openGauss AI特性创新实践课



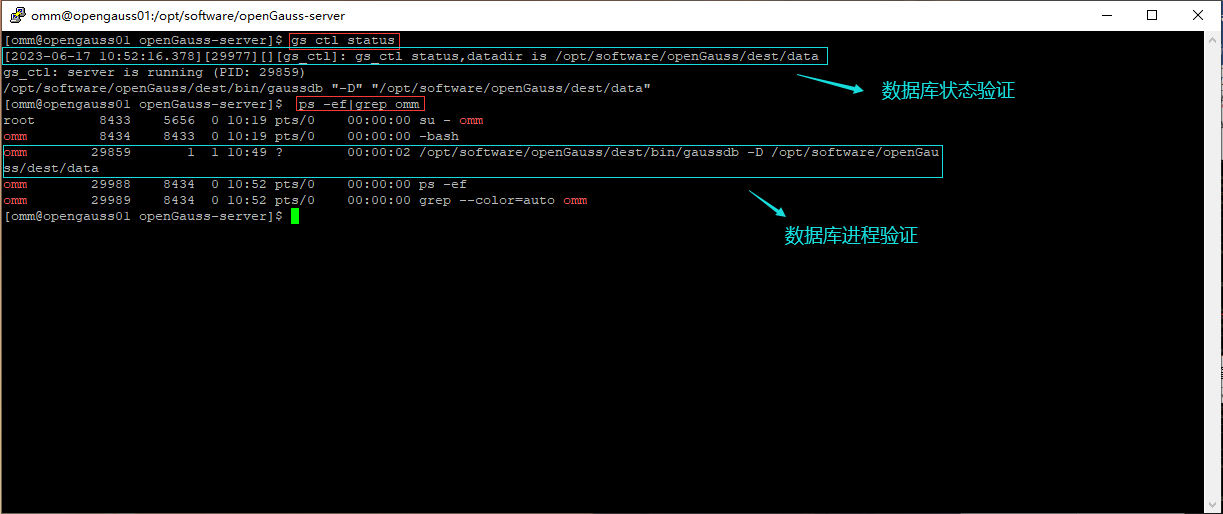
华为技术有限公司

# 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：

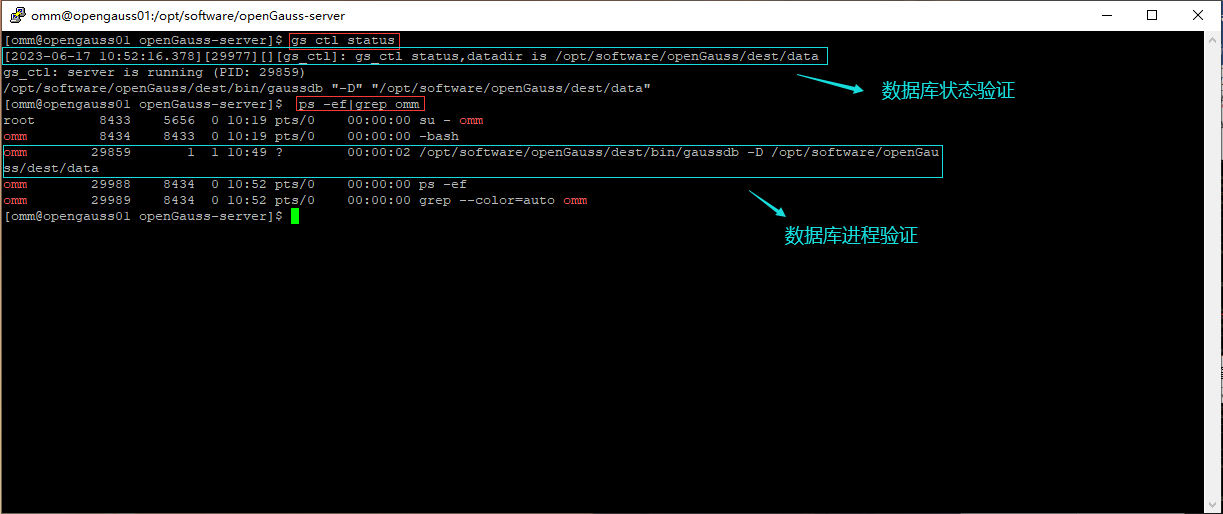
任务一：数据库状态验证

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？

通过源码编译安装数据库相较于使用预编译的二进制安装包，主要有以下几个好处：

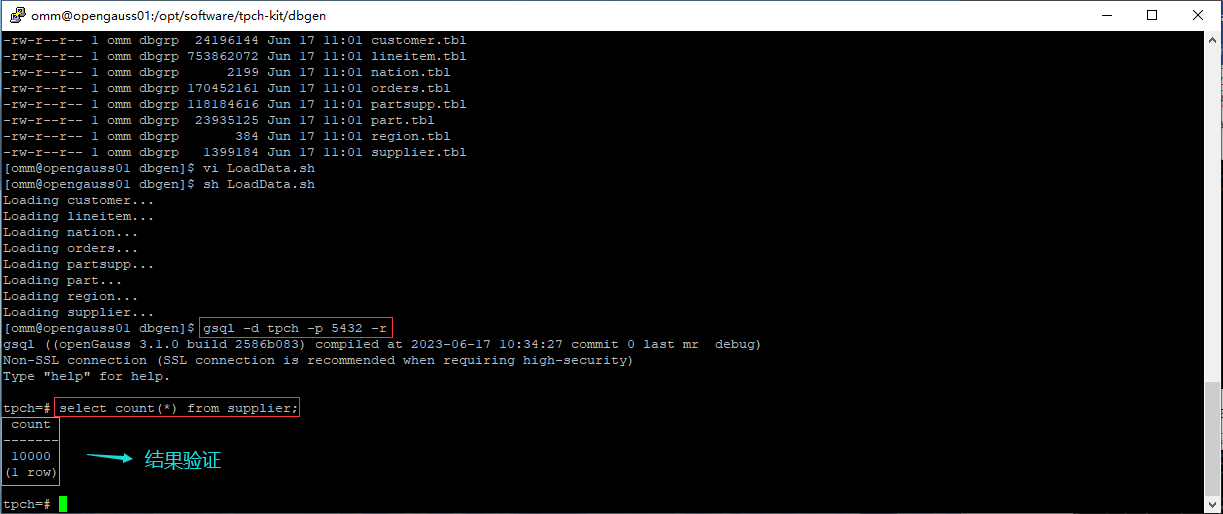
1. 定制化配置：通过源码编译，可以灵活地自定义数据库的安装选项和配置参数，以满足特定需求。你可以根据自己的环境和需求进行适配和优化，从而更好地发挥数据库性能和功能。
2. 最新版本和新特性：通过源码编译安装，你可以选择最新版本的数据库，并能即使地获取新发布的功能和修复的安全漏洞。这对于开发人员和系统管理员来说尤其重要，因为他们需要最新的特性来满足业务需求，并希望修复遗留问题和漏洞。
3. 安全性和可靠性：源码编译可以确保你使用的是官方发布的源代码，减少了潜在的恶意篡改和供应链攻击的风险。此外，通过源码编译可以进行更精细的调整和优化，提高数据库的稳定性和可靠性。
4. 高性能和扩展性：通过源码编译，你可以针对特定硬件环境和平台进行优化，从而获得更好的性能和扩展性。例如，可以针对特定的CPU架构进行代码优化，或者启用特定的编译选项以提高性能。

# 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：

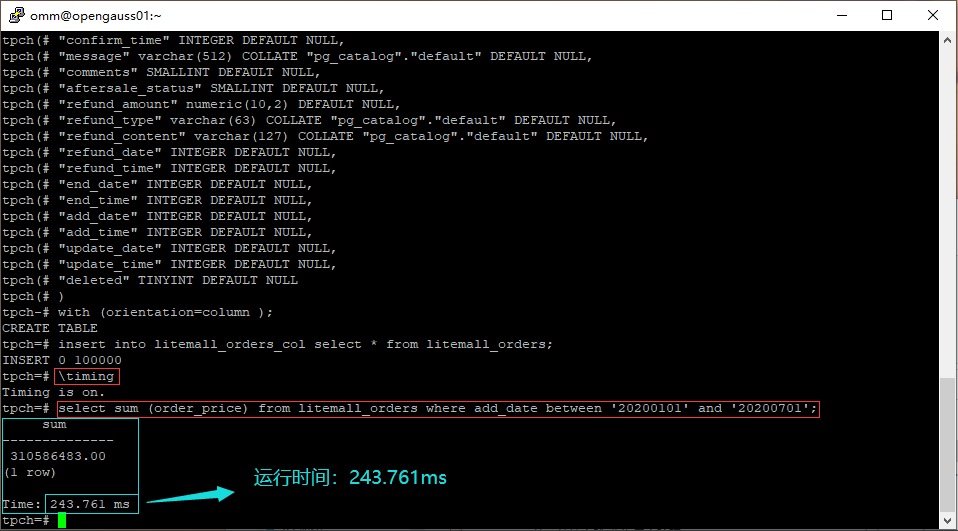
select count(\*) from supplier;;



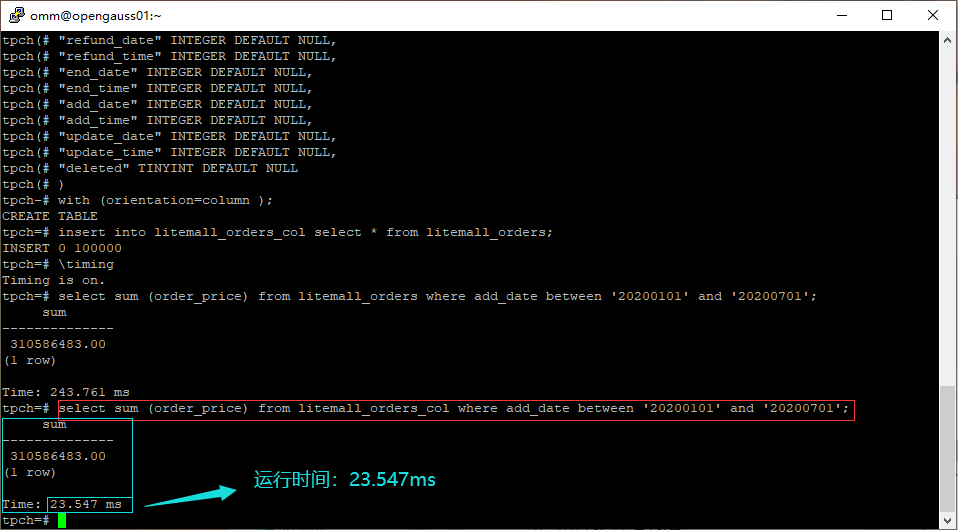
任务二：行存表与列存表执行效率对比

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';

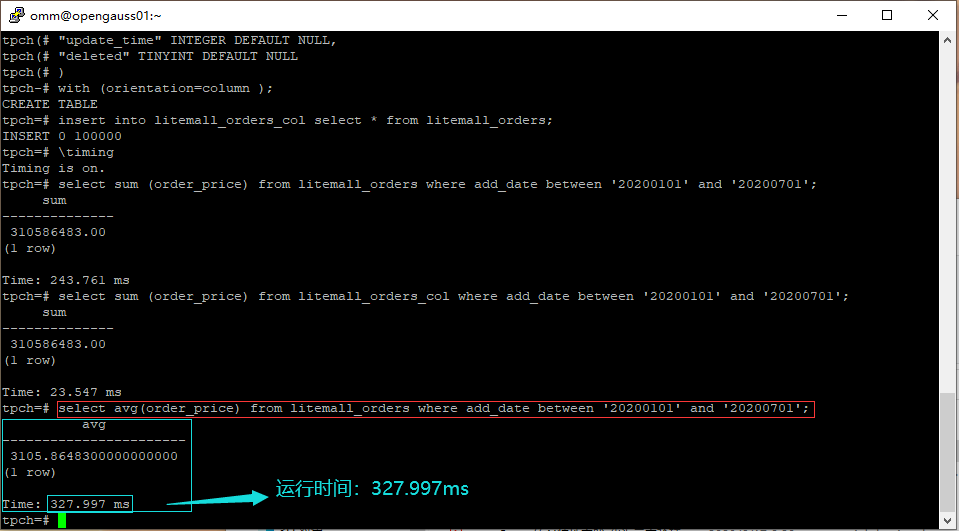


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

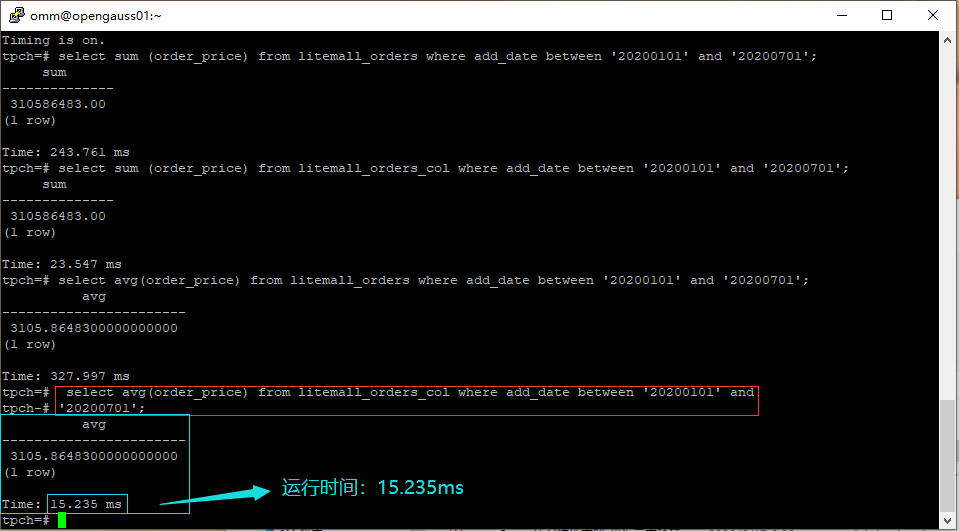


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

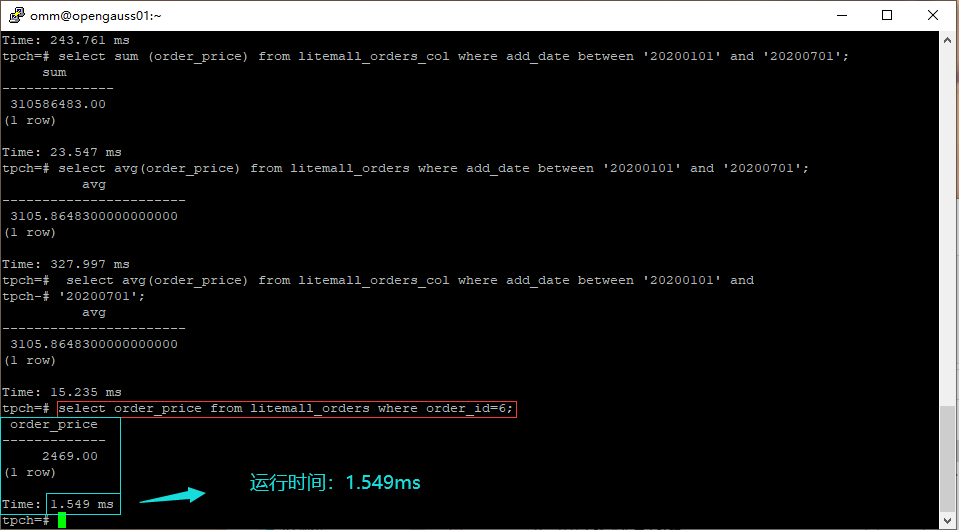


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

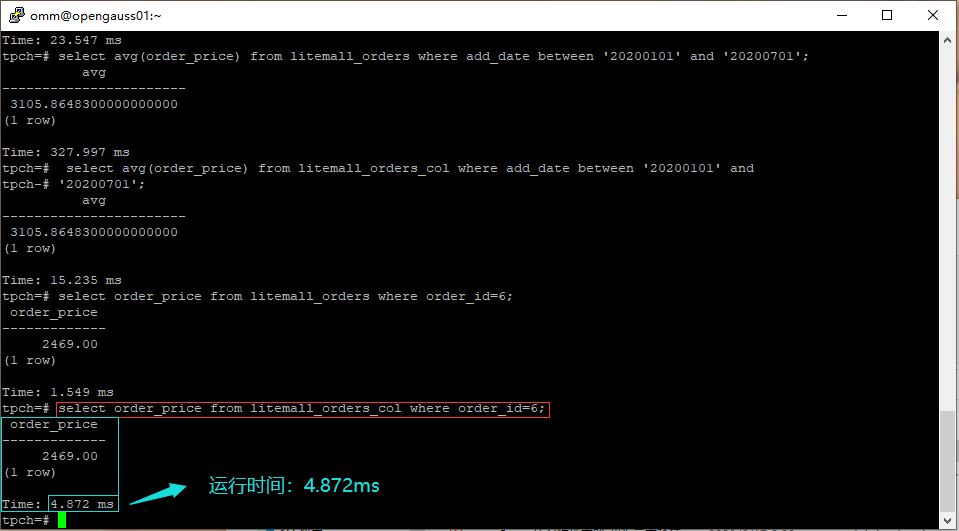


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

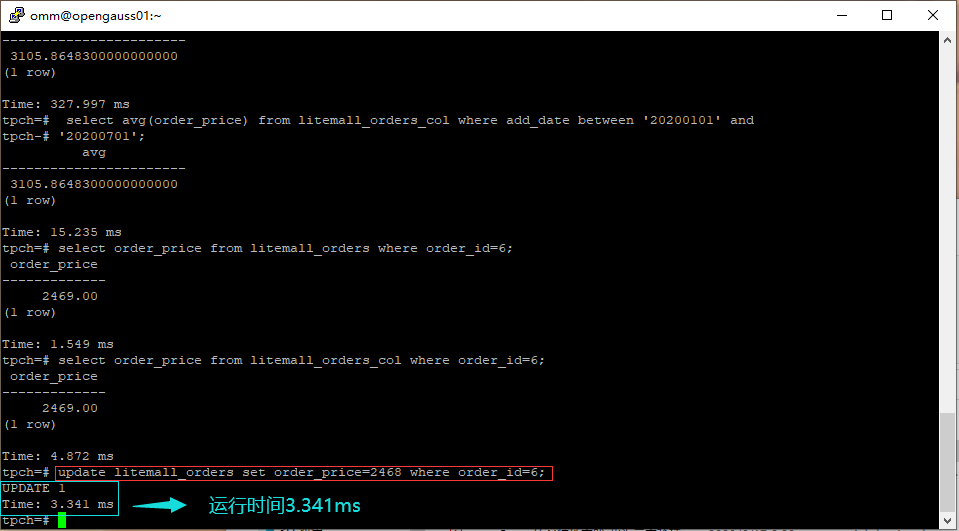


select order\_price from litemall\_orders\_col where order\_id=6;

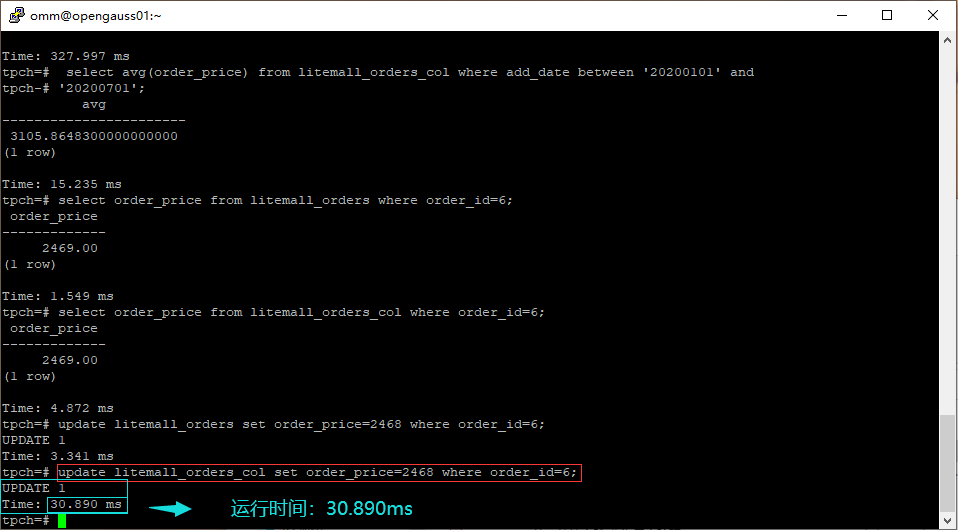


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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



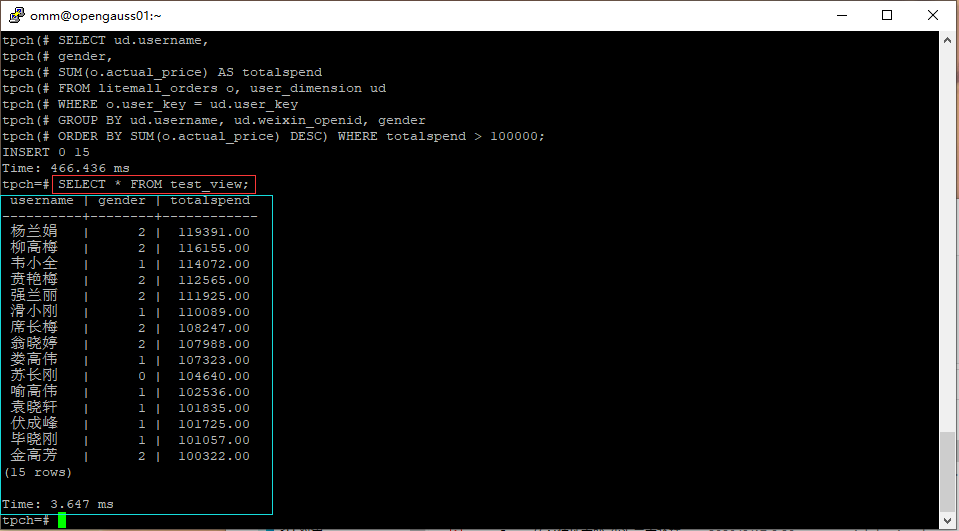
update litemall\_orders\_col set order\_price=2468 where order\_id=6;



任务三：物化视图的使用

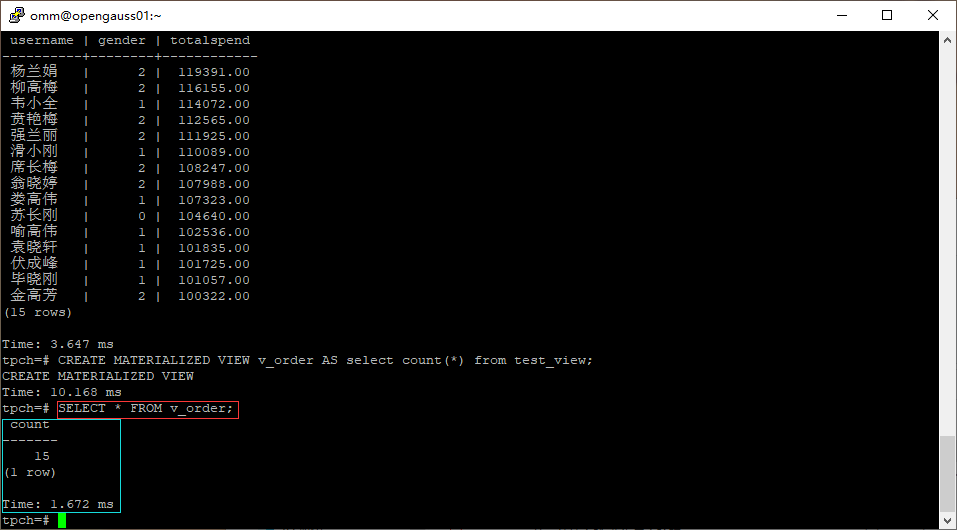
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



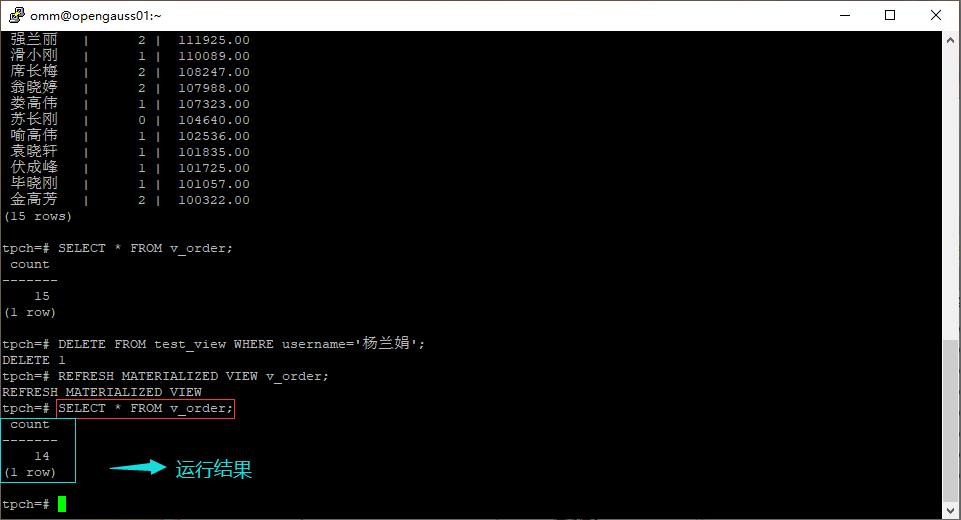
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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？

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主要有以下几点：

1. 数据存储方式不同：行存表以行为单位存储数据，而列存表以列为单位存储数据。这意味着在读取数据时，行存表需要读取整行数据，而列存表只需要读取所需列的数据。因此，在某些情况下，列存表可以避免读取不必要的数据，提高读取效率。
2. 压缩算法不同：列存表通常使用更高效的压缩算法来减少存储空间，而行存表通常使用较简单的压缩算法或没有压缩。这意味着列存表在存储数据时可以减少磁盘空间的使用，从而减少了IO操作的数量，提高了查询性能。

行存表效率更高的情况包括：

1. 需要频繁进行数据更新、插入和删除操作的场景，因为行存表的写入性能通常比列存表更好。
2. 需要进行复杂的联结操作和查询的场景，因为行存表可以通过访问整行数据，减少在联结操作时的数据访问次数和数据移动量。

列存表效率更高的情况包括：

1. 需要进行大规模的数据分析、聚合和统计查询的场景，因为列存表可以仅读取特定的列数据，减少不必要的IO操作和数据传输。
2. 需要进行高效的数据压缩的场景，因为列存表通常采用列式的压缩算法，可以极大地减少存储空间占用。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

全量物化视图和增量物化视图是物化视图的两种常见类型，它们之间的差别主要体现在以下几个方面：

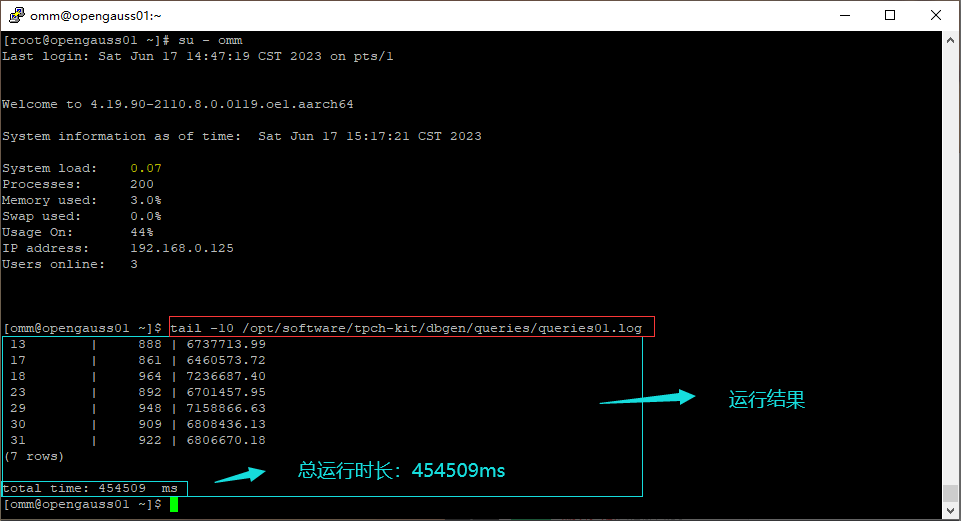
1. 数据更新方式：全量物化视图会将源表的所有数据完全复制到物化视图中，当源表数据发生变化时，需要重新计算和刷新整个物化视图的数据。而增量物化视图只存储源表数据的部分结果，通常是通过存储增量变化的方式来更新物化视图。增量物化视图会记录源表数据的变化，并根据变化来更新物化视图的结果，以减少刷新的开销。
2. 刷新频率：由于需要重新计算和复制整个数据集，全量物化视图的刷新通常比较耗时，因此刷新频率较低。而增量物化视图只需要处理源表数据的增量变化，因此刷新的开销较小，可以更频繁地进行刷新，以保持与源表数据的同步。
3. 数据一致性：全量物化视图在刷新时，可能会由于刷新过程中的并发更新操作或延迟而导致物化视图不与源表实时保持一致。增量物化视图通过记录源表变化并逐步应用变化来更新视图，可以更好地保持与源表数据一致性。
4. 查询性能：全量物化视图在刷新时需要重新计算和复制整个数据集，因此在查询时可以直接从物化视图中获取结果，查询性能较高。增量物化视图在刷新时只需要处理源表的增量变化，因此在查询时可能需要进行其他操作（例如合并增量变化），查询性能相对较低。
5. 使用场景：全量物化视图适用于数据量较小，刷新频率较低，但需要提供快速查询结果的场景。例如，用于报表、汇总查询等。增量物化视图则适用于数据量较大，刷新频率较高，需要实时或近实时保持与源表一致的场景。例如，用于实时分析、数据同步等。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

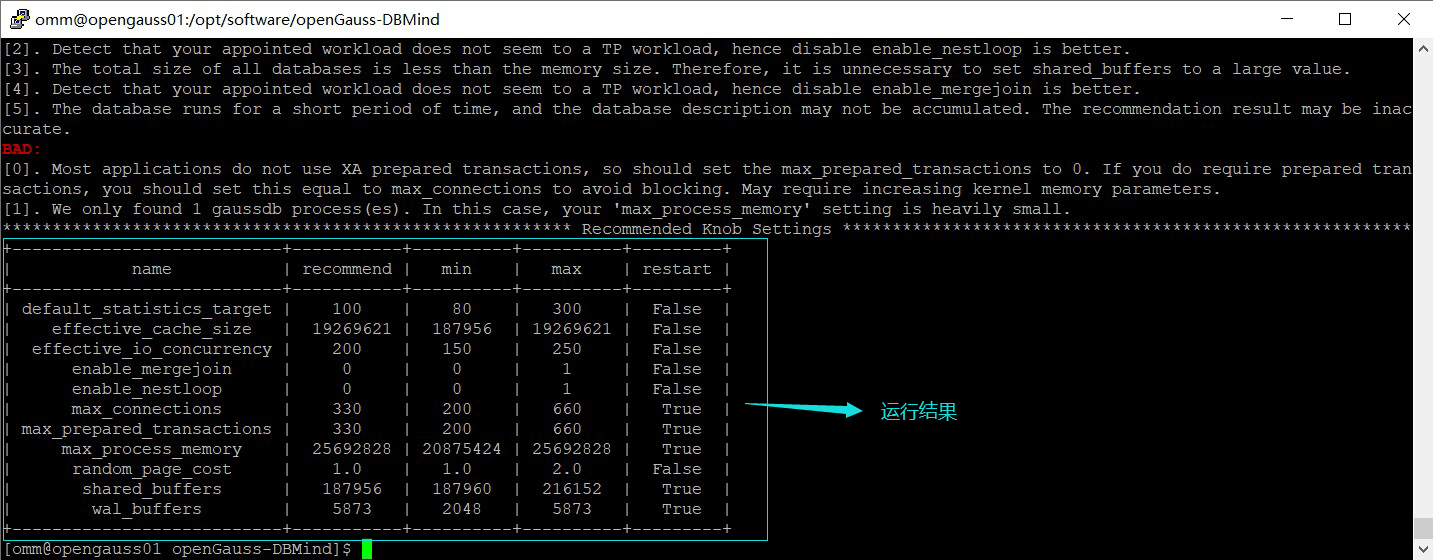
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

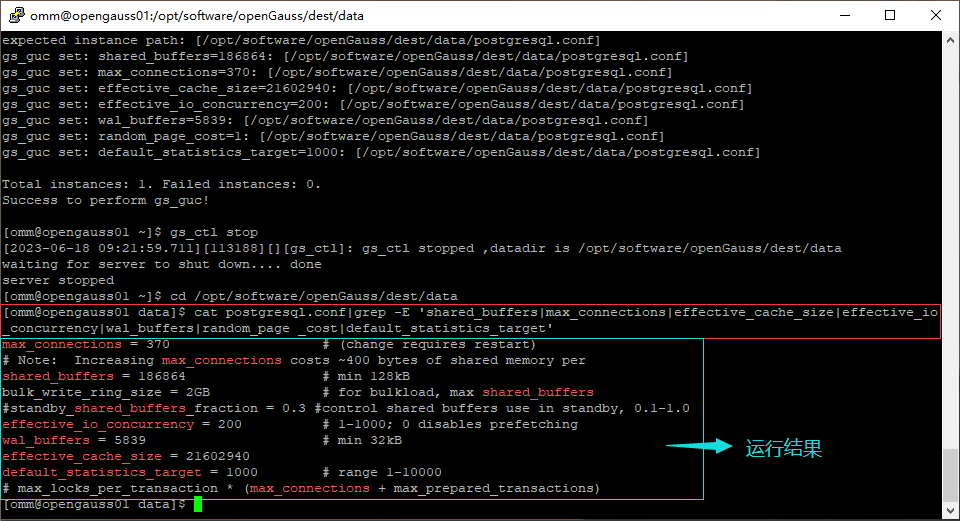
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

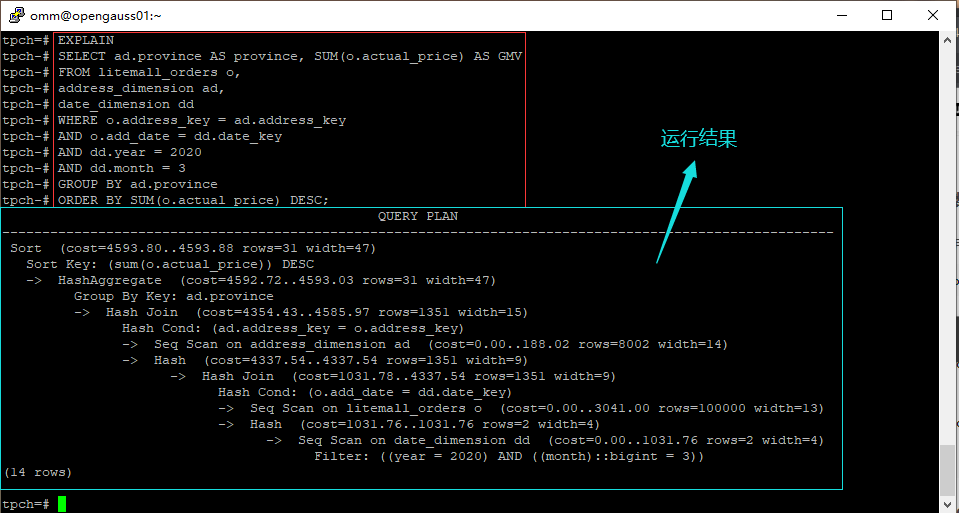
AND o.add\_date = dd.date\_key

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

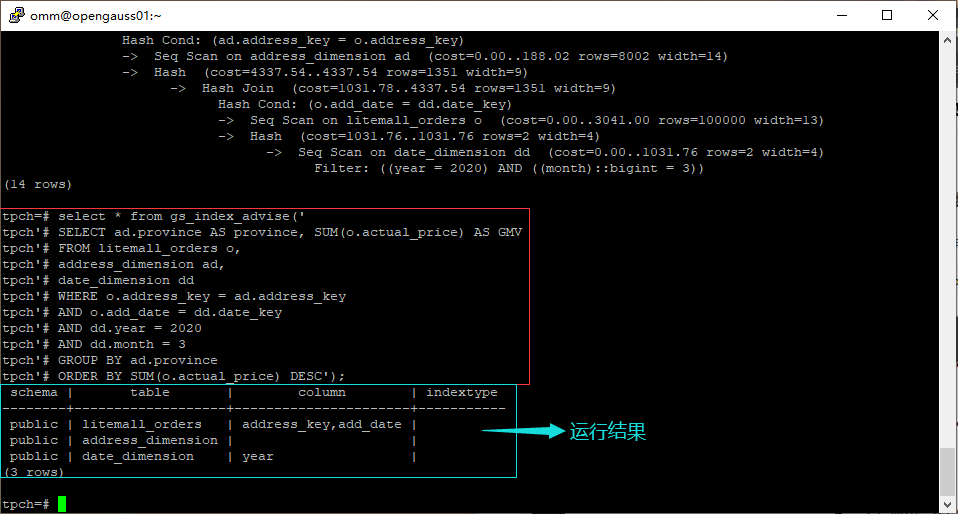
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

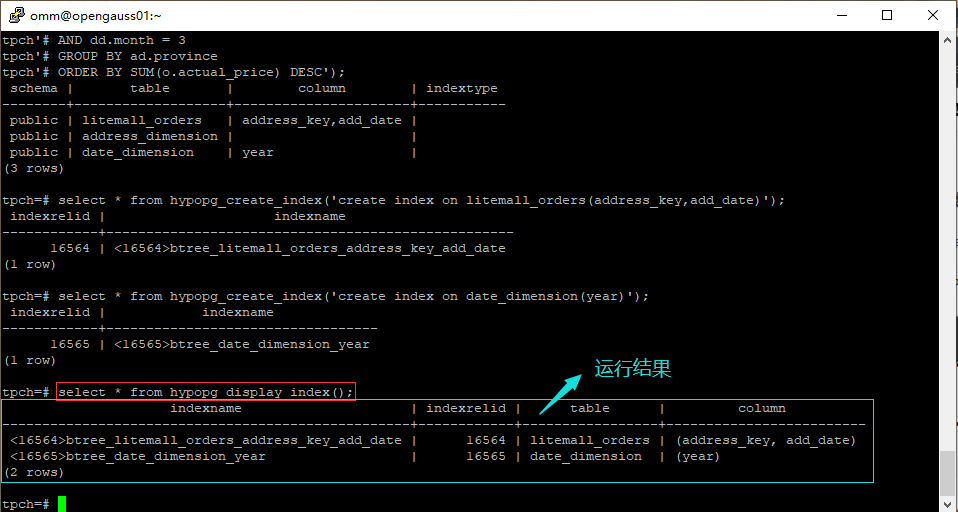
GROUP BY ad.province

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

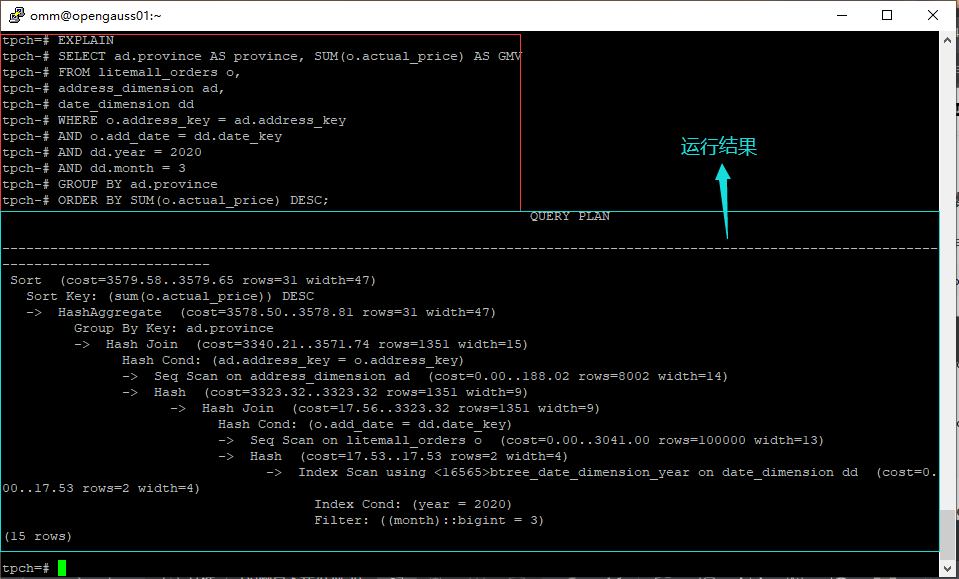
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

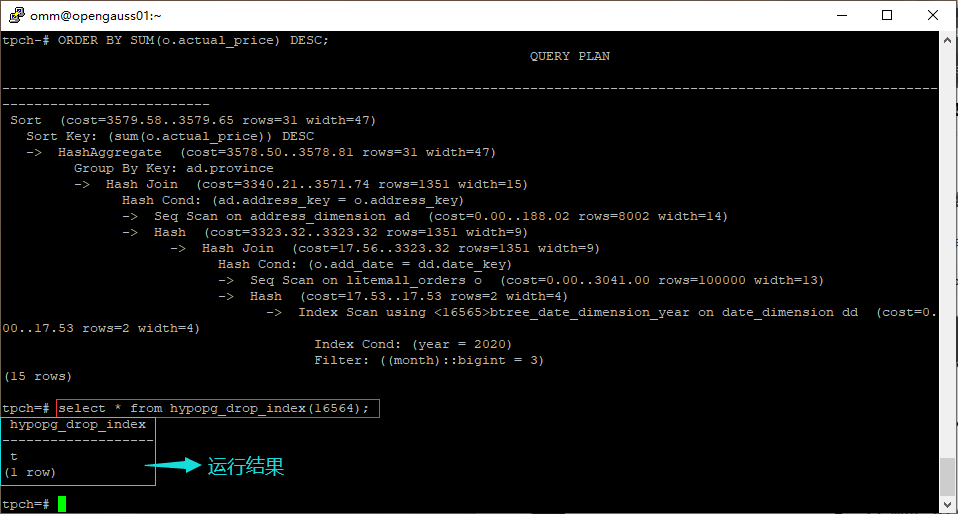
GROUP BY ad.province

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



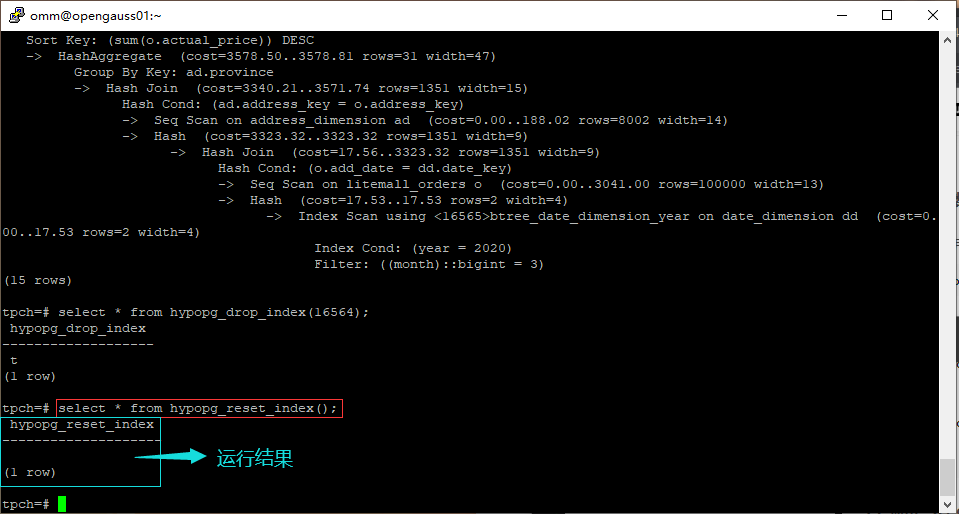
6. 删除某一个索引虚拟列，将执行结果截图。

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



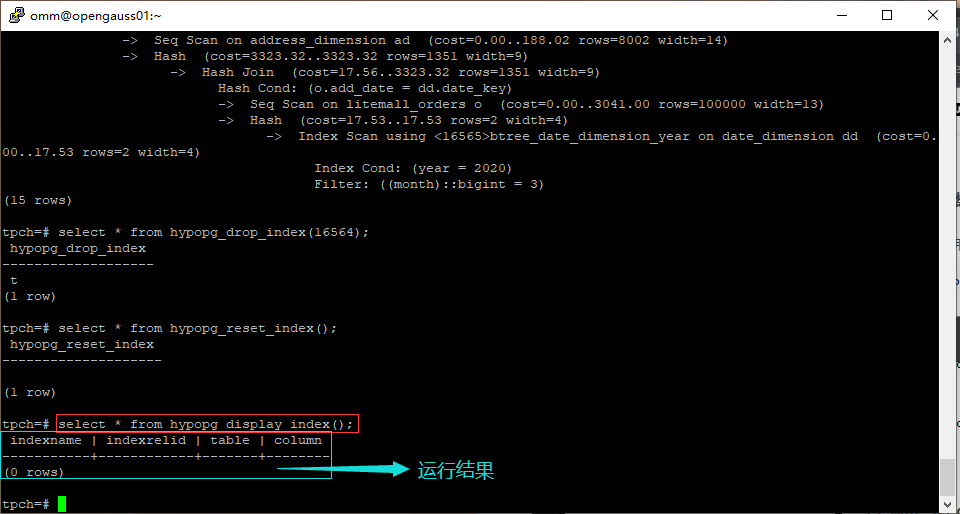
7. 删除所有索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

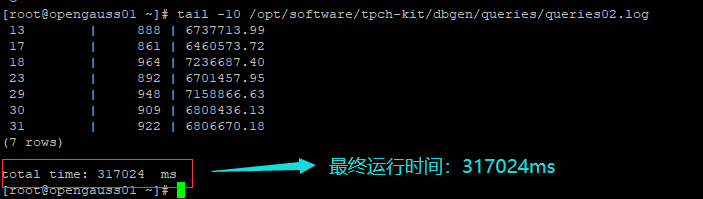
select \* from hypopg\_display\_index();



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：

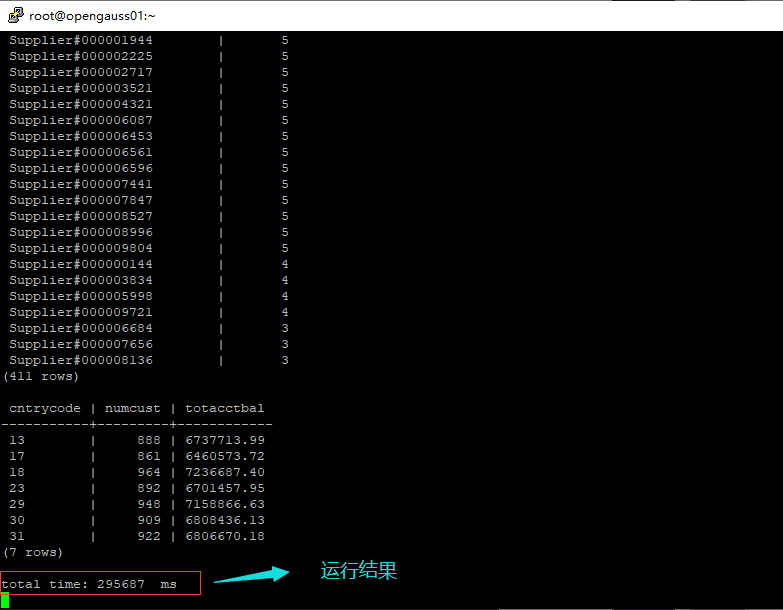
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log



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？

X-Tuner给出的参数优化表中每个参数的含义：

1. workload\_type: 工作负载类型
2. dirty\_background\_bytes: 后台脏数据处理的字节数
3. current\_locks\_count: 当前锁定的数量
4. current\_prepared\_xacts\_count: 当前准备事务的数量
5. rollback\_commit\_ratio: 回滚事务和提交事务的比例
6. average\_connection\_age: 平均连接使用时间
7. checkpoint\_proactive\_triggering\_ratio: 主动触发检查点的比例
8. fetched\_returned\_ratio: 获取和返回数据的比例
9. uptime: 系统运行时间
10. cache\_hit\_rate: 缓存命中率
11. os\_cpu\_count: 操作系统的CPU核心数
12. current\_connections: 当前的活动连接数
13. checkpoint\_avg\_sync\_time: 检查点平均同步时间
14. search\_modify\_ratio: 搜索和修改操作的比率
15. max\_processes: 最大进程数
16. track\_activity\_size: 追踪活动信息的大小
17. all\_database\_size: 所有数据库的总大小
18. temp\_file\_size: 临时文件的大小
19. current\_free\_mem: 当前空闲内存
20. shared\_buffer\_heap\_hit\_rate: 共享缓冲区堆表的命中率
21. used\_mem: 当前使用的内存
22. os\_mem\_total: 操作系统的总内存大小
23. checkpoint\_dirty\_writing\_time\_window: 检查点脏数据写入时间窗口
24. ap\_index: AP指数
25. shared\_buffer\_toast\_hit\_rate: 共享缓冲区TOAST表的命中率
26. read\_tup\_speed: 读取元组的速度
27. block\_size: 块大小
28. read\_write\_ratio: 读写比例
29. shared\_buffer\_tidx\_hit\_rate: 共享缓冲区索引表的命中率
30. shared\_buffer\_idx\_hit\_rate: 共享缓冲区堆表的命中率
31. write\_tup\_speed: 写入元组的速度
32. enable\_autovacuum: 自动清理功能是否启用
33. is\_64bit: 系统是否为64位
34. is\_hdd: 存储介质是否为HDD
35. load\_average: 系统负载平均值

X-Tuner的AI库通过分析数据库的性能指标、负载情况、配置和系统资源等多个方面的数据，使用机器学习和优化算法进行分析和计算，得出了上述优化参数的结果。通过收集数据库的各种指标数据，例如查询频率、缓存命中率、连接数、内存使用等，通过对这些数据进行分析和建模，预测不同参数对数据库性能的影响，并找出最优的参数设置方案，以达到优化数据库性能的目的。同时，X-Tuner也可能考虑了系统的配置和资源情况，例如CPU核心数、内存大小等，结合数据库的负载情况和工作负载类型，综合考虑了多个因素来确定最佳的参数配置。

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时有以下好处：

1. 提高查询性能：索引可以加速数据的查找和匹配过程。通过使用索引，数据库系统可以更快地定位到需要查询的数据，而不必扫描整个数据表。这样可以大大提高查询的速度和效率；
2. 减少IO操作：当数据库中的数据量较大时，使用索引可以减少磁盘IO的次数。通过使用索引，数据库可以直接定位到需要的数据页，而不必读取整个表或数据块。这样可以减少IO操作，提高系统的整体性能；
3. 优化排序和连接操作：索引不仅可以加速查询操作，还可以优化排序和连接操作。通过使用索引，数据库可以在执行排序和连接操作时更快地查找和匹配需要的数据，从而提高这些操作的性能。

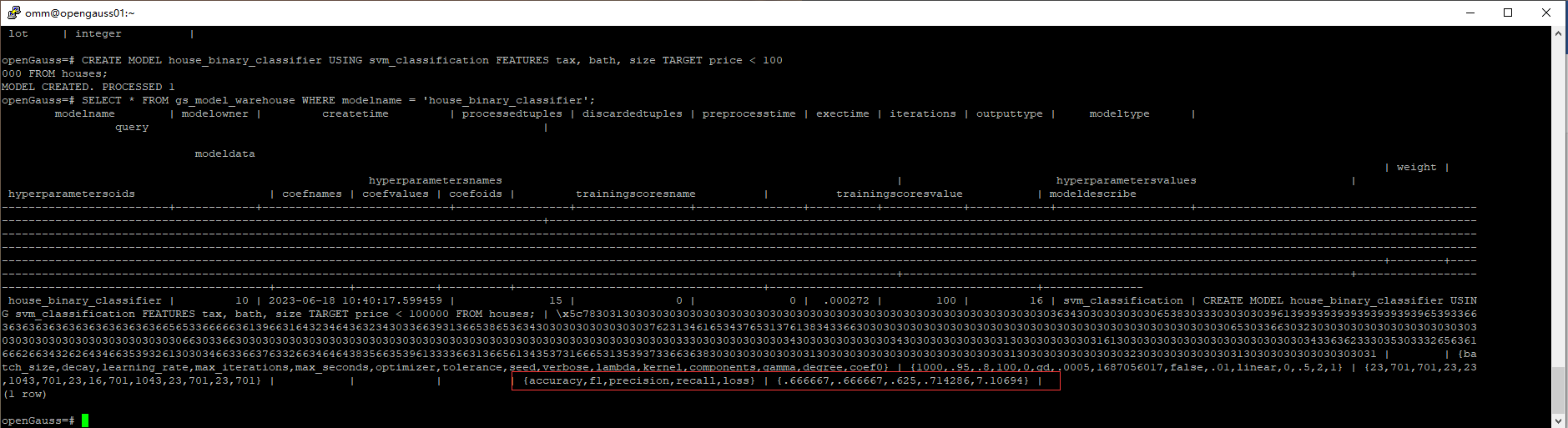
除了使用索引和参数之外，还有以下方面可以对数据库进行优化：

1. 范式设计和冗余数据的消除：通过合理的范式设计和消除冗余数据，可以减小数据表的大小，降低查询和更新的成本。
2. 数据库分区和分片：将大型数据库分区或分片存储可以提高数据的并行处理能力，从而提高系统的吞吐量和性能。
3. 缓存和缓存策略：通过使用缓存技术，如数据库查询缓存、应用程序级缓存等
4. 查询优化：数据库查询的性能可以通过优化查询语句、选择合适的查询方式（如索引选择、表连接方式）、调整查询顺序等进行提升。
5. 硬件升级和优化：通过升级硬件设备（如增加内存、加速存储介质、提升网络带宽等）可以提高数据库系统的整体性能。
6. 并发控制和事务管理：合理地使用并发控制机制和事务管理可以提高数据库系统的并发性能和数据一致性，避免锁冲突和死锁情况的发生。

# 关卡四、openGauss的DB4AI特性应用

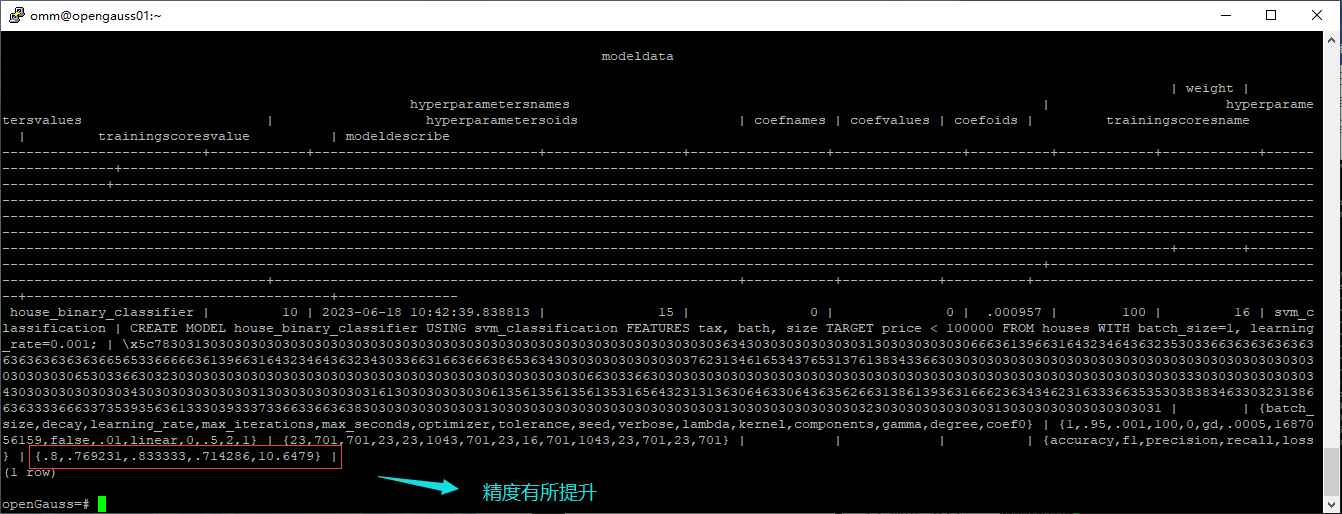
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



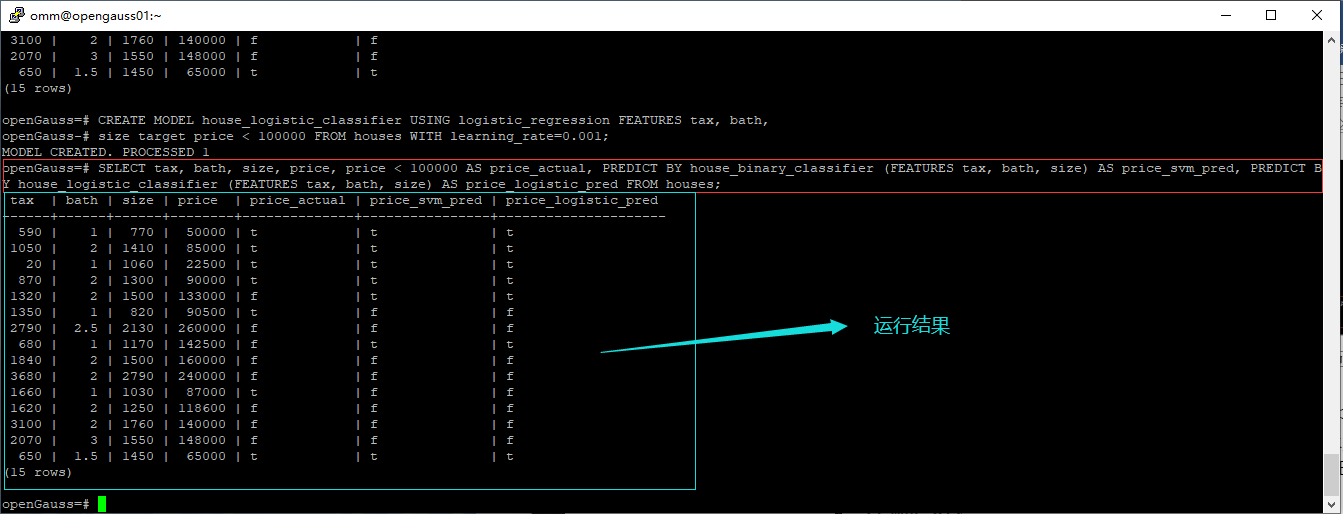
任务二：观察新模型的信息，将执行结果截图。

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常见的两种模型类型，它们在处理问题和建模方法上有一些区别：

1. 目标类型：分类模型的目标变量是离散的，代表不同的类别或标签。例如，预测一个图片是猫还是狗、判断邮件是垃圾邮件还是正常邮件等。回归模型的目标变量是连续的，可以是任意数值。例如，预测房屋的销售价格、预测用户的购买金额等。
2. 模型输出：分类模型的输出是对样本进行分类的结果，通常是一个离散的类别。可以是二分类（两个类别）或多分类（多个类别）问题。回归模型的输出是对样本进行预测的数值，可以是实数或整数。
3. 模型评估：分类模型的性能评估通常使用准确率、召回率、精确度等指标来衡量预测结果与真实标签的匹配程度。回归模型的性能评估通常使用均方误差（MSE）、平均绝对误差（MAE）、决定系数（R²）等指标来衡量预测结果与真实数值的接近程度。
4. 模型选择和建模方法：在分类模型中，常见的算法包括逻辑回归、决策树、支持向量机（SVM）、随机森林等。在回归模型中，常见的算法包括线性回归、岭回归、支持向量回归（SVR）、随机森林回归等。

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，常用于进行模式分类和回归分析。它属于监督学习算法的一种，通过建立一个能够将不同类别样本分隔开的超平面来进行分类或回归。

SVM算法的基本思想是找到一个最优的超平面，该超平面能够最大化不同类别样本之间的间隔（margin）。超平面是一个n-1维的子空间，n是特征维度。对于二分类问题，SVM的目标是找到一个能够将正负样本完全分开的最优超平面。

在SVM算法中，选择了一部分训练样本作为支持向量（support vectors），它们是离超平面最近的一些样本点，用于定义超平面的位置和方向。SVM通过使用核函数将高维特征映射到更高维的特征空间中，从而可以更容易地找到一个能够将样本分开的超平面。

SVM算法的优点包括：

1. 在高维特征空间中充分考虑样本分布，可以处理非线性问题；
2. 通过最大化间隔，具有较好的鲁棒性和泛化能力；
3. 对于小样本和大特征维度的情况，也能较好地进行分类；
4. 通过核技巧，可以处理非线性分类问题。

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？

在分类问题中，常用的评价指标包括：

1. 准确率（Accuracy）：准确率是一个简单而直观的评价指标，表示分类器预测正确的样本数占总样本数的比例。公式为：准确率 = (TP + TN) / (TP + TN + FP + FN)，其中TP表示真正例（True Positive），TN表示真负例（True Negative），FP表示假正例（False Positive），FN表示假负例（False Negative）。准确率越高，表示分类器预测正确的能力越强。
2. 精确度（Precision）：精确度衡量了分类器预测为正例的样本中，真正例的比例。公式为：精确度 = TP / (TP + FP)。精确度高表示分类器能够尽可能减少将负例错误预测为正例的情况，对于正例的预测准确性较高。
3. 召回率（Recall）：召回率衡量了分类器将正例正确预测的能力，也称为灵敏度（Sensitivity）或真正例率（True Positive Rate）。公式为：召回率 = TP / (TP + FN)。召回率高表示分类器能够尽可能地找出更多的正例样本。
4. F1值（F1-score）：F1值是精确度和召回率的调和平均，是一种综合评估指标。公式为：F1值 = 2 \* (精确度 \* 召回率) / (精确度 + 召回率)。F1值综合考虑了分类器的精确度和召回率，适用于不平衡类别分布的情况。
5. 特异性（Specificity）：特异性衡量了分类器正确预测负例的能力，即真负例中被正确预测的比例。公式为：特异性 = TN / (TN + FP)。特异性高表示分类器能够减少将正例错误预测为负例的情况。
6. FPR（False Positive Rate）：假正例率衡量了分类器将负例错误预测为正例的能力，即负例中被错误预测为正例的比例。公式为：FPR = FP / (FP + TN)。FPR低表示分类器对负例的预测准确性较高。
7. AUC-ROC（Area Under the Receiver Operating Characteristic Curve）：AUC-ROC是ROC曲线下的面积，用于评估二分类模型的分类性能。ROC曲线是TPR（真正例率）和FPR（假正例率）之间的关系曲线。AUC-ROC的取值范围在0~1之间，值越接近1表示模型性能越好，能有效衡量模型的整体分类能力。

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，常用的评价指标包括：

1. 均方误差（Mean Squared Error，MSE）：均方误差是最常见的回归评价指标之一，表示预测值与真实值之间差异的平方的平均值。公式为：，其中表示真实值，表示预测值，表示样本数量。MSE越小，表示模型的拟合效果越好。
2. 平均绝对误差（Mean Absolute Error，MAE）：平均绝对误差是另一种常用的回归评价指标，表示预测值与真实值之间差异的绝对值的平均值。公式为：。MAE可以更直观地衡量模型预测的平均偏差大小，而不关注偏差的方向。
3. 系数（Coefficient of Determination，）：是衡量回归模型拟合程度的指标，反映了模型对总变差的解释能力。决定系数的取值范围在0和1之间。，其中SSR为回归平方和（模型拟合残差的平方和），SST为总平方和（真实值与真实均值的平方和）。越接近1，表示模型对观测值的拟合较好。
4. 可解释方差（Explained Variance Score）：可解释方差是衡量回归模型解释样本方差的能力。公式为：。可解释方差的取值范围在0和1之间，越接近1表示模型对样本方差的解释能力越强。
5. 平均百分比误差（Mean Percentage Error，MPE）：平均百分比误差是一种相对误差的度量，表示预测值与真实值的百分比误差的平均值。公式为：。MPE可以衡量模型预测结果偏离真实值的平均百分比。
6. 均方根误差（Root Mean Squared Error，RMSE）：均方根误差是均方误差的平方根。。与MSE相比，RMSE可以更直观地表示预测误差的量级。