Sperasoft
Your game dev partner

SQL Server 2012: File Tables

File storage connected to RDBMS

Why we need it?

- There are lots of files
- We need to store them somewhere
- We need to query over them

- Files have metadata information like dates, size, content type etc.
- It needs to be queried as well

Who Needs It?

- Software that have to deal with files
- File libraries, archives, storages etc.
- Document-centric software
- Email systems with attachments
- Content Management Systems
- Version Control Systems
- and so on and so on…

Options available before – 1/3

1. Store files to any File System (or external storage), keep related information in DB Tables, sync both places

2. Store all files and related information in DB Tables

Options available before – 2/3

1. Store files to any File System (or external storage), keep related information in DB Tables, sync both places
   - the only problem here is to keep both places in sync
   - seems to be the most common scenario

Options available before – 3/3

2. Store all files and related information in DB Tables

- the files content is stored in BLOBs
- huge database size because of content stored within database file

New Option – SQL FileTables

SQL Server 2012 brings a new feature:

- FileTables

MSDN:

- “You can store files and documents in special tables in SQL Server called FileTables, but access them from Windows applications as if they were stored in the file system, without making any changes to your client applications”

Technically FileTables Are

Technically FileTables represent both options combined.

- but automated by SQL Server 2012 Engine
- based on SQL Server 2008/R2 FileStreams

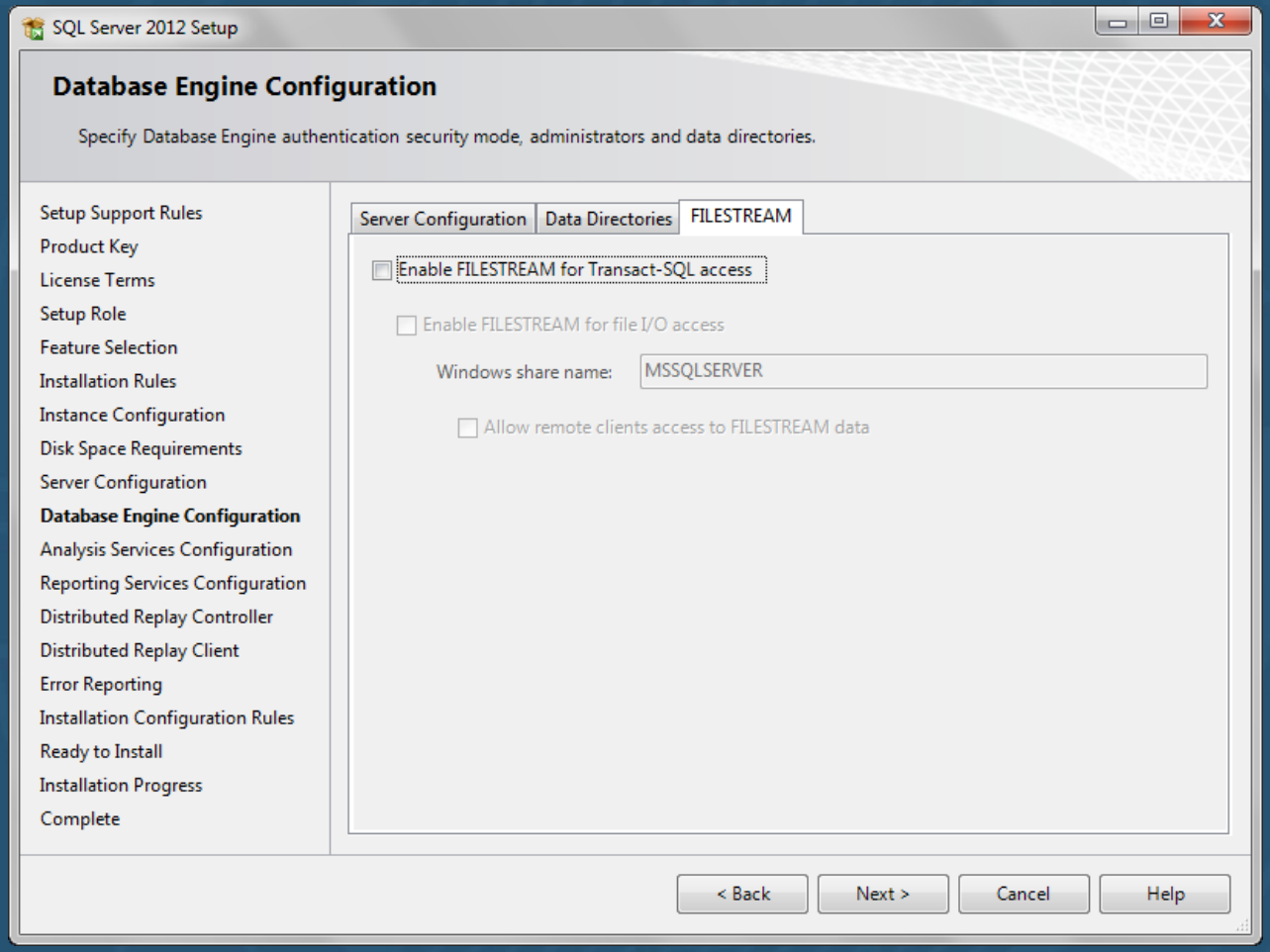
FileTables Benefits

- Windows File I/O compatibility:
  - non-transacted streaming access and in-place updates
  - A hierarchical namespace of directories and files
  - Storage of 10 file attributes such as Created Date and Modified Date
  - Support for both file and directory management Win API
- Full-text search over files and metadata
- Semantic search over files and metadata

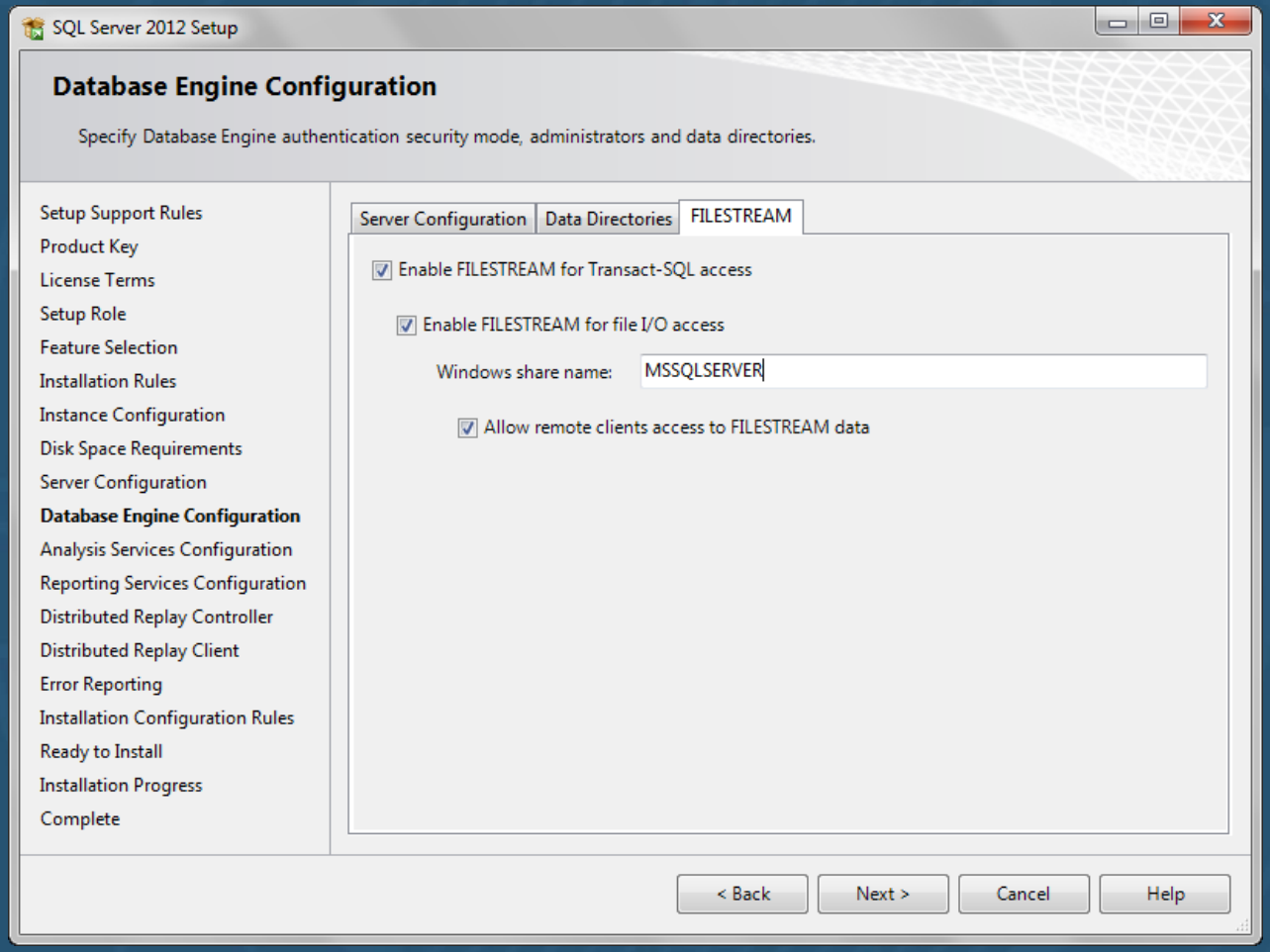
FileTables Restrictions

- Do not support Memory-Mapped Files
  - easily reproducible using Windows Notepad

Installation Options – 1/2



Installation Options – 2/2



UNC Directory Hierarchy

\\<machine-name>\
  <sql-instance-level FILESTREAM share>\
  <database-level directory>\
  <FileTable directory>\

Sample:
\\shirmanov\MSSQLSERVER\FileTablesDb\Documents\

Create FileStream-enabled Database

```
CREATE DATABASE [FileTablesDb]
ON
  PRIMARY (NAME = PrimaryFG, FILENAME =
N'c:\SqlFileStream\FileTablesDb.mdf'),
  FILEGROUP FileStreamFG CONTAINS FILESTREAM (NAME =
FileStreamFG, FILENAME = N'c:\SqlFileStream\FileStream')
  LOG ON (NAME = FileTablesDbLog, FILENAME =
N'c:\SqlFileStream\FileTablesDb.ldf')
  WITH FILESTREAM (NON_TRANSACTED_ACCESS = FULL,
DIRECTORY_NAME = N'FileTablesDb')
GO
```

Create a FileTable

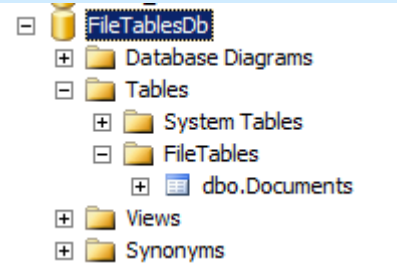
```
CREATE TABLE Documents
  AS FileTable
  WITH (
    FileTable_Directory = 'Documents',
    FileTable_Collate_Filename = database_default
);
GO
```

Create a FileTable

```
SELECT * FROM sys.filetables;
GO
```

Results | Messages

	object_id	is_enabled	directory_name	filename_collation_id	filename_collation_name
1	245575913	1	Documents	53269	Cyrillic_General_CI_AS



Enumerate All FileTables in Database

```
-- enumerate all FileTables in database

SELECT
  db_name() AS db_name,
  db_id() AS db_id,
  SC.[name] AS schema_name,
  SO.[schema_id],
  SO.[name] AS object_name,
  FT.[object_id],
  FT.[directory_name],
  FT.[filename_collation_id],
  FT.[filename_collation_name],
  FileTableRootPath() + '\' + FT.[directory_name] AS unc_path
FROM
  [FileTablesDb].[sys].[filetables] FT
LEFT JOIN [sys].[objects] SO
  ON FT.[object_id] = SO.[object_id]
LEFT JOIN [sys].[schemas] SC
  ON SC.[schema_id] = SO.[schema_id];
```

Demo

- Copy files to UNC share and show them in database table

How FileStream Catalog Looks Like

Name	Size	Date modified
FileTablesDbFileStream		02.05.2012 18:18
FileTablesDb.ldf	1 024 KB	02.05.2012 18:18
FileTablesDb.mdf	4 160 KB	02.05.2012 18:18

Name	Size	Date modified
$FSLOG		02.05.2012 18:18
filestream.hdr	1 KB	02.05.2012 18:18

Get SQL Instance-level FILESTREAM Share Name

```
SELECT SERVERPROPERTY
(N'FILESTREAMShareName');
```

Results | Messages

	(No column name)
1	MSSQLSERVER

Get Database-level Directory Name

```
SELECT DB_NAME(D.database_id) AS
"database_name", D.database_id,
D.non_transacted_access,
D.non_transacted_access_desc, D.directory_name
  FROM sys.database_filestream_options D
  WHERE directory_name IS NOT NULL;
GO
```

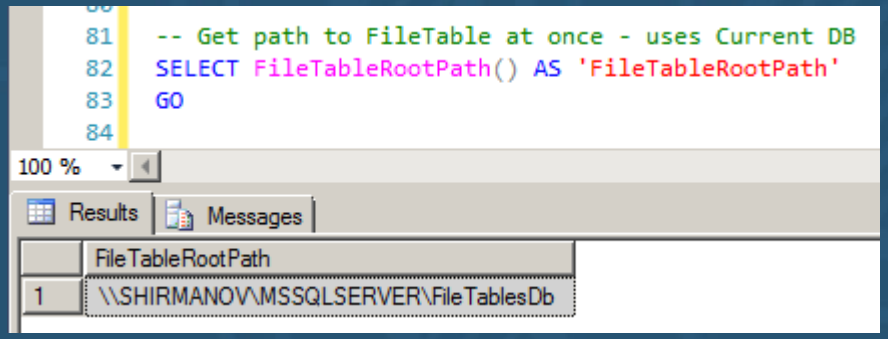
Results | Messages

	database_name	database_id	non_transacted_access	non_transacted_access_desc	directory_name
1	FileTablesDb	17	2	FULL	FileTablesDb

Get Path to FileTable at Once

```
-- Get path to FileTable at once - uses Current DB

SELECT FileTableRootPath() AS 'FileTableRootPath'
GO
```



Get a List of Open File Handles

```
-- Get a List of Open File Handles
-- Associated with a FileTable
SELECT * FROM
  sys.dm_filestream_non_transacted_handles;
GO
```

Kill All Open Handles

```
-- Kill all open handles in a single filetable
EXEC
  sp_kill_filestream_non_transacted_handles
   @table_name = 'Documents';
GO
```

Identify Open Files & Associated Locks

```
-- To identify open files and the associated locks
SELECT opened_file_name
  FROM
sys.dm_filestream_non_transacted_handles
  WHERE fcb_id IN
    ( SELECT request_owner_id FROM
sys.dm_tran_locks );
GO
```

Insert a file to the Root via T-SQL

```
INSERT INTO dbo.Documents
  (Name, file_stream)
  VALUES ('Testfile1.txt', 0x);
```

Insert a file to the folder via T-SQL

```
SELECT @pathstring = path_locator.ToString()
  from dbo.Documents
  where name = 'SQLFiles';

SET @newpath = @pathstring +  convert(varchar(20), convert(bigint,
substring(convert(binary(16), newid()), 1, 6))) +
'.' +  convert(varchar(20), convert(bigint, substring(convert(binary(16),
newid()), 7, 6))) + '.' +  convert(varchar(20), convert(bigint, substring(
convert(binary(16), newid()), 13, 4))) + '/';

INSERT INTO dbo.Documents
  (Name, path_locator, file_stream)
  VALUES ('SQLFilesTest.txt', @newpath, 0x);
```

Find Duplicate Files

```
-- Find duplicate files by name and size
SELECT
  COUNT(*) AS duplicates_number,
  MAX(D.stream_id) AS stream_id, D.name,
D.file_type, D.cached_file_size
FROM dbo.Documents D
GROUP BY D.name, D.file_type, D.cached_file_size
HAVING COUNT(*) > 1;
```

References

- http://msdn.microsoft.com/en-us/library/ff929144.aspx
- http://sqlskills.com/BLOGS/BOBB/post/SQL-Server-2012-FileTables-in-T-SQL-part-3-hierarchyid-methods.aspx