

# DRAFT

Input regarding this document appreciated - submission information available at:  
<http://www.borland.com/events/cdrom.html>.

## Introduction to the Database Components in C++Builder

### Introduction

C++Builder offers developers unparalleled power and flexibility for application development. In particular, the Borland Visual Component Library's built-in database components provide an incredibly easy-to-use set of tools for database application development. For a large part of the development process, the programmer simply points and clicks to set options for various components. In fact, as we'll see, a Client/Server one-to-many database form can be created rapidly and with little effort.

Database components also offer versatility. The components can be programmed to respond to events, and options can be set at runtime which transform a simple application into a very professional one. This paper will take an in-depth look at some of the components and tools through which C++Builder supports database application development. Specifically, the following issues will be covered:

1. A review of the C++Builder database development philosophy
2. Introduction of the TDataSet, TTable, TQuery, and TDataSource components
3. Demonstration of the "data-aware" controls including TDBEdit and TDBCombobox
4. Review the power and flexibility of the DBNavigator component
5. Discussion of the Database Form Expert
6. Demonstration of parameterized queries

All discussion will be occur in the context of a Client/Server environment to assist the reader of this paper in quickly developing a basic understanding of C++Builder's Client/Server development capabilities.

### C++Builder Database Development

C++Builder is a general purpose Windows development tool. It can be used to create standalone EXEs, DLLs, drivers, games - virtually anything. C++Builder's flexibility is due to its powerful language, C++, and its rich set of tools. Many other development environments are specifically targeted towards a particular type of development. C++Builder offers flexibility.

One of the strongest feature sets in the C++Builder development environment is its database application development toolset. C++Builder offers the corporate developer, consultant, or hobbyist an extensive array of tools and utilities geared specifically towards database applications. Most significantly, C++Builder offers database development related components. Components are reusable, generic objects that are used and/or built in C++Builder and are incorporated into C++Builder applications. Components can be compared to OCX and VBX controls, which may also be manipulated within a visual environment, but which provide less integrated, and often more limited, capabilities.

A major difference between VBX controls and components is that components are compiled into the final executable that C++Builder produces, while VBXs remain separate and must be distributed with the application. Thus, performance can suffer. Many "data-aware" VBX controls are great for developing very simple applications. They provide a means for developing or rapidly prototyping database applications. Many experienced developers of more sophisticated applications will choose, however, to call native APIs rather than use such controls. This can provide a more stable and much faster application, if done properly.

A significant drawback to this approach is the amount of code required to create even simple applications. In addition, only more advanced developers have the knowledge and experience necessary to make direct API calls reliably.

C++Builder's data-aware components are different. In developing the Visual Component Library components, Borland has ensured that they are easy to use, flexible, and fast. Because C++Builder can create an executable, using the database components can provide the stability and speed of direct API calls.

C++Builder provides two types of database-related components: data aware controls and data access components.

- Data-aware controls such as TDBEdit (a edit region or enterable field) and TDBGrid (a spreadsheet like display table) provide the display of data on a form. They allow the user of the application to look at, enter, change, or delete data.
- Data access components include Dataset components such as TTable and TQuery. These fundamental data access components point to the actual data residing in a database file.

Additionally, data access components include TDataSource components which provide a layer of 'abstraction' on top of the Dataset components. C++Builder supports 'three-tiered' application development in which a data-aware control is linked to a data source component which is in turn linked to actual data via a table or query component. The middle tier in this approach, TDataSource, allows the actual table that a form is based upon to be changed at design time without any redesign of the application - all display elements are linked to TDataSource, not the underlying table or query. Similarly, the underlying table can be changed on the fly at runtime with only a trivial amount of code. Additionally a TDataSource component can makes it easy to provide essential functionality such as synchronized data across more than one entry form.

The topics below delve into each of these components in detail. A section on 'putting the components together' follows.

It should be noted that one additional 'component' contributes to the database development support of C++Builder - the underlying data engine, Borland Database Engine (BDE). BDE is a unifying technology in all Borland products: C++Builder and Borland C++, Delphi, IntraBuilder, Paradox, and dBASE. BDE allows the developer to gain access to Paradox, dBASE or any ODBC data source transparently. Using Borland SQL Links native access to many SQL servers is also supported. A data-centric application need only to know the name of an 'alias' for any database to have access to that database's contents. (Aliases are set up in the BDE Configuration Utility (BDECFG.EXE) and specify driver parameters and database locations.)

## **Data Access Components**

As noted above, the data access components provide a means for the developer to indicate where the data to be presented resides. These components include TTable, TQuery and TDataSource.

### **TTable**

TTable defines a link to a database table. With this tool, all the data from a table is available to the application. TTables are useful for most application needs. TQuery objects provide additional functionality necessary where a subset of data, or data from more than one table is required.

Placing a TTable component on a form requires the developer to specify some basic properties using the Object Inspector. Below are descriptions of the most common properties of the TTable Object Inspector:

**Active:** This property specifies whether the connection to the table is active during design mode or at runtime. One of the nicest features of the C++Builder database components is that as a developer, you can see the actual data in design mode. This makes it much easier to see what the form will look like at run-time.

**DatabaseName:** The database name is either a DOS path or a BDE alias where the database resides. An alias is always preferred over a hard-coded path. By using an alias you can change the physical location of the data, from a local drive to a network drive, for example, without having to recompile the application - simply change the alias Path property in the BDE Configuration Utility. If you are developing using a client/server database, an alias is necessary to specify the various parameters that the database server will require.

**IndexName:** By specifying the IndexName property, you can display the data in an order other than the primary key order, which, if there is a primary key, will be the default.

**ReadOnly:** This property limits write access to table data and can be used to control access rights at run-time.

**TableName:** The table name is the actual table in which the data resides. By dropping down the combobox in the property editor, you will see all the tables in the directory pointed to by the DatabaseName property. If you are working on a client/server database, you will see all the table names in the server's database pointed to by the alias.

### **TQuery**

TQuery is similar to TTable in that it points to the actual data table. Unlike TTable, TQuery allows the developer to specify which records to retrieve, which columns to retrieve, and whether the data resides in two or more tables. The means of doing this is through queries written in SQL, the industry-standard query language. With SQL, developers can write very sophisticated queries against their databases. C++Builder passes queries to the database's processing engine where they are interpreted and from which the results are returned to the application.

The key properties that must be considered for TQueries are somewhat different than those for TTables.

**Params:** Parameters can be used in a query in order to make the query more generic. The Params property allows the developer to specify the datatypes and default values of parameters used in a query. Additional information follows below.

**RequestLive:** BDE has the ability to create "live" updatable queries for databases that support the necessary pointer or cursor functionality.

**SQL:** The SQL property allows you to specify the SQL query. If you double click on this property, a List Editor window will appear which presents an editor. Simply type in the SQL command for the TQuery and click the OK button.

**Active:** Similar to the Active property of the TTable, the Active property of the TQuery opens the DataSet and allows the developer to see the actual data that is returned to the application from the query during design or runtime mode.

**DatabaseName:** This property is required in order for the query to know where the data necessary to resolve the query reside. An alias is preferred over a hard-coded path.

### **TDataSource**

As discussed earlier, the TDataSource component provides a link between the dataset objects and the visual display, or data-aware controls.

**AutoEdit:** This property specifies whether or not changes to a data-aware control automatically put the record into "Edit" mode. In most cases this property should be set to True, the default.

**DataSet:** With this property, the developer specifies to which dataset component this data source is linked. As we'll see later, each data-aware component has a DataSource property which would complete the three-tiered relationship. The DataSource property in the TTable and TQuery objects allows the developer to easily create one-to-many forms. By hooking the detail table's DataSet object to the master table's DataSource, C++Builder and BDE will automatically keep the master-detail relationship in sync.

## Data Aware Controls

The data aware controls provide the means for an application to display data from a table to the user. They also allow the user to change the data presented or enter new information. These data-aware controls are counterparts to the standard Windows controls. There are labels, edit controls, comboboxes, radiobuttons, and list boxes. In addition, a spreadsheet-like control called TDBGrid is provided. Below is a list of some of the data-aware controls and descriptions of their functionality. Controls which have many properties in common are discussed first. Section on TDBGrid, and s and DBLookupCombo follow. Lastly, the DBNavigator components are discussed.

### TDBEdit, TDBCombobox, TDBListbox, TDBRadioGroup, TDBCheckBox, TDBImage, TDBMemo

These controls are very self explanatory and represent data-aware equivalents of standard Windows controls. The key properties for all these fields are:

**DataField:** This is the actual field in the table or query. A drop-down list of field names is available from which to chose. This property should be set once the DataSource property is set.

**DataSource:** The DataSource property links a data-aware control to a TDataSource object. This establishes the tiered relationship between the DataSet object, DataSource object, and Data Control object.

**ReadOnly:** As implied, the developer can specify that a particular field is read-only. This gives much greater control than specifying write access for an entire table.

### TDBGrid

TDBGrid offers a spreadsheet-like look for presenting data. In many situations, the spreadsheet metaphor is appropriate. Most users are familiar with its look and feel, and for presenting large amounts of data in a small space, it can't be beat. The DBGrid is also useful when designing forms displaying a 'one-to-many' relationship. Often, the detail records can be easily presented in a DBGrid.

A description of the key properties follows:

**DataSource:** Like all the data-aware controls the DataSource property links the DBGrid control to the TDataSource object, which is in turn linked to a TTable or TQuery object.

**Options:** The Options property allows the developer to fine-tune the look and operation of the DBGrid. For example, the developer can specify if they wish to see Grid Lines, Column Title, and/or a record indicator, among other options.

### **TDBLookupList & TDBLookupCombo**

These two very powerful data-aware controls offer the C++Builder database application developer an easy way to display "lookup" data to the user. Lookups are usually small tables which contains all the valid values a certain field might hold. For example, in a Customer database, the customer's address is usually entered. The state in which the customer can reside can be a "lookup" to a States table. This will ensure uniform, clean data in the database. The LookupList and LookupCombo allow you to display the full description of the possible lookup values, but store the lookup code. This serves the dual role of ease of use and smaller database tables - it is nice to show the words "California", or "Massachusetts" to support the selection process of users but store 'CA' or 'MA' in the Customers State field. The difference is about 15 characters per customer record for the one field!

Along with the normal data-aware control properties of DataField and DataSource, the properties that allow you to perform the special lookup functions are:

**LookupDisplay:** This is the field in the lookup table that contains the description of the lookup code. For example, part number 101 may have a description of "Size 10 Widgets". This property allows the developer to display the more intuitive description rather than the code.

**LookupField:** This property is the field in the lookup table that contains the lookup code itself.

**LookupSource:** This refers to a DataSource control that ties into the lookup table. Remember, to get information from the lookup table, it too will need a TTable (or TQuery) and TDataSource objects set up as well.

### **The DBNavigator Component**

The DBNavigator is an extraordinarily important and powerful component. This component allows the developer to add the now standard look and feel of 'VCR' controls to a form to allow users to move from record to record. In addition to the First, Last, Next, and Prior buttons, the DBNavigator also has options for Edit, Post, Insert, Delete, Cancel and Refresh! It provides the essential functionality every database application needs. Each button is optional and you can mix and match at will. For example, in a read-only form, you wouldn't want any of the editing type buttons. In a client/server application beware of allowing the user to click on the First or Last buttons as these could result in very long queries.

## **Putting it All Together**

Now that your familiar with all the components available to you for database access and manipulation, lets take a brief look at how these components can be put together to create a database form.

### **Three Steps to Design a Form**

Designing a C++Builder database form requires just three simple steps. First, place a DataSet object (TTable or TQuery) on the form and set its properties according to your preferences. Second, put a TDataSource on the form and set its DataSet property to reference the TTable or TQuery object from the first step. Third and last, drop your data-aware controls on the form and set their DataSource and

DataField properties. While your at it, drop on a DBNavigator so you can move from record to record. That's all it takes!

### **The Database Form Expert**

C++Builder provides a very powerful tool that can streamline even the minimal steps described above and allow you to build fully functional database applications with a few mouse clicks.

The Database Form Expert is a great utility that comes with C++Builder that can make the development of database forms a breeze. By following the prompts you can design simple single table forms or even one-to-many forms using the Expert. It even works on Client/Server tables! The results of the expert are neat and professional looking forms. They are the basis on which to build an application. Customization is, of course, fully supported whether involving adding code, error checking, or providing custom lookups. The Expert is extremely successful at meeting most needs for common database application development tasks.

### **Conclusion**

C++Builder offers the client/server database applications developer a plethora of tools, components and options. With this rich set of features, virtually any database application can be written in C++Builder.

Copyright © 1996. Borland International, Inc.