openGauss AI特性创新实践课



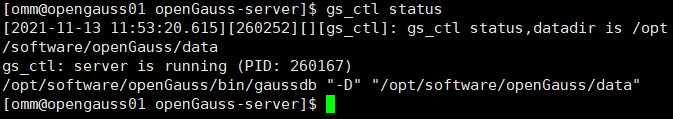
华为技术有限公司

# 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：

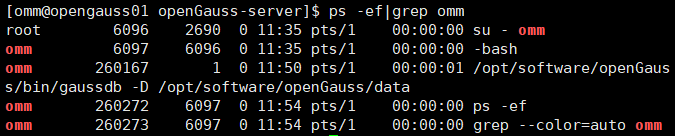
任务一：数据库状态验证

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？

1：源码包的可移植性好，几乎可以在任何 Linux 系统中安装使用，而 RPM 软件包是针对特定系统和架构编写的指令集，必须严格地符合执行环境才能顺利安装（即只会去“生硬地”安装服务程序）。

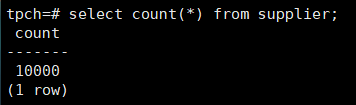
2：使用源码包安装服务程序时会有一个编译过程，因此可以更好地适应安装主机的系统环境，运行效率和优化程度都会强于使用 RPM 软件包安装的服务程序。也就是说，可以将采用源码包安装服务程序的方式看作是针对系统的“量体裁衣”。

# 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：

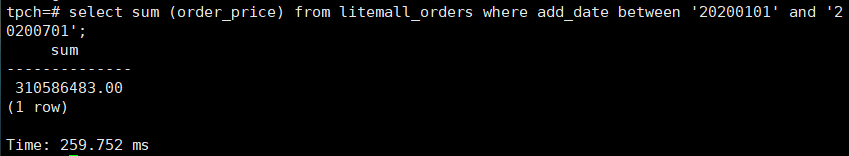
select count(\*) from supplier;;



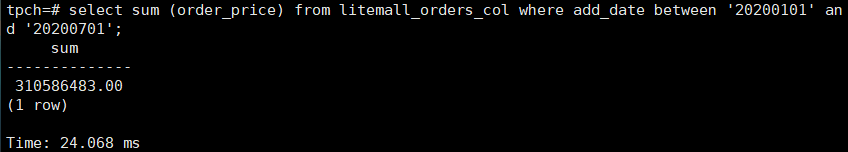
任务二：行存表与列存表执行效率对比

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';

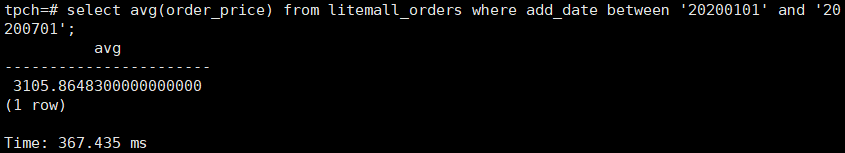


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

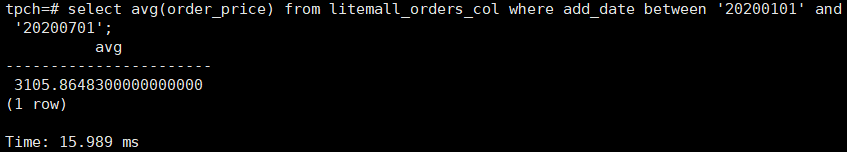


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

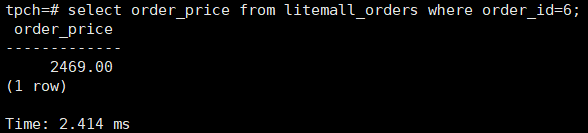


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

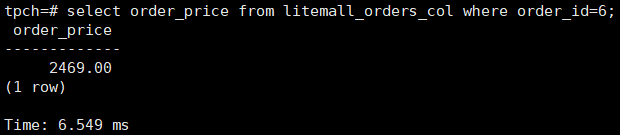


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

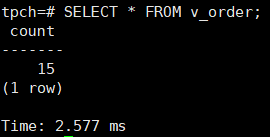
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



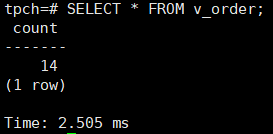
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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？

因为两者存储数据的形式不一致：行存储是以行的形式组织存储，一行是一个tuple，存在一起。当需要读取某列时，需要将这列前面的所有列都进行deform，所以访问第一列和访问最后一列的成本实际上是不一样的。列存储是以列为形式组织存储，每列对应一个或一批文件。读取任一列的成本是一样的，但是如果要读取多列，需要访问多个文件，访问的列越多，开销越大。

在执行仅需获取单列数据的情况时，列存表的效率更高；在执行需要对表中的数据进行更新或者插入等涉及到表中多列数据的查询或更新时，行存表的效率更高。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

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。增量物化视图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

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

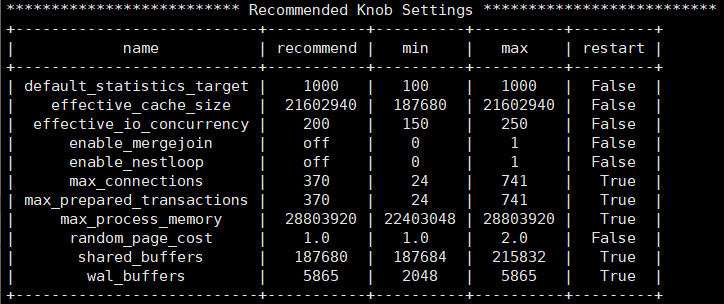
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

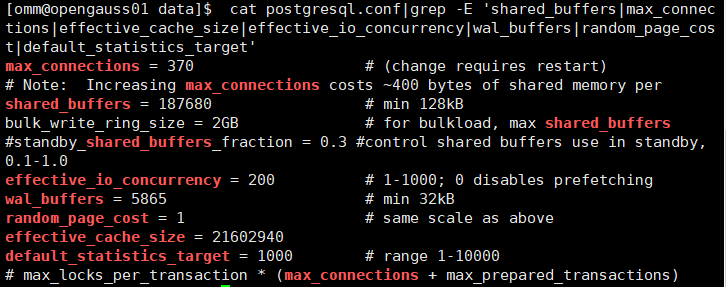
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

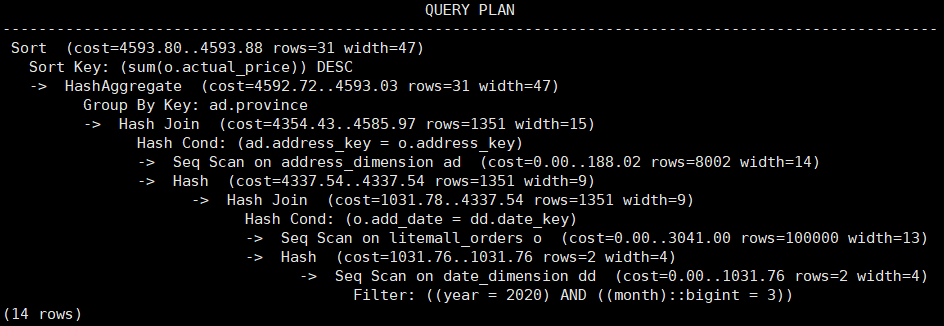
AND o.add\_date = dd.date\_key

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

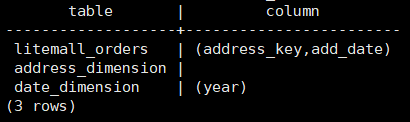
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

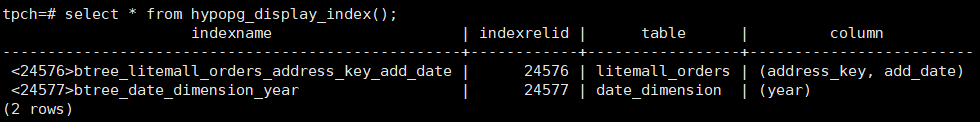
GROUP BY ad.province

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。

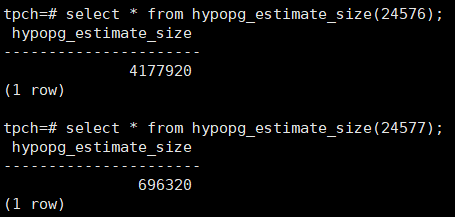
select \* from hypopg\_display\_index();



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

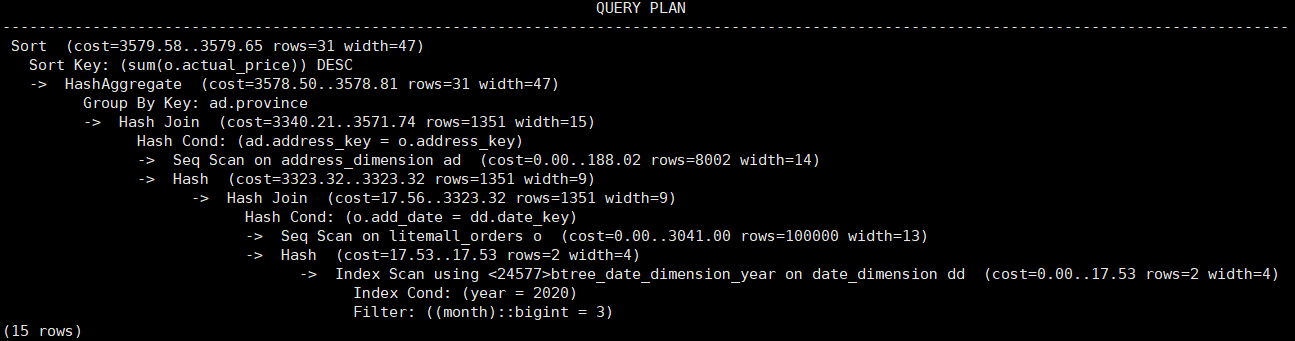
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

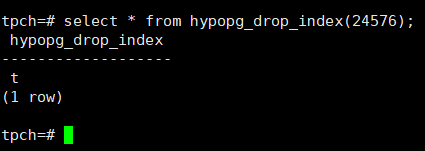
GROUP BY ad.province

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



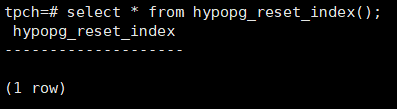
6. 删除某一个索引虚拟列，将执行结果截图。

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



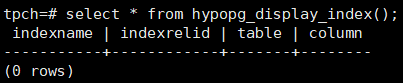
7. 删除某一个索引虚拟列，将执行结果截图。

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



因为本次实验使用的电脑CPU核数为8核，因此可以在SQL脚本使用利用SMP并行技术来降低查询时延。实验中设置了query\_dop=8，执行时间提升了将近2倍

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？

对shared\_buffers、max\_connections、effective\_cache\_size、effective\_io\_concurrency、wal\_buffers、random\_page\_cost、default\_statistics\_target这些参数进行了优化。

因为对这些参数进行调整后，可以更好地适应不同的硬件水平，让硬件对当前的SQL语句的执行有一个更好的支持。

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？

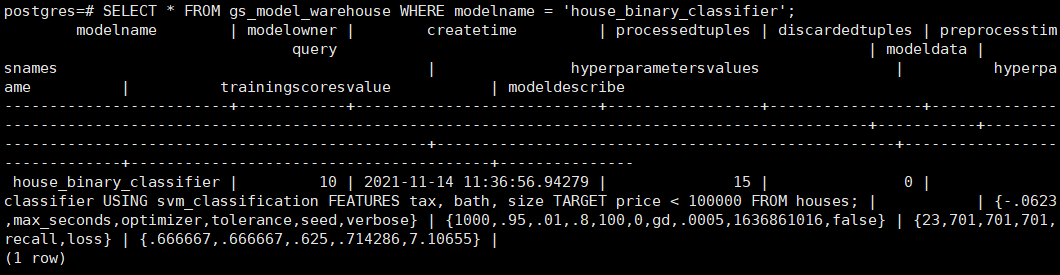
好处：通过唯一性索引（unique）可确保数据的唯一性、加快数据的检索速度、加快表之间的连接、减少分组和排序时间、使用优化隐藏器提高系统性能

优化数据库：可以对数据库进行拆分，将大的数据库分成小的数据，可以使数据的查询效率得到较好的提升。增加缓存，将查询热度较高的数据缓存下来，查询的时候直接返回，。

# 关卡四、openGauss的DB4AI特性应用

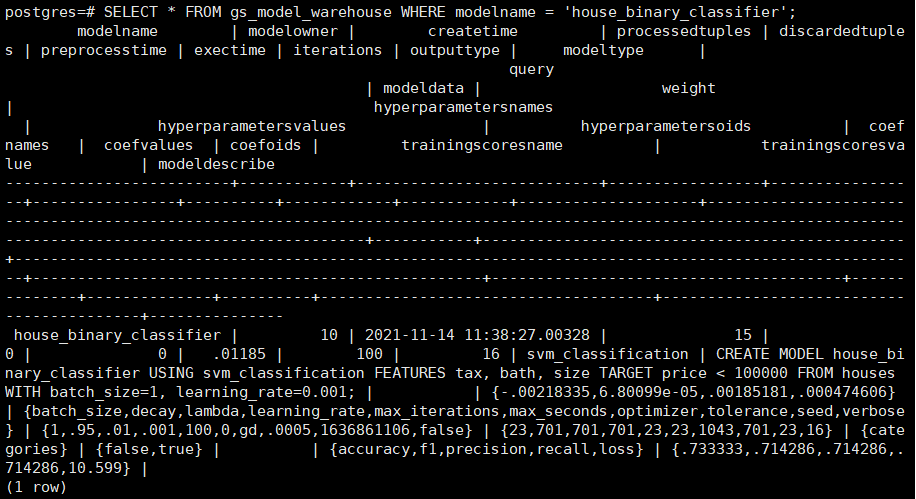
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



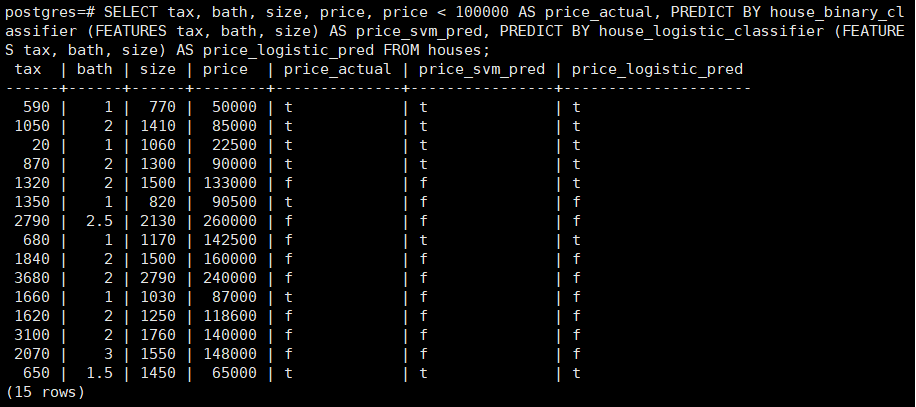
任务二：观察新模型的信息，将执行结果截图。

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？

分类模型和回归模型本质一样，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？

SVM，即支持向量机，是一种监督式学习的方法，可广泛地应用于统计分类以及回归分析。 支持向量机属于一般化线性分类器，这族分类器的特点是他们能够同时最小化经验误差与最大化几何边缘区，因此支持向量机也被称为最大边缘区分类器。

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？

分类问题的评价指标主要有：准确率，精确率，召回率，F1-Score， AUC, ROC

准确率：对于给定的数据，分类正确的样本数占总样本数的比例。

精确率：指在预测为正类的样本中真正类所占的比例

召回率：指在所有的正类中被预测为正类的比例,直观上来说召回率表示我现在预测为正的这些值中，占了所有的为正的样本的多大比例

F1-Score：精确率和召回率的调和平均

AUC：（Area Under roc CurveROC）曲线下的面积，引入AUC的原因是量化评价指标

ROC：在二分类问题中，我们最终得到的数据是对每一个样本估计其为正的概率值（Score），我们根据每个样本为正的概率大小从大到小排序，然后按照概率从高到低，一次将“Score”值作为阈值threshold，当测试样本的概率大于或等于这个threshold时，我们认为它为正样本，否则为负样本。每次选取一个不同的threshold，就可以得到一组FPR和TPR,即ROC曲线上的一点。

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？

回归问题的评价指标主要有：均方误差、均方根误差（标准误差）、平均绝对误差、R-squared

均方误差：观测值与真值偏差的平方和与观测次数的比值

均方根误差：标准误差是均方误差的算术平方根。它的意义在于开个根号后，误差的结果就与数据是一个级别的，可以更好地来描述数据

平均绝对误差：平均绝对误差能更好地反映预测值误差的实际情况

R-squared：R平方介于0~1之间，越接近1，回归拟合效果越好，一般认为超过0.8的模型拟合优度比较高