**西北农林科技大学信息工程学院**

**《云计算技术实践》**

**实习报告**

题 目： **实现超市商品管理系统**

|  |  |
| --- | --- |
| **学 号** | 2020012797 |
| **姓 名** | 邓星宇 |
| **专业班级** | 数科2003 |
| **指导教师** | 刘敬敏 |
| **实践日期** | 2022年12月12日-23日 |

目 录

[一、综合实践目的与要求 1](#_Toc271547901)

[二、综合实践任务 1](#_Toc271547902)

[三、总体设计 1](#_Toc271547903)

[四、详细设计说明 1](#_Toc271547904)

[五、调试与测试 1](#_Toc271547905)

[六、实习日志 1](#_Toc271547906)

[七、实习总结 1](#_Toc271547907)

[八、附录：核心代码清单 1](#_Toc271547908)

# 一、综合实践目的与要求

本综合训练是数据科学与大数据技术专业重要的实践性环节之一，是在学生学完《云计算》课程后进行的一次全面的综合练习。本课综合训练的目的和任务：

1. 巩固和加深学生对云计算课程的基本知识的理解和掌握；

2. 掌握基于云计算平台进行程序设计的能力；

3. 掌握书写数据并行，规约求和程序设计文档的能力；

4. 提高解决大数据相关实际问题的能力。

# 二、综合实践任务

功能需求：

任务：实现超市商品管理系统

数据：包括不少于100个商品，每个商品的登记内容包括：商品名称、商品ID、类型、价格、货架、重量等，具体数据可随机生成也可在网上查找。

使用HBase Java编程实现，运行程序，输出如下提示信息：

（1）添加数据：增加新商品，根据商品ID，增加到商品目录里；

（2）删除数据：超市不再售卖此商品，删除商品；

（3）修改数据：根据卖出量、进货量、商品位置变换等修改商品信息；

（4）查找数据：根据商品ID查找商品所在货架及价格等。

# 三、总体设计

实现超市商品管理系统需要考虑以下几个方面：

1.创建HBase表：首先，需要创建一个HBase表来存储商品信息。可以使用HBase的Java API来创建表并设置表的列族。

2.添加数据：可以使用HBase的Java API来向表中插入数据。可以使用Put类来插入一行数据，并使用addColumn()方法向Put对象中添加列族、列名和值。

3.删除数据：可以使用HBase的Java API来删除表中的数据。可以使用Delete类来删除一行数据，并使用deleteColumn()方法删除指定列族、列名的列。也可以使用deleteFamily()方法删除整个列族。

4.修改数据：可以使用HBase的Java API来修改表中的数据。可以使用Put类来修改一行数据，并使用addColumn()方法更新列族、列名和值。

5.查找数据：可以使用HBase的Java API来查找表中的数据。可以使用Get类来获取一行数据，并使用getValue()方法获取指定列族、列名的列值。

# 四、详细设计说明

## 添加数据：

**总体思路**

1.建立HBase连接：首先，需要使用HBase的Java API建立HBase的连接。可以使用ConnectionFactory类的createConnection()方法建立连接。

2.获取HBase表：然后，需要获取HBase表的引用。可以使用Connection对象的getTable()方法获取表的引用

。

3.创建Put对象：接着，需要创建一个Put对象，用于插入数据。可以使用Put类的构造函数创建Put对象，并指定行键。

4.向Put对象中添加列：然后，可以使用Put对象的addColumn()方法向其中添加列族、列名和值。

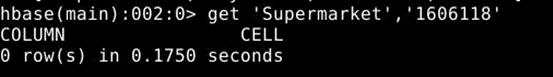
5.使用Table对象的put()方法将Put对象插入表中：最后，可以使用Table对象的put()方法将Put对象插入表中。

**程序代码**

**import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.hbase.\*;  
import org.apache.hadoop.hbase.client.\*;  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
  
public class AddData{  
 public static Configuration *configuration*;  
 public static Connection *connection*;  
 public static Admin *admin*;  
 public static void main(String[] args)throws IOException{  
 *init*();  
 *putData*("Supermarket");  
 *close*();  
 }  
  
 public static void init(){  
  
 *configuration* = HBaseConfiguration.create();  
 try{  
  
 *connection* = ConnectionFactory.createConnection(*configuration*);  
  
 *admin* = *connection*.getAdmin();  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
  
 public static void close(){  
 try{  
 if(*admin* != null){  
 *admin*.close();  
 }  
 if(null != *connection*){  
 *connection*.close();  
 }  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
 public static void putData(String tableName) throws IOException {  
 FileReader reader = new FileReader("/root/IdeaProjects/Test/data/add.dat");  
 BufferedReader br = new BufferedReader(reader);  
 String str = null;  
 Table table = *connection*.getTable(TableName.valueOf(tableName));  
 while((str = br.readLine()) != null) {  
 System.*out*.println("processing line:" + str);  
 String[] cols = str.split(",");  
 Put put = new Put(cols[0].getBytes());  
 put.addColumn("ID".getBytes(), "".getBytes(), cols[1].getBytes());  
 put.addColumn("Name".getBytes(), "".getBytes(), cols[2].getBytes());  
 put.addColumn("Type".getBytes(), "".getBytes(), cols[3].getBytes());  
 put.addColumn("Price".getBytes(), "".getBytes(), cols[4].getBytes());  
 put.addColumn("Place".getBytes(), "".getBytes(), cols[5].getBytes());  
 put.addColumn("Weight".getBytes(), "".getBytes(), cols[6].getBytes());  
 /\*put.addColumn("Course".getBytes(), "Chinese".getBytes(), cols[6].getBytes());\*/  
 table.put(put);  
 }  
 table.close();  
 br.close();  
 reader.close();  
 }  
}**

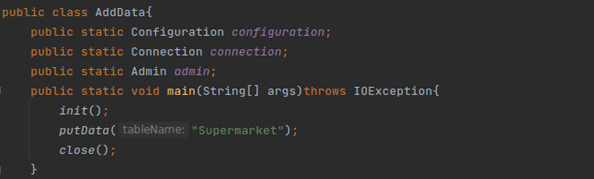
**执行结果**

执行前

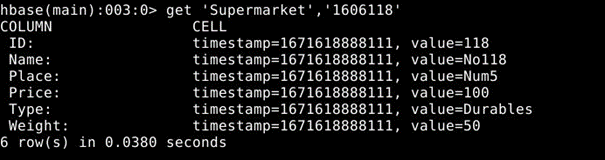


调用AddData()方法





执行后



## 删除数据：

**总体思路**

1.建立HBase连接：首先，需要使用HBase的Java API建立HBase的连接。可以使用ConnectionFactory类的createConnection()方法建立连接。

2.获取HBase表：然后，需要获取HBase表的引用。可以使用Connection对象的getTable()方法获取表的引用。

3.创建Delete对象：接着，需要创建一个Delete对象，用于删除数据。可以使用Delete类的构造函数创建Delete对象，并指定行键。

4.删除Delete对象中的列：然后，可以使用Delete对象的deleteColumn()方法删除指定列族、列名的列，或者使用deleteFamily()方法删除整个列族。

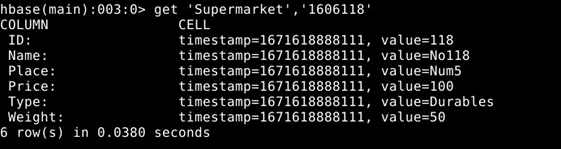
5.使用Table对象的delete()方法将Delete对象提交到表中：最后，可以使用Table对象的delete()方法将Delete对象提交到表中。

**程序代码**

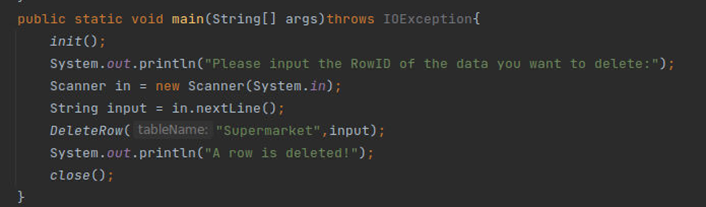
**import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.hbase.\*;  
import org.apache.hadoop.hbase.client.\*;  
import org.apache.hadoop.hbase.util.Bytes;  
import java.io.IOException;  
import java.util.Scanner;  
  
public class DeleteData{  
 public static Configuration *configuration*;  
 public static Connection *connection*;  
 public static Admin *admin*;  
  
 public static void init(){  
  
 *configuration* = HBaseConfiguration.create();  
 try{  
  
 *connection* = ConnectionFactory.createConnection(*configuration*);  
  
 *admin* = *connection*.getAdmin();  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 System.*out*.println("Connect to HBase successfully!");  
 }  
  
 public static void close(){  
 try{  
 if(*admin* != null){  
 *admin*.close();  
 }  
 if(null != *connection*){  
 *connection*.close();  
 }  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
 public static void main(String[] args)throws IOException{  
 *init*();  
 System.*out*.println("Please input the RowID of the data you want to delete:");  
 Scanner in = new Scanner(System.*in*);  
 String input = in.nextLine();  
 *DeleteRow*("Supermarket",input);  
 System.*out*.println("A row is deleted!");  
 *close*();  
 }  
  
 public static void DeleteRow(String tableName, String rowKey) {  
 try {  
 Table t = *connection*.getTable(TableName.valueOf(tableName));  
 Delete delete = new Delete(Bytes.toBytes(rowKey));  
 t.delete(delete);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}**

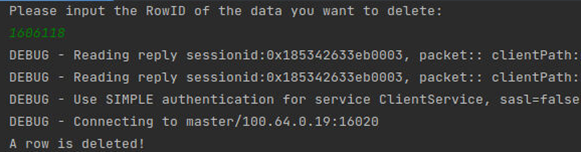
**执行结果**

执行前

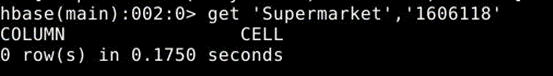


调用DeleteData()方法





执行后



## 修改数据

**总体思路**

1.建立HBase连接：首先，需要使用HBase的Java API建立HBase的连接。可以使用ConnectionFactory类的createConnection()方法建立连接。

2.获取HBase表：然后，需要获取HBase表的引用。可以使用Connection对象的getTable()方法获取表的引用。

3.创建Put对象：接着，需要创建一个Put对象，用于修改数据。可以使用Put类的构造函数创建Put对象，并指定行键。

4.向Put对象中添加或修改列：然后，可以使用Put对象的addColumn()方法向其中添加或修改列族、列名和值。

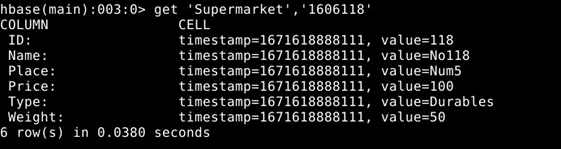
5.使用Table对象的put()方法将Put对象插入表中：最后，可以使用Table对象的put()方法将Put对象插入表中。

**程序代码**

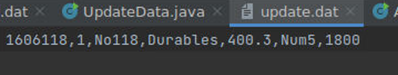
**import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.hbase.\*;  
import org.apache.hadoop.hbase.client.\*;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
  
public class UpdateData{  
 public static Configuration *configuration*;  
 public static Connection *connection*;  
 public static Admin *admin*;  
 public static void main(String[] args)throws IOException{  
 *init*();  
 *updateData*("Supermarket");  
 *close*();  
 }  
 //úËÞ¥  
 public static void init(){  
 //ËConfiguration ùa  
 *configuration* = HBaseConfiguration.create();  
 try{  
 //9n ConfigurationùaËConnection ùa  
 *connection* = ConnectionFactory.createConnection(*configuration*);  
 //·Ö Admin ùa  
 *admin* = *connection*.getAdmin();  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 System.*out*.println("Connect to HBase successfully!");  
 }  
 //síÞ¥  
 public static void close(){  
 try{  
 if(*admin* != null){  
 *admin*.close();  
 }  
 if(null != *connection*){  
 *connection*.close();  
 }  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
 //úh  
 public static void createTable(String myTableName,String[] colFamily) throws IOException {  
 TableName tableName = TableName.valueOf(myTableName);  
 if(*admin*.tableExists(tableName)){  
 System.*out*.println("Talbe is exists, going to delete the table...");  
 *admin*.disableTable(tableName);  
 *admin*.deleteTable(tableName);  
 }  
 HTableDescriptor hTableDescriptor = new HTableDescriptor(tableName);  
 for(String str:colFamily){  
 HColumnDescriptor hColumnDescriptor = new HColumnDescriptor(str);  
 hTableDescriptor.addFamily(hColumnDescriptor);  
 }  
 *admin*.createTable(hTableDescriptor);  
 }  
  
 public static void updateData(String tableName) throws IOException {  
 FileReader reader = new FileReader("/root/IdeaProjects/Test/data/update.dat");  
 BufferedReader br = new BufferedReader(reader);  
 String str = null;  
 Table table = *connection*.getTable(TableName.valueOf(tableName));  
 while((str = br.readLine()) != null) {  
 System.*out*.println("processing line:" + str);  
 String[] cols = str.split(",");  
 Put put = new Put(cols[0].getBytes());  
 put.addColumn("ID".getBytes(), "".getBytes(), cols[1].getBytes());  
 put.addColumn("Name".getBytes(), "".getBytes(), cols[2].getBytes());  
 put.addColumn("Type".getBytes(), "".getBytes(), cols[3].getBytes());  
 put.addColumn("Price".getBytes(), "".getBytes(), cols[4].getBytes());  
 put.addColumn("Place".getBytes(), "".getBytes(), cols[5].getBytes());  
 put.addColumn("Weight".getBytes(), "".getBytes(), cols[6].getBytes());  
 table.put(put);  
 }  
 table.close();  
 br.close();  
 reader.close();  
 }  
}**

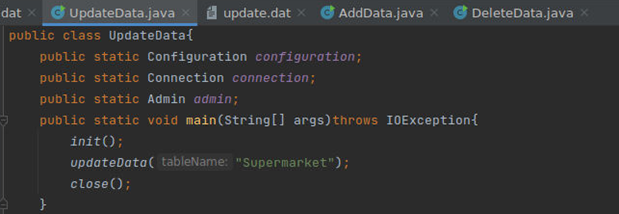
**执行结果**

执行前

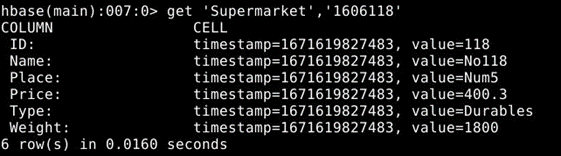


调用UpdateData()方法





执行后



## 查找数据

**总体思路**

1.建立HBase连接：首先，需要使用HBase的Java API建立HBase的连接。可以使用ConnectionFactory类的createConnection()方法建立连接。

2.获取HBase表：然后，需要获取HBase表的引用。可以使用Connection对象的getTable()方法获取表的引用。

3.创建Get对象：接着，需要创建一个Get对象，用于获取数据。可以使用Get类的构造函数创建Get对象，并指定行键。

4.获取Get对象中的列：然后，可以使用Get对象的get()方法获取指定列族、列名的列，或者使用getFamilyMap()方法获取整个列族的数据。

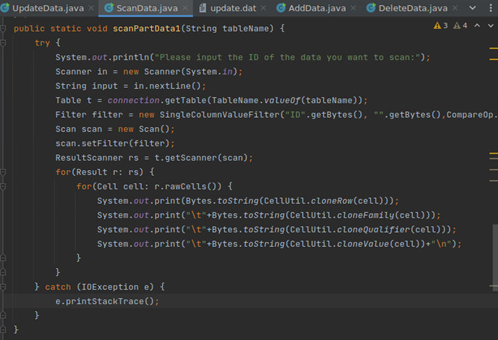
5.使用Table对象的get()方法将Get对象提交到表中：最后，可以使用Table对象的get()方法将Get对象提交到表中，并获取查询结果。

**程序代码**

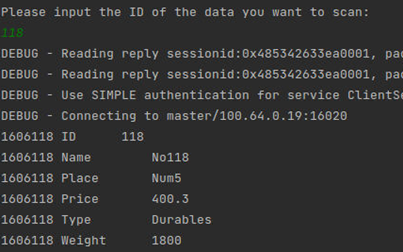
**import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.hbase.\*;  
import org.apache.hadoop.hbase.client.\*;  
import org.apache.hadoop.hbase.filter.CompareFilter.CompareOp;  
import org.apache.hadoop.hbase.filter.Filter;  
import org.apache.hadoop.hbase.filter.FilterList;  
import org.apache.hadoop.hbase.filter.SingleColumnValueFilter;  
import org.apache.hadoop.hbase.util.Bytes;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class ScanData{  
 public static Configuration *configuration*;  
 public static Connection *connection*;  
 public static Admin *admin*;  
  
 public static void init(){  
  
 *configuration* = HBaseConfiguration.create();  
 try{  
  
 *connection* = ConnectionFactory.createConnection(*configuration*);  
  
 *admin* = *connection*.getAdmin();  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 System.*out*.println("init success!");  
 }  
  
 public static void close(){  
 try{  
 if(*admin* != null){  
 *admin*.close();  
 }  
 if(null != *connection*){  
 *connection*.close();  
 }  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
 public static void main(String[] args)throws IOException{  
 *init*();  
 System.*out*.println("Scan part data1...");  
 *scanPartData1*("Supermarket");  
 *close*();  
 }  
  
 public static void scanPartData1(String tableName) {  
 try {  
 System.*out*.println("Please input the ID of the data you want to scan:");  
 Scanner in = new Scanner(System.*in*);  
 String input = in.nextLine();  
 Table t = *connection*.getTable(TableName.valueOf(tableName));  
 Filter filter = new SingleColumnValueFilter("ID".getBytes(), "".getBytes(),CompareOp.EQUAL, input.getBytes());  
 Scan scan = new Scan();  
 scan.setFilter(filter);  
 ResultScanner rs = t.getScanner(scan);  
 for(Result r: rs) {  
 for(Cell cell: r.rawCells()) {  
 System.*out*.print(Bytes.toString(CellUtil.cloneRow(cell)));  
 System.*out*.print("\t"+Bytes.toString(CellUtil.cloneFamily(cell)));  
 System.*out*.print("\t"+Bytes.toString(CellUtil.cloneQualifier(cell)));  
 System.*out*.print("\t"+Bytes.toString(CellUtil.cloneValue(cell))+"\n");  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }**

**执行结果**

调用ScanData()方法



输入想要查找的ID号后显示查询结果



# 五、调试与测试

















# 六、实习日志

12月12日

主要任务： 查看实验任务，确定实验思路

完成情况： 良好

存在问题： 对于实验方向和方法仍不确定

收获： 锻炼了思维能力和安排时间的能力

12月13日

主要任务： 继续确定实验思路，摸索方法

完成情况： 良好

存在问题： 对于HBASE JAVA API中一些操作的使用仍不熟练

收获： 学会了一些HBASE JAVA API的使用方法

12月14日

主要任务： 按照思路编写部分实验代码

完成情况： 良好

存在问题： 编写代码过程中遇到了困难

收获： 锻炼了编写代码的能力

12月15日

主要任务： 完善实验代码

完成情况： 良好

存在问题： 进一步完善代码

收获： 锻炼了编写代码的能力

12月16日

主要任务： 对代码进行部分测试

完成情况： 良好

存在问题： 测试时遇到了一些bug

收获：锻炼了debug能力和资料搜索能力

12月19日

主要任务： 完善代码，并进行调试

完成情况： 良好

存在问题： 遇到了一些困难

收获： 锻炼了调试代码的能力

12月20日

主要任务： 整体测试代码

完成情况： 良好

存在问题： 无明显问题

收获： 锻炼了配置环境和查询资料的能力

12月21日

主要任务： 制作ppt

完成情况： 良好

存在问题： 制作ppt时遇到了一些问题

收获： 锻炼了制作ppt的能力

12月22日

主要任务： 完善ppt

完成情况： 良好

存在问题： 锻炼了制作ppt的能力

收获：

12月23日

主要任务： 答辩

完成情况： 良好

存在问题： 情绪紧张，遗漏了一些环节

收获： 锻炼了答辩能力和口才

# 七、实习总结

通过本次实习，我收获了很多，也发现了自己的许多不足。在为期两周的实训生活中，在老师的指导下按着制定的实训计划，从一开始的迷茫、忙乱，到现在对大数据训练模型的建立有了一个较为清晰的认识。我的主要收获如下：

1.在信息时代，学习是不断地汲取新信息，获得事业进步的动⼒。作为⼀名青年学⼦更应该

把学习作为保持⼯作积极性的重要途径。实习，是开端也是结束。展现在⾃⼰⾯前的是⼀⽚任⾃⼰驰骋的沃⼟，也分明感受到了沉甸甸的责任。在今后的⼯作和⽣活中，我将继续学习，深⼊实践，不断提升⾃我，努⼒创造业绩，继续创造更多的价值。

2.只有将理论付诸于实践才能实现理论⾃⾝的价值，也只有将理论付诸于实践才能使理论得以检验。同样，⼀个⼈的价值也是通过实践活动来实现的，也只有通过实践才能锻炼⼈的品质，彰显⼈的意志。必须在实际的⼯作和⽣活中潜⼼体会，并⾃觉的进⾏这种⾓⾊的转换。

3.明白了大数据源于生活的方方面面。如果没有这次实训，我或许只会记得解决问题的方法是“自顶向下”这短短的四个字，但通过实训，用实践证明这短短四个字的精妙之处。通过实训也让我接触到了大数据真正的美，它是结合多方面学科知识，对于开发者的学习能力、规划能力、谨慎态度都有十分高的要求。也促使我更留心生活，因为大数据就源于生活需求的方方面面。

4.锻炼了课程设计成果总结能力。这次实训也锻炼了我编辑课程论文、编辑以及汇报PPT的能力，为今后还会无数次面临的课程设计打下基础。对于课程论文的排版要求、汇报PPT的版面以及内容要求有了初步的认识。

在实训中我学到了许多新的知识。是⼀个让我把书本上的理论知识运⽤于实践中的好机会，原来，学的时候感叹学的内容太难懂，现在想来，有些其实并不难，关键在于理解。在这次实训中还锻炼了我其他⽅⾯的能⼒，提⾼了我的综合素质。⾸先，它锻炼了我做项⽬的能⼒，提⾼了独⽴思考问题、⾃⼰动⼿操作的能⼒，在⼯作的过程中，复习了以前学习过的知识，并掌握了⼀些应⽤知识的技巧等。收获良多。

# 八、附录：核心代码清单

**import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.hbase.\*;  
import org.apache.hadoop.hbase.client.\*;  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
  
public class AddData{  
 public static Configuration *configuration*;  
 public static Connection *connection*;  
 public static Admin *admin*;  
 public static void main(String[] args)throws IOException{  
 *init*();  
 *putData*("Supermarket");  
 *close*();  
 }  
  
 public static void init(){  
  
 *configuration* = HBaseConfiguration.create();  
 try{  
  
 *connection* = ConnectionFactory.createConnection(*configuration*);  
  
 *admin* = *connection*.getAdmin();  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
  
 public static void close(){  
 try{  
 if(*admin* != null){  
 *admin*.close();  
 }  
 if(null != *connection*){  
 *connection*.close();  
 }  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
 public static void putData(String tableName) throws IOException {  
 FileReader reader = new FileReader("/root/IdeaProjects/Test/data/add.dat");  
 BufferedReader br = new BufferedReader(reader);  
 String str = null;  
 Table table = *connection*.getTable(TableName.valueOf(tableName));  
 while((str = br.readLine()) != null) {  
 System.*out*.println("processing line:" + str);  
 String[] cols = str.split(",");  
 Put put = new Put(cols[0].getBytes());  
 put.addColumn("ID".getBytes(), "".getBytes(), cols[1].getBytes());  
 put.addColumn("Name".getBytes(), "".getBytes(), cols[2].getBytes());  
 put.addColumn("Type".getBytes(), "".getBytes(), cols[3].getBytes());  
 put.addColumn("Price".getBytes(), "".getBytes(), cols[4].getBytes());  
 put.addColumn("Place".getBytes(), "".getBytes(), cols[5].getBytes());  
 put.addColumn("Weight".getBytes(), "".getBytes(), cols[6].getBytes());  
 /\*put.addColumn("Course".getBytes(), "Chinese".getBytes(), cols[6].getBytes());\*/  
 table.put(put);  
 }  
 table.close();  
 br.close();  
 reader.close();  
 }  
}**

**import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.hbase.\*;  
import org.apache.hadoop.hbase.client.\*;  
import org.apache.hadoop.hbase.util.Bytes;  
import java.io.IOException;  
import java.util.Scanner;  
  
public class DeleteData{  
 public static Configuration *configuration*;  
 public static Connection *connection*;  
 public static Admin *admin*;  
  
 public static void init(){  
  
 *configuration* = HBaseConfiguration.create();  
 try{  
  
 *connection* = ConnectionFactory.createConnection(*configuration*);  
  
 *admin* = *connection*.getAdmin();  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 System.*out*.println("Connect to HBase successfully!");  
 }  
  
 public static void close(){  
 try{  
 if(*admin* != null){  
 *admin*.close();  
 }  
 if(null != *connection*){  
 *connection*.close();  
 }  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
 public static void main(String[] args)throws IOException{  
 *init*();  
 System.*out*.println("Please input the RowID of the data you want to delete:");  
 Scanner in = new Scanner(System.*in*);  
 String input = in.nextLine();  
 *DeleteRow*("Supermarket",input);  
 System.*out*.println("A row is deleted!");  
 *close*();  
 }  
  
 public static void DeleteRow(String tableName, String rowKey) {  
 try {  
 Table t = *connection*.getTable(TableName.valueOf(tableName));  
 Delete delete = new Delete(Bytes.toBytes(rowKey));  
 t.delete(delete);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}**

**import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.hbase.\*;  
import org.apache.hadoop.hbase.client.\*;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
  
public class UpdateData{  
 public static Configuration *configuration*;  
 public static Connection *connection*;  
 public static Admin *admin*;  
 public static void main(String[] args)throws IOException{  
 *init*();  
 *updateData*("Supermarket");  
 *close*();  
 }  
 //úËÞ¥  
 public static void init(){  
 //ËConfiguration ùa  
 *configuration* = HBaseConfiguration.create();  
 try{  
 //9n ConfigurationùaËConnection ùa  
 *connection* = ConnectionFactory.createConnection(*configuration*);  
 //·Ö Admin ùa  
 *admin* = *connection*.getAdmin();  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 System.*out*.println("Connect to HBase successfully!");  
 }  
 //síÞ¥  
 public static void close(){  
 try{  
 if(*admin* != null){  
 *admin*.close();  
 }  
 if(null != *connection*){  
 *connection*.close();  
 }  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
 //úh  
 public static void createTable(String myTableName,String[] colFamily) throws IOException {  
 TableName tableName = TableName.valueOf(myTableName);  
 if(*admin*.tableExists(tableName)){  
 System.*out*.println("Talbe is exists, going to delete the table...");  
 *admin*.disableTable(tableName);  
 *admin*.deleteTable(tableName);  
 }  
 HTableDescriptor hTableDescriptor = new HTableDescriptor(tableName);  
 for(String str:colFamily){  
 HColumnDescriptor hColumnDescriptor = new HColumnDescriptor(str);  
 hTableDescriptor.addFamily(hColumnDescriptor);  
 }  
 *admin*.createTable(hTableDescriptor);  
 }  
  
 public static void updateData(String tableName) throws IOException {  
 FileReader reader = new FileReader("/root/IdeaProjects/Test/data/update.dat");  
 BufferedReader br = new BufferedReader(reader);  
 String str = null;  
 Table table = *connection*.getTable(TableName.valueOf(tableName));  
 while((str = br.readLine()) != null) {  
 System.*out*.println("processing line:" + str);  
 String[] cols = str.split(",");  
 Put put = new Put(cols[0].getBytes());  
 put.addColumn("ID".getBytes(), "".getBytes(), cols[1].getBytes());  
 put.addColumn("Name".getBytes(), "".getBytes(), cols[2].getBytes());  
 put.addColumn("Type".getBytes(), "".getBytes(), cols[3].getBytes());  
 put.addColumn("Price".getBytes(), "".getBytes(), cols[4].getBytes());  
 put.addColumn("Place".getBytes(), "".getBytes(), cols[5].getBytes());  
 put.addColumn("Weight".getBytes(), "".getBytes(), cols[6].getBytes());  
 table.put(put);  
 }  
 table.close();  
 br.close();  
 reader.close();  
 }  
}**

**import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.hbase.\*;  
import org.apache.hadoop.hbase.client.\*;  
import org.apache.hadoop.hbase.filter.CompareFilter.CompareOp;  
import org.apache.hadoop.hbase.filter.Filter;  
import org.apache.hadoop.hbase.filter.FilterList;  
import org.apache.hadoop.hbase.filter.SingleColumnValueFilter;  
import org.apache.hadoop.hbase.util.Bytes;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class ScanData{  
 public static Configuration *configuration*;  
 public static Connection *connection*;  
 public static Admin *admin*;  
  
 public static void init(){  
  
 *configuration* = HBaseConfiguration.create();  
 try{  
  
 *connection* = ConnectionFactory.createConnection(*configuration*);  
  
 *admin* = *connection*.getAdmin();  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 System.*out*.println("init success!");  
 }  
  
 public static void close(){  
 try{  
 if(*admin* != null){  
 *admin*.close();  
 }  
 if(null != *connection*){  
 *connection*.close();  
 }  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
 public static void main(String[] args)throws IOException{  
 *init*();  
 System.*out*.println("Scan part data1...");  
 *scanPartData1*("Supermarket");  
 *close*();  
 }  
  
 public static void scanPartData1(String tableName) {  
 try {  
 System.*out*.println("Please input the ID of the data you want to scan:");  
 Scanner in = new Scanner(System.*in*);  
 String input = in.nextLine();  
 Table t = *connection*.getTable(TableName.valueOf(tableName));  
 Filter filter = new SingleColumnValueFilter("ID".getBytes(), "".getBytes(),CompareOp.EQUAL, input.getBytes());  
 Scan scan = new Scan();  
 scan.setFilter(filter);  
 ResultScanner rs = t.getScanner(scan);  
 for(Result r: rs) {  
 for(Cell cell: r.rawCells()) {  
 System.*out*.print(Bytes.toString(CellUtil.cloneRow(cell)));  
 System.*out*.print("\t"+Bytes.toString(CellUtil.cloneFamily(cell)));  
 System.*out*.print("\t"+Bytes.toString(CellUtil.cloneQualifier(cell)));  
 System.*out*.print("\t"+Bytes.toString(CellUtil.cloneValue(cell))+"\n");  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }**