openGauss AI特性创新实践课



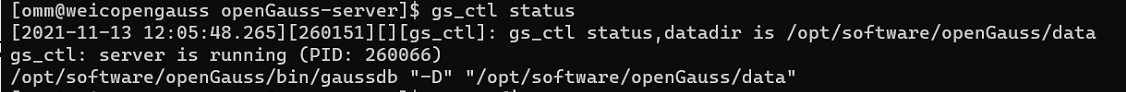
华为技术有限公司

# 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：

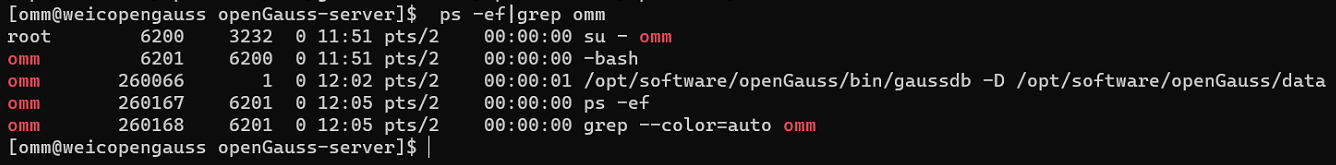
任务一：数据库状态验证

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？

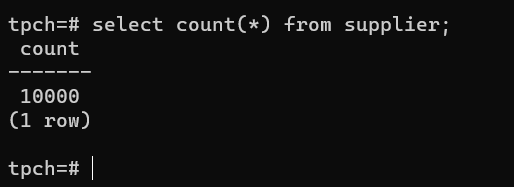
一个软件要在Linux上执行，必须是二进制文件，因此当我们拿到软件源码后，需要将它编译成二进制文件才能在Linux上运行。因此安装数据库首先需要获取openGauss的源码，之后使用make命令将源码编译成可执行的二进制文件，之后再使用make install命令将openGauss进行安装。

# 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：

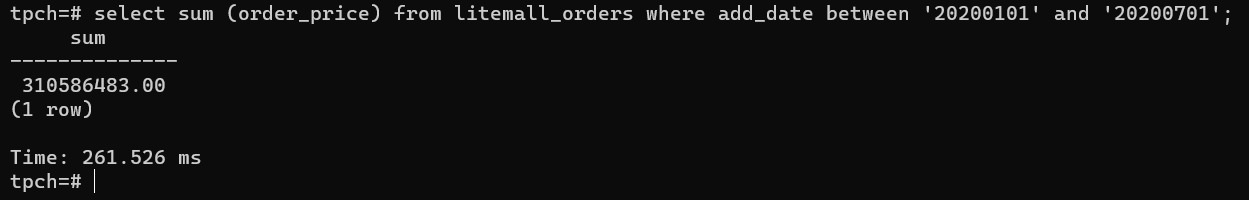
select count(\*) from supplier;



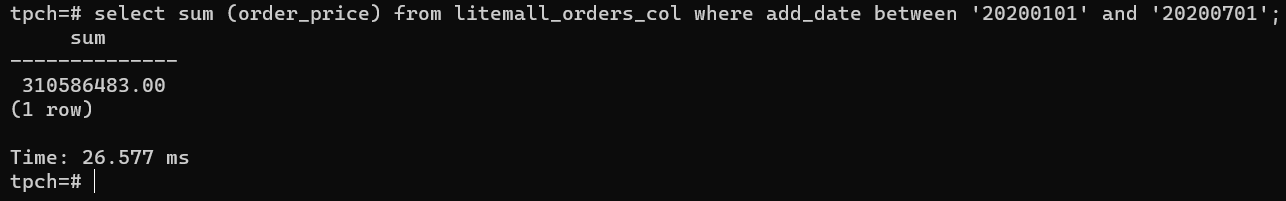
任务二：行存表与列存表执行效率对比

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';

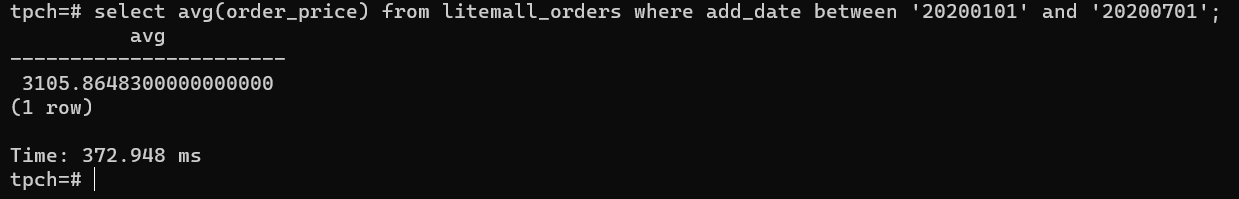


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

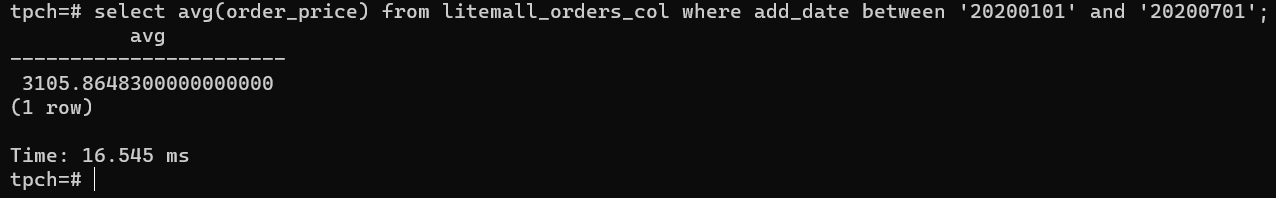


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

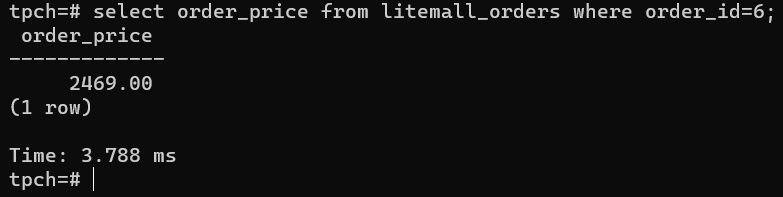


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

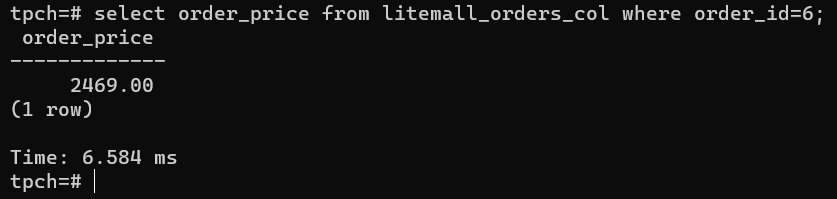


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

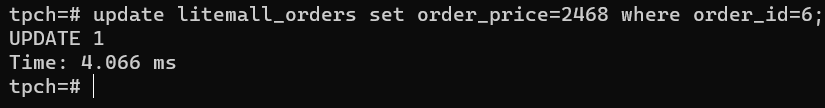


select order\_price from litemall\_orders\_col where order\_id=6;

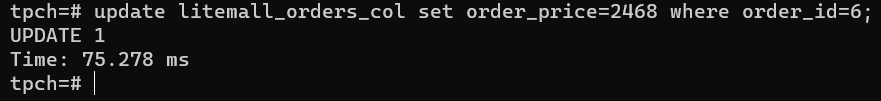


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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

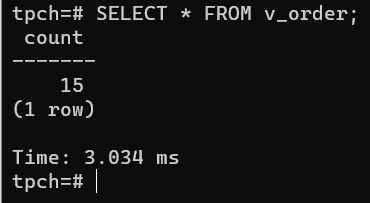
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



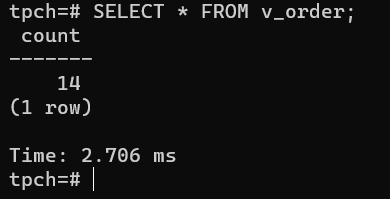
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



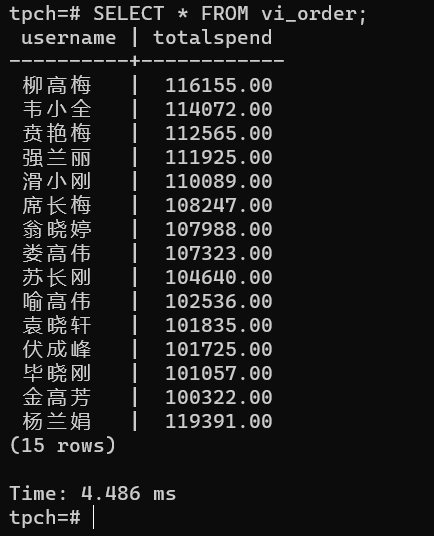
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



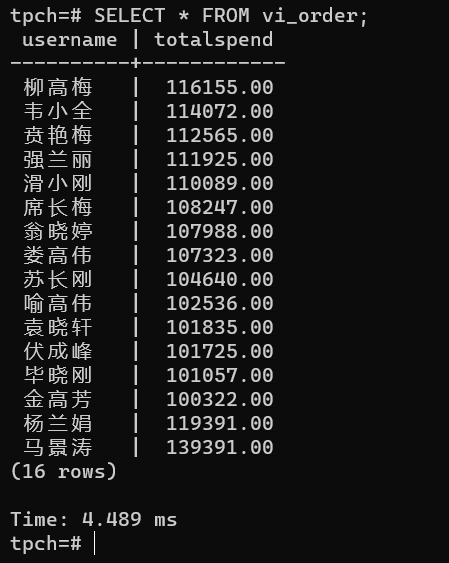
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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？

因为行存表的的连接方式是每一行的末尾去连接下一行，相当于每到下一行都需要遍历一

遍当前行的所有列；而列存表则是从上往下先将第一列的属性连接起来，之后再在末尾连接下一列的属性。由于两种连接方式不同导致了执行相同的SQL语句时执行的时间也不同。

在执行insert、update以及需要对数据进行的操作时，行存表的效率更高。

而在需要查询具体的某些或某个属性时，列存表的效率更高。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

全量物化视图刷新是将视图里的对应表的数据库全部删除，之后再从原表中用insert把数据重新插入。

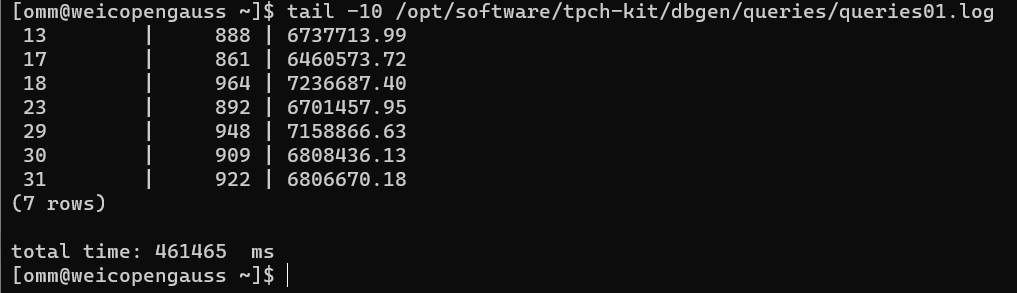
增量物化视图则是当原表进行了插入、更新或删除操作时，会在物化视图日志进行记录，之后刷新时根据物化视图日志的内容执行相应的操作，刷新后物化视图日志清空。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

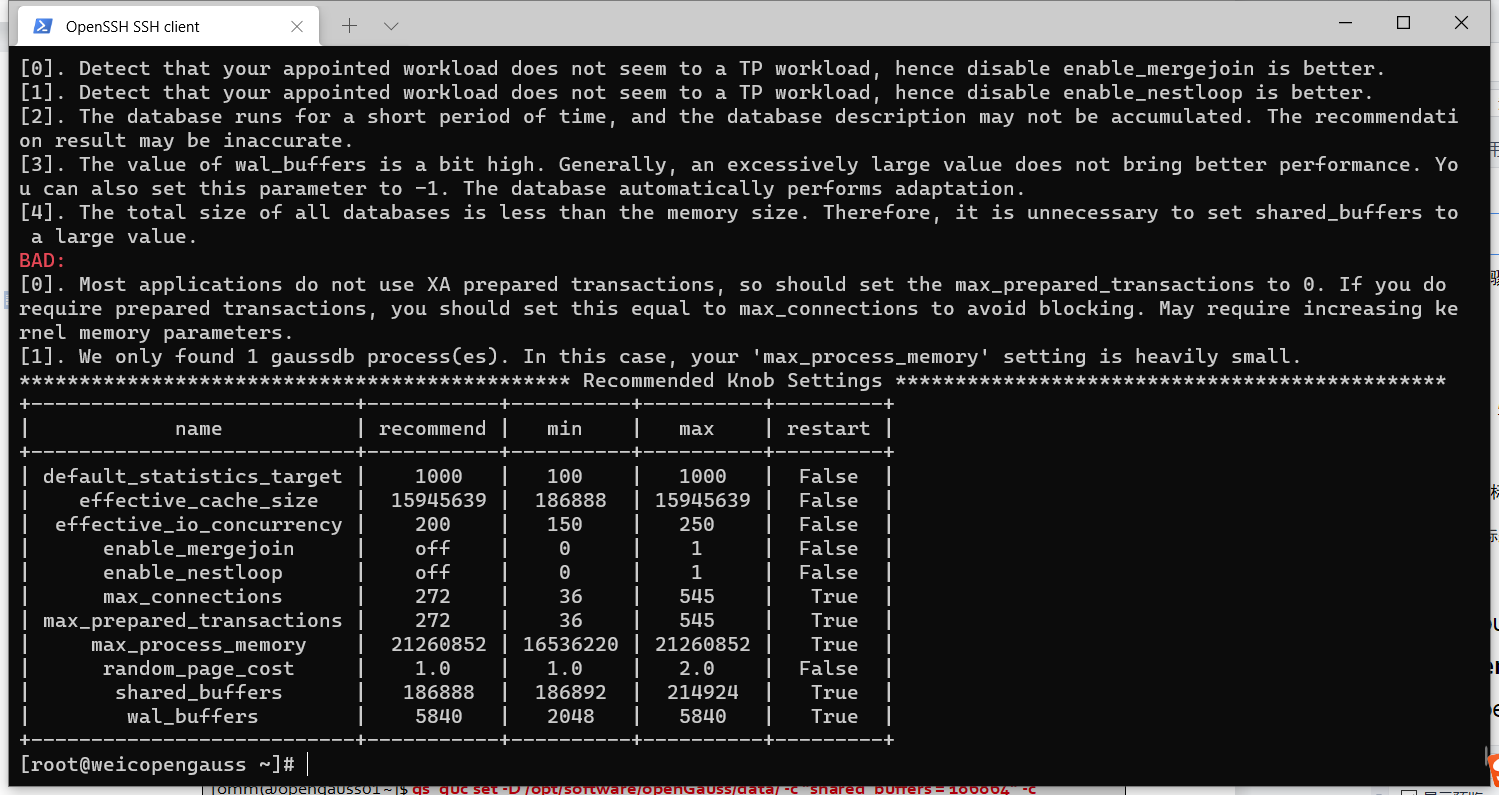
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

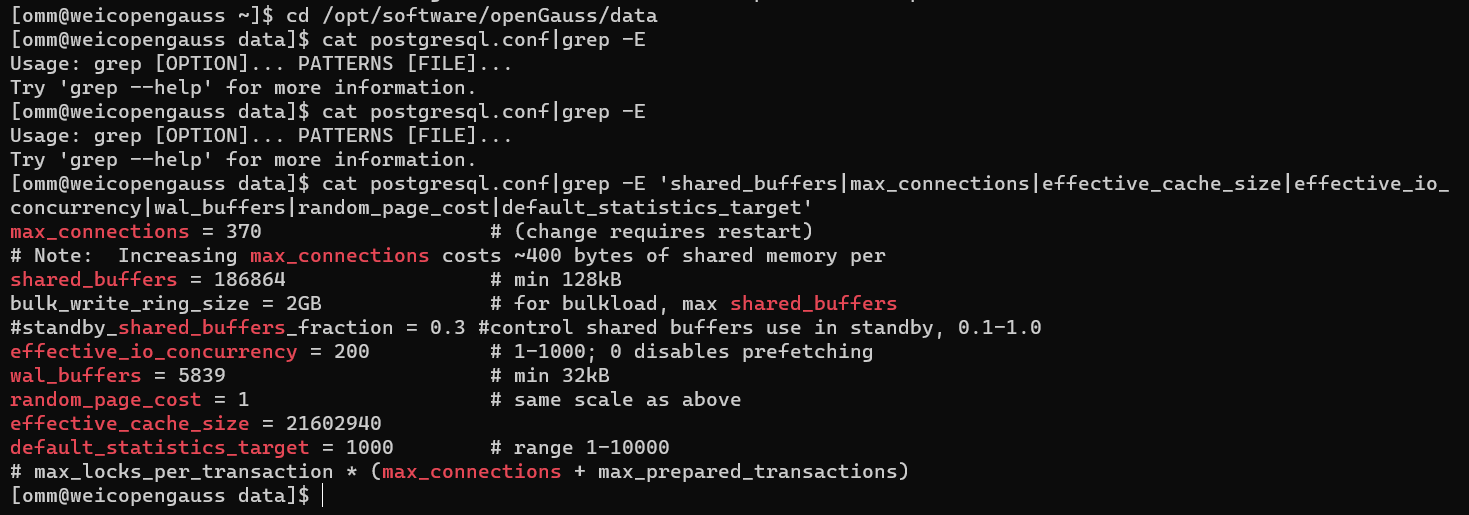
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

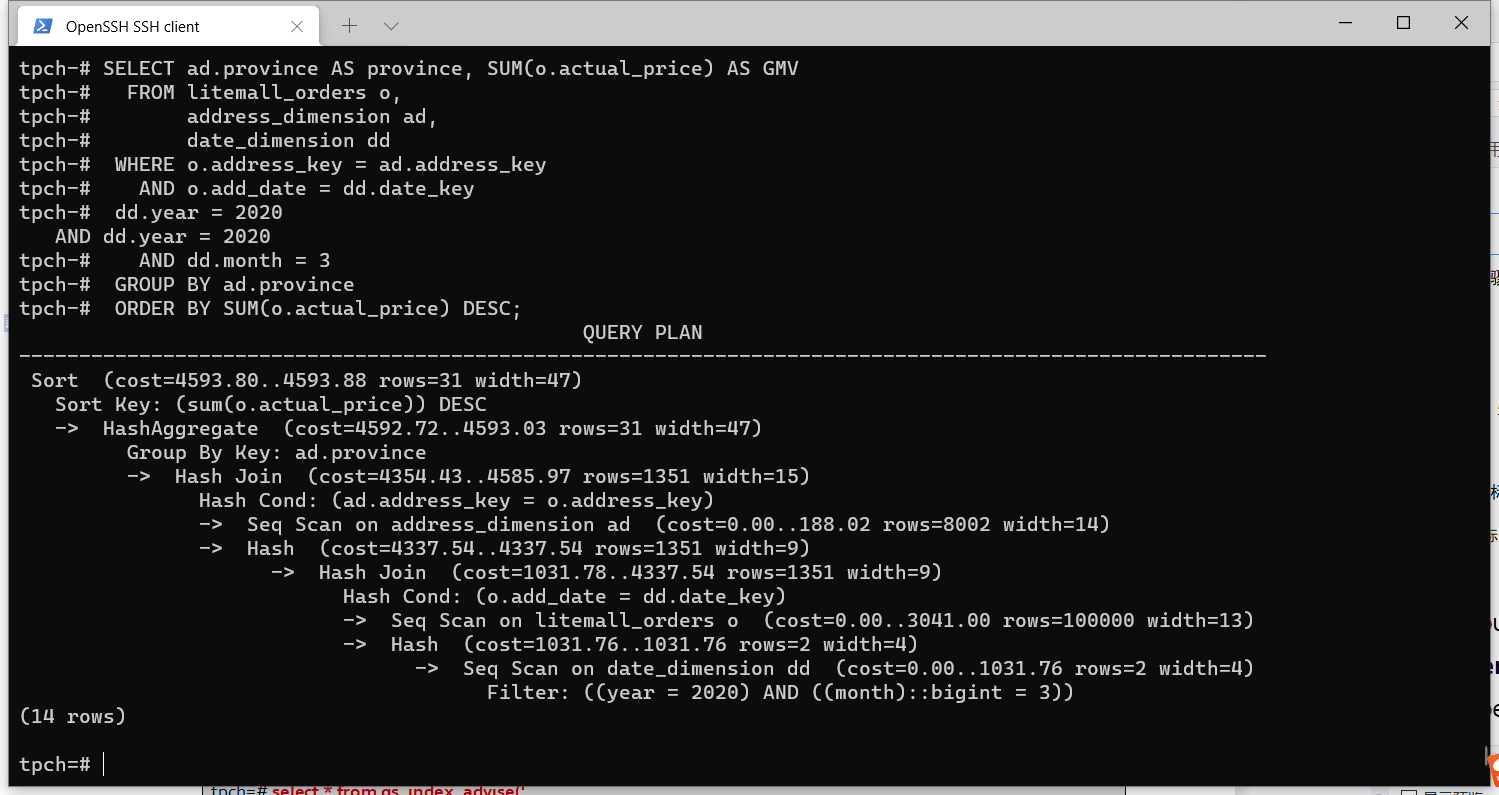
AND o.add\_date = dd.date\_key

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

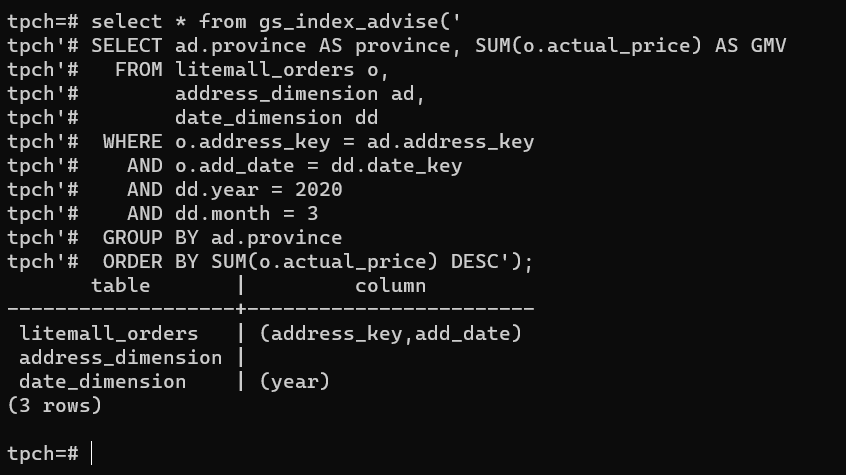
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

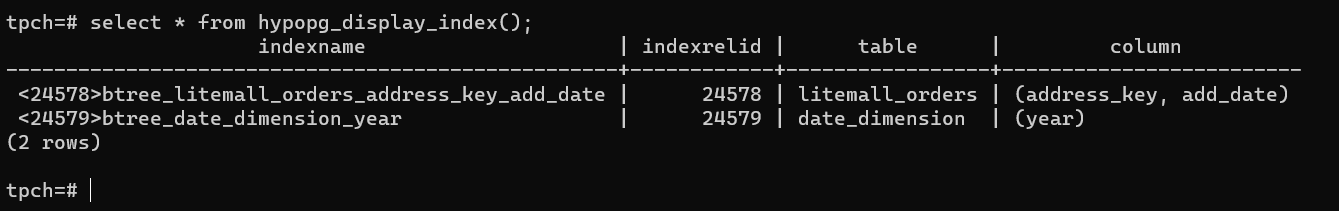
GROUP BY ad.province

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。

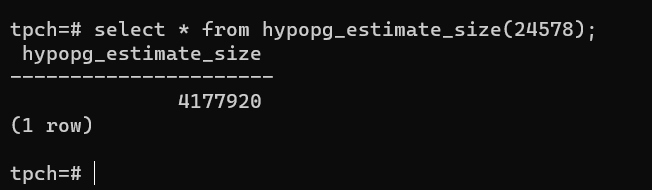
select \* from hypopg\_display\_index();

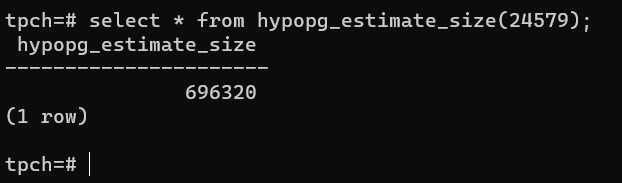


4. 获取索引虚拟列大小结果（单位为：字节），将执行结果截图。

select \* from hypopg\_estimate\_size(24578);

select \* from hypopg\_estimate\_size(24579);





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

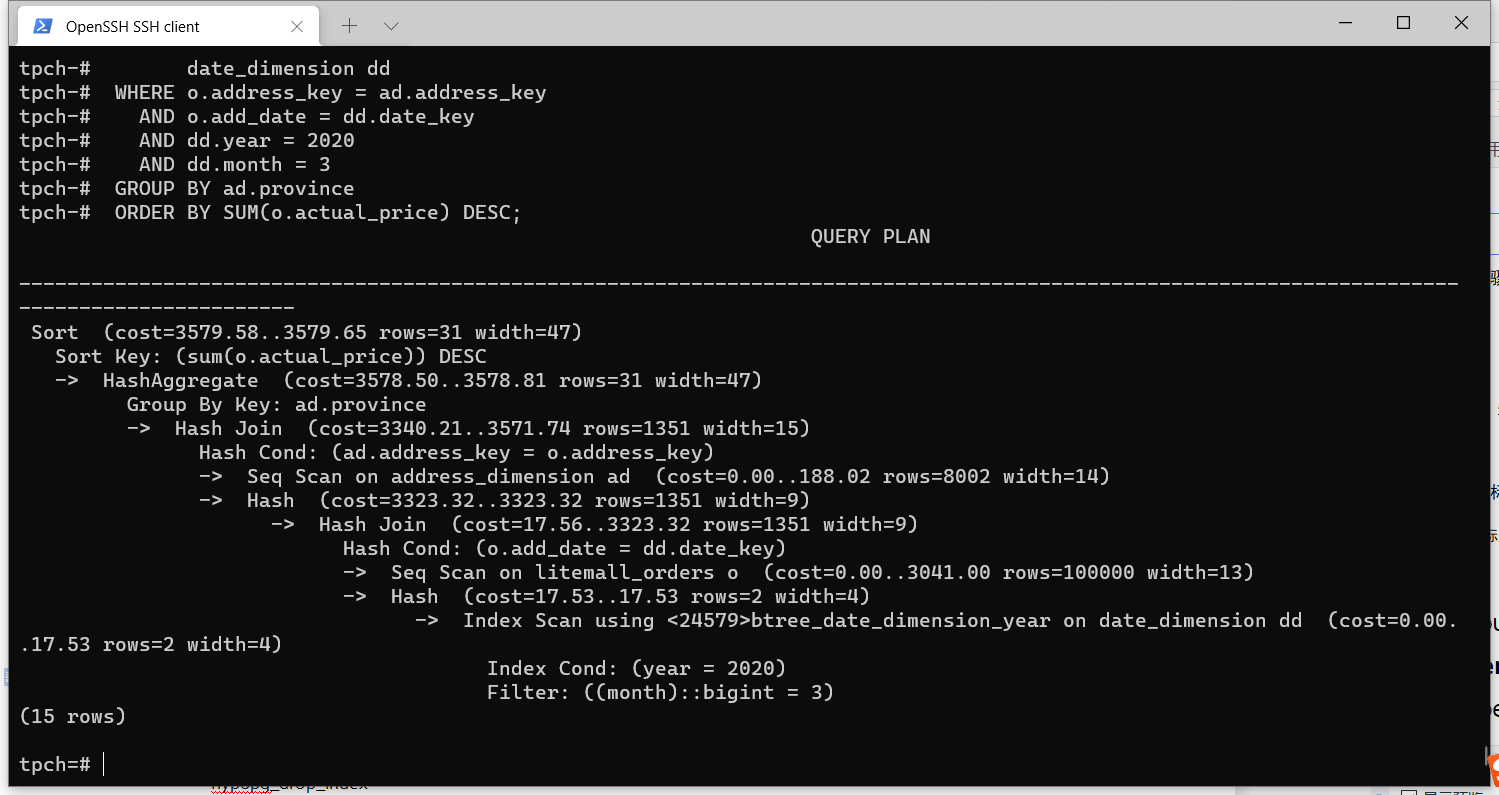
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

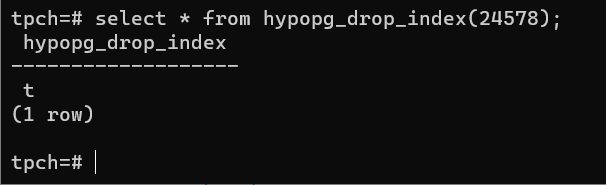
GROUP BY ad.province

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



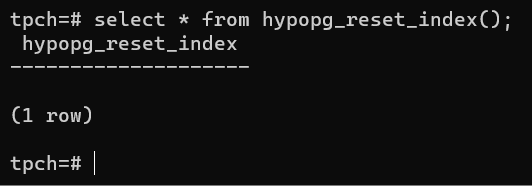
6. 删除某一个索引虚拟列，将执行结果截图。

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



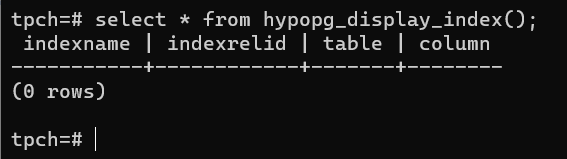
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

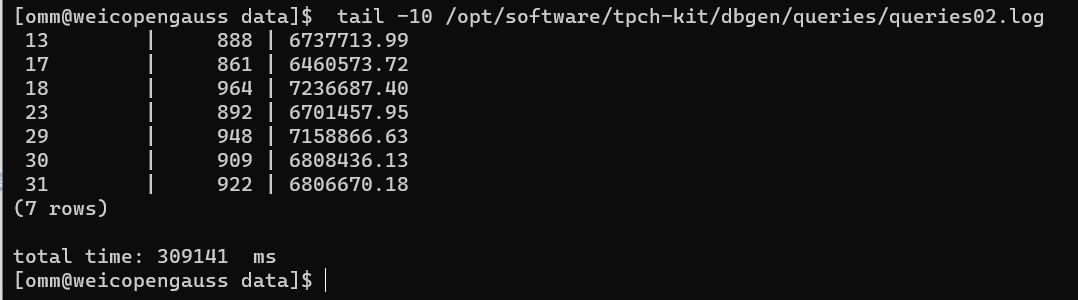
select \* from hypopg\_display\_index();



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

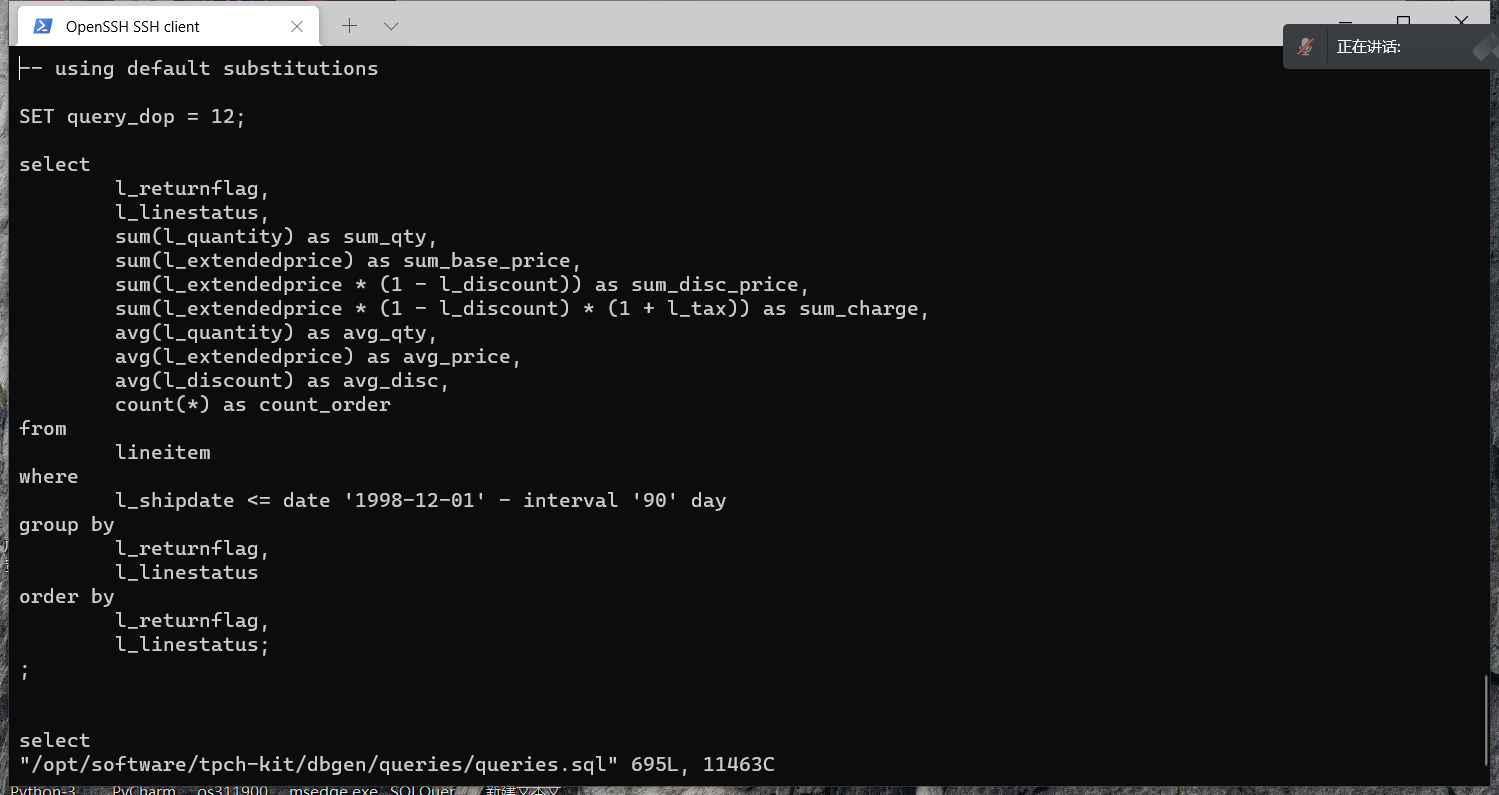


挑战一：进一步优化queries.sql中的查询语句，使得前后执行时间出现倍数级的提升。

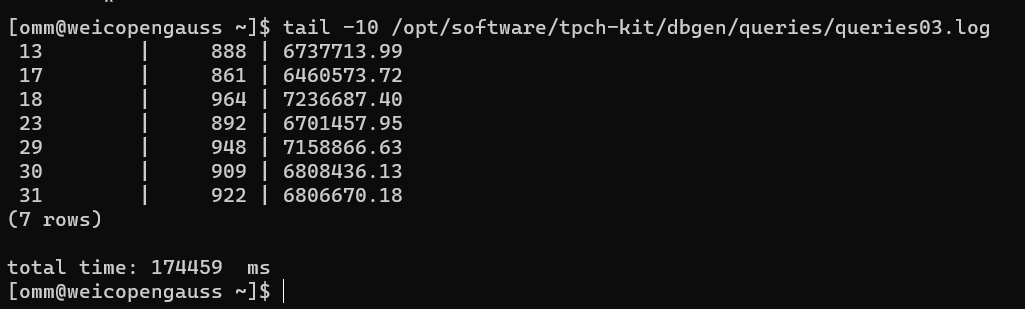
1. 重新执行queries.sql查询，将执行结果截图：

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

在queries.sql里面添加命令“SET query\_dop = 12”（本机linux是12核）之后再执行



执行结果：



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？

shared\_buffers: 增大设置中用于缓存的专用内存量以提高性能

max\_connections: 增加最大连接数来满足多个客户端的需求

effective\_cache\_size: 提高磁盘高速缓存的内存量的估计值以使索引扫描更多被使用

effective\_io\_concurrency: 增大以增加任何单个会话试图并行发起的 I/O 操作的数目

wal\_buffers: 增大设置中用于wal记录的缓冲区以提高性能

random\_page\_cost

default\_statistics\_target

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？

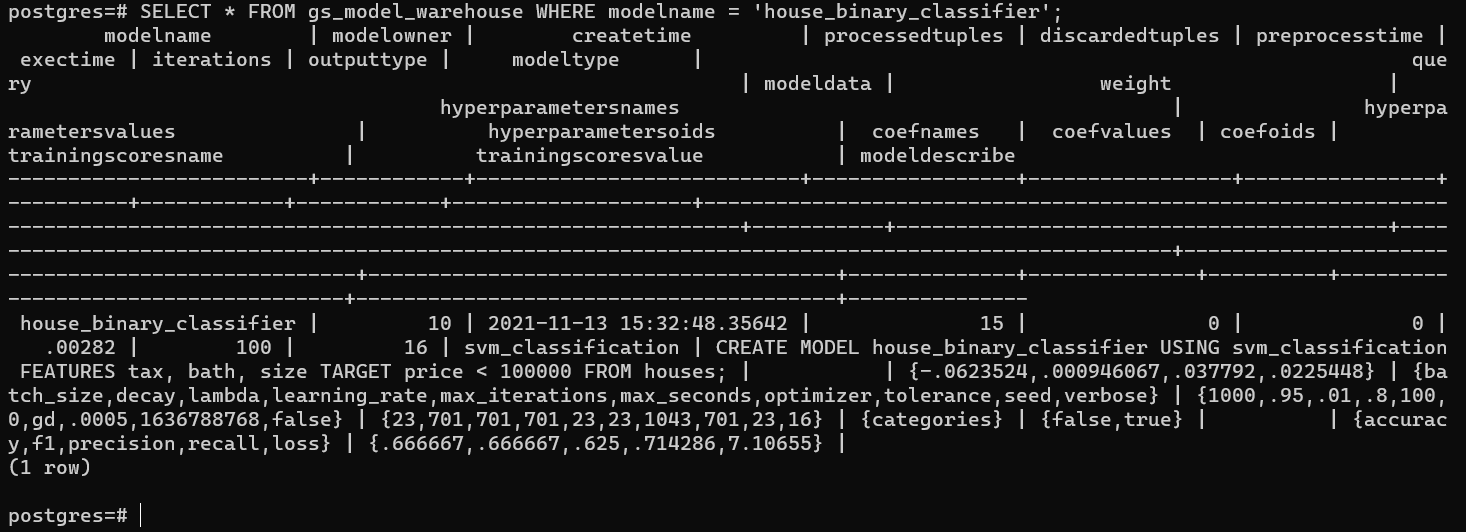
在有大量数据的数据库中使用索引，可以在执行SQL更快地找到需要的属性，优化执行SQL时的速度，用时更少。

对数据库进行优化的方面还有：使用连接来代替子查询；使用联合来代替手动创建的临时表；使用外键。

# 关卡四、openGauss的DB4AI特性应用

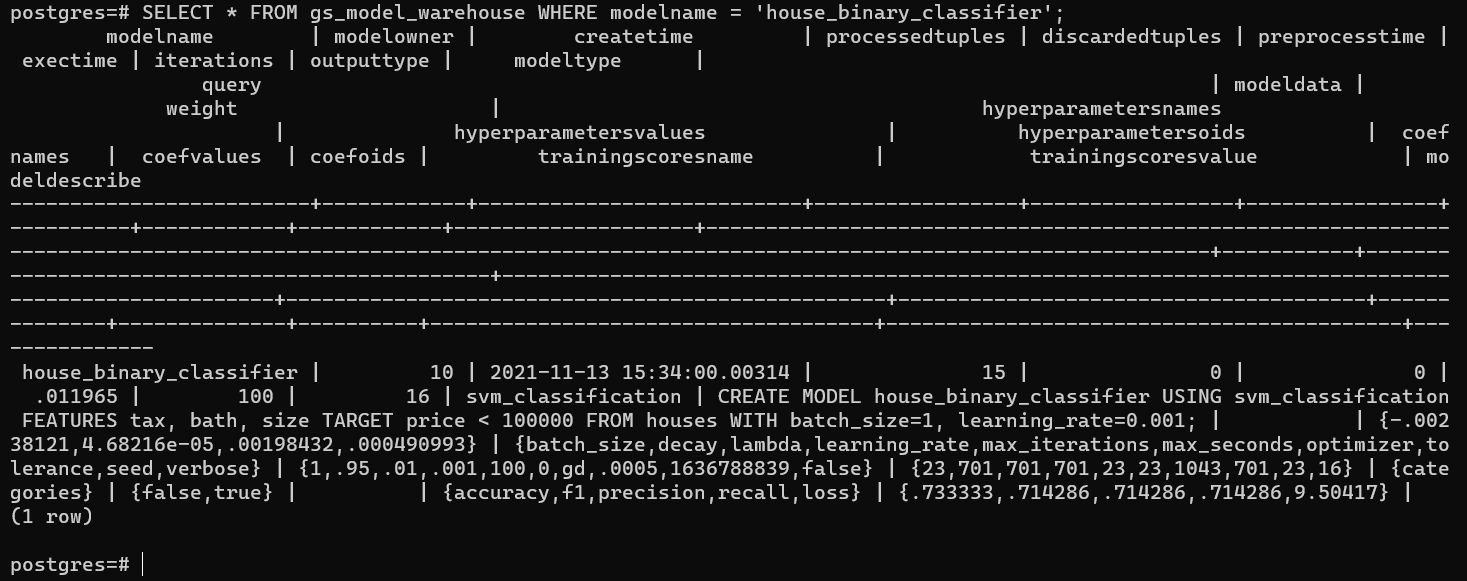
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



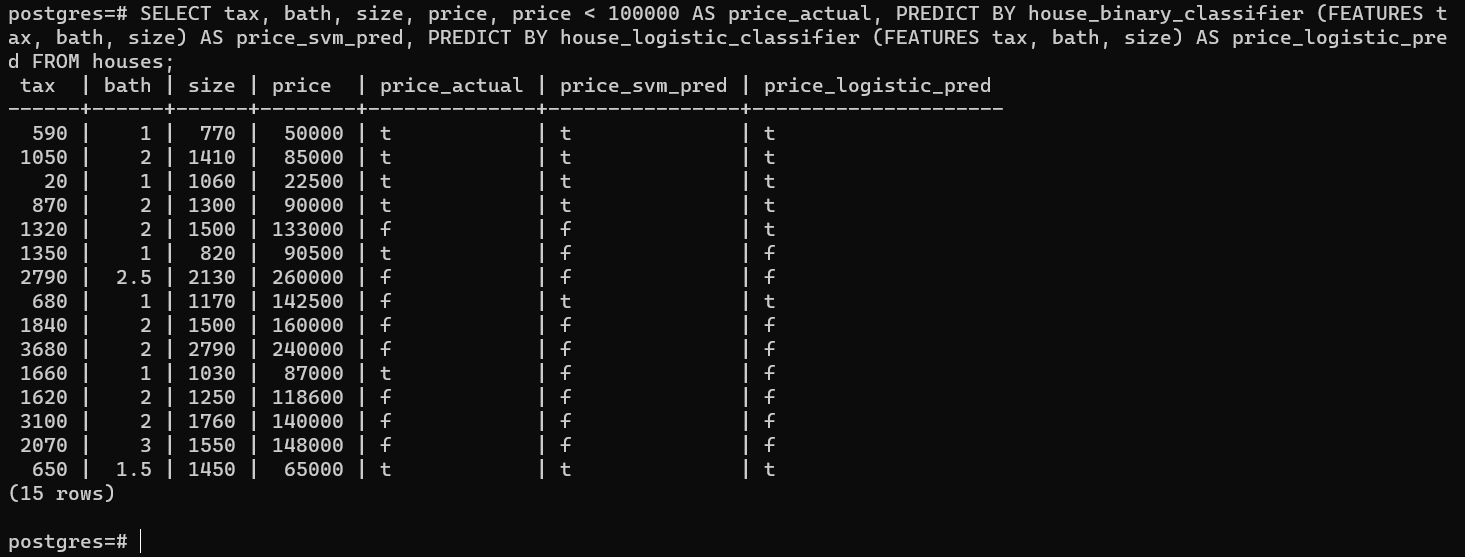
任务二：观察新模型的信息，将执行结果截图。

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？

定量输出称为回归，或者说是连续变量预测；

定性输出称为分类，或者说是离散变量预测。

实践思考题2：什么是SVM算法？

SVM（support Vector Machine）又称为支持向量机，是一种二分类的模型，可以分为线性核非线性两大类，经过演进，现在也可以支持多元分类，同时经过扩展，也能应用于回归问题。其主要思想为找到空间中的一个更够将所有数据样本划开的超平面，并且使得本本集中所有数据到这个超平面的距离最短。

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？

准确率（Accuracy）：准确率是分类问题中最为原始的评价指标，准确率的定义是预测正确的结果占总样本的百分比。

精确率（Precision）：精准率（Precision）又叫查准率，它是针对预测结果而言的，它的含义是在所有被预测为正的样本中实际为正的样本的概率，意思就是在预测为正样本的结果中，我们有多少把握可以预测正确。

召回率（Recall）：召回率（Recall）又叫查全率，它是针对原样本而言的，它的含义是在实际为正的样本中被预测为正样本的概率。

P-R曲线：P-R曲线（Precision Recall Curve）正是描述精确率/召回率变化的曲线。

F1-Score：F-Measure是P(Precision)和R(Recall)的加权调和平均。

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？

解释方差（explained\_variance）：解释方差的最大值是1.0，数值越大代表模型预测结果越好，越小代表模型预测结果越差

平均绝对值误差（MAE）：计算每一个样本的预测值和真实值的差的绝对值，然后求和再取平均值。用于评估预测结果和真实数据集的接近程度，其值越小说明拟合效果越好。

平均绝对百分比误差（MAPE）：范围[0,+∞)，MAPE 为0%表示完美模型，MAPE 大于 100 %则表示劣质模型。

均方误差（MSE）：计算每一个样本的预测值与真实值差的平方，然后求和再取平均值。该指标计算的是拟合数据和原始数据对应样本点误差的平方和的均值，其值越小说明拟合效果越好。

均方根误差（RMSE）：均方根误差就是在均方误差的基础上再开方，其值越小说明拟合效果越好。

可决系数（R²）：可决系数值在0~1之间。越接近于1，说明模型的预测效果越好，越接近于0，说明模型的预测效果越差，当然也存在负值，此时说明模型的效果非常差。