A method for creating a simple images database application using Microsoft Access 2016 and SQL Server 2008R2

# Overview

This article describes a way to create an image management database and application using Microsoft Access 2016 and Microsoft SQL Server 2008R2. The method relies on SQL Server's ability to store binary data in a VarBinary(Max) data type. Images are stored in the database tables, rather than as individual files on a file system. The trick is to stream the binary data (1s and 0s) in an image file into a database table's VarBinary(Max) column. The advantages are numerous:

* Images and metadata can stored in the same record.
* All images and associated data travel together in a single database package.
* No need to worry about synchronizing image file paths with database records the way you would if you stored image metadata in a database but images in a file system.

There are some drawbacks, however

* The method described here requires a good deal of technical skill. Converting images to and from bit streams for everyday use requires a good deal of knowledge.
* SQL Server has no 'understanding' of what data is in a VarBinary(Max) column type. The data could just as easily be a PDF, a spreadsheet or an image file. To SQL Server your data is simply a long string of ones and zeroes.

If you think this method is for you please find in this repository a working example Access application and a create table query to generate the Images table used in the example. You can create the database table on your SQL Server and link it to the Access application using an ODBC data source.

I also describe the major points of the example and how it works. Let's start by creating the database table. I used SQL Server 2008R2 but any recent version will be similar.

# Create the database table

I started by creating an SQL Server database table to store my images.

CREATE TABLE [dbo].[Images](

[ImageID] [int] IDENTITY(1,1) NOT NULL,

[SurveyImage] [varbinary](max) NULL,

[Filename] [varchar](255) NULL,

CONSTRAINT [PK\_SurveyImages] PRIMARY KEY CLUSTERED

(

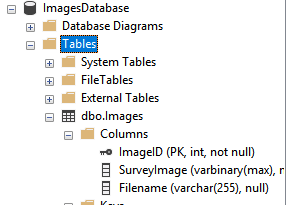
[ImageID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

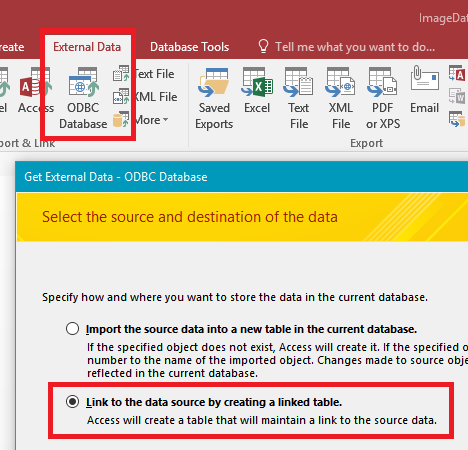
It looked like this when I was done:



The images will be stored in the SurveyImage (VarBinary(max) column.

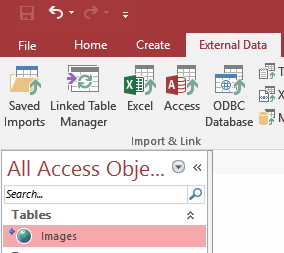
# Create an ODBC data source

In order to use Access as a front end to your database you will have to create an ODBC data source on your computer that links to your database. Use Microsoft's documentation to accomplish this task. The ODBC data source is the intermediate software that Access will use to communicate with your database.



# Create the Access application

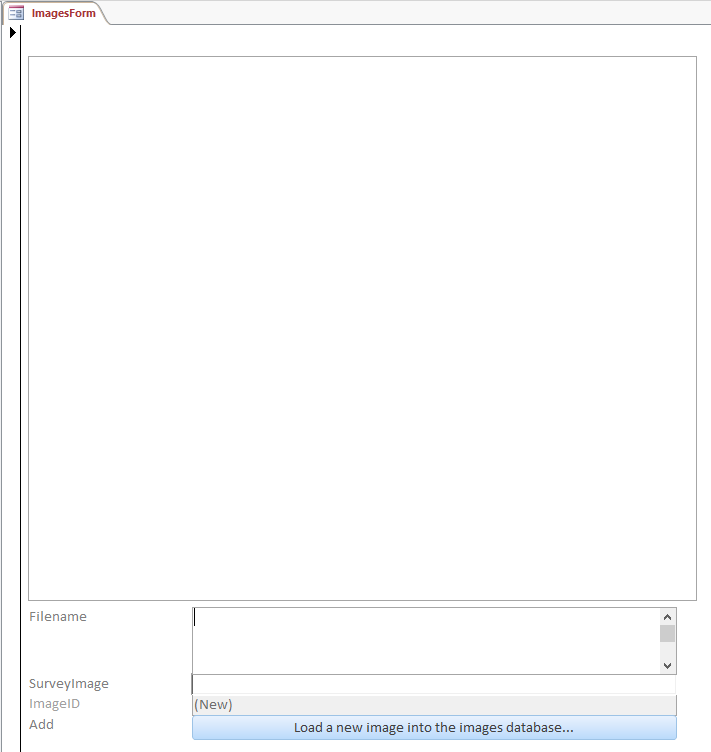
Next I created a new Access application that linked to my database table using the ODBC connection described above. I knew I succeeded when my Images database table showed up in the main Access interface.



# Create a form to show the images

The next step was to create a form that would allow me to view images, edit their metadata, delete them, if needed, and also to select images from my file system to load into the database.

My simple form looked like this:



The large control at the top is an ImageControl. The remainder of the controls are basic text controls. I’ll assume you have enough knowledge of Access forms to set this up so that the database fields link up with the controls. The ImageControl requires some explanation however. The trick in this form is to get the binary data in the SurveyImage field to render as an image in the ImageControl. Fortunately Microsoft has made this simple. The ImageControl has a .PictureData property which will do the conversion for us. All we have to do is create a Form\_Current event and pop the following code in it:

Private Sub Form\_Current()

'Set the picture control to pull the image from the SurveyImage field

Me.ImageControl.PictureData = Me.SurveyImage

End Sub

Now all that remains is to create a tool to load an image from our file system into the database table. I did this by creating a Sub. Note: you will have to add the Microsoft Office 16.0 Object Library and the Microsoft ActiveX Data Objects 6.1 Library to your project). The Sub is described by the inline comments. The objective is to convert the image file into a bit stream and insert it into a new database record that also includes the filename. See below:

Sub LoadImage(ServerName As String, Database As String, ImageFile As String)

'This sub accepts database connection details and the path to an image and inserts the image into

'the database's Images table

'modified from https://usefulgyaan.wordpress.com/2014/09/30/store-and-fetch-files-sql-server-tables/

On Error GoTo Error

'Define variables

Dim adoStream As Object

Dim adoCmd As Object

Dim strFilePath As String

Dim adoCon As Object

'Set up the connection, image stream and sql command objects

Set adoCon = CreateObject("ADODB.Connection")

Set adoStream = CreateObject("ADODB.Stream")

Set adoCmd = CreateObject("ADODB.Command")

'Open Connection to SQL server

adoCon.CursorLocation = adUseClient

adoCon.Open "Provider=SQLOLEDB;Data Source=" & ServerName & ";Initial Catalog = " & Database & ";Integrated Security=SSPI;"

'Stream image into object

adoStream.Type = adTypeBinary

adoStream.Open

adoStream.LoadFromFile ImageFile 'It fails if file is open

'Build an INSERT query command with two parameters, image and filename

With adoCmd

.CommandText = "INSERT INTO Images(Filename,SurveyImage) VALUES (?,?)" ' Query

.CommandType = adCmdText

'Build insert query parameters

.Parameters.Append .CreateParameter("@Filename", adVarChar, adParamInput, 255, ImageFile) 'Image filename

.Parameters.Append .CreateParameter("@Image", adVarBinary, adParamInput, adoStream.Size, adoStream.Read) 'The image stream

'---

End With

'Execute the INSERT query

adoCmd.ActiveConnection = adoCon

adoCmd.Execute

'Close the connection to free resources

adoCon.Close

Exit Sub

Error:

MsgBox Err.Description

End Sub

Now that I had a way to insert image records into my database I needed a way to quickly select multiple image files and process them. I created an event on the ‘Load a new image…’ button and wrote the following code to open a file browser dialog. Once the user selected image files the code loops through the files and submits them to the Sub above, inserting them into the database one by one.

Private Sub LoadImageButton\_Click()

'Open a file chooser dialog to allow the user to select images to import into the database

'Declare variables

Dim FD As FileDialog

Dim FileChosen As Integer

Dim FileName As String

Dim i As Integer

Dim SqlServer As String

Dim Database As String

'Database details

SqlServer = "YourSQLServerInstance"

Database = "ImagesDatabase"

'Create an open file dialog

Set FD = Application.FileDialog(msoFileDialogFilePicker)

FD.InitialView = msoFileDialogViewList

FD.AllowMultiSelect = True

FileChosen = FD.Show

If FileChosen = -1 Then

'Loop through the image files chosen in the file dialog and load the image into the database

For i = 1 To FD.SelectedItems.Count

FileName = FD.SelectedItems(i)

'Load the image file into the database

LoadImage SqlServer, Database, FileName

Next i

End If

'Refresh the form

Me.Requery

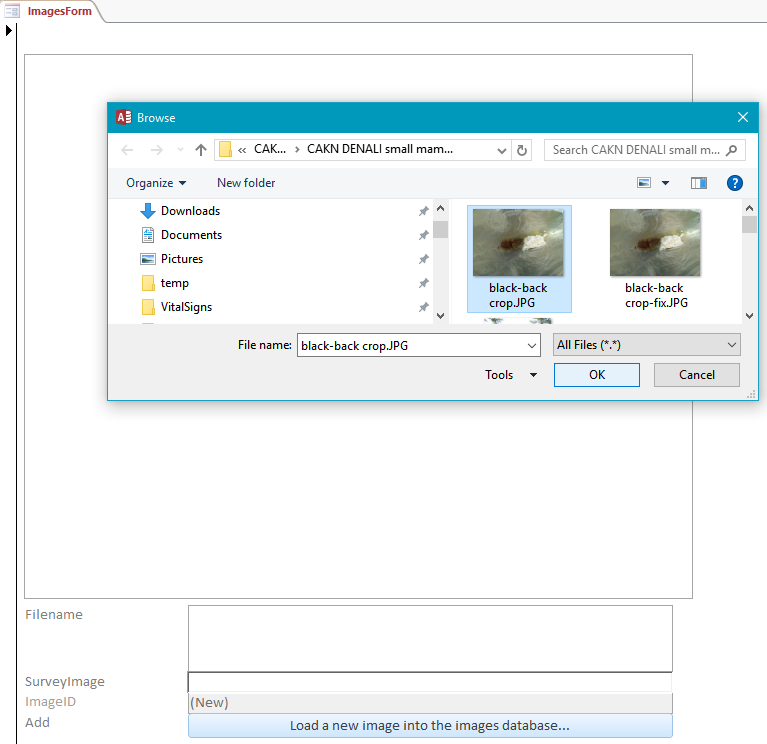
Exit Sub

Error:

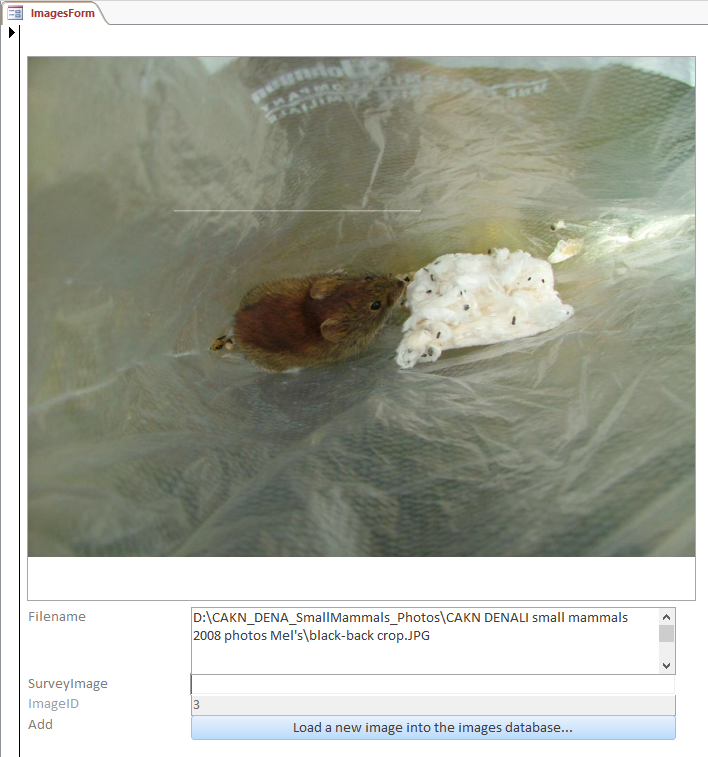
MsgBox Err.Description

End Sub

Now for a test. When I click the Load image button I get a file browser that allows me to select multiple image files.



The application loops through each selected file and calls the LoadImage Sub which then inserts them into the database and refreshes the form:



# Conclusion

I successfully created an image database application that stores, retrieves and adds images and image records to an SQL Server database.