The Results and Evaluation of the Filestream Based English Language Learning System

Mohamed Mhereeg[1], Asma Tawil[2] and Khairia Belghet[3]

***Abstract*—**The paper discusses the feasibility of constructing a SQL Server FILESTREAM based English Language Learning System (ELLS). It focuses on the results and evaluation phase of the system. It explains the prospect of storing and managing unstructured data (e.g. Images, video, Word, Excel, PDF, MP3, …etc) for educational purposes via utilizing FILESTREAM technique provided by SQL Server 2012, and explains how to maintain efficient storage and access to BLOB data. The paper seeks to utilize the combination of SQL Server 2012 features and the NTFS (New Technology File System) to improve the efficiency and performance of the ELLS system. The system also seeks to maintain the transactional consistency between the unstructured data and corresponding structured data. Three different BLOB storage techniques: SQL Server Varbinary(max), FILESTREAM through T_SQL, and FILESTREAM through NTFS were compared for performance purposes of the unstructured operations: insert, update, and delete. Different storage sizes have been used in the comparison ranging from 1MB to 6GB. According to the performance of the various sizes of BLOB data, FILESTREAM through T_SQL feature is best performed in managing and storing unstructured data, it is more efficient to use a Filestream when the typical file size is 1 MB or larger. The system also supports some maintenance operations such as backup, restore, and consistency checking.

***Keywords*—** Filestream, BLOB, NTFS, Varbinary, T_SQL.

I. INTRODUCTION

IN recent years, there has been an explosion in the volume of digital data created and stored by both individuals and organizations [1]. Historically, businesses have used computer systems and databases to store most of their business data in structured formats such as relational tables or fixed format files, and software applications have used these structured data stores to perform business tasks. Today however, a large proportion of an organization's data is typically stored in documents created with productivity tools such as Microsoft Office Excel and Microsoft Office Word, and advances in digital photography, document scanning, video production, and audio formats have further extended the range of unstructured data formats that are used for business data [1]. For example, in the context of relational database systems, it refers to data that can't be stored in rows and columns. Additionally, dramatic reductions in the cost of hardware storage and memory have significantly affected the amount and type of data that is stored in computer systems, and led to the emergence of a new generation of business application that merges traditional relational data structures with unstructured digital content [2]. This profusion of digital content means that organizations are now seeking to manage both relational data and unstructured data at the enterprise scale, and require a solution that comprehensively meets the needs of relational and non-relational data storage while reducing the cost of managing and building applications for that data [1]. Storing unstructured data such as text documents, images, and videos posed many challenges, such as how to maintain transactional consistency between the structured and unstructured data, how to manage backup and restore, and storage performance and scalability. Architects of applications that required the storage of binary large objects (BLOB) data could either store the data in the database or store it outside of the database with a reference stored in the database [3][4]. This paper focuses on the results and evaluation phase of a SQL Server FILESTREAM based English Language Learning System (ELLS). The system is evaluated taking into consideration the above mentioned issues including experiments, performance measurement, and capabilities.

Thus these issues are solved throughout the development phases of the project. The ELLS utilizes the FILESTREAM feature provided by SQL Server 2008/2012, which allows the storage and efficient access to BLOB data using a combination of SQL Server and the NTFS (New Technology File System). The ELLS has been evaluated in which these features have been applied. Three different methods have been used and compared in order to measure the performance changes over time; one uses a relational database to manage large objects, while the second method manages the objects as files in the file system, or as a combination of both. The outcome of the comparison will be presented throughout the paper

The system ultimately is intended to offer an excellent supporting material for in-class teaching, developing some language skills like reading, listening and speaking at certain levels. The system will also allow undertaking tests and providing automatic marking and feedback to students. This

[1]University of Tripoli, Tripoli, Libya, mmhereeg@msn.com
[2]Faculty of Engineering Technology, Tripoli, Libya, a.g.altawil@gmail.com
[3]The Libyan Academy, Tripoli, Libya, khbelgath@hotmail.com.

paper is based on the previous work entitled "Analysis and Design of a Filestream based English Language Learning System" and "The Implementation of a Filestream based English Language Learning System".

II. System Results

After the completion of the design [5] and the implementation [6] phases of the ELLS System through the utilization of Filestream feature in SQL Server 2012 to store and manage unstructured. The following sections discusses the results that have been accomplished.

A. Designing the Database with a FILESTREAM Feature Enabled

A special-purpose database has been created for an application called Learn_English. When using Filestream storage the folder where the binary data is placed must be specified. This folder is represented to SQL Server as a special file group defined to contain the Filestream. The primary file and the log files are defined automatically. All the files have been created within the "D:\Data" folder, when the database was created. All the FILESTREAM related data are stored in FileStreamData folder which is also known as the Learn_EnglishData Container.

B. Creating a Table with FILESTREAM Columns

Enhancing the storage and the performance of the unstructured content in the database was achieved by leveraging the NTFS file system via creating a table with FILESTREAM columns added to store educational lessons (video, image and audio). There is no need to make compromises in efficiency and complexity as it was used to be in the previous BLOB Storage Options when making the choice between storing BLOB data inside or outside the database. Instead, integrating BLOB data management with the rest of the data in the relational database without the need to manage the file system data separately. Defining the data as a FILESTREAM column in SQL Server also ensures data-level consistency between the relational data in the database and the unstructured data that is physically stored on the file system. A FILESTREAM column is a VARBINARY (MAX) column that has the FILESTREAM attribute enabled.

C. Insertion of Data

This system contributes in the unstructured data storage (Video, Audio and Image) of different sizes, ranging from under 1MB up to 6 GB, by using the FileStream feature to insert larger sizes.

Different files size have been inserted, which include structured and unstructured data. They files consist of tutorials exceeded 4 GB in volume, so as to make sure that the FileStream feature has the ability to deal with large-sizes exceeds the limit of T_SQL memory.

One could clearly see the overhead caused by executing several different statements, and sizes using Sql FileStream. The file handling overhead was seen in both file stream based solutions compared to a Varbinary solution.e.

Three different storage techniques are utilized and compared; one uses a relational database to store large objects, while the second stores the objects as files in the file system, or as a combination of both. The performance changes have been measured over time. Table 1 shows the collected readings and how this comparison was conducted.

Note: The measurement of storage has been made by milliseconds (ms).

TABLE I
COMPARISON BETWEEN THE MECHANISMS FOR DATA INSERTION

Type	Size	Sqlparameter	SqlFilestream + Filestream	Sqlparameter + Filestream
Jpg	70.5 Kb	10.0001 ms	70.0001 ms	30.0001 ms
Docx	95.1 Kb	30.0001 ms	90.0001 ms	60.0001 ms
mp4	1.39 Mb	23.0001 ms	100 ms	20 ms
Pdf	1.67 Mb	25.0001 ms	160 ms	20 ms
Rar	2.196 Mb	26.0001 ms	140 ms	19 ms
Exe	13.4 Mb	210 ms	393.002 ms	130 ms
mp4	27 Mb	631.0009 ms	892.0014 ms	310.0004 ms
wmv	169 Mb	4594.0707 ms	3065.0475 ms	1102.517 ms
wmv	227 Mb	7753.2136 ms	4385.0097 ms	1466.403 ms
wmv	233Mb	6742.0095 ms	4581.0065 ms	1591.002 ms
wmv	453 Mb	15518.0361 ms	12064.017 ms	2241.005 ms
wmv	453Mb	16903.0182 ms	12414.019 ms	3471.006 ms
wmv	453Mb	18903.0182 ms	13414.019 ms	4471.006 ms
VOB	800 Mb	25025.02 ms	14024.02 ms	5241.007 ms
VOB	1 Gb	55025.02 ms	35024.02 ms	10241.007 ms

D. Updating of Data

Modification of FILESTREAM data can be done via using streaming API, which is pretty much the same as what we saw in the Insert function. Prior accessing the data the user needs to access the PathName(), starts a transaction and then obtains a transaction context before modifying the data using the SqlFileStream class.

Three different storage techniques are utilized and compared; one uses a relational database to update large objects, while the second updates the objects as files in the file system, or as a combination of both. The performance changes have been measured over time. Table 2 shows the collected readings and how this comparison was conducted.

TABLE II
COMPARISON BETWEEN THE DATA UPDATING MECHANISMS

Type	Size	Sqlparameter	SqlFilestream + Filestream	Sqlparameter + Filestream
JPG	70.5 Kb	20.0003 ms	149.0085 ms	95.0054 ms
exe	573 Kb	30.0062 ms	150.091 ms	98.0373 ms
wmv	1.3 Mb	20 ms	60.0013 ms	10 ms
wmv	1.77 Mb	30.0001 ms	150.0002 ms	20 ms
wmv	56 Mb	3941.0084 ms	1040.0015 ms	370.0006 ms
wmv	180 Mb	6444.8114 ms	5214.8077 ms	2460.01 ms
wmv	188 Mb	7411.339 ms	4114.2354 ms	1019.0926 ms
wmv	196 Mb	7387.0118 ms	3720.0066 ms	1340.0018 ms
wmv	196 Mb	8440.0135 ms	3949.01 ms	1530.0049 ms
wmv	196 Mb	8538.0136 ms	4019.01 ms	1946.01 ms
wmv	227 Mb	8455.2149 ms	5335.2094 ms	3260.406 ms
wmv	250 Mb	8233.0158 ms	4821.0068 ms	3730.005 ms
wmv	321 Mb	9423.0092 ms	5872.0083 ms	1680.0024 ms
mp4	358 Mb	15831.5623 ms	6244.3571 ms	2894.1655 ms
mp4	500 Mb	25415.8265 ms	17695.0121 ms	4360.2494 ms

E. Deletion of Data

When the entire row containing the FILESTREAM data is deleted from the table, a regular DELETE command will delete the row from the table and remove all the FILESTREAM data associated with it from the FILESTREAM data container using the CHECKPOINT command.

Three different storage techniques are utilized and

compared; one uses a relational database to delete large objects, while the second stores the objects as files in the file system, or as a combination of both. The performance changes have been measured over time. Table 3 shows the collected readings and how this comparison was conducted:

TABLE III
COMPARISON BETWEEN THE MECHANISMS FOR DATA DELETION

Type	Size	Sqlparameter	SqlFilestream +Filestream	Sqlparameter + Filestream
jpg	70 Kb	20.0001 ms	40.0009 ms	35.0002 ms
docx	95 Kb	35.0021ms	65.0025 ms	45.0037 ms
exe	573 Kb	30.0017 ms	70.0041 ms	55.0013 ms
exe	1.39 Mb	50.0003 ms	23.0007 ms	20.0002 ms
wmv	1.77 Mb	61.003 ms	50.0042 ms	25.0002 ms
mp3	13.4Mb	63.0016 ms	40.001 ms	30.0004 ms
mp3	27Mb	70.0012 ms	45.0009 ms	35.0004 ms
mp3	36Mb	76.0044 ms	57.0033 ms	22.0012 ms
wmv	180 Mb	80.0002 ms	73.0042 ms	30.0017 ms
wmv	196 Mb	147.009 ms	69.0046 ms	33.0019 ms
wmv	196Mb	148.0085 ms	70.0031 ms	35.0004 ms
wmv	196 Mb	184.491 ms	100.1099 ms	61.2717 ms
wmv	233 Mb	344.008 ms	100.002 ms	50.0003 ms
mp4	358 Mb	566.564 ms	209.019 ms	96.1628 ms
mp4	500 Mb	545.037 ms	160 ms	60.0012 ms

F. Access and Retrieval of FILESTREAM Data

After storing data in a FILESTREAM column, the files can be accessed using T_SQL transactions or using Win32 application programming interfaces (APIs). T_SQL can access the data as if they were stored in the database.

a) Accessing FILESTTREAM Data Using TSQL

Even though the actual data of the FILESTREAM enables column stored in the NT File System, it would be completely transparent to the T_SQL code. Data can be accessed in the FILESTREAM column just like any other column of the table. The user can observe the page loading the details of the items in the database and displaying the thumbnail images associated with those items

b) Accessing FILESTREAM Data with Managed API

There are two key pieces of information that need to be obtained in order to access the FILESTREAM data using Win32 Streaming. First, the file system transaction context is needed, which is returned by the GET_FILESTREAM_TRANSACTION_CONTEXT function, this function returns NULL if a transaction has not yet been established. Second, the logical UNC path to the file holding the BLOB on the server is also needed, which is returned by the PathName method on a varbinary(max) FILESTREAM value instance.

G. Backup and Restore for FILESTREAM Database

The benefit of the FILESTREAM feature over the traditional methods of storing the BLOB data in the file system is that a FILESTREAM database backup contains both the relational data and the BLOB data stored in the FILESTREAM data container. This removes the administrative overhead of having to maintain separate backups of the disk files, which is necessary when using the traditional BLOB storage methods. Likewise, when the full backup of a FILESTREAM database is restored to another location, the FILESTREAM data is also restored, and is available along with the relational data

H. Enhancing the Security of the System

Protection and management of users' access permissions in terms of storage methods have been configured as the following:

1. When LOB data is stored within the database, the security can be managed at the SQL Server level. This reduces the administrative complexity and aligns LOB security with the same security processes applied to the associated relational data.
2. If LOB data is stored outside the database, separate measures must be put in place to secure the file system. The relational data and LOB data are disconnected, and managing user access permissions is a major challenge.

III. SYSTEM EVALUATION

This section discusses the evaluation of the system to ensure that the system is designed and developed to meet the specified requirements of the system. The main reason for the evaluation is to verify how well the system fulfills the intended objectives to maintain the unstructured data via utilizing the Filestream technique. This evaluation was done by the system developer only and no evaluation was done by the users. The following criteria used in the system evaluation was done by the developer.

a) Functionality

The functionality feature of the criteria evaluates how well the developed system meets the predefined functional requirements [5]. Each functional requirement was evaluated and checked against the capability of the developed system to ensure that the functions operate in an efficient manner. The entire system was also evaluated to test the functionality, efficiency, and the correctness of the outputs. For example, links and buttons were tested to ensure that the users are redirected to the correct pages and that the data are being inserted in the correct tables.

b) The performance of the Criteria

This criterion determines how the system performs in terms of responsiveness and accuracy of data. Measurement of the data accuracy was done in all forms, where the fields were validated whether the correct data types are inserted, and also checking whether the error messages for wrong inputs are fired whenever an error occurs.

A. Evaluation of the Interface

The performance of standards for GUIs have been evaluated. The system is carefully designed to be user friendly. Its design is very intuitive and informative so that even people with little experience in computers can easily use this system. A user friendly feature of this system is 'title based' information for each item in each page. Moving the mouse pointer over any item such as buttons, text boxes, images, etc will display the title information below the mouse pointer indicating its functionality or intent. This greatly simplifies interaction between the system and the users.

The following features can also be obtained:

Allows user to complete tasks effectively.

The system should prevent the user from making errors and allow error recovery.

Informative error messages when things go wrong.

All menu functions were tested, by invoking the corresponding functionality properly.

Providing a meaningful naming system for all output data

All menu functions and sub functions were verified for correctness.

All required fields were not left blank. For example when leaving a username or password text fields blank, the error messages will appear.

B. Performance Evaluation

With a small amount of binary data, it is not efficient to use a file stream. This is because it needs extra overhead like file creation and handling. These operations are not needed when predefined database files are used. However, with larger files, file streams are quite efficient. The following charts show the elapsed times for managing data in milliseconds using different techniques. The key specifications for the computer used were:

SQL Server and client application on the same machine

Processor: Intel Core7 Duo, 1.8 MHz

8 GB physical memory

Database files on drive C

Files uploaded from drive D

Drives C: and D: on separate physical SATA disk drives

As the below figures illustrate, compared with the time consumption in data entry for different storage methods, the relative performance for read, update and delete operations are presented:

BLOB data are stored in FILESTREAM format and accessed through the WIN32 streaming APIs. The times includes getting a transaction context from SQL Server, getting the file path, doing the operation, closing the file, and committing the transaction in SQL Server.

BLOB data are stored in FILESTREAM format and manipulated through T-SQL.

1. BLOB data are stored in Varbinary(max) format (and obviously manipulated through T_SQL).

C. Performance of Data Insertion

Figure 1 shows the performance of the insertion of unstructured data in accordance with the table 1.

In this measurements, it can clearly be seen the overhead caused by executing several different statements using the Sql FileStream. In addition, the file handling overhead is seen in both file stream based solutions compared to a varbinary inside the database. Testing the speed of insertion is applied on 70.55Mb, 95.1Kb, 1.39Mb, 1.67MB, 2.19MB, 13.4Mb, 27MB, 169MB, 227Mb, 233MB, 453MB, 800MB, and 1GB of data from the table. This figure below shows the difference in the performance results.

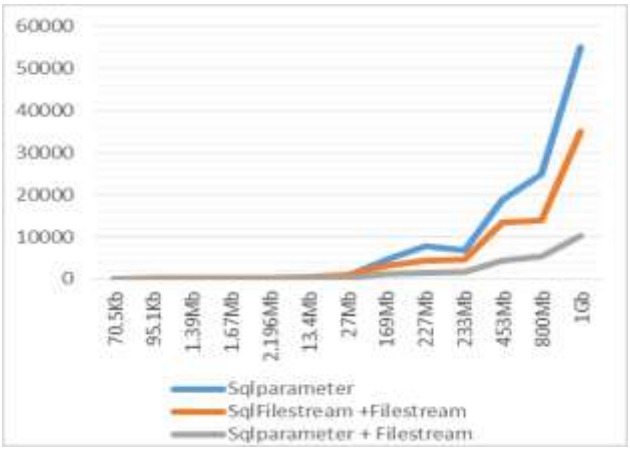

Fig. 1: Performance of the data insertion.

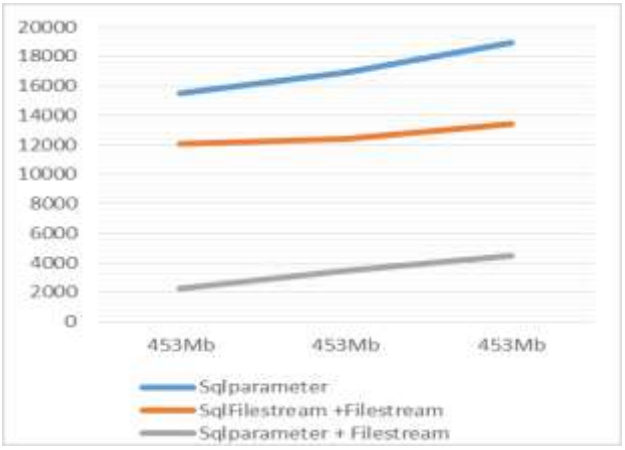

Fig. 2: Data Insertion (Equal row size of 453 MB)

According to figure 1, FILESTREAM through T_SQL has shown best performance. FILESTREAM through NTFS is a bit slower, but when inserting data larger than 1MB in volume, Varbinary(max) needed more time to perform the storing operation.

Figure 2 above shows the relative throughput of inserting same size of BLOB data using Varbinary(max), FILESTREAM through T_SQL, and FILESTREAM through NTFS. This graph shows 453KB file repeated 3 times for each measurement. According to this figure FILESTREAM through T_SQL has shown the best performance. However, FILESTREAM through NTFS and Varbinary(max) are noticeably slower

Based on these measurements, it's more efficient to use a file stream when the file size is about 1MB or larger. If files are small (clearly under 1MB), a traditional varbinary performs better.

D. Performance of Data Updating

Figure 3 shows the relative throughput of update of various sizes of BLOB data (70kb, 573kb, 1.39Mb, 1.77Mb, 56Mb,180Mb, 188Mb, 196Mb, 227Mb, 250Mb, 321Mb,358Mb and 500Mb) using Varbinary(max), FILESTREAM through T_SQL, and FILESTREAM through NTFS. This figure shows the performance results of the update operations of unstructured data in accordance with the table 2.

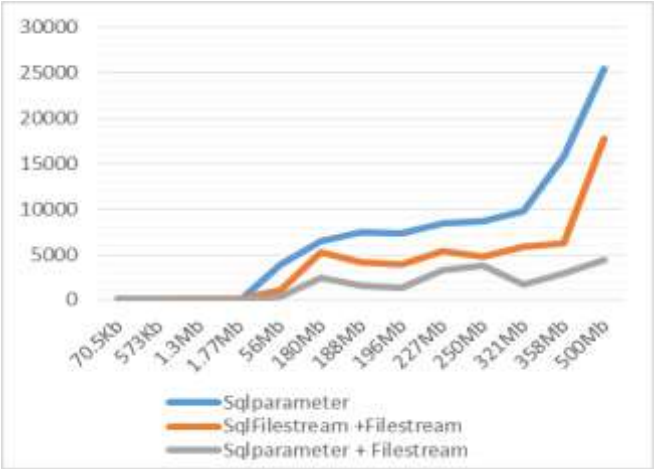

Fig. 3: Performance of the data updating

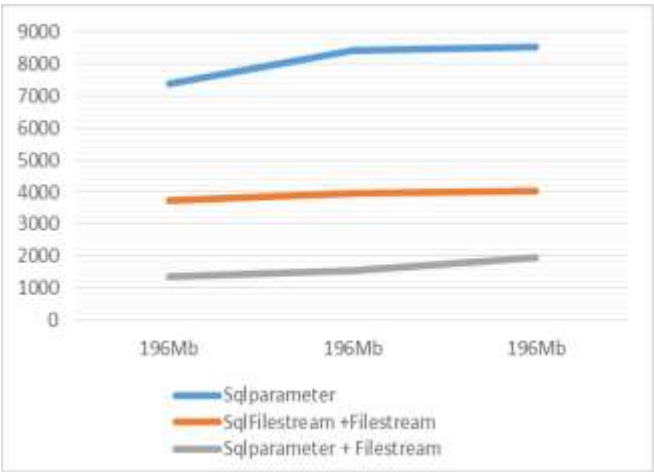

Fig. 4: Data Updating (Equal row size of 196 MB)

According to the Figure 3, FILESTREAM through T_SQL has shown best performance. FILESTREAM through NTFS is a bit slower. But when updating data larger than or equals 1MB in volume, Varbinary(max) needed more time to perform the update operations.

Figure 4 above shows the relative throughput of update of same size of BLOB data using Varbinary(max), FILESTREAM through T_SQL, and FILESTREAM through NTFS. This figure shows 196MB file repeated 3 times for each measurement. According to this figure, FILESTREAM through T_SQL has shown best performance. However, FILESTREAM through NTFS is a bit slower, but when updating data larger than or equals 1MB in volume, Varbinary (max) needed more time to perform the updating operations.

Based on these measurements, it's more efficient to use a T_SQL file stream technique for data updating.

E. Performance of Data Deletion

Figure 5 shows the relative throughput of deletion of various sizes of BLOB data (70.5Kb, 95Kb, 573Kb, 1.39Mb, 1.77Mb, 13.4Mb, 27Mb, 56Mb, 180MB, 196Mb, 233MB, 358MB, and 500MB) using Varbinary(max), FILESTREAM through T_SQL, and FILESTREAM through NTFS. Figure 5 shows the performance results of the deletion of unstructured data in accordance with the table 3.

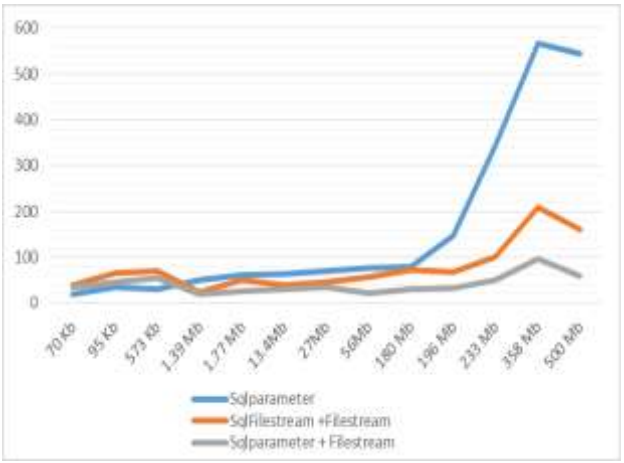

Fig. 5: Performance of the data deletion

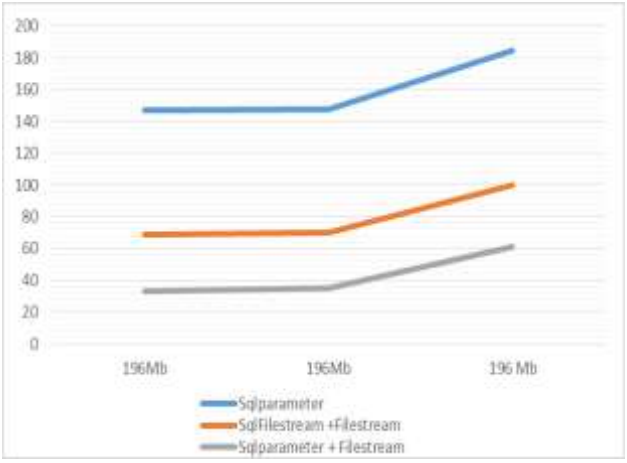

Fig. 6: Data Deletion (Equal row size of 196 MB)

According to the Figure 5, FILESTREAM through T_SQL has shown best performance. However, FILESTREAM through NTFS is a bit slower. But when deleting data larger than 1MB in volume, Varbinary(max) needed more time to perform the data deletion operations.

Graph 6 shows the relative throughput of Deletion of same size of BLOB data using Varbinary(max), FILESTREAM through Transact-SQL, and FILESTREAM through NTFS. This figure shows a file size of 196MB being repeated 3 times for each measurement. Based on these measurements, it is more efficient and faster to use a T_SQL Filestream to delete files. However, using Sql FileStream would not give any performance advantage, but when all the data is deleted using a different storage techniques, FILESTREAM through NTFS is much faster than FILESTREAM through T_SQL and Varbinary(max). Table 4 below shows the size of the data in each table.

TABLE VI
COMPARISON BETWEEN THE MECHANISMS OF ALL DATA DELETION FROM THE TABLE

Type	Size	Sqlparameter	Sqlparameter + Filestream	SqlFilestream +Filestream
All data	2Gb	20453.542 ms	774.0442 ms	429.0245 ms
All data	3.5 Gb	24092.1426 ms	1660.002 ms	340.002 ms
All data	5 Gb	41547.848 ms	1520.0014 ms	326.00 ms

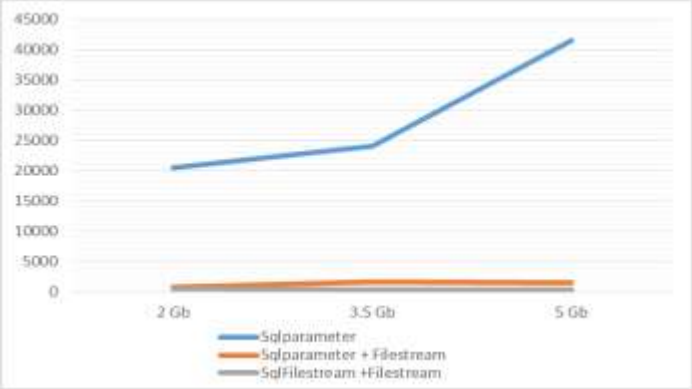

Fig. 7: Performance of the deletion of all data

IV. CONCLUSION

The Filestream through T_SQL was utilized and proved successful in retrieving and storing unstructured data in the NTFS file system. This is accomplished by storing the structured data in the database files and the unstructured BLOB data in the file system, while maintaining transactional consistency between the two stores. The comparison between the different BLOB storage techniques was conducted for performance purposes of the unstructured operations: insert, update, and delete. According to the performance of various sizes of BLOB data ranging from below 1MB up to 6 GB using Varbinary(max), FILESTREAM through T_SQL, and FILESTREAM through NTFS, It was clearly noticed that T_SQL access of FILESTREAM data is several times faster than T_SQL access of Varbinary(max) data as data size increases. Furthermore, Based on these comparisons, FILESTREAM Feature through T_SQL is best performed in managing and storing unstructured data, it is more efficient to use this technique when the typical file size is about 1 MB or larger. If files are small (clearly under 1 MB), a traditional varbinary performs better. Based on the outcome obtained from comparing these BLOB Storage techniques, The FILESTREAM feature is the only solution that provides transactional consistency of structured and unstructured data as well as security, and excellent streaming performance.

REFERENCES

[1] Microsoft (July 2008). Managing Unstructured Data with SQL Server 2008. Microsoft SQL Server 2008. [Online]. pp 1-2. Available: http://download.microsoft.com/download/a/c/d/acd8e043-d69b-4f09-bc9e-4168b65aaa71/ SQL2008UnstructuredData.doc

[2] J. Azemović. (2012). Varbinary vs. Filestream and other BLOB issues. BEGIN TRANSACTION. [online] Available: http://sqltales.wordpress.com/2012/05/15/varbinary-vs-filestream-and-other-blob-issues .

[3] S. Tinline-Jones. (2011). FILESTREAM Design and Implementation Considerations. SQL Server Technical Article. [online]. pp 5-6. Available:http://download.microsoft.com/download/d/9/4/d948f981-926e-40fa-a026 5bfcf076d9b9/ FILESTREAM%20Design%20and%20Implementation%20Considerations.docx

[4] J. Sebastian and S. Aelterman, The Art of SQL Filestream , Simple Talk Publishing, 2012,ch1,pp.23-45.

[5] M. R. Mhereeg, A. G. Tawil, "The Analysis and Design of a Filestream Based English Language Learning System," in Proc. 2015 ICCIT15 Conf., 2015.

[6] A. G. Tawil, M. R. Mhereeg, "The Implementation of a Filestream Based English Language Learning System," in Proc. 2015 ITMC15 Conf., 2015, pp. 39-47.