openGauss AI特性创新实践课



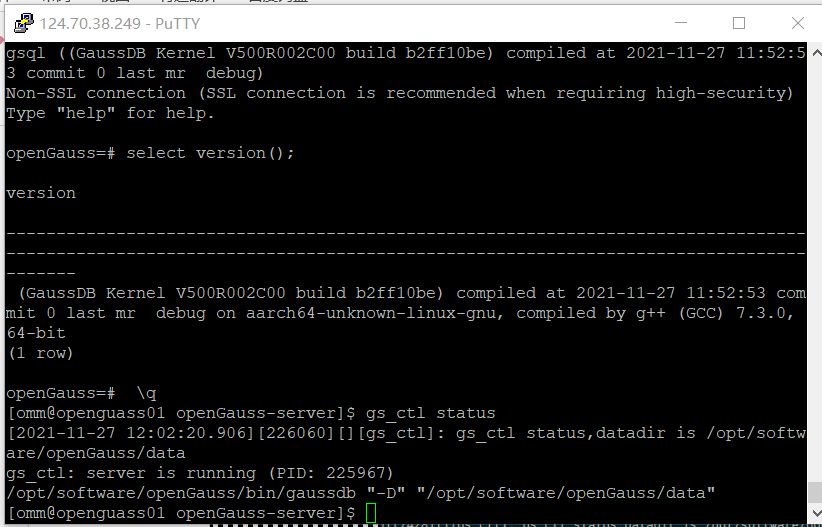
华为技术有限公司

# 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：

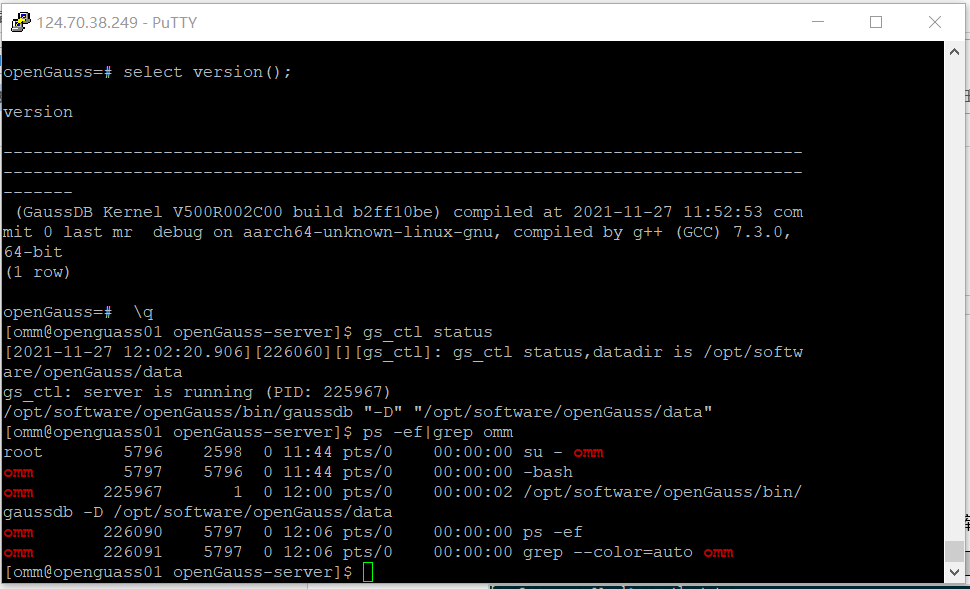
任务一：数据库状态验证

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？

源码编译可以自定义下载内容，避免下载了不必要的冗余，且根据自己需求定制。另外，也方便了运营人员管理维护。

# 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：

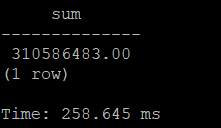
select count(\*) from supplier;;

C:\Users\admin\Documents\Tencent Files\153132495\Image\C2C\QPEO$943XO]Y`[P{]1$LF$E.png

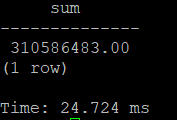
任务二：行存表与列存表执行效率对比

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';

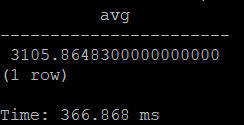


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

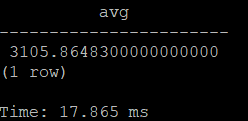


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

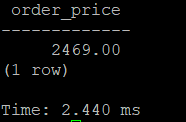


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

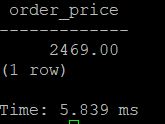


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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;

C:\Users\admin\Documents\Tencent Files\153132495\Image\C2C\}I9)G)IPGI(635~BMFO4_O8.png

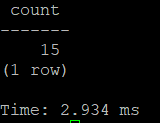
update litemall\_orders\_col set order\_price=2468 where order\_id=6;

C:\Users\admin\Documents\Tencent Files\153132495\Image\C2C\RBC8X`2CIC[HT__$$X$OGJU.png

任务三：物化视图的使用

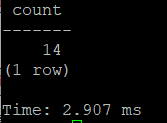
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

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，以行为形式组织存储，一行是一个tuple，存在一起。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。

列存，以列为形式组织存储，每列对应一个或一批文件。读取任一列的成本是一样的，但是如果要读取多列，需要访问多个文件，访问的列越多，开销越大。

所以根据任务需求不同，两种方法执行时间也不同。

在执行关注某几行的具体数据时，使用行存效率较高

在执行聚集函数相关的与列的数据操作时，使用列存效率较高。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

全量只能支持全量刷新，增量的话全量刷新和增量刷新都支持，只是目前运用较少。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

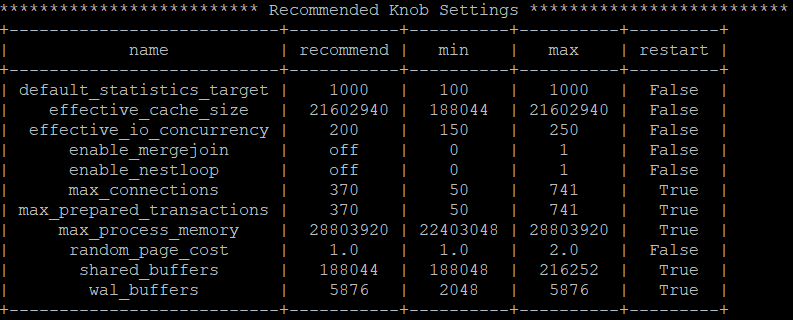
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

C:\Users\admin\Documents\Tencent Files\153132495\Image\C2C\~%KJ`POETN%F4H%X]_%_B]D.png

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

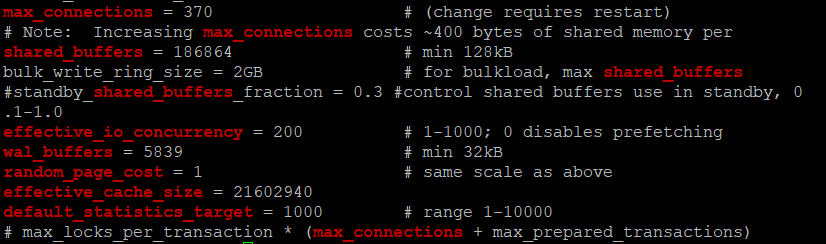
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

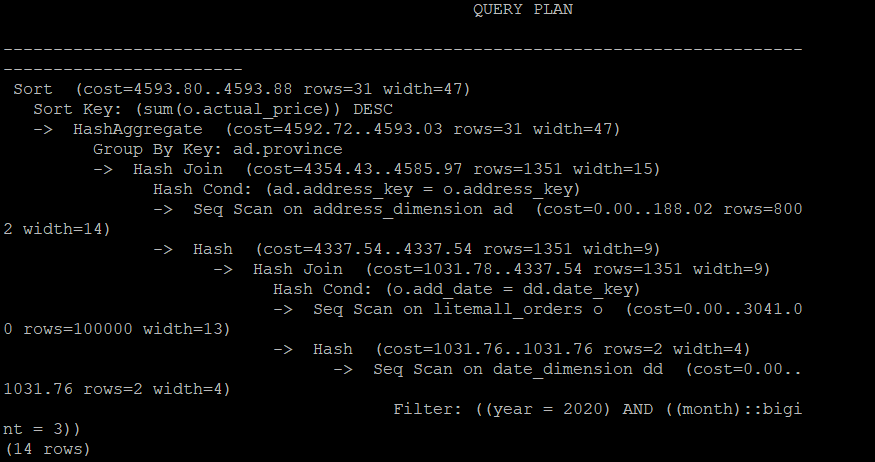
AND o.add\_date = dd.date\_key

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

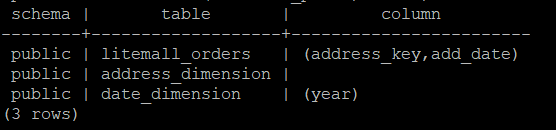
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

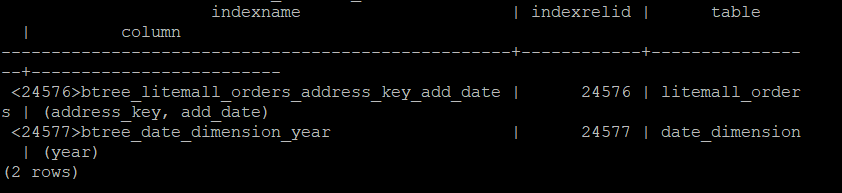
GROUP BY ad.province

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。

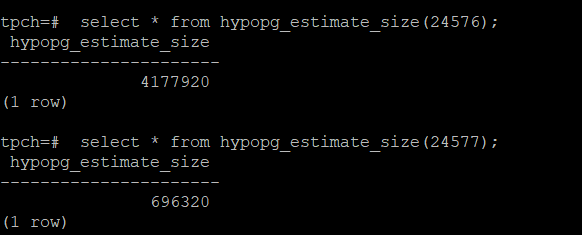
select \* from hypopg\_display\_index();



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

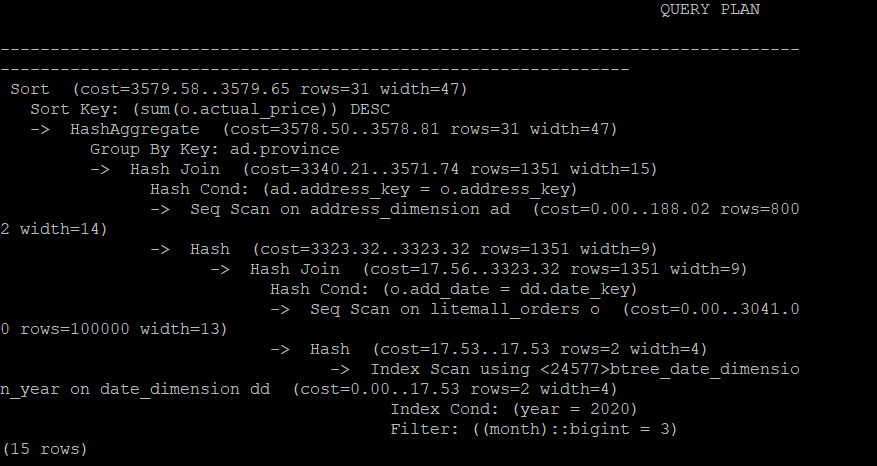
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

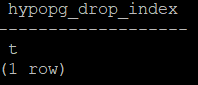
GROUP BY ad.province

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



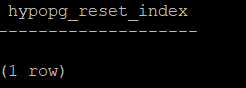
6. 删除某一个索引虚拟列，将执行结果截图。

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



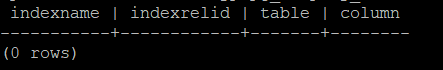
7. 删除某一个索引虚拟列，将执行结果截图。

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

C:\Users\admin\Documents\Tencent Files\153132495\Image\C2C\QD~U[FD48R}~}DPS7(88BVT.png

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？

在max\_connections,max\_prepared\_transactions,max\_process\_memory,shared\_buffers,wal\_buffers、effective\_io\_concurrency,effective\_cache\_size上进行了优化，

max\_connections：最大连接数。使数据库在服务器允许范围内提高效率；

max\_prepared\_transactions：设置可在准备状态中同时进行的事务的最大数量

max\_process\_memory：最大使用内存大小

shared\_buffers：数据页缓冲区。

wal buffer：预写日志(wal)缓冲区

effective\_cache\_size: 查询规划器确定实际缓存的数量，并有助于调整I/O缓存。如果effecve\_cache\_size的值太低，那么查询规划器可能会决定不使用某些索引，即使它们可以极大地提高查询速度。所以换句话说，这个值是PG用来估计索引的成本用的。

优化这些参数，使数据库连接量更大，处理速度更快，提高效率。

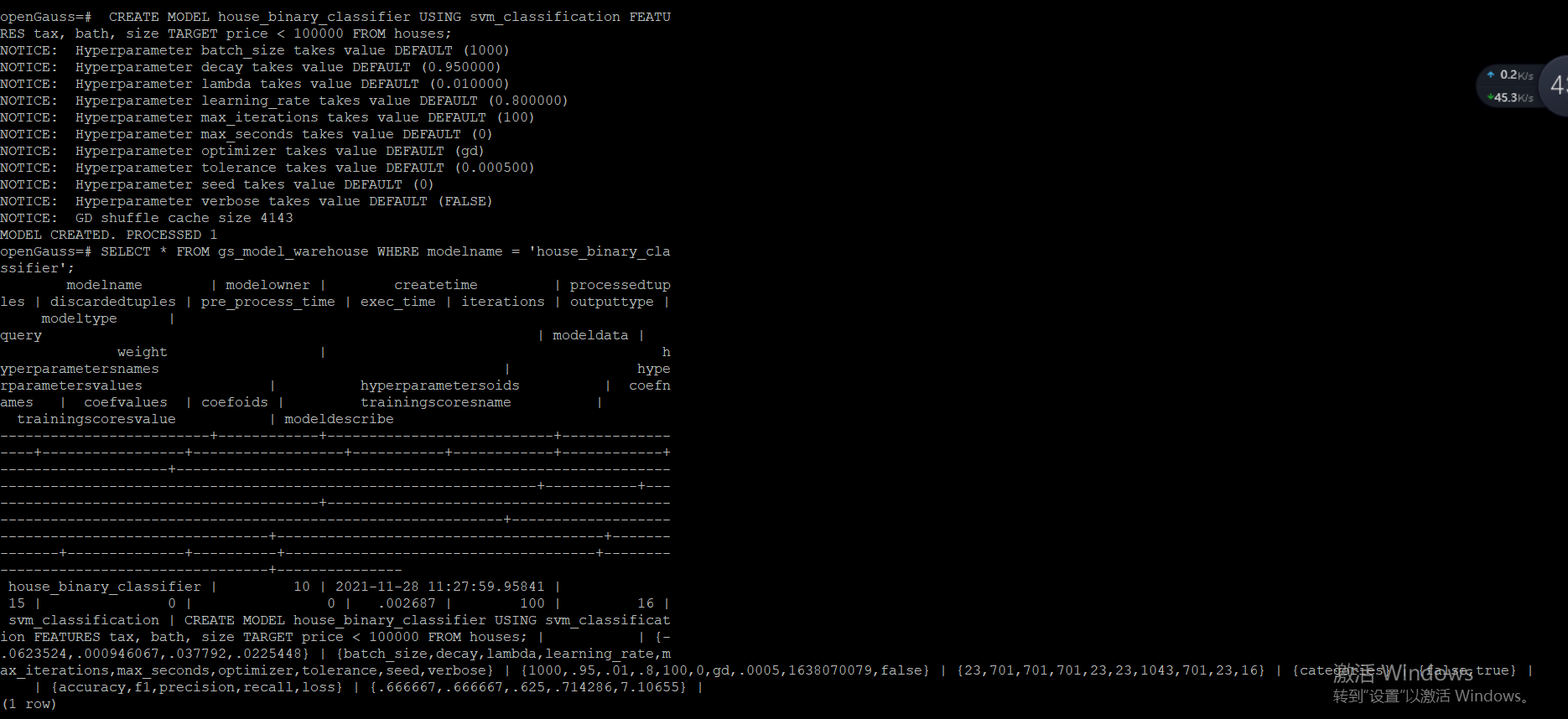
实践思考题2：索引的使用，对于执行SQL有什么好处？除了使用索引和参数外，还有哪些方面可以对数据库进行优化？

索引的使用，可以使数据库服务器以更快的速度检索特定的行，除了索引和参数，还可以通过优化数据结构、优化查询语句等方法来优化数据库

# 关卡四、openGauss的DB4AI特性应用

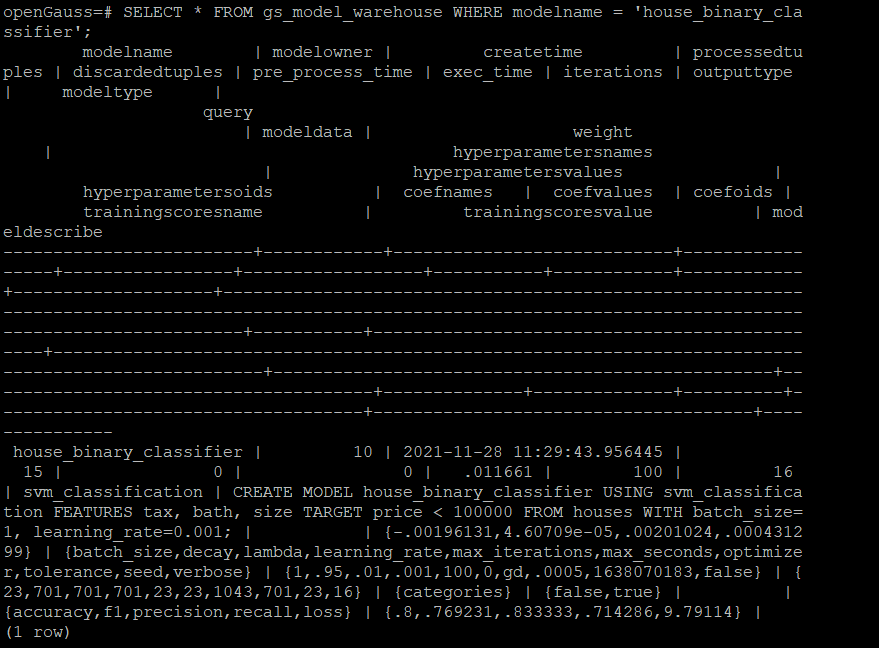
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



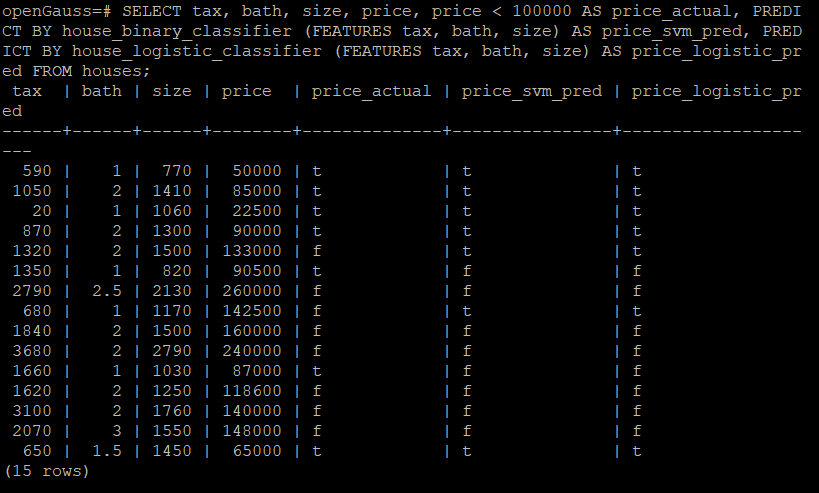
任务二：观察新模型的信息，将执行结果截图。

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？

分类模型输出的是物体所属的类别，是离散的；回归模型输出的是物体具体的值，是连续的。

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，它将实例的特征向量映射为空间中的一些点，SVM 的目的就是想要画出一条线，以 “最好地” 区分这两类点，以至如果以后有了新的点，这条线也能做出很好的分类。SVM 适合中小型数据样本、非线性、高维的分类问题

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有

准确率：正确预测样本数/所有样本数

精确率：label预测为1且预测正确的样本数/ label预测为1样本数

召回率：label预测为1且预测正确的样本数/真实样本中所有 label为1样本数

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？

回归问题的评价指标有：

RMSE(平方根误差)：衡量观测值与真实值之间的偏差。

MAE（平均绝对误差）：绝对误差的平均值。

MSE(平均平方误差)：真实值与预测值的差值的平方然后求和平均。

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。