openGauss AI特性创新实践课



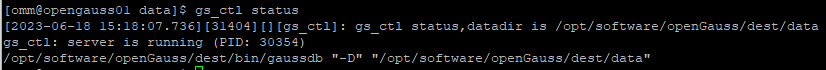
华为技术有限公司

# 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：

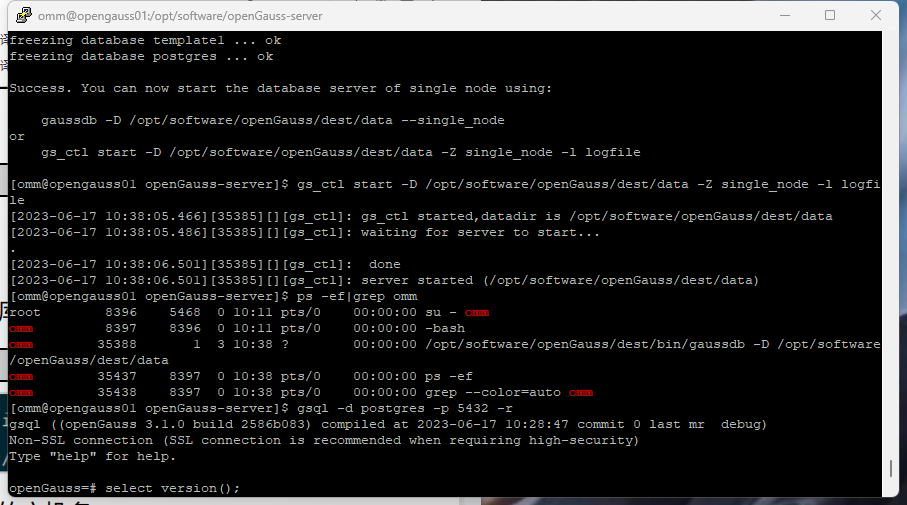
任务一：数据库状态验证

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：通过源码编译安装数据库的主要原因是为了更好地控制数据库的配置和性能。当我们从源代码编译和安装数据库时，可以选择编译选项和配置参数，以满足特定需求。这样可以确保数据库在系统上以最佳性能运行，并且可以避免不必要的功能和组件的安装。

此外，通过源码编译和安装数据库还可以提供更高的安全性。由于源代码是公开的，因此可以更容易地检查和修复潜在的安全漏洞。此外，通过自己编译和安装数据库，我们可以确保我们使用的是最新版本，并且可以及时更新和修复任何已知的漏洞。

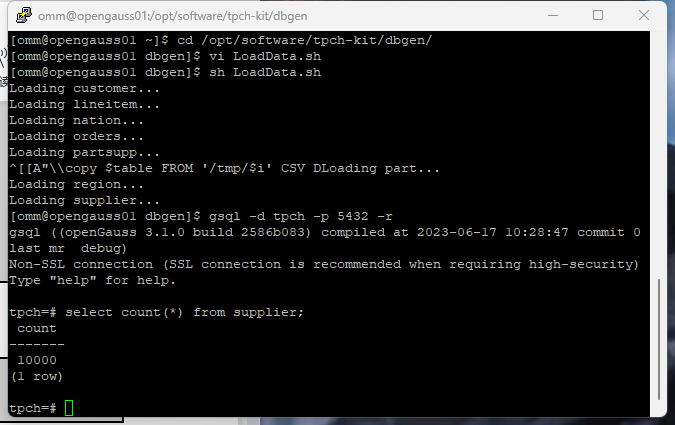
总之，通过源码编译和安装数据库可以提供更好的性能、安全性和灵活性，但需要更多的技术知识和时间。如果不熟悉编译和安装过程，或者需要快速部署数据库，那么使用预编译的二进制包可能更为方便。

# 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：

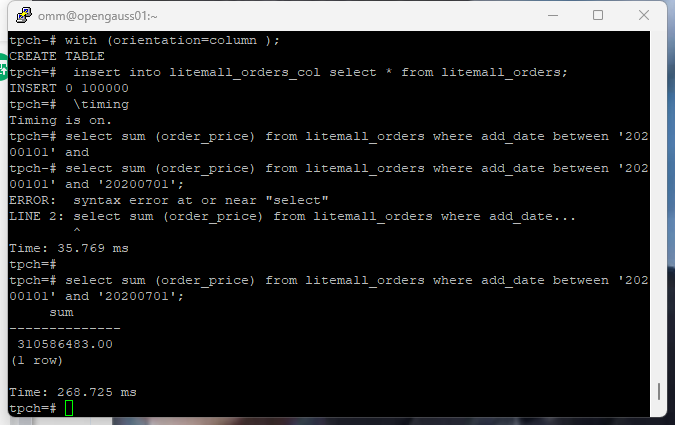
select count(\*) from supplier;;



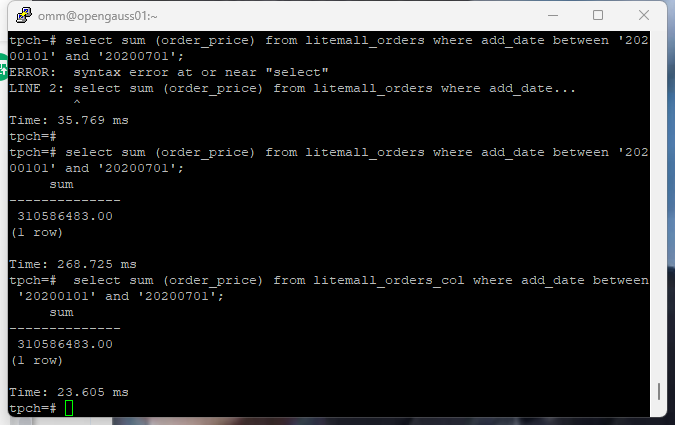
任务二：行存表与列存表执行效率对比

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

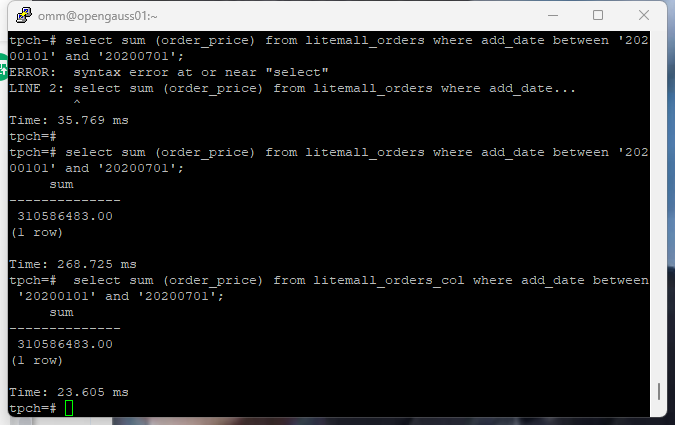
select sum (order\_price) from litemall\_orders where add\_date between '20200101' and '20200701';



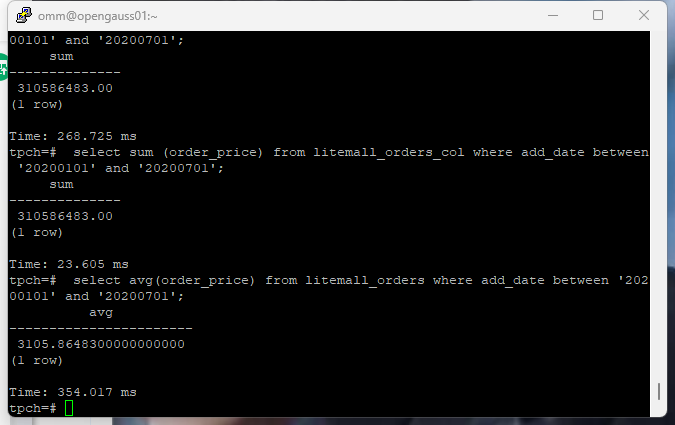
select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

 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';

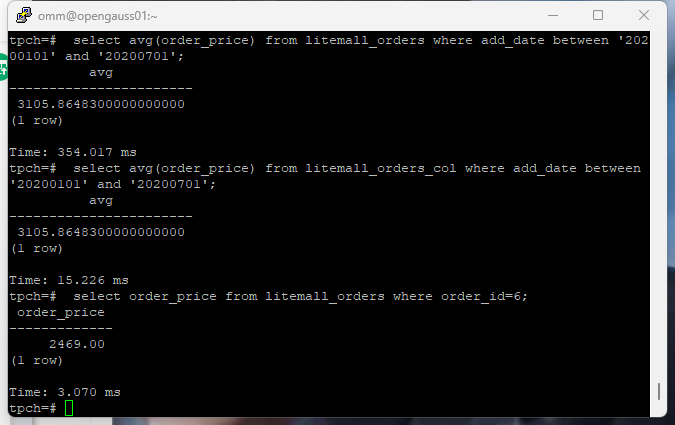


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

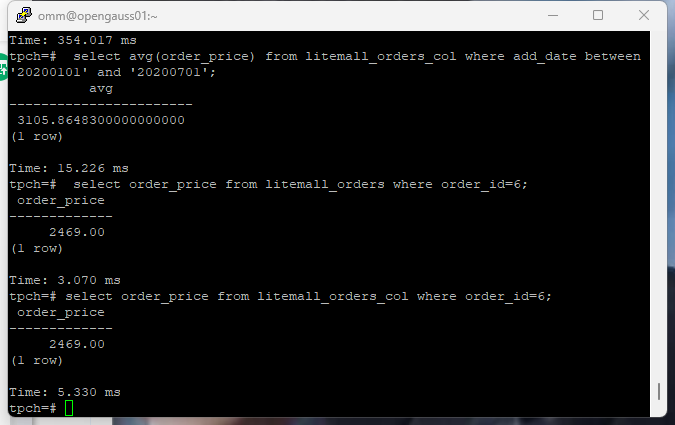


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

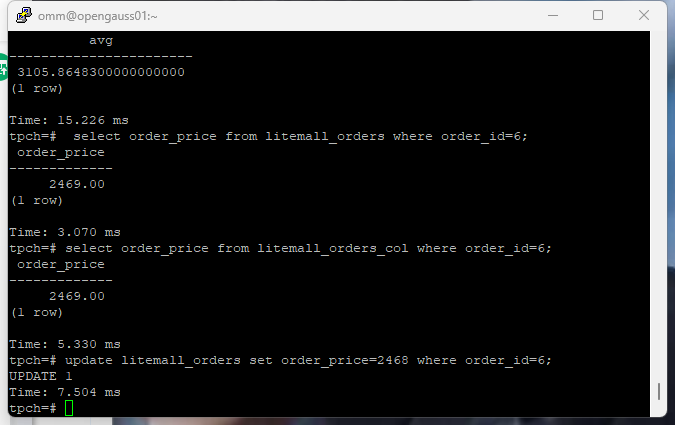


select order\_price from litemall\_orders\_col where order\_id=6;



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;



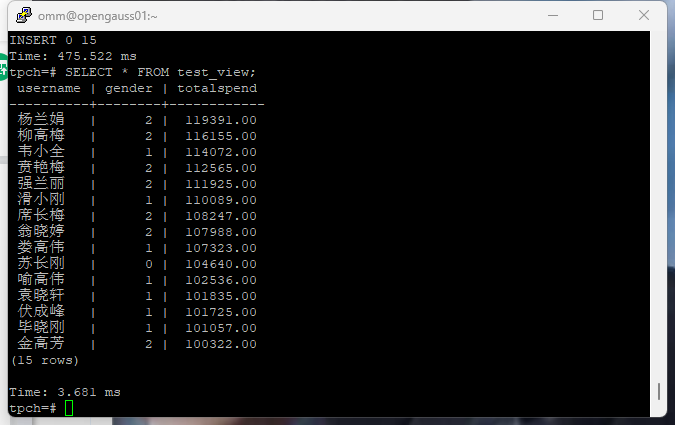
update litemall\_orders\_col set order\_price=2468 where order\_id=6;

))R~NH%[)U@L@SKFA`(%A75

任务三：物化视图的使用

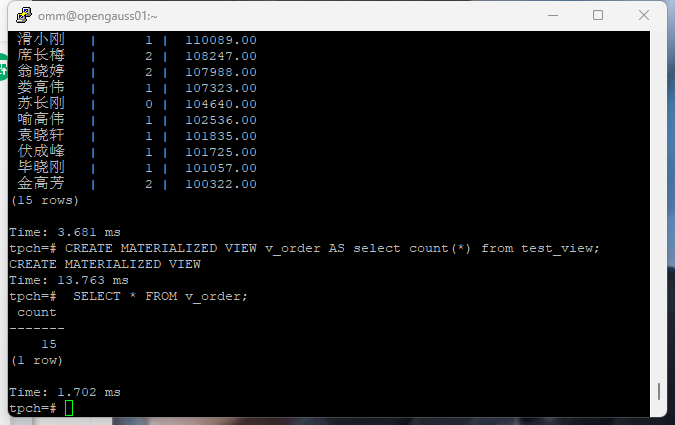
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



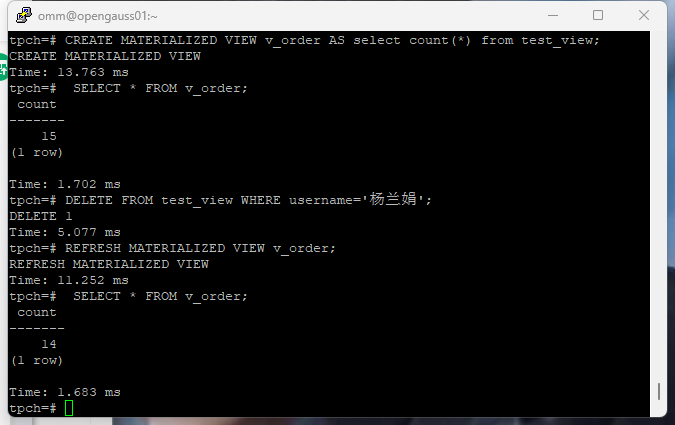
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



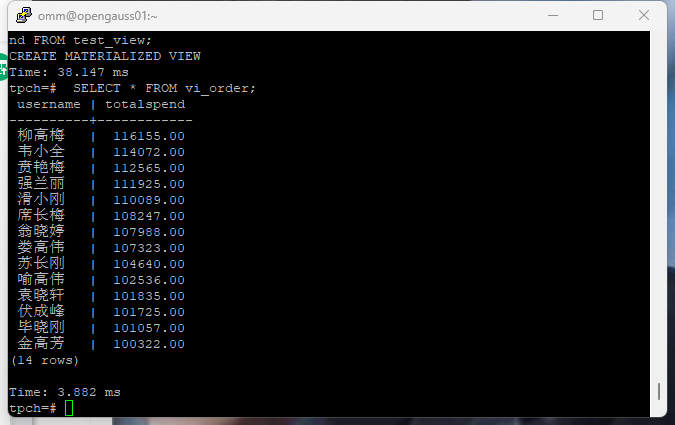
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



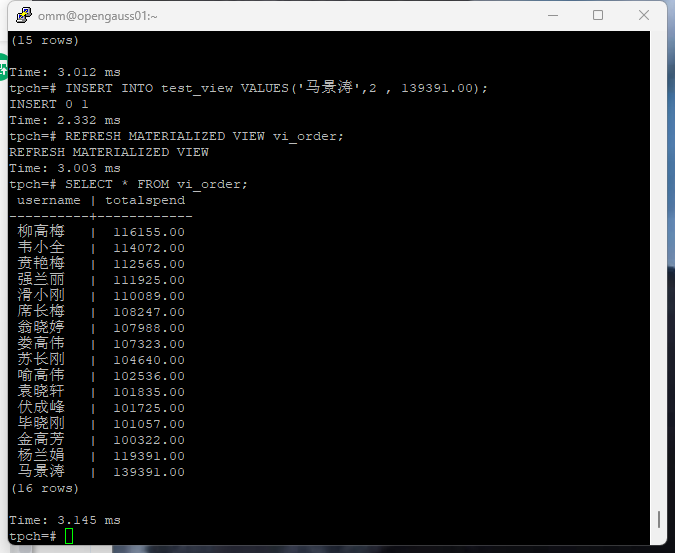
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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：行存表和列存表在执行相同的SQL语句时，执行时间不同的原因是它们的数据存储方式不同。行存表将每一行数据存储在一起，而列存表将每一列数据存储在一起。因此，在执行不同类型的SQL语句时，它们的效率也会有所不同。

在执行需要读取大量行数据的SQL语句时，行存表通常更高效。这是因为行存表可以更快地读取整行数据，而列存表需要读取多个列才能获取完整的行数据。例如，需要查询某个客户的所有订单时，行存表可以更快地返回结果。

在执行需要聚合或分组的SQL语句时，列存表通常更高效。这是因为列存表可以更快地扫描单个列，并且可以更好地利用向量化处理和压缩技术。例如，需要计算某个产品的销售总额时，列存表可以更快地执行聚合操作。

总之，行存表和列存表各有优劣，适用于不同类型的SQL操作。在选择使用哪种表类型时，需要根据具体的业务需求和数据特征进行评估和选择。

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：全量物化视图和增量物化视图是两种不同的物化视图类型，它们的主要差别在于刷新方式和刷新效率。

全量物化视图是一种完全重新计算的物化视图，它在每次刷新时都会重新计算整个视图的数据。这意味着全量物化视图的刷新时间可能会很长，特别是当数据量很大时。但是，全量物化视图的优点是它的数据是最新的，因为它总是重新计算整个视图。

增量物化视图是一种只计算更改数据的物化视图，它只计算源表中发生更改的数据，并将这些更改应用于物化视图中。这意味着增量物化视图的刷新时间通常比全量物化视图短，因为它只需要计算更改的数据。但是，增量物化视图的缺点是它可能不是最新的，因为它只计算更改的数据。

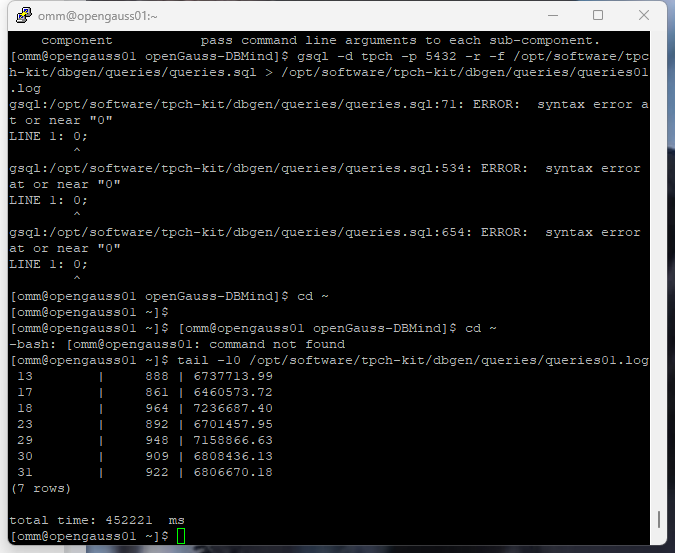
总之，全量物化视图和增量物化视图各有优缺点，适用于不同的场景。如果你需要最新的数据并且可以承受较长的刷新时间，那么全量物化视图可能更适合。如果你需要更快的刷新时间并且可以接受不是最新的数据，那么增量物化视图可能更适合。

# 关卡三、openGauss的AI4DB特性应用

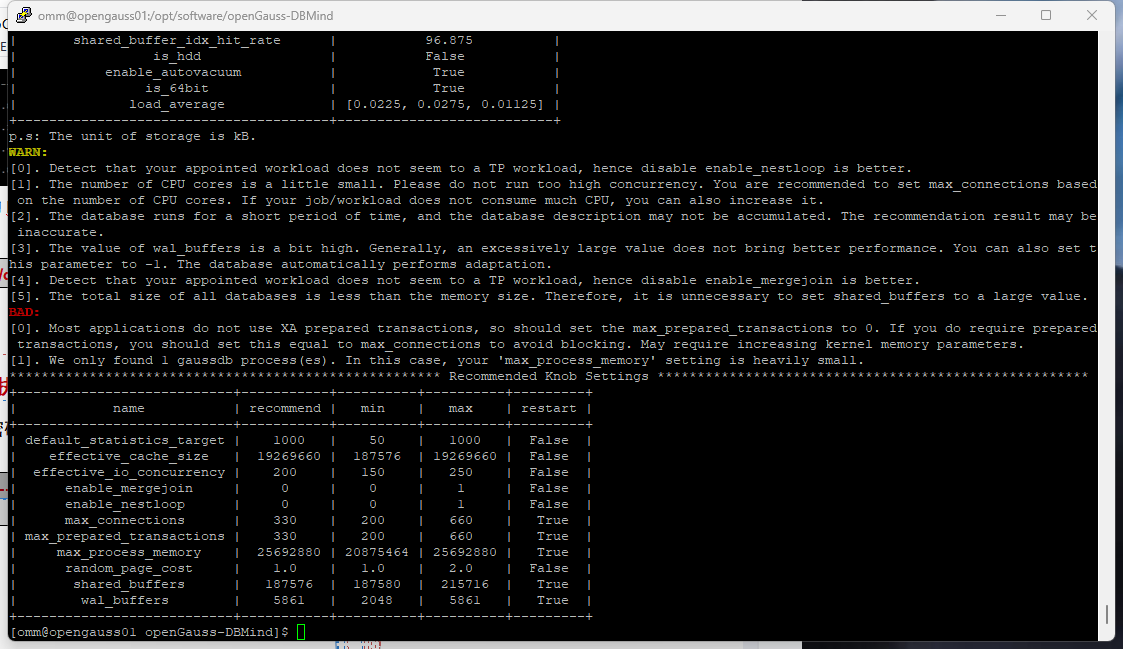
任务一：使用X-Tuner进行参数优化

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

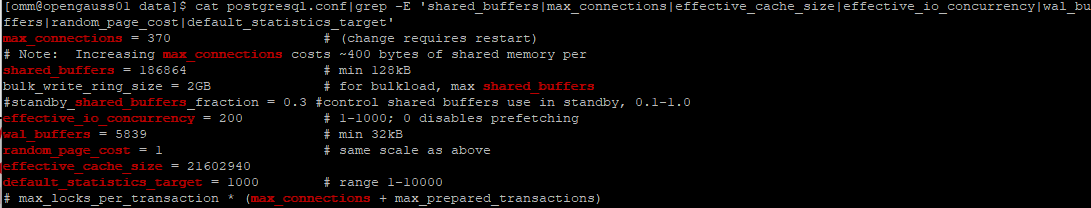
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

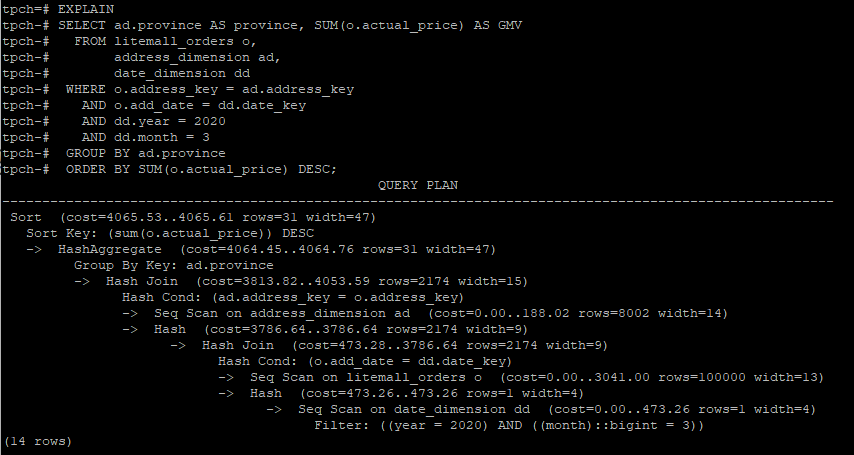
AND o.add\_date = dd.date\_key

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

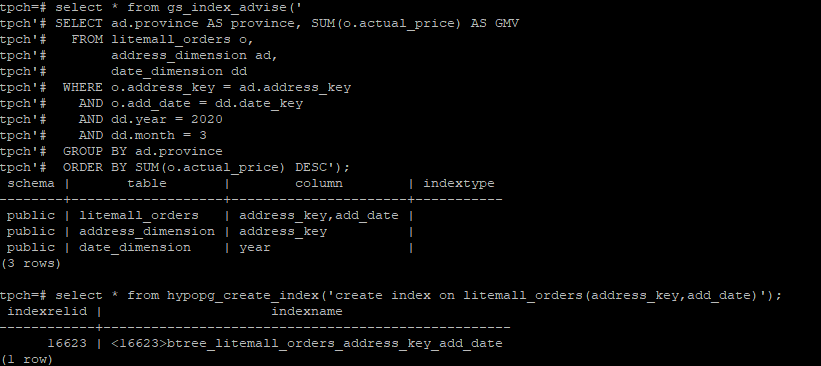
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

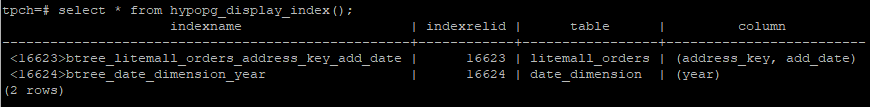
GROUP BY ad.province

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。

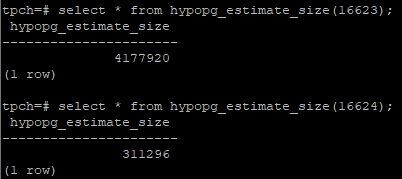
select \* from hypopg\_display\_index();



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(16623);

select \* from hypopg\_estimate\_size(16624);



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

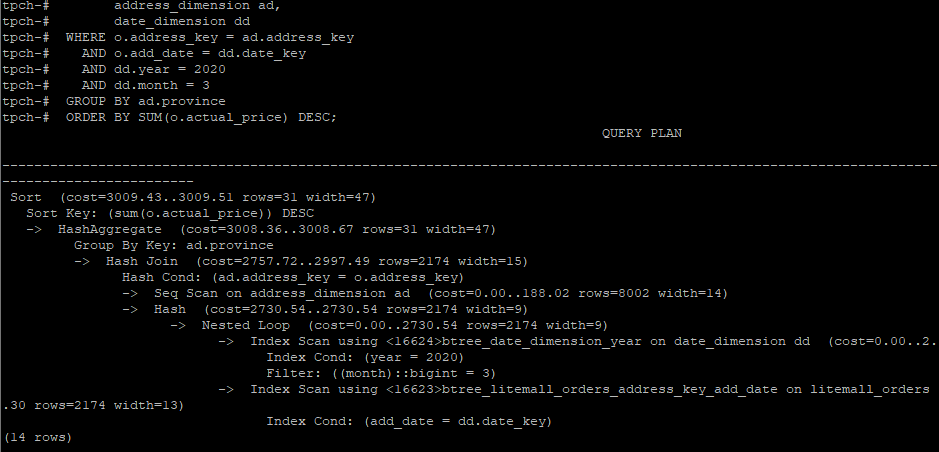
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

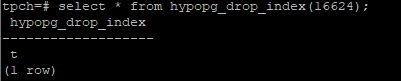
GROUP BY ad.province

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



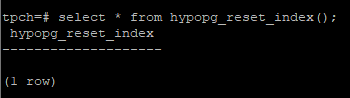
6. 删除某一个索引虚拟列，将执行结果截图。

select \* from hypopg\_drop\_index(16624);



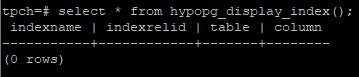
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

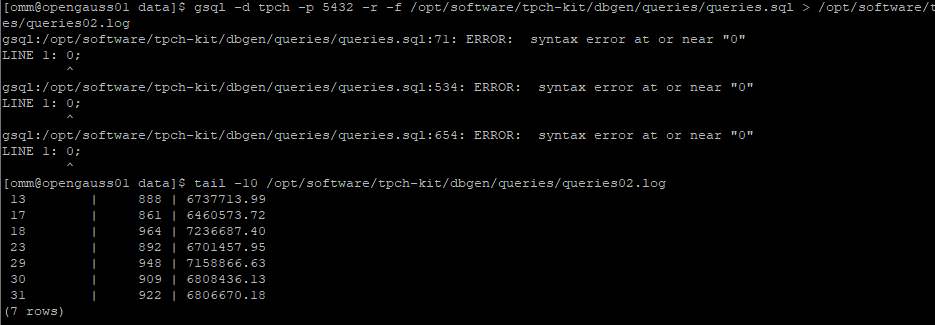
select \* from hypopg\_display\_index();



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：

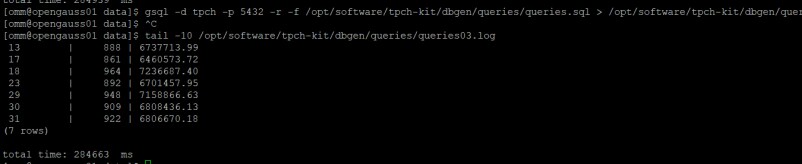
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log



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：X-Tuner是一种自动化参数优化工具，它可以根据给定的参数范围和目标函数，自动搜索最优的参数组合。

在X-Tuner进行参数优化时，它会在指定的参数范围内搜索最优的参数组合，并根据目标函数的值来评估每个参数组合的性能。X-Tuner通常会优化那些对模型性能影响较大的参数，例如学习率、正则化系数、批量大小等。这些参数对模型的性能有很大的影响，因此需要进行优化。

总之，X-Tuner可以帮助我们自动搜索最优的参数组合，从而提高模型的性能。在使用X-Tuner进行参数优化时，需要指定待优化的参数范围、目标函数和优化算法，并优化那些对模型性能影响较大的参数。

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？

索引是一种用于加速数据库查询的数据结构，它可以提高查询的速度和效率。使用索引可以使数据库在执行查询时更快地定位到需要的数据，从而减少查询时间和资源消耗。具体来说，索引可以带来以下好处：

提高查询速度：索引可以帮助数据库快速定位到需要的数据，从而提高查询速度。

减少资源消耗：使用索引可以减少数据库的资源消耗，例如CPU、内存和磁盘IO等。

改善并发性能：索引可以减少锁的竞争，从而提高并发性能。

除了使用索引和参数外，还有以下几个方面可以对数据库进行优化：

数据库设计：良好的数据库设计可以提高数据库的性能和可维护性。例如，使用适当的数据类型、规范化数据模型、避免冗余数据等。

查询优化：优化查询可以提高数据库的性能。例如，使用合适的查询语句、避免使用子查询、避免使用全表扫描等。

硬件优化：优化硬件可以提高数据库的性能。例如，使用高速磁盘、增加内存、使用RAID等。

数据库参数优化：优化数据库参数可以提高数据库的性能。例如，调整缓存大小、调整日志记录级别、调整并发连接数等。

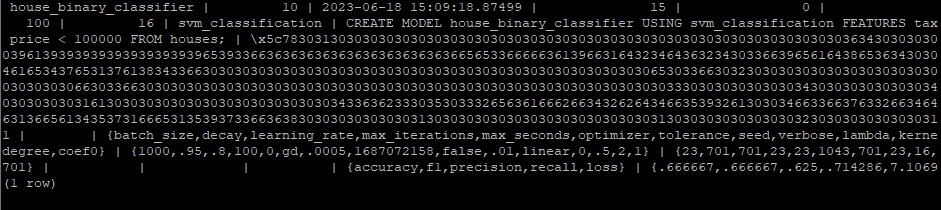
数据库分区：将数据库分成多个分区可以提高查询性能和管理效率。例如，按时间、地理位置、业务类型等进行分区。

总之，对于数据库的优化，需要综合考虑多个方面，包括数据库设计、查询优化、硬件优化、数据库参数优化和数据库分区等。

# 关卡四、openGauss的DB4AI特性应用

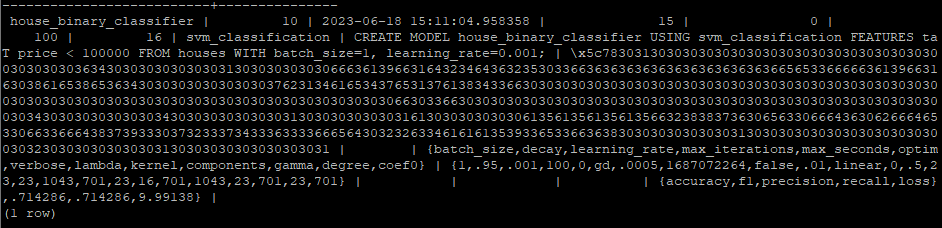
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



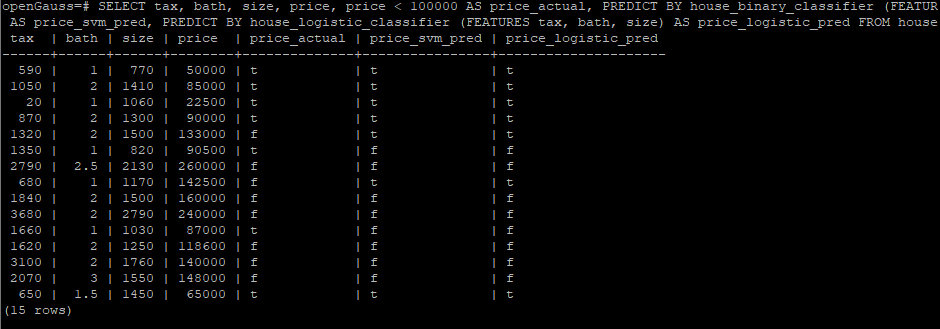
任务二：观察新模型的信息，将执行结果截图。

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？

答：分类模型和回归模型是两种不同的机器学习模型，它们的主要区别在于它们所预测的变量类型不同。

分类模型是一种用于预测离散变量的模型，它将输入数据映射到一个有限的、离散的输出集合中。例如，分类模型可以用于预测一个人是否患有某种疾病、一封电子邮件是否是垃圾邮件等。分类模型通常使用逻辑回归、决策树、支持向量机等算法。

回归模型是一种用于预测连续变量的模型，它将输入数据映射到一个连续的输出空间中。例如，回归模型可以用于预测房价、股票价格等连续变量。回归模型通常使用线性回归、多项式回归、岭回归等算法。

总之，分类模型和回归模型是两种不同的机器学习模型，它们所预测的变量类型不同。分类模型用于预测离散变量，回归模型用于预测连续变量。

实践思考题2：什么是SVM算法？

答：SVM（Support Vector Machine，支持向量机）是一种常用的机器学习算法，它可以用于分类和回归问题。SVM算法的基本思想是将数据映射到高维空间中，然后在该空间中找到一个最优的超平面，将不同类别的数据分开。SVM算法的优点是可以处理高维数据、具有较好的泛化能力和鲁棒性。

SVM算法的核心是寻找最优的超平面，使得不同类别的数据点之间的间隔最大化。在二分类问题中，SVM算法的目标是找到一个超平面，使得正样本和负样本之间的间隔最大化。如果数据不是线性可分的，SVM算法可以使用核函数将数据映射到高维空间中，从而使得数据线性可分。

SVM算法的优化问题可以转化为一个凸优化问题，可以使用现有的优化算法进行求解。常用的SVM算法包括线性SVM、非线性SVM、多分类SVM等。

总之，SVM算法是一种常用的机器学习算法，它可以用于分类和回归问题。SVM算法的核心是寻找最优的超平面，使得不同类别的数据点之间的间隔最大化。SVM算法具有较好的泛化能力和鲁棒性，可以处理高维数据。

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：分类问题的评价指标有很多，以下是常用的几个评价指标及其含义：

准确率（Accuracy）：准确率是指分类器正确分类的样本数占总样本数的比例。准确率越高，分类器的性能越好。但是，当样本不平衡时，准确率可能会失去意义。

精确率（Precision）：精确率是指分类器预测为正样本的样本中，真正为正样本的比例。精确率越高，分类器预测为正样本的准确性越高。

召回率（Recall）：召回率是指真正为正样本的样本中，被分类器预测为正样本的比例。召回率越高，分类器能够正确识别正样本的能力越强。

F1分数（F1 Score）：F1分数是精确率和召回率的调和平均数，它综合了精确率和召回率的性能。F1分数越高，分类器的性能越好。

ROC曲线（Receiver Operating Characteristic Curve）：ROC曲线是一种用于评估二分类器性能的曲线，它以真正例率（True Positive Rate）为纵轴，以假正例率（False Positive Rate）为横轴。ROC曲线越靠近左上角，分类器的性能越好。

AUC（Area Under Curve）：AUC是ROC曲线下的面积，它可以用来评估分类器的性能。AUC越大，分类器的性能越好。

总之，分类问题的评价指标有很多，需要根据具体情况选择合适的指标。准确率、精确率、召回率和F1分数是常用的评价指标，用于评估分类器的性能。ROC曲线和AUC是用于评估二分类器性能的常用指标。

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：回归问题的评价指标有很多，以下是常用的几个评价指标及其含义：

均方误差（Mean Squared Error，MSE）：均方误差是指预测值与真实值之差的平方的平均值。MSE越小，模型的性能越好。

均方根误差（Root Mean Squared Error，RMSE）：均方根误差是均方误差的平方根，它可以将误差转化为与目标变量相同的单位。RMSE越小，模型的性能越好。

平均绝对误差（Mean Absolute Error，MAE）：平均绝对误差是指预测值与真实值之差的绝对值的平均值。MAE越小，模型的性能越好。

决定系数（Coefficient of Determination，R2）：决定系数是指模型解释目标变量方差的比例。R2越接近1，模型的性能越好。

相关系数（Correlation Coefficient，r）：相关系数是指预测值与真实值之间的线性关系强度。相关系数越接近1，模型的性能越好。

总之，回归问题的评价指标有很多，需要根据具体情况选择合适的指标。均方误差、均方根误差和平均绝对误差是常用的评价指标，用于评估模型的预测误差。决定系数和相关系数是用于评估模型的预测能力的常用指标。