Implementation of Production Normalize DB based on Functional Dependency Table (FDT)

Maitham Ali Naji

Foundation of Technical Education, College of Electrical and Electronic Techniques, Power Department

ABSTRACT

This paper presents the result of previous papers. It is provides computerized technique to design and create normalize relational database based on Functional Dependency Table (FDT). The system use Data Access Object (DAO) to define tables to the table collection. After the tables and fields have been created, the system will define an index and primary key attributes to the first field of each table, finally the relationships type is created between tables depend on the primary keys and foreign keys.

Keywords Relational Database, DAO Jet Engine, Visual Basic.

تطبيق انتاج قواعد بيانات منسقة استنادا على جدول الوظيفة الاعتمادية

المستخلص

هذه البحث يمثل نتيجة لبحوث سابقة والتي تهدف الى توفير تقنية حاسوبية لتصميم وبناء قواعد بيانات علائقية منسقة من خلال جدول الوظيفة الاعتمادية (FDT). النظام استخدم كائن DAO لتعريف الجداول الى مجموعاتها. بعد تكوين الجداول وحقولها، سوف يعرف النظام الفهرس والمفتاح الرئيسئ للحقل الاول لكل جدول. واخيرا" سوف تنشاء العلاقات بين الجداول اعتمادا" على المفتاح الرئيسي والثانوي.

List of Abbreviations:

RDBMS	Relational Data Base Management System
SQL	Structure Query Language
OODBM	Object Oriented Data Base Management
ERD	Entity Relationship Diagram
ERT	Entity Relationship Table
PK	Primary Key

Introduction

The relational database model focuses on logical representation of the data and its relationships, rather than on the physical storage details. The relational database perceived by the user to be a collection of tables in which data are stored [1]. RDBMS A DBMS that manages relational databases (and no others); equivalently, a DBMS that implements the relational model. Note: SQL DBMSs must be regarded as only approximately relational at best, since SQL involves so many departures from the relational model [2].

Each table is a matrix consisting of a series of row/column intersections. Tables, also called relation, are sharing to each other by sharing a common entity characteristic. The relationship type (1:1, 1: M or M: N) is often shown in relational schema to connect entities [3].

We define object orientation as a set of design and development principles based on conceptually autonomous computer structure known as objects. OODBM is a database management system that integrates the benefits of typical database systems with more powerful modeling and programming characteristics of the object oriented data model [4].

Paper [5] applies normalization rules to ERT to produce FDT. The current system produce normalize database based on FDT.

The DAO Object Hierarchy

The DAO is the name given to the object model for the Jet engine. The DAO have many levels, the top level object of the DAO model, there is a hierarchy of other objects and their collection, each with their own properties and methods, which allow you to access the data in your database [6]. The object hierarchy is shown in figure 1.

1. DBEngine

The top level object in the DAO object hierarchy is the DBEngine. There can be only one DBEngine object; it contains all the other objects in the hierarchy. You do not create the DBEngine object, because it is available by default.

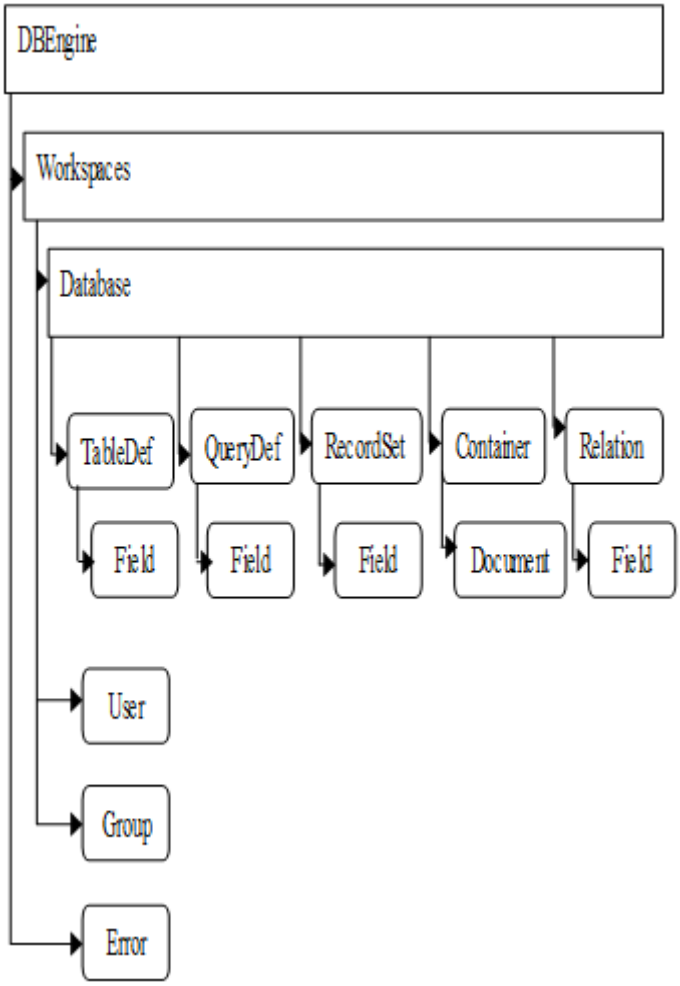


Figure 1. The Data Access Object (DAO) hierarchy [6].

2. Workspaces

Most of the DAO Hierarchy lies under the workspaces collection. A workspace is an organizational structure that allows you to group database connections into a named area or session. You will need more than one workspace if you want to access groups of databases in different ways, such as one group through the jet engine and the other through ODBCDirect.

3. Groups and Users

In a Jet database, such as Microsoft Access, you may define workgroups consisting of Groups and Users, and their

associated permissions for accessing the database.

4.Relation

A Relation object represents a relationship between fields in tables or quires. You can use the Relation object to create new relationships [7].

5. Database

The Database object is the most important object we will be dealing with because it allows us to interface with the database.

To look at the structure of a database, we need first look at the objects, collections, and properties that exist in our DAO hierarchy; figure 2 shows the relevant structure and properties [6].

1.TableDefs

Just as every object under the DBEngine object can be a collection or object, TableDefs is a collection of TableDef objects.

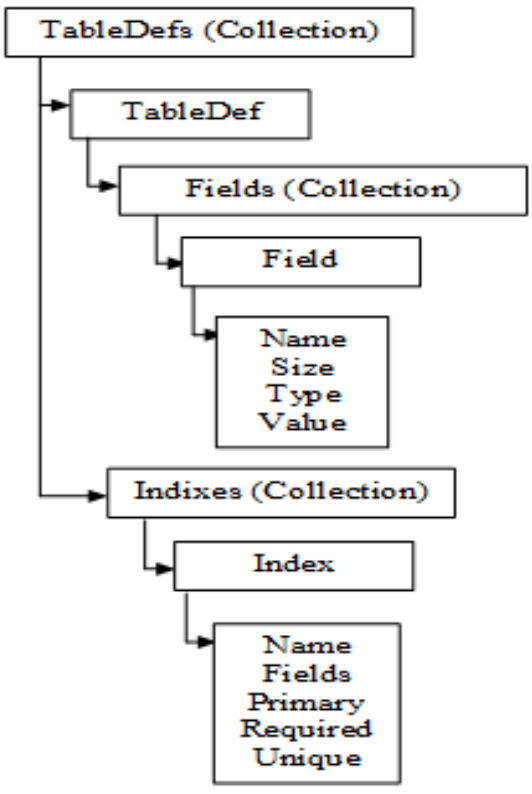


Figure 2. The tabledefs collection hierarchy [6].

2-Fields

Of more interest than the table names are the fields that exist in the table. Each tableDef object contains a Fields collection, which in turn contains one or more Field objects. The most important properties are (name, size, source field, source table, type and value).

3-Indexes

Just as you can look at the various fields you have in a table, you can also view the indexes on it. Each TableDef object can have an associated Indexes collection with one or more Index Object in it. The most important properties are (fields, foreign, name, primary and unique).

2-The System's Flowcharts

In [8], the program divides the entered sentence into three parts, Subject Group, Verb Group and complement group. Subject and complement group are representing the source and destination entities or attributes with its relationship. Verb group is representing the relation name that connects the source and destination entities.

In [9] the system converts the subject, verb, and complement groups to ERT instead of ERD.

In [5] the author applies normalization rules based on primary keys on ERT. The program will split the primary key attributes from non key attributes to build the functional dependency for each table. The Functional Dependency will store in file called FDT that contains Table Name, PK Fields, and Table Fields. As shown in figure 3.

The Functional Dependency Table (FDT) for the following examples shown in figure 3. Employee has Emp ID, Emp name and Dept ID. Department has Dept ID , Dept Name and Dept Location. Skill Master has Skill Code and Skill Name. Employee Skill has Skill Code, Skill Name and Skill level.

FD Table

	Table Name	PK Fields	Table Fields
▶	Employee	Emp ID	Emp Name , Dept ID
	Department	Dept ID	Dept Name , Dept Location
	Skill Master	Skill Code	Skill Name
	Employee Skill	Skill Code	Skill Name , Skill level

Display FD

Figure 3. Functional Dependency Table (FDT).

The FDT will be an input to the current system. The system will pass through six stages to produce relational database. As shown in figure 4.

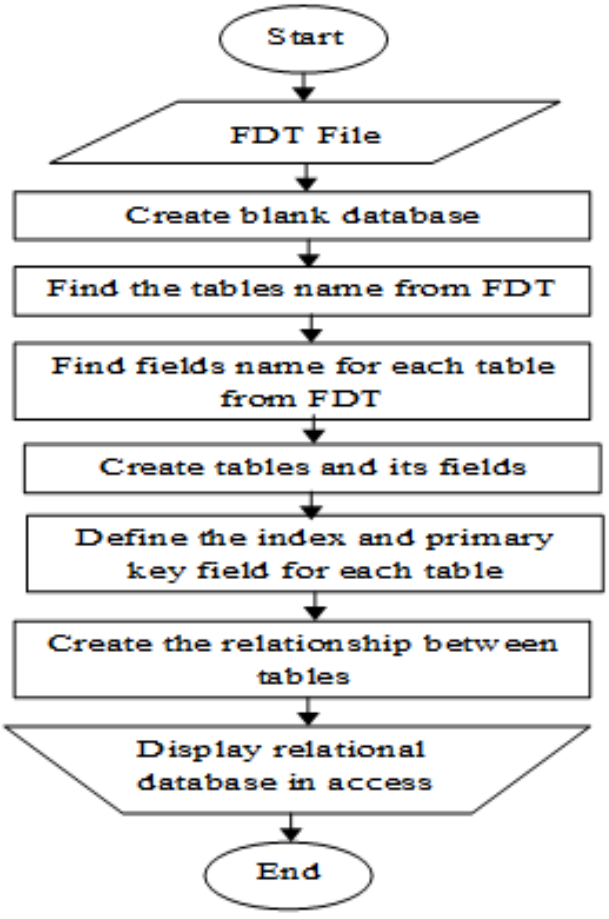


Figure 4. Block diagram of production normalize DB based on FDT.

The first stage, the system will delete the old Database if it's existing; then create new one in first workspace. As shown in figure 5.

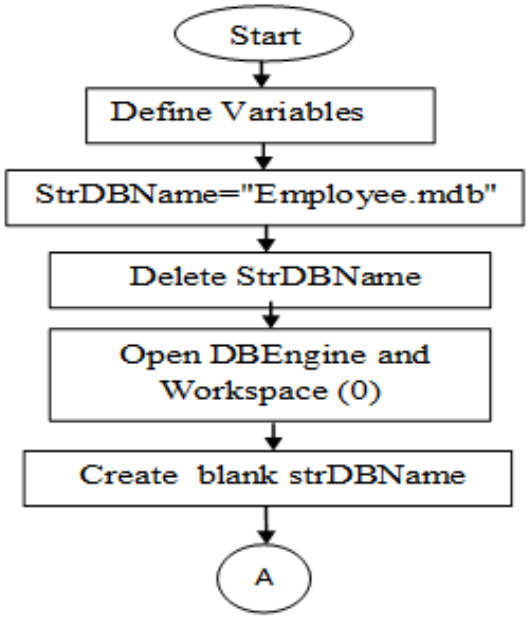


Figure 5. Create new database.

The second stage extraction tables names; the system will store the content of FDT's "Table Name" field that contains tables names in list called "Table List". As shown in figure 6.

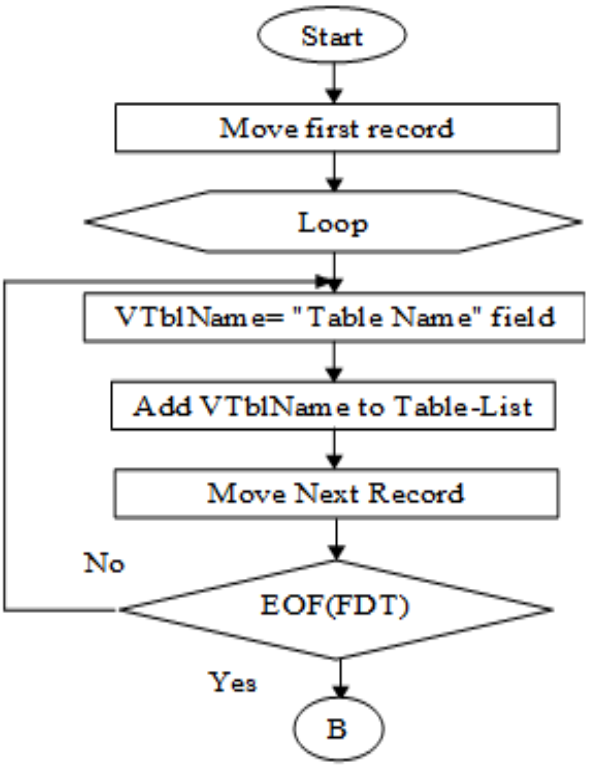


Figure 6. Extracting tables name from FDT.

The output of this stage will pass the table name and its fields to procedure called "Create Table and Fields" to create table in DAO object. This is done by store the primary key attribute and non key attribute in list called "Field List". The primary key attribute will store from PK-Field of FDT in first location of the Field list, while the non key attributes will store from Table Fields to Field List. The Table Fields may contain more than one attributes. These attributes converted to an array by using Split instruction, and then each element in array will be added to Field List. This process is repeated until end of FDT file. As shown in figure 7.

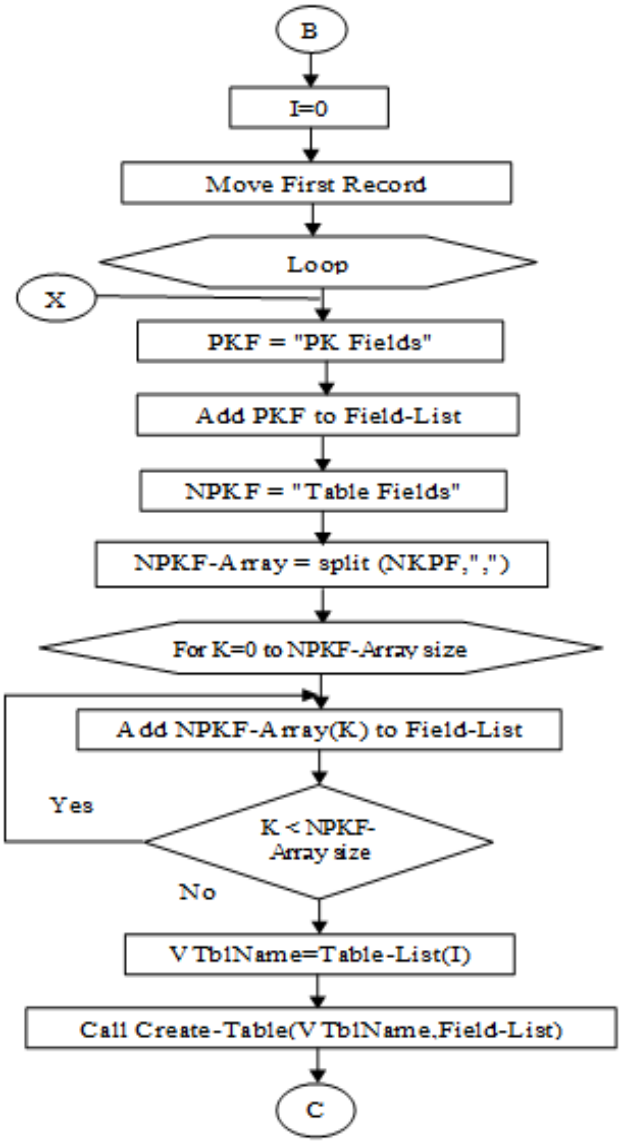


Figure 7. Extracting fields list from FDT.

The output of previous flowchart will be List-Table(0)=Employee

List-Field of Employee =[Emp ID, Emp Name, Dept ID]

List-Table(1)=Department

List-Field of Department =[Dep ID, Dept Name, Dept Location]

List-table(2)=Employee Skills

List-Field of Employee Skill =[Skill Code, Emp ID, Skill Level]

List-table(3)=Skill Master

List-Field of Skill Master =[Skill Code, Skill Name]

The Create Table and fields procedure shown in figure 8 does the following:

1. Create table and added it to the tables collection in DAO object.
2. Create the table's fields after the user define the field type (Text, Number, Date,…, etc) and added it to field collection.
3. Assign index to first field for each table; then added the index to index collection. As illustrated in DAO Object Hierarchy section.

The last stage; the system will establish the relationships between tables. This is done by storing the tables' names and their fields in two dimensional arrays. As shown in figure 9. The result of the flowchart is shown in table 1.

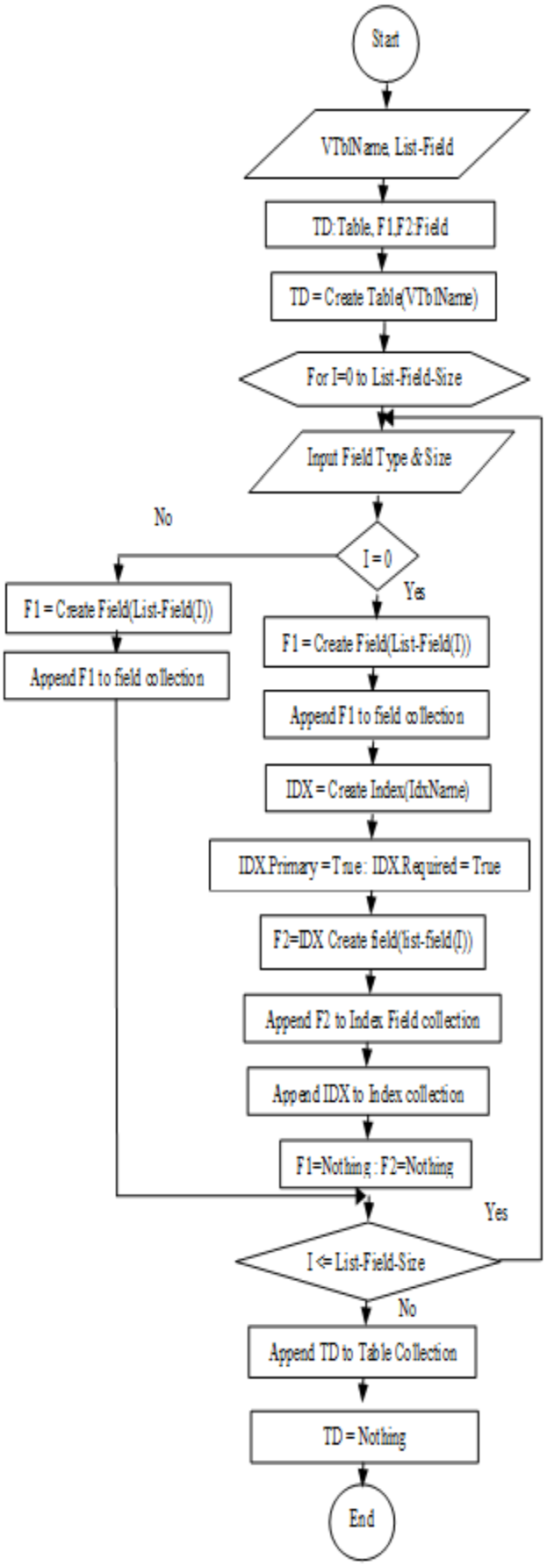


Figure 8.Create table and fields

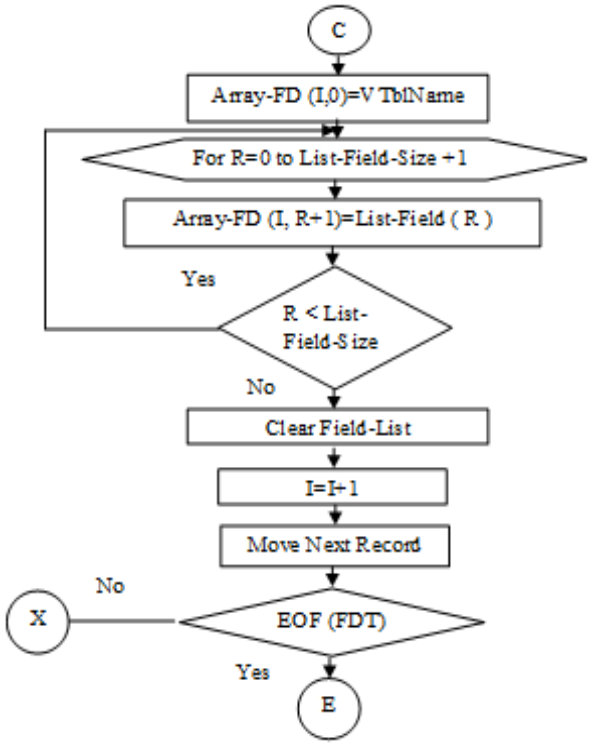


Figure 9. Conversion table into two dimensional arrays.

Table 1. Tables names and their fields.

	Column 0	Column 1	Column 2	Column 3	…	Col 10
1	Employee	Emp ID	Emp Name	Dept ID		
2	Department	Dept ID	Dept Name	Dept Location		
3	Employee Skills	Skill Code	Emp ID	Skills Level		
4	Skill Master	Skill Code	Skill Name			
.						
10						

The system generates two types of relationships; the first type is (1:1). Like the relationship between Employee Skill and Skill Master Tables. The relation called (1:1) if two tables connect via primary key.

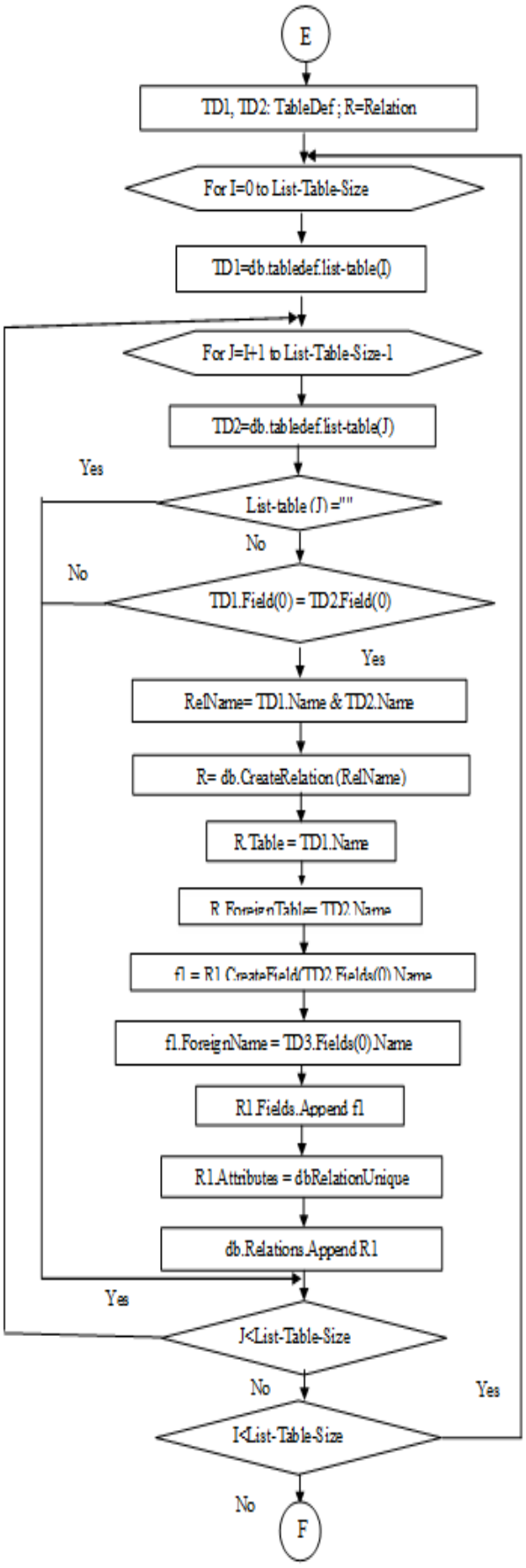


Figure 10. 1:1 Relationship.

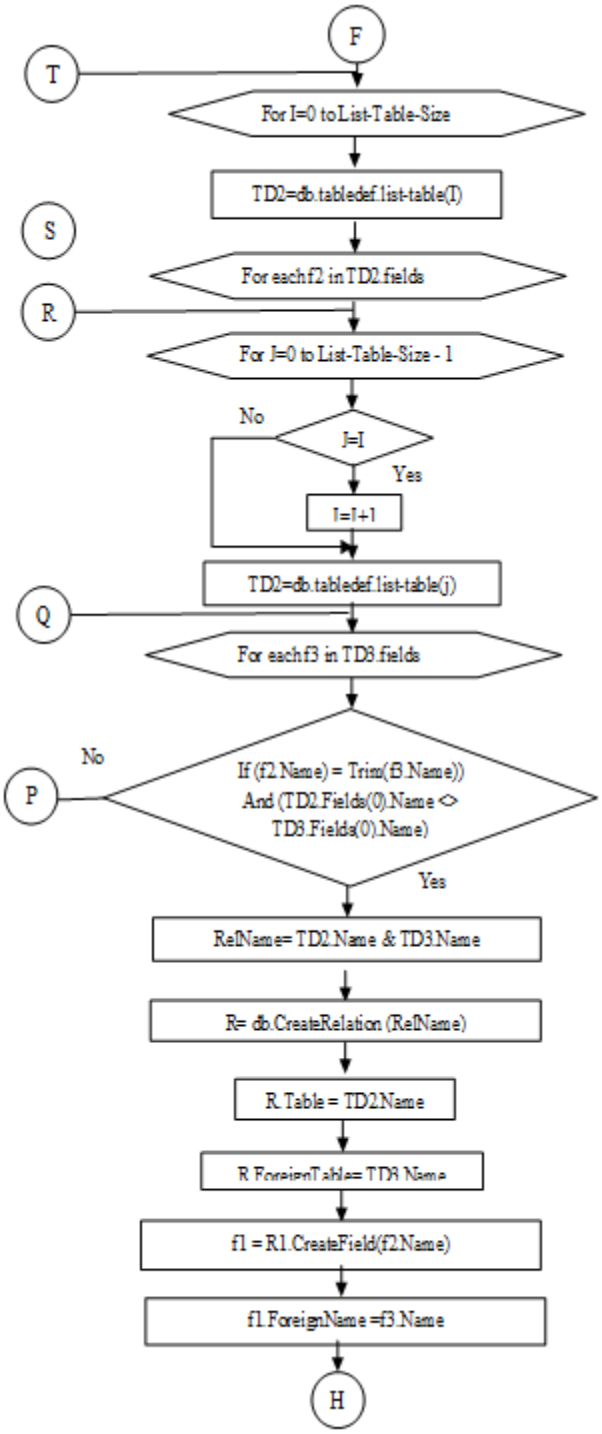
The flowchart shown in figure (10) does the following:

1. Add the tables to the collection of tablesdefs.
2. It has two nested loop; will compare the first item of column one (Shaded column) in two dimensional array with all other items in the same column; if the match success the system creates relationship, otherwise will take another item.
3. The program will define relationship name.
4. Create the relationship between tables.
5. Add the relationship to the collection.

The second type is (1: N). Like the relation between Departments and Employee tables. The relation called (1:N) if one tables has primary key and the second table has foreign key.

The flowchart shown in figure 11 does the following:

1. Add the tables to the collection of tablesdefs.
2. It has four nested loop; will take the primary key (item in shaded column) and compare it with all other items in array (except item in shaded column). If the match success the system creates relationship, otherwise will take another item. The steps (3-5) are similar to above steps.



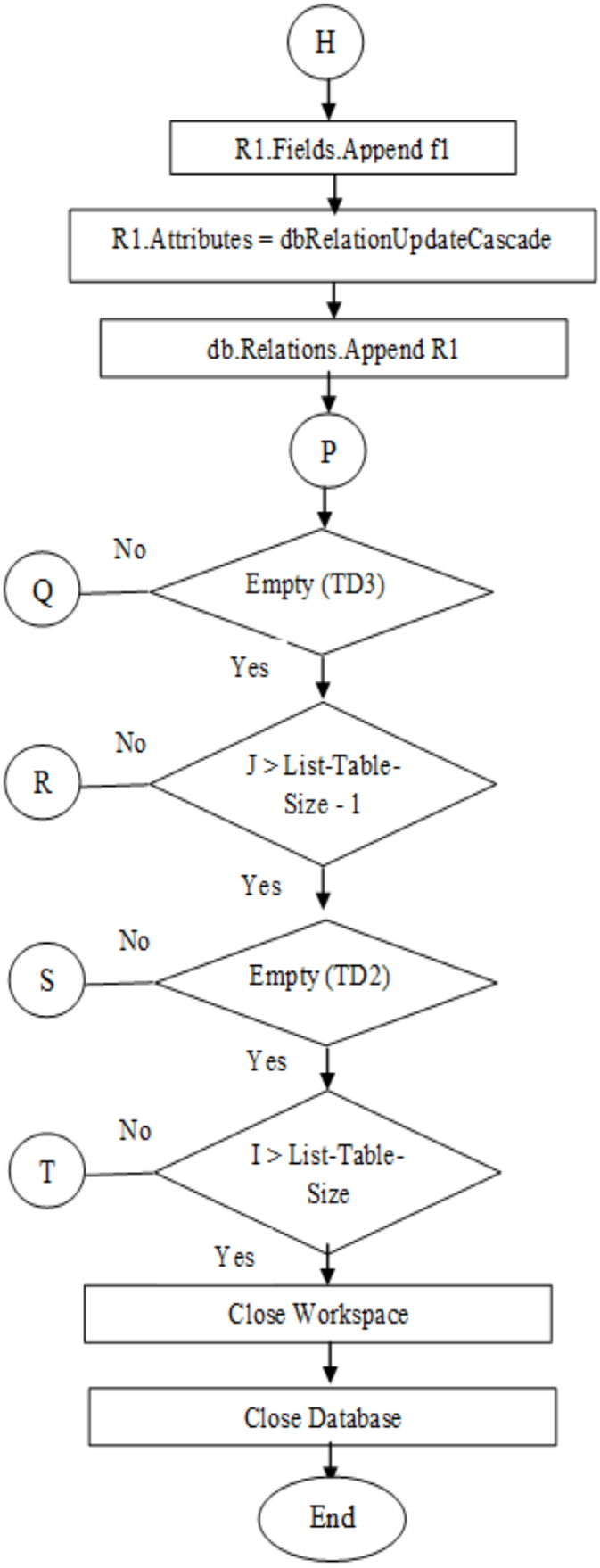


Figure 11. 1: N Relationship.

3-The Implementation

The system has been implemented in Visual Basic language. The program contains the following main windows as shown in figure 12.

Figure 12. Main Form Window.

By click on "Create Database" button, the program will create blank database, and the process will be done on FDT to extract the tables and it is fields, the system will ask from the user to define the data type (Number, Text, Date, etc) for each field before adding the field to the collection. As shown in figure 13.

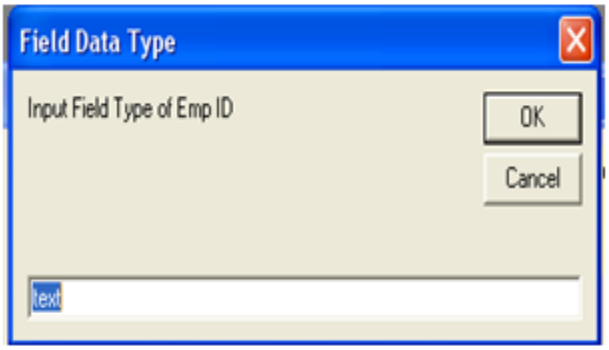


Figure (13): Input Field Data Type.

This process is repeated to give data type to all fields of the tables as shown in Figure (14).

Figure 14. Creating Tables in Access.

Finally the system will establish the relationship between tables shown in figure 15 and displays message "The Relational Database is done" as.

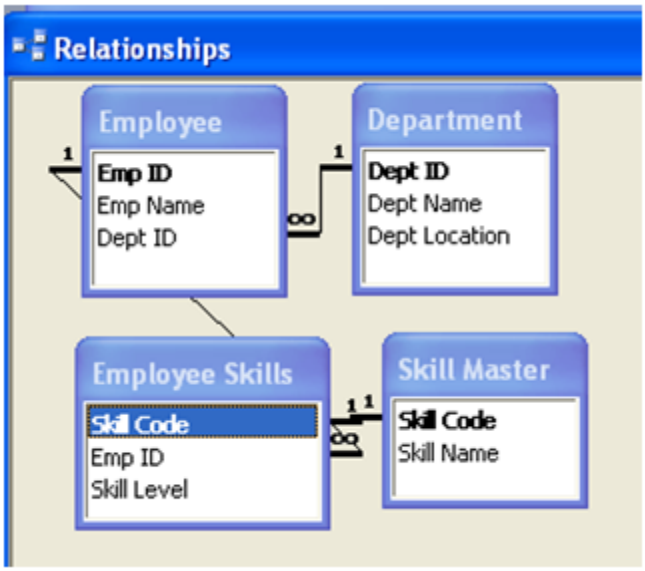


Figure 15. Creating Relationship between Tables.

Conclusion

The majors points can be concluded from this system, the first one the system is easy to use, since the user just enter the business rules (sentences) the output system will be normalize DB.

The second one, some time you find some persons had high skill in DB design but he/she does not like to learn application that help him to apply his/her design, so the system is very useful for this type of people.

The last point the system is very useful to provide secure DB by transmit the DB as a FDT to end user. The end user can convert the FDT to DB by using this program.

References

1. Wisley, Addison; (2010). Fundamentals of Database System. USA, 1172.
2. Date, C.J., (2007). The Relational Database Dictionary. Trafford Publishing, 215.
3. Carter, John, (2000). Database Design and Programming with Access, SQL and Visual Basic. McGraw-Hill Publishing Company, 483.
4. Rob, Morris, Coronel, (2010). Database Systems: Design, Implementation, and Management. McGraw- Hill Edition, USA, 675.
5. Naji, Maitham Ali, (2013). Improve Entity Relationship Table (ERT) by Using Primary Key. Accepted not published in Al-Mustansiriyah Journal of Science.
6. Snowdow, Nick, (2002). Oracle Programming with Visual Basic. SYBEXInc,USA,715.
7. VB Help, (2001). Microsoft Developer Network (MSDN).
8. Naji, Maitham Ali, (2011). Implementation Sentence Analysis via Verb. Foundation of Technical Education, 24(8):83-90.
9. Naji, Maitham Ali, (2011). Proposal of Creating Entity-Relationship Table (ERT) From English Sentence Group. Iraqi Journal of Computers, Communication, Control and Systems Engineering, 11(2): 79-88.