openGauss AI特性创新实践课



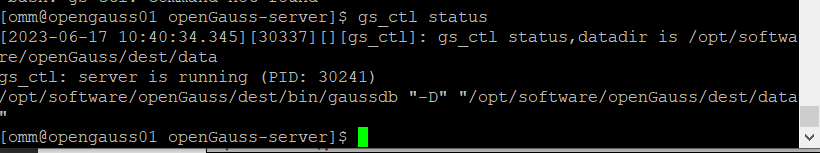
华为技术有限公司

# 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：

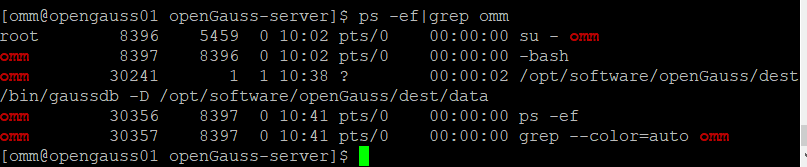
任务一：数据库状态验证

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？

以下原因可以解释为什么需要通过源码编译来安装数据库：

1. 自定义编译选项：通过源码编译数据库，可以自定义编译选项以满足特定需求。可以根据您的系统配置、硬件需求和性能要求进行调整。这允许您优化数据库的功能、性能和兼容性，并根据特定的环境进行调整。

2. 平台和操作系统兼容性：源码编译可以确保数据库与操作系统和平台兼容。不同的操作系统和版本可能有不同的要求和依赖项，通过源码编译可以根据环境进行自定义适配，确保数据库在系统上正确运行。

3. 版本控制和更新：通过源码编译和安装数据库，可以更好地控制数据库的版本和更新。可以选择安装最新版本或特定版本，并根据需要进行升级和回滚。这使用户能够更灵活地管理数据库的功能和修复程序，并在需要时进行安全性和性能更新。

4. 安全性和稳定性：源码编译可以提供更高的安全性和稳定性。源码通常经过广泛的测试和审查，可以修复已知的漏洞和问题。此外，可以根据自己的需求选择启用或禁用特定的功能和插件，以减少潜在的安全风险。

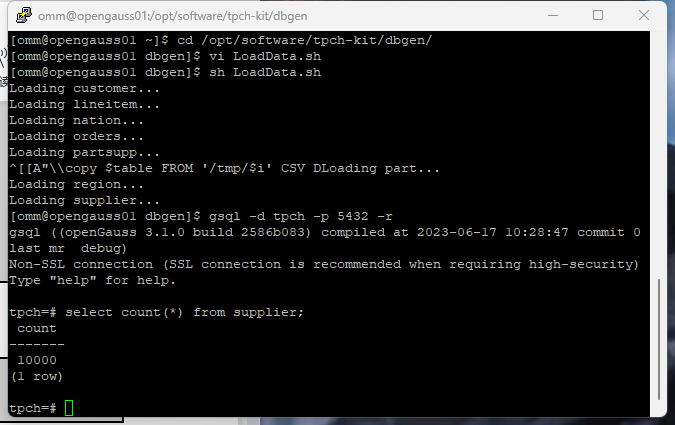
需要注意的是，通过源码编译安装数据库需要一定的技术知识和经验，以及适当的系统和开发环境。对于非专业用户来说，可能更倾向于使用预编译的二进制发行版，这些发行版已经经过编译和打包，并且易于安装和配置。但对于需要更高度定制和控制的用户和环境，源码编译是一种更为灵活和可行的选择。

# 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：

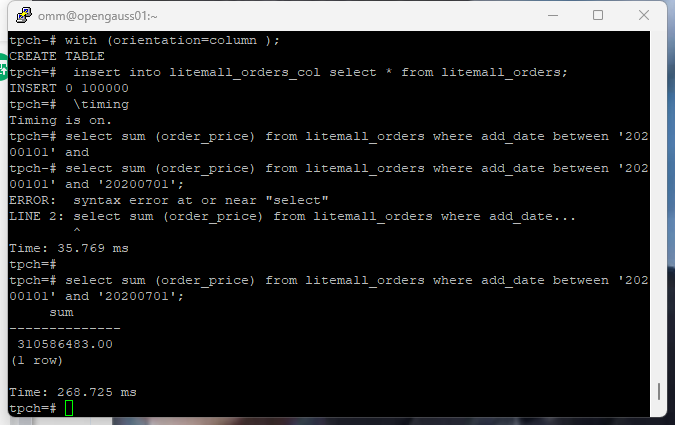
select count(\*) from supplier;;



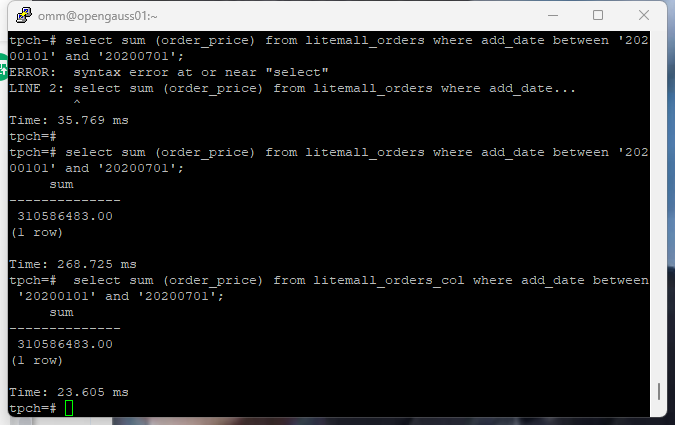
任务二：行存表与列存表执行效率对比

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';

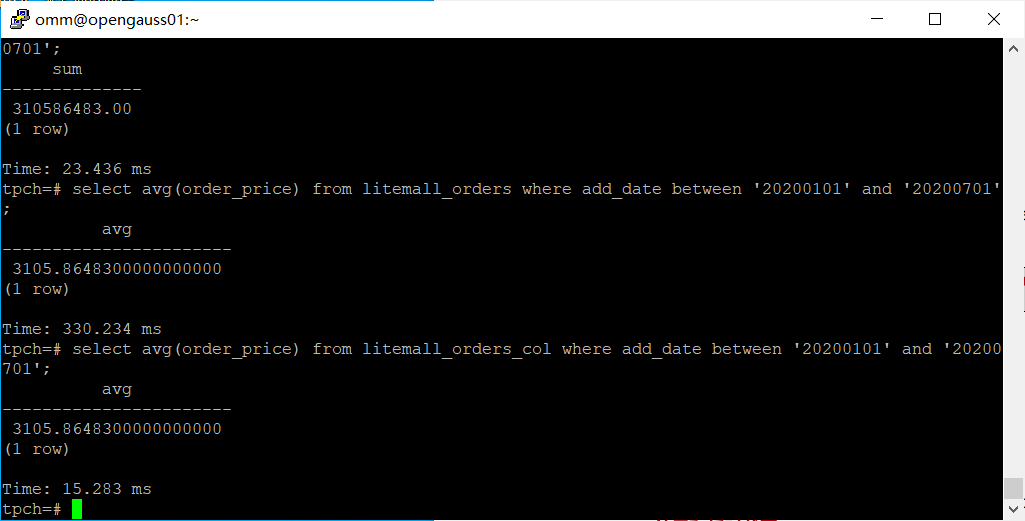


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

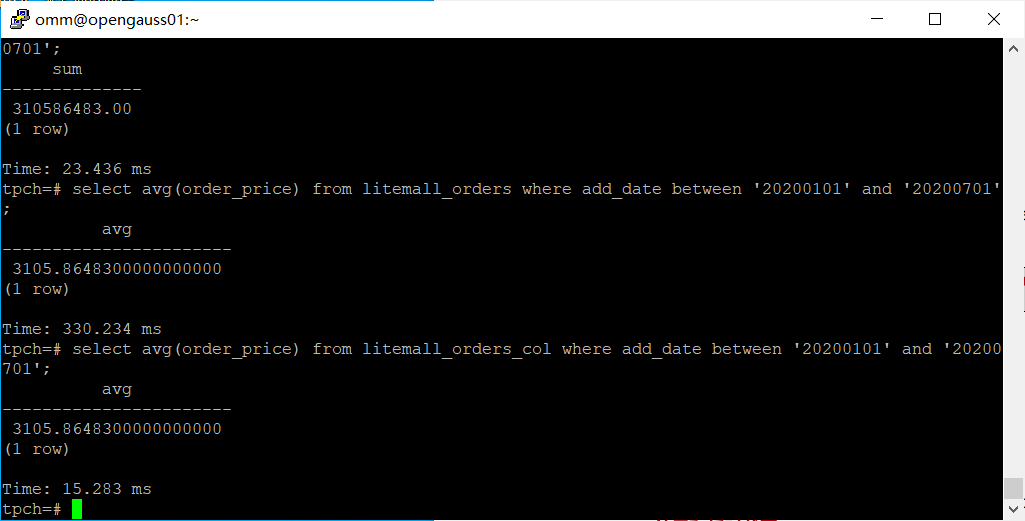


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

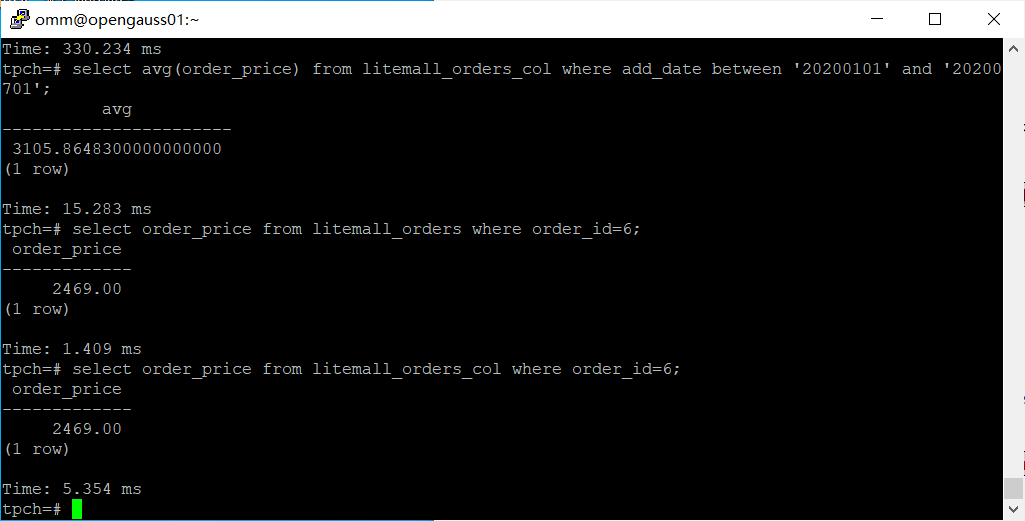


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

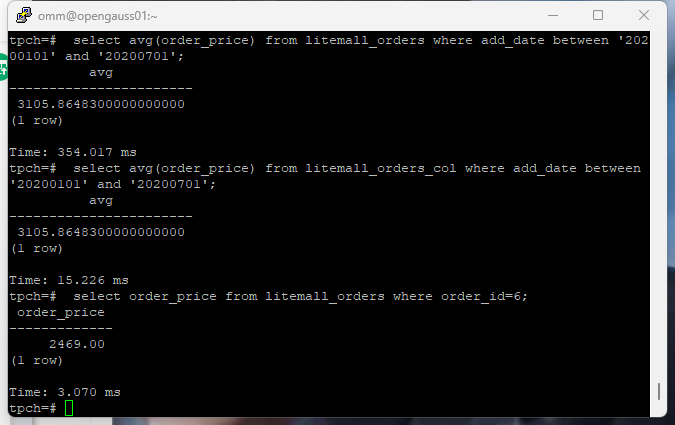


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

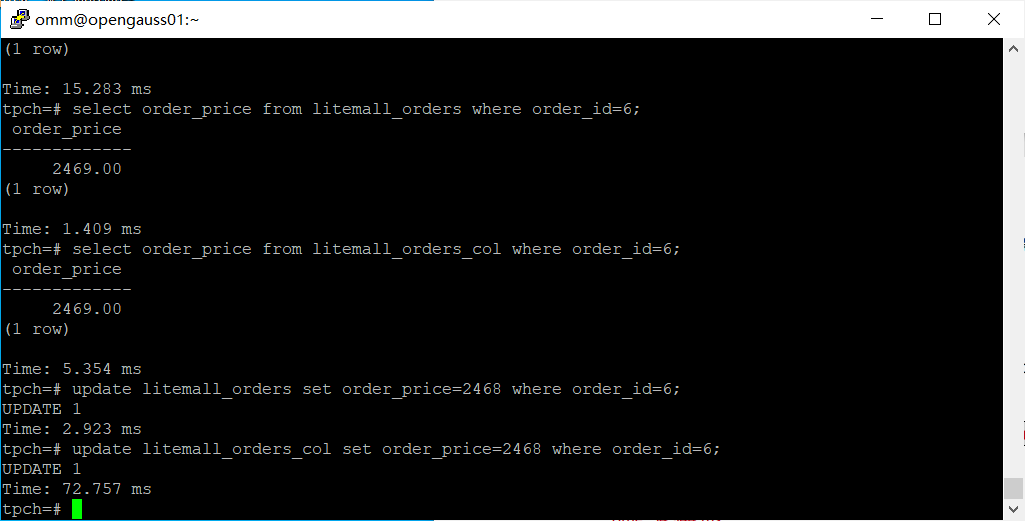


select order\_price from litemall\_orders\_col where order\_id=6;



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;



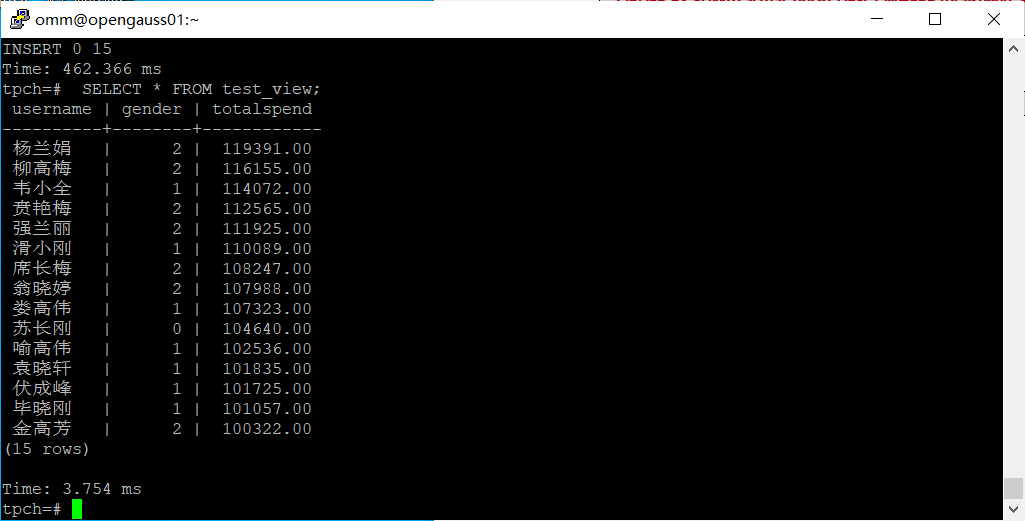
update litemall\_orders\_col set order\_price=2468 where order\_id=6;

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

任务三：物化视图的使用

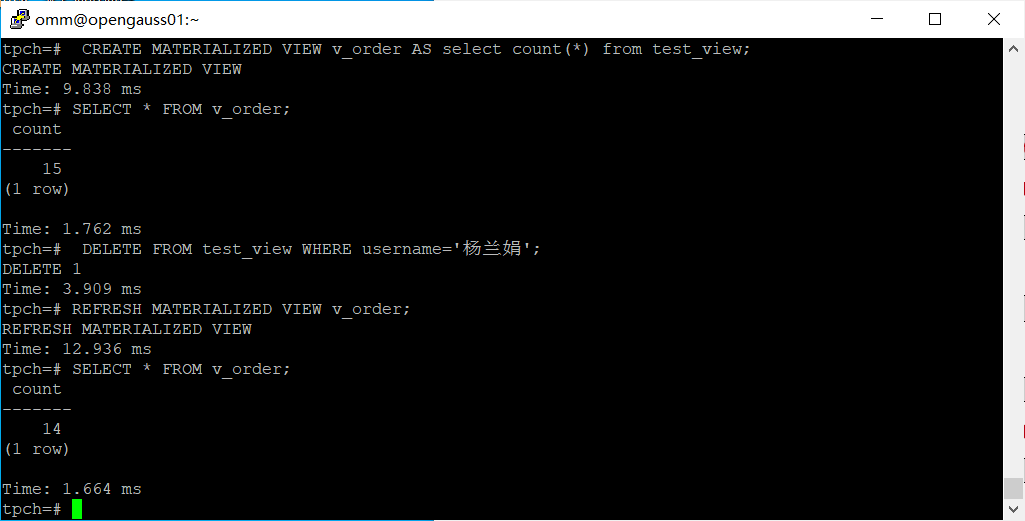
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



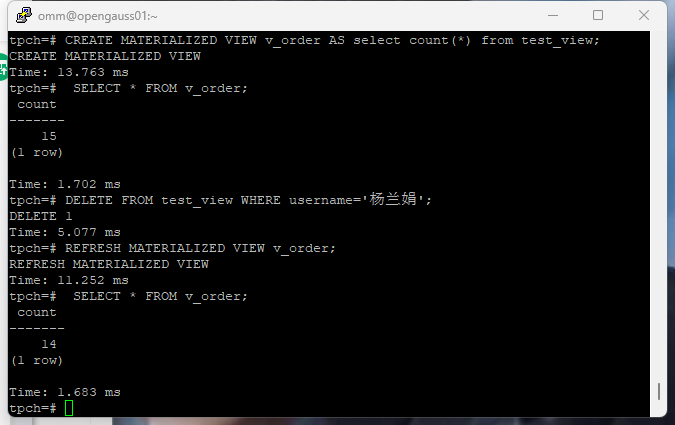
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



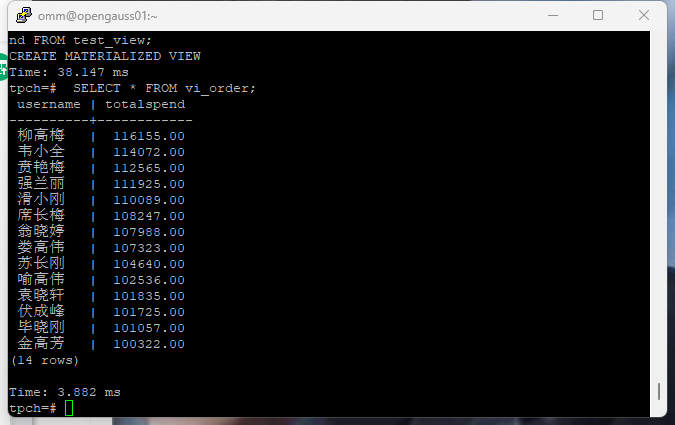
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



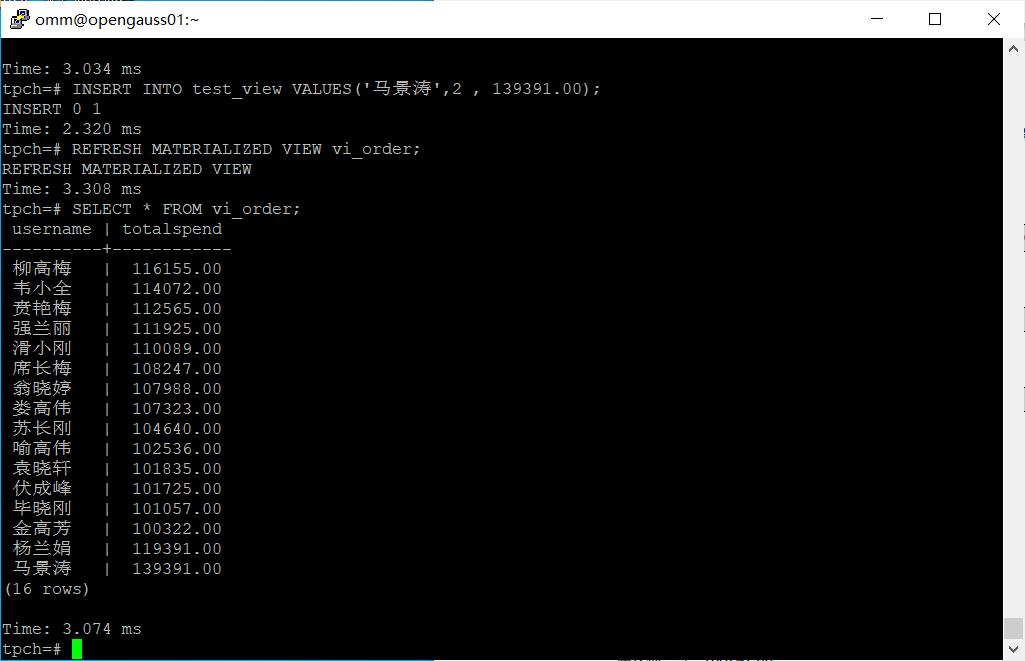
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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？

行存表和列存表是在数据库中使用的两种不同的存储方式。

行存表（Row-based table）是将数据按行存储的方式。每一行数据都包含了记录中的所有字段信息。这种存储方式适合于需要读取整行数据的场景，例如查询某个特定的记录，或者需要获取完整数据行的所有字段值的查询。

列存表（Columnar table）是将数据按列存储的方式。每一列数据都是连续存储的，所有行的同一列数据存储在一起。这种存储方式适合于对某个特定字段的聚合操作，例如对某个字段进行求和、计数等统计操作。

执行相同的SQL语句时，行存表和列存表执行时间不同的原因主要有两个方面：

1. 数据访问模式：行存表适合读取整行数据，因此在执行需要读取整行数据的查询时效率较高；而列存表适合进行聚合操作，当SQL语句需要对某个特定字段进行统计或聚合时，列存表的效率较高。

2. 数据存储方式：行存表在存储时将一整行数据存储在一起，因此在查询时可能需要读取多个字段的数据，增加了I/O操作和数据传输的开销；而列存表将同一列的数据存储在一起，可以只读取需要的列，减少了数据传输的开销。

因此，在执行以下类型的SQL语句时，行存表的效率更高：

- 需要读取整行数据的查询，例如根据主键查询某个记录。

- 需要频繁更新或插入数据的操作，因为行存表的数据存储方式更适合这类操作。

而在执行以下类型的SQL语句时，列存表的效率更高：

- 需要对某个特定字段进行聚合、统计操作的查询，例如求和、计数等。

- 需要查询大量数据但只需要部分字段的查询，因为列存表可以只读取需要的列，减少了数据传输的开销。

需要注意的是，选择行存表还是列存表还应考虑具体的数据库系统、数据量、查询频率等因素，没有绝对的优劣之分，需根据实际情况进行选择。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

全量物化视图（Materialized View，简称MV）和增量物化视图是两种不同的物化视图类型，它们在刷新方式和使用场景上存在差异。

1. 刷新方式：

- 全量物化视图：全量物化视图在刷新时重新计算和获取所有数据，即重新执行查询并更新整个物化视图的内容。这通常需要较长的时间，特别是当基础数据量大且更新频繁时。

- 增量物化视图：增量物化视图只刷新被影响到的数据部分，而不是重新计算和获取全部数据。它跟踪源数据变化，并根据变化的情况更新物化视图的内容。这种刷新方式更加高效，因为它只处理变更的部分数据。

2. 使用场景：

- 全量物化视图：全量物化视图适用于静态或相对稳定的数据集。当基础数据更新频率较低或数据量相对较小，并且查询需要获取整个数据集时，全量物化视图是一个很好的选择。例如，用于生成报表的物化视图。

- 增量物化视图：增量物化视图适用于数据更新频繁的情况。当基础数据的变更集较小且查询仅需要最新的部分数据时，增量物化视图可以提供更好的性能。例如，用于缓存频繁查询的结果集，以避免重复计算。

需要注意的是，增量物化视图通常需要更复杂的维护机制来跟踪和处理源数据的变化，以确保物化视图的准确性和一致性。而全量物化视图的维护相对简单，因为它只需要重新计算和刷新整个数据集。

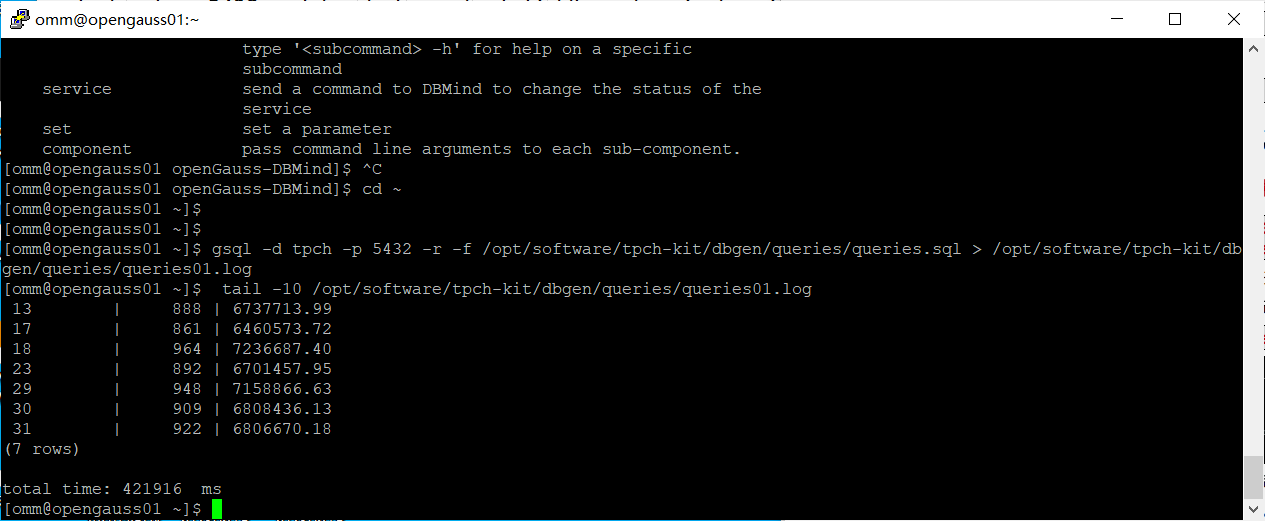
选择使用全量物化视图还是增量物化视图应根据具体的业务需求和数据特点进行评估。增量物化视图在处理大规模数据和高频更新时可能更具优势，但也需要考虑增加的维护复杂性。全量物化视图则适用于数据变更较少或查询需要整个数据集的场景。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

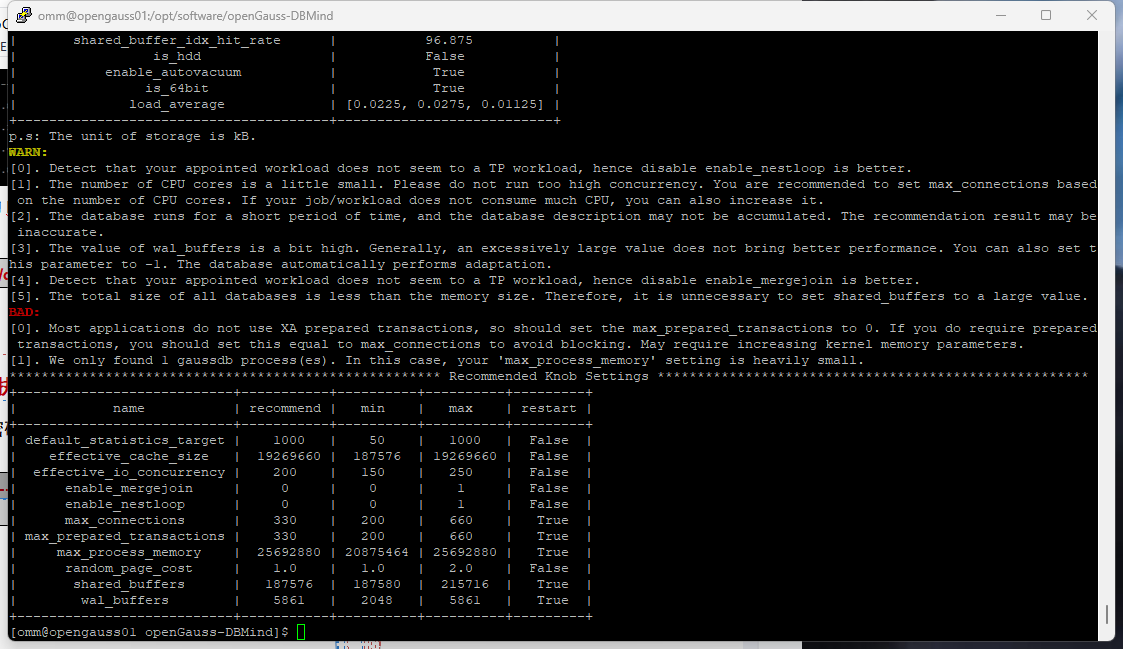
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

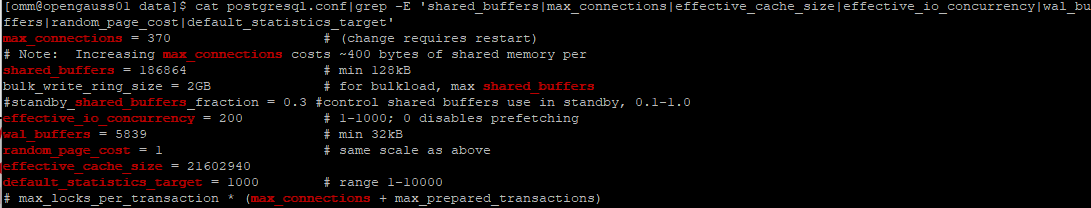
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

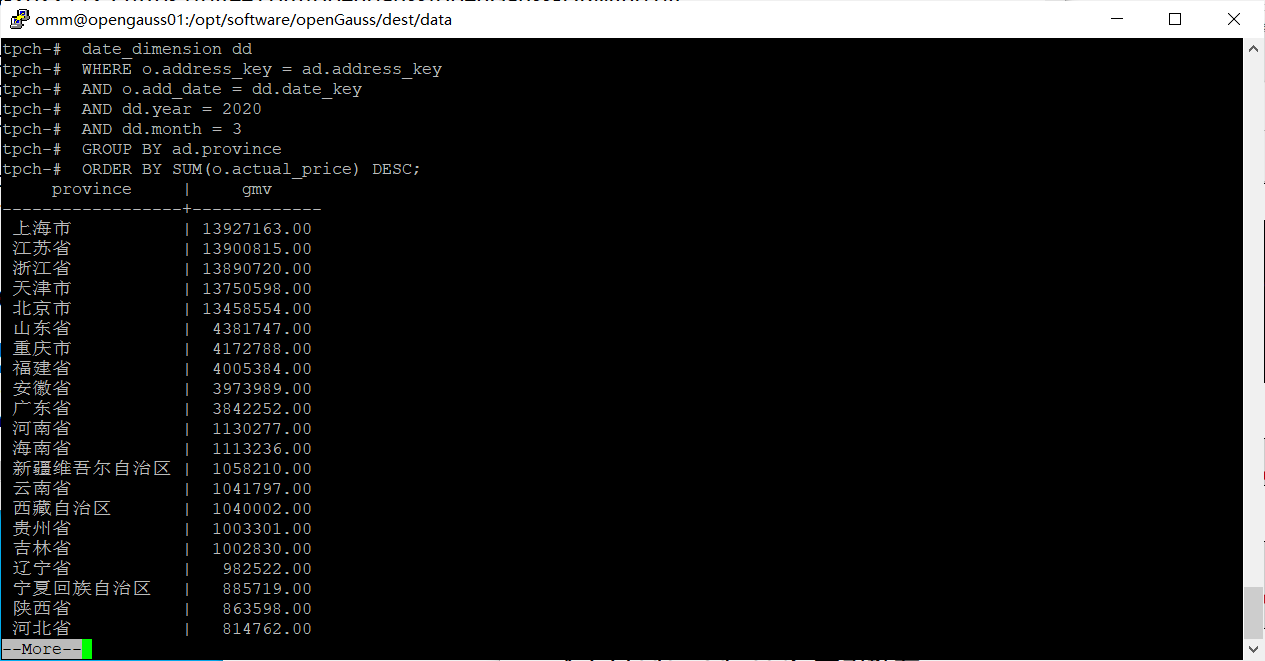
AND o.add\_date = dd.date\_key

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

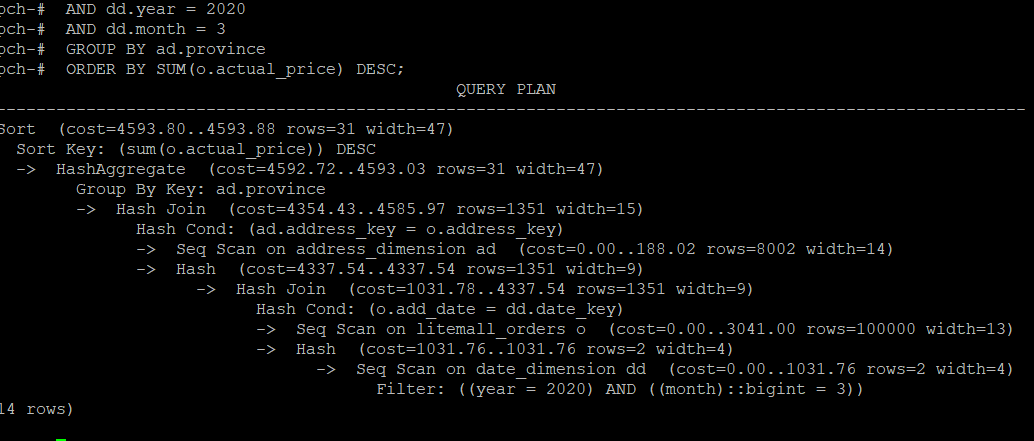
AND o.add\_date = dd.date\_key

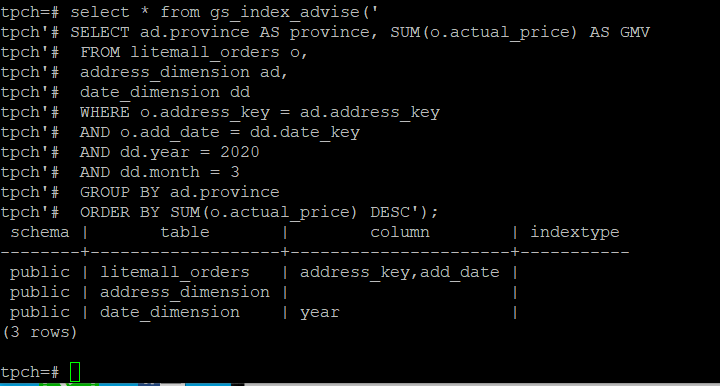
AND dd.year = 2020

AND dd.month = 3

GROUP BY ad.province

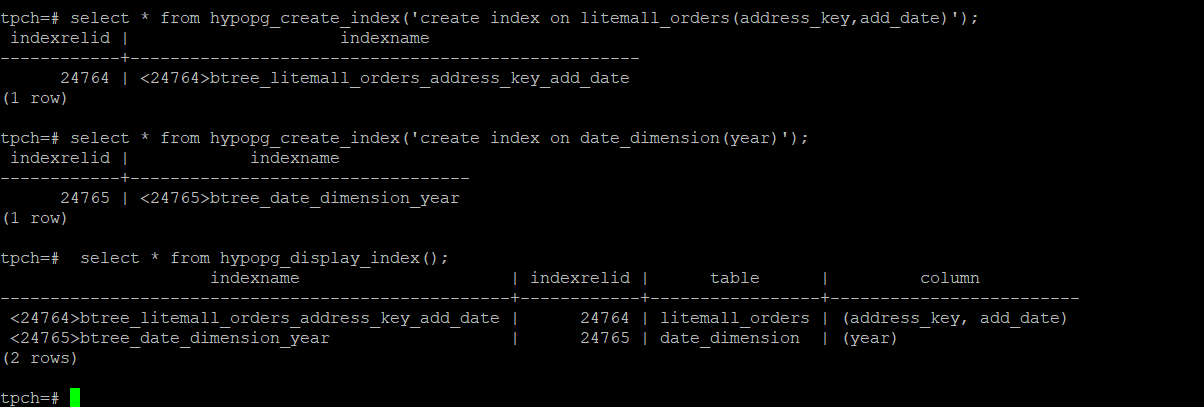
ORDER BY SUM(o.actual\_price) DESC');





3. 查看创建的虚拟索引列，将执行结果截图。

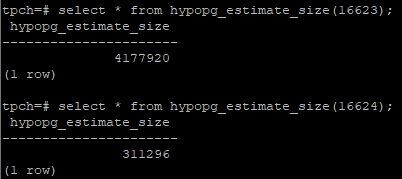
select \* from hypopg\_display\_index();



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

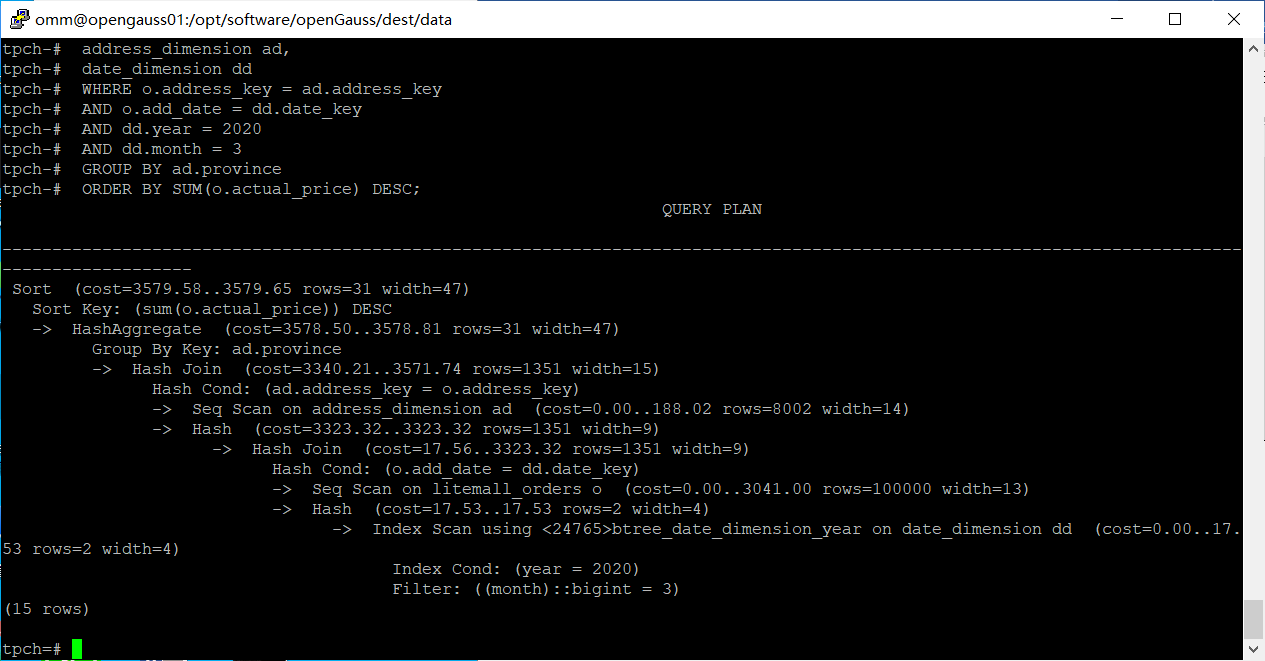
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

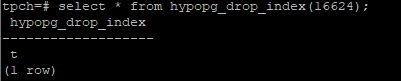
GROUP BY ad.province

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



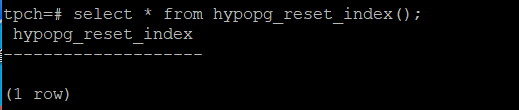
6. 删除某一个索引虚拟列，将执行结果截图。

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



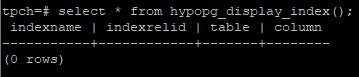
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

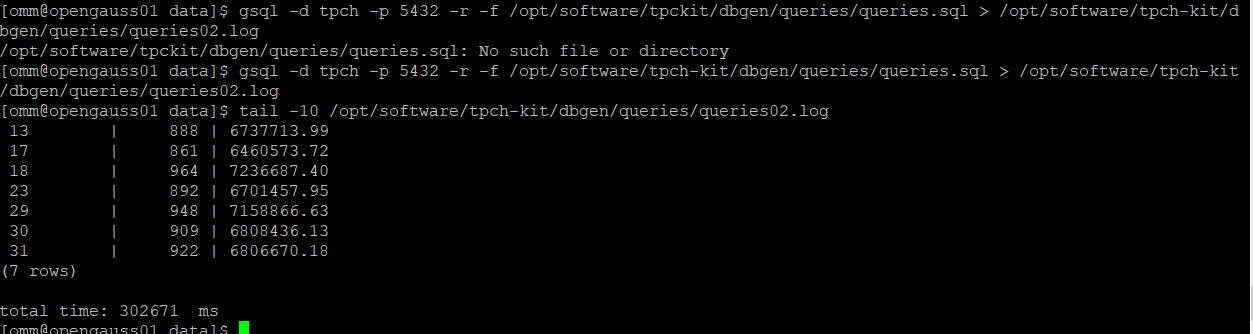
select \* from hypopg\_display\_index();



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：

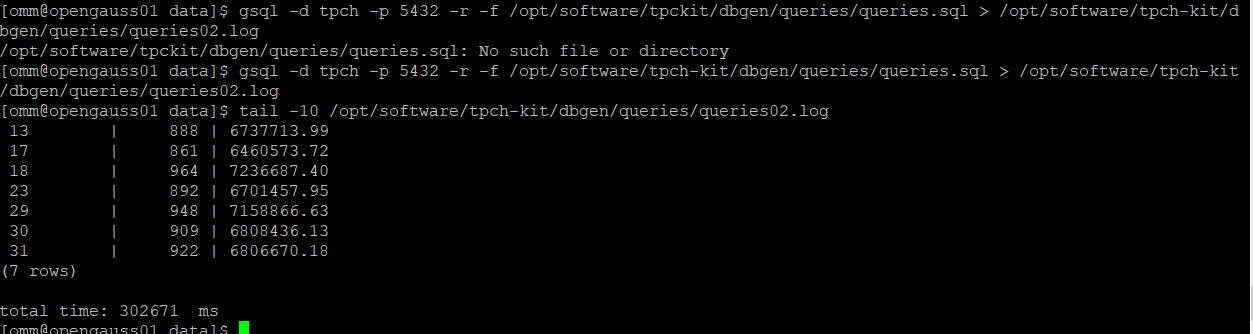
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log



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：

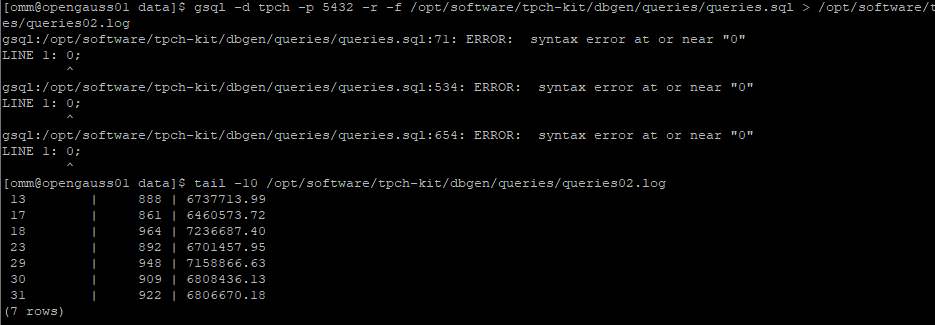
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries03.log



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？

缓冲池和缓存参数：这些参数涉及数据库的内存管理和数据缓存。例如，可以调整缓冲池大小、页大小、缓存策略和算法，以优化磁盘 I/O 操作和数据访问速度。

查询优化器参数：查询优化器决定了数据库如何选择和执行查询计划。通过调整查询优化器的参数，可以改变查询计划的选择和执行方式，从而优化查询性能。

并发控制参数：这些参数控制数据库的并发访问和锁定机制。通过调整并发控制参数，可以平衡并发访问的性能和数据一致性，避免死锁和资源争用。

日志和恢复参数：这些参数控制数据库事务日志的写入和恢复机制。通过调整日志和恢复参数，可以平衡事务的持久性和性能，以及故障恢复的速度和可靠性。

网络和连接参数：这些参数涉及数据库服务器的网络通信和连接管理。通过调整网络和连接参数，可以优化客户端与服务器之间的通信性能和连接资源的利用。

这些参数进行优化的原因是为了提高数据库的性能、可伸缩性、稳定性和适应性。通过优化这些参数，可以根据特定的工作负载和系统需求，改善查询响应时间、减少资源消耗、提高并发处理能力和减少潜在的性能瓶颈。

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 查询时有以下好处：

1. 提高查询性能：索引可以加速数据的查找和访问过程，减少数据库需要扫描的数据量。通过索引，数据库可以更快地定位和获取符合查询条件的数据，从而提高查询性能。

2. 减少磁盘 I/O 操作：索引可以帮助数据库避免全表扫描，减少磁盘 I/O 操作的次数。通过减少磁盘读取操作，可以降低查询的响应时间并提高系统的整体吞吐量。

3. 优化排序和聚合操作：索引可以加速排序和聚合操作，使数据库能够更快地执行排序、分组和聚合操作。通过使用索引，可以避免对大量数据进行排序或聚合，提高相关查询的执行效率。

除了使用索引和参数来优化数据库性能，还可以考虑以下方面：

1. 数据库设计和规范化：良好的数据库设计和规范化可以提高数据库的性能和可扩展性。合理划分数据表、优化数据结构、避免冗余数据和设计合适的关系模式等，都可以减少查询复杂度和提高数据访问效率。

2. 查询优化和调整：通过分析和优化查询语句，可以改进查询执行计划和索引的使用方式。使用合适的连接方式、选择恰当的操作符、避免不必要的子查询等，可以提高查询性能。

3. 硬件和系统配置：适当的硬件和系统配置对数据库性能至关重要。考虑合适的服务器配置、磁盘和存储优化、内存管理、并发连接数的设置等，可以提高数据库的吞吐量和响应时间。

4. 缓存和缓冲区管理：通过适当设置缓存和缓冲区参数，可以提高数据访问的速度和效率。例如，调整缓冲池大小、合理设置缓存策略、使用适当的缓存技术等，都可以减少磁盘 I/O 操作并加速数据的读取。

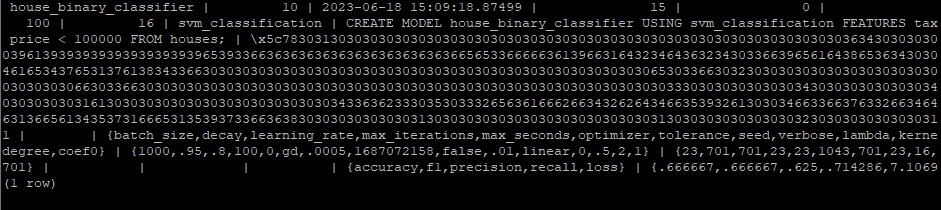
5. 定期维护和优化：定期进行数据库维护和优化是保持数据库性能的重要方面。包括索引重建、统计信息更新、数据库统计和监控、定期备份和恢复等，都可以确保数据库的稳定性和高性能运行。

需要根据具体的数据库系统、应用需求和环境特点进行综合考虑和优化。数据库性能优化是一个复杂的任务，需要综合多个因素进行调整和优化，以获得最佳的性能和效果。

# 关卡四、openGauss的DB4AI特性应用

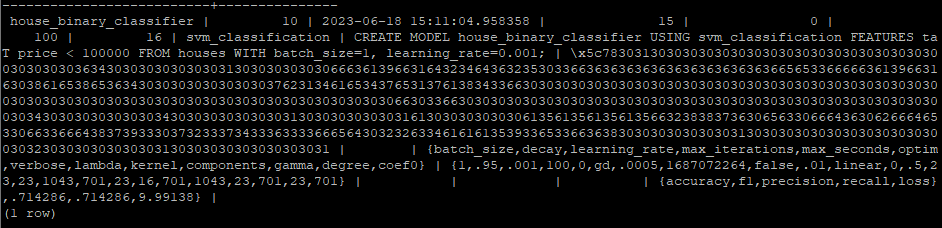
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



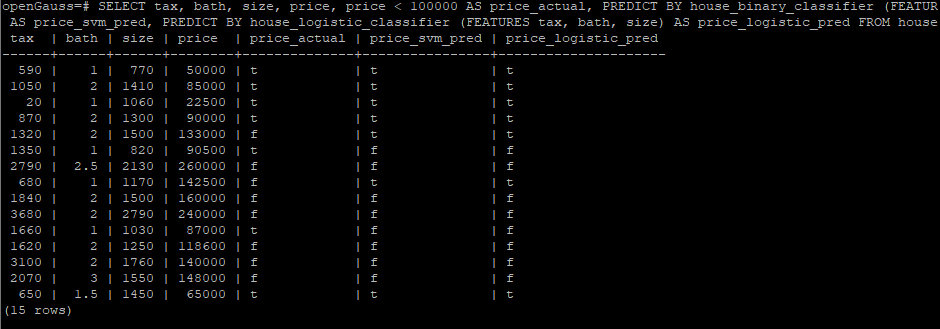
任务二：观察新模型的信息，将执行结果截图。

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？

分类模型和回归模型是机器学习中两种不同类型的预测模型，它们在预测目标和输出结果上存在一些基本的区别。

分类模型：

分类模型用于预测离散的类别或标签。它将输入数据映射到预定义的类别中的一个。分类模型的输出是有限且离散的类别标签。一些常见的分类算法包括逻辑回归、决策树、随机森林、支持向量机和神经网络。分类模型可用于识别图像中的物体、预测电子邮件是否为垃圾邮件、预测疾病的患病与否等。

回归模型：

回归模型用于预测连续的数值输出。它根据输入数据建立一个函数关系，用于预测数值型目标变量。回归模型的输出是连续的数值。一些常见的回归算法包括线性回归、多项式回归、支持向量回归和决策树回归。回归模型可用于预测房价、销售量、气温变化等连续数值的预测问题。

主要区别：

1. 输出类型：分类模型输出离散的类别标签，而回归模型输出连续的数值。

2. 模型选择：选择分类模型还是回归模型取决于预测目标的性质。如果预测的目标是有限且离散的类别，应选择分类模型。如果预测的目标是连续的数值，应选择回归模型。

3. 损失函数：分类模型通常使用交叉熵、对数损失或误分类率等损失函数，用于度量预测结果与真实标签之间的差异。而回归模型通常使用均方误差、平均绝对误差等损失函数。

1. 模型评估：分类模型的性能评估通常使用准确率、精确度、召回率、F1 分数等指标来衡量。回归模型的性能评估则使用均方误差、平均绝对误差、决定系数等指标来衡量。

实践思考题2：什么是SVM算法？

SVM（支持向量机）是一种机器学习算法，用于进行分类和回归任务。它通过在特征空间中构建一个最优的超平面，将不同类别的样本尽可能地分开。SVM 在解决二分类和多分类问题中都有广泛的应用。

SVM 的基本思想是寻找一个能够最大化样本间间隔（margin）的超平面，将不同类别的样本分开。超平面可以理解为一个 n-1 维的线性决策边界，其中 n 是特征空间的维度。在二维空间中，超平面是一条直线，而在高维空间中，超平面是一个超平面。

SVM 的核心概念包括：

1. 支持向量：这些是距离超平面最近的训练样本点，它们对于定义超平面具有重要的作用。支持向量决定了超平面的位置和方向。

2. 间隔（margin）：间隔是指超平面与支持向量之间的距离，用于度量分类器的鲁棒性和泛化能力。SVM 通过最大化间隔来寻找最优的超平面，从而提高分类器的性能。

3. 核函数：SVM 可以使用核函数将低维特征映射到高维特征空间中。通过核函数，SVM 可以处理非线性可分的数据，并在高维空间中找到线性可分的超平面。

SVM 的训练过程可以通过求解一个凸优化问题来实现，目标是最小化损失函数和正则化项。一旦训练完成，SVM 可以用于对新的样本进行分类或回归预测。

SVM 算法具有以下优点：

1. 在高维空间中有效：SVM 可以通过核函数将低维特征映射到高维空间中，以处理非线性可分的数据。

2. 对于小样本数据有效：SVM 的决策边界仅取决于支持向量，因此对于小样本数据具有较好的泛化能力。

3. 可以控制间隔和容错率：SVM 可以通过调整正则化参数来控制间隔的大小，从而平衡分类的准确性和容错率。

4. 在处理高维特征时具有较好的性能：由于 SVM 只涉及支持向量，因此在处理高维特征数据时具有较好的计算效率和内存利用率。

需要注意的是，SVM 对于大规模数据集和高度噪声的数据可能会面临挑战。此外

，SVM 的参数选择和核函数的选择也需要根据具体问题进行调整和优化。

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？

分类问题的评价指标可以帮助我们评估分类器的性能和准确度。以下是一些常见的分类问题评价指标：

1. 准确率（Accuracy）：准确率是最常用的评价指标之一，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整体性能较好，但在类别不平衡的情况下可能会产生误导。

2. 精确度（Precision）：精确度是指分类器预测为正例的样本中实际为正例的比例。精确度高表示分类器将负例误判为正例的能力较低，即避免了"假阳性"错误。

3. 召回率（Recall）：召回率是指分类器正确预测为正例的样本占实际正例样本的比例，也被称为灵敏度或真阳性率。召回率高表示分类器能够较好地捕捉到实际为正例的样本，避免了"假阴性"错误。

4. F1 分数（F1 Score）：F1 分数是精确度和召回率的调和平均值，综合考虑了两者的性能。F1 分数高表示分类器在精确度和召回率之间取得了较好的平衡。

5. 特异度（Specificity）：特异度是指分类器将负例正确预测为负例的能力，即避免了"假阴性"错误。特异度与召回率互为补数。

6. ROC 曲线和AUC：ROC 曲线（Receiver Operating Characteristic Curve）以分类器的真阳性率（召回率）为纵轴，假阳性率（1-特异度）为横轴绘制。AUC（Area Under the Curve）是ROC 曲线下方的面积，用于评估分类器的整体性能。AUC 值越接近1，表示分类器性能越好。

以上评价指标的选择取决于具体的问题和应用场景。在类别不平衡、成本敏感或特定需求下，可能需要重点关注某些评价指标。同时，还可以根据具体问题的需求自定义和使用其他的评价指标。

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？

在回归问题中，我们使用不同的评价指标来衡量预测结果与真实值之间的差异。以下是一些常见的回归问题评价指标：

1. 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。它计算预测值与真实值之间差异的平方的均值，用于衡量预测结果的平均误差大小。MSE 越小表示模型预测越准确。

2. 均方根误差（Root Mean Squared Error，RMSE）：均方根误差是均方误差的平方根。它与均方误差类似，但在度量误差时采用了与原始数据相同的单位。RMSE 也常用于衡量模型的预测准确性，越小表示模型的性能越好。

3. 平均绝对误差（Mean Absolute Error，MAE）：平均绝对误差计算预测值与真实值之间差异的绝对值的均值。MAE 衡量了预测结果的平均绝对误差大小，对异常值不敏感。MAE 越小表示模型的性能越好。

4. 决定系数（Coefficient of Determination，R^2）：决定系数是用来衡量回归模型对因变量变异的解释程度。它表示预测值与真实值之间的方差占总方差的比例，取值范围从0到1。R^2 值越接近1表示模型的拟合程度越好。

5. 相关系数（Correlation Coefficient）：相关系数衡量了预测值与真实值之间的线性关系强度和方向。它的取值范围在-1到1之间，接近1表示存在强正相关，接近-1表示存在强负相关，接近0表示没有线性相关性。

需要根据具体的回归问题选择合适的评价指标。有些指标更关注整体误差的大小（如MSE、RMSE、MAE），而有些指标更关注模型拟合程度和相关性（如R^2、相关系数）。根据问题的特点和需求，选择最合适的评价指标进行模型评估。