Storing Images and BLOB files in SQL Server Part 2
By Don Schlichting

http://www.databasejournal.com/features/mssql/article.php/3724556

Introduction

This article will focus on using the SQL Server Binary data type to store small image files. In Part 1 of this series, the definition of binary and BLOBs (Binary Large Objects) was discussed, as well as when to store them in SQL Server. A few reasons to store images or binary data inside SQL Server include security, client access, and transactional control. This article will focus on the varBinary(MAX) data type. It is available in SQL 2005 and SQL 2008. The (MAX) extension to the Binary type means there is no upper size limit. The "var" means the size is variable rather than fixed as in the case of the standard Binary data type. SQL BOL (Books On Line) gives a good example of the three binary types:

Use Binary when the sizes of the column data entries are consistent (and less than 8,000 bytes).

Use varBinary when the sizes of the column data entries vary considerably (and are less than 8,000 bytes).

Use varBinary(max) when the column data entries exceed 8,000 bytes.

For SQL 2000, use the Image data type. However, be aware that Microsoft has stated the Image data type is for backwards compatibility only, and may be discontinued on a future version.

Example Table

To begin, we'll create a test database and table to hold our images. Use the following TSQL statement:

```
USE master;
GO

CREATE DATABASE Test;
GO

USE Test;
GO

CREATE TABLE BLOBTest
(
TestID int IDENTITY(1,1),
BLOBName varChar(50),
BLOBData varBinary(MAX)
);
```

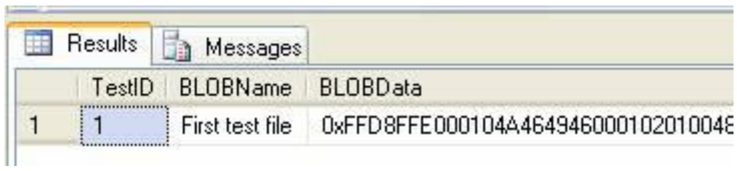
In this example, the column name is BLOBData, but the column name can be any standard SQL name. Binary data types do not have any special naming restrictions.

If you’re running SQL 2005 or SQL 2008, test the database inserting an image using a TSQL statement. This statement will not work in SQL 2000 because only a (MAX) data type can be the target. Find a small image then execute the following statement:

```
INSERT INTO BLOBTest
        (BLOBName, BLOBData)
        SELECT 'First test file',
                BulkColumn FROM OPENROWSET(
                        Bulk 'C:\temp\nextup.jpg', SINGLE_BLOB) AS BLOB
```

Substatute the ‘C:\temp\nextup.jpg’ for the file system path to your file. The OPENROWSET statement allows SQL to access data from an external provider. Bulk is a special provider for OPENROWSET created for inserting documents and images. For addiational details, see BOL “Importing Large Objects by using the OPENROWSET Bulk Rowset Provider”. A Select of the table should produce one record as shown below.

```
SELECT *
FROM BLOBTest
```



	TestID	BLOBName	BLOBData
1	1	First test file	0xFFD8FFE000104A46494600010201004E

The SELECT statement will verify data has been inserted, but there isn’t a way in SQL to view the image. For that we’ll create a small Visual Studio application.

Binary Write

This example will read the stored image from SQL and display it on a web page using Visual Studio. Create a new page without code behind. The code behind example will be shown later.

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<%
    string sConn = @"server=.; database=Test; Integrated
Security=True";
    SqlConnection objConn = new SqlConnection(sConn);
    objConn.Open();

    string sTSQL = "SELECT BLOBData FROM BLOBTest";

    SqlCommand objCmd = new SqlCommand(sTSQL, objConn);
    objCmd.CommandType = CommandType.Text;

    SqlDataReader dr = objCmd.ExecuteReader();
    dr.Read();

    Response.BinaryWrite((byte[])dr["BLOBData"]);

    objConn.Close();

%>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
    </form>
</body>
</html>
```

The web page will use the sConn variable to find the database. The phrase "Integatred Security = True" specifies Windows security will be used. The Server name should be the name of your server, or use a dot if SQL and web server are on the same local machine. The SQL statement is only getting the image, but in future examples, we'll retrive the file name as well and build a more real life application. The command type being set to "text" means a SQL statement will be passed in, as opposed to the name of a stored procedure.

The next statement:

```
        SqlCommand objCmd = new SqlCommand(sTSQL, objConn);
```

Creates a command object that ties the sql statement and the connection together. The command object is execute in the next statement:

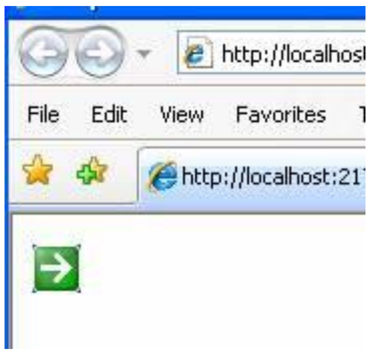
```
        SqlDataReader dr = objCmd.ExecuteReader();
```

A new data reader is declared and set to the command objects return (the sql statement executed against the connection). The dr.Read statement loads the first result (the only result in this example). If we wern't sure of a successful read, an IF statement could be used as a test. An exmaple will be shown in the next article.

So now, the data reader has the image binary data, and it will be streamed to the page using Response.BinaryWrite. Notice the html for the web page is empty. The BinaryWrite dosent need an any html objects. In fact, any html on the page would be not be displayed. For example, we'll stick a sting between the div:

```
<div>
        Hello World
    </div>
```

But When the page is run, only the image is displayed as shown below:



In our example database, we have additional data stroed in SQL that we would like presented along with the image, such as the file name. Because the Binary Write would overwrite any other html on the page, a workaround is needed. Save this page as GetPicture.aspx. Create a new page and put an image control on it as shown below. Set the imge url to the GetPicture page.

```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        Here is the image <asp:Image ID="Image1" runat="server"
ImageUrl="GetPicture.aspx" />
    </div>
    </form>
</body>
</html>
```

Because the binary stream stored in SQL can not be passed directly to the image control, the web page containing the stream was passed.

Conclusion

Binary data can be stored in SQL and retreived by a web application. In the next article, a web appliction will be created for inserting images into SQL. In addition, the exmpales in this article will be expanded to include passing the file name of the picture back to the appliciton, passing in a specific image name to retrieve, and converting these applictions to use stored procedures and code behind pages.