openGauss AI特性创新实践课



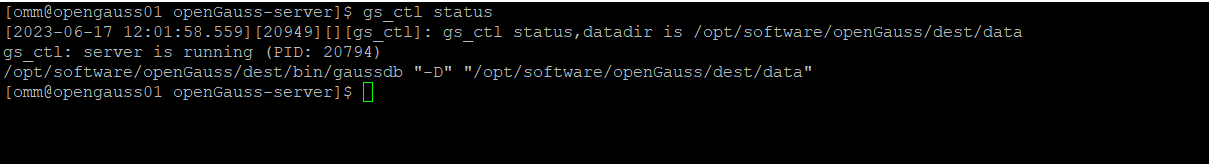
华为技术有限公司

# 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：

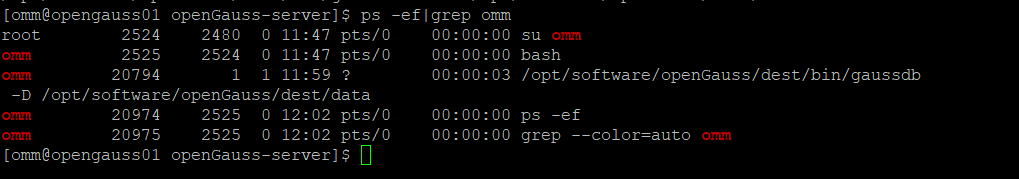
任务一：数据库状态验证

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有如下原因：

（1）不同操作系统和架构对数据库的支持程度不同，有的只支持数据库源码编译安装，而使用源码编译安装数据库可以将其移植到不同操作系统硬件平台上，使得部署过程大大简化；

（2）数据库软件往往具有高度定制性要求，需要在特定的操作系统、硬件架构、网络环境下运行。因此通过源码编译可以更好地定制和控制MySQL编译参数，以满足用户特定需求；

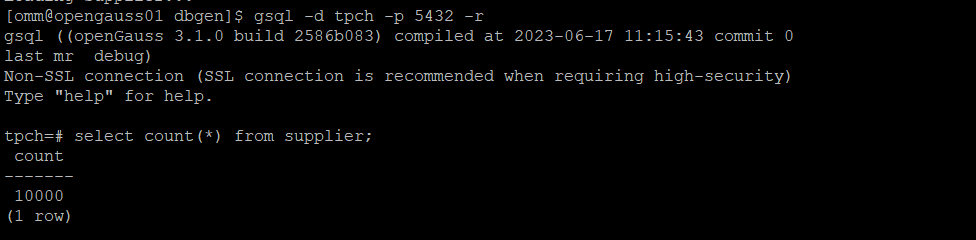
（3）通过源码编译安装数据库可以提高性能并减少资源消耗。

# 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：

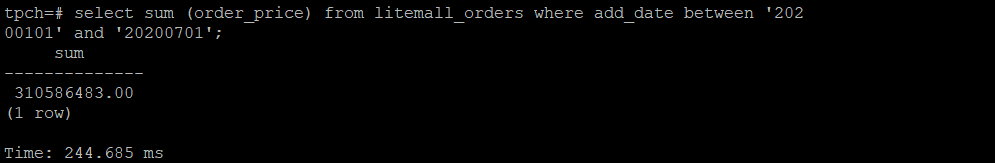
select count(\*) from supplier;;



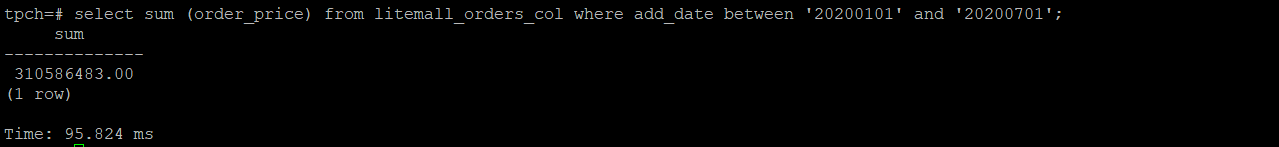
任务二：行存表与列存表执行效率对比

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';

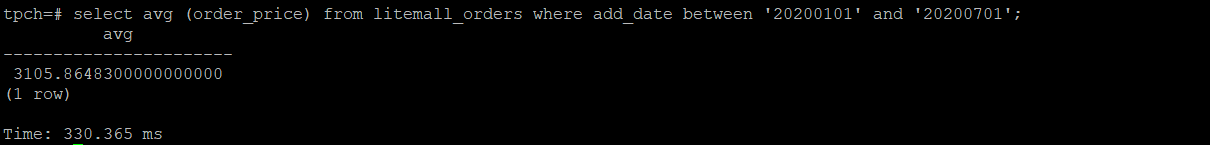


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

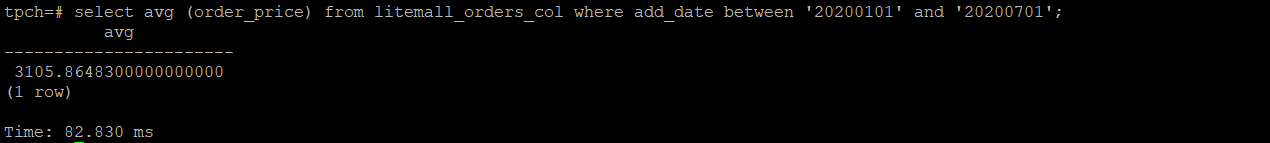


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

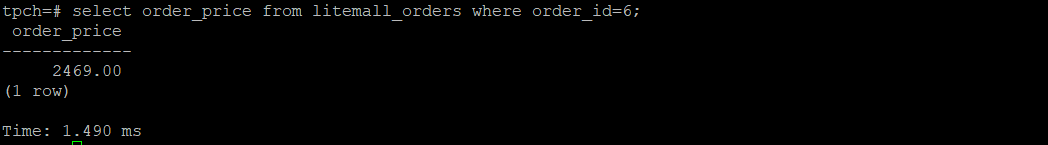


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

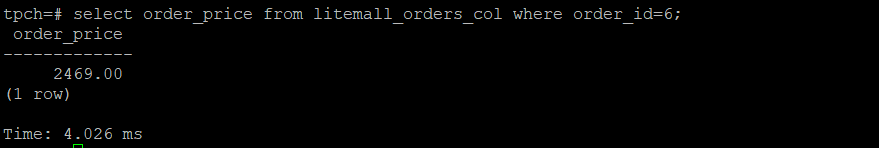


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

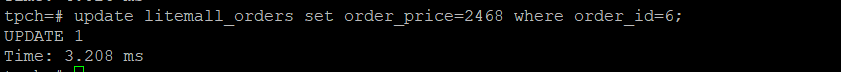


select order\_price from litemall\_orders\_col where order\_id=6;

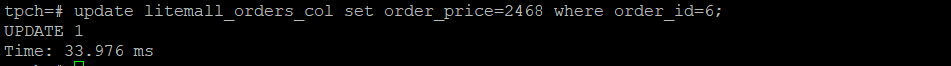


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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



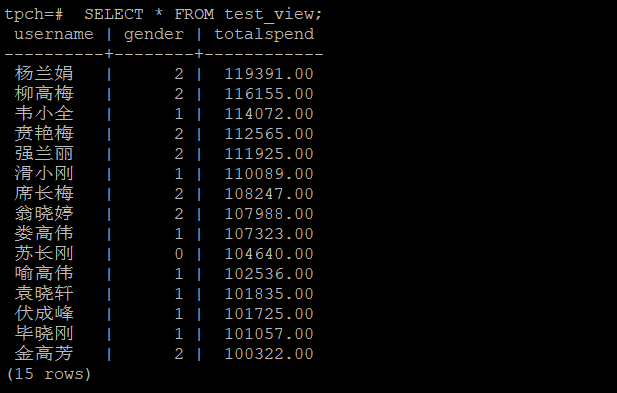
update litemall\_orders\_col set order\_price=2468 where order\_id=6;



任务三：物化视图的使用

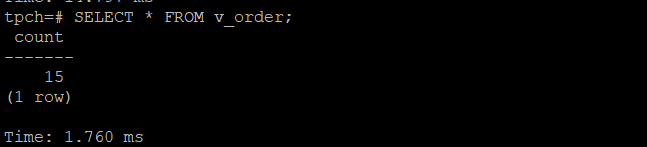
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



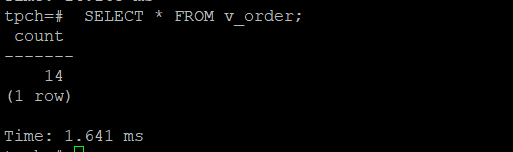
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



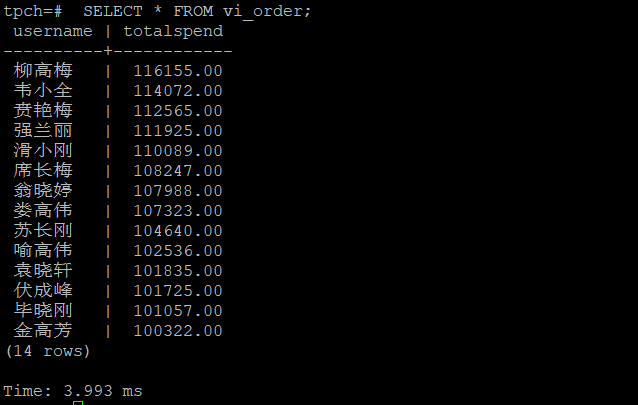
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



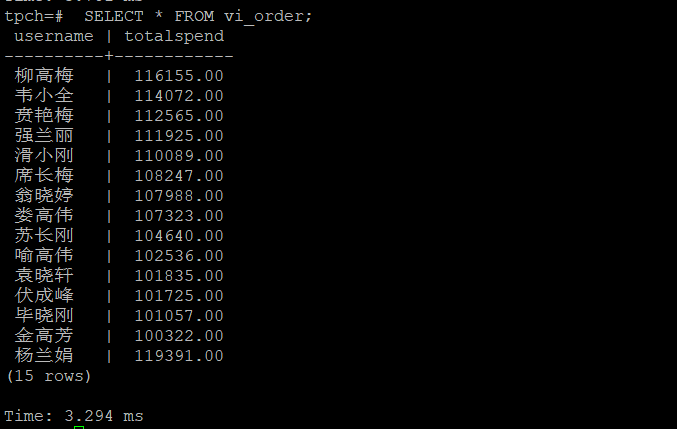
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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？

1. 行存表和列存表在执行相同sql语句时，执行时间不同的根本原因在于二者存储方式不同，行存表是将每一行数据存储到一起，列存表是将同一列数据存储到一起；
2. 通过上面实验发现，行存表在执行频繁更新数据的sql、同时查询多个表的sql以及需要查询少量数据的sql时，效率更高，包括INSERT、UPDATE、DELETE、JOIN、SELECT等语句，在上面实验中，查询特定值、修改值时行存表效率更高；
3. 通过上面实验发现，列存表在执行查询大量数据的sql、需要聚合计算的sql以及需要对某些列进行频繁查询的sql时，效率更高，包括SELECT、SUM、AVG、MAX、GROUP BY等语句，在上面实验中，求某一字段总和、某一字段平均值，列存表效率更高。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

全量物化视图与增量物化视图主要差异在于如何处理数据的更新。全量物化视图在每次刷新时，都会重新计算和存储所有的数据，因此需要更多的存储空间和计算资源，但可以提供更高的查询性能。增量物化视图只存储最近的更改，因此所需的存储空间和计算资源更少，查询性能也有所下降。

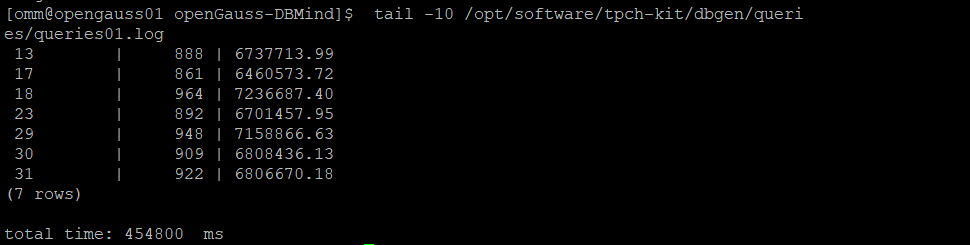
全量物化视图适用于数据变化不频繁的情况，而增量物化视图适用于数据变化频繁的情况。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

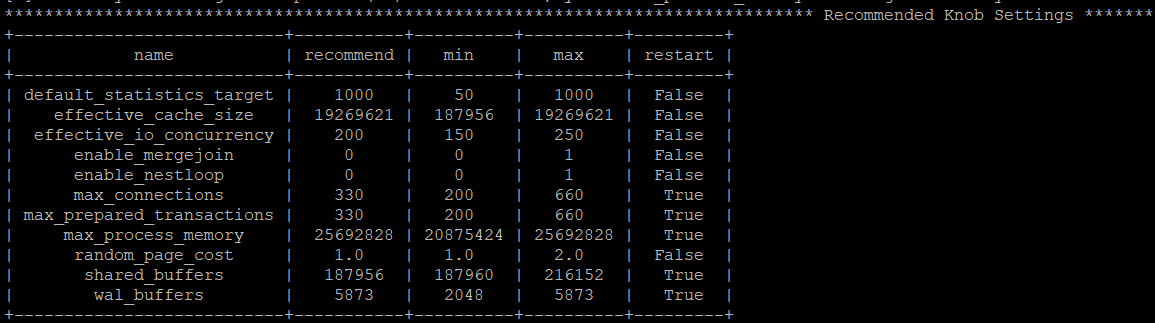
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

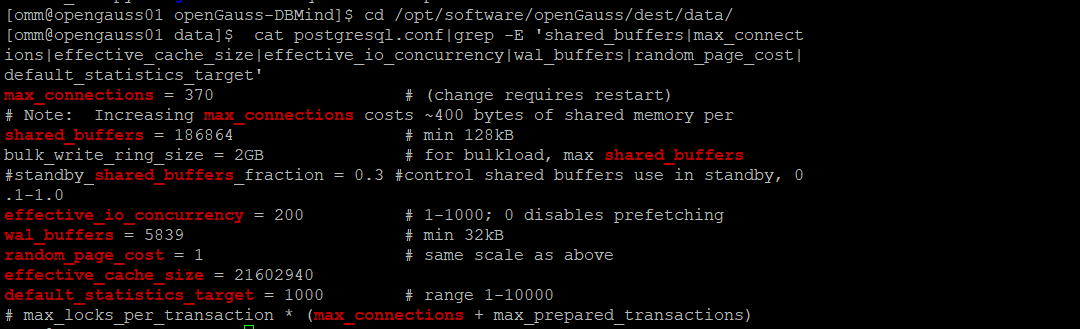
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

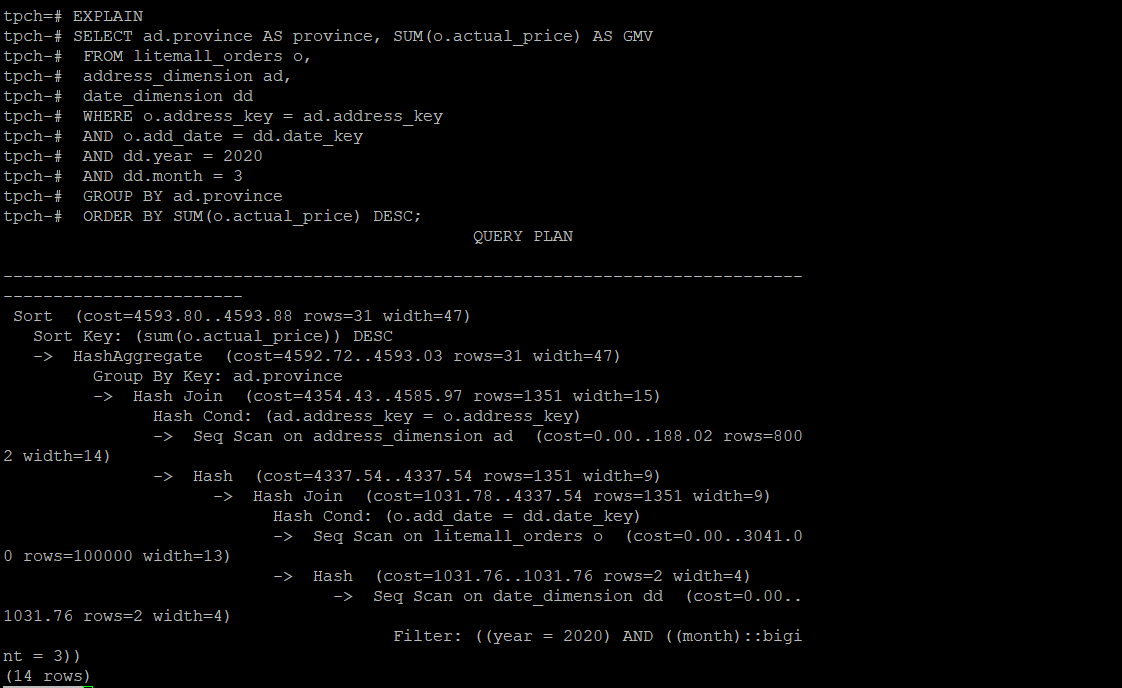
AND o.add\_date = dd.date\_key

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

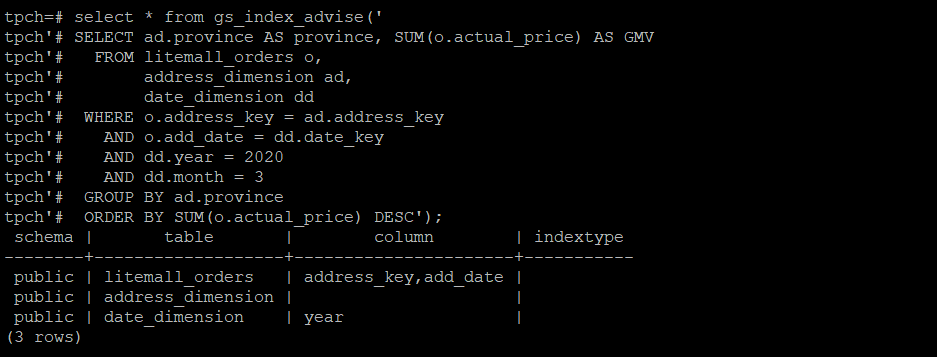
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

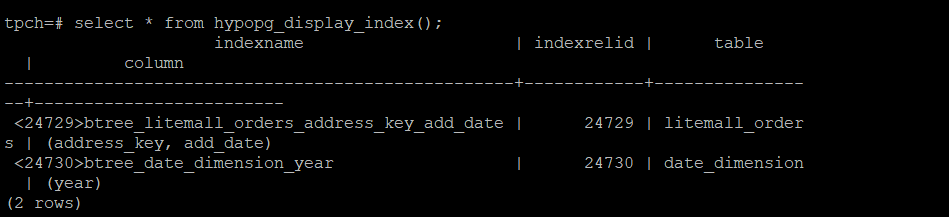
GROUP BY ad.province

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。

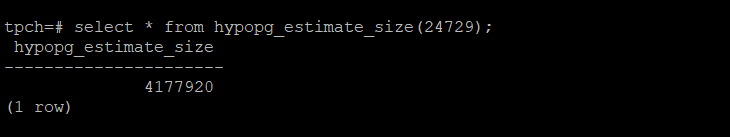
select \* from hypopg\_display\_index();

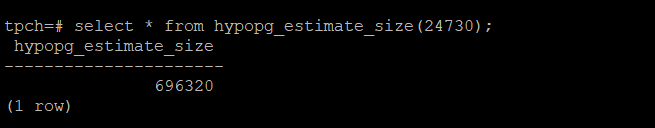


4. 获取索引虚拟列大小结果（单位为：字节），将执行结果截图。

select \* from hypopg\_estimate\_size(16715); //24729

select \* from hypopg\_estimate\_size(16716); //24730





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

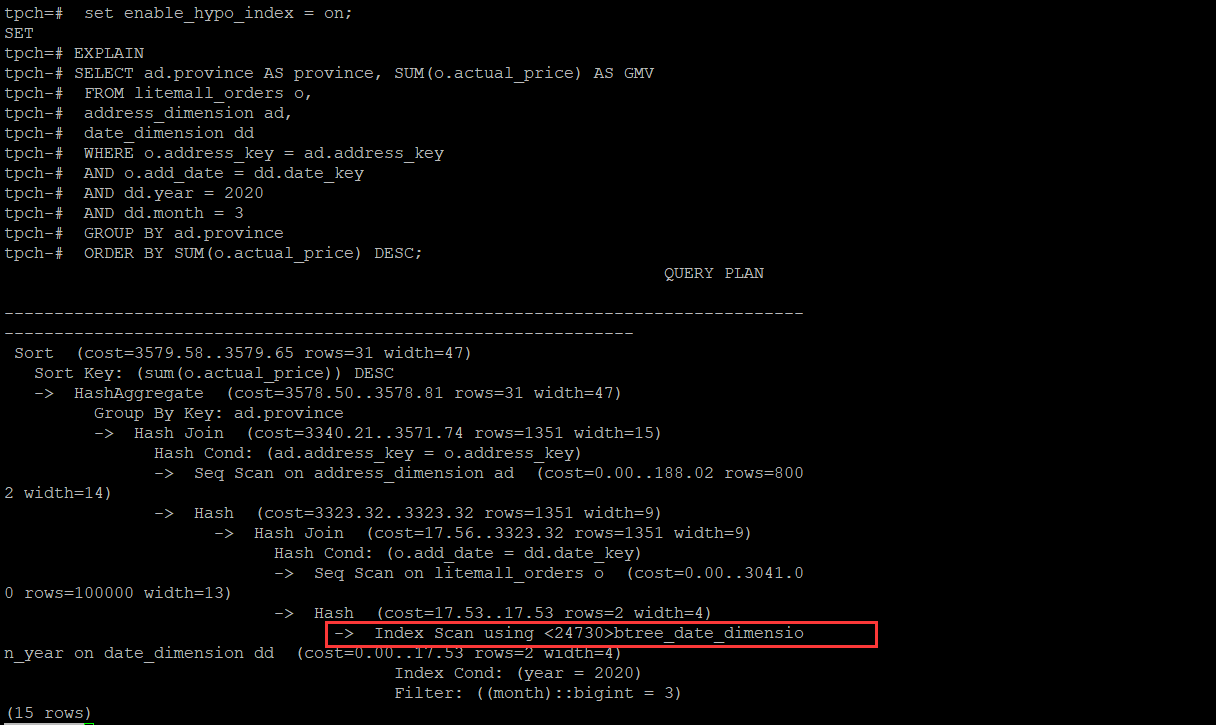
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

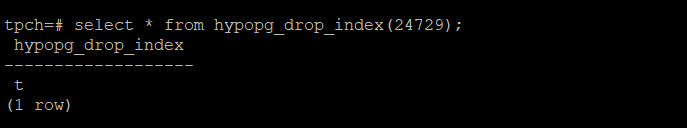
GROUP BY ad.province

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



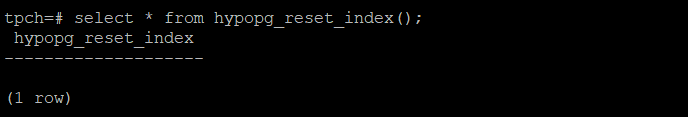
6. 删除某一个索引虚拟列，将执行结果截图。

select \* from hypopg\_drop\_index(16715); #24729



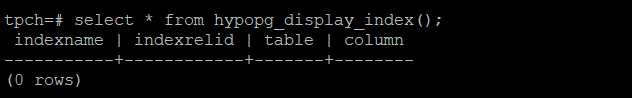
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

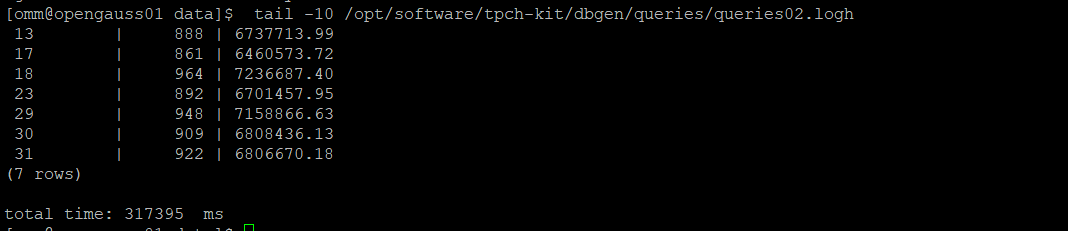
select \* from hypopg\_display\_index();



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：

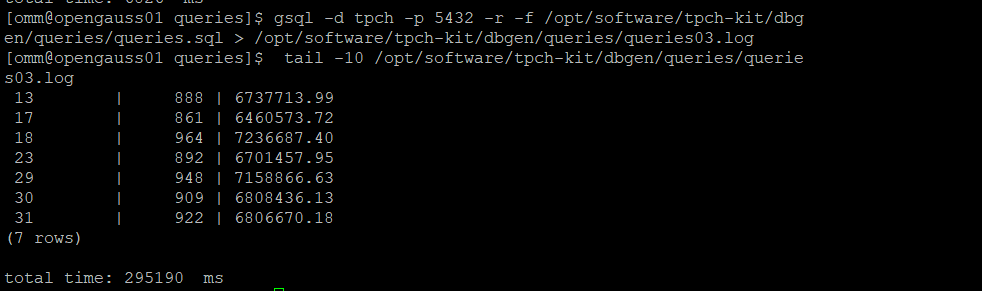
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log



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？

根据上面实验结果，X-Tuner会对缓冲池大小、最大连接数、查询缓存、日志文件大小、磁盘读写速度等参数进行优化。优化缓冲池大小可以减少磁盘I/O次数，从而提高查询性能；优化最大连接数可以提高并发访问能力，优化查询缓存可以减少重复的查询次数，优化读写参数可以提高磁盘I/O效率，减少人工干预和错误，提高数据库的稳定性和可靠性。

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？

当数据库中数据量很大时，使用索引可以提高查询性能。具体包括减少磁盘I/O次数，加速数据检索。索引可以快速定位查找数据，减少不必要的间接查询，因此可以加速查询，提升查询效率。

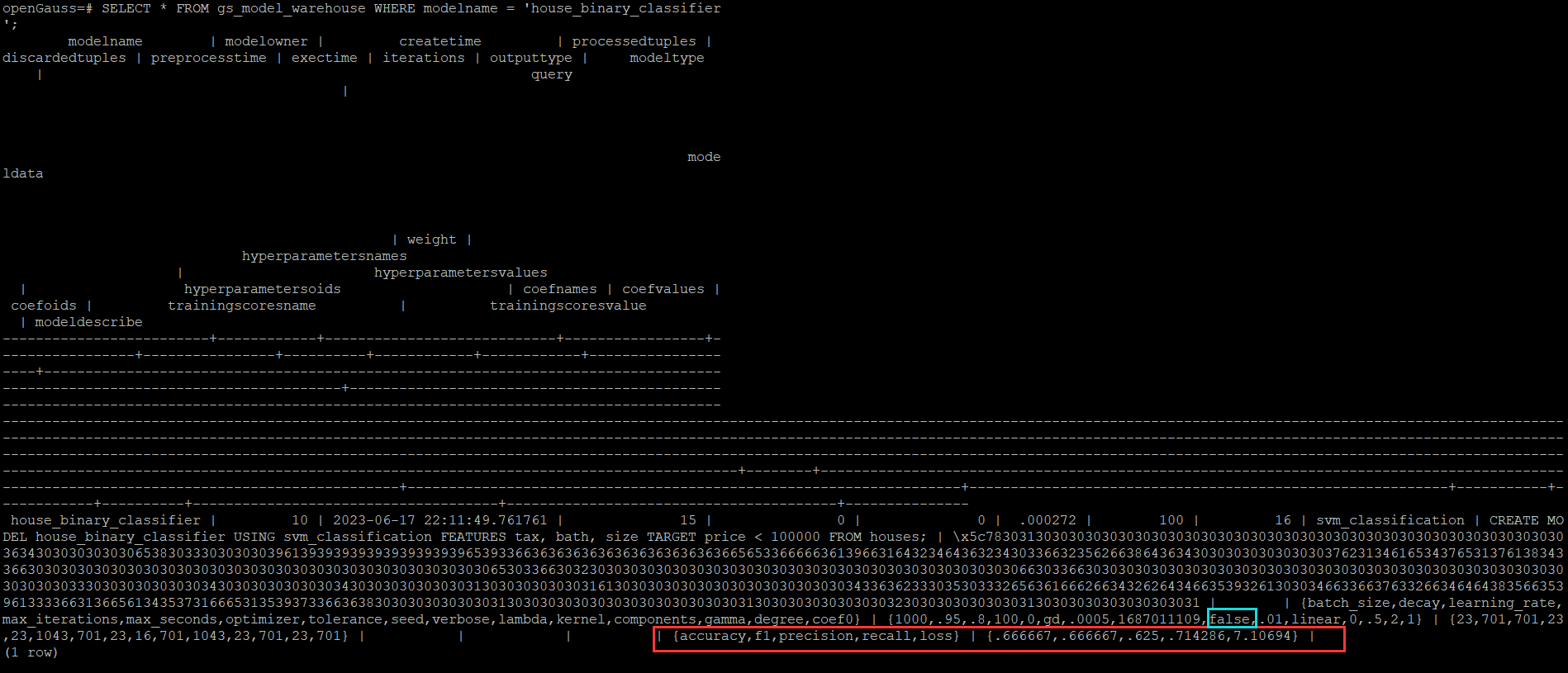
除了使用索引和参数外，还可以对数据库分区、硬件优化、SQL语句优化等角度对数据库优化。数据库分区可以提高并发访问能力，从而提高查询性能；硬件优化则是通过增加CPU、内存、磁盘等硬件资源，提高数据库的处理能力；SQL优化是通过调整SQL查询语句的结构和语法，减少查询时需要访问的数据量，进而提高查询性能。

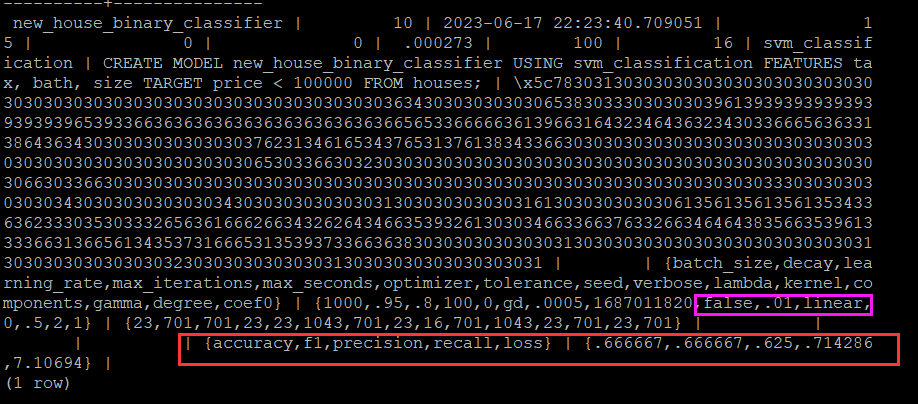
还可以利用openGauss的AI特性优化数据库，包括使用Slow Query Diagnosis对慢sql进行归因分析，用SQLdiag发现慢sql，进而对慢sql改进；也可以使用SQL Rewriter对SQL语句改写，提升语句执行性能。

# 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：

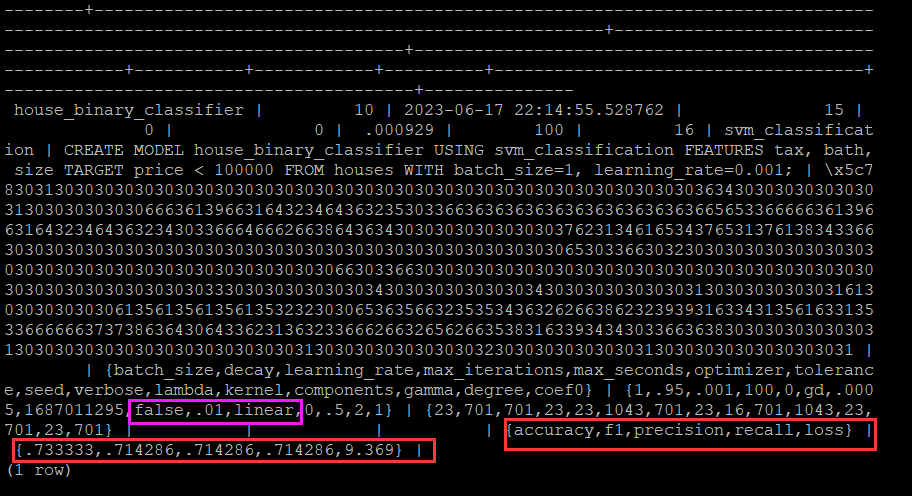
postgres=# SELECT \* FROM gs\_model\_warehouse WHERE modelname = 'house\_binary\_classifier';





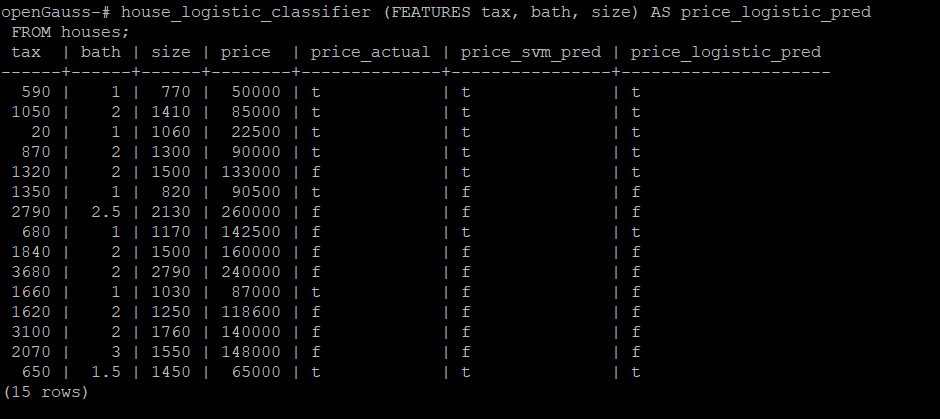
任务二：观察新模型的信息，将执行结果截图。

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？

分类模型与回归模型最根本的不同在于解决的问题不同。分类模型用于预测离散型变量，得到的是样本的离散值（类别）；回归模型用于预测连续型变量，得到的是连续值（例如价格、年龄等）。

实践思考题2：什么是SVM算法？

SVM算法属于监督学习算法的一种，核心思想在于构建一个最优超平面来将数据分为两个类别或用最优超平面拟合数据。SVM中每个样本都可以表示为一个向量，每个向量代表一个标签，SVM则优化一个目标函数找到最优平面，目标函数优化目标为间隔最大同时误分类点数量最少，间隔仅与支持向量有关，因此SVM算法具有更好的鲁棒性。

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？

分类问题评估指标，包括AUC、F1-Score、Precision、Recall、Acc、Loss。AUC是在随机抽取一对样本的条件下，预测得到正样本的概率大于负样本概率的概率，也可以是ROC曲线面积的衡量。ROC曲线是以真正率为纵轴，假正率为横轴绘制的曲线，可以衡量模型在不同阈值下的分类性能。AUC越接近1，代表模型分类效果越好。多分类中AUC的计算方式是ovo或者ovr，即根据不同划分方式得到多个“二分类”AUC结果，再求取平均值。Precision是指在模型预测为正样本中实际为正样本的比例，值越大代表模型预测为正样本的准确性越高；Recall是指实际为正样本中被模型预测为正样本的比例，值越大代表对正样本的覆盖率越高。F1-Score是Precision和Recall的调和平均数，可用于衡量分类模型的综合性能，即不同模型在同一数据集上的性能。

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？

回归问题评估指标包括r2分数、rmse、mae等。r2分数用于衡量模型对目标变量的解释能力，评估回归模型拟合度优度，其取值范围为0到1，越接近1表示模型的拟合程度越好。rmse是均方根误差，衡量了预测值与真实值之间的平均误差的标准方差，其值越小代表模型预测误差越小。mae是平均绝对误差，衡量了预测值与真实值之间的平均绝对误差，其值越小代表模型预测误差越小。rmse易受异常值影响，而mae对异常值不敏感。r2分数、rmse、mae更新公式如下：







其中为真实值的均值，为样本预测值，为样本真实值。