openGauss AI特性创新实践课



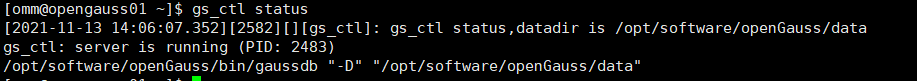
华为技术有限公司

# 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：

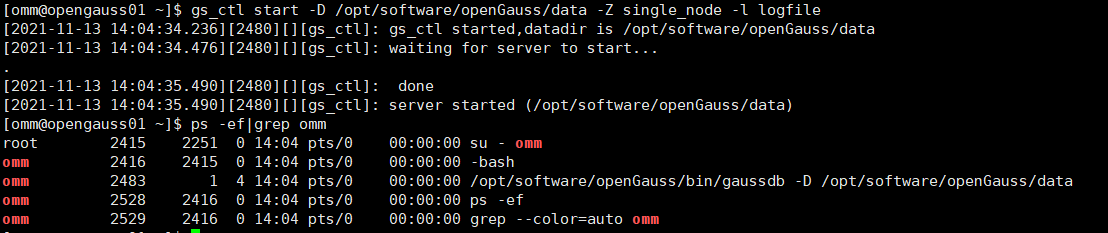
任务一：数据库状态验证

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？

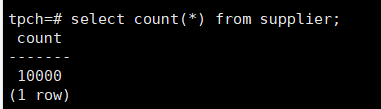
编译安装过程，可以设定参数，按照需求，进行安装，并且安装的版本，可以自己选择，灵活性比较大。

# 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：

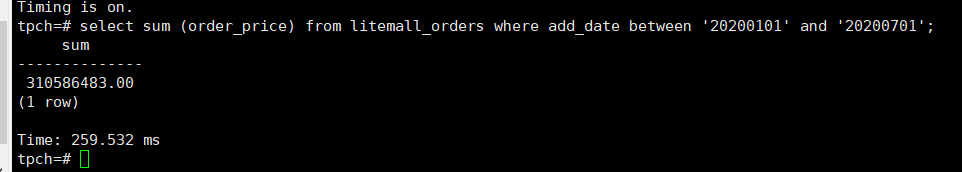
select count(\*) from supplier;;



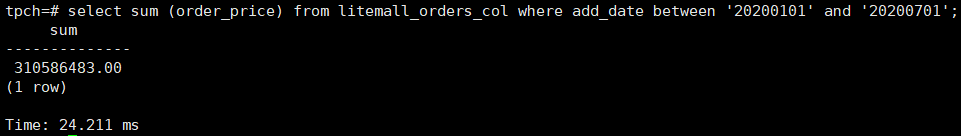
任务二：行存表与列存表执行效率对比

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';

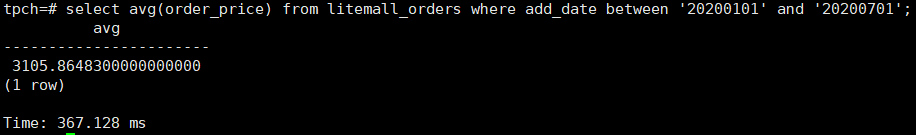


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

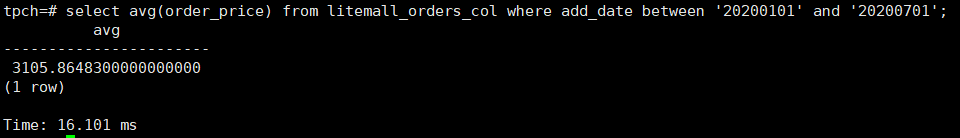


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

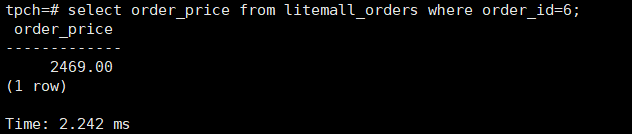


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

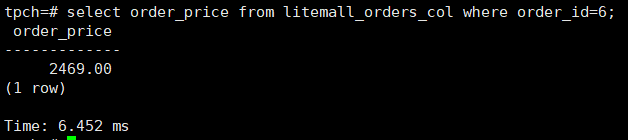


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

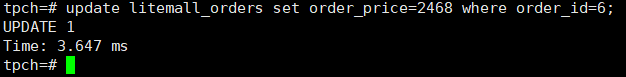


select order\_price from litemall\_orders\_col where order\_id=6;

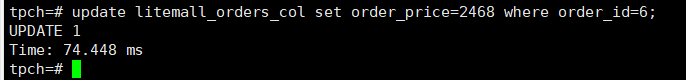


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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

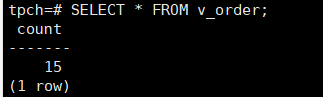
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



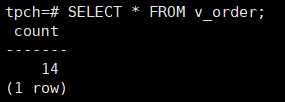
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



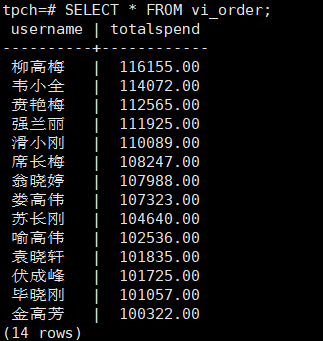
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

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？

行存表是将数据以行的形式存储，列存表是将数据以列的形式存储，存储结构不同导致在增删改查时处理不同类型时执行时间不同。在执行以id或某种值查询一整行或多个列时，行存表比较块，在执行一个列的查询时，列存表比较快。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

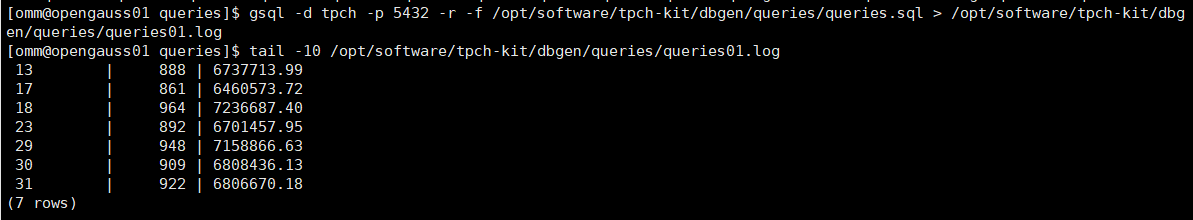
全量物化视图仅支持对已创建的物化视图进行全量更新，而不支持进行增量更新。创建全量物化视图语法和CREATE TABLE AS语法类似。  
 增量物化视图可以对物化视图增量刷新，需要用户手动执行语句完成对物化视图在一段时间内的增量数据刷新。与全量创建物化视图的不同在于目前增量物化视图所支持场景较小。目前物化视图创建语句仅支持基表扫描语句或者UNION ALL语句。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

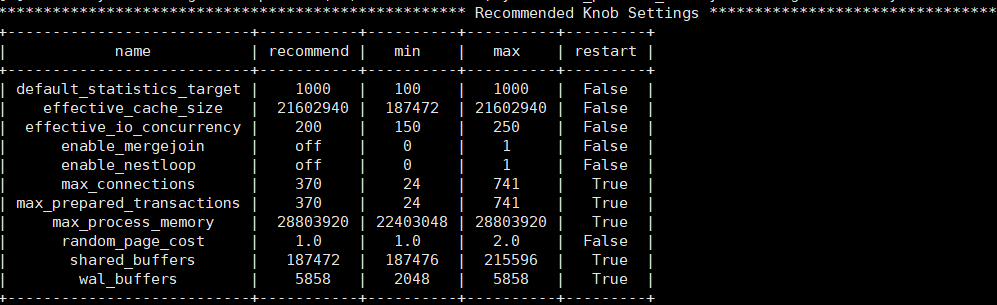
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

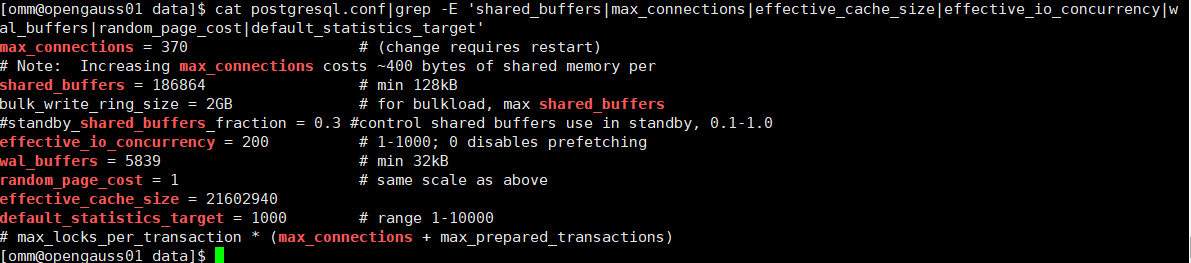
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

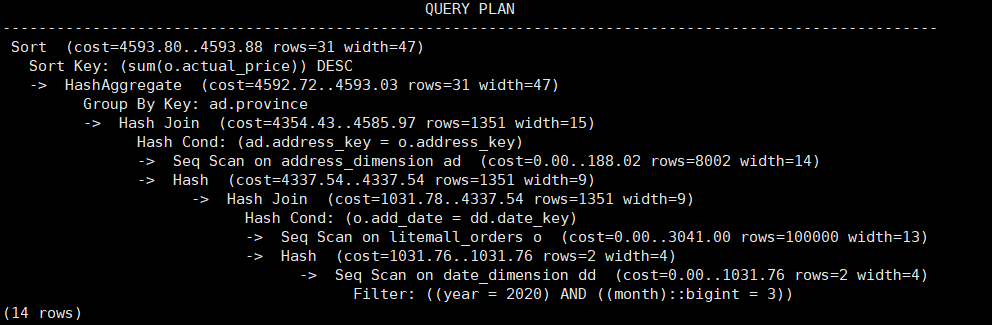
AND o.add\_date = dd.date\_key

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

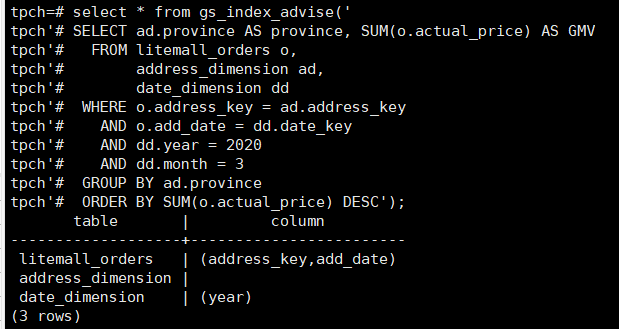
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

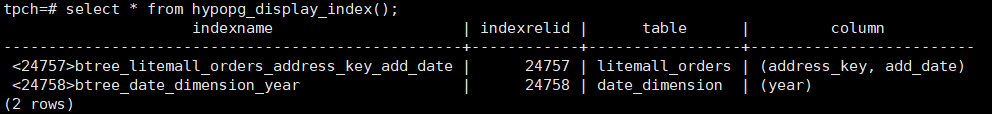
GROUP BY ad.province

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。

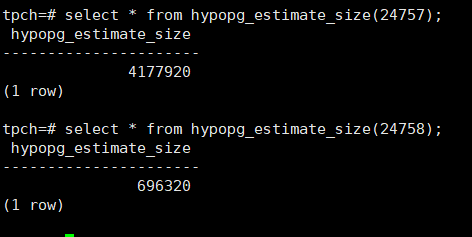
select \* from hypopg\_display\_index();



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(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

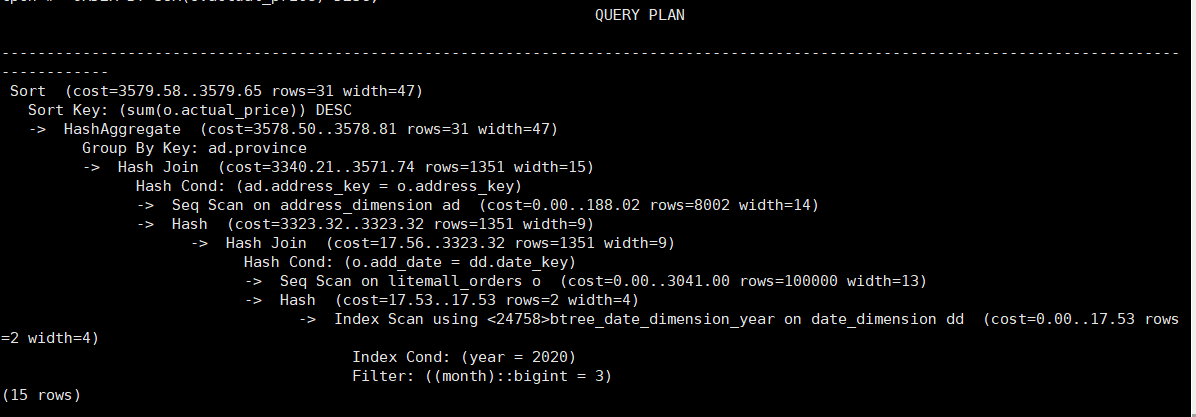
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

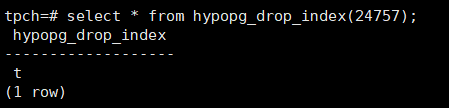
GROUP BY ad.province

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



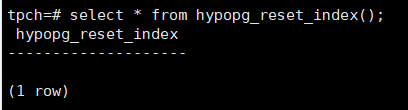
6. 删除某一个索引虚拟列，将执行结果截图。

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



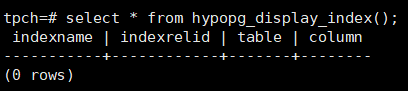
7. 删除所有索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

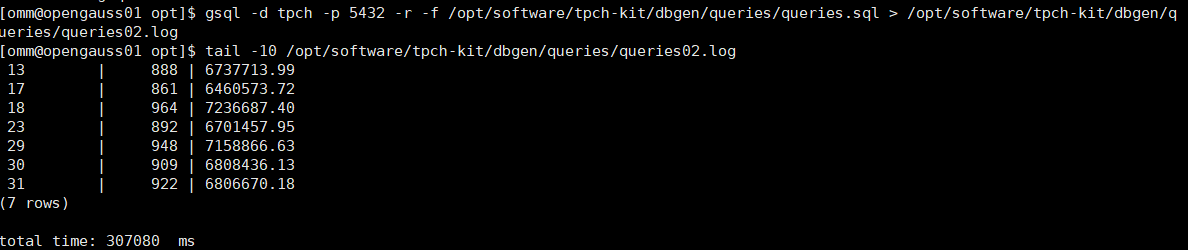
select \* from hypopg\_display\_index();



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

优化之后 能够提高数据库的运行效率

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？

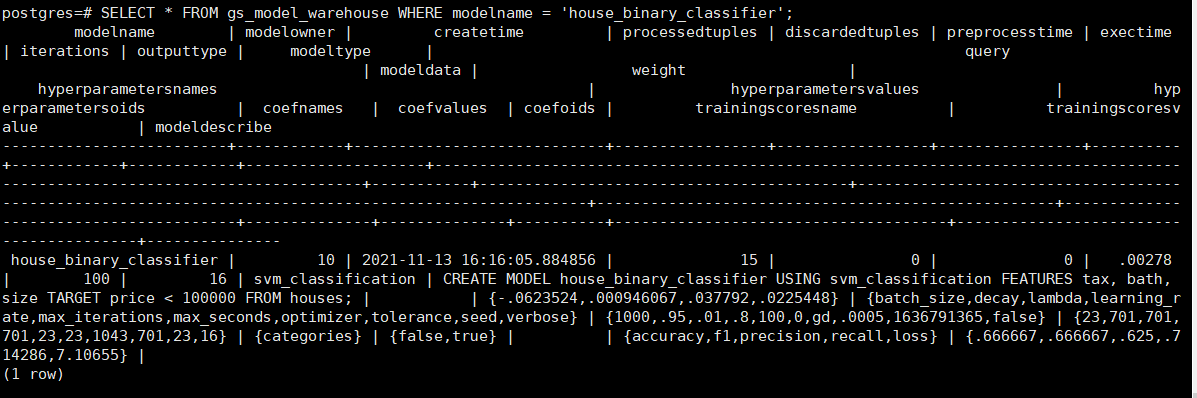
索引可以使数据库的增删改查的效率提高，但是索引会有代价。

将查询语句进行优化，减少时间和空间的开销。

# 关卡四、openGauss的DB4AI特性应用

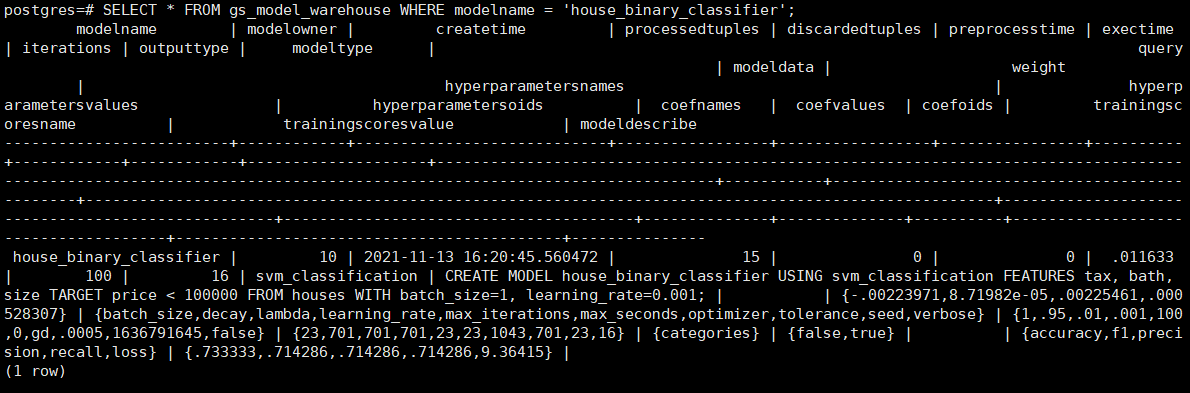
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



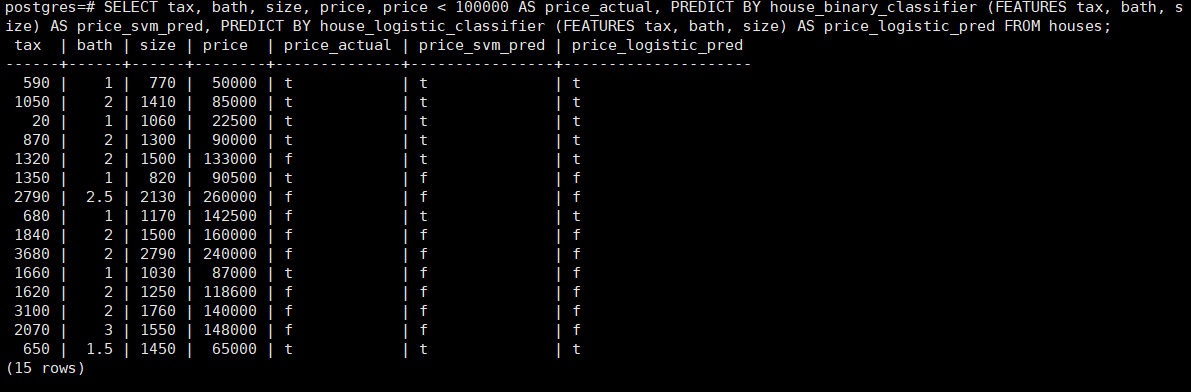
任务二：观察新模型的信息，将执行结果截图。

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？

分类和回归的区别在于输出变量的类型。

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。

分类模型和回归模型本质一样，分类模型可将回归模型的输出离散化，回归模型也可将分类模型的输出连续化

实践思考题2：什么是SVM算法？

支持向量机（support vector machines）是一个二分类的分类模型（或者叫做分类器）。如图：

它分类的思想是，给定给一个包含正例和反例的样本集合，svm的目的是寻找一个超平面来对样本根据正例和反例进行分割

SVM的核心思想是**尽最大努力使分开的两个类别有最大间隔，这样才使得分隔具有更高的可信度。而且对于未知的新样本才有很好的分类预测能力**

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？

TP、True Positive   真阳性：预测为正，实际也为正

FP、False Positive  假阳性：预测为正，实际为负

FN、False Negative 假阴性：预测与负、实际为正

TN、True Negative 真阴性：预测为负、实际也为负

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？

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。这个指标是对绝对误差损失的预期值。MAE对极端值比较敏感，即MAE 对异常值更加稳健，因为它不使用平方。

MAE=\frac{1}{n}\sum_{i=1}^{n}|y_{i}-\hat{y_{i}}|

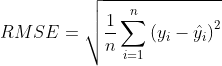
均方误差（MSE）

（Mean Squared Error）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。该指标对应于平方(二次)误差的期望。是线性回归的损失函数，在线性回归的时候我们的目的就是让这个损失函数最小。受到异常值的影响很大。

MSE=\frac{1}{n}\sum_{i=1}^{n}{(y_{i}-\hat{y_{i}})}^{2}

均方根误差（RMSE）

（Root Mean Squard Error）均方根误差就是在均方误差的基础上再开方。该指标对应于平方(二次)误差的期望。受到异常值的影响很大，使用时应先剔除异常值。



平均绝对百分比误差(MAPE)

（Mean Absolute Percentage Error）这个指标是对相对误差损失的预期值。所谓相对误差，就是绝对误差和真值的百分比。

MAPE=\frac{1}{n}\sum_{i=1}^{n}\frac{||y_{i}-\hat{y_{i}}||}{||y_{i}||}

均方误差对数(MSLE)

（Mean Squared Log Error）该指标对应平方对数(二次)差的预期。当数据当中有少量的值和真实值差值较大的时候，使用log函数能够减少这些值对于整体误差的影响。

MSLE=\frac{1}{n}\sum_{i=1}^{n}{(log(1+y_{i})-log(1+\hat{y_{i}}))}^{2}

中位绝对误差(MedAE)

（Median Absolute Error）通过取目标和预测之间的所有绝对差值的中值来计算损失。

MedAE=median(|y_{1}-\hat{y_{1}}|,\dots,|y_{n}-\hat{y_{n}}|)

R Squared

又叫可决系数(coefficient of determination)/拟合优度，取值范围为0~1，反映的是自变量 对因变量 的变动的解释的程度。越接近于1，说明模型拟合得越好。

R^{2}=1-\frac{MSE(y,\hat{y})}{Var(y))}=1-\frac{\sum_{i=1}^{n}{(y_{i}-\hat{y_{i}})}^{2}}{\sum_{i=1}^{n}{(y_{i}-\bar{y}})^{2}}=\frac{SSE}{SST}=1-\frac{SSR}{SST}

SST={\sum_{i=1}^{n}{(y_{i}-\bar{y}})^{2}表示的是 y 的变动的程度，正比于方差；

SSR={\sum_{i=1}^{n}(y_{i}-\hat{y_{i}})^{2}表示的是模型和真实值的残差；

SSE={\sum_{i=1}^{n}(\hat{y_{i}}-\bar{y})^{2}表示的是模型对 y 的变动的预测。

SST=SSR+SSE.