openGauss AI特性创新实践课



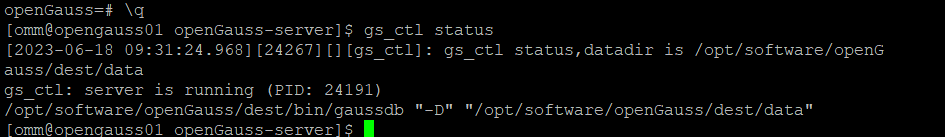
华为技术有限公司

# 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：

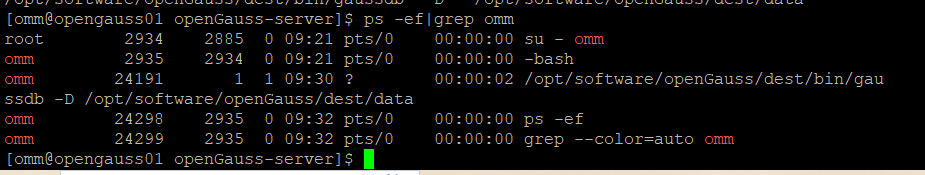
任务一：数据库状态验证

1.查询数据库状态成功截图



任务二：数据库服务进程验证

2.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？

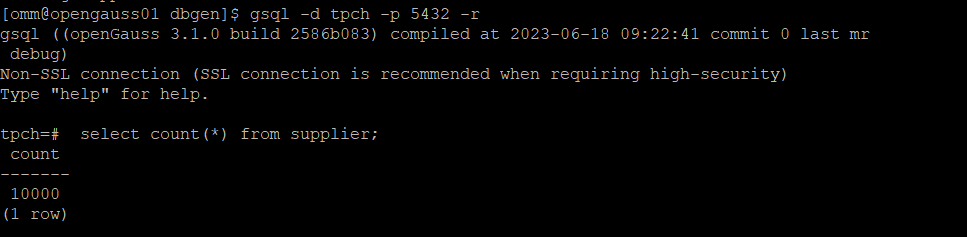
通过源码编译，可以自定义一些参数和配置，根据系统进行特定的优化，也可以指定特定的版本，在遇到问题时，可以通过调试符号和日志信息等来分析和解决问题，更好地进行调试。

# 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：

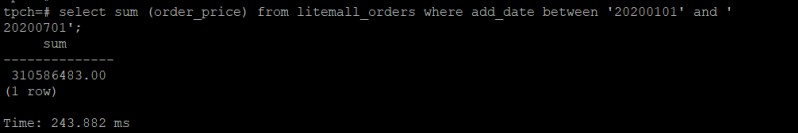
select count(\*) from supplier;;



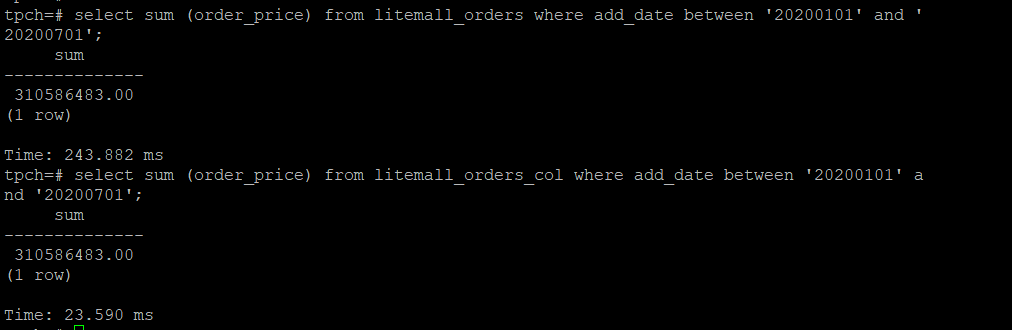
任务二：行存表与列存表执行效率对比

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';

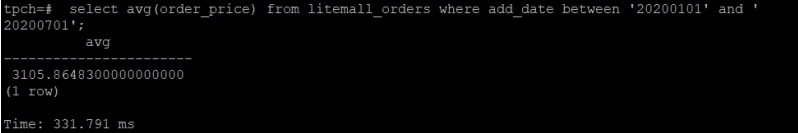


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

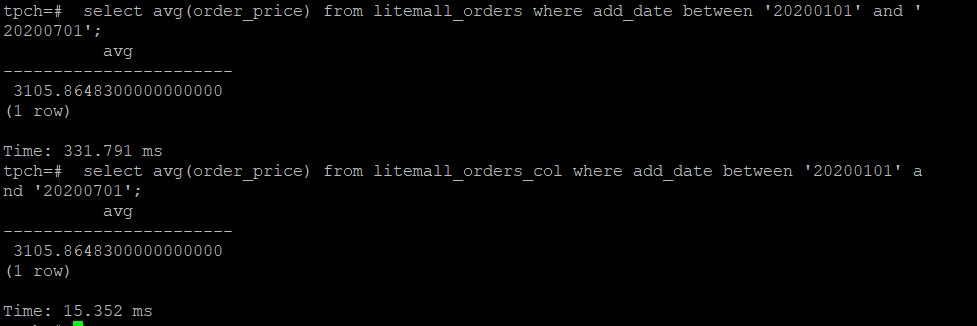


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

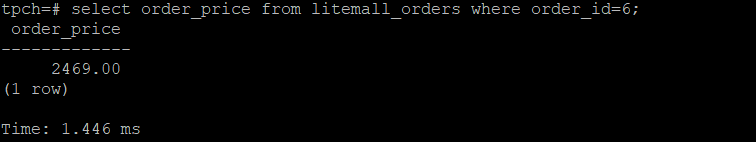


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

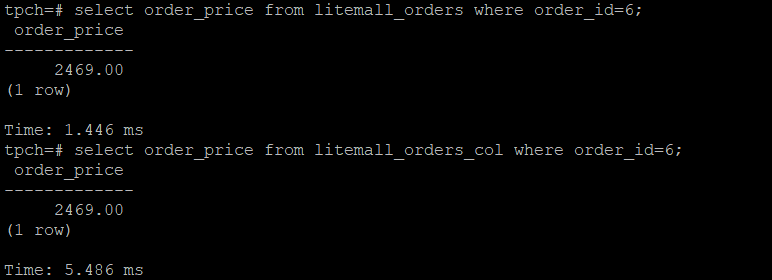


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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



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

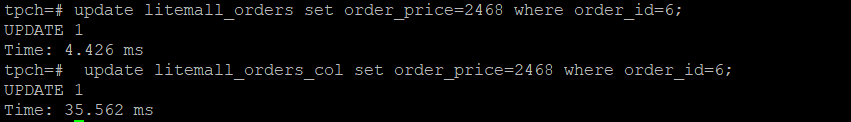


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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

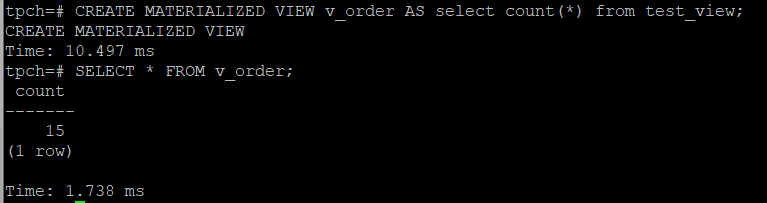
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



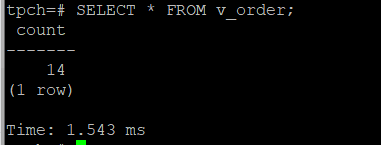
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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;



再次插入数据后：



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？

行存表是按照行的方式存储数据的，它将一行完整的数据存储在一起。在执行语句时，需要读取整行的数据，包括需要的列和其他列。列存表是按照列的方式存储数据的，它将同一列的数据存储在一起。在执行查询时，只需要读取所需的列数据，而不必读取其他列的数据。

行存表更高：单行查询，因为他可以一次读取整行数据。插入、更新和删除，对大部分的增删改操作，行存表效率更高，因为它可以快速定位到需要操作的行并进行修改

列存表更高：聚合查询、复杂查询如多表连接，在这些查询里，列存表可以减少很多不必要的数据读取和处理，只读取所需的列数据，可以减少磁盘 I/O 和数据传输的开销。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

1.刷新方式：全量物化视图重新计算并完全重建视图数据，而增量物化视图只更新变化的数据部分。

2.资源消耗：全量物化视图可能需要更多的计算资源和时间来执行全量刷新，而增量物化视图可以减少资源消耗和刷新时间。

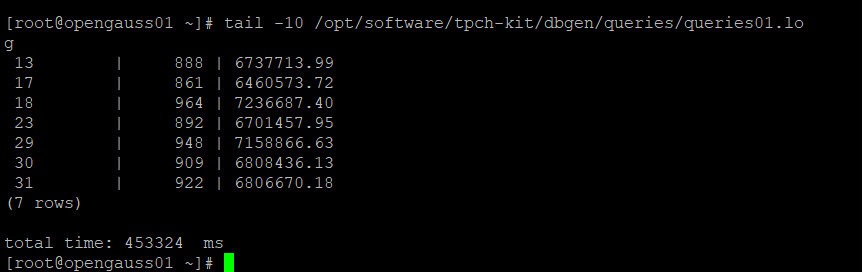
3.数据一致性：全量物化视图确保物化视图中的数据与基表数据完全一致，而增量物化视图在刷新过程中可能存在一定的延迟，因为它只更新变化的数据部分。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

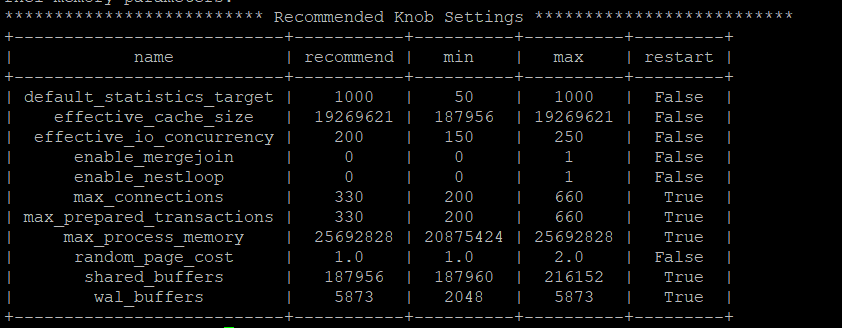
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

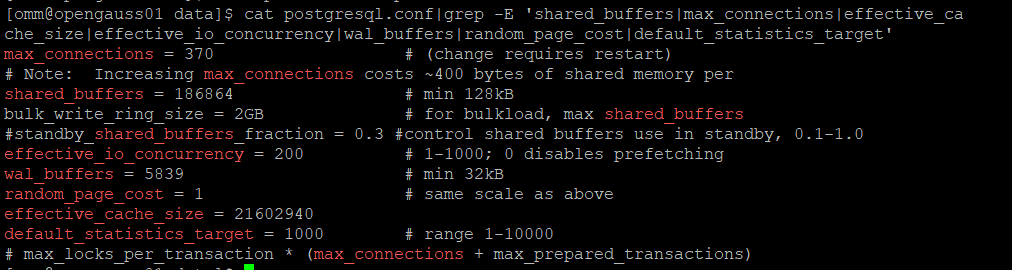
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

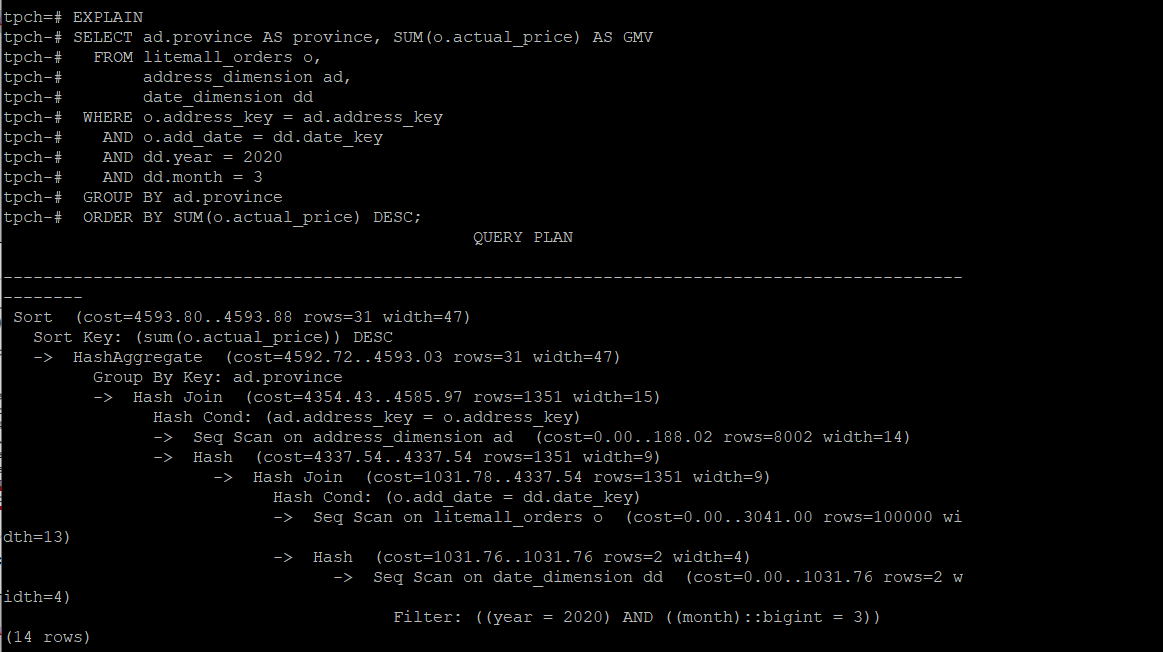
AND o.add\_date = dd.date\_key

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

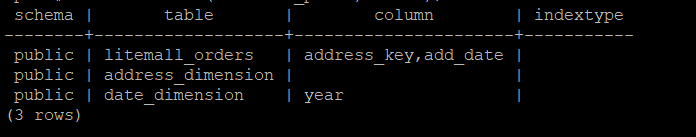
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

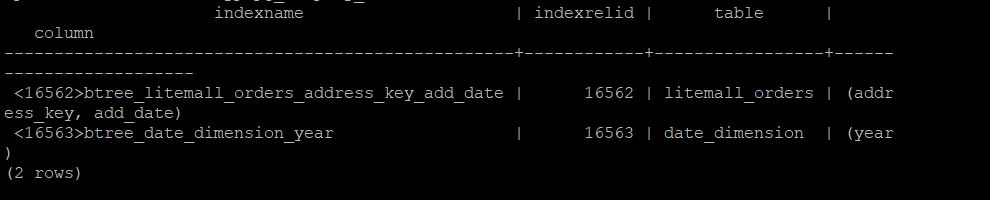
GROUP BY ad.province

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。

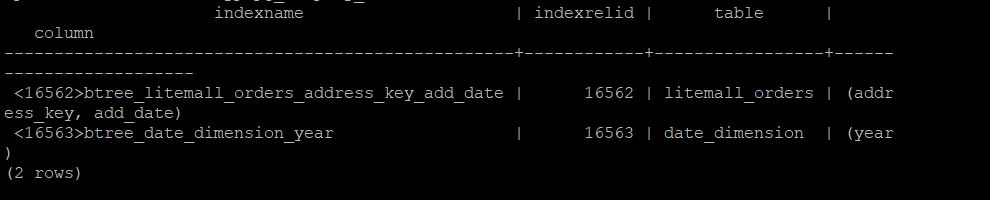
select \* from hypopg\_display\_index();



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

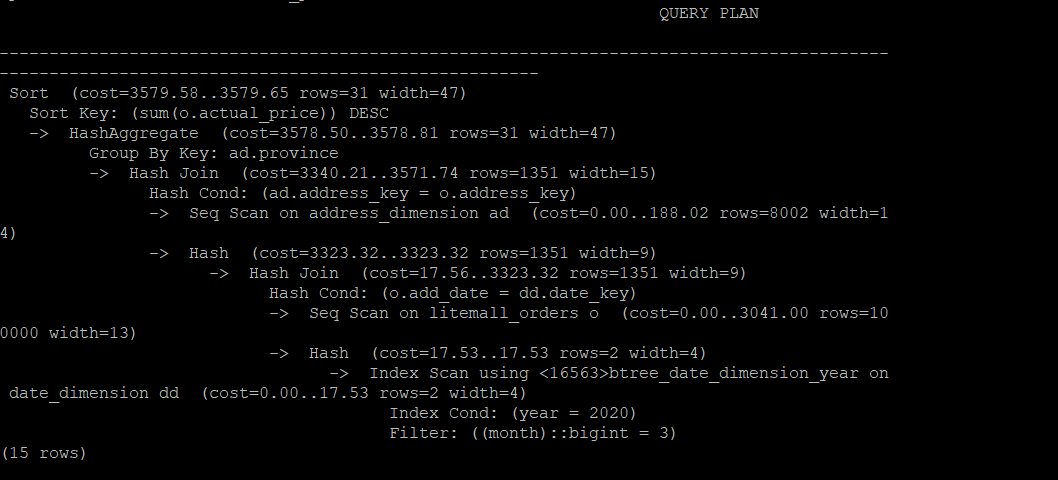
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

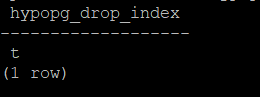
GROUP BY ad.province

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



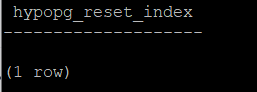
6. 删除某一个索引虚拟列，将执行结果截图

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



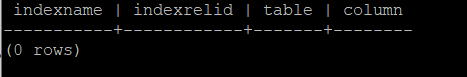
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

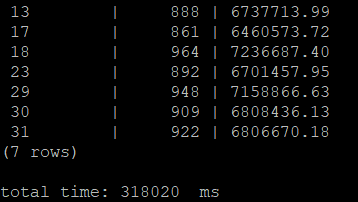
select \* from hypopg\_display\_index();



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：

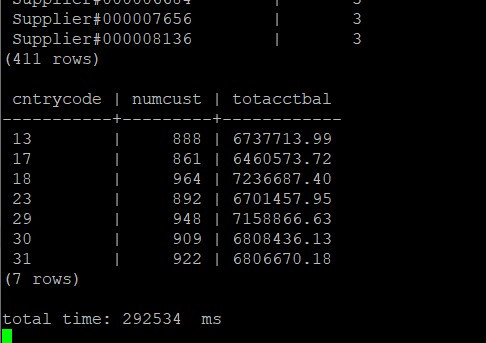
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log



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？

1.shared\_buffers（共享缓冲区）：设置为186864表示在共享缓冲区中分配186,864个8KB页面。共享缓冲区用于存储常用数据块，提高数据读取的速度。适当调整该参数可以提高查询性能和减少磁盘 I/O。

2.max\_connections（最大连接数）：设置为370表示数据库允许最多370个并发连接。该参数控制数据库可以同时处理的连接数。根据应用需求和系统资源，适当调整该参数可以避免连接数过多导致系统负荷过大。

3.effective\_cache\_size（有效缓存大小）：设置为21602940表示估计系统可用于缓存的内存大小为21,602,940 KB。用于指导查询优化器在执行查询计划时估算数据在缓存中的可用空间，可以更好地选择合适的查询计划。

4.effective\_io\_concurrency（有效IO并发数）：设置为200表示在IO操作期间允许的并发度为200。该参数用于指导系统在执行IO操作时的并发度设置，以提高IO操作的效率和吞吐量。

5.wal\_buffers（WAL缓冲区）：设置为5839，表示为WAL日志记录分配5839个8KB页面。适当改变参数可以提高写入性能和减少磁盘IO。

6.random\_page\_cost（随机页面成本）：设置为1表示在查询优化中将随机页面访问与顺序页面访问看作具有相同的成本。适当调整这个参数可以影响查询计划的选择。

7.default\_statistics\_target（默认统计目标）：设置为1000表示指定为每个列的默认统计目标数。适当调整这个参数可以提高查询计划的准确性和性能。

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？

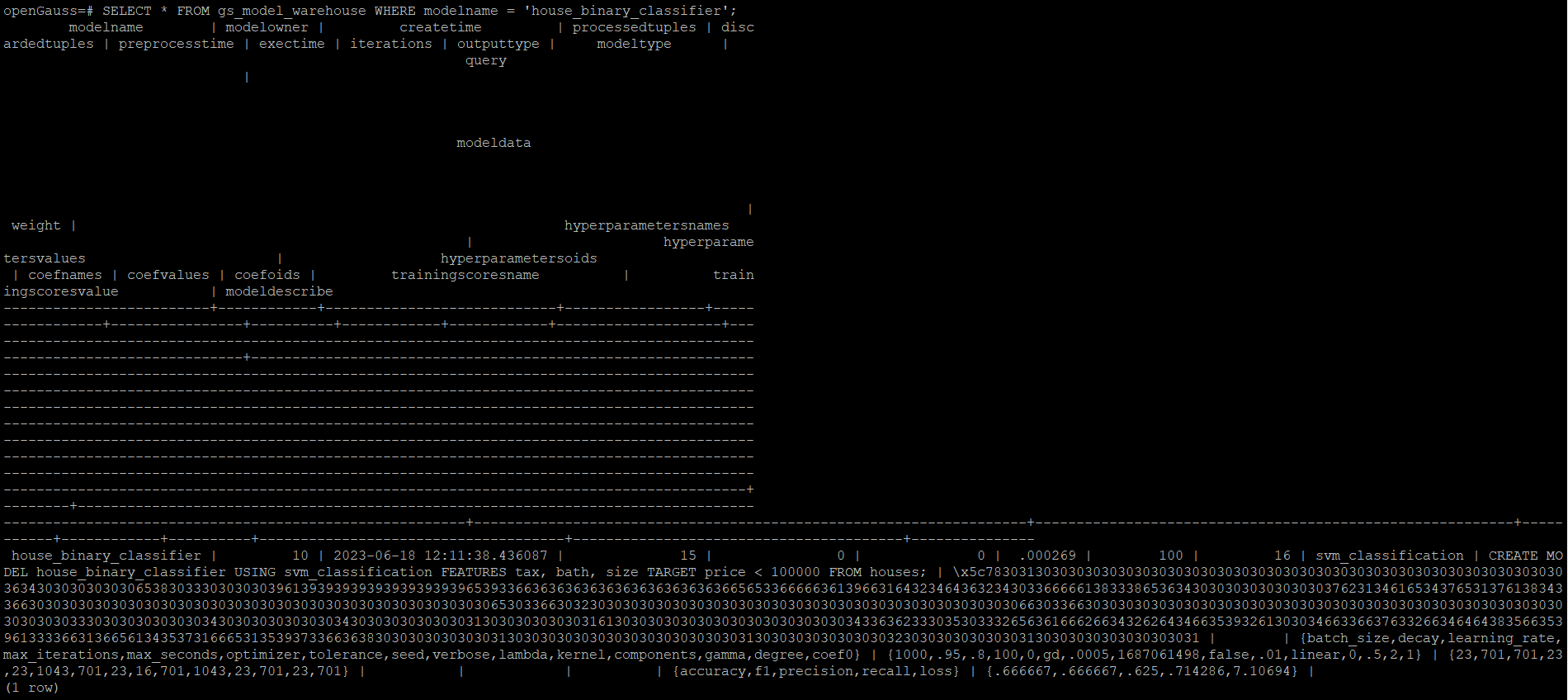
索引的好处：所以可以提高查询性能，减少相关的IO开销，也支持更多的操作如排序、聚合，无需读取整个表，可以避免临时表的创建和排序过程的开销。

还可以优化数据库的设计，合理设计数据库的结构，进行硬件的优化，提升硬件配置等等。

# 关卡四、openGauss的DB4AI特性应用

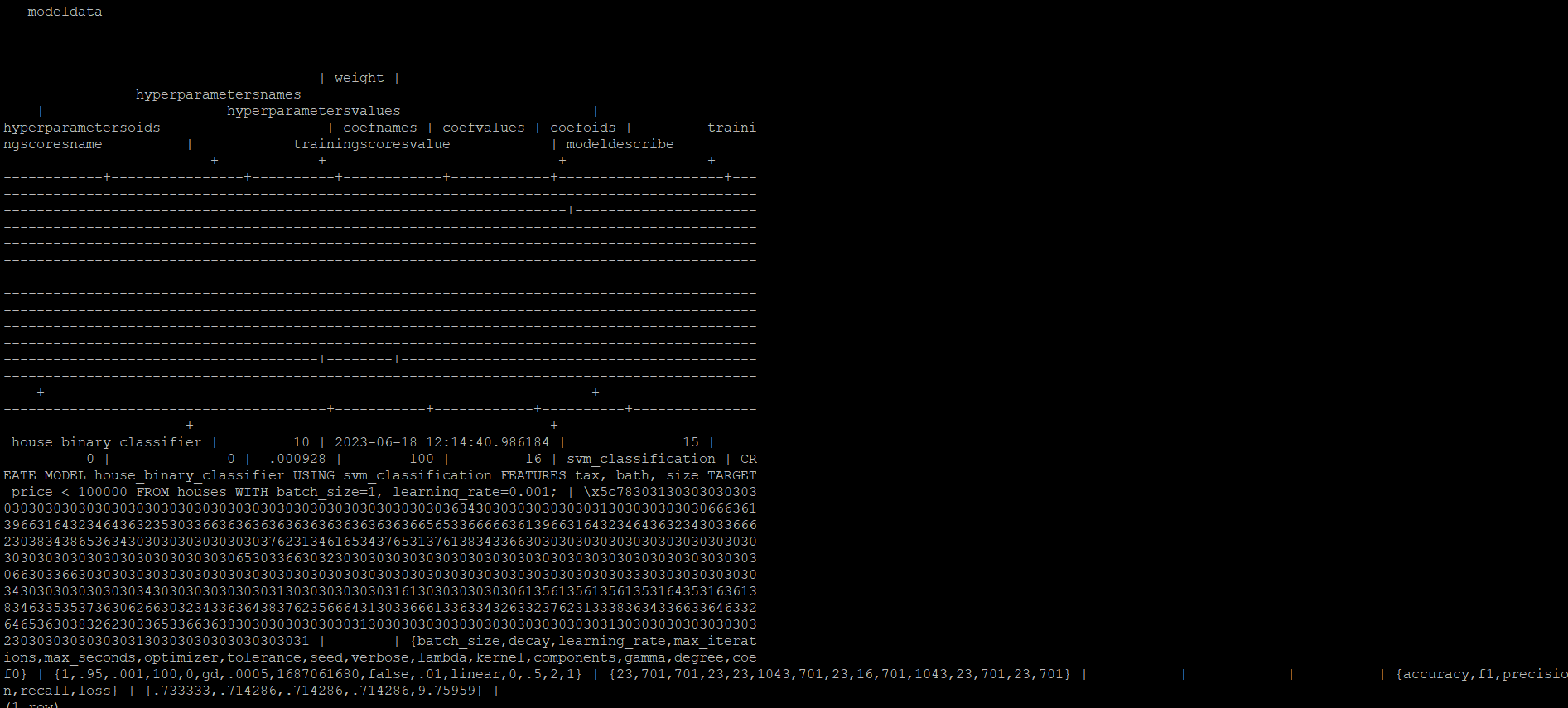
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



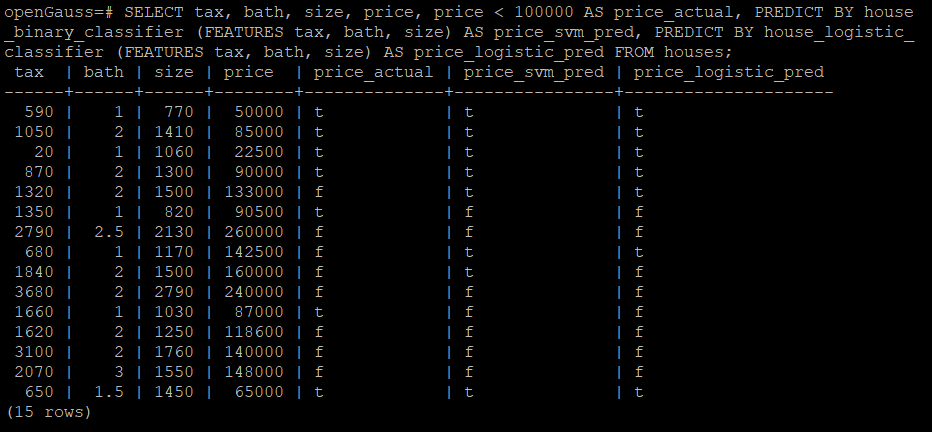
任务二：观察新模型的信息，将执行结果截图。

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：用于预测离散的类别或标签。输出的是某个特定的类别。回归模型：用于预测连续的数值。输出的是所预测的值。

实践思考题2：什么是SVM算法？

SVM（支持向量机）是一种监督学习算法，常用于分类和回归任务。它的主要目标是找到一个最优的超平面，可以将不同类别的样本数据有效地分开。在SVM中，每个样本被表示为特征空间中的一个点，并且每个样本都被赋予一个类别标签。SVM的目标是找到一个超平面，它可以最大程度地将不同类别的样本点分开，并且离超平面最近的样本点被称为支持向量。

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？

1.准确率：准确率是最简单和直观的评价指标，表示分类器正确预测的样本数占总样本数的比例。准确率高表示分类器的预测结果更接近真实结果，但在不平衡类别分布的情况下可能会受到误导。

2.精确率：精确率衡量分类器预测为正例的样本中有多少是真正的正例。精确率较高表示分类器将负例误判为正例的可能性较低，但它不考虑分类器是否漏掉了一些真正的正例。

3.召回率：召回率衡量分类器能够正确识别出多少真正的正例。召回率较高表示分类器能够找到更多的正例，但它不考虑分类器是否将一些负例错误地分类为正例。

4.AUC：AUC是基于ROC曲线的一个评价指标，衡量分类器在不同阈值下的分类性能。AUC值越大表示分类器具有更好的分类能力。

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.均方误差：预测值与真实值之间的平均偏差的平方，将预测值与真实值的差值平方后求平均值。MSE越小表示预测结果与真实值越接近。

2.均方根误差：均方误差的平方根，使用与目标变量相同的单位来表示误差。RMSE越小表示预测结果与真实值之间的误差越小。

3.平均绝对误差：预测值与真实值之间的平均绝对偏差，是预测值与真实值之间的绝对差值的平均值。它越小表示预测结果与真实值之间的绝对误差越小。

4.相关系数：预测值与真实值之间的线性关系强度，取值范围从-1到1。