openGauss AI特性创新实践课



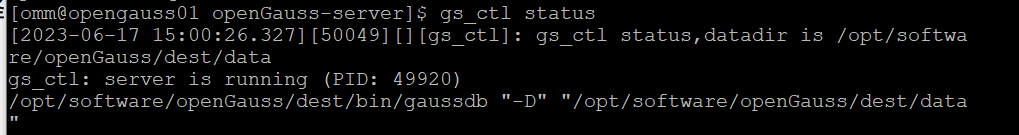
华为技术有限公司

# 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：

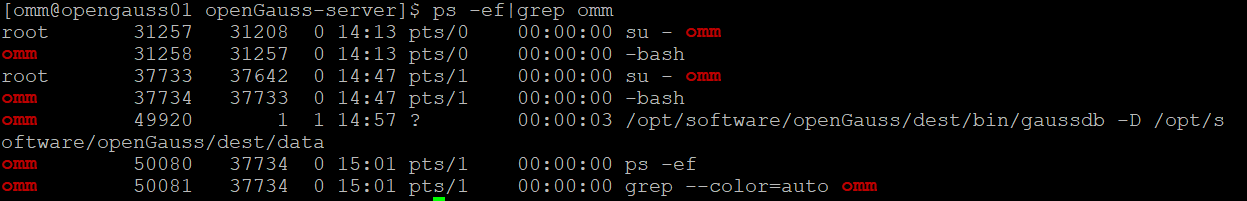
任务一：数据库状态验证

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：回答如下：

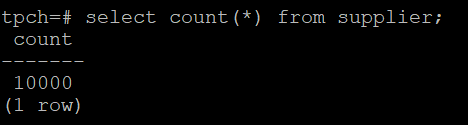
1. 定制化需求：源码编译允许你根据特定的需求进行定制。你可以选择启用或禁用某些功能、优化编译选项以获得更好的性能，或者根据你的硬件和操作系统环境进行适配。这种灵活性使得你可以根据具体需求对数据库进行配置和优化，以达到最佳的性能和功能。
2. 最新版本和修复补丁：源码编译可以确保你使用的是最新版本的数据库软件。新版本通常包含性能改进、新功能和安全修复。此外，当发现数据库软件中的漏洞或错误时，开发者会发布修复补丁。通过源码编译安装，你可以及时获取这些更新和修复，从而提高数据库的安全性和稳定性。
3. 平台适配性：通过源码编译，你可以确保数据库软件与你的操作系统和硬件平台完全兼容。不同的操作系统和硬件架构可能有不同的二进制格式和依赖库，通过源码编译可以根据你的环境进行定制编译，确保数据库能够正常运行。
4. 深入理解和研究：对于那些对数据库内部工作原理感兴趣的人来说，通过源码编译可以提供更深入的学习和研究机会。你可以阅读源代码、理解数据库的实现细节，并对其进行修改和实验，以满足特定的需求或进行性能优化。

# 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：

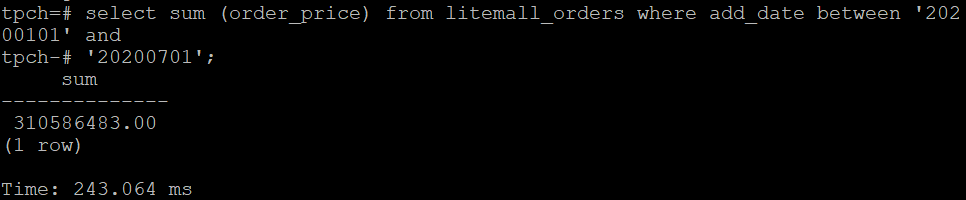
select count(\*) from supplier;;



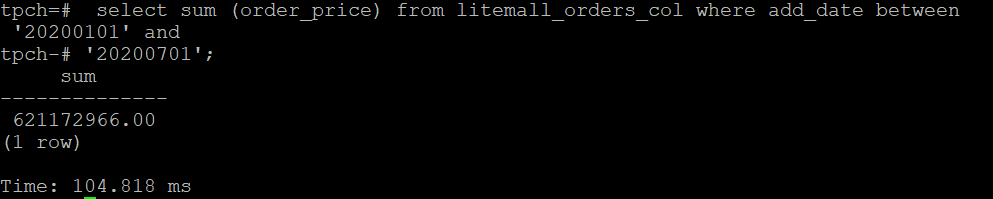
任务二：行存表与列存表执行效率对比

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';

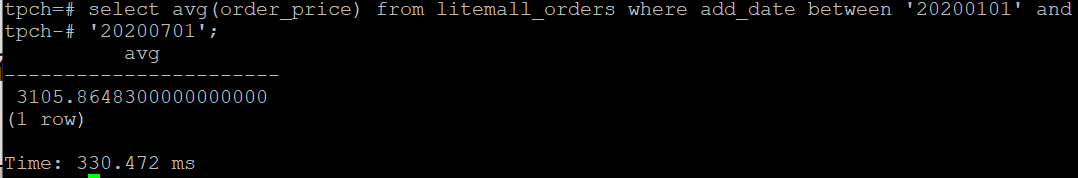


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

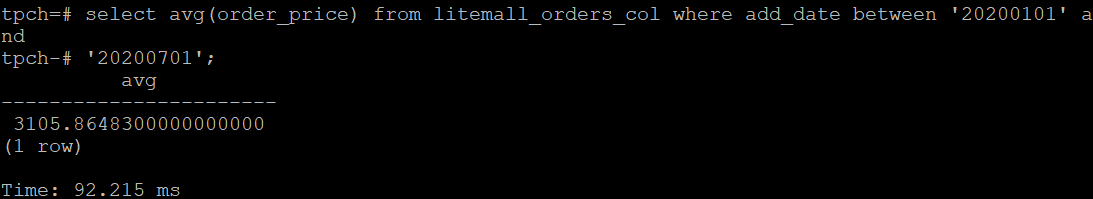


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

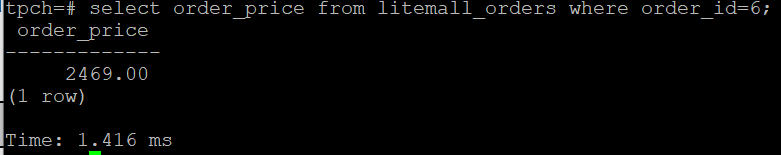


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

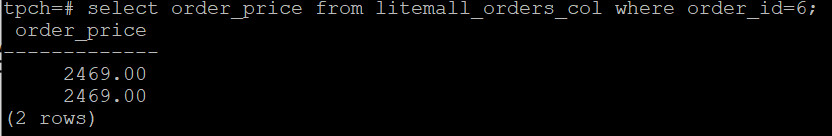


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

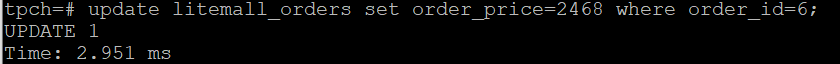


select order\_price from litemall\_orders\_col where order\_id=6;

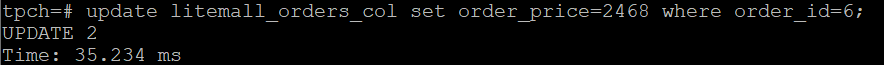


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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

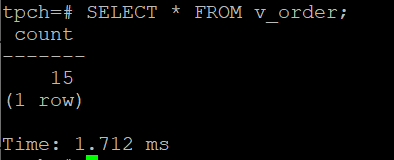
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



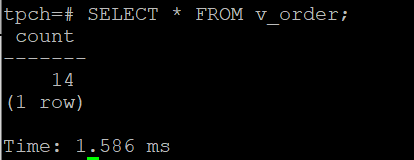
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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;



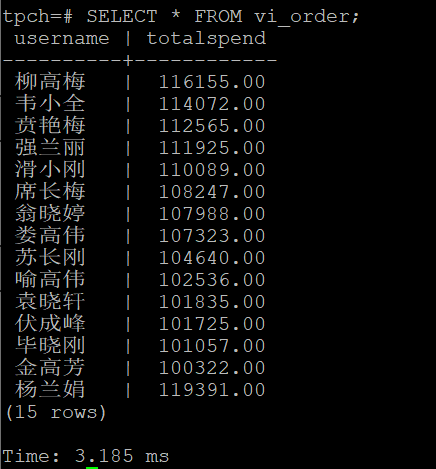
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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：回答如下：

1. 行存表和列存表在执行相同的SQL语句时，执行时间不同的主要原因是它们的数据存储和访问方式不同。行存表是将数据按行存储在磁盘上，每一行包含多个列的数据。当执行一条SQL语句时，行存表需要读取整行数据，包括那些没有在查询结果中使用的列。这种存储方式适合于需要查询和修改完整行数据的操作，比如事务处理和OLTP（联机事务处理）应用。行存表在执行小规模数据查询和频繁的数据更新时效率较高，因为它可以快速地定位和读取整行数据。列存表是将数据按列存储在磁盘上，每一列的数据连续存储在一起。当执行一条SQL语句时，列存表只需要读取所需的列数据，而不需要读取未使用的列。这种存储方式适合于需要聚合、过滤和分析大规模数据集的操作，比如数据仓库和OLAP（联机分析处理）应用。列存表在执行大规模数据查询、复杂的分析查询和聚合操作时效率较高，因为它可以减少磁盘I/O和内存使用。
2. 行存表效率更高情形：小规模数据查询：当查询的数据量相对较小，需要获取整行数据时，行存表的效率更高。这包括针对特定行的查找、单个或少量行的插入、更新和删除操作。
3. 列存表效率更高情形：大规模数据查询和分析：当查询的数据量较大，需要聚合、过滤和分析操作时，列存表的效率更高。这包括复杂的多表关联查询、聚合函数的计算、分组和排序操作。

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：回答如下：

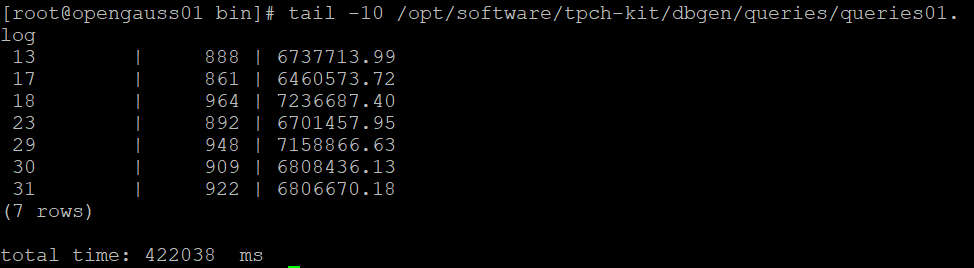
1. 全量物化视图是基于源表的完整数据集创建的物化视图。当源表发生变化时，刷新全量物化视图需要重新计算和加载整个数据集，将源表的变化反映到物化视图中。这可能涉及大量的数据操作和计算，因此全量物化视图的刷新成本较高。全量物化视图适用于源表的变化频率较低、数据量不大、刷新时间要求不高的场景。
2. 增量物化视图是在全量物化视图的基础上引入增量更新机制的物化视图。它记录源表的变化，只更新物化视图中受影响的部分，而无需重新计算和加载整个数据集。增量物化视图通过使用事务日志（如Oracle的归档日志）或触发器等机制来捕获源表的变化，并在刷新时只处理发生变化的数据。这样可以大大减少刷新的时间和成本，提高物化视图的实时性。增量物化视图适用于源表的变化频率较高、数据量较大、刷新时间要求较低的场景。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

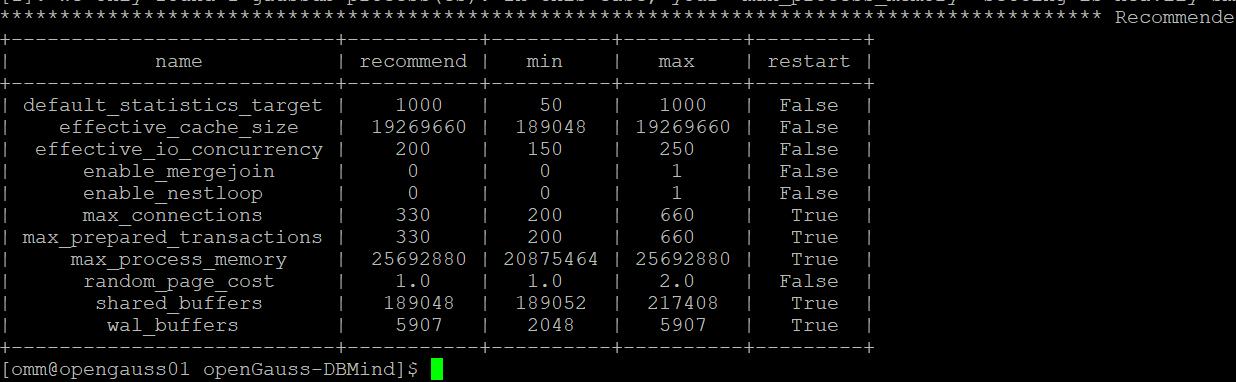
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

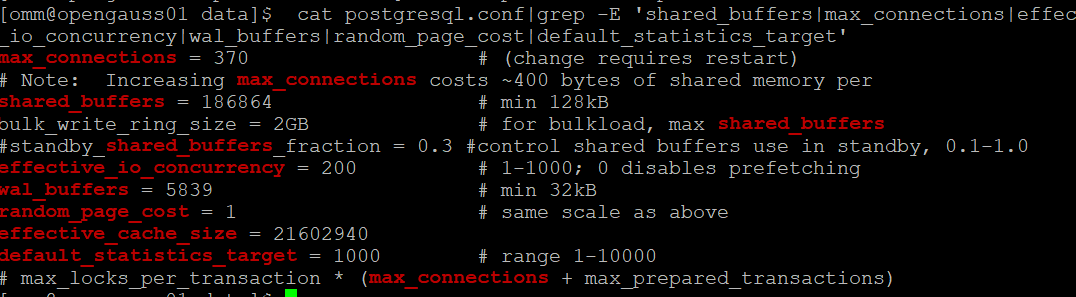
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

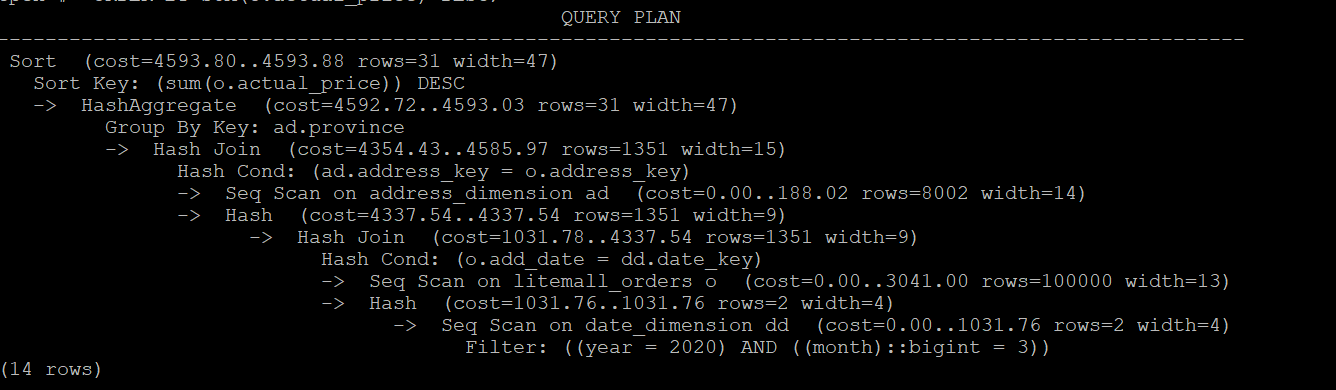
AND o.add\_date = dd.date\_key

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

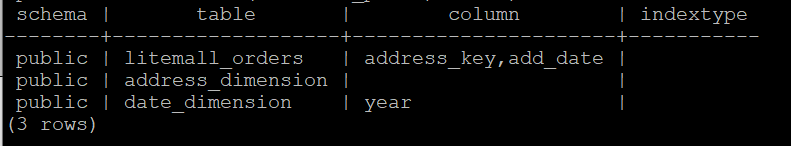
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

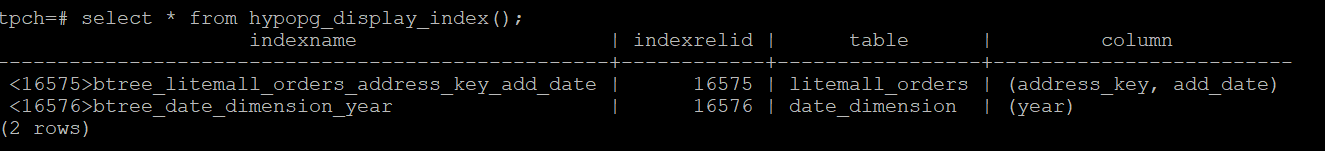
GROUP BY ad.province

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。

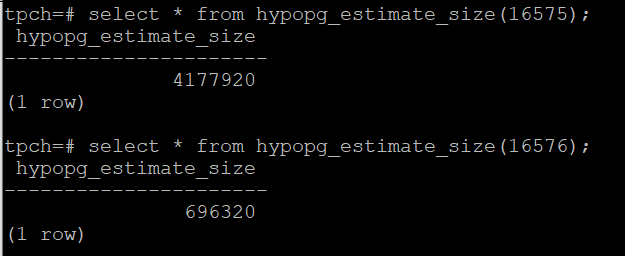
select \* from hypopg\_display\_index();



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

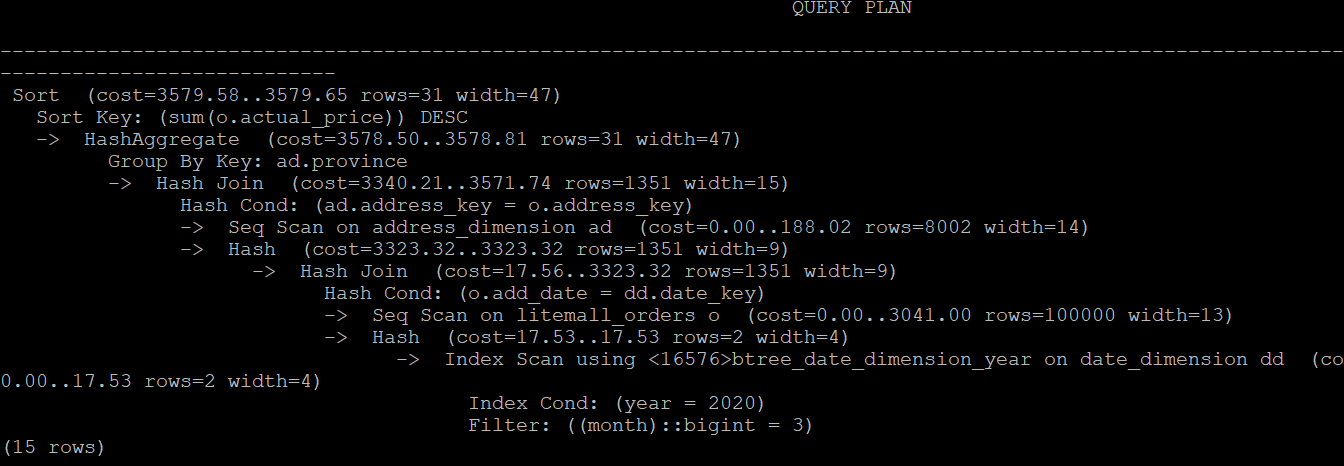
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

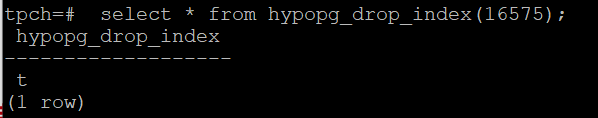
GROUP BY ad.province

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



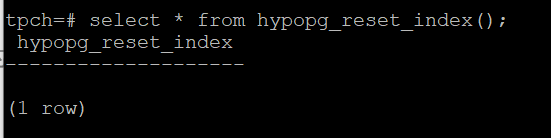
6. 删除某一个索引虚拟列，将执行结果截图。

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



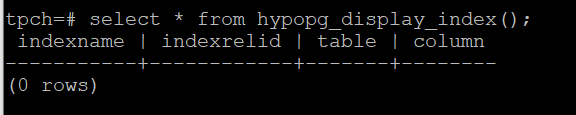
7. 删除某一个索引虚拟列，将执行结果截图。

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。

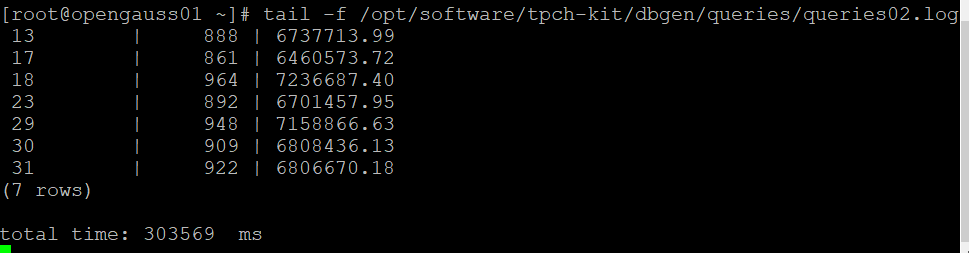
1. 重新执行queries.sql查询，将执行结果截图：

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



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：回答如下：

1. 对shared\_buffers、max\_connections、effective\_cache\_size、effective\_io\_concurrency、wal\_buffers、random\_page\_cost、default\_statistics\_target等参数进行了优化
2. 优化这些参数的目的是根据具体的工作负载和硬件环境，使数据库系统的性能达到最佳状态。通过调整参数配置，可以提高数据库的吞吐量、响应时间和稳定性，以满足应用的性能需求。X-Tuner通过自动化的参数调整和优化算法，帮助用户在不同的场景中找到最优的参数配置，提高数据库系统的性能和效率。

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？

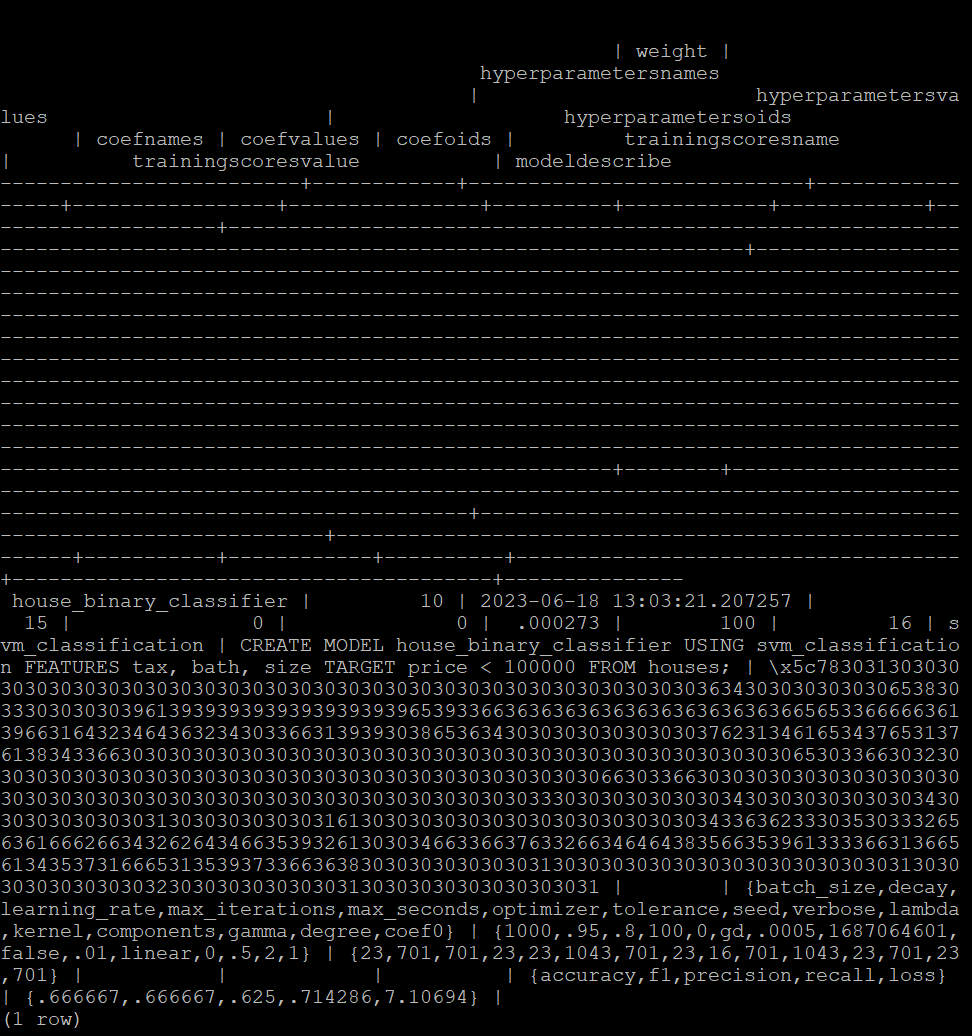
答：回答如下：

1. 索引的使用对于执行SQL有以下好处：
2. 提高查询性能：索引可以加速数据检索，通过创建合适的索引，数据库系统可以快速定位到符合查询条件的数据，减少全表扫描的开销，从而提高查询的执行速度。
3. 减少磁盘I/O操作：索引存储了部分或全部数据的副本，减少了需要读取的磁盘数据量，从而减少磁盘I/O操作的次数。这对于大型表和频繁的数据检索操作特别重要。
4. 优化排序和分组操作：对于排序和分组操作，索引可以提供有序的数据集，减少排序和分组的开销，提高执行效率。
5. 除了使用索引和参数外，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：
6. 数据库设计优化：合理的数据库设计是数据库性能优化的基础。包括选择适当的数据类型、设计良好的表结构、建立关联和约束等。合理的数据库设计可以减少冗余数据和数据一致性问题，提高查询和更新的效率。
7. 查询优化：通过编写高效的SQL查询语句，使用合适的连接方式、合理的过滤条件和优化的查询计划，可以提高查询的性能。例如，避免不必要的联接、使用合适的索引覆盖查询、避免使用过多的子查询等。
8. 服务器配置和硬件优化：合理配置数据库服务器的硬件资源，如内存、磁盘、CPU等，以满足数据库的性能需求。优化数据库服务器的参数配置，如缓冲区大小、并发连接数等，以提高性能和资源利用率。
9. 数据库分区和分表：对于大型数据库，可以通过分区和分表来分散数据存储和查询的压力。将数据分成更小的块，可以提高查询和维护的效率。
10. 数据库缓存和缓存策略：通过使用数据库缓存（如Redis）或应用程序缓存，可以减少对数据库的频繁访问，提高响应速度和整体性能。
11. 定期维护和优化：定期进行数据库的备份、压缩、重建索引和统计信息收集等维护工作，可以保持数据库的健康状态，并提高性能。

# 关卡四、openGauss的DB4AI特性应用

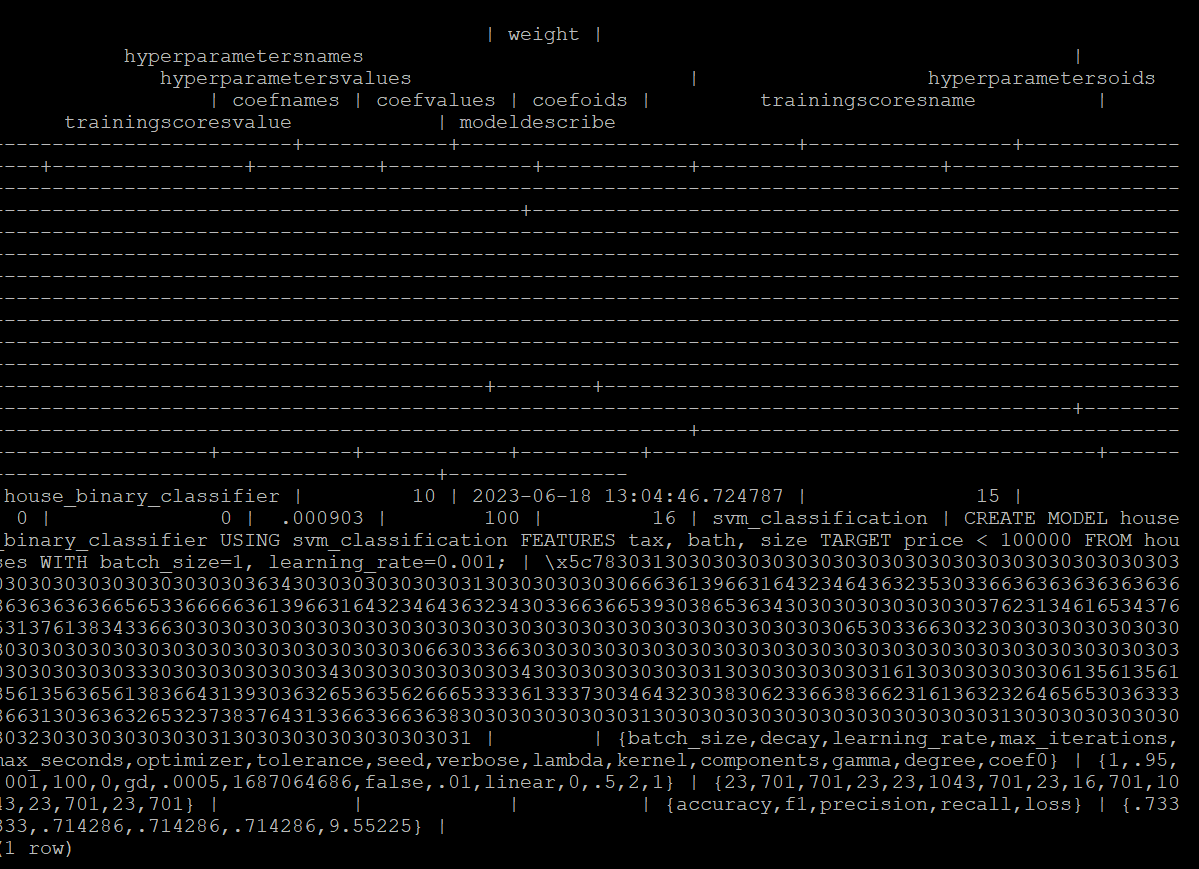
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



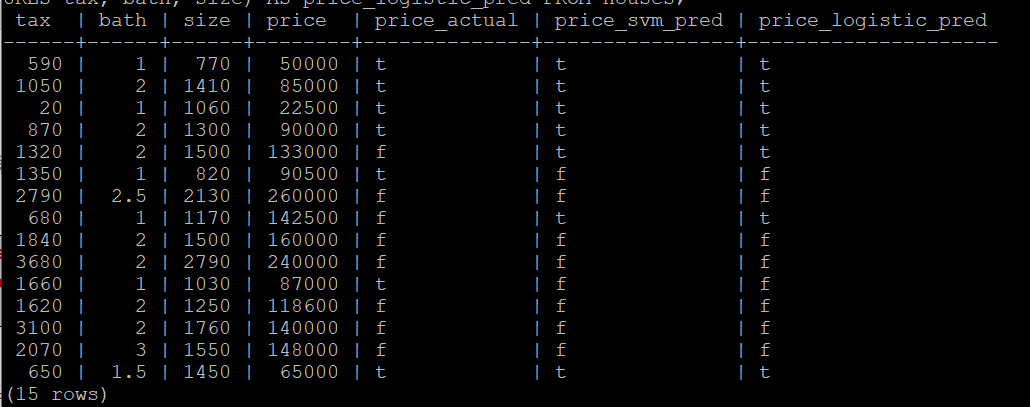
任务二：观察新模型的信息，将执行结果截图。

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：回答如下：

1. 分类模型和回归模型是机器学习中两种常见的预测模型，它们有以下不同之处：
2. 预测目标：分类模型用于预测离散的类别标签，而回归模型用于预测连续的数值。在分类模型中，预测结果通常是一个类别或类别概率；而在回归模型中，预测结果是一个实数值。
3. 输出类型：分类模型的输出是离散的类别标签，通常表示为整数或文本；而回归模型的输出是连续的数值，可以是任意实数。
4. 模型类型：分类模型常用的算法包括逻辑回归、决策树、支持向量机（SVM）和随机森林等；回归模型常用的算法包括线性回归、决策树回归、支持向量回归（SVR）和神经网络等。这些算法在处理不同类型的问题时有不同的特点和适用性。
5. 损失函数：分类模型通常使用交叉熵等损失函数来衡量预测结果与真实类别标签之间的差异；回归模型通常使用均方误差（MSE）或平均绝对误差（MAE）等损失函数来衡量预测结果与真实数值之间的差异。
6. 评估指标：分类模型的评估指标包括准确率、精确率、召回率、F1值等，用于衡量分类模型的分类性能；回归模型的评估指标包括均方误差（MSE）、均方根误差（RMSE）、平均绝对误差（MAE）等，用于衡量回归模型的预测精度。

实践思考题2：什么是SVM算法？

答：回答如下：

1. SVM是一种常用的机器学习算法，用于进行分类和回归任务。它基于统计学习理论和结构风险最小化的原理，具有良好的泛化性能。
2. 在分类任务中，SVM通过在特征空间中构建一个超平面，将不同类别的样本分隔开。超平面的选择是通过最大化样本间的间隔来实现的，即找到能够最好地将样本划分为两个类别的超平面。这些样本中的一部分被称为支持向量，它们位于超平面附近，对决策边界的确定起着重要作用。
3. SVM的核心思想是将低维空间的样本映射到高维空间，以便能够更好地将它们分离。这是通过核函数实现的，核函数可以计算低维样本之间的相似度或内积，而不需要显式进行高维空间的计算。常用的核函数有线性核、多项式核和径向基函数（RBF）核等。

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：回答如下：

在分类问题中，常用的评价指标包括以下几种：

1. 准确率（Accuracy）：准确率是最常见的分类评价指标，表示分类器正确预测的样本比例。计算方式为正确预测的样本数除以总样本数。准确率无法区分不同类别之间的预测效果，特别是在类别不平衡的情况下容易被误导。
2. 精确率（Precision）：精确率衡量了分类器在预测为正例的样本中的真正正例数量。计算方式为真正正例数除以预测为正例的样本数。精确率高表示分类器更少地将负例错误地预测为正例，具有较低的误报率。
3. 召回率（Recall）：召回率衡量了分类器对于真实正例的识别能力，即分类器正确预测为正例的样本数占真实正例的比例。计算方式为真正正例数除以真实正例的样本数。召回率高表示分类器更少地将正例错误地预测为负例，具有较低的漏报率。
4. F1值（F1 Score）：F1值综合考虑了精确率和召回率，是精确率和召回率的调和平均值。F1值可以评估分类器的整体性能，并且对精确率和召回率给予了相等的权重。计算方式为2 \* (精确率 \* 召回率) / (精确率 + 召回率)。
5. 特异度（Specificity）：特异度衡量了分类器对于真实负例的识别能力，即分类器正确预测为负例的样本数占真实负例的比例。计算方式为真真负例数除以真实负例的样本数。特异度高表示分类器更少地将负例错误地预测为正例。
6. AUC-ROC（Area Under the ROC Curve）：AUC-ROC是基于接收者操作特征曲线（ROC曲线）的指标，用于评估分类器在不同阈值下的预测性能。ROC曲线横轴表示1 - 特异度，纵轴表示召回率，AUC-ROC表示ROC曲线下的面积。AUC-ROC的取值范围在0.5到1之间，越接近1表示分类器的性能越好。

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：回答如下：

在回归问题中，常用的评价指标包括以下几种：

1. 平均绝对误差（MAE，Mean Absolute Error）：MAE衡量了预测值与真实值之间的平均绝对差异，表示预测值与真实值之间的平均偏差程度。计算方式为预测值与真实值之差的绝对值的平均值。
2. 均方误差（MSE，Mean Squared Error）：MSE衡量了预测值与真实值之间的平均平方差异，表示预测值与真实值之间的平均误差程度。计算方式为预测值与真实值之差的平方的平均值。
3. 均方根误差（RMSE，Root Mean Squared Error）：RMSE是MSE的平方根，它具有与原始观测值相同的单位。RMSE表示预测值与真实值之间的平均误差程度，并对较大的误差值给予较高的惩罚。
4. R平方（R-Squared）：R平方度量了模型对观测值变异性的解释程度，表示因变量方差中被模型解释的比例。R平方的取值范围在0到1之间，越接近1表示模型对观测值的解释能力越强。
5. 可决系数（Coefficient of Determination）：可决系数也是R平方的一种度量，它表示模型能够解释的因变量方差所占的比例。可决系数的取值范围在0到1之间。
6. 相对绝对误差（RAE，Relative Absolute Error）：RAE是预测值与真实值之间的平均绝对误差与真实值的平均绝对误差之比，可以衡量模型相对于基准模型的性能。
7. 相对均方误差（RSE，Relative Squared Error）：RSE是预测值与真实值之间的平均均方误差与真实值的平均均方误差之比，也可以用于衡量模型相对于基准模型的性能。