openGauss AI特性创新实践课



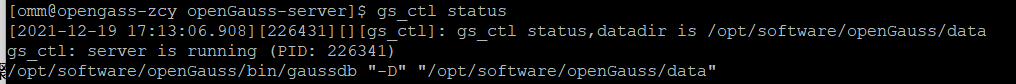
华为技术有限公司

# 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：

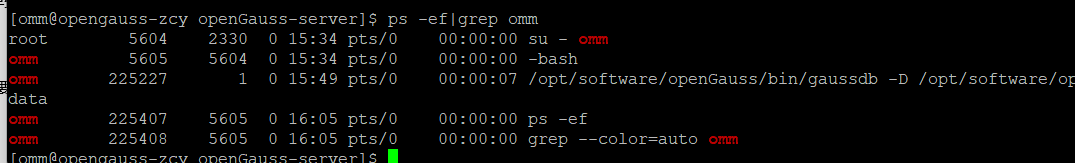
任务一：数据库状态验证

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？

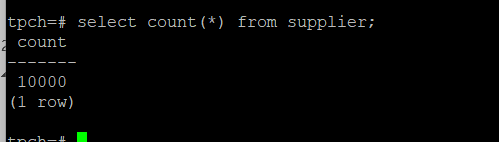
满足不同的运行平台，Linux发行版本众多，但是每个版本采用的软件或者内核版本不一样，而二进制包所依赖的环境不一定能正常运行，所以大部分软件直接用源码进行安装。

# 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：

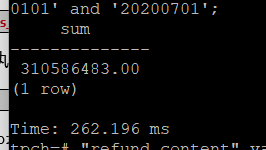
select count(\*) from supplier;;



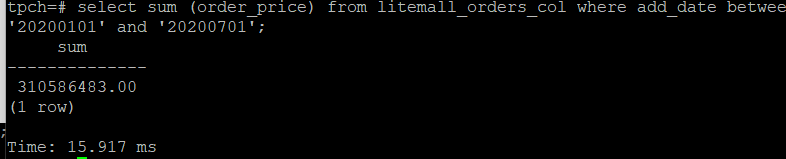
任务二：行存表与列存表执行效率对比

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';

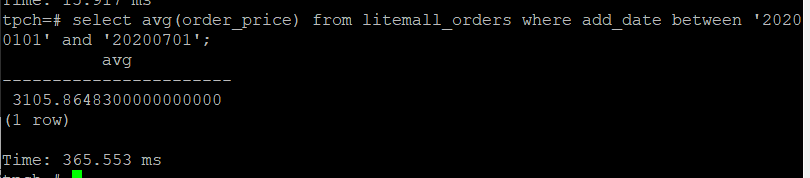


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

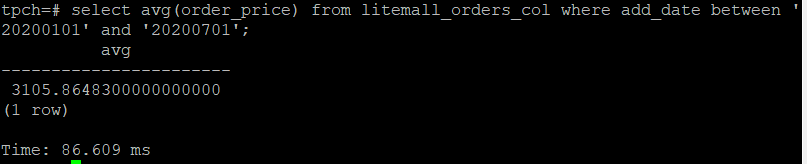


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

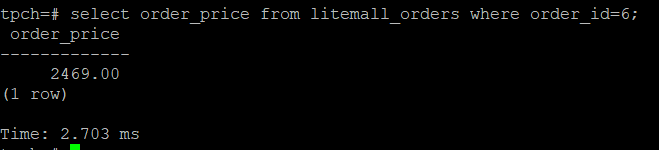


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

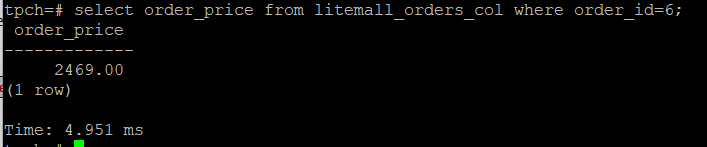


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

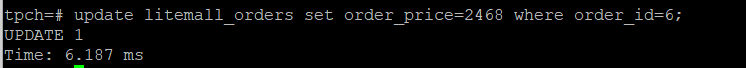


select order\_price from litemall\_orders\_col where order\_id=6;



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;



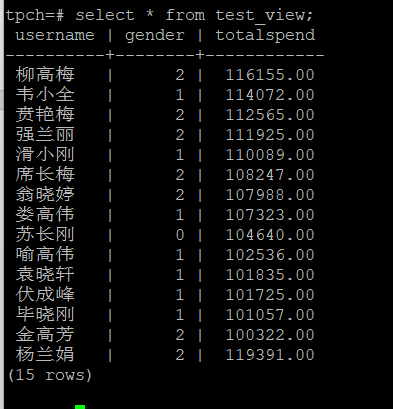
update litemall\_orders\_col set order\_price=2468 where order\_id=6;



任务三：物化视图的使用

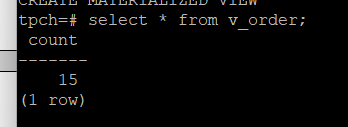
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



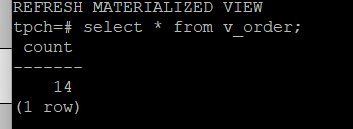
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



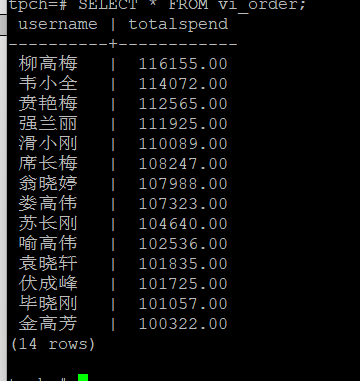
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

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某些相同的SQL语句时，执行效率不同，在执行的SQL涉及到多行数据时，列存表速度更快，甚至是行存表的几倍，但是执行只涉及到一行数据时，行存表更快。

1.存储方式不一样，行存储是指将表按行存储到硬盘分区上，即一行数据是连续存储。列存储是指将表按列存储到硬盘分区上，即一列所有数据是连续存储的。

2.行存表支持完整的增删改查。适用于对业务数据频繁更新场景。列存表主要应用于数据追加和分析场景

实践思考题2：全量物化视图与增量物化视图有哪些差别？

物化视图：是包括一个查询结果的数据库对象，它是远程数据的本地副本

全量物化视图仅支持对创建好的物化视图做全量更新，而不支持做增量更新。创建全量物化视图语法和CREATE TABLE AS语法一致，不支持对全量物化视图指定NodeGroup创建。

增量物化视图顾名思义就是可以对物化视图增量刷新，需要用户手动执行语句完成对 物化视图在一段时间内的增量数据进行刷新。

全量物化视图仅支持对创建好的物化视图做全量更新，而不支持做增量更新

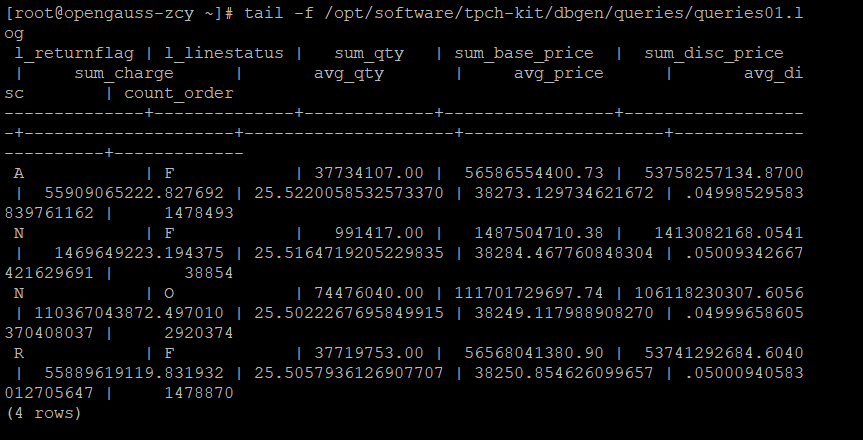
增量物化视图可以对物化视图增量刷新

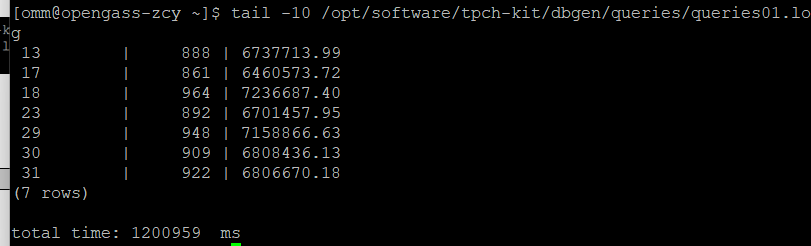
# 关卡三、openGauss的AI4DB特性应用

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：

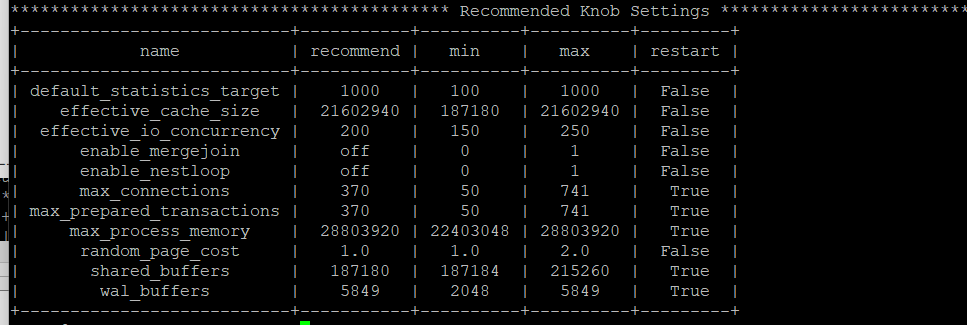
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log





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

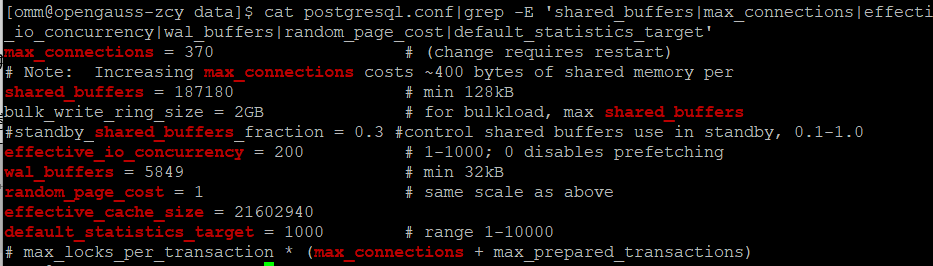
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

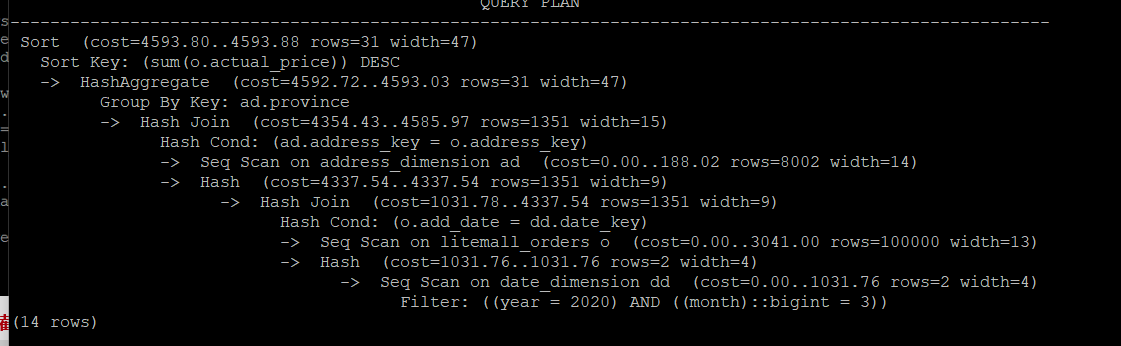
AND o.add\_date = dd.date\_key

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

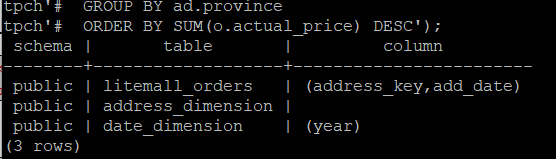
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

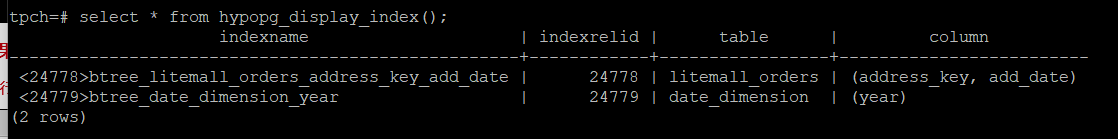
GROUP BY ad.province

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。

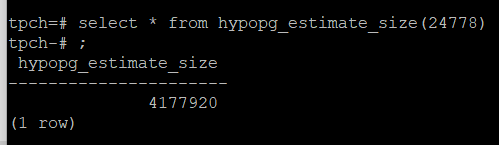
select \* from hypopg\_display\_index();

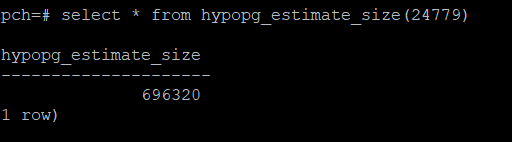


4. 获取索引虚拟列大小结果（单位为：字节），将执行结果截图。

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

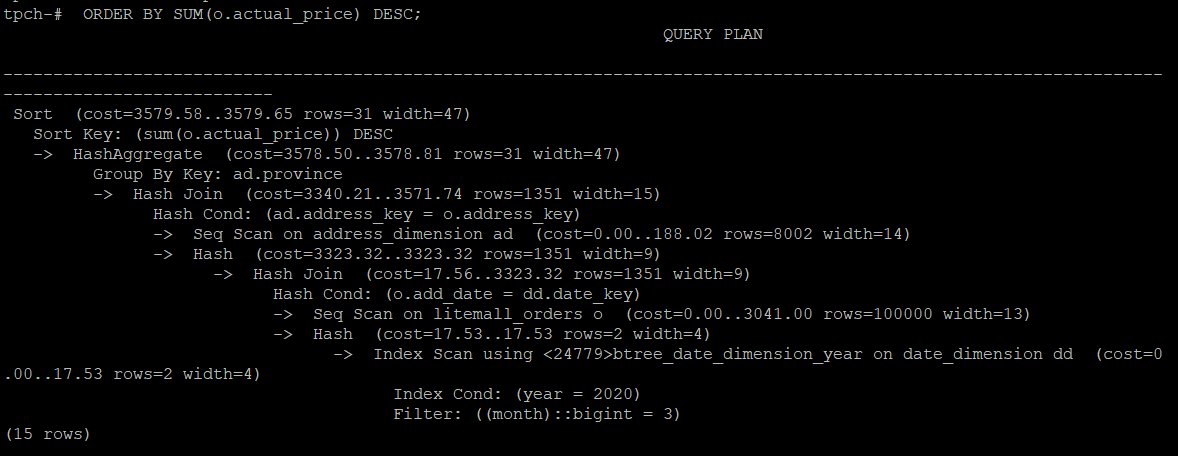
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

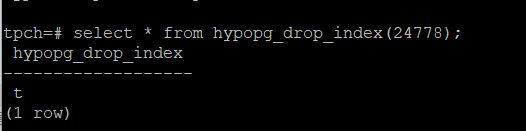
GROUP BY ad.province

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



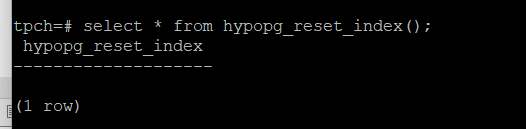
6. 删除某一个索引虚拟列，将执行结果截图。

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



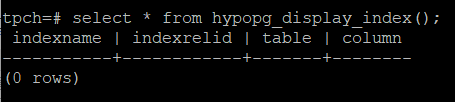
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

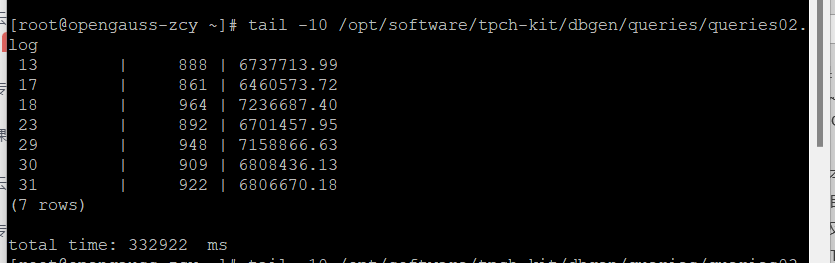
select \* from hypopg\_display\_index();



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。

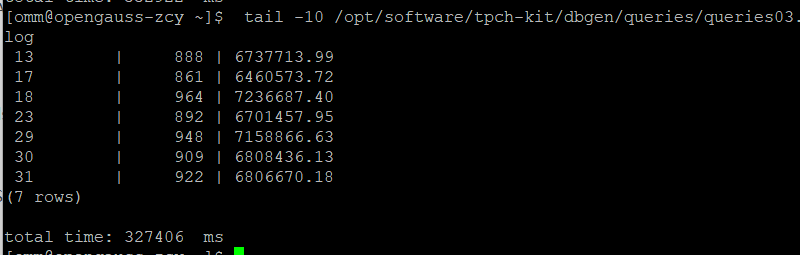
1. 重新执行queries.sql查询，将执行结果截图：

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？

shared\_buffers共享缓冲区

max\_connections最大连接数

effective\_cache\_size有效缓存大小

effective\_io\_concurrency有效io并发

wal\_buffers wal\_缓冲区

random\_page\_cost 随机页面成本

default\_statistics\_target 默认系统目标

使数据库效率更高。

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？

1.通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。

2.可以大大加快数据的检索速度，这也是创建索引的最主要的原因。

3.可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义。

4.在使用分组和排序子句进行数据检索时，同样可以显著减少查询中分组和排序的时间。

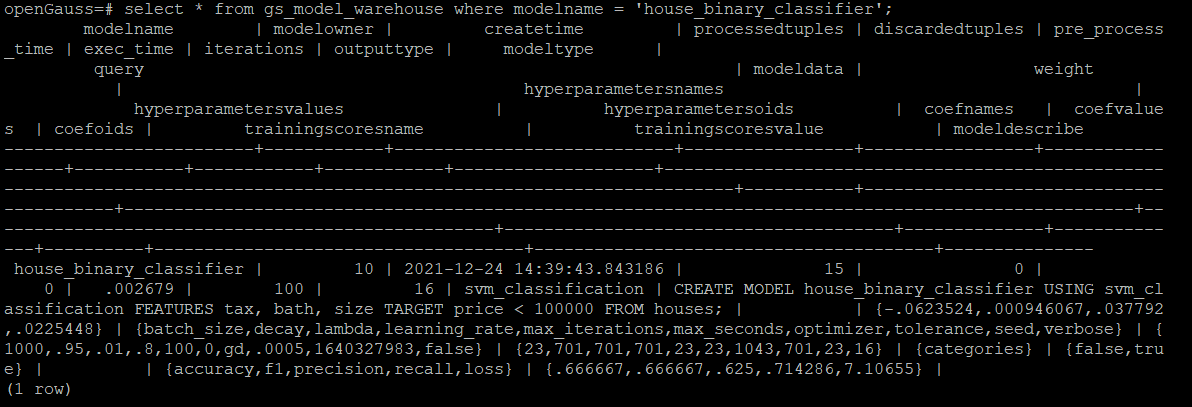
5.通过使用索引，可以在查询的过程中，使用优化隐藏器，提高系统的性能。

除此之外，还可以优化表结构，优化SQL语句等

# 关卡四、openGauss的DB4AI特性应用

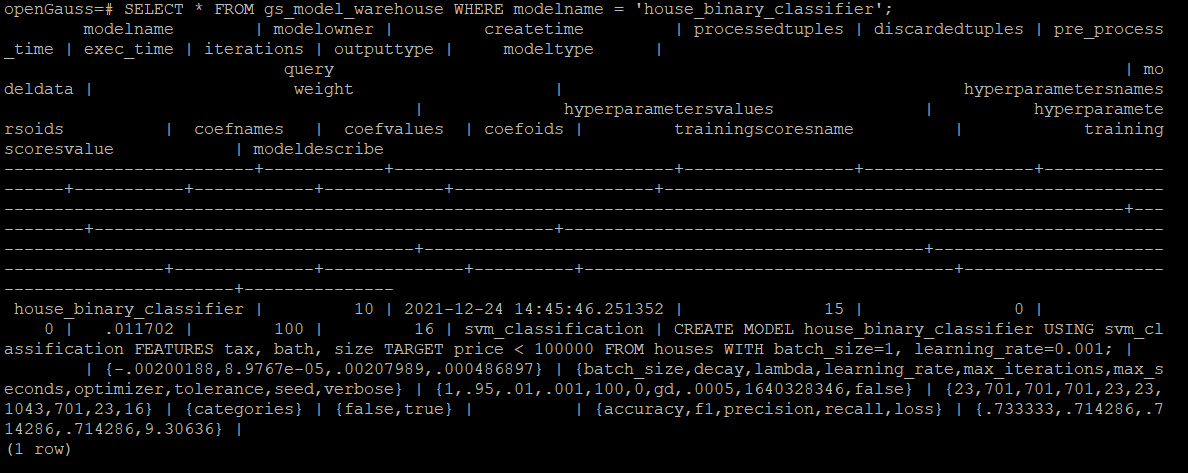
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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

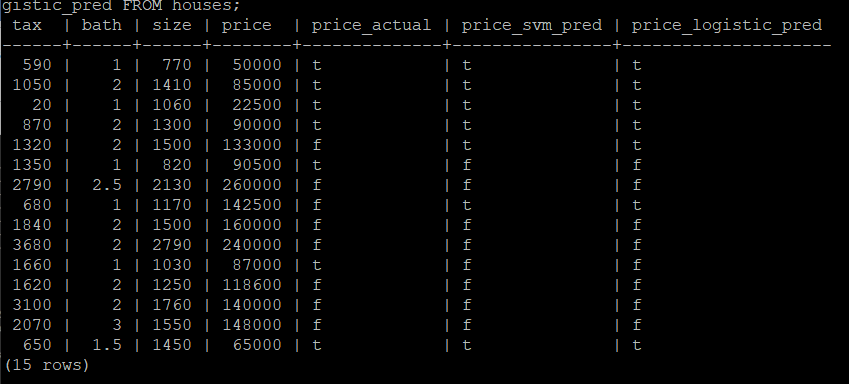


任务二：观察新模型的信息，将执行结果截图。

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？

回归会给出一个具体的结果，例如房价的数据，根据位置、周边、配套等等这些维度，给出一个房价的预测。

 分类相信大家都不会陌生，生活中会见到很多的应用，比如垃圾邮件识别、信用卡发放等等，就是基于数据集，作出二分类或者多分类的选择。

浅层： 两者的预测目标变量类型不同，回归问题是连续变量，分类问题离散变量。

中层： 回归问题是定量问题，分类问题是定性问题。

高层： 回归与分类的根本区别在于输出空间是否为一个度量空间。

定量输出称为回归，或者说是连续变量预测

定性输出称为分类，或者说是离散变量预测

实践思考题2：什么是SVM算法？

SVM 支持向量机（Support Vecor Machine）是一种二分类模型，将实例饿特征向量映射为空间中的一些点，SVM的目的就是想要画出一条线，区分这两类点。适合中小型数据样本、非线性、高维的分类问题。

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？

准确率(Accuracy)：分类正确的样本数量与总样本数量的比例

FPR（False Positive rate）：所有实际为假的样本中有百分之多少被系统错分为真了

FNR(False Negative rate)：是所有实际为真的样本中有百分之多少被系统错分为假了

Recall(True Positive rate)：所有实际为真的样本中有百分之多少被系统正确分为真了

Precision：系统预测为真的样本中有百分之多少实际上也是真的

F-score：不同领域使用不同标准，在信息检索中常用到F-score是Recall和Precision的调和平均。Fβ=（β2+1）·P·R/（β2P+R）

MAP(Mean Average Precision)：用于可以给出候选预测结果排序（或者分数）的系统

ROC曲线 ：平面的横坐标是false positive rate(FPR)，纵坐标是true positive rate(TPR)。对某个分类器而言，我们可以根据其在测试样本上的表现得到一个TPR和FPR点对。这样，此分类器就可以映射成ROC平面上的一个点。

AUC： ROC曲线与坐标轴围成的面积

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？

MSE(mean square error)：是预测值和真实值之差的平方的均值

MAE(Mean Absolute Error)：测值和真实值之差的绝对值求均值

PCC/CC 相关系数，又称皮尔逊相关系数