openGauss AI特性创新实践课



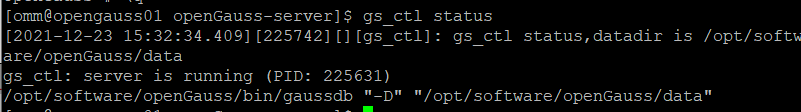
华为技术有限公司

# 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：

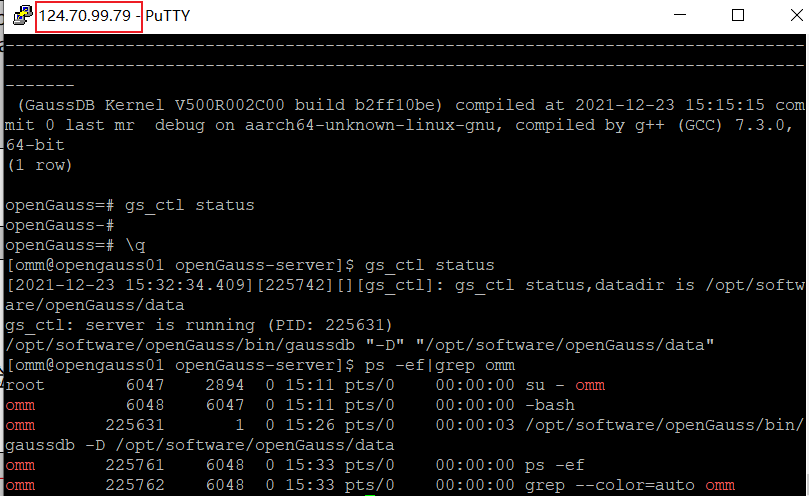
任务一：数据库状态验证

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？

通过源码编译可以更加理解数据库，特别需要修改编译文件时

源码编译一般会安装最新版本的数据库

mysql编译安装可以定制字符集，并且目录清晰

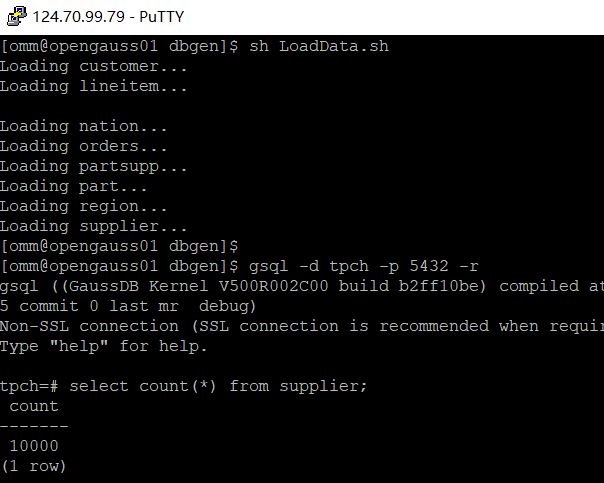
可以满足当前场景

# 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：

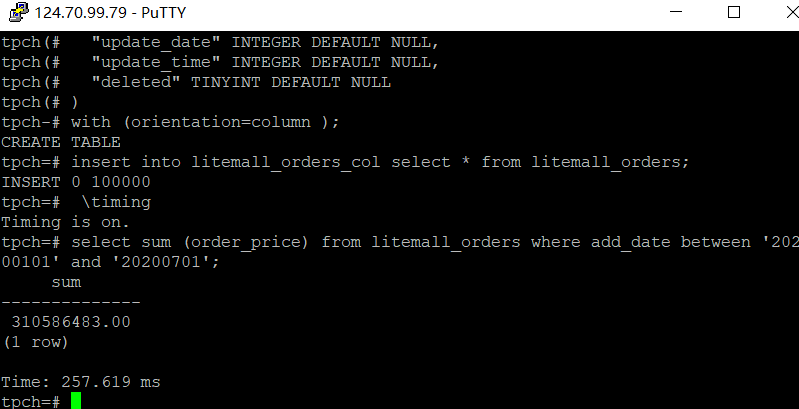
select count(\*) from supplier;;



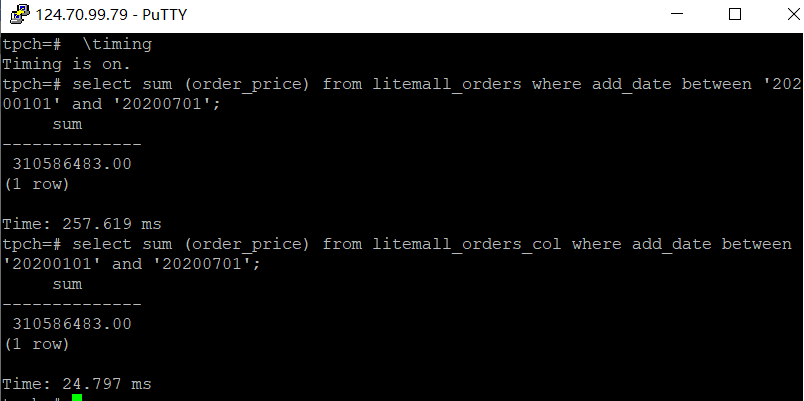
任务二：行存表与列存表执行效率对比

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';

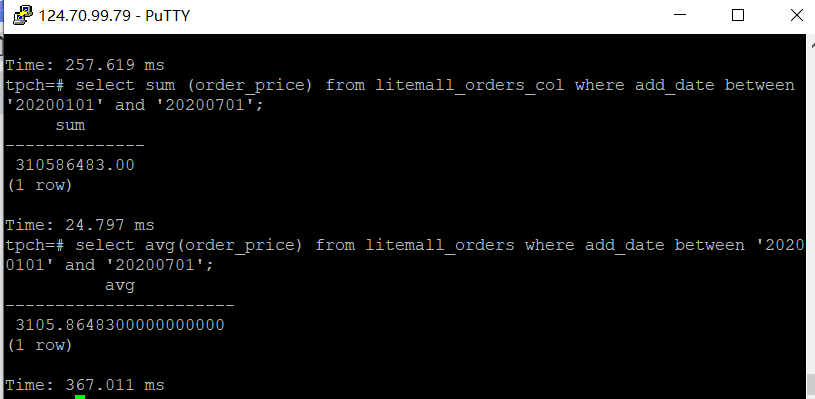


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

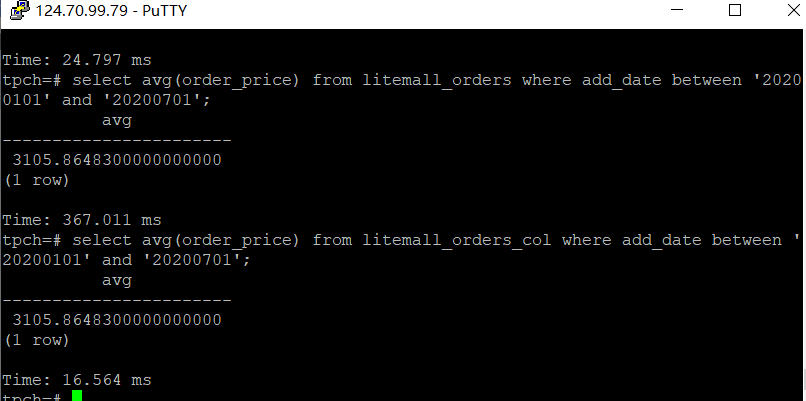


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

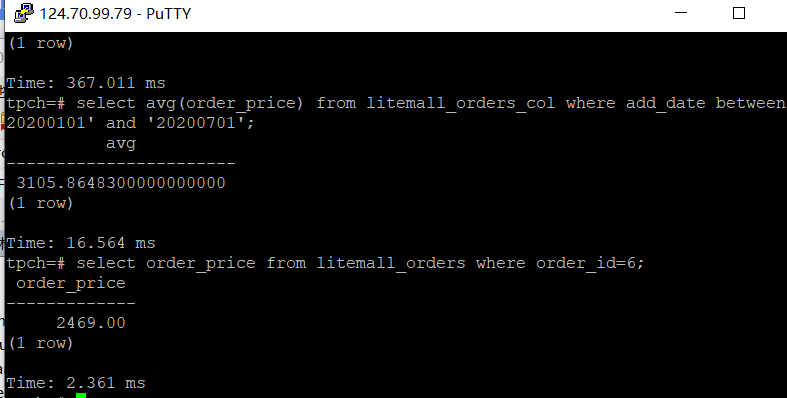


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

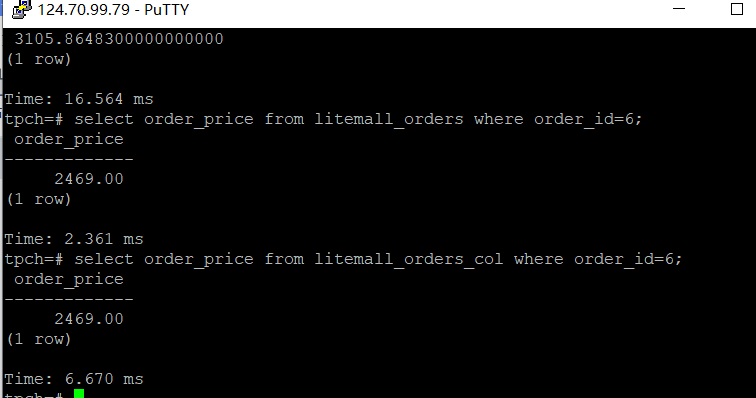


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

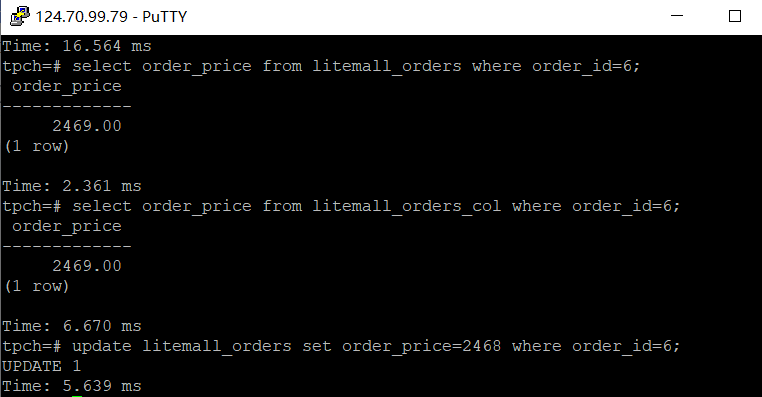


select order\_price from litemall\_orders\_col where order\_id=6;

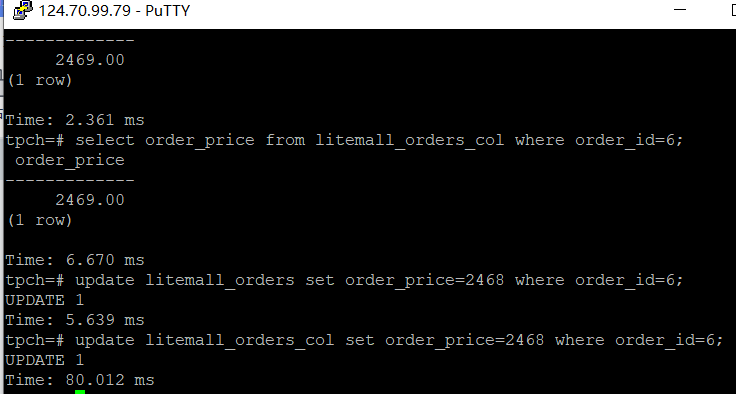


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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



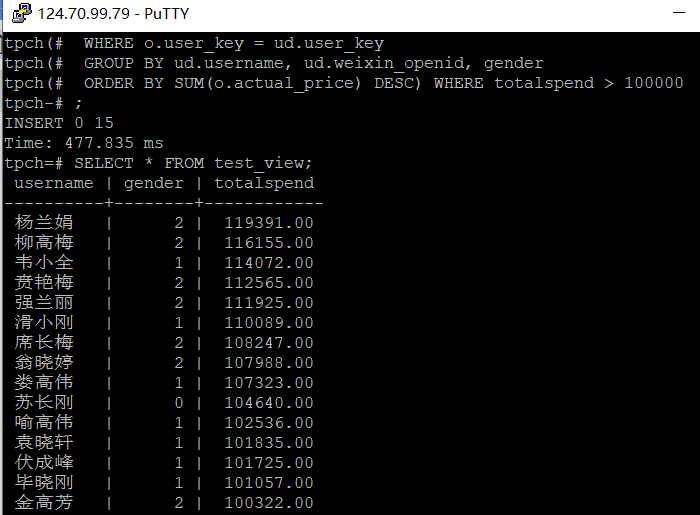
update litemall\_orders\_col set order\_price=2468 where order\_id=6;



任务三：物化视图的使用

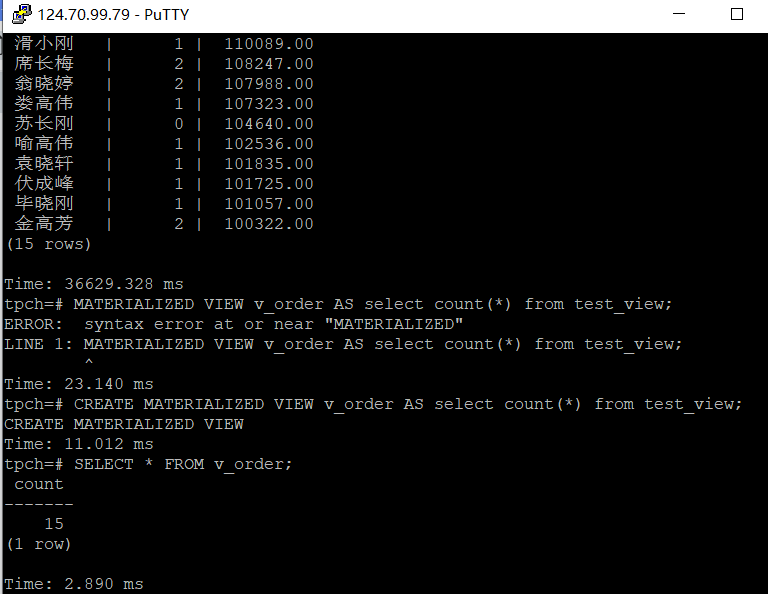
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



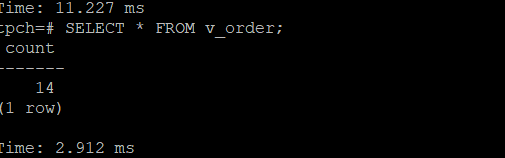
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



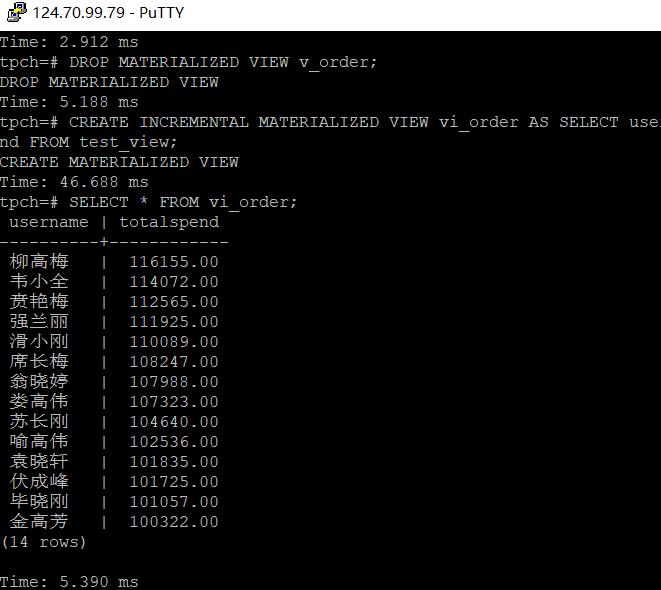
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



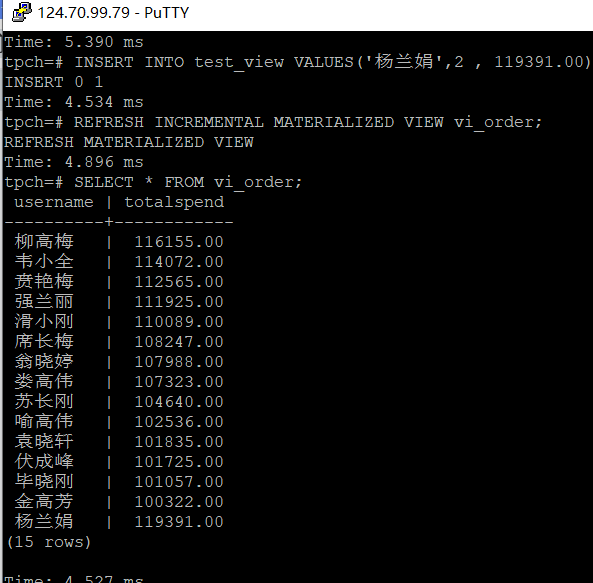
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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？

1. 列存表由于需要把一行记录拆分成单列保存，写入次数明显比行存表多（意味着磁头调度次数多，而磁头调度是需要时间的，一般在1ms~10ms)，再加上磁头需要在盘片上移动和定位花费的时间，实际时间消耗会更大。
2. 行存表在数据写入和数据修改时效率更高
3. 数据读取时，如果只需要其中几列数据的情况，就会存在冗余列，出于缩短处理时间的考量，消除冗余列的过程通常是在内存中进行的；由于列存储的每一列数据类型是同质的，不存在二义性问题，所以列存储的解析过程更有利于分析大数据。

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：每进行一次插入、更新等操作都会全局更新所有数据，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。

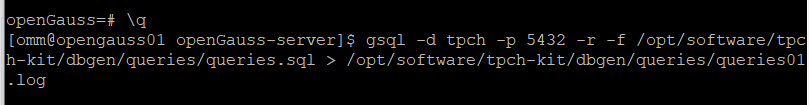
增量物化视图：需要用户手动执行语句完成对物化视图在一段时间内的增量数据进行刷新。

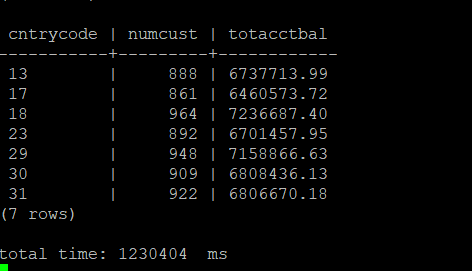
# 关卡三、openGauss的AI4DB特性应用

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：

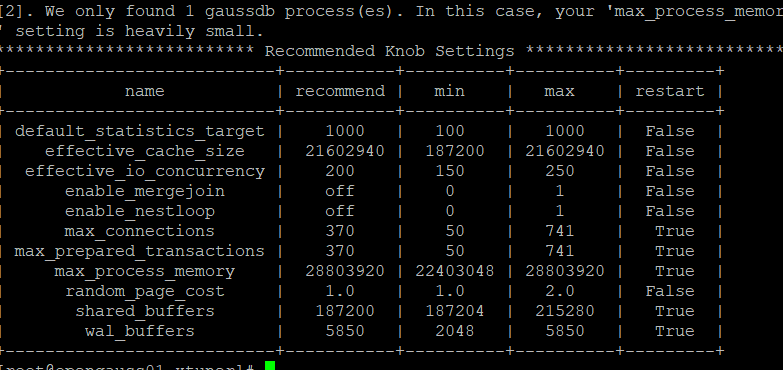
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log





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

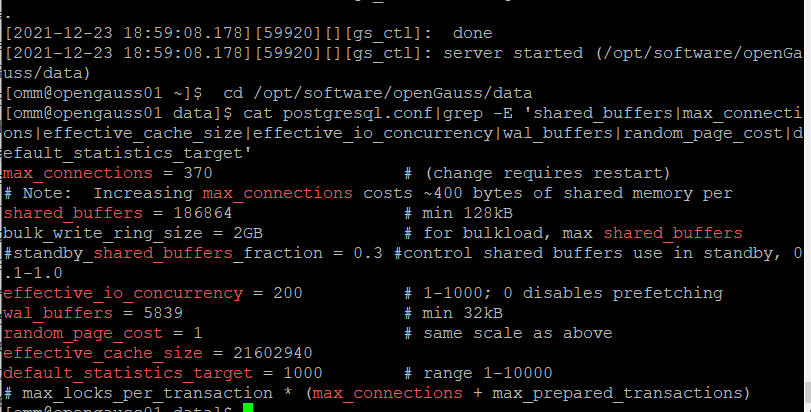
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

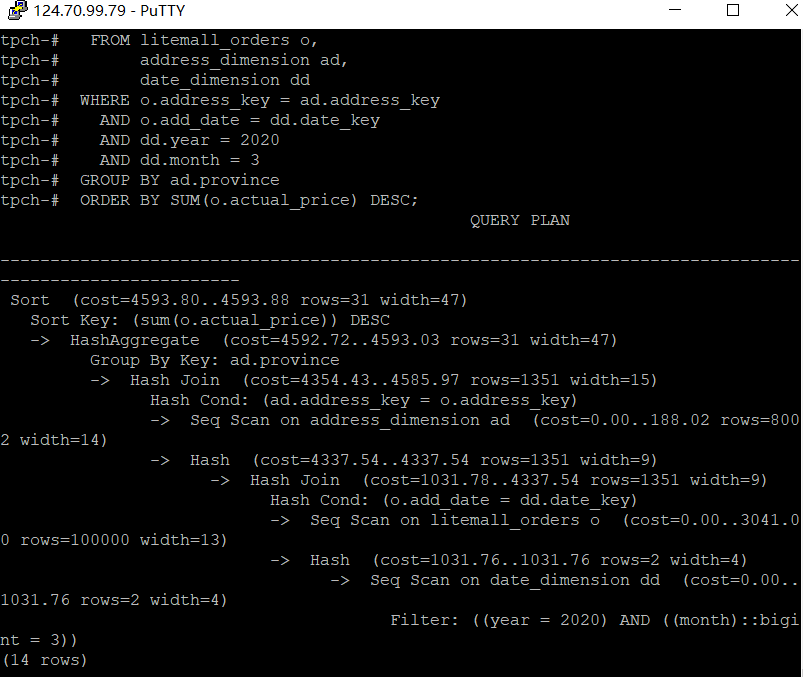
AND o.add\_date = dd.date\_key

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

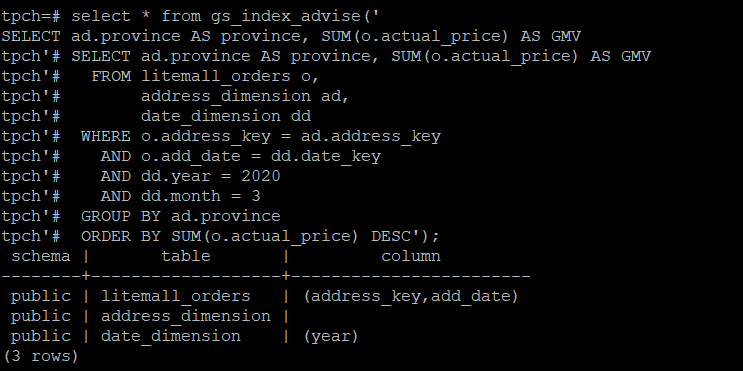
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

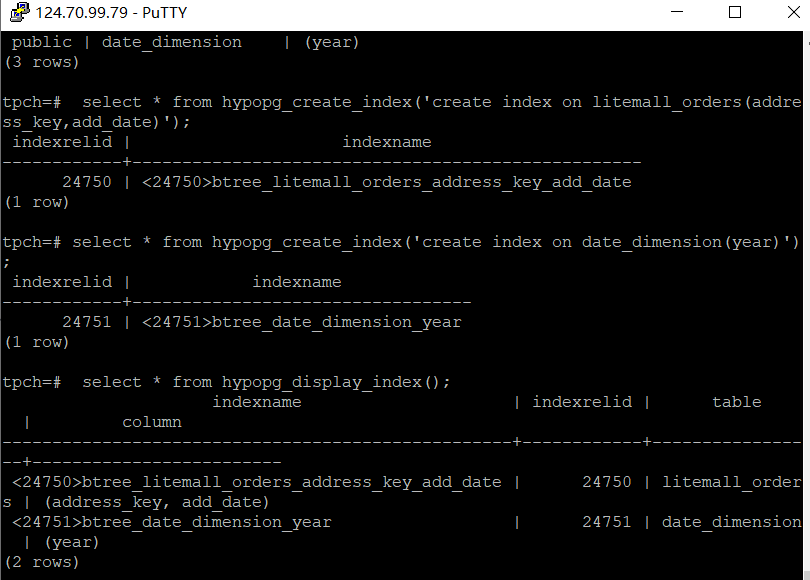
GROUP BY ad.province

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。

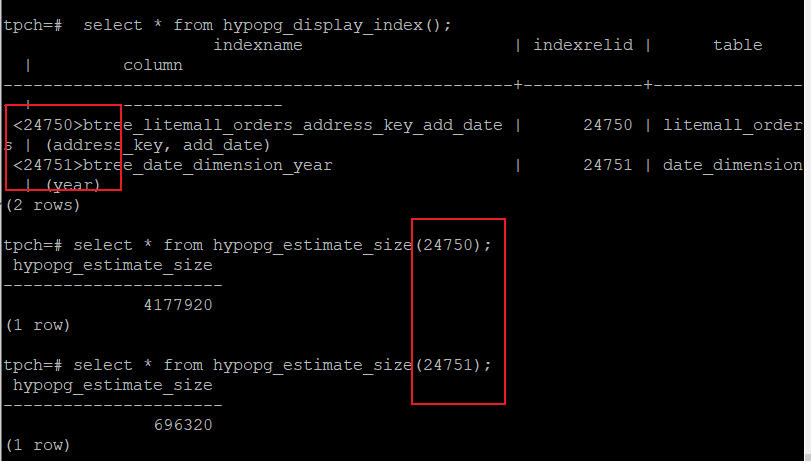
select \* from hypopg\_display\_index();



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(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

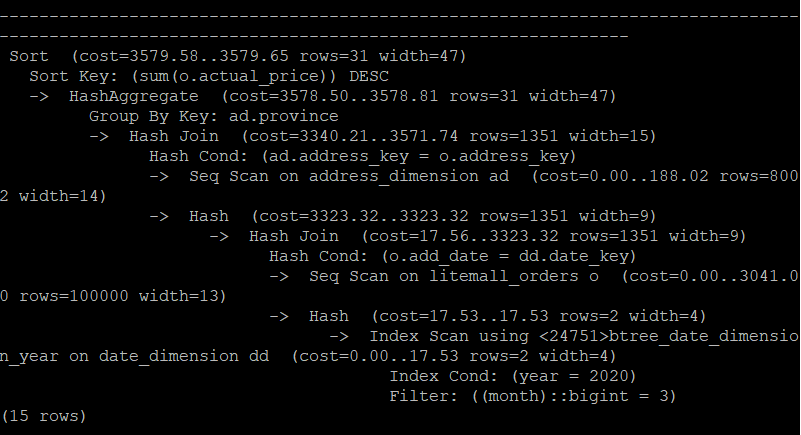
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

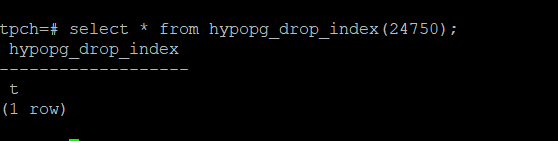
GROUP BY ad.province

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



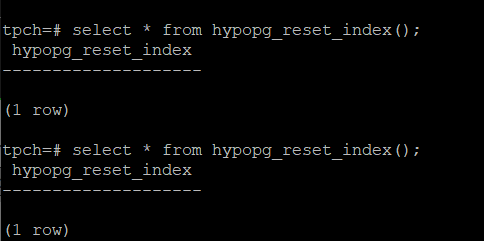
6. 删除某一个索引虚拟列，将执行结果截图。

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



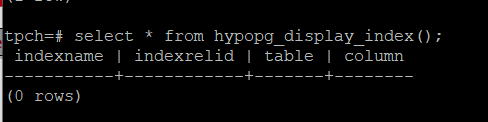
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

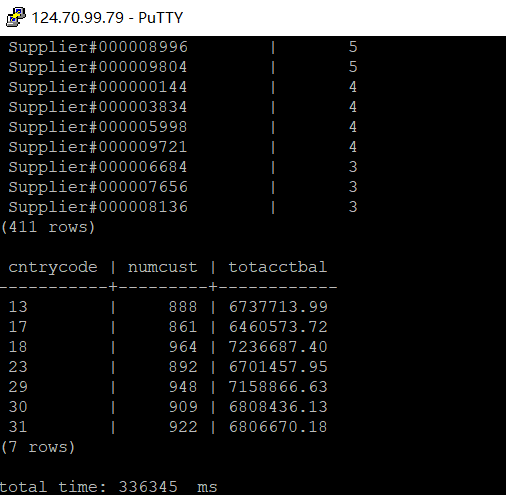
select \* from hypopg\_display\_index();



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max\_connections、shared\_buffers、effective\_cache\_size 、effective\_io\_concurrency、wal\_buffers、random\_page\_cost、default\_statistics\_target进行了优化。

在生产环境中，将shared\_buffers设置为较大的值通常可以提供非常好的性能，但应当时刻注意找到平衡点；max\_connections是允许客户端连接的最大数目；effective\_cache\_size更高的数值会使得索引扫描更可能被使用，更低的数值会使得顺序扫描更可能被使用；effective\_io\_concurrency是执行节点为bitmap heap scan时，预读的块数；wal\_buffers设置为一个较高的值可以提供更好的性能；random\_page\_cost是离散扫描的成本因子；default\_statistics\_target是进行analyze的时候，参考的生成的列的柱状图的大小，可以理解为采样颗粒度。

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？

SQL好处

①创建唯一性索引可以保证数据库表中每一行数据的唯一性；

②加快检索速度；

③可以通过外键加速表和表之间的连接；

④在使用分组和排序子句进行数据检索时，能减少查询中分组和排序的时间；

⑤使用索引在查询的过程中使用优化隐藏器，提高系统的性能。

优化数据库

①优化数据库表结构设计

②数据量很大的收，可以将表拆分成多张表

③事务。当多个用户同时使用相同的数据源时，它可以利用锁定数据库的方法来为用户提供一种安全的访问方式，这样可以保证用户的操作不被其它的用户所干扰

④数据库参数配置优化

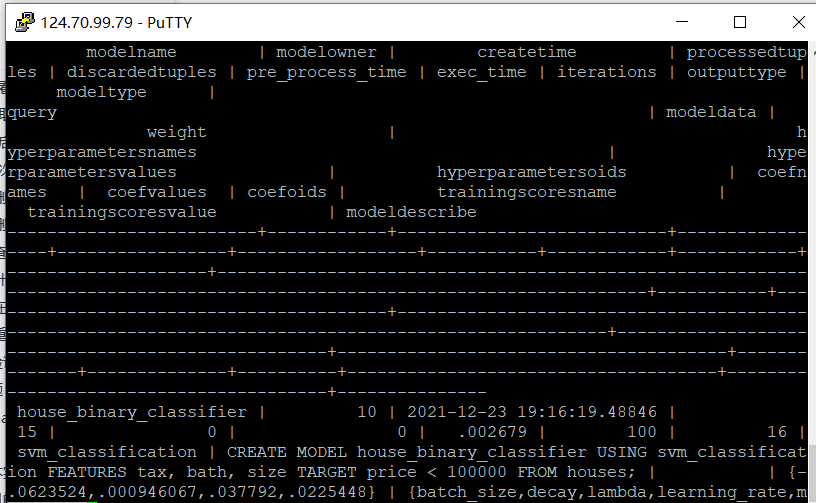
⑤主从复制，读写分离

⑥优化的查询语句。，在建有索引的字段上尽量不要使用函数进行操作在搜索字符型字段时，减少使用LIKE关键字和通配符等

# 关卡四、openGauss的DB4AI特性应用

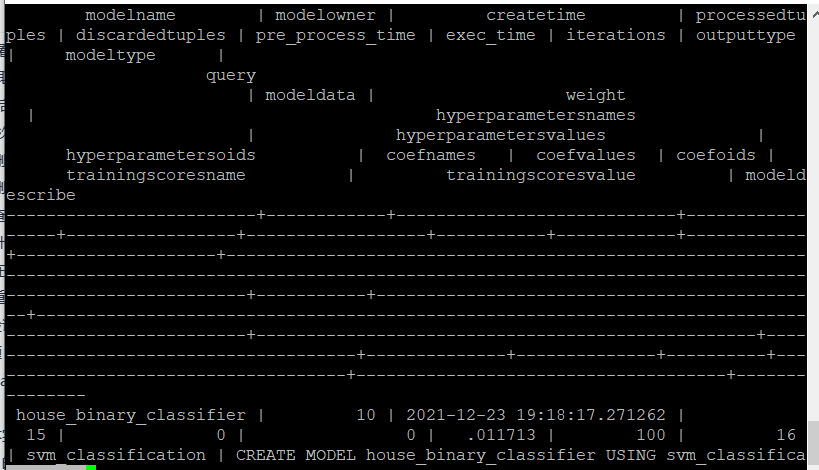
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



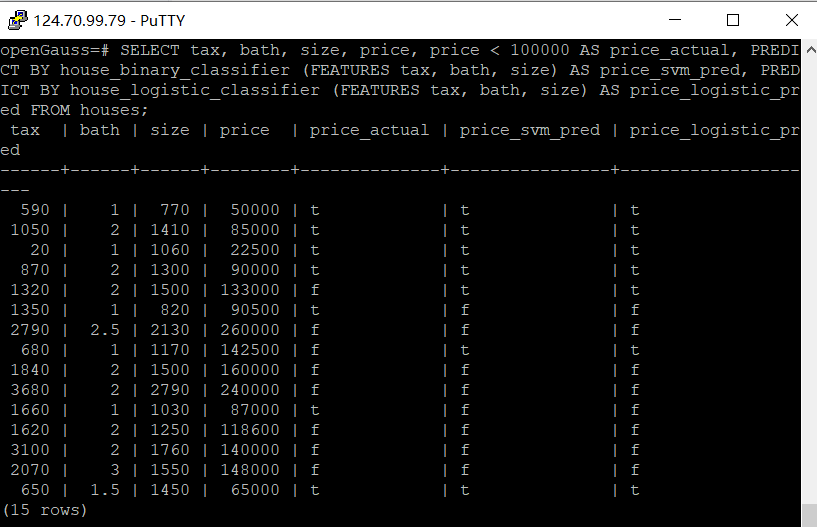
任务二：观察新模型的信息，将执行结果截图。

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：结果为离散变量，给定一个模型，根据训练集推断出他所对应的类别，通常会使用softmax，

回归模型：结果为连续变量，输出为具体的值

实践思考题2：什么是SVM算法？

SVM(Support Vector Machine)算法，即支持向量机算法，以其简单的理论构造了复杂的算法，又以其简单的用法实现了复杂的问题。SVM算法属于有监督学习算法是基于统计学习理论的一种机器学习方法，通过寻求结构化风险最小来提高学习机泛化能力，实现经验风险和置信范围的最小化，从而达到在统计样本量较少的情况下，亦能获得良好统计规律的目的。它常用来对小样本、非线性及高维数据进行模式识别、分类以及回归分析，并能够取得很好的效果。

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？

主要是基于混淆矩阵的

①准确率(Accuracy)：分类正确的样本数与总样本数之比；

②精确率(Precision)：（TP真阳性）/（TP真阳性+FP假阳性）

③召回率(Recall)：（TP真阳性）/（TP真阳性+FN假阴性）

④综合评价指标F-Measure（又称为F-Score）：Precision和Recall的加权调和平均；

⑤ROC曲线：接收者操作特征（receiver operating characteristic），ROC曲线上每个点反映着对同一信号刺激的感受性横轴，负正类率（FPR，特异度）、纵轴：真正类率（TPR，灵敏度）；

⑥AUC(Area Under Curve)：ROC曲线下的面积，介于0.1和1之间。AUC作为数值可以直观的评价分类器的好坏，值越大越好。

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？

①MSE（Mean Squared Error）均方误差：观测值与真值偏差的平方和与观测次数的比值，线性回归中最常用的损失函数，线性回归过程中尽量让该损失函数最小。模型之间的对比也可以用它来比较。可以评价数据的变化程度，MSE的值越小，说明预测模型描述实验数据具有更好的精确度；

②RMSE（Root Mean Squared Error）均方根误差；

③MAE （Mean absolute Error）平均绝对误差：预测值与真实值之间平均相差多大；

④R-Squared：分母理解为原始数据的离散程度，分子为预测数据和原始数据的误差，二者相除可以消除原始数据离散程度的影响；

⑤交叉验证（Cross-Validation）