

INFORMATION PROCESSING TECHNIQUES

SQL File Stream

WEEK 14

MURTAZA MUNAWAR FAZAL

CHARACTER(n)	Character string. Fixed-length n
char(n)	Fixed width character string. Maximum 8,000 characters
varchar(n)	Variable width character string. Maximum 8,000 characters
varchar(max)	Variable width character string. Maximum 1,073,741,824 characters
text	Variable width character string. Maximum 2GB of text data
nchar	Fixed width Unicode string. Maximum 4,000 characters
nvarchar	Variable width Unicode string. Maximum 4,000 characters
nvarchar(max)	Variable width Unicode string. Maximum 536,870,912 characters
ntext	Variable width Unicode string. Maximum 2GB of text data
bit	Allows 0, 1, or NULL
binary(n)	Fixed width binary string. Maximum 8,000 bytes
varbinary	Variable width binary string. Maximum 8,000 bytes
varbinary(max)	Variable width binary string. Maximum 2GB
image	Variable width binary string. Maximum 2GB

Types of Data

Structured

Semi-structured

Unstructured

Structured Data

“normal” RDBMS data

Format is known and defined

Example: Sales Order

- [-] Sales.SalesOrderHeader
 - [-] Columns
 - 🔑 SalesOrderID (PK, int, not null)
 - RevisionNumber (tinyint, not null)
 - OrderDate (datetime, not null)
 - DueDate (datetime, not null)
 - ShipDate (datetime, null)
 - Status (tinyint, not null)
 - OnlineOrderFlag (Flag(bit), not null)
 - SalesOrderNumber (Computed, nvarchar(25), not null)
 - PurchaseOrderNumber (OrderNumber(nvarchar(25)), null)
 - AccountNumber (AccountNumber(nvarchar(15)), null)
 - 🔑 CustomerID (FK, int, not null)
 - 🔑 SalesPersonID (FK, int, null)
 - 🔑 TerritoryID (FK, int, null)
 - 🔑 BillToAddressID (FK, int, not null)
 - 🔑 ShipToAddressID (FK, int, not null)
 - 🔑 ShipMethodID (FK, int, not null)
 - 🔑 CreditCardID (FK, int, null)
 - CreditCardApprovalCode (varchar(15), null)
 - 🔑 CurrencyRateID (FK, int, null)
 - SubTotal (money, not null)
 - TaxAmt (money, not null)
 - Freight (money, not null)
 - TotalDue (Computed, money, not null)
 - Comment (nvarchar(128), null)
 - rowguid (uniqueidentifier, not null)
 - ModifiedDate (datetime, not null)

Semi Structured Data

some structure, but it is fluid

changes in structure should not break code

example: XML

Semi Structured Data

```
<SalesOrder DueDate="20120201">
  <OrderID>12</OrderID>
  <Customer>John Doe</Customer>
  <OrderDate>2012/01/15</OrderDate>
  <Items>
    <Item>
      <Product>Widget</Product>
      <Quantity>12</Quantity>
    </Item>
    <Item>
      <Product>Whatchamacallit</Product>
      <Quantity>2</Quantity>
    </Item>
  </Items>
</SalesOrder>
```

Unstructured Data

structure is merely encoding.

meta data may be in the structure

examples:

- Audio files
- Word Documents
- PDF
- Movies

Unstructured Data - Pre 2008

In SQL Server large binary files handling had two solutions:

- store in the file system, but a reference in the database
 - administrative issues (backup, security)
 - synchronization issues
- store the binary file in the database
 - Text/image/varbinary data type
 - complex storage, manipulation, and retrieval.

Filestream

Added in SQL Server 2008

Allows storage in the filesystem, but appears to be in the database.

SQL Server Instance

Filegroup_1

File 1 - c:\Program Files\...\Data\MyDB.mdf

File 2 - c:\Program Files\...\Data\MyDB2.ndf

File 3 - c:\Program Files\...\Data\MyDB3.ndf

Table: MyMP3s

Name varchar(200)

MP3 varbinary(max)

SQL Server Instance

Filegroup_1

File 1 - c:\Program Files\...\Data\MyDB.mdf

File 2 - c:\Program Files\...\Data\MyDB2.ndf

File 3 - c:\Program Files\...\Data\MyDB3.ndf

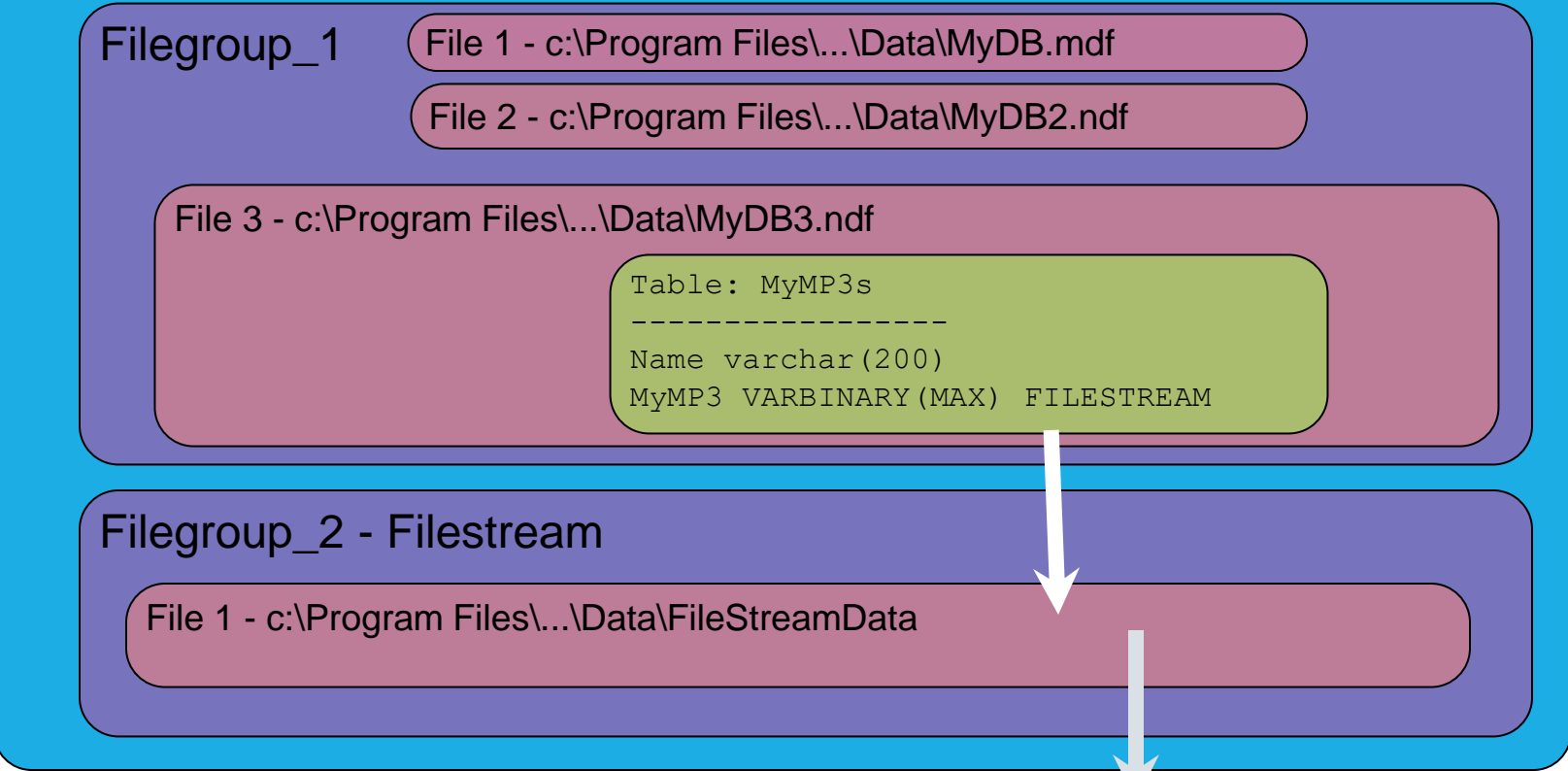
Table: MyMP3s

Name varchar(200)

MyMP3 VARBINARY(MAX) FILESTREAM

Filegroup_2 - Filestream

File 1 - c:\Program Files\...\Data\FileStreamData



Name	Date modified	Type
0000001f-000004fc-00d0	4/12/2010 10:01 AM	File
0000001f-0000051c-000e	4/12/2010 10:01 AM	File
0000001f-0000051c-001c	4/12/2010 10:01 AM	File
0000001f-0000051c-002a	4/12/2010 10:01 AM	File
0000001f-0000051c-0007	4/12/2010 10:01 AM	File
0000001f-0000051c-0015	4/12/2010 10:01 AM	File
0000001f-0000051c-0023	4/12/2010 10:01 AM	File
0000001f-0000051c-0031	4/12/2010 10:01 AM	File
0000001f-0000051c-0038	4/12/2010 10:01 AM	File

Filestream

Comparison point	Storage solution	
	File server / file system	SQL Server (using varbinary(max))
Maximum BLOB size	NTFS volume size	2 GB – 1 bytes
Streaming performance of large BLOBs	Excellent	Poor
Security	Manual ACLs	Integrated
Cost per GB	Low	High
Manageability	Difficult	Integrated
Integration with structured data	Difficult	Data-level consistency
Application development and deployment	More complex	More simple
Recovery from data fragmentation	Excellent	Poor
Performance of frequent small updates	Excellent	Moderate

Filestream

Comparison point	Storage solution		
	File server / file system	SQL Server (using varbinary(max))	FILESTREAM
Maximum BLOB size	NTFS volume size	2 GB – 1 bytes	NTFS volume size
Streaming performance of large BLOBs	Excellent	Poor	Excellent
Security	Manual ACLs	Integrated	Integrated + automatic ACLs
Cost per GB	Low	High	Low
Manageability	Difficult	Integrated	Integrated
Integration with structured data	Difficult	Data-level consistency	Data-level consistency
Application development and deployment	More complex	More simple	More simple
Recovery from data fragmentation	Excellent	Poor	Excellent
Performance of frequent small updates	Excellent	Moderate	Poor

Filestream - Caveats

TDE does not encrypt filestream data

Encryption is not supported on Filestream data

Containers cannot be nested

In a cluster – filestream container must be on shared resources.

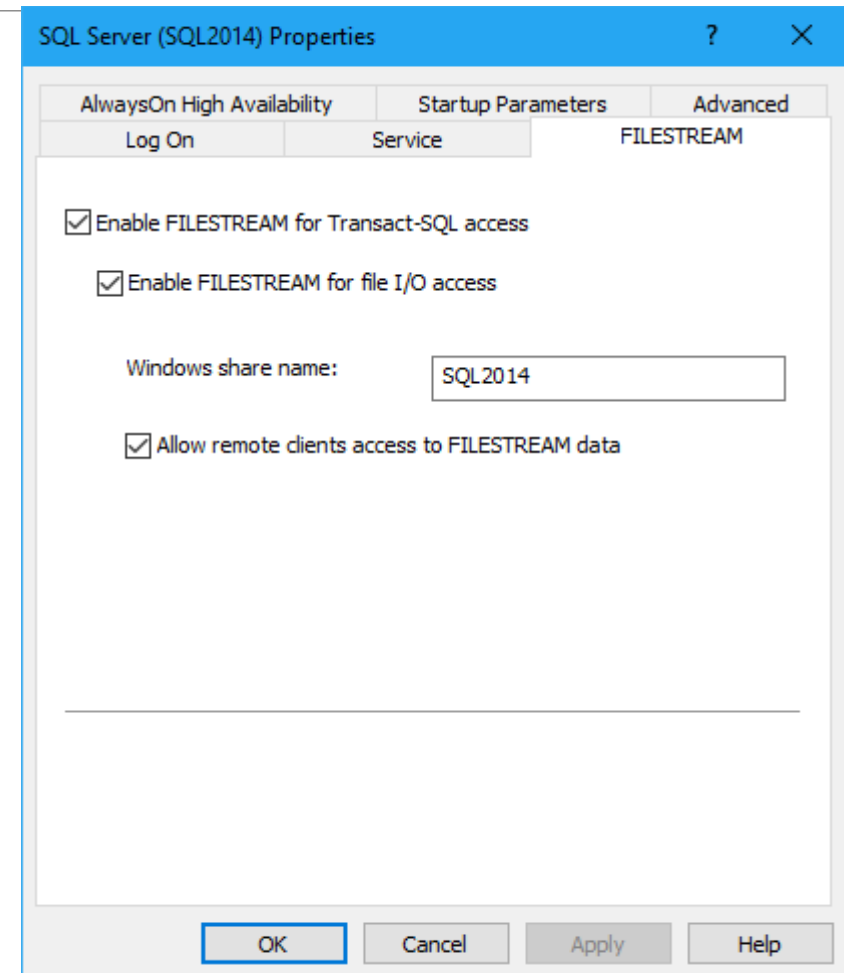
File Stream

- Enabling the FILESTREAM feature for a given SQL Server instance
- Setting the access level for the FILESTREAM data
- Adding FILESTREAM storage to an existing database
- Adding a FILESTREAM column to an existing table
- Inserting BLOB data into the FILESTREAM column.

-
- FILESTREAM is a hybrid feature that requires both the Windows Administrator and the SQL Server Administrator to perform actions before the feature is enabled. Because of this, the configuration is often a two-step process.
 - The Windows Administrator manages the Windows-related configuration changes needed for the FILESTREAM feature. For example, Windows Administrator privileges are required to create a Windows file share for the FILESTREAM feature. The Windows Administrator uses the SQL Server Configuration Manager to enable, disable, and configure FILESTREAM.
 - The SQL Server Administrator manages the configuration changes within the boundaries of the given SQL Server instance. The SQL Administrator enables, disables, and configures the FILESTREAM Access Level, either by using SQL Server Management Studio or by running `sp_configure`.

Enabling FILESTREAM

Our first step is to enable the FILESTREAM feature at the instance level. Briefly, on the machine on which the SQL Server 2008 instance is installed, navigate as follows (this path may change in future versions): Start | All Programs | Microsoft SQL Server 2008 | Configuration Tools | SQL Server Configuration Manager, then select SQL Server Services. Right-click on the appropriate instance, select Properties then move to the FILESTREAM tab. In here, you can enable the instance for T-SQL and file I/O streaming access.



Setting the access level

All the rest of the work can be done from within SQL Server Management Studio (SSMS), so open it up and connect to the instance you just FILESTREAM enabled. Since we are enabling FILESTREAM access on an instance that did not have FILESTREAM enabled as part of the SQL Server installation process, we will also need to set the FILESTREAM Access Level.

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

Adding FILESTREAM storage

Next, we need to add a FILESTREAM filegroup and FILESTREAM file (data container) to the database

```
ALTER DATABASE NorthPole
    ADD FILEGROUP NorthPole_fs
    CONTAINS FILESTREAM;
GO

ALTER DATABASE NorthPole
    ADD FILE (
        NAME = NorthPole_fs,
        FILENAME = 'C:\ArtOfFS\Demos\Chapter1\NorthPole_fs' )
TO FILEGROUP NorthPoleFS;
GO
```

Note that the C:\ArtOfFS\Demos\Chapter1 directory must exist on the appropriate SQL Server instance machine

Adding a FILESTREAM column

A FILESTREAM column can be created only on a table that has a UNIQUEIDENTIFIER column with the ROWGUIDCOL attribute. Notice, also, that the table contains an AuthorImage column of type VARBINARY(MAX) to which we've applied the FILESTREAM attribute.

```
CREATE TABLE [dbo].[Authors]
(
    [AuthorID] UNIQUEIDENTIFIER ROWGUIDCOL
                        NOT NULL
                        UNIQUE ,
    [AuthorName] VARCHAR(50) ,
    [AuthorImage] VARBINARY(MAX) FILESTREAM
                        NULL
)
```

Size of the files

- As per Microsoft Research paper, To BLOB or Not To BLOB:
 - For BLOBs that are smaller than 256 KB, the best performance will probably be obtained by simply storing the data directly in the relational tables.
 - Files larger than 1 MB are better stored in the file system, and managed through SQL Server using FILESTREAM.
 - In that in-between region of 256 KB to 1 MB, the decision of which way to go can only be taken after analyzing the specific data usage scenarios

Data usage patterns

- FILESTREAM does not support in-place (also referred to as "partial") updates of BLOB data files. Instead, every time a BLOB value is modified, SQL Server creates an entirely new copy of the file with the modified content, and discards the old file. The old file will be deleted by an asynchronous garbage collector process.
- If your business requirement includes making frequent changes to the BLOB data values, then FILESTREAM may not be the right option. However, this decision should be taken only after running some tests with the expected usage pattern and deciding whether to go with FILESTREAM or VARBINARY(MAX) storage.

Writing File to Database Using File Stream

- Insert into the desired table with the File Content as empty
- Get File Path from SQL File Stream
 - `comm.CommandText = string.Format(@"SELECT TOP(1) FileContent.PathName() FROM <table> WHERE <condition>");`
 - `string path = comm.ExecuteScalar().ToString();`
- Write Byte Stream Data to Memory Stream
 - `MemoryStream FileContent = new MemoryStream();`
 - `BinaryWriter writer = new BinaryWriter(FileContent);`
 - `writer.Write(BinaryData);`

■ Write data from Memory Stream to SQL (File Stream)

■ Get SQL Write Destination:

- `comm.CommandText = string.Format(@"SELECT TOP(1) GET_FILESTREAM_TRANSACTION_CONTEXT()
FROM <table> where <condition>");`
- `byte[] transactionContext = (byte[])comm.ExecuteScalar();`

■ Write to Database

- `System.Data.SqlTypes.SqlFileStream destination = new
System.Data.SqlTypes.SqlFileStream(path, transactionContext, System.IO.FileAccess.Write);`
- `FileContent.CopyTo(destination);`

References

FILESTREAM Overview - <http://msdn.microsoft.com/en-us/library/bb933993.aspx>

FILESTREAM Best Practices – <http://msdn.microsoft.com/en-us/library/dd206979.aspx>

FILESTREAM Storage in SQL Server 2008 - <http://msdn.microsoft.com/en-us/library/cc949109.aspx>

Remote BLOB store - <http://technet.microsoft.com/en-us/library/gg638709.aspx>

Enable the Prerequisites for FileTable - [http://msdn.microsoft.com/en-us/library/gg509097\(v=sql.110\).aspx](http://msdn.microsoft.com/en-us/library/gg509097(v=sql.110).aspx)

Create, Alter, and Drop FileTables - <http://msdn.microsoft.com/en-us/library/gg509088%28v=sql.110%29.aspx>