Generic Data Mining Application

Dr Bruce E. Segee (email:segee@eece.maine.edu),
Binaya Acharya (email:bacharya@eece.maine.edu)
Department of Electrical and Computer Engineering,
Instrumentation Research Laboratory,
University of Maine.

Abstract

Using instrumentation and automated data collection technologies, it is possible to accumulate large amount of data. This data can be efficiently stored, sorted and retrieved using database software. However, processing data collected in a factory or in a research application can be time consuming and tedious. Sometimes, similar processing needs to be done to spot trends or relationship in the data. The Instrumentation Research Laboratory at the University of Maine developed a Generic Data Mining Application. This application uses the queries in a database via ODBC or OLEDB and processes the data appropriately.

Upon initialization, the application uses a special query to identify queries that are available and the type of processing to be performed. For example, a query may return data suitable for creating a Bar Chart. Other queries may return data for populating a spreadsheet. The application creates a menu of choices for the user based on available queries. When a user makes a selection, the appropriate query is run to retrieve data. In the case where the data is to be used in a bar graph, the results from the query are loaded into an Excel worksheet using OLE and are used to create the appropriate graph. Other cases similarly use OLE to handle the data plotting and manipulation. The result is an application whose functionality is automatically extended whenever a new query is added to the database. The application does not need to be recompiled to make use of this functionality.

1. Overview of the Problem

Essentially, data gathering is the first step in achieving understanding of an underlying process, or relationship. Using many available techniques and methods a large volume of data can be accumulated. Understanding data, however, is very different from merely collecting data. A flowchart is given in Figure 1 that implies that data gathering is not an end, but rather the means for developing a better understanding of a problem, process or relationship. In order to get meaning from the data, however, it must be "looked at" in a variety of ways. The "understanding" that results may be a complete understanding of

the process or relationship, but is more often an understanding of what additional data must be gathered. Data processing is often tedious, time consuming, and error prone and is an area that can benefit greatly from automated methods.

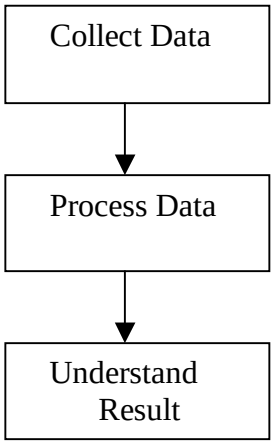

Fig 1. A Flowchart for Data Collection and analysis

2. The Need

What is needed is a set of tools that is capable of processing data in a manner that is not specific to the type of data. That is, in determining a relationship between two things, it should not matter what those two things actually are. At minimum, it is necessary to be able to quickly and easily see fundamental trends and relationships in the data. These can often be identified by the use of a bar graph display or an X-Y scatter plot. In addition to visualization, however, it is necessary to be able to get the data from an existing database quickly and easily, with a minimum amount of effort by the user.

3. The Software

With all the needs in mind we developed a relatively generic data mining application using Microsoft Visual Basic. In order to show the viability of this approach we developed the application to have three visualization tools, but added all of the necessary framework to allow many more to be developed in the future. Specifically, this application is capable of interfacing to a Microsoft Access database, running queries, and producing graphical output, using Microsoft Excel, consisting of a Bar graph, an X-Y scatter plot, or a double X-Y scatter plot. The data that can be viewed is not a function of the application, only the database, and it is possible to use exactly the same application to analyze the data from a wide variety of databases. The information on data retrieval is contained in the database itself. The application uses the results of database queries in order to customize its user interface at run time. Figure 2 shows the user interface presented by the application.

3.1 The User Interface

As can be seen in Figure 2, the user interface of the application is very simple. The "Change Database" button launches the standard "Open" dialog box allowing the user to select any Access database (the default is db1.mdb). Upon selection of a database, the form reloads itself and retrieves the list of Graphs that can be processed, from the special table "Graphs" in the database. If a database is not selected, the application uses the default database to retrieve the list of graphs in the "Select a Graph" combo box. When the user selects any of the available graphs, the application refreshes the text box containing the parameter name and the combo box containing the parameter list.

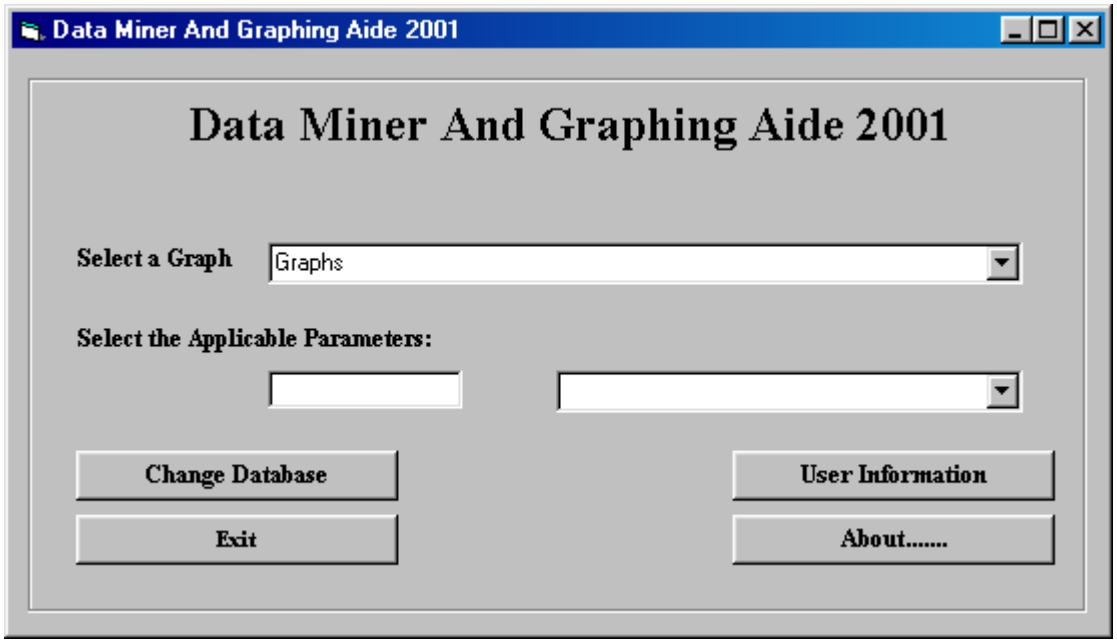

Figure 2. The User Interface

Both, the text box and the combo box retrieve the parameter name and the parameter list from the table "Graphs" in the database. When the user changes the graph name, in the "Select a Graph" combo box, the text box and the combo box containing the parameter list refreshes itself. The user can then select any parameter from the list in the combo box. If the selected parameter is no acceptable for a particular graph, a message box is displayed, but if the selected parameter is acceptable the application loads the graph form and the plotted graph.

We have developed a help file containing useful user information using Microsoft Word. A click on the "User Information" button launches word and loads an instance of this help file. This help contains useful information regarding the "Model Database" and the application. The remaining two buttons, the "Exit" and the "About…," exits the program and displays programmer information respectively.

3.2 Creating a Graph

After the user chooses a graph name (non-parameter passing query) from the “Select a Graph” combo box or after the user clicks on a parameter (parameter passing query), the application loads the graphing interface. This consists of an OLE container for Microsoft Excel Charts and a “Print Graph” button as shown in Figure 3. The OLE container displays the graph plotted by the application. Most of the functionability of Microsoft Excel can be used from the OLE container. This is a real help as the user can work on the graph to extract more information from the data using the features of Excel, such as drawing trend lines in a XY plot.

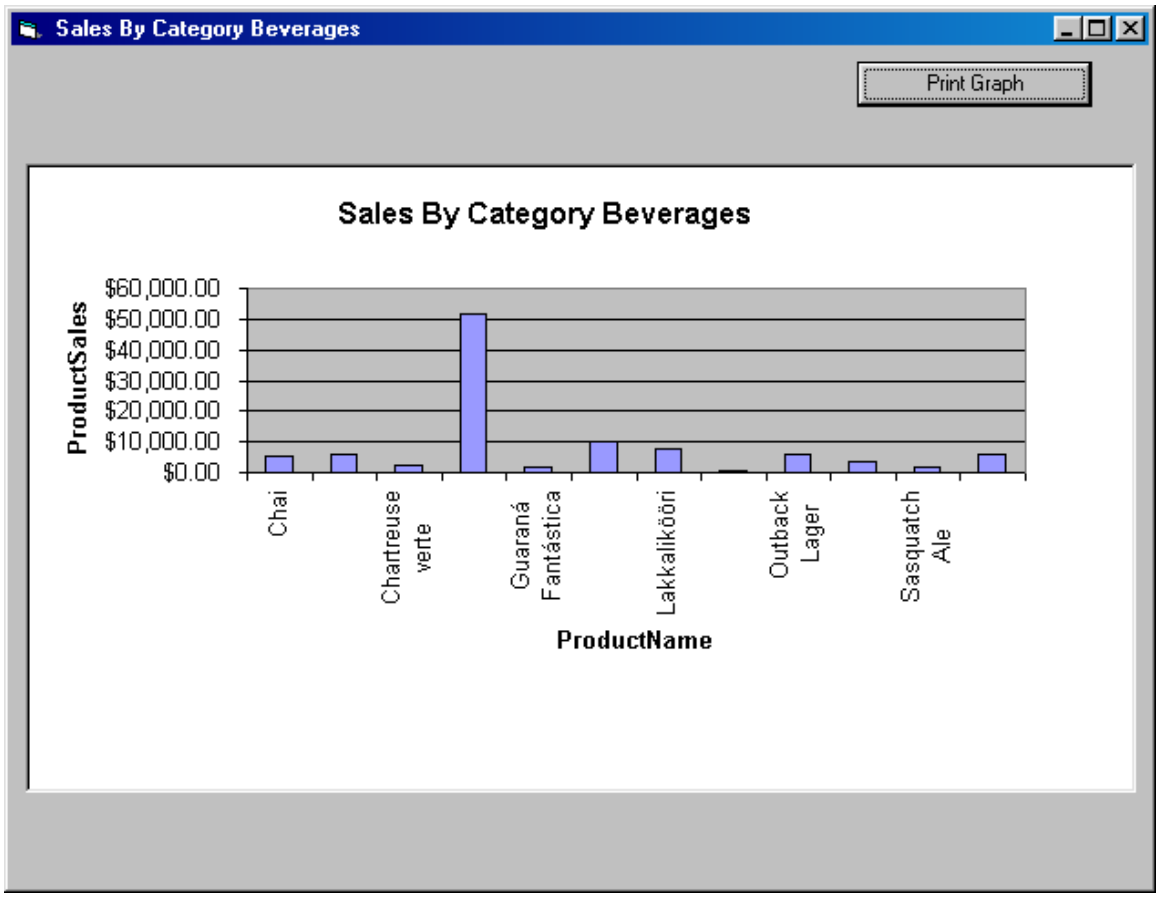

Figure 3 The Graphing Interface with a sample Graph

3.3 The Model Database

The default database for the application is “db1.mdb” and it resides in the same folder as the application. However, the default database can be changed at runtime by using the “Change Database” button. The application will run on any database provided that it has the special tables and follows the special query nomenclature used by our application.

3.3.1 The Special Table “Graphs”

The Access database must contain a table called “Graphs” (or this table must be added). The table may have an arbitrary number of records, each of which contain three fields as described below:

1. Graph Name
   This field contains the list of names of the queries to be processed. The names of the queries to be processed begin with a special character called the “Graph Type”. This application is capable of plotting three kinds of graphs -– the Bar Graph, the XY Scatter and a Double XY Scatter. There are three letters used to designate those types.
   “B” for Bar Graph
   “X” for XY Scatter
   “D” for double XY Scatter.

   The Graph Type is followed by the word “query”, a space and the name of the graph. For example: Xquery Sample Acceptability Trend per Day over Time. Here “X” is the query type.

2. Parameter
   This field contains the Parameter name that the query takes. The parameter names are preceded by “@”. If a query doesn’t take any parameters, the application expects a “@NA” (not applicable) instead of the parameter name.
   The parameter names must match the parameter names used within the query. Note that the SQL statements are case sensitive.

3. Parameter Source
   The parameter source is the name of query that returns a recordset containing acceptable parameters.

4. An Example

This section gives an example of using the program “Data Mining and Graphing Aide 2001” with the “Northwinds” Database distributed with Access by Microsoft.
Using “Sales By Category” query from the Northwinds Database we can make a query suitable for this application to plot a Bar Graph. The following simple SQL statements can be used to build the applicable query. Note that [@CategoryName] is the parameter that needs to be passed to the query.

SELECT [Sales by Category].CategoryName, [Sales by Category].ProductName, [Sales by Category].ProductSales
FROM [Sales by Category]
where [Sales by Category].CategoryName =[@CategoryName];

This query is named “Bquery Sales By Category”, since it produces a Bar Graph.

A query containing all the acceptable parameter values can be built with the following SQL statement.

SELECT DISTINCT ([Sales by Category].CategoryName)
FROM [Sales by Category];

And given the name "query category." This query can be given any name but whatever name is chosen must go in the "Graphs" table.

Similarly, we can use a non- parameter-taking query –"Order Subtotals", which is part of the database already, by simply changing its name. This query fits XY Scatter format so lets name it "Xquery Order Subtotals"

Figure 4 shows the "Graphs" table added to the "Northwinds" database. The fields in the table consists of the name of the query generating data suitable for graphing (with the graph type encoded in the name as described in section 3.3.1), the parameter name, if any and the name of a query returning valid parameter values.

.

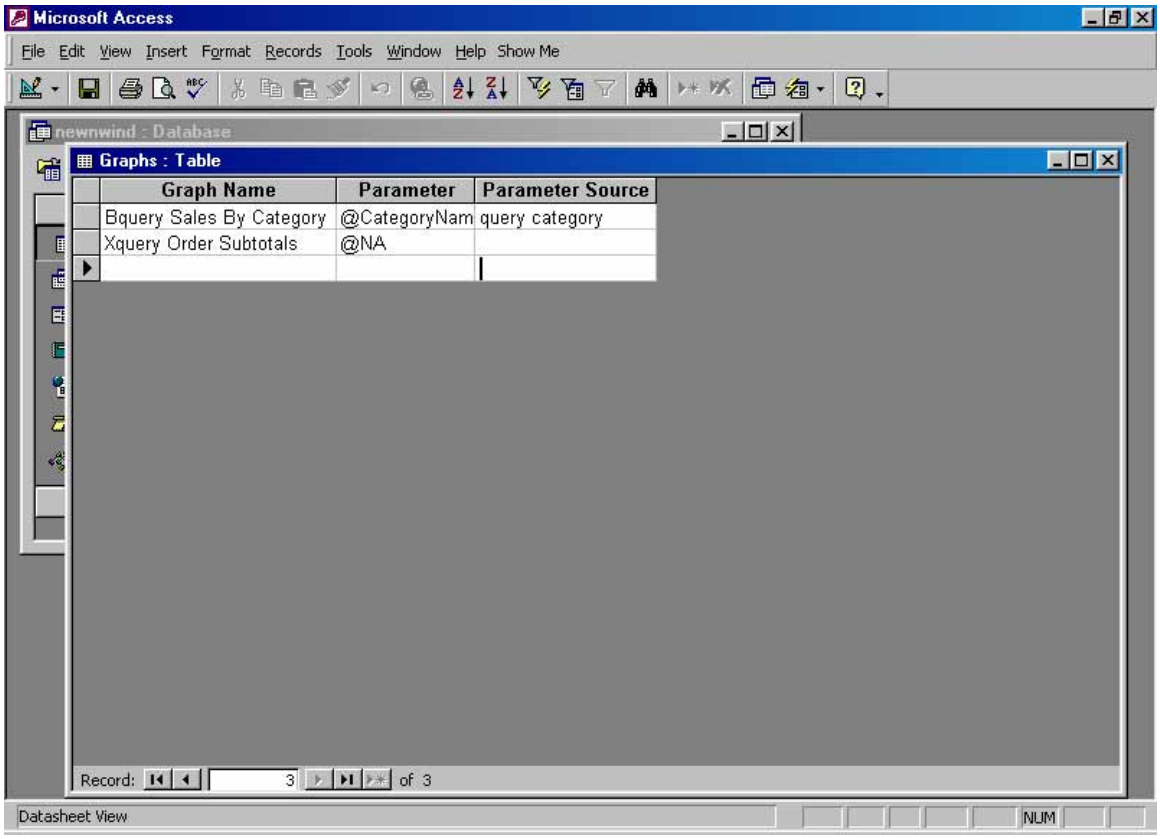

Figure 4 the special table "Graphs"

Figure 5 shows what the user sees when the modified "Northwinds" database is selected and a graph is clicked on the "Select a Graph" combo box. The name of the parameter

that the graph requires appears in the text box while all the parameter values are added to the combo box.

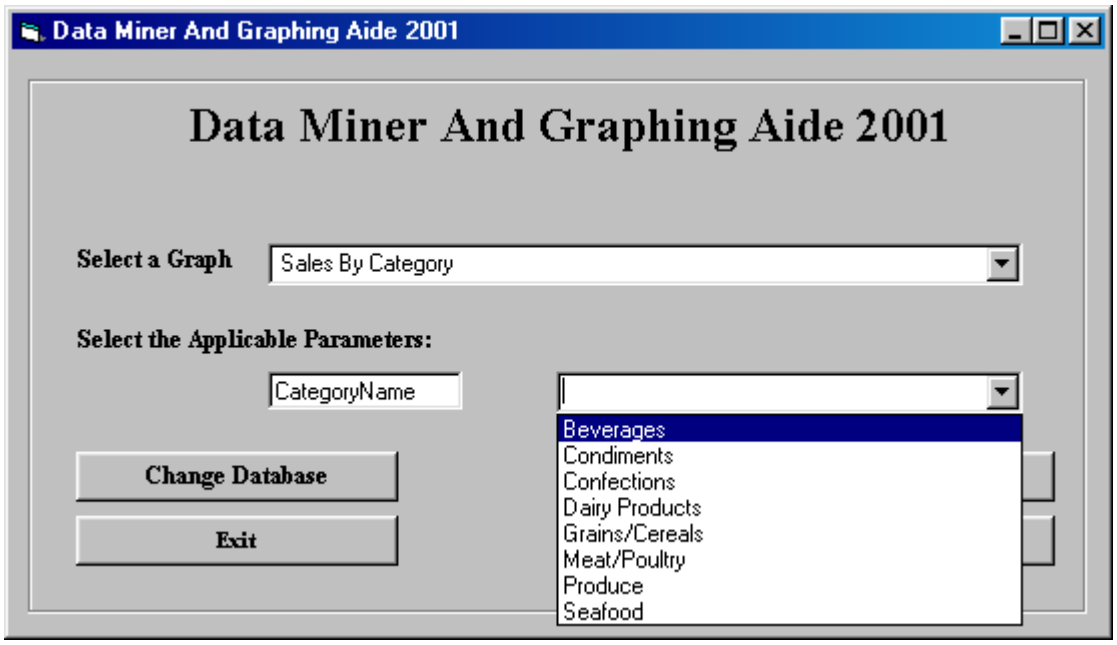

Figure 5. The User Interface with a parameter name and parameter list

Once, the user selects the right parameter value, the application loads the graphing interface displaying the graph as shown in Figure 3.

5.Conclusion

A collection of any data can be valuable only after we dig into it to process it and make sense out it. Graphical displays are usually effective in learning from the processed data. This whole data processing is tedious but the result will usually be fruitful. This generic data mining and graphing application can be a significant aide in processing and displaying data.

6.Bibliography

[Microsoft Corporation, 2000] "Retrieving and Modifying Data" MSDN Library, July 2000

[Microsoft Corporation, 2000] "Understanding Object Models" MSDN Library, July 2000

[James Hoffman, 1996-1998.] Introduction to Structured Query Language
http://www.netplus.bg/SQL/SQL%20tutorial/

[Microsoft Corporation, 2000] 'Visual Basic Documentation"
MSDN Library, July 2000

[Microsoft Corporation, 2000] "Microsoft Access Help"
Microsoft Access 2000

BINAYA ACHARYA
Binaya Acharya is currently enrolled as an undergraduate student at the University of Maine as an Electrical Engineering major. He is employed in the Instrumentation Research Lab, Department of Electrical and Computer Engineering at the University of Maine as a Research Assistant.

BRUCE E. SEGEE
Bruce E. Segee is an Associate Professor of Electrical and Computer Engineering at the University of Maine. His research interests include Instrumentation, Automation, and Intelligent Systems. He is the Director of the Instrumentation Research Laboratory and a Member of the Intelligent Systems Group at the University of Maine. His work focuses on real-world deployable systems for use in manufacturing environments. Dr. Segee received his PhD from the Department of Electrical and Computer Engineering at University of New Hampshire in 1992.