openGauss AI特性创新实践课



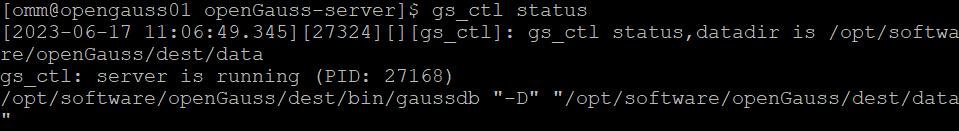
华为技术有限公司

# 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：

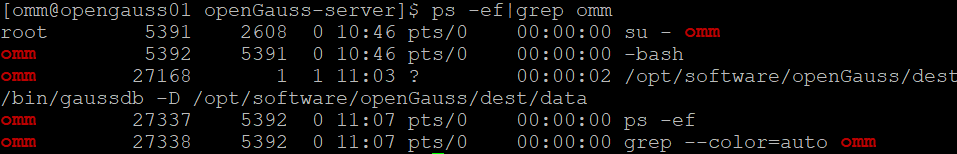
任务一：数据库状态验证

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？

通过源码编译和安装数据库的过程可以提供更大的灵活性和控制权。

定制化需求，通过源码编译和安装数据库，可以自定义编译选项，以满足特定的硬件、操作系统或性能需求。可以根据自己的要求进行优化和配置，以获得最佳性能和功能。

获取最新功能，使用源码编译，可以访问最新版本的数据库软件，并及时获得最新的功能和修复程序。

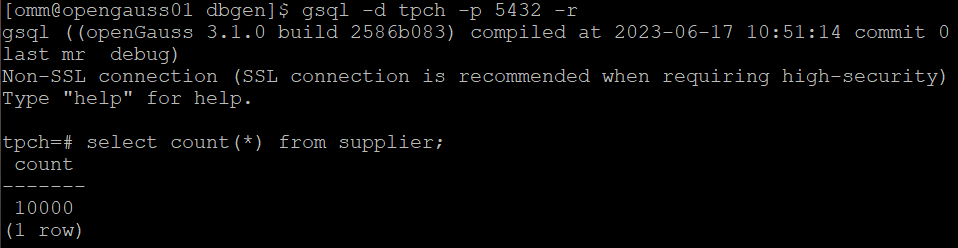
安全性和可信度，通过从官方源获取源码，可以确保所安装的数据库是来自可信的来源，并且可以审查和验证源代码的安全性。

# 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，并将结果进行截图：

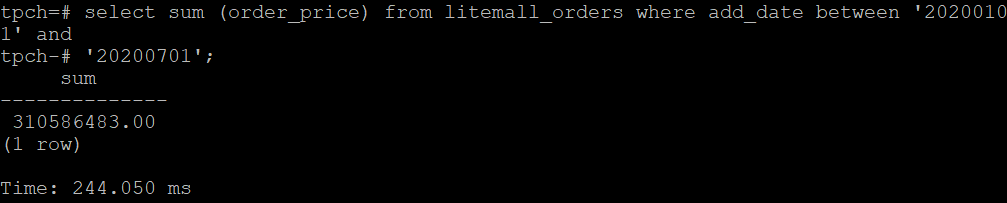
select count(\*) from supplier;;



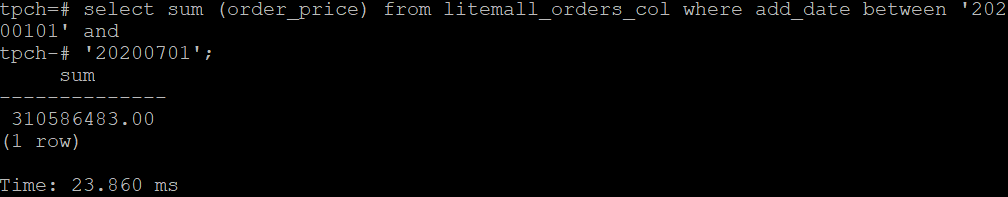
任务二：行存表与列存表执行效率对比

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';

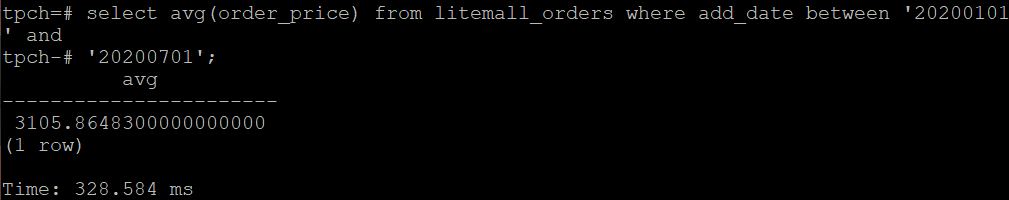


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

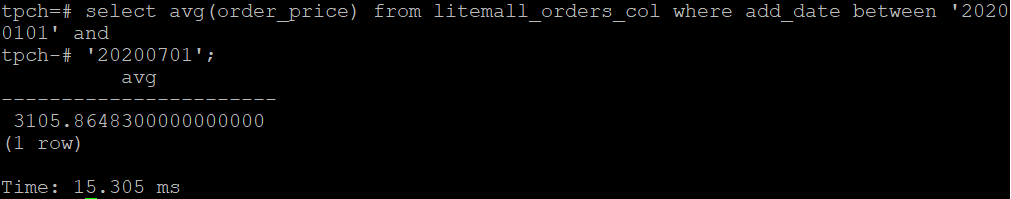


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

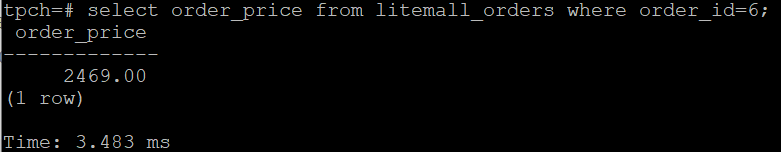
select avg (order\_price) from litemall\_orders where add\_date between '20200101' and '20200701';



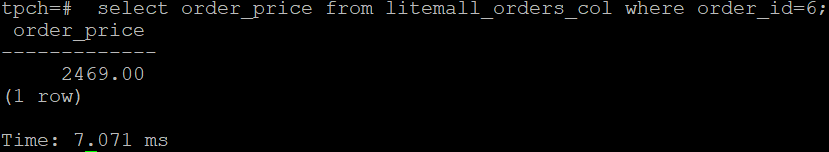
select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';



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。

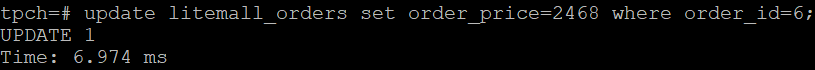
select order\_price from litemall\_orders where order\_id=6;

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;

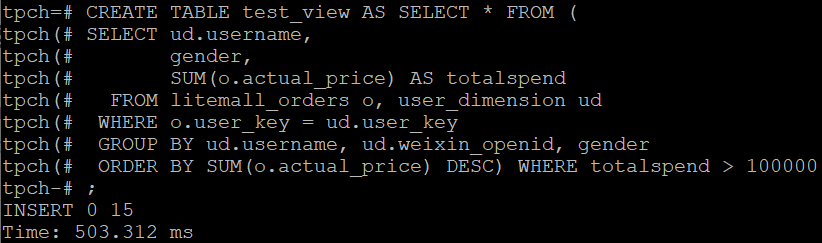
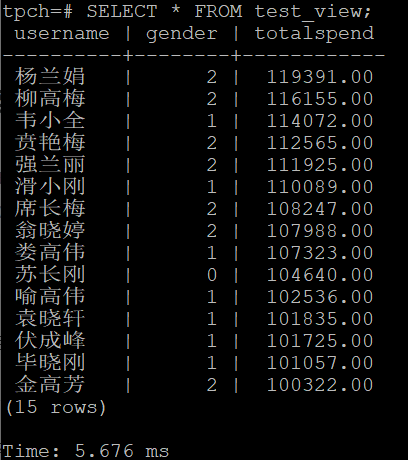


update litemall\_orders\_col set order\_price=2468 where order\_id=6;

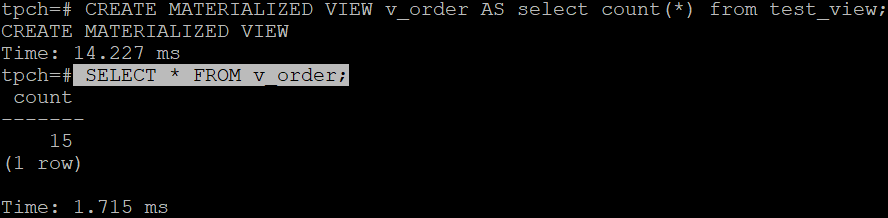
任务三：物化视图的使用

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;

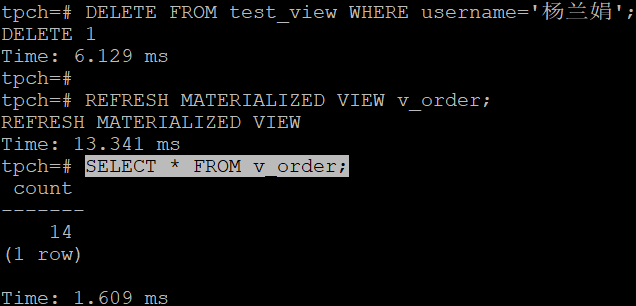


2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

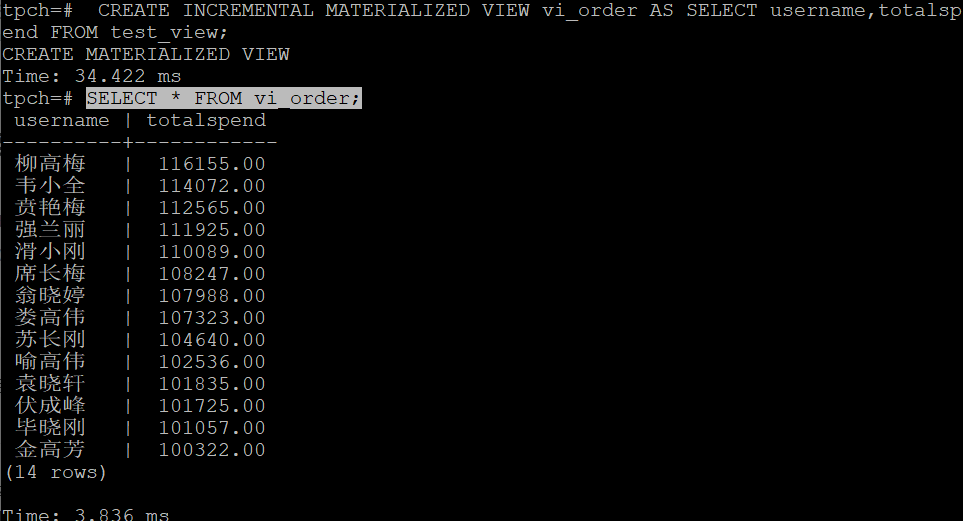
SELECT \* FROM v\_order;

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;

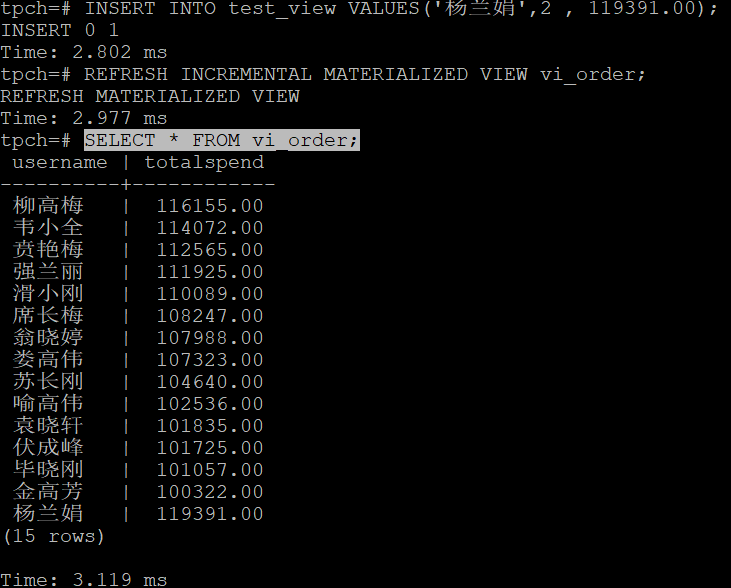
 4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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？

行存表和列存表是两种不同的数据库存储方式，它们在执行相同的SQL语句时执行的时间不同，

这是由它们内部的数据组织方式所决定的。行存表将数据按照行的方式存储在磁盘上，每个数

据行包含多个列的值。列存表则将数据按照列的方式存储在磁盘上，每个列的值被存储在一起。

行存表效率更高的SQL语句类型：需要读取整行数据的SQL语句，需要频繁更新或插入整行的据的SQL语句。

列存表效率更高的SQL语句类型：只需要读取特定列数据的SQL语句，需要进行聚合查询、分组操作或者对特定列进行计算的SQL语句。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

数据更新方式：全量物化视图是通过完全重新计算或重新加载整个视图的数据来进行更新。增量物化视图则是通过仅计算或加载源数据的增量变化来进行更新，它只处理源数据表中发生变化的部分，而不需要重新计算或加载整个视图的数据。

更新频率和实时性：全量物化视图通常具有较低的更新频率，因为在每次更新时需要重新计算或重新加载整个视图，这可能需要较长的时间。增量物化视图可以具有较高的更新频率，因为它只需要处理源数据表中的增量变化。

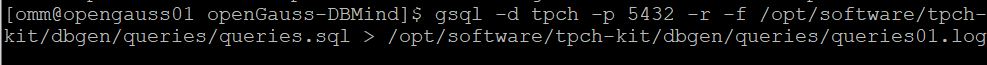
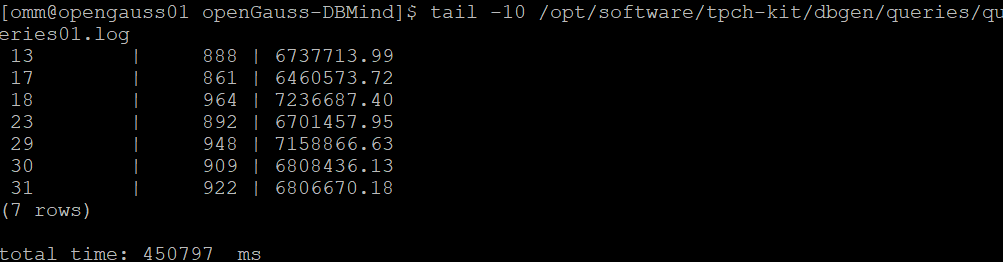
存储和性能：全量物化视图通常需要占用较大的存储空间，因为它们存储了完整的计算结果。增量物化视图通常需要较少的存储空间，因为它们仅存储了源数据的增量变化。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

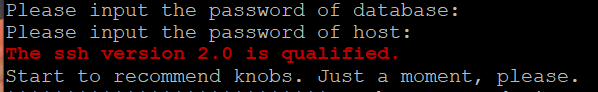
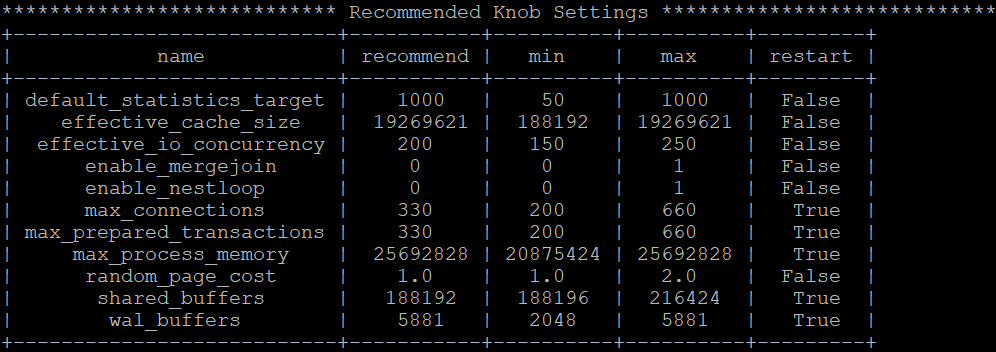
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

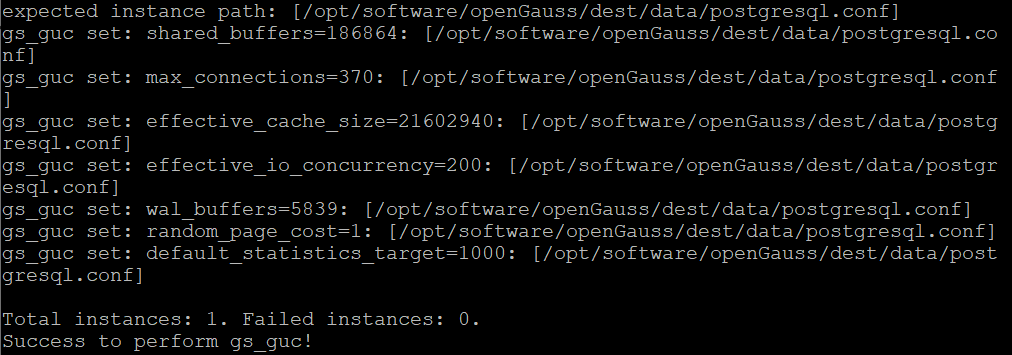
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log



2. 使用omm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

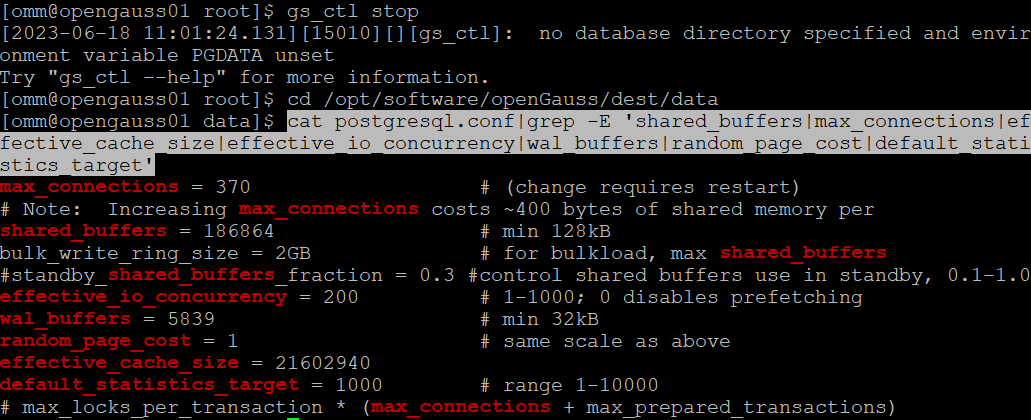




3.重启完成后，获取参数值：

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

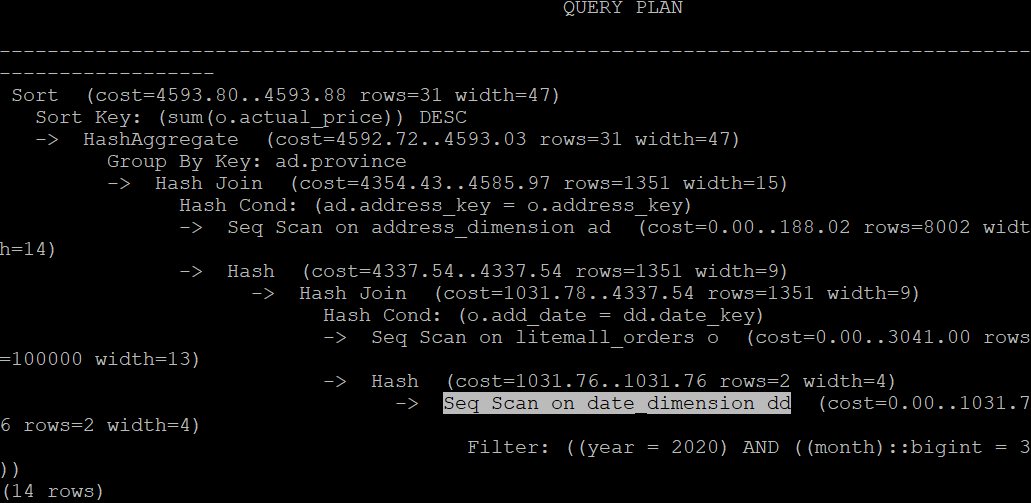
AND o.add\_date = dd.date\_key

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

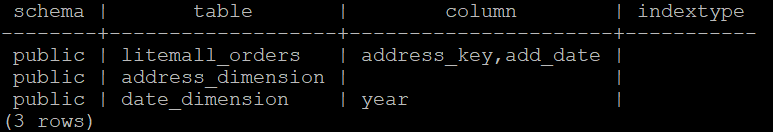
AND o.add\_date = dd.date\_key

AND dd.year = 2020

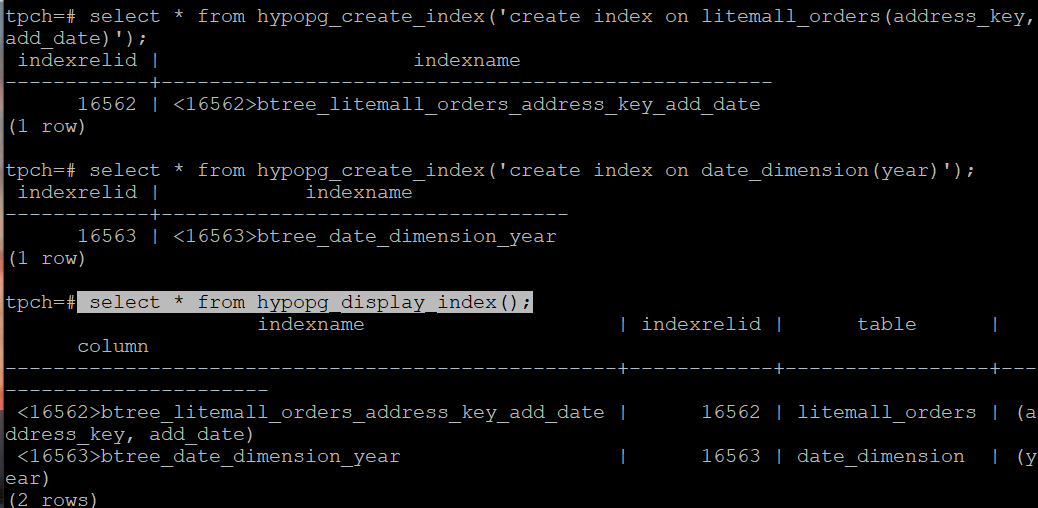
AND dd.month = 3

GROUP BY ad.province

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



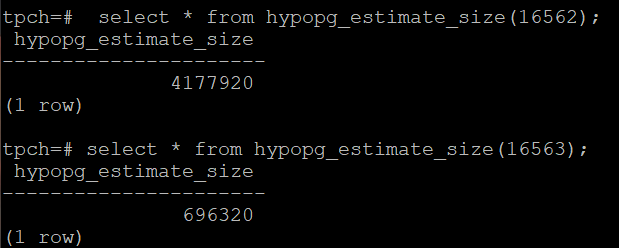
3. 查看创建的虚拟索引列，将执行结果截图。

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

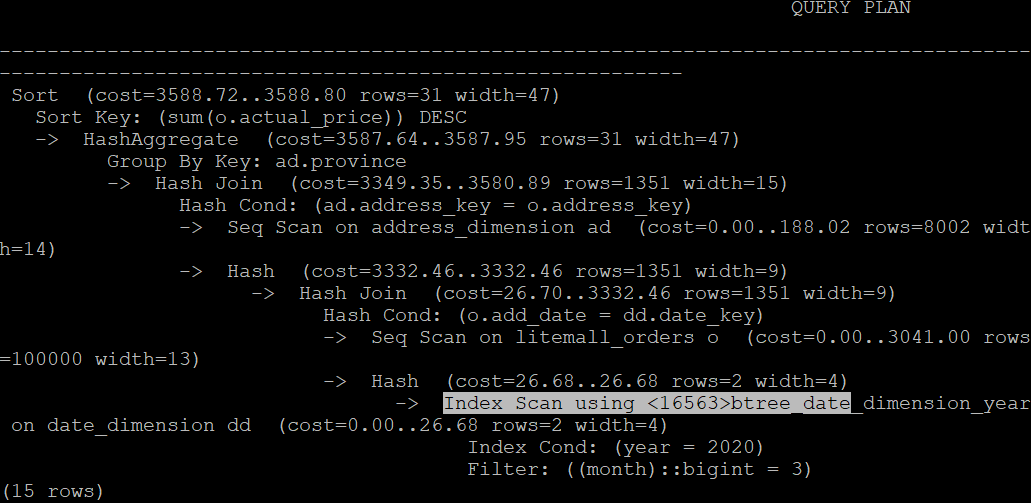
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

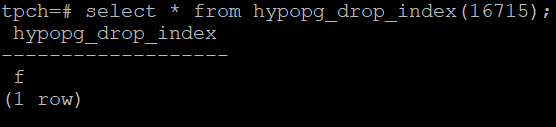
GROUP BY ad.province

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

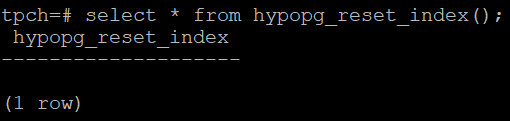


6. 删除某一个索引虚拟列，将执行结果截图。

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

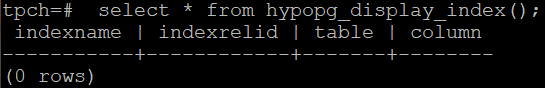


7. 删除所有索引虚拟列，将执行结果截图。

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

8. 查看索引虚拟列，将执行结果截图。

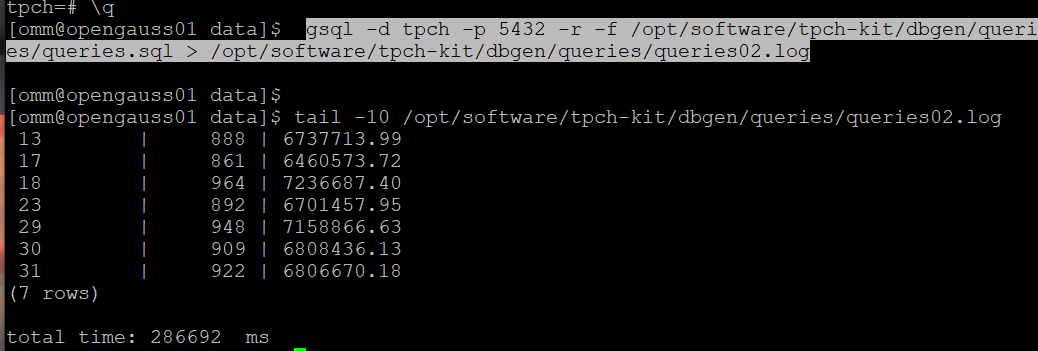
select \* from hypopg\_display\_index();



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？

对以下参数进行了优化：shared\_buffers、max\_connections、effective\_cache\_size、effective\_io\_concurrency 、wal\_buffers、random\_page\_cost、default\_statistics\_target。

这些参数的优化涉及到数据库的内存配置、并发连接数、磁盘IO操作等方面。通过合理调整这些参数，可以根据具体的硬件配置、工作负载和性能需求，提高数据库的整体性能和响应能力。

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？

索引在执行SQL时具有以下好处：

提高查询性能：索引可以加快数据检索的速度，特别是在大型表或包含大量数据的表中。它们提供了快速访问数据的路径，减少了全表扫描的需求，从而加快了查询的执行速度。

减少IO操作：索引存储了数据的逻辑排序，使得数据库可以更快地定位和读取所需的数据页，减少了磁盘IO操作的次数，提高了数据访问的效率。

改善连接操作性能：当进行连接查询（JOIN）时，索引可以提供连接字段的快速匹配，减少了连接操作所需的计算量和IO操作，从而加速了连接查询的执行。

支持排序和聚合操作：索引可以提供数据的逻辑排序，使得排序和聚合操作更加高效。当SQL语句需要按特定字段排序或进行聚合函数计算时，索引可以减少排序或聚合所需的处理时间。

除了使用索引和参数来优化数据库性能外，还有其他方面可以进行数据库优化，包括：

数据库架构设计：良好的数据库架构设计可以优化数据表之间的关系、索引的选择和使用、数据划分等，从而提高数据库的性能和可扩展性。

查询优化：通过分析和优化查询语句，如使用适当的连接方式、使用合适的过滤条件、避免使用不必要的子查询等，可以改善查询的执行计划，提高查询性能。

缓存和缓冲区管理：通过使用适当的缓存策略和缓冲区管理技术，可以减少磁盘IO操作，提高数据访问速度。

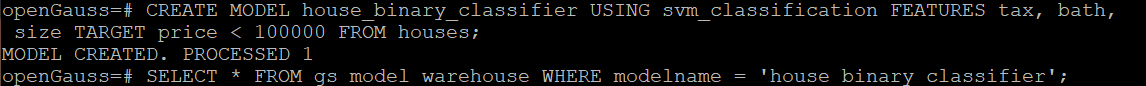
硬件和系统配置：优化数据库服务器的硬件配置和操作系统参数设置，如适当分配内存、磁盘配置、网络优化等，可以改善数据库的性能。

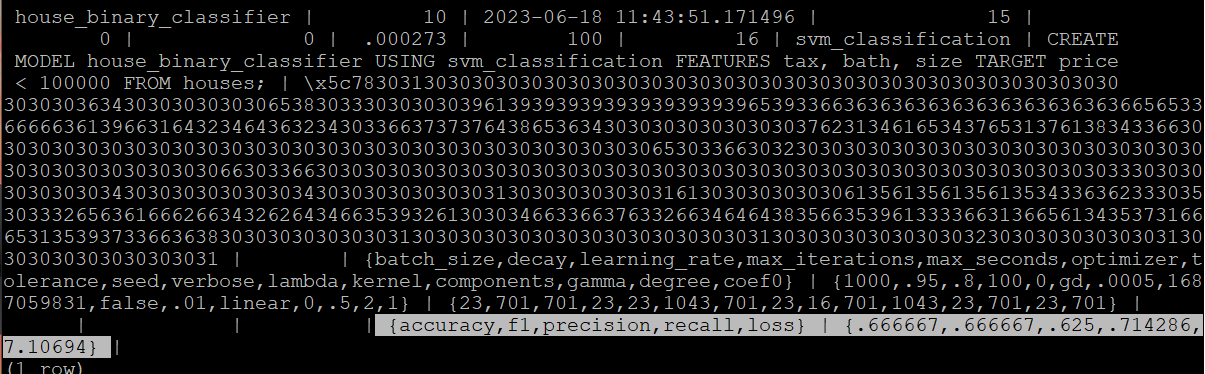
数据备份和恢复策略：合理的备份和恢复策略可以保证数据的安全性，并在需要时快速恢复数据库的运行。

定期维护和优化：定期进行数据库维护任务，如重新构建索引、优化表结构、清理过期数据等，可以保持数据库的良好性能。

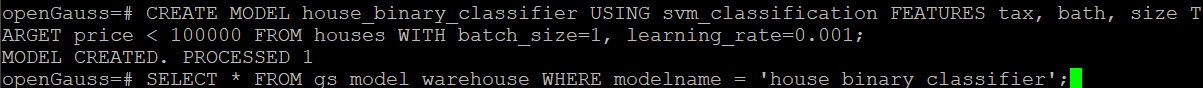
# 关卡四、openGauss的DB4AI特性应用

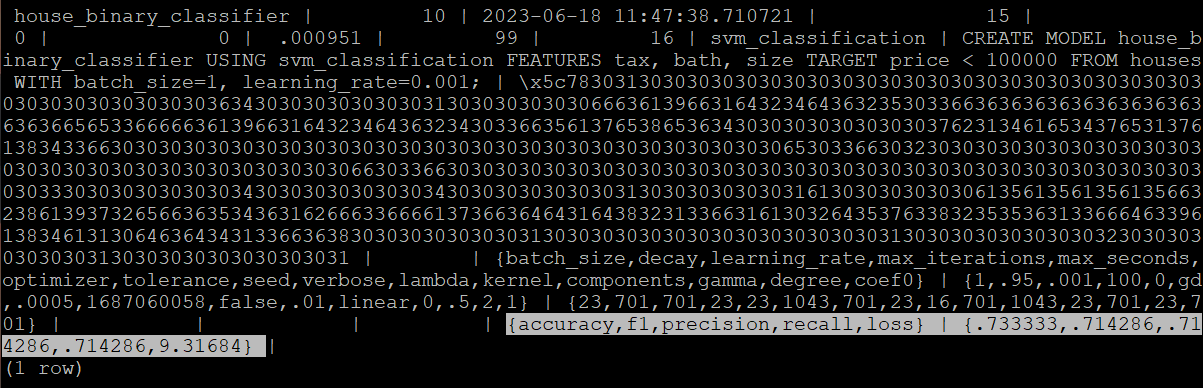
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



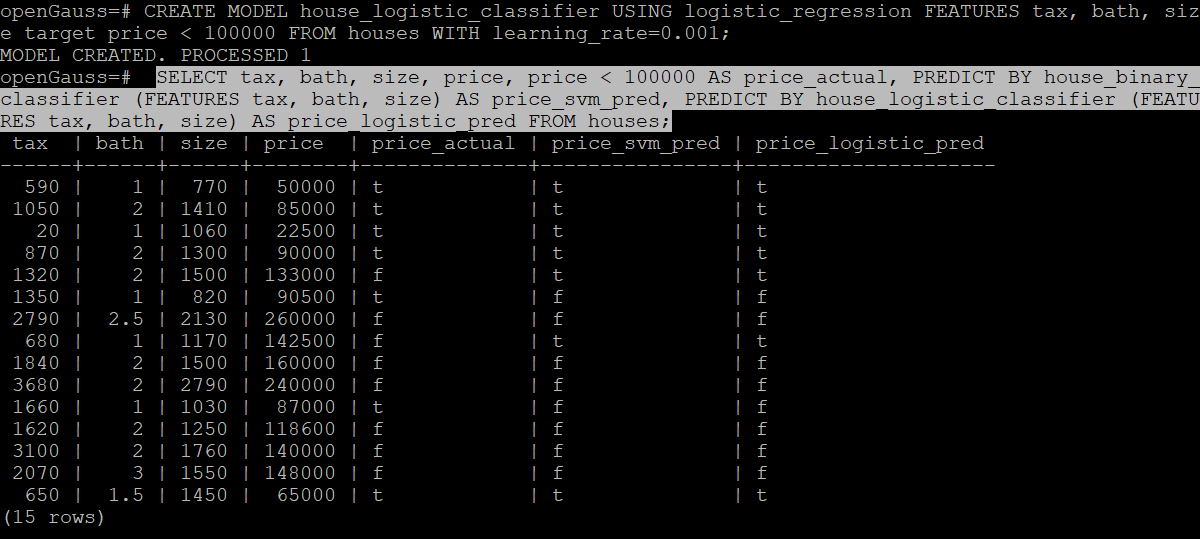
任务二：观察新模型的信息，将执行结果截图。

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？

输出结果类型：分类模型输出的是离散的类别标签或类别概率，而回归模型输出的是连续的数值。

目标类型：分类模型适用于解决分类问题，即将输入数据分为不同的类别；回归模型适用于解决回归问题，即对输入数据进行数值预测。

模型选择：根据问题的性质和目标，选择合适的模型类型。如果要进行分类预测，则选择分类模型；如果要进行数值预测，则选择回归模型。

模型评估：分类模型通常使用准确率、精确度、召回率、F1分数等指标来评估性能；而回归模型通常使用均方误差、均方根误差、平均绝对误差等指标来评估预测的准确性。

实践思考题2：什么是SVM算法？

支持向量机是一种用于分类和回归的监督学习算法。它通过构建一个最优的超平面或者间隔最大化来实现分类任务。

SVM 的主要思想是将数据映射到高维空间，找到一个最优的超平面来分隔不同类别的数据点。超平面是一个N-1维的线性子空间，其中N是数据的特征数目。当数据线性可分时，SVM通过最大化类别之间的间隔，选择一个最优的超平面，使得同一类别的数据点都在超平面的同一侧。这个最优的超平面由支持向量决定，它们是离超平面最近的数据点。

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？

准确率（Accuracy）：准确率是最直观的评价指标，表示分类正确的样本数占总样本数的比例。它衡量了分类器对所有样本的整体分类准确程度。准确率越高，分类器的性能越好。

精确度（Precision）：精确度是衡量分类器在预测为正例的样本中有多少真正是正例的指标。它表示分类为正例的样本中，正确预测为正例的比例。精确度高意味着分类器将负例误判为正例的概率较低。

召回率（Recall）：召回率衡量分类器对所有正例样本中有多少被正确预测为正例的指标。它表示正确预测为正例的样本数占实际正例样本数的比例。召回率高意味着分类器将正例样本正确找出的能力较强。

F1值（F1 Score）：F1值是精确度和召回率的调和平均值，综合考虑了分类器的准确度和召回能力。F1值越高，表示分类器在平衡精确度和召回率方面的性能越好。

特异度（Specificity）：特异度是衡量分类器对所有负例样本中有多少被正确预测为负例的指标。它表示正确预测为负例的样本数占实际负例样本数的比例。特异度高意味着分类器将负例样本正确排除的能力较强。

假阳性率（False Positive Rate，FPR）：假阳性率衡量分类器在所有负例样本中错误预测为正例的比例。它是特异度的补数，表示分类器将负例误判为正例的概率。

假阴性率（False Negative Rate，FNR）：假阴性率衡量分类器在所有正例样本中错误预测为负例的比例。它是召回率的补数，表示分类器无法将正例样本正确找出的概率。

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？

均方误差（Mean Squared Error，MSE）：均方误差是回归问题中最常用的评价指标之一，它衡量预测值与真实值之间的平均差异的平方。MSE越小，表示回归模型的拟合效果越好。

均方根误差（Root Mean Squared Error，RMSE）：均方根误差是MSE的平方根，它与原始数据的单位相同，因此更易于解释和理解。RMSE同样用于衡量预测值与真实值之间的平均差异。

平均绝对误差（Mean Absolute Error，MAE）：平均绝对误差是预测值与真实值之间绝对差值的平均值。它衡量了预测值与真实值之间的平均偏差，不受离群值的影响。

决定系数（Coefficient of Determination，R²）：决定系数衡量回归模型对因变量变化的解释能力。它表示模型能够解释的因变量方差占总方差的比例，取值范围在0到1之间，越接近1表示模型的拟合效果越好。

相关系数（Correlation Coefficient，R）：相关系数衡量预测值与真实值之间的线性关系强度和方向。相关系数取值范围在-1到1之间，当相关系数为正值时表示正相关，为负值时表示负相关，接近0表示无相关性。

解释方差（Explained Variance）：解释方差衡量回归模型能够解释的响应变量的方差。它表示模型能够解释的数据方差所占比例，越接近1表示模型的拟合效果越好。