openGauss AI特性创新实践课



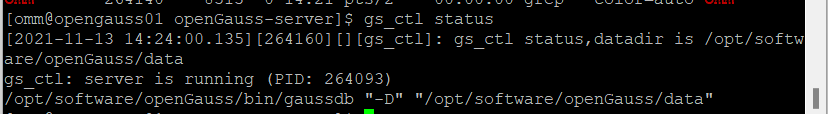
华为技术有限公司

# 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：

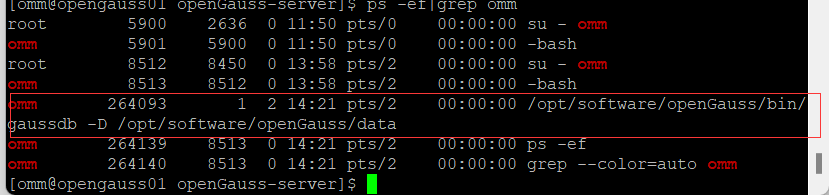
任务一：数据库状态验证

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？

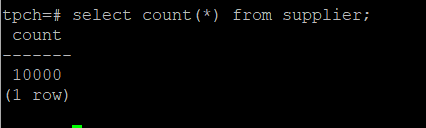
答：通过源码安装可以定制数据库，能更灵活地满足需求。

# 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：

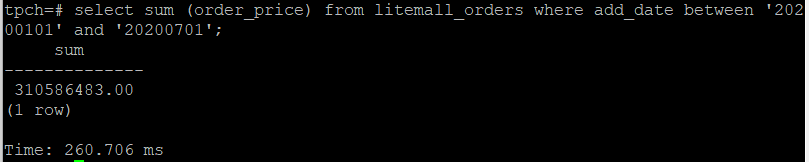
select count(\*) from supplier;;



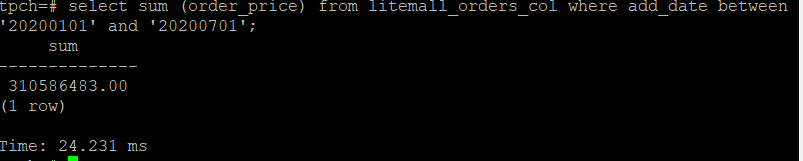
任务二：行存表与列存表执行效率对比

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';

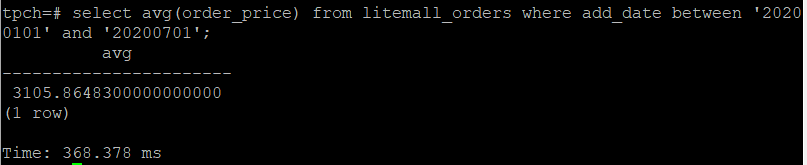


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

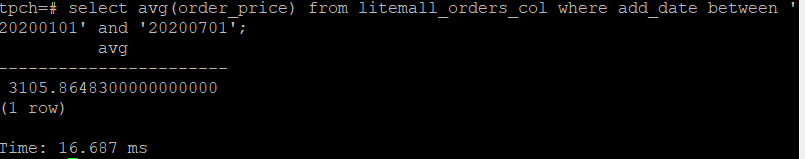


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

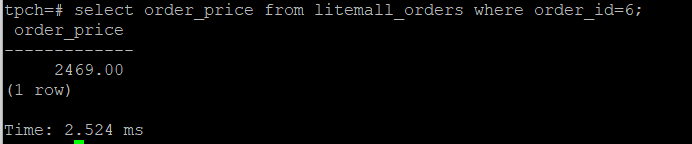


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

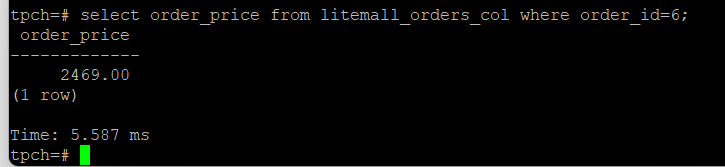


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

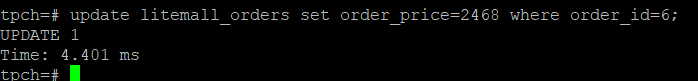


select order\_price from litemall\_orders\_col where order\_id=6;

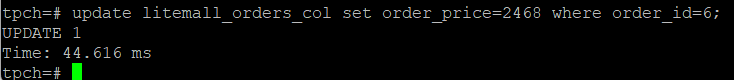


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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

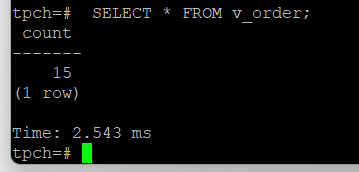
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



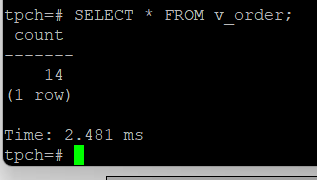
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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？

答：在GaussDB数据库中，表里的数据总是存放在一个或多个物理的数据文件中。相应的数据文件又分为多个固定大小的数据块，数据存放在这些数据块中。 在传统的行存储中，当需要读取某列时，需要将这列前面的所有列都进行deform，所以访问第一列和访问最后一列的成本实际上是不一样的，使得读取任意列的成本不一样，越靠后的列，成本越高。在添加有默认值的字段时，进行REWRITE表，需要对全表进行REWRITE。对于列存表中，当读取任一列时，都不需要读取排在前面列的数据，读取任意一列的成本是一样，但如果要读取多列，需要访问多个文件，访问的列越多，开销越大。每个CU存放的是具有相同格式的数据（即同一字段），使其更容易进行向量化和更高的压缩比。向量化的数据，在进行大批量数据访问和统计方面具有更高的效率。更高的压缩比也意味着，相同的数据，列存表占用的磁盘空间更少。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

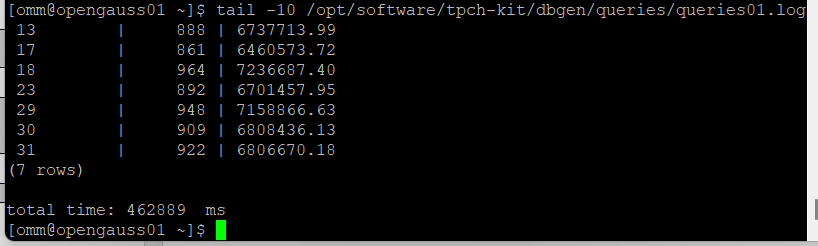
答：全量物化视图只支持全量更新。增量物化视图可实现异步更新功能，用户可通过执行语句把新增数据刷新到物化视图中。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

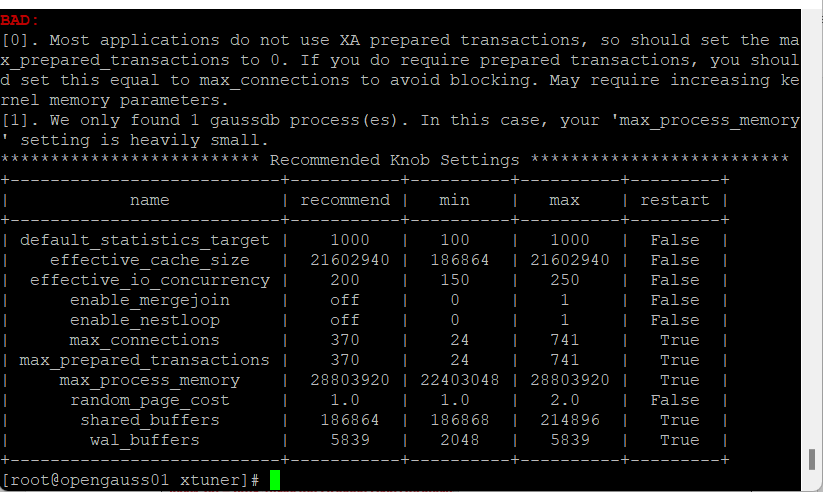
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

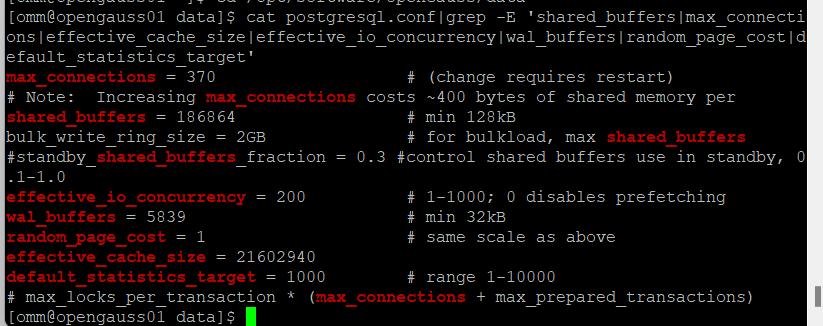
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

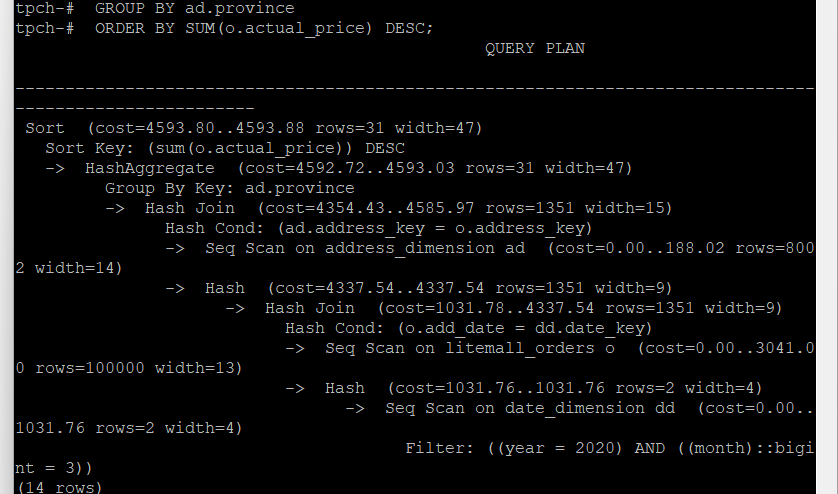
AND o.add\_date = dd.date\_key

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

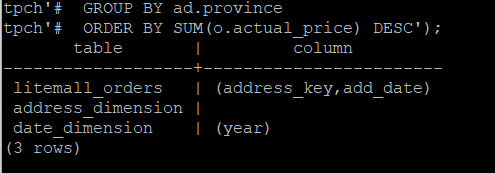
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

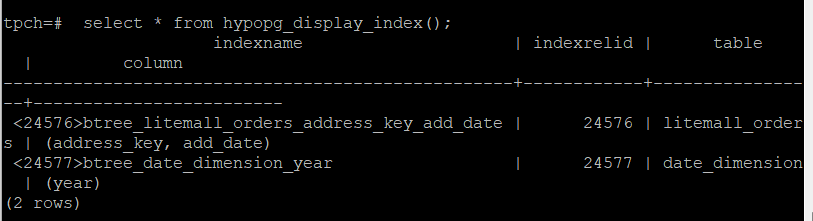
GROUP BY ad.province

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。

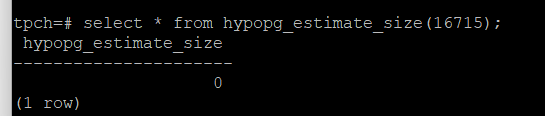
select \* from hypopg\_display\_index();

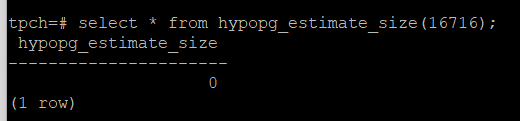


4. 获取索引虚拟列大小结果（单位为：字节），将执行结果截图。

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

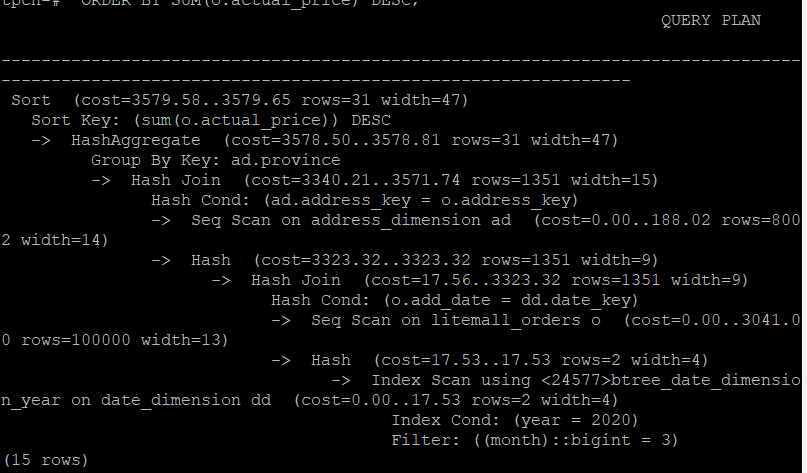
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

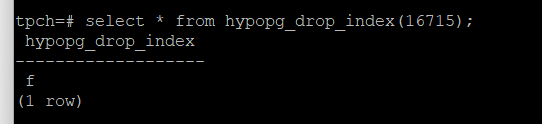
GROUP BY ad.province

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



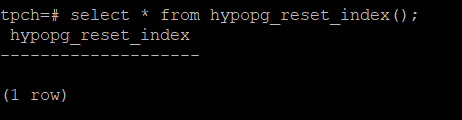
6. 删除某一个索引虚拟列，将执行结果截图。

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



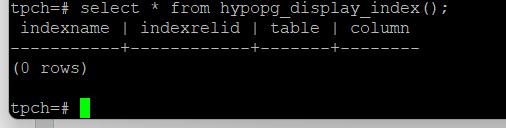
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

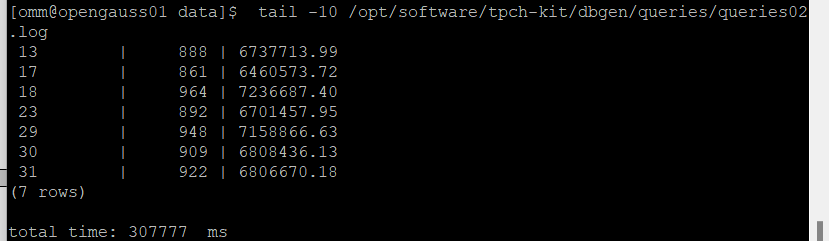
select \* from hypopg\_display\_index();



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？

答：对shared\_buffers， max\_connections，

effective\_cache\_size， effective\_io\_concurrency ,

wal\_buffers, random\_page\_cost,

default\_statistics\_target

等参数上进行了优化。这些参数都是影响数据库性能的一些参数，比如最大连接数，缓存大小等等，只有调到合适的数值才能发挥最大的性能。

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？

答：创建索引可以大大提高系统的性能。

第一，   通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。  
第二，   可以大大加快数据的检索速度，这也是创建索引的最主要的原因。  
第三，   可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义。  
第四，   在使用分组和排序子句进行数据检索时，同样可以显著减少查询中分组和排序的时间。  
第五，   通过使用索引，可以在查询的过程中，使用优化隐藏器，提高系统的性能。

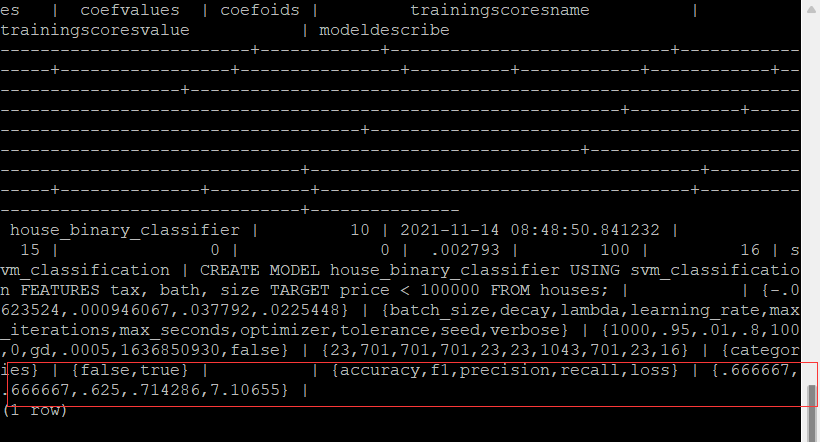
优化数据库的方法：

1. 选取最适用的字段属性；
2. 使用连接（JOIN）来代替子查询(Sub-Queries)；
3. 使用联合(UNION)来代替手动创建的临时表；
4. 优化查询语句；
5. 事务；

# 关卡四、openGauss的DB4AI特性应用

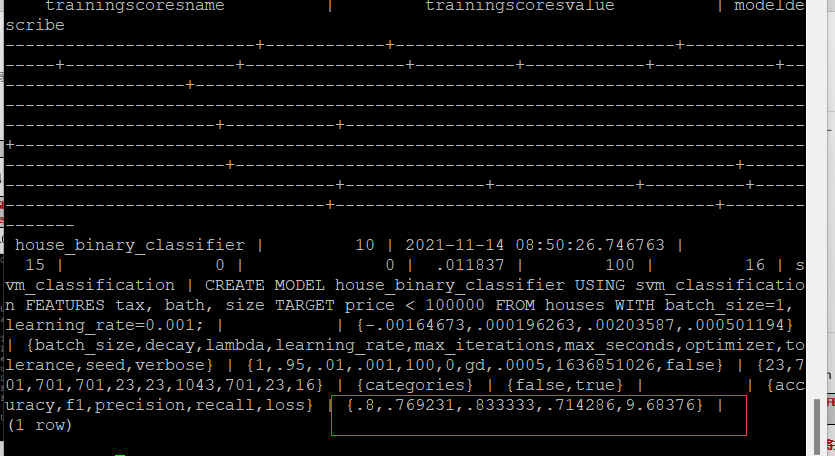
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



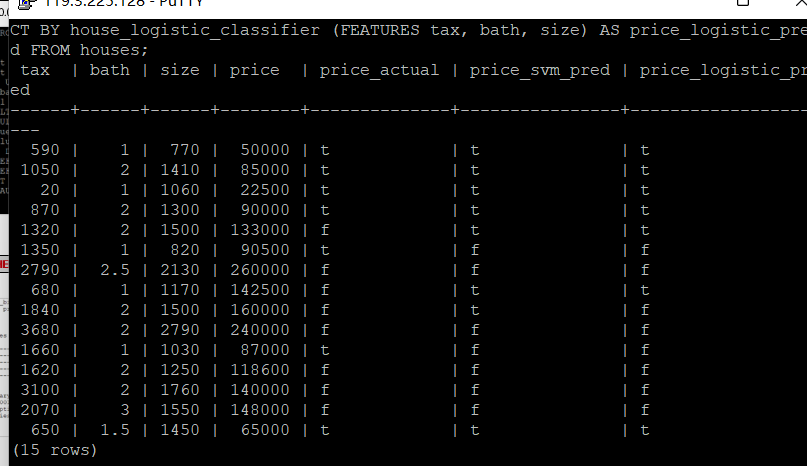
任务二：观察新模型的信息，将执行结果截图。

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：分类模型与回归模型的区别主要在于输出变量类型是否连续，预测连续值的为回归问题、预测离散值的为分类问题。

实践思考题2：什么是SVM算法？

答：支持向量机(support vector machine)是一种分类算法，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。通俗来讲，它是一种二类分类模型，其基本模型定义为特征空间上的间隔最大的线性分类器，即支持向量机的学习策略便是间隔最大化，最终可转化为一个凸二次规划问题的求解。

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：（1）准确率（Accuracy）：正确预测的样本数/所有样本数；

（2）精确率（Precision）：预测为正中的正确率；

（3）召回率（Recall）：正样本中被预测正确的概率；

（4）F1-Score：精确率和召回率的调和平均值；

（5）Macro-F1：宏平均；

（6）Micro-F1：微平均；

（7）ROC曲线及其AUC值：ROC曲线中横坐标代表FPR，纵坐标代表TPR，

其中， 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 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。

（8）P-R曲线：P-R 曲线其横坐标为 Recall， 纵坐标为 Precision， 其能帮助我们很

好的做出权衡。

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？

答：（1）MSE(Mean Square Error)：均方误差描述了样本真实值与预测值差方求和的平均

值，在机器学习中，利用均方误差最小化优化模型的方法称为**“最小二乘法”；**

（2）RMSE（Root Mean Square Error）：即对MSE开平方根；

（3）MAE（Mean Absolute Erroe）：真实值与预测值差的绝对值和求平均；

（4）MAPE(Mean Absolute Percentage Error)：求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；

（5）MASE（Mean Scaled Percentage Error）：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。