openGauss AI特性创新实践课



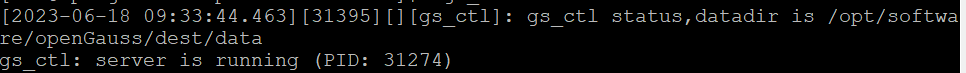
华为技术有限公司

# 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：

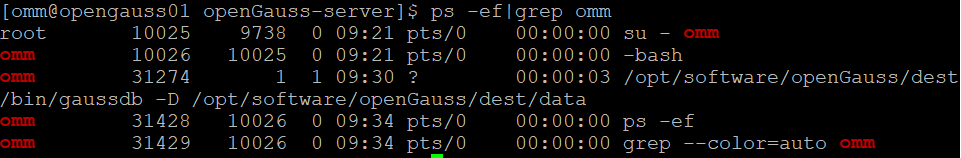
任务一：数据库状态验证

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主要原因有以下几点：

1. 定制化安装：源码编译允许用户根据自己的需求进行定制化安装。用户可以选择编译特定的功能模块、配置参数，以及优化编译选项，以满足其特定的使用场景和需求。

2. 最新版本和功能：源码编译通常可以确保使用最新的数据库版本，包括最新的功能和安全修复。一些数据库的官方发行版或二进制包可能会有一些延迟，而通过源码编译可以更及时地获取到最新的更新。

3. 平台适配性：不同的操作系统和硬件平台可能需要特定的编译选项和优化参数。通过源码编译，用户可以针对自己的操作系统和硬件环境进行定制，以获得更好的性能和稳定性。

4. 调试和优化：源码编译允许用户在需要时进行调试和优化。如果在使用过程中遇到问题或性能瓶颈，可以通过编译选项和调试工具进行诊断和改进。

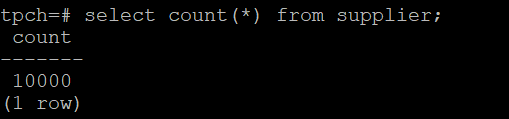
5. 教学和学习目的：通过编译和安装数据库源码，用户可以深入了解数据库的内部工作原理和架构。这对于数据库管理员、开发人员和研究人员来说是非常有价值的学习和教学资源。

# 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：

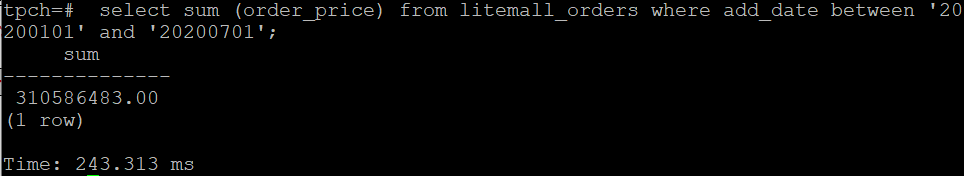
select count(\*) from supplier;;



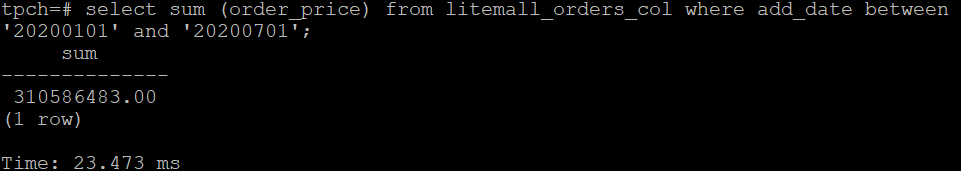
任务二：行存表与列存表执行效率对比

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';

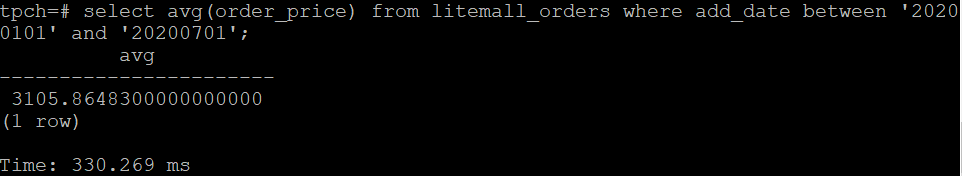


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

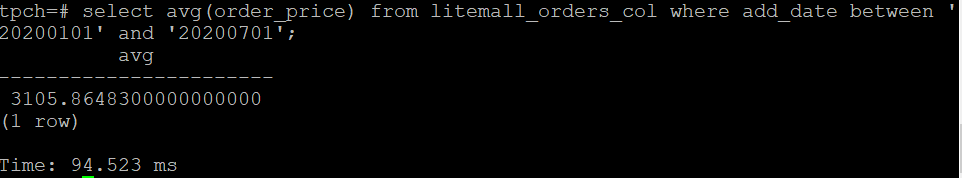


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

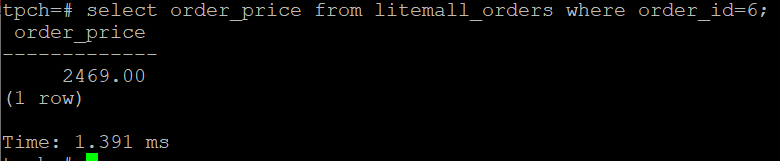


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

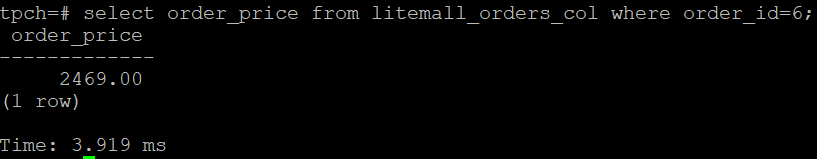


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

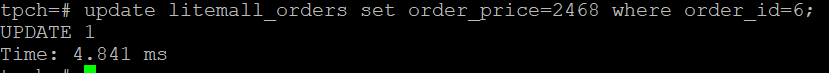


select order\_price from litemall\_orders\_col where order\_id=6;

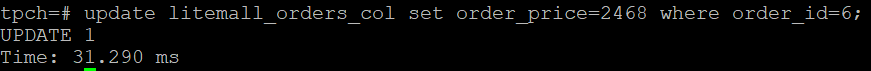


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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

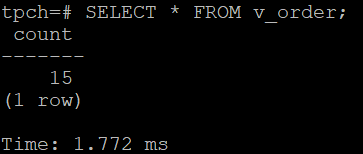
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



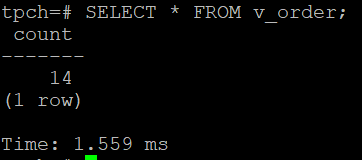
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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，执行时间可能会不同，这是由于它们在数据存储和查询处理方面的不同特点所导致的。行存表是将数据按照行的方式存储在磁盘上。当执行需要读取或修改整行数据的SQL语句时，行存表可以更高效地执行，因为它可以直接读取或修改整个行，无需对行中的每个列进行额外的处理。因此，在执行需要检索多个列或需要更新大部分列的SQL语句时，行存表通常效率更高。

列存表是将数据按照列的方式存储在磁盘上。当执行需要聚合、过滤或仅涉及特定列的SQL语句时，列存表可以更高效地执行。由于列存表将每一列的数据连续存储在一起，它可以更有效地利用压缩算法、减少IO访问量和提高查询性能。因此，在执行需要聚合函数、涉及大量数据列的查询或仅需要特定列的查询时，列存表通常效率更高。

综上所述，在以下情况下，行存表效率更高：

- 需要读取或修改整行数据的SQL语句。

- 需要检索多个列或需要更新大部分列的SQL语句。

而在以下情况下，列存表效率更高：

- 需要聚合函数、涉及大量数据列的查询或仅需要特定列的查询。

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在更新和维护方面的两种不同策略。它们之间的主要差别在于如何处理对基表数据的更改和更新。

1. 全量物化视图：

全量物化视图是通过在物化视图中存储完整的结果集来构建的。当基表发生更新时，需要重新计算和刷新整个物化视图的内容。 更新物化视图需要执行一系列的操作，如对基表进行扫描、计算新的结果集，并将结果存储到物化视图中。这可能需要较长的时间，特别是对于大型数据集或复杂的查询。全量物化视图适用于基表更新频率较低，但需要提供实时或定期刷新的查询结果的场景。

2. 增量物化视图：

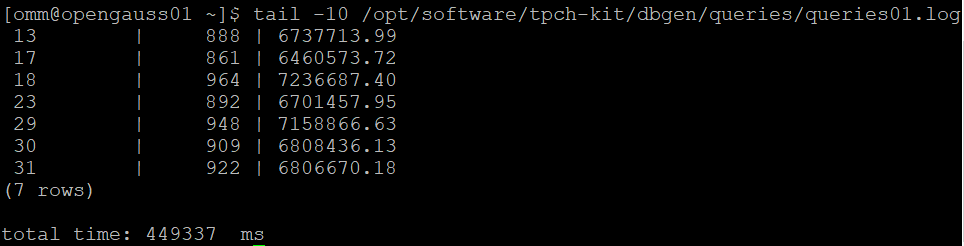
增量物化视图是通过在物化视图中存储基表的更改记录（如插入、更新、删除）来构建的。它仅记录基表的变更操作，而不需要重新计算整个结果集。当基表发生更新时，增量物化视图只需要应用相应的更改记录，而不需要重新计算整个结果集。这样可以显著减少更新物化视图的开销和时间。增量物化视图适用于基表更新频率较高的场景，可以更快速地维护和更新物化视图，并提供实时或准实时的查询结果。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

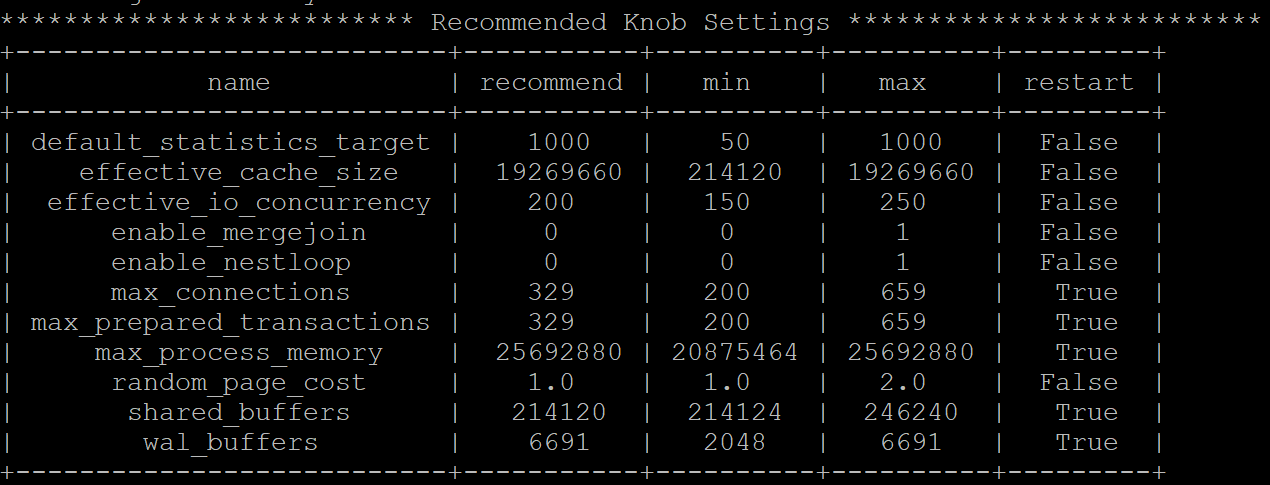
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

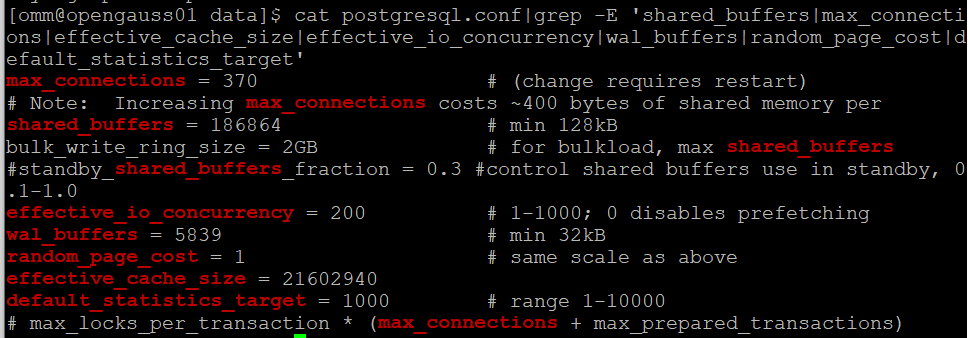
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

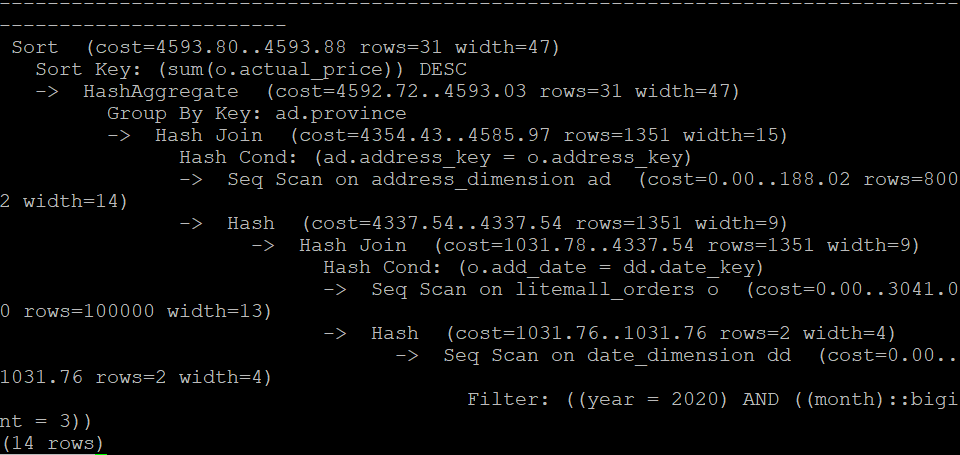
AND o.add\_date = dd.date\_key

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

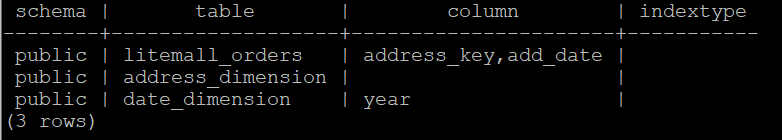
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

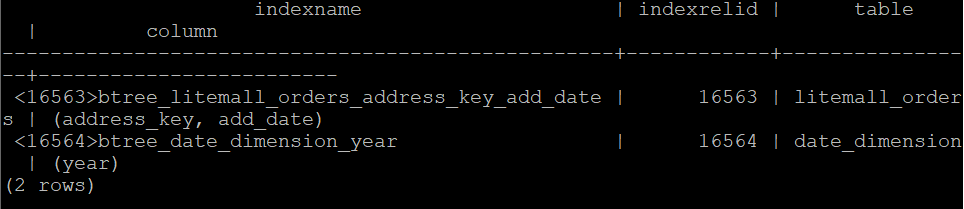
GROUP BY ad.province

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。

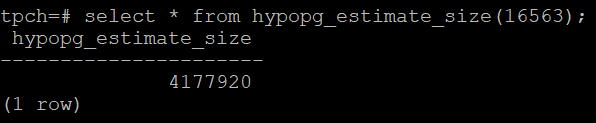
select \* from hypopg\_display\_index();

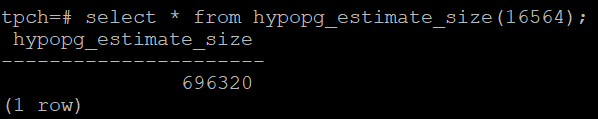


4. 获取索引虚拟列大小结果（单位为：字节），将执行结果截图。

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

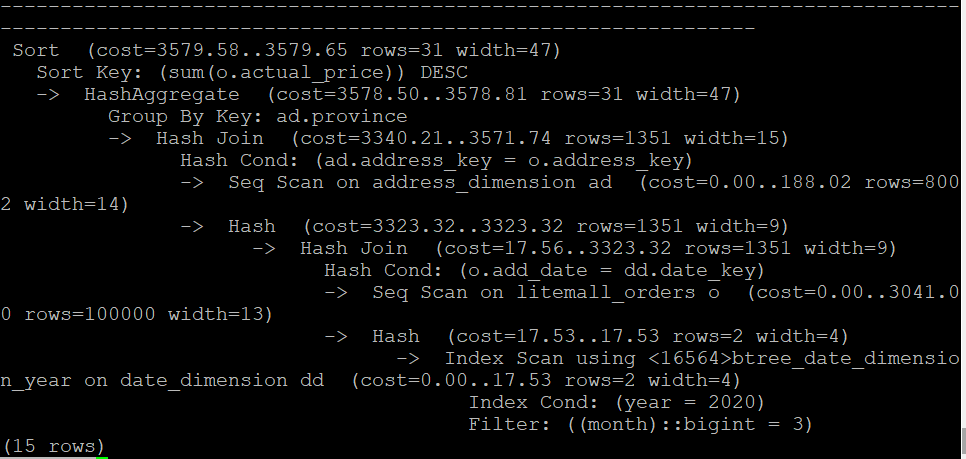
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

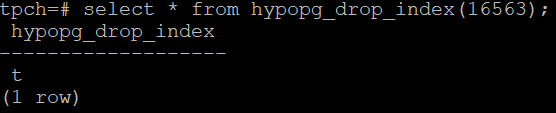
GROUP BY ad.province

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



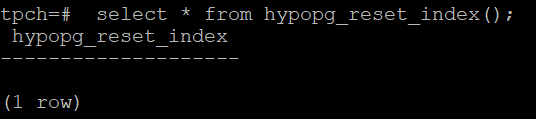
6. 删除某一个索引虚拟列，将执行结果截图。

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



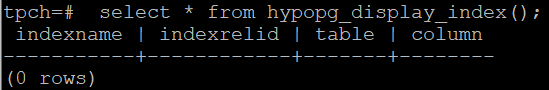
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

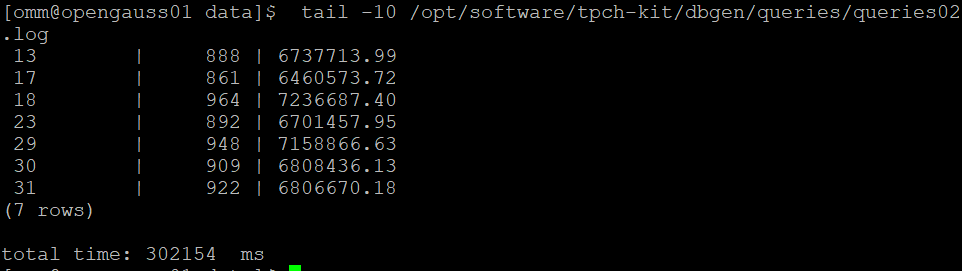
select \* from hypopg\_display\_index();



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



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？

对shared\_buffers、max\_connections、effective\_cache\_size、effective\_io\_concurrency、wal\_buffers、random\_page\_cost、default\_statistics\_target参数进行了优化

1. shared\_buffers（共享缓冲区）：

- 优化目标：共享缓冲区用于存储从磁盘读取的数据块，提高数据库的性能。通过优化该参数，可以使数据库更有效地使用内存，减少磁盘IO，提高查询性能。

- 优化原因：增加共享缓冲区大小可以提高数据缓存的命中率，减少从磁盘读取数据的次数，加快查询和数据访问的速度。

2. max\_connections（最大连接数）：

- 优化目标：最大连接数指定了数据库系统同时支持的最大客户端连接数。通过优化该参数，可以确保系统具有足够的连接资源来处理并发请求，避免资源瓶颈和性能下降。

- 优化原因：如果并发连接数超过了数据库系统的处理能力，可能会导致连接排队、延迟和性能下降。适当增加最大连接数可以提高系统的并发处理能力。

3. effective\_cache\_size（有效缓存大小）：

- 优化目标：有效缓存大小表示数据库可以使用的系统缓存大小。通过优化该参数，可以告诉查询优化器有关系统缓存大小的信息，以便它可以更好地估计查询的成本和选择合适的查询执行计划。

- 优化原因：查询优化器使用有效缓存大小来估计查询的成本和选择索引或顺序扫描等查询执行计划。正确配置有效缓存大小可以提高查询计划的准确性，从而提高查询性能。

4. effective\_io\_concurrency（有效IO并发度）：

- 优化目标：有效IO并发度表示数据库在执行IO操作时可以并行进行的数量。通过优化该参数，可以提高并发IO操作的效率，减少IO等待时间，提高查询和数据访问的速度。

- 优化原因：数据库通常需要执行许多IO操作，如从磁盘读取数据块。通过增加有效IO并发度，可以让数据库并行执行多个IO操作，从而提高整体IO吞吐量和响应速度。

5. wal\_buffers（WAL缓冲区）：

- 优化目标：WAL缓冲区用于存储事务日志（Write-Ahead Log），用于持久化数据修改操作。通过优化该参数，可以提高WAL写入的性能，减少事务提交的延迟。

- 优化原因：较大的WAL缓冲区可以减少频繁的磁盘写入操作，提高写入性能和持久化的效率

。较小的WAL缓冲区可能会导致频繁的磁盘写入操作，降低事务提交的速度。

6. random\_page\_cost（随机页面成本）：

- 优化目标：随机页面成本表示随机IO操作相对于顺序IO操作的成本。通过优化该参数，可以影响查询优化器在选择查询执行计划时对随机IO成本的考虑程度。

- 优化原因：随机IO通常比顺序IO更加昂贵，因为它需要在磁盘上进行磁头的寻道操作。通过适当调整随机页面成本，可以让查询优化器更准确地估计不同查询执行计划的成本，从而选择更高效的查询计划。

7. default\_statistics\_target（默认统计目标）：

- 优化目标：默认统计目标用于指定自动生成表和索引的统计信息的精度级别。通过优化该参数，可以提高查询优化器对表和索引数据的统计分析准确性，从而生成更好的查询执行计划。

- 优化原因：查询优化器使用统计信息来估计查询成本、选择索引和执行计划。通过增加默认统计目标，可以增加生成统计信息的详细程度，提高查询优化器的决策准确性，从而提高查询性能。

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语句时有以下好处：

1. 提高查询性能：索引可以加速查询操作，减少扫描整个表的需要。通过使用索引，数据库可以快速定位到匹配条件的行，从而减少IO访问和减少查询时间。

2. 优化排序和聚合操作：索引可以加速排序和聚合操作，如ORDER BY、GROUP BY和聚合函数等。通过使用索引，数据库可以避免对整个表进行排序或聚合，从而提高这些操作的性能。

3. 减少磁盘IO：索引允许数据库更有效地利用磁盘IO。通过减少需要读取的数据块数量，索引可以减少磁盘IO操作的次数，从而提高查询性能。

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

1. 数据库设计：合理的数据库设计可以显著影响性能。包括选择适当的数据类型、定义正确的主键和外键、规范化数据模型等。

2. 查询优化：编写高效的SQL查询语句可以提高数据库性能。这包括选择合适的查询方式、避免全表扫描、优化WHERE子句和JOIN操作等。

3. 缓存和缓冲区管理：通过使用适当的缓存和缓冲区管理策略，可以减少IO访问和提高数据访问速度。包括使用合理的缓存策略、调整内存参数和缓冲区大小等。

4. 硬件和服务器优化：优化数据库服务器的硬件配置和操作系统参数，如磁盘配置、内存分配、并发连接设置等，可以提升整体性能。

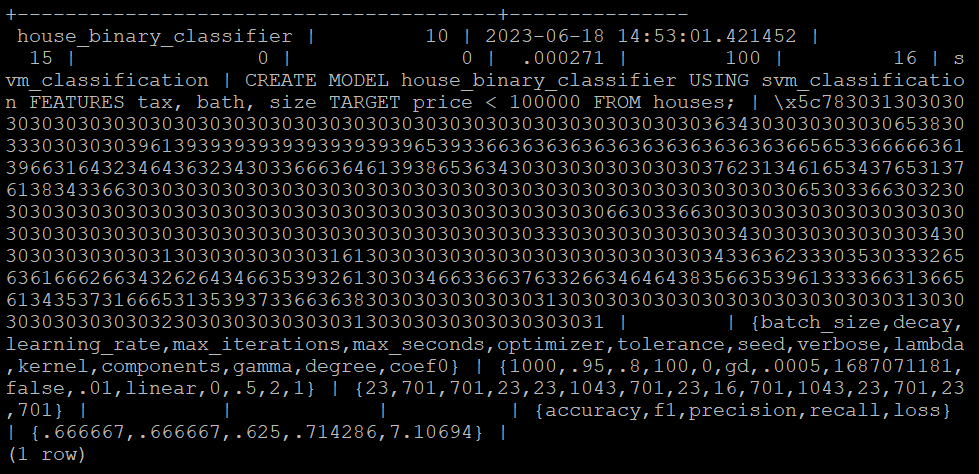
5. 定期维护和优化：定期执行数据库维护操作，如索引重建、统计信息更新、日志清理和备份等，可以保持数据库的健康状态和高性能。

6. 负载均衡和分布式架构：对于高负载或大规模应用，考虑使用负载均衡和分布式架构来分担单个数据库服务器的负载，提高整体性能和可扩展性。

# 关卡四、openGauss的DB4AI特性应用

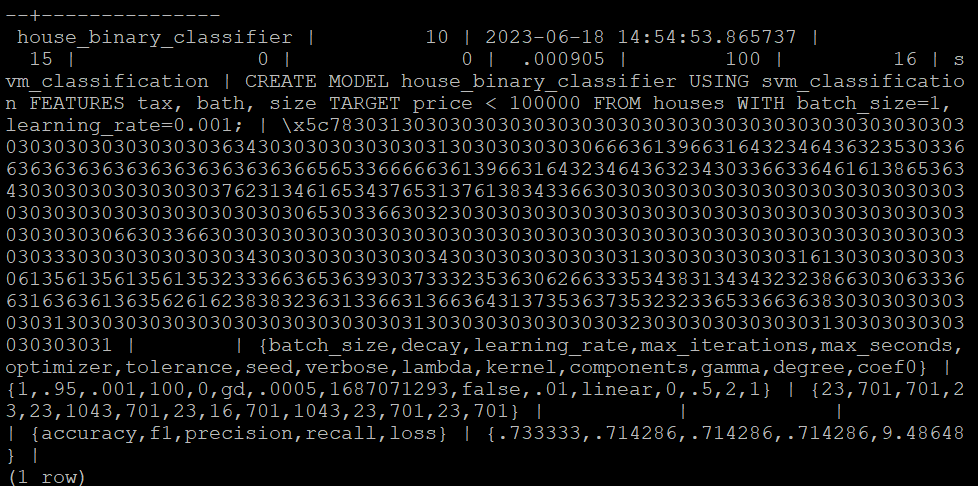
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



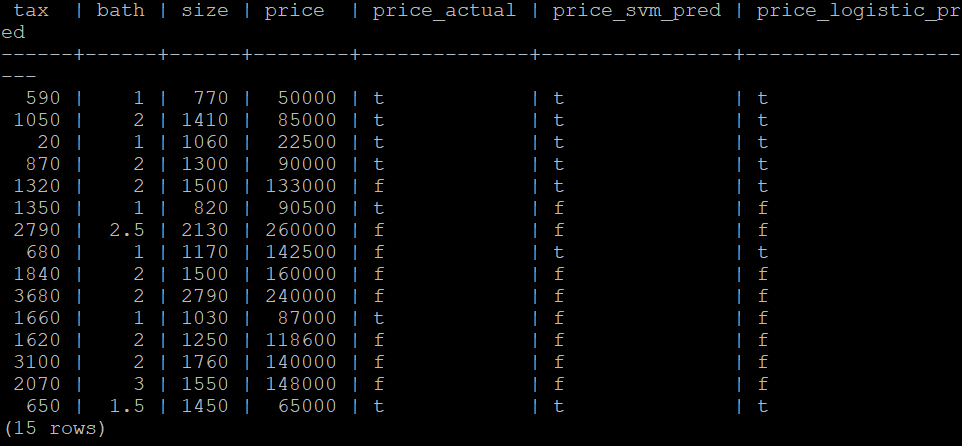
任务二：观察新模型的信息，将执行结果截图。

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？

分类模型和回归模型是机器学习中两种不同的预测模型，它们在处理不同类型的问题和输出变量上有所区别。分类模型用于预测类别，而回归模型用于预测连续数值。

分类模型：

- 用途：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，并进行分类判定。

- 输出变量：输出变量是有限个离散的类别标签，例如二元分类（两个类别）或多元分类（多个类别）。

- 例子：垃圾邮件过滤、图像分类、疾病诊断等问题常常使用分类模型。

回归模型：

- 用途：回归模型用于预测连续的数值输出。它通过学习输入特征与输出变量之间的关系，对连续数值进行预测。

- 输出变量：输出变量是连续的数值或实数，可以是任意范围内的实数值。

- 例子：房价预测、销售量预测、股票价格预测等问题通常使用回归模型。

主要区别：

1. 输出变量类型：分类模型的输出是离散的类别标签，而回归模型的输出是连续的数值。

2. 预测目标：分类模型旨在将数据分为不同的类别，回归模型旨在预测数值。

3. 模型类型：分类模型通常使用分类算法，如逻辑回归、决策树、支持向量机等。回归模型通常使用回归算法，如线性回归、决策树回归、神经网络回归等。

实践思考题2：什么是SVM算法？

SVM（Support Vector Machine）算法是一种监督学习算法，常用于分类和回归任务。SVM 在机器学习领域具有广泛应用，并且在处理线性和非线性分类问题上表现出色。

SVM 的基本思想是通过找到一个最优的超平面来将不同类别的样本分开，同时最大化与该超平面最近的样本点之间的间隔，这些样本点被称为支持向量。该超平面可以用来预测新样本的类别。

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常用于衡量分类模型的性能和准确性。以下是常见的分类问题评价指标：

1. 准确率（Accuracy）：

- 含义：准确率是分类正确的样本数与总样本数之比。它衡量模型在整个数据集上分类正确的能力。

- 计算方法：准确率 = 分类正确的样本数 / 总样本数。

2. 精确率（Precision）：

- 含义：精确率衡量模型在预测为正类别的样本中，真正为正类别的比例。它关注的是模型的预测结果中真正为正类别的准确性。

- 计算方法：精确率 = 真正为正类别的样本数 / 预测为正类别的样本数。

3. 召回率（Recall）：

- 含义：召回率衡量模型在所有真实为正类别的样本中，正确预测为正类别的比例。它关注的是模型对正类别的覆盖程度。

- 计算方法：召回率 = 真正为正类别的样本数 / 所有真实为正类别的样本数。

4. F1 值（F1 Score）：

- 含义：F1 值是精确率和召回率的综合指标，用于平衡两者之间的关系。它是精确率和召回率的调和平均值。

- 计算方法：F1 值 = 2 \* (精确率 \* 召回率) / (精确率 + 召回率)。

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？

回归问题的评价指标用于衡量回归模型的性能和准确性。以下是常见的回归问题评价指标：

1. 均方误差（Mean Squared Error，MSE）：

- 含义：均方误差衡量模型预测值与真实值之间的平均差异程度。它计算预测值与真实值之间差的平方，并求其平均值。

- 计算方法：MSE = (1/n) \* Σ(yi - ŷi)²，其中 yi 是真实值，ŷi 是模型的预测值，n 是样本数量。

2. 均方根误差（Root Mean Squared Error，RMSE）：

- 含义：均方根误差是均方误差的平方根，它是对均方误差的标准化指标。RMSE 衡量模型预测值与真实值之间的平均差异程度。

- 计算方法：RMSE = √MSE

3. 平均绝对误差（Mean Absolute Error，MAE）：

- 含义：平均绝对误差衡量模型预测值与真实值之间的平均绝对差异程度。它计算预测值与真实值之间的绝对差，并求其平均值。

- 计算方法：MAE = (1/n) \* Σ|yi - ŷi|，其中 yi 是真实值，ŷi 是模型的预测值，n 是样本数量。

4. 决定系数（Coefficient of Determination，R²）：

- 含义：决定系数衡量模型对因变量的解释能力，即模型对观测数据的方差的解释程度。它表示模型预测值与真实值之间的方差比例。

- 计算方法：R² = 1 - (Σ(yi - ŷi)² / Σ(yi - ȳ)²)，其中 yi 是真实值，ŷi 是模型的预测值，ȳ 是真实值的平均值。

这些评价指标在回归问题中用于度量模型的预测精度和误差程度。MSE、RMSE、MAE 衡量预测值与真实值之间的差异程度，越小表示模型越准确。R² 衡量模型解释观测数据方差的能力，越接近1表示模型拟合较好。根据具体问题的需求，选择适合的指标进行评估和比较。