openGauss AI特性创新实践课



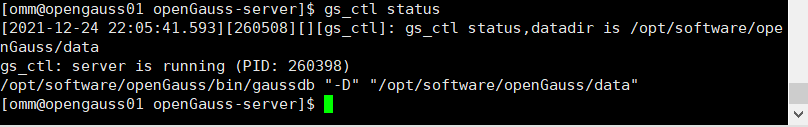
华为技术有限公司

# 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：

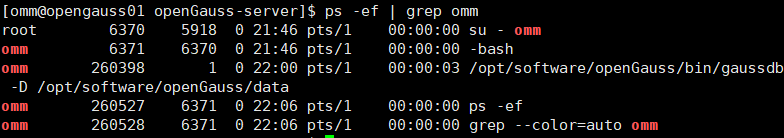
任务一：数据库状态验证

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？

直接 通过下载二进制文件 是打发安装数据库的话可能会因为底层不兼容而失败，因为二进制格式的程序 啊打强依赖于底 层的操作系统和处理器架构，，而通过编译 源码的方式安装，则保证可以得到的二进制文件适配当前处 理器架构和操作系统。

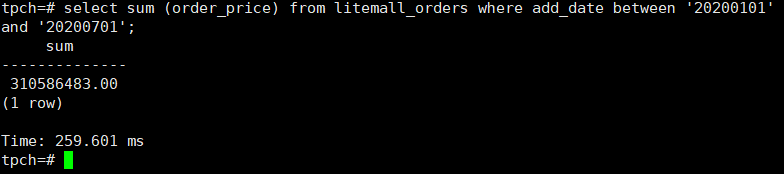
# 关卡二、openGauss数据导入及基本操作

任务一：数据初始化验证

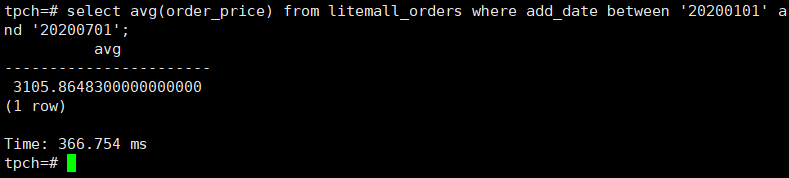
1. 查询supplier表的行数，并将结果进行图：

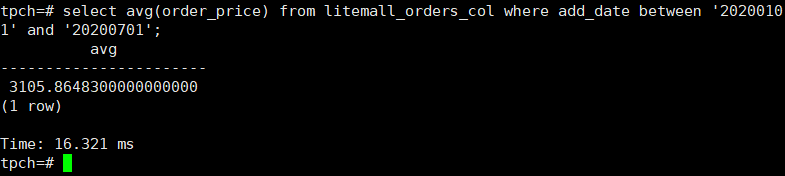
任务二：行存表与列存表执行效率对比

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

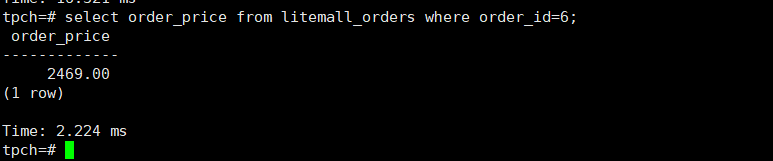


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图



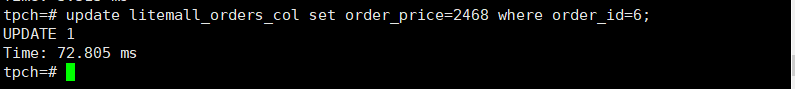


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。



直接通过下载二进制文件安装数据库的话可能会因为底层不兼容而失败

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。

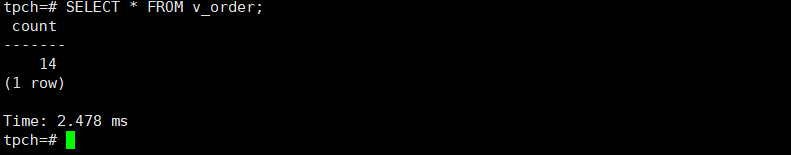


任务三：物化视图的使用

1. 创建物化视图所需要的表后，对表内容进行查询

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。

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。



4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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？

DBMS对行存表和列存表的存储方式不同，所以对于相同的执行计划执行时间不同。在执行对特定行的增加、查找、更新或删除操作时行存表效率更高，在执行对所有行某一列或某些列的操作时，比如聚合函数sum()、avg()等，列存表效率更高。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

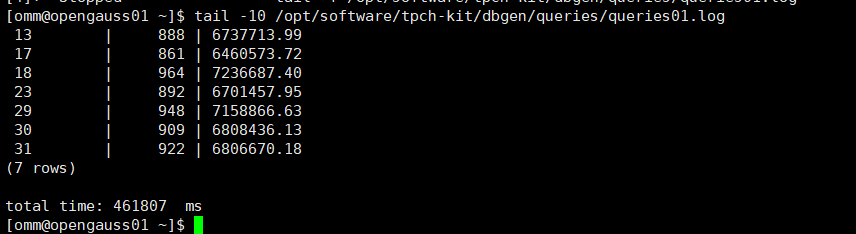
增量物化视图顾名思义就是可以对物化视图增量刷新，需要用户手动执行语句完成对物化视图在一段时间内的增量数据进行刷新。与全量创建物化视图不同在于目前增量物化视图所支持场景较小，目前物化视图创建语句仅支持基表扫描语句或者UNION ALL语句，全量物化视图仅支持对创建好的物化视图做全量更新，而不支持做增量更新；创建全量物化视图语法和CREATE TABLE AS语法一致，不支持对全量物化视图指定NodeGroup创建。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

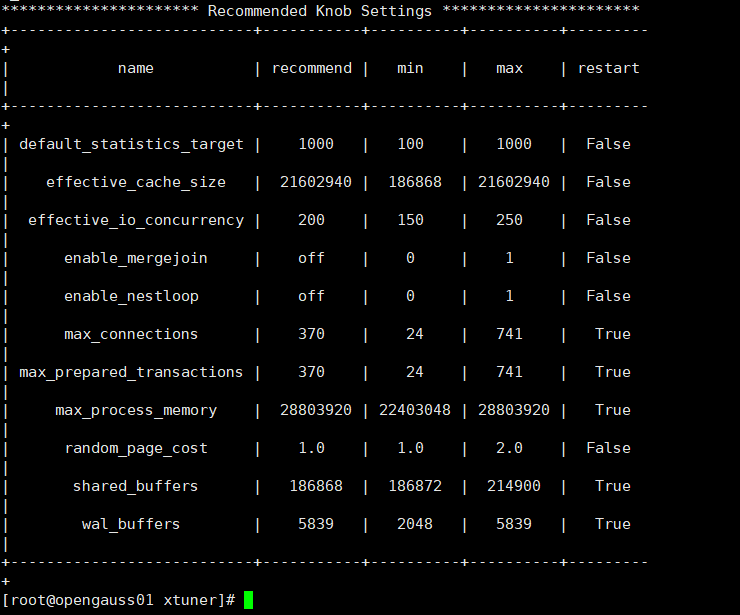
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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



1. 使用root用户，执行X-Tuner进行参数建议优化，将执行结果截图

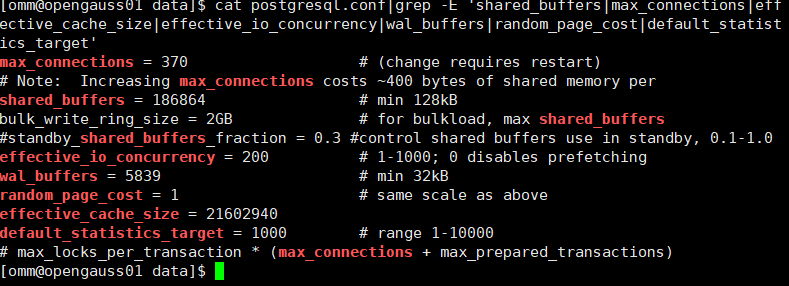
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



1. 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

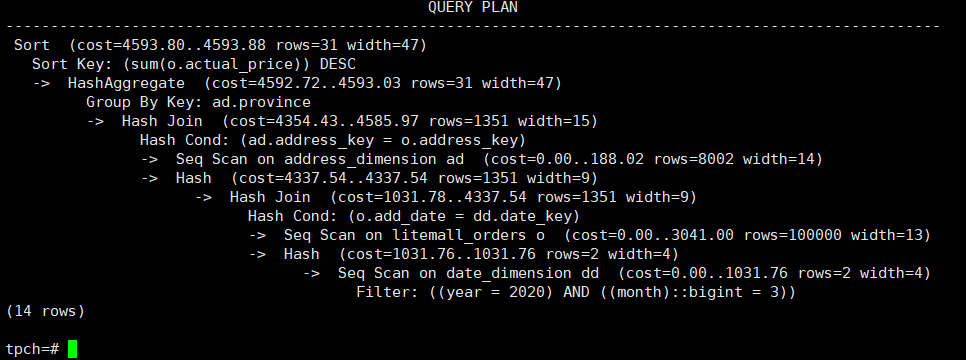
AND o.add\_date = dd.date\_key

AND dd.year = 2020

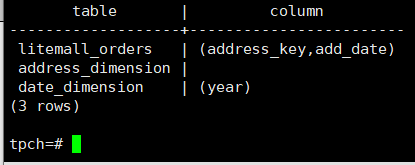
AND dd.month = 3

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

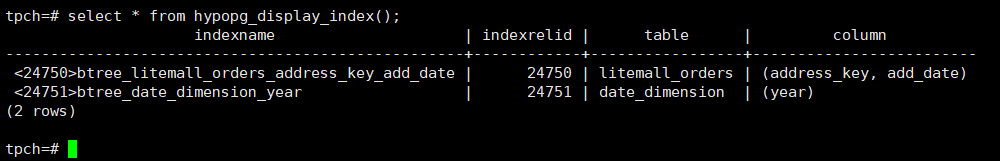
AND dd.month = 3

GROUP BY ad.province

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

1. 查看创建的虚拟索引列，将执行结果截图。

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



1. 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);

1. 再次使用explain，对该SQL加以分析，将执行结果截图。

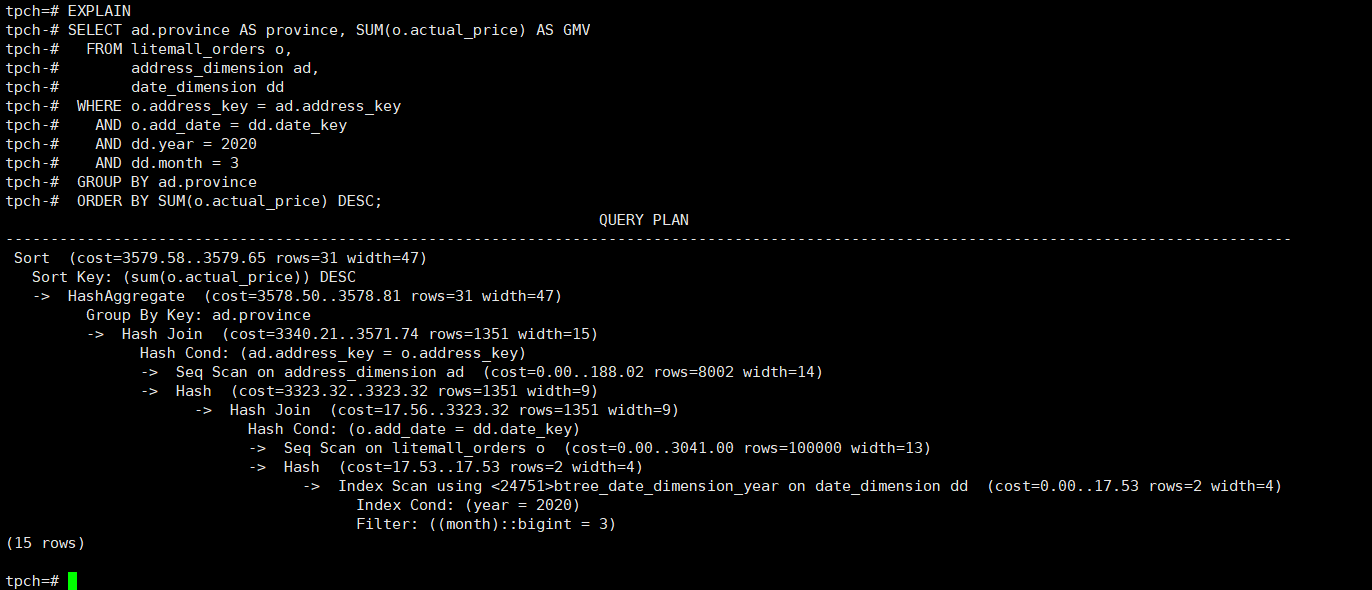
EXPLAIN

SELECT ad.province AS province, SUM(o.actual\_price) AS GMV

FROM litemall\_orders o,

address\_dimension ad,

date\_dimension dd



1. 删除某一个索引虚拟列，将执行结果截图。

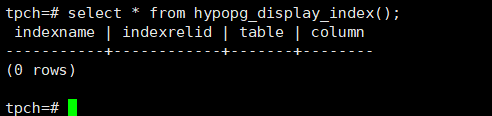
select \* from hypopg\_drop\_index(16715);

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？

答：对以下参数进行了优化：

我们对这个数据库的查询是有特点的（即TPCH测试脚本的内在的特征），针对这些特征，我们可以通过使用X-Tuner分析得到的参数优化建议，来使得对仍符合之前操作特征的后续操作而言，DBMS性能得到提高。

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？

索引的使用，有利于DBMS快速的定位到数据的存储位置，加速SQL的执行。分析日常使 用时对表操作的方式，对于经常进行行增删查改的数据库表，使用行存表，对于经常进行某些列的聚合操作的数据库表，使用列存表。

# 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：分类模型与回归模型有何不同？

答：分类模型是认为模型的输出是离散的，例如大自然的生物被划分为不同的种类，是离散的。 回归模型的输出是连续的 ，例如人的身高变化过程是一个连续过程，而不是离散的。

实践思考题：分类问题有哪些评价指标，请分别说明他们的含义？

学习中遇到的分类任务中的评价指标有准确率（Accuracy）、FPR、FNR、Recall、Precision、F-score、MAP、ROC曲线和AUC等，回归任务中的指标有 (r)MSE、MAE、CC/PCC等。

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。

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。 感知机的模型就是尝试找到一条直线，能够把二元数据隔离开。