**本科生课程论文**

课程名称 企业工程实践

教师姓名 阎继宁

本科生姓名 张宇豪

本科生学号 20181001905

本科生专业 数据科学与大数据技术

所在院系 计算机学院

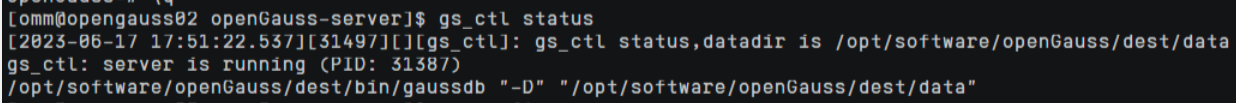
日期: 2023年6月17日

# 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：

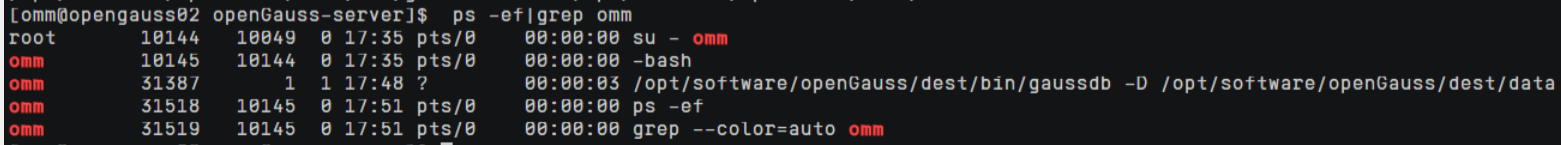
任务一：数据库状态验证

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？

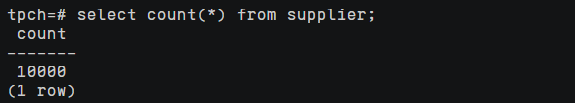
在不同的操作系统和硬件环境下，往往需要有针对性的优化配置，有时候就需要根据实际情况进行源码编译安装。通过源码编译安装，可以更好地满足用户的特定需求或者应用场景，并根据具体的硬件架构和配置选择所需的功能模块和优化参数，进一步提升OpenGauss的性能和稳定性，从而更好地满足用户的需求和要求。另外，源码编译安装OpenGauss还能更好地理解OpenGauss的运行原理和机制，使用户更好地学习和应用OpenGauss。

# 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;;

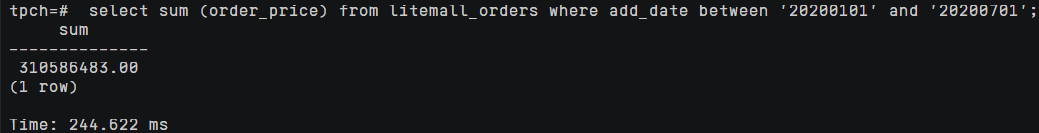


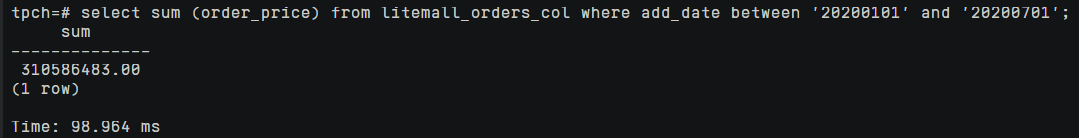
任务二：行存表与列存表执行效率对比

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';

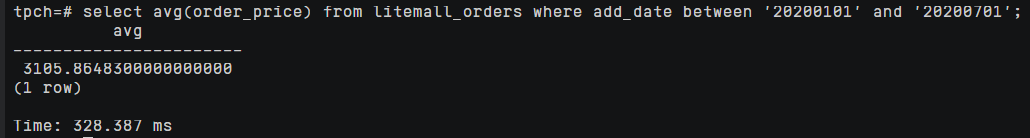


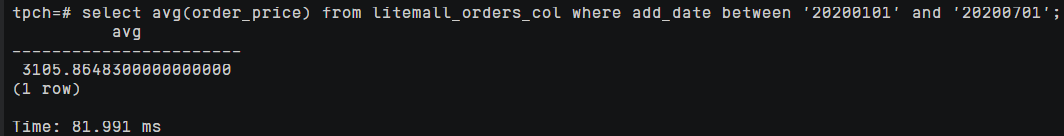


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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';

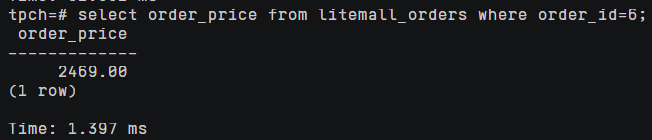


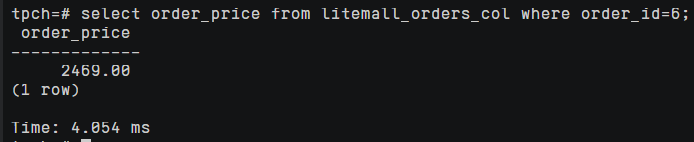


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

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

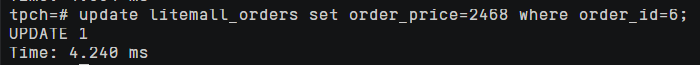


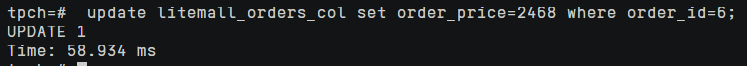


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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

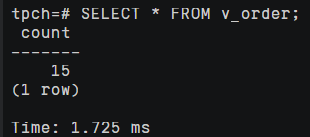
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



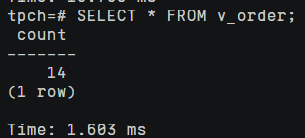
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



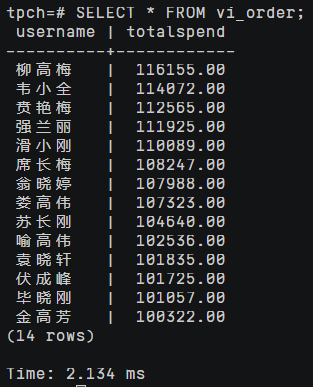
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



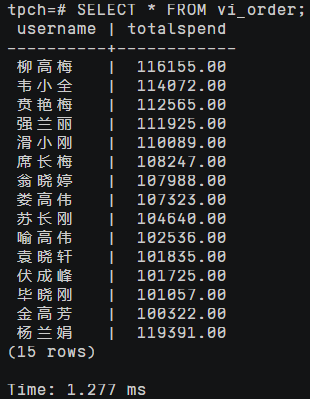
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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？

opengauss行存表和列存表的存储方式不同，在执行相同的SQL语句时，由于其物理存储方式的不同，会导致执行时间不同。

一般来说，行存表在执行INSERT和UPDATE语句时比较高效，因为这些语句通常需要更新整行数据。而列存表在执行SELECT语句时比较高效，因为SELECT通常只需要获取某些列的数据。

在执行大量的聚合类查询时，列存表往往更高效，因为列存表可以减少IO访问次数和CPU消耗。例如，在执行带有GROUP BY和SUM函数的查询时，列存表可以更快地统计结果。而在执行简单的单行查询等操作时，行存表由于只需要读取少量数据，效率往往更高。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

1. 刷新方式不同

全量物化视图每次要进行全量刷新，即对所有基表进行重新计算和读取。增量物化视图只需要对那些发生了变化的基表进行刷新，可以有效地提高刷新效率。

2. 刷新时间不同

全量物化视图在处理多个基表时，需要对所有基表进行全量刷新，因此刷新时间相对较长。而增量物化视图只需要刷新那些发生变化的基表，因此刷新时间相对较短。

3. 存储方式不同

为了支持快速刷新，增量物化视图需要维护一些特殊的数据结构来存储基表记录的变化信息。而全量物化视图则不需要维护这些信息。

4. 查询效率不同

由于增量物化视图只需要刷新那些发生变化的基表，因此在查询时可以快速返回结果。而全量物化视图需要对所有基表进行全量刷新，因此查询效率相对较低。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

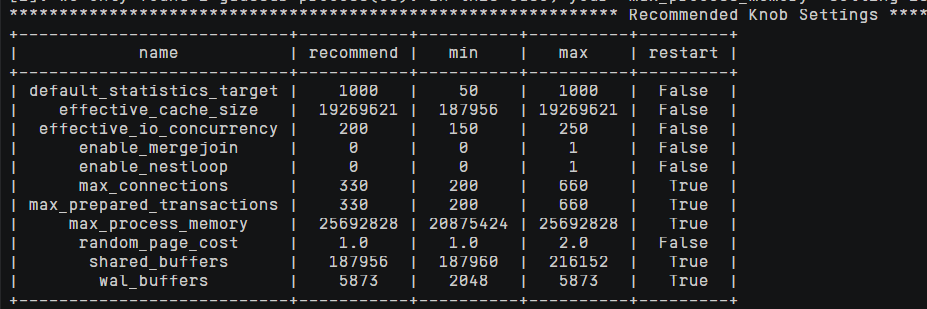
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

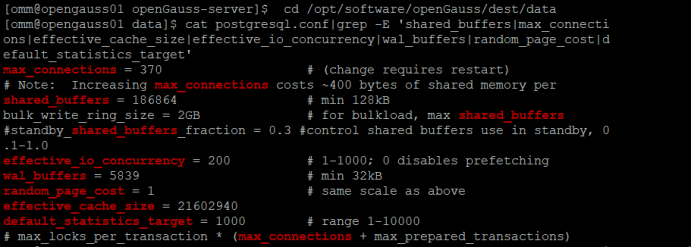
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

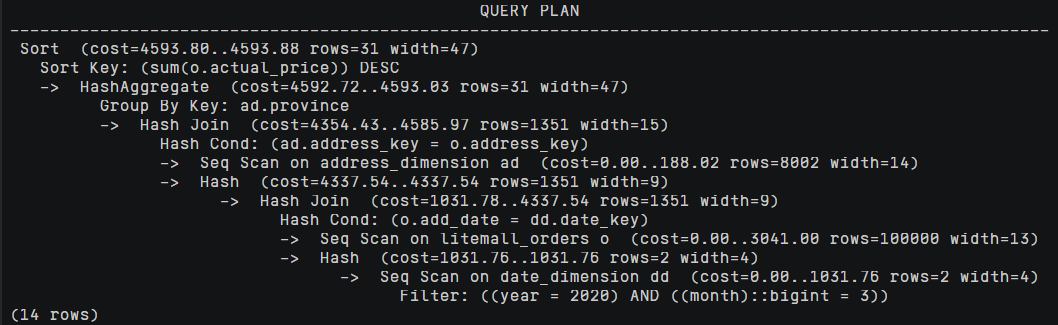
AND dd.year = 2020

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

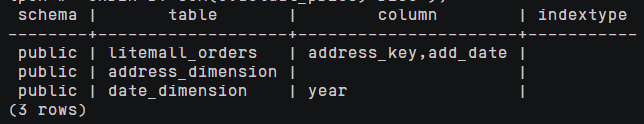
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

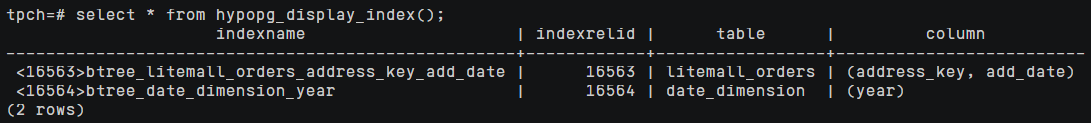
GROUP BY ad.province

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。

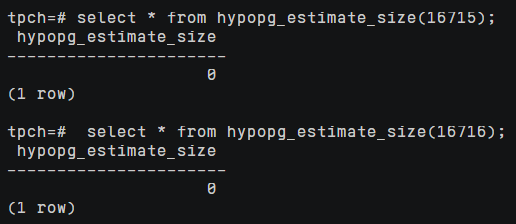
select \* from hypopg\_display\_index();



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

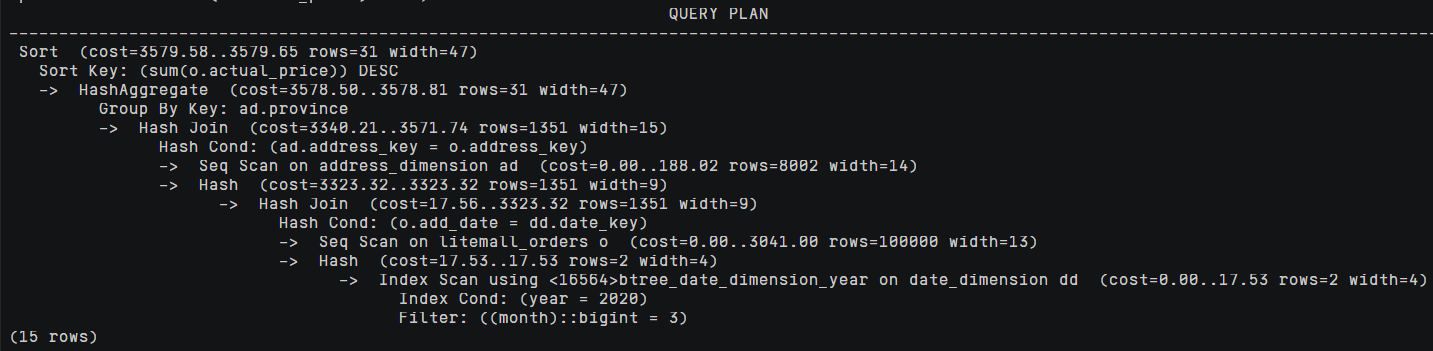
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

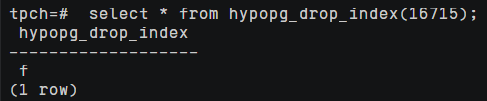
GROUP BY ad.province

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



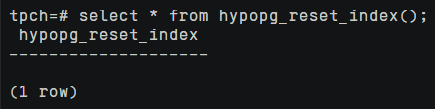
6. 删除某一个索引虚拟列，将执行结果截图。

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



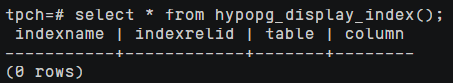
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

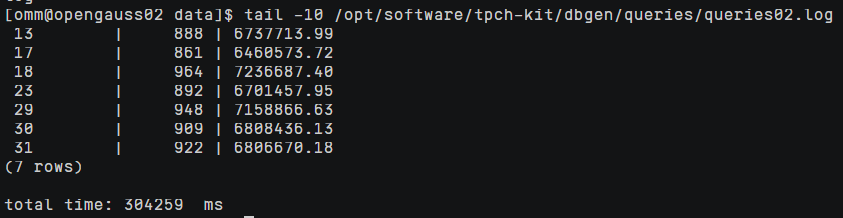
select \* from hypopg\_display\_index();



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：

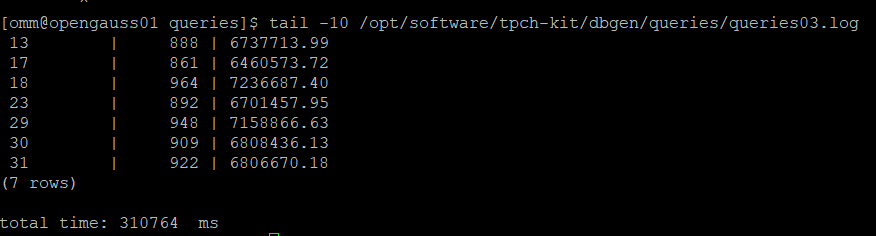
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log



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



使用gs\_index\_advise对query.sql中语句计算并建立索引，但没有显著效果。

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？

优化了shared\_buffers, max\_connections, effective\_cache\_size, effective\_io\_concurrency, wal\_buffers, random\_page\_cost, default\_statistics\_target，为了提升数据库事务操作的执行效率

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？

OpenGauss数据库使用索引可以提高查询效率，因为索引可以加速WHERE语句中的条件筛选和排序操作。

1. 合理设计数据模型。采用合理的数据模型可以减少表之间的关联操作，降低IO消耗和数据库负载。

2. 使用分区表。对于大数据量的表，可以使用分区表来分割数据，提高查询效率。

3. 优化SQL语句。通过优化SQL语句，如避免使用SELECT \*、避免在WHERE子句中使用函数等，可以提高查询效率。

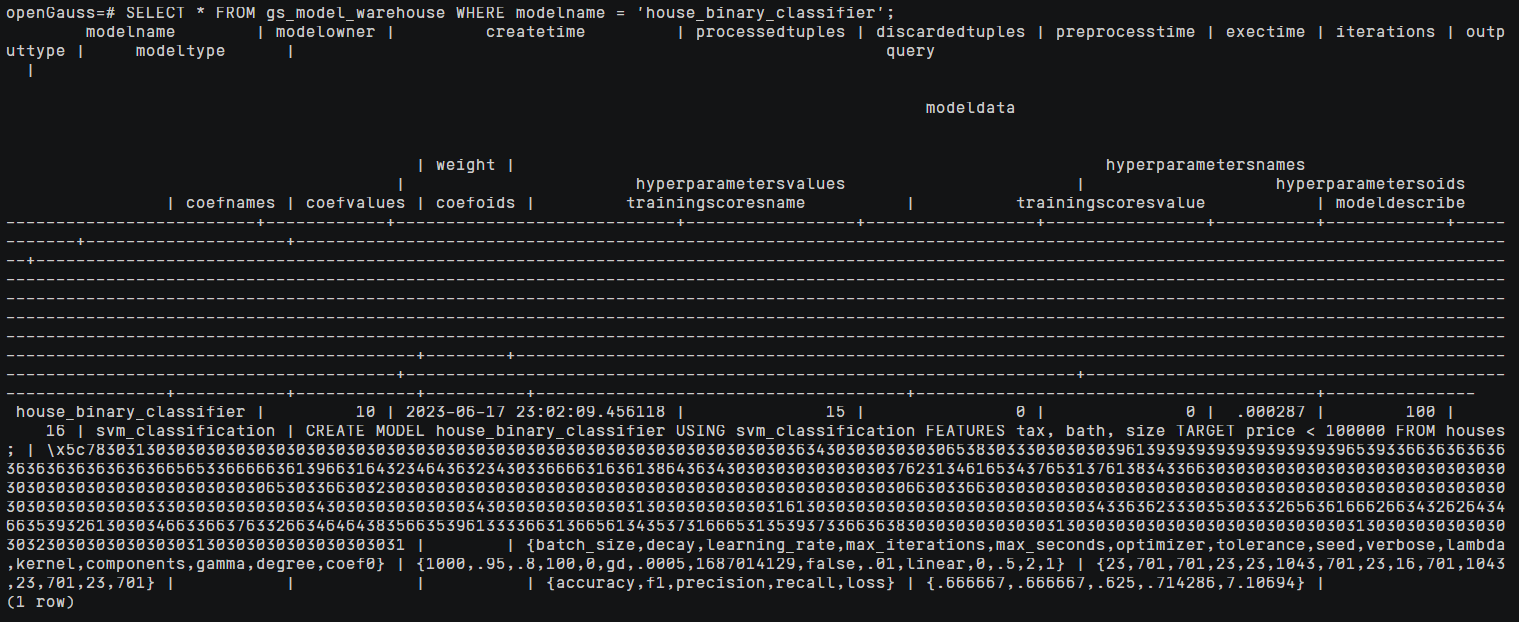
4. 增加硬件资源。增加硬件资源，如加内存、加CPU等，可以提高数据库处理数据的能力，提高查询效率。

5. 合理配置数据库参数。通过合理配置数据库参数，如调整max\_connections、shared\_buffers、work\_mem等参数，可以进一步提高数据库效率和减少资源占用。

# 关卡四、openGauss的DB4AI特性应用

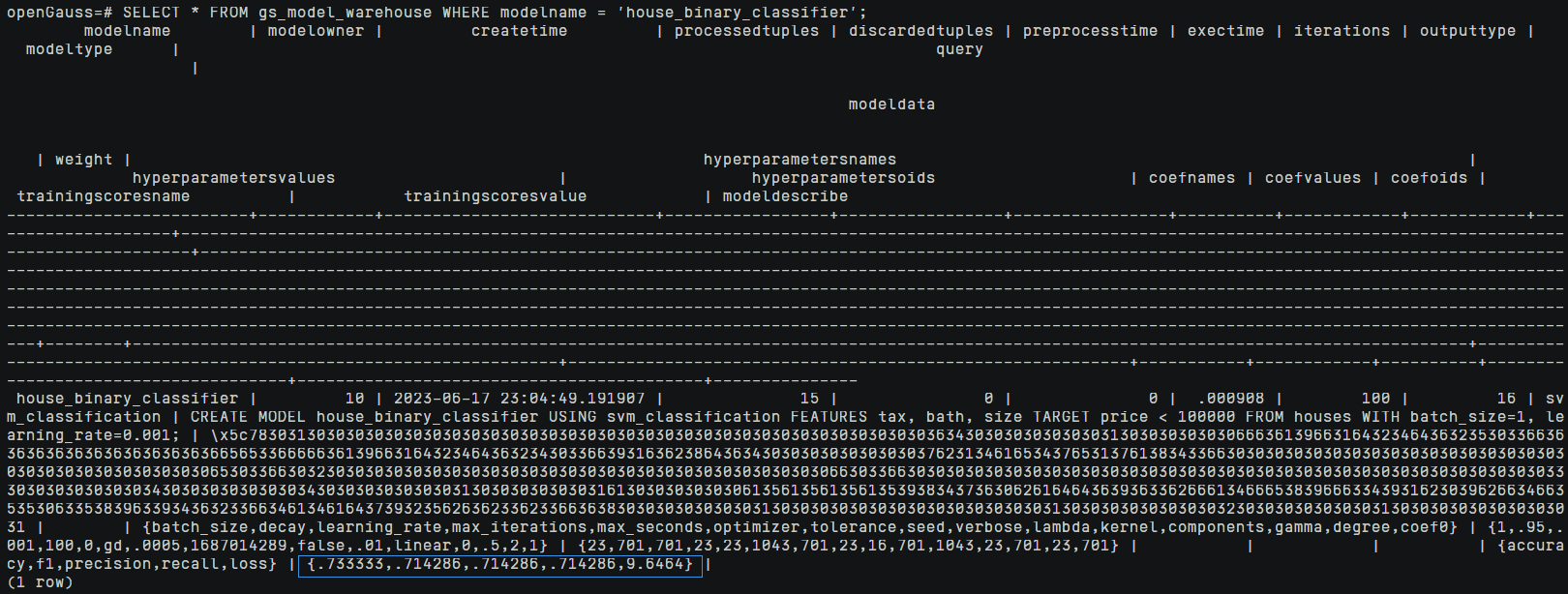
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



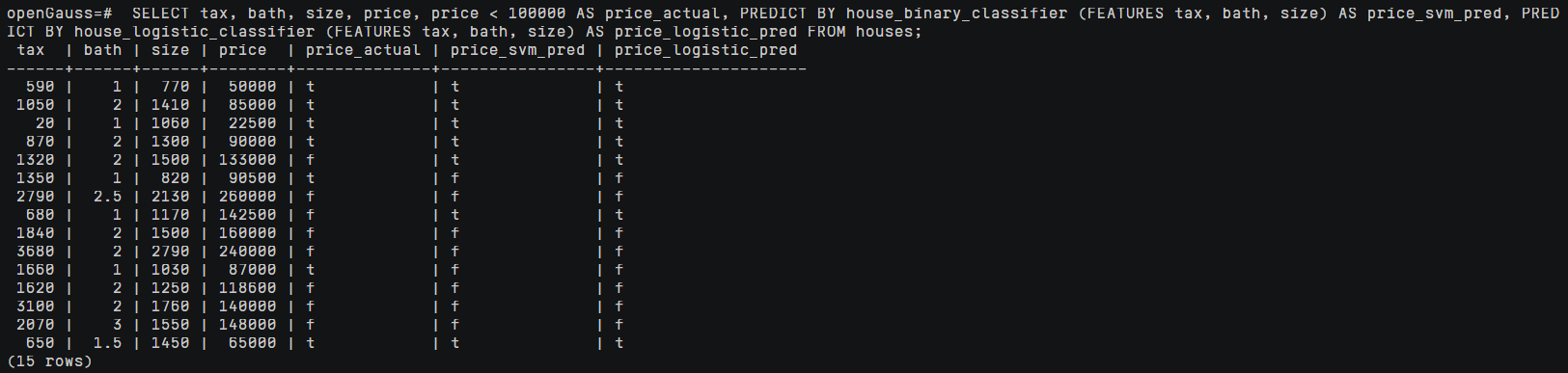
任务二：观察新模型的信息，将执行结果截图。

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？

1. 目标不同

分类模型的目标是预测一个给定样本属于哪一类别，即离散型输出。回归模型的目标是根据给定的输入数据，预测一个连续型输出结果。

2. 建模方式不同

分类模型建立的是离散分类模型，它的处理方式是根据实际数据去学习分类规律。而回归模型建立的是数学模型，它不是去学习分类规律，而是拟合输入和输出之间的函数关系。

3. 变量类型不同

分类模型的自变量通常是一些离散或分类变量，因为只考虑样本所属的类别，不考虑数值大小。回归模型的自变量通常是连续型变量。

4. 评估方式不同

分类模型的评估指标包括准确率、精确率、召回率等；回归模型的评估指标包括均方误差、平均绝对误差等。

实践思考题2：什么是SVM算法？

SVM是一个二类分类器，它的目标是找到一个超平面，使用两类数据离超平面越远越好，从而对新的数据分类更准确，即使分类器更加健壮。

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（Accuracy）：准确率指的是分类器预测正确的样本数占总样本数的比例。

2. 精确率（Precision）：精确率是指分类器预测为正样本并且分类器预测正确的正样本数占预测集中所有预测为正样本的样本数的比例。

3. 召回率（Recall）：召回率是指分类器预测为正样本并且分类器预测正确的正样本数占实际上是正样本的所有样本数的比例。

4. F1度量值（F1 Measure）：精确率和召回率可以结合成为F1度量值，F1度量值是准确率和召回率的调和平均数，它可以综合评价分类器的效果。

5. ROC曲线（Receiver Operating Characteristic Curve）：ROC曲线是以假正率为横轴，真正率为纵轴画出的图形，它可以用来评价分类器的效果。ROC曲线下的面积（AUC）越大，分类器的效果越好。

6. PR曲线（Precision-Recall Curve）：PR曲线是以召回率为横轴，精确率为纵轴画出的图形，它可以用来评价分类器的效果。PR曲线下的面积（AUC）越大，分类器的效果越好。

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？

1. 平均绝对误差（Mean Absolute Error, MAE）：平均绝对误差指预测值与真实值之差的绝对值的平均值。它对预测值和真实值的差异进行了均等化处理。

2. 均方误差（Mean Squared Error, MSE）：均方误差指预测值与真实值之差的平方的平均值。它度量的是预测值与真实值之间的平均残差的大小，更注重较大的误差值。

3. R平方（R-squared）： R平方也称为决定系数，它指的是数据拟合优度的一种度量，可以评估分析建模的好坏。它是真实值和预测值之间的平均差的比率。

4. 均方根误差（Root Mean Squared Error, RMSE）：均方根误差是均方误差的平方根，它很像标准偏差，但是它是对预测值和真实值之间差异的平均性进行评估。

5. 平均绝对百分误差（Mean Absolute Percentage Error, MAPE）：平均绝对百分误差是百分误差的平均数。百分误差指的是预测值和真实值之间的差异和真实值之间的比率。