openGauss AI特性创新实践课



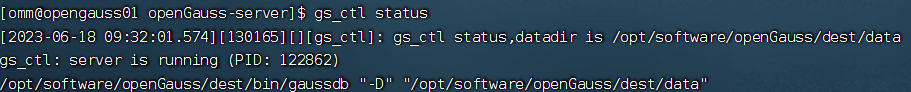
华为技术有限公司

# 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：

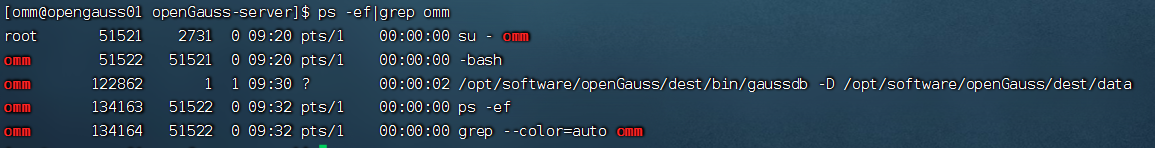
任务一：数据库状态验证

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？

在编译时，可以指定编译选项来对数据库进行优化，以提高性能。

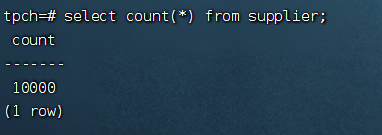
如果在使用数据库时遇到问题，源码编译可以方便地进行调试。

# 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：

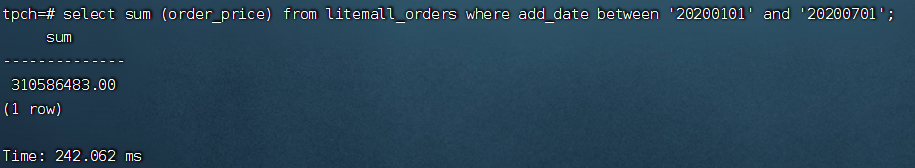
select count(\*) from supplier;;



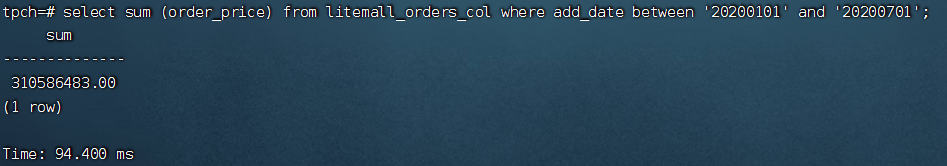
任务二：行存表与列存表执行效率对比

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';

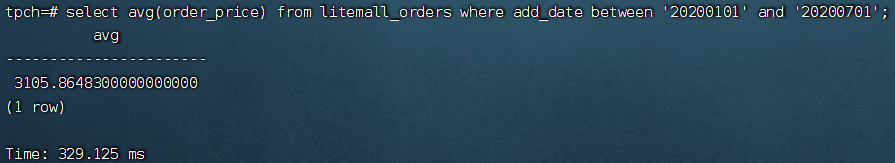


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

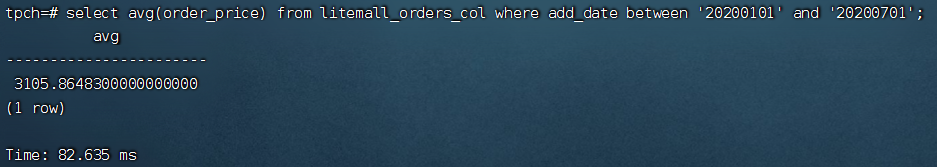


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

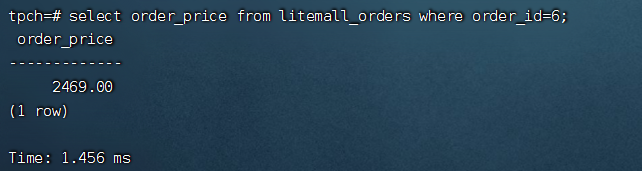


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

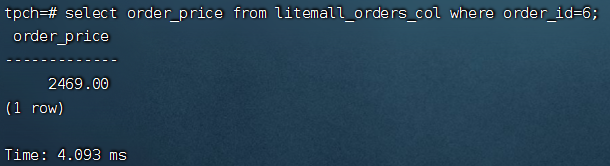


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

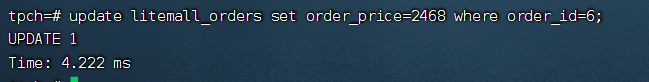


select order\_price from litemall\_orders\_col where order\_id=6;

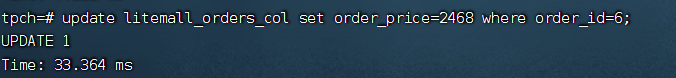


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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

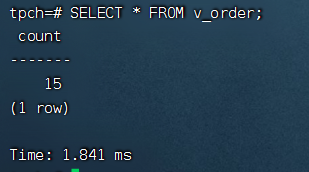
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



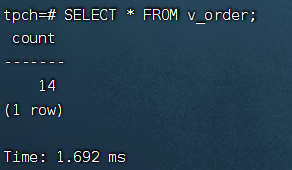
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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？

物理存储方式不同，行存表通过按行存储数据，适合于经常需要访问完整记录的应用场景。而列存表则是将每个列单独存储，这种存储方式适合于大量同一列的数据查询，

实践思考题2：全量物化视图与增量物化视图有哪些差别？

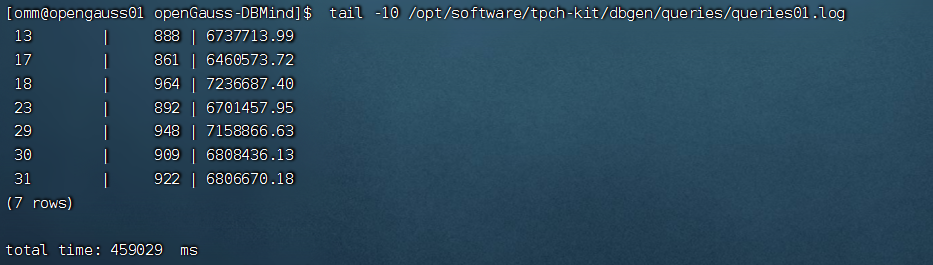
全量物化视图在每次刷新时重新计算所有数据，它们的计算是基于原始表中的数据。而增量物化视图在每次刷新时只计算新增、修改、删除等变化的数据

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

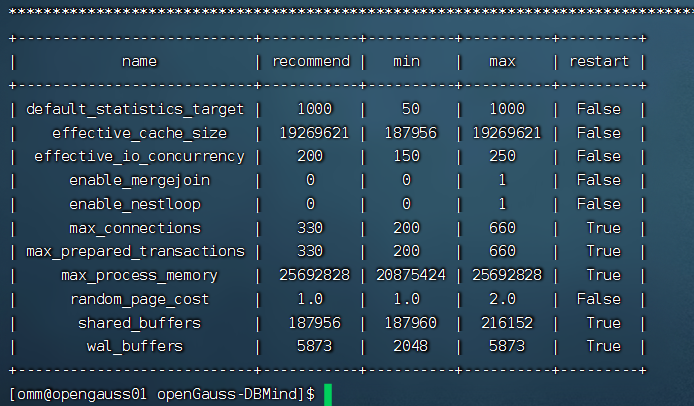
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

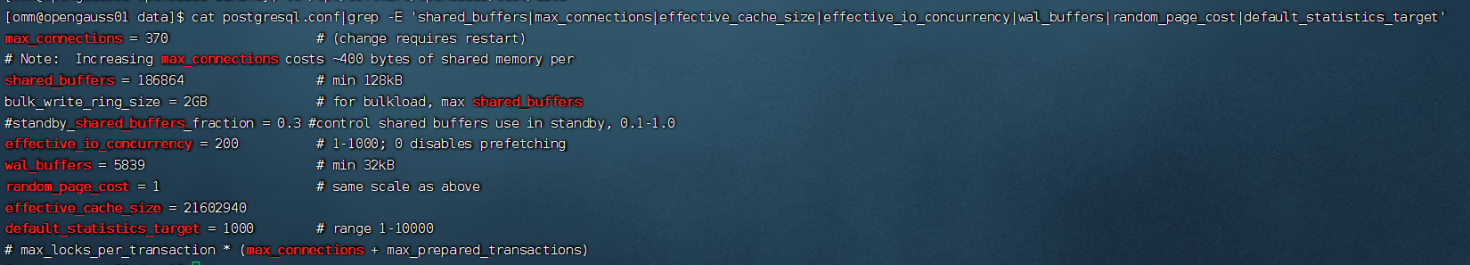
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

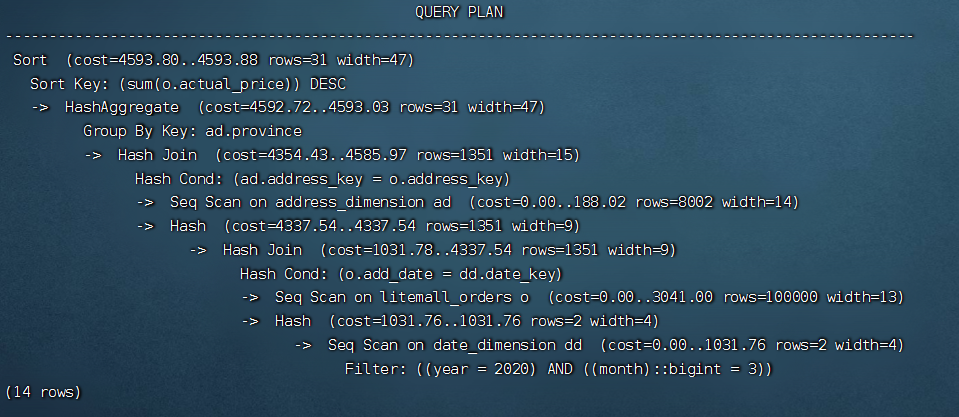
AND dd.year = 2020

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

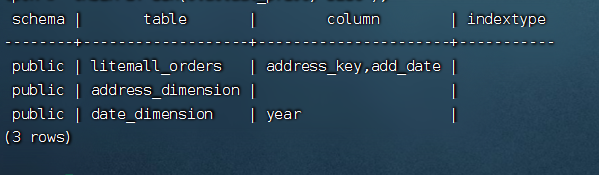
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

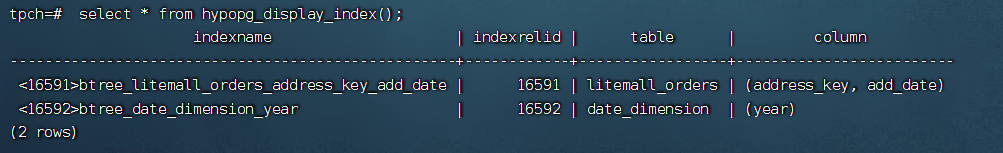
GROUP BY ad.province

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。

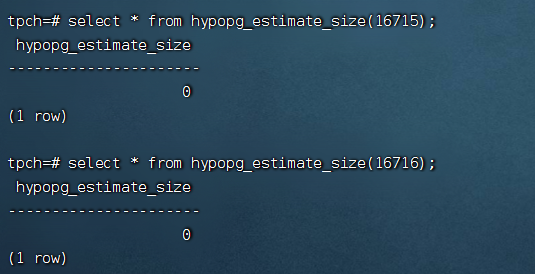
select \* from hypopg\_display\_index();



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

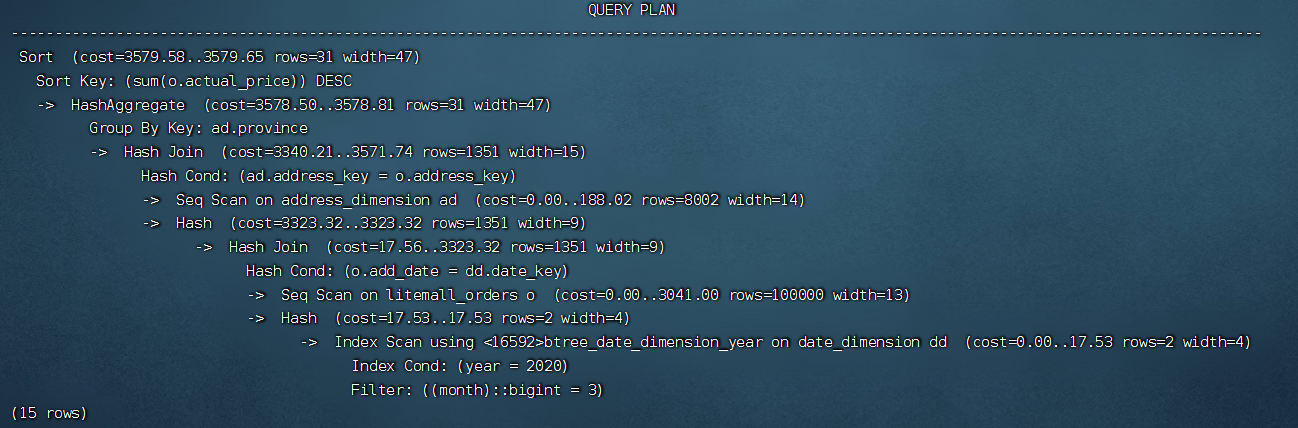
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

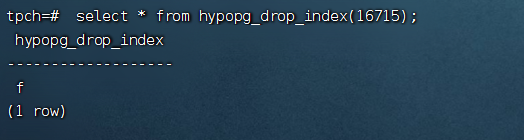
GROUP BY ad.province

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



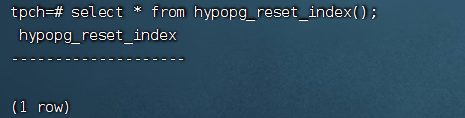
6. 删除某一个索引虚拟列，将执行结果截图。

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



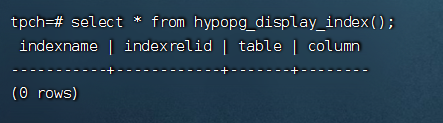
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

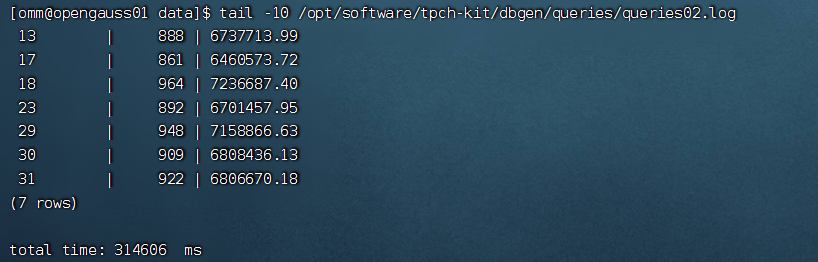
select \* from hypopg\_display\_index();



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？

“shared\_buffers”: 应用程序和操作系统之间共享的内存缓冲区大小，改为了186864个块。可以增加内存缓冲区的大小，从而减少磁盘I/O操作，减少数据库的响应时间和提高并发能力。

“max\_connections”: 允许数据库并行连接的最大数目，改为了370个连接。可以支持更多的并发连接。

“effective\_cache\_size”: PostgreSQL用于确定查询计划的缓存大小，改为了21602940KB。

“effective\_io\_concurrency”: 控制硬盘IO并发性的参数，改为了200。

“wal\_buffers”: WAL缓冲区的大小，默认的WAL缓冲区是8MB，改为了5839个块。

“random\_page\_cost”: 计算查询成本的参数，表达了一个数据页和一个随机磁盘块的相对花费，改为了1。

“default\_statistics\_target”: 显示统计信息的默认目标，改为了1000。

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？

好处：

索引是一种特殊的数据结构，用于加快数据库中的查询速度和性能。索引允许在表中查找特定值和数据范围的子集，而不必扫描整个表。这可以大大缩短查询的响应时间，并减少了数据库IO操作的次数。

方案：

数据库设计优化：优化数据库结构和架构设计，包括选择正确的数据类型，适当的表分割和合并，避免冗余数据等。

查询优化：通过优化SQL语句的编写来提高数据库查询性能，例如使用JOIN代替子查询，使用UNION代替OR等。

缓存策略优化：通过改善数据库缓存策略来优化性能，例如增加缓存大小，调整缓存时间和频率等。

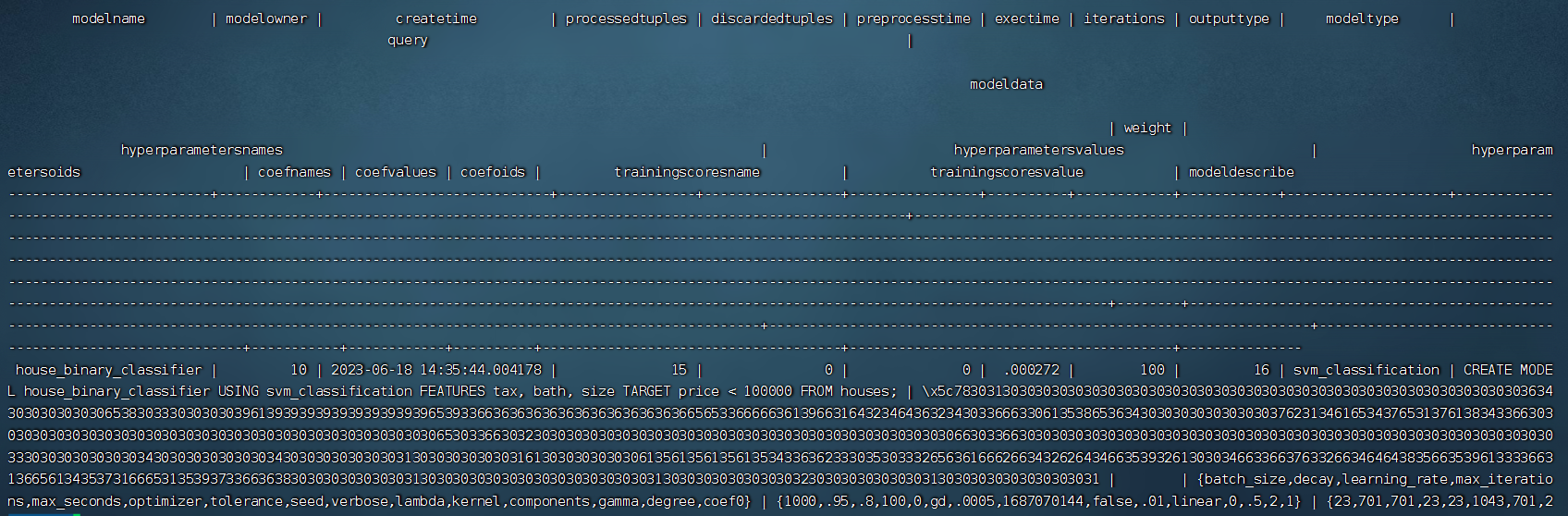
存储引擎调优：选择合适的存储引擎，例如InnoDB、MyISAM、PostgreSQL等，并调整其参数以最大化性能。

硬件优化：包括添加更多硬件资源，例如内存、CPU和存储设备，并调整其配置以最大化数据库性能。

# 关卡四、openGauss的DB4AI特性应用

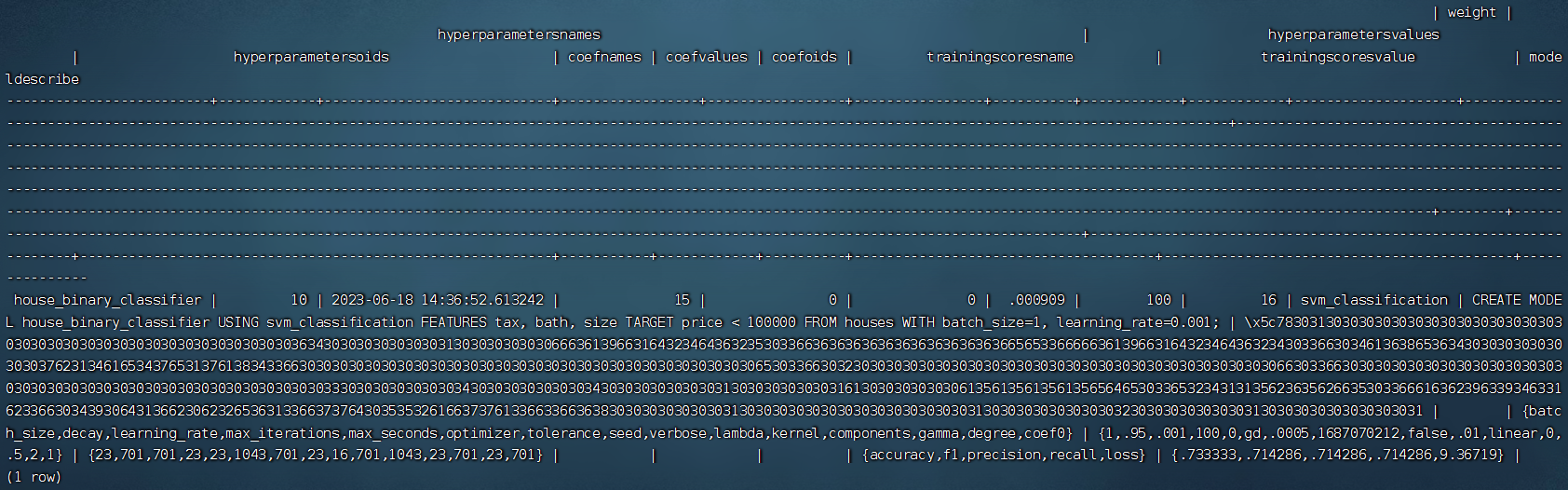
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



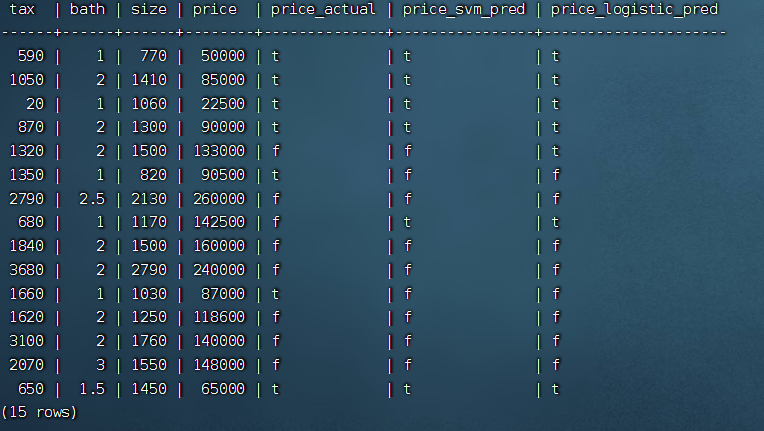
任务二：观察新模型的信息，将执行结果截图。

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？

分类模型用于预测一个样本属于哪个类别，即进行分类任务。回归模型用于预测一个连续的数值，即进行回归任务。

实践思考题2：什么是SVM算法？

SVM即支持向量机，是一种常见的机器学习算法，主要用于分类和回归问题。其主要思想是寻找一个最优超平面，将不同分类的数据点分开并最大化他们之间的间隔，从而实现分类与回归任务。

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）：指分类正确的样本数占总样本数的比例，是最常用和直观的评价指标。但是在样本不均衡的情况下，准确率容易被较多的样本主导而失去判别力。

精确率（Precision）：指被分类器预测为正的样本中，真正为正的样本数占总预测为正的样本数的比例。精确率比较适用于偏向少数样本的分类问题，例如欺诈检测。

召回率（Recall）：指分类器预测为正的样本中，真正为正的样本数占总正样本数的比例。召回率比较适用于偏向多数样本的分类问题，例如病人诊断。

AUC值：ROC曲线下的面积，可以衡量模型在不同阈值下的分类性能。AUC值越接近1，说明模型的性能越好。

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？

平均绝对误差（MAE）：指预测值与真实值之差的绝对值的平均数，用于度量预测值与真实值之间的平均差异。

均方误差（MSE）：指预测值与真实值之差的平方的平均数，用于度量预测值与真实值之间的平均差异，和MAE类似，但对误差大的数据更加敏感。

均方根误差（RMSE）：指MSE的平方根，用于度量预测值与真实值之间的平均误差，可以将误差平方化，避免了误差被抵消的情况。

R2（决定系数）：反映模型对观测结果变异的解释能力，通俗地讲，是预测方程的回归能力的一种评价。R2的取值范围在0-1之间，越趋近于1，表示模型对实际情况的解释能力越强。