openGauss AI特性创新实践课



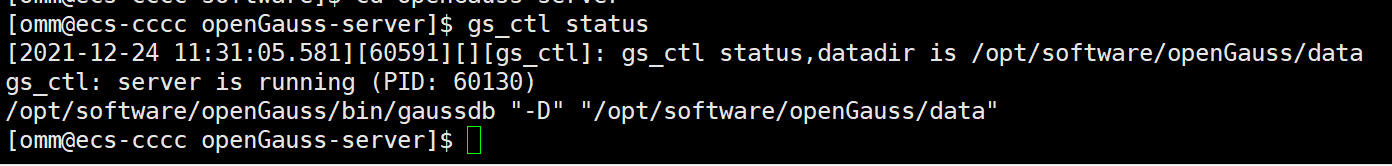
华为技术有限公司

# 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：

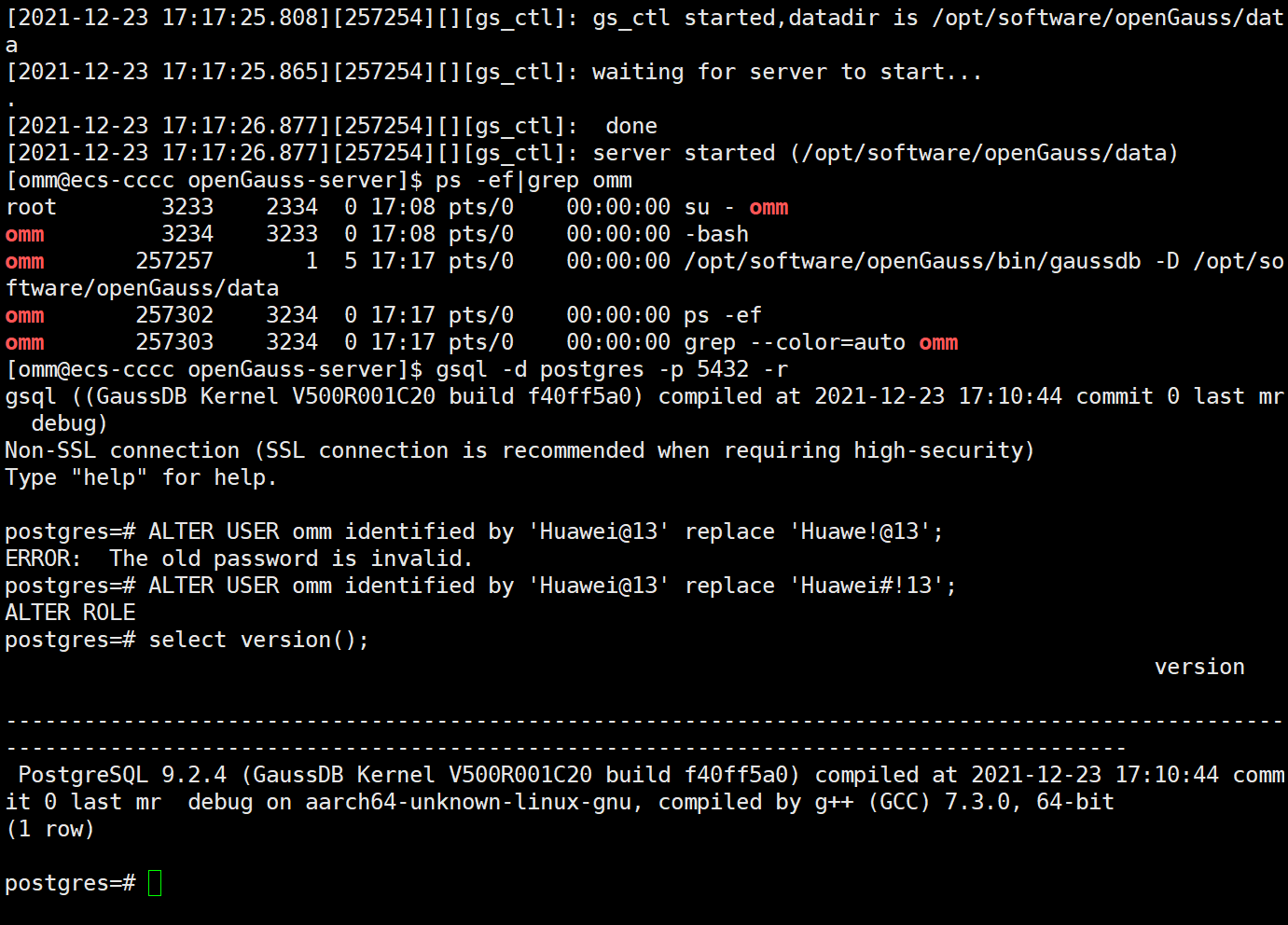
任务一：数据库状态验证

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？

因为x86 和鲲鹏使用的指令是不一样的，在鲲鹏上使用的是精简指令集，而在 x86 上使用的是一个复杂的指令集，在不同平台上的程序需经过编译才能运行。通过源码编译安装数据库，可以在安装过程中设定参数，按照需求，进行安装，并且可以自己选择安装的版本与内容，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：

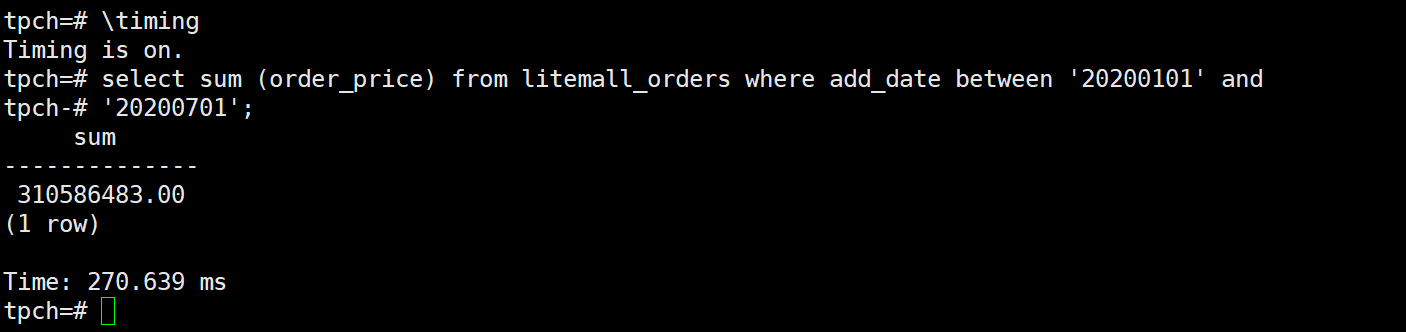
select count(\*) from supplier;;



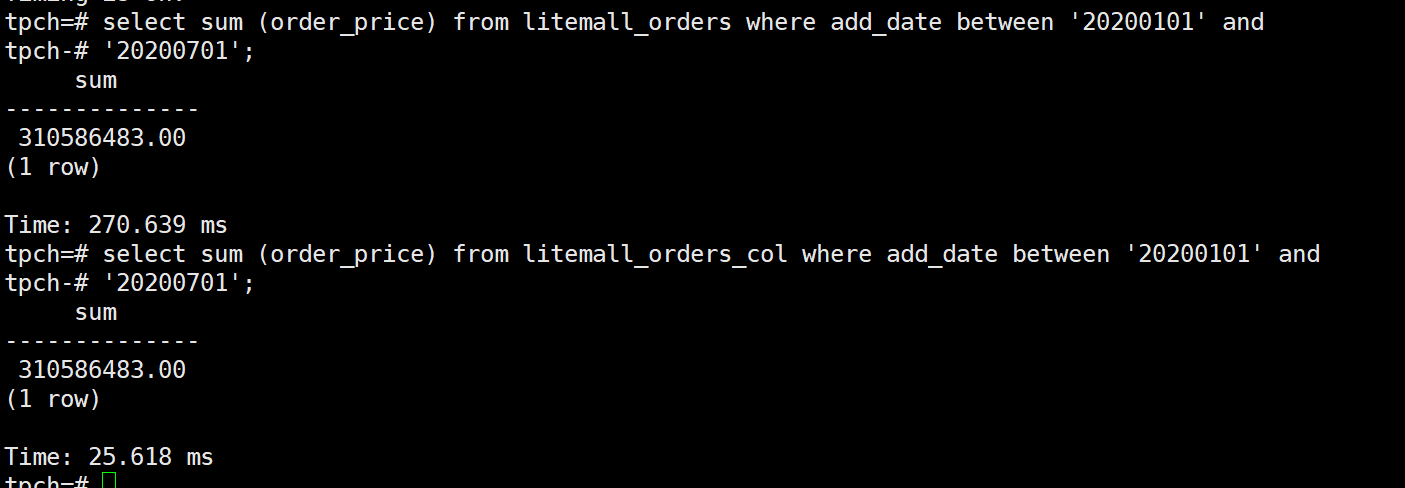
任务二：行存表与列存表执行效率对比

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';

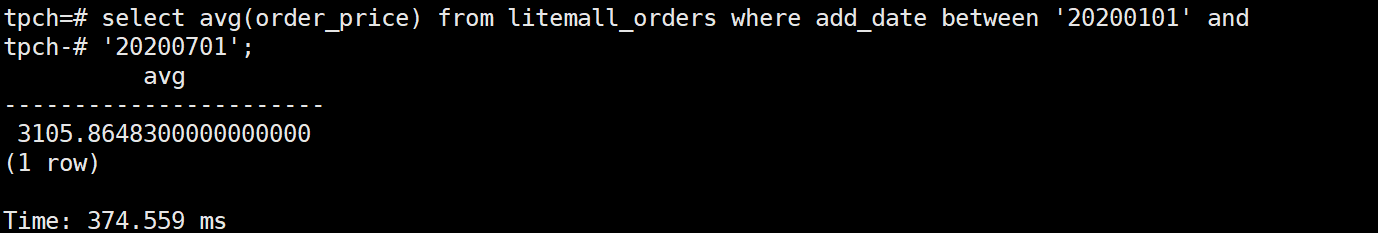


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

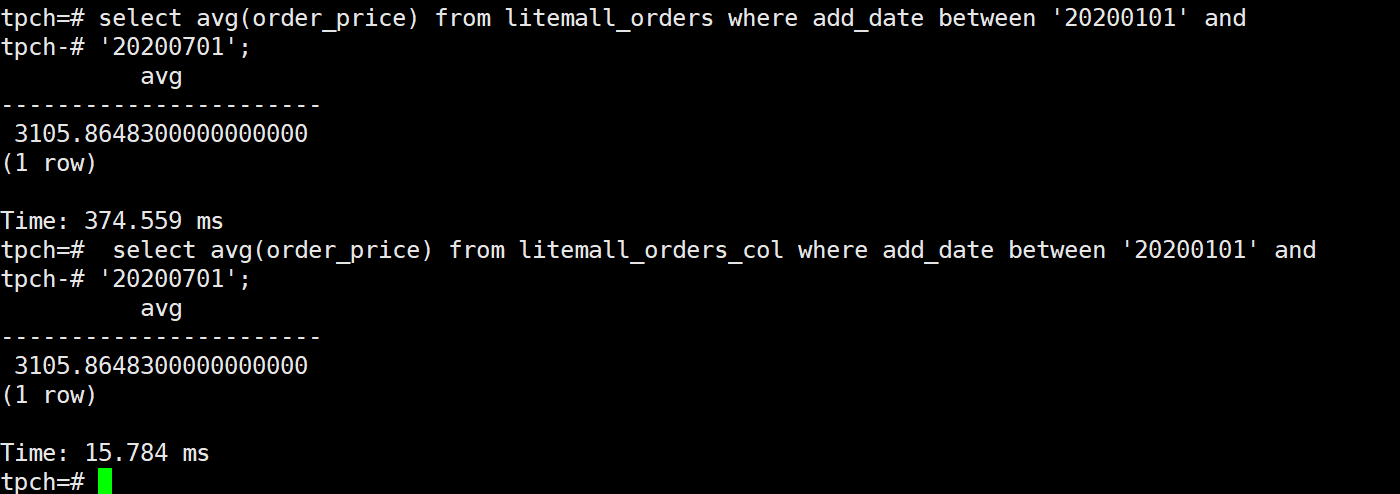


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

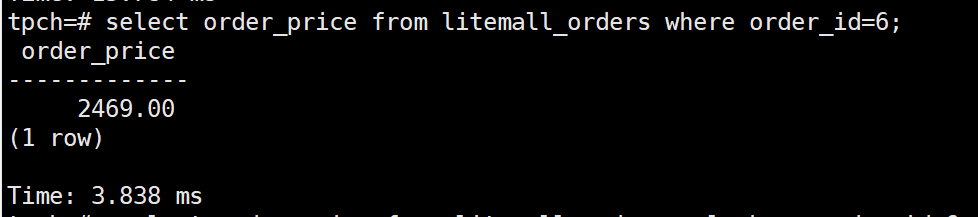


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

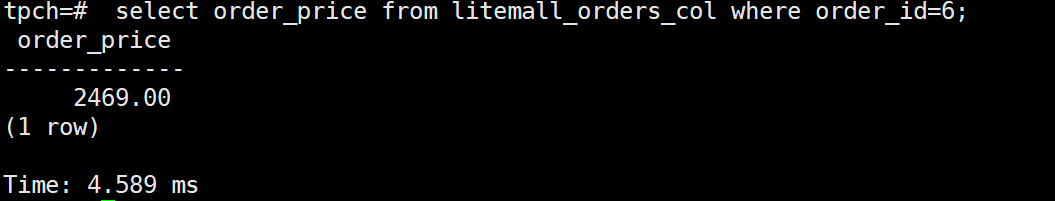


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

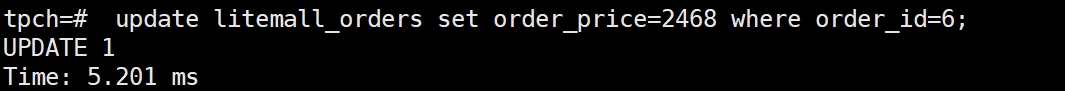


select order\_price from litemall\_orders\_col where order\_id=6;

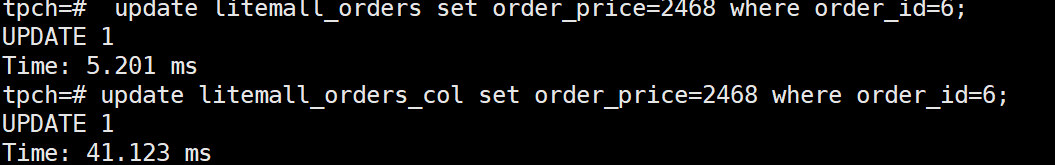


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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



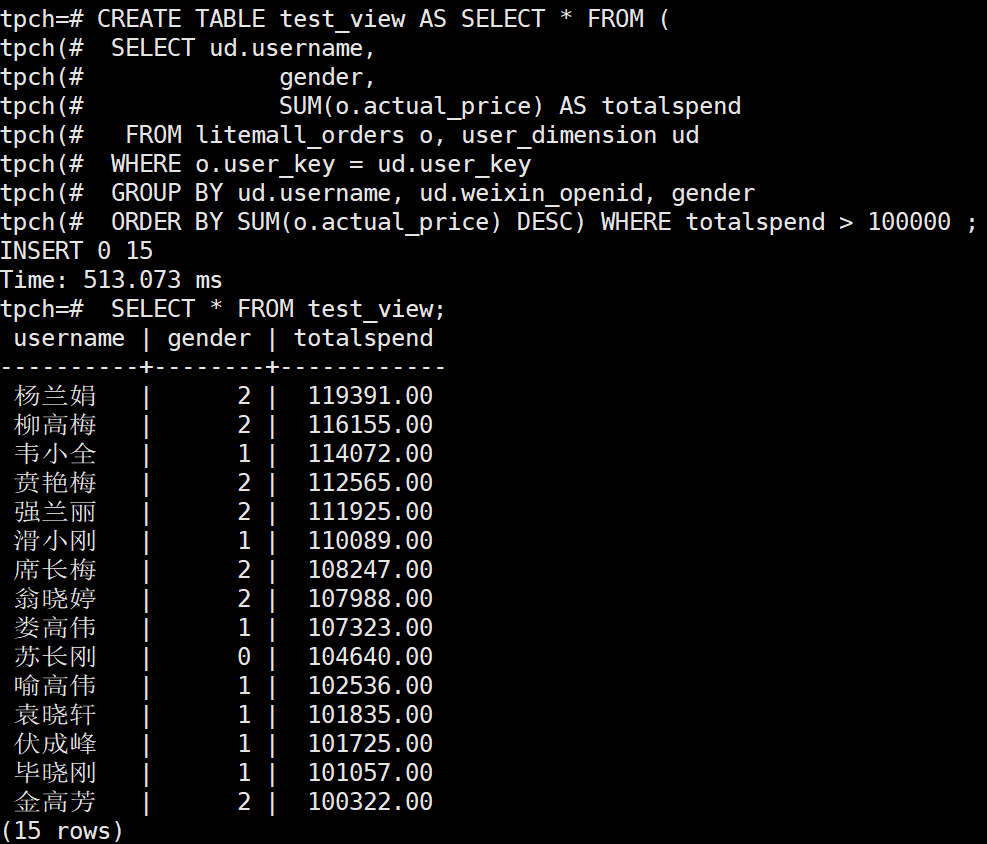
update litemall\_orders\_col set order\_price=2468 where order\_id=6;



任务三：物化视图的使用

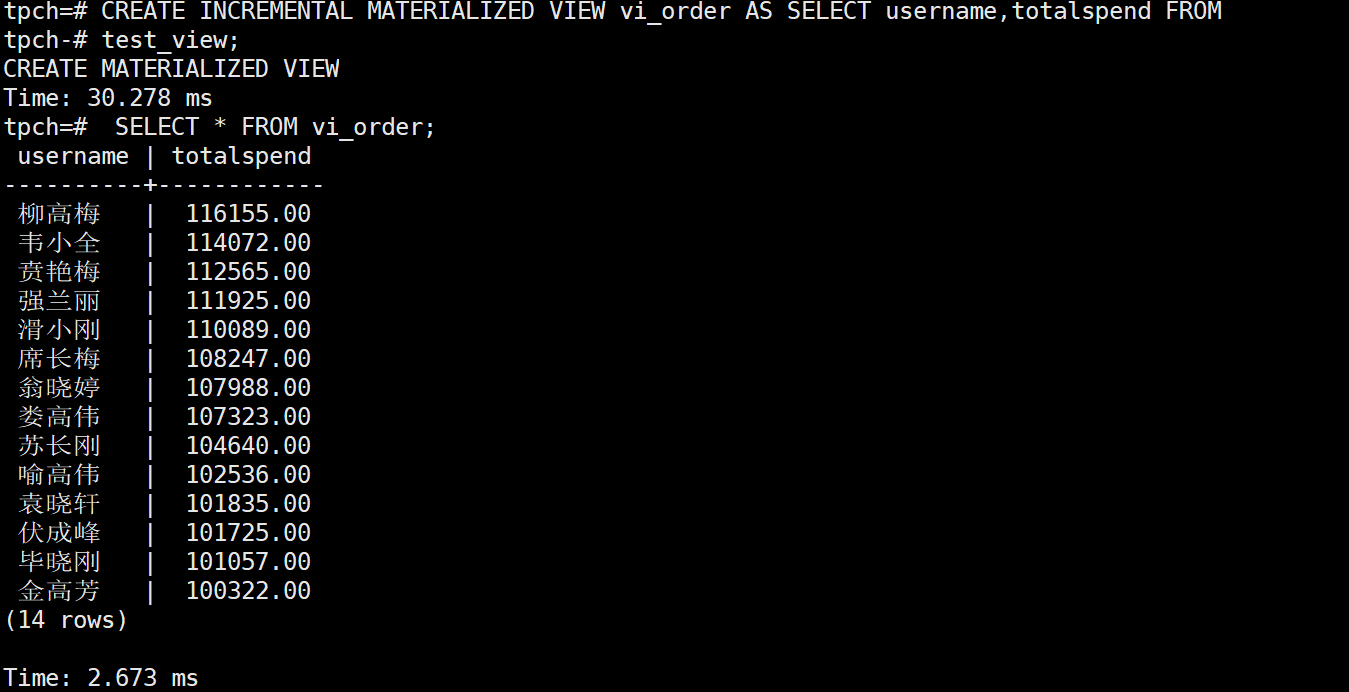
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



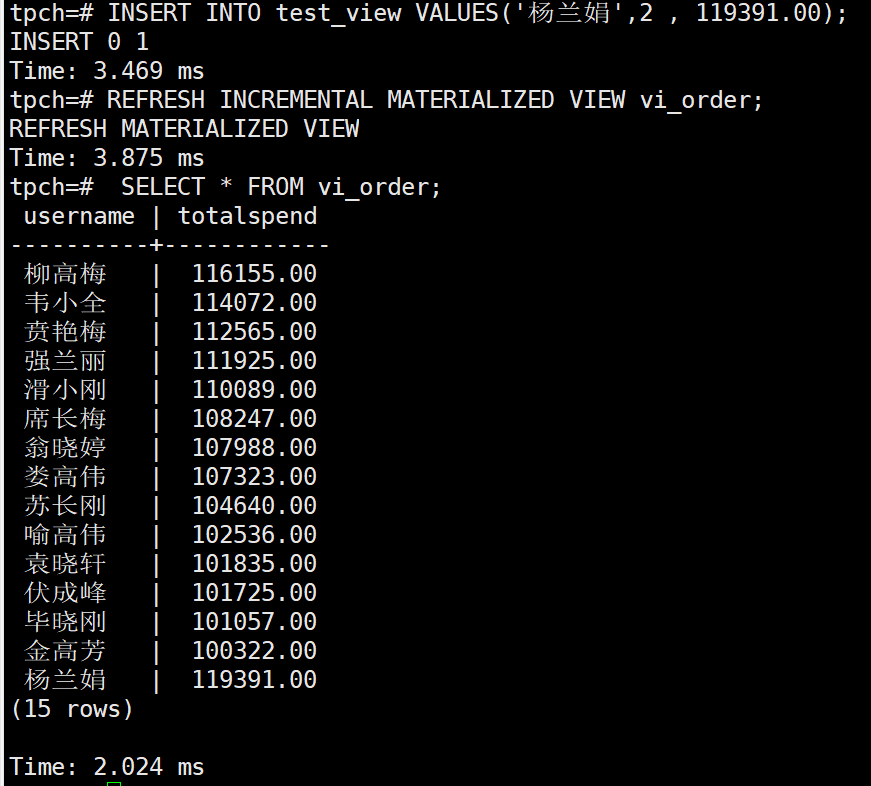
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



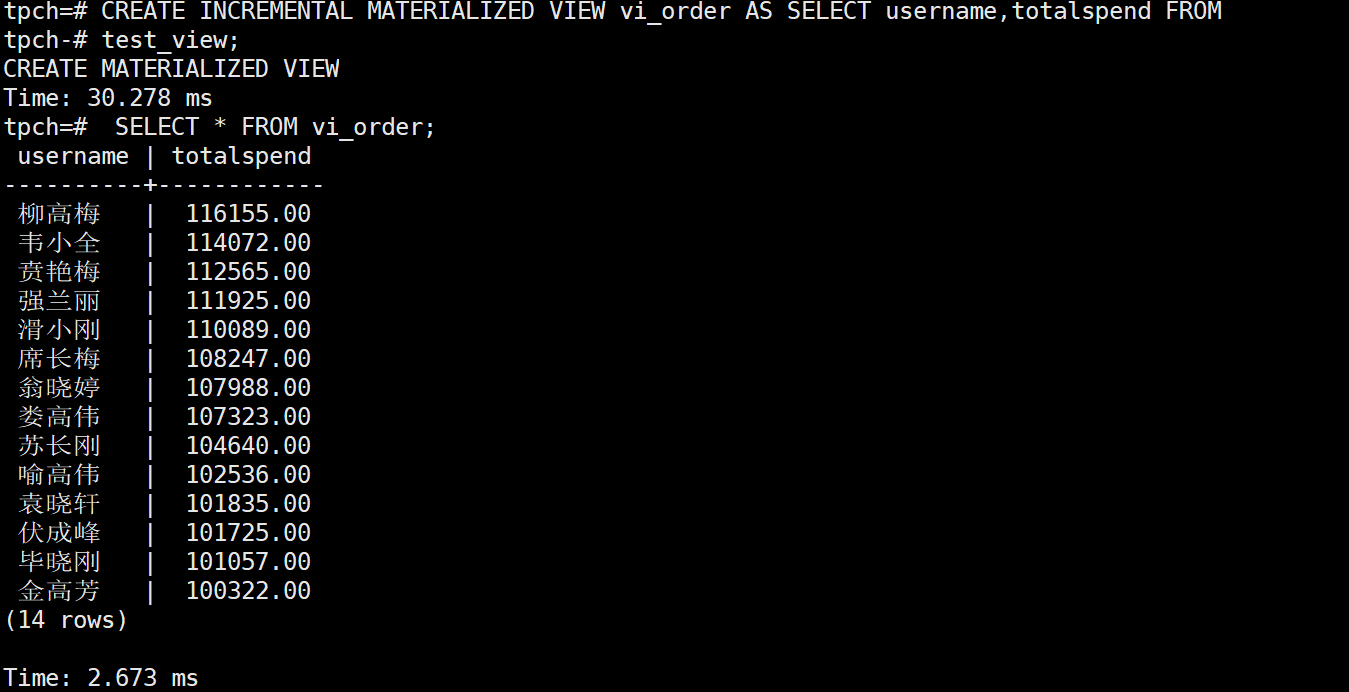
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



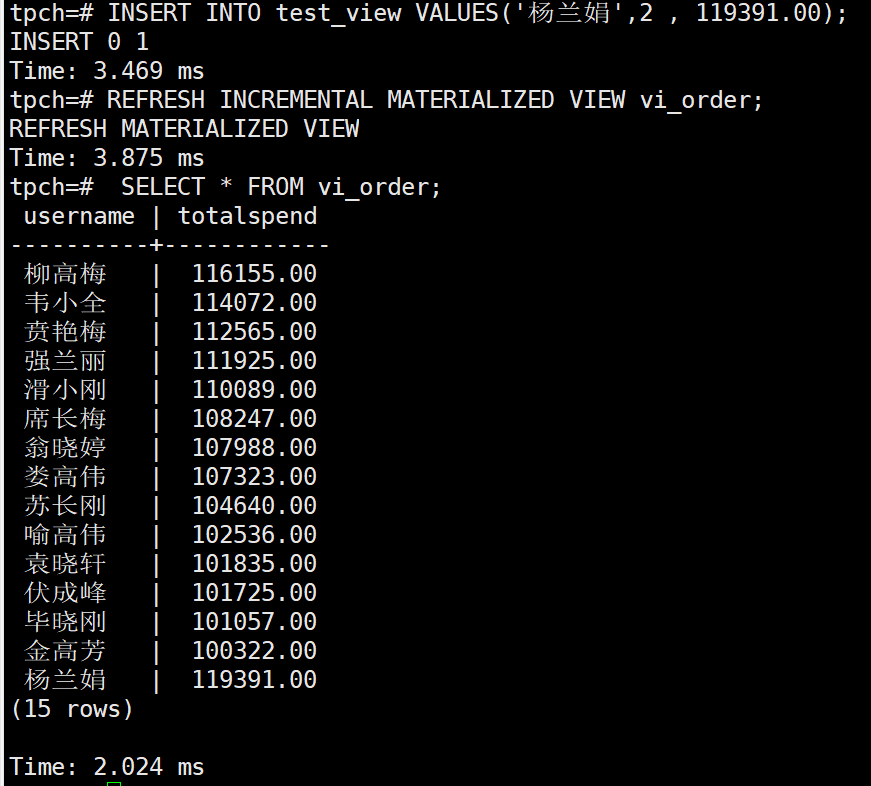
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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？

从数据写入方面来看，行存储的写入是一次完成，数据的完整性因此可以确定。而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，从实际时间消耗上看行存储相较于列存储更快。因此行存储相对于列存储在执行数据写入方向的SQL操作上效率更高。

从数据读取方面看，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，而列存储每次读取的数据是集合的一段或者全部，不存在冗余性问题，因此列存储相对于行存储在执行数据读取方向的SQL操作上更有优势。

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。

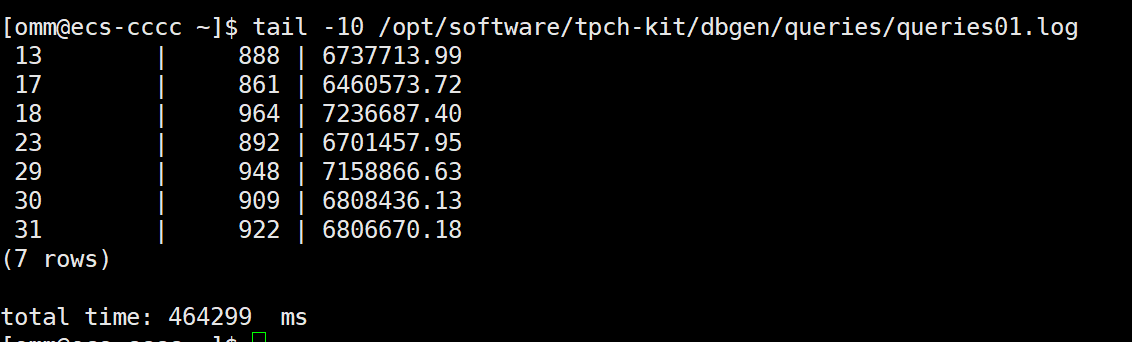
增量物化视图顾名思义就是可以对物化视图增量刷新，需要用户手动执行语句完成对物化视图在一段时间内的增量数据进行刷新。与全量创建物化视图不同在于目前增量物化视图所支持场景较小，目前物化视图创建语句仅支持基表扫描语句或者UNION ALL语句。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

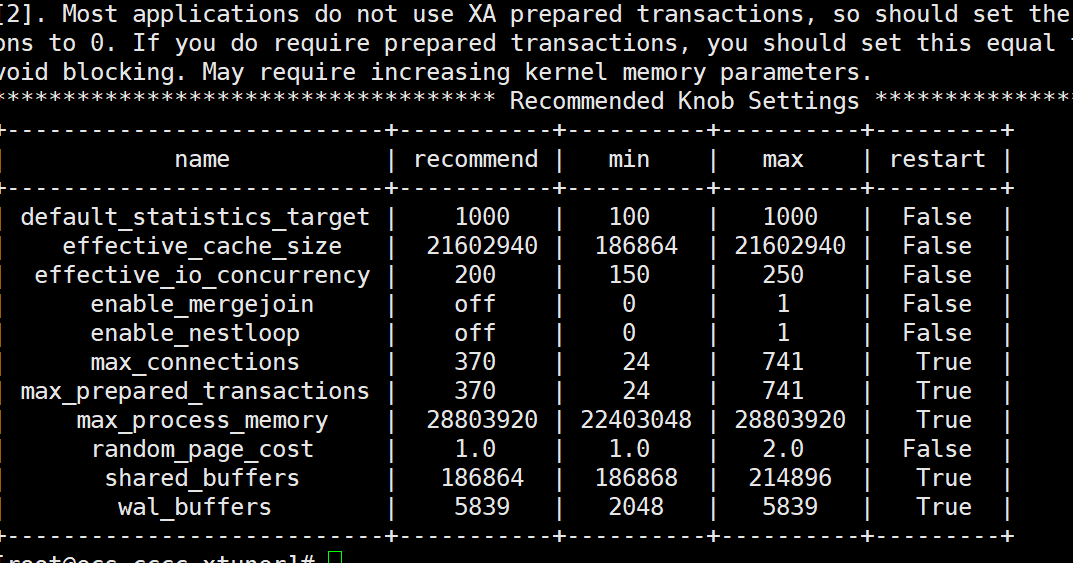
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

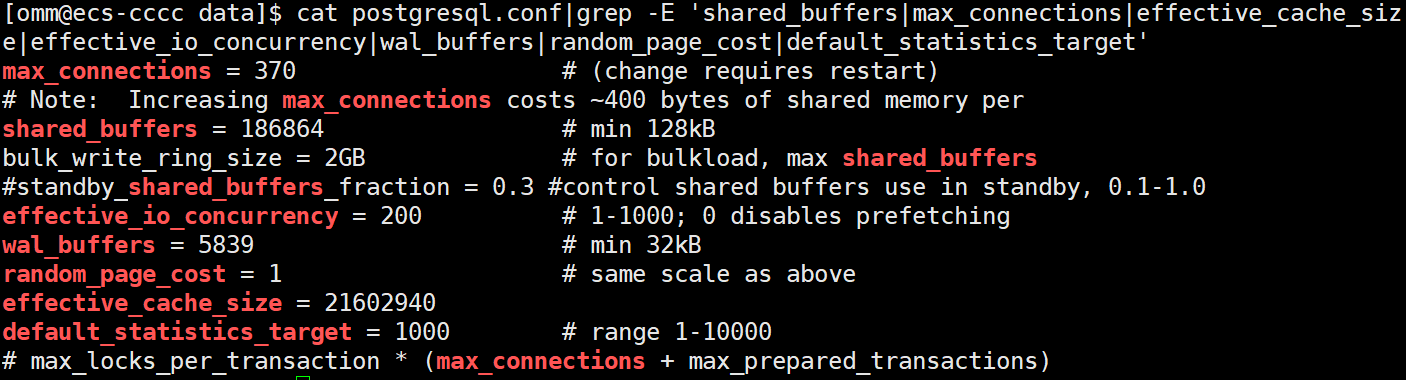
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

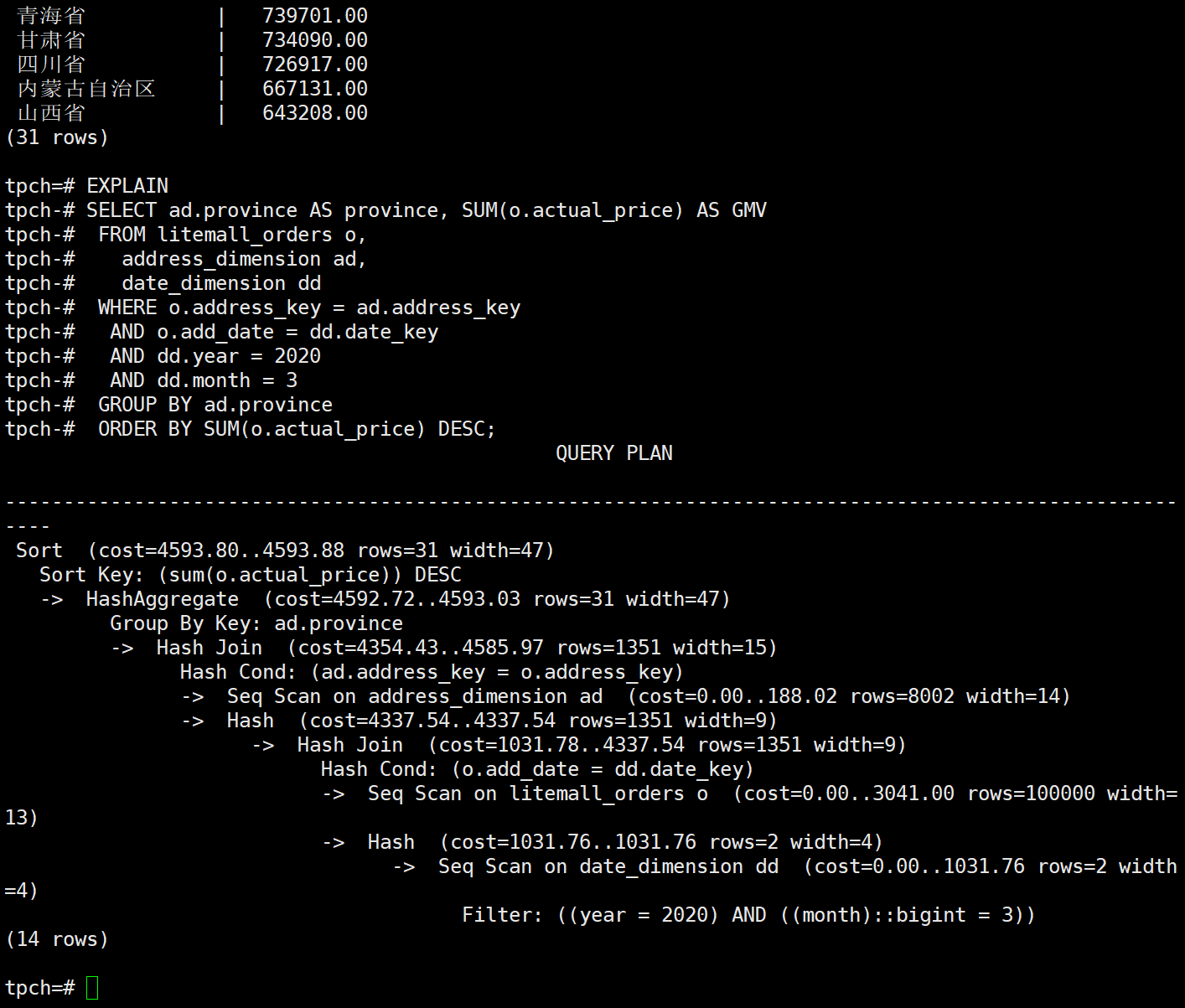
AND o.add\_date = dd.date\_key

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

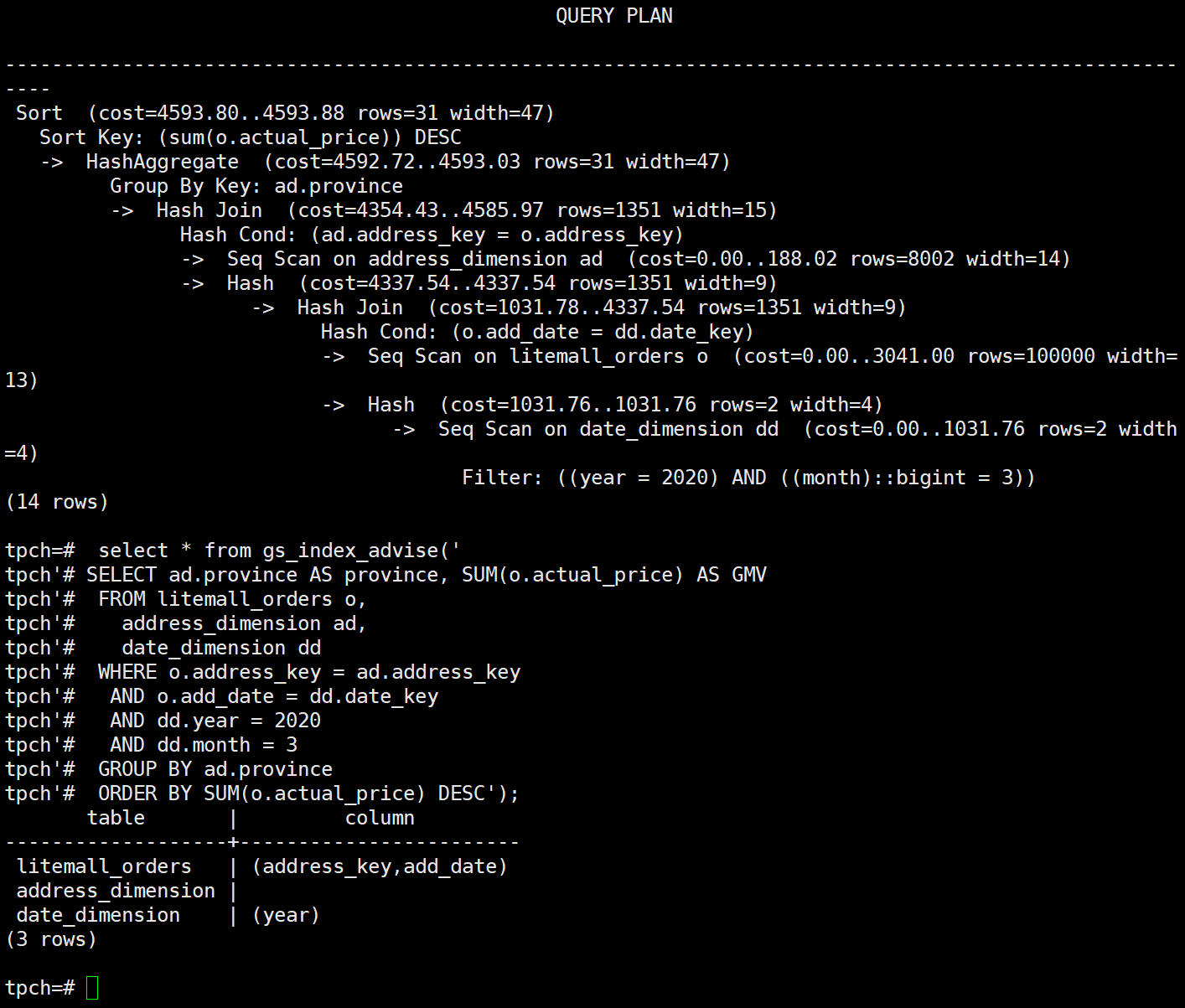
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

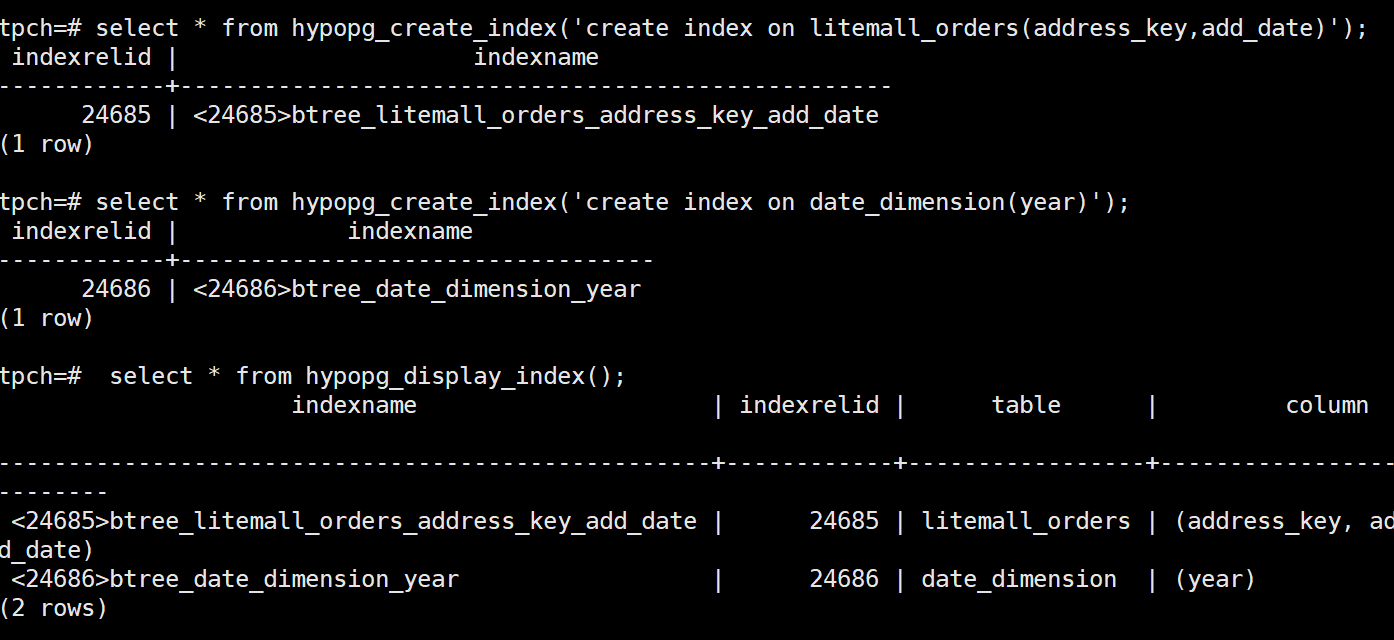
GROUP BY ad.province

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。

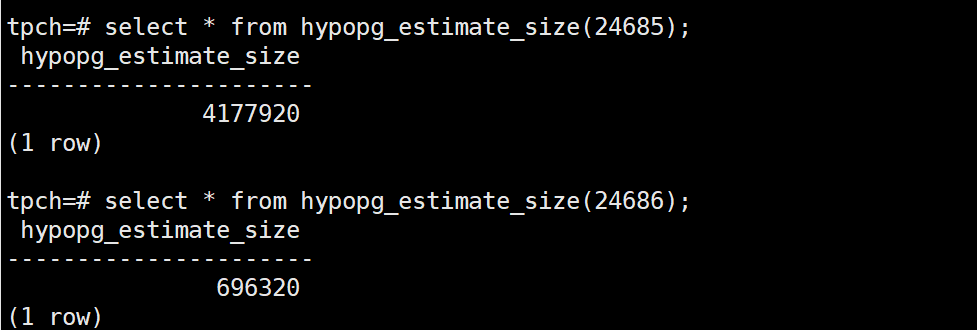
select \* from hypopg\_display\_index();



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

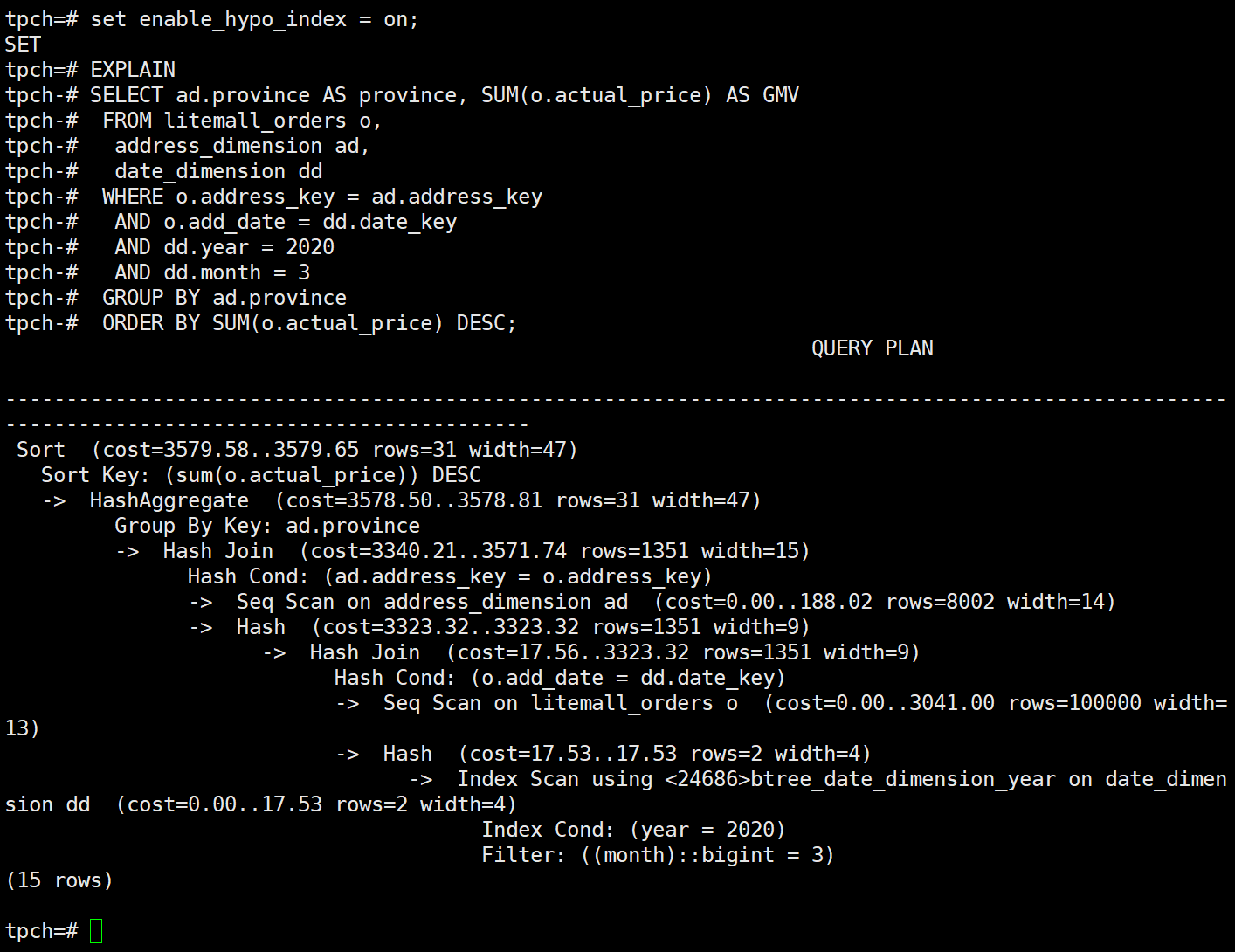
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

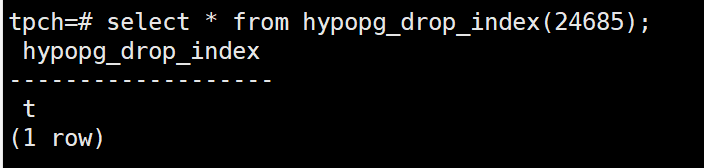
GROUP BY ad.province

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



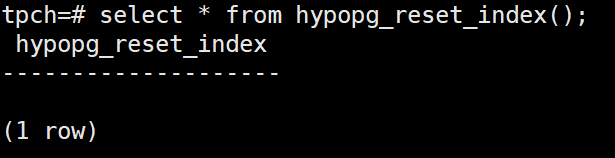
6. 删除某一个索引虚拟列，将执行结果截图。

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



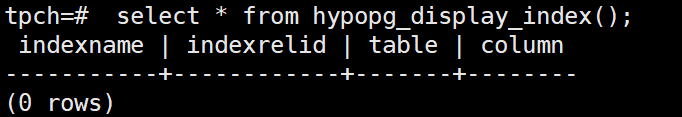
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

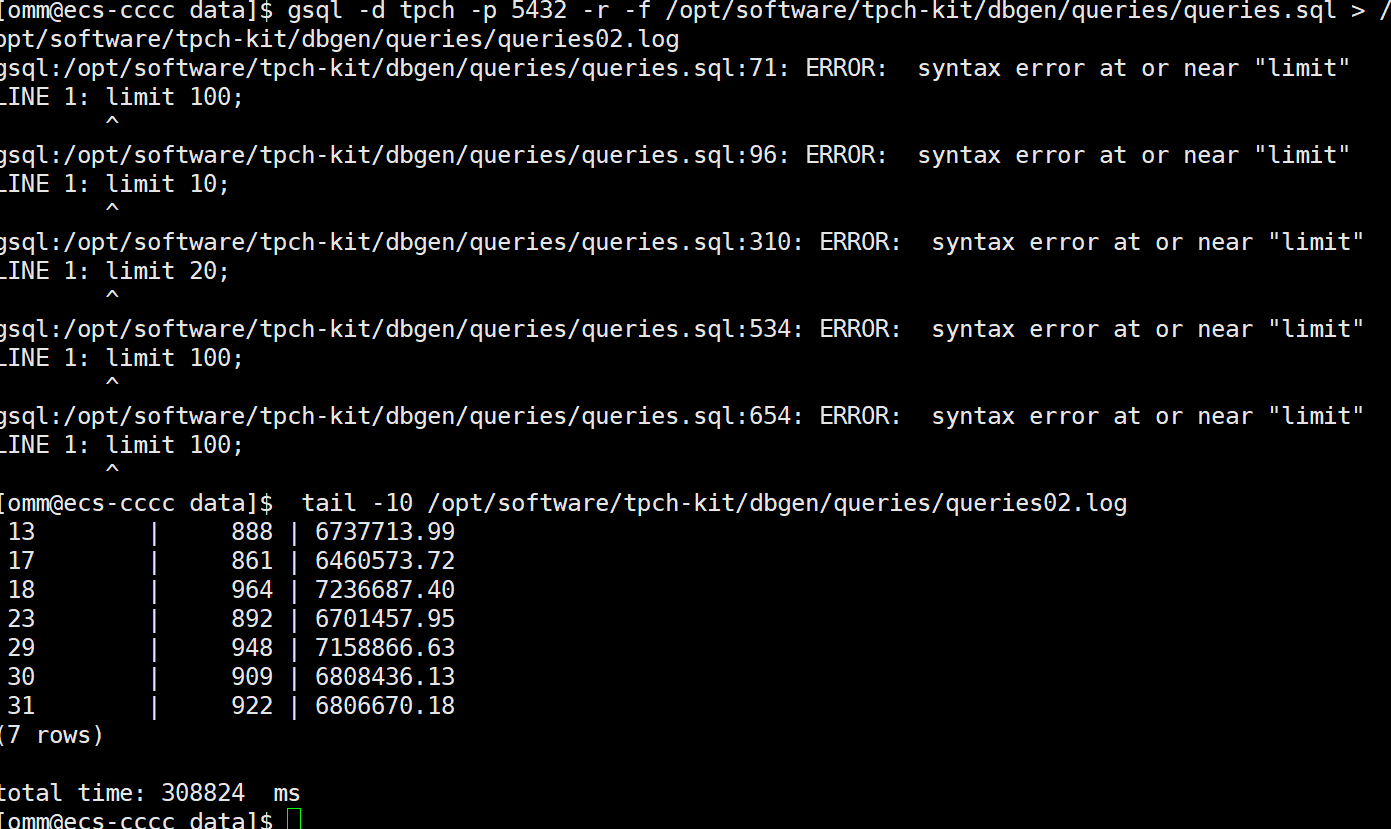
select \* from hypopg\_display\_index();



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：

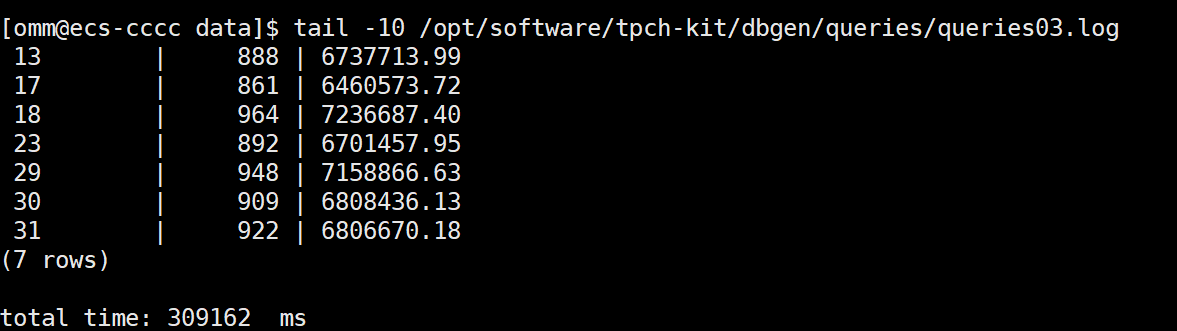
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log



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？

在order表的o\_orderdate列，以及lineitem表的l\_shipdate列上创建了索引，因为这两个参数是经常需要根据范围进行搜索和排序的参数。

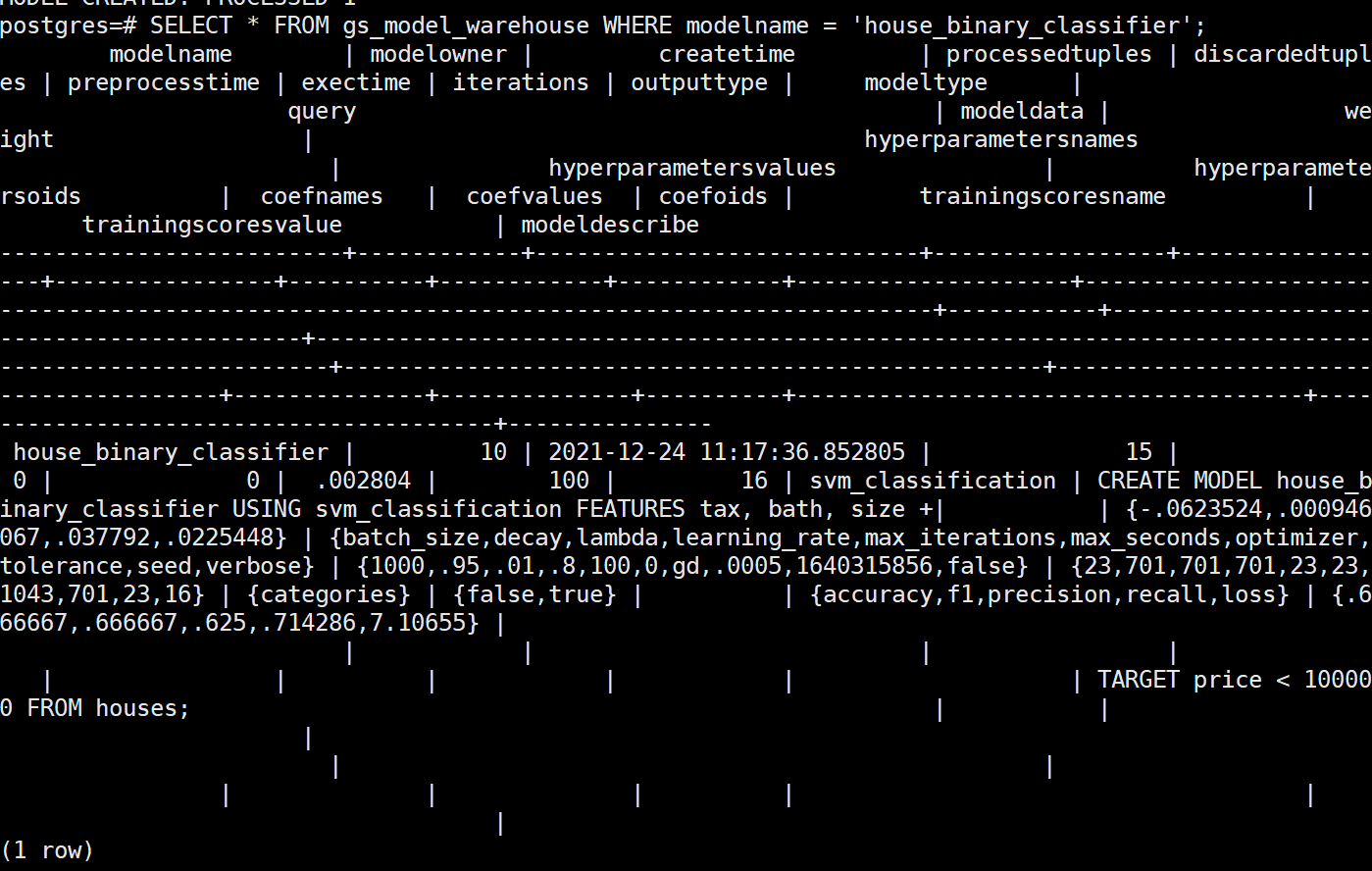
实践思考题2：索引的使用，对于执行SQL有什么好处？除了使用索引和参数外，还有哪些方面可以对数据库进行优化？

1. 索引使用的好处：索引的使用可以加快数据的检索速度，在使用分组和排序子句进行数据检索时，同样可以减少查询中分组和排序的时间提高检索效率，使用WHERE子句中的列上面创建索引，可以加快条件的判断速度。
2. 除了使用索引和参数外还可以通过优化表的设计结构，选择数据库引擎时事务型选 InnoDB ，非事务型选 MyISAM，设计表结构应符合第三范式：更新类应用可让表的个数多些单表的列少一些；分析类的应用可让表个数少些单表的列多些，列名尽量不超过 18 个字符，表的数据类型选择：数字类型优于字符类型；长度小的优于长度大的。变长的 VARCHAR优于定长的 CHAR 。 BLOB 类型用 BINARY VARCHAR 替代，替代不了则用单独的表存放。

# 关卡四、openGauss的DB4AI特性应用

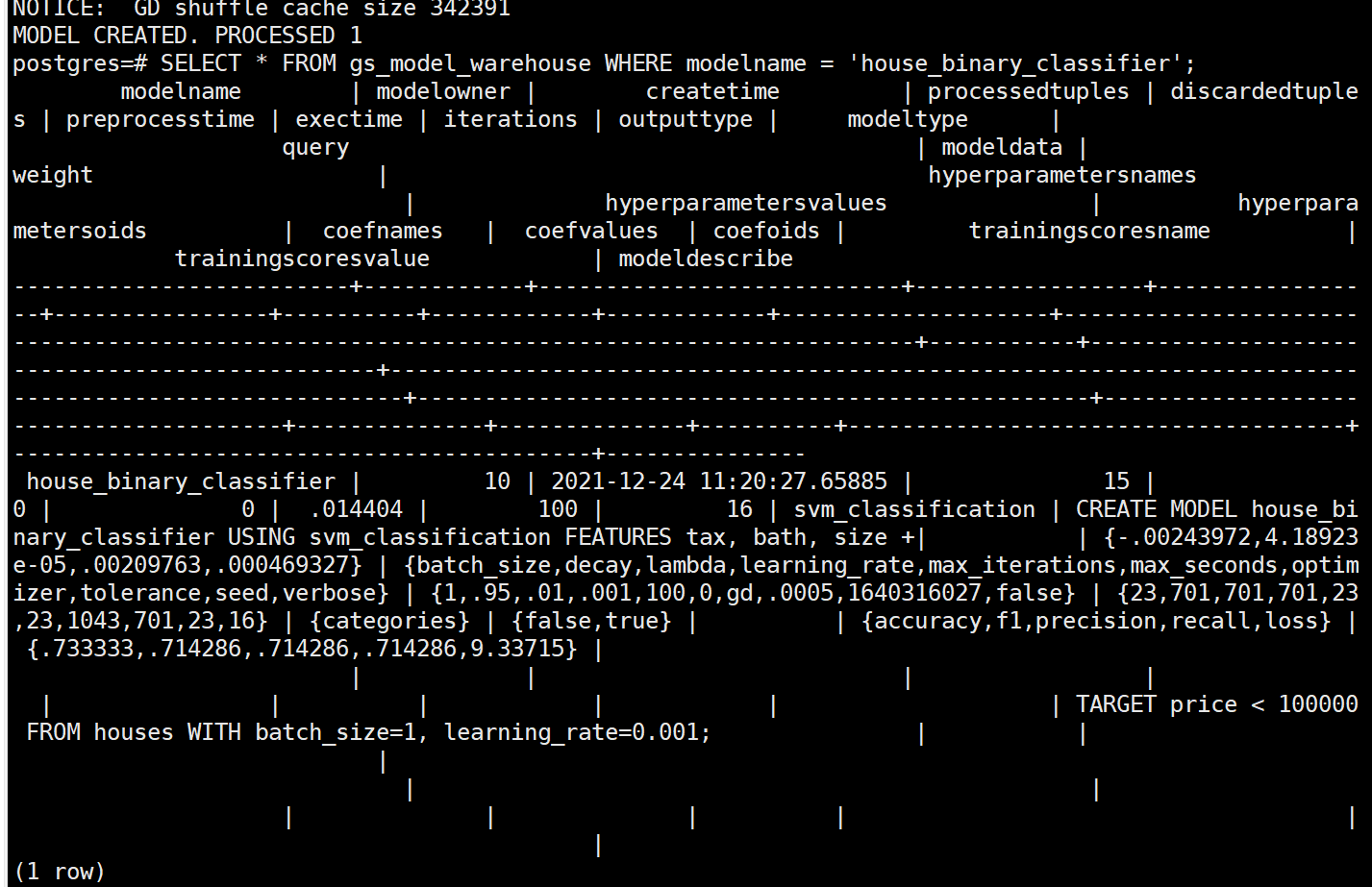
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



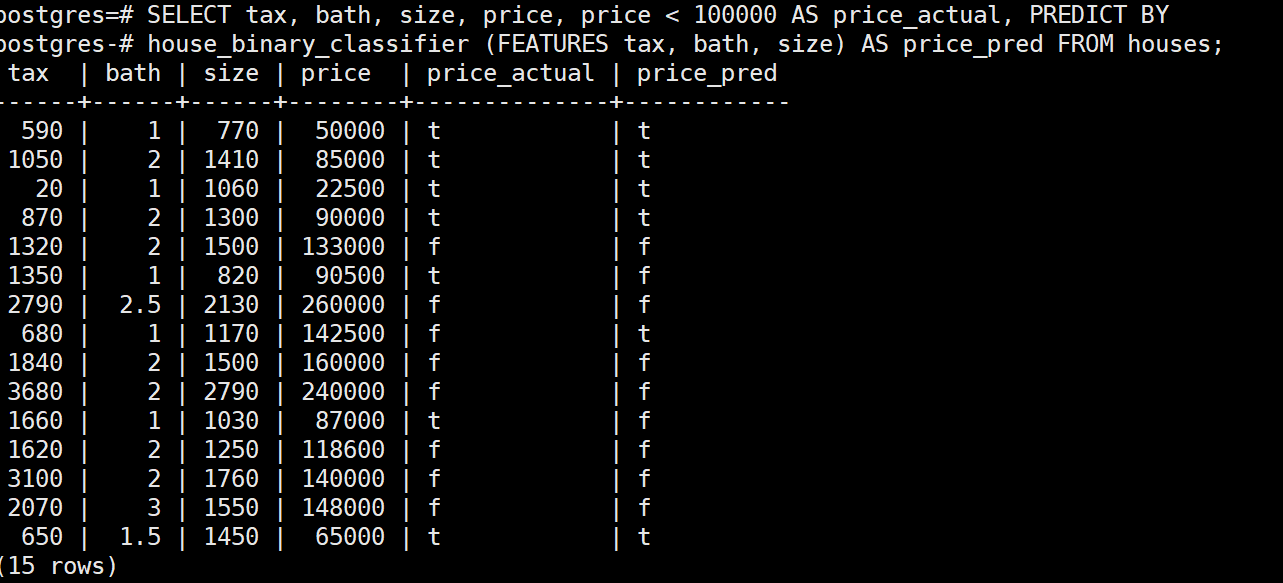
任务二：观察新模型的信息，将执行结果截图。

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？

两者的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。 回归问题是定量问题，分类问题是定性问题。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。

实践思考题2：什么是SVM算法？

SVM是一个二元分类算法，线性分类和非线性分类都支持。经过演进，现在也可以支持多元分类，同时经过扩展，也能应用于回归问题；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。

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？

首先是几个常见的模型评价术语分别是：

1）True positives(TP): 被正确地划分为正例的个数，即实际为正例且被分类器划分为正例的实例数（样本数）；

2）False positives(FP): 被错误地划分为正例的个数，即实际为负例但被分类器划分为正例的实例数；

3）False negatives(FN):被错误地划分为负例的个数，即实际为正例但被分类器划分为负例的实例数；

4）True negatives(TN): 被正确地划分为负例的个数，即实际为负例且被分类器划分为负例的实例数。

分类问题的评价指标有以下几种：

1）正确率（accuracy）

正确率是我们最常见的评价指标，accuracy = （TP+TN）/(P+N)，这个很容易理解，就是被分对的样本数除以所有的样本数，通常来说，正确率越高，分类器越好；

2）错误率（error rate)

错误率则与正确率相反，描述被分类器错分的比例，error rate = (FP+FN)/(P+N)，对某一个实例来说，分对与分错是互斥事件，所以accuracy =1 - error rate；

3）灵敏度（sensitive）

sensitive = TP/P，表示的是所有正例中被分对的比例，衡量了分类器对正例的识别能力；

4）特效度（specificity)

specificity = TN/N，表示的是所有负例中被分对的比例，衡量了分类器对负例的识别能力；

5）精度（precision）

精度是精确性的度量，表示被分为正例的示例中实际为正例的比例，precision=TP/（TP+FP）；

6）召回率（recall）

召回率是覆盖面的度量，度量有多个正例被分为正例，recall=TP/(TP+FN)=TP/P=sensitive，可以看到召回率与灵敏度是一样的。

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？

回归算法的评价指标就是MSE，RMSE，MAE、R-Squared。MSE和MAE适用于误差相对明显的时候，大的误差也有比较高的权重，RMSE则是针对误差不是很明显的时候；MAE是一个线性的指标，所有个体差异在平均值上均等加权，所以它更加凸显出异常值，相比MSE；RMSLE: 主要针对数据集中有一个特别大的异常值，这种情况下，data会被skew，RMSE会被明显拉大，这时候就需要先对数据log下，再求RMSE，这个过程就是RMSLE。对低估值（under-predicted）的判罚明显多于估值过高(over-predicted)的情况（RMSE则相反）

1）平均绝对误差就是指预测值与真实值之间平均相差多大 ：



平均绝对误差能更好地反映预测值误差的实际情况.

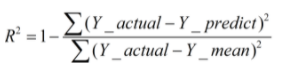
2）均方误差观测值与真值偏差的平方和与观测次数的比值：



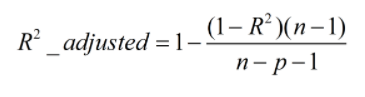
这也是线性回归中最常用的损失函数，线性回归过程中尽量让该损失函数最小。那么模型之间的对比也可以用它来比较。

MSE可以评价数据的变化程度，MSE的值越小，说明预测模型描述实验数据具有更好的精确度。

3）决定系数



4）Adjusted R-Square (校正决定系数）



n样本数量，p为特征数量。消除了样本数量和特征数量的影响