Catalog Manager 模块

要求

Catalog Manager 负责管理数据库的所有模式信息，包括：

1. 数据库中所有表的定义信息，包括表的名称、表中字段（列）数、主键、定义在该表上的索引。
2. 表中每个字段的定义信息，包括字段类型、是否唯一等。
3. 数据库中所有索引的定义，包括所属表、索引建立在那个字段上等。

类

index

功能：记录索引的信息

储存的信息

```
  public String indexName;//Ë÷ÒýÃû£¬Î¨Ò»±ê¾ÇË÷Òý
  public String tableName;//±íÃû
  public String attriName;//ÊôÐÔÃû
  public int column;     //on which column the index is created
  public int columnLength;//
  public int rootNum;
  public int blockNum=0;  //number of block the datas of the index occupied in
the file index_name.table
```

类中的方法

类中的方法定义了两个构造函数和一个获得信息的方法。

```
  public index(String indexName,String tableName,String attriName, int
blockNum, int rootNum){
    this.indexName=indexName;
    this.tableName=tableName;
    this.attriName=attriName;
    this.blockNum = blockNum;
    this.rootNum = rootNum;
  }

  public index(String indexName,String tableName,String attriName){
    this.indexName=indexName;
    this.tableName=tableName;
    this.attriName=attriName;
  }
  public void PickInfo(){
    column = CatalogManager.getAttriOffest(tableName, attriName);
    columnLength = CatalogManager.getLength(tableName, attriName);
  }
```

table

储存的信息

```
  String tableName;
  String primaryKey;
  Vector<attribute>attributes;
  Vector<index> indexes;
  int indexNum;
  int attriNum;
  int tupleNum;
  int tupleLength;
```

类中的方法

类中的方法定义两个构造函数

```
  public table(String tableName,Vector<attribute> attributes,String
primaryKey){
    this.tableName=tableName;
    this.primaryKey=primaryKey;
    this.indexes=new Vector<index>();
    this.indexNum=0;
    this.attributes=attributes;
    this.attriNum=attributes.size();
    this.tupleNum=0;
    //¼ÆËã×ÜtupleLength
    for(int i=0;i<attributes.size();i++){
      if(attributes.get(i).attriName.equals(primaryKey))
        attributes.get(i).isUnique=true;
      this.tupleLength+=attributes.get(i).length;
    }
  }
  public table(String tableName, Vector<attribute> attributes, Vector<index>
indexes, String primaryKey,int tupleNum) {//initial table
    this.tableName=tableName;
    this.primaryKey=primaryKey;
    this.attributes=attributes;
    this.indexes=indexes;
    this.attriNum=attributes.size();
    this.indexNum=indexes.size();
    this.tupleNum=tupleNum;
    for(int i=0;i<attributes.size();i++){
      this.tupleLength+=attributes.get(i).length;
    }
  }
```

attribute

储存的信息

```
  String attriName;
  String type;
  int length;
  boolean isUnique;
```

类中方法

类中的方法定义了一个构造函数

```
  public attribute(String attriName,String type,int length,boolean isU){
    this.attriName=attriName;
    this.type=type;
    this.length=length;
    this.isUnique=isU;
  }
```

Catalog Manager中定义的静态变量

```
  private static Hashtable<String,table> tables=new Hashtable<String, table>()
;
  private static Hashtable<String,index> indexes=new Hashtable<String, index>
();
  private static String tableFilename="table catalog";
  private static String indexFilename="index catalog";
```

两个哈希表用于储存表和索引，实行实例和名字的一一对应。

Catalog Manager中定义的函数

Catalog Manager的初始化

要初始化Catalog Manager这个模块，要分别从文件中读取Index和Table的信息，因此分为两个函数进行读取，在初始化时，一并调用。

```
  public static void InitialCatalog() throws IOException {
    InitialTableCatalog();
    InitialIndexCatalog();
```

```java
  }
  private static void InitialIndexCatalog() throws IOException  {
    // TODO Auto-generated method stub
    File file=new File(indexFilename);
    if(!file.exists()) return;
    FileInputStream fis = new FileInputStream(file);
    DataInputStream dis = new DataInputStream(fis);
    String tmpIndexName,tmpTableName,tmpAttriName;
    int tmpIndexBlockNum,tmpRootNum;
    while(dis.available()>0) {
      tmpIndexName=dis.readUTF();
      tmpTableName=dis.readUTF();
      tmpAttriName=dis.readUTF();
      tmpIndexBlockNum=dis.readInt();
      tmpRootNum = dis.readInt();
      indexes.put(tmpIndexName, new
index(tmpIndexName,tmpTableName,tmpAttriName,tmpIndexBlockNum,tmpRootNum));
    }
    dis.close();

  }
  private static void InitialTableCatalog() throws IOException {
    // TODO Auto-generated method stub
    File file=new File(tableFilename);
    if(!file.exists()) return;
    FileInputStream fis = new FileInputStream(file);
    DataInputStream dis = new DataInputStream(fis);
    String tmpTableName,tmpPriKey;
    int tmpIndexNum,tmpAttriNum,tmpTupleNum;

    while(dis.available()>0) {
      Vector<attribute> tmpAttributes=new Vector<attribute>();
      Vector<index> tmpIndexes=new Vector<index> ();
      tmpTableName=dis.readUTF();
      tmpPriKey=dis.readUTF();
      tmpTupleNum=dis.readInt();//dos.writeInt(tmpTable.tupleNum);
      tmpIndexNum=dis.readInt();
      for(int i=0;i<tmpIndexNum;i++){
        String tmpIndexName,tmpAttriName;
        tmpIndexName=dis.readUTF();
        tmpAttriName=dis.readUTF();
        tmpIndexes.addElement(new
index(tmpIndexName,tmpTableName,tmpAttriName));
      }
      tmpAttriNum=dis.readInt();
      for(int i=0;i<tmpAttriNum;i++){
        String tmpAttriName,tmpType;
        int tmpLength;boolean tmpIsU;
        tmpAttriName=dis.readUTF();
```

```
            tmpType=dis.readUTF();
            tmpLength=dis.readInt();
            tmpIsU=dis.readBoolean();
            tmpAttributes.addElement(new
attribute(tmpAttriName,tmpType,tmpLength,tmpIsU));
          }
          tables.put(tmpTableName, new
table(tmpTableName,tmpAttributes,tmpIndexes,tmpPriKey,tmpTupleNum));

        }
        dis.close();
      }
```

储存Catalog Manage信息

同初始化，储存信息要同时储存Index和Table信息到文件中。

```
  public static void storeCatalog() throws IOException{
    storeTableCatalog();
    storeIndexCatalog();
  }
  private static void storeIndexCatalog() throws IOException {
    // TODO Auto-generated method stub

    File file=new File(indexFilename);
    if(file.exists())file.delete();
    FileOutputStream fos = new FileOutputStream(file);
    DataOutputStream dos = new DataOutputStream(fos);
    index tmpIndex;
    Enumeration<index> en = indexes.elements();
    while(en.hasMoreElements()) {
      tmpIndex=en.nextElement();
      dos.writeUTF(tmpIndex.indexName);
      dos.writeUTF(tmpIndex.tableName);
      dos.writeUTF(tmpIndex.attriName);
      dos.writeInt(tmpIndex.blockNum);
      dos.writeInt(tmpIndex.rootNum);
    }
    dos.close();
  }
  private static void storeTableCatalog() throws IOException {
    // TODO Auto-generated method stub
    File file=new File(tableFilename);
    //if(file.exists())file.d;
    FileOutputStream fos = new FileOutputStream(file);
    DataOutputStream dos = new DataOutputStream(fos);
    table tmpTable;
    Enumeration<table> en = tables.elements();
        while(en.hasMoreElements()) {
          tmpTable=en.nextElement();
          dos.writeUTF(tmpTable.tableName);
          dos.writeUTF(tmpTable.primaryKey);
          dos.writeInt(tmpTable.tupleNum);
          dos.writeInt(tmpTable.indexNum);
          for(int i=0;i<tmpTable.indexNum;i++){
            index tmpIndex=tmpTable.indexes.get(i);
            dos.writeUTF(tmpIndex.indexName);
            dos.writeUTF(tmpIndex.attriName);
            }
          dos.writeInt(tmpTable.attriNum);
          for(int i=0;i<tmpTable.attriNum;i++){
            attribute tmpAttri=tmpTable.attributes.get(i);
            dos.writeUTF(tmpAttri.attriName);
            dos.writeUTF(tmpAttri.type);
            dos.writeInt(tmpAttri.length);
            dos.writeBoolean(tmpAttri.isUnique);
          }
        }
    dos.close();
  }
```

展示Catalog Manager信息

和前面一样，要一并输出table和index的信息，因此用了两个函数，一并调用

```
    zpublic static void showCatalog(){
      showTableCatalog();
      System.out.println();
      showIndexCatalog();
    }
    public static void showIndexCatalog() {
      // TODO Auto-generated method stub
      index tmpIndex;
      Enumeration<index> en = indexes.elements();
      int cnt=1;
      System.out.println("There are "+indexes.size()+" indexes in the database:
  ");
          System.out.println("\tIndex name\tTable name\tAttribute name:");
      while(en.hasMoreElements()) {
        tmpIndex=en.nextElement();

  System.out.println(cnt+++"\t"+tmpIndex.indexName+"\t\t"+tmpIndex.tableName+"\t
  \t"+tmpIndex.attriName);
      }
    }
    public static void showTableCatalog() {
      // TODO Auto-generated method stub
      table tmpTable;
    index tmpIndex;
    attribute tmpAttribute;
    Enumeration<table> en = tables.elements();
    int cnt=1;
    System.out.println("There are "+tables.size()+" tables in the database:
");
        while(en.hasMoreElements()) {
            tmpTable=en.nextElement();
            System.out.println("\nTable "+cnt++);
            System.out.println("Table name: "+tmpTable.tableName);
            System.out.println("Number of Columns: "+tmpTable.attriNum);
            System.out.println("Primary key: "+tmpTable.primaryKey);
            System.out.println("Number of tuples: "+tmpTable.tupleNum);
            System.out.println("Index keys: "+tmpTable.indexNum);
            System.out.println("\tIndex name\tTable name\tAttribute name:");
            for(int i=0;i<tmpTable.indexNum;i++){
              tmpIndex=tmpTable.indexes.get(i);

System.out.println("\t"+tmpIndex.indexName+"\t"+tmpIndex.tableName+"\t\t"+tmpI
ndex.attriName);
            }
            System.out.println("Attributes: "+tmpTable.attriNum);
            System.out.println("\tAttribute name\tType\tlength\tisUnique");
            for(int i=0;i<tmpTable.attriNum;i++){
              tmpAttribute=tmpTable.attributes.get(i);
 System.out.println("\t"+tmpAttribute.attriName+"\t\t"+tmpAttribute.type+"\t"+
tmpAttribute.length+"\t"+tmpAttribute.isUnique);
            }
        }
  }
```

获得catalog manager信息的接口

利用哈希表的一些函数，来读取信息。

```
    public static boolean isPrimaryKey(String tableName,String attriName){
    if(isTableExist(tableName)){
      table tmpTable=getTable(tableName);
      if(tmpTable.primaryKey.equals(attriName))return true;
      else return false;
    }
    else{
      System.out.println("The table "+tableName+" doesn't exist");
      return false;
    }
  }
  public static boolean inUniqueKey(String tableName,String attriName){
    if(isTableExist(tableName)){
      table tmpTable=getTable(tableName);
      int i;
      for(i=0;i<tmpTable.attributes.size();i++){
        attribute tmpAttribute=tmpTable.attributes.get(i);
        if(tmpAttribute.attriName.equals(attriName)){
          return tmpAttribute.isUnique;
        }
      }
      if(i>=tmpTable.attributes.size()){
        System.out.println("The attribute "+attriName+" doesn't exist");
        return false;
      }
    }
    System.out.println("The table "+tableName+" doesn't exist");
    return false;

  }
  public static boolean isIndexKey(String tableName,String attriName){
    if(isTableExist(tableName)){
      table tmpTable=getTable(tableName);
      if(isAttributeExist(tableName,attriName)){
        for(int i=0;i<tmpTable.indexes.size();i++){
          if(tmpTable.indexes.get(i).attriName.equals(attriName))
            return true;
        }
        //System.out.println(" The attribute "+attriName+" is not an index
key");Áô¸øinterpreter
      }
      else{
        System.out.println("The attribute "+attriName+" doesn't exist");
      }
    }
    else
      System.out.println("The table "+tableName+" doesn't exist");
    return false;
  }
  public static boolean isTableExist(String tableName){
    return tables.containsKey(tableName);
      }
  public static boolean isIndexExist(String indexName){
    return indexes.containsKey(indexName);
  }
  public static boolean isAttributeExist(String tableName,String attriName){
    table tmpTable=getTable(tableName);
    for(int i=0;i<tmpTable.attributes.size();i++){
      if(tmpTable.attributes.get(i).attriName.equals(attriName))
        return true;
    }
    return false;
  }
  public static String getIndexName(String tableName,String attriName){
    if(isTableExist(tableName)){
      table tmpTable=getTable(tableName);
      if(isAttributeExist(tableName,attriName)){
        for(int i=0;i<tmpTable.indexes.size();i++){
          if(tmpTable.indexes.get(i).attriName.equals(attriName))
            return tmpTable.indexes.get(i).indexName;
        }
      }
      else{
        System.out.println("The attribute "+attriName+" doesn't exist");
      }
    }
    else
      System.out.println("The table "+tableName+" doesn't exist");
    return null;
  }
  public static String getAttriName(String tableName,int i){//ÓÃÓÚinsert
Õë¶ÔµÚi¸öÊôÐÔ
    return tables.get(tableName).attributes.get(i).attriName;
  }
  public static int getAttriOffest(String tableName,String attriName){
    table tmpTable=tables.get(tableName);
    attribute tmpAttri;
    for(int i=0;i<tmpTable.attributes.size();i++){
      tmpAttri=tmpTable.attributes.get(i);
      if(tmpAttri.attriName.equals(attriName))
        return i;
    }
    System.out.println("Error: The attribute "+attriName+" doesn't exist");
    return -1;
  }
  public static String getType(String tableName,String attriName){//ÓÃÓÚwhere
    table tmpTable=tables.get(tableName);
    attribute tmpAttri;
    for(int i=0;i<tmpTable.attributes.size();i++){
      tmpAttri=tmpTable.attributes.get(i);
      if(tmpAttri.attriName.equals(attriName))
        return tmpAttri.type;
    }
    System.out.println("Error: The attribute "+attriName+" doesn't exist");
    return null;
  }
  public static int getLength(String tableName,String attriName){//ÓÃÓÚwhere
    table tmpTable=tables.get(tableName);
    attribute tmpAttri;
    for(int i=0;i<tmpTable.attributes.size();i++){
      tmpAttri=tmpTable.attributes.get(i);
      if(tmpAttri.attriName.equals(attriName))
```

```
        return tmpAttri.length;
      }
      System.out.println("Error: The attribute "+attriName+" doesn't exist");
      return -1;
    }
  public static String getType(String tableName,int i){//ÓÃÓÚinsert
Õë¶ÔµÚi¸öÊôÐÔ
      table tmpTable=tables.get(tableName);

//System.out.println(tmpTable.attributes.get(i).type+tmpTable.attributes.get(i
).attriName);
      return tmpTable.attributes.get(i).type;
    }
  public static int getLength(String tableName,int i){//ÓÃÓÚinsert
Õë¶ÔµÚi¸öÊôÐÔ
      table tmpTable=tables.get(tableName);
      return tmpTable.attributes.get(i).length;
    }
  public static boolean isAttributeExist(Vector<attribute> attributes, String
attriName) {
      for(int i=0;i<attributes.size();i++){
        if(attributes.get(i).attriName.equals(attriName))
          return true;
      }
      return false;
    }
```

创建，更改matalog manager储存的信息

利用哈希表的方法，修改matalog manager储存的信息。

```
  public static void addTupleNum(String tableName){
      tables.get(tableName).tupleNum++;
    }
  public static void deleteTupleNum(String tableName,int num){
      tables.get(tableName).tupleNum-=num;
    }
  public static boolean updateIndexTable(String indexName,index indexinfo){
      indexes.replace(indexName, indexinfo);
      return true;
    }
public static boolean createTable(table newTable){
      try{
        tables.put(newTable.tableName, newTable);
        //indexes.put(newTable.indexes.firstElement().indexName,
newTable.indexes.firstElement());
        return true;
      }
      catch(NullPointerException e){
        e.printStackTrace();
        return false;
      }

    }
    public static boolean dropTable(String tableName){
      try{
        table tmpTable=tables.get(tableName);
        for(int i=0;i<tmpTable.indexes.size();i++){
          indexes.remove(tmpTable.indexes.get(i).indexName);
        }
        tables.remove(tableName);
        return true;
      }
      catch(NullPointerException e){
        System.out.println("Error: drop null table. "+e.getMessage());
        return false;
      }
    }
    public static boolean createIndex(index newIndex){
      try{
      table tmpTable=getTable(newIndex.tableName);
      tmpTable.indexes.addElement(newIndex);
      tmpTable.indexNum=tmpTable.indexes.size();
      indexes.put(newIndex.indexName, newIndex);
      return true;
      }
      catch(Exception e){
        e.printStackTrace();
        return false;
      }
    }
    public static boolean dropIndex(String indexName){
      try{
        index tmpIndex=getIndex(indexName);
        table tmpTable=getTable(tmpIndex.tableName);
        tmpTable.indexes.remove(tmpIndex) ;
        tmpTable.indexNum=tmpTable.indexes.size();
        indexes.remove(indexName);
        return true;
      }
      catch(Exception e){
        e.printStackTrace();
        return false;
      }

    }
```

bonus natural join实现

基于catalog模块的natural join 功能设计，分为两种情况：一、无参数进行natural join，检查是否有重复名字和类型的列，并根据这个列进行join。二、如果为有列输入，则检查对应的关系属性是否一致，若不一致则返回错误，若一致则生成新的表。

```
 public static boolean NaturalJoin(table table_r, table table_l, String table_name, String primaryKey) {
    try {
      Vector<attribute> current_vector = new Vector<>(table_r.attributes);
      for (attribute s : table_l.attributes) {
        boolean flag = true;
        for(attribute ts : table_r.attributes){
          if(ts.attriName.equals(s.attriName) && ts.type.equals(s.type)){
            flag = false;
            break;
          }
        }
        if(flag){
          current_vector.add(s);
        }
      }
      table new_table = new table(table_name, current_vector, primaryKey);
      tables.put(table_name, new_table);
      //indexes.put(newTable.indexes.firstElement().indexName, newTable.indexes.firstElement());
      return true;
    } catch (NullPointerException e) {
      e.printStackTrace();
      return false;
    }
  }

  public static boolean NaturalJoin(table table_r, table table_l, String table_name, Vector<attribute> columns1, Vector<attribute> columns2, String primaryKey) {
    try {
      if(columns1.size() != columns2.size())
        return false;
      for(int i = 0 ; i < columns1.size() ; i++){
        if(!columns1.get(i).type.equals(columns2.get(i).type))
          return false;
      }
      Vector<attribute> current_vector = new Vector<>(table_l.attributes);
      for(attribute s : table_r.attributes){
        boolean flag = true;
        for(attribute ts : columns2){
```

```
                    if(ts.attriName.equals(s.attriName) &&
ts.type.equals(s.type)){
                        flag = false;
                        break;
                    }
                }
                if(flag){
                    current_vector.add(s);
                }
            }
            table new_table = new table(table_name, current_vector,
primaryKey);
            tables.put(table_name, new_table);
            //indexes.put(newTable.indexes.firstElement().indexName,
newTable.indexes.firstElement());
            return true;
        } catch (NullPointerException e) {
            e.printStackTrace();
            return false;
        }
    }
```