An Introduction to SQL Server FileStream

03 August 2009
by Jacob Sebastian

Filestream allows us to store and manage unstructured data in SQL Server more easily. Initially, the accounts of FILESTREAM assumed prodigious powers of concentration and cognition, and we mortals all recoiled numbly. However, it became clear that we were missing out on some extraordinarily useful functionality, so we asked Jacob Sebastian to come up with a simple and clear-cut account of the FILESTREAM feature in SQL Server 2008. You'll agree he has managed the feat superbly.

Introduction

Problems with storing unstructured data

Storing and managing unstructured data was tricky prior to the release of SQL Server 2008. Before then, there were two approaches to storing unstructured data in SQL Server. One approach suggests storing the data in a VARBINARY or IMAGE column. This ensures transactional consistency and reduces management complexities, but is bad for performance. The other approach is to store the unstructured data as disk files and store the location of the file in the table along with the other structured data linked to it. This approach was found to be good in terms of performance, but does not ensure transactional consistency. Moreover management (backup, restore, security etc) of the data will be a pain too.

FILESTREAM Feature

FILESTREAM was introduced in SQL Server 2008 for the storage and management of unstructured data. The FILESTREAM feature allows storing BLOB data (example: word documents, image files, music and videos etc) in the NT file system and ensures transactional consistency between the unstructured data stored in the NT file system and the structured data stored in the table.

Storing BLOB data in NT file system, allows SQL Server to take advantage of the NTFS I/O streaming capabilities and at the same time, maintain transactional consistency of the data. FILESTREAM uses NT System Cache for caching file data. This minimizes the effect that FILESTREAM data might have on the Database Engine performance. When accessing FILESTREAM data through the streaming API, SQL Server buffer pool is not used and hence it does not reduce the amount of memory available for Database Engine query processing.

The term 'FILESTREAM data type' or 'FILESTREAM column' is very commonly used and it gives an indication that the FILESTREAM feature is implemented as a data type. This is not true. FILESTREAM is not a data type; instead, it is an attribute that can be assigned to a VARBINARY (MAX) column. When the FILESTREAM attribute of a VARBINARY (MAX) column is set, it becomes a 'FILESTREAM enabled' column. Any data that you store in such columns will be stored in the NT file system as a disk files and a pointer to the disk file is stored in the table. A VARBINARY (MAX) column with FILESTREAM attribute is not restricted to the 2 GB limit SQL Server imposes on Large Value Types. The size of the file is limited by the size of the disk volume only.

When the FILESTREAM attribute is set, SQL Server stores the BLOB data in the NT file system and keeps a pointer the file, in the table. SQL Server ensures transactional consistency between the FILESTREAM data stored in the NT file system and the structured data stored in the tables.

Installing and Configuring FILESTREAM

The default installation of SQL Server 2008 disables FILESTREAM feature. It is quite easy to enable the FILESTREAM feature as part of a new SQL Server 2008 installation. If the SQL Server 2008 instance is already installed without FILESTREAM feature, it can still be enabled following the steps explained later in this section.

Enabling FILESTREAM as part of installation

The easiest way to enable FILESTREAM feature is to do so as part of the installation process. You will see a new tab labeled "FILESTREAM" on the "Database Engine Configuration" page of the installation wizard. This tab allows you to specify the FILESTREAM configuration options that you wish to enable on your new SQL Server Instance.

FILESTREAM feature may be enabled with three different levels of access to the FILESTREAM data, namely:

1. Enable FILESTREAM for Transact-SQL access
2. Enable FILESTREAM for file I/O streaming access
3. Allow remote clients to have streaming access to FILESTREAM data

Enabling FILESTREAM during an Unattended installation

If you are installing SQL Server 2008 in 'Unattended mode', you can use /FILESTREAMLEVEL and /FILESTREAMSHARENAME configuration options to configure FILESTREAM features. For more information about installing SQL Server 2008 in 'Unattended mode', see How to: Install SQL Server 2008 from the Command Prompt

Enabling FILESTREAM after installation

Enabling FILESTREAM feature on a SQL Server 2008 Instance, installed without FILESTREAM features, would require a little more work. To enable FILESTREAM feature, go to 'SQL Server configuration Manager' and right click on instance of SQL Server Service and select 'properties'. The property page will show an additional tab labeled 'FILESTREAM', which looks exactly the same as the FILESTREAM configuration page shown as part of the installation wizard. The FILESTREAM feature can be enabled by setting the appropriate options on this page.

If you wish to automate this process, you could try running the VBScript given at How to enable FILESTREAM from the command line. to do it.

Once FILESTREAM feature is enabled, the next step is to configure FILESTREAM Access Level. This is an additional step that is required only if you configure FILESTREAM after the installation of SQL Server. Open SQL Server Management Studio and open the properties of the SQL Server 2008 instance. Select the 'Advanced' tab and change the 'FILESTREAM Access Level' to 'Transact-SQL Access enabled' or 'Full access enabled'.

Alternatively, 'FILESTREAM Access Level' can be configured using TSQL by running the following statement.

```
EXEC sp_configure filestream_access_level, 2
GO
RECONFIGURE
GO
```

The last parameter to **sp_configure** specifies the Access Level, where 0 means 'Disabled', 1 means 'Transact-SQL Access Enabled' and 2 means 'Full Access Enabled'

Using FILESTREAM

Once you have an instance of SQL server 2008 with FILESTREAM feature enabled, you are ready to go ahead and create FILESTREAM enabled databases. When you create a FILESTREAM enabled database, the FILESTREAM data should be placed in a separate file group having a special attribute that indicates that the file group contains FILESTREAM data.

SQL Server allows FILESTREAM file groups to be on compressed volumes. If you are using FILESTREAM on a clustering environment, FILESTREAM file groups must be on shared disk resources.

Creating a FILESTREAM enabled database

A basic FILESTREAM enabled database should have the following storage components:

- MDF File
- LOG File
- FILESTREAM Data Container

MDF File and LOG files are familiar to us and do not need any explanation. However, FILESTREAM Data Container may be new to some. FILESTREAM Data Container is a special folder in the NT File System where SQL Server will store FILESTREAM data as disk files. We will examine FILESTREAM Data container a little later in this article.

Here is the script that creates a FILESTREAM enabled database.

```
CREATE DATABASE NorthPole
ON
PRIMARY (
      NAME = NorthPoleDB,
      FILENAME = 'C:\Temp\NP\NorthPoleDB.mdf'
), FILEGROUP NorthPoleFS CONTAINS FILESTREAM(
      NAME = NorthPoleFS,
    FILENAME = 'C:\Temp\NP\NorthPoleFS')
LOG ON (
      NAME = NorthPoleLOG,
    FILENAME = 'C:\Temp\NP\NorthPoleLOG.ldf')
GO
```

Note the portion highlighted in yellow. That is the part that adds the FILESTREAM Data Container to the database. One important point to note here is that the last sub directory in the FILESTREAM Data Container path should not already exist. In the above example, "c:\temp\np" should exist, however, the root level folder "NorthPoleFS" should not exist in "c:\temp\np" at the time of creating the database. SQL Server will create the root folder and configure it. If the folder already exists, the operation will fail with an error.

Understanding FILESTREAM Data Container

The root folder where FILESTREAM data of a database is stored is called *FILESTREAM Data Container*. When you create a database with FILESTREAM feature enabled, the FILESTREAM Data Container will be created with the path you specify in your CREATE statement. The root directory in the FILENAME parameter will be configured as FILESTREAM Data Container. It is important that this folder should not exist at the time of creating the database. SQL Server will create this folder and will configure it as FILESTREAM Data Container.

Reviewing the FILESTREAM Data Container

Let us go ahead and take a look at the FILESTREAM data container. You may not do this every time you create a FILESTREAM enabled database or table. However, for the purpose of understanding the FILESTREAM Data Container structure, let us take a look at how the folder looks like.

When we created the database, we had specified the path to the location that we wanted to have configured as the FILESTREAM Data Container for our database. If you have not modified the location I specified in the code snippet, it will be "C:\temp\NP\NorthPoleFS". Open windows explorer and navigate to the folder you specified as FILESTREAM Data Container while creating the database.

You will see a folder named "$FSLOG" and a file "filestream.hdr" there. The folder "$FSLOG" is the FILESTREAM equivalent of the Database Transaction Log file. "filestream.hdr" contains important metadata information used by SQL Server internally. Make sure that you do not tamper with this file.

Creating a table with FILESTREAM columns

Once we have a FILESTREAM enabled database, we are ready to go ahead and create tables having FILESTREAM columns. A FILESTREAM column is a VARBINARY(MAX) column that has the FILESTREAM attribute enabled. Remember, FILESTREAM is not a new DATA TYPE, it is a new attribute added to a VARBINARY(MAX) column.

```
CREATE TABLE [dbo].[Items](
   [ItemID] UNIQUEIDENTIFIER ROWGUIDCOL NOT NULL UNIQUE,
   [ItemNumber] VARCHAR(20),
   [ItemDescription] VARCHAR(50),
   [ItemImage] VARBINARY(MAX) FILESTREAM NULL
)
```

Note that every table that has a FILESTREAM column should have a **UNIQUEIDENTIFIER** column with **ROWGUIDCOL** and **UNIQUE** attributes.

Reviewing the FILESTREAM Data Container Changes

Let us go back to the FILESTREAM Data Container folder and see if we can detect any change since the last time we visited it. This time you will notice that a new folder is created in the root folder with a GUID value as its name. SQL Server will create a folder for each table that uses FILESTREAM enabled columns. Within this folder, a sub folder will be created for each FILESTREAM enabled column in the table.

Inserting FILESTREAM data

We have got the table created. Let us now add a couple of rows to the table we created. As you must have figured out by looking at the column names, this table is intended for storing product information. Along with the item information (ItemNumber and ItemDescription) we will also store an image of the item. The image will be stored in the FILESTREAM enabled column.

Let us add 'Microsoft Mouse' as the first item to this table. I have a cute image of a Microsoft Mouse on my hard disk and let us load the content of the image file and store that to the 'ItemImage' column.

For the purpose of this example, let us use OPENROWSET(BULK..) to load the content of the image file (jpg) from the disk to a VARBINARY(MAX) variable. Once the content of the image is loaded to the variable, we can store it to the FILESTREAM enabled column. Let us see the code that does it.

```
-- Declare a variable to store the image data
DECLARE @img AS VARBINARY(MAX)

-- Load the image data
SELECT @img = CAST(bulkcolumn AS VARBINARY(MAX))
      FROM OPENROWSET(
            BULK
            'C:\temp\MicrosoftMouse.jpg',
            SINGLE_BLOB ) AS x

-- Insert the data to the table
INSERT INTO Items (ItemID, ItemNumber, ItemDescription, ItemImage)
SELECT NEWID(), 'MS1001','Microsoft Mouse', @img
```

Before you run the above code, make sure that you change the name of the file ("c:\temp\MicrosoftMoust.jpg") to a valid image file name that exists in your hard disk.

Accessing FILESTREAM Data using TSQL

Even though the actual data of a FILESTREAM enabled column is stored in the NT File System, it will be completely transparent to the TSQL code. You can access the data in the FILESTREAM enabled column just like any other column of the given table. The following example queries all the information from the "Items" table which returns the FILESTREAM data too.

```
SELECT * FROM items
/*
ItemID           ItemNumber ItemDescription ItemImage
---------------- ---------- --------------- ------------
EE9A1CB8-3514... MS1001     Microsoft Mouse 0xFFD8FFE...
*/
```

Usually the TSQL code (stored procedure) does not really do any manipulation of the FILESTREAM data except for saving and retrieving it. It will be the client applications that actually use the FILESTREAM data. Let us look at a VB.NET client application that reads the image data that we just stored into the FILESTREAM data store and displays it in an image control. (To keep the source code listing minimal, I have not added any error handling code)

```
'Create a connection to the database
```

```
Dim ConStr As String
ConStr = "Data Source=JACOBXPS\katmaifs;Initial Catalog=NorthPole" & _
         ";Integrated Security=True"
Dim con As New SqlConnection(ConStr)
con.Open()

'Retrieve the FilePath() of the image file
Dim sqlCommand As New SqlCommand()
sqlCommand.Connection = con
sqlCommand.CommandText = "SELECT ItemImage FROM items " + _
                         "WHERE ItemNumber = 'MS1001'"
Dim buffer As Byte() = sqlCommand.ExecuteScalar()

'Bind the image data to an image control
Dim ms As MemoryStream = New MemoryStream(buffer)
Dim bmp As Bitmap = New Bitmap(ms)
ItemImage.Image = bmp

'Cleanup
con.Close()
```

Accessing FILESTREAM data with Managed API

Accessing FILESTREAM data using Win32 Streaming has a number of advantages over accessing it using TSQL. When accessing FILESTREAM data using TSQL, SQL Server reads the content of the FILESTREAM data file and serves it to the client. SQL Server memory is used for reading the content of the data file. Accessing FILESTREAM data using Win32 Streaming does not use SQL Server memory. In addition it allows the application to take advantage of the Streaming capabilities of the NT File System.

Though accessing FILESTREAM data using Win32 Streaming has a number of advantages, it is a bit complicated to use, compared to the syntax needed to access it from TSQL. Before a client application can access the FILESTREAM data, it needs to find out the logical path that uniquely identifies the given file in the FILESTREAM data store. This can be achieved by using the "**PathName**" method of a FILESTREAM column. Note that the **PathName()** function is Case Sensitive. The following example shows how to retrieve the **PathName()** associated with the FILESTREAM data of a column.

```
SELECT
      ItemNumber,
      ItemImage.PathName() AS FilePath
FROM Items
/*
ItemNumber FilePath
---------- --------------------------------------------
MS1001     \\JACOBXPS\KATMAIFS\v1\NorthPole\dbo\Items\
           ItemImage\EE9A1CB8-3514-4115-8227-4E8F080E22E0
*/
```

The next step is to begin a transaction and obtain a transaction context. A client application can start a transaction by calling the '**BeginTransaction**' method of the **SqlConnection** object. After starting a transaction, the transaction context can be obtained by running the following query.

```
SELECT GET_FILESTREAM_TRANSACTION_CONTEXT() AS TransactionContext
```

```
/*
TransactionContext
-----------------------------------
0xCB261797D1878D4A9F9562660124BBD8
*/
```

Once the transaction context to the FILESTREAM data file is obtained, the file can be accessed using the **'SqlFileStream'** class. Here is the complete listing of a basic VB.NET application that reads FILESTREAM data using the Managed API and displays the picture on an Image Control. (To keep the code listing minimal, I have not added any error handling code)

```
'Create a connection to the database
Dim ConStr As String
ConStr = "Data Source=JACOBXPS\katmaifs;Initial Catalog=NorthPole" & _
         ";Integrated Security=True"
Dim con As New SqlConnection(ConStr)
con.Open()

'Retrieve the FilePath() of the image file
Dim sqlCommand As New SqlCommand()
sqlCommand.Connection = con
sqlCommand.CommandText = "SELECT ItemImage.PathName() AS PathName " + _
                         "FROM items WHERE ItemNumber = 'MS1001'"
Dim filePath As String = sqlCommand.ExecuteScalar()

'Obtain a Transaction Context
Dim transaction As SqlTransaction = con.BeginTransaction("ItemTran")
sqlCommand.Transaction = transaction
sqlCommand.CommandText = "SELECT GET_FILESTREAM_TRANSACTION_CONTEXT()"
Dim txContext As Byte() = sqlCommand.ExecuteScalar()

'Open and read file using SqlFileStream Class
Dim sqlFileStream As New SqlFileStream(filePath, txContext, FileAccess.Read)
Dim buffer As Byte() = New Byte(sqlFileStream.Length) {}
sqlFileStream.Read(buffer, 0, buffer.Length)

'Bind the image data to an image control
Dim ms As MemoryStream = New MemoryStream(buffer)
Dim bmp As Bitmap = New Bitmap(ms)
ItemImage.Image = bmp

'Cleanup
sqlFileStream.Close()
sqlCommand.Transaction.Commit()
con.Close()
```

Updating FILESTREAM Data

FILESTREAM data may be modified using either TSQL or using the FILESTREAM API. As mentioned earlier, modifying FILESTREAM data using the Streaming API has a number of advantages over accessing it using TSQL.

Using TSQL to update one or more FILESTREAM columns is not different from updating regular columns.

Though a FILESTREAM column can store more than 2 GB of data, it is not possible to access more than 2 GB of FILESTREAM data using TSQL, because 2 GB is the maximum size of any Large Value Type (VARBINARY, VARCHAR, NVARCHAR, TEXT, NTEXT) can store.

Modifying FILESTREAM data using streaming API is pretty much the same as what we saw in the previous example. Before accessing the data you need to access the **PathName**(), start a transaction and obtain a transaction context before modifying the data using the **SqlFileStream** class.

Deleting FILESTREAM Data

When a row is deleted from a table having FILESTREAM enabled columns, the record is removed from the table and the FILESTREAM data file is removed from the FILESTREAM Data Container. FILESTREAM data from the FILESTREAM data container is removed by the FILESTREAM garbage collector. Since this is done in a separate background thread, you may still see the deleted file in the FILESTREAM data container until the garbage collector runs again.

FILESTREAM Garbage Collector

SQL Server triggers the FILESTREAM Garbage Collector thread when a CHECKPOINT occurs. So, after deleting FILESTREAM data using TSQL you might notice that the physical file is not removed from the FILESTREAM data container right away. The file will remain in the FILESTREAM data container until the next CHECKPOINT occurs and the garbage collector runs. FILESTREAM operations generate minimal log in the TRN log file and hence it might take longer for a CHECKPOINT to occur if the database is not highly transactional. In such a case, if you want to trigger the garbage collector thread, you should issue an EXPLICIT CHECKPOINT.

FILESTREAM Feature Summary

With FILESTREAM, the SQL Server team not only added a feature to handle unstructured data, but also made sure that it smoothly integrates with many of the existing features of SQL Server.

- FILESTREAM feature is available with all versions of SQL Server 2008, including SQL Server Express.
- SQL Server Express database has a 4 GB limitation; however this limitation does not apply to the FILESTREAM data stored in a SQL Server Express database.
- FILESTREAM Columns can be replicated.
- FILESTREAM enabled databases can be used with LOG Shipping
- FILESTREAM columns can be used in Full Text Indexes
- FILESTREAM works with all recovery models
- FILESTREAM File Groups can be placed on compressed disk volumes
- The maximum size of the file that can be stored into the FILESTREAM data storage is limited by the size of the disk volume only.

Restrictions on existing features

Though the FILESTREAM feature smoothly integrates with many of the existing features of SQL Server, it adds restrictions to or completely disables a few other important features of SQL Server.

- A FILESTREAM enabled database cannot be used for mirroring. This is one of the key restrictions that I dislike. Mirroring is a very interesting feature and it cannot be used with FILESTREAM.

- FILESTREAM data is not available in database snapshots. If you create database snapshots and run a "SELECT * FROM table" query on a table with FILESTREAM columns, you will get an error. All queries that you run on a FILESTREAM enabled table of a database snapshot should exclude FILESTREAM columns.

FILESTREAM Limitations

The FILESTREAM implementation in SQL Server 2008 comes with a few limitations. The following limitations make complete sense to me and I have no complaints on them.

- FILESTREAM columns cannot be used in Index Keys.
- FILESTREAM columns cannot be used in the INCLUDED columns of a Non Clustered Index
- FILESTREAM columns cannot be used in a Table Valued Parameter
- FILESTREAM columns cannot be used in a memory table
- FILESTREAM columns cannot be used in a global or local temp table
- Statistics cannot be created on FILESTREAM columns
- Computed columns having reference to FILESTREAM columns cannot be indexed
- When FILESTREAM Data is accessed through Win 32 APIs, only READ COMMITTED ISOLATION level is supported.
- FILESTREAM data can be stored only on local disk volumes

However the following limitations are little restrictive and I really wish to have them removed in the future versions of SQL Server.

- FILESTREAM data is not supported on Database Snapshots. Well, that is acceptable, however what is the real pain is the error that we get if we run a "SELECT * FROM table" query on a FILESTREAM enabled table in a database snapshot. SQL Server should ideally return NULL values for FILESTREAM enabled columns, if it cannot create snapshots of FILESTREAM data. At present, it is a real pain to exclude all FILESTREAM enabled columns from all queries that runs on a Database Snapshot.
- Transparent Data Encryption (TDE) does not encrypt FILESTREAM data. It would be interesting to have this restriction removed in the future.

FILESTREAM Feature - Points to Remember

Take note of the following points if you intend to use FILESTREAM in your application.

- You cannot do database mirroring if you use FILESTREAM features
- If you replicate FILESTREAM enabled columns, make sure that all subscribers are running on SQL Server 2008 or later versions.
- If you intend to use LOG shipping, make sure that both primary and secondary servers are running on SQL Server 2008 or later versions.
- If you are on a FAILOVER CLULSTERING environment, make sure that you place the FILESTREAM file groups in a shared disk and FILESTREAM should be enabled on each node.
- You cannot access FILESTREAM data if you create database snapshots. If you try to access a FILESTREAM enabled column, SQL Server will raise an error.
- SQL Server Instance should be configured with Integrated Security if Win 32 access to the FILESTREAM data is required

FILESTREAM Best Practices

- Place each FILESTREAM data container in a separate volume
- Use the correct RAID level depending upon the nature of the application (read intensive, write intensive), expected work load etc
- Do periodical disk defragmentation
- Decide diligently on whether or not to use a compressed disk volume
- Disable 8.3 names in NTFS
- Disable last access time tracking in NTFS
- FILESTREAM data container should not be on a fragmented volume
- Make sure that the data being stored is appropriate for FILESTREAM storage. FILESTREAM storage is only good if the size of the data is more than 1 MB (approx)
- Avoid multiple small appends to the FILESTREAM file
- If FILESTREAM files are large, avoid using TSQL access to the FILESTREAM data.
- If reads require only the first few bytes, then consider using TSQL access using substring() function
- If the entire file is needed Win 32 access is desirable

Conclusions

FILESTREAM solves many of the problems related to the storage and management of unstructured data. The FILESTREAM feature of SQL Server allows taking advantage of the Streaming capabilities of NT File System and ensures transactional consistency between the unstructured data stored in the FILESTREAM Data Container and the structured data stored in the relational tables.