openGauss AI特性创新实践课



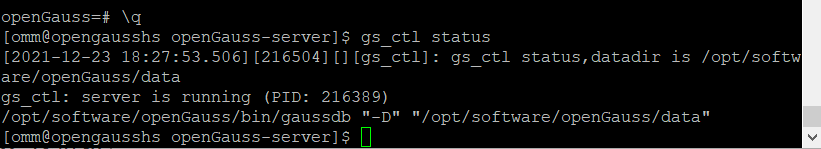
华为技术有限公司

# 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：

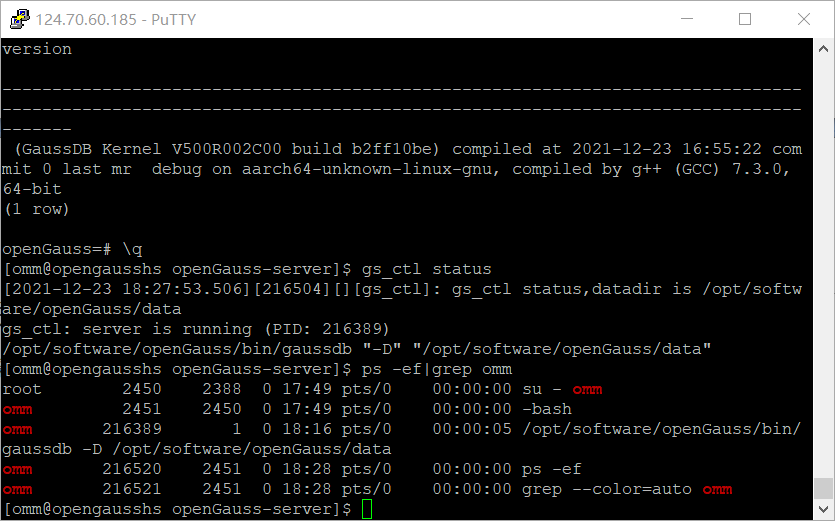
任务一：数据库状态验证

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？

1：满足不同的运行平台， Linux发行版本众多，但是每个版本采用的软件或者内核版本都不一样，而二进制包所依赖的环境不一定能够正常运行，所以大部分软件直接提供源码

2：方便定制，满足不同的需求，很多时候我们所需要的软件都是可以定制的，我需要什么就安装什么，大多数二进制代码都是一键装全，所以自由度并不高

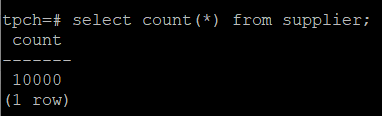
3：方便运维、开发人员维护，源码是可以打包二进制的，但是对于这个软件的打包都会有一份代价不小的额外工作，包括维护，所以如果是源码的话，软件产商会直接维护，但是如果是二进制的话，一般都是Linux发行商提供

# 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：

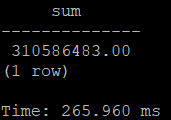
select count(\*) from supplier;;



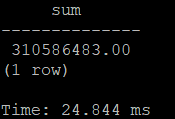
任务二：行存表与列存表执行效率对比

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';

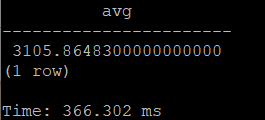


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';



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

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

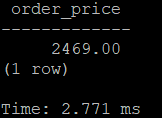


selectavg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

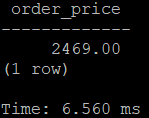


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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



selectorder\_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。

updatelitemall\_orders set order\_price=2468 where order\_id=6;

E:\newUser\Documents\Tencent Files\3044243151\FileRecv\MobileFile\Image\GFRQZO5MZX}7`][00H12@I8.png

updatelitemall\_orders\_col set order\_price=2468 where order\_id=6;

E:\newUser\Documents\Tencent Files\3044243151\FileRecv\MobileFile\Image\2X(]W_W`2R~KV)M1WC8E3EK.png

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？

行存储的写入是一次性完成，消耗的时间比列存储少，并且能够保证数据的完整性，缺点是数据读取过程中会产生冗余数据，如果只有少量数据，此影响可以忽略;数量大可能会影响到数据的处理效率。

列存储在写入效率、保证数据完整性上都不如行存储，它的优势是在读取过程，不会产生冗余数据，这对数据完整性要求不高的大数据处理领域，比如互联网，犹为重要。查询过程中，可针对各列的运算并发执行(SMP)，\*\*\*在内存中聚合完整记录集，\*\*\*可能降低查询响应时间;可在数据列中高效查找数据，无需维护索引(任何列都能作为索引)，查询过程中能够尽量减少无关IO，避免全表扫描;因为各列独立存储，且数据类型已知，可以针对该列的数据类型、数据量大小等因素动态选择压缩算法，以提高物理存储利用率;如果某一行的某一列没有数据，那在列存储时，就可以不存储该列的值，这将比行式存储更节省空间。

**行式存储的适用场景：**

　　1、适合随机的增删改查操作;

　　2、需要在行中选取所有属性的查询操作;

　　3、需要频繁插入或更新的操作，其操作与索引和行的大小更为相关。

**列式存储的适用场景：**

　　1、OLAP类型的查询可能需要访问几百万甚至几十亿个数据行，

2、列族（column group，Bigtable系统中称为locality group）

实践思考题2：全量物化视图与增量物化视图有哪些差别？

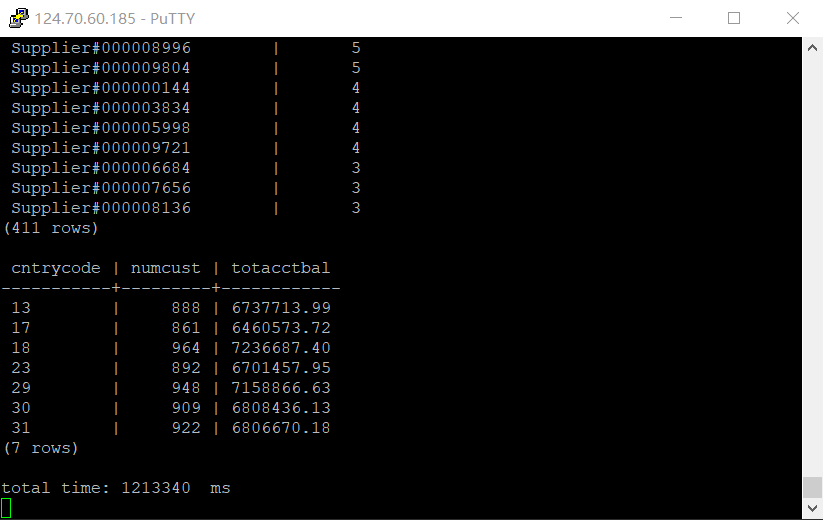
全量简单，但是数据量大；增量，相对复杂，要求对数据差异准确性高，对业务系统的性能不能有太大压力

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

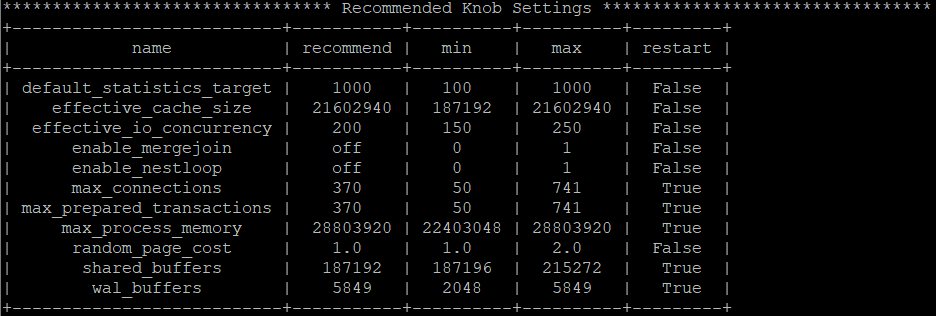
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

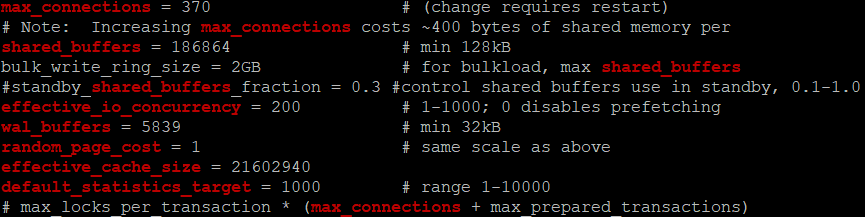
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



3.重启完成后，获取参数值：

cd /opt/software/openGauss/data

catpostgresql.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\_dimensiondd

WHERE o.address\_key = ad.address\_key

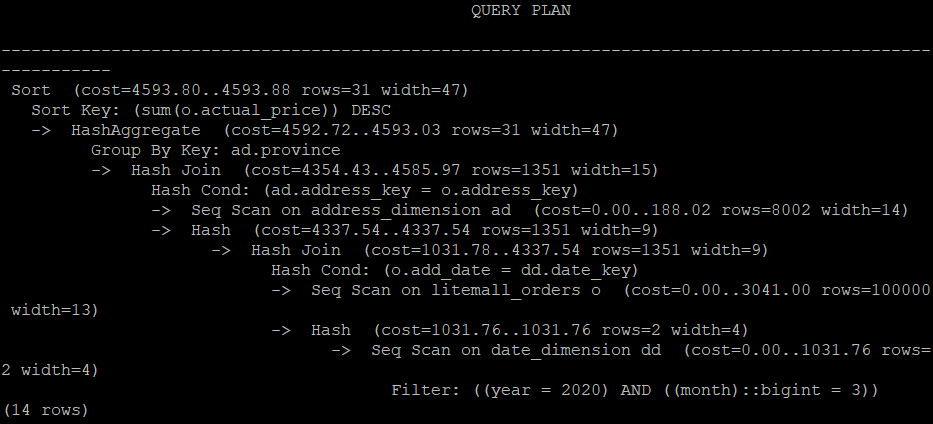
AND o.add\_date = dd.date\_key

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\_dimensiondd

WHERE o.address\_key = ad.address\_key

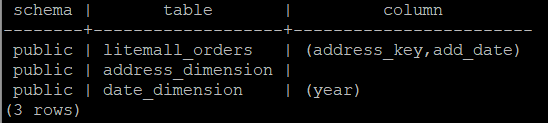
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

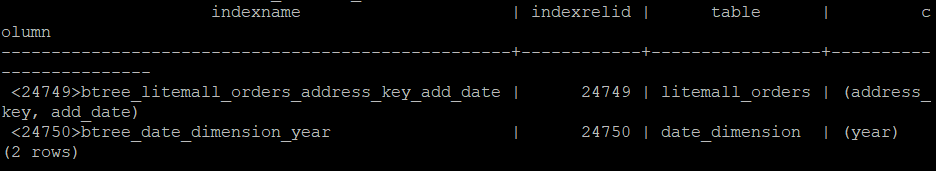
GROUP BY ad.province

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。

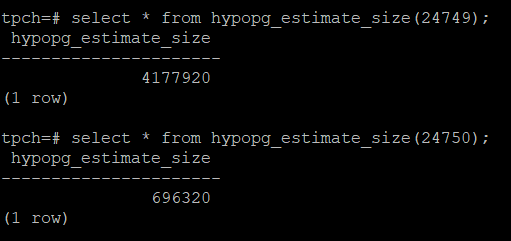
select \* from hypopg\_display\_index();



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\_dimensiondd

WHERE o.address\_key = ad.address\_key

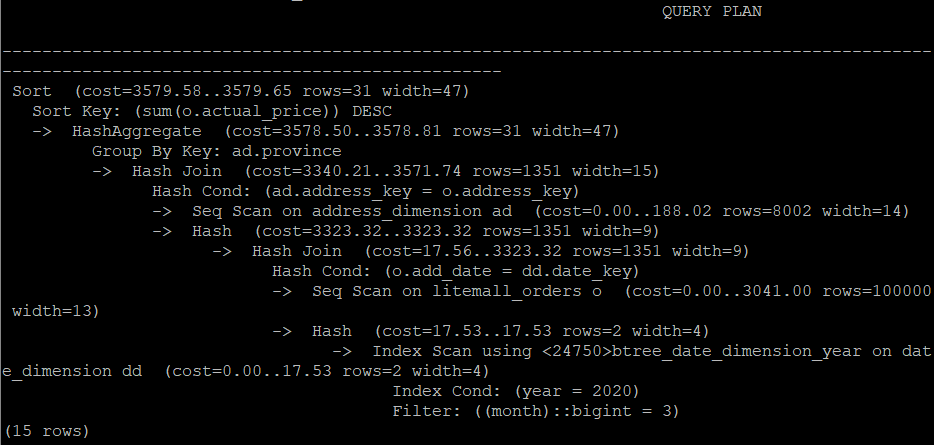
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

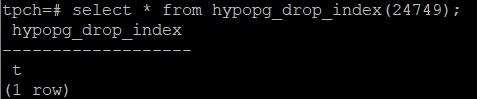
GROUP BY ad.province

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



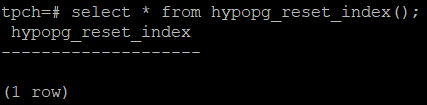
6. 删除某一个索引虚拟列，将执行结果截图。

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



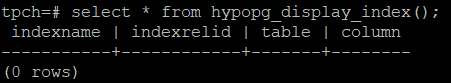
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

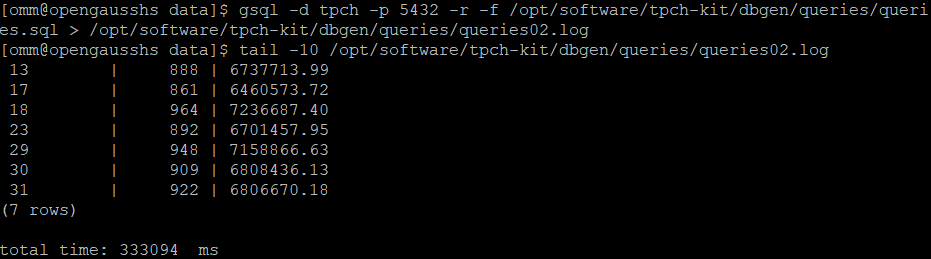
select \* from hypopg\_display\_index();



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？

**shared\_buffers**

**max\_connections**

**effective\_cache\_size**

**effective\_io\_concurrency**

**wal\_buffers**

**random\_page\_cost**

**default\_statistics\_target**

TPC-H 是一个面向分析型业务(AP)的基准测试，它由一系列热点查询组成，这些热点查询都是高度复杂的，因此执行时间往往都比较长。默认TPC-H数据库的表缺少索引，数据库的参数并没有做任何优化，因此执行效率会比较差。使用openGauss的参数自调优(X-Tuner：gs\_xtuner)功能，对数据库进行参数优化，以提升数据库运行性能

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？

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](http://lib.csdn.net/base/mysql)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。

优化目标

1.减少 IO 次数

2.降低 CPU 计算

优化方法

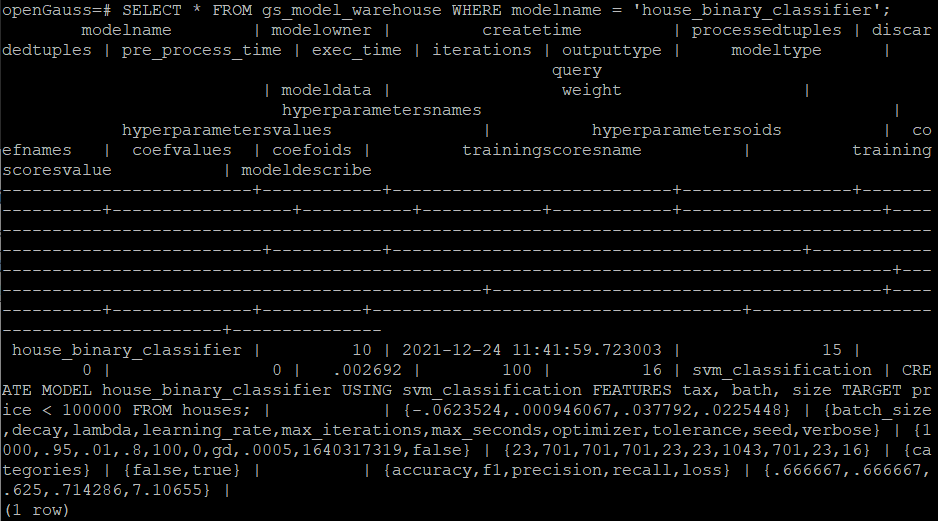
1.改变 SQL 执行计划

优化表结构

# 关卡四、openGauss的DB4AI特性应用

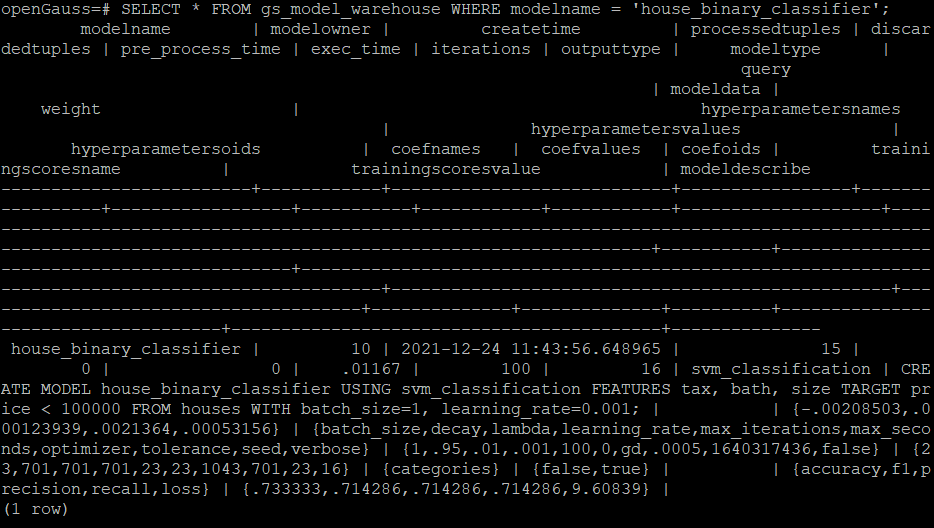
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



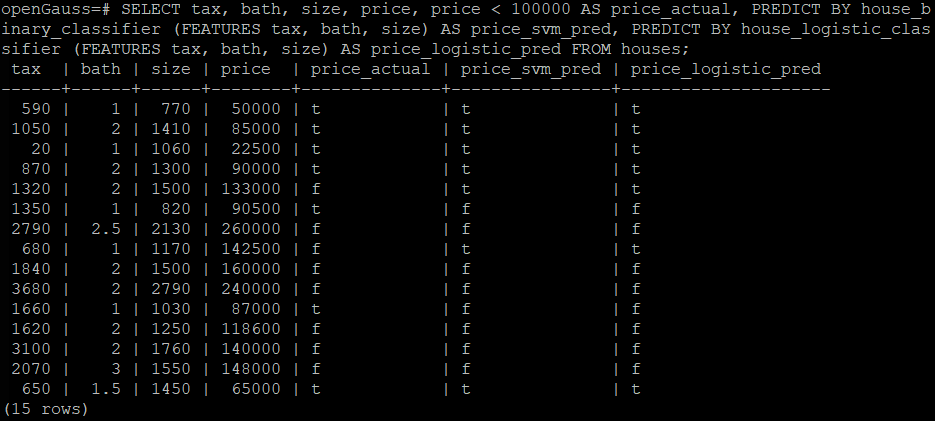
任务二：观察新模型的信息，将执行结果截图。

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。

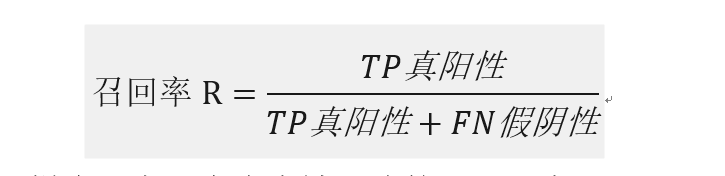
实践思考题2：什么是SVM算法？

支持向量机（support vector machines, 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。SVM的的学习策略就是间隔最大化，可形式化为一个求解凸二次规划的问题，也等价于正则化的合页损失函数的最小化问题。SVM的的学习算法就是求解凸二次规划的最优化算法。

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？

精确率和召回率





F1分数（F1 Score），是统计学中用来衡量二分类模型精确度的一种指标。它同时兼顾了分类模型的精确率和召回率。F1分数可以看作是模型精确率和召回率的一种加权平均，它的最大值是1，最小值是0。

Micro-F1和Macro-F1。在第一个多标签分类任务中，可以对每个“类”，计算F1，显然我们需要把所有类的F1合并起来考虑。

这里有两种合并方式：

第一种计算出所有类别总的Precision和Recall，然后计算F1。

第二种方式是计算出每一个类的Precison和Recall后计算F1，最后将F1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？

1.绝对误差与相对误差

2.MAE（平均绝对误差）又被称为 L1范数

优点：可以把绝对误差和相对误差里面正负相互抵消的问题去掉。

缺点：因为用了绝对值，所以会有一些点无法求导~

3.MSE(平均平方误差)

优点：解决了不光滑的问题。

缺点：MSE与我们的目标变量的量纲不一致

4.RMSE（平方根误差）

优点：解决了上述的缺点。

缺点：它是使用平均误差，而平均值对异常点（outliers）较敏感，如果回归器对某个点的回归值很不理性，那么它的误差则较大，从而会对RMSE的值有较大影响，即平均值是非鲁棒的。

5.Coefficient of determination (决定系数)

决定系数是相关系数的平方。 相关系数是用来描述两个变量之间的线性关系的，但决定系数的适用范围更广，可以用于描述非线性或者有两个及两个以上自变量的相关关系。

SST = SSR + SSE

缺点：均值对异常点（outliers）较敏感，如果样本中有一些异常值出现，会对以上指标的值有较大影响，即均值是非鲁棒的。