A Guide to FoxPro CursorAdapter Classes

Version: **1.6**
Author: **Stamati Crook**
redware research ltd
http://www.redware.com
Date: **29 January 2005**
Document: **cursorAdapter07.doc**

Contents

Copyright 2004, 2005

redware research ltd, 104 Tamworth Road, Hove BN3 5FH, England.
http://www.redware.com

This document is updated regularly and we would be pleased to send you an update if you register on our website. You will need this password: **`adapter.`** `We also have` other documents covering database development with Visual FoxPro and SQL Server and also Open Source web database applications available from our website and **`amazon.com.`**

Introduction

Visual FoxPro has excellent facilities for database application development utilising a client-server database management system to store data as a replacement for the native FoxPro table files. The advantages of this approach are too numerous to mention here and there are various FoxPro features available to the developer requiring access to client-server data:

- Remote Views created in a database container to reference a client server database query representing retrieved data as a cursor.
- Parameterised Remote Views that limit the amount of data retrieved from the server.
- SQL pass-through queries that allow commands to be executed directly on the database server with the results returned as a cursor.
- Table buffering and commit and rollback commands to control the timing of updates sent to the database server.
- Connection control allowing performance optimisation, asynchronous queries, and so forth.

These features have been available since version 3.0 of Visual FoxPro and are now supplemented with the CursorAdapter class (first available with FoxPro 8.0) that allows an object-oriented software object that accesses client-server data in a flexible manner. Some notable features include:

- Transparent definition of native FoxPro or client-server database offering the best way to migrate an existing application to client-server.
- Seamless integration with ADO and XML.
- Manipulation of an extensive range of properties compatible with existing remote views and connection properties.
- Integration into the data environment of a form.
- Event driven model allowing sophisticated client-side validation and processing.

This document complements the redware Visual FoxPro Client-Server Handbook which describes in detail the techniques and optimisations to access client-server databases using all the available features of Visual FoxPro through to version 7.0. The use of the **`CursorAdapter`** object is discussed in detail with regards to accessing client-server data (ADO and XML interfaces are not covered here).

If you are not familiar with making connections to client-server databases and the use of table buffering with Visual FoxPro you will need further study or purchase of the redware FoxPro SQL Server Handbook (available from www.redware.com). Optimisation features are also not covered in this document which will be integrated into the next edition of the Handbook.

CursorAdapter Class

The CursorAdapter class is new for Visual FoxPro 8.0 and provides an object-oriented base class for creating cursors that control access to data.

CursorAdapters work well with various different data sources:

- Data stored in native FoxPro format on your local disk drive or local area network servers.
- Client-Server data stored in a central database management system accessed using ODBC drivers.
- ADO recordsets created on the workstation or by a middle tier business component server.
- XML documents.

CursorAdapters can be created programmatically or using the CursorAdapter builder that forms part of the visual design tools for the DataEnvironment of a Form. Third party tools such as the **cabuilder** from **www.mctweedle.com** build CursorAdapter class libraries directly from existing local or remote databases.

There are several advantages of using CursorAdapter classes:

- Object oriented inheritance allows a base class to be defined with application specific properties and methods.
- CursorAdapters are suitable for implementing systems that need to operate with either local or remote data according to the installation.
- The event model allows for powerful validation and trigger functionality to be implemented on the client-side of the application (perhaps to support triggers without needing to program for different database management systems).
- CursorAdapter definitions can be created and stored in a Visual Class Library and can be added into a DataEnvironment for a Form in a similar manner to local tables or views defined in a database container.

Some disadvantages include:

- CursorAdapters are objects and the object variable needs to remain in scope for the data to be available.
- Visual Class Libraries have a limit of 255 characters for properties and these properties often need to be defined in a method of the class.
- Views in the Database Container have additional properties for the individual fields defined for the cursor.
- A cursor created by a CursorAdapter class is instantiated in program code instead of with the **USE** command.

CursorAdapters combine many of the best qualities of views defined in a database container with the flexibility of programmatic control and object oriented inheritance. Storing CursorAdapter definitions within a Visual Class Library in combinations with the builder tool allows visual design and persistence of the definitions (although this needs some improvement). The main benefit however is a single object oriented technique for accessing data from a variety of data sources comprising local tables, client-server tables, ADO recordset objects and XML documents.

Note: This chapter does not discuss interfacing a CursorAdapter object with XML or ADO.

CursorAdapter

A CursorAdapter uses a Connection to communicate with a client server database when accessing data through with ODBC. A **SQLCONNECT** or **SQLSTRINGCONNECT** command is issued to get a Connection handle and the **DATASOURCETYPE** and assigned to the **DATASOURCE** property defined as shown below.

The **SELECTCMD** property is given a command that is executed on the server to create a local cursor from server data when the **CURSORFILL** method is called.

```
lnHandle = SQLSTRINGCONNECT( ;
  'DRIVER=SQL Server;SERVER=(local);UID=sa;PWD=;DATABASE=pubs')
loPubs = CREATEOBJECT('cursoradapter')
loPubs.DATASOURCETYPE ='ODBC'
loPubs.DATASOURCE = lnHandle
loPubs.ALIAS = 'caAuthors'
loPubs.SELECTCMD=[select * from authors]
IF loPubs.CURSORFILL()
      BROWSE
ELSE
      ? 'Error'
ENDIF
```

Note: This cursor will be closed as the program ends and the variable holding a reference to the CursorAdapter goes out of scope.

An identical procedure is followed for local FoxPro data where the **DATASOURCE** is blank and the **DATASOURCE** type is set to **'NATIVE'**.

```
lcTable =  HOME()+'samples\northwind\customers'
IF SPACE(1) $ lcTable
      lcTable = '"' + lcTable + '"'
ENDIF
loCustomer = CREATEOBJECT('cursoradapter')
loCustomer.DATASOURCETYPE ='NATIVE'
loCustomer.ALIAS = 'caCustomer'
loCustomer.SELECTCMD=[select * from ] + lcTable
IF loCustomer.CURSORFILL()
      BROWSE
ELSE
      ? 'Error'
ENDIF
```

*Note: To avoid confusion it is best to use a different **alias** name to the name of the table as FoxPro also opens the original table in the data session.*

Setup Table Buffering

The **BUFFERMODEOVERRIDE** property of the cursorAdapter object can be used to set up table or row level buffering to be used in conjunction with the **TABLEUPDATE** and **TABLEREVERT** commands.

Remember also to:

```
SET MULTILOCKS ON
```

Use Parameters to Filter Data

Accessing client server data without specifying a filter is expensive as all the data must be retrieved from the server into a cursor on the local machine. A paramerised query may be specified in the **SELECTCMD** property to specify a selection of data.

The following example shows a CursorAdapter object retrieving an empty cursor by specifying the **NODATA** property when filling the cursor with the **CURSORFILL** command. This adds a **WHERE 1=2** clause onto the SELECT command and returns an empty cursor with zero records.

```
RELEASE loPubs
PUBLIC loPubs
lnHandle = SQLSTRINGCONNECT(;
  'DRIVER=SQL
Server;SERVER=(local);UID=pubsuser;PWD=pubspassword;DATABASE=pubs')
```

```
loPubs = CREATEOBJECT('cursoradapter')
loPubs.DATASOURCETYPE ='ODBC'
loPubs.DATASOURCE = lnHandle
lopubs.alias = 'caAuthors'

loPubs.SELECTCMD=[select * from authors where state=?lcState]
IF loPubs.CURSORFILL(.F.,.T.)
      BROWSE TITLE 'NODATA'
      ELSE
      ? 'Error'
ENDIF

lcState = 'CA'
IF loPubs.CURSORREFRESH()
      BROWSE TITLE [state=] + lcState
      ELSE
      ? 'Error'
ENDIF
```

An empty cursor might typically be created when initially running a form until the user can be prompted for some selection criteria. At this stage the parameters can be set and the **CURSORFILL** command reissued to populate the cursor as shown below. This process can be repeated as often as required.

```
lcState = 'TX'
IF loPubs.CURSORREFRESH()
      BROWSE TITLE [state=] + lcState
      ELSE
      ? 'Error'
ENDIF
```

The **SELECTCMD** property of the CursorAdapter can be updated as required provided that a **CURSORFILL** command is issued to repopulate the cursor rather than a **CURSORREFRESH**. This removes a major limitation of Database Container Remote Views that do not allow the where clause to be specified between queries.

```
lcWhere = [WHERE state='CA' AND contract=0]
lcSQL = [SELECT * FROM authors ] + lcWhere
loPubs.SELECTCMD = lcSQL
IF loPubs.CURSORFILL(.F.,.F.)
      BROWSE TITLE lcWhere
ELSE
      ? 'Error'
ENDIF
```

Fill a CursorAdapter using a Stored Procedure

It seems that the **SELECTCMD** property for a CursorAdapter can contain any command that can be executed against the server. The following example executes a paramerised stored procedure on the server and returns the results as a cursor:

```
lnHandle = SQLCONNECT('dsnpubs','pubsuser','pubspassword')
loPubs = CREATEOBJECT('cursoradapter')
loPubs.DATASOURCETYPE ='ODBC'
loPubs.DATASOURCE = lnHandle
lopubs.alias = 'caRoyalty'
lnPerCent = 40
loPubs.SELECTCMD=[exec byroyalty ?lnPerCent]
IF loPubs.CURSORFILL()
      BROWSE
      ELSE
```

```
      ? 'Error'
ENDIF
```

Update Data

The process for making a cursor updatable involves specifying properties to allow the system to automatically generate commands to execute on the server to update the changed records in the cursor.

The required properties are the same properties used to make a cursor created with SQL pass-through updatable and are the same as those set against a cursor created by a remote view:

- **TABLES** contains a list of the tables on the server to be updated.
- **KEYFIELDLIST** is a list of the fields that form the primary key of the tables.
- **UPDATEABLEFIELDLIST** is a list of the cursor fields that are updatable.
- **UPDATENAMELIST** is a translation from local cursor field names to the field names on the server with the correct table prefix.

The following example allows updates to occurs only on the **AU_LNAME** and **AU_FNAME** fields of the **AUTHORS** table:

```
lnHandle = SQLCONNECT('dsnpubs','pubsuser','pubspassword')
loPubs = CREATEOBJECT('cursoradapter')
loPubs.DATASOURCETYPE ='ODBC'
loPubs.DATASOURCE = lnHandle
loPubs.ALIAS = 'caAuthors'
loPubs.SELECTCMD=[select * from authors where state=?lcState]
lcState = 'CA'
IF NOT loPubs.CURSORFILL()
      ? 'Error'
ENDIF

loPubs.TABLES='authors'
loPubs.KEYFIELDLIST='au_id'
loPubs.UPDATABLEFIELDLIST='au_lname,au_fname'
* Primary key must be defined in updatenamelist
loPubs.UPDATENAMELIST = 'au_id authors.au_id, au_lname authors.au_lname,
au_fname authors.au_fname'

BROWSE TITLE 'au_lname and au_fname are updatable'
? TABLEUPDATE(.T.,.T.)
```

Note: The primary key fields must be specified in the UPDATENAMELIST property to allow the update commands to be automatically created by the system. They do not need to be updatable if the key values are automatically generated on the server when adding a new record.

The **ALLOWUPDATE**, **ALLOWINSERT**, **ALLOWDELETE** and **SENDUPDATES** properties are True as the default settings for each CursorAdapter object. These values can be changed to restrict the appropriate data modification operation.

The **WHERETYPE** property can also be set to specify the type of update command that is applied when records are changed. A good default value to chose is to set **Key and Modified fields** so that the system checks any modified fields for changes by other users but allows two users to change the same record provided they change different fields.

UPDATETYPE by default notifies the system to automatically generate a single update command to update records on the server. Change this setting to generate separate **DELETE** and **UPDATE** commands if required by your database server.

CONVERSIONFUNC is an interesting property allowing local field values to be converted with a FoxPro function immediately prior to updating the data on the server. This is particularly useful when data is stored as variable length character strings on the server but recognised as fixed length character strings in the local cursor. Specifying the **RTRIM** conversion function for each character field ensures that trailing spaces are removed. **STRCONV** is also useful for performing Unicode conversions.

Note: Adding spaces onto variable length character fields may cause unpredictable results when selecting records from the server which may now expect the correct number of trailing spaces when comparing values. The ODBC driver for SQL Server also allows you to override this property in the datasource configuration (do not set ANSI paddings on).

FoxPro automatically generates the update commands for any changes in the local cursor and controls their execution against the server system. This functionality works perfectly for both local and remote client server tables using ODBC. Properties such as **UPDATECOMMAND**, **UPDATEDATASOURCE**, and **UPDATEDATASOURCETYPE** allow detailed specification of update, insert, or delete commands if required.

Save a CursorAdapter as a Visual Class

The following code specifies an updatable CursorAdapter class programmatically and then saves the object into a Visual Class Library.

```
lnHandle = SQLCONNECT('dsnpubs','sa','')
loPubs = CREATEOBJECT('cursoradapter')
loPubs.DATASOURCETYPE ='ODBC'
loPubs.DATASOURCE = lnHandle
loPubs.ALIAS = 'caAuthors'
loPubs.SELECTCMD=[select * from authors where state=?lcState]
lcState = 'TX'
IF NOT loPubs.CURSORFILL()
      ? 'Error'
ENDIF

loPubs.TABLES='authors'
loPubs.KEYFIELDLIST='au_id'
loPubs.UPDATABLEFIELDLIST='au_lname,au_fname'
* Primary key must be defined in updatenamelist
loPubs.UPDATENAMELIST = 'au_id authors.au_id, au_lname authors.au_lname,
au_fname authors.au_fname'

loPubs.SAVEASCLASS('ca07','cauthors','authorsw - only fname and lname
updatable')
```

The CursorAdapter object can subsequently be opened using the **AUTOPEN** method which automatically calls **CURSORFILL**. The method simply fills the cursor but is the method used by a Form **DataEnvironment** to open a CursorAdapter.

```
SET CLASSLIB TO ca07
lcState = 'CA'
loAuthors = CREATEOBJECT('cauthors')
loAuthors.AUTOOPEN
loAuthors.BUFFERMODEOVERRIDE= 3
BROWSE
? TABLEUPDATE(.t.,.t.)
```

Use the CursorAdapter Event Model for Data Validation

CursorAdapter cursors have an event model that allows methods to be defined before and after major events such as filling a cursor or modifying or inserting data.

These events can be particularly useful for defining the equivalent of database validation and triggers in an object oriented fashion. Although is it perhaps better to encapsulate this functionality on the database server, this approach is useful if defining data validation rules on the client where a variety of database servers are used some of which may not allow data constraints or triggers to be specified. A validation rule can easily be defined by modifying the **BEFOREUPDATE** event method. The **DODEFAULT** command is issued to continue with the update if the current record satisfies the validation rule.

The following example prevents BILL being entered as an author's first name:

```
* BEFOREUPDATE
LPARAMETERS cFldState, lForce, nUpdateType, cUpdateInsertCmd, cDeleteCmd

* Do not allow update if firstname is bill
LOCAL llFailUpdateRule
STORE .F. TO llFailUpdateRule
LOCAL lcFirstName
STORE SPACE(0) TO lcFirstName
LOCAL lcField
STORE SPACE(0) TO lcField

lcFirstName = EVALUATE(THIS.ALIAS+[.au_fname])
IF TYPE('lcFirstName')=='C'
      IF ALLTRIM(UPPER(lcFirstName))= 'BILL'
            llFailUpdateRule = .T.
            WAIT WINDOW 'Sorry. You cannot have BILL as the first name.'
      ENDIF
ENDIF

IF llFailUpdateRule
      RETURN .F.
ELSE
      RETURN DODEFAULT(cFldState, lForce, nUpdateType, cUpdateInsertCmd,
cDeleteCmd)
ENDIF
```

*Note: The **BREAKONERROR** property defaults to allowing a CursorAdapter class to trap its own errors within an **ERROR** method. Set this property to **TRUE** to allow standard errors to be generated if there is a problem with your code.*

Overcome the Property Length Problem in a Visual Class

Classes stored in Visual Class Libraries have a limit of 2,048 characters for a property value. This is easily exceeded when defining your CursorAdapter Classes and a programmatic workaround is required:

```
text to This.SelectCmd noshow
select authors.ADDRESS, authors.AU_FNAME, authors.AU_ID,
authors.AU_LNAME, authors.CITY, authors.CONTRACT, authors.PHONE,
authors.STATE, authors.ZIP from authors
endtext
```

Note: This can be seen in action when using the CursorAdapter builder tool.

Attach an Existing Cursor

An existing cursor can be attached to a CursorAdapter object and the properties of the object manipulated as required. The following example shows a read-only cursor created with SQL pass-through attached to a CursorAdapter with the **CURSORATTACH** method and then specified as an updatable cursor.

```
lnHandle = SQLCONNECT('dsnpubs','sa','')
lnExec = SQLEXEC( lnHandle, ;
  [SELECT * FROM authors],'sptAuthors')

loPubs = CREATEOBJECT('cursoradapter')
loPubs.CURSORATTACH('sptAuthors')

loPubs.DATASOURCETYPE ='ODBC'
loPubs.DATASOURCE = lnHandle
lopubs.Tables='authors'
lopubs.KeyFieldList='au_id'
lopubs.UpdatableFieldList='au_lname,au_fname'
* Primary key must be defined in updatenamelist
lopubs.UpdateNameList = 'au_id authors.au_id, au_lname authors.au_lname,
au_fname authors.au_fname'
```

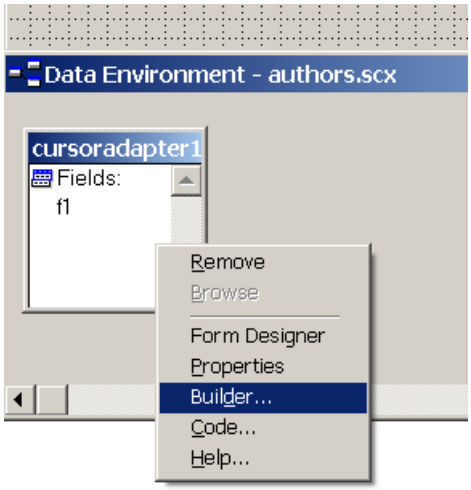
The **CURSORDETACH** method is used to detach a cursor from the CursorAdapter object. The object reference can then go out of scope but the cursor remains in the environment as a standard local cursor. The CursorAdapter properties will need to be specified again if the cursor is subsequently attached to a CursorAdapter object.

Note: Re-attaching a detached cursor is problematic as some of the original properties are lost during detachment.

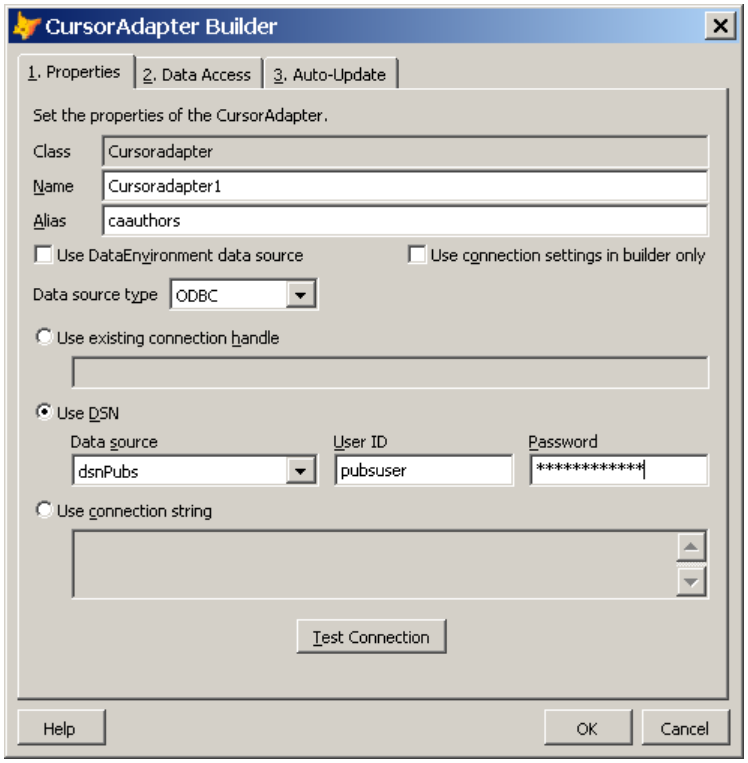
Create a Cursor with the CursorAdapter Builder

CursorAdapter classes are the new object oriented way to access both local and remote client server databases as well as access data using ADO RecordSets and XML.

An easy way to build a CursorAdapter object directly into a form is to create a new form and right-click to view the data environment. Ignore the initial prompt for a table and right-click to add a CursorAdapter. Right-click on the newly added CursorAdapter and select the builder:



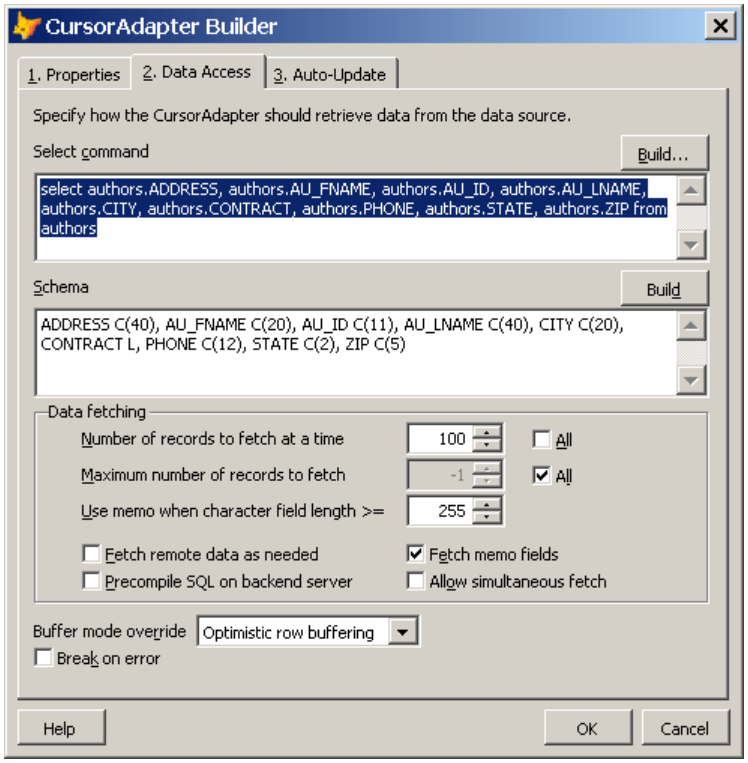
Name the alias and select the ODBC datasource type and define the connection characteristics. An existing workstation datasource is used below and the password and username specified.



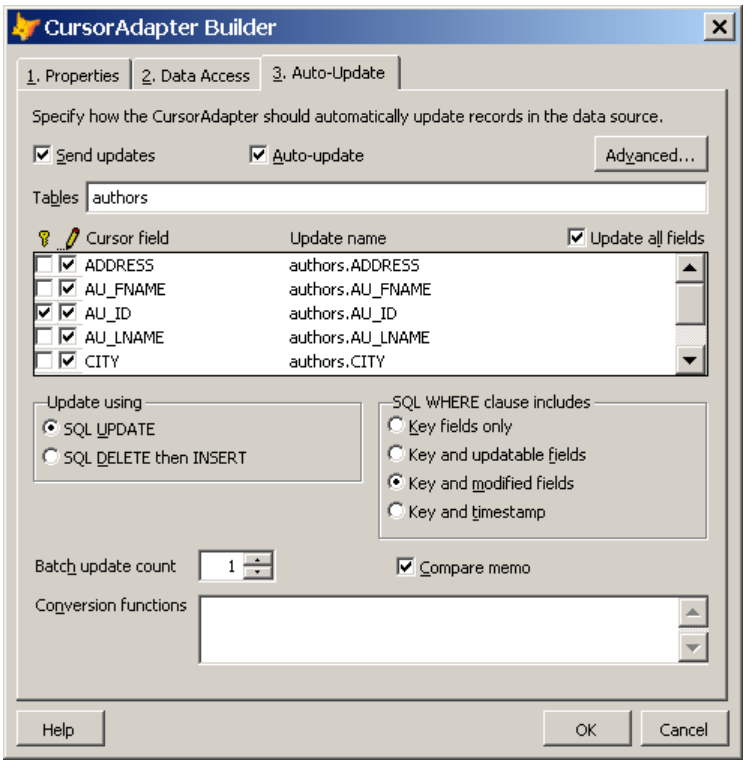
Click the **`Data Access`** tab and enter the following SELECT command:

```
SELECT * FROM authors
```

Now press the **`BUILD`** button just above the select command to select the fields required and automatically update the **`SCHEMA`** values. Select all the fields in the AUTHORS table:



Now select the **AUTO-UPDATE** tab to specify the **SENDUPDATES** and **AUTOUPDATE** check boxes and the updatable fields. Specify the **AU_ID** field as the primary key and remember to click the updatable checkbox for each updatable field.

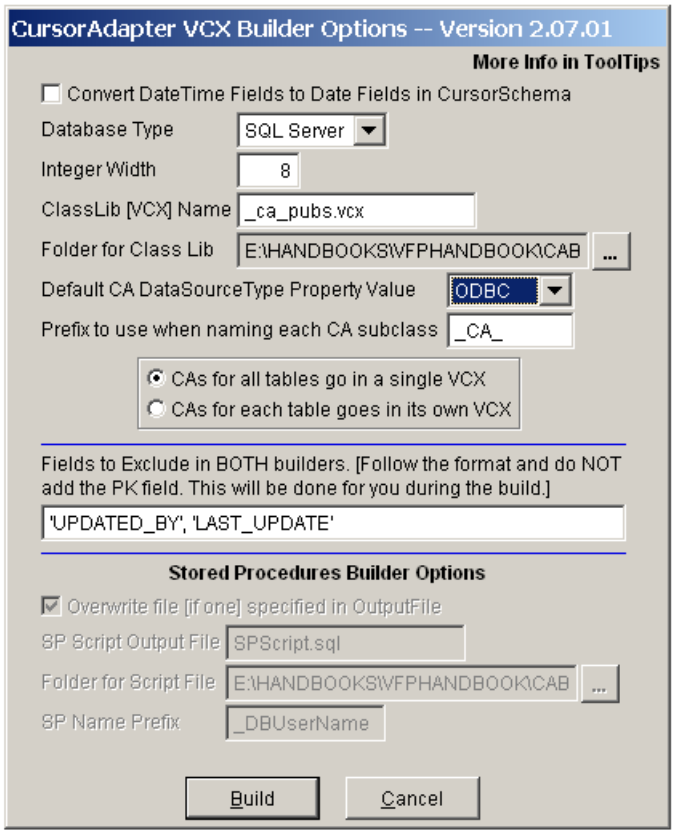


The CursorAdapter object is now complete and will create a cursor that updates the table on the client server database. Close the builder down and drag the fields onto the form to create a form in the usual way.

Build a CursorAdapter with cabuilder.prg

The CABUILDER.PRG utility from **www.mctweedle.com** is very useful for creating cursorAdapter visual class definitions from an existing client-server database. The utility is a free download from **M&J Software** at the above website and the following form appears after running the program:

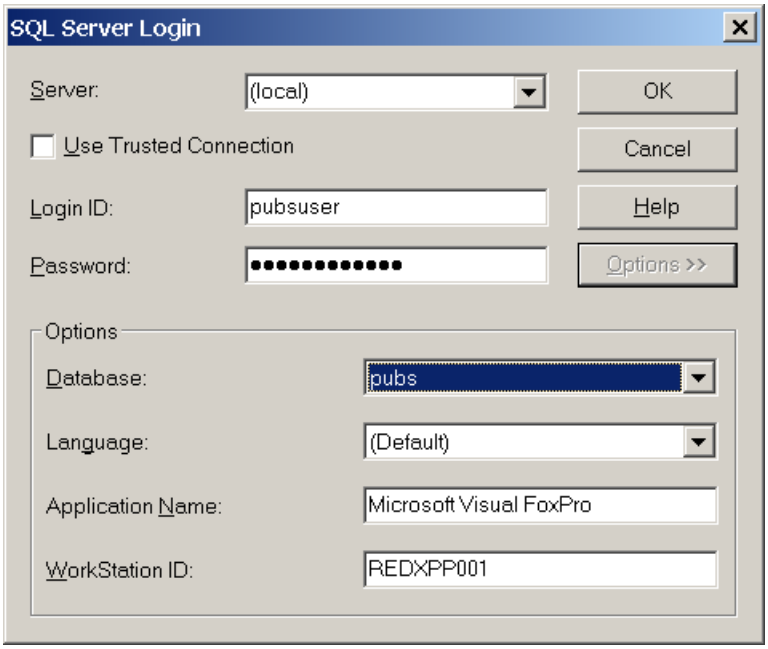
```
DO cabuilder
```



This example is creating a set of cursorAdapter visual classes from the pubs database with the following parameters set:

- Database Type is SQL Server.
- The DataSourceType is ODBC.
- The class library is *_CA_*PUBS and will be created in the `classes` subfolder.
- Each class will have a *_CA_ prefix.*

Running the utility by pressing the BUILD button prompts for the login details for SQL Server:



Copy the **_CA_PUBS** Visual Class Library to your program folder and instantiate the cursorAdapter for the **authors** table as follows:

```
RELEASE loAuthors
PUBLIC loAuthors

loAuthors = NEWOBJECT('_ca_authors','_ca_pubs')
lnhandle= SQLCONNECT('dsnpubs','pubsuser','pubspassword')
loauthors.DataSource = lnhandle

loauthors.cselectcmdfilter=''
loauthors.CursorFill
```

You can now browse the cursor and update values:

```
SET MULTILOCKS ON
BROWSE
? TABLEUPDATE(.T.)
```

*Note: You must **SET MULTILOCKS ON**.*

Conclusion

This concludes the overview of the cursorAdapter class and you are now in a position to:

- Programmatically define an updatable cursorAdapter object.
- Interface to native FoxPro and to ODBC client-server datasources.
- Save a cursorAdapter object as a Visual Class.
- Overcome problems with the text limit on a Visual Class property.
- Use the cursorAdapter builder.
- Integrate a cursorAdapter object into the DataEnvironment of a form.
- Use the CABUILDER tool from www.mctweedle.com to quickly create a set of cursorAdapter classes from an existing client-server database.

Please refer to the FoxPro documentation or to the redware FoxPro client-server Handbook for more information on:

- Defining asynchronous connections.
- Optimising and sharing connections.
- Optimising the performance of a cursorAdapter in relation to the ODBC Connection.
- Using cursorAdapters with XML or ADO (not in the Handbook).