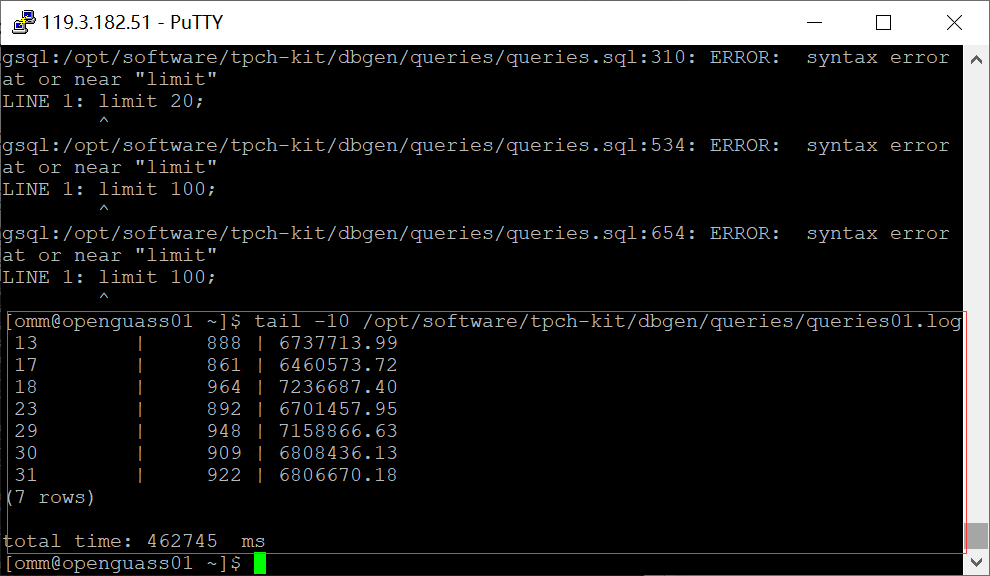
# 关卡三、openGauss的AI4DB特性应用

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：

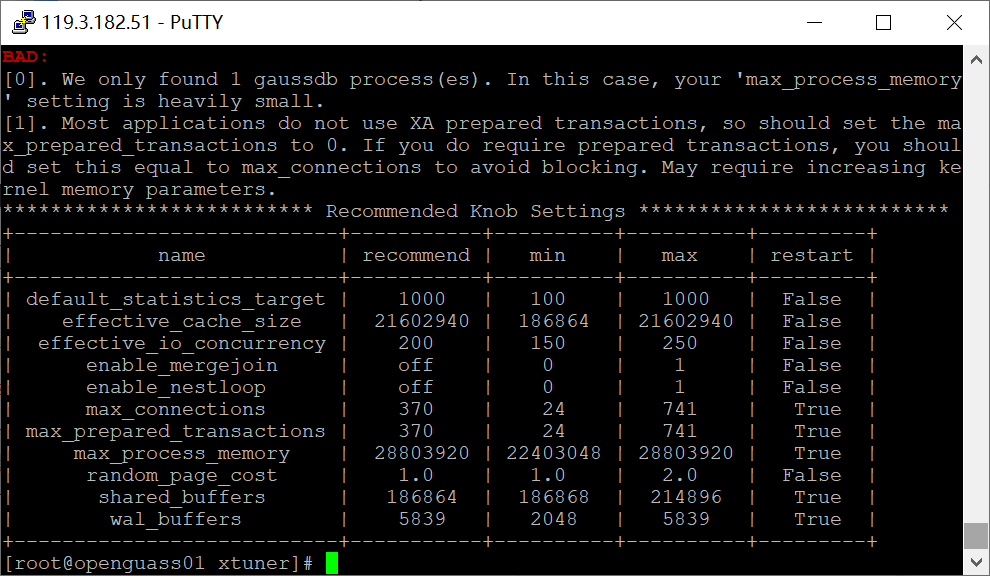
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log





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

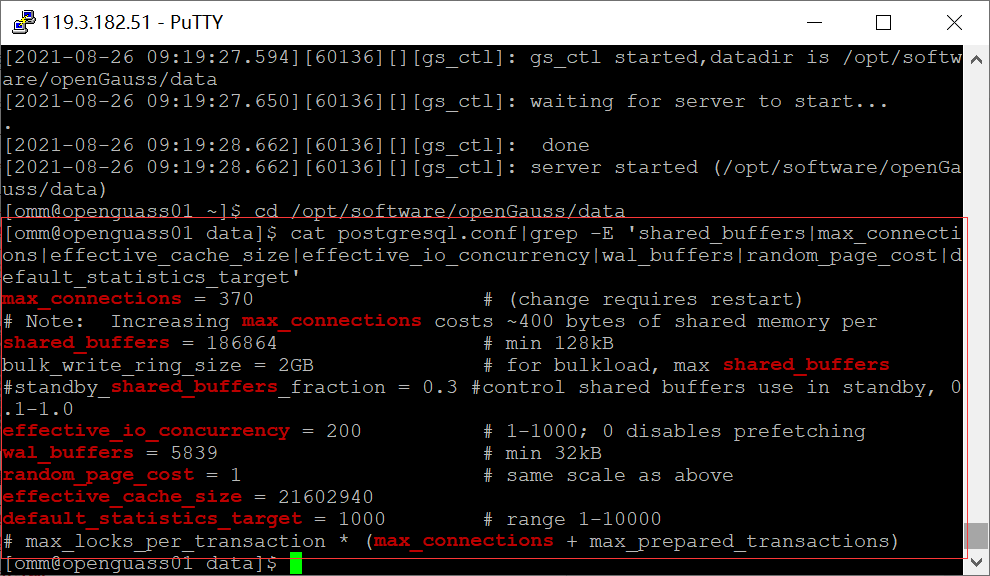
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



3.重启完成后，获取参数值：

cd /opt/software/openGauss/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

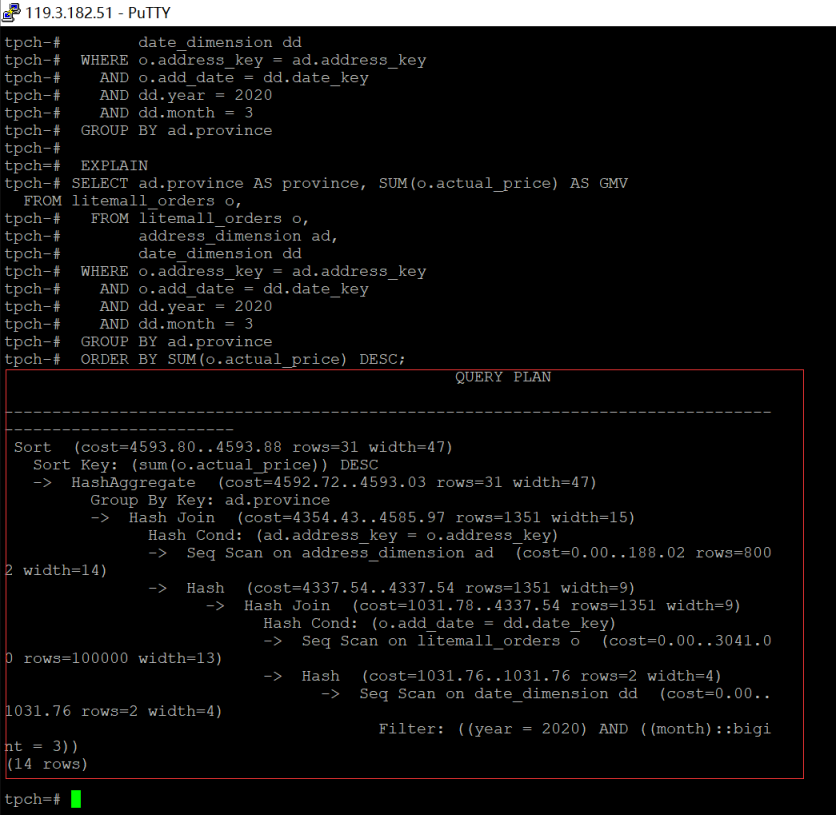
AND o.add\_date = dd.date\_key

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

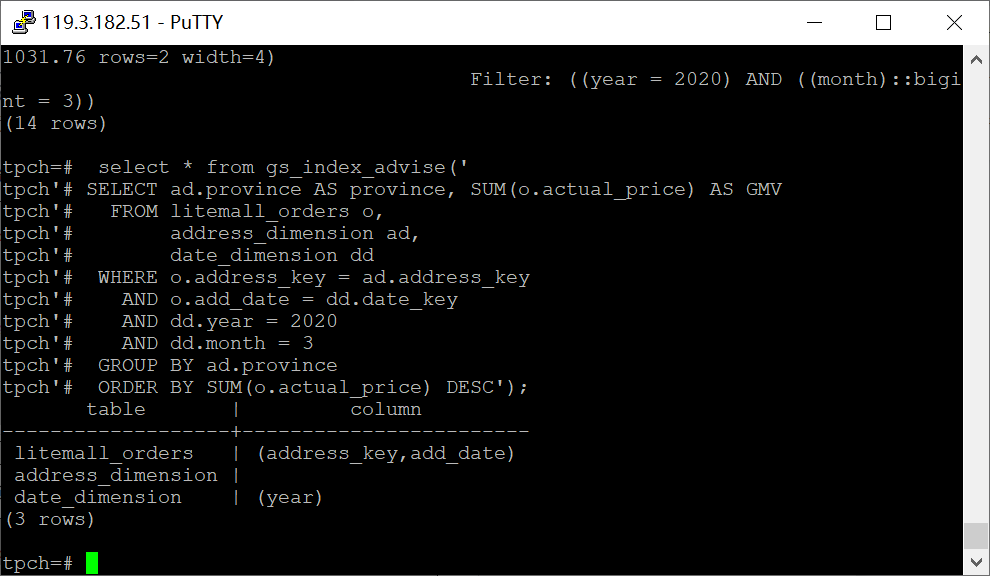
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

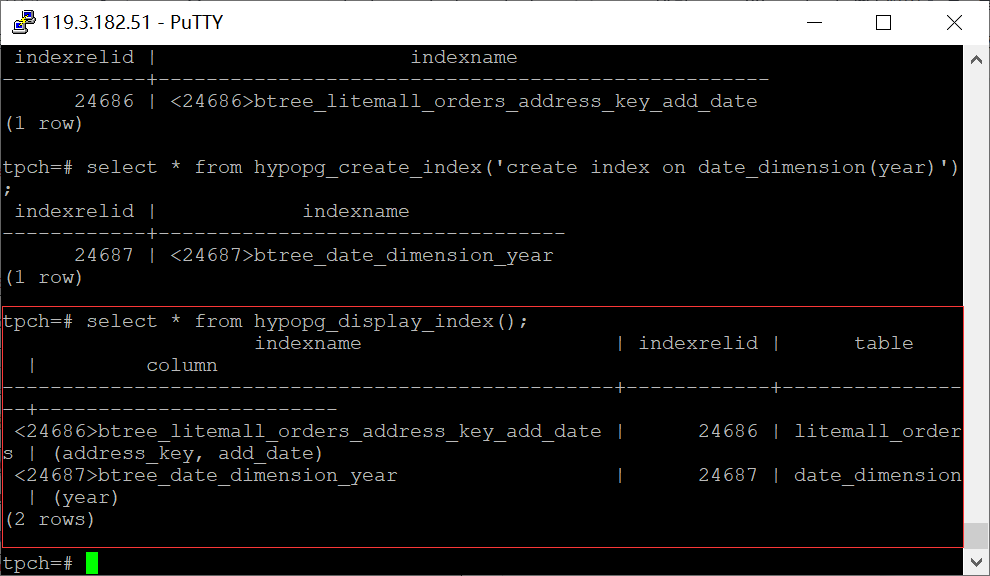
GROUP BY ad.province

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。

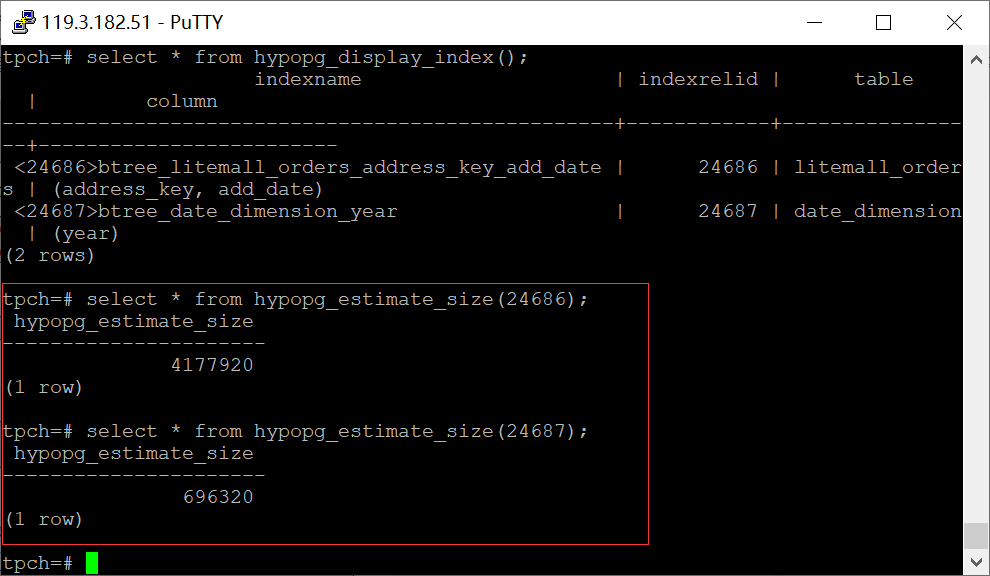
select \* from hypopg\_display\_index();



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(24686);

select \* from hypopg\_estimate\_size(24687);



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

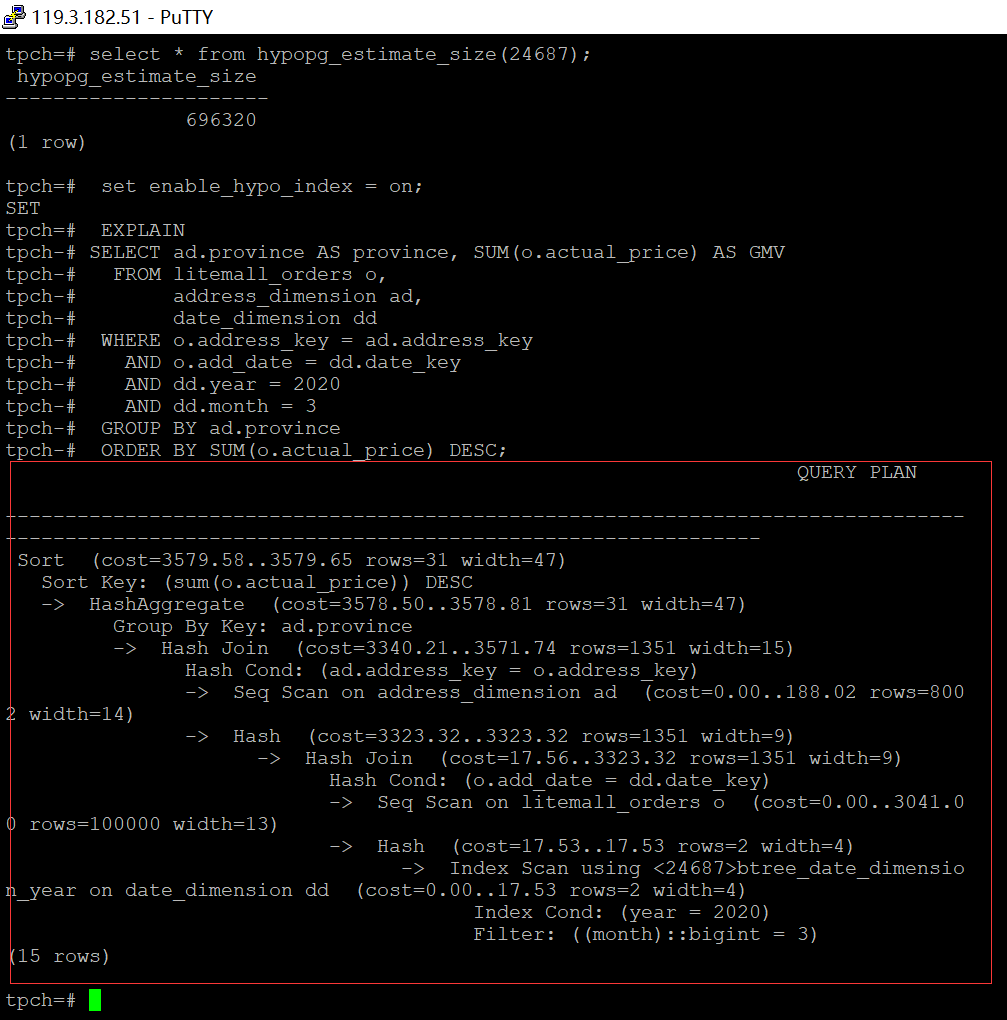
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

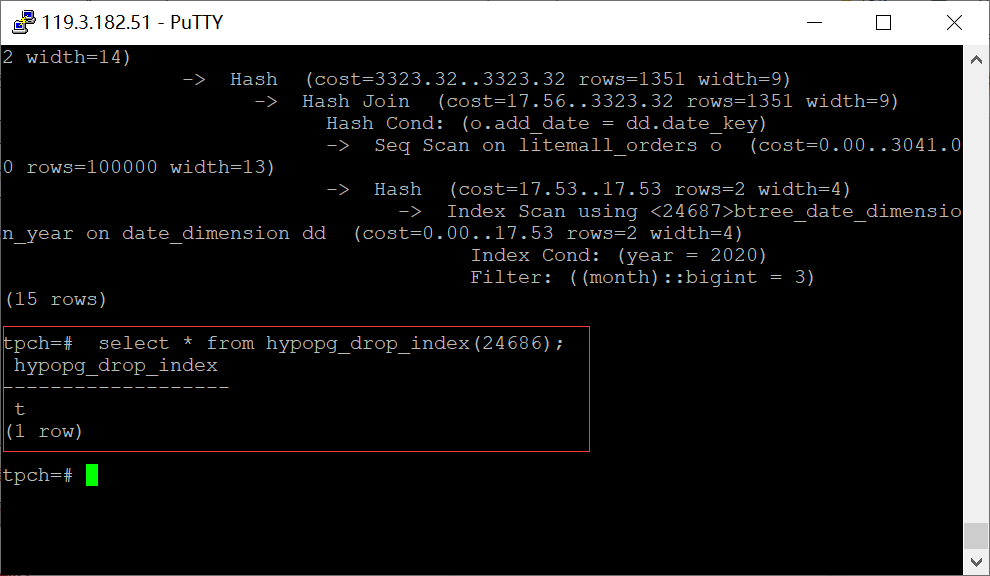
GROUP BY ad.province

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



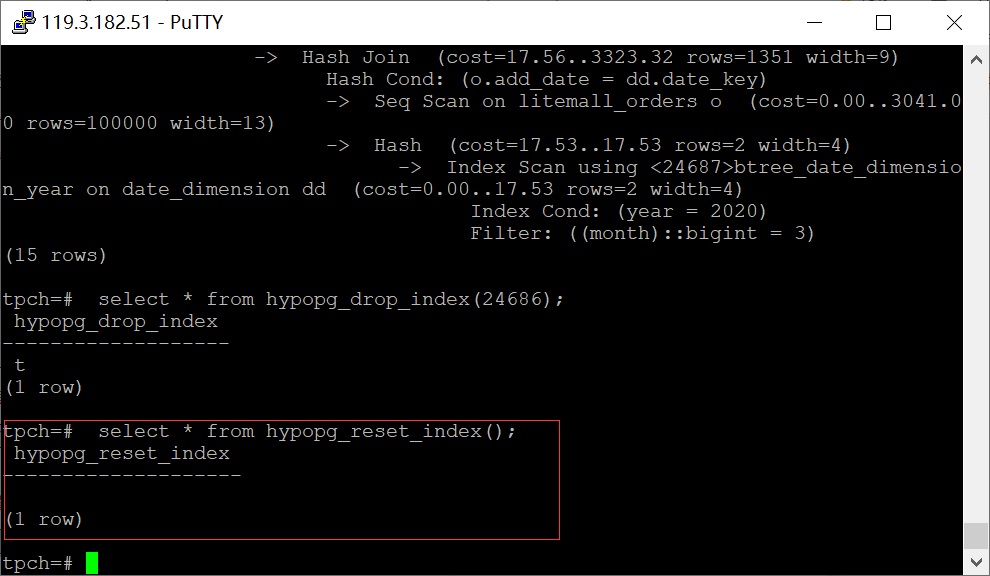
6. 删除某一个索引虚拟列，将执行结果截图。

select \* from hypopg\_drop\_index(24686);



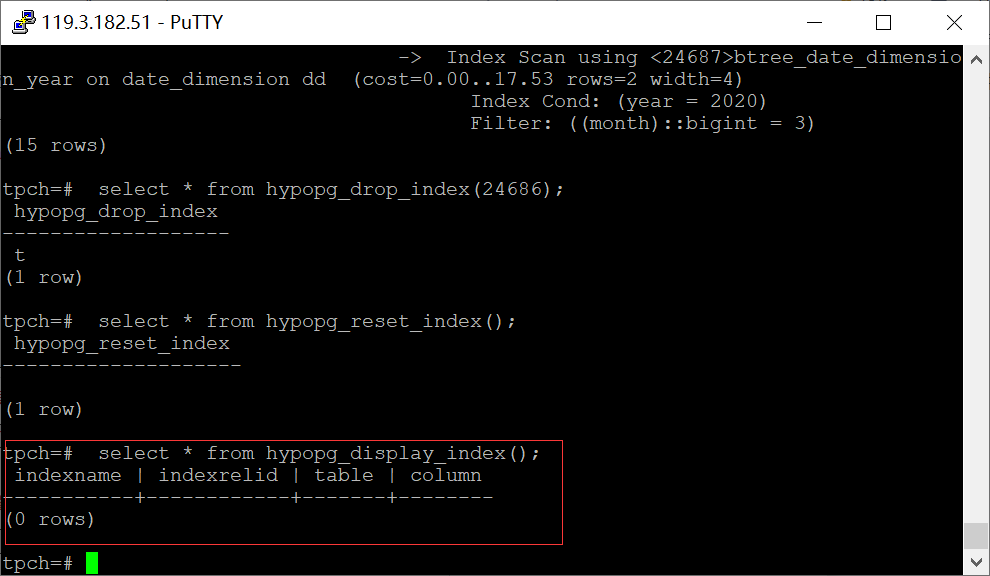
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

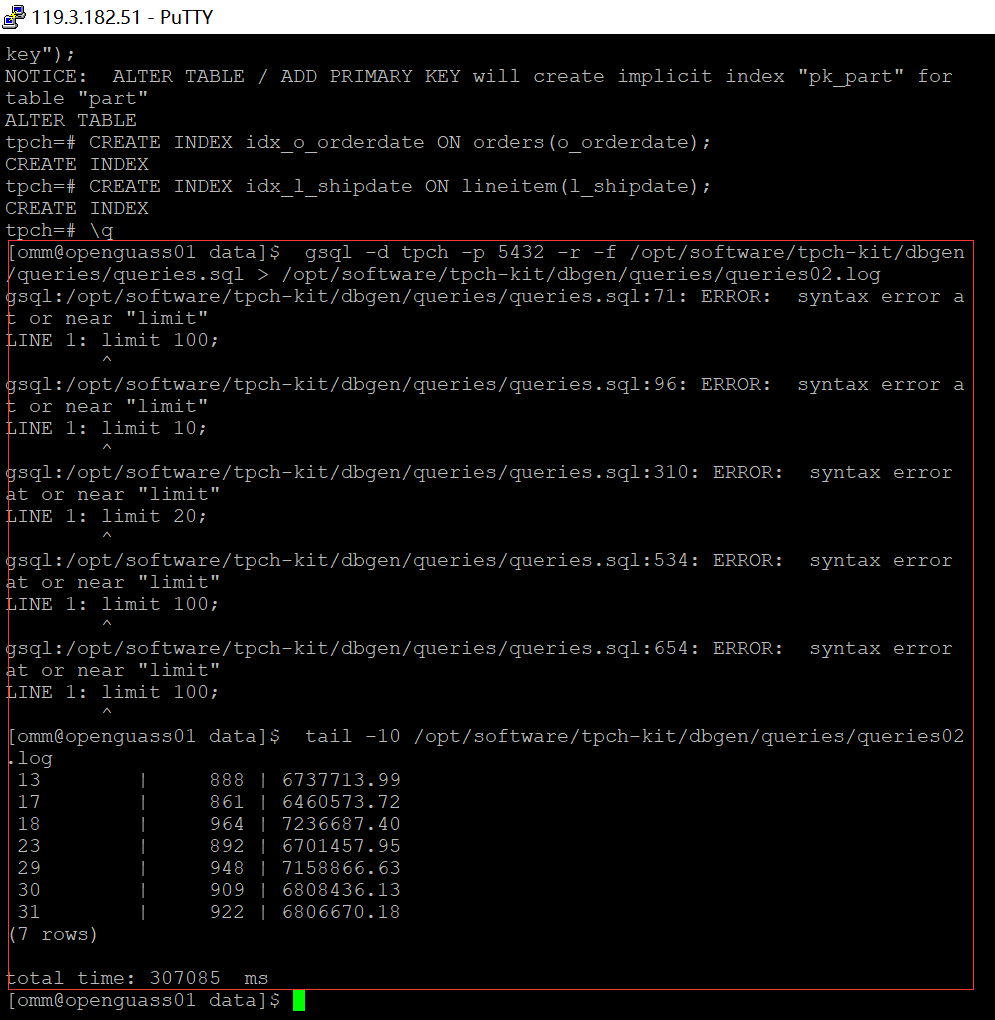
select \* from hypopg\_display\_index();



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？

default\_statistics\_target = 1000

effective\_cache\_size = 21602940

effective\_io\_concurrency = 200

enable\_mergejoin = off

enable\_nestloop = off

max\_connections = 370

max \_prepared\_transactions = 370

max\_process\_memory = 28803920

random\_page\_cost = 1.0

shared\_buffers = 186864

wal\_buffers = 5839

优化以上数据库参数，可提高资源利用率，提升数据库运行性能。

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？

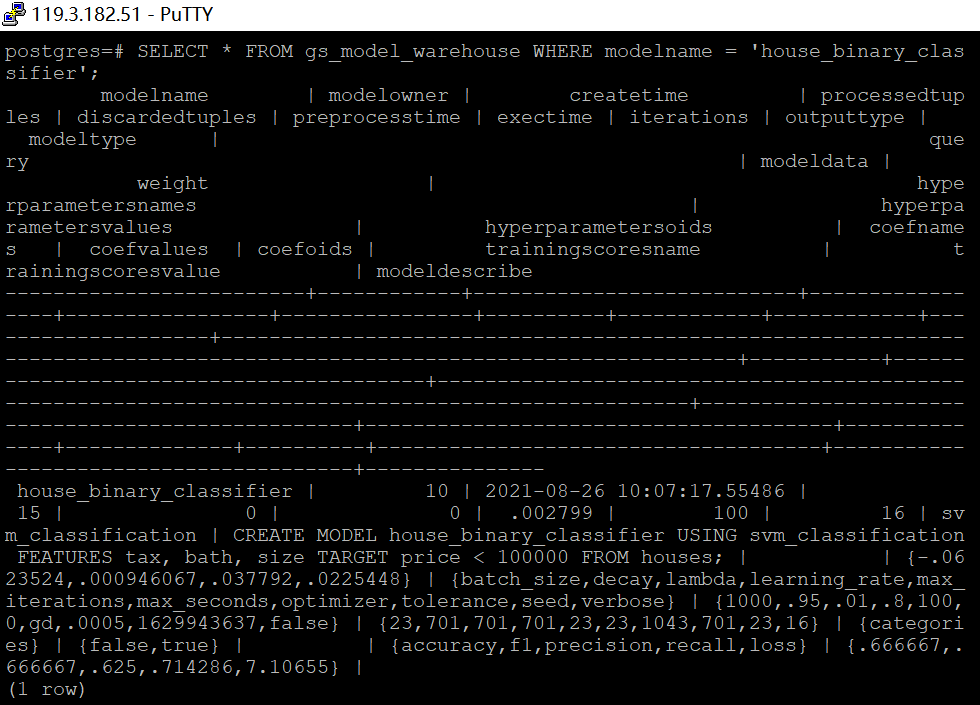
索引对数据库表中一个或多个列的值进行排序，能够加快数据库的查询速度。

其他优化方法：查询语句优化（如使用JOIN来代替子查询，减少系统开销）、表结构优化（如分解表、中间表、增加冗余字段等）、分库分区、配置多核高频CPU+提高内存+高速磁盘等。

# 关卡四、openGauss的DB4AI特性应用

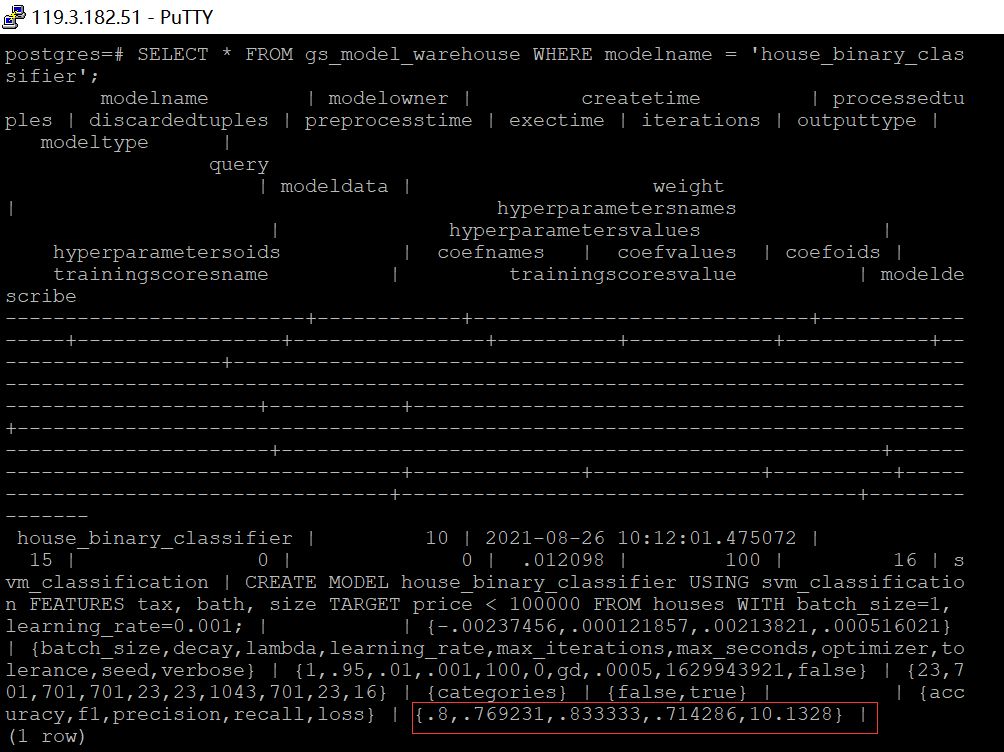
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



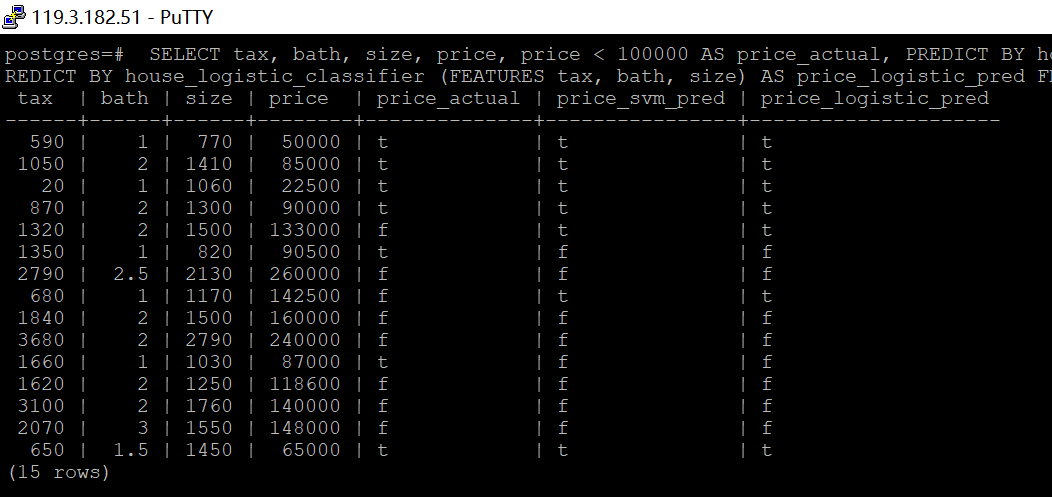
任务二：观察新模型的信息，将执行结果截图。

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？

分类模型和回归模型的输出方式与损失函数都不同。

分类模型输出离散化，回归模型的输出连续化。

（注：逻辑回归是一种分类模型）

实践思考题2：什么是SVM算法？

SVM是支持向量机，用于监督学习中二元线性分类。SVM对学习样本求解的最大边距超平面，其中每一个决策面对应一个线性分类器，SVM找到分类间隔最大的最优决策面，即一个分类器的最优化问题。

从几何意义上看，支持向量指的是观察的样本在 n 维空间中的坐标，SVM 需要找到将样本分成两类的最佳超平面。

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？

混淆矩阵如下所示。

|  |  |  |
| --- | --- | --- |
|  | 预测为正 | 预测为负 |
| 实际为正 | TP | FN |
| 实际为负 | FP | TN |

① 准确率（accuracy）。分类正确的样本数占总样本数的比例。

即A＝（TP＋TN）／（TP+FN+FP+TN）。

② 精确率（precision）、召回率（recall）、F1值。

精确率：正确预测为正的占全部预测为正的比例。即P = TP / (TP + FP)。

召回率：正确预测为正的占全部实际为正的比例。即R = TP / (TP + FN)。

F1值：精确率和召回率的调和平均。即 2/F1 = 1/P + 1/R。

③ ROC和AUC。

TPR = TP / (TP + FN)

FPR = FP / (FP + TN)

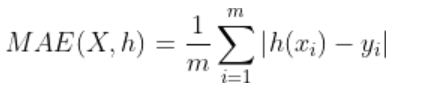
ROC曲线使用FPR作为横坐标，TPR作为纵坐标；ROC曲线越靠近左上角，模型的准确性就越高。

AUC为ROC曲线下与坐标轴围成的面积，取值范围在0.5和1之间。AUC越接近1.0，检测方法真实性越高；等于0.5时，则真实性最低。

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？

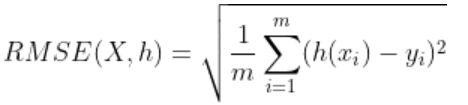
① MAE（平均绝对误差）

是绝对误差的平均值，可以更好地反映预测值误差的实际情况。



② RMSE（均方根误差）

衡量观测值与真实值之间的偏差，常用来作为机器学习模型预测结果衡量的标准。



③ MSE（均方误差）

MSE是真实值与预测值的差值的平方然后求和平均，常被用作线性回归的损失函数。

