openGauss AI特性创新实践课



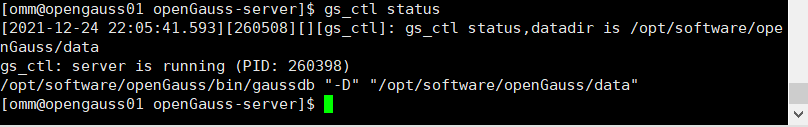
华为技术有限公司

# 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. 查询数据库的截图如下：

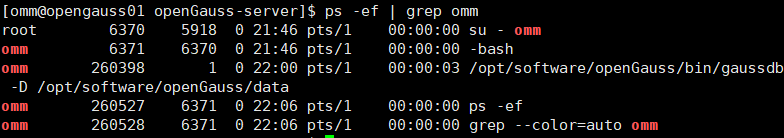


任务二：数据库服务进程验证

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：因为二进制格式的程序强依赖于底层，而使用这种方式就可以避免这种缺陷。

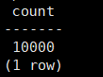


# 关卡二、openGauss数据导入及基本操作

任务一：数据初始化验证

1. 查询supplier表的行数，结果的截图如下所示：

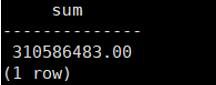
命令为：select count(\*) from supplier;



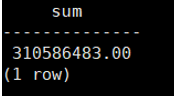
任务二：行存表与列存表执行效率对比

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

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

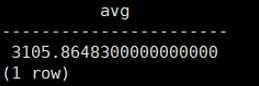


命令为：select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

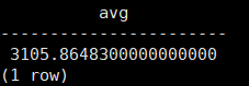


命令为：select avg (order\_price) from litemall\_orders where add\_date between '20200101' and '20200701';

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

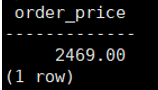


命令为：select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

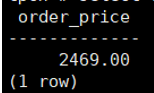


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

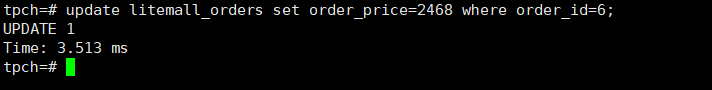


命令为：select order\_price from litemall\_orders\_col where order\_id=6;

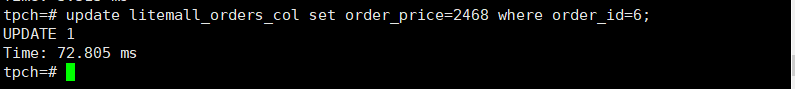


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

命令为：update litemall\_orders set order\_price=2468 where order\_id=6;



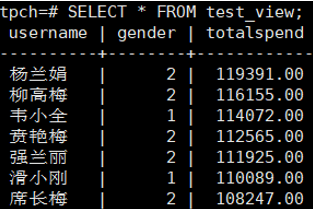
命令为：update litemall\_orders\_col set order\_price=2468 where order\_id=6;



任务三：物化视图的使用

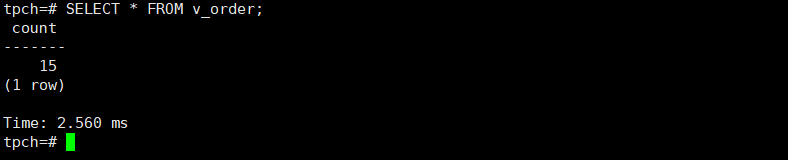
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

命令为：SELECT \* FROM test\_view;



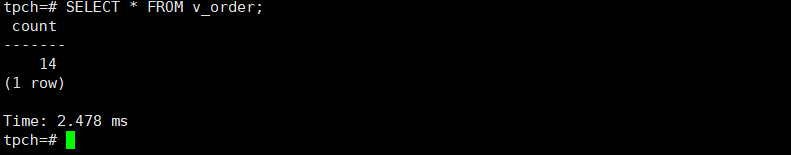
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

命令为：SELECT \* FROM v\_order;

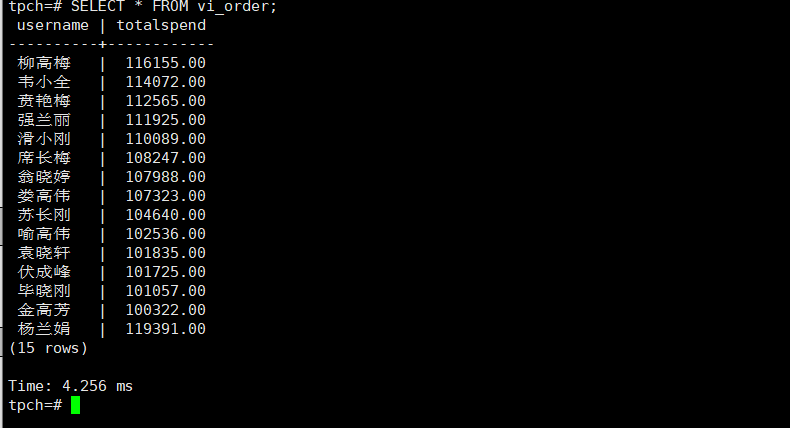


3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

命令为：SELECT \* FROM v\_order;

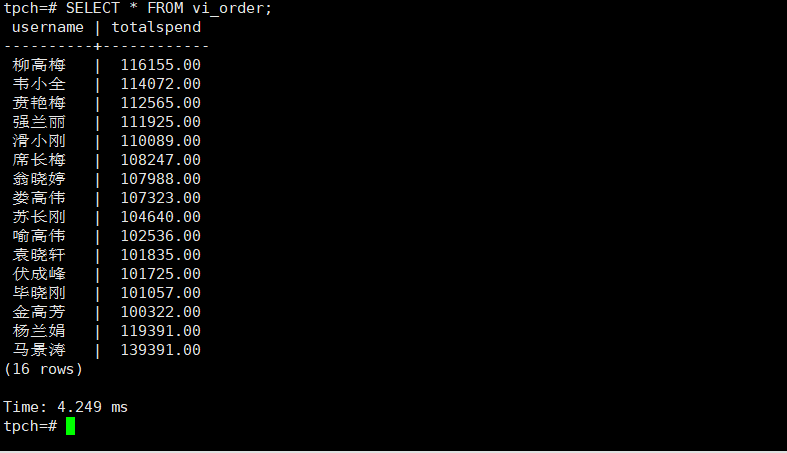


4. 创建增量物化视图，查询物化视图结果，将执行结果截图。



命令为：SELECT \* FROM v\_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。

命令为：SELECT \* FROM v\_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：DBMS对行存表和列存表的存储方式不同，同时根据时间复杂度和空间复杂度，以及计算机组成和操作系统的性质上来说，这样做的效率也会要高很多。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

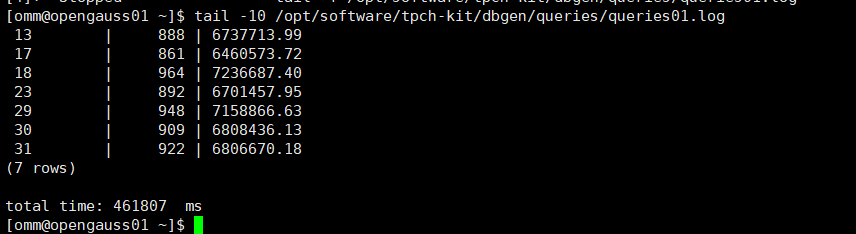
答：全量物化视图仅支持对创建好的物化视图做全量更新，而不支持做增量更新；创建全量物化视图语法和CREATE TABLE AS语法一致，不支持对全量物化视图指定NodeGroup创建。增量物化视图顾名思义就是可以对物化视图增量刷新，而这需要用户自己手动去进行一个刷新。与全量创建物化视图不同在于目前增量物化视图所支持场景较小，目前物化视图创建语句仅支持基表扫描语句或者UNION ALL语句。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

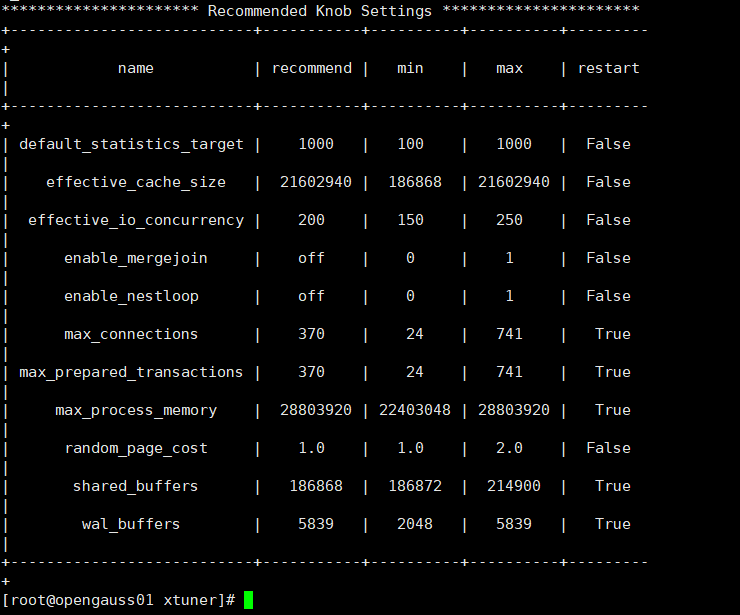
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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



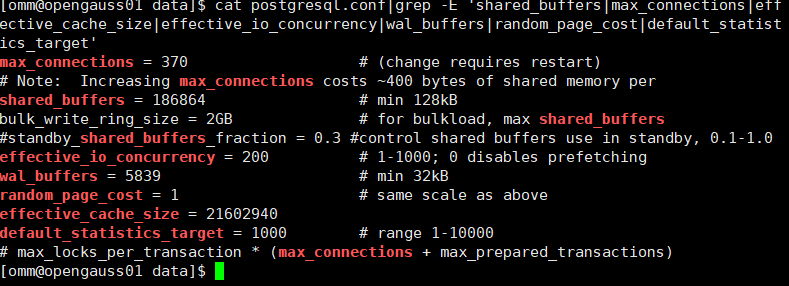
2. 使用root用户，执行X-Tuner进行参数建议优化，将执行结果截图

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



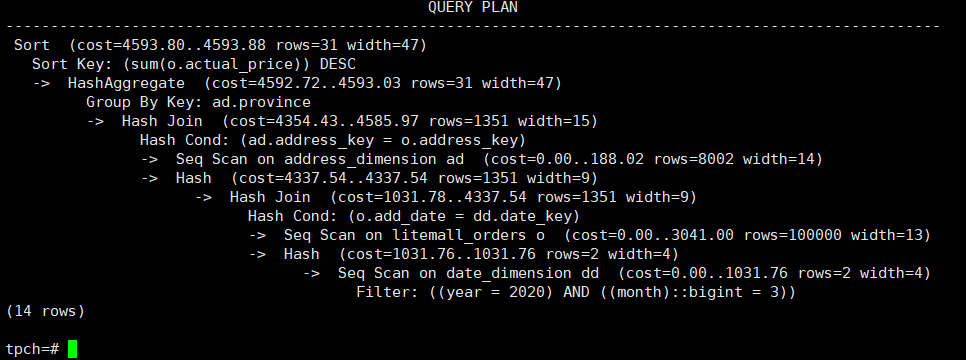
3.重启完成后，获取参数值：

命令为：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'

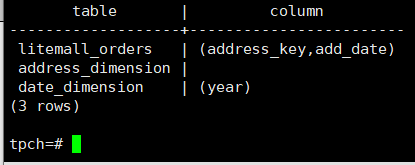


任务二：使用Index-advisor对select 查询语句进行优化，并通过对比执行计划，得到优化前后的不同。

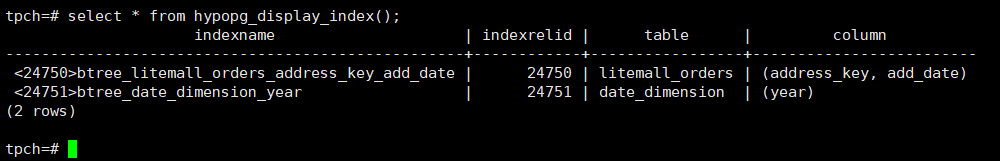
1. 使用explain，对查询2020年3月订单表收入并进行排序的SQL加以分析，将结果截图。

命令为：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。

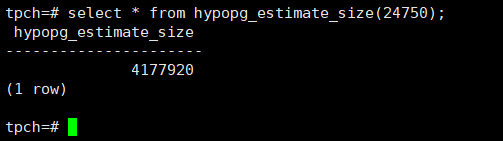
命令为：select \* from gs\_index\_advise(' SELECT ad.province AS province, SUM(o.actual\_price) AS GMV FROM litemall\_orders o, address\_dimension ad, date\_dimension dd WHERE o.address\_key = ad.address\_key AND o.add\_date = dd.date\_key AND dd.year = 2020 AND dd.month = 3 GROUP BY ad.province ORDER BY SUM(o.actual\_price) DESC');

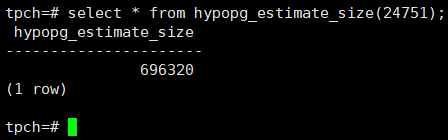
3. 查看创建的虚拟索引列，将执行结果截图。

命令为：select \* from hypopg\_display\_index();

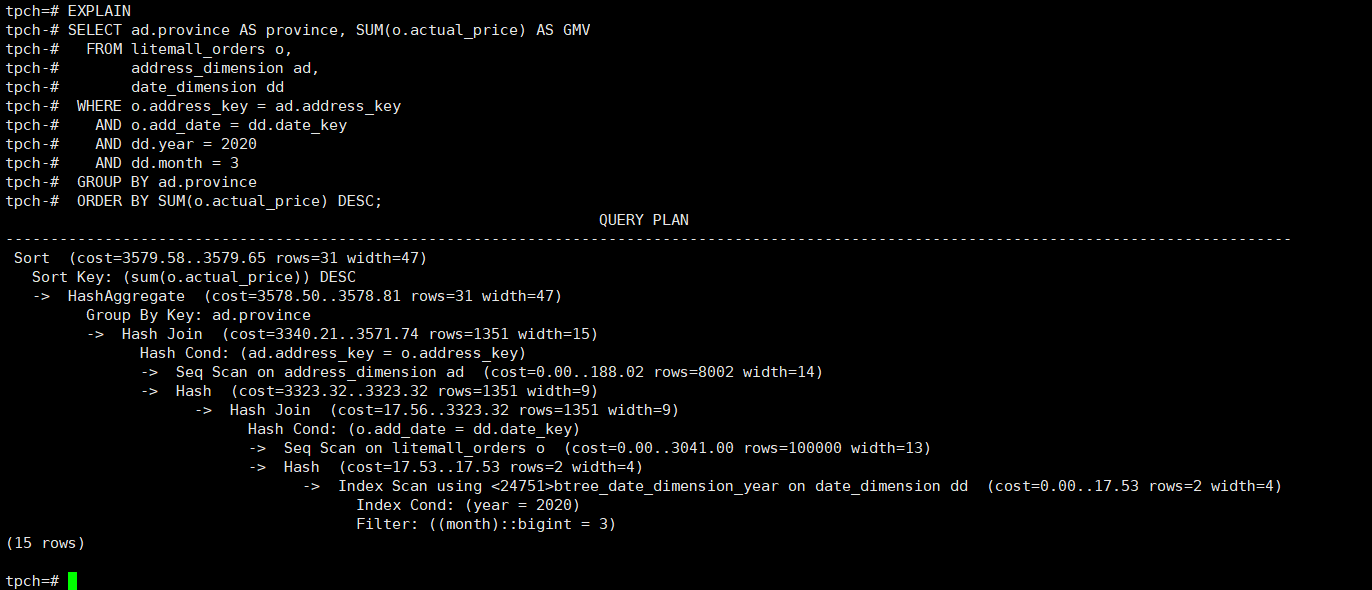
4. 获取索引虚拟列大小结果（单位为：字节），将执行结果截图。

命令为：select \* from hypopg\_estimate\_size(16715);

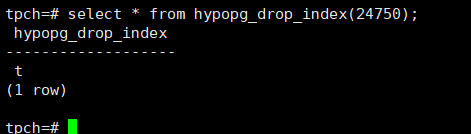
命令为：select \* from hypopg\_estimate\_size(16716);



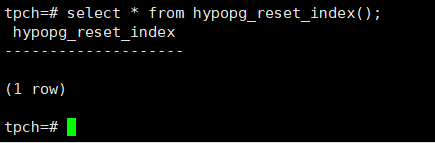
5.再次使用explain，对该SQL加以分析，将执行结果截图。

命令为：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;

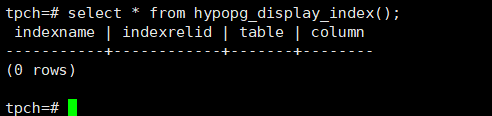
6. 删除某一个索引虚拟列，将执行结果截图。

命令为：select \* from hypopg\_drop\_index(16715);

7. 删除所有索引虚拟列，将执行结果截图。

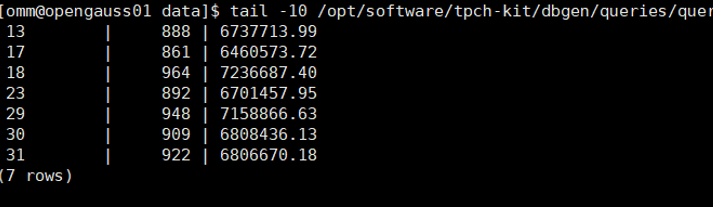
命令为：select \* from hypopg\_reset\_index();

8. 查看索引虚拟列，将执行结果截图。

命令为：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：

命令为：gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log 

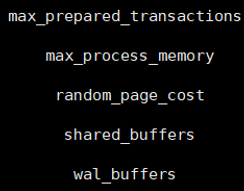
挑战一：进一步优化queries.sql中的查询语句，使得前后执行时间出现倍数级的提升。

1. 重新执行queries.sql查询，将执行结果截图：

命令为：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：对以下参数进行了优化：



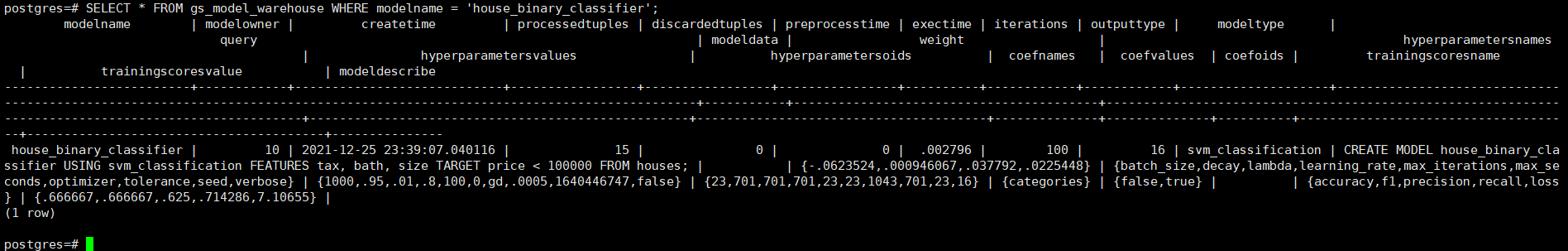
我们对这个数据库的查询是有特点的（即TPCH测试脚本的内在的特征），针对这些特性来使用X-TUNNER可以让其相对于DBMS有一个更好的性能提升。

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？

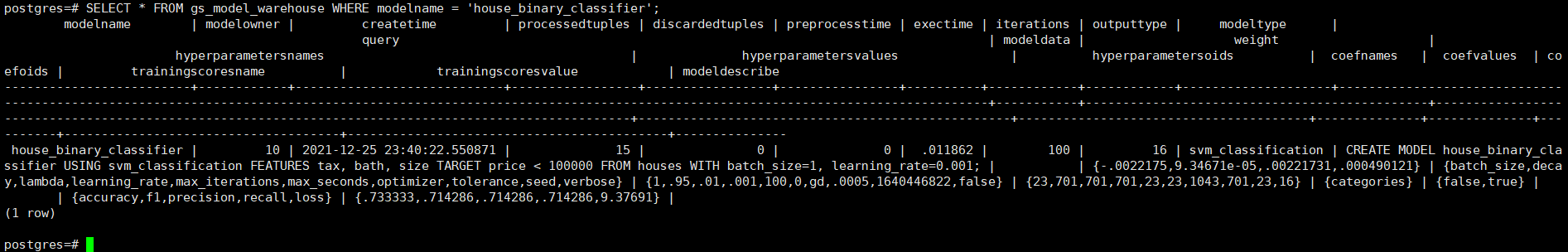
答：索引的使用，有利于DBMS快速的定位到数据的存储位置，加速SQL的执行。从而更好的为用户服务，分析日常使用时对表操作的方式，对于接下来改进也非常重要。

# 关卡四、openGauss的DB4AI特性应用

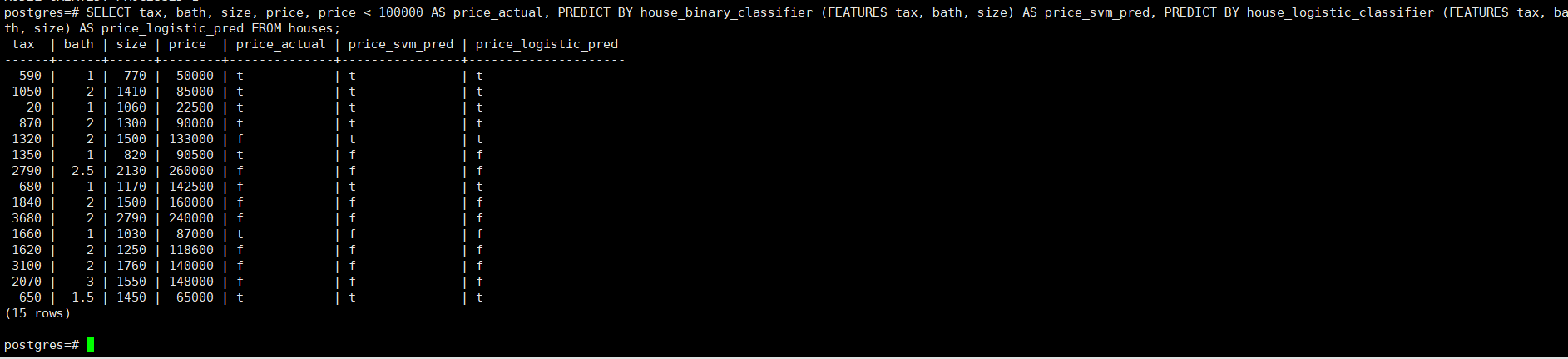
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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：分类模型是认为模型的输出是离散的，例如大自然的生物被划分为不同的种类，是离散的。 回归模型的输出是连续的 ，例如人的身高变化过程是一个连续过程，而不是离散的。

实践思考题2：什么是SVM算法？

答：SVM是一个二元分类算法，线性分类和非线性分类都支持。 经过演进，现在也可以支持多元分类，同时经过扩展，也能应用于回归问题。 感知机的模型就是尝试找到一条直线，能够把二元数据隔离开。

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：学习中遇到的分类任务中的评价指标有准确率（Accuracy）、FPR、FNR、Recall、Precision、F-score、MAP、ROC曲线和AUC等，回归任务中的指标有 (r)MSE、MAE、CC/PCC等。

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：分类问题的评价指标是 准确率 ，那么回归算法的评价指标就是 MSE，RMSE，MAE、R-Squared。MSE和MAE适用于误差相对明显的时候，大的误差也有比较高的权重，RMSE则是针对误差不是很明显的时候；MAE是一个线性的指标，所有个体差异在平均值上均等加权，所以它更加凸显出异常值，相比MSE； RMSLE: 主要针对数据集中有一个特别大的异常值，这种情况下，data会被skew，RMSE会被明显拉大，这时候就需要先对数据log下，再求RMSE，这个过程就是RMSLE。