Przetwarzanie dokumentów XML za pomocą TSQLa

Adam Paleczny

Czerowiec 2023

1 Założenia projektowe

Celem projektu jest stworzenie API do obsługi plików XML w TSQLu. API ma umożliwiać funkcje zapisu i usuwania wybranego dokumentu oraz wyszukania wybranych dokumentów po określonych węzłach i/lub atrybutach oraz modyfikacji wybranych węzłów i/lub atrybutów. Całość powinna zostać zrealizowana w ramach Microsoft SQL Server Management Studio 2019, a część aplikacji napisana w C# w Microsoft Visual Studio 2019.

2 Procedury API

2.1 Wstęp

2.1.1 Pliki XML

Biblioteka obługuje pliki, o określonej strukturze:

```
<list>
  <table_name>
    <element> ... </element>
  </table_name>

  <table_name>
    <element> ... </element>
  </table_name>
</list>
```

Każdy plik xml znajduje się pomiędzu <list></list>, a *table_name* to nazwa tabeli w jakiej są przechowywane wartości. Znaczniki wewnątrz tabeli mogę być oczywiście zypełnie różne i niezależne od tych w innych tabelach.

2.1.2 Przechowywanie dokumentów XML

Każdy dokument XML przechowywany jest wewnątrz tabeli XmlContent:

Dzięki takiej strukturze znacznie ułatwione zostało odwoływanie się do odpowiednich dokumentów XML.

Nazwa	Typ	Opis
ID	INTEGER	Identyfikator pliku
Name	NVARCHAR	Nazwa tabeli wewnątrz pliku XML
XmlData	XML	Plik XML o opisanej strukturze

Tabela 1: Tabela XmLContent

2.2 Procedury biblioteki

2.2.1 InsertEndXml

Procedura służąca do dodanie nowego znacznika (@ElementName) z wartością (@Value) do wskazanej tabeli (@TableName) na końcu elementów pomiędzy znacznikami nazwy tabeli. Znacznik ten pojawia się pomiędzy **pierwszym** znacznikiem o nazwie tabeli w pliku XML.

Nazwa	Typ	Opis
TableName	NVARCHAR(MAX)	Nazwa tabeli
ElementName	NVARCHAR(MAX)	Nazwa nowego znacznika
Value	NVARCHAR(MAX)	Wartość wewnątrz znacznika

Tabela 2: InsertEndXml procedura

2.2.2 InsertIndexXml

Procedura, która działa podobnie jak InsertEndXml, natomiast dodaje ona wartości do indeksu wskazanego przez wartość @Value. Sprawdzany zostaje również czy dany element istnieje. Jeśli nie to nie zostaje on dodany.

Nazwa	Typ	Opis
TableName	NVARCHAR(MAX)	Nazwa tabeli
ElementName	NVARCHAR(MAX)	Nazwa nowego znacznika
Value	NVARCHAR(MAX)	Wartość wewnątrz znacznika
Number	INT	Numer znacznika o nazwie tabeli, w którym ma pojawić się nowy znacznik

Tabela 3: InsertIndexXml procedura

2.2.3 AddXMLToDB

Procedura służąca do dodanie pliku XML do bazy danych.

Nazwa	Typ	Opis
FileName	NVARCHAR(255)	Nazwa pliku
DirectoryPath	NVARCHAR(255)	Ścieżka bezwzględna do pliku

Tabela 4: AddXMLToDB procedura

2.2.4 DeleteValues

Procedura służąca do usuwania znaczników z pliku XML.

Nazwa	Typ	Opis
TABLE	NVARCHAR(MAX)	Nazwa tabeli
ATTRIBUTE	NVARCHAR(MAX)	Nazwa znacznika
VALUE	NVARCHAR(MAX)	Wartość pomiędzy znacznikami

Tabela 5: DeleteValues procedura

2.2.5 Delete_XML

Procedura służąca do usuwania pliku XML o podanej nazwie.

Nazwa	Typ	Opis
Name	VARCHAR(MAX)	Nazwa tabeli

Tabela 6: Delete_XML procedura

2.2.6 DeleteSubtree

Procedura usuwająca poddrzewo pliku xml o podanej nazwie. Należy również podać numer, które poddrzewo od początku usunąć. W przypadku jeśli poddrzewo o podanym numerze nie istnieje nić nie jest usuwane. Jeśli nie zostało już żadne poddrzewo o podanej nazwie usuwa cały plik xml.

Nazwa	Typ	Opis
TableName	NVARCHAR(MAX)	Nazwa tabeli
Numer	NVARCHAR(MAX)	Numer znacznika w pliku xml od początku

Tabela 7: DeleteSubtree procedura

2.2.7 FindInTableXML

Procedura służąca do znaleznia atrybutów w podanym pliku. Jeśli element jest znaleziony to zwracana jest jedynka, w przeciwnym wypadku 0.

Nazwa	Typ	Opis
TabeleName	NVARCHAR(MAX)	Nazwa tabeli
Attribute	NVARCHAR(MAX)	Nazwa znacznika
Value	NVARCHAR(MAX)	Wartość pomiędzy znacznikami

Tabela 8: FindInTableXML procedura

2.2.8 FetchXmlData

Procedura służąca do wyciągnięcia nazw znaczników z dokumentu XML o wskazanej nazwie. Wraca tabelę nazw znaczników

Nazwa	Typ	Opis
Element	NVARCHAR(MAX)	Nazwa tabeli i pliku XML

Tabela 9: FetchXmlData procedura

2.2.9 ReplaceXML

Procedura służąca do zmiany wartości wewnątrz podanych znaczników we wskazanej tabeli. Po wykonaniu procedury stara wartość zostanie zastąpiona nową.

Nazwa	Typ	Opis
TABLE	NVARCHAR(MAX)	Nazwa tabeli
ATTRIBUTE	NVARCHAR(MAX)	Nazwa znacznika
VALUE	NVARCHAR(MAX)	Wartość między znacznikami do podmienienia
NEWVALUE	NVARCHAR(MAX)	Nowa wartość

Tabela 10: ReplaceXML procedura

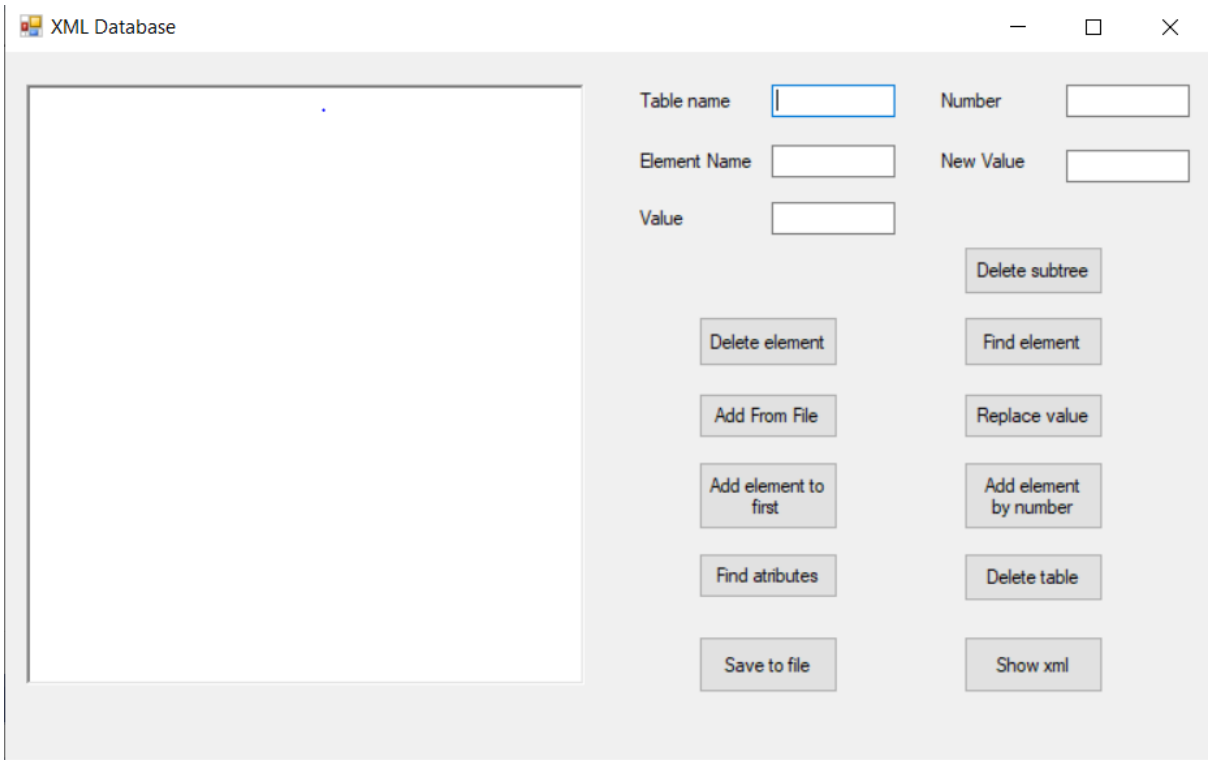
2.2.10 GetXmlTable

Procedura zwracająca plik XML o podanej nazwie tabeli.

Nazwa	Typ	Opis
TableName	NVARCHAR(MAX)	Nazwa tabeli

Tabela 11: GetXmlTable procedura

3 Przykład - Aplikacja okienkowa



Rysunek 1: Przykład działającej aplikacji

Na rysunku przedstawiony jest przykładowa aplikacja wykorzystująca bibliotekę do XMLa. Aplikacja wykorzystuje bibliotekę WxWidgets oraz język C#.

Aplikacja posiada panel wyświetlający dokument w formacie XML, 5 pól, w które można wpisać tekst do wykorzystywanych funkcji oraz 11 przycisków włączających poszczególne funkcje do obsługi bazy danych.

3.1 Obsługa

Każda metoda ma zaimplementowane sprawdzenie czy istnieje taka tabela. W przypadku występienia problemu komunikat zostanie wyświetlony.

3.1.1 Delete subtree

Przycisk ten działa jak metoda **DeleteSubtree**. W polu *Table name* wpisujemy nazwę tabeli, a w polu *Number* liczbę, która odpowiada poddrzewie, które chcemy usunąć. W przypadku zadziałania metody zostanie wyświetlony komunikat ile poddrzew zostało usuniętych.

3.1.2 Delete element

Naciśnięcie przycisku powoduje wywołanie metody **DeleteValues**. W polu *Table name* wpisujemy nazwę tabeli, w *Element Name* nazwę znacznika, a w *Value* wartość wewnątrz znacznika. Po wywołaniu metody usuwamy poddrzewo i wyświetla komunikat na ekranie.

3.1.3 Find Element

Naciśnięcie przycisku powoduje wywołanie metody **FindInTableXML**. W polu *Table name* wpisujemy nazwę tabeli, w *Element Name* nazwę znacznika, a w *Value* wartość wewnątrz znacznika. Jeśli element zostanie znaleziony to wyświetlamy cały dokument XML, a w przeciwnym przypadku otrzymujemy komunikat, że nie znajduje się w tym dokumencie znacznik o wskazanej wartości.

3.1.4 Add From File

Przycisk wykorzystuje procedurę **AddXMLToDB**. Należy jedynie wpisać nazwę tabeli. Należy oczywiście pamiętać, żeby zachować wszystkie zasady pliku xml, które są wskazane w dokumentacji. Plik XML musi się znajdować w tym samym folderze co plik wykonywalny. W przypadku dobrze wykonanej procedury otrzymamy informację.

3.1.5 Raplace Value

Po naciśnięciu przycisku wykonamy procedurę **ReplaceXML**. W polu *Table name* wpisujemy nazwę tabeli, w *Element Name* nazwę znacznika, a w *Value* wartość wewnątrz znacznika, którą chcemy podmienić, natomiast w *New Value* nową wartość. Po wykonanej zamianie otrzymujemy informację.

3.1.6 Add element to first

Naciśnięcie przycisku powoduje wywołanie metody **InsertEndXml**. W polu *Table name* wpisujemy nazwę tabeli, w *Element Name* nazwę znacznika, a w *Value* wartość wewnątrz znacznika. Element zostanie dodany na końcu pierwszego poddrzewa o wskazanej nazwie.

3.1.7 Add element by number

Naciśnięcie przycisku powoduje wywołanie metody **InsertIndexXml**. W polu *Table name* wpisujemy nazwę tabeli, w *Element Name* nazwę znacznika, a w *Value* wartość wewnątrz znacznika, w polu *Number* wpisujemy do którego poddrzewa dodajemy znacznik. Informacje o wyniku działania otrzymamy w panelu.

3.1.8 Find attributes

Przycisk wykorzystuje procedurę **FetchXmlData**. Należy jedynie wpisać nazwę tabeli w *Table Name*. W polu zostają wyświetlone znaczniki znajdujące się w elementach pliku xml o wskazanej nazwie.

3.1.9 Delete table

Wynikiem naciśnięcia przycisku jest usunie rekoru z bazy danych o wskazanej nazwie. Wykorzystywana jest do tego procedura **Delete_XML**. Wynik jest wyświetlany na ekranie.

3.1.10 Save to file

Po wciśnięciu przycisku zapisujemy obecny w bazie danym plik xml do pliku o nazwie tabeli. Wykorzystujemy procedurę **GetXmlTable**. Należy wpisać jedynie wartość w *Table name*. Po wykonaniu informujemy o tym wyniku w panelu.

3.1.11 Show xml

Wciśnięcie przycisku wywołuje procedurę **GetXmlTable**. Należy wpisać jedynie nazwę tabeli. Plik XML zostaje wyświetlony na panelu. W przeciwnym wypadku wyświetlany jest odpowiedni komunikat.

4 Wnioski i Podsumowania

Przedstawiona biblioteka implementuje operacje na danych XML i zapisuje je w typie natywnym. Opracowane API udostępnia funkcje zapisu i usuwania wybranego pliku, wyszukania wybranych dokumentów po określonych atrybutach oraz modyfikacji wybranych węzłów i atrybutów.

W przyszłej iteracji biblioteki możaby dodać możliwość obsługi wszystkich struktur XML, natomiast przedstawiony schamat bardzo dobrze mógłby sobie poradzić z zapisywaniem danych do tabeli SQL.

5 Bibliografia

- Laboratorium 11 – MS SQL Server 2019, Temat: Przegląd możliwości SQL Server 2019 związanych z wsparciem dla XML - dr. inż. Grażyna Krupińska, dr. inż. Antoni Dydejczyk
- Dokumentacja języka C# - learn.microsoft.com/pl-pl/dotnet/csharp - Microsoft, ostatnio wyświetlono: 09.06.2024
- XML w C#: learn.microsoft.com/pl-pl/sql/relational-databases/xml/xml-data-sql-server?view=sql-server-ver16, ostatnio wyświetlono: 09.06.2024