Visual Basic Data Access

By

Dave Jezak

Software Test Engineer, Microsoft Corp.

December 7, 2024

This document is intended as an overview of the information discussed at the VBits '94 Visual Basic seminar. It contains key points meant to show the reader what is available in this area of Visual Basic from a fairly high level. Additional information will be provided in the slide presentation and by way of sample application code.

1. Visual Controls

Visual Basic has a solid set of visual controls that enable the creation of data aware applications quickly and easily. It is entirely possible to create a simple data browsing application without ever writing a single line of code. The controls are also easily enhanced with the addition of code that performs such tasks as validation, formatting and error checking.

1.1. Data Control

The Data Control is the foundation for all other data-aware controls in Visual Basic. It has the ability to connect to one or more data sources and produce a set of records called a Dynaset. It can produce a Dynaset from a single table name, a saved QueryDef name in an Access database or any valid SQL statement.

1.1.1. Properties

There are six special properties associated with the data control (Connect, DatabaseName, Exclusive, Options, ReadOnly and RecordSource). They allow the user complete control over the Dynaset that will be created when the data control is loaded. The only two that are required to open a set of records are the DatabaseName and RecordSource. If either of these is missing, the data control will do nothing until some code places values in both of them and executes a refresh method.

The Connect property is used when the datasource is not an Access MDB file. It provides information that the OpenDatabase call needs to make the connection to the data. The other properties mentioned above map directly to flag parameters in the OpenDatabase and CreateDynaset calls available in the basic language.

All other properties are standard Visual Basic properties. The most useful is probably the caption property which may be used to display record or status information.

1.1.2. Methods

The two interesting methods are Refresh and UpdateControls. Refresh will cause the entire Dynaset to be recreated from scratch. The new Dynaset will include any records that were added or deleted by other users. It can also serve as a way to filter or sort the current Dynaset. By changing the RecordSource property and issuing a Refresh method, the Dynaset will take on the new criteria. The UpdateControls method is useful when you need to reset the data in the bound controls to the contents of the record in the Dynaset after a user has requested to cancel any changes made.

1.1.3. Events

There are three special events for the data control (Error, Reposition and Validate). They are used to add programming such as field validation, error handling, status and cancellation.

1.2. Bound Controls

Bound controls provide a mean by which many of the common data oriented operations take place internally so the programmer does not have to write code to perform them. These operations include displaying data, updating the Dynaset and moving from record to record. Visual Basic comes with many bound controls including the TextBox, Label, Picture, Image, CheckBox and some 3D controls. There are many third party controls available or being developed as well. These include spreadsheet, calendar, ListBox and other controls useful for viewing database result sets.

1.2.1. Special Properties

Each bound control must have DataSource property. This is the property used to connect the control with the Dynaset opened by the data control. The other property that a bound control might have is DataField. This tells the control which column in the Dynaset from which to get and set data. Some controls will not have a DataField property because they use the entire set of columns. An example of this would be a spreadsheet control that displays all columns from the same control.

1.3. Third Party Controls

Third party software developers are busy developing and selling custom bound controls utilizing the additions made to the CDK (Control Developers Kit) in Visual Basic 3.0. These additions supply the hooks to get at the data attached to a data control. These controls will not work without a data control.

2. Data Access Objects

The Visual Basic language has been greatly enhanced in version 3.0 to include all of the data access elements available in Access Basic. This allows the creation of any number of powerful database application.

2.1. DDL (Data Definition Language)

This area takes the place of Access's table-design UI. It enables programmers to add and delete tables, add fields and add and delete indexes. It is available on all backends with some exceptions specific to each type of database being used. It operates off the database object and has TableDefs, Fields and Indexes collections. Collections are simply sets of objects associated with a parent object. For example, the Fields Collection contains information about the fields in a table.

2.2. DML (Data Manipulation Language)

This set of commands allow programmers to browse, edit, add and delete records using the same source code on any type of database backend. It is all based on the Database object that is created when the OpenDatabase function is called. Hierarchically, below the database object lies recordset objects that come in three flavors: Table, Dynaset and SnapShot. Each has unique characteristics and similarly unique uses. Tables and Dynasets may be updatable while SnapShots are read-only. Tables objects are not available from ODBC data sources.

2.3. DML Optimization

Trying to decide which DML object to use can get a little tricky. The best way to decide is usually to perform some benchmarks using each object type that is available for the given task.

2.3.1. Tables

In general, the table object will be best for single table operations where changing and adding data is needed. The Seek method on the table object is always the fastest way to get to a given record because it makes direct access to the underlying table using a specified index. The biggest limitation of tables is that they contain the entire table.

With a table, only the indexed field's data is transferred into memory so loading data from a network is much quicker than bringing all the data. Then when a specific record is needed, the data is transferred to the client.

2.3.2. Dynasets

Dynasets create a local keyset index on your workstation. As you move from record to record, the actual data is fetched into memory. A Dynaset can be created to reference an entire table. In addition, you can specify limited views of table data based on a SQL statement. This way the data can be sorted or grouped in different ways and include only the columns you need to see. Dynasets can also be filtered and sorted after their

initial creation. Dynasets are usually updatable but may be read only for certain types of multi-table joins or some backends where the table has no unique index.

Only the first few hundred or so keys are loaded into memory when a Dynaset is originally created so focus will be returned to the app more quickly. Certain operations such as Movelast will cause all of the keys to be transferred into the memory of the local system. This makes subsequent moves quicker.

2.3.3. Snapshots

Snapshots are created by copying the data from the database into the local workstation memory. As with Dynasets, you can specify the fields to fetch and the sort order by using a SQL statement when creating the Snapshot. Data is not fetched until your application moves through the records (as with Movenext). Using the Movelast method will bring all selected data records into the workstation memory. Because of this, Snapshots are most appropriate for small read-only sets of data. The main differences between the Snapshot and Dynaset are that a Snapshot is always read-only and all data for each record is transferred into memory where a Dynaset may be updatable and only the keys are fetched until the record data is explicitly requested.

3. Crystal Reports

Crystal Reports for Visual Basic is a special edition of the standalone report writer modified to work optimally with Visual Basic. It contains two main parts, the design environment and the runtime environment. Crystal Reports is implemented as a custom control for ease of integration into Visual Basic applications. It uses a two pass report generator, enabling reports that are sorted on information obtained after a pass through the data has been done. For example, a report that is sorted by highest sale volume per office may be created where each office is a section or group in the report but they are ordered by their individual totals. Visual Basic applications cab specify record or group selection criterion and the destination for reports.

3.1. Creating Reports

Creating reports is fairly straightforward with the Crystal Reports design environment. It has a rich set of functionality that lends itself to creating everything from simple tabular reports to sophisticated grouped reports with graphics and other visual enhancements.

3.2. Running Reports from Visual Basic

Adding the ability to run reports from a Visual Basic application is quick and easy using the Crystal custom control. Setting the controls Action property to 1 will launch the runtime report previewer and also allow the user to send the report to the printer, a file, or the screen if desired.

The main stumbling block when using Crystal Reports is sending a query to the report to determine the criteria by which to select the records for the report. The Crystal syntax is not the same as the SQL found in Visual Basic so you must translate it into a statement that Crystal will understand.

4. ODBC Specifics

ODBC allows the connection to a growing number of database backends while maintaining the same base code.

4.1. Setup

Setup has been a very active area for users of ODBC based applications. It requires that the DLL files be made available and the INI file settings are correct. The ODBC.DLL and ODBCINST.DLL must be on the path along with the specific driver DLLs and support files. The ODBC.INI and ODBCINST.INI files must be present in the Windows directory and must contain information such as the following example:

ODBCINST.DLL
[ODBC Drivers]
SQL Server=Installed

[SQL Server]
Driver=SQLSRVR.DLL
Setup=SQLSRVR.DLL

This tells the driver manager what drivers are installed and what DLL to use when adding new data sources for that driver.

ODBC.INI
[ODBC Data Sources]
MySQLServer=SQL Server

[MySQLServer]
Driver=SQLSRVR.DLL
Description=SQLTest
Database=pubs
LastUser=guest
OemToAnsi=No
Network=dbnmp3
Address=\\MySQLServer\PIPE\SQL\QUERY

This information tells the driver manager what DLL to use to access the data, where it is located and the last accessed database and last user.

If this data is not intact, the user is likely to receive a "Data Source not found" error when attempting to open the database from within Visual Basic.

All ODBC INI files are generated by the ODBC Setup application and manipulated by the ODBC Data Source Name management utility launched from the Windows Control Panel. In addition, the RegisterDatabase function in Visual Basic may be used to add new data sources to the ODBC.INI file.

4.2. ODBC vs. ISAM

When using an ODBC driver, it may be more efficient to use an Access database with attachments to each ODBC table needed. This allows for much better connection management and ultimately much better Dynaset creation performance. The downside is that you must abide with the Access SQL syntax unless you want to make direct connection to the ODBC data for certain operations that require SQL passthrough capability. A major advantage is the ability to store and execute QueryDefs from the Access database as well as use Access itself to manipulate the data when it is more appropriate than Visual Basic.

4.3. Drivers

There are currently many drivers shipping with the recent release of the Q+E ODBC Pack that includes 12 drivers in a single package. Many other companies are busy developing drivers and gateways to various backends with most major backends accounted for within the next 6 months. A catalog is available from Microsoft at 800-426-9400.

4.4. Levels

Functionality in an ODBC driver depends on the level to which it conforms with the overall ODBC specification in two areas, API support and SQL Grammar. The higher the level, the more functionality a driver will be able to provide. SQL Grammar is more important to Visual Basic as it is capable of performing most DML functions with minimum level drivers while it may have

severe limitations in the DDL area. SQL Grammar levels range from minimum through extended. If you are having trouble doing certain things with a particular driver, it may be due to the level of conformance. A simple example is that the minimum level does not support Alter Table so adding fields to existing tables is impossible

5. Common Pitfalls and Problem Areas

There are several common areas where users may run into some confusion trying to achieve the behavior that they need in their applications. They are presented in no particular order.

5.1. Using the Standard Grid Control

5.2. ListBoxes

5.3. Linking Unbound Controls

5.4. "No Current Record" Error

5.5. ODBC Setup

5.6. Non-Updatable Dynasets

5.7. Defining Queries

5.8. Crystal Reports Syntax

5.9. SQL Passthrough

5.10. DML Methods (Table, Dynaset, SnapShot, Execute, ExecuteSQL)