Homework3 Database Accessor 2

本实验要求两人完成，其中一个人扮演的角色是类实现者、另一个人扮演的角色是类使用者。

类实现者应与类使用者使用相同的编程语言。每个人需给出所写程序的设计图（使用UML绘制类图）、代码注释说明。

|  |
| --- |
| 编程要求（**以下是强制要求，不是可选要求，任何一条不满足，扣10分**）：  一个程序之中类的个数不少于2个；  类名采用像DBAccessor、TableConfig这样的每个单词第一个字母用大写的命名方式、禁止使用**汉语拼音**命名类名、方法名、成员变量名、临时变量名；  一个文件只能存放一个类；  每个类的方法数量不少于5个，不多于50个；  每个类必须提供至少一个方法：打印其所有成员变量的值  每个方法体内的代码长度**不大于50行**；（请在打印代码时代码左侧提供代码行号）  每个方法内部的嵌套深度**不大于2层**；  每个方法体内的if块、for/while/switch/case/块之前或之后**必须空一行**；  每个方法体内if块内如果所有出口都通过return跳出，则**禁止在**if块后连接else块或elseif块；  每个方法的输入参数的个数不大于5个；  每个方法需提供说明：输入/输出参数的类型+含义+缺省值、可能抛出什么类型的异常；  测试方法采用testConnect, testInsertRecords, testXXX这样的命名；  方法体内变量/成员变量/方法命名采用updateTable、insertRow、tableName这样的驼峰命名方式；  除有特殊含义（如a11表示矩阵第一行第一列元素），禁止使用a1、t2、flag这样的命名。  如果能用普通代码完成任务的，禁止用反射编程。 |

**类实现者的工作：**

1. 扩展Database访问类Accessor为两个子类，一个是MySQLAccessor，一个是PostgreSQLAccessor，以支持两种数据库引擎的不同连接方式，同时把数据操作的公共部分放入Accessor父类中，包括建表、读取/插入/更新/删除表中满足条件的记录等。

Accessor类及其子类所提供的每一个数据库表的操作方法（如：建表、增、删、改、查等）都必须提供在操作失败的情况下，重新打开连接并重新尝试执行失败的操作，尝试次数由配置文件确定。

2）实现一个CSVLoader类，将（事先未给定的）任意但符合CSV格式的时间序列数据装入与CSV文件名相同的表中。

CSVLoader类还支持一个功能：将数据库的（事先未给定的）任意表，导出为CSV文件。

（附件提供了两个CSV测试文件－请在编程时误以这个文件的格式为准，因为后继还有其它格式文件）

3）实现一个DBConvolution类，实现一个方法***void conv(patternVector, tableName, fieldName, resultFieldName)***，将patternVector与tableName表中的fieldName列组成的向量做卷积，并将卷积得到的结果向量存入tableName表中的resultFieldName。

**类使用者的工作：**

1. 用Java/C++/Python之一，实现对MySQLAccessor、PostgreSQLAccessor进行测试、编写**自动化**测试程序，针对下述三个验证，分别写一个测试函数：

增加N条记录{A\_i}是否成功：验证方法是再读入N条记录{B\_i}，并逐个比对A\_i 与B\_j是否相同

删除N条记录{A\_i}是否成功：验证方法是再读入N条记录{B\_i| B\_i.id = A\_i.id}，验证读不到结果。

更新N条记录{A\_i}是否成功：验证方法是再读入N条记录{B\_i}，并逐个比对A\_i 与B\_j是否相同

1. 开发最大收益随机搜索器ProfitSearcher，该类依赖类DBConvolution, BuyPointer, ProfitComp，来完成如下方法：

实现RandomConv方法

(以下参数tableName, fieldname,RAfield,alpha,...等，请设计一种方法作为ProfitSearcher的成员变量、初始化时传入。

pre\_profit = 0

profit = 1

While( abs(profit-pre\_profit) >1e^-6)：

随机生成两个正向量A、B，A的所有分量之和为1，B的所有分量之和也为1.调用

***DBConvolution.conv(A, tableName, fieldname, resultFieldName= RAfield)***

***DBConvolution.conv(B, tableName, fieldname, resultFieldName=RBfield)***

***BuyPointer. setBuySellPoint(tableName, RAfield, RBfield, alpha, beta, setBuyPoint, setSellPoint)***

***pre\_profit = profit***

***profit = ProfitComp. compProfit(intialWealth, tableName, buypointField, sellpointField, closeField, setWealth, setStock, precision)***

BuyPointer、ProfitComp类说明如下：

2.1）开发BuyPointer类

该类实现的一个主要方法是***setBuySellPoint(tableName, RAfield, RBfield, alpha, beta, setBuyPoint=true, setSellPoint=true)***，即

逐行读取表tableName中的每条记录，若当前记录的RAfield字段>alpha \* RBfield字段，且***setBuyPoint***=true则在tableName的buypoint字段上记录1；若当前记录的RAfield字段<beta \* RBfield字段，且***setSellPoint*** =true则在tableName的字段sellpoint字段上记录1。

setBuySellPoint方法体内部必须检查表tableName存在，字段RAfield、RBfield存在，若sellpoint字段不存在，且***setBuyPoint*** =true，则创建之，同理对于***setSellPoint=true***的情况做类似处理。

2.3) 开发ProfitComp类

该类实现一个主要方法是***compProfit(intialWealth, tableName, buypointField, sellpointField, closeField, setWealth=true, setStock=true, precision=6)***，其中precision定义了小数点后的计算精度，其它字段含义如下：

若setWealth=true且表tableName没有wealth字段，则在表tableName中创建wealth字段，反之创建wealth变量。同理，对于setStock参数做类似处理。

以w=initialWealth元本金作为tableName的第一条记录的wealth字段/变量的值、以0股份（以下用变量s表达）作为tableName表的第一条记录的stock字段/变量的值。

逐行读取***tableName***中的每条记录：

1）若当前记录的***buypointField***字段为1且当前本金w足够的情况下，买入新增股份t=1000元/当前记录***closeField***字段值，并将

s更新为s+t

w更新为w-1000

当前记录的stock字段/变量的值设置为t的值

当前记录的wealth字段/变量的值设置为-1000的值

2) 若当前记录的***sellpointField***字段为1则卖出当前全部股份s，获得新增本金v=s\*当前记录***sellpointField***字段值，并将

s更新为0

w更新为w+v

当前记录的stock字段/变量的值设置为0的值

当前记录的wealth字段/变量的值设置为v的值

3) 当处理完tableName最后一条记录时，若s≠0，则全部卖出股份s，获得新增本金v=s\*当前记录***closeField***字段值，并将

s更新为0

w更新为w+v

当前记录的stock字段/变量的值设置为0的值

当前记录的wealth字段/变量的值设置为v的值

计算收益return = (w- ***intialWealth***)/ ***intialWealth***，精确到小数点后precision位

4) 验证收益

若***setWealth=true***，对***tableName***表的wealth字段所有列求和，假定为sw，计算return2=(sw- ***intialWealth***)/ ***intialWealth***，验证return2与3)中的return相等，并打印误差。

11月19号，以两个人一组提交打印版作业。