openGauss AI特性创新实践课



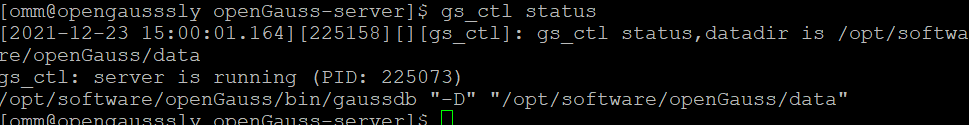
华为技术有限公司

# 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：

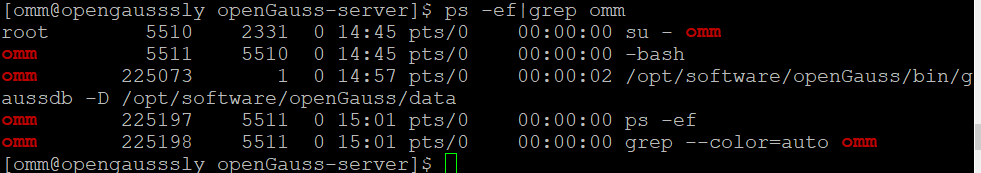
任务一：数据库状态验证

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）



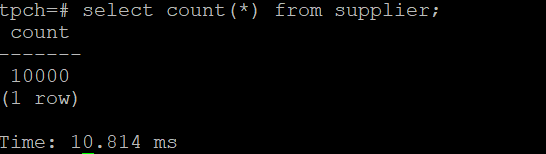
实验思考题：为什么需要通过源码编译，安装数据库？

# 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;;

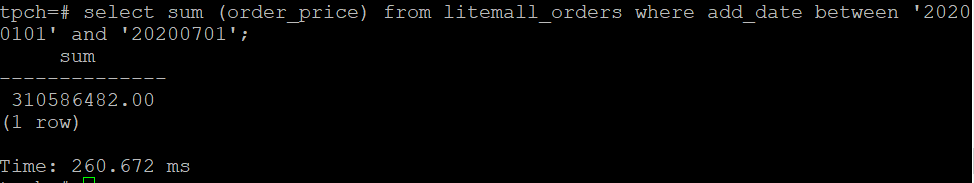


查询到supplier表的总行数为10000行。

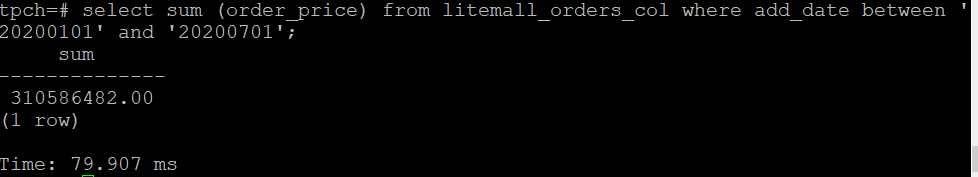
任务二：行存表与列存表执行效率对比

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';

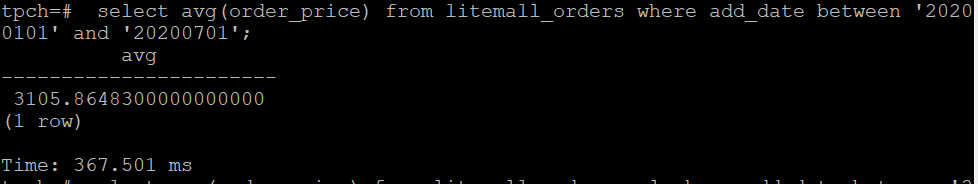


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

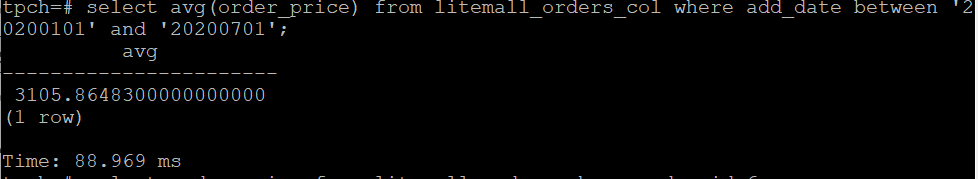


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

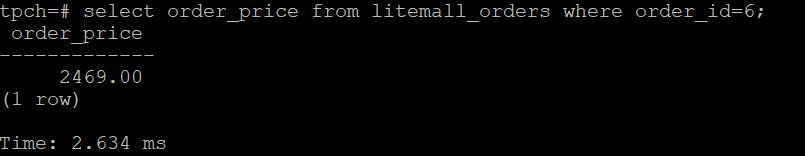


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

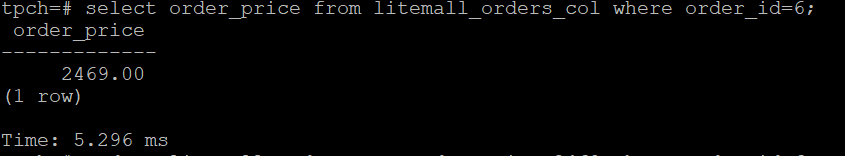


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

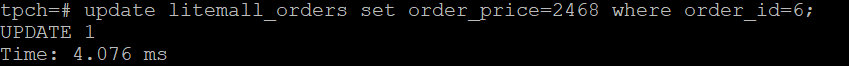


select order\_price from litemall\_orders\_col where order\_id=6;

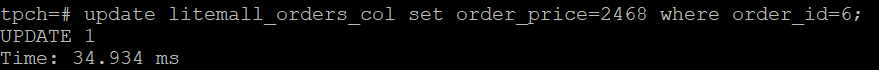


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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

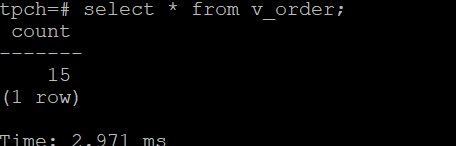
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



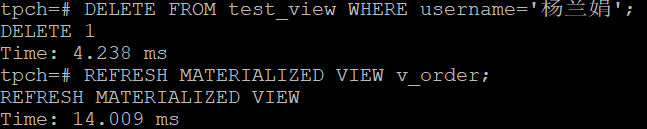
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

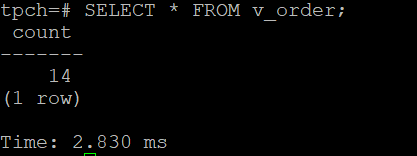
SELECT \* FROM v\_order;



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。

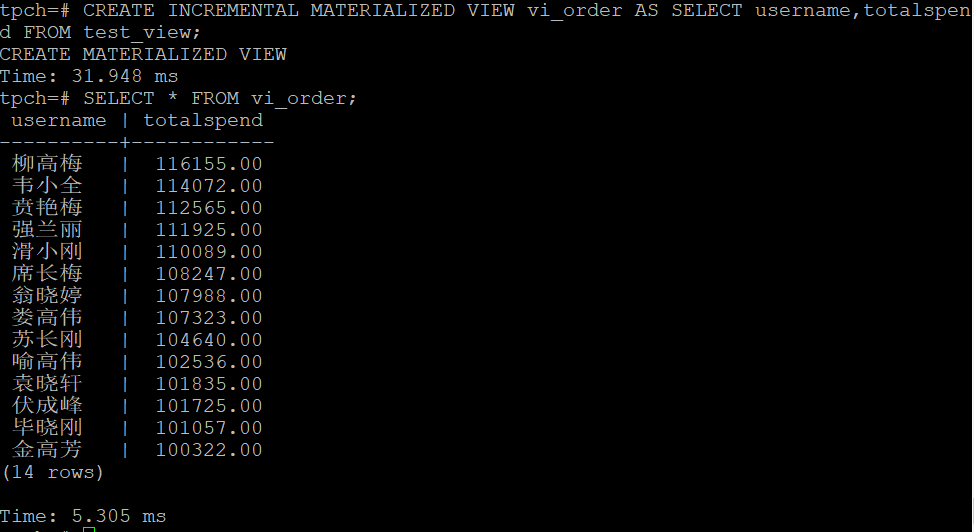
SELECT \* FROM v\_order;





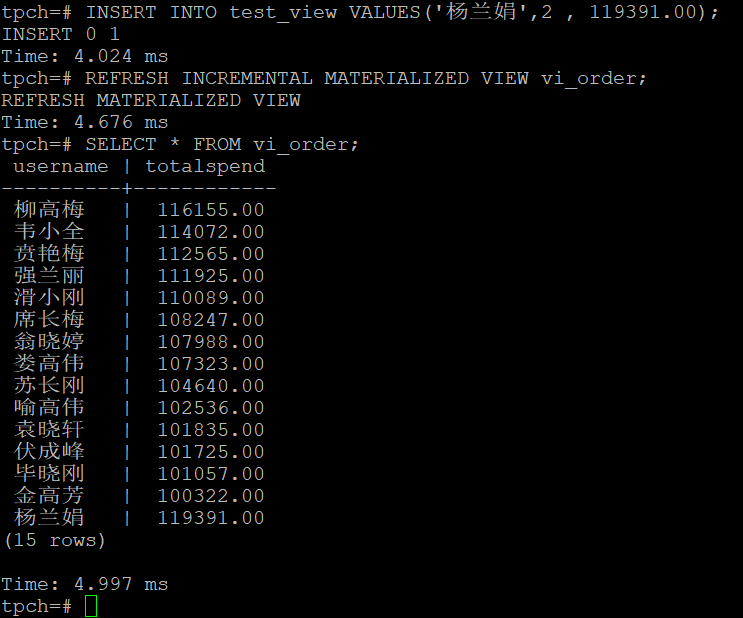
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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？

答：

(1)数据读取时，行存储通常将一行数据完全读出，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。列存储每次读取的数据是集合的一段或者全部，不存在冗余性问题。

(2)在进行数据更新和单条数据项的查询时，行存表的效率更高

(3)在执行求和，求平均值等计算操作时，以及查询多列数据时列存表效率更高

实践思考题2：全量物化视图与增量物化视图有哪些差别？

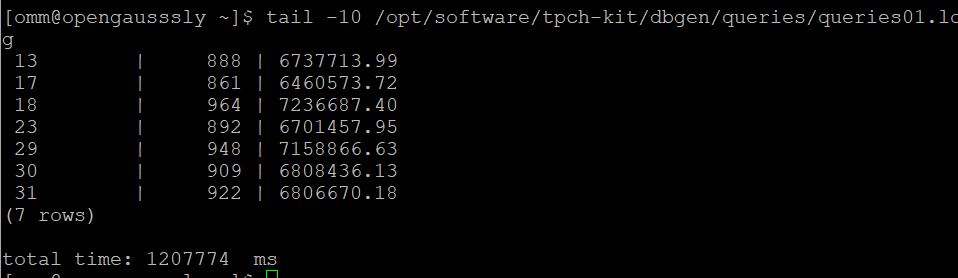
答：全量物化视图仅支持对创建好的物化视图做全量更新，而不支持做增量更新。创建全量物化视图语法和CREATE TABLE AS语法一致，不支持对全量物化视图指定NodeGroup创建。增量物化视图顾名思义就是可以对物化视图增量刷新，需要用户手动执行语句完成对物化视图在一段时间内的增量数据进行刷新。与全量创建物化视图不同在于目前增量物化视图所支持场景较小，目前物化视图创建语句仅支持基表扫描语句或者UNION ALL语句。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

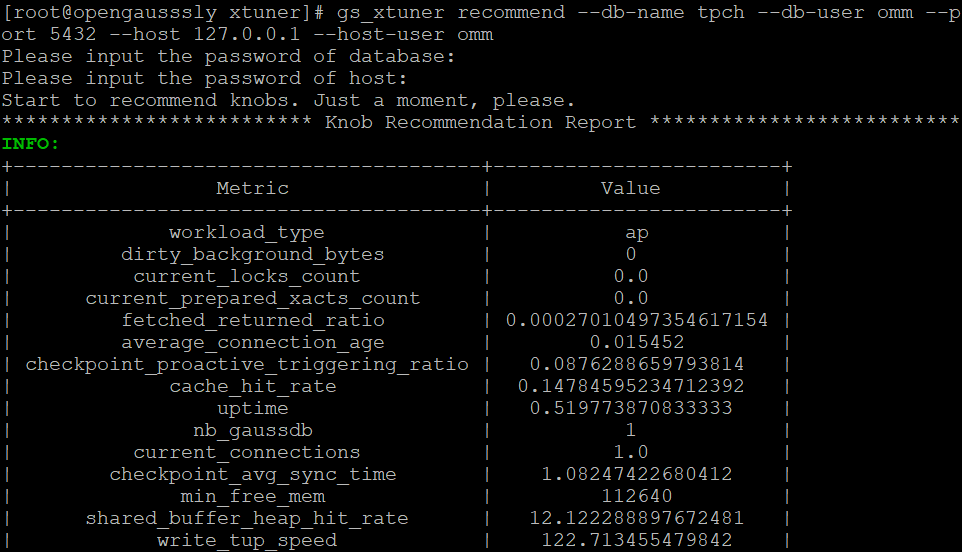
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

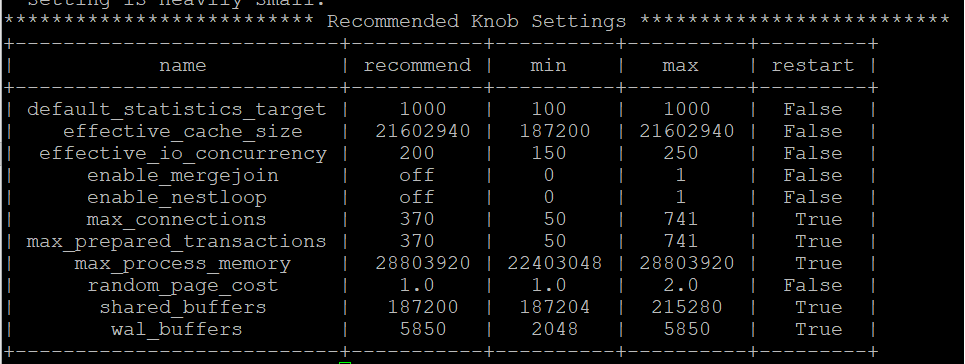
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log



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

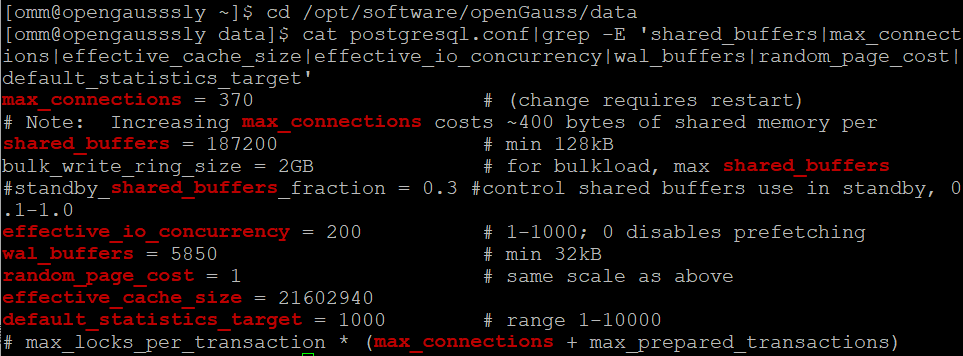




3.重启完成后，获取参数值：

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

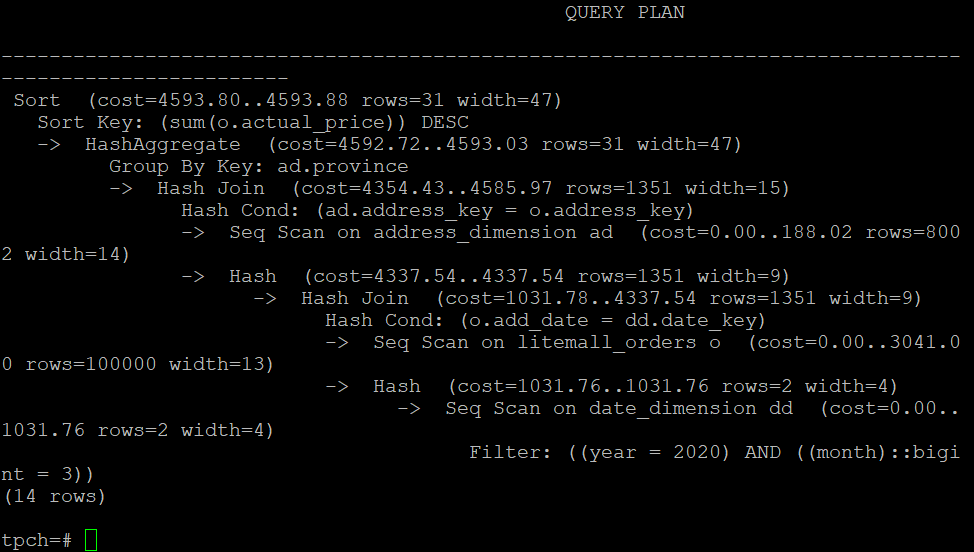
AND o.add\_date = dd.date\_key

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

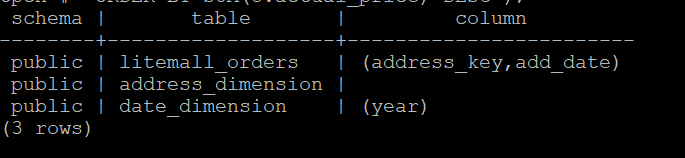
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

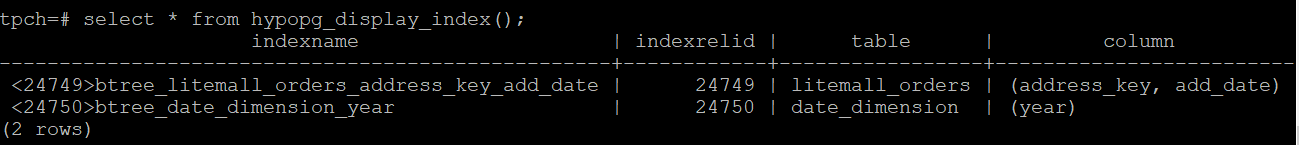
GROUP BY ad.province

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。

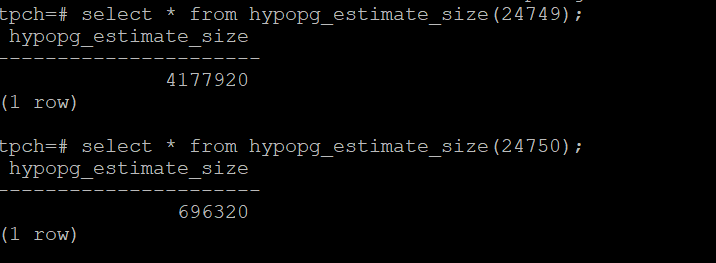
select \* from hypopg\_display\_index();



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(24749);

select \* from hypopg\_estimate\_size(24750);



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

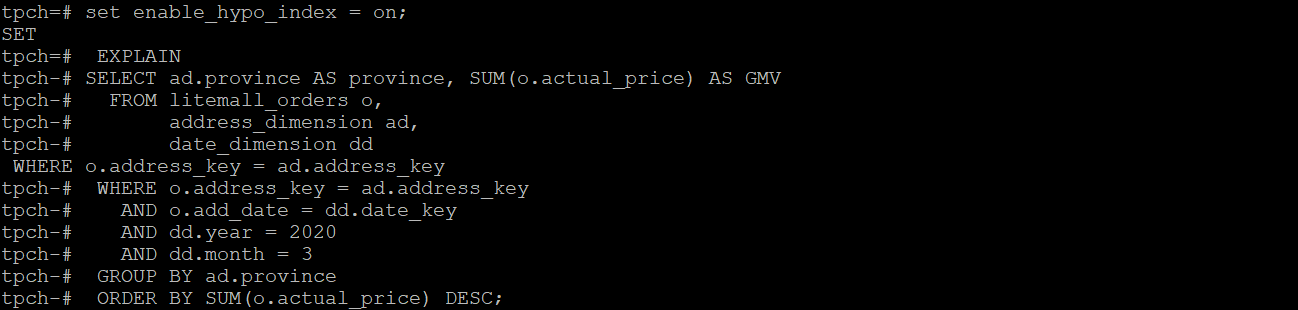
AND o.add\_date = dd.date\_key

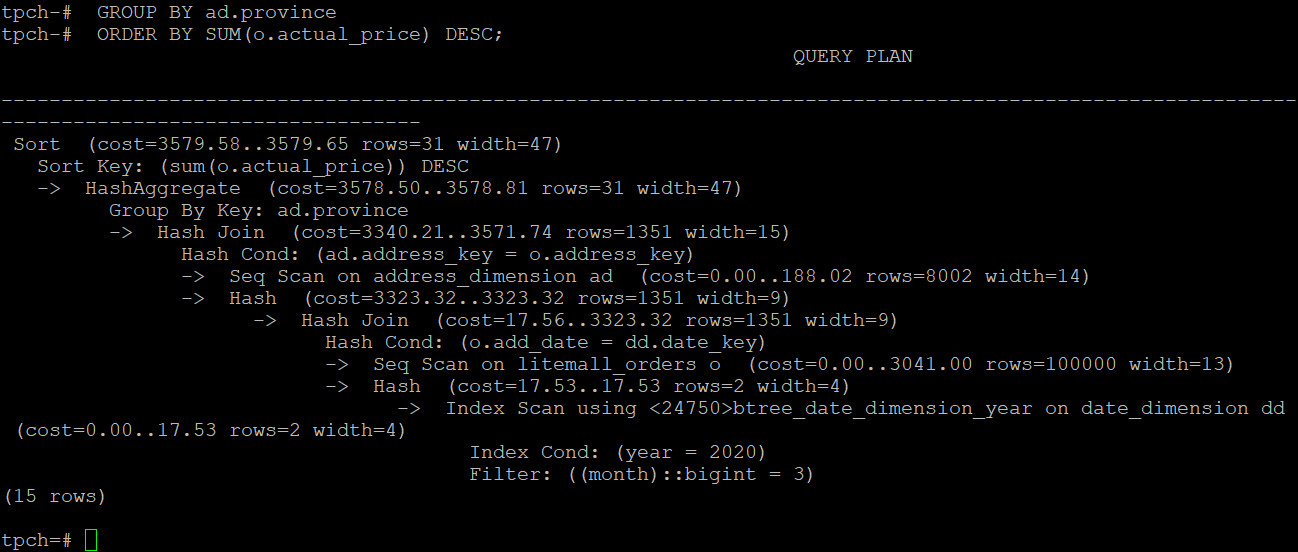
AND dd.year = 2020

AND dd.month = 3

GROUP BY ad.province

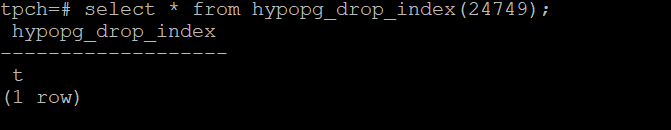
ORDER BY SUM(o.actual\_price) DESC;





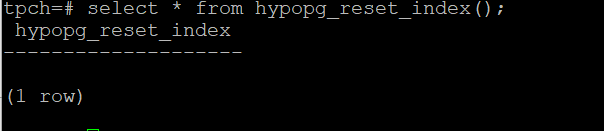
6. 删除某一个索引虚拟列，将执行结果截图。

select \* from hypopg\_drop\_index(24749);



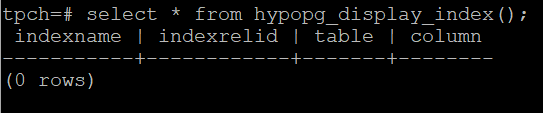
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

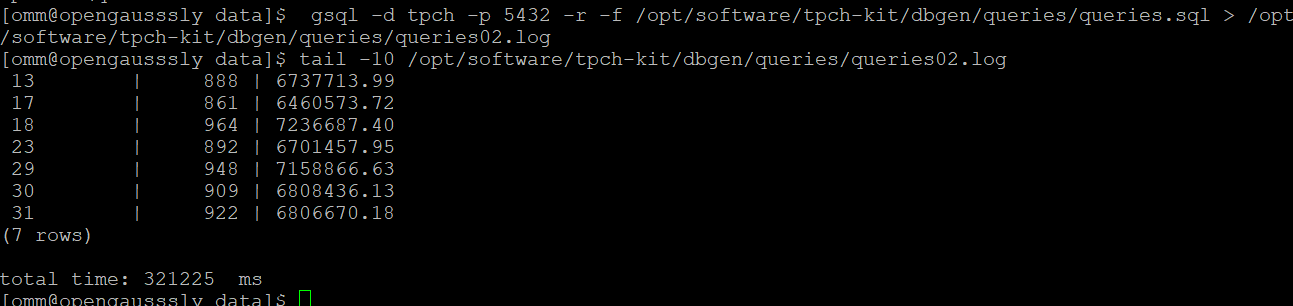
select \* from hypopg\_display\_index();



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？

答：根据X-Tuner给出的参数优化，主要优化的参数有：effective\_cache\_size,

effective\_io\_concurrency、shared\_buffers、max\_connections、wal\_buffers、random\_page\_cost、default\_statistics\_target，优化之后可以提高数据库运行效率，减少资源消耗。

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？

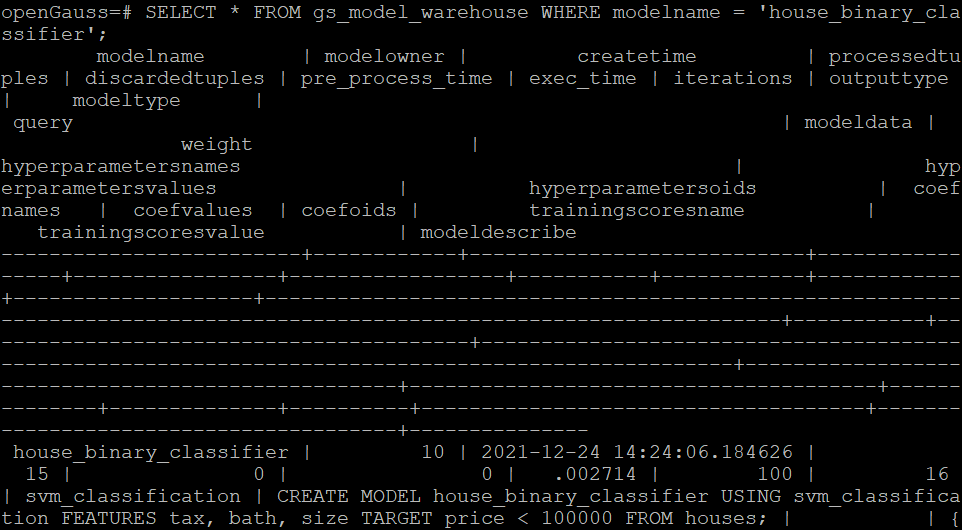
答：可以加快数据库查询速度：DB在执行一条Sql语句的时候，默认的方式是根据搜索条件进行全表扫描，遇到匹配条件的就加入搜索结果集合。如果我们对某一字段增加索引，查询时就会先去索引列表中一次定位到特定值的行数，大大减少遍历匹配的行数，所以能明显增加查询的速度。

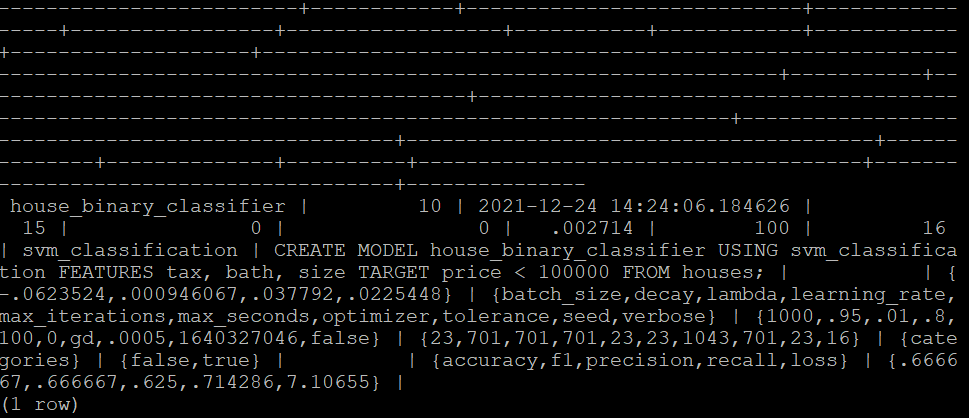
例如数据库结构的改变：行存储和列存储在不同操作时可以显著增加运行速度。数据库表的范式化。更改数据库的相关配置。提高硬件水平等。

# 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：

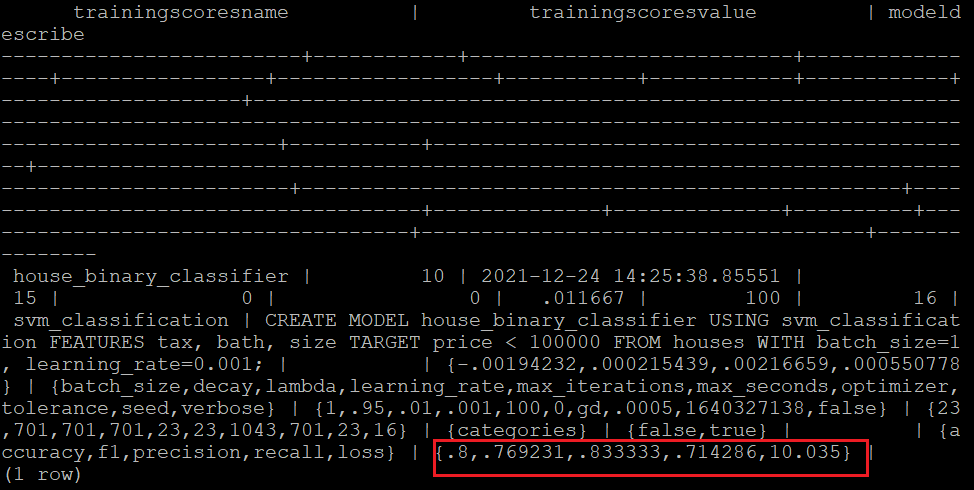
postgres=# SELECT \* FROM gs\_model\_warehouse WHERE modelname = 'house\_binary\_classifier';





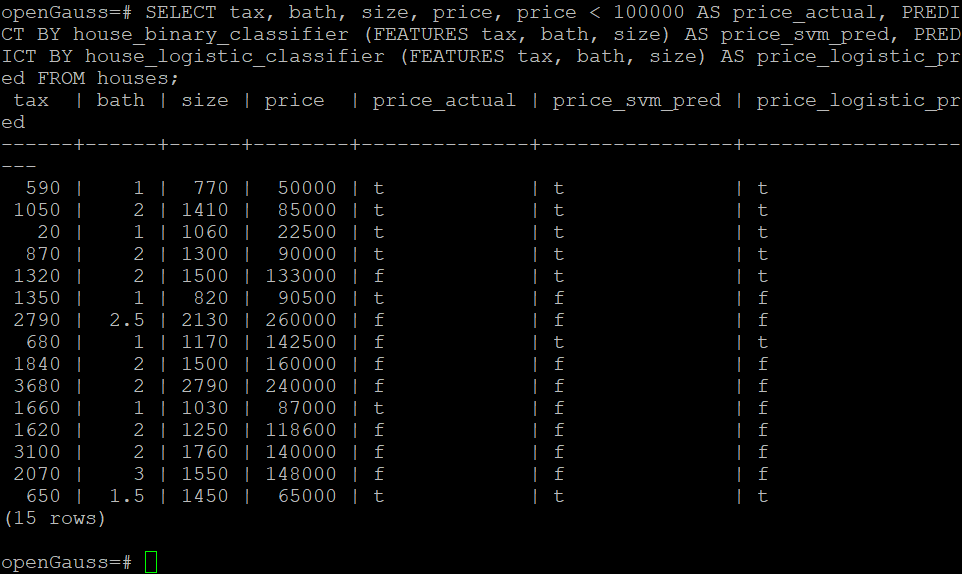
任务二：观察新模型的信息，将执行结果截图。

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？

答：分类指的是定性输出，或者也可以有条件地表述为离散变量预测。给定一个新的模式，根据训练集推断它所对应的类别（如：+1，-1），是一种定性输出，也叫离散变量预测。

回归指的是定量输出，或者也可以有条件地表述为连续变量预测。给定一个新的模式，根据训练集推断它所对应的输出值（实数）是多少，是一种定量输出，也叫连续变量预测。

回归问题通常是用来预测一个值，如预测房价、未来的天气情况等等，分类问题是用于将事物打上一个标签，通常结果为离散值。例如判断一幅图片上的动物是一只猫还是一只狗，分类通常是建立在回归之上，分类的最后一层通常要使用softmax函数进行判断其所属类别。

实践思考题2：什么是SVM算法？

答：SVM的全称是Support Vector Machine，即支持向量机，主要用于解决模式识别领域中的数据分类问题，属于有监督学习算法的一种。SVM是一种二分类模型，它的基本模型是定义在特征空间上的间隔最大的线性分类器，间隔最大使它有别于感知机；SVM还包括核技巧，这使它成为实质上的非线性分类器。

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？

答：首先明确几个定义：混淆矩阵

TP： True Positives， 表示实际为正例且被分类器判定为正例的样本数

FP： False Positives， 表示实际为负例且被分类器判定为正例的样本数

FN： False Negatives， 表示实际为正例但被分类器判定为负例的样本数

TN： True Negatives， 表示实际为负例且被分类器判定为负例的样本数

分类问题的评价指标如下：

Accuracy：准确率。评价分类问题的性能的指标一般是分类准确率，其定义是对于给定的数据，分类正确的样本数占总样本数的比例。Accuracy = 预测正确的样本数 / 所有样本数。

Precision and Recall(精准率和召回率)：精准率可以解释为，预测为正例的样本中，有多少是真的正例。召回率可以解释为，真实的正例的样本中，有多少被预测出来。

F1-Score：F1Score = 2\*(Precision \* Recall) / (Precision + Recall).

ROC 曲线：ROC 曲线主要关注两个指标：FPR和TPR。FPR 代表将负例错分为正例的概率，TPR 表示能将正例分对的概率， 如果我们增大阈值， 则 TPR 会增加，而对应的FPR也会增大， 而绘制ROC曲线能够帮助我们找到二者的均衡点。

AUC：Area under Curve 。AUC 为ROC 曲线下的面积，这个面积的数值介于0到1之间，能够直观的评价出分类器的好坏，AUC的值越大，分类器效果越好。

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？

回归问题的评价指标有：MSE，RMSE，MAE、R-Squared。

MSE（Mean Square Error）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。通过平方的形式便于求导，所以常被用作线性回归的损失函数。

RMSE（Root Mean Square Error）：均方根误差是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。

MAPE(Mean Absolute Percentage Error)：所有样本真实值与预测值差绝对值与真实值的比例的和求平均。

MASE（Mean Scaled Percentage Error）平均平方百分比误差：即求所有样本的真实值与预测值的差与真实值的比例求平方的和求平均。