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

文本

描述已自动生成

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）

文本

描述已自动生成

实验思考题：为什么需要通过源码编译，安装数据库？

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;;

文本

描述已自动生成

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';

文本

描述已自动生成

文本

描述已自动生成

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';

文本

描述已自动生成

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

文本

描述已自动生成

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;

文本

描述已自动生成

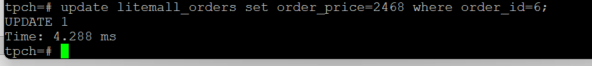
select order\_price from litemall\_orders\_col where order\_id=6;

文本

描述已自动生成

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;

图形用户界面

中度可信度描述已自动生成

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;

图形用户界面, 文本

描述已自动生成

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;

文本

描述已自动生成

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;

文本

中度可信度描述已自动生成

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;

图片包含 文本

描述已自动生成

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？

**行式存储**

优点1.数据被保存在一起; 2.INSERT/UPDATE容易

缺点 1.选择(Selection)时即使只涉及某几列，所有数据也都会被读取

**列式存储**

优点1.查询时只有涉及到的列会被读取; 2.投影(projection)很高效; 3.任何列都能作为索引

缺点：1.选择完成时，被选择的列要重新组装;2.INSERT/UPDATE比较麻烦

实践思考题2：全量物化视图与增量物化视图有哪些差别？

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

文本

描述已自动生成

电脑的屏幕

描述已自动生成

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

图形用户界面, 文本

描述已自动生成

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'

图形用户界面, 文本

描述已自动生成

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;

文本

描述已自动生成

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

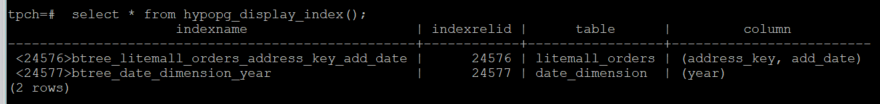
ORDER BY SUM(o.actual\_price) DESC');

图片包含 图形用户界面

描述已自动生成

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);

文本

描述已自动生成

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;

文本

描述已自动生成

6. 删除某一个索引虚拟列，将执行结果截图。

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

文本

描述已自动生成

7. 删除某一个索引虚拟列，将执行结果截图。

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

文本

描述已自动生成

8. 查看索引虚拟列，将执行结果截图。

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

日程表

中度可信度描述已自动生成

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

文本

描述已自动生成

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

文本

描述已自动生成

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？

通过命令行指定：分别传递 –db-name –db-user –port –host –host-user 参数，可选 –host-ssh-port 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？

好处：

第一通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。  
第二，可以大大加快 数据的检索速度，这也是创建索引的最主要的原因。  
第三，可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义。  
第四，在使用分组和排序 子句进行数据检索时，同样可以显著减少查询中分组和排序的时间。  
第五，通过使用索引，可以在查询的过程中，使用优化隐藏器，提高系统的性能。

还有事务，锁定表，外键等可以优化数据库进行优化

# 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';

文本

描述已自动生成

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';

文本

描述已自动生成

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;

文本

描述已自动生成

实践思考题1：分类模型与回归模型有何不同？

 回归会给出一个具体的结果，例如房价的数据，根据位置、周边、配套等等这些维度，给出一个房价的预测。         分类相信大家都不会陌生，生活中会见到很多的应用，比如垃圾邮件识别、信用卡发放等等，就是基于数据集，作出二分类或者多分类的选择。

浅层： 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。 中层： 回归问题是定量问题，分类问题是定性问题。 高层： 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的全称是Support Vector Machine，即支持向量机，主要用于解决模式识别领域中的数据分类问题，属于有监督学习算法的一种。SVM要解决的问题可以用一个经典的二分类问题加以描述。

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？

分类问题是用于将事物打上一个标签，通常结果为离散值。例如判断一幅图片上的动物是一只猫还是一只狗，分类通常是建立在回归之上，分类的最后一层通常要使用softmax函数进行判断其所属类别。分类并没有逼近的概念，最终正确结果只有一个，错误的就是错误的，不会有相近的概念。最常见的分类方法是逻辑回归，或者叫逻辑分类。

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？

回归问题通常是用来预测一个值，如预测房价、未来的天气情况等等，例如一个产品的实际价格为500元，通过[回归分析](https://so.csdn.net/so/search?q=%E5%9B%9E%E5%BD%92%E5%88%86%E6%9E%90&spm=1001.2101.3001.7020)预测值为499元，我们认为这是一个比较好的回归分析。一个比较常见的回归算法是线性回归算法（LR）。另外，回归分析用在神经网络上，其最上层是不需要加上softmax函数的，而是直接对前一层累加即可。回归是对真实值的一种逼近预测。