openGauss AI特性创新实践课



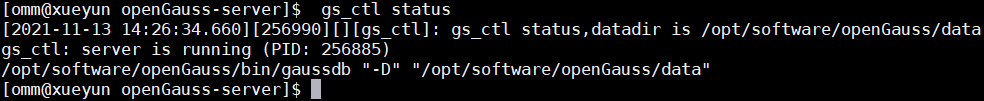
华为技术有限公司

# 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：

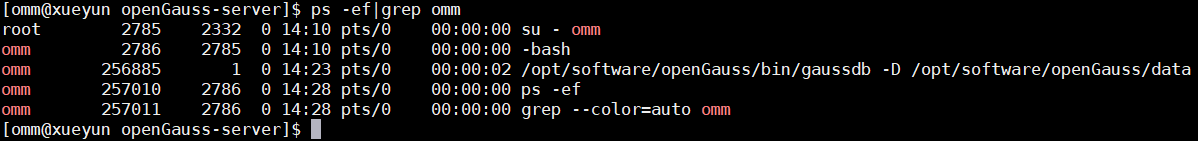
任务一：数据库状态验证

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？

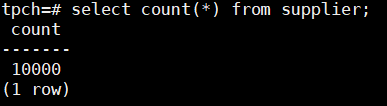
答：源码安装的优点：编译安装过程，可以设定参数，按照需求，进行安装，并且安装的版本，可以自己选择，灵活性比较大。

# 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：

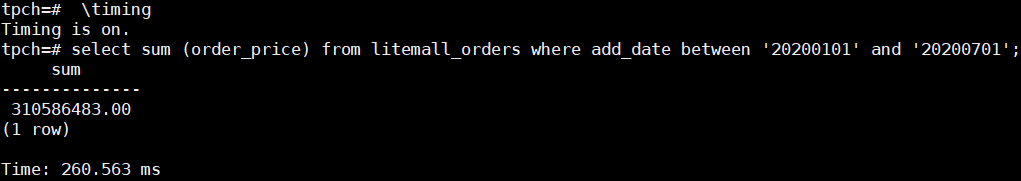
select count(\*) from supplier;;



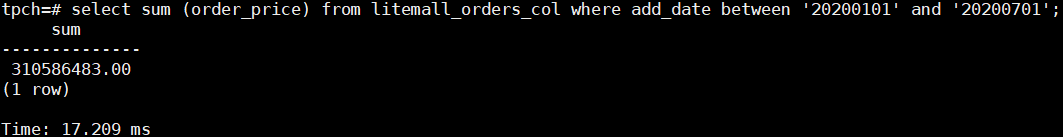
任务二：行存表与列存表执行效率对比

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';

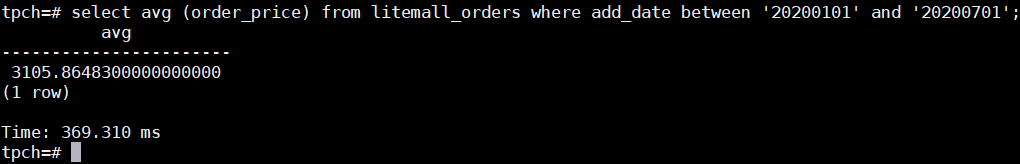


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

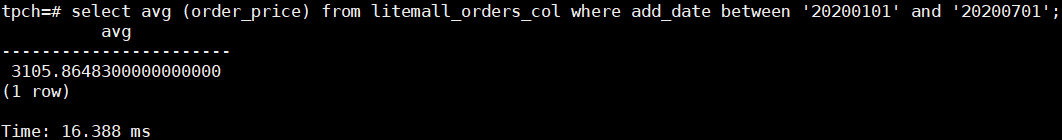


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

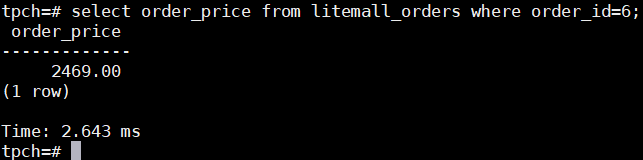


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

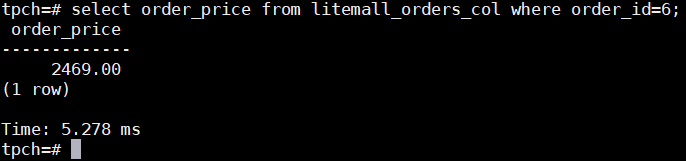


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

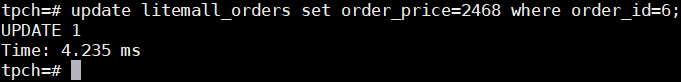


select order\_price from litemall\_orders\_col where order\_id=6;

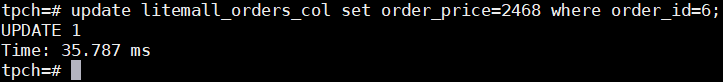


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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

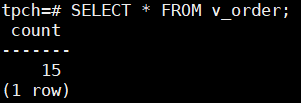
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



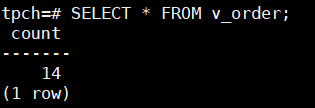
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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、行存储与列存储在数据写入时的对比

（1）行存储的写入是一次完成。如果这种写入建立在操作系统的文件系统上，可以保证写入过程的成功或者失败。【**行存储数据的完整性可以确定**】

（2）列存储由于需要把一行记录拆分成单列保存，写入次数明显比行存储多（意味着磁头调度次数多，而磁头调度是需要时间的，一般在1ms~10ms)，再加上磁头需要在盘片上移动和定位花费的时间，实际时间消耗会更大。【**对比列存储，行存储在写入上占有很大的优势**】

（3）还有数据修改,这实际也是一次写入过程。不同的是，数据修改是对磁盘上的记录做删除标记。行存储是在指定位置写入一次，列存储是将磁盘定位到多个列上分别写入，这个过程仍是行存储的列数倍。【**数据修改也是以行存储占优**】

2、行存储与列存储在数据读取时的对比

（1）数据读取时，行存储通常将一行数据完全读出，如果只需要其中几列数据的情况，就会存在冗余列，出于缩短处理时间的考量，消除冗余列的过程通常是在内存中进行的。【**行存储在读取数据时可能会有数据的冗余**】

（2）列存储每次读取的数据是集合的一段或者全部。【**行存储读取数据不存在冗余性问题**】

（3）由于【**列存储的每一列数据类型是同质的，不存在二义性问题**。】比如说某列数据类型为整型(int)，那么它的数据集合一定是整型数据。这种情况使数据解析变得十分容易。相比之下，行存储则要复杂得多，【**行存储在一行记录中保存了多种类型的数据，数据解析需要在多种数据类型之间频繁转换，这个操作很消耗CPU，增加了解析的时间。**】所以，列存储的解析过程更有利于分析大数据。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

答：物化视图实际上就是一种特殊的物理表，物化视图是相对普通视图而言的。普通视图是虚拟表，应用的局限性较大，任何对视图的查询实际上都是转换为对SQL语句的查询，性能并没有实际上提高。而物化视图实际上就是存储SQL所执行语句的结果，起到缓存的效果。

（1）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。

（2）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。

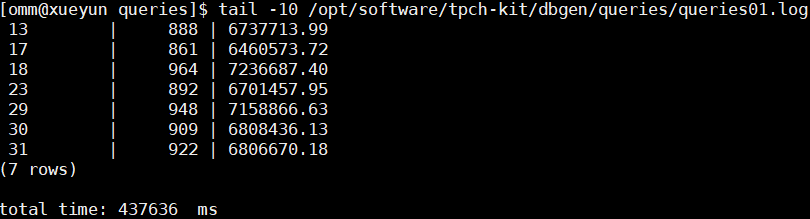
# 关卡三、openGauss的AI4DB特性应用

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：

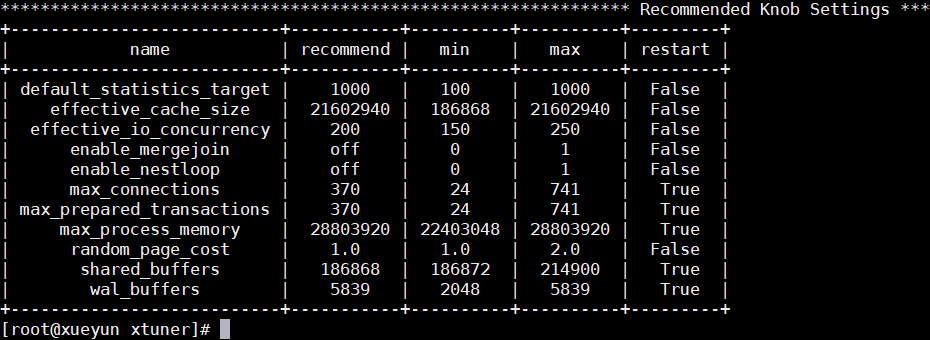
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log





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

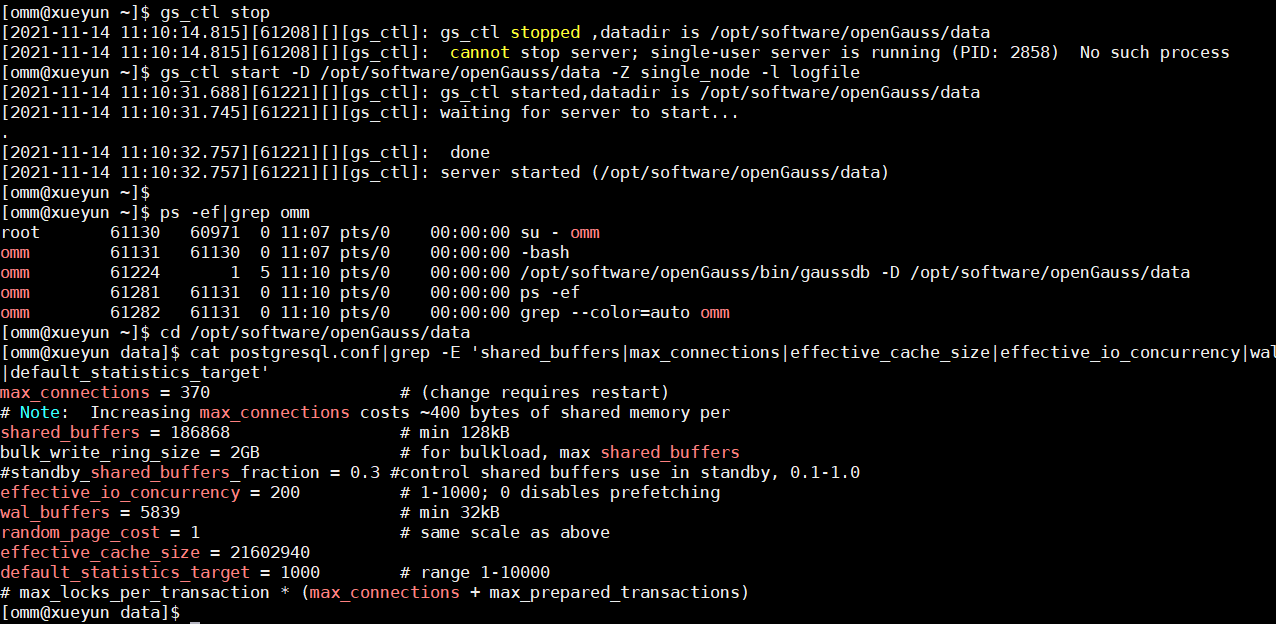
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

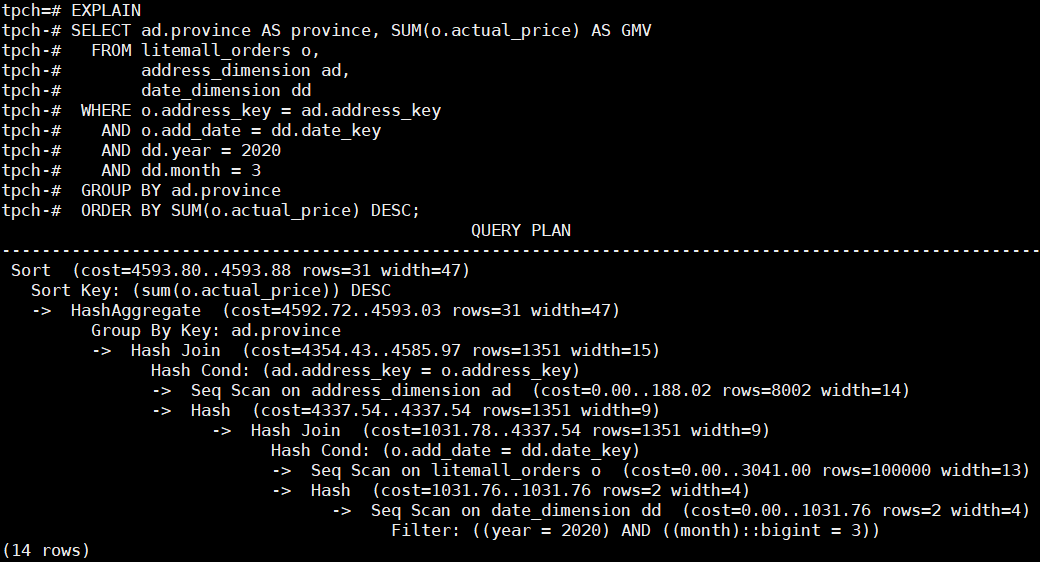
AND o.add\_date = dd.date\_key

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

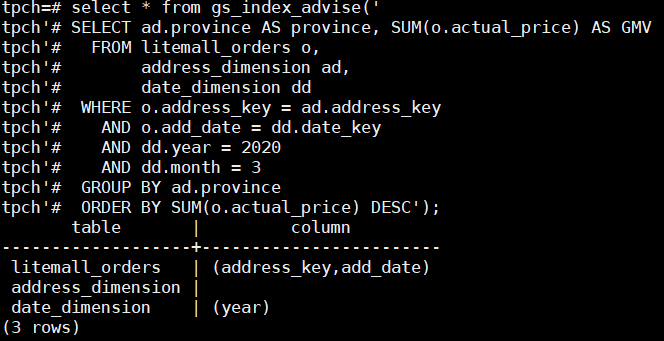
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

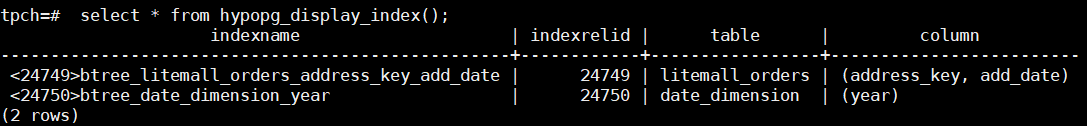
GROUP BY ad.province

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。

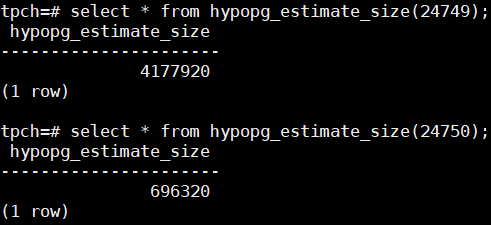
select \* from hypopg\_display\_index();



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

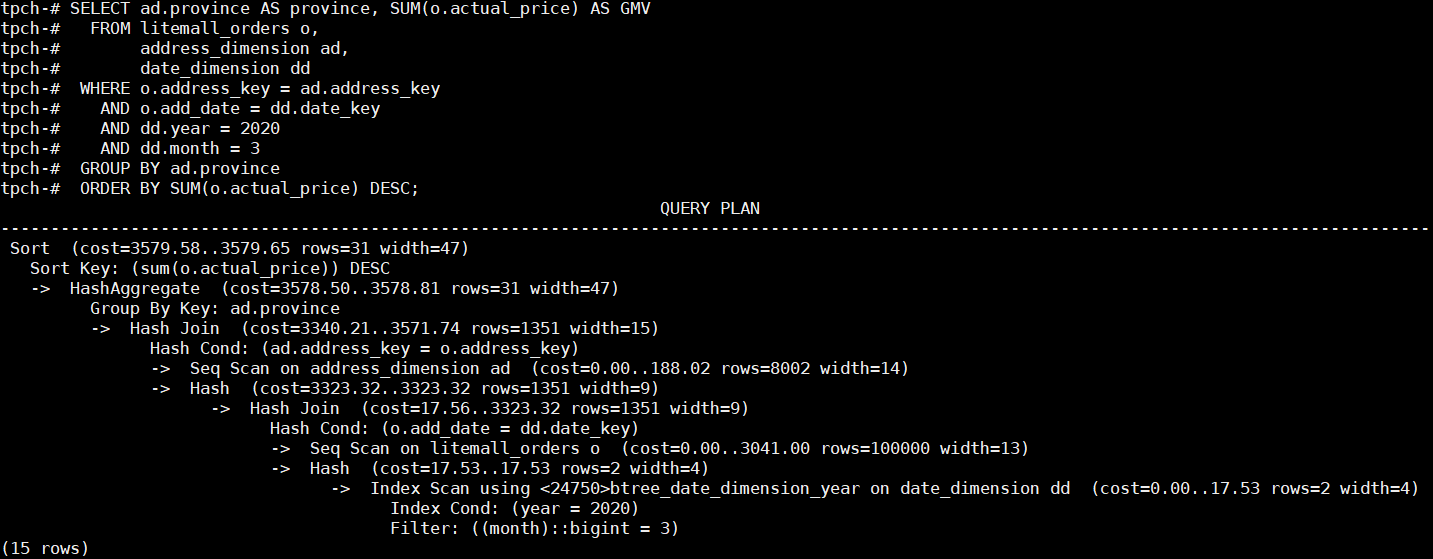
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

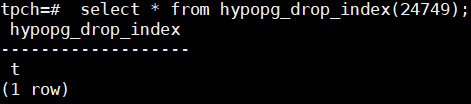
GROUP BY ad.province

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



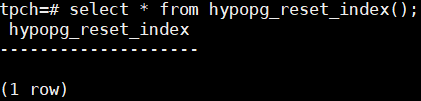
6. 删除某一个索引虚拟列，将执行结果截图。

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



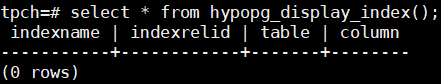
7. 删除所有索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

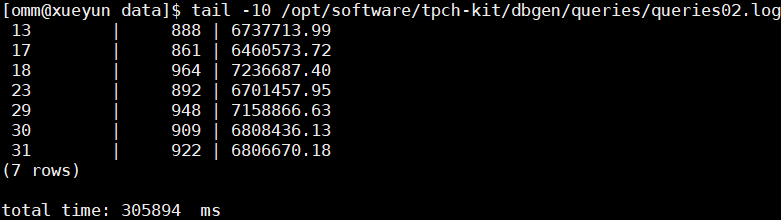
select \* from hypopg\_display\_index();



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：

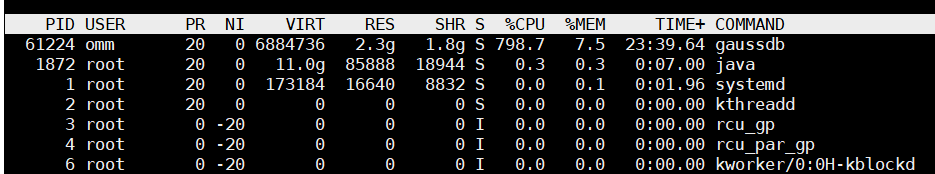
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log

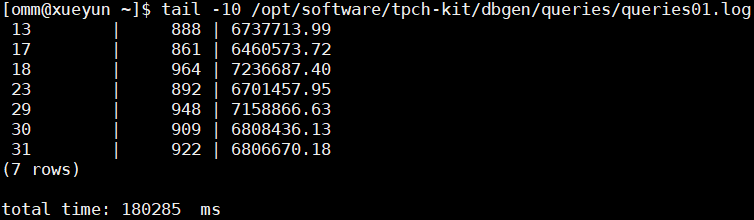


挑战一：进一步优化queries.sql中的查询语句，使得前后执行时间出现倍数级的提升。

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？

**（1）max\_connections**，允许的最大客户端连接数。这个参数设置大小和work\_mem有一些关系。配置的越高，可能会占用系统更多的内存。

**（2）shared\_buffers**，PostgreSQL使用自己的缓冲区，也使用Linux操作系统内核缓冲OS Cache。这就说明数据两次存储在内存中，首先是PostgreSQL缓冲区，然后是操作系统内核缓冲区。与其他数据库不同，PostgreSQL不提供直接IO，所以这又被称为双缓冲。PostgreSQL缓冲区称为shared\_buffer，建议设置为物理内存的1/4。而实际配置取决于硬件配置和工作负载，如果你的内存很大，而你又想多缓冲一些数据到内存中，可以继续调大shared\_buffer。

**（3）Effective\_cache\_size，**这个参数主要用于Postgre查询优化器。是单个查询可用的磁盘高速缓存的有效大小的一个假设，是一个估算值，它并不占据系统内存。由于优化器需要进行估算成本，较高的值更有可能使用索引扫描，较低的值则有可能使用顺序扫描。

**（4）wal\_buffers，**事务日志缓冲区的大小，PostgreSQL将WAL记录写入缓冲区，然后再将缓冲区刷新到磁盘。在PostgreSQL 12版中，默认值为-1，也就是选择等于shared\_buffers的1/32 。

**（5）default\_statistics\_target，**PostgreSQL使用统计信息来生成执行计划。统计信息可以通过手动Analyze命令或者是autovacuum进程启动的自动分析来收集，default\_statistics\_target参数指定在收集和记录这些统计信息时的详细程度。

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）[索引](https://baike.baidu.com/item/%E7%B4%A2%E5%BC%95/5716853)是对[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728)表中一列或多列的值进行排序的一种结构，使用索引可快速访问数据库表中的特定信息。如果想按特定职员的姓来查找他或她，则与在表中搜索所有的行相比，索引有助于更快地获取信息。

（2）除了使用索引和参数外，对数据库进行优化的方式还有：

1、SQL 语句优化，在进行查询时，应尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引；

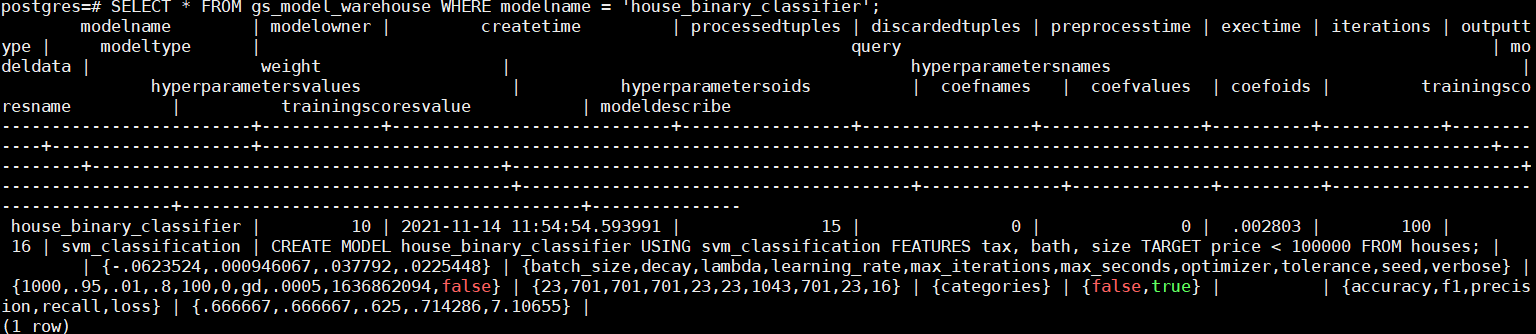
2、选取最适用的字段属性，MySQL可以很好的支持大数据量的存取，但是一般说来，数据库中的表越小，在它上面执行的查询也就会越快。因此，在创建表的时候，为了获得更好的性能，我们可以将表中字段的宽度设得尽可能小；

3、适当对数据库进行分库分表。

# 关卡四、openGauss的DB4AI特性应用

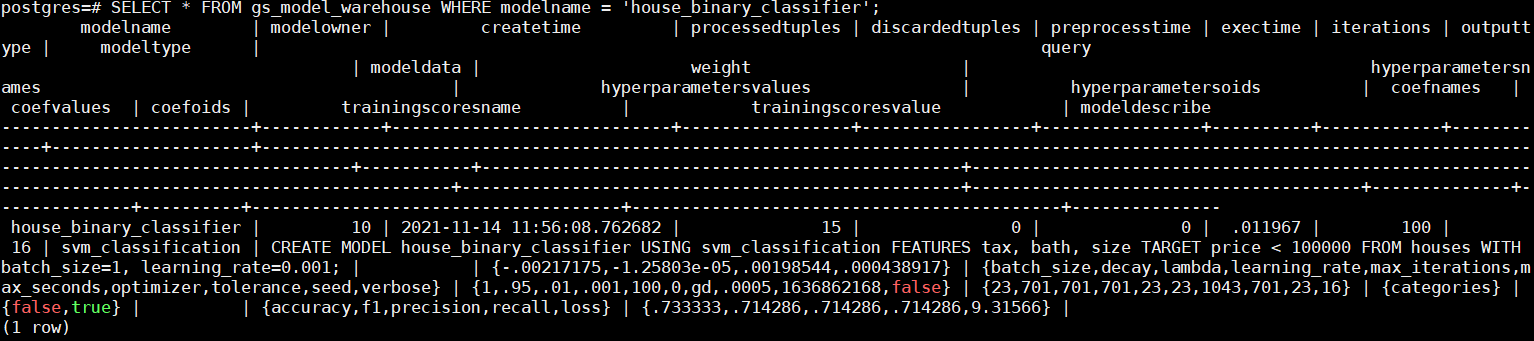
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



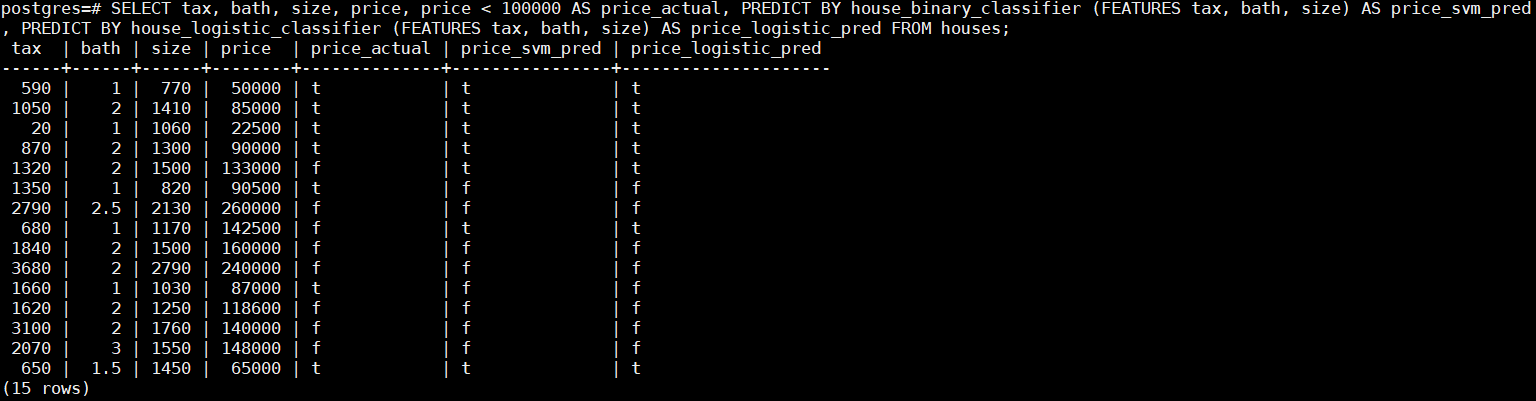
任务二：观察新模型的信息，将执行结果截图。

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？

答：分类和回归的区别在于输出变量的类型。

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？

答：支持向量机（support vector machines，SVM）是一种二分类模型，它将实例的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，以 “最好地” 区分这两类点，以至如果以后有了新的点，这条线也能做出很好的分类。SVM 适合中小型数据样本、非线性、高维的分类问题。

SVM 最早是由 Vladimir N. Vapnik 和 Alexey Ya. Chervonenkis 在1963年提出，目前的版本（soft margin）是由 Corinna Cortes 和 Vapnik 在1993年提出，并在1995年发表。深度学习（2012）出现之前，SVM 被认为机器学习中近十几年来最成功，表现最好的算法。

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？

（1）准确率：正确预测的样本占所有样本的比重 ；

（2）精确率：预测为正的样本中真正为正的比例；

（3）召回率：所有正样本中被正确预测的比重；

（4）F1 值：精确率和召回率的调和平均值；

（5）ROC：ROC (Receiver operating characteristic)接收者操作特征曲线，是反映敏感性和特异性连续变量的综合指标，roc曲线上每个点反映着对同一信号刺激的感受性；

（6）AUC：AUC（Area Under Curve），是另一种评价二分类算法的指标，被定义为 ROC 曲线下的面积，这个面积的数值不会大于 1，和 F1 score 差不多，都是综合评价精准率和召回率的指标，只不过绘制 ROC 曲线使用了另外两个此消彼长的指标。

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？

（1）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。

（2）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。

（3）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。

（4）SD（Standard Deviation）标准差，方差的算术平均根。用于衡量一组数值的离散程度。