ASSOCIATIVE DATA MODEL AND CONTEXT MAPS

MINGHUI HAN

A MAJOR REPORT

IN

THE DEPARTMENT

Of

COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE

CONCORDIA UNIVERSITY

MONTREAL, QUEBEC, CANADA

AUGUST 2001

0-612-64083-3

Abstract

Associative Data Model and Context Maps

Minghui Han

This report presents the possibility of using *context maps* to represent *associative data model.* This new technology for *associative data model* can be presented as the *joined maps* (*jMaps*) of concepts and relationships. The solution for converting a set of *context maps* into one database or retrieving information from the database to *context maps* was developed. The software was developed by using *VBA* (Visual Basic for Application), which can give us access to *Microsoft Office* for integration with databases. The implementation for this technology was demonstrated by using *MS Excel* spreadsheet to display the *associative model* of data and *MS Access* to store a set of converted *context maps*.

Acknowledgements

I wish to thank all those who made the final realization of this dissertation possible. It is not possible to mention all their names, however I would like to express my special gratitude to the following contributors.

I am pleased to express my gratitude to Professor T. Radhakrishnan, Chair of Department of Computer Science, for acceptance to be my co-supervisor. He made many useful comments for this report. My thanks are extended to Prof. Peter Grogono, an examiner for my work, for his valuable comments on this report. My thanks also go to Halina Monkiewicz, the Graduate Program Secretary, for her support and collaboration. I am greatly indebted to my supervisor, Prof. W.M. Jaworski, who encouraged my interest in the subject of software information system, for his technical advice, generous attention, constant help and critical remarks throughout this work. He patiently guided to me with his knowledge about information systems and encouraged me in tackling various difficult issues.

My sincere thanks are due to the support given by my company, Motorola Canada Software Center, Montreal, Quebec, Canada. My special thanks to Gerald Dunkelman, Operation Manager and Alf Lund, project Manager for their inspiration, encouragement in many ways during my master program study.

Finally, I would like to express my deep gratitude to my wife, Dawei, for her all the support and she always believe me and I can always rely on her.

Contents

List of Figures

List of Tables

Chapter 1

1. Introduction

1.1 Background

There are many representations and methodologies for information systems and software engineering, such as *CASE* tools and Rational Rose *UML*, which can be presented with graphical notations for the information system views.

However it is a challenge to develop a methodology with safety critical systems which needs to be simple and easy to implement. The *joined maps* viewed as *context maps* in this report is one way to represent the above requirement. The *joined maps*, or *jMaps*, are a notation and a method for representing systems architecture, structures, processes and reusable templates. The *jMaps* can be synonyms with syntax maps. This technology was first introduced by W.M. Jaworski [1995]. The technology was initially developed as a means of recovering and refining knowledge from legacy system. This technology has a history of names. During the late 1970s and early 1980s, based on conceptual graphs introduced by J.F.Sowa [1984], it was named as *ABL*, or *Array Based Language* (Jaworski [1987]). In the late 1980s, it was renamed as *ABL/W4*. *W4* represents as what, when, where and which. In the early 1990s, Prof. Jaworski [1995], by considering existing notations and methodologies, named this technology as *jMaps*. In the late 1990s until now, *jMaps* can be presented as *Context Maps* (Jaworski [1999]). With *jMaps* or *Context Maps* technology, by using the popular concept of a spreadsheet it is feasible to communicate the design information to different audiences. The *jMaps* notation allows efficient recovery and modeling of generic schemata for processes, objects and views of information systems.

The *associative data model* was developed by Simon Williams [2000]. The *associative model* treats the information in the same way as the human brain, i.e. treats the things with association between them. Those associations can be expressed through the simple subject-verb-object syntax of an English sentence. The *associative model* divides the real–world things with two kinds of sorts: *Entities* and *Associations*. According to Simon Williams [2000], *Entities* are the things that have discrete, independent existence. An entity's existence dose not depend on any other thing. *Associations* are the things whose existence depends on one or more other things, if any of those things ceases to exist, the thing itself ceases to exist or becomes meaningless.

The *associative model* overcomes the limitations of the relational model and avoids the complexities of the object model by structuring information in a more accessible and intuitive manner than either. The *associative model* overcomes two fundamental limitations of current programming practice: the need to write new programs for every new application, and the need to store identical types of information about each instance. It also offers a superior distributed data model, allowing one database to be distributed over many geographically dispersed web servers. Moreover, associative databases may be readily tailored to serve different requirements simultaneously, and different databases may be easily combined and correlated without extra programming

By considering the basic concepts of *associative data model*, it becomes possible for us to use *context maps* to represent *associative data model*.

1.2 Objective of Study

The main purpose of the research work reported herein is to introduce the new method of using *context maps* to represent *associative data model*. Based on this new technology, we will design and develop a software for converting a set of *context maps* into one database or retrieving data information from the database. The *associative data model* will be presented as the *joined maps* (*jMaps*) of concepts and relationships in the *MS Excel* spreadsheet.

The main purpose of this project is based on the *associate model* of information to produce the related *jMap* in the form of *MS Excel* spreadsheet. By considering *context maps* for associative data model, it is focused on using *context maps* to represent the *associative data model*, and exporting *context maps* into database or recovering the data from a database to spreadsheets in the *jMaps* format.

The application software was written by *VB* with emphasis on using Micro Office application. Since our developing software is a small project, the *MS Excel* spreadsheet and *MS Access* database are sufficient in using this project.

1.3 Project Scope

The research work for this project was supervised by Prof. W.M. Jaworski. The work study was started in January 2001. The procedure to develop this project is structured in the following way:

1) Try to get familiar in using associative data model, especially in understanding the basic concepts of this new technology for representing the database model.
2) Analyze the basic requirements for this project. List the relationships between entities and associations for a special example.
3) Do research on the *jMap* notation, and converting the *associative model* with *jMaps* notations into a spreadsheet.
4) Project design, source coding in *MS Excel* by using *VBA*, with special emphasis on converting a set of data into the database and restoring the *jMaps* from the database.
5) Integrate the program, and make all functions work.
6) A deliverables project package will contain a full description of manual, sample Excel file and sample database file
7) Make a conclusion for this research work and provide recommendations for future works.

Chapter 2

2. Associative Model Introduction

2.1 Data Model

In the database management system, we can record the existence and properties of things in the real world. The transition from things which we want to record information into a database relies on using a modeling system. The modeling system consists of three layers of abstraction: a conceptual layer, a logical layer and a physical layer.

- The conceptual layer is the highest level and is more abstract than the other layers. It describes what should the modeling system in representing things in the real world, and sets the rules about how they may be used in the modeling system.
- The logical layer describes the logical building blocks which the database uses to store and access data, and how to map the conceptual layer into logical layer.
- The physical layer is the lowest level which describes the physical building blocks which exist in the computer's memory and are stored and retrieved in its hardware storage. The physical layer decides how the logical building blocks map into physical layer.

In above layers, the conceptual and logical layers together make up the data model. In this case, we can conclude that the data model is a scheme for structuring data in databases, the logical and the physical layers together make up the database aspects.

The data model is fundamental for database management systems. According to Simon Williams [2000], five data models have been proposed and used since computers became available. Those five data models are: the *network model*, the *hierarchical model*, the *relational model*, the *object model*, and the *object/relational model*. In the above models, the two most significant and widely adopted models are the relational model and the object model. Today's database market is dominated by products based on the *relational model*.

2.2 Relational Model

The *relational model* was first described by Dr. Edgar Codd of IBM's San Jose Research Laboratory in 1970. Nowadays, the *relational model* is the foundation of almost every commercial database. The *relational model* stores data in special tables called "relations".

In the *relational model*, each table holds data for a particular type of thing or entity, such as customers, orders, students and so on. Within a table, each row represents one instance of the type of things that the tables stores and each column represents a piece of information that is stored.

Here is a simple example of customers and orders for which the source was taken from Simon Willams [2000]. The customers table has columns for customer number, name, telephone number, credit limit, outstanding balance and so on. The Orders table has columns for order number, date, customer number item, quantity and so on

Customers				
Customer number	*Name*	*Telephone no*	*Credit limit*	*O/S balance*
456	Avis	0171 123 4567	£10,000	£4,567
567	Boeing	0181 345 6789	£2,500	£1,098
678	CA	0123 45678	£50,000	£14,567
789	Dell	0134 56789	£21,000	£6,789

Table 1 Customers Relational Table

Orders				
Order no	*Date*	*Customer number*	*Item*	*Quantity*
11234	2-Mar-99	**567**	ABC345	150
11235	15-Mar-99	**789**	GGI765	25
11236	21-Apr-99	**789**	KLM012	1,000
11237	7-May-99	**456**	GHJ999	£6,789

Table 2 Orders Relational Table

Within each table, rows are uniquely identified by one or more special columns called primary keys. The relationship between an order and the customer who placed it is recorded by putting the customer's number into the "customer number" column of the order's row in the *Orders* table. This is an example of a foreign key. The foreign keys in table are shown in bold.

The *relational model* is the standard architecture for the database management systems. However it has some fundamental limitations such as the following:

- Each new relational database application requires a new set of programs. So the cost of application software increases.

- The relational database applications are difficult to customize for individual users.
- A relational database can not record a piece of data about a particular thing that is not relevant to all others of same type.
- It is difficult and sometimes not possible to combine two relational database.

2.3 Associative Model

2.3.1 General

The *Associative Model* is the first major advance beyond the *Relational Model.* The *Associative Model* of Data is the name given by Simon Williams [2000] to the set of concepts, structures and techniques underlying the *Sentences* database management system. The *Sentences(TM)* is an innovative database management system written in the Java language and based on the *Associative Model of Data.* The *associative model* builds on a body of academic research that includes: semantic networks, binary-relational techniques and the entity relationship model. We have added several important and unique concepts.

The *associative model* sees information in the same way as our own brains: as things and associations between them. These associations are expressed through the simple **subject-verb-object** syntax of an English sentence. For example:

The lake *is* coloured blue
Sherry *is* sister to Jim
Lee *has* a credit limit of $5,000
Montreal *is* located in Province of Quebec

A sentence may itself be the subject or object of another sentence, so the *associative model* can express quite complex concepts:

(Flight BA123 *arrives at* 20:15) *on* Monday
The Bible *says* (God created the World)

For previous Customers relational table, the sentence in the *associative model* can be described as following

Avis *is* a Customer
Avis *has* telephone number 0171 123 4567
Avis *has* credit limit £10,000
Avis *has* outstanding balance of £4,567
Boeing *is* a Customer
Boeing *has* telephone number 0181 345 6789
Boeing *has* credit limit £2,500
Boeing *has* outstanding balance £1,098

...and so on.

2.3.2 Associative Model Structure

According to Simon Williams [2000], an associative database comprises two data structures:

- **Items**, each of which has a unique identifier, a name and a type.
- **Links**, each of which has a unique identifier, together with the unique identifiers of three other things, that represent the source, verb and target of a fact that is

recorded about the source in the database. Each of the three things identified by the source, verb and target may each be either a link or an item.

The following example shows how the *associative model* would use these two structures to store the piece of information.

Example sentence:

"Flight AC1234 arrived at Montreal Doval on 12-Aug-2001 at 10:25am".

In the above sentence, we could divide seven items with:

the four things:

Flight BA1234,
Montreal Doval,
12-Aug-2001
10:24am

and the three verbs or prepositions

arrived at
on
at.

In this case, we need three links to store the data. They are:

Flight AC1234 arrived at Doval Airport
... on 12-Aug-2001
... at 10:25am

We can see that each line is one link. The first link uses "arrived at" to associate *Flight AC1234* and *Doval Airport*. The second link uses "on" to associate the first link and *12-Aug-2001*. The third link uses "at" to associate the second link and *10:25am*.

We can simply put brackets around each link. Written this way, our example would look like this:

((Flight BA1234 arrived at Doval Airport) on 12-Aug-2001) at 10:25am

This may look more like human language than the contents of a database, but if we chose for a moment to view the *associative model* through the eyes of the *relational model*, we see that any associative database can be stored in just two tables: one for items and one for links. Each item and link has a meaningless number to act as its primary key.

Items	
Identifier	Name
01	Flight AC1234
02	Montreal Doval
03	12-Aug-2001
04	10:25am
05	arrived at
06	On
07	At

Table 3 Items Associative Table

Links			
Identifier	Source	Verb	Target
11	01	05	02

12	11	06	03
13	12	07	04

Table 4 Links Associative Table

2.3.3 The Benefits of Associative Model

The *associative model* has following advantages:

- One program can be used to implement many different applications without being altered or rewritten in any way. The *associative mode* allows users to create new applications from existing ones. This will significantly reduce the costs of software development.

- By using the *associative model*, applications can permit features to be used or ignored selectively by individual users without the need for complex parameters or customisation.

- A database can record information that is relevant only to one thing of a particular type, without demanding that it be relevant to all other things of the same type.

- Separate databases can be readily correlated or merged without extra programming, and multiple databases distributed across many servers can be accessed by applications as though they were a single database.

2.4 The Bookseller Example

In this section, we will describe the bookseller example to look at the more sophisticated problem and to show how the *associative model* deals with this problem. The example was taken directly from Simon Williams [2000]. This example will be also represented by *context maps* in later chapter.

The domain of bookseller problem as described following:

> *An Internet retail bookseller operates through legal entities in various countries. Any legal entity may sell books to anyone. People are required to register with the legal entity before they can purchase.*
>
> *For copyright and legal reasons not all books are sold in all countries, so the books that each legal entity can offer a customer depend on the customer's country of residence.*
>
> *Each legal entity sets its own prices in local currency according to the customer's country of residence. Price increases may be recorded ahead of the date that they become effective.*
>
> *Customers are awarded points when they buy, which may be traded in against the price of a purchase. The number of points awarded for a given book by a legal entity does not vary with the currency in which it is priced.*

With *associative data model*, the schema that describes the structure of orders for this problem is as follows. The items in bold are entity types.

Legal entity sells **Book**
... worth **Points**
... in **Country**
... from **Date**
... at **Price**
Person lives in **Country**

Person customer of **Legal entity**

... has earned **Points**

... orders **Book**

... on **Date**

... at **Price**

In above data itself, the items in italics are entities. Now we define the group of them that we are using; two legal entities, two books, two customers and two countries:

Amazon is a **Legal entity**

Bookpages is a **Legal entity**

Dr No is a **Book**

Simon Williams is a **Person**

Simon Williams lives in *Britain*

Mary Davis is a **Person**

Mary Davis lives in *America*

Britain is a **Country**

America is a **Country**

Spycatcher is a **Book**

Next comes the price list:

Amazon sells *Dr No*

... worth *75 points*

... in *Britain*

... from *1-Jan-00*

... at *£10*

... in *America*

... from *1-Mar-00*

... at *$16*

Amazon sells *Spycatcher*

... worth *50 points*

... in *Britain*

... from *1-Jun-00*

... at £7

... in *America*

... from *1-Jun-00*

... worth *35 points*

... in *Britain*

... from *1-Jan-00*

... at *£8*

... in *America*

... from *1-Jan-00*

... at *$14*

Bookpages sells Spycatcher

... worth *35 points*

... in *America*

... from *1-Jun-00*

... at *$13*

Here, for each of our two customers we record the number of points awarded to date, together with a single order:

Simon Williams customer of *Bookpages*

... has earned *1,200 points*

... orders *Dr No*

... on *10-Oct-00*

... at *£10*

Mary Davis customer of *Amazon*

... has earned *750 points*

... orders *Spycatcher*

... on *19-Oct-00*

... at *$12*

Here is the metadata for the bookseller problem in diagrammatic form. The ovals represent items; the lines represent links. The circles on the lines are the anchor points for links between items and other links.

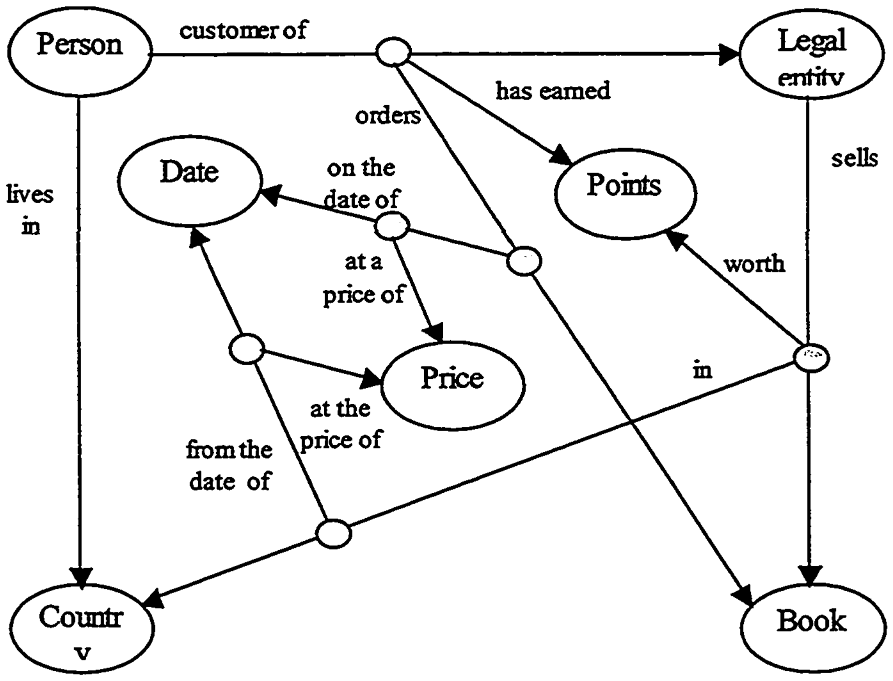


Figure 1 The Bookseller Problem in the Diagrammatic Form

Comparison with *associative model* and *relational model*, the associative schema usually take much less lines that record the same data as *relational model* requires to store an equivalent database.

Chapter 3

3. Context Maps

3.1 Context Paradigm

From the source of Dr. Jaworski at the website of www.gen-strategies.com, *Context maps* introduces the concept of creating style sheets to control knowledge-based information access and navigation. *Context maps* enable us to create virtual information maps for the information system. In a technical sense, *Context maps* describe what an information set is about, by formally declaring topics, and by linking the relevant parts of the information set to the appropriate topics.

Context tuple is a generic association of set members cast in roles. In the extended spreadsheet a column of roles and the related set members define context tuple. From the graphical view, context tuple, in fact, is represented by a compound edge and the connected compound nodes. A directed edge object consists of tail object, middle object and head object. Context can be defined by an aggregation of context tuples. While context tuples represents action-able system behaviors, processes, tasks, procedures or programs. The aggregated context tuples will form a *context map*.

3.2 *jMap* Technology

The *joined maps* or *jMaps* is a notation and method for representing systems architecture, structures, processes and reusable templates. This technology was first introduced by Dr. W.M. Jaworski [1995]. The technology was initially developed as a means of recovering and refining knowledge from legacy systems. By using the popular concept of spreadsheet structure, it is feasible to describe and process conceptual information. The *jMaps* notation allows efficient recovery and modeling of generic schemata for processes, objects and views of information systems.

jMaps represents the knowledge in a spreadsheet format with the relationships represented by vertical *tuples/columns*. Connecting the words jointed and map produced the term "*jMap*". The *jMaps* represent the relationship between different information nodes and provide functionality of arrays, graphs, relational tables, etc. The 'j' stands for *jointed*, because a *jMap* can be a collection of different information connected together in a strong logical way. By that we mean that you can manipulate the logical query to get the specific information that you seek from the map.

3.3 *jMap* Syntax and Process

The syntax of *jMaps* is based on the Relationship Oriented paradigm, or on relating sets and set members. In *jMaps* the relationships are represented by (vertical) tuples/columns. The *kTuple* (knowledge tuple) construct is the fundamental structure defined by the concepts and instances related by roles.

The relating mechanism is implemented by allocating roles to sets in schema and their instance to set members/instances in map. Compared to diagrams, maps are very compact, offering a rich context within limited space of a computer screen. Maps are created or edited within an organized electronic page – spreadsheet which assures efficient manipulation of relationships (columns) and heavy reuse of components (row).

Figure 2 (source from W.M.Jaworski) demonstrates associations of descriptor strings to arcs and nodes. The character "f" ("t") associate the strings to the "tail" ("head') of an arc. The character "m" signifies that the string is attached to the "body" of arc or node. Clustering of arcs - and connected nodes - into graphs is shown by tagging columns with character "&". Graphs are connected if they share at least one node. As is illustrated by graph (A) and (B), reordering of columns and/or rows is an information-preservation operation, i.e. the shape of the graph might change but not the meaning. Descriptors of arcs and nodes are set members. Sets are identified by {<set name>} and are defined by enumeration. The schema-level view of a map is obtained first by hiding set instances and then by hiding redundant columns (Figure 3). The schema provides information about *joined maps* (*jMaps*) structure and size.

A	**A**	**A**	**3**	**1**	**{Graph}**
&	&	&	3		{A}
M	**M**	**M**	**3**	**3**	**{Arc}**
m			1		string4
	m		1		string5
		m	1		string6
F	**F**	**F**	**3**	**3**	**{Node}**
	t	f	2		string1
t	f	t	3		string2
f		t	2		string3

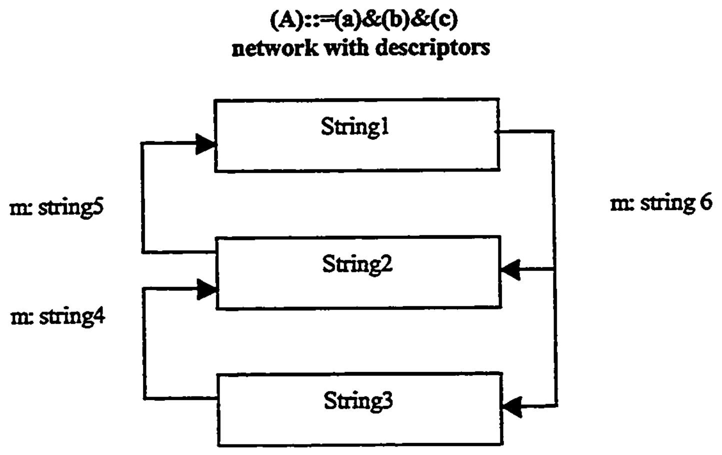


Figure 2 Diagrams Defined by Map Patterns

In Figure 2, for the diagrams on the right side, we can have the map as shown on the left side of figure. Map with Patterns contains three sets namely **{Graph}**, **{Arc}, and {Node}**. There are three roles namely **'A'**, **'M'** and **'F'** and four instance roles namely '&', 'm', 'f' and 't'. Role **'A'** was allocated to **{Graph}** to allow clustering of columns (i.e. relationships of instances) with instance role '&'. Role **'M'** was allocated to **{Arc}** to allow allocating of instance role 'm' to the instances 'string4', 'string5' and 'string6'. Role **'F'** was allocated to **{Node}** to allow allocating the instance roles 'f' and 't' to the members/instance of these sets.

A	A	A	3	1	{Graph}
&	&	&	3		{A}
M	M	M	3	3	{Arc}
F	F	F	3	3	{Node}

Figure 3 Schema view of Map with Pattern

If we need to develop large *jMaps* models, we can hide irrelevant columns and rows, editing visible cells and inserting new columns and new rows.

In general, abstract concepts appear on the right of the map in bold and between curly brackets. They can be thought of as a heading of a table column or a row. The instances would then be the actual contents listed in the table. Each column is to be read vertically using the syntax that was described above. For every "variable" you come across when reading down a column, you must read across towards the right of the map, to see which concept or instance the variable is referring to. Beside each instance, and under the total number of instances, represents the number of times the instance is referred to.

3.4 The *Joined Map* Notation

jMap notation can addresses many topics such as following:

- Information system architecture
- Recovery and reuse of system patterns
- Evolving information systems
- Software evaluation and renewal
- Systems workstations
- Automation of system design
- Modeling of web sites and knowledge hubs

Following are explaining for some *jMap* notations

- **The concepts could be one of the following:**

  A - Template Aggregation
  T – Template
  Y – Dominant
  Z – Descriptive
  K – Identifier
  O – Identity
  H – Hierarchy
  I - Generalization - "parent" or "heir"
  P - Aggregation - "whole" or "part"

U - Uses or used

D - Dependence

S - Sequence - position in a sequence

F - Flow "from" or "to"

L - Flow "from", "to" and "loop"

X - Unique Qualifier

M - Association

G - Guard or Goal

E - Event

V - Value

? - User defined

- **The different instances that exist for the concepts:**

1 ... * - identifier or value

o - column marker

h - tree root

1 ... * - branch

f - from:

t - to:

b - both

m - many or middle:

d - destination:

s - source:

l - loop

a - assertion

e - exception

x - unique row marker

v - related

c - composite

t - true

f - false

o - otherwise

t - implied true

f – implied false

e – enabled

d - disabled

? - user defined

u - update

3.5 *Associative Model* Recovered with *jMaps*

We will go back to earlier *Customer* and *Order* tables (Table 1 and Table 2) and convert *associative model* data to *jMaps*. Rewriting of the *associative model* with *jMaps* should be done by performing of the following activities:

1) Identify component types i.e. identification of sets by name.
2) Enumerate sets and identify connector types
3) Create connectivity columns/map by `connecting' components with characters "f" and "t".
4) Use "M" to identify association. Enhance connectivity columns by using characters "m" to represent association between the attributes
5) Use characters "v" to stress uniqueness/identity of an entity.
6) Use characters "F" to identify columnwise for the sets with members connected by "f" or "t".
7) Create schema view by first hiding set members and then hiding redundant columns.

Products of the process for this example, i.e. relevant *jMaps* and schema are shown in Figure 4 and Figure 5 .

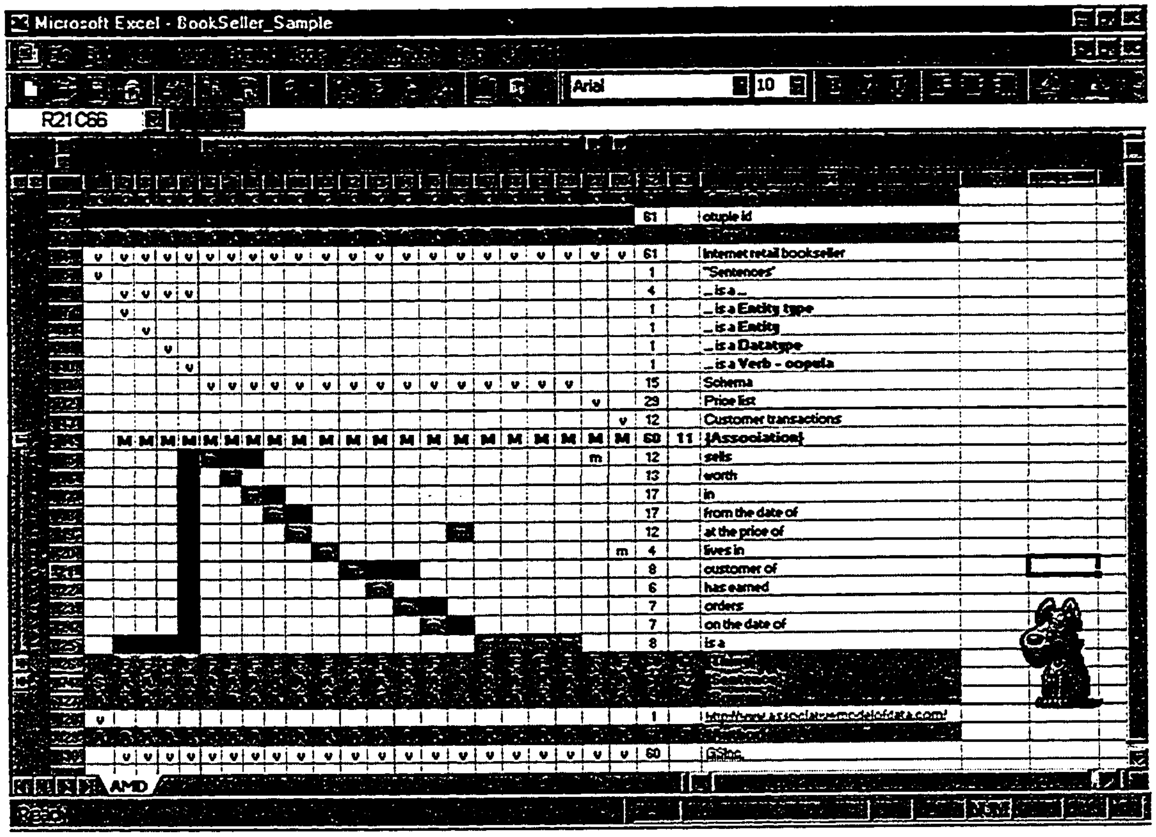
For more complex example as described in earlier Book Seller problem, Figure 6 shows recovered *associative model* with relevant *jMap* and schema.

A	A	A	A	A	A	A	A	8	4	{View}
v	v	v	v	v	v	v	v	8		ADM
A	A	A	A	A	A	A	A	8	2	{Entity}
v	v	v	v					4		Customers
				v	v	v	v	4		Orders
F	F	F	F	F	F	F	F	8	4	{Customer number}
F	F	F	F					4	4	{Name}
F	F	F	F					4	4	{Telephone no}
F	F	F	F					4	4	{Credit limit}
F	F	F	F					4	4	{O/S balance}
				F	F	F	F	4	4	{Order number}
				F	F	F	F	4	4	{Date}
				F	F	F	F	4	4	{Item}
				F	F	F	F	4	4	{Quantity}
M	M	M	M	M	M	M	M	8	1	{Association}

Figure 4 Schema of Customers and Order *Associative Model* represented by *jMaps*

			f		t	t		3		789
F	F	F	F					4	4	{Name}
t								1		Avis
	t							1		Boeing
		t						1		CA
			t					1		Dell
F	F	F	F					4	4	{Telephone no}
t								1		020 7123 4567
	t							1		020 8345 6789
		t						1		0123 45678
			t					1		0134 56789
F	F	F	F					4	4	{Credit limit}
t								1		£10,000
	t							1		£2,500
		t						1		£50,000
			t					1		£21,000
F	F	F	F					4	4	{O/S balance}
t								1		£4,567
	t							1		£1,098
		t						1		£14,567
			t					1		£6,789
				F	F	F	F	4	4	{Order number}
				f				1		11234
					f			1		11235
						f		1		11236
							f	1		11237
				F	F	F	F	4	4	{Date}
				t				1		02/03/1999
					t			1		15/03/99
						t		1		21/04/99
							t	1		07/05/1999
				F	F	F	F	4	4	{Item}
				t				1		ABC345
					t			1		GGI765
						t		1		KLM012
							t	1		GHJ999
				F	F	F	F	4	4	{Quantity}
				t				1		150
					t			1		25
						t		1		1000
							t	1		50
M	M	M	M	M	M	M	M	8	1	{Association}
m	m	m	m	m	m	m	m	8		has / is

Figure 5 Customer and Orders *Associative Model* represented by *jMap*


Microsoft Excel - BookSeller_Sample

R21C66

61		otuple id
61		Internet retail bookseller
1		"Sentences"
4		_is a _
1		_is a Entity type
1		_is a Entity
1		_is a Datatype
1		_is a Verb - copula
15		Schema
29		Price list
12		Customer transactions
60	11	{Association}
12		sells
13		worth
17		in
17		from the date of
12		at the price of
4		lives in
8		customer of
6		has earned
7		orders
7		on the date of
8		is a
1		http://www.associativemodelofdata.com/
60		GSInc.

Figure 6 Book Seller Problem with *jMap* converted *Associative Model*

Chapter 4

4. Application Program

4.1 Introduction

This project deals with the recovery of the information structure knowledge from database, to generate the *jMap* (in the background) and then launches *MS Excel* with the resultant map. The program runs only on computers equipped with *MS Excel.* The subsequent reuse of this recovered knowledge can be represented as *associative data model.* The central element in the process of information manipulation is based on the *jMap* formal notation technology.

The program will provide the user with a number of options including the options to recover information from the database, and the options to normalize *jMap* sheet to save into the database. A comprehensive on-line help about what each of these options mean will be provided. For users seeking more detailed sample will also be provided.

4.2 Development Tool

The development tool is described as the follows: (most are members of the Microsoft family of products)

- **Microsoft Excel:**

Excel is a spreadsheet that allow you to organize data, complete calculations, make decisions, graph data, develop professional-looking reports, convert Excel files for use on the database, access the database.

The three major parts of Excel are:

1) Worksheets, that allow you to better calculate, manipulate, and analyze data such as numbers and text (the term worksheet means the same as spreadsheet.).
2) Charts, that pictorially represent data. Excel can draw a variety of two-dimensional and three-dimensional charts.
3) Databases, that manage data. For example, once you enter that data, you can search for specific data, and select data that meets the criteria.

- **Microsoft Access**

Microsoft Access is a database which makes difficult database technology accessible to general business users. *Microsoft Access* ensures that the benefits of using a database can be quickly realized. With its integrated technologies, *Microsoft Access* is designed to make it easy for all users to find answers, share timely information, and build faster solutions.

Microsoft Access has a powerful database engine and a robust programming language, making it suitable for many types of complex database applications. For small project, to chose *Microsoft Access* is suitable to store the data information.

- **VBA**

  A Visual Basic Application can provide us with the means to accomplish a wide range of the programmatic results. With *VBA*, we can create full-fledged custom applications in *Microsoft Excel.*

  Visual Basic support a set of objects that correspond directly to elements in *Microsoft Excel.* Every element in *Microsoft Excel*, such as workbook, worksheet, chart, cell, and so on, can be represented by an object in *Visual Basic.* By creating procedures that control these objects we can automate tasks in *Microsoft Excel.*

4.3 Project Functions

One of the main features needed for this project is the seamless nature of its operation. This entails minimum work by the system's user. The main system's functions as seen by the user can be summarized as:

1) Providing a mechanism through which the user can handle operation
2) Providing a mechanism for the user to enter his/her selected options,
3) Providing context-sensitive help,
4) Providing a visual indicator for the user to know the process' progress, and
5) Seeing the resultant *jMap* in *MS Excel.*

The system has more functions that are done in the background. These include:

1) Create the Unique Ids for each Context Tuple (aka Column ID) and Context Item (aka Set X Member ID), then introduce Unique IDs for each of Sets, Members, Spreadsheets, Workbooks, Tuples, Chapters, DBs etc.
2) The identifier is a surrogate, that is automatically assigned by the system.
3) Enable new item data be grouped into two tables: cTuple and cItem, and then could be saved into the database.
4) For obtained information from database, extracting and then refining the those data needed for the *jMap* generation.
5) Generating the associated data model *jMap* with the needed features, and launching *MS Excel* with the resultant *jMap*.
6) For both directions: database convert to *jMap* or *jMap* to Database, it will be taken care about the larger data with constrain of few spreadsheets and few workbooks
7) Query database and display results as *jMaps*.

4.4 General Constraints

The software is constrained only to run MS Windows operating system (WIN NT or WIN95/98/2000). The user also needs to know basic operations of *MS Excel*.

Chapter 5

5. Program User Manual

This manual concerns the extraction of an associative data model information and conversion of the selected information to and represented in *jMap* notation. The *jMap* is based on the *Excel* spreadsheets. Therefore, it is necessary for users to have elementary *Excel* knowledge.

The syntax, schema, maps, and styles of *jMaps* are protected by copyright and trade secret law and may not be disclosed, used or produced in any manner, or for any purpose, except with written permission from Dr. W. M. Jaworski.

5-1 System Requirement

Before you try to run this program, you need check if your system meets following requirements:

Hardware:

Pogrom shall operate with the following hardware requirements:

- CPU 486 or later
- Monitor – SVGA (800x600) or latter
- RAM – 16 MB
- Mouse or equivalent pointing device

Software:

You have to set up the following software in your machine

- Microsoft Excel
- Microsoft Access
- Visual Basic

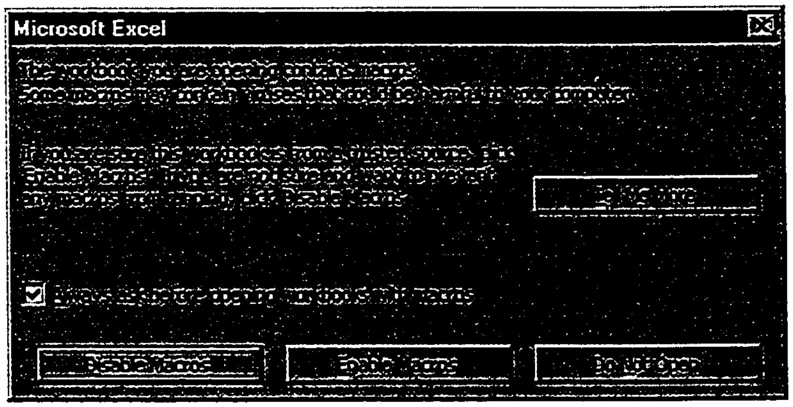
Platform:

The program can run on the following platforms

- Windows 95
- Windows 98SE
- Windows NT
- Windows 2000

5-2 Start Program

In the software package, soon after open **ADMjMap.xls** file, there will be a Microsoft Excel popup dialogue as shown in following:



Microsoft Excel

The workbook you are opening contains macros.
Some macros may contain viruses that could be harmful to your computer.

If you are sure this workbook is from a trusted source, click
'Enable Macros'. If you are not sure and want to prevent
any macros from running, click 'Disable Macros'.

☑ Always ask before opening workbooks with macros

Figure 7 MS Excel Macros Enable Dialogue Interface

Click **Enable Macros** button to open the file, if you select **Disable Macros** button, then you will be unable to run the Macros in the program. After **Enable Macros** button is clicked, it will show following Welcome Interface:

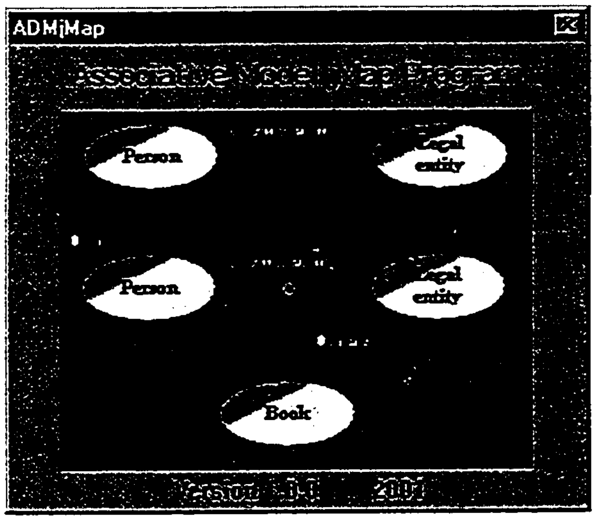


Figure 8 The Welcome Interface of ADMjMap Software

By clicking any area of welcome interface, you will hear one beep sound, after that the ADMjMap Excel file is ready to use.

5-3 Program Functionality

After ADMjMap is opened and is ready to use, you will find there is a *jMap* test sheet in the book. This test sheet is just for user to test the program's functionality. In Figure 9, you will find that a menu bar named ADMjMap has been created. When open this menu, as we can see, there are six operation sub-menus:

- **Create Tables**——to create the new sheet with generated ID for Sets, Members, Spreadsheets, and Workbooks.
- **Remove Tables** ——to remove the created sheets of {cItem}, {cTuples} and all sheet name which have brace {} covered will be removed
- **jMap->DB** ——to save the created tables to Access Database
- **DB->jMap** ——recover tables information from the Access Database with formatted *jMap* notation to the new sheet
- **DB jMap Analysis** ——analyze data tables from Database to produce the *jMap* results
- **Help** ——to get help information for using this program

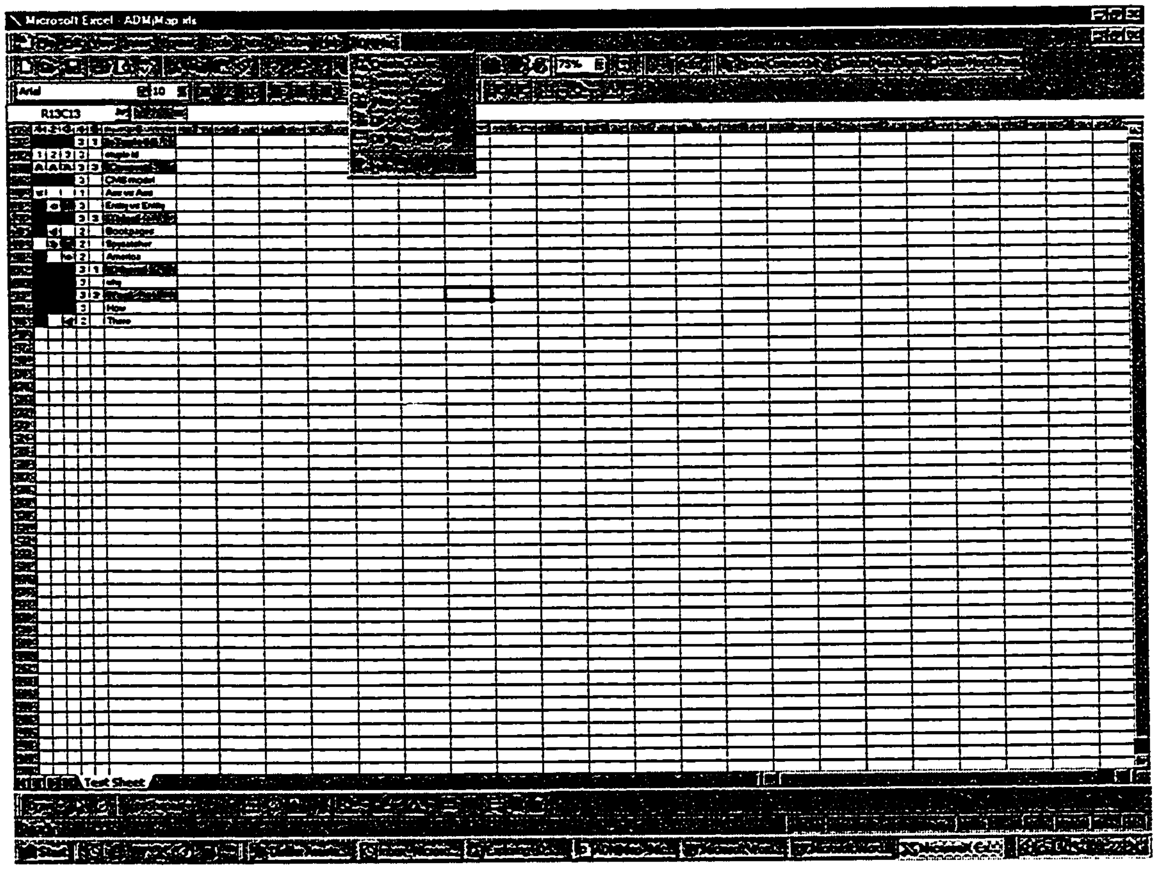


Figure 9 ADMjMap Menus and Test Sheet

5-4 Create Normalized *jMap* Tables

On the top menu "**ADMjMap**", by clicking "**Create Tables**".

- You will be asked to select sheet ID and book ID from a given Combo Box interface.

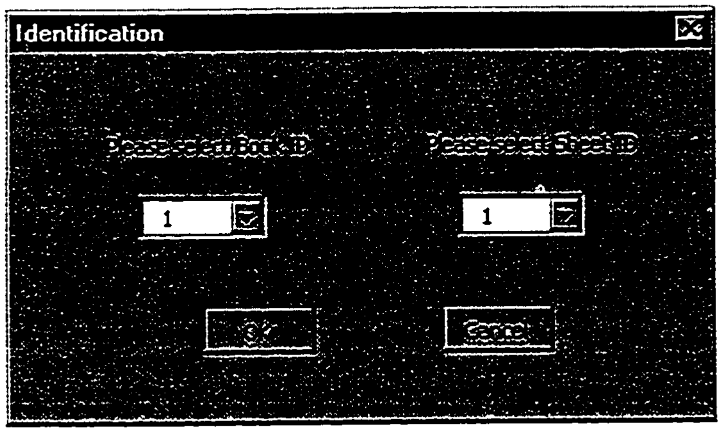
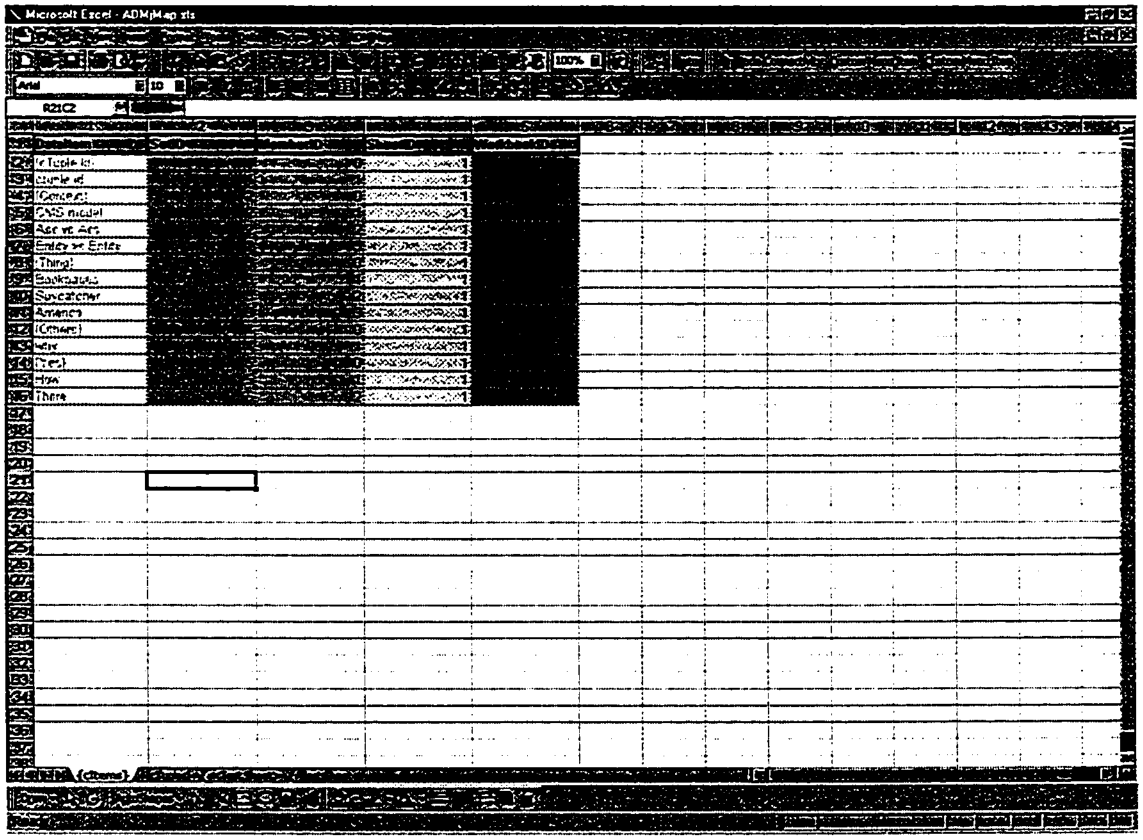


Figure 10 Sheets and Book Identify Dialogue

- Based on Normalized information from original active sheet, the program will create two tables which present as cItem and cTuple properties.
- It will create a new sheet name as: "< + "Original Sheet Name" + > ". This new sheet will present generated ID for Sets, Members, spreadsheet and workbook from the original sheet. The two tables will be in two new created sheet named as: {cItem} and {cTuples.
- If two sheet tables already exist, the created sheet name will be changed to {cItem}1 and {cTuples}1, or {cItem}2 and {cTuples}2, and so on. As the same, new sheet name for generated ID for Sets, Members, spreadsheet and workbook, if it exists, its name will also be updated with increasing number.

Figure 11 shows the created new cItem sheet based on generated ID for Sets, Members, spreadsheet and workbook from original sheet. Figure 12 also shows the results of new cTuple sheet normalized from original sheet.



[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
(cTuple Id)	[illegible]	[illegible]	[illegible]	[illegible]
ctuple id	[illegible]	[illegible]	[illegible]	[illegible]
(Context)	[illegible]	[illegible]	[illegible]	[illegible]
CMS model	[illegible]	[illegible]	[illegible]	[illegible]
Asc vs Arts	[illegible]	[illegible]	[illegible]	[illegible]
Entity vs Entity	[illegible]	[illegible]	[illegible]	[illegible]
(Thing)	[illegible]	[illegible]	[illegible]	[illegible]
Bookstores	[illegible]	[illegible]	[illegible]	[illegible]
Suncatcher	[illegible]	[illegible]	[illegible]	[illegible]
America	[illegible]	[illegible]	[illegible]	[illegible]
(Others)	[illegible]	[illegible]	[illegible]	[illegible]
Who	[illegible]	[illegible]	[illegible]	[illegible]
(Yes)	[illegible]	[illegible]	[illegible]	[illegible]
How	[illegible]	[illegible]	[illegible]	[illegible]
There	[illegible]	[illegible]	[illegible]	[illegible]

Figure 11 Created cItems Table Sheet

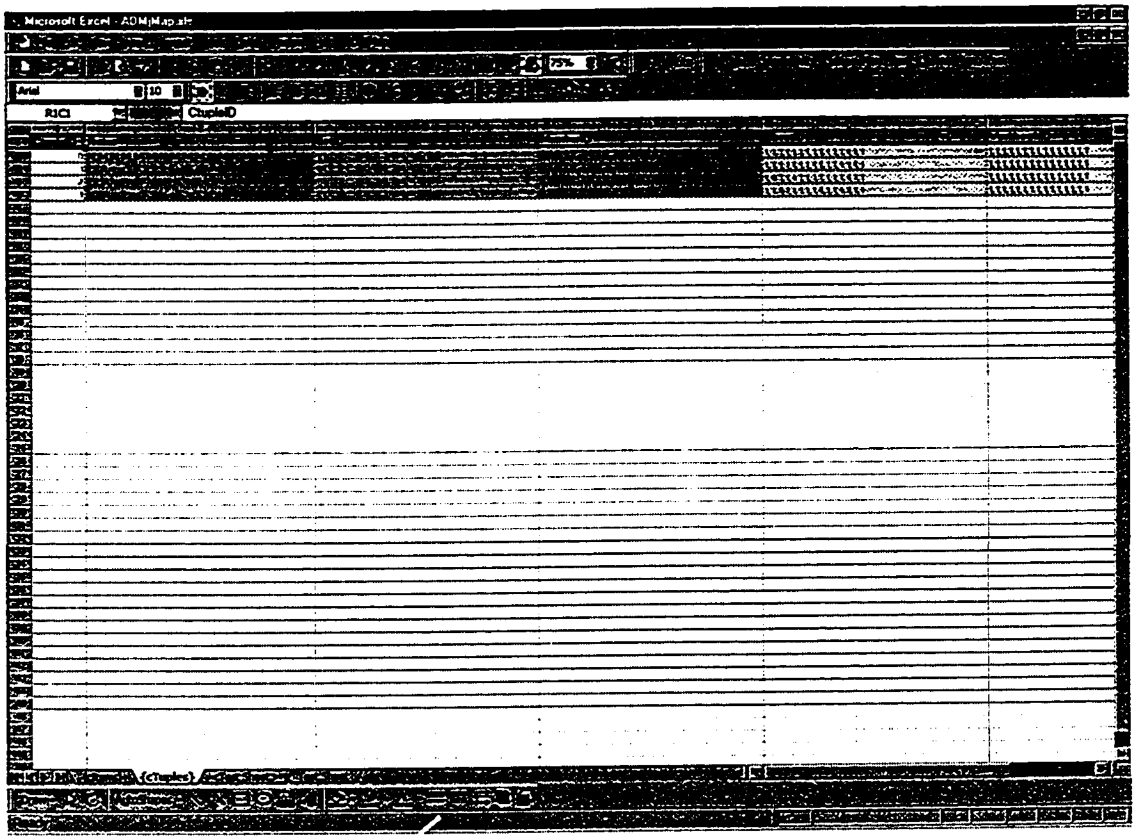


Figure 12 Created cTuples Table Sheet

Figure 13 shows normalized new sheet for original sheet in which the new sheet has been generated ID for Sets, Members, spreadsheet and workbook. This sheet information will be ready for creating the cItems and cTuple tables.

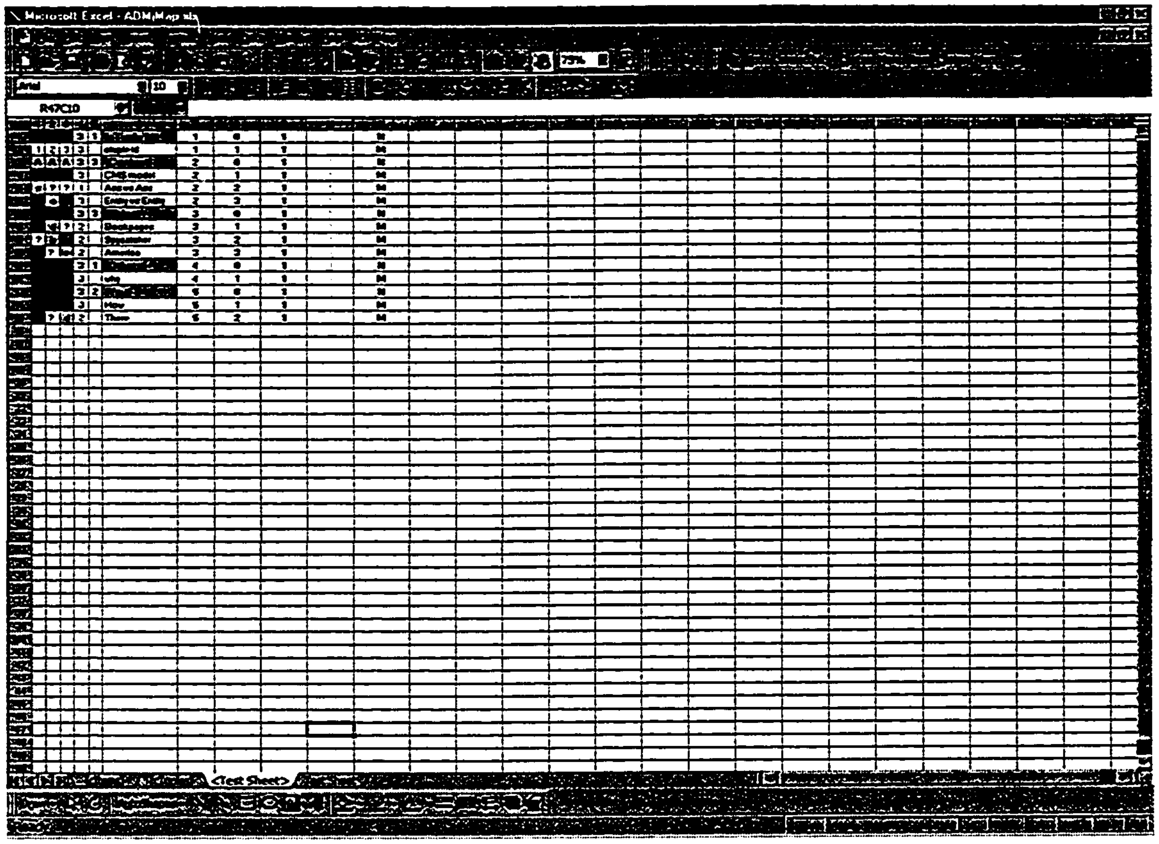


Figure 13 Normalized New Sheet

5-5 Remove Tables

On the top menu "**ADMjMap**", by clicking "**Remove Tables**", the created sheets of {cItem}, {cTuples} and all sheet names with {}or <> covered will be removed.

5-6 Save *jMap* to Database

On the top menu "**ADMjMap**", by clicking "**jMap->DB**", it will display a dialog box allowing the user to save the created tables to *Access Database.*

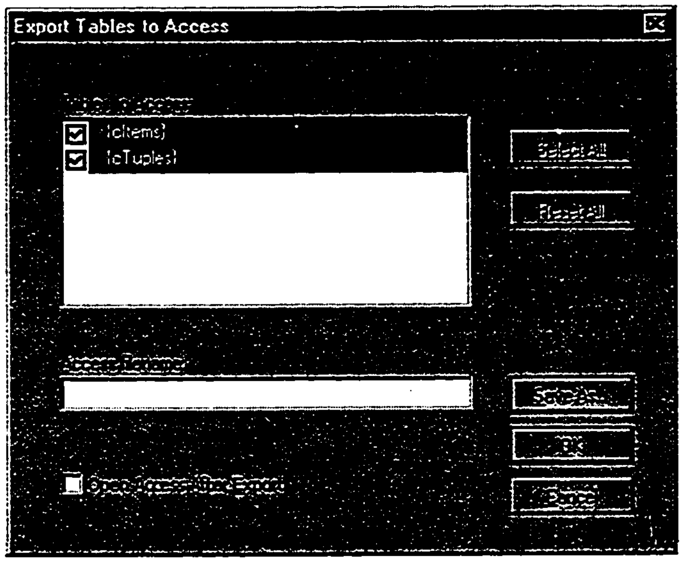


Figure 14 Export Tables to Access Dialog Box

In Figure 14, by clicking **Save As** button, the program will show following dialog box with default file name, if select Save button, the cItem Table and cTuple Table will be saved to *Access Database*. User can change the file name. If file name already exists, the tables information will be still added into this database in a changed table name as {cItem}1 and {cTuples}1, or {cItem}2 and {cTuples}2, and so on.

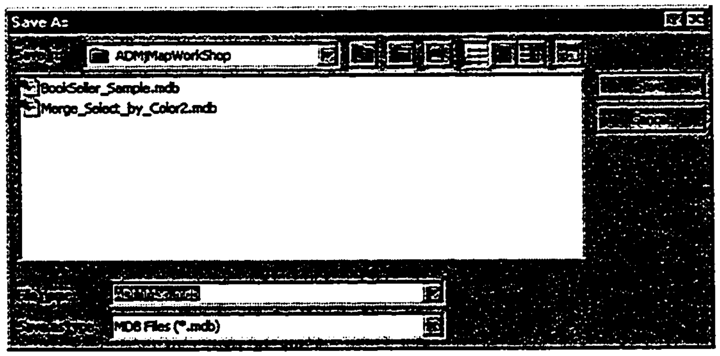


Figure 15 Save As Dialog Box

After the database file has been saved, the following message box will inform the user that the file has been saved.

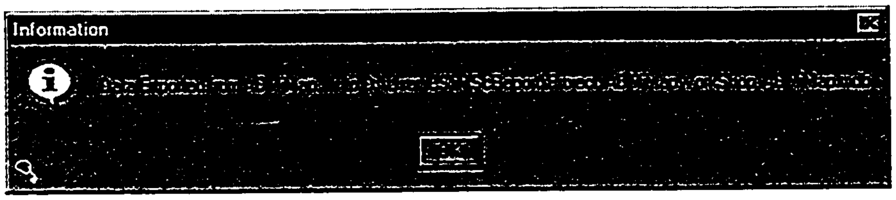


Figure 16 Information Message Box for Data Export from *jMap*

If check box **"Open Access after Export"** has been checked in the Export Tables to Access Dialog Box (see Figure 14), after Database File has been saved, computer system will automatically open the *Microsoft Access* for user to review the saved information.

5-7 Recover Database to *jMap*

On the top menu **"ADMjMap"**, by clicking **"DB->jMap"**, it will display a dialog box allowing the user to customize for recovering Database to *jMap*.

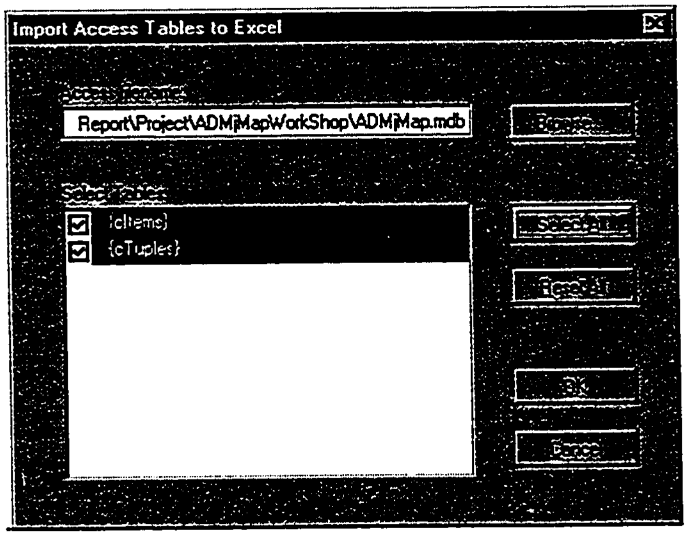
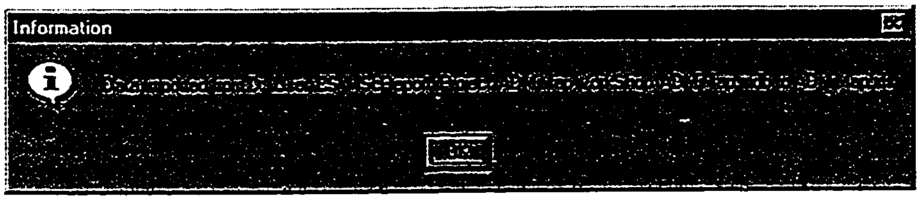

Import Access Tables to Excel
Report\Project\ADMjMapWorkShop\ADMjMap.mdb
{cItems}
{cTuples}

Figure 17 Import Tables to Excel Dialog Box

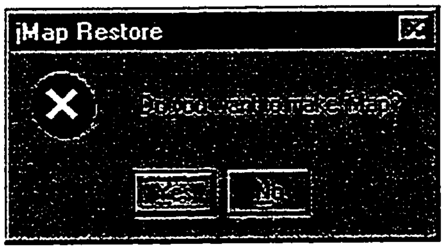
In Figure 17, after by clicking **Select All** button or check selected Table, with clicking **OK** button, the program will load the cItem Table and cTuple Table data to Excel importing in different sheets. When it has been finished import Tables to the sheet, the following message box will display the file from the path has been import to Excel



[illegible]

Figure 18 Information Message Box for Data Import from Access

At end, the program will display a pop-up message box which will ask user if user wants to convert table information to *jMap*. Click Yes button, it will restore the *jMap* into the sheet.



Do you want to make jMap?

Figure 19 *jMap* Restoring Message Box

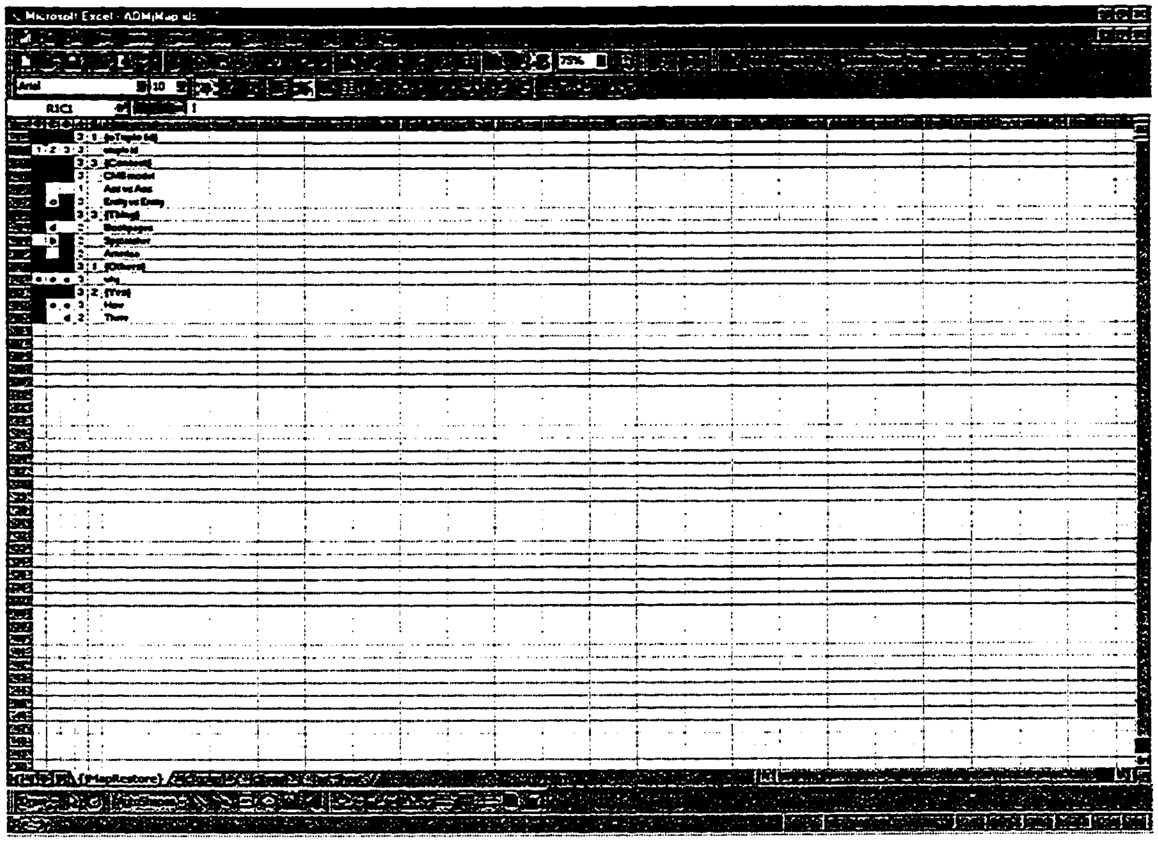


Figure 20 Restored *jMap* Results

5-8 Analyze Database Property with *jMap*

On the top menu "**ADMjMap**", by clicking "**DB jMap Analysis**", computer system will display a dialog box allowing the user to select a Table for analysis.

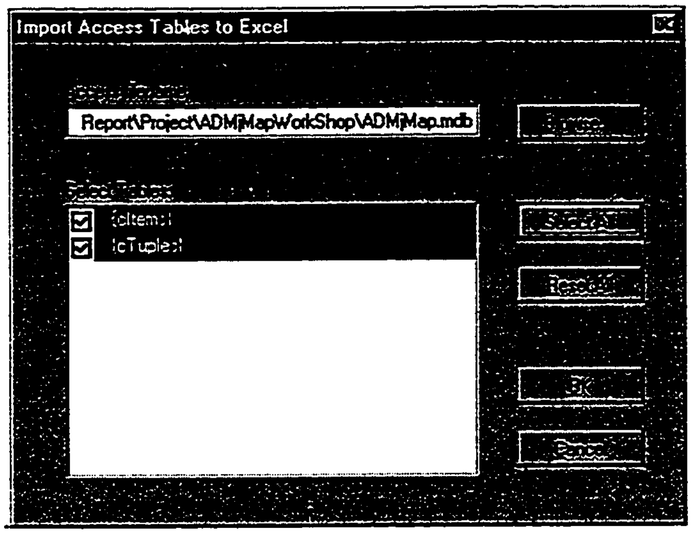
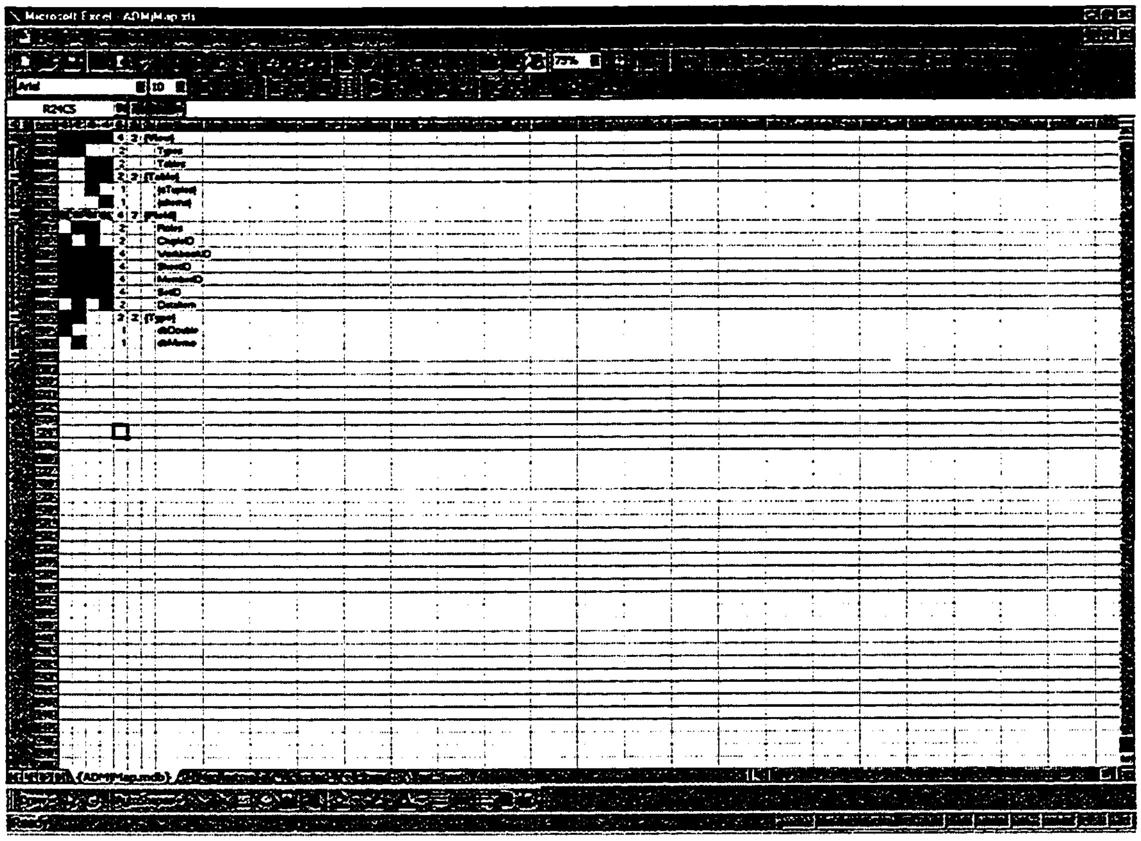

Import Access Tables to Excel
Report\Project\ADMjMapWorkShop\ADMjMap.mdb
{cItem:}
{cTuples:}
OK

Figure 21 Import Tables to Excel Dialog Box for Analysis Table Properties

In Figure 21, after by clicking Select All button or check selected Table, with clicking OK button, it will analysis the saved tables information from the Access Database to produce the *jMap* results

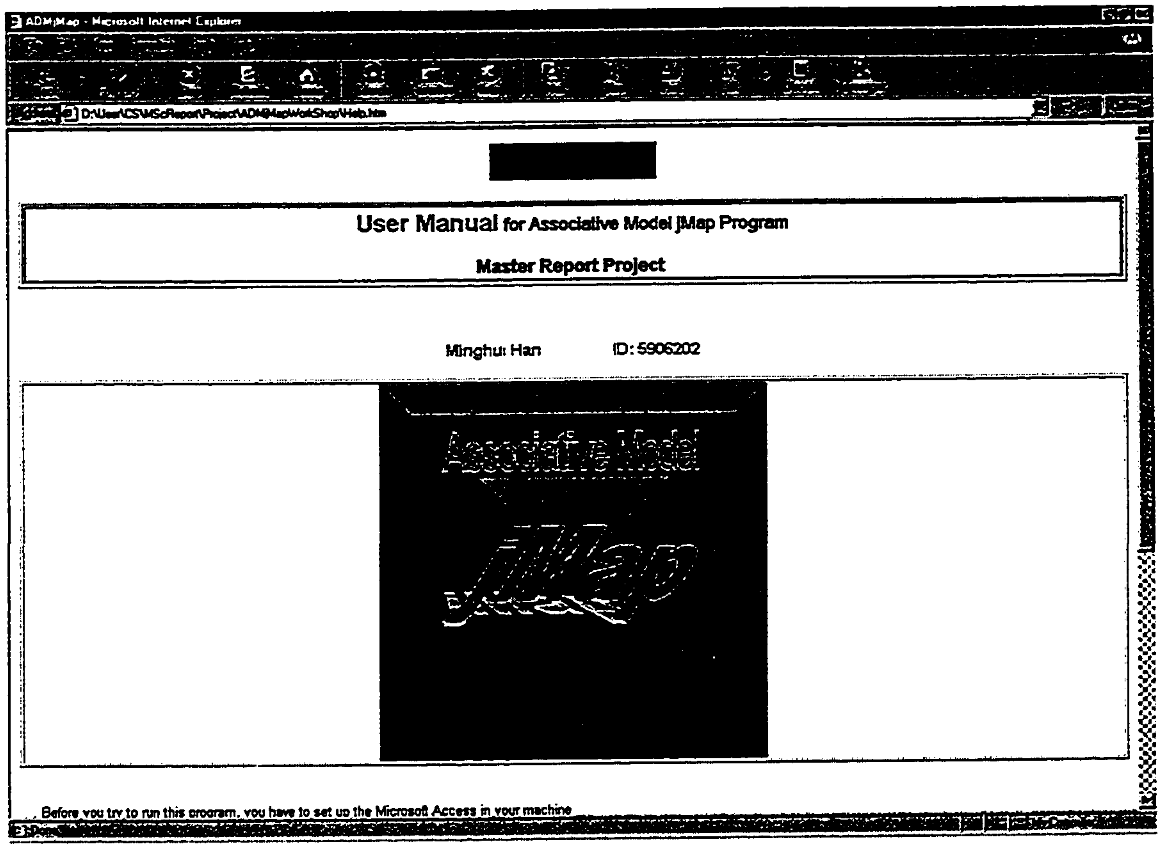


```
4 2 [illegible]
2       Types
2       Tables
2 2 [Tables]
1       [illegible]
1       [illegible]
4 7 [Fields]
2       [illegible]
2       ChapterID
4       [illegible]
4       SheetID
4       MemberID
4       SetID
2       [illegible]
2 2 [Types]
1       dbDouble
1       dbMemo
```

Figure 22 The Tables Analysis Results with *jMap* Notation

5-9 Get Help

On the top menu "**ADMjMap**", by clicking "**Help**", the program will open a help HTML web page for user to get help. Figure 20 shows this help page.



ADMjMap - Microsoft Internet Explorer

D:\User\CS\MScReport\Project\ADMjMapWorkShop\Help.htm

User Manual for Associative Model jMap Program

Master Report Project

Minghui Han ID: 5906202

Before you try to run this program, you have to set up the Microsoft Access in your machine

Figure 23 Program Help Page

Chapter 6

6. Conclusion And Recommendation

6.1 General Conclusion

The following conclusions are drawn from the results of this study:

1) The *associative model* views the information in the same way as the human brain, i.e. treats the things with association between them. Those associations can be expressed through the simple subject-verb-object syntax of an English sentence.
2) The *associative model* is simple. It overcomes the limitations of the *relational model* and avoids the complexities of the object model by structuring information in a more accessible and intuitive manner than either of the other model.
3) *Context maps* enable us to create virtual information maps for the information system. *Joined maps - jMaps* are a notation and method for representing systems architecture, structures, processes and reusable templates. The *jMaps* notation allows easy recovery and modeling of generic schemata for processes, objects and views of information systems.
4) *jMaps* syntax is simple and robust. *jMaps* models are pattern rich, allow users to specify, query and control the model views. Different views are generated algorithmically to be useful for compilers or end users
5) The *associative data* model can be presented as the *joined maps* (*jMaps*) of concepts and relationships using the popularly available *MS Excel* spreadsheet.

6) An application program was developed by considering *context maps* for associative data model. This program can present *context maps* exported into database or recovery data from a database to spreadsheets with *jMaps* notation which represented as the associative data model.

7) The application program can also treat any standard *jMap* sheet to convert *jMap* into a database system.

6.2 Recommendations for Future Works

From the results of this study, it is noted that there are still more detail works need to be carried out for improving use the application program. The following are recommended for future enhancement.

1) There is much future work in implementation of *joined maps* for dealing with complex systems. Future work is expected to lead to a better and more complete theory of *Context Maps*.
2) A more complete application program to convert *jMaps* into Database, or from Database to *jMaps*, needs to be developed.
3) In developed application program, to query different tables and data types from *Associative Model* database is necessary for future work.
4) For a larger data *jMap* sheet, it really takes time to get results in running the current program on a PC. It is necessary to improve program-running speed.
5) Designing of more user-friendly interface is yet another work needs to be done.

6) For large amounts of data, using *Excel* as a repository of *jMaps* has its limitations. Only 256 columns are available in the *Excel*. Although to some extent this project has considered this issue, to develop more efficient method for storing "context tuples" is necessary.

Bibliography

A- Printed Materials

1) **Grady Booch, James Rumbaugh, Ivar Jacobson,** "The UML User's Guide", Addison Wesley, 1998.

2) **Derek Coleman, etc.,** "Object-Oriented Development: the Fusion Method", Prentice-Hall, Inc., 1994.

3) **Minghui Han, etc.,** "jMapper, Web-Page *jMap* Generator For Key words and Keyphrase", Concordia University, COMP657, 2000.

4) **Minghui Han,** "jMapper, Web-Page *jMap* Generator For Key words, Keyphrase and XML tree view, Version 2.0", Concordia University, COMP695, 2000.

5) **W.M. Jaworski,** "Comp 457/657 Course Notes", Concordia University, 2000.

6) **W.M. Jaworski,** "*jMaps*: Conceptual Spreadsheets for Data and Knowledge", Warehousing, 1995.

7) **W.M. Jaworski,** "System Analysis and Design in the Classroom: InfoMAPs Teaching Factory", *Modeling and Simulation Conference*, Pittsburgh, Pa.,May 3-4, 1990.

8) **W.M. Jaworski,** "Michailidis A. A., Recovery and Enhancement of System Patterns: InfoSchemata and InfoMaps", *NATW94*, University of Massachusetts - Lowell, Massachusetts, June 1994.

9) **W.M. Jaworski,** "Conceptual Spreadsheets for Data and Knowledge Warehousing", *ATW95 - USA 1995*, University of New Hampshire, Durham, New Hampshire, May 31 - June 1, 1995.

10) **W.M. Jaworski**, "Cooperative Engineering Issues by Examples: Mapping of Mil498 and NSDIR with *jMaps*", *ATW96-USA 1996*, Electronic Systems Center, Hanscom Air Force Base, August 6-9, 1996.

11) **W.M. Jaworski**, "Representing Processes, Schemata and Templates with *jMaps*", *Expanded version of the paper presented at Conference on Notational Engineering (a.k.a. NOTATE96)*, The George Washington University, Washington, DC., May 23-25, 1996.

12) **W.M. Jaworski, Michailidis A. A.**, "Recovery and Enhancement of System Patterns: InfoSchemata and InfoMaps", *ATW '94*, University of Massachusetts - Lowell, Lowell, Massachusetts, June 1994.

13) **W.M. Jaworski**, "InfoMaps: Conceptual Spreadsheets for Data and Knowledge Warehousing", *ATW '95*, University of New Hampshire, Durham, New Hampshire, June 1995.

14) **W.M. Jaworski**, et al. "The ABL/W4 methodology for system modeling", *System Research Journal 4(1)*, 23-37, 1987.

15) **W.M. Jaworski**, et al. "Representing processes, schemata and templates with *jMaps*", *Semiotica* 125(1/3), 229-47, 1999.

16) **W.M. Jaworski**, "Representing System Schemata and Templates with *jMaps*", *NOTATE'96*, George Washington University, Washington, D.C., May 23-25, 1996.

17) **James Rumbaugh, etc.**, "Object-Orinted Modeling and Design, Prentice-Hall, Inc.", 1991.

18) **Ian Sommerville**, "Software Engineering", Addison-Wesley, 5th edition, 1995.

19) **Simon Williams**, "The *Associative Model* of Data", Lazy Software, 2000.

B- On-Line Sources

1) **General Strategies Inc.** http://www.gen-strategies.com

2) **Lazy Software,** http://www.lazysoft.com

3) **Lazy Software,** http://www.associativemodelofdata.com/

4) **Concordia University,** Thesis preparation and thesis examination regulations, http://www-gradstudies.concordia.ca/SGS_WWW/publications.html

5) **Rob Kremer,** A Concept Map Meta-Language, http://www.cpsc.ucalgary.ca/~kremer/dissertation/index.html

6) **Joseph D. Novak,** The Theory Underlying Concept Maps and How To Construct Them, http://cmap.coginst.uwf.edu/info/printer.html

Appendix Source Code

The program was coded by using VB language, the project consists of three parts: user forms, modules and class modules

- A user form contains user interface controls, such as command buttons and text boxes
- A module is a set of declarations followed by procedures—a list of instructions that a program performs.
- A class module defines an object, its properties, and its methods. A class module acts as a template from which an instance of an object is created at run time.

A-1 User Form Source Code

The source code for User Form includes as following created forms:

- **frmBookSheetInfo**
- **frmExportTablesToAccess**
- **frmImportAccessToWks**
- **frmWelcome**

All source code in above forms are listed as following:

A-1-1 frmBookSheetInfo

```
Option Explicit

Private Sub CancelButton_Click()
    On Error Resume Next

    Unload Me
End Sub


Private Sub OKButton_Click()

    Dim varBookID As String
    Dim varSheetID As String
    varBookID = ComboBox_Book.value
    varSheetID = ComboBox_Sheet.value
    Call MTables.CreateID(varSheetID, varBookID)
    Call MTables.CreateTables

    If MStartup.bjMaptoAccess = True Then

       frmExportTablesToAccess.Show

    End If

    Unload Me

End Sub

Private Sub UserForm_Initialize()
   Dim varCounter       ' Declare variables.

   For varCounter = 1 To 100   ' Count from 1 to 100.
      ComboBox_Book.AddItem varCounter   ' Add the Counter number for Book.
      ComboBox_Sheet.AddItem varCounter  ' Add the Counter number for Sheet
   Next varCounter

End Sub
```

A-1-2 frmExportTablesToAccess

```
'
'Purpose:   this form allows the user to select the worksheets from the active
'           workbook to export to access
'
Option Explicit

Private colSheets As Collection
Private blnOpenADBM As Boolean
Private blnSaveAsClicked As Boolean

Public Property Get SaveAsClicked() As Boolean
    SaveAsClicked = blnSaveAsClicked
End Property

Public Property Get OpenADBM() As Boolean
    OpenADBM = blnOpenADBM
End Property

Public Property Get SelectedSheets() As Collection
    Set SelectedSheets = colSheets
End Property

Private Sub EnableOKAsNecessary()
Dim lngItemCurr As Long

    cmdSaveAs.Enabled = False
    cmdOK.Enabled = False
    With lstTables
        For lngItemCurr = 0 To .ListCount - 1
            If .Selected(lngItemCurr) Then
                cmdSaveAs.Enabled = True
                cmdOK.Enabled = True
            End If
        Next lngItemCurr
    End With

End Sub

Private Sub chkOpenADBM_Click()
    If chkOpenADBM.value = -1 Then
        blnOpenADBM = True
    Else
        blnOpenADBM = False
    End If
End Sub

Private Sub cmdCancel_Click()
    On Error Resume Next

    Set colSheets = Nothing

    MTables.RemoveTables

    Unload Me


End Sub

Private Sub cmdResetAll_Click()
    On Error Resume Next

    ChangeSelection (False)

End Sub

Private Sub cmdOK_Click()
Dim lngItemCurr As Long

    If txtADBMName = "" Then
        MsgBox "Access Filename(*.mdb) must be entered", vbExclamation, "Error"
        Exit Sub
    End If

    If UCase(Right(txtADBMName, 4)) <> ".MDB" Then
        txtADBMName = txtADBMName + ".mdb"
    End If

    Set colSheets = New Collection

    With lstTables
        For lngItemCurr = 0 To .ListCount - 1
            If .Selected(lngItemCurr) Then
                colSheets.Add .List(lngItemCurr)
            End If
        Next lngItemCurr
    End With

    MExportTablesToAccess.Export txtADBMName
    MTables.RemoveTables

    Set colSheets = Nothing
    Unload Me

End Sub

Private Sub ChangeSelection(ByVal Selected As Boolean)
    Dim lngItemCurr As Long

    On Error Resume Next

    With lstTables
        For lngItemCurr = 0 To .ListCount - 1
            .Selected(lngItemCurr) = Selected
        Next lngItemCurr
    End With

End Sub

Private Sub cmdSaveAs_Click()
    'Defines the variable as a variant data type
    Dim X As Variant

    'Opens the dialog
    X = Application.GetSaveAsFilename(, "MDB Files (*.mdb), *.mdb", 2, "Save As")
    If X <> False Then
        txtADBMName.Text = X
        blnSaveAsClicked = True
    End If

    txtADBMName.SetFocus

End Sub

Private Sub cmdSelectAll_Click()
    On Error Resume Next

    ChangeSelection (True)

End Sub

Private Sub lstTables_Change()
    EnableOKAsNecessary

End Sub

Private Sub lstTables_Click()
    EnableOKAsNecessary

End Sub

Private Sub UserForm_Initialize()
Dim Wks As Worksheet

    chkOpenADBM.value = 0
    blnOpenADBM = False
    cmdSaveAs.Enabled = False
    blnSaveAsClicked = False
    cmdOK.Enabled = False
    txtADBMName.Text = ""
    lstTables.Clear

    For Each Wks In Worksheets
        If Wks.type = xlWorksheet Then
            If Wks.Visible Then
                If InStr(Wks.Name, "{") Then
                    lstTables.AddItem (Wks.Name)
                End If
            End If
        End If
    Next Wks

    Dim lngItemCurr As Long

    On Error Resume Next

    With lstTables
        For lngItemCurr = 0 To .ListCount - 1
            .Selected(lngItemCurr) = True
        Next lngItemCurr
    End With

End Sub
```

A-1-3 frmImportAccessToWks

```
'
'Purpose:   this form allows the user to specify an access database and choose
'           which tables to import from access

Option Explicit

Private colTables As Collection
Private blnBrowseClicked As Boolean

Public Property Get BrowseClicked() As Boolean
    BrowseClicked = blnBrowseClicked
End Property

Public Property Get SelectedTables() As Collection
    Set SelectedTables = colTables
End Property

Private Sub EnableOKAsNecessary()
Dim lngItemCurr As Long

    cmdOK.Enabled = False
    With lstTables
        For lngItemCurr = 0 To .ListCount - 1
            If .Selected(lngItemCurr) Then
                cmdOK.Enabled = True
            End If
        Next lngItemCurr
    End With

End Sub

Private Sub cmdBrowse_Click()
    'Defines the variable as a variant data type
    Dim X As Variant

    blnBrowseClicked = False

    'Opens the dialog
    X = Application.GetOpenFilename("MDB Files (*.mdb), *.mdb", 2, "Open", ,
False)

    If X <> False Then
        blnBrowseClicked = True
        txtADBMName.Text = X
        ListTables
    End If

    txtADBMName.SetFocus

End Sub

Private Sub cmdCancel_Click()
    On Error Resume Next

    Set colTables = Nothing
    Unload Me

End Sub

Private Sub cmdResetAll_Click()
    On Error Resume Next

    ChangeSelection (False)

End Sub

Private Sub ListTables()
Dim wrkdefault As Workspace
Dim db As Database
Dim tblList As TableDef
Dim Message As String
Dim Title As String

    On Error GoTo Handler

    ' Get default Workspace.
    Set wrkdefault = DBEngine.Workspaces(0)
    ' Open database
    If blnBrowseClicked = True Then
        Set db = wrkdefault.OpenDatabase(txtADBMName)
    Else
        Set db = wrkdefault.OpenDatabase(ActiveWorkbook.Path & "\" & txtADBMName)
    End If

    lstTables.Clear

    ' Fetch all the tables
    For Each tblList In db.TableDefs
        If Left(tblList.Name, 4) <> "MSys" Then
            lstTables.AddItem (tblList.Name)
        End If
    Next

    Set db = Nothing

    lstTables.Enabled = True
    cmdSelectAll.Enabled = True
    cmdResetAll.Enabled = True

    Exit Sub

Handler:
    Message = _
                "Error Number      : " & Err _
        & Chr(10) & "Error Description: " & Error()

    Title = "An error has occured"
    MsgBox Message, , Title
    Message = ""
    Title = ""
    cmdResetAll_Click
    lstTables.Clear

End Sub

Private Sub cmdOK_Click()
    Dim lngItemCurr As Long

    'will add the question Box
    Dim Msg, Style, Title, Help, Ctxt, Response, MyString
    Msg = "  Do you want to make jMap?  "     ' Define message.
    Style = vbYesNo + vbCritical + vbDefaultButton1    ' Define buttons.
    Title = "jMap Restore"     ' Define title.
    Help = "DEMO.HLP"
    Ctxt = 1000     ' Define topic
        ' context.
        ' Display message.

    If txtADBMName = "" Then
        MsgBox "Access Filename must be entered", vbExclamation, "Error"
        Exit Sub
    End If

    If UCase(Right(txtADBMName, 4)) <> ".MDB" Then
        txtADBMName = txtADBMName + ".mdb"
    End If

    Set colTables = New Collection

    With lstTables
        For lngItemCurr = 0 To .ListCount - 1
            If .Selected(lngItemCurr) Then
                colTables.Add .List(lngItemCurr)
            End If
        Next lngItemCurr
    End With

    If MStartup.bAccesstojMap = True Then
```

```
        MImportAccessToWks.Import txtADBMName

        Response = MsgBox(Msg, Style, Title, Help, Ctxt)
        If Response = vbYes Then    ' User chose Yes.

            MRestorejMap.MapTable

        End If

    Else

        MAnalysisAccessToWks.Import txtADBMName

    End If

    Set colTables = Nothing

   Unload Me

End Sub

Private Sub ChangeSelection(ByVal Selected As Boolean)
Dim lngItemCurr As Long

    On Error Resume Next

    With lstTables
        For lngItemCurr = 0 To .ListCount - 1
            .Selected(lngItemCurr) = Selected
        Next lngItemCurr
    End With

End Sub


Private Sub cmdSelectAll_Click()
    On Error Resume Next

    ChangeSelection (True)

End Sub

Private Sub lstTables_Change()
    EnableOKAsNecessary

End Sub

Private Sub lstTables_Click()
    EnableOKAsNecessary

End Sub

Private Sub txtADBMName_AfterUpdate()

    If txtADBMName <> "" Then
        ListTables
    End If

End Sub

Private Sub UserForm_Initialize()

    blnBrowseClicked = False
    txtADBMName.Text = ""
    lstTables.Clear

    cmdSelectAll.Enabled = False
    cmdResetAll.Enabled = False
    lstTables.Enabled = False
    cmdOK.Enabled = False
```

```
End Sub
```

A-1-4 frmWelcome

```
' Show welcome interface when open the workbook
Sub show_Beep()

    On Error Resume Next

    Beep
    Show

End Sub

Private Sub UserForm_Click()
    Beep
    End
End Sub
```

A-2 Modules Source Code

The source code for Modules includes as following created modules:

- **MAnalysisAccessToWks**
- **MColor**
- **MExportTablesToAccess**
- **MimportAccessToWks**
- **MRestorejMap**
- **MShellExecute**
- **MStartup**
- **MTables**

All source code in above modules are listed as following:

A-2-1 MAnalysisAccessToWks

```
Option Explicit

Private db As Database

Sub Import(strADBM As String)
'
'Purpose:   imports an access database into an excel workbook and builds a jmap
'Arguments: string containing the database to import

    On Error GoTo Handler

    'create a jmap object
    Dim map As New AccessJMapBuilder

    ' Get default Workspace.
    Dim wrkdefault As Workspace
    Set wrkdefault = DBEngine.Workspaces(0)

    ' Open database
    Dim db As Database

    If frmImportAccessToWks.BrowseClicked = True Then
        Set db = wrkdefault.OpenDatabase(strADBM)
    Else
        Set db = wrkdefault.OpenDatabase(ActiveWorkbook.Path & "\" & strADBM)
    End If



    'name the sheet
    map.NameSheet strADBM

    'insert some set's and set members
    map.InsertSetMember "View", "Tables"
    map.InsertSet "Table", "F"
    map.InsertSet "Field", "N"
    map.InsertSetMember "View", "Types"
    map.InsertSet "Type", "F"

    Dim colTables As Collection
    Set colTables = frmImportAccessToWks.SelectedTables

    'For every selected table, import table information
    Dim Tb
    Dim Rs As Recordset
    Dim I As Integer
    Dim RsSql As String
    For Each Tb In colTables

        RsSql = "SELECT * FROM [" & Tb & "]"
        Set Rs = db.OpenRecordset(RsSql, dbOpenDynaset)

        'insert a set member for the tables set
        map.InsertSetMember "Table", Tb

        ' Loop through the Microsoft Access field names and insert into
        ' the set of fields
        map.AddColumn
        For I = 0 To Rs.fields.Count - 1
            map.InsertSetMember "Field", Rs.fields(I).Name
            map.InsertAssociation "Table", Tb, "f", "Field", Rs.fields(I).Name,
"t", "Tables"
        Next I
    Next Tb

    'For every selected table, get the type information
    For Each Tb In colTables
        ' Loop through the Microsoft Access field types and insert into
        ' the set of types
        RsSql = "SELECT * FROM [" & Tb & "]"
        Set Rs = db.OpenRecordset(RsSql, dbOpenDynaset)

        For I = 0 To Rs.fields.Count - 1
            If map.FindSetMember("Type", FieldType(Rs.fields(I).type)) = False
Then
                map.AddColumn
                map.InsertSetMember "Type", FieldType(Rs.fields(I).type)
            End If
            map.InsertAssociation "Type", FieldType(Rs.fields(I).type), "f",
"Field", Rs.fields(I).Name, "t", "Types"
        Next I
    Next Tb

    'Close the database
    db.Close

    'group the sets
    map.DoRowGrouping "View"
    map.DoRowGrouping "Table"
    map.DoRowGrouping "Field"
    map.DoRowGrouping "Type"

    MColor.ColorItem

    MsgBox "Data Imported from " & strADBM & " to " & ActiveWorkbook.Name,
vbInformation, "Information"

    Exit Sub

Handler:
    Dim Message As String
    Dim Title As String
    Message = _
                    "Error Number      : " & Err _
        & Chr(10) & "Error Description: " & Error()

    Title = "An error has occured"
    MsgBox Message, , Title
    Message = ""
    Title = ""

End Sub

Function FieldType(intType As Integer) As String
'
'Purpose:   converts field type integer to return a field type string
    Select Case intType
        Case dbBoolean
            FieldType = "dbBoolean"
        Case dbByte
            FieldType = "dbByte"
        Case dbInteger
            FieldType = "dbInteger"
        Case dbLong
            FieldType = "dbLong"
        Case dbCurrency
            FieldType = "dbCurrency"
        Case dbSingle
            FieldType = "dbSingle"
        Case dbDouble
            FieldType = "dbDouble"
        Case dbDate
            FieldType = "dbDate"
        Case dbText
            FieldType = "dbText"
        Case dbLongBinary
            FieldType = "dbLongBinary"
        Case dbMemo
            FieldType = "dbMemo"
        Case dbGUID
            FieldType = "dbGUID"
    End Select

End Function
```

A-2-2 MColor

```
Sub ColorItem()
  Dim colNum As Integer
  Dim rowNum As Integer
  Dim rgnSheet As Excel.Range

  AutoAqua = RGB(60, 186, 196)
  AutoLime = RGB(153, 178, 51)
  Autogreen = RGB(0, 251, 0)
```

```
autored = RGB(255, 0, 0)
AutoLightOrg = RGB(222, 144, 51)
AutoPink = RGB(255, 0, 255)
Autopaleblue = RGB(153, 204, 255)
AutoLightPink = RGB(255, 166, 205)
AutoYellow = RGB(238, 192, 65)
AutoBrightYellow = RGB(255, 255, 0)
AutoGray = RGB(128, 128, 128)
AutoLightPurple = RGB(204, 137, 255)
AutoPurple = RGB(255, 0, 255)
AutoDarkGreen = RGB(0, 95, 0)
AutoBrightBlue = RGB(0, 204, 255)

Set rgnSheet = ActiveSheet.UsedRange

For colNum = 1 To rgnSheet.Columns.Count
  For rowNum = 1 To rgnSheet.Rows.Count

  If UCase(rgnSheet.Cells(rowNum, colNum).value) = "" Then
       rgnSheet.Cells(rowNum, colNum).Interior.ColorIndex = xlNone
     ElseIf UCase(rgnSheet.Cells(rowNum, colNum).value) = "A" Then
       rgnSheet.Cells(rowNum, colNum).Interior.ColorIndex = 16
     ElseIf UCase(rgnSheet.Cells(rowNum, colNum).value) = "v" Then
       rgnSheet.Cells(rowNum, colNum).Interior.ColorIndex = 15
     ElseIf UCase(rgnSheet.Cells(rowNum, colNum).value) = "E" Then
       rgnSheet.Cells(rowNum, colNum).Interior.Color = AutoLime
     ElseIf UCase(rgnSheet.Cells(rowNum, colNum).value) = "T" Then
       rgnSheet.Cells(rowNum, colNum).Interior.Color = Autogreen
     ElseIf UCase(rgnSheet.Cells(rowNum, colNum).value) = "F" Then
       rgnSheet.Cells(rowNum, colNum).Interior.Color = autored
     ElseIf UCase(rgnSheet.Cells(rowNum, colNum).value) = "M" Then
       rgnSheet.Cells(rowNum, colNum).Interior.Color = AutoLightOrg
     ElseIf UCase(rgnSheet.Cells(rowNum, colNum).value) = "L" Then
       rgnSheet.Cells(rowNum, colNum).Interior.Color = AutoPink
     ElseIf UCase(rgnSheet.Cells(rowNum, colNum).value) = "S" Then
       rgnSheet.Cells(rowNum, colNum).Interior.Color = AutoBrightYellow
     ElseIf UCase(rgnSheet.Cells(rowNum, colNum).value) = "N" Then
       rgnSheet.Cells(rowNum, colNum).Interior.Color = AutoLightPink
     ElseIf UCase(rgnSheet.Cells(rowNum, colNum).value) = "V" Then
       rgnSheet.Cells(rowNum, colNum).Interior.Color = AutoYellow
     ElseIf UCase(rgnSheet.Cells(rowNum, colNum).value) = "I" Then
       rgnSheet.Cells(rowNum, colNum).Interior.Color = AutoGray
     ElseIf UCase(rgnSheet.Cells(rowNum, colNum).value) = "G" Then
       rgnSheet.Cells(rowNum, colNum).Interior.Color = AutoDarkGreen
     ElseIf UCase(rgnSheet.Cells(rowNum, colNum).value) = "X" Then
       rgnSheet.Cells(rowNum, colNum).Interior.Color = AutoLightPurple
     ElseIf UCase(rgnSheet.Cells(rowNum, colNum).value) = "R" Then
       rgnSheet.Cells(rowNum, colNum).Interior.Color = AutoBrightBlue
     ElseIf UCase(rgnSheet.Cells(rowNum, colNum).value) = "L" Then
       rgnSheet.Cells(rowNum, colNum).Interior.Color = AutoPurple
     ElseIf IsNumeric(rgnSheet.Cells(rowNum, colNum).value) Then
       rgnSheet.Cells(rowNum, colNum).Font.Color = autored
    End If
  Next rowNum
 Next colNum
End Sub
```

A-2-3 MExportTablesToAccess

```
Option Explicit
Private db As Database

Sub Export(strADBM As String)
'
'Purpose:   exports worksheets from the active workbook into access
'Arguments: string containing the database name to create in access

    Dim colSheets As Collection
    Dim Wks, WksTemp
    Dim wrkdefault As Workspace
    Dim dataSource As String
    Dim Message As String
    Dim Title As String
    Dim TbIndex As Integer

    On Error GoTo Handler

    ' Get default Workspace.
    Set wrkdefault = DBEngine.Workspaces(0)

    ' Create a new encrypted database
    If frmExportTablesToAccess.SaveAsClicked = True Then
        Set db = wrkdefault.CreateDatabase(strADBM, dbLangGeneral, dbEncrypt)
    Else
        Set db = wrkdefault.CreateDatabase(ActiveWorkbook.Path & "\" & strADBM,
dbLangGeneral, dbEncrypt)
    End If

    Set colSheets = frmExportTablesToAccess.SelectedSheets

    'Create a new Table, and use the Worksheet Name as the
    'Table Name. Or Change the Table if the name already exist
    Dim tdfLoop As TableDef

    'For every selected worksheet, export to access
    For Each Wks In colSheets
        WksTemp = Wks
        TbIndex = 1
        With db

        ' Enumerate TableDefs collection.
        For Each tdfLoop In .TableDefs
            'For every table, compare if it exist
            If Wks = tdfLoop.Name Then
                Worksheets(Wks).Select
                Wks = WksTemp + Format(TbIndex)
                ActiveSheet.Name = Wks
                TbIndex = TbIndex + 1
            End If
         Next tdfLoop

        End With

        WksToAccess (Wks)

    Next

    MsgBox "Data Exported from " & ActiveWorkbook.Name & " to " & strADBM,
vbInformation, "Information"

    'Close the database

    db.Close

    'Check whether the open mdb flag is set.  If so, open the newly created access
database.
    'If user clicked on SaveAs, do not access the MDB file using the path name
    'If user entered the MDB filename, then insert active workbook path name in
the MDB string to
    'avoid "file not found" error when opening the database in access.

    If frmExportTablesToAccess.OpenADBM = True Then
        If frmExportTablesToAccess.SaveAsClicked = True Then
            ShellExec strADBM
        Else
            ShellExec ActiveWorkbook.Path & "\" & strADBM
        End If
    End If

    Exit Sub
```

```
Handler:

    ' Error 3204 means that the database already exist
    If DBEngine.Errors(0).Number = 3204 Then
        ' Open the database
         Set db = wrkdefault.OpenDatabase(strADBM)
         Resume Next

    Else
        Message = _
                    "Error Number      : " & Err _
        & Chr(10) & "Error Description: " & Error()

        Title = "An error has occured"
        MsgBox Message, , Title
        Message = ""
        Title = ""

    End If

End Sub

Sub WksToAccess(ByVal Wks)
'
'Purpose:   exports worksheets in active workbook to access
'Arguments: worksheet object
'Returns:

' Declare variables.
Dim Rs As Recordset
Dim td As TableDef
Dim Fd As Field
Dim X As Integer
Dim f As Integer
Dim r As Integer
Dim c As Integer
Dim Message As String
Dim Title As String
Dim LastColumn As Integer
Dim NumberTest As Double
Dim StartCell As Object
Dim LastCell As Object
Dim Response
Dim CreateFieldFlag As Integer
Dim flag As Integer

     CreateFieldFlag = 0
     flag = 0

     ' Turn off Screen Updating.
     Application.ScreenUpdating = False
     On Error GoTo ErrorHandler

     ' Select the worksheet and Cell "A1."
     ' In this example, you need column headers in the first row.
     ' These headers will become field names.
     Worksheets(Wks).Select
     Range("A1").Select

     ' If the ActiveCell is blank, open a message box.
     If ActiveCell.value = "" Then
         Message = "There is no data in the active cell: " & _
             ActiveSheet.Name & "!" & ActiveCell.Address & Chr(10) & _
             "Please ensure that all your worksheets have data on " & _
             "them " & Chr(10) & _
             "and the column headers start in cell A1" & Chr(10) & _
             Chr(10) & "This process will now end."

         Title = "Data Not Found"
    MsgBox Message, , Title
    Exit Sub
End If


Set td = db.CreateTableDef(Wks)

' Find the number of fields on the sheet and store the number
' of the last column in a variable.
Selection.End(xlToRight).Select
LastColumn = Selection.Column

' Select the current region. Then find what the address
' of the last cell is.
Selection.CurrentRegion.Select
Set LastCell = Range(Right(Selection.Address, _
    Len(Selection.Address) - _
    Application.Search(":", Selection.Address)))

' Go back to cell "A1."
Range("A1").Select

' Enter a loop that will go through the columns and
' create fields based on the column header.
For f = 1 To LastColumn
    flag = 0

    ' Enter a select case statement to determine
    ' the cell format.
    Select Case Left(ActiveCell.Offset(1, 0).NumberFormat, 1)
        Case "G"    'General format
            ' The "General" format presents a special problem.
            ' See above discussion for explanation
            If ActiveCell.value Like "*Zip*" Then
                Set Fd = td.CreateField(ActiveCell.value, _
                    dbMemo)
                Fd.AllowZeroLength = True
                r = LastCell.row - 1
                flag = 1
            Else
                If ActiveCell.value Like "*Postal*" Then
                    Set Fd = td.CreateField(ActiveCell.value, _
                        dbMemo)
                    Fd.AllowZeroLength = True
                    r = LastCell.row - 1
                    flag = 1
                End If
            End If

            ' Set up a text to determine if the field contains
            ' "Text" or "Numbers."
            For r = 1 To LastCell.row - 1
                If flag = 1 Then r = LastCell.row
                CreateFieldFlag = 1
                NumberTest = ActiveCell.Offset(r, 0).value / 2
            Next r

            ' If we get all the way through the loop without
            ' encountering an error, then all the values are
            ' numeric, and we assign the data type to be "dbDouble"
            If flag = 0 Then

               Set Fd = td.CreateField(ActiveCell.value, dbDouble)
            End If

        ' Check to see if the cell below is formatted as a date.
        Case "m", "d", "y"
            Set Fd = td.CreateField(ActiveCell.value, dbDate)

        ' Check to see if the cell below is formatted as currency.
```

```
            Case "$", "_"
                Set Fd = td.CreateField(ActiveCell.value, dbCurrency)

            ' All purpose trap to set field to text.
            Case Else
                Set Fd = td.CreateField(ActiveCell.value, dbMemo)
            End Select

        ' Append the new field to the fields collection.
        td.fields.Append Fd

        ' Move to the right one column.
        ActiveCell.Offset(0, 1).Range("A1").Select

    ' Repeat the procedure with the next field (column).
    Next f

    ' Append the new Table to the TableDef collection.
    db.TableDefs.Append td

    ' Select Cell "A2" to start the setup for moving the data from
    ' the worksheet to the database.
    Range("A2").Select

    ' Define the StartCell as the Activecell. All record addition
    ' will be made relative to this cell.
    Set StartCell = Range(ActiveCell.Address)

    ' Open a recordset based on the name of the activesheet.
    Set Rs = db.OpenRecordset(Wks)

    ' Loop through all the data on the sheet and add it to the
    ' recordset in the database.
    For X = 0 To LastCell.row - 2
        Rs.AddNew
        For c = 0 To LastColumn - 1
            Rs.fields(c) = StartCell.Offset(X, c).value

        Next c
        Rs.Update
    Next X

    Application.ScreenUpdating = True

    Exit Sub

ErrorHandler:
  Select Case Err
     Case 3204   ' Database already exists.
        Message = "There has been an error creating the database." & _
             Chr(10) & _
             Chr(10) & "Error Number: " & Err & _
             Chr(10) & "Error Description: " & Error() & _
             Chr(10) & _
             Chr(10) & "Would you like to delete the existing" & _
             "database:" & Chr(10) & _
             Chr(10) & _
             Left(ActiveWorkbook.Name, Len(ActiveWorkbook.Name) - 4) & _
             ".mdb"
         Title = "Error in Database Creation"
         Response = MsgBox(Message, vbYesNo, Title)
         If Response = vbYes Then
             Kill _
               Left(ActiveWorkbook.Name, Len(ActiveWorkbook.Name) - 4) _
               & ".mdb"
             Message = ""
             Title = ""
             Resume
         Else
             Message = "In order to run this procedure you need" & _
                 Chr(10) & "to do ONE of the following:" & _
                Chr(10) & _
                Chr(10) & "1.  Move the existing database to a " & _
                "different directory, or " & _
                Chr(10) & "2.  Rename the existing database, or" & _
                Chr(10) & "3.  Move the workbook to a different " & _
                "directory, or" & _
                Chr(10) & "4.  Rename the workbook"
            Title = "Perform ONE of the following:"
            MsgBox Message, , Title
            Message = ""
            Title = ""
            Exit Sub
        End If

    ' Check to see if the error was Type Mismatch. If so, set the
    ' file to dbMemo.
    Case 13 ' Type mismatch.
        If CreateFieldFlag = 1 Then
            Set Fd = td.CreateField(ActiveCell.value, dbMemo)
            Fd.AllowZeroLength = True
            flag = 1
            r = LastCell.row - 1
            CreateFieldFlag = 0
            Resume Next
        Else
            Message = _
                            "Worksheet Name    :  " & Wks _
                & Chr(10) _
                & Chr(10) & "Error Number      :  " & Err _
                & Chr(10) & "Error Description:  " & Error() _
                & Chr(10) & Chr(10) & "Worksheet cannot be exported!"

            Title = "Type Mismatch"
            MsgBox Message, , Title
            Message = ""
            Title = ""
        End If

    ' For any other error, display the error.
    Case Else
       Message = _
                        "Worksheet Name    :  " & Wks _
            & Chr(10) _
            & Chr(10) & "Error Number      :  " & Err _
            & Chr(10) & "Error Description:  " & Error() _
            & Chr(10) & Chr(10) & "Worksheet cannot be exported!"

        Title = "An error has occured"
        MsgBox Message, , Title
        Message = ""
        Title = ""
  End Select
End Sub
```

A-2-4 MImportAccessToWks

```
Option Explicit

Private db As Database

Sub Import(strADBM As String)
'
'Purpose:   imports an access database into an excel workbook and builds a jmap
'Arguments: string containing the database to import

    On Error GoTo Handler

    'Get default Workspace.
    Dim wrkdefault As Workspace
    Set wrkdefault = DBEngine.Workspaces(0)
```

```
'Open database
Dim db As Database

If frmImportAccessToWks.BrowseClicked = True Then
    Set db = wrkdefault.OpenDatabase(strADBM)
Else
    Set db = wrkdefault.OpenDatabase(ActiveWorkbook.Path & "\" & strADBM)
End If


Dim colTables As Collection
Set colTables = frmImportAccessToWks.SelectedTables

'For every selected table, import table information
Dim Tb
Dim Rs As Recordset
Dim I, J As Integer
Dim RsSql As String
Dim newsheet, shtName As String
Dim fldName As Field
Dim fldValue As String
Dim strColunmWdLen As Integer

Dim intCount, numRow As Integer

For Each Tb In colTables

'new sheet name
    newsheet = Tb


    On Error Resume Next
    Sheets(newsheet).Select
    On Error Resume Next

    'add new sheet
    Sheets.Add
    ActiveSheet.Name = newsheet
    ActiveWindow.Zoom = 75

    RsSql = "SELECT * FROM [" & Tb & "]"
    Set Rs = db.OpenRecordset(RsSql, dbOpenDynaset)

    'Loop through the Microsoft Access field names and insert into
    ' the set of fields

    For I = 0 To Rs.fields.Count - 1

    intCount = 1
   'strColunmWdLen = 1
    Cells(intCount, I + 1) = Rs.fields(I).Name
    Cells(intCount, I + 1).Interior.ColorIndex = 8
    Cells(intCount, I + 1).Font.Bold = True
    Cells(intCount, I + 1).Borders.LineStyle = xlDouble
    strColunmWdLen = 16
    Worksheets(newsheet).Columns(I + 1).ColumnWidth = strColunmWdLen

    Do Until Rs.EOF
        Set fldName = Rs.fields(I)
        fldValue = fldName.value

        intCount = intCount + 1

        Cells(intCount, I + 1) = fldValue

        If (strColunmWdLen < Len(fldValue)) Then
         strColunmWdLen = Len(fldValue)
         Worksheets(newsheet).Columns(I + 1).ColumnWidth = strColunmWdLen
        End If
```

```
            Rs.MoveNext
            Loop

            Rs.MoveFirst

        Next I

    Next Tb

    db.Close

    MsgBox "Data Imported from " & strADBM & " to " & ActiveWorkbook.Name,
vbInformation, "Information"

    Exit Sub

Handler:
    Dim Message As String
    Dim Title As String
    Message = _
                    "Error Number      : " & Err _
        & Chr(10) & "Error Description: " & Error()

    Title = "An error has occured"
    MsgBox Message, , Title
    Message = ""
    Title = ""

End Sub
```

A-2-5 MRestorejMap

```
Dim rstSheetName, cItemSheetName, cTupleSheetName As String

Sub MapTable()
'
' CreateTables Macro
' Macro recorded 07/12/2001 by Minghui Han

  Dim rstSheetNameTemp, cItemSheetNameTemp, cTupleSheetNameTemp, sTempName As
String
  Dim nCount, nSheetCount As Integer
  Dim nEndStep As Integer

  rstSheetNameTemp = "{jMapRestore}"
  cItemSheetNameTemp = "{cItems}"
  cTupleSheetNameTemp = "{cTuples}"

  nCount = 1

  rstSheetName = rstSheetNameTemp
  cItemSheetName = cItemSheetNameTemp
  cTupleSheetName = cTupleSheetNameTemp

  Call MakejMap
  nEndStep = Sheets.Count

  For nSheetCount = nEndStep To 1 Step -1

    sTempName = cItemSheetNameTemp + Format(nCount)

    If Sheets(nSheetCount).Name = sTempName Then
        rstSheetName = rstSheetNameTemp + Format(nCount)
        cItemSheetName = sTempName
        cTupleSheetName = cTupleSheetNameTemp + Format(nCount)
        nCount = nCount + 1

        Call MakejMap
        nSheetCount = nSheetCount + 1
    End If

Next nSheetCount

End Sub
Sub MakejMap()

  Dim newsheet As Sheets
  Dim nHorPos, nVerPos, nColumnStart, nColumnEnd, nRowStart, nRowEnd, nToRow As
Integer
  Dim numItems, fndlen As Integer
  Dim strCellValue, searchString, subString, searchChar As String


  On Error Resume Next
  Sheets(rstSheetName).Select
  On Error Resume Next

  'add new sheet
  Sheets.Add
  ActiveSheet.Name = rstSheetName
  ActiveWindow.Zoom = 75

  Sheets(cTupleSheetName).Select

  'Get the Activesheet's range
  Set rgnSheet = ActiveSheet.UsedRange

  nRowEnd = rgnSheet.Rows.Count
  nColumnEnd = rgnSheet.Columns.Count
  nToRow = nRowEnd

  'find the started column for "Roles" data item
  For nRowStart = 1 To nRowEnd
        For nColumnStart = 1 To nColumnEnd

            strCellValue = Cells(nRowStart, nColumnStart)

            If InStr(1, strCellValue, "Roles", vbTextCompare) = 1 Then

                nHorPos = nRowStart         'find first row position of text with
"Roles"
                nVerPos = nColumnStart      'find column position of text with
"Roles"
                GoTo getValue

            End If
        Next nColumnStart
    Next nRowStart

getValue:

  For nRowStart = nHorPos + 2 To nRowEnd

            searchString = Cells(nRowStart, nVerPos)


            fndlen = InStr(1, searchString, ",")

            numItems = 1

            Do Until fndlen = 0

              subString = Left(searchString, fndlen - 1)

              searchString = Mid(searchString, fndlen + 1) ' Returns rest string

              fndlen = InStr(1, searchString, ",")
            Sheets(rstSheetName).Select

            subString = Trim(subString)
            Cells(numItems, nRowStart - 2).value = subString

            numItems = numItems + 1

            Sheets(cTupleSheetName).Select

          Loop

          'put last char to the new sheet
          Sheets(rstSheetName).Select
          subString = Trim(searchString)
          Cells(numItems, nRowStart - 2).value = subString
          Sheets(cTupleSheetName).Select

 Next nRowStart

 Sheets(rstSheetName).Select
 Set rgnSheet = ActiveSheet.UsedRange

 nRowEnd = rgnSheet.Rows.Count
 nColumnEnd = rgnSheet.Columns.Count

  'replace "?" with empty
  For nRowStart = 1 To nRowEnd

      For nColumnStart = 1 To nColumnEnd

          strCellValue = Cells(nRowStart, nColumnStart)

          If (strCellValue = "?") Then

              Cells(nRowStart, nColumnStart).value = ""

          End If

      Next nColumnStart

  Next nRowStart

  ...........

 Sheets(cItemSheetName).Select

 'Get the Activesheet's range
 Set rgnSheet = ActiveSheet.UsedRange

 nRowEnd = rgnSheet.Rows.Count
 nColumnEnd = rgnSheet.Columns.Count

 'find the started column for "Roles" data item
 For nRowStart = 1 To nRowEnd
       For nColumnStart = 1 To nColumnEnd

           strCellValue = Cells(nRowStart, nColumnStart)

           If InStr(1, strCellValue, "DataItem", vbTextCompare) = 1 Then

               nHorPos = nRowStart         'find first row position of text with
"Roles"
               nVerPos = nColumnStart      'find column position of text with
"Roles"
               GoTo getValueItem

           End If
       Next nColumnStart
   Next nRowStart
```

```
getValueItem:

  numItems = 1

  For nRowStart = nHorPos + 1 To nRowEnd

            searchString = Cells(nRowStart, nVerPos)


            Sheets(rstSheetName).Select

            searchString = Trim(searchString)
            Cells(numItems, nToRow + 1).value = searchString

            numItems = numItems + 1

            Sheets(cItemSheetName).Select


   Next nRowStart


   Sheets(rstSheetName).Select

   'add column numbers for each row
    For nRowStart = 1 To nRowEnd - 1
        numItems = 0
        For nColumnStart = 1 To nToRow - 1

                If Not Cells(nRowStart, nColumnStart) = "" Then
                   numItems = numItems + 1
                End If
        Next nColumnStart

        Cells(nRowStart, nToRow - 1) = numItems

    Next nRowStart

    numItems = 0
    'find the started column for data item
    For nRowStart = nRowEnd - 1 To 1 Step -1

        numItems = numItems + 1

        strCellValue = Cells(nRowStart, nToRow + 1)

        If InStr(1, strCellValue, "{", vbTextCompare) = 1 Then

            Cells(nRowStart, nToRow) = numItems - 1

            Range(Cells(nRowStart, 1), Cells(nRowStart, nToRow + 1)).Select
            Selection.Font.Bold = True

            numItems = 0

        End If

    Next nRowStart

    'fomat column width
    Range(Cells(1, 1), Cells(nRowEnd - 1, nToRow)).Select
    Selection.ColumnWidth = 2

    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .ShrinkToFit = False
        .MergeCells = False

    End With

    Range(Cells(1, nToRow + 1), Cells(nRowEnd - 1, nToRow + 1)).Select
    Selection.ColumnWidth = 20
    MColor.ColorItem

    Cells(1, 1).Select

End Sub
```

A-2-6 MShellExecute

```
'
'Purpose:   this module is needed to display a html page in the default web
'           browser

Option Explicit

Private Declare Function ShellExecute Lib "shell32.dll" Alias _
        "ShellExecuteA" (ByVal hwnd As Long, ByVal lpszOp As _
        String, ByVal lpszFile As String, ByVal lpszParams As String, _
        ByVal lpszDir As String, ByVal FsShowCmd As Long) As Long

Private Declare Function GetDesktopWindow Lib "user32" () As Long

Private Const SW_SHOWNORMAL = 1
Private Const SW_SHOWMAXIMIZED = 3

Private Const SE_ERR_FNF = 2&
Private Const SE_ERR_PNF = 3&
Private Const SE_ERR_ACCESSDENIED = 5&
Private Const SE_ERR_OOM = 8&
Private Const SE_ERR_DLLNOTFOUND = 32&
Private Const SE_ERR_SHARE = 26&
Private Const SE_ERR_ASSOCINCOMPLETE = 27&
Private Const SE_ERR_DDETIMEOUT = 28&
Private Const SE_ERR_DDEFAIL = 29&
Private Const SE_ERR_DDEBUSY = 30&
Private Const SE_ERR_NOASSOC = 31&
Private Const ERROR_BAD_FORMAT = 11&

Sub ShellExec(DocName As String)
    Dim r As Long, Msg As String
    Dim Scr_hDC As Long

    Scr_hDC = GetDesktopWindow()

    r = ShellExecute(Scr_hDC, "Open", DocName, "", "C:\", SW_SHOWNORMAL)

    If r <= 32 Then
        'There was an error
        Select Case r
            Case SE_ERR_FNF
                Msg = "File not found"
            Case SE_ERR_PNF
                Msg = "Path not found"
            Case SE_ERR_ACCESSDENIED
                Msg = "Access denied"
            Case SE_ERR_OOM
                Msg = "Out of memory"
            Case SE_ERR_DLLNOTFOUND
                Msg = "DLL not found"
            Case SE_ERR_SHARE
                Msg = "A sharing violation occurred"
            Case SE_ERR_ASSOCINCOMPLETE
                Msg = "Incomplete or invalid file association"
            Case SE_ERR_DDETIMEOUT
```

```
                Msg = "DDE Time out"
            Case SE_ERR_DDEFAIL
                Msg = "DDE transaction failed"
            Case SE_ERR_DDEBUSY
                Msg = "DDE busy"
            Case SE_ERR_NOASSOC
                Msg = "No association for file extension"
            Case ERROR_BAD_FORMAT
                Msg = "Invalid EXE file or error in EXE image"
            Case Else
                Msg = "Unknown error"
        End Select

        MsgBox Msg, vbInformation

    End If

End Sub
```

A-2-7 MStartup

```
Option Explicit

'=================================================================
'Module Level Constant Declaration Section
'=================================================================

Private Const MACRO_MENU_CAPTION As String = "ADM&jMap" ' added by han

Public bjMaptoAccess As Boolean
Public bAccesstojMap As Boolean


Sub RemovejMapMacroMenu()
  Dim cbct As CommandBarControl
  On Error Resume Next

  For Each cbct In CommandBars.ActiveMenuBar.Controls
    If 0 = StrComp(cbct.Caption, MACRO_MENU_CAPTION, vbBinaryCompare) Then
      Call cbct.Delete
    End If
  Next cbct
End Sub

Public Sub AddjMapMacroMenu()
'
' MStartup Macro
' Macro recorded 3/23/2001 by Minghui Han
'
' Keyboard Shortcut: Ctrl+b
'


  Dim cbpopTopMenu As CommandBarPopup
  Dim cbpopSubMenu As CommandBarPopup
  Dim cbctls As CommandBarControls

  On Error Resume Next

  ' Ensure we have no duplicates
  Call RemovejMapMacroMenu

  bjMaptoAccess = False
  bAccesstojMap = False

  Set cbpopTopMenu = CommandBars.ActiveMenuBar.Controls.Add(type:=msoControlPopup)
  With cbpopTopMenu
    .Caption = MACRO_MENU_CAPTION
    .OnAction = "mnujMap_OnAction"
    .Visible = True
  End With

  Set cbctls = cbpopTopMenu.Controls

  ' Add the sub items to the menu
  With cbctls.Add(msoControlButton)
      .Caption = "&Create Tables"
      .OnAction = "mnuCreateTables_OnAction"
      .FaceId = 240
  End With

  ' Add the sub items to the menu
  With cbctls.Add(msoControlButton)
      .Caption = "&Remove Tables"
      .OnAction = "mnuRemoveTables_OnAction"
      .FaceId = 2002
  End With

  ' Add the sub items to the menu
  With cbctls.Add(msoControlButton)
      .Caption = "&jMap->DB"
      .OnAction = "mnujMaptoAccess_OnAction"
      .FaceId = 2116
  End With

  ' Add the sub items to the menu
  With cbctls.Add(msoControlButton)
      .Caption = "&DB->jMap"
      .OnAction = "mnuAccesstojMap_OnAction"
      .FaceId = 2109
  End With

   ' Add the sub items to the menu
  With cbctls.Add(msoControlButton)
      .Caption = "DB jMap &Analysis"
      .OnAction = "mnuAnalysisAccesstojMap_OnAction"
      .FaceId = 2114
  End With

  With cbctls.Add(msoControlButton)
        .Caption = "&Help..."
        .OnAction = "mnuHelp_OnAction"
        .BeginGroup = True
        .FaceId = 49
  End With

End Sub

Private Sub mnujMap_OnAction()
  On Error Resume Next

End Sub

Private Sub mnujMaptoAccess_OnAction()

    On Error Resume Next

    bjMaptoAccess = True
    frmBookSheetInfo.Show
    bjMaptoAccess = False


End Sub

Private Sub mnuAccesstojMap_OnAction()

    On Error Resume Next
    bAccesstojMap = True
    frmImportAccessToWks.Show
    bAccesstojMap = False

End Sub
Private Sub mnuAnalysisAccesstojMap_OnAction()

    On Error Resume Next

    bAccesstojMap = False
    frmImportAccessToWks.Show

End Sub

Private Sub mnuHelp_OnAction()

    On Error Resume Next

    ShellExec ActiveWorkbook.Path & "\" & "Help.htm"

End Sub

Private Sub mnuCreateTables_OnAction()

    On Error Resume Next

    frmBookSheetInfo.Show

End Sub

Private Sub mnuRemoveTables_OnAction()

    On Error Resume Next

    MTables.RemoveTables

End Sub
```

A-2-8 MTables

```
Sub CreateID(ByVal strSheetID As String, ByVal strBookID As String)
'
' CreateTables Macro
' Macro recorded 07/03/2001 by Minghui Han
'
' Keyboard Shortcut: Ctrl+t
'
    Dim nHorPos, nVerPos, nColumnStart, nColumnEnd, nRowStart, nRowEnd As Integer
    Dim numSetID, numMemberID, numItems, numTables, numField As Integer '
    Dim oldsheet, newsheet, strCellValue As String

    nHorPos = 1
    nVerPos = 1
    numSetID = 0
    numMemberID = 0

    oldsheet = ActiveSheet.Name
    newsheet = "<" + oldsheet + ">"

   'copy oldsheet contents to the new sheet and renamed as newsheet name
    Sheets(oldsheet).Copy before:=Sheets(oldsheet)
    ActiveSheet.Name = newsheet

    'Get the Activesheet's range
    Set rgnSheet = ActiveSheet.UsedRange

    nRowEnd = rgnSheet.Rows.Count
    nColumnEnd = rgnSheet.Columns.Count
    'replace empty cell with "?"
    For nRowStart = 1 To nRowEnd
        For nColumnStart = 1 To nColumnEnd

            strCellValue = Cells(nRowStart, nColumnStart)

            If ((strCellValue = "") And (nColumnStart < nColumnEnd - 2)) Then

                Cells(nRowStart, nColumnStart).value = "?"

            End If

        Next nColumnStart

    Next nRowStart

    'find the started column for data item
    For nRowStart = 1 To nRowEnd
        For nColumnStart = 1 To nColumnEnd

            strCellValue = Cells(nRowStart, nColumnStart)

            If InStr(1, strCellValue, "{", vbTextCompare) = 1 Then

                nHorPos = nRowStart         'find first row position of "{"
                nVerPos = nColumnStart      'find column position of "{"
                GoTo setID

            End If
        Next nColumnStart
    Next nRowStart

setID:
    For nRowStart = 1 To nRowEnd

            strCellValue = Cells(nRowStart, nVerPos)

            If InStr(1, strCellValue, "{", vbTextCompare) = 1 Then
              numSetID = numSetID + 1
              numMemberID = 0
              Cells(nRowStart, nVerPos + 5).value = "N"

            Else
              Cells(nRowStart, nVerPos + 5).value = "M"
            End If

            Cells(nRowStart, nVerPos + 1).value = numSetID
            Cells(nRowStart, nVerPos + 2).value = numMemberID
            Cells(nRowStart, nVerPos + 3).value = strSheetID
            Cells(nRowStart, nVerPos + 4).value = strBookID

            numMemberID = numMemberID + 1

    Next nRowStart

    'color and format cell's property

    'for SetID column
    Range(Cells(1, nVerPos + 1), Cells(nRowEnd, nVerPos + 1)).Select
    Selection.Font.ColorIndex = 3
    Selection.Font.Bold = True
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .ShrinkToFit = False
        .MergeCells = False
    End With
    'for MemeberID column
    Range(Cells(1, nVerPos + 2), Cells(nRowEnd, nVerPos + 2)).Select
    Selection.Font.ColorIndex = 9
    Selection.Font.Bold = True
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .ShrinkToFit = False
        .MergeCells = False
    End With

    'for SheetID column
    Range(Cells(1, nVerPos + 3), Cells(nRowEnd, nVerPos + 3)).Select
    Selection.Font.ColorIndex = 52
    Selection.Font.Bold = True
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .ShrinkToFit = False
        .MergeCells = False
    End With

    'for WoorkBookID column
    Range(Cells(1, nVerPos + 4), Cells(nRowEnd, nVerPos + 4)).Select
    Selection.Font.ColorIndex = 7
    Selection.Font.Bold = True
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .ShrinkToFit = False
        .MergeCells = False
    End With

    'for WoorkBookID column
    Range(Cells(1, nVerPos + 5), Cells(nRowEnd, nVerPos + 5)).Select
    Selection.Font.ColorIndex = 10
    Selection.Font.Bold = True
    Selection.ColumnWidth = 10

    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .ShrinkToFit = False
        .MergeCells = False

    End With

End Sub
Sub CreateTables()
'
' CreateTables Macro
' Macro recorded 03/22/2001 by Minghui Han
'
' Keyboard Shortcut: Ctrl+t
'
Dim nHorPos, nVerPos, nColumnStart, nRowStart As Integer
Dim numItems, numTables, numField As Integer '
Dim oldsheet, newsheet, strCellValue As String
Dim strField1(1 To 500) As String
Dim strField2(1 To 500) As String
Dim strField3(1 To 500) As String
Dim strField4(1 To 500) As String
Dim strField5(1 To 500) As String
Dim strField6(1 To 500) As String
Dim strColunmWdLen1 As Integer
Dim strColunmWdLen2 As Integer
Dim strColunmWdLen3 As Integer
Dim strColunmWdLen4 As Integer
Dim strColunmWdLen5 As Integer

'Get the Activesheet's range
Set rgnSheet = ActiveSheet.UsedRange

nHorPos = 1
nVerPos = 1
oldsheet = ActiveSheet.Name

For nRowStart = 1 To rgnSheet.Rows.Count
    For nColumnStart = 1 To rgnSheet.Columns.Count

        strCellValue = Cells(nRowStart, nColumnStart)

        If InStr(1, strCellValue, "{", vbTextCompare) = 1 Then
                nHorPos = nRowStart         'find first row position of "{"
                nVerPos = nColumnStart      'find column position of "{"
                strField1(1) = strCellValue
                GoTo firstTable
        End If
    Next nColumnStart
Next nRowStart

firstTable:
    Sheets(oldsheet).Select

    strField1(1) = "DataItem"
    strField2(1) = "SetID"
    strField3(1) = "MemberID"
    strField4(1) = "SheetID"
    strField5(1) = "WorkbookID"

    numItems = 2   'start to get second row's value, and so on

    For nRowStart = nHorPos To rgnSheet.Rows.Count ' - 1

        strField1(numItems) = Cells(nRowStart, nVerPos)
        strField2(numItems) = Cells(nRowStart, nVerPos + 1)
        strField3(numItems) = Cells(nRowStart, nVerPos + 2)
        strField4(numItems) = Cells(nRowStart, nVerPos + 3)
        strField5(numItems) = Cells(nRowStart, nVerPos + 4)

        numItems = numItems + 1

    Next nRowStart

    'ready to add new sheet

    'new sheet name
    newsheet = "{cItems}"
    On Error Resume Next
    Sheets(newsheet).Select
    On Error Resume Next

    'add new sheet
    Sheets.Add
    ActiveSheet.Name = newsheet
    Columns("A:E").ColumnWidth = 15
    strColunmWdLen1 = 1

    ActiveSheet.Name = newsheet
    ActiveWindow.Zoom = 75
    For nRowStart = 1 To numItems - 1
        Cells(nRowStart, 1) = strField1(nRowStart)
        Cells(nRowStart, 2) = strField2(nRowStart)
        Cells(nRowStart, 3) = strField3(nRowStart)
        Cells(nRowStart, 4) = strField4(nRowStart)
        Cells(nRowStart, 5) = strField5(nRowStart)

        If (strColunmWdLen1 < Len(strField1(nRowStart))) Then
            strColunmWdLen1 = Len(strField1(nRowStart))
            Columns("A:A").ColumnWidth = strColunmWdLen1
        End If

        If nRowStart = 1 Then

             Cells(nRowStart, 1).Interior.ColorIndex = 8
             Cells(nRowStart, 2).Interior.ColorIndex = 8
             Cells(nRowStart, 3).Interior.ColorIndex = 8
             Cells(nRowStart, 4).Interior.ColorIndex = 8
             Cells(nRowStart, 5).Interior.ColorIndex = 8
             Cells(nRowStart, 1).Borders.LineStyle = xlDouble
             Cells(nRowStart, 2).Borders.LineStyle = xlDouble
             Cells(nRowStart, 3).Borders.LineStyle = xlDouble
             Cells(nRowStart, 4).Borders.LineStyle = xlDouble
             Cells(nRowStart, 5).Borders.LineStyle = xlDouble

        Else
             Cells(nRowStart, 1).Font.ColorIndex = 32
             Cells(nRowStart, 2).Font.ColorIndex = 3
             Cells(nRowStart, 3).Font.ColorIndex = 3
             Cells(nRowStart, 4).Font.ColorIndex = 3
             Cells(nRowStart, 5).Font.ColorIndex = 3

             Cells(nRowStart, 1).Interior.ColorIndex = 2
             Cells(nRowStart, 2).Interior.Color = RGB(255, 204, 153)
             Cells(nRowStart, 3).Interior.Color = RGB(204, 255, 204)
             Cells(nRowStart, 4).Interior.Color = RGB(200, 204, 253)
             Cells(nRowStart, 5).Interior.Color = RGB(100, 250, 200)

             Cells(nRowStart, 1).Borders.LineStyle = xlDot
             Cells(nRowStart, 2).Borders.LineStyle = xlDot
             Cells(nRowStart, 3).Borders.LineStyle = xlDot
             Cells(nRowStart, 4).Borders.LineStyle = xlDot
             Cells(nRowStart, 5).Borders.LineStyle = xlDot

        End If

    Next nRowStart

    Range("A1:E1").Select
    Selection.Font.Bold = True
    Cells(1, 1).Select

otherTable:

     strColunmWdLen1 = 1
     strColunmWdLen2 = 1
     strColunmWdLen3 = 1
     strColunmWdLen4 = 1
     strColunmWdLen5 = 1

    'go back to old sheet ready to read the each column's value
     Sheets(oldsheet).Select

     strField1(1) = "CtupleID"
     strField2(1) = "Roles"
     strField3(1) = "SetID"
     strField4(1) = "MemberID"
     strField5(1) = "SheetID"
     strField6(1) = "WorkbookID"
    'new sheet name
    newsheet = "{cTuples}"
    On Error Resume Next
    Sheets(newsheet).Select
    On Error Resume Next

    'add new sheet
    Sheets.Add
    ActiveSheet.Name = newsheet
    Columns("A:F").ColumnWidth = 10

    ActiveSheet.Name = newsheet
    ActiveWindow.Zoom = 75

    'add field name in the first row to the new sheet, and format them
    For numField = 1 To 6

        Cells(1, numField).Interior.ColorIndex = 8
        Cells(1, numField).Borders.LineStyle = xlDouble

        If numField = 1 Then
            Cells(1, numField) = strField1(1)
        ElseIf numField = 2 Then
            Cells(1, numField) = strField2(1)
        ElseIf numField = 3 Then
            Cells(1, numField) = strField3(1)
        ElseIf numField = 4 Then
            Cells(1, numField) = strField4(1)
        ElseIf numField = 5 Then
            Cells(1, numField) = strField5(1)
        ElseIf numField = 6 Then
            Cells(1, numField) = strField6(1)
        End If

    Next numField

    Sheets(oldsheet).Select

   'For all other columns tables
    For nColumnStart = 0 To nVerPos - 3

        numItems = 2   'start to get second row's value, and so on
        strField2(numItems) = Cells(1, nColumnStart)
        strField3(numItems) = Cells(1, nVerPos + 1)
        strField4(numItems) = Cells(1, nVerPos + 2)
        strField5(numItems) = Cells(1, nVerPos + 3)
        strField6(numItems) = Cells(1, nVerPos + 4)

        numItems = numItems + 1

        If nColumnStart = 0 Then

            strField2(numItems - 1) = Cells(1, nVerPos + 5)
            'to get last colume's value i.e. N, M...
            For nRowStart = 2 To rgnSheet.Rows.Count

                If Not Cells(nRowStart, nVerPos + 4) = "" Then
                    strField2(numItems) = CStr(strField2(numItems - 1)) + ", " +
CStr(Cells(nRowStart, nVerPos + 5))
                    strField3(numItems) = CStr(strField3(numItems - 1)) + ", " +
CStr(Cells(nRowStart, nVerPos + 1))
                    strField4(numItems) = CStr(strField4(numItems - 1)) + ", " +
CStr(Cells(nRowStart, nVerPos + 2))
                    strField5(numItems) = CStr(strField5(numItems - 1)) + ", " +
CStr(Cells(nRowStart, nVerPos + 3))
                    strField6(numItems) = CStr(strField6(numItems - 1)) + ", " +
CStr(Cells(nRowStart, nVerPos + 4))

                    numItems = numItems + 1
                End If
            Next nRowStart

        Else

            'to retrive the jMap notation value
            'since the sheet has one more row for title, minus one to get real
rows
            For nRowStart = 2 To rgnSheet.Rows.Count

                If Not Cells(nRowStart, nColumnStart) = "" Then
                    strField2(numItems) = CStr(strField2(numItems - 1)) + ", " +
CStr(Cells(nRowStart, nColumnStart))
                    strField3(numItems) = CStr(strField3(numItems - 1)) + ", " +
CStr(Cells(nRowStart, nVerPos + 1))
                    strField4(numItems) = CStr(strField4(numItems - 1)) + ", " +
CStr(Cells(nRowStart, nVerPos + 2))
                    strField5(numItems) = CStr(strField5(numItems - 1)) + ", " +
CStr(Cells(nRowStart, nVerPos + 3))
                    strField6(numItems) = CStr(strField6(numItems - 1)) + ", " +
CStr(Cells(nRowStart, nVerPos + 4))

                    numItems = numItems + 1
                End If

            Next nRowStart
        End If



       'ready to add new sheet
        On Error Resume Next
        Sheets(newsheet).Select

        'assign the value to new sheet
        Cells(nColumnStart + 2, 1) = CStr(nColumnStart)
        Cells(nColumnStart + 2, 2) = strField2(numItems - 1)
        Cells(nColumnStart + 2, 3) = strField3(numItems - 1)
        Cells(nColumnStart + 2, 4) = strField4(numItems - 1)
        Cells(nColumnStart + 2, 5) = strField5(numItems - 1)
        Cells(nColumnStart + 2, 6) = strField6(numItems - 1)


        'add field data in the new sheet, and format them
        Cells(nColumnStart + 2, 1).Font.ColorIndex = 32
        Cells(nColumnStart + 2, 2).Font.ColorIndex = 3
        Cells(nColumnStart + 2, 3).Font.ColorIndex = 32
        Cells(nColumnStart + 2, 4).Font.ColorIndex = 3
        Cells(nColumnStart + 2, 5).Font.ColorIndex = 32
        Cells(nColumnStart + 2, 6).Font.ColorIndex = 3

        Cells(nColumnStart + 2, 1).Interior.ColorIndex = 2
        Cells(nColumnStart + 2, 2).Interior.Color = RGB(255, 255, 153)
        Cells(nColumnStart + 2, 3).Interior.Color = RGB(204, 255, 204)
        Cells(nColumnStart + 2, 4).Interior.Color = RGB(255, 204, 153)
        Cells(nColumnStart + 2, 5).Interior.Color = RGB(240, 200, 223)
        Cells(nColumnStart + 2, 6).Interior.Color = RGB(220, 250, 230)

        Cells(nColumnStart + 2, 1).Borders.LineStyle = xlDot
        Cells(nColumnStart + 2, 2).Borders.LineStyle = xlDot
        Cells(nColumnStart + 2, 3).Borders.LineStyle = xlDot
        Cells(nColumnStart + 2, 4).Borders.LineStyle = xlDot
        Cells(nColumnStart + 2, 5).Borders.LineStyle = xlDot
        Cells(nColumnStart + 2, 6).Borders.LineStyle = xlDot



        If (strColunmWdLen1 < Len(strField2(numItems - 1))) Then
           strColunmWdLen1 = Len(strField2(numItems - 1))
           Columns("B:B").ColumnWidth = strColunmWdLen1
        End If
        If (strColunmWdLen2 < Len(strField3(numItems - 1))) Then
            strColunmWdLen2 = Len(strField3(numItems - 1))
            Columns("C:C").ColumnWidth = strColunmWdLen2
        End If

        If (strColunmWdLen3 < Len(strField4(numItems - 1))) Then
            strColunmWdLen3 = Len(strField4(numItems - 1))
            Columns("D:D").ColumnWidth = strColunmWdLen3
        End If

        If (strColunmWdLen4 < Len(strField5(numItems - 1))) Then
            strColunmWdLen4 = Len(strField5(numItems - 1))
            Columns("E:E").ColumnWidth = strColunmWdLen4
        End If

        If (strColunmWdLen5 < Len(strField6(numItems - 1))) Then
            strColunmWdLen5 = Len(strField6(numItems - 1))
            Columns("F:F").ColumnWidth = strColunmWdLen5
        End If

        Range("A1:F1").Select
        Selection.Font.Bold = True
        Cells(1, 1).Select

        Sheets(oldsheet).Select

    Next nColumnStart

End Sub
Sub RemoveTables()
'
' RemoveTables Macro
' Macro recorded 03/22/2001 by Minghui Han
'
'
   Dim Wks As Worksheet

   For Each Wks In Worksheets
        If Wks.type = xlWorksheet Then
            If Wks.Visible Then
                If InStr(Wks.Name, "(") Or InStr(Wks.Name, "<") Then
                    Application.DisplayAlerts = False
                    Wks.Delete
                    Application.DisplayAlerts = True
                End If
            End If
        End If
    Next Wks

End Sub
```

A-3 Class Modules Source Code

The source code for Class Modules includes as following created class:

- **AccessJMapBuilder**
- **Table**

All source code in above Classes are listed as following:

A-3-1 AccessJMapBuilder

```
'Option Compare Database
Option Explicit

Private App As Excel.Application     'pointer to excel application
Private Book As Excel.Workbook       'pointer to excel workbook
Private Sheet As Excel.Worksheet     'pointer to excel worksheet

Private SetTable As Table  'pointer to table, holds the cordinates of the written
sets

Private Sub Class_Initialize()
'
'Purpose: create the workbook and sheet to work in
'BECAUSE: we cannot run build database on a new book

    'create application and workbook, make pointer point to created objects
    Set App = Workbooks.Application
    Set Sheet = App.Sheets.Add
    Sheet.Activate

    'inputting first entry in the spreadsheet
    'first entry is the {View} set
    'done explicitly because insert functions (later in class)
    'depend on this entry to determine where to insert new sets
    Sheet.Cells(1, 1) = 0  'for 0 associations
    Sheet.Cells(1, 2) = 0  'for 0 set members
    Sheet.Range("A1:B1").Select
    App.Selection.Font.ColorIndex = 3  'change color to red

    Sheet.Cells(1, 3) = "{View}"
    Sheet.Range("A1:C1").Select
    App.Selection.Font.Bold = True  'make bold
    App.Selection.ColumnWidth = 2  'column width

    'create the table and insert the first row that we just wrote into the
spreadsheet
    Set SetTable = New Table
    SetTable.InsertRow "View", 1, 3, "V"
End Sub


Private Sub Class_Terminate()
'
'Purpose:  Quit the application and free all allocated space
'Arguments: Filename, full path: where to save workbook

  'free allocated resources
    Set App = Nothing
    Set Book = Nothing
    Set Sheet = Nothing
    Set SetTable = Nothing
End Sub



Public Sub NameSheet(Name As String)
'
'Purpose: Assigns a name to the Active sheet

Dim CurPos As Integer
Dim StartPos As Integer

    CurPos = Len(Name)
    Do Until CurPos < 1
        StartPos = InStr(CurPos, Name, "\")
        If StartPos <> 0 Then
```

```
            Exit Do
        End If
        CurPos = CurPos - 1
    Loop

    Sheet.Name = Mid(Name, StartPos + 1)

    Sheet.Name = "{" + Sheet.Name + "}"

    ActiveSheet.Select
    ActiveWindow.Zoom = 75

End Sub



Public Sub Save(Filename As String)
'
'Purpose:  save the file and close the book
'Arguments: Filename, full path: where to save workbook

    Mid(Filename, Len(Filename) - 2) = "xls"
    Book.SaveAs Filename

End Sub



Public Sub InsertSet(set_name As String, association As String)
'
'Purpose:  Insert a new set into the Jmap
'Arguments: set_name, name of the set, not including curly brackets

    If SetTable.Exists(set_name) = True Then
        Exit Sub
    End If

    Dim row As Integer, Column As Integer
    NewCell row, Column     'get next available place

    'write the set_name in the sheet and make it bold
    Sheet.Cells(row, Column) = "{" & set_name & "}"
    Sheet.Cells(row, Column).Select
    App.Selection.Font.Bold = True

    'insert a value of 0 next to the set, meaning 0 set members
    'then make bold and red
    Sheet.Cells(row, Column - 1) = 0
    Sheet.Cells(row, Column - 1).Select
    App.Selection.Font.ColorIndex = 3
    App.Selection.Font.Bold = True

    'left of the number of set members, onsert another 0 meaning 0 associations
    'then again make bold and red
    Sheet.Cells(row, Column - 2) = 0
    Sheet.Cells(row, Column - 2).Select
    App.Selection.Font.ColorIndex = 3
    App.Selection.Font.Bold = True

    'update our table
    SetTable.InsertRow set_name, row, Column, association

End Sub



Public Sub InsertSetMember(set_name As String, ByVal value As String)
'
```

```
'Purpose:  Insert a new set member into the given set
'Arguments: set_name, name of the set, not including curly brackets
'           value, a string for the set member value

    'cannot insert into a non existant set
    If SetTable.Exists(set_name) = False Then
        Exit Sub
    End If

    'if set exists make sure this is not a duplicate entry
    If FindSetMember(set_name, value) = True Then
        Exit Sub
    End If


    Dim row As Integer, Column As Integer
    SetTable.GetNamedIndexes set_name, row, Column     'get coordinates of set

    'add a new row under the set name
    Sheet.Cells(row + 1, Column).Select
    App.Selection.EntireRow.Insert

    'insert the value into the new row, 1 column to the right of the set
    Sheet.Cells(row + 1, Column + 1) = value

    'insert value of 0 for the number of associations
    '2 columns left of where we just put the set member
    Sheet.Cells(row + 1, Column - 2) = 0
    Sheet.Cells(row + 1, Column - 2).Select
    App.Selection.Font.ColorIndex = 3
    App.Selection.Font.Bold = False


    'update the counter for set members next to the set (add 1). it's 1 column to
the left of the set
    Sheet.Cells(row, Column - 1) = Sheet.Cells(row, Column - 1) + 1

    'update our table (we added a new row at row + 1)
    SetTable.ShiftDown row + 1

End Sub



Public Function FindSetMember(set_name As String, ByVal value As String, Optional
ret_row As Integer, Optional ret_col As Integer) As Boolean
'
'Purpose:  Checks if a set member is already part of a set, if yes returns the
index
'Arguments: set_name, name of the set, not including curly brackets
'           value, a string for the set member value

    ' value on error
    FindSetMember = False

    'cannot search into a non existant set
    If SetTable.Exists(set_name) = False Then
        Exit Function
    End If

    'set exists, get coordinates of set
    Dim row As Integer, Column As Integer
    SetTable.GetNamedIndexes set_name, row, Column

    'if we find the set return it's coordinates else return -1
    Dim I As Integer
    Dim limit As Integer
    Dim set_member_name As String
    limit = Sheet.Cells(row, Column - 1).value
    I = 0
    For I = 0 To limit
        set_member_name = Sheet.Cells(row + I, Column + 1)
        If StrComp(set_member_name, value) = 0 Then
            ret_row = row + I
            ret_col = Column + 1
            FindSetMember = True
            Exit Function
        End If
    Next I
End Function



Public Sub AddColumn()
'
'Purpose: adds a column to the bigenning of the worksheet

    Sheet.Cells(1, 1).Select
    App.Selection.EntireColumn.Insert
    App.Selection.ColumnWidth = 2  'column width

    'populate the blank columns with the correct association headings

    'update the values in the table
    SetTable.ShiftRight
End Sub



Private Sub NewCell(ByRef row As Integer, ByRef Column As Integer)
'
'Purpose: find the next free space to insert a set
'Arguments: row, places the value of new row where to insert
'           column, places the value of the new column where to insert

'gets location of last set
SetTable.GetLastEntry row, Column

'add the number of set members for last set member
row = row + Sheet.Cells(row, Column - 1)

'one more for a new row
row = row + 1

End Sub



Public Sub DoRowGrouping(Name As String)
'
'Purpose: Groups the rows under a set
'Arguments: name is the name of the set to be grouped

    Dim row As Integer, row2 As Integer, Column As Integer

    SetTable.GetNamedIndexes Name, row, Column  'find coordinates of set
    row2 = row + Sheet.Cells(row, Column - 1)   'calculate the limit
    Sheet.Range(Sheet.Cells(row + 1, Column), Sheet.Cells(row2, Column)).Select
    App.Selection.Rows.Group  'group
End Sub



Public Sub DoGroupSettings()
'
'Purpose: Edits the settings for the row groupings
```

```
    Sheet.Outline.AutomaticStyles = False
    Sheet.Outline.SummaryRow = xlAbove
    Sheet.Outline.SummaryColumn = xlLeft
    Sheet.Outline.ShowLevels RowLevels:=1
End Sub


Public Sub InsertAssociation(Name As String, ByVal Member As String, RelationType
As String, Name2 As String, ByVal Member2 As String, RelationType2 As String, View
As String)
'
'Purpose: Adds an association between two set members

    'find the view part of it. exit if it dosen't exist
    Dim row As Integer, col As Integer
    If FindSetMember("View", View, row, col) = False Then
        Exit Sub
    End If

    'extend the {View} header of the map
    Dim row_h As Integer, col_h As Integer, a As String
    SetTable.GetNamedIndexes "View", row_h, col_h, a

    'if statement to avoid overwriting and over counting
    If Sheet.Cells(row_h, 1) = "" Then
        Sheet.Cells(row_h, 1) = a
        Sheet.Cells(row_h, 1).Select
        App.Selection.Font.Bold = True  'make bold
        Sheet.Cells(row_h, col_h - 2) = Sheet.Cells(row_h, col_h - 2) + 1
    End If


    'insert a "v" for the set member of the {View}
    If Sheet.Cells(row, 1) = "" Then
        Sheet.Cells(row, 1) = "v"
        Sheet.Cells(row, 1).Select
        App.Selection.Font.Bold = False
        'Sheet.Cells(row, 1).Interior.ColorIndex = 8

        Sheet.Cells(row, col - 3) = Sheet.Cells(row, col - 3) + 1

    End If

'---------------- First part of the association -----------------'

    'get the first part of the association and extend the header
    SetTable.GetNamedIndexes Name, row_h, col_h, a

    If Sheet.Cells(row_h, 1) = "" Then
        Sheet.Cells(row_h, 1) = a
        Sheet.Cells(row_h, 1).Select
        App.Selection.Font.Bold = True  'make bold
        Sheet.Cells(row_h, col_h - 2) = Sheet.Cells(row_h, col_h - 2) + 1
    End If

    'insert the actual association
    FindSetMember Name, Member, row, col

    'find the correct column to add the association to
    Dim I As Integer, col2 As Integer
    col2 = 1
    For I = 1 To col - 3
        If Sheet.Cells(row, I) <> "" And I <> col - 3 Then
            col2 = I
            Exit For
        End If
    Next I

    If Sheet.Cells(row, col2) = "" Then
        Sheet.Cells(row, col2) = RelationType
        Sheet.Cells(row, col2).Select
```

```
            App.Selection.Font.Bold = False
            Sheet.Cells(row, col - 3) = Sheet.Cells(row, col - 3) + 1
        End If


    '---------------- Second part of the association -----------------'

        'get the second part of the association and extend it's header
        SetTable.GetNamedIndexes Name2, row_h, col_h, a

        If Sheet.Cells(row_h, 1) = "" Then
            Sheet.Cells(row_h, 1) = a
            Sheet.Cells(row_h, 1).Select
            App.Selection.Font.Bold = True  'make bold
            Sheet.Cells(row_h, col_h - 2) = Sheet.Cells(row_h, col_h - 2) + 1
        End If

        'insert the actual association
        FindSetMember Name2, Member2, row, col

        'use col2 from first part of association
        If Sheet.Cells(row, col2) = "" Then
            Sheet.Cells(row, col2) = RelationType2
            Sheet.Cells(row, col2).Select
            App.Selection.Font.Bold = False
            'increment the counter
            Sheet.Cells(row, col - 3) = Sheet.Cells(row, col - 3) + 1
        End If
    End Sub
```

A-3-2 Table

```
    'MADE the table object DYNAMIC

    'Option Compare Database
    Option Explicit

    Private max As Integer      ' maximum number of entries in the table
    Private next_free As Integer  'index of next available place in table

    Private Names() As String   'choose not to use variant for effency reasons
    Private association() As String  'determiones the type of association
    Private Indexes() As Integer 'stored [(row,column),(row,column)...]



    Private Sub Class_Initialize()
    '
    'Purpose: constructor type method used to initilize table.

        next_free = 0  'on creation index 0 is available
        max = 20

    'NOTE: indexed from 0 to n, (c++ style)
        ReDim Preserve Names(max + 1) As String
        ReDim Preserve association(max + 1) As String
        ReDim Preserve Indexes(2, max + 1) As Integer
    End Sub

    'does nothing but if needed add destruction code here
    Private Sub Class_Terminate()

    End Sub
```

```
Public Sub InsertRow(Name As String, row As Integer, Column As Integer, a As
String)
'
'Purpose: Inserts a row into our table, a row contains the name of a set and it's
coordinates
'Arguments: name = name of set
'           row = row index of set location
'           column = column index of set location


    If next_free = max Then
        'double the size of the array
        max = max * 2

        'NOTE: indexed from 0 to n, (c++ style)
        ReDim Preserve Names(max + 1) As String
        ReDim Preserve association(max + 1) As String
        ReDim Preserve Indexes(2, max + 1) As Integer

    End If

    'insert the actual data
    Names(next_free) = Name
    association(next_free) = a
    Indexes(0, next_free) = row
    Indexes(1, next_free) = Column
    next_free = next_free + 1

End Sub


Public Function FindIndex(Name As String) As Integer
'
'Purpose: Gets the array index in the table of the given set name.
'Arguments: name = name of set
'Returns: an integer which is the index of the given set name or -1 if not found


    FindIndex = -1 'return -1 if not found
    Dim I As Integer
    For I = 0 To max
        If StrComp(Names(I), Name) = 0 Then
            FindIndex = I  'means set name was found reset return to current index
        End If
    Next I
End Function



Public Function Exists(Name As String) As Boolean
'
'Purpose: Checks if a set already exists
'Arguments: name = name of set
'Returns: BOOL true if found false if not

    Exists = False 'return false if not found
    Dim I As Integer
    For I = 0 To max
        If StrComp(Names(I), Name) = 0 Then
            Exists = True  'means set name was found reset return true
            Exit Function
        End If
    Next I
End Function
```

```
Public Function GetNamedIndexes(Name As String, row As Integer, Column As Integer,
Optional a As String)
'
'Purpose: Takes the name of a set and places it's coordinates into the row and
column arguments
'Arguments: name = name of set
'           row = argument to place found row value
'           column = argument to place found column value
'Returns: 0 on success -1 on failure


    Dim I As Integer
    I = FindIndex(Name)
    If I <> -1 Then
        row = Indexes(0, I)
        Column = Indexes(1, I)
        a = association(I)
        GetNamedIndexes = 0
    Else
        Debug.Print "set name not found"
        GetNamedIndexes = -1
    End If

End Function




Public Sub ShiftRight()
'
'Purpose: increment all the column values by one used when we insert a column into
the spreadsheet
'Arguments:
'Returns:

    Dim I As Integer
    For I = 0 To max
        Indexes(1, I) = Indexes(1, I) + 1
    Next I
End Sub



Public Sub ShiftDown(row_index As Integer)
'
'Purpose:  shift all values under the given row
'Arguments: row_index = the row in the spreadsheet where you insert a new row
'           this forces all the entries in the table to be wrong
'           so must call ShiftDown to correct entries in the table

    Dim I As Integer
    For I = 0 To max
        If Indexes(0, I) >= row_index Then
            Indexes(0, I) = Indexes(0, I) + 1
        End If
    Next I
End Sub



Public Sub GetLastEntry(ByRef row As Integer, ByRef Column As Integer)
'
'Purpose:  puts the coordinates of the last set into the arguments
'Arguments: row = place the row index here
'           column = place the column index here
'

'    Debug.Assert next_free <> 0
```

```
    If next_free > 0 Then
        row = Indexes(0, next_free - 1)
        Column = Indexes(1, next_free - 1)
    Else
        Debug.Print "the table is empty"
    End If
End Sub


Public Function GetAllEntries() As Collection
'
'Purpose:  Returns a collection with the names of all the entries in the table

    Dim I As Integer
    Set GetAllEntries = New Collection
    For I = 0 To max
        GetAllEntries.Add (Names(I))
    Next I
End Function
```