Andrzej PTASZNIK[1]

OBSŁUGA TYPU DANYCH XML W MS SQL SERVER 2008

Streszczenie

Przedmiotem wykładu jest wykorzystanie dokumentów XML w relacyjnych bazach danych. W pierwszej części wykładu zaprezentowana zostanie krótka historia standardu XML i podstawowe zasady tworzenia dokumentów XML. Wprowadzenie w MS SQL Server 2005 typu danych XML udostępniło nowe możliwości projektowania i implementacji baz danych. Metody typu XML pozwalają na realizację zapytań w języku SQL wykorzystujących jednocześnie dane relacyjne i dane zapisane w dokumentach XML. Możliwość definiowania schematów XSD umożliwia rozwiązanie problemów walidacji danych zapisanych w dokumentach XML na poziomie mechanizmów bazy danych.

Abstract

The subject of the lecture is the use of XML documents in relational databases. In the first part of the lecture a brief history of XML standard and basic rules for creating XML documents will be presented. The introduction of XML data type in MS SQL Server 2005 has made new possibilities accessible for database design and implementation. XML type methods allow the execution of SQL queries simultaneously using relational data and data saved as XML documents. The feature of XSD schemas defining enables solving problems of the validation of data saved as XML documents at the level of the database engine.

1. WPROWADZENIE

Relacyjne bazy danych towarzyszą twórcom aplikacji już od wielu lat i ciężko znaleźć projektanta lub programistę, który nie zetknął się z tą problematyką. Podobna sytuacja występuje w obszarze związanym z językiem XML. Jego intensywny rozwój, a właściwie dynamiczne rozszerzanie zastosowań oraz rozbudowywanie technologii wspomagających XML możemy obserwować od ponad dziesięciu lat. W takiej sytuacji jest dość oczywiste, że język XML pojawił się w otoczeniu relacyjnych baz

[1] Mgr inż. Andrzej Ptasznik jest wykładowcą w Warszawskiej Wyższej Szkole Informatyki.

danych i rozpoczęła się ich „współpraca". Przejawami tej współpracy i wzajemnego przenikania się obu technologii zajmiemy się w niniejszym wykładzie.

W jego ramach zostanie zaprezentowana geneza oraz podstawy języka XML, co ułatwi lepsze poruszanie się po omawianej problematyce. Zasadniczą częścią wykładu będzie jednak związek języka XML z SQL Server 2008 i wsparcie oferowane przez tę technologię w zakresie obsługi danych w formacie XML. Zaprezentowane zostaną polecenia służące do zwracania rezultatów zapytań w postaci dokumentów XML, przedstawione będą możliwości typu danych XML, który służy do przechowywania dokumentów XML w bazie w formie natywnej, a także zostaną opisane przykłady stosowania XML Schema do definiowania dopuszczalnej struktury dokumentów XML.

2. JĘZYK XML

Wraz z intensywnym rozwojem Internetu pojawił się (a właściwie nabrał większego znaczenia) problem przekazywania danych pomiędzy różnymi systemami. Problem ten istniał od momentu pierwszych prób łączenia komputerów w sieć. Od samego początku istniały dwa podejścia: korzystanie z binarnego zapisu danych oraz z postaci tekstowej. W okresie, gdy komputery miały bardzo ograniczone zasoby (pamięć operacyjną, pojemność dysków, transfer w sieci) popularniejsze były rozwiązania korzystające z postaci binarnej. Istniejące protokoły binarne powodowały jednak konieczność „tłumaczenia" danych pomiędzy heterogenicznymi systemami (np. w jednym systemie liczby całkowite są cztero– a w drugim dwu-bajtowe), co powodowało wzrost poziomu komplikacji systemów rozproszonych. Równocześnie obserwowano burzliwy rozwój usługi WWW, dla której pokonanie bariery pomiędzy różnymi architekturami nie stanowiło problemu, gdyż stosowanym rozwiązaniem było przekazywanie danych w postaci tekstowej, czego przejawem stał się język HTML. Jego sukces i rosnąca popularność ujawniły jednak wiele niedoskonałości. Wystarczy wspomnieć o tzw. wojnie przeglądarek, polegającej z grubsza na różnym sposobie interpretowania (czyli renderowania) dokumentu HTML przez różne przeglądarki oraz na dodawaniu przez producentów przeglądarek nowych elementów (spoza specyfikacji HTML), które były interpretowane przez ich produkt. W efekcie idea HTML – jako języka służącego do przekazywania treści wraz z informacjami nadającymi znaczenie poszczególnym fragmentom tekstu – została wypaczona, a sam język został sprowadzony do poziomu języka opisu układu (ang. *layout*) dokumentu.

W 1996 roku powstał pomysł stworzenia nowego języka dla potrzeb Internetu. Miał to być prosty język, z prostymi regułami dotyczącymi jego składni, czytelny

zarówno dla człowieka jak i dla maszyn, oraz możliwie uniwersalny jako nośnik danych i informacji o ich strukturze. W efekcie tych prac powstał **język XML** (ang. *Extensible Markup Language*). Wizualnie zbliżony do HTML (stosowanie znaczników i elementów atrybutów), choć od samego początku rygorystycznie podchodzący do poprawnego formułowania dokumentu. Mimo, że XML jest tylko podzbiorem istniejącego od dawna, **języka SGML** (ang. *Standarized General Markup Language*) o potężnych możliwościach, szybko okazało się, że z racji swojej prostoty, XML wyparł SGML z większości zastosowań.

Reguły określone dla struktury dokumentu XML są proste i przejrzyste, oraz nie zawierają wyjątków. Aby dokument XML mógł być uznany za poprawnie sformułowany (tzn. zgodny z wszystkimi wymaganiami co do syntaktyki) musi spełniać następujące reguły:

- składać się z elementów, przy czym jeden z elementów jest elementem głównym i zawiera w sobie pozostałe;
- element składa się (podobnie jak w HTML) ze znaczników: otwierającego i zamykającego;
- elementy muszą być poprawnie zagnieżdżone – zakres objęty elementem musi być jednoznacznie określony. Innymi słowy: jeden element może zawierać w sobie inny, ale tylko w całości (wraz ze znacznikiem zamykającym);
- każdy element może zawierać atrybuty. Są one definiowane wewnątrz znacznika otwierającego po nazwie elementu. Każdy atrybut musi mieć wartość – w przeciwieństwie do HTML, gdzie takiej konieczności nie było. Wartość atrybutu musi być ujęta w apostrofy lub cudzysłów;
- w dokumentach XML jest rozróżniana wielkość liter i jest ona istotna. Elementy **<dane /> <Dane />** i **<DANE />** to trzy różne elementy.

Poprawność dokumentów XML można rozpatrywać na dwóch poziomach: syntaktycznym i semantycznym. Pierwszy z nich jest określony przez opisane wyżej reguły dotyczące tworzenia dokumentów XML. Jeżeli dokument pod względem składniowym spełnia te reguły, to jest **dokumentem poprawnie sformułowanym** (ang. *well formed*).

Jeżeli ten poziom weryfikacji nie wystarcza, można sięgnąć po dodatkowe narzędzia, aby nałożyć ograniczenia semantyczne (np. w każdym elemencie „osoba" musi wystąpić co najmniej jeden element „imie", wartością atrybutu pesel musi być ciąg 11 cyfr). Najczęściej stosowane są rozwiązania: **DTD** (ang. *Document Type Definition*) i **XML Schema**.

DTD jest rozwiązaniem starszym i o ograniczonych możliwościach. Obecnie znacznie częściej stosuje się **XML Schema**, co nie znaczy że zapewnia ono

możliwość zdefiniowania dowolnych reguł dotyczących dopuszczalnej zawartości dokumentu XML. Jeżeli dokument jest poprawnie sformułowany oraz spełnia reguły opisane w DTD lub XML Schema, to jest **dokumentem poprawnym** (ang. *valid*). Istnieje wiele narzędzi służących do sprawdzania poprawności dokumentów XML, są to tzw. **parsery walidujące**. Połączenie XML z XML Schema, przy wsparciu ze strony narzędzi, umożliwia bardzo łatwe rozwiązywanie problemów z projektowaniem formatów przekazywania czy przechowywania danych. Wystarczy opracować dla konkretnego problemu dokument XML Schema i przekazać zainteresowanym stronom. Twórcy dokumentów XML będą mieli wszystkie informacje, niezbędne do utworzenia dokumentów w postaci zgodnej z założeniami. Z kolei, twórcy oprogramowania, które ma takie dane interpretować, będą wiedzieli czego mogą spodziewać się w dokumencie oraz będą mieli możliwość łatwego sprawdzenia poprawności dokumentu przed rozpoczęciem jego przetwarzania. Taki schemat sprzyja rozkwitowi technologii związanych ze stosowaniem XML. Można tu wymienić dla przykładu **język SVG** (ang. *Scalable Vector Graphics)* czy **MathML**. Oba definiują postać dokumentu XML, który następnie jest przetwarzany na grafikę wektorową lub równanie matematyczne.

3. JĘZYK XML A RELACYJNE BAZY DANYCH

Skoro język XML umożliwia łatwe budowanie dokumentów o hierarchicznej strukturze, to czy może stanowić alternatywę dla relacyjnych baz danych? Można przecież zastąpić relacje odpowiednim zagnieżdżaniem elementów. Teoretycznie jest to jak najbardziej możliwe, ale praktycznie raczej nie.

W dokumencie XML, co prawda, można zastąpić relacje odpowiednim zagnieżdżaniem elementów oraz stosowaniem atrybutów, ale to podejście sprawdza się wyłącznie przy małej ilości danych. Przy niewielkim rozmiarze dokumentu może on śmiało rywalizować z bazą danych, lecz wraz ze wzrostem rozmiaru dokumentu, XML szybko zostaje w tyle i osiąga „masę krytyczną" co powoduje ogromny spadek wydajności przy jego przetwarzaniu. Podobnie wygląda definiowanie ograniczeń stosowanych do zawartości i struktury dokumentu XML. Narzędzie XML Schema, mimo rozbudowanych możliwości, okazuje się jednak niewystarczające i powstaje konieczność dobudowywania własnych mechanizmów służących zapewnieniu spójności danych. To wszystko powoduje, że stosowanie języka XML w charakterze bazy danych staje się coraz bardziej złożone, co szybko przerasta stopień złożoności prawdziwej bazy relacyjnej zastosowanej do rozwiązania tego samego problemu, a co za tym idzie stosowanie XML w tym przypadku staje się ekonomicznie nieuzasadnione.

Obszarem, w którym XML sprawdza się jednak bardzo dobrze w roli bazy danych jest pełnienie funkcji magazynu danych *off-line*. Polega to na tworzeniu aplikacji, które łącząc się z bazą danych, pobierają dane niezbędne do pracy aplikacji i przechowują je na komputerze użytkownika w postaci XML. Dalsza praca odbywa się na danych z XML już bez połączenia z bazą danych i dopiero w momencie synchronizacji dane są uaktualniane w bazie. Przykładem tego typu rozwiązania jest klasa DataSet z .NET Framework. Ma ona bardzo rozbudowane możliwości przechowywania danych i śledzenia ich modyfikacji.

Zasadniczo jednak język XML należy raczej rozpatrywać jako pewnego rodzaju uzupełnienie funkcjonalności baz danych, które może być stosowane w przypadkach, wymagających tworzenia rozbudowanej struktury tabel, aby zamodelować zbiór cech informacyjnych o zróżnicowanej strukturze. Tu XML sprawdza się znakomicie, co potwierdza coraz większe wsparcie dla korzystania z XML w relacyjnych bazach danych oferowane przez poszczególnych producentów. W ramach niniejszego wykładu prezentowane będą takie właśnie mechanizmy w odniesieniu do systemu SQL Server 2008. Pierwszym elementem, o który rozszerzono możliwości polecenia SELECT języka SQL jest klauzula FOR XML umożliwiająca zwracanie wyników zapytań w relacyjnych bazach danych w postaci odpowiednio sformatowanych dokumentów XML. Klauzulę FOR XML wprowadzono już w MS SQL Server 2000, co biorąc pod uwagę datę publikacji pierwszego standardu XML, czyli rok 1998, wyraźnie wskazuje wagę jaką od początku istnienia języka XML przywiązywano. W tej samej wersji MS SQL Server wprowadzono funkcję OPENXML, która wraz z systemowymi procedurami składowanymi ***sp_XML_preparedocument*** oraz ***sp_XML_removedocument*** stwarzała możliwość realizacji zapytań pobierających dane z dokumentów XML. Największym problemem było przechowywanie dokumentów XML w tabelach relacyjnej bazy danych, ponieważ jedynym typem danych możliwym do wykorzystania był typ ***ntext,*** ale zapisane w ten sposób dane były uciążliwe przy programowej ich obróbce.

4. TYP DANYCH XML W MICROSOFT SQL SERVER 2008

Konsekwencją wzrostu popularności języka XML było wprowadzenie do standardu SQL typu danych XML. Wprowadzenie tego typu całkowicie zmieniło sposób obsługi danych zapisanych jako dokumenty XML w środowisko relacyjnych baz danych. Służy on do przechowywania dokumentów lub fragmentów dokumentów XML bezpośrednio w bazie danych oraz do wygodnego manipulowania nimi i walidowania z zastosowaniem XML Schema. Dane XML w bazie mogą występować w dwóch wariantach:

- skojarzone z kolekcją dokumentów XML Schema (*typed XML*),
- nieskojarzone z XML Schema (*untyped XML*).

Skojarzenie kolumny typu XML ze schematem XSD powoduje nadanie ograniczeń strukturze dokumentów XML, które mogą być umieszczone w tej kolumnie. Ograniczenia te są weryfikowane automatycznie przy każdej operacji dodania czy modyfikacji zawartości kolumny XML.

Zanim skorzysta się z typu danych XML warto zapoznać się z jego dokumentacją. Szczególnie chodzi tu o sposób przechowywania dokumentu oraz o ograniczenia związane z samym typem danych. Warto również zastanowić się nad korzystaniem z kolekcji XML Schema oraz indeksów XML. Istotną informacja jest to, że dokument nie jest zapisywany w bazie wprost, tylko przechodzi proces normalizacji (modyfikacja dokumentu, kodowanie w Unicode, eliminowanie niepotrzebnych ciągów i znaków itp.). Eliminuje to możliwość korzystania z tego typu danych wszędzie tam, gdzie ważna jest oryginalna postać dokumentu (np. kwestie podpisu elektronicznego).

Deklarowanie kolumn typu XML nie odbiega od deklarowania kolumn każdego innego typu. Jedyną specyficzną rzeczą jest – w przypadku kolumny ze skojarzoną kolekcja dokumentów XML Schema – umieszczenie w nawiasie w deklaracji typu nazwy tej kolekcji.

Sama kolekcja dokumentów XML Schema zawierać może jedną bądź wiele pojedynczych schematów XSD. Po skojarzeniu kolekcji z kolumną typu XML, każda wartość wpisywana do tej kolumny będzie walidowana pod kątem zgodności, z którymś ze schematów XSD z kolekcji. W przypadku braku zgodności – operacja zapisu zostanie anulowana.

Tworzenie dokumentów XML Schema (zwane też modelowaniem dopuszczalnej struktury dokumentów XML) jest zagadnieniem bardzo rozbudowanym i wykracza poza ramy niniejszego wykładu. Przy założeniu, że mamy już określoną postać dokumentu XML Schema, stworzenie kolekcji schematów XSD jest proste i ogranicza się do wykonania jednego polecenia – CREATE XML SCHEMA COLLECTIONS. Od tego momentu można używać zdefiniowanej w ten sposób kolekcji przy deklaracjach kolumn typu XML.

Kiedy w praktyce należy stosować typ danych XML? W tej kwestii zdania są podzielone. Jedni z definicji odrzucają XML traktując go jako niepotrzebny, a wręcz szkodliwy element odbiegający od relacyjnego modelu danych (stawiając m.in. zarzuty co do kiepskiej wydajności), drudzy używają go gdzie tylko się da, zastępując jedną kolumną XML strukturę kilku tabel lub tworząc procedury składowane, którym przekazuje się tylko jeden parametr typu XML, z którego są potem pobierane

konkretne wartości. W skrajnych przypadkach cała komunikacja z bazą danych sprowadza się do wymiany dokumentów XML. Z aplikacji przychodzi żądanie z parametrami w postaci XML, na które baza odpowiada zwracając dokument XML z odpowiednią strukturą danych. Sprowadza to komunikację z bazą danych do postaci zbliżonej do korzystania z usług sieciowych (*webservices*), które stają się w ostatnich latach coraz bardziej popularne. Nowością, którą typ danych XML wprowadza do technologii relacyjnych baz danych są metody typu danych.

Aby korzystać z metod typu XML należy opanować dodatkowo podstawy języka XQuery oraz XPath, gdyż wyrażenia zbudowane w oparciu o nie są stosowane w parametrach wywołania metod typu XML.

Krótki opis metod typu XML zawiera poniższe zestawienie:

- Metoda ***value(xquery,typ)*** służy do wskazania poprzez wyrażenie XPath elementu lub atrybutu, którego wartość będzie pobrana z dokumentu XML, a następnie skonwertowana do typu wskazanego w drugim parametrze wywołania metody ***value()***.
- Metoda ***exist(xquery)*** służy do sprawdzenia, czy kolumna XML zawiera w swojej wartości element lub atrybut wskazany przez wyrażenie XQuery. Podobny efekt da się osiągnąć za pomocą metody **value().** Jeżeli jednak nie ma konieczności pobierania wartości z XML, a tylko sprawdzenia jej istnienia, to metoda ***exist()*** jest zalecana ze względu na szybsze działanie.
- Metoda ***query(xquery)*** służy do wskazania przez wyrażenie XQuery zbioru węzłów z dokumentu XML, które są następnie zwracane także jako zmienna typu XML.
- Metoda ***nodes(xquery)*** służy do przetworzenia danych zawartych w dokumencie XML na postać relacyjną. Z jej pomocą (oraz z wykorzystaniem operatora CROSS APPLY) można wybrać węzły dokumentu, które będą tworzyły kolumny wiersza danych w zbiorze wynikowym zapytania SELECT. Rezultatem działania metody ***nodes()*** jest zbiór wierszy zawierający logiczne kopie węzłów dokumentu XML wybranych przez wyrażenie XQuery. Funkcja ta jest szczególnie użyteczna i wygodna, gdy tworzymy zapytanie, które ma zawierać w kolumnach poszczególne informacje zaszyte w strukturze elementów i atrybutów dokumentu XML
- Metoda ***modify(XMLdml)*** służy do modyfikowania zawartości dokumentu XML. Modyfikacje te są realizowanie za pomocą poleceń języka XML DML (ang. *XML Data Manipulation Language*). W skład XML DML wchodzą trzy polecenia:
  - ***insert*** – służące do dodawania nowych węzłów (elementów, atrybutów, węzłów tekstowych itp.) do dokumentu XML

- ***delete*** – służące do usuwania węzłów z dokumentu
- ***replace value of*** – służące do zastępowania zawartości węzła dokumentu inna zawartością

Możliwości tych trzech poleceń są dość ograniczone i łatwe do zastosowania wyłącznie w przypadku prostych modyfikacji operujących z reguły na wartościach podawanych w postaci stałych łańcuchów znaków. Gdy potrzebne są możliwości dynamicznego budowania wartości, która ma być wstawiona do dokumentu, to szybko okazuje się, że jest to trudne bądź wręcz niemożliwe. Dlatego, gdy wymagane są bardziej złożone operacje na dokumencie XML, to są realizowane one po stronie aplikacji, a ich gotowy wynik jest przekazywany do bazy.

Możliwości manipulowania danymi zapisanymi w kolumnach lub zmiennych typu XML są istotnym elementem składowym funkcjonalności obsługi XML w bazie danych. Umożliwiają realizowanie typowych operacji na danych XML w sposób zbliżony do znanego ze świata XML. Jest to bardzo istotne ze względu na łatwość zastosowania tych rozwiązań w praktyce.

5. PODSUMOWANIE

W ramach niniejszego wykładu zaprezentowany został pokrótce język XML oraz rozważyliśmy możliwości korzystania z niego w relacyjnych bazach danych. Z racji rosnącej popularności tego języka oraz coraz bogatszego wsparcia ze strony producentów serwerów baz danych, warto się nim zainteresować, aby móc rozważać go jako jedną z opcji przy projektowaniu baz danych. Jako przykład serwera baz danych, oferującego wsparcie dla XML, wykorzystany został SQL Server 2008. W ramach omawiania jego możliwości w kontekście XML, zasygnalizowane zostały główne obszary, w których serwer oferuje narzędzia służące do obsługi dokumentów XML. Przyjrzeliśmy się klauzuli FOR XML stosowanej do zwracania wyników zapytania w postaci dokumentów lub fragmentów dokumentów XML. Zapoznaliśmy się, także z typem danych XML, który służy nie tylko do przechowywania danych w tym formacie, ale także do zaawansowanego odpytywania dokumentów XML oraz manipulowania ich zawartością. Dodatkowo, wspomnieliśmy o wykorzystywanych w ramach SQL Server 2008 innych technologiach i narzędziach wspierających XML. Warto tu wymienić choćby XQuery, XPath oraz XML Schema. Wszystkie te rozwiązania są otwarte, nie stanowią własności żadnej firmy i są rozwijane przez konsorcjum W3C i szeroko adoptowane w świecie IT. Umożliwia to ekspertom od XML na łatwe wykorzystanie nabytej wiedzy również przy korzystaniu z serwerów baz danych.

Jest jednak także druga strona medalu. Jeśli zachłyśniemy się wsparciem dla XML w SQL Server 2008, to bardzo szybko może nas czekać rozczarowanie. Otóż mechanizmy wspierające XML na pierwszy rzut oka wyglądają na potężne i wygodne. Faktycznie jest tak, ale tylko w zakresie przewidzianym przez ich twórców. Okazuje się, że nie zaimplementowali oni w pełni standardów XQuery i XPath (można szybko natrafić na wiele nieobsługiwanych funkcji bądź ograniczeń). Podobnie jest w przypadku XML Schema. Nie wszystkie możliwości tej technologii są dopuszczalne w ramach SQL Server 2008. W zakresie manipulowania strukturą dokumentów XML również okazuje się, że metoda ***modify()*** z poleceniami XML DML ma bardzo dużo ograniczeń, szczególnie gdy chce się polecenia tworzyć bardziej dynamicznie. To wszystko nie umniejsza jednak faktu, ze wsparcie XML w relacyjnych bazach danych jest ciekawą możliwością, z której warto korzystać, po uprzednim gruntownym zapoznaniu się z dokumentacją, aby uniknąć przykrych niespodzianek w trakcie realizacji projektu.

Literatura

1. I. Ben-Gan, L. Kollar, D. Sarka: *MS SQL Server 2005 od środka: Zapytania w języku T-SQL,* APN PROMISE, Warszawa 2006
2. R. Coburn: *SQL dla każdego*, Helion, Gliwice 2001
3. T. Rizzo, A. Machanic, R. Dewson, R. Walters, J. Sack, J. Skinner: *SQL Server 2005*, WNT, Warszawa 2008
4. M. Szeliga: ABC języka SQL, Helion, Gliwice 2002
5. R. Vieira: *SQL Server 2005. Programowanie. Od Podstaw,* Helion, Gliwice 2007
6. E. Castro: Po prostu XML, Helion, Gliwice 2001
7. P. Walmsley: Wszystko o XML Schema, Wydawnictwo Naukowo-Techniczne, Warszawa 2007