[**Таблицы файлов в SQL Server 2012**](http://ts-soft.ru/blog/filetable)

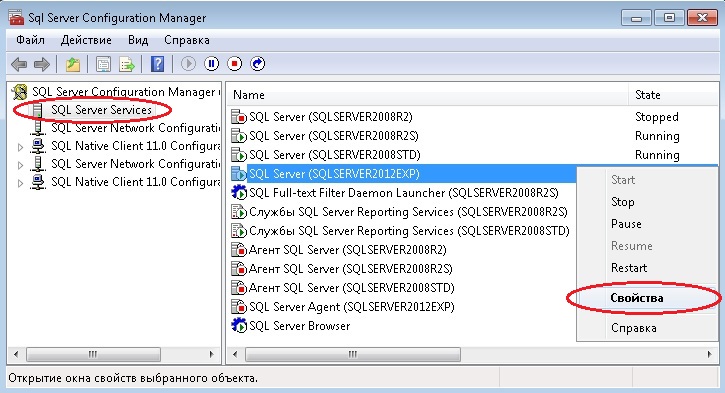
21 марта 2013 в 13:23

[Разработка](http://ts-soft.ru/blog/category/development)

Традиционным способом хранения в БД неструктурированных двоичных файлов, таких как видео, аудио, картинки и т. п. является определение в таблице поля типа varbinary(max) или применение FILESTREAM, появившегося в SQL Server 2008. В этой статье будет рассмотрен новый подход с использованием таблиц FileTable, которые появились в SQL Server 2012. Технология FileTable основывается на функционале FileStream, при этом расширяя его возможности. В FileTable для организации структуры каталогов используется тип HierarchyId, о котором вы можете прочитать в [моей предыдущей статье](http://ts-soft.ru/blog/hierarchyid/).

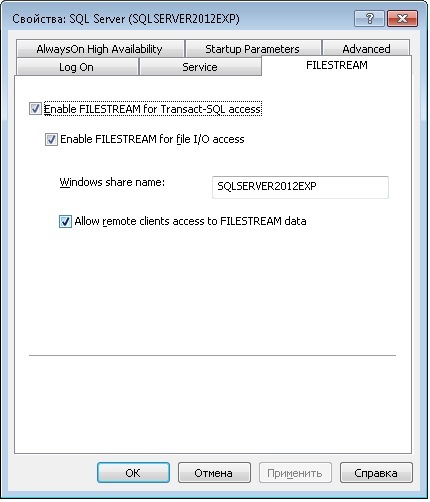
**Настройка сервера для поддержки FileTable**

Для корректной работы FileTable сначала необходимо произвести настройку SQL Server. В первую очередь нужно включить поддержку FileStream на уровне сервера. Это можно сделать через SQL Server Configuration Manager. Перейдите в меню  
Пуск > Все программы > Microsoft SQL Server 2012 > Configuration Tools > SQL Server Configuration Manager. Далее в левой панели выберите SQL Server Services, при этом в правой панели откроются все службы SQL Server. Выберите экземпляр SQL Server, на котором нужно включить поддержку FileStream, и перейдите в «Свойства» (рис. 1).

[](http://ts-soft.ru/blog/wp-content/uploads/2013/02/ChooseSqlServerInstatnce.jpg)

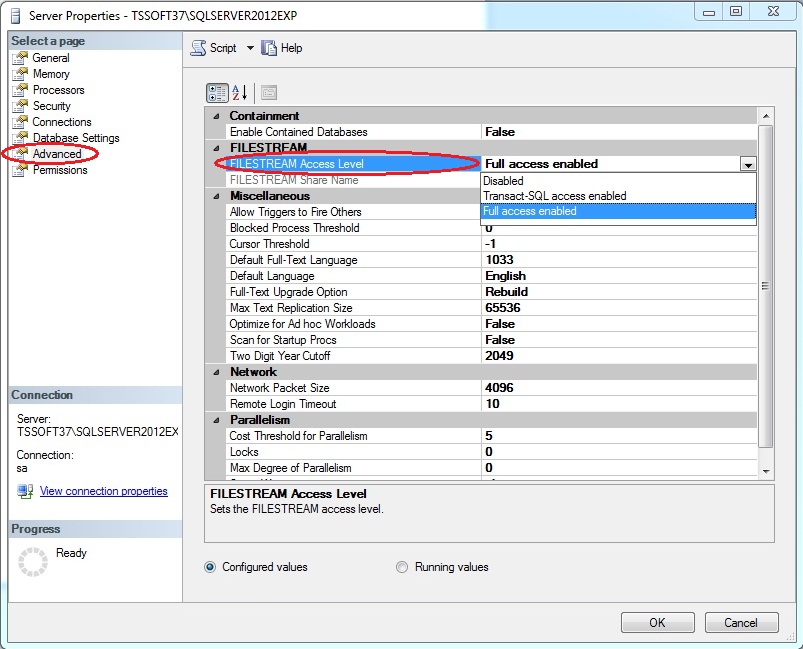
*Рис. 1. Выбор экземпляра SQL Server, на котором необходимо включить поддержку FileStream*

В окне свойств перейдите на вкладку FILESTREAM. Чтобы включить поддержку FileStream на сервере, отметьте флаг **«Enable FILESTREAM for Transact-SQL access»**. Если вы хотите иметь возможность доступа к FileStream из файловой системы Windows, то отметьте флаг **«Enable FILESTREAM for file I/O streaming access»**. Также здесь вы можете задать имя общей папки Windows для данных FileStream. Если нужно, чтобы удалённые клиенты могли иметь доступ к данным FileSteam, хранящимся в общей папке, то отметьте флаг **«Allow remote clients to have streaming access to FILESTREAM data»** (рис. 2). Все эти настройки можно также произвести при установке SQL Server.



*Рис. 2. Включение поддержки FILESTREAM в SQL Server Configuration Manager*

Далее откройте SQL Server Management Studio, зайдите в свойства сервера, перейдите на вкладку Advanced и в поле FileStream Access Level выберите **«Full access enabled»**, как показано на рис. 3.

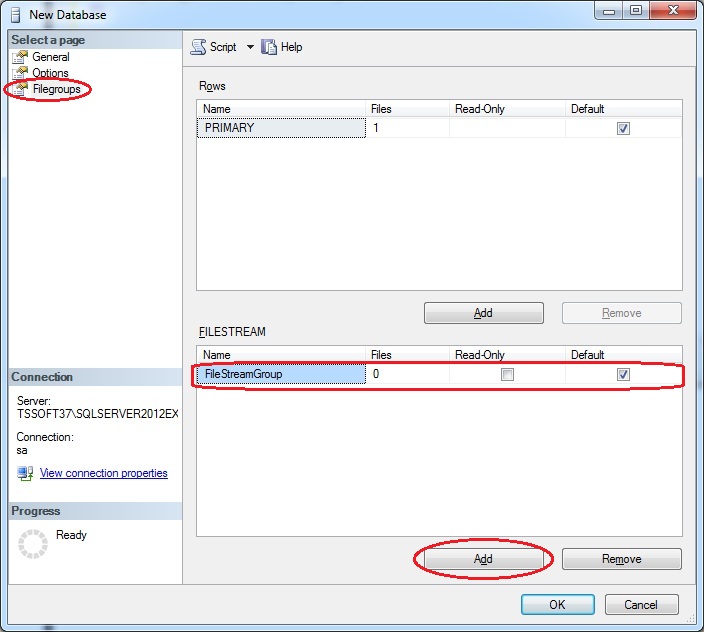
[](http://ts-soft.ru/blog/wp-content/uploads/2013/02/EnableFileStreamInManagementStudio.jpg)

*Рис. 3. Включение поддержки FILESTREAM в SQL Server Management Studio*

Это также можно сделать с помощью скрипта, приведённого ниже:

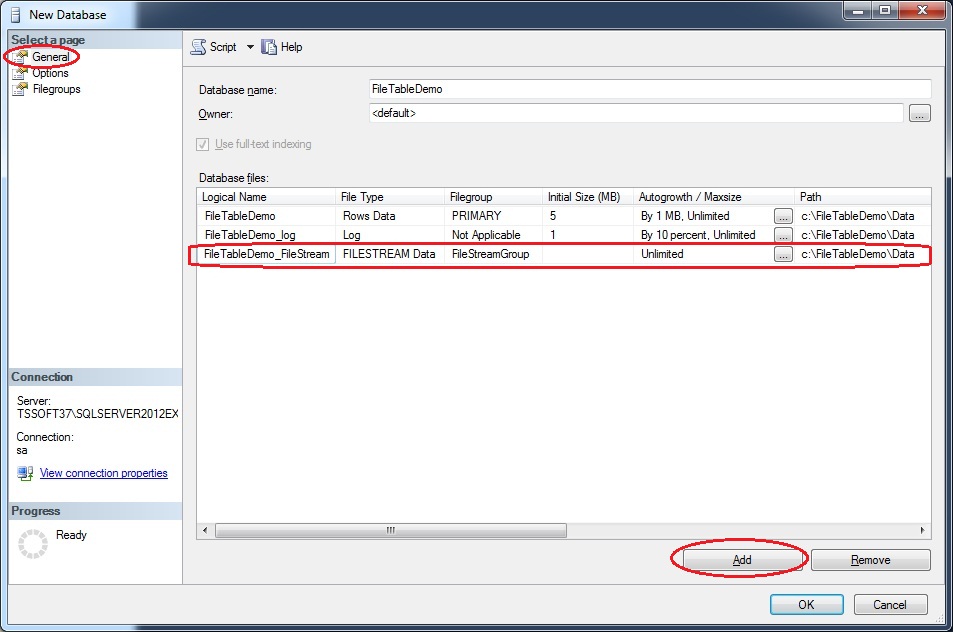
1. USE master
2. GO
4. EXEC sp\_configure filestream\_access\_level, 2
5. GO
7. RECONFIGURE
8. GO

Далее необходимо создать или модифицировать базу данных, которая будет поддерживать FileStream. Для этого в свойствах базы данных перейдите на вкладку Filegroups и создайте файловую группу согласно рис. 4.

[](http://ts-soft.ru/blog/wp-content/uploads/2013/02/AddFileGroup.jpg)

*Рис. 4. Создание файловой группы для поддержки FileStream*

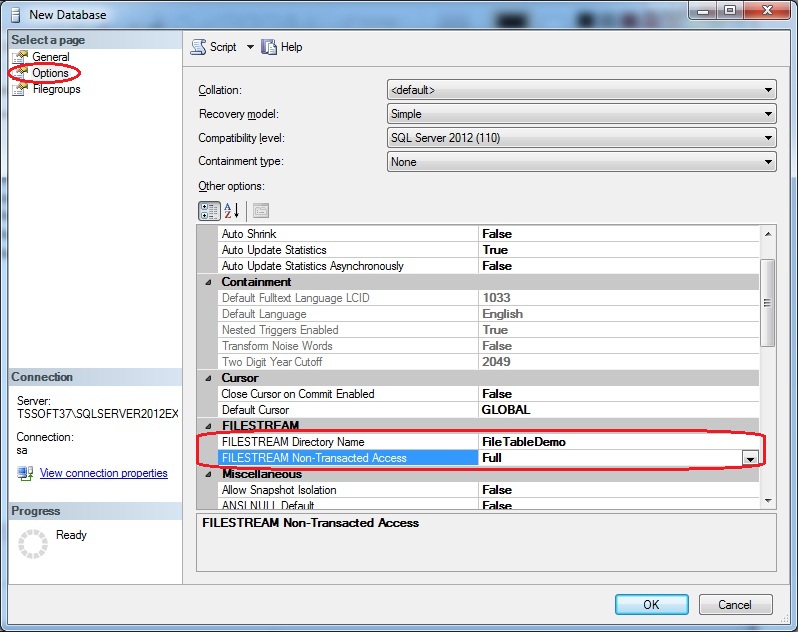
Затем перейдите на вкладку General и создайте файл, как показано на рис. 5. Тип данных установите в значение **«FILESTREAM Data»**. В поле Filegroup выберите ранее созданную файловую группу. Также в поле Path задайте путь для хранения данных FileStream, например, c:FileTableDemoData. Этот путь должен существовать в файловой системе.

[](http://ts-soft.ru/blog/wp-content/uploads/2013/02/AddFileToFileGroup.jpg)

*Рис. 5. Создание файла базы данных для поддержки FileStream*

*(=Option для созданной базы)*

Теперь перейдите на вкладку Options. В разделе FILESTREAM в поле FILESTREAM Directory Name задайте имя директории. Эта директория будет создана в общей папке сервера, которая была задана при включении FileStream, а в неё, в свою очередь, будут помещаться папки для каждой таблицы FileTable, созданной в этой базе данных. Также, если вы хотите иметь доступ к файлам, хранящимся в FileTables, из файловой системы, необходимо установить нетранзакционный доступ, выбрав в поле FILESTREAM Non-Transacted Access значение **«Full»** (рис. 6).

[](http://ts-soft.ru/blog/wp-content/uploads/2013/02/AddFileStreamDirectoryName.jpg)

*Рис. 6. Настройка нетранзакционного доступа и имени директории для FileTables*

Все эти действия можно сделать с помощью следующего скрипта:

1. USE master
2. GO
4. CREATE DATABASE FileTableDemo ON PRIMARY
5. (NAME = 'FileTableDemo', FILENAME = 'c:FileTableDemoDataFileTableDemo.mdf')
6. LOG ON
7. (NAME = 'FileTableDemo\_log', FILENAME = 'c:FileTableDemoDataFileTableDemo\_log.ldf')
8. WITH FILESTREAM
9. (
10. NON\_TRANSACTED\_ACCESS = FULL,
11. DIRECTORY\_NAME = 'FileTableDemo'
12. )
13. GO
15. ALTER DATABASE FileTableDemo
16. ADD FILEGROUP FileStreamGroup CONTAINS FILESTREAM
17. GO
19. ALTER DATABASE FileTableDemo
20. ADD FILE
21. (
22. NAME= 'FileTableDemo\_FileStream',
23. FILENAME = 'c:FileTableDemoDataFileTableDemo\_FileStream'
24. )
25. TO FILEGROUP FileStreamGroup
26. GO

**Создание таблицы FileTable**

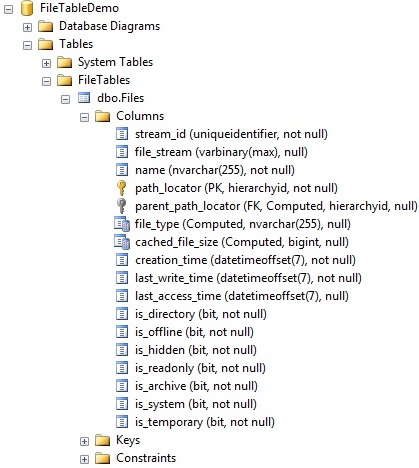
Таблица FileTable является специализированной пользовательской таблицей с заранее заданной и фиксированной схемой, поэтому при её создании нет необходимости указывать список столбцов. Синтаксис для создания такой таблицы выглядит следующим образом:

1. USE FileTableDemo
2. GO
4. CREATE TABLE Files AS FILETABLE
5. WITH
6. (
7. FILETABLE\_DIRECTORY = 'Files',
8. FILETABLE\_COLLATE\_FILENAME = database\_default
9. )
10. GO

FILETABLE\_DIRECTORY — это корневой каталог для всех файлов и каталогов, хранящихся в FileTable. Если при создании не указано имя каталога, то в качестве него используется имя самой таблицы.  
FILETABLE\_COLLATE\_FILENAME указывает имя параметров сортировки, применяемых к столбцу Name в таблице FileTable. Если значение не указано или задано как database\_default, столбец унаследует параметры сортировки текущей базы данных.  
С учётом вышесказанного, синтаксис создания FileTable может быть упрощён до следующего:

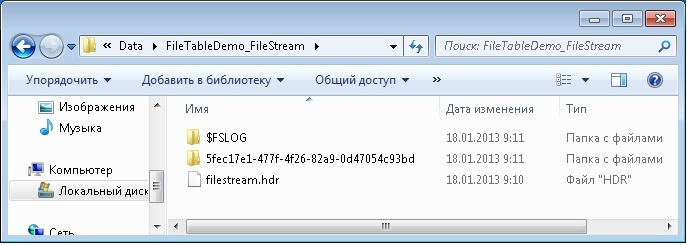
1. USE FileTableDemo
2. GO
4. CREATE TABLE Files AS FILETABLE
5. GO

После создания FileTable перейдите к базе данных, раскройте узел FileTables и затем раскройте список столбцов созданной таблицы. Вы увидите структуру, показанную на рис. 7.



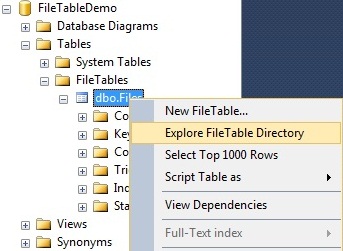
*Рис. 7. Структура таблицы FileTable*

В таблице хранится большинство атрибутов файла, а также его имя, размер, тип и родительский католог. Для указания каталогов существует флаг is\_directory. В таблицы такого типа нельзя добавлять новые пользовательские столбцы, а также удалять или изменять существующие, но можно создавать пользовательские индексы, триггеры или ограничения.  
После проделанных операций вы можете посмотреть структуру папок, которые создал SQL Server. В директории c:FileTableDemoData была создана папка FileTableDemo\_FileStream. Все данные FileStream будут сохраняться в этой папке. Сейчас, если вы перейдёте в неё, то увидите там файл filestream.hdr и две директории (рис. 8). Они нужны для корректной работы FileStream и не должны удаляться или модифицироваться вручную.



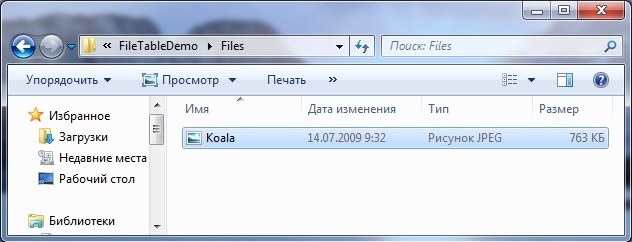
*Рис. 8. Папка, созданная SQL Server для хранения данных FileStream*

Также вы можете перейти в общую сетевую папку, созданную SQL Server. В обобщённом виде путь к этой папке выглядит следующим образом:\SERVERNAMEFILESTREAM\_WINDOWS\_SHARE\_NAMEFILESTREAM\_TABLE\_NAME FILETABLE\_DIRECTORY.  
В нашем случае это будет выглядеть так: \ServerSqlServer2012ExpFileTableDemoFiles.  
В эту директорию можно попасть и с помощью SQL Server Management Studio. Для этого кликните правой кнопкой на созданной таблице FileTable и выберите пункт Explore FileTable Directory (рис. 9) Для доступа к директории FileTable требуются соответствующие разрешения SQL Server для таблицы. Если у пользователя нет таких разрешений, то доступ к этой папке из файловой системы будет запрещен.



*Рис. 9. Доступ к папке FileTable из SQL Server Management Studio*

В настоящий момент эта директория пуста. Вы можете помещать в неё файлы и папки обычным копированием, как показано на рис. 10.

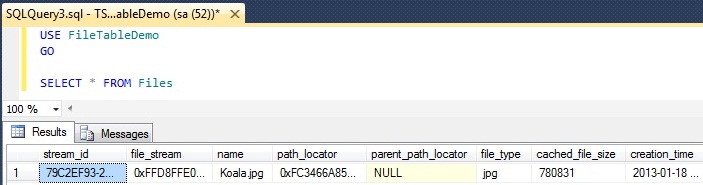


*Рис. 10. Копирование файлов в папку FileTable через файловую систему*

Теперь сделаем выборку из нашей таблицы с помощью обычного скрипта:

1. USE FileTableDemo
2. GO
4. SELECT \* FROM DemoFileTable
5. GO

Получим следующий результат, показанный на рис. 11.



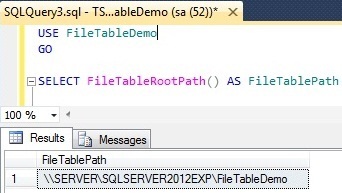
*Рис. 11. Выборка файлов из таблицы FileTable*

Как и следовало ожидать, наш файл появился в таблице со всеми необходимыми атрибутами.  
Также данные в таблицу можно добавлять, обновлять и удалять с помощью обычных инструкций INSERT, UPDATE и DELETE языка Transact-SQL.

**Рабта с каталогами и путями в таблицах FileTable**

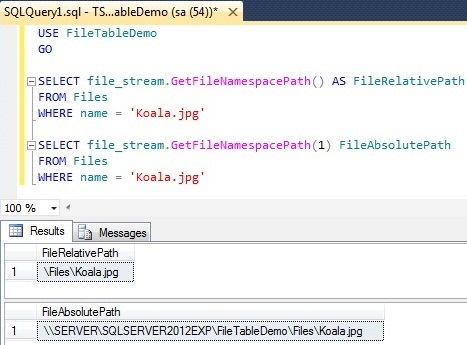
Для работы с каталогами и путями таблиц FileTables существуют 3 функции:

* FileTableRootPath([<’FileTable\_Name’>], [<option>]) — получает корневой путь для конкретной таблицы FileTable или для текущей базы данных. Первый параметр — это имя таблицы FileTable, для которой необходимо вернуть путь. Он является необязательным, значение по умолчанию — это текущая база данных. Второй параметр — это целочисленное выражение, определяющее способ форматирования серверных компонентов пути. Использование этой функции приведено на рис. 12.



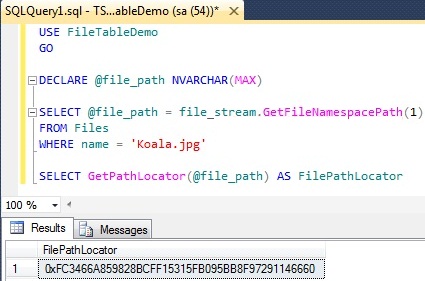
*Рис. 12. Пример функции FileTableRootPath*

* <column\_name>.GetFileNamespacePath([<is\_full\_path>], [<option>]) — возвращает путь к файлу или каталогу в таблице FileTable. <column\_name> — это имя столбца file\_stream типа varbinary(max) в таблице FileTable. <is\_full\_path> — это целочисленное выражение, указывающее, какой путь возвращать — относительный или абсолютный. Он является необязательным, и по умолчанию выводится относительный путь внутри каталога уровня базы данных. Параметр <option> аналогичен одноимённому параметру в пункте выше. Использование этой функции приведено на рис. 13.



*Рис. 13. Пример функции GetFileNamespacePath*

* GetPathLocator(<’filenamespace\_path’>) — возвращает значение идентификатора path\_locator для заданного файла или каталога в таблице FileTable. <filenamespace\_path> — это путь к пространству имен в FileTable. Он имеет тип nvarchar(max). Использование этой функции приведено на рис. 14.



*Рис. 14. Пример функции GetPathLocator*

**Преимущества использования FileTable**

Особенность FileTables заключается в том, что эта технология объединяет компонент SQL Server Database Engine с файловой системой NTFS, размещая данные больших двоичных объектов (BLOB) типа varbinary(max) в файловой системе. Исходя из этого, можно выделить следующие преимущества:

* возможен нетранзакционный доступ через файловую систему Windows, при этом производительность равняется производительности файловой системы;
* для кэширования файлов в хранилище FileTable используется системный кэш NT, при этом SQL-буфер используется только для обработки запросов;
* размер двоичного файла ограничен только размером NTFS-раздела;
* возможен также транзакционный доступ с помощью обычных инструкций SELECT, INSERT, UPDATE и DELETE;
* доступно использование интегрированных служб SQL Server, таких как резервное копирование, встроенная поддержка безопасности, полнотекстовый поиск и т. д.

### Сохранение в базу данных файлов

Рассмотрим, как мы можем сохранять файлы, в частности, файлы изображений в базу данных. Для этого добавим в базу данных новую таблицу Images с четырьмя столбцами: Id (первичный ключ и идентификатор, имеет тип int), FileName (будет хранить имя файла и имеет тип nvarchar), Title (будет хранить заголовок файла и также имеет тип nvarchar) и ImageData (будет содержать бинарные данные файла и имеет тип varbimary(MAX)).

Определим код, в котором будут загружаться данные в таблицу:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | private static void SaveFileToDatabase()  {      string connectionString = @"Data Source=.\SQLEXPRESS;Initial Catalog=usersdb;Integrated Security=True";      using (SqlConnection connection = new SqlConnection(connectionString))      {          connection.Open();          SqlCommand command = new SqlCommand();          command.Connection = connection;          command.CommandText = @"INSERT INTO Images VALUES (@FileName, @Title, @ImageData)";          command.Parameters.Add("@FileName", SqlDbType.NVarChar, 50);          command.Parameters.Add("@Title", SqlDbType.NVarChar, 50);          command.Parameters.Add("@ImageData", SqlDbType.Image, 1000000);            // путь к файлу для загрузки          string filename = @"C:\Users\Eugene\Pictures\cats.jpg";          // заголовок файла          string title = "Коты";          // получаем короткое имя файла для сохранения в бд          string shortFileName = filename.Substring(filename.LastIndexOf('\\')+1); // cats.jpg          // массив для хранения бинарных данных файла          byte[] imageData;          using (System.IO.FileStream fs = new System.IO.FileStream(filename, FileMode.Open))          {              imageData = new byte[fs.Length];              fs.Read(imageData, 0, imageData.Length);          }          // передаем данные в команду через параметры          command.Parameters["@FileName"].Value = shortFileName;          command.Parameters["@Title"].Value = title;          command.Parameters["@ImageData"].Value = imageData;            command.ExecuteNonQuery();      }  } |

После выполнения этой программы в базе данных появится соответствующая запись:

### Извлечение файлов из базы данных

В прошлой теме мы рассмотрели, как добавить файл в базу данных. Теперь произведем обратную операцию - получим файл из БД. Вначале определим класс файла, который упростит работу с данными:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | public class Image  {      public Image(int id, string filename, string title, byte[] data)      {          Id = id;          FileName = filename;          Title = title;          Data = data;      }      public int Id { get; private set; }      public string FileName { get; private set; }      public string Title { get; private set; }      public byte[] Data { get; private set; }  } |

Затем в коде программы определим следующий метод:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | private static void ReadFileFromDatabase()  {      string connectionString = @"Data Source=.\SQLEXPRESS;Initial Catalog=usersdb;Integrated Security=True";      List<Image> images = new List<Image>();      using (SqlConnection connection = new SqlConnection(connectionString))      {          connection.Open();          string sql = "SELECT \* FROM Images";          SqlCommand command = new SqlCommand(sql, connection);          SqlDataReader reader = command.ExecuteReader();            while(reader.Read())          {              int id = reader.GetInt32(0);              string filename = reader.GetString(1);              string title = reader.GetString(2);              byte[] data = (byte[])reader.GetValue(3);                Image image = new Image(id, filename, title, data);              images.Add(image);          }      }      // сохраним первый файл из списка      if(images.Count>0)      {          using (System.IO.FileStream fs = new System.IO.FileStream(images[0].FileName, FileMode.OpenOrCreate))          {              fs.Write(images[0].Data, 0, images[0].Data.Length);              Console.WriteLine("Изображение '{0}' сохранено", images[0].Title);          }      }  } |

В этом методе с помощью SqlDataReader мы получаем значения из БД и по ним создаем объект Image, который потом добавляется в список. И в конце смотрим, если в списке есть элементы, то берем первый элемент и сохраняем его на локальный компьютер. И после сохранения в папке нашей программы появится загруженный из базы данных файл.