openGauss AI特性创新实践课



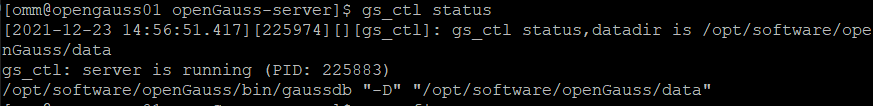
华为技术有限公司

# 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：

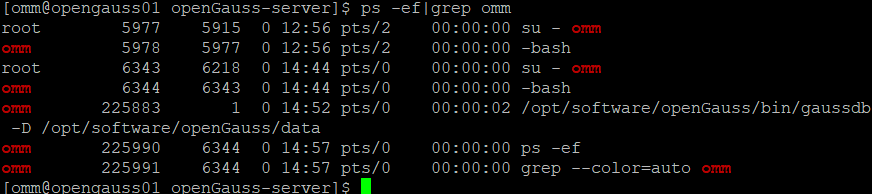
任务一：数据库状态验证

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？

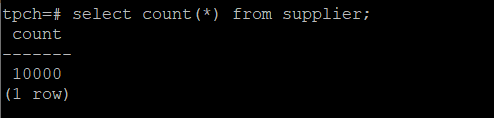
答：通过源码编译可以在安装过程中设定参数，修改源码来定制自己需要的功能，具有更强的开源性。

# 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：

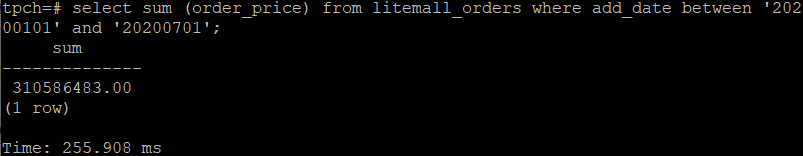
select count(\*) from supplier;;



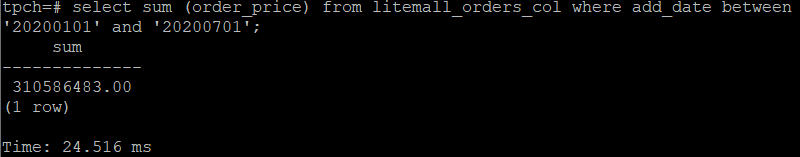
任务二：行存表与列存表执行效率对比

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';

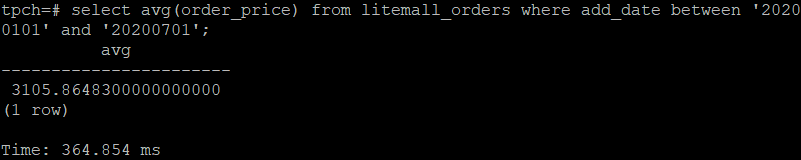


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

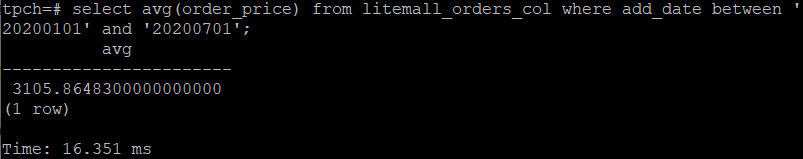


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

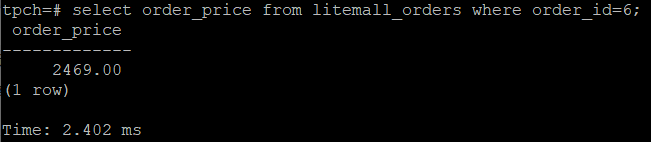


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

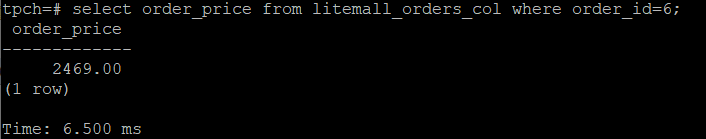


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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;



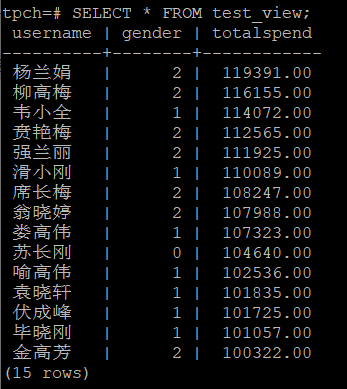
update litemall\_orders\_col set order\_price=2468 where order\_id=6;



任务三：物化视图的使用

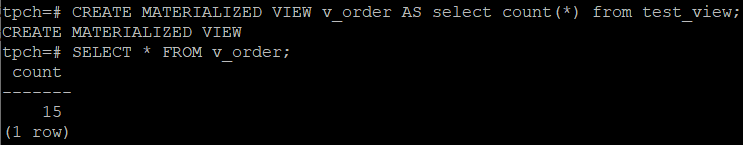
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



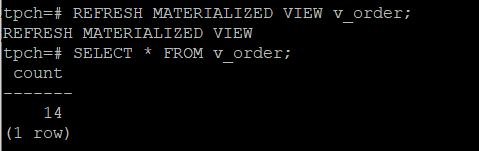
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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;



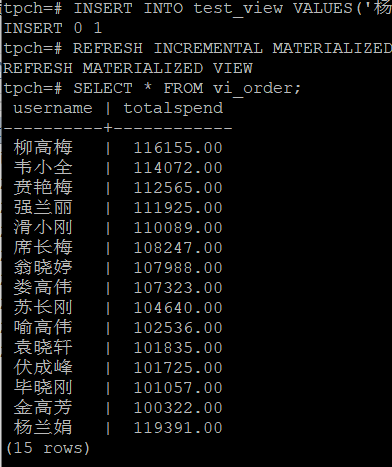
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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：在写入操作上，行存表的写入是一次完成。如果这种写入建立在操作系统的文件系统上，可以保证写入过程的成功或者失败，数据的完整性因此可以确定,，列存表由于需要把一行记录拆分成单列保存，写入次数明显比行存表多,实际消耗时间大。

在读取操作上，行存表通常将一整行数据读出，会造成冗余列，列存表数据存储每一列单独存放， 数据即是索引。数据的存储方式使得行存表和列存表在执行相同sql语句时，执行时间不同。

在执行整个数据行查询如查询某个条件下的数据行，或者关注整个表的整体数据，则可以使用行存表。如果会关注列数据的聚集运算，如计算列的总和平均等，则可以选择列存表。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

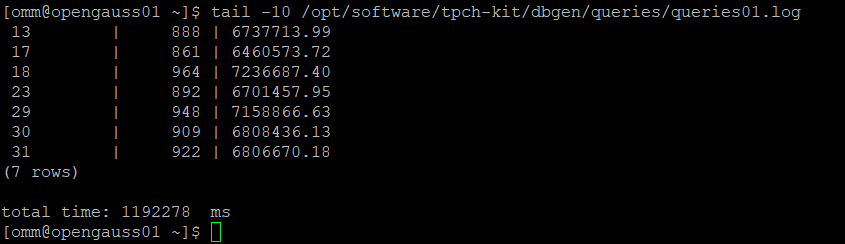
答：全量物化视图仅支持对创建好的物化视图做全量更新，而不支持做增量更新；增量物化视图顾名思义就是可以对物化视图增量刷新，需要用户手动执行语句完成对物化视图在一段时间内的增量数据进行刷新。与全量创建物化视图不同在于目前增量物化视图所支持场景较小，目前物化视图创建语句仅支持基表扫描语句或者UNION ALL语句。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

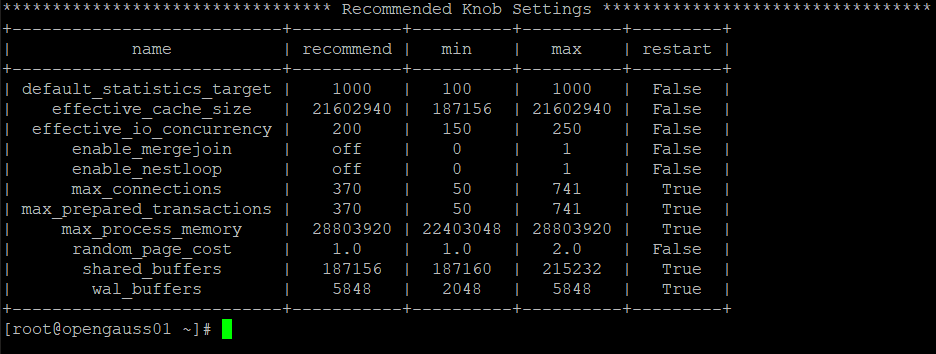
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

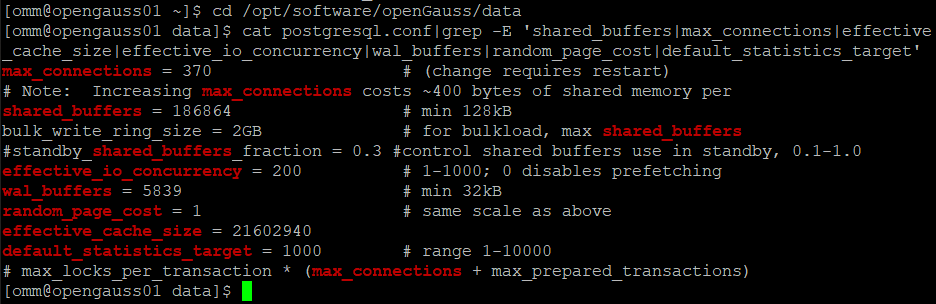
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

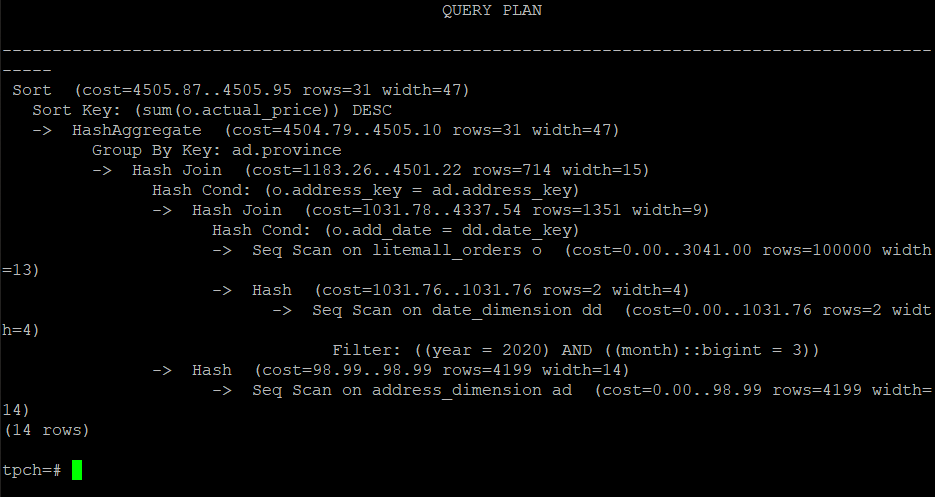
AND o.add\_date = dd.date\_key

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

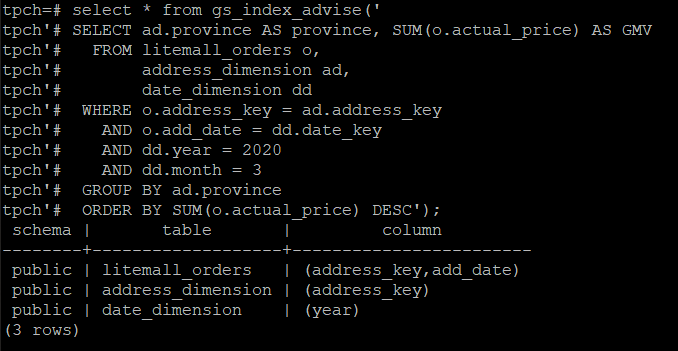
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

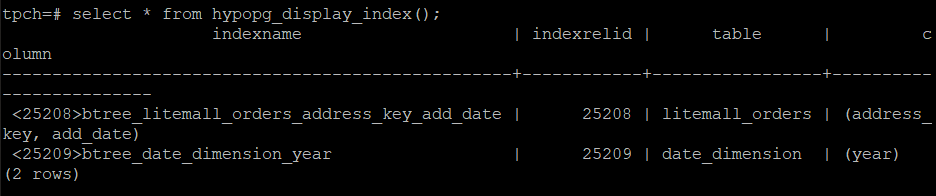
GROUP BY ad.province

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。

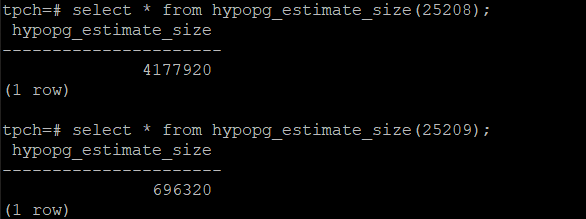
select \* from hypopg\_display\_index();



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

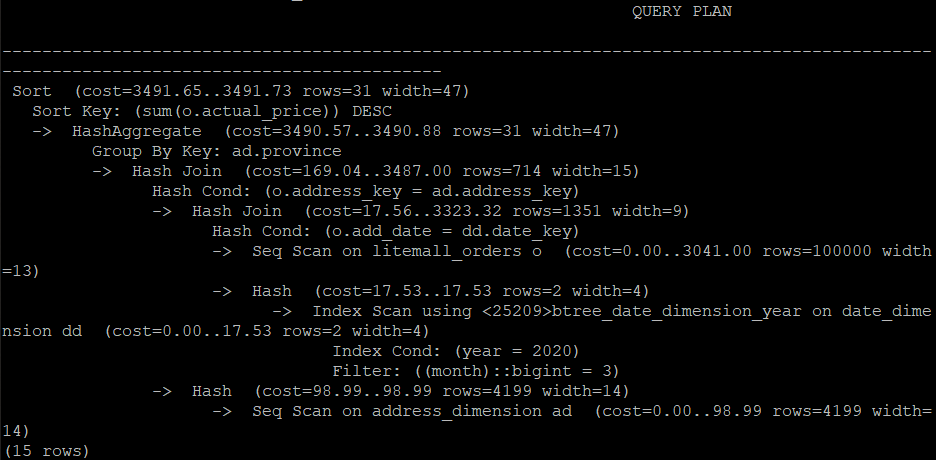
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

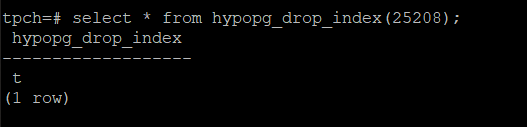
GROUP BY ad.province

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



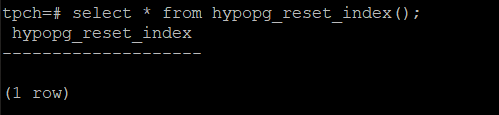
6. 删除某一个索引虚拟列，将执行结果截图。

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



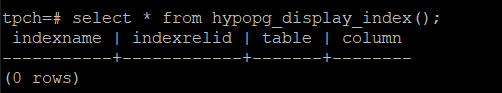
7. 删除所有索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

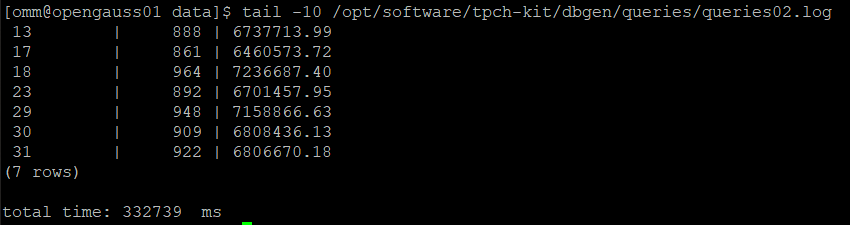
select \* from hypopg\_display\_index();



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？

答：（1）max\_connections，允许的最大客户端连接数。这个参数设置大小和work\_mem有一些关系。配置的越高，可能会占用系统更多的内存。

（2）shared\_buffers，PostgreSQL使用自己的缓冲区，也使用Linux操作系统内核缓冲OS Cache。这就说明数据两次存储在内存中，首先是PostgreSQL缓冲区，然后是操作系统内核缓冲区。与其他数据库不同，PostgreSQL不提供直接IO，所以这又被称为双缓冲。PostgreSQL缓冲区称为shared\_buffer，建议设置为物理内存的1/4。而实际配置取决于硬件配置和工作负载，如果你的内存很大，而你又想多缓冲一些数据到内存中，可以继续调大shared\_buffer。

（3）Effective\_cache\_size，这个参数主要用于Postgre查询优化器。是单个查询可用的磁盘高速缓存的有效大小的一个假设，是一个估算值，它并不占据系统内存。由于优化器需要进行估算成本，较高的值更有可能使用索引扫描，较低的值则有可能使用顺序扫描。

（4）wal\_buffers，事务日志缓冲区的大小，PostgreSQL将WAL记录写入缓冲区，然后再将缓冲区刷新到磁盘。在PostgreSQL 12版中，默认值为-1，也就是选择等于shared\_buffers的1/32 。

（5）default\_statistics\_target，PostgreSQL使用统计信息来生成执行计划。统计信息可以通过手动Analyze命令或者是autovacuum进程启动的自动分析来收集，default\_statistics\_target参数指定在收集和记录这些统计信息时的详细程度。

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？

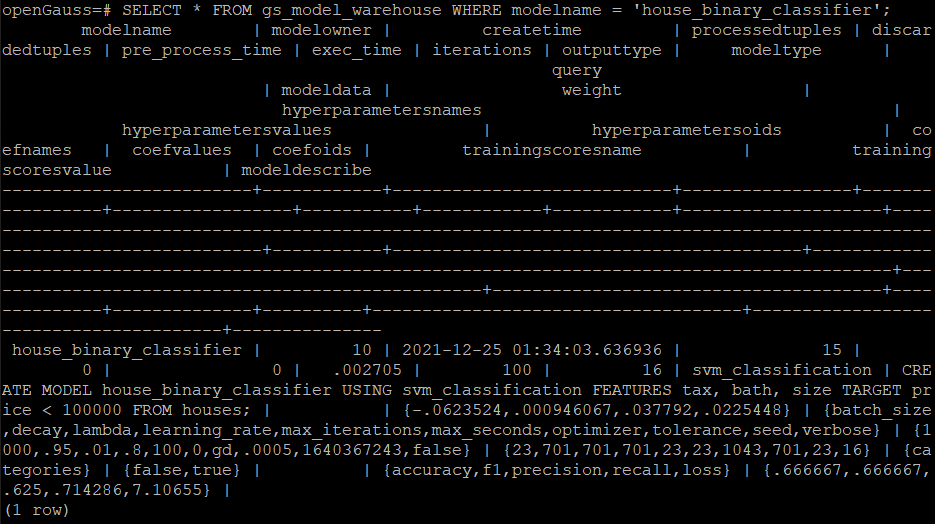
答：好处：可快速访问数据库表中的特定信息；加快表之间的连接；减少分组和排序时间；使用优化隐藏器提高系统性能。

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

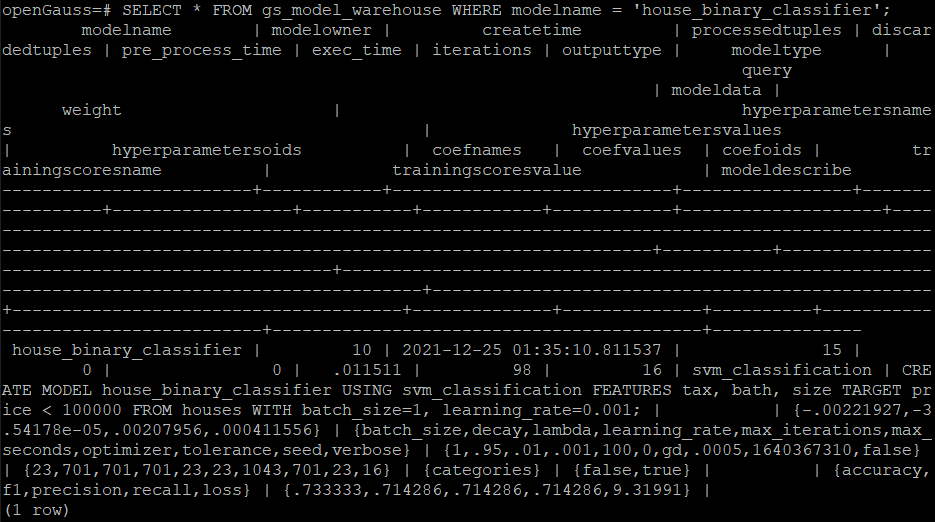
# 关卡四、openGauss的DB4AI特性应用

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：

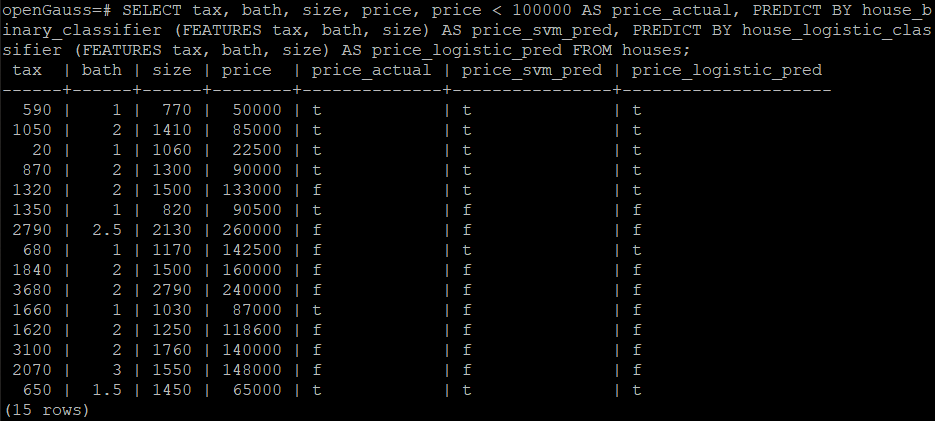
postgres=# SELECT \* FROM gs\_model\_warehouse WHERE modelname = 'house\_binary\_classifier';

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不同点在于输出变量的类型。

回归模型是定量输出，也可以说是连续变量预测；

分类模型是定性输出，也可以说是离散变量预测。

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。

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）RMSE（Root Mean Square Error）均方根误差，衡量观测值与真实值之间的偏差。常用来作为机器学习模型预测结果衡量的标准。

（2）MSE（Mean Square Error）均方误差，MSE是真实值与预测值的差值的平方然后求和平均。通过平方的形式便于求导，所以常被用作线性回归的损失函数。

（3）MAE（Mean Absolute Error）平均绝对误差，是绝对误差的平均值。可以更好地反映预测值误差的实际情况。

（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。

（5）SD（Standard Deviation）标准差，方差的算术平均根。用于衡量一组数值的离散程度。