openGauss AI特性创新实践课



华为技术有限公司

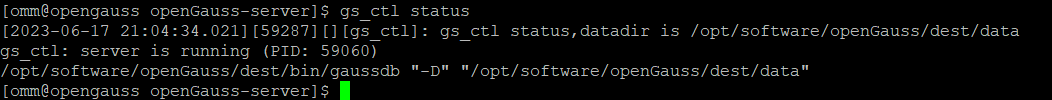
# 关卡一、openGauss数据安装及基本操作

openGauss数据安装及基本操作, 作业提交任务如下：

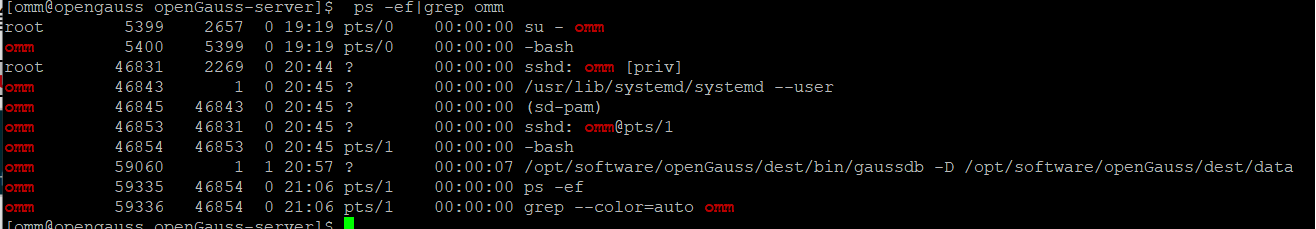
任务一：数据库状态验证

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）



1. 查看数据库状态



**实验思考题：为什么需要通过源码编译，安装数据库？**

**回答·**

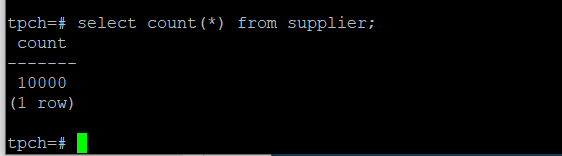
通过源码编译安装数据库，可以根据具体需求进行定制化配置。使得数据库系统能够更好地适应你的应用程序和环境。提高安全性。可以精确地控制所安装的软件版本和组件此外，通过源码编译，你可以排除依赖项冲突和其他兼容性问题。在需要更新组件的过程中也可以更好的依据需求进行更新。

# 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：

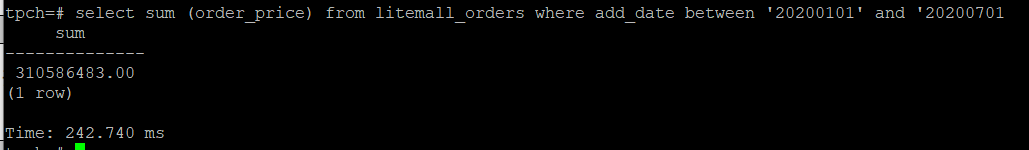
select count(\*) from supplier;



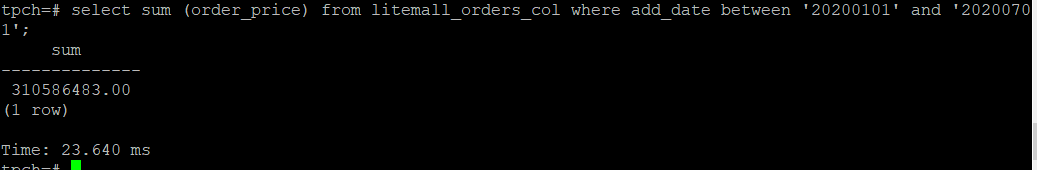
任务二：行存表与列存表执行效率对比

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';

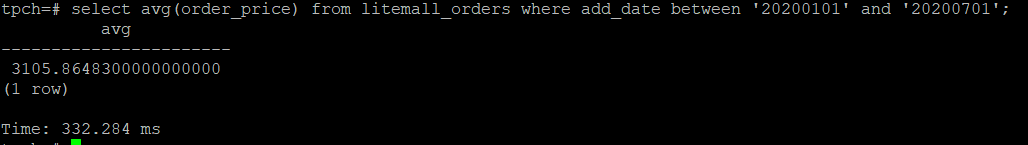


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

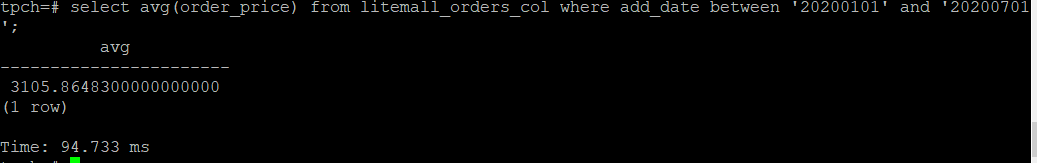


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

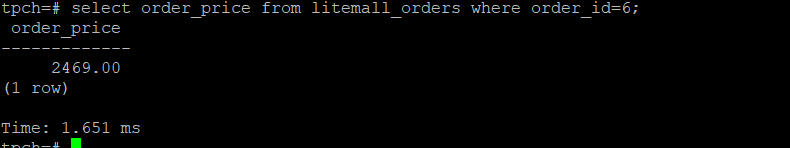


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

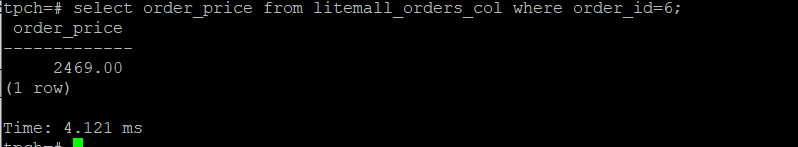


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

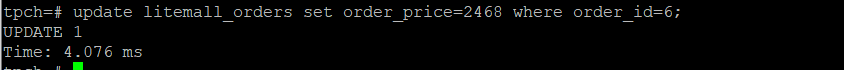


select order\_price from litemall\_orders\_col where order\_id=6;

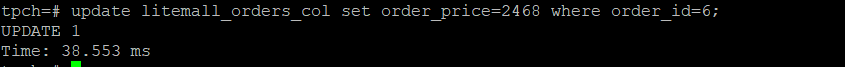


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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

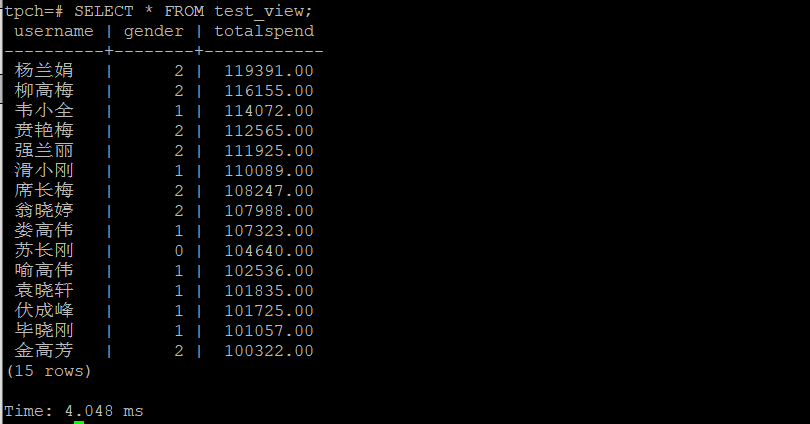


update litemall\_orders\_col set order\_price=2468 where order\_id=6;

任任务三：物化视图的使用

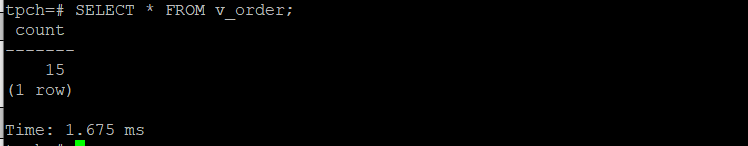
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



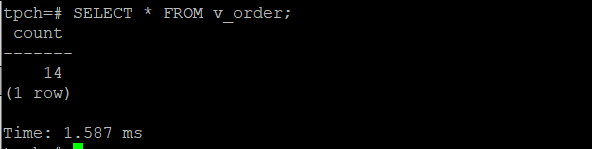
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



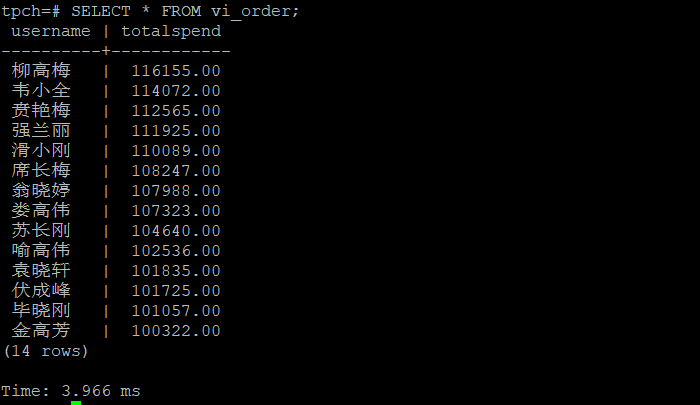
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



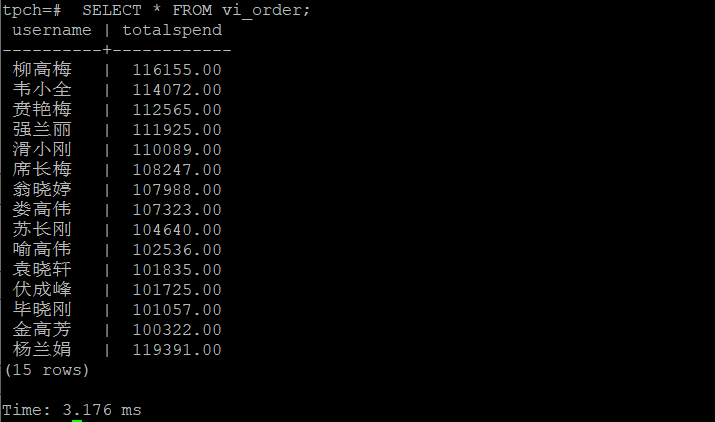
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM vi\_order;



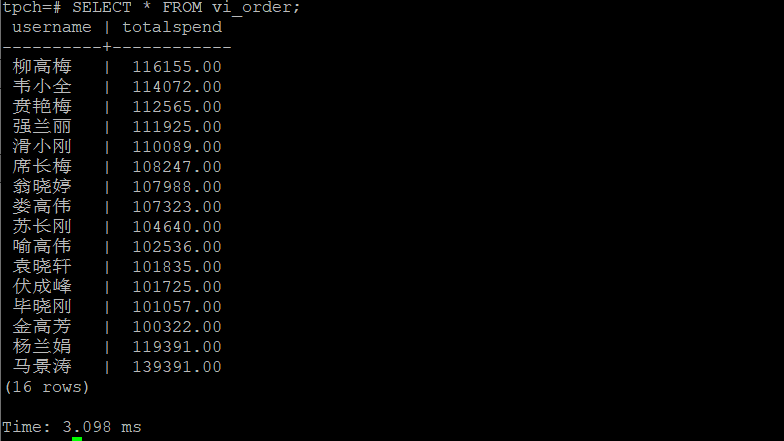
5. 对表进行操作后，刷新增量物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM vi\_order;



6.查询物化视图结果，**将执行结果截图**。

tpch=# **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？**

**回答：**

行村表与列存表在物理存储上的组织方式不同。行存表是以一个存储对象为单位存储，其中包含着不同的字段可能是不同的类型。列存表，是以特征字段的的单位进行存储，全部都是同一类型，便于压缩和批量处理。

行存表的效率更高：对单个或少量行进行增删改操作时，行存表通常更高效。

列存表的效率更高：对大量数据进行聚合、分组和分析查询。

**实践思考题2：全量物化视图与增量物化视图有哪些差别？**

**回答：**

全量物化视图：创建时会完全计算和存储结果集。当底层表发生变化时，需要手动或自动地刷新全量物化视图。全量物化视图适用于对静态数据进行频繁的查询和聚合操作。

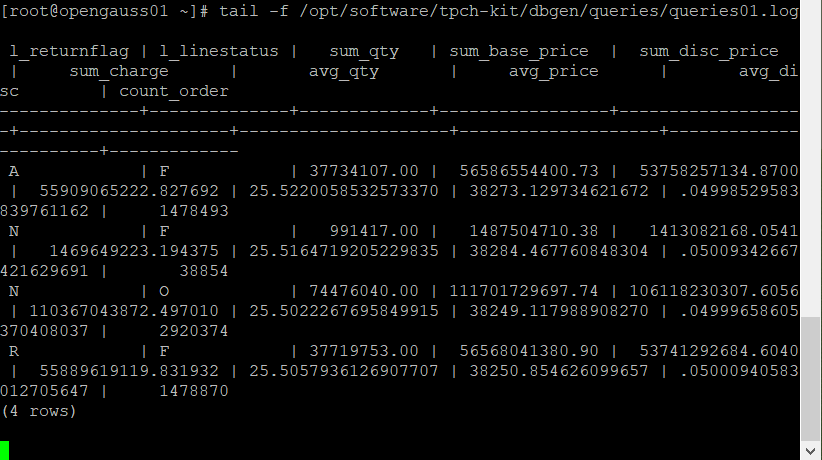
增量物化视图：增量物化视图是一种基于增量变更的物化视图，它会记录底层表的变更，并通过变更日志来实时维护物化视图的数据。无需手动更新。适合频繁更新的数据。存储效率较高。

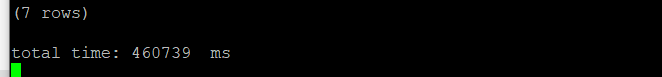
# 关卡三、openGauss的AI4DB特性应用

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：

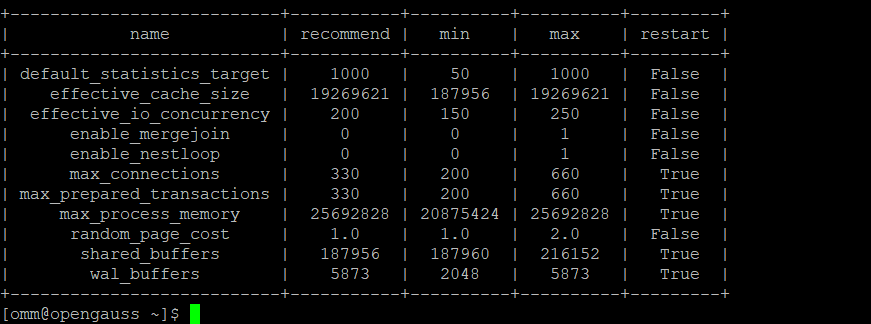
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log





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

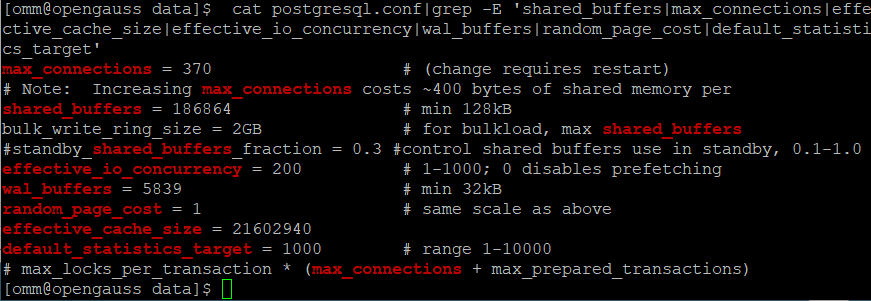
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

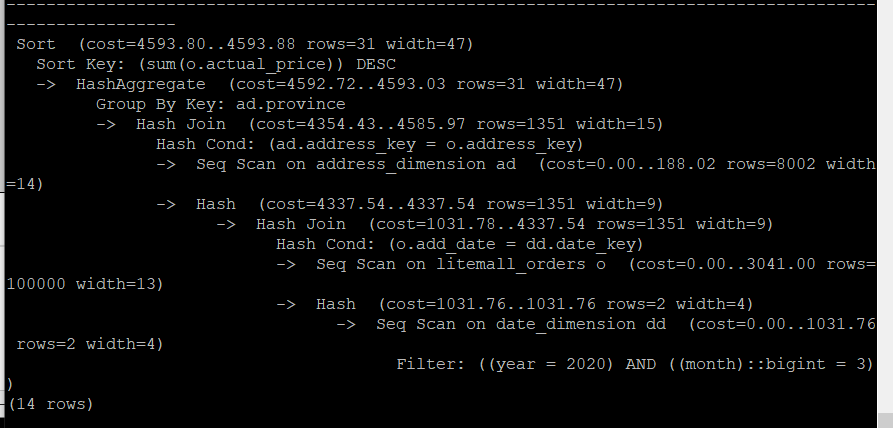
AND o.add\_date = dd.date\_key

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

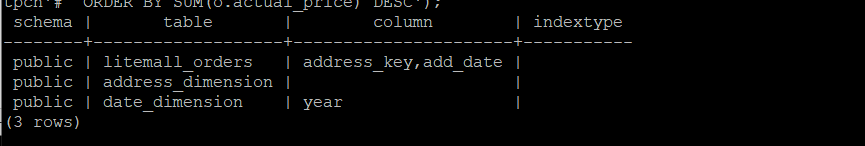
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

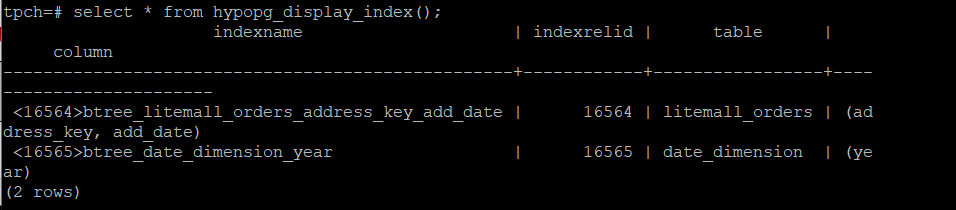
GROUP BY ad.province

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。

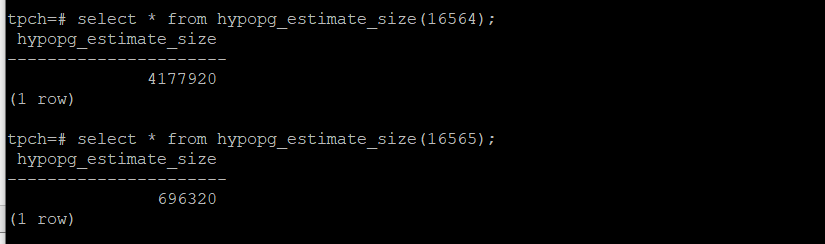
select \* from hypopg\_display\_index();



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(16564);

select \* from hypopg\_estimate\_size(16565);



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

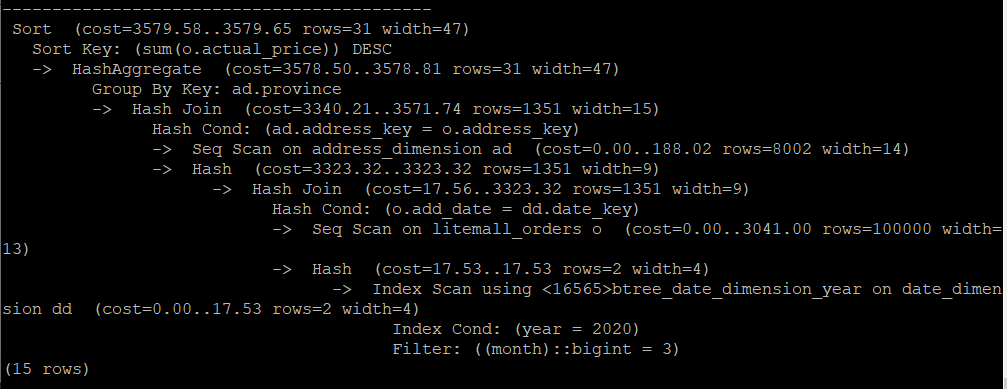
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

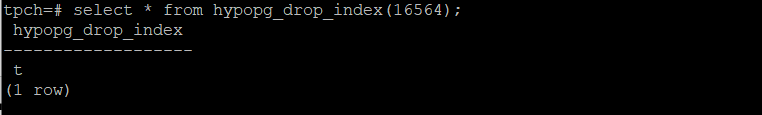
GROUP BY ad.province

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



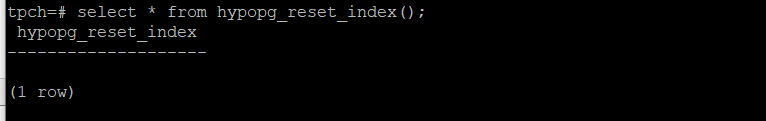
6. 删除某一个索引虚拟列，将执行结果截图。

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



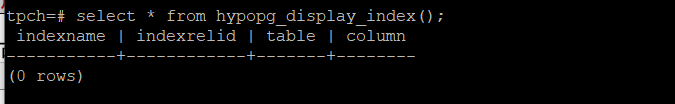
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

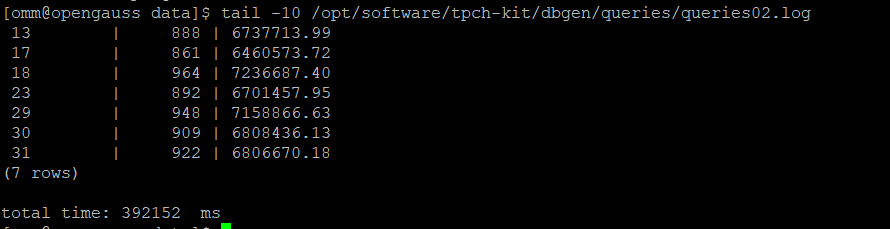
select \* from hypopg\_display\_index();



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？**

**回答**

Max-connection

max\_prepared\_transactions

max\_process\_memory

shared\_buffers

wal\_buffers

由之前的实验结果（截图）一种，对以上参数进行了调整。尽可能提升数据库的性能、响应速度和并发处理能力。同时又不造成资源的浪费

**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？**

**回答：**

提高查询性能：索引可以加快查询操作的速度。

减少数据读取量：索引可以帮助减少需要读取的数据量。

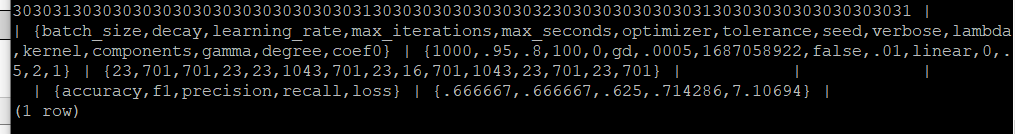
优化排序和聚合操作：对于排序和聚合操作，索引可以提供有序的数据访问路径，减少排序和聚合。

但是索引的创建同样会有空间开销，在使用时同样会有时间开销。所以，我们需要在索引的优缺点方面做好平衡。

# 关卡四、openGauss的DB4AI特性应用

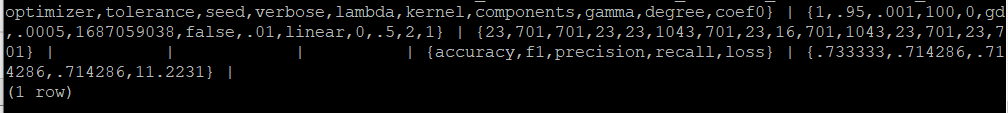
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



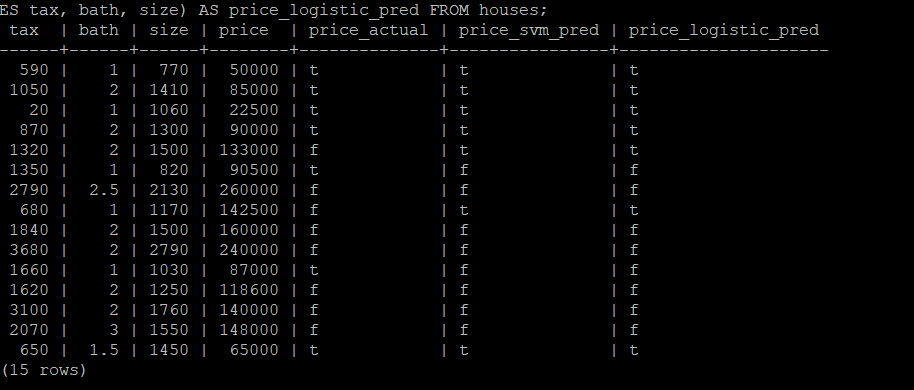
任务二：观察新模型的信息，将执行结果截图。

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？**

**回答**

分类的目的是为了寻找决策边界，预测之后的输出值是一个定性的分类类别。是离散的

回归的目的是为了找到最优拟合，通过回归算法得到是一个最优拟合线，标签是连续的，输出的是一个连续值。

**实践思考题2：什么是SVM算法？**

**回答**

是一种二类分类模型。它的基本模型是定义在特征空间上的间隔最大的线性分类器，间隔最大使它有别于感知机；。我们可以通过引入核函数的概念，可将支持向量机拓展到非线性分类问题上。优点：有坚实的理论基础作为支撑，模型结果容易解释，间隔最大化让模型拥有更好的鲁棒性的有带你

**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？**

**回答**

准确率（Accuracy）：表示分类器正确预测的样本数占总样本数的比例。计算公式为：准确率 = (预测正确的样本数) / (总样本数)。

召回率（Recall）：召回率是分类器正确预测为正类的样本占实际正类样本数的比例。计算公式为：召回率 = (真正例) / (真正例 + 假负例)。

一般情况下，我们可以使用混淆矩阵来对两者进行整体的分析和计算。

**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？**

**回答**

均方误差(MSE):预测值与真实值之差的平方和的平均值。

平均绝对误差(MAE):预测值与真实值之差的绝对值的平均值。

平均绝对百分比误差(MAPE)：这个指标是对相对误差损失的预期值.所谓相对误差,就是绝对误差和真值的百分比.

均方根误差（ RMSE）：均方根误差是均方误差的平方根，与MSE相比，RMSE更加重视较大误差的影响。计算公式为：RMSE = sqrt(MSE)。