openGauss AI特性创新实践课



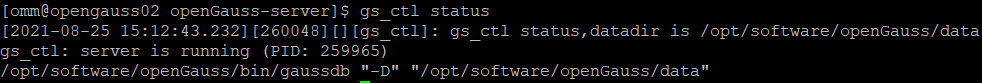
华为技术有限公司

# 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：

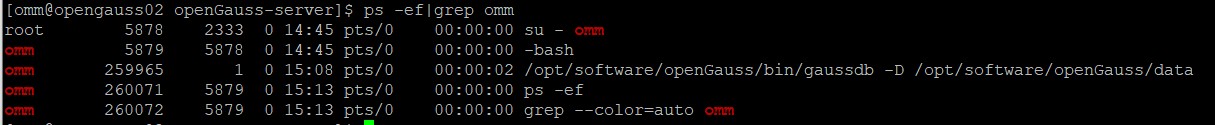
任务一：数据库状态验证

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？

因为软件在编译期间需要配置。openGauss 数据库通过编译安装，是因为要定制存储引擎。此外，软件也需要统一的安装路径。

安装数据库所需要的步骤：

1. 编译前，要使用root用户名登录ECS。创建openGauus数据库的安装用户omm和其属组dbgrp。
2. 创建openGauss源码存放和安装路径后，下载第三方编译库如openEuler。解压安装包之后进行重命名为binarylibs。下载openGauss源码后，使用WinSCP工具吧cmake的安装包上传到服务器内。
3. 使用yum安装依赖包，之后替换python版本，并修改/opt/software路径的用户所属组以及权限。
4. 切换到omm用户，设置omm用户的环境变量，添加环境变量，并写入.bash\_profile，使环境变量生效。
5. 在openGauss源码下生成配置文件，执行make命令来进行编译。
6. 执行make install来进行安装。安装完毕后，可以对数据库的密码进行设置。
7. 启动数据库后，登录数据库，若是子想要更换omm账号密码也可以。
8. 登陆成功后查询数据库版本，数据库就安装成功了。
9. 可以退出数据库。

# 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;;

Text

Description automatically generated

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';

Graphical user interface, text

Description automatically generated

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

Text

Description automatically generated

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';

Text

Description automatically generated

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

Text

Description automatically generated

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;

Text

Description automatically generated

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

Text

Description automatically generated

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;

Text

Description automatically generated

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

Text

Description automatically generated with medium confidence

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;

A picture containing text

Description automatically generated

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;

Text

Description automatically generated

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;

Text

Description automatically generated

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;

A picture containing text

Description automatically generated A picture containing table

Description automatically generated

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;

A picture containing table

Description automatically generated

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？

行存表与列存表在执行相同的SQL语句时，执行时间不同。因为行存储是一次性完成的，而列存储在每次读取的数据是集合中的一段或者全部。

在执行数据查询和修改数据类型的SQL时，行存表效率更高。

在执行总和查询和平均值查询的SQL时，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？

全量物化视图只能支持对于创建好的区划试图做全量更新，但是不能支持做增量更新。创建全量物化视图语法和CREATE TABLE AS 语法不一样，不能支持对全量物化指定的NodeGroup创建

增量物化视图可以对物化视图增量刷新，需要我们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。

这两者不同的地方就是增量物化视图所支持的场景较小，物化视图创建的语句也只是支持及表扫描语句或者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

Text

Description automatically generated with medium confidence

Text

Description automatically generated

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

Text

Description automatically generatedA picture containing graphical user interface

Description automatically generated

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'

A screen shot of a computer

Description automatically generated with low confidence

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;

Text

Description automatically generated

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

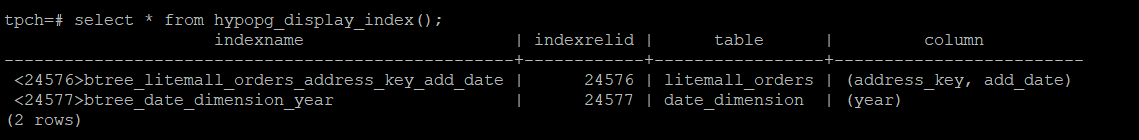
ORDER BY SUM(o.actual\_price) DESC');

Text

Description automatically generated

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);

Text

Description automatically generated

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;

Text

Description automatically generated

6. 删除某一个索引虚拟列，将执行结果截图。

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

Text

Description automatically generated

7. 删除某一个索引虚拟列，将执行结果截图。

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

Text

Description automatically generated

8. 查看索引虚拟列，将执行结果截图。

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

A picture containing graphical user interface

Description automatically generated

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

Text

Description automatically generated

Text

Description automatically generated

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

A computer screen capture

Description automatically generated with low confidence

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？

对以下的参数进行优化：

A picture containing graphical user interface

Description automatically generated

要对这些参数做优化是为了提升数据库运行的性能。在默认的TPC-H数据库的表中缺少了索引，所以如果数据库的参数没有做任何的优化，执行效率会比较差。

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？

使用索引对于执行SQL的好处：

* 可以通过唯一性索引 (unique) 来确保数据的唯一性。
* 索引可以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
* 使用优化隐藏器来以提高系统性能

除了使用索引和参数外，还可以通过SQL语句以及有效索引、数据结构、系统配置和硬件来对数据库进行优化。

* SQL以及索引优化最重要。根据需求要写结构良好的SQL，再根据SQL在表中建立有效的索引
* 设计表结构时，要考虑如何能有有效的查询
* 系统配置的优化。openGauss 数据库的内核是基于Taishan服务器，所以在openEuler操作系统上，为了达到数据库的极致性能，所以来的关键系统要进行调优配置
* 硬件配置的优化。由于openGauss支持使用SSD盘作为数据的组存储设备，但必须是SAS接口的SSD盘，以RAID的方式部署使用。

# 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';

A picture containing text, screenshot, monitor, black

Description automatically generated

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';

A screenshot of a computer

Description automatically generated

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;

Graphical user interface, text

Description automatically generated

实践思考题1：分类模型与回归模型有何不同？

分类和回归的却别是它们输出的变量的类型。回归模型输出变量的类型也成为定量输出，或者连续变量预测；而分类模型则成为定性输出，或者离散变量预测。分类模型可以将回归模型的输出离散化，而回归模型可以将分类模型的输出连续化。

实践思考题2：什么是SVM算法？

SVM算法是一种二分类模型，他的基本模型可以定义为特征空间上的间隔最大的线性分类器，其学习策略是间隔最大化，最后可以转化为一个凸二次规划问题的求解。

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？

* 混淆矩阵(Confuse Matrix)

将实力分成正类 (positive) 或负类 (negative)，在实际分类中会出现一下四种情况：

1. 若一个实力是正类，并且被预测为正类，即是真正类TP (True Positive)
2. 若一个实力是正类，并且被预测为负类，即是假负类FN (False Negative)
3. 若一个实力是负类，并且被预测为正类，即是假正类FP (False Positive)
4. 若一个实力是负类，并且被预测为负类，即是真负类TN (True Negative)

|  |  |  |
| --- | --- | --- |
| 真实标签  预测标签 | 正例 | 反例 |
| 正例 | TP（真正类） | FN（假负类） |
| 反例 | FP（假正类） | TN（真负类） |

* 准确率(Accuracy)

准确率的缺点是数据的样本不平均，不能用来评价模型的性能优劣

* 精确率(Precision)

精确率是在模型预测为正样本的结果中，真正是正样本所占的百分比，分类阈值比较高。

* 召回率(Recall)

在史记为正样本中，被预测为正样本所占的百分比，它尽量检测数据，不遗漏数据，所以分类阈值较低。

* F1 Score

精准率和召回率都有各自的缺点，如果阈值较高，精准率就会高，但同时回一楼很多数据，反之阈值较低，召回率就会高，但预测的不准确。

F1 Score 是一种调和平均数。

* P-R曲线(Precesion-Recall Curve)

对于所有正样本，用来描述精准率和召回率变化的曲线。

模型月座标围城的面积越大，模型的性能就越好。由于曲线下的面积很难估算，BEP (Break-Event Point)，也称为平衡点，就是当P = R时的取值，平衡点的取值越高，性能更优。

Chart, radar chart

Description automatically generated

* ROC (Receiver Operating Characteristic)

又称为接收者操作特征曲线。理想的目标为TPR=1, FPR=0。所以ROC曲线越靠拢(0,1)点，越偏离45度对角线越好，对应的就是TPR越大越好，FPR越小越好。

* AUC (Area Under Curve)

这个曲线是处于ROC曲线下的那部分面积的大小。AUC越大，代表模型的性能越好。

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, Mean Absolute Error)

对绝对误差损失的预期值

* 平均绝对百分比误差 (MAPE, Mean Absolute Percentage Error)

对相对误差损失的预期值，也就是绝对误差和真值的百分比

* 均方误差 (MSE, Mean Squared Error)

对应于平方（二次）误差的期望

* 均方误差跟 / 均方根误差 (RMSE, Root Mean Squared Error)

对应于平方（二次）误差的期望

* 均方误差对数 (MSLE, Mean Squared Log Error)

对应平方对数（二次）差的预期

* 中位绝对误差 (MedAE, Median Absolute Error)

通过取目标和预测之间的所有绝对差值的中值来计算损失

* R Squared (R2 Score)

反映了自变量x对因变量y的变动的解释的程度。越接近1，说明模型的拟合越好。

表示了y的变动的程度，正比于方差

表示了模型和真实值的残差

表示了模型对y的变动的预测