ANNALS of the ORADEA UNIVERSITY.

Fascicle of Management and Technological Engineering, Volume VII (XVII), 2008

DATA CONTROLS

Emil COSMA,
"Ovidius" University of Constanța,Faculty of Economic Sciences,
e-mail: ecosma@univ-ovidius.ro

Keywords: Server Explorer, Data Adapter, Binding Navigator, Binding Source

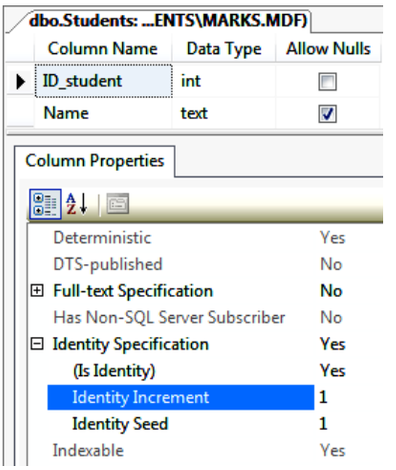
Abstract: The most important thing to know about databases is the fact that they are organized into tables. The software application used for data arrangement is called *relational database*. The SQL (*Structured Query Language*) is the most frequent utilized language program, designed for the management and retrieval of data in relational database management systems. *Visual Basic* controls within database play a significant role in establishing the relationships between the applications and the data. In order to initiate a database working session, this has to be activated. In order to connect to an existing database or to create a new one, the *Server Explorer* Window might be utilized

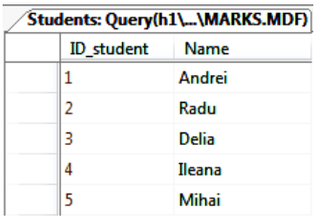
1. INTRODUCTION

Server Explorer is a visual interface utilized by the *Visual Studio 2005* which permits you to log on to servers (database servers for example), to obtain information concerning the structure and content of presented data within the databases server's context, and it also facilitates simple operations with databases servers. In order to connect to a database server, one may require the access from the administrator – with other words, an identification name and a password. A functional *SQL Server* version cannot be accessed by everybody – but almost every person who utilizes the *Windows* operation system may create a database, type *Microsoft SQL Server Database File* (by utilizing *Visual Studio 2005*) or *Microsoft Access Database File* (by running the *Access* application within the *Microsoft Office*).

2. EXAMPLE

- One may use the *Marks.mdf* database – a database type *Microsoft SQL Server* – which is comprised of two tables (*Students* and *Tests*). The values within the *ID_student* field will be automatically incremented inside the *Students* table (*Identity Increment,* fig. 1).





dbo.Tests: Table(h1\...\MARKS.MDF)

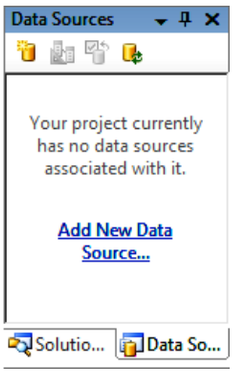
Column Name	Data Type	Allow Nulls
ID_student	int	☑
NrTest	char(1)	☑
Mark	numeric(3, 0)	☑

Tests: Query(h1\...\MARKS.MDF)

ID_student	NrTest	Mark
2	A	85
2	B	86
1	A	81
1	B	80
4	A	58
4	B	85
3	A	79
3	B	89
5	A	75
5	B	83

fig. 1

- The *Add New Data Source…* will be activated within the *Data Sources* window if a project, type *Windows Applications,* is opened in the display (fig. 2).



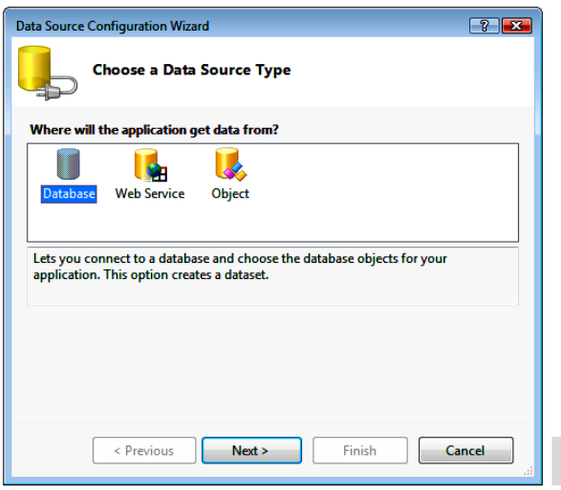


fig. 2

- One may press the *New Connection…* button within the *Data Source Configuration Wizard* window in order to select the database (*Marks.mdf* in our example, fig. 3)*:*

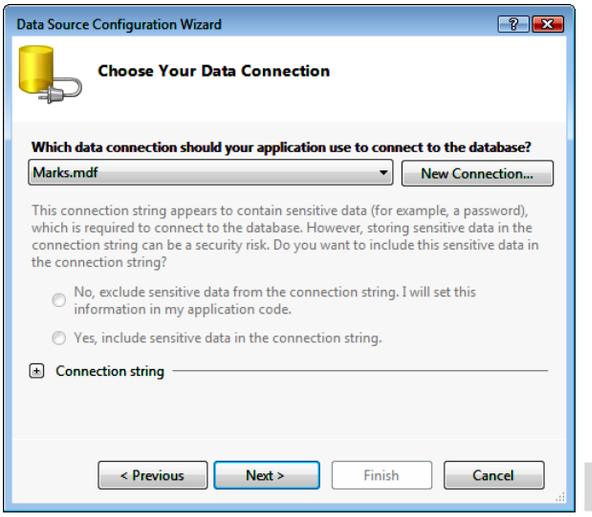


fig. 3

- One should select the elements which are going to be included in the data source (the tables called *Students* and *Tests,* fig. 4):

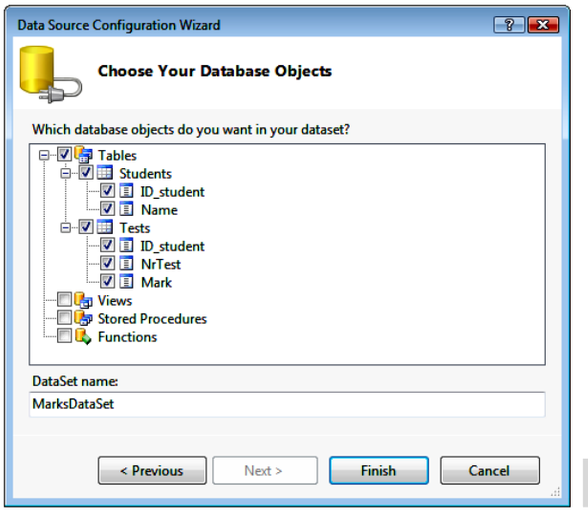


fig. 4

- When pressing the Finish button, the *Solution Explorer* and *Data Sources* will comprise (fig. 5):

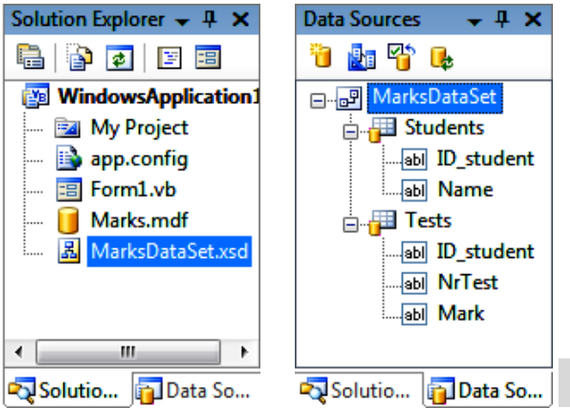


fig. 5

- The editor data set will be activated by double-clicking on the *.xsd (MarksDataSet.xsd)* extensive folder. The line between the two tables indicates the fact that those are connected through a "*one to many*" relationship. This indicates that every *ID_student* value appears once in the *Students* table and one or many times in the *Tests* table. The "*TableAdapter*" elements allow the program to move data from/in the data source (fig. 6).

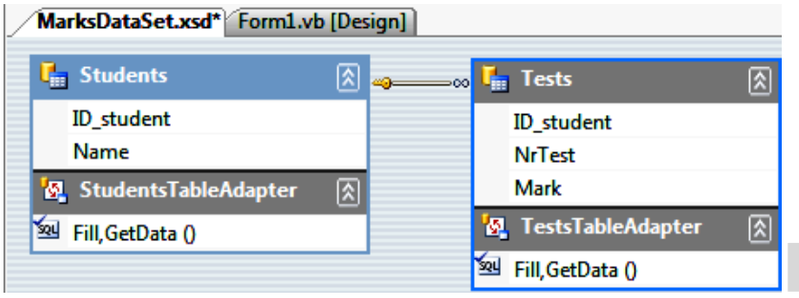


fig. 6

- The *Parent-Child* relationships between the following tables might be established by right-clicking on the connection line between the two tables or by selecting the *Edit Relation…* option from the contextual meniu (fig. 7)*:*

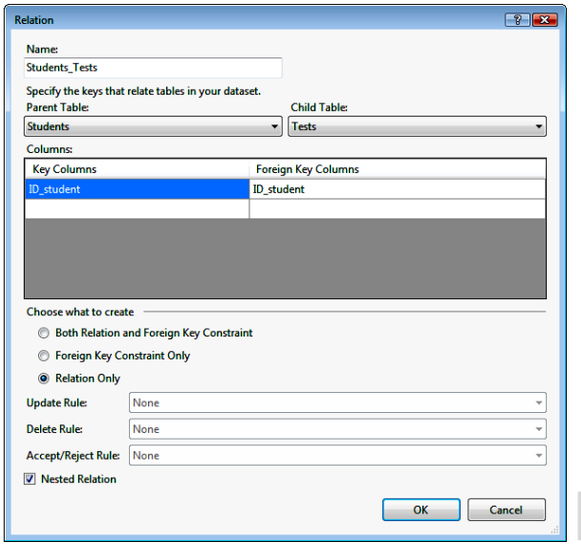


fig. 7

- Within *Data Sources*, the data set will be configured in the *Parent-Child* relationship (*Configure DataSet with Wizard…,* fig. 8):

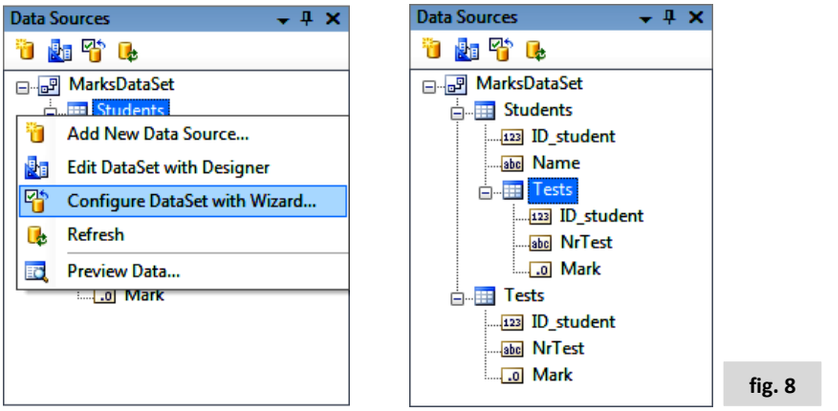


fig. 8

- *Visual Basic* creates automatically the **BindingNavigator** and the Da**taGridView** objects by dragging (using the *mouse*) the tables from *Data Sources* into the form. It also creates: **DataSet, TableAdapter, BindingSource** and **BindingNavigator** (fig. 9):

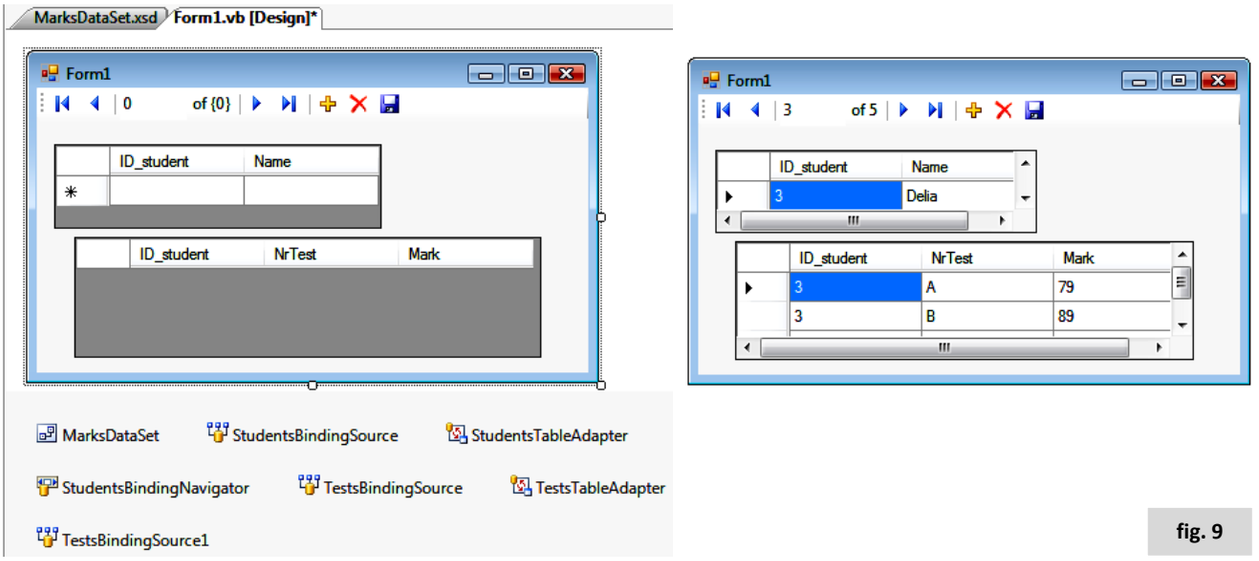


fig. 9

- *Right-clicking* on the table within *Data Sources* will modify its display style within the form *(DataGridView, Details,* fig .10*)*:

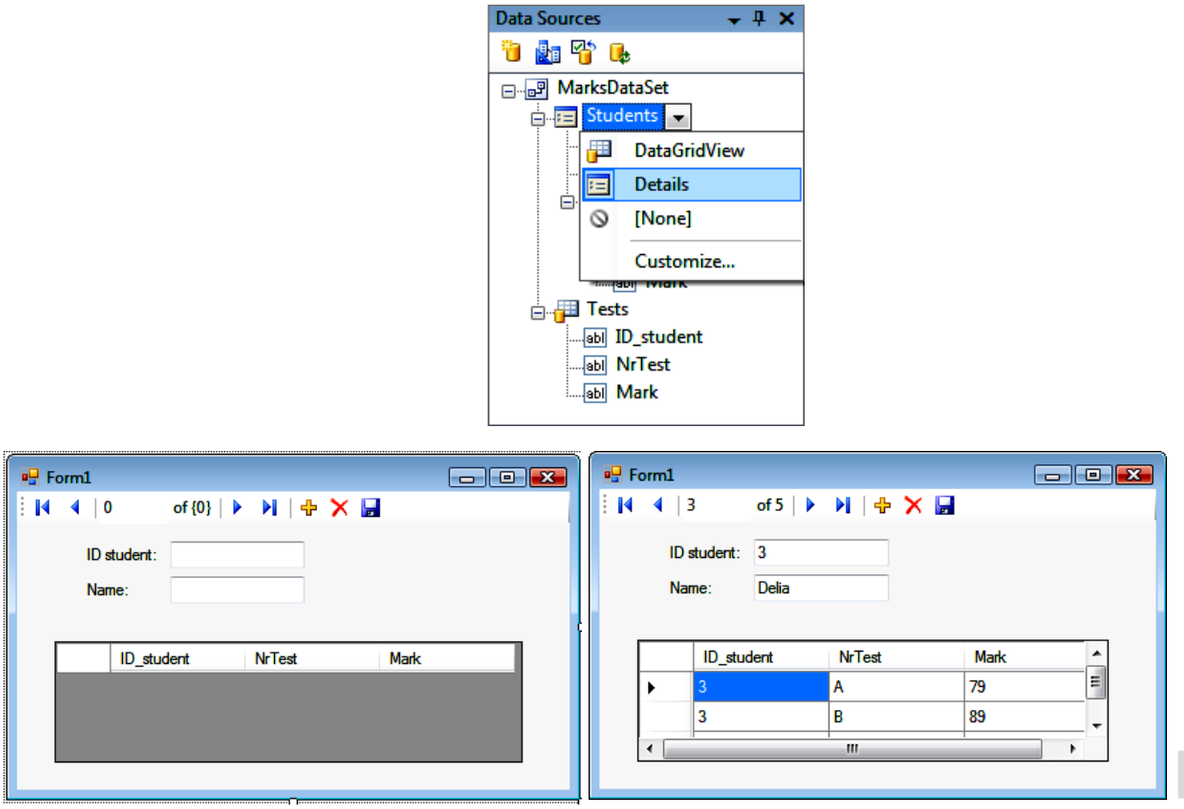


fig. 10

The program memorizes the data within the **DataSet** (fig. 11) object which represents the entire database.

The **TableAdapter** object copies data between database and **DataSet**. It also includes methods to facilitate the operations in relation to the database (the selection, the insertion, the update, the deletion of recordings).

The **BindingSource** object encapsulates all data from inside the **DataSet** and controls the data movement actions, as well as the procedures of elements' addition and deletion.

BindingNavigator offers the interface through which the user may interact with the data source.

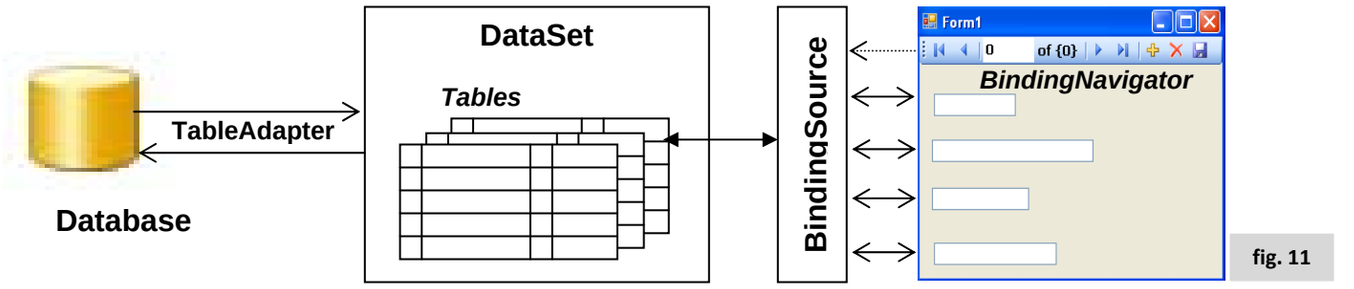


fig. 11

The *Visual Basic* generates automatically code sequences. When loading the form, these instructions intermediate the data copying (through the **TableAdapter**) from the database into the **DataSet**. The **StudentiBindingNavigatorSaveItem_Click** event (control from **Binding Navigator**) helps the tables' modifications saving in the database.

```
Public Class Form1

    Private Sub Form1_Load(ByVal sender As System.Object, ...
        'TODO: This line of code loads data into the
        'PunctajeDataSet.Teste' table. You can move, or remove
        'it, as needed.
        Me.TesteTableAdapter.Fill(Me.PunctajeDataSet.Teste)
        'TODO: This line of code loads data into the
        'PunctajeDataSet.Studenti' table. You can move, or remove
        'it, as needed.
        Me.StudentiTableAdapter.Fill(Me.PunctajeDataSet.Studenti)

    End Sub

    Private Sub StudentiBindingNavigatorSaveItem_Click(ByVal …

      Me.Validate()
      Me.StudentiBindingSource.EndEdit()
      Me.StudentiTableAdapter.Update(Me.PunctajeDataSet.Studenti)

    End Sub
End Class
```

3. DATA ADAPTER CONFIGURATION

The Adapter may use diverse methods for manipulating the objects inside database; it invokes implicitly the use of SQL phrases. Its configuration is realized as usual through the contextual menu (in *DataSet*, *right-click* on the *TableAdapter*, *Configure*…).

Example:

- *StudentsTableAdapter* Configuration (fig. 12):

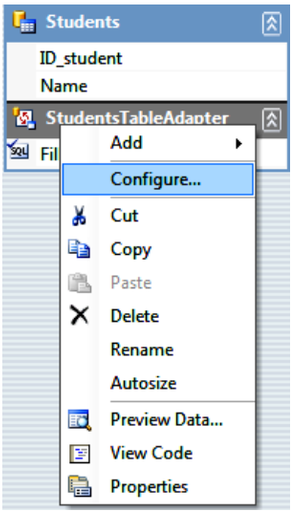


fig. 12

- In the following screen will be specified what *DataSet* effectivily contains (the *Id_student, Name* fields). One could also indicate the data set's format and content in two different manners:
  - using an explicite SQL interrogation which is written by the programmer in the

text box that is given by *Wizard*;

- using the *Query Builder* assistant through which the interactive format gets constructed. The interrogation within this context is pretty simple: **SELECT ID_student, Name FROM dbo.Students** (fig. 13).

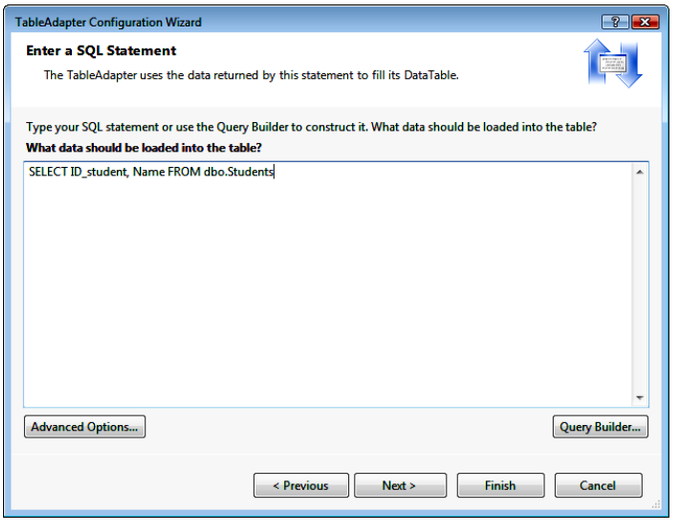


fig. 13

- When pressing the *Next* button, the insertion, modification and selection commands will be generated by the adapter (fig. 14).

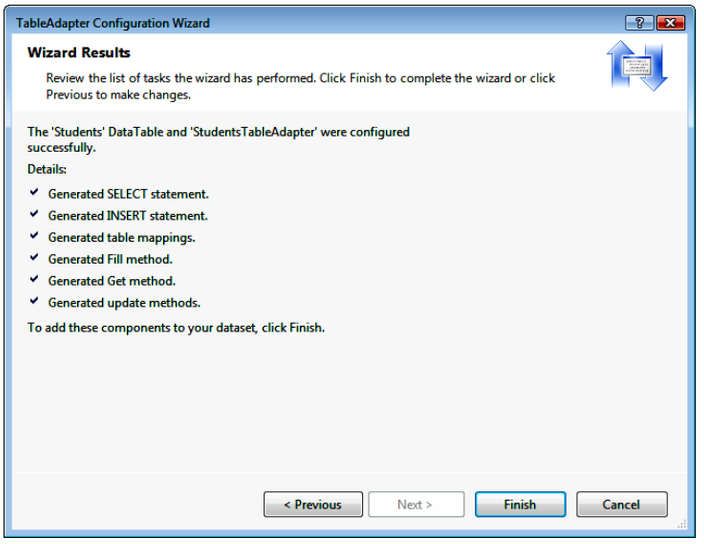


fig. 14

- The *InsertCommand* and *SelectCommand* properties, used for the content update, are to be found in the adapter Properties window. These commands may be visualized and modified depending on the application's interest (fig. 15).

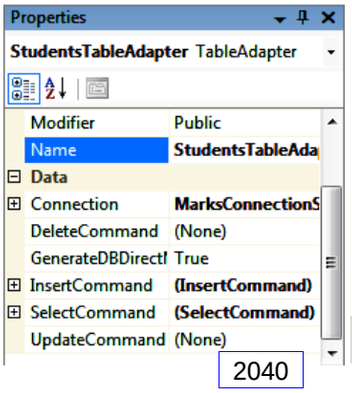


fig. 15

4. DATA SET MANIPULATION

When running the form, in order to fill the data set, one may use the *Fill* method founded at the adapter level. The data set content may be displayed by using the *Preview Data…* option from the contextual menu. This method receives as arguments the data set's name and the name of the table that is to be updated; in fact, the *Fill* method supposes: the attainment of a connection with the database; the SELECT phrase transmission which was specified by the programmer when he/she projected the adapter and the data set; the recordings receiving and the filling of the local data set; and finally the disabling of database connection.

Example:

- When preceding the **Click** event, one may determine the recordings number within the table by placing a command button on the form. These recordings numbers may be sequentially inspected, from the first to the last, the fields' values being also available.

```
Private Sub Test_Click(ByVal sender As System.Object…
 Dim i As Integer, name As String

 MsgBox("Total Rows: " & MarksDataSet.Tables("Students").Rows.Count)

 For i = 0 To MrksDataSet.Tables("Students").Rows.Count - 1

   name = MarkDataSet.Tables("Students").Rows(i).Item("Name")

   MessageBox.Show("Record " & i & " " & name)

 Next
```

- Even if it may be perceived as a difficult manner at the first sight, in order to grant access to the elements within the data set, one should demand a hierarchical class structure (collections and properties) that is to be represented in the following way (fig. 16):

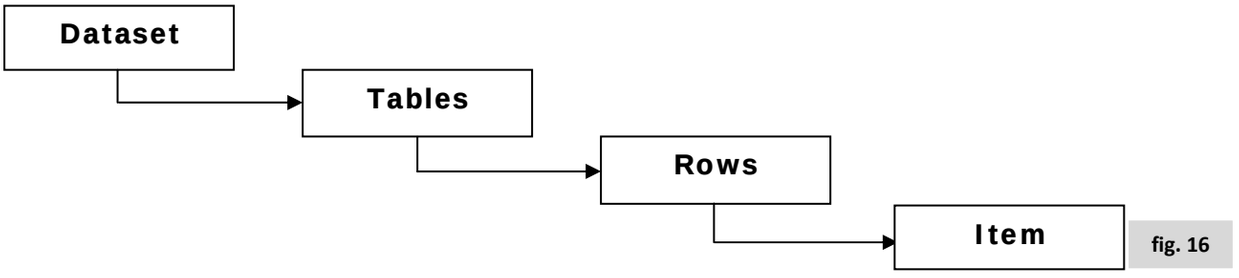
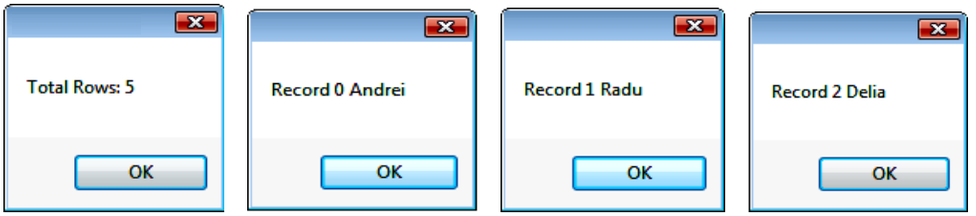


fig. 16

- The result of the previous sequence execution will be concretized as a suite of message cassettes whose number depends directly on the recordings number within the table.



▪ ▪ ▪ ▪ fig. 17

- The application steps should be followed in the prescribed order:
  - ÷ The form loading (*FormLoad*);
  - ÷ The execution of the **Form1_Load** event procedure where the connection with the database is established and where the *Fill* method is used for the filling of the local

data set.

- ÷ The database interrogation should be executed and the result is sent in the **MarksDataSet**.
- ÷ When pressing the *Test* button the recordings within the local set are examined and the fields' values are displayed.

- ✔ When referring to the data update, we should take into consideration the fact that the local set has been filled once when loading the form and it will behave itself as a variable which preserves the value established through assigning. As a consequence, in order to update the data, depending on the modifications that take place in the database, one should invoke again the adapter's *Fill* method.

REFERENCES

[1] Emil Cosma (2007), *VISUAL BASIC …VBA 2007 … Studio 2005,* Ed. MATRIX ROM, Bucureşti
[2] Harold Davis (2004), *VISUAL BASIC pentru Windows*, Ed. Corint, Bucureşti
[3] McFedries (2006), *VBA – Ghid pentru începători,* Ed. TEORA, Bucureşti
[4] Parsons Andrew (®2005), *Wrox's Visual Basic, Express Edition,* www.wrox.com
[5] Walnum Clayton (2003) , *VISUAL BASIC.NET,* Ed. B.I.C. ALL, Bucureşti