O SqlFileStream é uma classe do .NET que permite trabalhar com dados armazenados no SQL Server usando o recurso de FILESTREAM, que possibilita o armazenamento de arquivos binários grandes (como documentos, imagens, vídeos) no sistema de arquivos, mantendo a integridade e gestão do SQL Server. Abaixo está um exemplo sobre como utilizar o SqlFileStream de maneira eficiente em C#.

**Passos para usar o SqlFileStream:**

1. **Configurar o SQL Server para Suportar FILESTREAM**:
   * Certifique-se de que o FILESTREAM está habilitado no seu SQL Server.
   * Crie uma tabela que suporte FILESTREAM:

sql

Copiar código

CREATE TABLE Documentos (

Id UNIQUEIDENTIFIER ROWGUIDCOL NOT NULL UNIQUE,

Nome NVARCHAR(255) NOT NULL,

Conteudo VARBINARY(MAX) FILESTREAM NULL,

PRIMARY KEY (Id)

);

1. **Códigos em C#:**
   * Exemplo para **inserir** um arquivo:

csharp

Copiar código

using System;

using System.Data;

using System.Data.SqlClient;

using System.Data.SqlTypes;

using System.IO;

using System.Transactions;

public class FileStreamExample

{

private string connectionString = "your\_connection\_string\_here";

public void SaveFileToDatabase(Guid fileId, string filePath)

{

using (var transactionScope = new TransactionScope())

{

using (var connection = new SqlConnection(connectionString))

{

connection.Open();

// Inserir um novo registro e obter o PathName

string insertQuery = @"

INSERT INTO Documentos (Id, Nome, Conteudo)

OUTPUT INSERTED.Conteudo.PathName()

VALUES (@Id, @Nome, DEFAULT)";

string filePathName;

using (var command = new SqlCommand(insertQuery, connection))

{

command.Parameters.AddWithValue("@Id", fileId);

command.Parameters.AddWithValue("@Nome", Path.GetFileName(filePath));

filePathName = (string)command.ExecuteScalar();

}

// Obter o contexto de transação para o SqlFileStream

string transactionContextQuery = "SELECT GET\_FILESTREAM\_TRANSACTION\_CONTEXT()";

byte[] transactionContext;

using (var command = new SqlCommand(transactionContextQuery, connection))

{

transactionContext = (byte[])command.ExecuteScalar();

}

// Escrever o arquivo no SQL FileStream

using (var sqlFileStream = new SqlFileStream(filePathName, transactionContext, FileAccess.Write))

using (var fileStream = new FileStream(filePath, FileMode.Open, FileAccess.Read))

{

fileStream.CopyTo(sqlFileStream);

}

transactionScope.Complete();

}

}

}

}

* + Exemplo para **ler** um arquivo:

csharp

Copiar código

public void ReadFileFromDatabase(Guid fileId, string destinationPath)

{

using (var transactionScope = new TransactionScope())

{

using (var connection = new SqlConnection(connectionString))

{

connection.Open();

// Obter o PathName do arquivo FILESTREAM

string selectQuery = @"

SELECT Conteudo.PathName()

FROM Documentos

WHERE Id = @Id";

string filePathName;

using (var command = new SqlCommand(selectQuery, connection))

{

command.Parameters.AddWithValue("@Id", fileId);

filePathName = (string)command.ExecuteScalar();

}

// Obter o contexto de transação para o SqlFileStream

string transactionContextQuery = "SELECT GET\_FILESTREAM\_TRANSACTION\_CONTEXT()";

byte[] transactionContext;

using (var command = new SqlCommand(transactionContextQuery, connection))

{

transactionContext = (byte[])command.ExecuteScalar();

}

// Ler o arquivo do SQL FileStream e salvar em disco

using (var sqlFileStream = new SqlFileStream(filePathName, transactionContext, FileAccess.Read))

using (var fileStream = new FileStream(destinationPath, FileMode.Create, FileAccess.Write))

{

sqlFileStream.CopyTo(fileStream);

}

transactionScope.Complete();

}

}

}

**Considerações Importantes:**

* **Transações:** O uso de TransactionScope é necessário porque o SqlFileStream precisa de uma transação de contexto do SQL Server.
* **Permissões:** Certifique-se de que as permissões do SQL Server estejam configuradas corretamente para permitir o acesso ao FILESTREAM.
* **Desempenho:** O FILESTREAM é útil para armazenar grandes arquivos binários, especialmente quando você precisa de integridade transacional e deseja armazenar dados diretamente no sistema de arquivos.

Usar o SqlFileStream dessa forma garante que você possa ler e gravar arquivos de forma eficiente, mantendo-os integrados ao banco de dados, mas armazenados fisicamente no sistema de arquivos.

4o

**Você disse:**

deu acess define, tem como corrigir isso?

**O ChatGPT disse:**

ChatGPT

O erro "Access Denied" ao usar SqlFileStream geralmente está relacionado a permissões insuficientes ou problemas de configuração no SQL Server. Aqui estão algumas etapas para resolver esse problema:

**1. Verifique as Configurações do SQL Server:**

Certifique-se de que o FILESTREAM está habilitado e configurado corretamente. Para isso, siga os passos abaixo:

1. Abra o SQL Server Configuration Manager.
2. Navegue até **SQL Server Services**.
3. Clique com o botão direito no seu serviço do SQL Server e selecione **Properties**.
4. Na aba **FILESTREAM**, certifique-se de que:
   * **Enable FILESTREAM for Transact-SQL access** está marcado.
   * **Enable FILESTREAM for file I/O streaming access** está marcado.
   * **Allow remote clients to have streaming access to FILESTREAM data** está marcado (se necessário).

**2. Conceda Permissões ao Diretório FILESTREAM:**

Certifique-se de que a conta sob a qual o SQL Server está sendo executado tenha permissões de leitura e gravação no diretório FILESTREAM. Para isso:

1. Encontre o caminho do diretório FILESTREAM. Você pode verificar isso no Management Studio, executando:

EXEC sp\_configure 'filestream access level'

1. Navegue até o diretório e clique com o botão direito nele.
2. Selecione **Propriedades** e, em seguida, vá para a aba **Segurança**.
3. Adicione a conta de serviço do SQL Server (ex.: NT SERVICE\MSSQLSERVER) e conceda **Controle Total**.

**3. Certifique-se de que a Conta do SQL Server tenha Permissões de FILESTREAM:**

No SQL Server Management Studio (SSMS), você pode verificar e habilitar permissões de FILESTREAM com o seguinte comando:

EXEC sp\_configure filestream\_access\_level, 2;

RECONFIGURE;

**Níveis de acesso possíveis:**

* 0: FILESTREAM desativado.
* 1: Ativado apenas para Transact-SQL.
* 2: Ativado para Transact-SQL e I/O de arquivos (necessário para SqlFileStream).

**4. Verifique o Código .NET:**

Certifique-se de que o código está abrindo uma conexão em um contexto de transação. O SqlFileStream exige que todas as operações de leitura e gravação sejam feitas dentro de uma transação explícita.

Exemplo de como criar e usar uma transação no código:

using (var transactionScope = new TransactionScope(TransactionScopeOption.Required,

new TransactionOptions { IsolationLevel = IsolationLevel.ReadCommitted }))

{

using (var connection = new SqlConnection(connectionString))

{

connection.Open();

// Insira seu código para operar com FILESTREAM aqui.

transactionScope.Complete(); // Confirma a transação

}

}

**5. Reinicie o Serviço do SQL Server:**

Após fazer alterações nas configurações do FILESTREAM, é recomendado reiniciar o serviço do SQL Server para garantir que todas as configurações sejam aplicadas corretamente.

**6. Confirme que o FILESTREAM está Ativo:**

Para verificar se o FILESTREAM está corretamente ativado e configurado, use o seguinte comando SQL:

EXEC sp\_configure 'filestream access level';

Isso deve retornar um valor de 2 se o FILESTREAM estiver habilitado para leitura e gravação de arquivos.

**Resumo**

Verifique as permissões do sistema operacional e configurações do SQL Server para garantir que tudo esteja habilitado corretamente e que as contas de serviço tenham acesso total. No código .NET, certifique-se de usar TransactionScope para garantir que a operação esteja dentro de um contexto transacional.