openGauss AI特性创新实践课



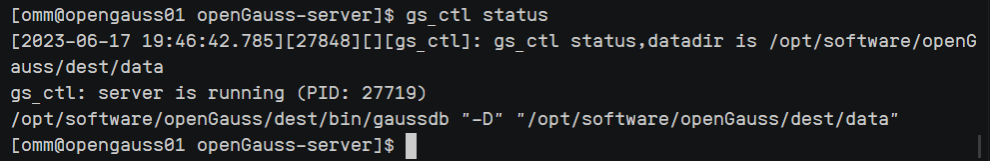
华为技术有限公司

# 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：

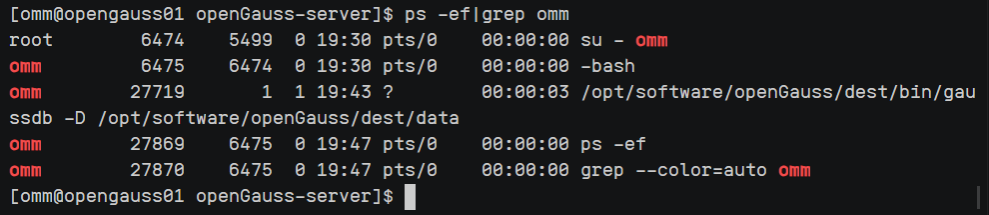
任务一：数据库状态验证

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：1. 定制化配置：通过源码编译安装数据库，可以提供更多的配置选项和灵活性，可以根据自己的需求选择特定的编译选项，启用或禁用某些功能，以满足特定的使用场景或性能要求。从而确保数据库的安装和配置与你的应用程序完全匹配。

2. 最新版本和功能：源码编译安装数据库可以获得最新的版本和功能。官方预编译的二进制包可能会有一定的延迟，而通过源码编译可以保持与开发团队最新的代码同步，获取最新的功能和安全补丁。

3. 性能优化：通过源码编译，可以根据自己的硬件环境和需求进行性能优化。你可以选择适当的编译选项、优化参数和配置设置，以确保数据库在你的硬件上以最佳性能运行。

4. 平台兼容性：有时候预编译的二进制包可能无法满足特定的操作系统或平台要求。通过源码编译，你可以确保数据库能够在你所使用的特定操作系统或平台上正确编译和运行。

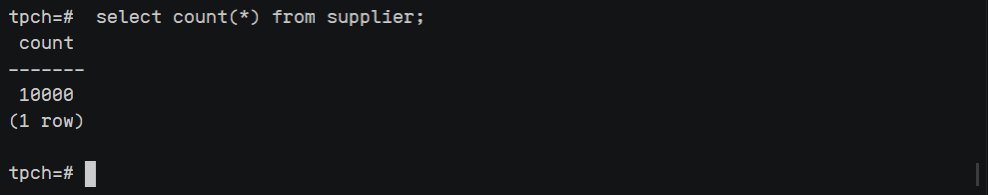
5. 学习和调试：源码编译安装数据库可以更好地了解数据库的内部工作原理。可以学习数据库的实现细节，甚至对数据库进行调试和定制开发。

# 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：

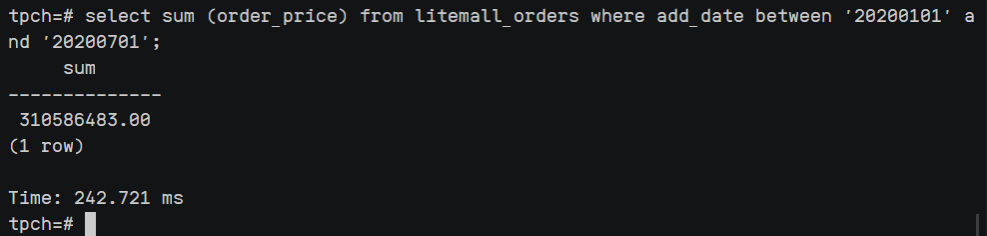
select count(\*) from supplier;;



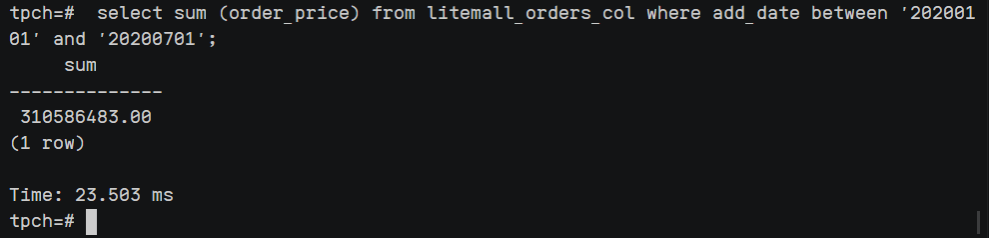
任务二：行存表与列存表执行效率对比

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';

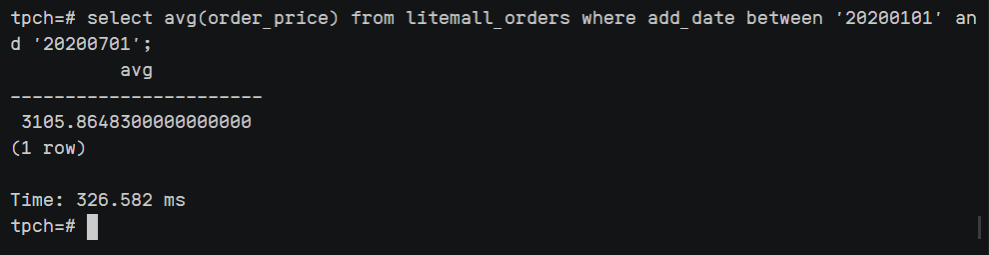


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

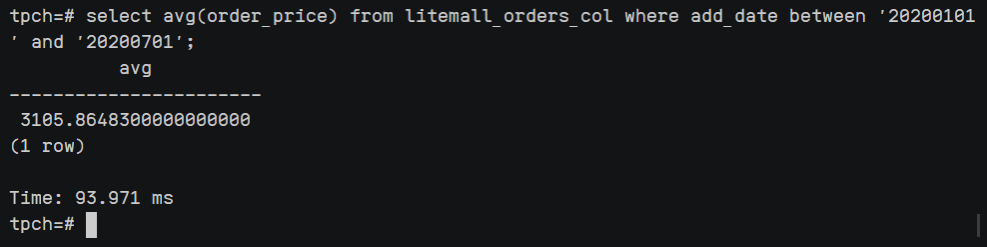


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

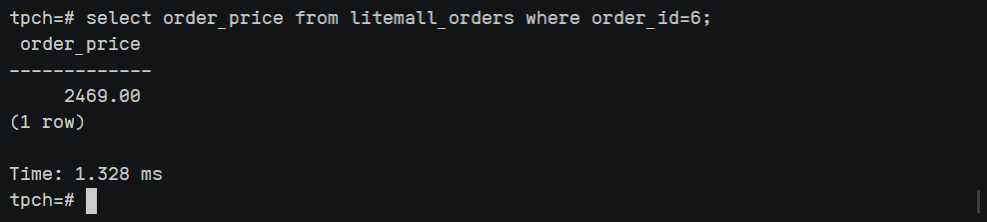


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

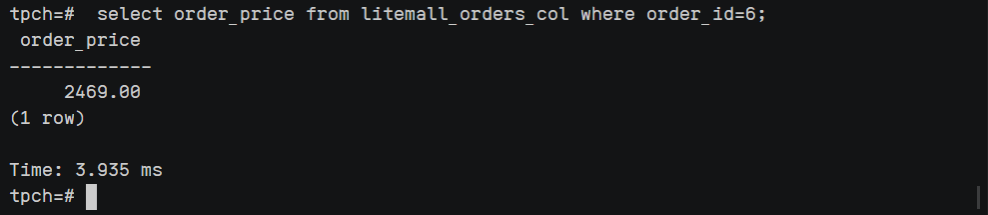


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

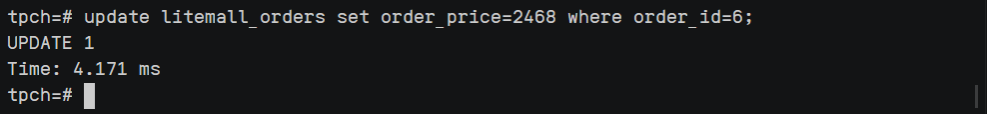


select order\_price from litemall\_orders\_col where order\_id=6;

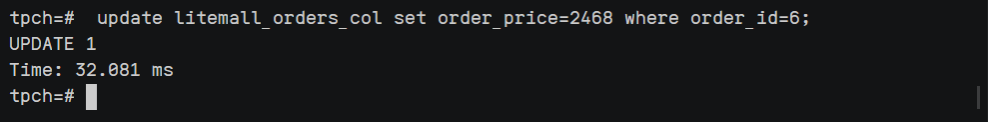


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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



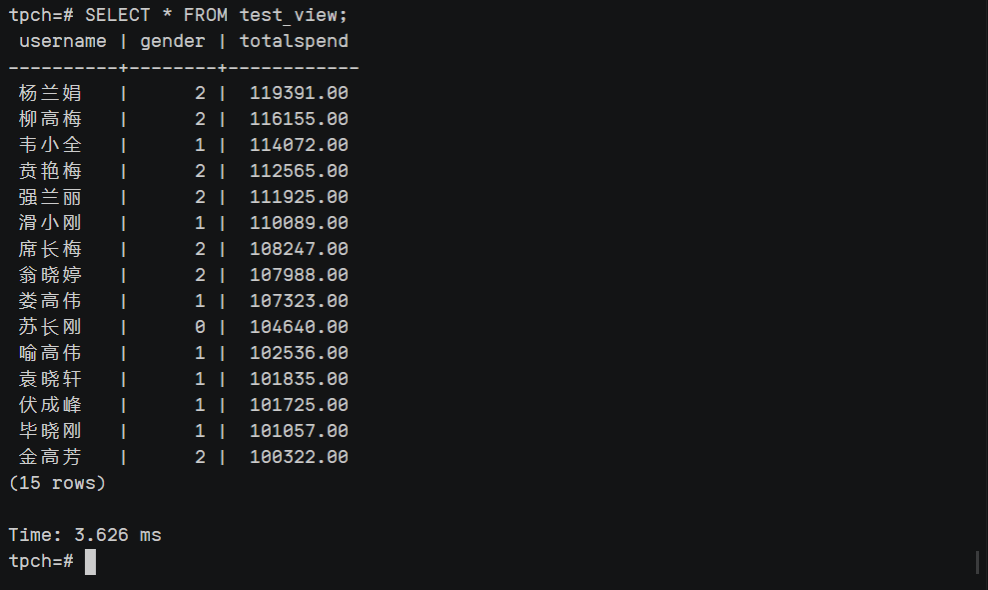
update litemall\_orders\_col set order\_price=2468 where order\_id=6;



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;



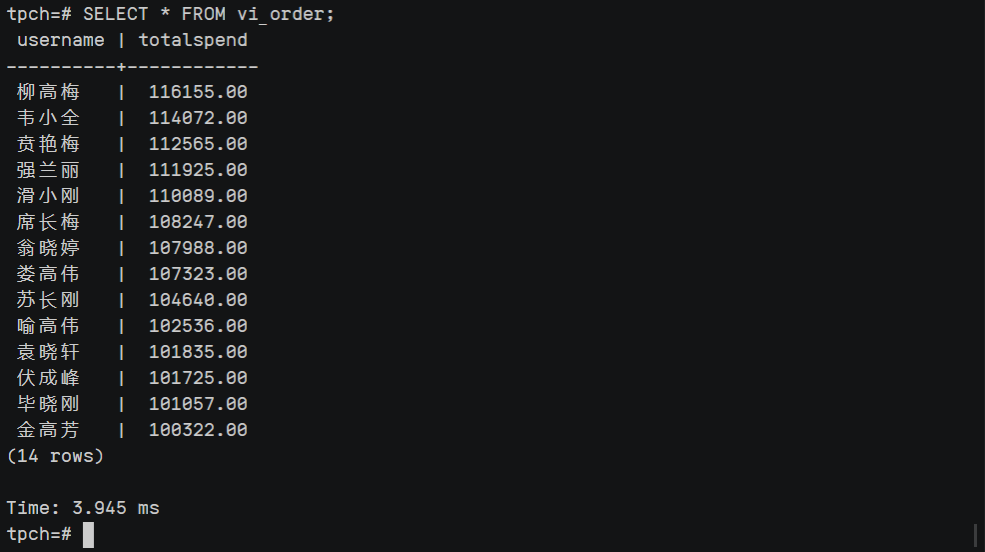
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



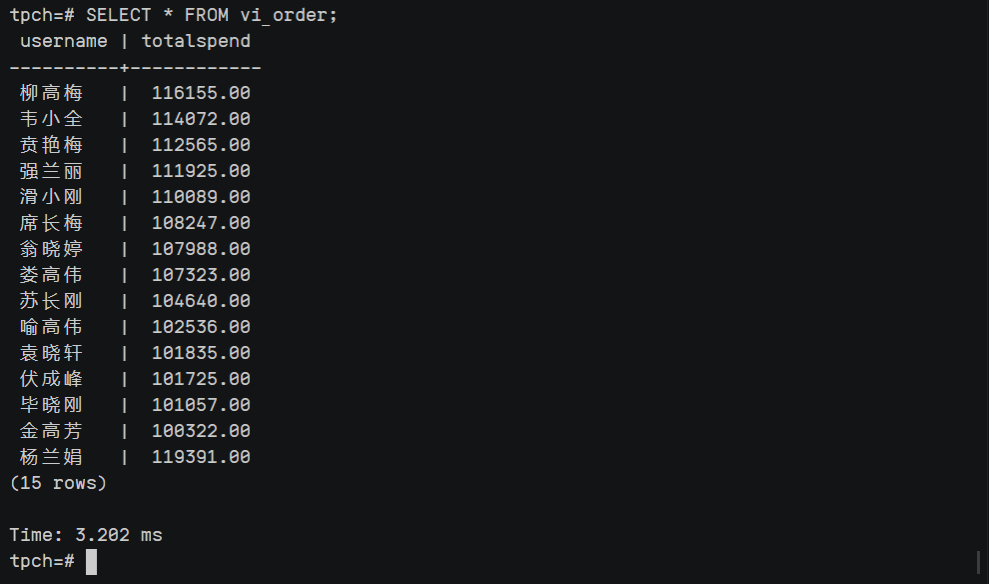
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

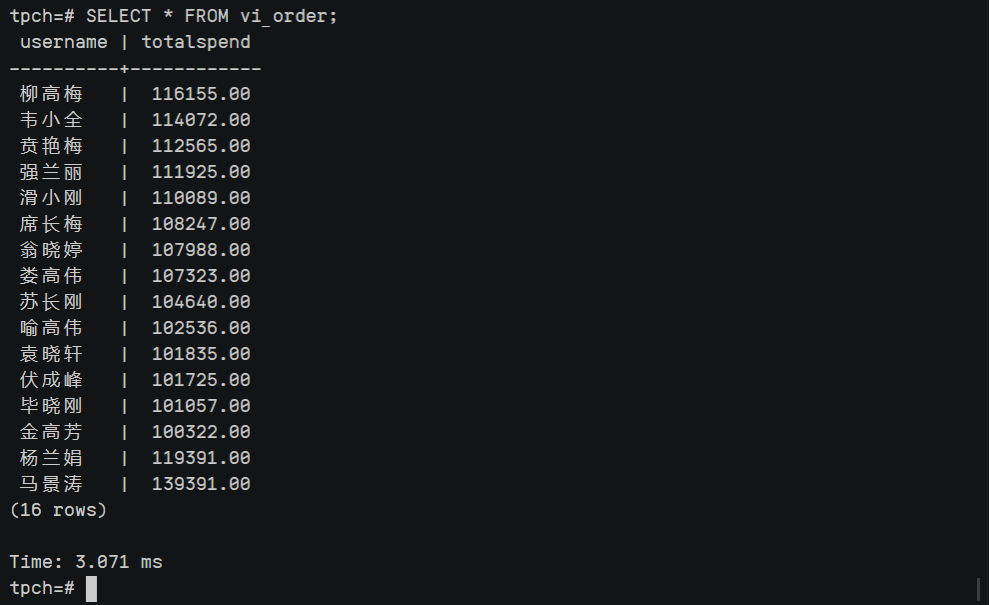
SELECT \* FROM vi\_order;



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？

答：行存表与列存表在执行相同的SQL语句时执行时间不同的原因是由于它们在数据存储和查询执行上的不同原理所致。行存表是将数据按行存储在磁盘上。当执行查询时，数据库需要读取整行的数据，包括不需要的列，然后进行计算和过滤操作。这意味着即使查询只需要部分列的数据，也需要读取整行的数据，导致读取的数据量相对较大，对磁盘 IO 和网络传输造成负担。因此，在行存表中执行查询时，可能会有较多的数据传输和处理，导致执行时间较长。而列存表是将数据按列存储在磁盘上。每个列单独存储，相同列的数据连续存放。这样，在执行只需要部分列的数据查询时，数据库只需要读取所需的列数据，而不需要读取其他列的数据。由于只读取所需的列，可以减少磁盘 IO 和网络传输的数据量，提高查询的效率。此外，列存表通常还会使用诸如数据压缩和批量处理等技术，进一步提高查询性能。

行存表适合于事务处理（OLTP）场景，例如频繁的插入、更新和删除操作。行存表效率更高的情况是需要读取整行数据的查询，如果 SQL 查询需要读取一整行的数据，而不仅仅是部分列数据，行存表的效率可能会更高。因为行存表将整行数据连续存储在磁盘上，读取整行数据的开销相对较小。

存表适合于分析处理（OLAP）场景，例如对大量数据进行聚合、统计和分析操作。列存表在聚合查询和统计分析等需要对列数据进行计算和聚合操作的场景中效率更高。列存表查询涉及大量列的数据子集，当查询只涉及部分列的数据子集时，列存表通常比行存表更高效。列存表只需要读取所需的列数据，而不需要读取其他列的数据，减少了不必要的数据传输和处理开销。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

答：

1. 数据更新方式：全量物化视图通过周期性或手动刷新来更新视图中的数据。每次刷新时，完整地重新计算和存储物化视图的数据。而增量物化视图通过捕获和应用变更日志来更新视图中的数据。只计算和存储发生变化的数据部分，而不需要重新计算整个物化视图。

2. 刷新频率和延迟：全量物化视图需要定期刷新，可以根据需求设置刷新频率。刷新的时间间隔决定了视图中的数据与源数据的一致性，可能存在一定的数据延迟。而增量物化视图实时地捕获和应用变更日志，可以保持与源数据的近实时同步，提供较低的数据延迟。

3. 存储和性能开销：全量物化视图存储和计算整个数据集，占用更多的存储空间，并且在每次刷新时需要重新计算和加载全部数据，可能导致较高的性能开销。而增量物化视图只存储和计算变更的数据部分，占用更少的存储空间，并且只需要处理变更数据，可以降低性能开销。

4. 使用场景：全量物化视图适用于对源数据进行周期性统计、聚合或计算的场景，数据更新频率较低且可以接受一定的数据延迟。而增量物化视图适用于需要实时或近实时分析和查询的场景，数据更新频率高且需要较低的数据延迟。

综上，全量物化视图适用于较稳定的数据和低延迟要求，而增量物化视图适用于频繁变更的数据和近实时的查询需求。

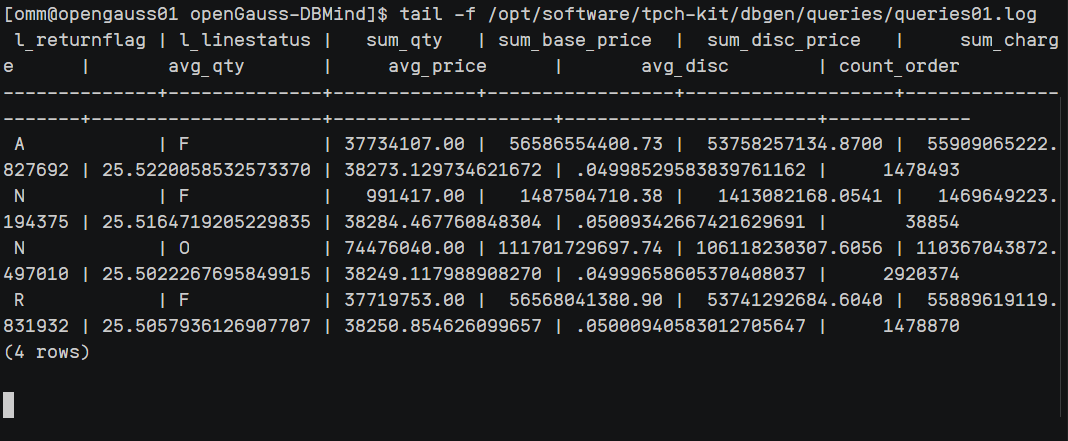
# 关卡三、openGauss的AI4DB特性应用

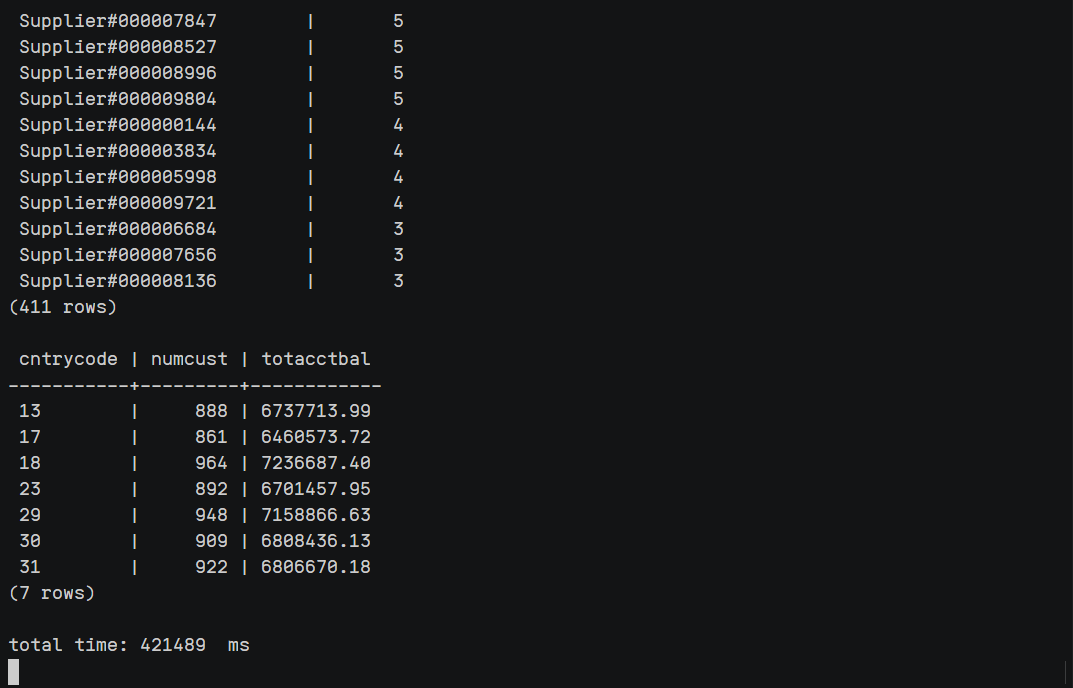
任务一：使用X-Tuner进行参数优化

1. 执行TPCH脚本，获得测试时间，将执行结果截图：

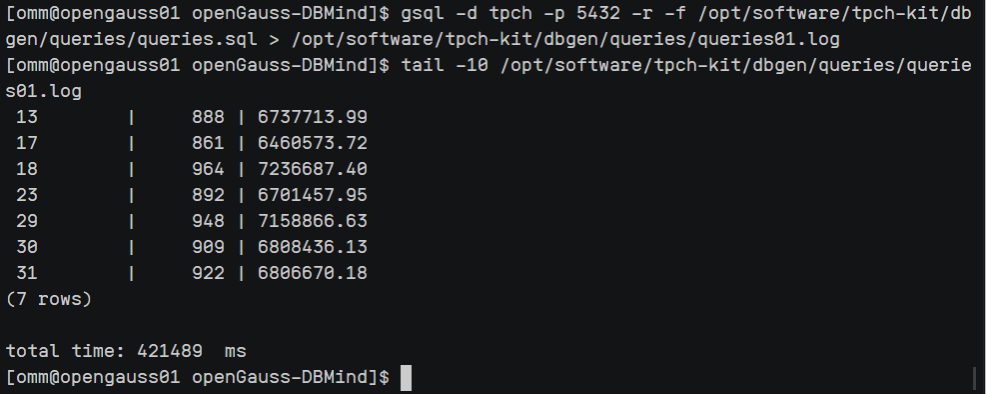
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log

在另一个连接窗口中执行如下内容进行监控，内容如下（部分截图）：



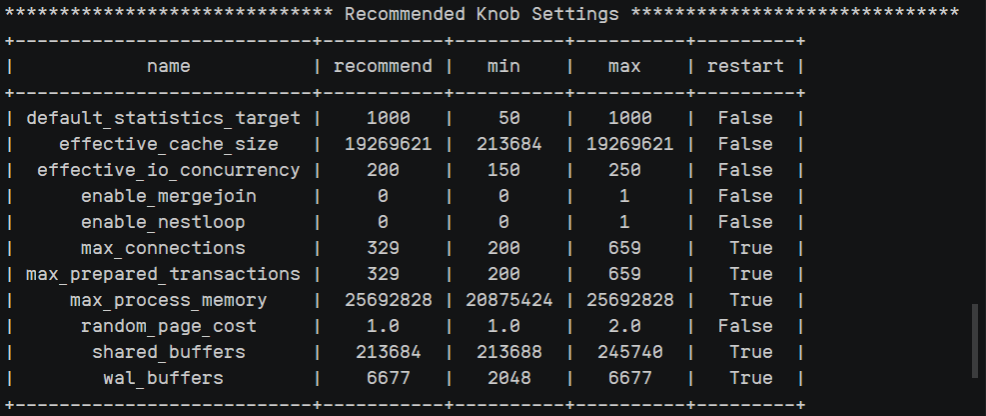


在原来连接窗口中查看queries01.log，结果如下：



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

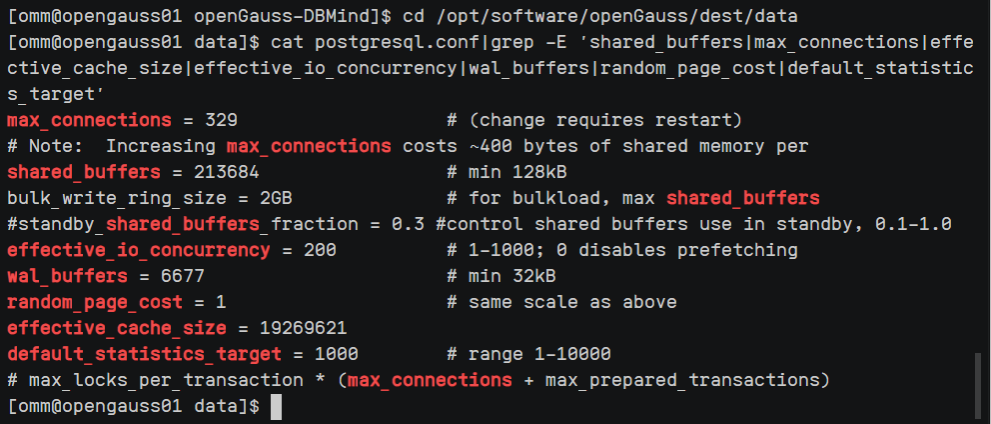
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

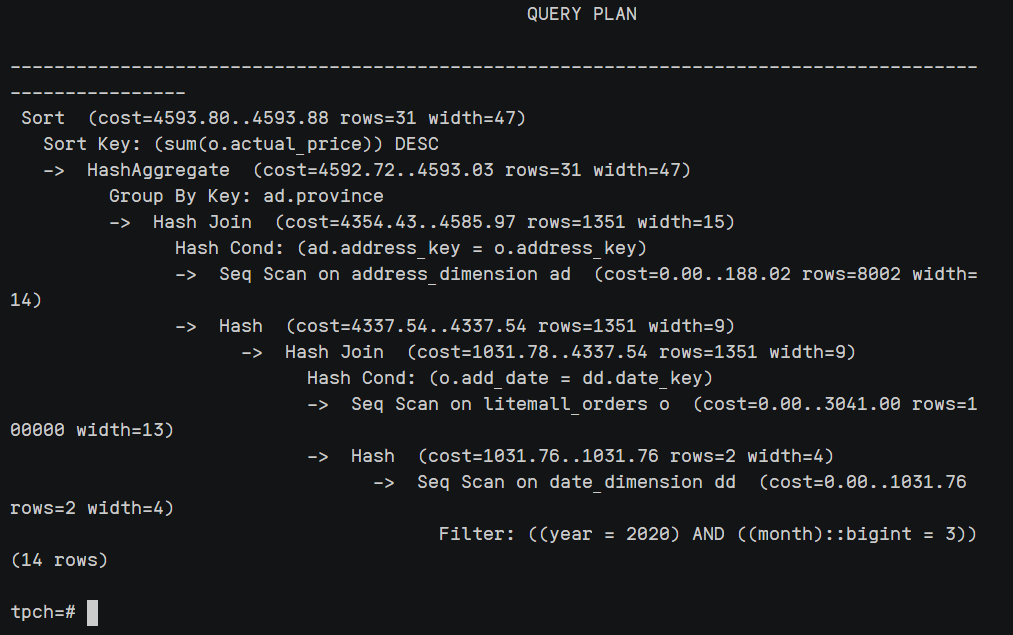
AND o.add\_date = dd.date\_key

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

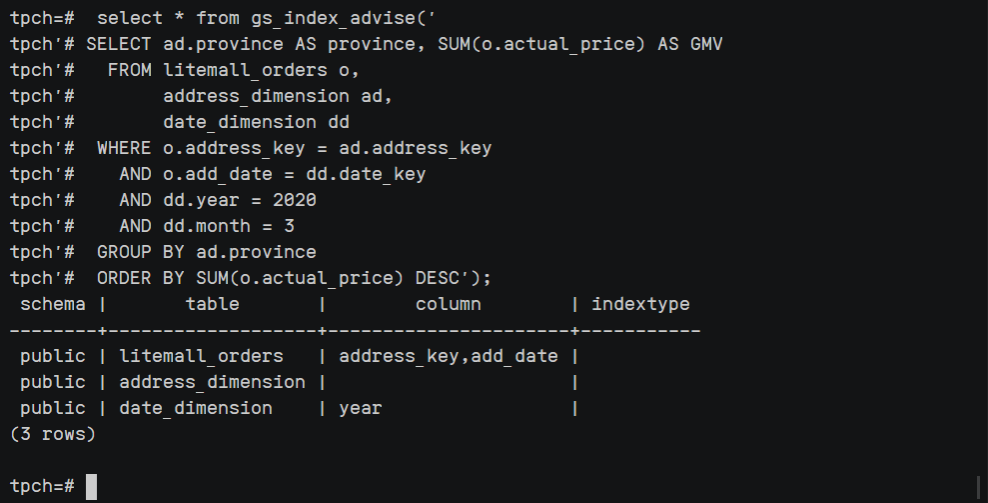
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

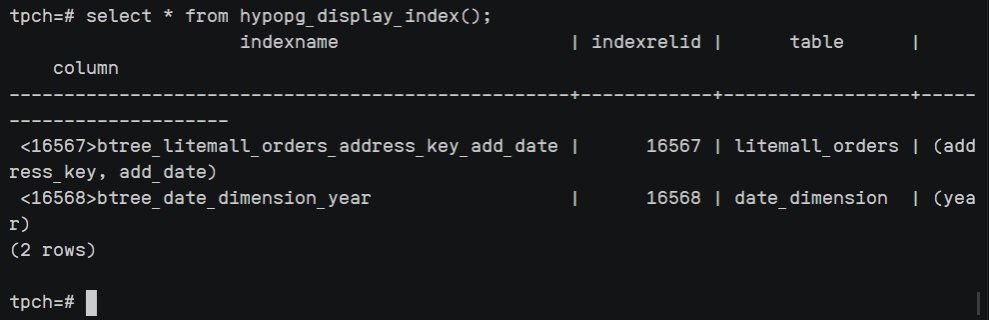
GROUP BY ad.province

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。

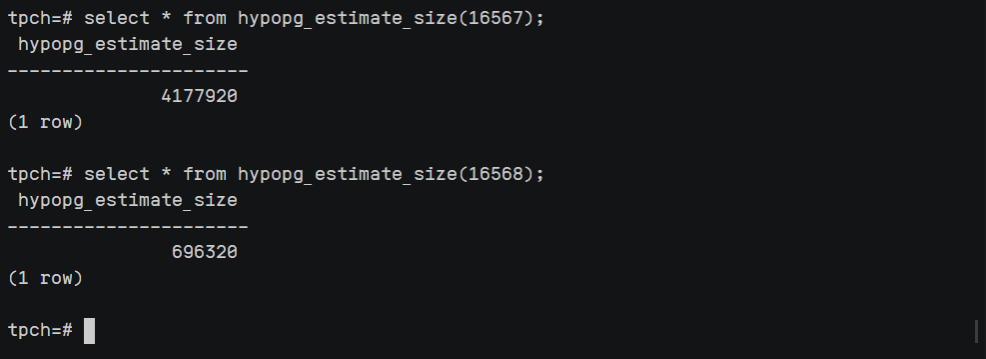
select \* from hypopg\_display\_index();



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

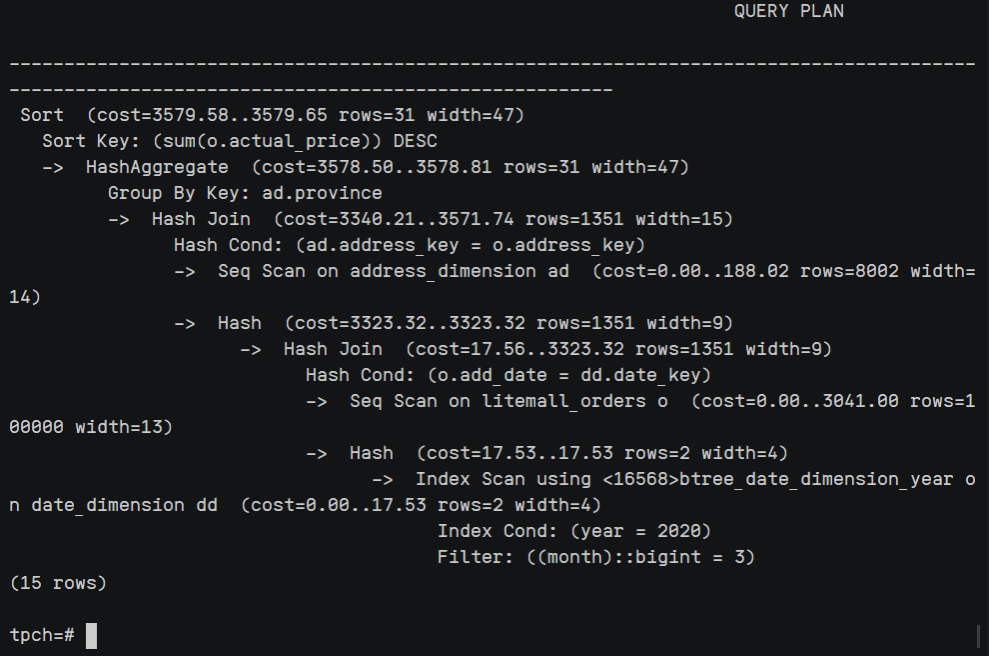
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

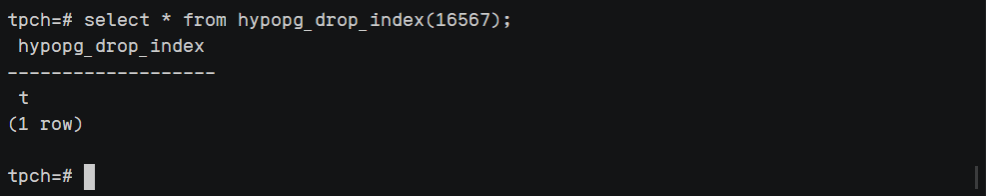
GROUP BY ad.province

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



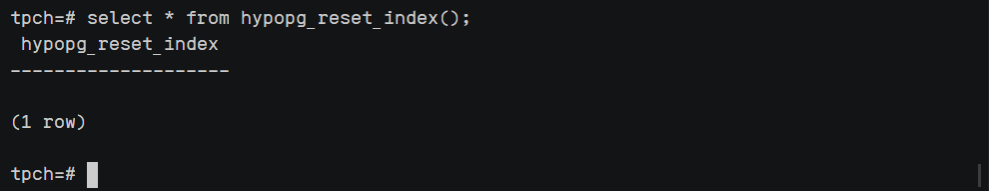
6. 删除某一个索引虚拟列，将执行结果截图。

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



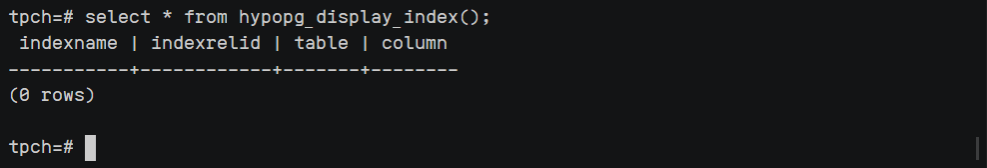
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

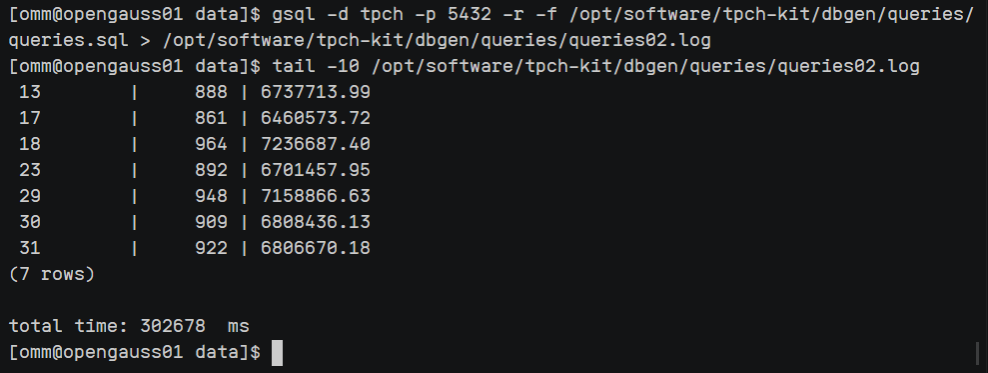
select \* from hypopg\_display\_index();



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：

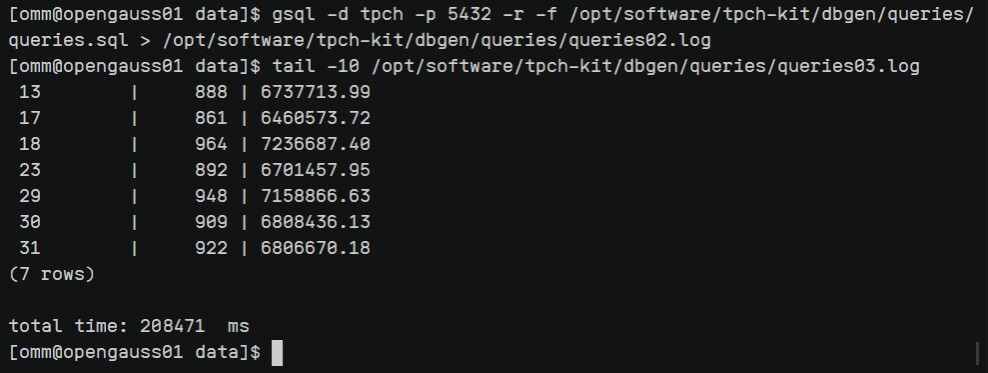
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log



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？

答：在default\_statistics\_target、effective\_cache\_size、effective\_io\_concurrency、max\_connections、random\_page\_cost、shared\_buffers、wal\_buffers这些参数上进行了优化。使用 gs\_xtuner recommend 命令，gs\_xtuner 工具将分析数据库的性能和配置，并推荐最优的参数配置。优化这些参数可以根据实际的工作负载、硬件配置和性能要求，提高查询执行的效率、减少资源消耗、加快事务处理速度，并改善数据库系统的整体性能。

具体每个参数优化的原因如下：

1.default\_statistics\_target：该参数决定了统计信息的采样率。通过优化该参数，可以提高查询优化器对表数据的统计分析准确性，从而产生更好的查询计划，提高查询性能。2.effective\_cache\_size：该参数指定了 PostgreSQL 用于查询计划估算的缓存大小。通过优化该参数，可以确保查询计划的成本估算更加准确，提高查询执行的效率。

3.effective\_io\_concurrency：该参数指定了并行 I/O 操作的并发度。通过优化该参数，可以提高对存储设备的并发读取和写入操作，从而改善数据访问性能。

4.max\_connections：该参数限制了数据库的最大并发连接数。通过优化该参数，可以合理配置连接数，以适应实际的并发负载，并避免资源耗尽和性能下降。

5.random\_page\_cost：该参数用于估算随机 I/O 操作的成本。通过优化该参数，可以更准确地评估随机页面访问的开销，从而改进查询计划的选择和性能。

6.shared\_buffers：该参数指定了数据库共享缓冲区的大小。通过优化该参数，可以增加缓存的数据量，减少磁盘 IO 操作，提高数据访问性能。

7.wal\_buffers：该参数指定了 WAL 缓冲区的大小。通过优化该参数，可以控制写入日志的缓冲区大小，从而提高事务提交的性能和可靠性。

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？

答：

使用索引在执行 SQL 查询时有以下好处：

1. 提高查询性能：索引可以加速查询操作，减少数据库系统需要扫描的数据量。通过索引，数据库系统可以更快地定位到满足查询条件的数据行，从而减少了磁盘 IO 操作和数据的读取时间，提高了查询的执行速度。

2. 减少数据排序和聚合的开销：索引可以为排序和聚合操作提供有序的数据，减少了数据库系统需要进行的排序和聚合操作的开销。对于包含大量数据的表，使用索引可以避免全表扫描和临时存储，提高了排序和聚合操作的效率。

3. 改善并发性能：索引可以减少数据访问冲突，提高并发性能。通过使用索引，数据库系统可以更好地处理并发查询，减少了锁冲突和资源争用，提高了并发查询的吞吐量和响应性能。

4. 提供唯一性约束：索引可以用于实施唯一性约束，保证表中的某个列或列组合的值是唯一的。这样可以防止重复数据的插入和更新，确保数据的完整性和一致性。

5. 支持连接操作：索引可以加速连接操作（如 JOIN 操作），减少连接过程中需要比较的数据量。通过索引，数据库系统可以更快地定位连接操作所需的关联数据行，提高连接操作的执行效率。

除了使用索引和参数外，还有以下方面可以对数据库进行优化：

1. 数据库设计优化：良好的数据库设计可以提高数据库的性能和扩展性。这包括合理的表结构设计、正确选择数据类型、定义适当的约束和关系等。优化数据库设计可以减少数据冗余、提高数据一致性，同时优化查询和操作的效率。

2. 查询优化：优化数据库查询是提高性能的重要步骤。这包括编写有效的查询语句、避免全表扫描、合理使用连接操作、避免重复或不必要的查询等。使用正确的查询语法和优化技巧可以减少查询时间和资源消耗。

3. 硬件优化：数据库性能受到底层硬件设备的影响。优化硬件可以包括增加内存容量、优化磁盘子系统、使用高性能的存储设备、提升网络带宽等。通过合理配置和优化硬件，可以提高数据库的读写性能和响应速度。

4. 数据库连接池管理：对于应用程序连接数据库的管理也是优化的一部分。合理配置数据库连接池的大小、连接超时时间和闲置连接的回收策略，可以提高连接的复用性和应用程序的性能。

5. 数据库备份和恢复策略：合理的备份和恢复策略可以确保数据的安全性和可靠性。定期备份数据库并测试恢复过程，以防止数据丢失和减少系统故障对业务的影响。

6. 数据库统计信息维护：数据库统计信息对于查询优化和查询计划的选择非常重要。定期更新统计信息并进行统计信息的收集和维护，可以提供准确的查询计划和优化器的决策依据，提高查询性能。

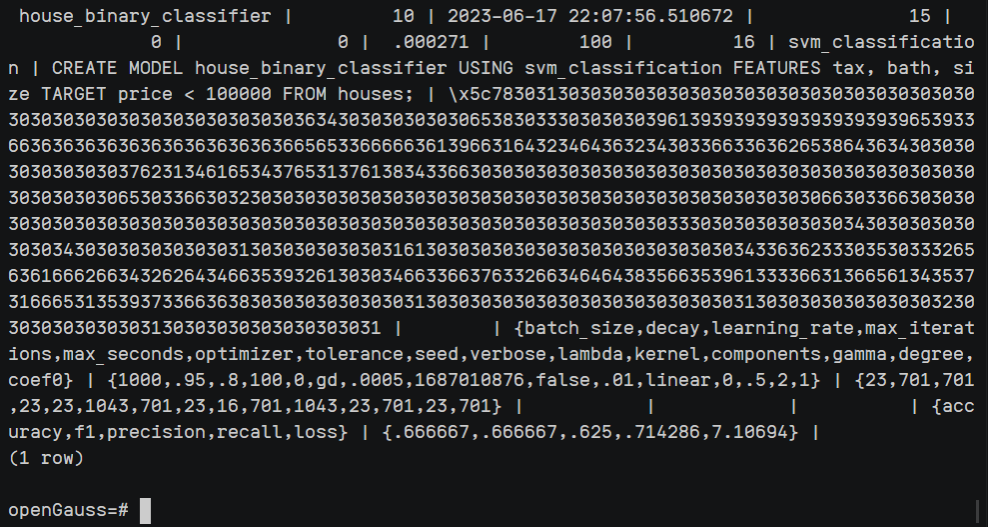
7. 查询缓存和缓存管理：数据库查询缓存可以减少对磁盘的读取操作，提高查询性能。优化查询缓存的大小和缓存的存储策略，可以更好地利用缓存，提高查询的响应速度。

8. 并发控制和事务管理：合理管理并发访问和事务的并发控制，可以避免锁冲突和死锁问题，提高并发性能和系统的可用性。

# 关卡四、openGauss的DB4AI特性应用

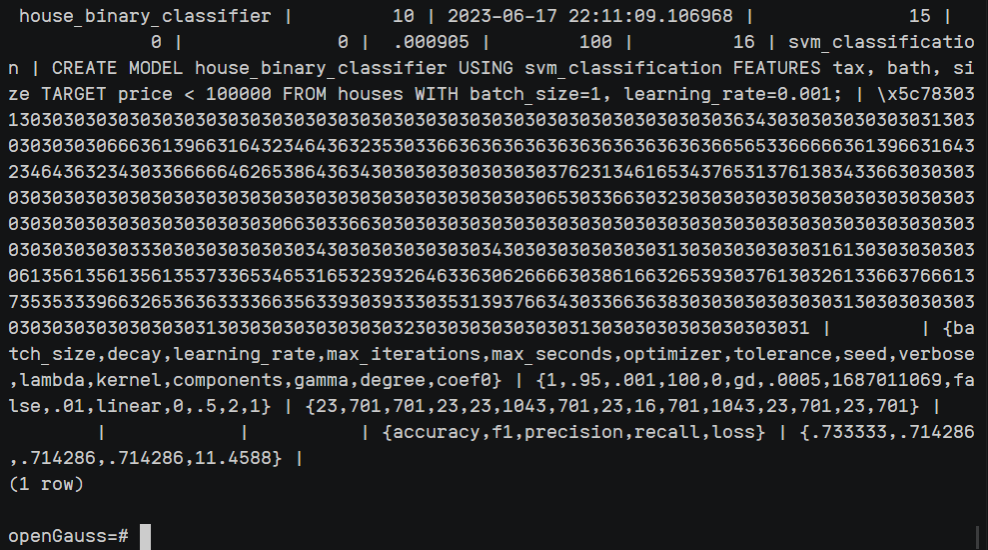
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



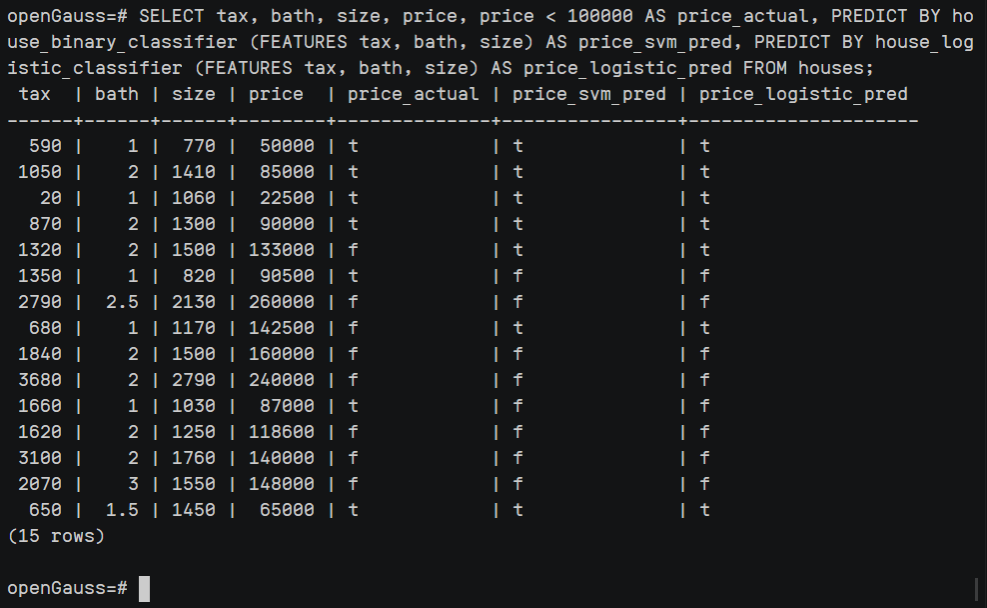
任务二：观察新模型的信息，将执行结果截图。

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：

1. 目标变量类型：分类模型用于预测离散的类别或标签，目标变量是有限个预定义的类别。而回归模型用于预测连续的数值，目标变量是一个连续的数值。

2. 输出类型：分类模型的输出是对样本进行分类的结果，通常是一个类别标签或类别的概率分布。回归模型的输出是一个连续的数值，可以是实数或浮点数。

3. 模型类型：分类模型使用的是分类算法，例如逻辑回归、决策树、支持向量机、随机森林等。回归模型使用的是回归算法，例如线性回归、多项式回归、岭回归、神经网络等。

4. 损失函数：分类模型通常使用交叉熵等损失函数来衡量分类错误的程度。回归模型通常使用均方误差、平均绝对误差等损失函数来衡量预测值与真实值之间的差异。

5. 模型解释性：分类模型通常更容易解释和理解，因为结果是对应离散类别的预测。回归模型的结果可能更难以解释，因为它们是对连续数值的预测。

6. 数据分布假设：回归模型通常对数据的分布做出一些假设，例如线性回归假设数据呈现线性关系。分类模型不对数据分布做出明确的假设，而是通过训练数据中的模式来推断类别之间的关系。

实践思考题2：什么是SVM算法？

答：SVM是一种常用的监督学习算法，广泛用于分类和回归任务。SVM 的基本思想是通过在特征空间中找到一个最优的超平面来实现样本的分类。超平面是一个 N-1 维的线性子空间，其中 N 是特征的维数。对于二分类任务，SVM 的目标是找到一个能够最大化两个类别之间的间隔（即支持向量）的超平面。间隔是指超平面到最近的训练样本点的距离。在现实情况中，数据往往不是完全线性可分的。SVM 引入了松弛变量来容忍一些样本分类错误和噪声。松弛变量允许一些样本位于超平面错误的一侧或在间隔内部。当数据不是线性可分时，SVM 通过使用核函数将特征空间映射到更高维的特征空间，从而使数据在新的特征空间中更容易线性分离。常用的核函数有线性核、多项式核、高斯核等。

SVM 算法的优势在于对于高维数据的处理能力、对噪声数据的鲁棒性以及泛化能力较强。此外，SVM 可以通过调整超参数和选择合适的核函数进行灵活的模型适应。然而，对于大规模数据集和非线性问题，SVM 的计算复杂性可能会增加，并且对于不平衡数据集和噪声敏感。

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：

1. 准确率：准确率是最简单直观的评价指标，表示分类正确的样本占总样本数的比例。

2. 精确率：精确率衡量的是被分类器判断为正例的样本中，真正为正例的比例。精确率的高值表示分类器在判断为正例时更准确。

3. 召回率：召回率衡量的是真正为正例的样本中，被分类器正确判断为正例的比例。召回率的高值表示分类器在捕捉正例方面更全面。

4. F1 值：F1 值是精确率和召回率的调和平均值，综合考虑了分类器的准确性和完整性。计算公式为：



F1 值的范围在 0 到 1 之间，值越高表示分类器性能越好。

5. 特异度（Specificity）：特异度衡量的是被分类器判断为负例的样本中，真正为负例的比例。计算公式为：



特异度的高值表示分类器在判断为负例时更准确。

6. AUC-ROC：AUC-ROC 是综合考虑分类器在不同阈值下的准确性和召回率的评价指标。ROC 曲线表示分类器在不同阈值下的真正例率（TPR）和假正例率（FPR）之间的关系。AUC-ROC 的取值范围在 0.5 到 1 之间，值越接近 1 表示分类器性能越好。

准确率适用于类别平衡的情况，而精确率和召回率适用于类别不平衡或关注特定类别的情况。F1 值综合考虑了精确率和召回率，适用于在准确性和完整性之间需要平衡的情况。AUC-ROC 则适用于评估分类器在不同阈值下的整体性能。

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：

1. 均方误差（MSE）：均方误差是回归预测结果与真实值之间差异的平方的平均值。计算公式为：



其中 y 是真实值，是预测值，n 是样本数量。MSE 值越小表示模型预测的结果越接近真实值。

2. 均方根误差（RMSE）：均方根误差是均方误差的平方根，用于衡量模型预测结果与真实值之间的差异。计算公式为：



RMSE 的值与原始数据的单位相同，越小表示模型的预测精度越高。

3. 平均绝对误差（MAE）：平均绝对误差是回归预测结果与真实值之间差异的绝对值的平均值。计算公式为：

。

MAE 衡量的是预测结果的平均偏差程度，数值越小表示模型的预测准确性越高。

4. 决定系数（R²）：决定系数衡量模型对目标变量方差的解释程度，即模型拟合数据的程度。取值范围在 0 到 1 之间，越接近 1 表示模型对目标变量的解释能力越强。计算公式为：



其中 y 是真实值，y\_pred 是预测值，y\_mean 是真实值的均值。

MSE、RMSE 和 MAE 衡量了模型预测结果与真实值之间的差异，越小表示预测的准确性越高。R² 衡量了模型对目标变量的解释程度，越接近 1 表示模型的拟合能力越好。