openGauss AI特性创新实践课



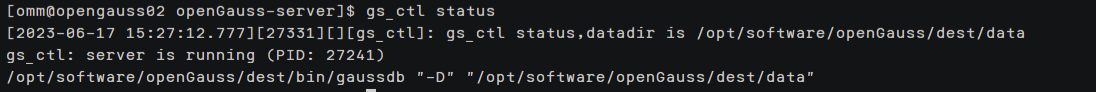
华为技术有限公司

# 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：

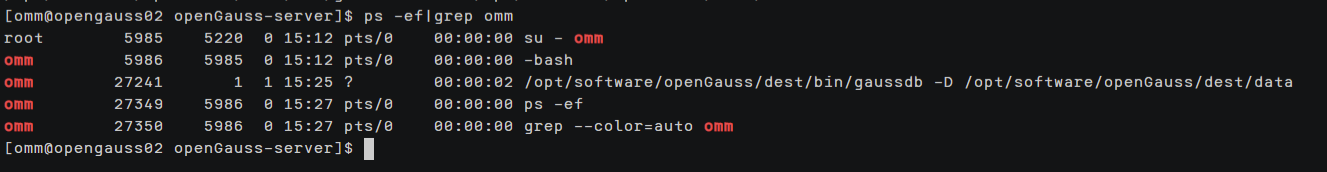
任务一：数据库状态验证

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方式有以下几个主要原因：

定制安装：通过源码编译和安装数据库，您可以进行定制安装，选择需要的功能和配置选项。这样可以根据您的具体需求和环境进行优化和配置，以实现最佳性能和适应性。

最新版本：源码编译可以让您获得数据库的最新版本。在发行版发布之前，数据库供应商通常会提供源代码供用户自行编译和安装。这样可以及时获取最新的功能、性能优化和错误修复。

平台适配性：通过源码编译，您可以确保数据库在您的操作系统和硬件平台上正常运行。不同的操作系统和硬件有不同的特性和要求，源码编译可以为特定平台进行适配，以确保数据库的稳定性和性能。

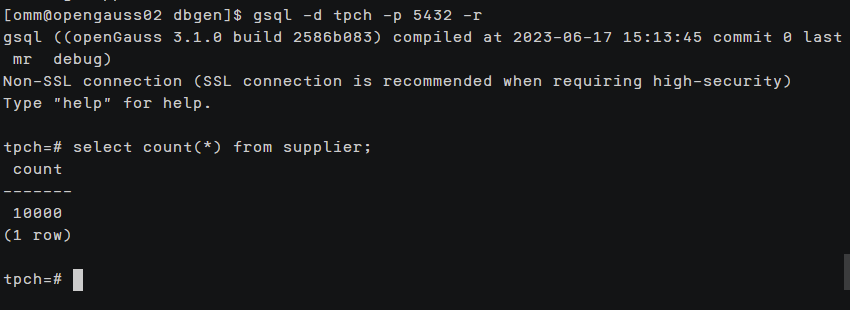
安全性和审查：通过查看和审查数据库的源代码，您可以验证其安全性和可靠性。这对于敏感数据和安全要求较高的环境特别重要。通过自行编译和审查源代码，您可以确保数据库没有包含恶意代码或漏洞，并且可以对其进行自定义安全增强。

# 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：

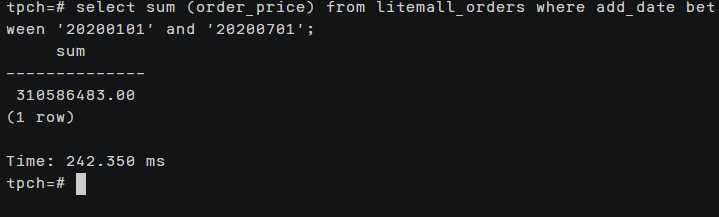
select count(\*) from supplier;;



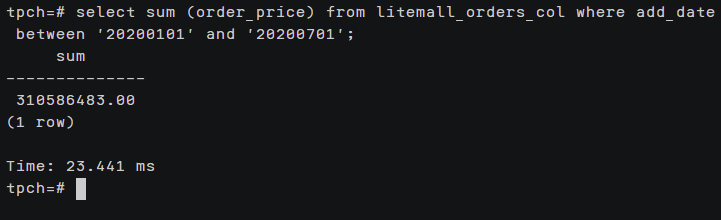
任务二：行存表与列存表执行效率对比

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';

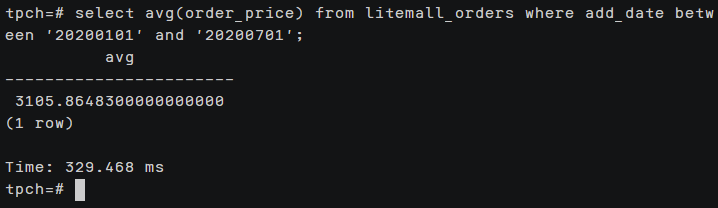


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

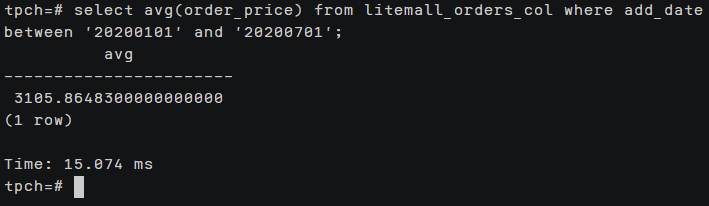


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

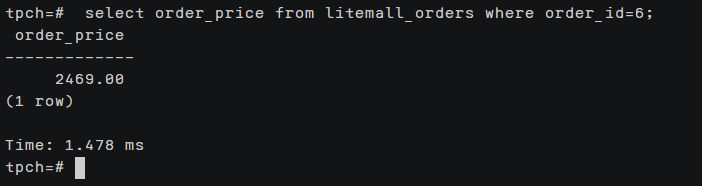


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

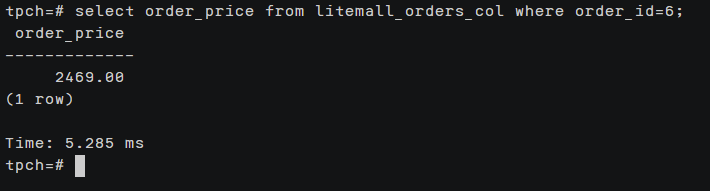


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

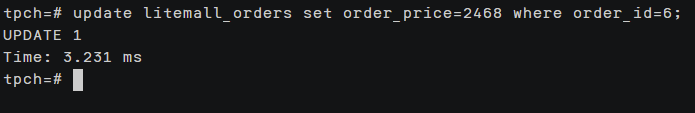


select order\_price from litemall\_orders\_col where order\_id=6;

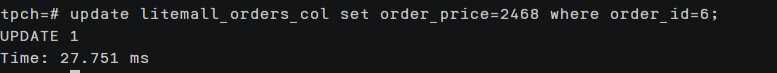


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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

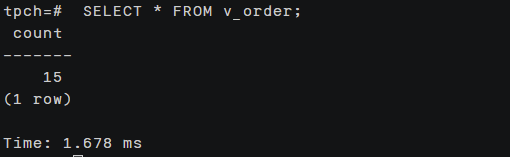
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



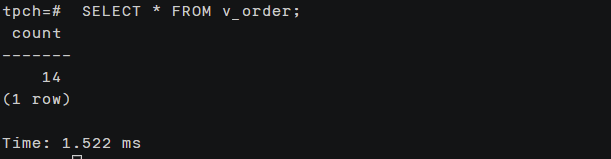
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



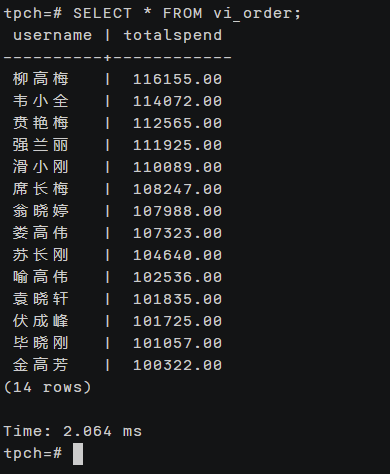
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



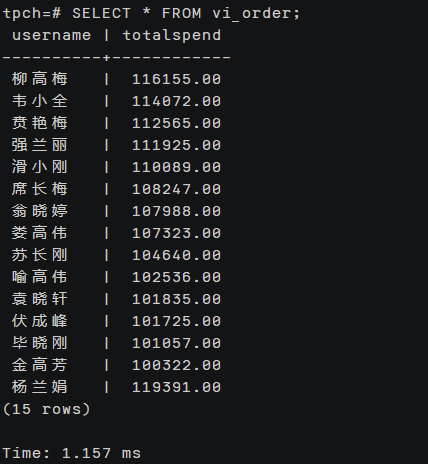
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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是由于它们在数据存储和读取方式上的差异。

行存表（Row-based table）以行为单位存储数据，每一行包含了多个列的数据。当执行涉及整行数据的操作时，行存表效率较高。例如，当需要获取完整的行数据，进行增删改查操作时，行存表可以直接读取或写入整个行，效率较高。

列存表（Columnar-based table）以列为单位存储数据，将同一列的数据连续存放。当执行涉及某些列的操作时，列存表效率较高。例如，当需要进行大量聚合计算、分析查询或只需要某些列的数据时，列存表可以仅读取所需的列数据，避免读取不相关的数据，从而提高查询性能。

具体而言，在以下情况下，行存表效率更高：

需要获取完整的行数据。

需要频繁进行增删改操作，特别是涉及到多个列的操作。

数据访问模式是基于事务的 OLTP（联机事务处理）工作负载，需要频繁地进行单行查询和更新操作。

而在以下情况下，列存表效率更高：

需要进行大规模的聚合计算，如SUM、COUNT等。

需要进行复杂的分析查询，涉及多个列的过滤、排序和分组操作。

数据访问模式是基于分析的 OLAP（联机分析处理）工作负载，需要进行大规模数据查询和分析。

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）和增量物化视图（Incremental Materialized View）是在数据仓库和数据库中用于提高查询性能的技术。它们之间的主要差别在于刷新方式和维护成本。

全量物化视图：全量物化视图是通过将查询结果完整地计算和存储在物化视图中，以提供快速的查询响应。全量物化视图在刷新时重新计算和加载完整的数据集，通常需要执行较长时间的操作，因为它们需要处理整个数据集。因此，全量物化视图的刷新过程可能对系统资源有较高的需求，并且在数据更新频繁的情况下，刷新时间可能较长。

增量物化视图：增量物化视图是通过跟踪源数据的变化，仅更新物化视图中受影响的数据部分，以实现更快速的刷新。增量物化视图使用日志、触发器或其他机制来捕获源数据的变化，并根据变化的数据更新物化视图。相比于全量物化视图，增量物化视图的刷新操作通常更快，因为它只需要处理受影响的数据部分。这种方式适用于数据更新频繁的场景，可以减少刷新时间和系统资源的开销。

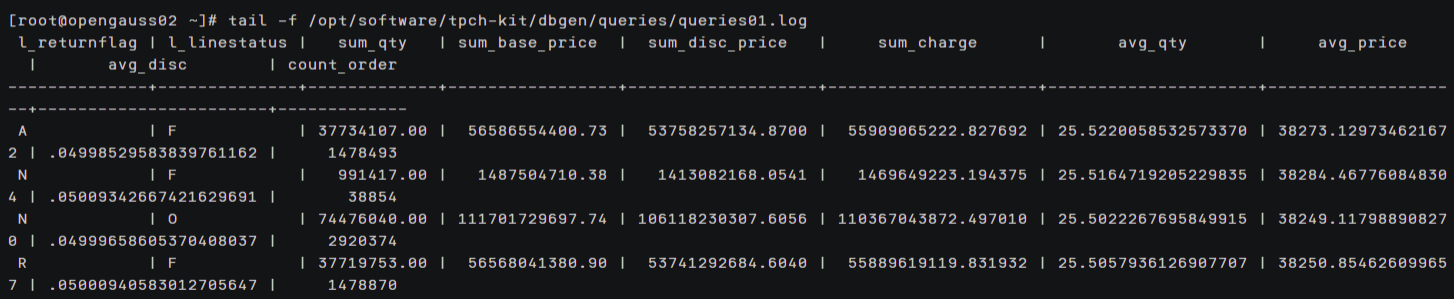
总结来说，全量物化视图通过完整计算和存储查询结果，适用于数据更新较少或查询需求不频繁的情况。增量物化视图则通过跟踪源数据的变化，只更新受影响的数据部分，适用于数据更新频繁或对实时性要求较高的情况。选择合适的物化视图类型需要综合考虑数据更新频率、查询需求和系统资源等因素。

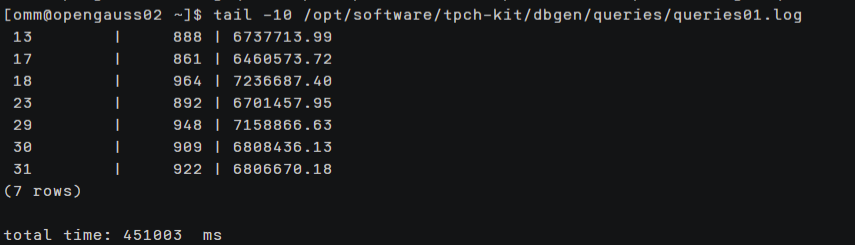
# 关卡三、openGauss的AI4DB特性应用

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：

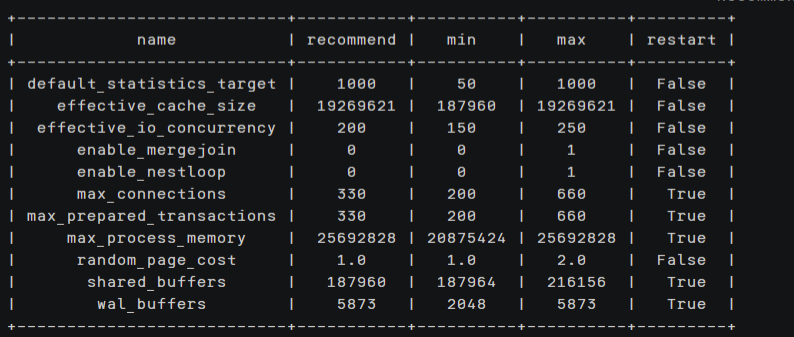
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log





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

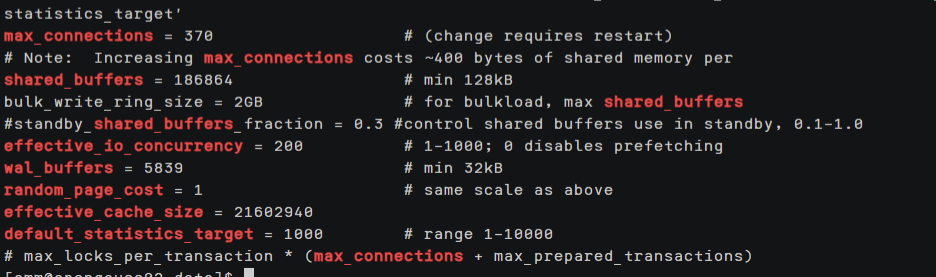
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

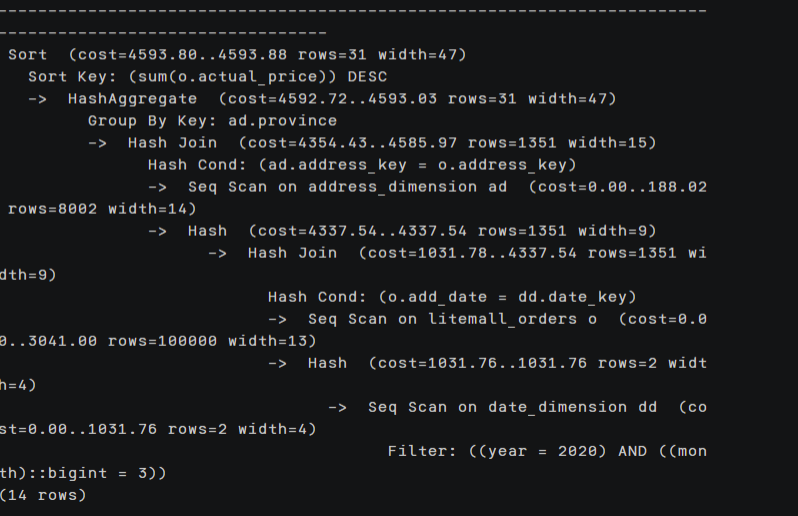
AND o.add\_date = dd.date\_key

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

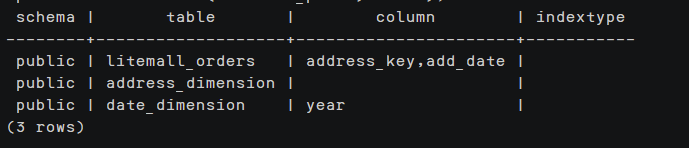
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

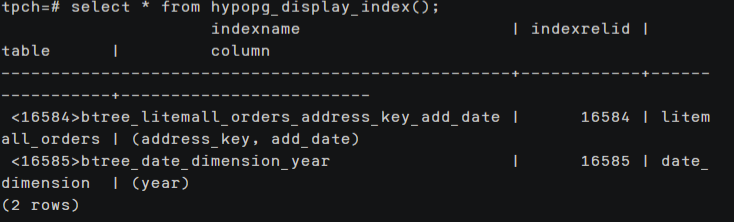
GROUP BY ad.province

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



1. 查看创建的虚拟索引列，将执行结果截图。

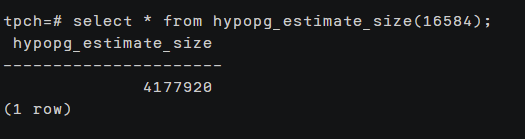
select \* from hypopg\_display\_index();

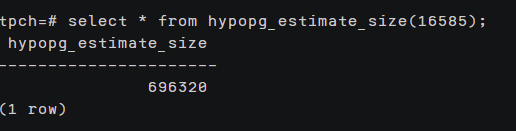


4. 获取索引虚拟列大小结果（单位为：字节），将执行结果截图。

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

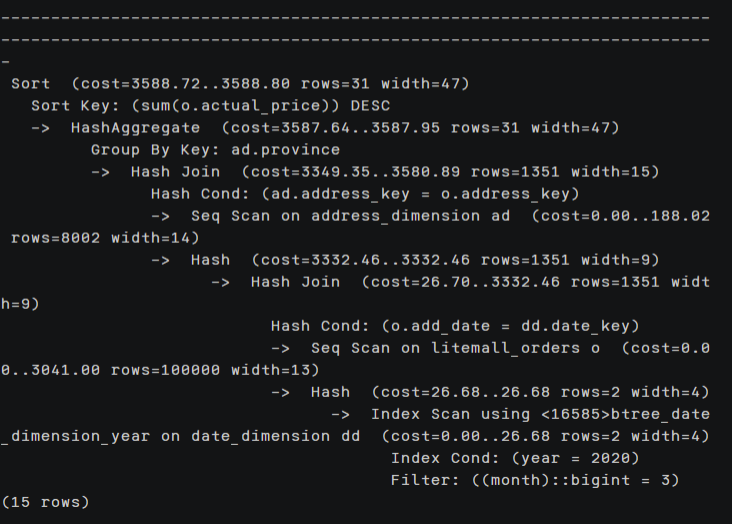
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

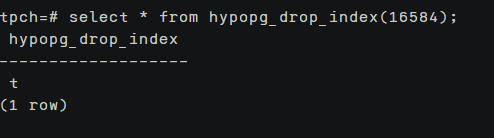
GROUP BY ad.province

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



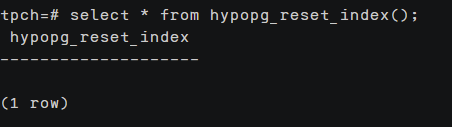
6. 删除某一个索引虚拟列，将执行结果截图。

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



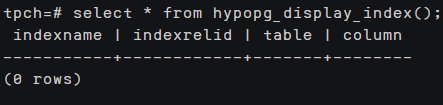
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

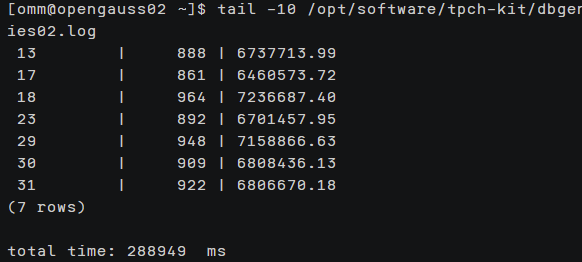
select \* from hypopg\_display\_index();



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？

根据X-Tuner进行的参数优化通常涉及以下几个方面的参数：

查询优化器参数：查询优化器是数据库系统中负责生成查询执行计划的组件。通过调整查询优化器相关的参数，可以影响查询执行计划的选择和性能。这些参数包括连接算法、连接顺序、索引选择、成本估算等。优化这些参数可以改善查询的执行效率，减少查询的响应时间。

缓存管理参数：数据库系统通常具有各种缓存，如数据缓存、查询缓存、日志缓存等。通过调整缓存管理参数，可以优化缓存的大小、替换策略、缓存清理频率等，以提高数据访问的性能。合理的缓存管理参数可以减少磁盘IO操作，加快数据读取速度。

并发控制参数：并发控制是数据库系统中用于处理并发事务的机制。通过调整并发控制参数，可以影响并发事务的并发度、锁的粒度、死锁检测机制等。优化这些参数可以提高并发事务的吞吐量和性能，减少锁冲突和死锁的发生。

日志管理参数：数据库系统通常有日志机制用于记录数据修改操作，以支持事务的持久性和恢复能力。通过调整日志管理参数，可以控制日志的写入策略、日志文件大小、日志刷新频率等。优化这些参数可以提高事务的提交性能和系统的恢复性能。

这些参数的优化是为了提高数据库系统的性能和效率。通过调整这些参数，可以根据具体的应用场景和硬件配置，使数据库系统能够更好地适应工作负载，提供更快速、高效的数据处理和查询响应。不同的参数优化策略可能适用于不同的应用场景，因此根据具体情况进行参数优化是重要的性能优化手段之一。

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？

使用索引可以为执行SQL语句带来以下好处：

提高查询性能：索引可以加速查询操作，减少需要扫描的数据量。通过使用索引，数据库可以快速定位到满足查询条件的数据，提高查询效率，减少响应时间。

减少磁盘IO操作：索引可以减少磁盘IO操作，避免全表扫描。当索引覆盖了查询所需的列时，数据库可以直接从索引中获取所需的数据，而无需访问主要存储区域，从而减少磁盘IO的开销。

支持排序和分组：索引可以为排序和分组操作提供支持。当SQL语句包含ORDER BY、GROUP BY或DISTINCT等操作时，索引可以提供有序的数据，减少排序和分组的开销。

除了使用索引和参数，还有其他方面可以对数据库进行优化，包括但不限于以下几点：

数据库模式设计：合理的数据库模式设计可以提高数据访问效率。通过规范的表结构、关系定义、数据类型选择等，可以减少数据冗余、提高数据一致性，并优化查询操作。

数据库范式化与反范式化：根据具体需求，选择合适的数据库范式化级别和反范式化技术。范式化可以提高数据的一致性和更新效率，而反范式化可以减少查询的复杂性和提高查询性能。

优化SQL语句：编写高效的SQL语句可以显著提高查询性能。合理使用查询条件、避免使用不必要的JOIN操作、避免全表扫描等，都可以对SQL语句进行优化。

数据库连接池配置：对于应用程序使用连接池连接数据库的情况，合理配置连接池的参数可以提高数据库连接的管理和利用率，减少连接的创建和销毁开销。

内存管理：合理配置数据库的内存参数，包括缓存大小、排序区大小等，可以提高数据的内存访问速度，减少磁盘IO操作。

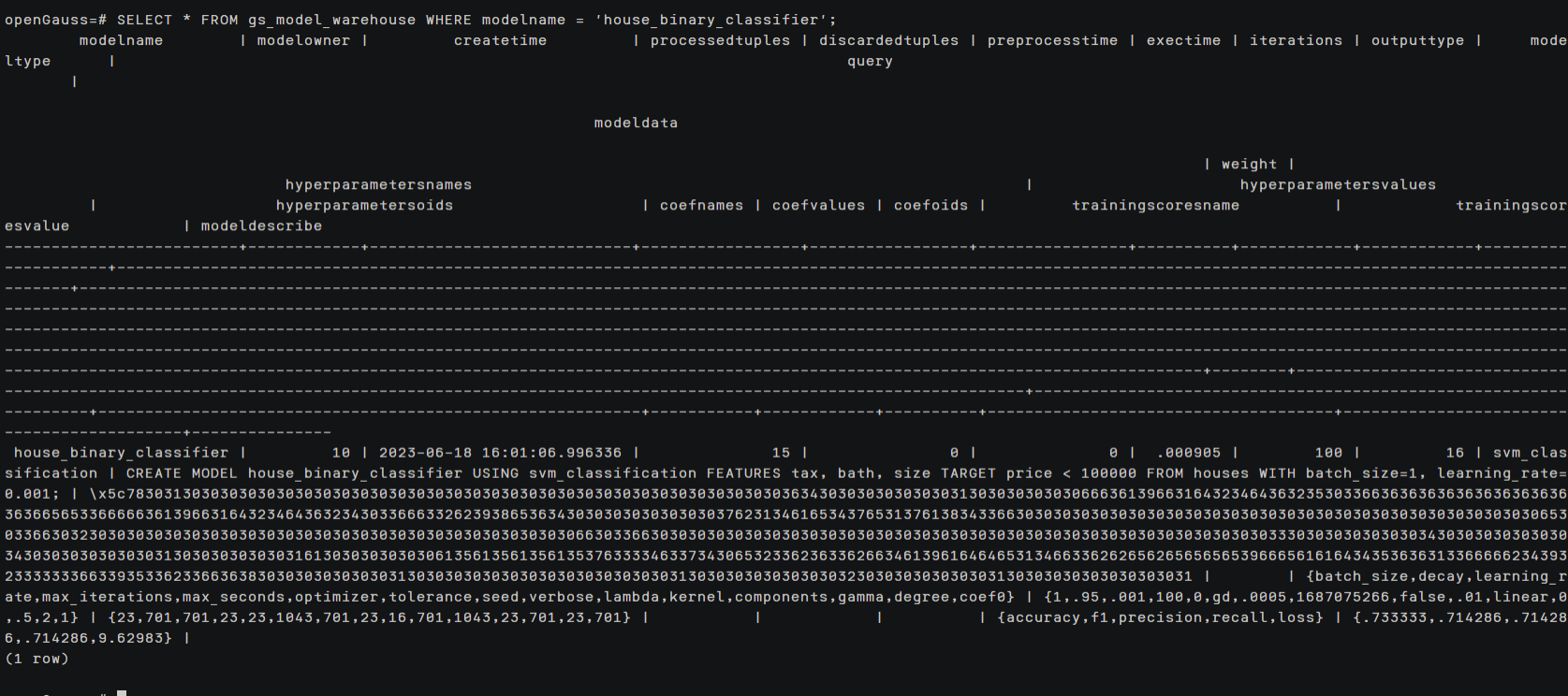
查询缓存：对于频繁执行相同查询的场景，可以启用查询缓存来缓存查询结果，减少重复查询的开销。

定期维护：定期执行数据库维护操作，如索引重建、统计信息收集、碎片整理等，可以保持数据库的性能和稳定性。

# 关卡四、openGauss的DB4AI特性应用

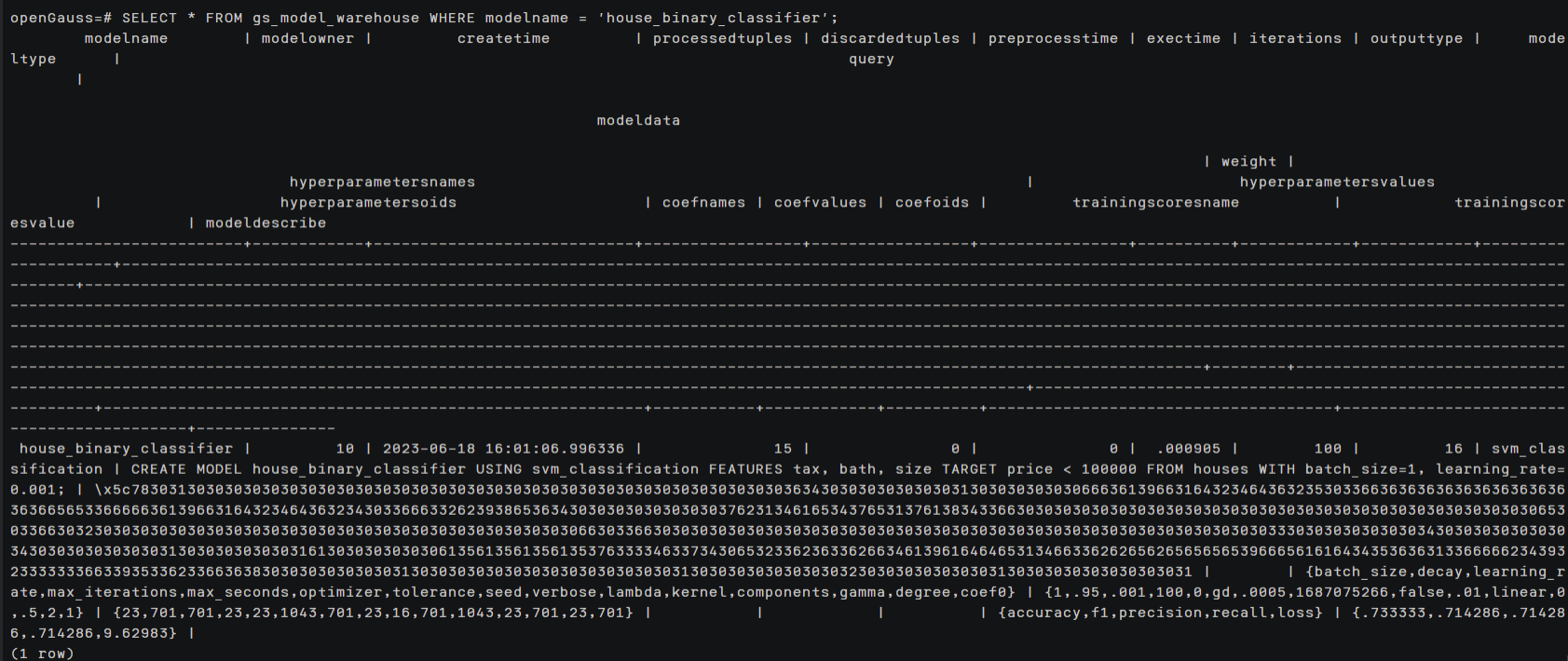
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



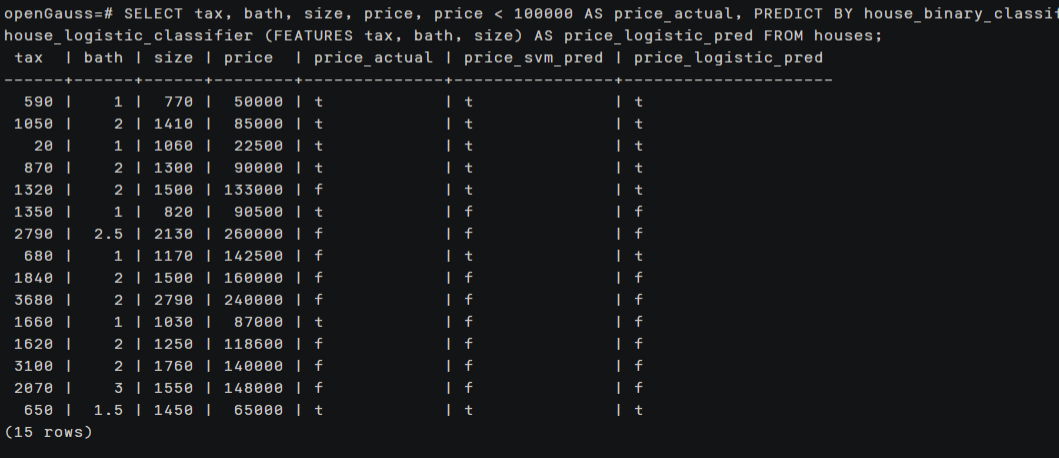
任务二：观察新模型的信息，将执行结果截图。

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？

分类模型是用于解决分类问题的模型。分类问题是将样本分为不同的离散类别或标签。分类模型的输出通常是一个离散的类别或标签，表示样本所属的类别。分类模型可以根据输入样本的特征进行学习，并根据学习的模型对新样本进行分类。常见的分类模型包括逻辑回归、决策树、支持向量机（SVM）、随机森林和神经网络等。

回归模型是用于解决回归问题的模型。回归问题是预测一个连续的数值输出。回归模型的输出通常是一个连续的数值，可以是实数或整数。回归模型通过学习输入样本的特征与输出之间的关系，建立一个函数模型来预测新样本的输出。常见的回归模型包括线性回归、岭回归、Lasso回归、多项式回归和支持向量回归等。

总结来说，分类模型用于解决离散分类问题，其输出是离散的类别或标签；而回归模型用于解决连续回归问题，其输出是一个连续的数值。选择分类模型还是回归模型取决于问题的性质和预测目标的要求。

实践思考题2：什么是SVM算法？

SVM（支持向量机，Support Vector Machine）是一种常用的监督学习算法，用于二分类和多分类问题。它可以用于解决线性可分和线性不可分的问题，并且在处理高维数据和非线性数据时表现良好。

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进行分类。超平面是一个(d-1)维的子空间，将d维特征空间划分为两个部分，每个部分对应一个类别。最优超平面是具有最大间隔（margin）的超平面，可以最大程度地将两个类别的样本分开。间隔是指离超平面最近的样本点与超平面之间的距离。SVM的目标是找到一个最优超平面，使得间隔最大化。

当样本无法被线性分割时，SVM采用核函数（Kernel Function）的方式将样本从原始特征空间映射到一个高维特征空间，使得样本在高维空间中线性可分。常用的核函数包括线性核函数、多项式核函数和径向基函数（RBF）核函数等。

SVM的训练过程是一个凸优化问题，通过最大化间隔来确定最优超平面。其求解过程可以转化为一个二次规划（Quadratic Programming）问题，并通过拉格朗日乘子法进行求解。

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通常包括以下几个指标：

准确率（Accuracy）：准确率是最常用的分类评价指标，表示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。然而，准确率在不平衡数据集中可能存在问题，因为它无法反映出不同类别之间的分类效果差异。

精确率（Precision）：精确率表示分类器预测为正例的样本中，真正为正例的比例。它衡量了分类器在预测为正例时的准确性。高精确率表示分类器的正例预测准确率较高。

召回率（Recall）：召回率表示真实为正例的样本中，被分类器正确预测为正例的比例。它衡量了分类器对正例的识别能力。高召回率表示分类器对正例的覆盖能力较好。

F1值（F1-Score）：F1值是精确率和召回率的调和平均值，用于综合评估分类器的性能。F1值越高，分类器的综合性能越好。它可以解决准确率和召回率之间的平衡问题。

特异度（Specificity）：特异度表示分类器在预测为负例时的准确性。它衡量了分类器对负例的识别能力。高特异度表示分类器对负例的区分能力较好。

AUC-ROC：AUC（Area Under Curve）指的是ROC（Receiver Operating Characteristic）曲线下的面积，用于评估二分类器的性能。AUC-ROC值越接近1，分类器的性能越好。ROC曲线反映了分类器在不同阈值下的召回率和假阳率之间的关系。

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以下几个：

平均绝对误差（MAE，Mean Absolute Error）：MAE是预测值与真实值之间绝对差值的平均值。

均方误差（MSE，Mean Squared Error）：MSE是预测值与真实值之间差值的平方的平均值。MSE相较于MAE更加敏感，因为它将误差的平方进行了放大。与MAE相比，MSE更注重较大误差的惩罚，数值越小表示模型的拟合程度越好。

均方根误差（RMSE，Root Mean Squared Error）：RMSE是MSE的平方根，它可以将误差的单位恢复到与原始数据相同的量级。RMSE与MSE具有相同的趋势和相似的解释。

决定系数（R2 Score）：决定系数是预测值与真实值之间的方差占比，表示模型对数据变异的解释能力。取值范围为0到1，越接近1表示模型的拟合程度越好，0表示模型不能解释目标变量的任何变异。

解释方差分数（Explained Variance Score）：解释方差分数衡量了模型解释目标变量方差的能力。取值范围为0到1，越接近1表示模型的解释能力越好。