openGauss AI特性创新实践课



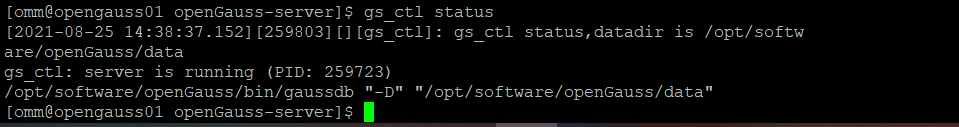
华为技术有限公司

# 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：

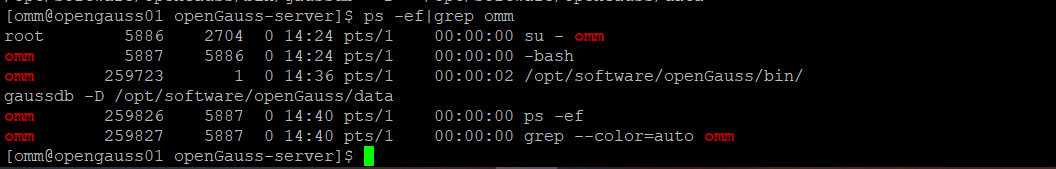
任务一：数据库状态验证

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？

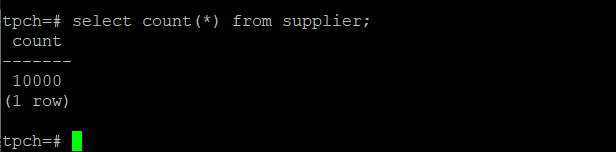
**答：编译安装数据库可以使得数据库根据机器进行优化处理，可以设定参数，按照需求进行安装，并且可以选择安装的版本，灵活性比较大。**

# 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：

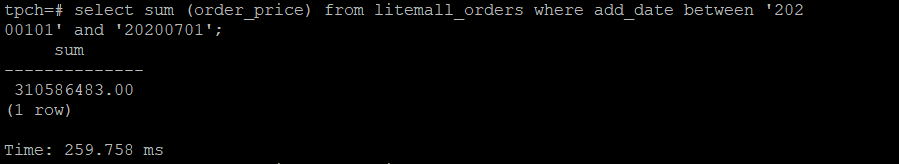
select count(\*) from supplier;;



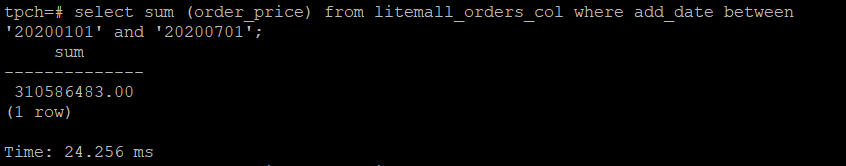
任务二：行存表与列存表执行效率对比

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';

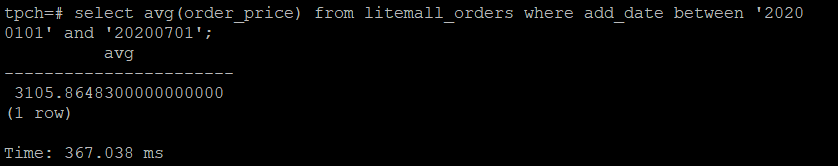


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

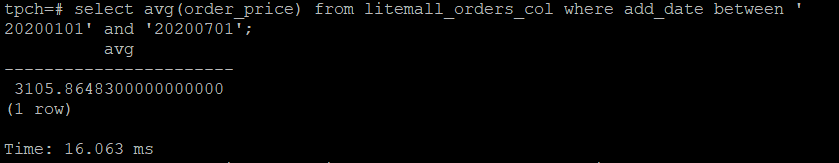


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

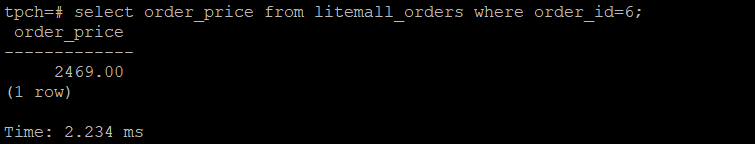


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

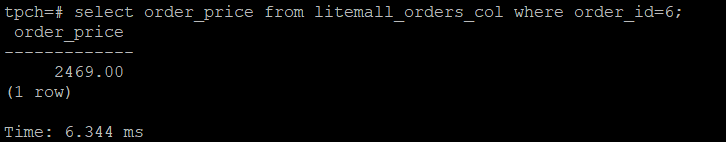


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

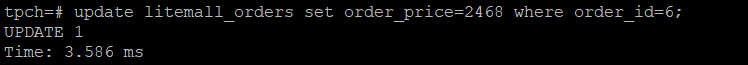


select order\_price from litemall\_orders\_col where order\_id=6;

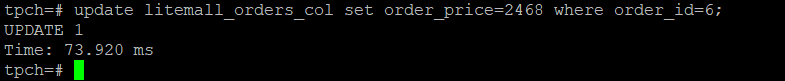


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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

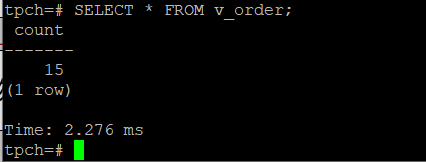
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



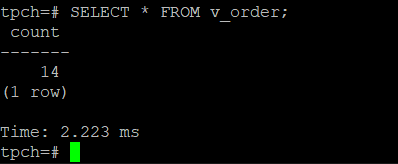
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



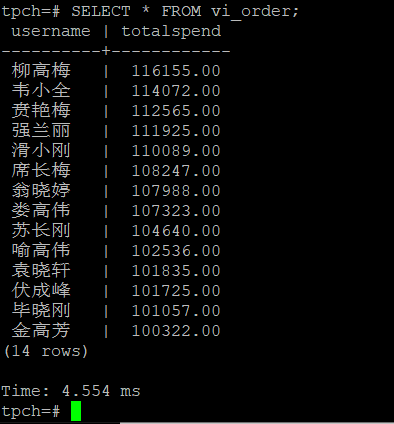
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

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。

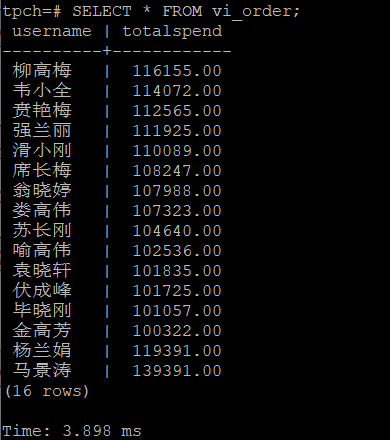
SELECT \* FROM vi\_order;



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？

**答：在列存表中，一列的数据在存储介质中以连续存储的形式存在，因此在进行某一列的聚合查询操作时，执行的IO操作少，所以速度较快，而在查询某一行数据获改变某一行数据时，需要多次进行IO操作，因此速度较慢。因此在执行更新，普通查询操作时，行存表的效率更高，在执行聚合查询操作时，列存表的效率更高**

实践思考题2：全量物化视图与增量物化视图有哪些差别？

**答：全量物化视图在刷新时会重新构建表，刷新较慢，但查询时较快。而增量物化视图增量更新时针对新增的数据，刷新较快，但以特殊数据结构的形式存储在内容中，因此查询全部内容时较慢。**

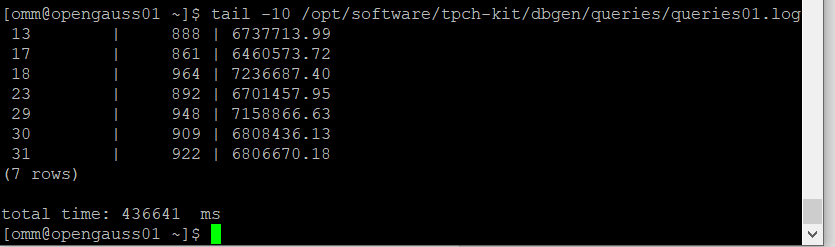
# 关卡三、openGauss的AI4DB特性应用

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：

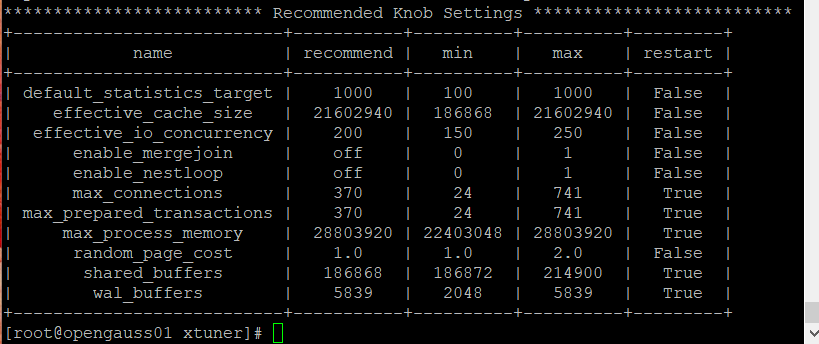
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log





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

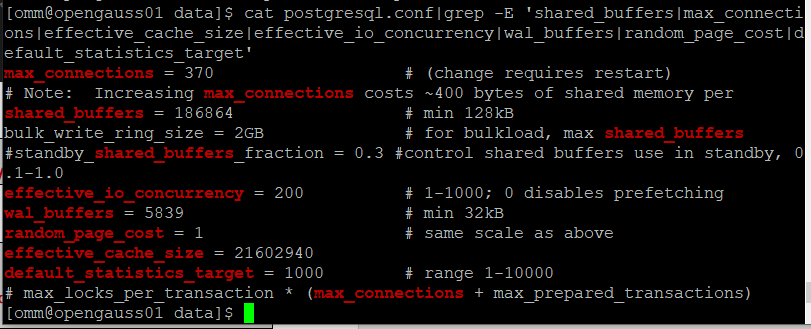
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

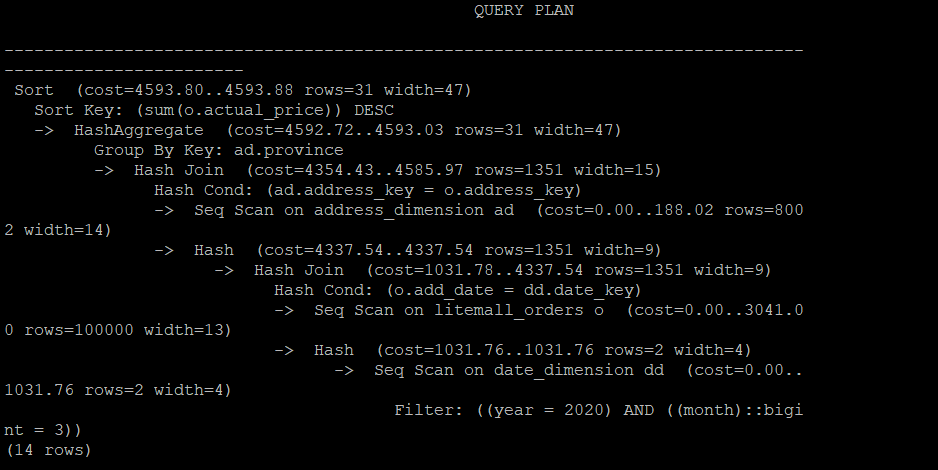
AND o.add\_date = dd.date\_key

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

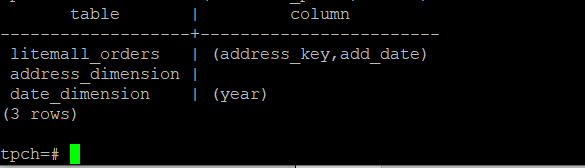
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

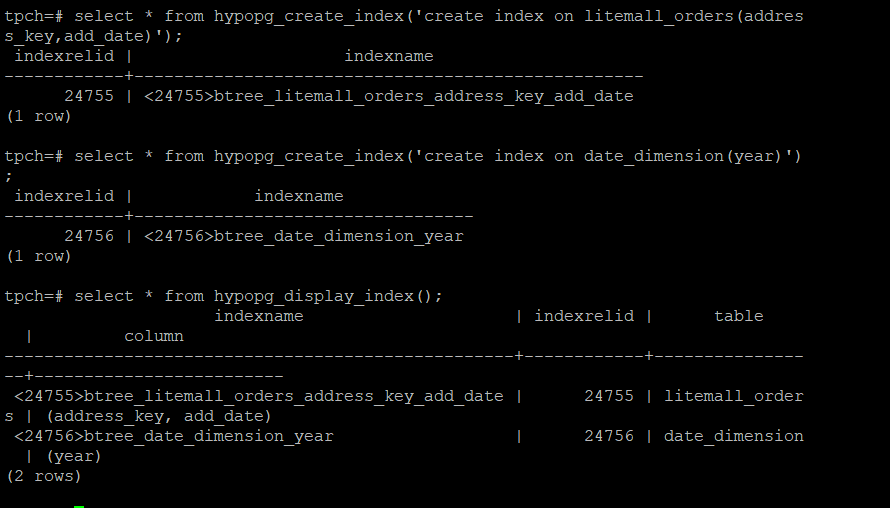
GROUP BY ad.province

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。

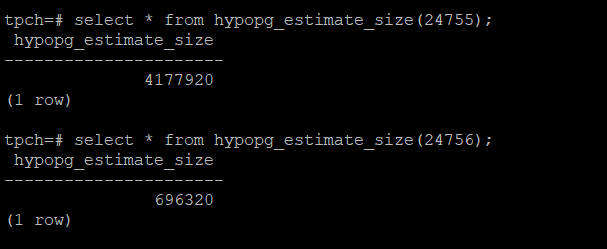
select \* from hypopg\_display\_index();



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

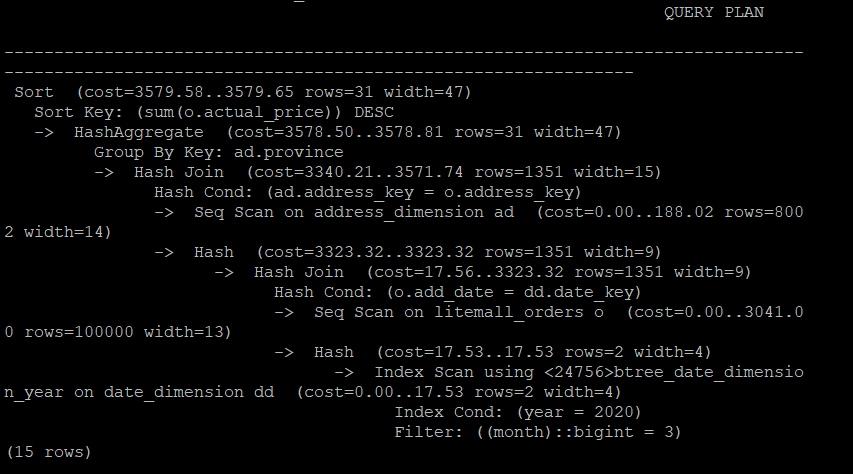
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

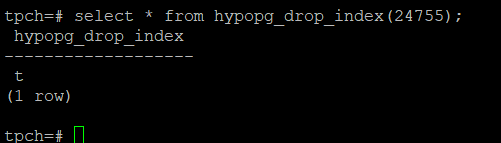
GROUP BY ad.province

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



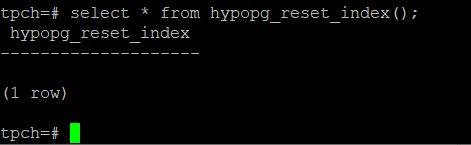
6. 删除某一个索引虚拟列，将执行结果截图。

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



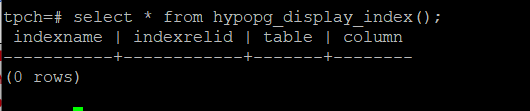
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

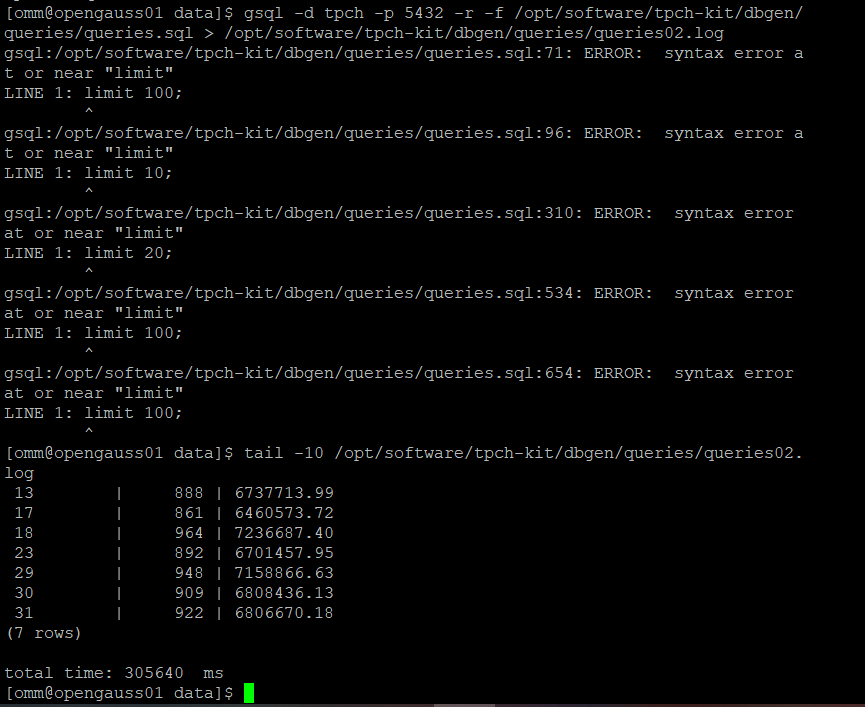
select \* from hypopg\_display\_index();



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



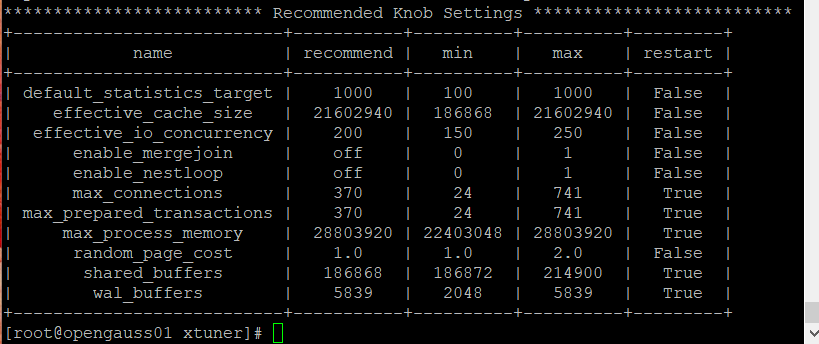
挑战一：进一步优化queries.sql中的查询语句，使得前后执行时间出现倍数级的提升。

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？

答：



给出的参数优化如上图所示，default\_statistics\_target为没有通过ALTER TABLE SET STATISTICS设置的列特定目标设置表列的默认统计目标。该参数告诉 PostgreSQL 应该抽样多少数据来填充存储元数据的表，提高后可以改进规划器的执行计划；effective\_cache\_size为优化器假设可以用于单个查询的磁盘缓存的有效大小，该值越大，使用索引扫描的可能性越大，因此查询更快；effective\_io\_concurrency为PostgreSQL可以同时被执行的并发磁盘 I/O 操作的数量，最佳值根据CPU，磁盘决定；enable\_mergejoin为是否先对各表排序，再连接，由于hash join比merge join更快，因此关闭；enable\_nestloop是否允许走nestloop连接，关闭原因与enable\_megejoin相同；max\_connections为最大连接数，最佳值需根据实际查询情况设置；max\_prepared\_transactions至少与max\_connections一样大，以便每个会话都可以有一个待处理的预备事务；random\_page\_cost为设置规划器对非顺序获取的磁盘页面的成本的估计，减少这个值可以使得系统更倾向索引扫描；shared\_buffer，数据缓存区，功能类似cache，设置值与机器内存大小相关；wal\_buffers，PostgreSQL将其WAL（[预写日志](https://www.cnblogs.com/VicLiu/p/11854782.html)）记录写入缓冲区，然后将这些缓冲区刷新到磁盘。由wal\_buffers定义的缓冲区的默认大小为16MB，但如果有大量并发连接的话，则设置为一个较高的值可以提供更好的性能。

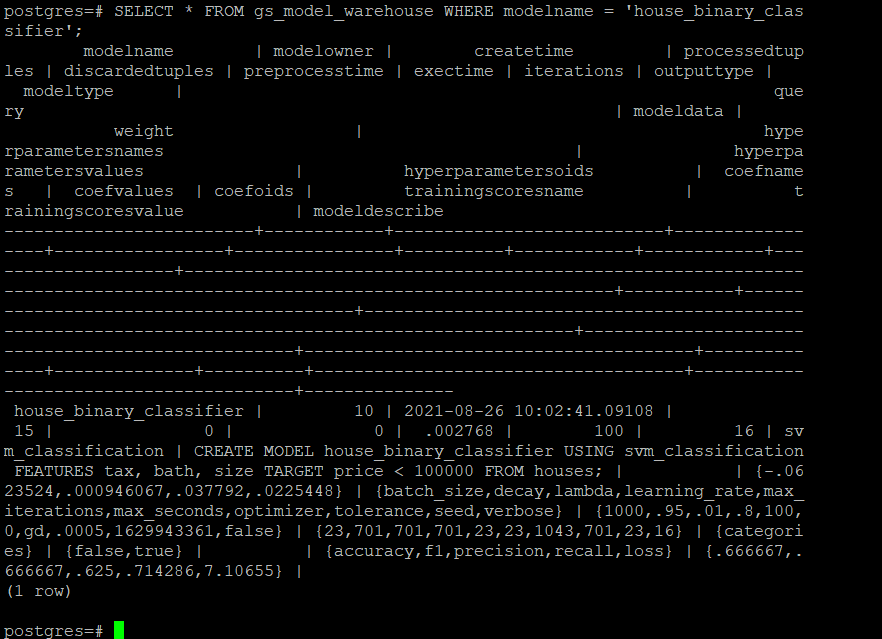
实践思考题2：索引的使用，对于执行SQL有什么好处？除了使用索引和参数外，还有哪些方面可以对数据库进行优化？

答：**索引的使用可以加速表扫描查询的时间，使得扫描查询平均复杂度由O（n）降为O（logn）。除了使用索引和参数外，还可以对数据库表的结构，系统配置，硬件等方法优化数据库。**

# 关卡四、openGauss的DB4AI特性应用

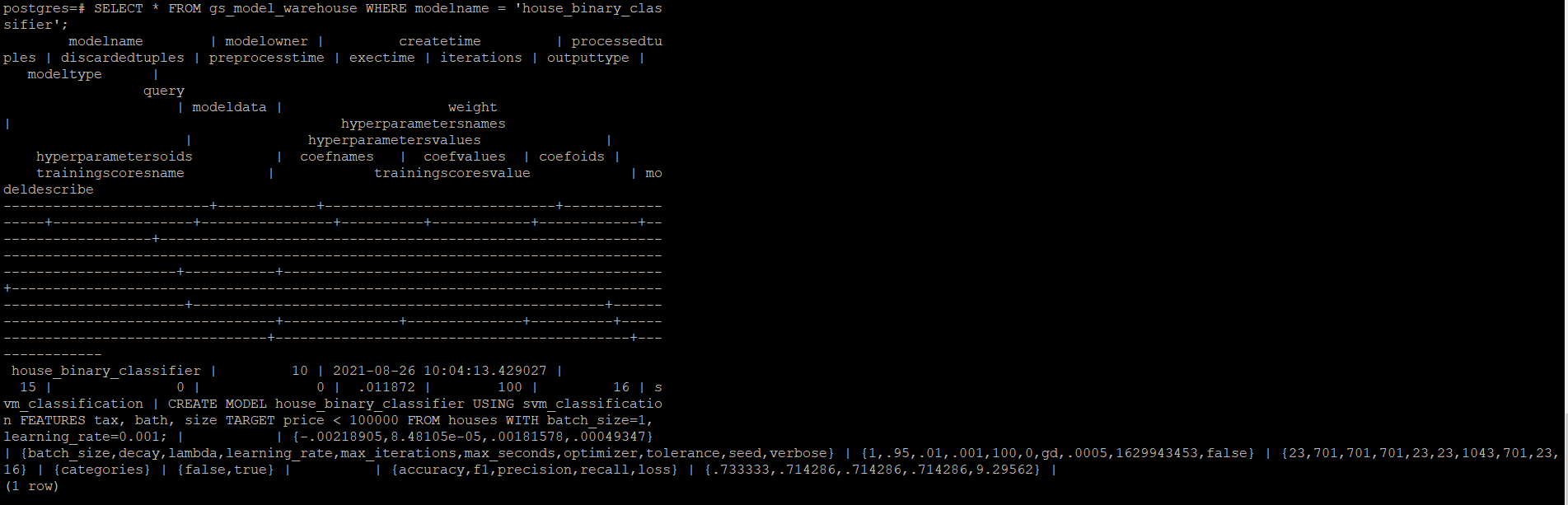
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



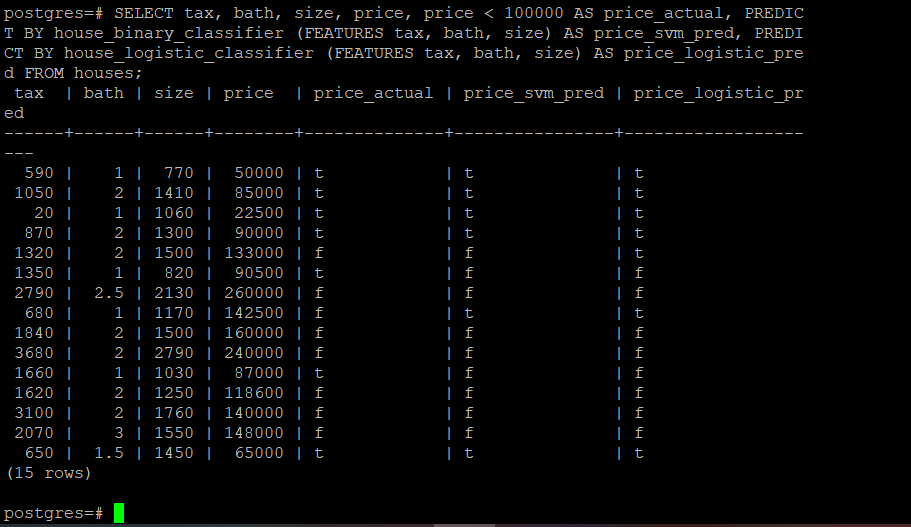
任务二：观察新模型的信息，将执行结果截图。

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？

答：**分类模型输出为离散化的数据，回归模型输出为连续的数据。**

实践思考题2：什么是SVM算法？

答：**SVM的全称是Support Vector Machine，即支持向量机，主要用于解决模式识别领域中的数据分类问题，属于有监督学习算法的一种。**

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：

**True Positive(真正, TP)：将正类预测为正类数。  
True Negative(真负 , TN)：将负类预测为负类数。**

**False Positive(假正, FP)：将负类预测为正类数。**

**False Negative(假负 , FN)：将正类预测为负类数。**

**Precision(精确率)：TP / (TP + FP)**

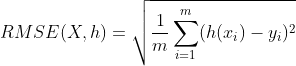
**Recall(召回率)：TP / (TP + FN)**

**Accurary(准确率)：(TP + TN) / (TP + FP + TN + FN)**

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：

**MSE（均方误差）**

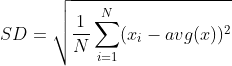
****

**RMSE（均方根误差）=sqrt(MSE)**

**MAE（平均绝对误差）**

**MAE(X, h)=\frac{1}{m}\sum_{i=1}^{m}\left | h(x_{i})-y_{i} \right |**

**SD（标准差）**

****