openGauss AI特性创新实践课

姓名：**张敬泽**

学号：**1820181080**

班级：**08011802**



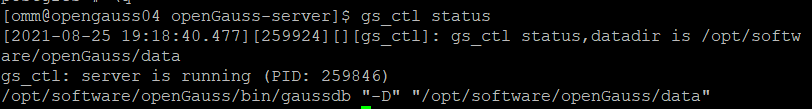
华为技术有限公司

# 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：

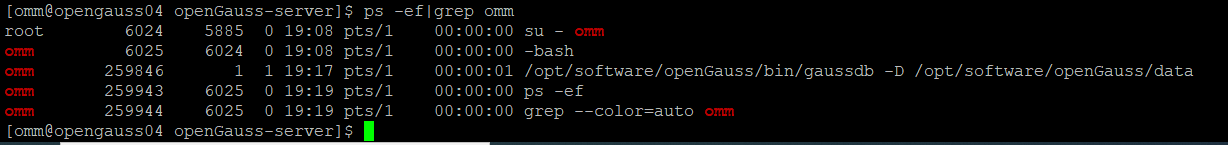
任务一：数据库状态验证

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）



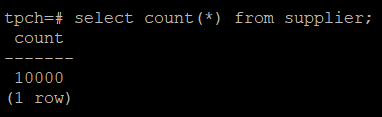
实验思考题：为什么需要通过源码编译，安装数据库？

# 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：

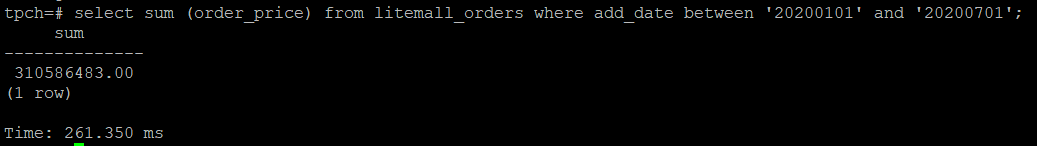
select count(\*) from supplier;;



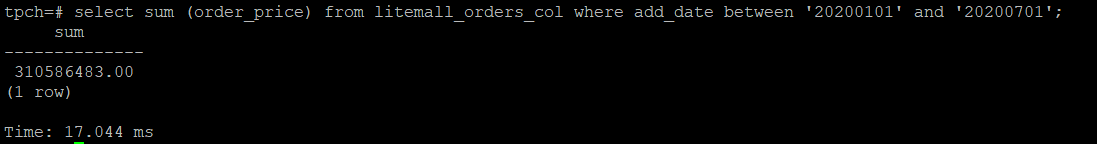
任务二：行存表与列存表执行效率对比

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';

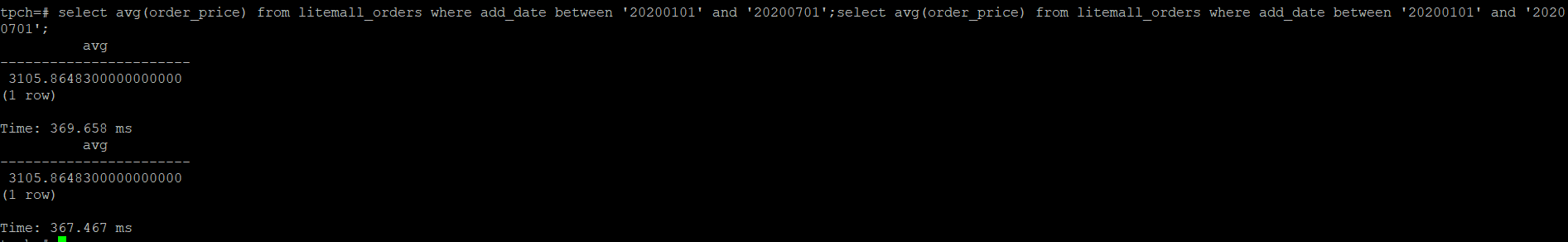


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

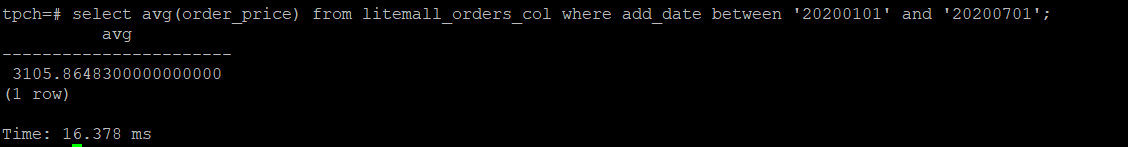


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

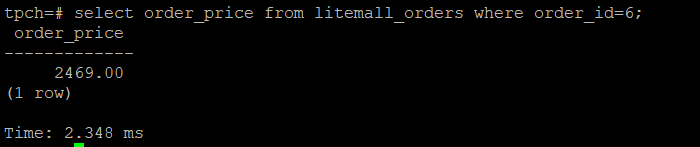


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

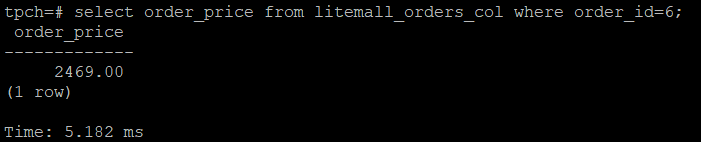


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

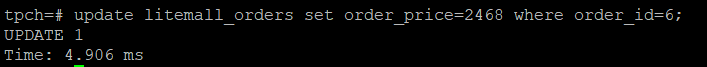


select order\_price from litemall\_orders\_col where order\_id=6;

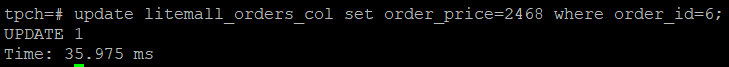


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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。

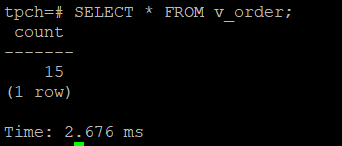
SELECT \* FROM v\_order;

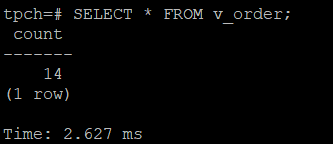
Text

Description automatically generated

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;

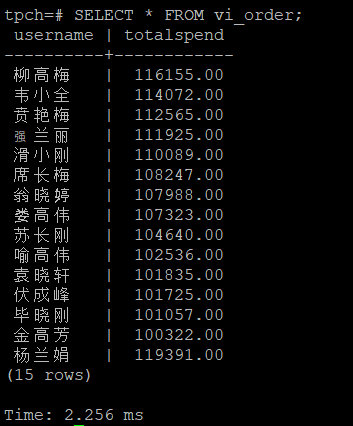




4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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？

行存储的写入是一次性完成，而列存储每次读取的数据是集合中的一段或者全部，因此在SQL语句中行存表消耗的时间比列存表少。

行存表：行存储的写入是一次性完成，消耗的时间比列存储少，并且能够保证数据的完整性，适用于少量数据

列存表：在读取过程，不会产生冗余数据，适用于对数据完整性要求不高的大数据处理领域（互联网）

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。与全量创建物化视图不同在于目前增量物化视图所支持场景较小，目前物化视图创建语句仅支持基表扫描语句或者UNION ALL语句。

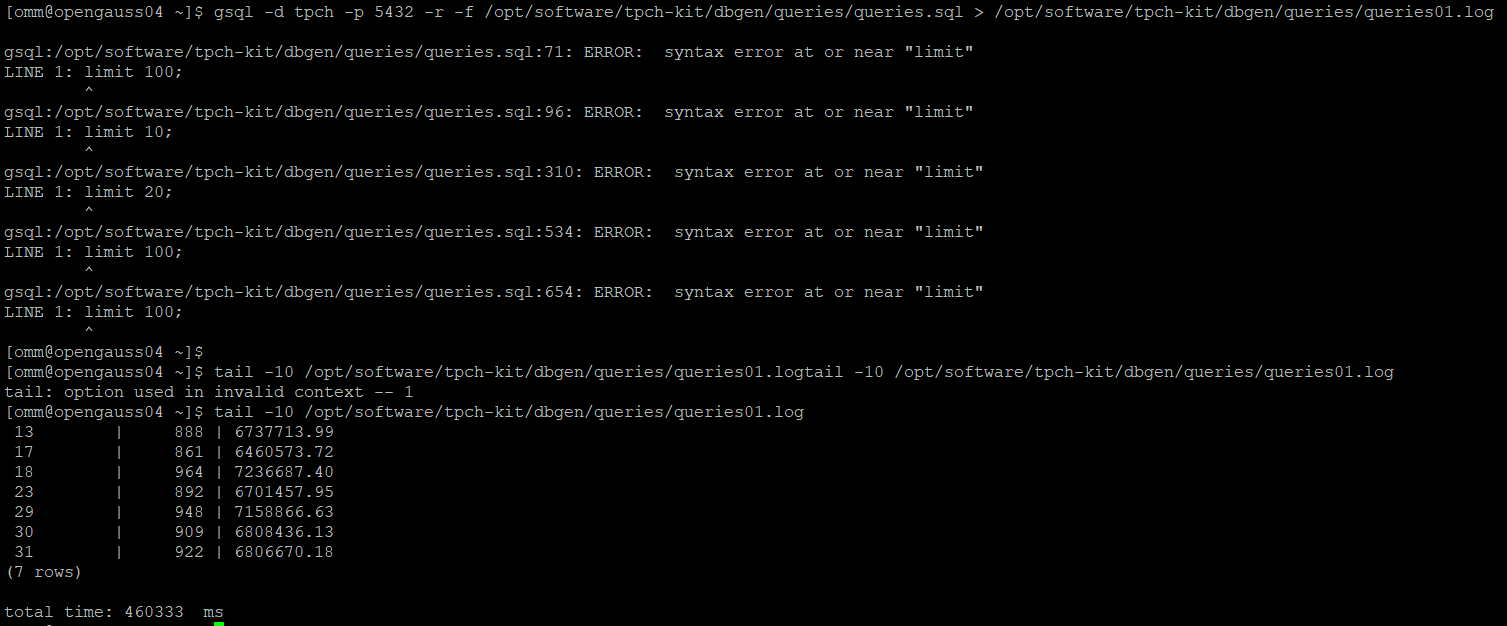
两者区别在于增量物化视图适用于场景较小的地方，而全量物化视图仅支持及表扫描或UNION ALL语句

# 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

Text

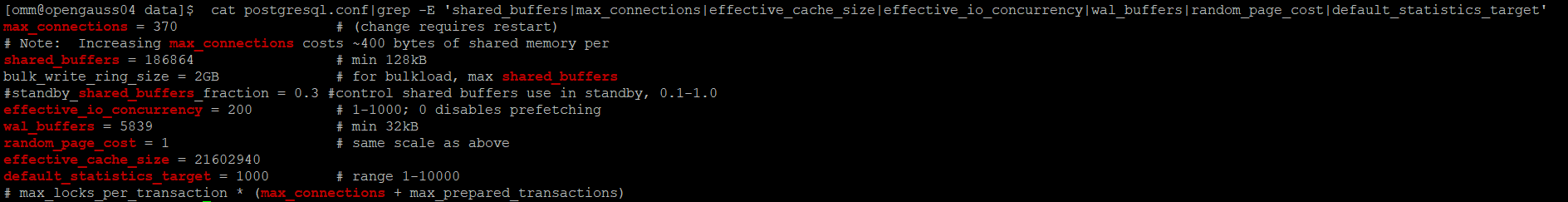
Description automatically generated A picture containing graphical user interface

Description automatically generated

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

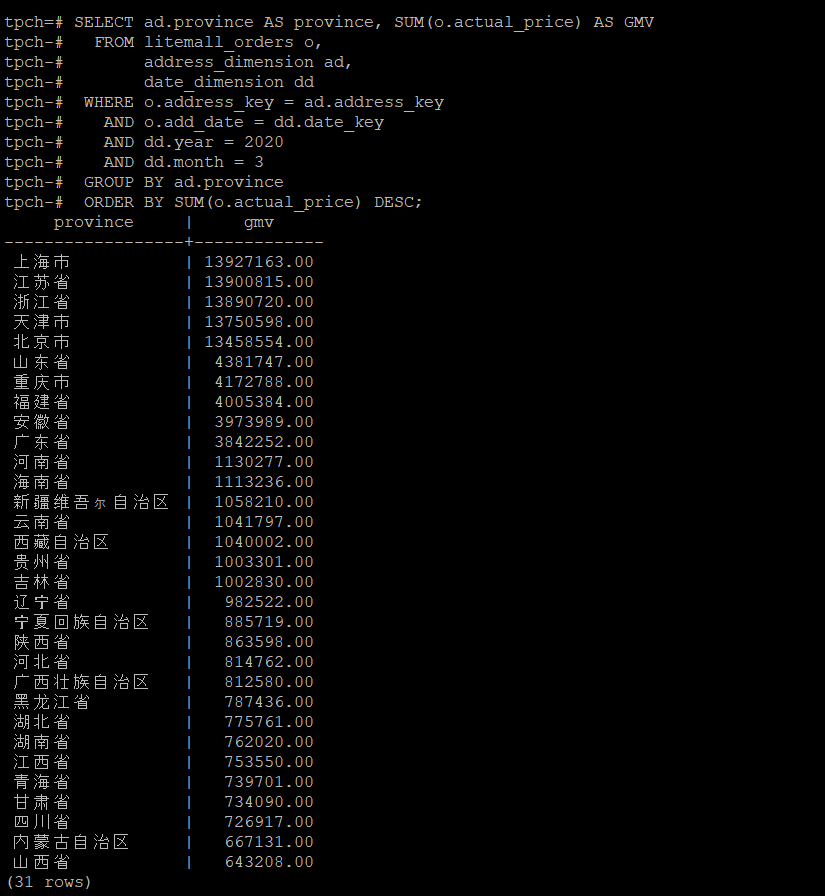
AND o.add\_date = dd.date\_key

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

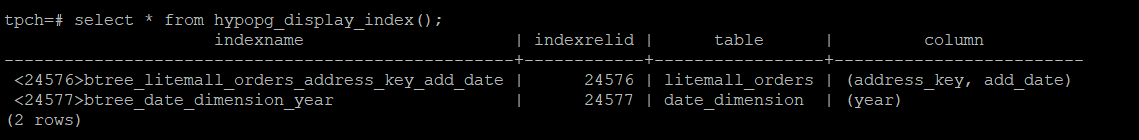
ORDER BY SUM(o.actual\_price) DESC');

Text

Description automatically generated

3. 查看创建的虚拟索引列，将执行结果截图。

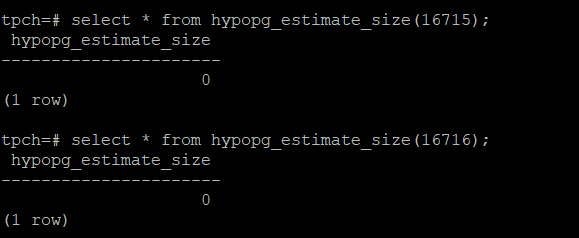
select \* from hypopg\_display\_index();



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

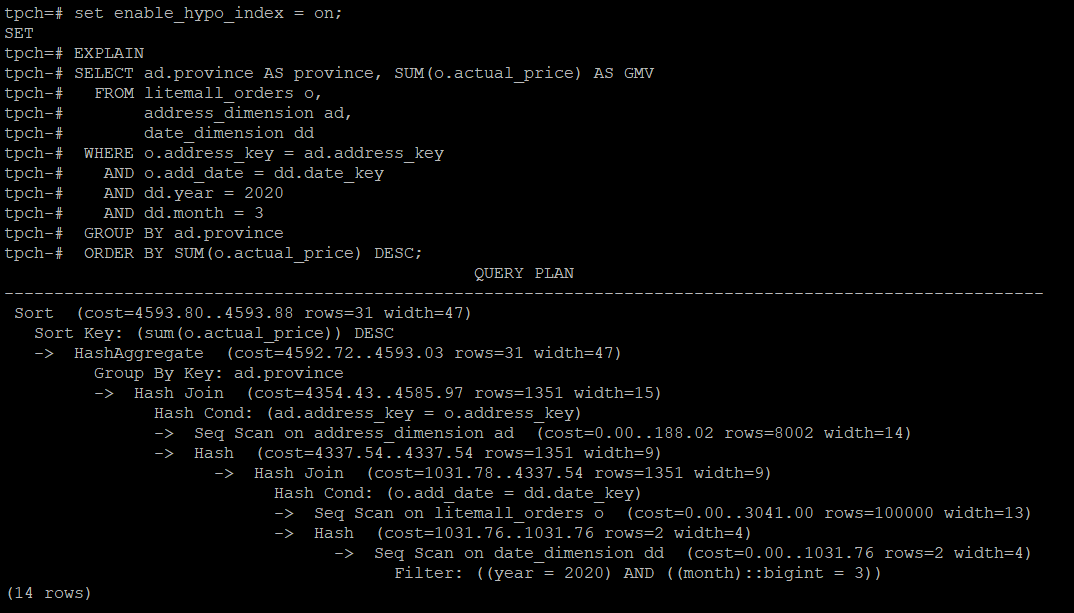
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

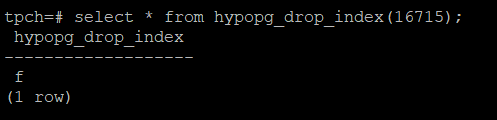
GROUP BY ad.province

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



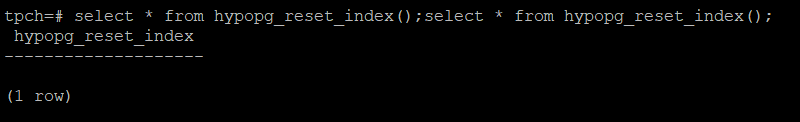
6. 删除某一个索引虚拟列，将执行结果截图。

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



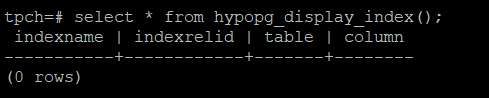
7. 删除某一个索引虚拟列，将执行结果截图。

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

Text

Description automatically generated

Text

Description automatically generated

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

A computer screen capture

Description automatically generated with low confidence

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？

A picture containing graphical user interface

Description automatically generated

在TPC-H数据库的表中缺少了索引，如果未对数据库进行任何的优化，这样数据库的运行效率会明显的比已优化的效率来的差。对参数进行优化可以提升数据库提升处理数据的性能与效率。

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？

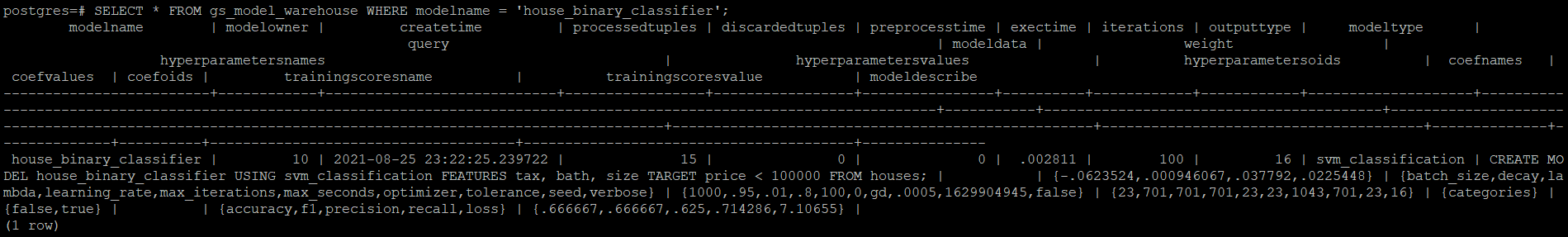
索引在执行SQL中的好处可以加快数据的搜索效率，加快表之间的联通，并且减少分组与排序时间和使用优化隐藏器来以提高系统性能。它也可以通过唯一性索引 (unique) 来确保数据的唯一性。

此外，我们可以通过SQL可通过SQL语句以及有效索引、数据结构、系统配置和硬件来对数据库进行优化。如通过需求分析来结构一个良好的SQL语句在根据SQL矮表中建立有效的索引。并且在设计表结构时，也不忘对结构设计出一个良好和有效的查询架构。其次是系统配置优化。Taishan是openGauss数据库的主要服务器，它采用高性能低功耗的华为鲲鹏ARM64处理器，通过openEulerOS的搭配，可以为数据库优化带来显著的效果。最后是硬件配置的优化，通过openGauss提供的SSD服务器硬盘，它在处理速度上比传统的HDD存/读上拥有高达5倍之多，提升硬件设备也是对数据库进行优化的首选之一。

# 关卡四、openGauss的DB4AI特性应用

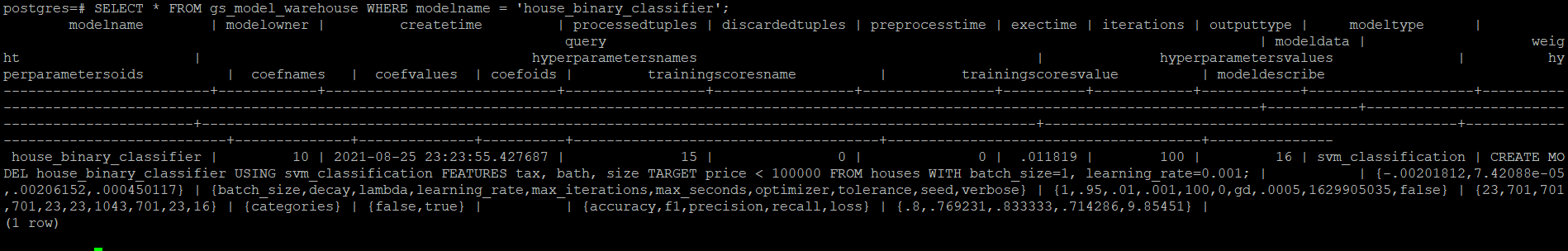
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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？

定量输出称为回归，或者说是连续变量预测,可用于测量气温温度

定性输出称为分类，或者说是离散变量预测，可用于判断天气类型

实践思考题2：什么是SVM算法？

支持向量机（SVM）是一种二分类模型，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；SVM还包括核技巧，这使它成为实质上的非线性分类器。

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**

预测正确的结果占总样本的百分比

**精确率**

所有被预测为正的样本中实际为正的样本的概率

通常以关注的类为正类，其他类为负类，分类器在数据集上的预测或者正确或者不正确，我们有4中情况：

* TP：True Positive， 把正类预测为正类；
* FP：False Positive，把负类预测为正类；
* TN：True Negative， 把负类预测为负类；
* FN：False Negative，把正类预测为负类

|  |  |  |
| --- | --- | --- |
| 预测值 实际值 | 正 | 负 |
| 正 | TP | FN |
| 负 | FP | TN |

**召回率**

实际为正的样本中被预测为正样本的概率

**F1分数**

把精确率（Precision）和召回率（Recall）之间的关系用图来表达，就是下面的**PR曲线**

综合两者的表现，在两者之间找一个平衡点，就出现了一个 F1分数

**P-R曲线 (Precesion-Recall Curve)**

对于所有正样本，用来描述精准率和召回率变化的曲线

**AUC（Area Under Curve）**

AUC被定义为ROC曲线下的面积 (ROC的积分) ，通常在0.5至1之间 。随机挑选一个正样本以及一个负样本，分类器判定正样本的值高于负样本的概率就是 AUC 值。AUC值的分类器越大，其性能就越好

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？

**MSE（Mean Square Error）均方误差**

MSE是真实值与预测值的差值的平方然后求和平均。通过平方的形式便于求导，所以常被用作线性回归的损失函数。

**MAE（Mean Absolute Error）平均绝对误差**

是绝对误差的平均值。可以更好地反映预测值误差的实际情况。

**RMSE（Root Mean Square Error）均方根误差**

衡量观测值与真实值之间的偏差。常用来作为机器学习模型预测结果衡量的标准

**R Squared (R2 Score)**

R Squared可称呼为可决系数 (coefficient of determination) 也叫拟合优度, 反映的是自变量x对因变量y的变动的解释的程度。 越接近于1,说明模型拟合得越好。常用此评价指标在中回归树上。