

openGauss AI特性创新实践课

专业班级： 195202

学生姓名： 方寒飞

学生学号： 20201002538

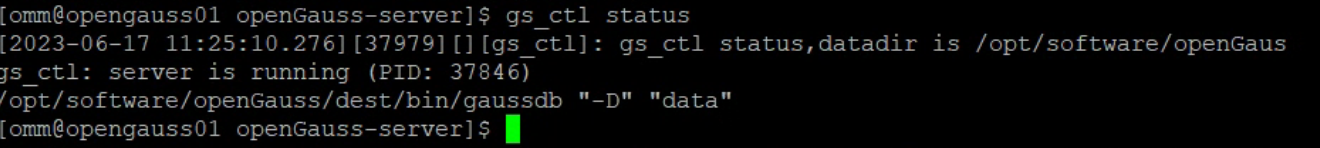
**中国地质大学计算机学院**

# 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：

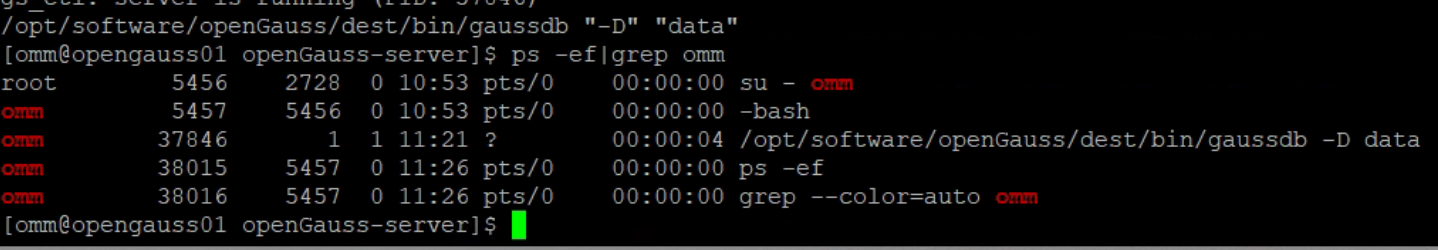
任务一：数据库状态验证

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）



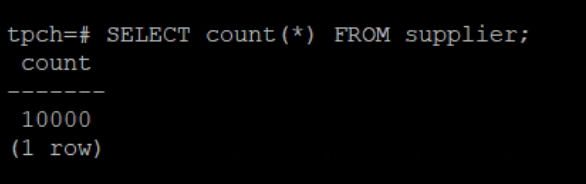
实验思考题：为什么需要通过源码编译，安装数据库？

# 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：

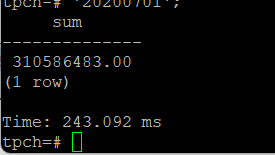
select count(\*) from supplier;;



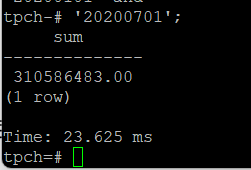
任务二：行存表与列存表执行效率对比

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';

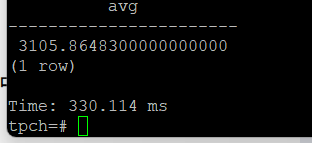


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

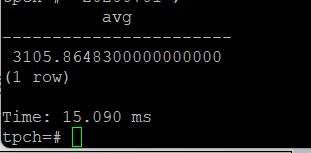


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

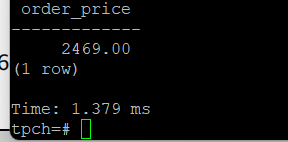


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

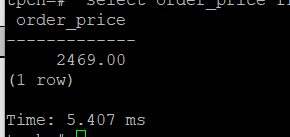


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

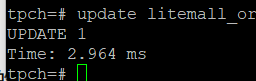


select order\_price from litemall\_orders\_col where order\_id=6;

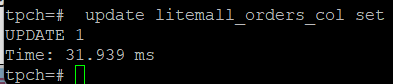


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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

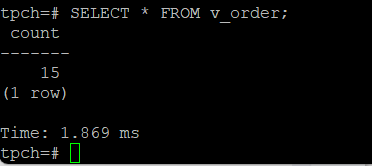
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



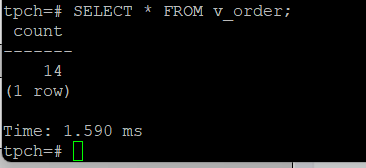
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



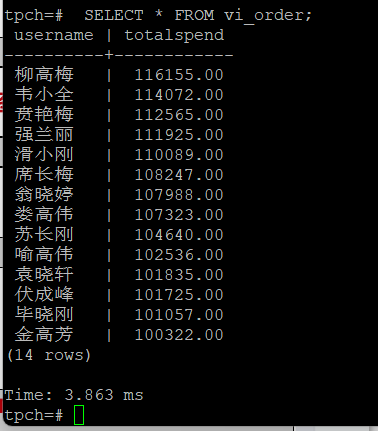
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

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？

由于行存表和列存表的存储方式不同，它们在执行相同的SQL语句时会有不同的执行时间。在执行需要读取整行数据的SQL语句时，行存表的效率更高。因为行存表的数据在磁盘上是按行存储的，当需要读取整行的数据时，可以通过一次磁盘读取将整行数据加载到内存中，从而减少了磁盘I/O操作的次数。在执行需要读取部分列数据的SQL语句时，列存表的效率更高。因为列存表的数据在磁盘上是按列存储的，当需要读取部分列的数据时，只需要读取包含所需列的数据块，而不必读取整行数据。这可以减少磁盘I/O操作的数据量，提高查询的速度。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

全量物化视图和增量物化视图之间的主要差别在于数据刷新方式、刷新频率和存储要求。全量物化视图通过重新计算和加载源表的全部数据来更新，需要读取和处理整个源表的数据，通常需要定期或按需手动刷新。相比之下，增量物化视图仅处理源表中发生变化的数据，只更新物化视图中的变化部分，刷新操作更快且可以实时或定期自动刷新。此外，全量物化视图需要足够的存储空间来容纳源表的全部数据，而增量物化视图只存储源表中变化的部分数据。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

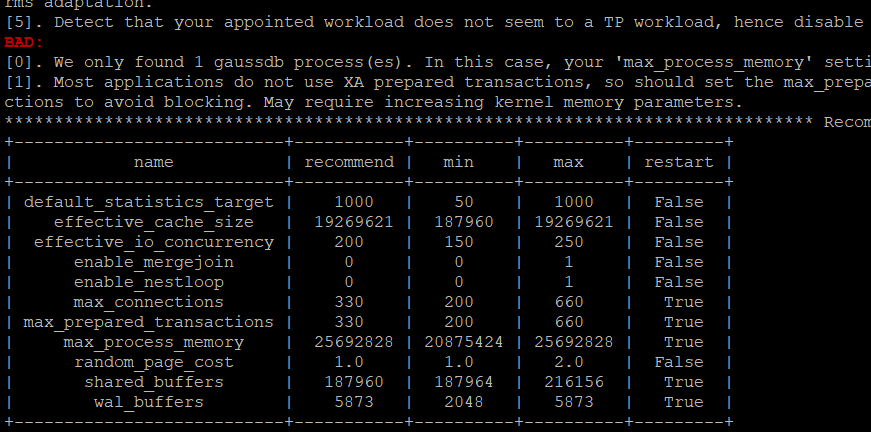
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

IMG_256

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

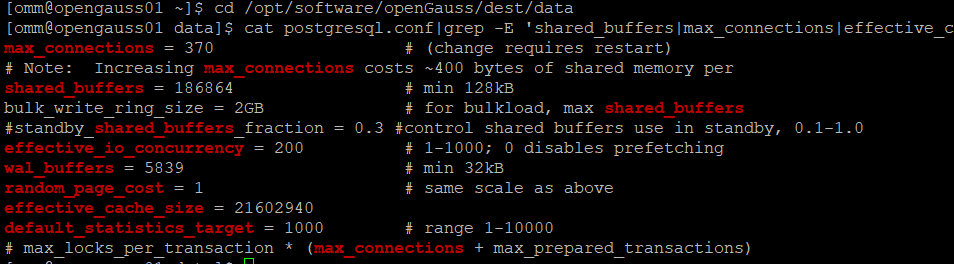
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

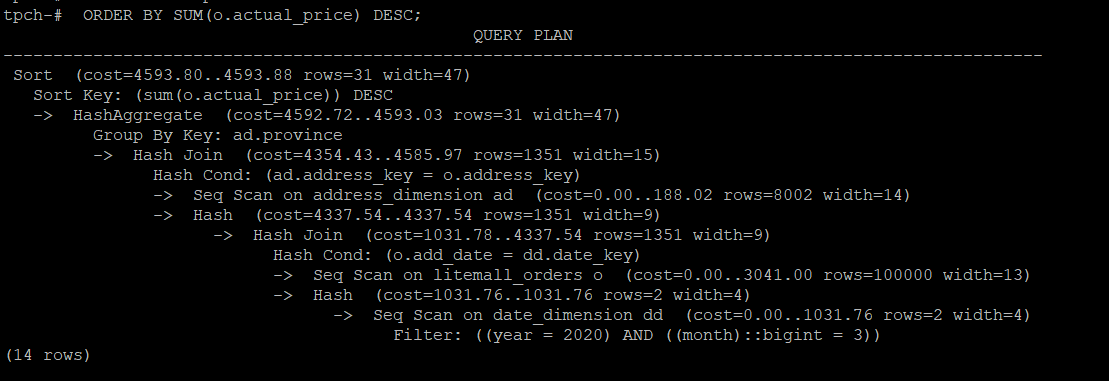
AND o.add\_date = dd.date\_key

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

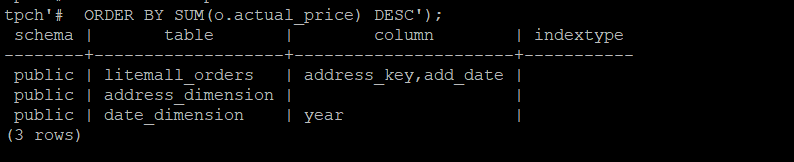
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

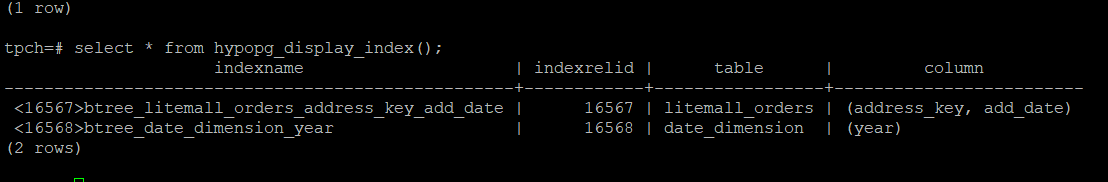
GROUP BY ad.province

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。

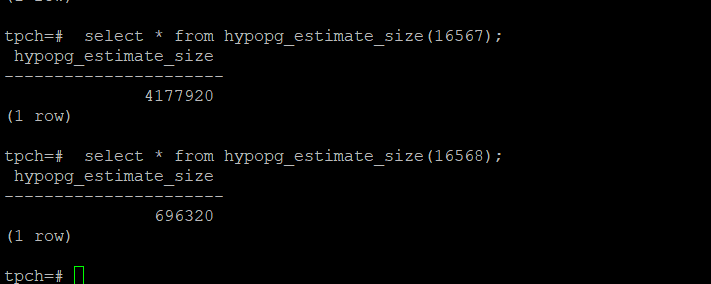
select \* from hypopg\_display\_index();



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

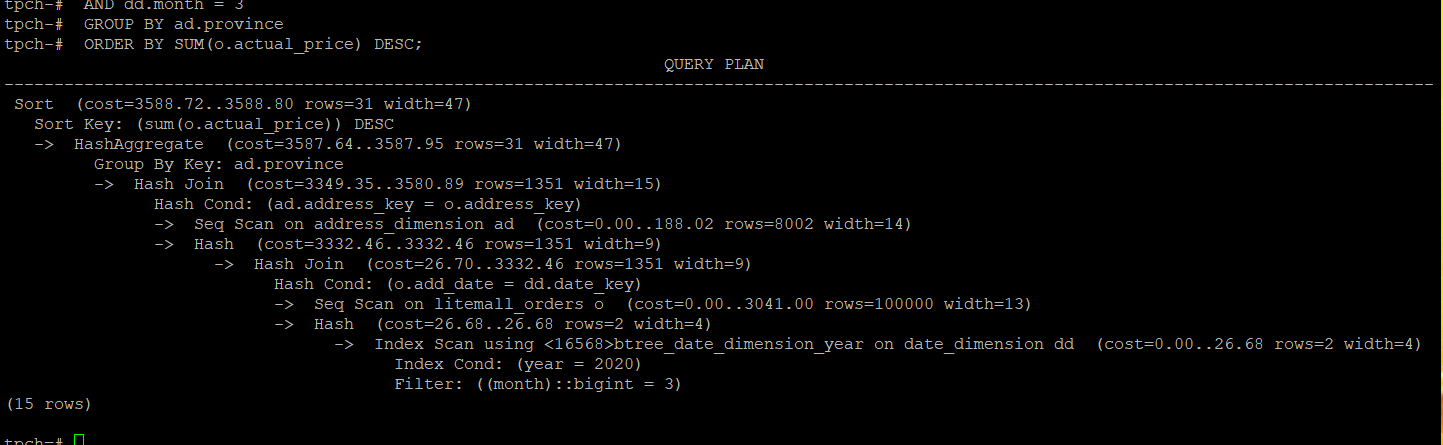
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

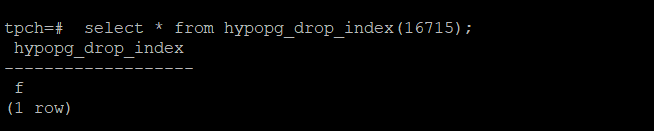
GROUP BY ad.province

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



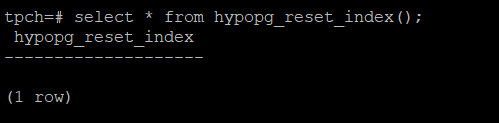
6. 删除某一个索引虚拟列，将执行结果截图。

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



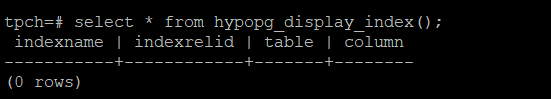
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

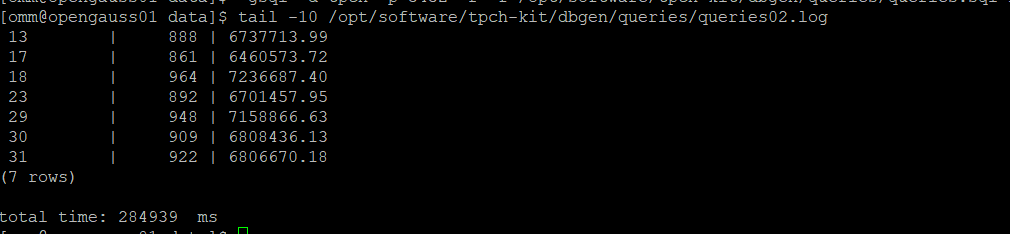
select \* from hypopg\_display\_index();



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：

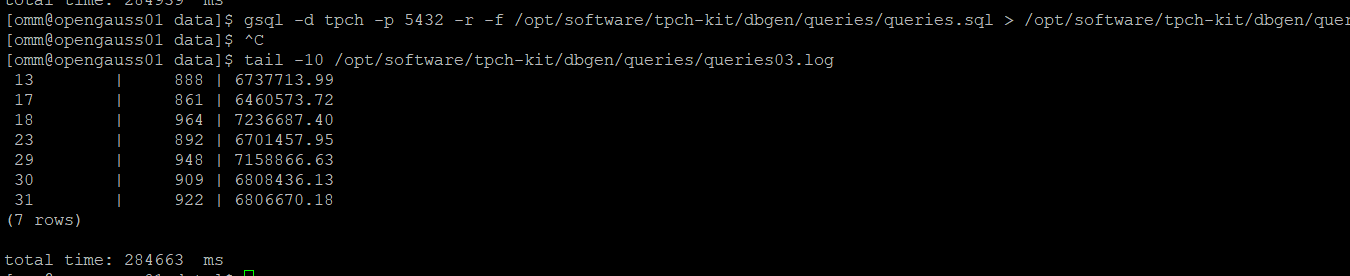
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log



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？

gsql：是执行Greenplum数据库中SQL命令的客户端工具。

-d tpch：指定要连接的数据库名称为tpch。

-p 5432：指定要连接的数据库的端口号为5432。

-r：表示将查询结果以文本形式输出。

-f /opt/software/tpch-kit/dbgen/queries/queries.sql：指定要执行的查询文件的路径为/opt/software/tpch-kit/dbgen/queries/queries.sql。

> /opt/software/tpch-kit/dbgen/queries/queries03.log：将查询结果输出重定向到/opt/software/tpch-kit/dbgen/queries/queries03.log文件中。

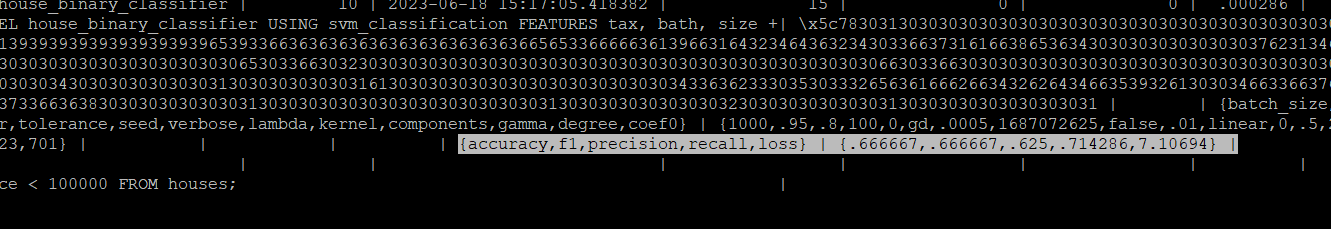
实践思考题2：索引的使用，对于执行SQL有什么好处？除了使用索引和参数外，还有哪些方面可以对数据库进行优化？

索引在执行SQL查询时有以下好处：它们可以提高查询性能，加速查询操作，减少磁盘I/O次数，使数据库能够更快地定位和检索符合查询条件的数据，从而提高查询的执行速度。除了使用索引和参数，还有其他方面可以对数据库进行优化，如合理的表结构设计、正确的数据类型选择、适当的查询重写和优化、合理的内存和磁盘配置、并发控制和事务管理等。综合考虑这些因素，可以进一步优化数据库的性能和效率。

# 关卡四、openGauss的DB4AI特性应用

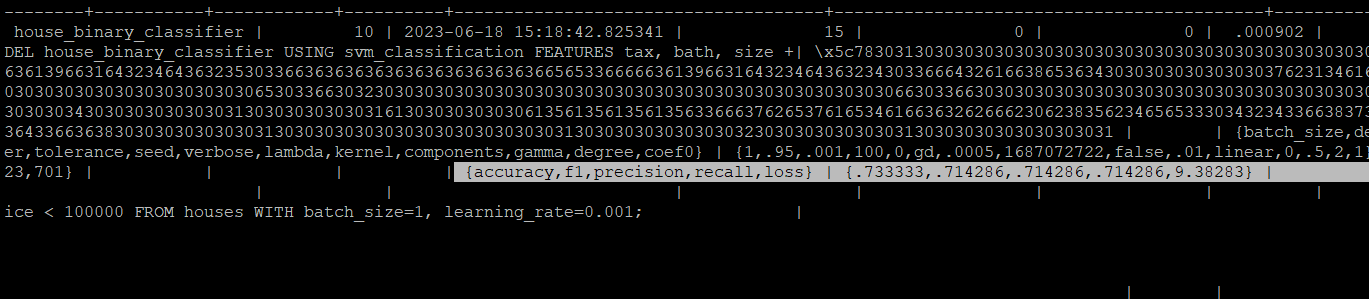
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



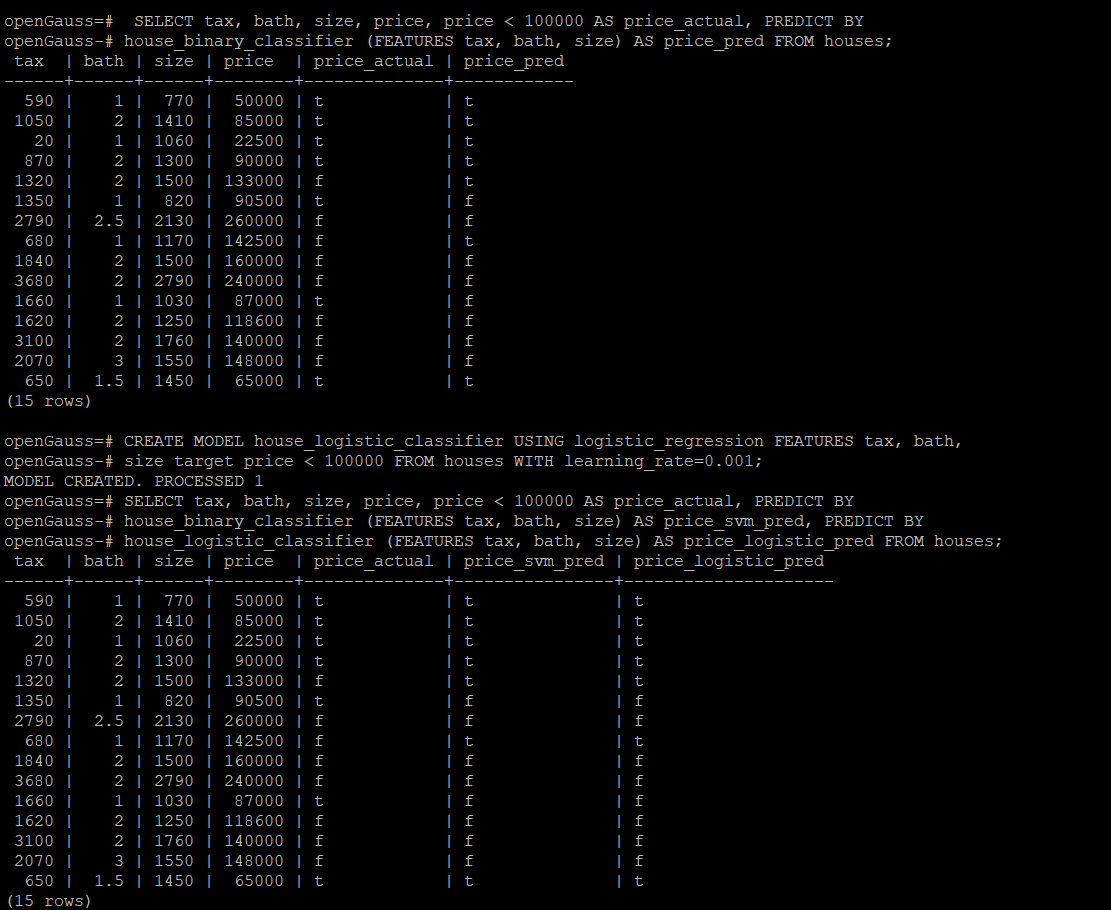
任务二：观察新模型的信息，将执行结果截图。

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是机器学习中两种不同类型的预测模型。分类模型用于预测离散的类别或标签，如判断邮件是否是垃圾邮件。而回归模型则用于预测连续的数值，如预测房屋价格。区别主要在于预测目标和输出结果的性质，分类模型关注离散的类别，而回归模型关注连续的数值。

实践思考题2：什么是SVM算法？

SVM算法的中文名称是支持向量机算法，是一种用于分类和回归分析的机器学习算法。它是一种非概率性的监督学习方法，其目标是通过找到一个最优的超平面（或者在高维空间中的超平面）来将不同类别的样本实例分隔开。

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？

分类问题的评价指标常见有以下几种：

1. 准确率：准确率是最常用的评价指标之一，表示分类正确的样本数与总样本数之比。即准确率 = (TP + TN) / (TP + TN + FP + FN)，准确率越高，模型的分类效果越好。
2. 精确率：精确率是指模型预测为正例的样本中，真正例的比例。即精确率 = TP / (TP + FP)。精确率衡量了模型预测为正例的准确性，较高的精确率表示模型将负例误判为正例的概率较低。
3. 召回率：召回率是指模型正确预测为正例的样本占真实正例的比例。即召回率 = TP / (TP + FN)。召回率衡量了模型对正例的查全率，较高的召回率表示模型更能识别出真正的正例。
4. F1值：F1值综合了精确率和召回率，是精确率和召回率的调和平均值。即 F1 = 2 \* (精确率 \* 召回率) / (精确率 + 召回率)。F1值在评估分类器的性能时较为常用，尤其在正负样本不平衡的情况下。

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：均方误差是预测值与真实值之间差异的平方的平均值。MSE = (1/n) \* Σ(yᵢ - ȳ)²，其中 yᵢ 是真实值，ȳ 是预测值，n 是样本数量。MSE衡量了模型预测值与真实值之间的平均偏差，数值越小表示模型拟合效果越好。
2. 均方根误差：均方根误差是均方误差的平方根，即 RMSE = √MSE。RMSE与MSE相比更加直观，表示预测值与真实值之间的平均误差。
3. 平均绝对误差：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。MAE = (1/n) \* Σ|yᵢ - ȳ|。MAE衡量了模型预测值与真实值之间的平均绝对偏差，数值越小表示模型的预测精度越高。
4. 决定系数：决定系数衡量了模型对因变量变异性的解释程度。R²的取值范围在0到1之间，越接近1表示模型的拟合效果越好，能够更好地解释因变量的变异。