Design of Recursive Algorithms for Manipulating Relational Database, Using Abstract Data Type, Relation

O.E. Oguike and C.I. Ugwu

Department of Computer Science
University of Nigeria
Nsukka, Nigeria

osondu.oguike@unn.edu.ng, celestiyke@yahoo.com

ABSTRACT

Relational database is a collection of two or more database tables with common attributes, in form of primary and foreign keys, which are used to establish relationships between the database tables. Most application software were developed using relational database management system, which allowed the application developer to store and retrieve data from the database tables, using appropriate commands that the database programming language had provided. Data abstraction is an important software engineering concept that allows us to separate the specification of an abstract data type from its implementation. This paper designs and implements the algorithms that can be used to manipulate relational database using the abstract data type, relation and its underlying abstract data types, in an object oriented programming style.

Keywords: Relational database, table/relation, set, list, dynamic data structure, recursive algorithm, data abstraction, object oriented programming

1. INTRODUCTION

Relational database is very important for storing and manipulating structured data in any application software that has been developed using a relational database management system, like Foxpro, Oracle, Paradox, MySql etc. All relational database management systems provide data manipulation languages that application programmers can use to store and manipulate data stored in the database. Most of them provide query languages that application programmers can use to query the database in order to produce useful reports. It has became a matter of academic concern to consider some of the underlying abstract data types that can be used to store and manipulate data stored in a relational database, which include abstract data types, table/relation, set, list and tree.

Figure 1: Hierarchy of the Abstract Data Types
Furthermore, all the algorithms for each of the abstract data types will be designed and they will be used to design novel recursive algorithms that will be used to query and manipulate a relational database. For each database table, the abstract data type, table/relation will be used to represent it. Data stored in this abstract data type will be stored in a file to form the database file.

The underlying abstract data type that will be used to represent a table/relation is the abstract data type, set, while the underlying abstract data type that will be used to represent the abstract type set is the abstract data type, list. The hierarchy of the abstract data types used to represent the relational database can be shown in figure 1. An Academic Library Management System will be used as a specific application to illustrate the use of these abstract data types and their algorithms for manipulating a relational database.

2. RELATIONAL DATABASE DESIGN OF THE SYSTEM

In order to identify the various database tables of an Academic Library Management System, the Entity Relationship Model in figure 2 will be used.

![Figure 2. Entity Relationship Diagram of the System](image)

The various database tables from the Entity Relationship Model can be shown in the table 1.

<table>
<thead>
<tr>
<th>TABLE 1: DATABASE TABLES OF ACADEMIC LIBRARY SYSTEM.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Books Table</strong></td>
</tr>
<tr>
<td>Field Name</td>
</tr>
<tr>
<td>ISBN</td>
</tr>
<tr>
<td>Title</td>
</tr>
<tr>
<td>Author</td>
</tr>
<tr>
<td>Edition</td>
</tr>
<tr>
<td>Publisher</td>
</tr>
<tr>
<td>Data Published</td>
</tr>
<tr>
<td>No Copies</td>
</tr>
<tr>
<td>Loan Loan</td>
</tr>
<tr>
<td>username</td>
</tr>
<tr>
<td>Password</td>
</tr>
</tbody>
</table>
For each of the database tables, the following ADTs, list or tree, set, table and file will be used to represent it as shown in figure 1.

### 2. SURVEY OF RELATED LITERATURE

The concept of ERM as a tool for designing relational database tables for a specific application was discussed extensively in literature, [1],[2],[3]. This paper uses the concept to design the database tables of an Academic Library Management System. Though abstract data types were used to implement relational database tables in [4], but recursive algorithms were not presented, except the implementations in Modula 2. However, this paper uses an object-oriented approach to implement the database tables. In [5], the author proposed some high-level language construct for the abstract data type, relation. One of the constructs was based on repetition, which was controlled by relation. However, this paper uses recursion to implement repetition. It was proposed in [6] that databases had a type system interface and it described a representation of a type system in terms of relations.

This paper uses some object-oriented concepts like, class hierarchy, inheritance, interface, collections and methods to implement the various abstract data types. These concepts were used in [7] to model and implement information retrieval in an object-oriented database environment. Distinguishing between relational databases and object-oriented databases, [8] remarked that object-oriented databases represented the data or real world entity to be organized as objects. According to the author, a record in a relational model would correspond roughly to an object in object-oriented models. Furthermore, [8] noted that while a record in a relational model had only a state, an object in object-oriented model had state and methods, and could represent structure as well as behaviour. Therefore, each object according to [8] could encapsulate many properties (attributes) and behaviours (methods).

However, [9] maintained that a relation or a table in a relational database could be considered to be analogous to a class in an object oriented database. According to them a tuple was similar to an instance of a class, but it was different in that it had attributes but no behaviours.

This paper uses these similarities and differences to implement all the algorithms for all the ADTs. According to [9], object oriented databases included abstract data types that allowed the user to define the characteristics of the data to be stored when developing an application. In recent years, object oriented relational database had been developed, which combined the relational database framework with the capability to store complex data types [9], [10].

### 3 DESIGN OF ALGORITHM FOR THE OPERATIONS OF ABSTRACT DATA TYPE, LIST

There are four database tables, which are users, books, copies and loans. Each of them will be represented as a tree or a list at the lowest level of data abstraction. Suppose the table, users is represented as a user defined abstract data type, userlist with the following attributes defined in a java class userlist.

- mystring userid;
- mystring surname;
- mystring firstname;
- mystring lastname;
- mystring category;
- mystring adminno;
- mystring dept;
- mystring address;
- mystring username;
- mystring password;
- userlist link;

The following are the algorithms that will be used to store and access the data in the abstract data type, userlist.

```java
userlist emptyuser()
```

1. Determine emptyuser
   1.1 emptyuser = null
2. Display emptyuser
The algorithm, emptyuser returns the null list and it takes nothing as parameter.

Boolean isemptyuser(userlist l)
1. Read l
2. Determine isemptyuser
   2.1 IF l = null THEN
       2.1.1 isemptyuser = true
   ELSE
       2.1.2 isemptyuser = false
3. Display isemptyuser

The algorithm, isemptyuser takes a list as parameter and it returns true if the list is empty, otherwise it returns false.

mystring headuserid(userlist l)
1. Read l
2. Determine headuserid
   2.1 IF isemptyuser(l) THEN
       2.1.1 headuserid = (“No head for empty list”)
   ELSE
       2.1.2 headuserid = l.userid
3. Display headuserid

The algorithm, headuserid takes a list, userlist as parameter. If the list is not empty, it returns userid attribute of the list, otherwise, it returns an appropriate error message.

mystring headsurname(userlist l)
1. Read l
2. Determine headsurname
   2.1 IF isemptyuser(l) THEN
       2.1.1 headsurname = (“No head for empty list”)
   ELSE
       2.1.2 headsurname = l.surname
3. Display headsurname

The algorithm, headsurname takes the list, userlist as parameter. If the list is not empty, it returns surname attribute of the list, otherwise, it returns an appropriate error message.

mystring headfirstname(userlist l)
1. Read l
2. Determine headfirstname
   2.1 IF isemptyuser(l) THEN
       2.1.1 headfirstname = (“No head for empty list”)
   ELSE
       2.1.2 headfirstname = l.firstname
3. Display headfirstname

The algorithm, headfirstname takes the list, userlist as parameter. If it is not empty, it returns firstname attribute of the list, otherwise, it returns an appropriate error message.

mystring headcategory(userlist l)
1. Read l
2. Determine headcategory
   2.1 IF isemptyuser(l) THEN
       2.1.1 headcategory = (“No head for empty list”)
   ELSE
       2.1.2 headcategory = l.category
3. Display headcategory

The algorithm, headcategory takes the list, userlist as parameter. If it is not empty it returns lastname, which is the user defined list of characters called mystring, otherwise it returns an appropriate error message.

mystring headadminno(userlist l)
1. Read l
2. Determine headadminno
   2.1 IF isemptyuser(l) THEN
       2.1.1 headadminno = (“No head for empty list”)
   ELSE
       2.1.2 headadminno = l.adminno
3. Display headadminno

The algorithm, headadminno takes the list, userlist as parameter. If it is not empty it returns adminno attribute of the list, otherwise, it returns an appropriate error message.

mystring headdept(userlist l)
1. Read l
2. Determine headdept
   2.1 IF isemptyuser(l) THEN
       2.1.1 headdept = (“No head for empty list”)
   ELSE
       2.1.2 headdept = l.dept
3. Display headdept

The algorithm, headdept takes the list, userlist as parameter. If it is not empty, it returns adminno attribute of the list, otherwise, it returns an appropriate error message.
The algorithm, headdept takes the list, userlist as parameter. If it is not empty, it returns dept attribute of the list, otherwise it returns an appropriate error message.

```plaintext
mystring headaddress(userlist l)
1   Read l
2   Determine headaddress
2.1  IF isemptyuser(l) THEN
2.1.1  headaddress = ("No head for empty list")
ELSE
2.1.2  headaddress = l.address
3   Display headaddress
```

The algorithm, headusername takes the list, userlist as parameter. If it is not empty, it returns address attribute of list, otherwise it returns an appropriate error message.

```plaintext
mystring headusername(userlist l)
1   Read l
2   Determine headusername
2.1  IF isemptyuser(l) THEN
2.1.1  headusername = ("No head for empty list")
ELSE
2.1.2  headusername = l.username
3   Display headusername
```

The algorithm, headpassword takes the list, userlist as parameter. If it is not empty, it returns password attribute of the list, otherwise, it returns an appropriate error message.

```plaintext
mystring headpassword(userlist l)
1   Read l
2   Determine headpassword
2.1  IF isemptyuser(l) THEN
2.1.1  headpassword = ("No head for empty list")
ELSE
2.1.2  headpassword = l.password
3   Display headpassword
```

The algorithm, tailuser takes the list, userlist as parameter. If the list is not empty, it returns the link attribute of the list, otherwise, it returns an empty list.

```plaintext
mystring tailuser(userlist l)
1   Read l
2   Determine tailuser
2.1  IF isemptyuser(l) THEN
2.1.1  tailuser = l
ELSE
2.1.2  tailuser = l.link
3   Display tailuser
```

In a similar manner, for each of the other data entities, books, copies and loans, the algorithms for the operations of the lowest underlying ADT, list or tree will be designed, with the appropriate attributes defined in table 1.

### 4. DESIGN OF ALGORITHMS FOR THE OPERATIONS OF ABSTRACT DATA TYPE, SET

For each of the data entities, users, books, loans and copies, the algorithms for the operations of the underlying abstract data type, list or tree will be used to design the algorithms for the operations of set. For the data entity, users, the abstract data type, userset will be considered as userlist, therefore, the algorithms for the operations of the abstract data type, userlist will be used to design the algorithms for the operations of the abstract data type, userset. The following are the algorithms for the operations of the abstract data type, userset.

```plaintext
userset emptyuserset()
1   emptyuserset = emptyuserset()
2   Display emptyuserset
```
The algorithm, emptyuserset takes nothing as parameter and it returns an empty set.

\[
\text{Boolean isemptyuserset(userset s)}
\]

1. Read s
2. Determine isemptyuserset
   2.1. isemptyuserset = isemptyuser(s)
3. Display isemptyuserset

The algorithm, isemptyuserset takes a set called, userset as parameter, and it returns true if the set is empty, otherwise it returns false.

\[
\text{Boolean memberuserset(mystring x, userset s)}
\]

1. Read data
   1.1. Read x
   1.2. Read s
2. Determine memberuserset
   2.1. IF isemptyuserset(s) THEN
       2.1.1. memberuserset = false
       ELSE
       2.1.2. IF equalstring((x, headuserid(s))) THEN
           2.1.2.1. memberuserset = true
           ELSE
           2.1.2.2. memberuserset = memberuserset
                        (x, tailuserid(s))
   3. Display memberuserset

The algorithm, memberuserset takes two parameters, x and s. The algorithm uses another algorithm called equalstring to determine if x is a member of the set, s.

\[
\text{userset insertuserset(mystring x1,x2, x3, x4, x5, x6, x7, x8, x9, x10, userset s)}
\]

1. Read data
   1.1. Read x1
   1.2. Read x2
   1.3. Read x3
   1.4. Read x4
   1.5. Read x5
   1.6. Read x6
   1.7. Read x7
   1.8. Read x8
   1.9. Read x9
   1.10. Read x10
   1.11. Read s
2. Determine insertuserset
   2.1. IF memberuserset(x1, s) THEN
       2.1.1. insertuserset = s
       ELSE
       2.1.2. insertuserset = consuser(x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, s)
   3. Display insertuserset

The algorithm, insertuserset takes all the attributes of userset, each of them is of the type mystring, and a userset. The algorithm uses one of the algorithms for the operations of userset, called memberuserset to check if the parameter, x1 is a member of the set of userid. If it is a member, it does not insert the record into the set, otherwise, it inserts it into the set.

\[
\text{Myset removeuserset(mystring x, user set s)}
\]

1. Read data
   1.1. Read x
   1.2. Read s
2. Determine removeuserset
   2.1. IF memberuserset(x, s) THEN
       2.1.1. IF equalstring(x, headuserid(s)) THEN
           2.1.1.1. removeuserset = tailluser(s)
           ELSE
           2.1.1.2. removeuserset = insertuserset(headuserid(s),
                                                  headsurname(s),
                                                  headfirstname(s),
                                                  headlastname(s),
                                                  headcategory(s),
                                                  headadminno(s),
                                                  headdept(s),
                                                  headaddress(s),
                                                  headusername(s),
                                                  headpassword(s),
                                                  removeuserset
                                                  (x, tailluser(s)))
       ELSE
       2.1.2. removeuserset = s
   3. Display removeuserset

The algorithm, removeuserset takes a userid, which is of the type mystring, and userset. The algorithm checks for the membership of the userid in the set of all userid in userset. If it is a member, it removes the whole record that has userid, otherwise, it does not remove it from the set.

5. DESIGN OF ALGORITHMS FOR THE OPERATIONS OF ABSTRACT DATA TYPE, TABLE/RELATION

For each of the data entities, users, books, loans and copies, the algorithms for the operations of its abstract data type, set will be used to design its abstract data type, table/relation. Example, for the data entity, users, the algorithms for the operations of userset will be designed using the algorithms for the operations of
abstract data type, usertable. The following are the design of algorithms for the operations of the abstract data type, usertable.

usertable emptyusertable()
1 Determine emptyusertable
1.1 emptyusertable = emptyuserset()
2 Display emptyusertable

The algorithm, emptyusertable takes nothing as parameter and it uses the algorithm for the operation of userset to return an empty table.

Boolean isemptyusertable(usertable t)
1 Read t
2 Determine isemptyusertable
2.1 isemptyusertable = isemptyuserset(t)
3 Display isemptyusertable

The algorithm, isemptyusertable takes one parameter, which is a table of the type usertable. The algorithm uses the algorithm for the operation of userset to return true if the table is empty, otherwise, it returns false.

Boolean memberusertable(mystring x; usertable t)
1 Read data
1.1 Read x
1.2 Read t
2 Determine memberusertable
2.1 memberusertable = memberuserset(x, t)
3 Display memberusertable

The algorithm, memberusertable takes two parameters, which are the primary key of the table, x and a table, t. The algorithm uses the algorithm for the operation of userset to return true if x is a member of the primary key of the table, otherwise, it returns false.

mystring locatesurname(mystring x; usertable t)
1 Read data
1.1 Read x
1.2 Read t
2 Determine locatesurname
2.1 IF memberusertable(x, t) THEN
2.1.1 IF equalstring(x, headuserid(t)) THEN
2.1.1.1 locatesurname = headsurname(l)
ELSE
2.1.1.2 locatesurname = locatesurname(x, tailuser(t))
ELSE
2.1.2 locatesurname = “The primary key does not exist”
3 Display locatesurname

The algorithm, locatesurname takes two parameters, which are the primary key x, of the type mystring and a table, t, of the type usertable. The algorithm uses the algorithms for the operations of userset to determine if x is a member of the set of primary keys, if it is, it locates the primary key x in the table and returns the corresponding surname that has that primary key, otherwise, an appropriate error message is returned.

mystring locatefirstname(mystring x, usertable t)
1 Read data
1.1 Read x
1.2 Read t
2 Determine locatefirstname
2.1 IF memberusertable(x, t) THEN
2.1.1 IF equalstring(x, headuserid(t)) THEN
2.1.1.1 locatefirstname = headfirstname(l)
ELSE
2.1.1.2 locatefirstname = locatefirstname(x, tailuser(t))
ELSE
2.1.2 locatefirstname = “The primary key does not exist”
3 Display locatefirstname

The algorithm, locatefirstname takes two parameters, which are the primary key x, of the type mystring and a table, t, of the type usertable. The algorithm uses the algorithms for the operations of userset to determine if x is a member of the set of primary keys, if it is, it locates the primary key x in usertable and returns the corresponding firstname that has that primary key, otherwise, an appropriate error message is returned.

In a similar manner, the following locate algorithms can be designed, locatelastname, locatecategory, locateadminno, locatedept, locateaddress, locateusername, and locatepassword.
2 Determine insertusertable
2.1 insertusertable = insertuserset(x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, t)
3 Display insertusertable

The algorithm, insertusertable takes eleven parameters, which represent all the attributes of the table, and a table, t, of the type usertable. The algorithm uses the algorithm for the operation of userset to insert all the attributes of the table into the table in such a way that there is no duplication of the primary key.

usertable removeusertable(mystring x, usertable t)
1 Read data
1.1 Read x
1.2 Read t
2 Determine removeusertable
2.1 removeusertable = removeuserset(x, t)
3 Display removeusertable

The algorithm, removeusertable takes two parameters, which represent primary key of the table, x and the table, t. The algorithm uses the algorithm for the operation of the userset to remove the record with the primary key x, if it exist in the table, otherwise, nothing will be removed.

6. DESIGN OF ALGORITHM FOR THE OPERATIONS OF ABSTRACT DATA TYPE, FILE

For each of the data entities, users, books, copies and loans, data stored in its abstract data type, table will be written to a text file, and be read from a text file into the table. Example for the data entity, users, data stored in the ADT, usertable will be written to a text file called userfile, and be read from the userfile into usertable. Therefore, for each of the following data entities, users, books, copies and loans, the file higher language data management facilities will be used to write data from its table into its file, read data from its file into its table.

7. DESIGN OF RECURSIVE ALGORITHMS FOR MANIPULATING RELATIONAL DATABASE

The recursive algorithms for manipulating the database will relate two or more tables, using the primary and foreign keys. Illustrating with an example, suppose we want to design the recursive algorithm that will extract the detail information about all the books that are on loan to a particular user. The recursive algorithm follows below.

Void userloan(loantable loan, copytable copy, booktable book, mystring iduser)
1 Read data
1.1 Read loan
1.2 Read copy
1.3 Read book
1.4 Read iduser
2. Determine userloan
2.1 IF isemptyloantable(loan) THEN
// Do nothing for base case
ELSE
2.2.1 catno = headcatalogno(loan)
2.2.2 loandate = locateloandate(catno, iduser, loan)
2.2.3 loandue = locateloandue(catno, iduser, loan)
2.2.4 isbn = locatecopyisbn(catno, copy)
2.2.5 title = locatetitle(isbn, book)
2.2.6 author = locateauthor(isbn, book)
2.2.7 edition = locatedition(isbn, book)
2.2.8 Display iduser
2.2.9 Display catno
2.2.10 Display loandate
2.2.11 Display loandue
2.2.12 Display isbn
2.2.13 Display title
2.2.14 Display author
2.2.15 Display edition
2.2.16 userloan(loantail(loan), copy, book, iduser)

The above recursive algorithm related three tables, loantable, copytable and booktable in order to produce the required information. In terms of relational databases, the same information can be produced using SQL. Illustrating with another example, suppose we want to design a recursive algorithm that will retrieve information about all copies of books that are on loan. This algorithm, listallloan takes four tables as parameters, which are the usertable, loantable, copytable and booktable. The algorithm relates these four tables with the aim of displaying the catalogue number, isbn, title and author of all the copies of books that are on loan; together with the userid, surname, first name, last name and department of the borrowers; and the due date of each copy of a book that is on loan.
Void listallloan(usersetable user, loantable loan, copytable copy, booktable book)

1. Request data
   1.1 Read user
   1.2 Read loan
   1.3 Read copy
   1.4 Read book

2. Determine listallloan
   2.1 IF isemptyloantable(loan) THEN
       //Do nothing for base case
   ELSE
     2.2.1 catno = headcatalogueno(loan)
     2.2.2 iduser = headuserid(loan)
     2.2.3 loandue = locateloandue(catno, iduser, loan)
     2.2.4 surname = locatesurname(iduser, user)
     2.2.5 firstname = locatefirstname(iduser, user)
     2.2.6 lastname = locatelastname(iduser, user)
     2.2.7 dept = locatedept(iduser, user)
     2.2.8 isbn = locatecopyisbn(catno, copy)
     2.2.9 title = locatetitle(isbn, book)
     2.2.10 author = locateauthor(isbn, book)
     2.2.11 Display catno
     2.2.12 Display isbn
     2.2.13 Display title
     2.2.14 Display author
     2.2.15 Display firstname
     2.2.16 Display lastname
     2.2.17 Display dept
     2.2.18 Display isbn
     2.2.19 Display title
     2.2.20 Display loandue
     2.2.21 listallloan(user, tail(loan), copy, book)

8. IMPLEMENTATION OF THE ALGORITHMS
All the algorithms for each of abstract data types for the following data entities, users, books, copies and loans were implemented as Java methods. Each abstract data type for each data entity was implemented as a Java class. Each implemented class had its interface class, thereby separating the specification of the class from its implementation. Object oriented concepts of inheritance and composition were used to establish relationships between the classes. For the data entity, users, the following java classes were implemented, userlist, userset, usertable and userfile, as shown in the class diagram of figure 3. Another java class, called relational was used to implement all the recursive algorithms that related the various tables, and a test program was written, which was used to store data into the four tables, and write data into the four files, including reading from the files into the tables.

9. RESULT OF THE IMPLEMENTED ALGORITHM
The interesting results that the implemented algorithms produced were the relevant reports, which were obtained by relating two or more tables. After entering data for the data entities, users, books, copies and loans, the test program was able to produce the following reports: Detail information about each copy of a book that a particular user has on loan; for each copy of book on loan, it produces the detail information about the user that has the book on loan, and the detail information about that copy of a book, together with its loan information.

Figure 3: Class Diagram for the User Data Entity
10. CONCLUSION

This paper has been able to use the algorithms for the following underlying abstract data types, list, set to design and implement the algorithms for the abstract data type table/relatin, using an object oriented programming language, Java. For each of the following data entities, users, books, copies and loans, novel recursive algorithms that related two or more tables have been developed and implemented.

REFERENCES


Authors’ Brief

Oguike, Osondu Everestus is a Senior Lecturer in the Department of Computer Science, University of Nigeria, Nsukka, Enugu State, Nigeria. He has received many academic prizes and scholarships as a result of his outstanding academic performance. He is interested in modeling the performance of parallel computer system. He can be reached by phone on +2348035405100 and through E-mail osondu.oguike@unn.edu.ng

Ugwu Celestine Ikechukwu is a System Analyst/Programmer in the Department of Computer Science, University of Nigeria, Nsukka, Enugu State, Nigeria. He obtained his B.Sc and M.Sc in the Department of Computer Science, University of Nigeria, Nsukka. He is proficient in Java programming. He can be reached through Email: celestiyke@yahoo.com and phone number +2348030899997.