informatyka+

Algorytmika i programowanie

Bazy danych

Multimedia, grafika i technologie internetowe

Sieci komputerowe

Tendencje w rozwoju informatyki i jej zastosowań

Człowiek – najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Warszawska Wyższa Szkoła Informatyki

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

informatyka+

Kuźnia Talentów: Bazy danych

Wykorzystanie XML w relacyjnych bazach danych

Andrzej Ptasznik

Człowiek – najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Warszawska Wyższa Szkoła Informatyki

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Wykorzystanie XML w relacyjnych bazach danych

Rodzaj zajęć: Kuźnia Talentów Informatycznych
Tytuł: Wykorzystanie XML w relacyjnych bazach danych
Autor: mgr inż. Andrzej Ptasznik

Redaktor merytoryczny: prof. dr hab. Maciej M Sysło

Zeszyt dydaktyczny opracowany w ramach projektu edukacyjnego **Informatyka+** — ponadregionalny program rozwijania kompetencji uczniów szkół ponadgimnazjalnych w zakresie technologii informacyjno-komunikacyjnych (ICT).
www.informatykaplus.edu.pl
kontakt@informatykaplus.edu.pl

Wydawca: Warszawska Wyższa Szkoła Informatyki
ul. Lewartowskiego 17, 00-169 Warszawa
www.wwsi.edu.pl
rektorat@wwsi.edu.pl

Projekt graficzny: FRYCZ I WICHA

Warszawa 2010

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Warszawska Wyższa Szkoła Informatyki

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Wykorzystanie XML w relacyjnych bazach danych

Andrzej Ptasznik
Warszawska Wyższa Szkoła Informatyki
aptaszni@wwsi.edu.pl

Streszczenie

Kurs jest poświecony szeroko rozumianym aspektom wykorzystania dokumentów XML w bazach danych. Słuchacze będą realizowali ćwiczenia polegające na wykorzystaniu danych relacyjnych łącznie z danymi zapisanymi w dokumentach XML. Przedstawione zostaną aspekty praktyczne i korzyści wynikające z zastosowania typu danych XML w definicji tabel. Wprowadzone zostaną także pojęcia związane z walidacja dokumentów XML (schematy XSD).

Spis treści

1 WPROWADZENIE

Historia relacyjnego modelu danych rozpoczęła się w roku 1970 wraz z publikacją E.F. Codda pt. *A Relational Model of Data for Large Shared Data Banks*. (pol. *Relacyjny model danych dla dużych współdzielonych banków danych*). Ten artykuł wzbudził duże zainteresowanie, ponieważ przedstawiał możliwości realizacji i praktyczne zastosowania komercyjnego systemu baz danych. Praca ta stworzyła teoretyczne podstawy budowania baz danych w oparciu o relacyjne podejście do jej modelowania. Opublikowana teoria bardzo szybko zainteresowała potencjalnych twórców i producentów systemów zarządzania bazami danych. Droga od teorii do praktyki bywa często długa i wyboista ale w tym przypadku, ze względu na pilne potrzeby rynku, przebiegała dość szybko i sprawnie.

W ramach naszego kursu zapoznamy się z Systemem Zarządzania Relacyjnymi Bazami Danych MS SQL Server 2008 i skoncentrujemy się na nowej tendencji w relacyjnych bazach danych, czyli wykorzystaniu **dokumentów XML** (ang. *Extensible Markup Language*, w wolnym tłumaczeniu *Rozszerzalny Język Znaczników*). 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 „z otoczki" XML, można obserwować od ponad dziesięciu lat. W takiej sytuacji jest dość oczywiste, że język XML pojawił się w otoczeniu relacyjnych baz danych i rozpoczęła się ich „współpraca". Przejawami tej współpracy i wzajemnego przenikania się obu technologii zajmiemy się w niniejszym kursie.

Podczas zajęć zostanie zaprezentowana geneza oraz podstawy języka XML, co ułatwi poruszanie się po omawianej problematyce. Zasadniczą częścią zajęć zajmie omówienie związku języka XML z SQL Server 2008 i wsparcie oferowane przez to narzędzie 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.

Podstawowe pojęcia związane z bazami danych

Wprowadzimy na początku kilka definicji wyjaśniających podstawowe pojęcia, którymi będziemy się posługiwali w dalszej części.

- **Dane** to liczby, znaki, symbole (i cokolwiek innego) zapisane w celu ich przetwarzania.
- **Informacja** to taki czynnik, któremu człowiek może przypisać określony sens (znaczenie), aby móc ją wykorzystywać do różnych celów.
- **Wiedza** to, zgodnie z prastarą definicją Platona, *ogół wiarygodnych informacji o rzeczywistości wraz z umiejętnością ich wykorzystywania*.

Można teraz uporządkować te pojęcia w kontekście baz danych i sposobu i wykorzystania. Dane zbieramy i zapisujemy, by na ich podstawie uzyskiwać informacje, które będą stawały się wiedzą, gdy uzupełnimy je o sposoby i możliwości ich praktycznego wykorzystania. Nie trzeba chyba udowadniać tezy, że współczesne społeczeństwo, społeczeństwo informacyjne, opiera swoje działania i podstawy rozwoju na wiedzy, która jest między innymi pozyskiwana z baz danych. Zgodnie z powyższymi definicjami dane to prawie wszystko co może być zapisane i dlatego jednym z podstawowych zadań, żeby zbiór danych mógł stać się bazą danych, jest zapewnienie odpowiedniego sposobu uporządkowania. Bazy danych mogą gromadzić gigantyczne ilości danych zapewniając właściwy sposób ich uporządkowania. Teraz możemy spróbować zdefiniować pojęcie bazy danych :

Baza danych to zbiór danych zapisanych w ściśle określony sposób w strukturach odpowiadających założonemu modelowi danych.

2 XML – OMÓWIENIE STANDARDU

Język XML wywodzi się z opracowanego na początku lat 80' XX wieku **języka SGML** (ang. *Standard Generalized Markup Language*). Standardowy uogólniony język znaczników (SGML) służy do ujednolicania struktury i formatu różnego typu informacji (danych). Umożliwia zapisanie ich w postaci dokumentu tekstowego i dzięki temu można łatwo je przenosić, wyświetlać i drukować w różnych systemach elektronicznego przekazu danych. SGML w odróżnieniu od języków znaczników dedykowanych do konkretnych zastosowań (takich jak np.

HTML), nie jest zbiorem określonych znaczników i reguł ich użytkowania, lecz ogólnym, nadrzędnym językiem służącym do definiowania dowolnych znaczników i ustalania zasad ich poprawnego użytkowania. Poniżej przedstawiona została krótka historia rozwoju języków opartych na znacznikach:

- lata 60' XX wieku: poszukiwanie standardu dla dokumentów drukarskich.
- 1969: firma IBM zaproponowała GML wraz z hierarchiczna struktura oznaczeń.
- 1978: ANSI rozpoczyna prace nad normą.
- 1983: szósta wersja – SGML – staje się standardem Urzędu Kontroli Skarbowej USA.
- 1985: SGML standardem Komisji UE oraz Departamentu Obrony USA.
- 1986: SGML w wersji ISO 9979:1986.
- Dziś mamy nowe standardy ISO oraz HTML, XML, XSL i inne – wszystko na podstawie założeń GML/SGML
- 1986: SGML – Standard Generalized Markup Language, ISO 8879:1986.
- 1991: powstaje World Wide Web.
- 1994: HTML 2.0 zdefiniowany jako zastosowanie języka SGML.
- 1998: XML – Extensible Markup Language, World Wide Web Consortium.

Warto tutaj zaznaczyć, że ze względu na dużą złożoność, standard SGML nie doczekał się powszechnego zastosowania i nie powstały narzędzia i technologie, które by w pełni implementowały ten standard. Twórcy języka XML za jedno z podstawowych założeń przyjęli możliwie największa prostotę języka, żeby nie stwarzać barier przed jego rozwoju.

XML (ang. *Extensible Markup Language*, w wolnym tłumaczeniu *Rozszerzalny Język Znaczników*) – uniwersalny język formalny przeznaczony do reprezentowania różnych danych w strukturalizowany sposób. Język XML jest niezależny od platformy, co umożliwia łatwą wymianę dokumentów pomiędzy heterogenicznymi (różnymi) systemami i znacząco przyczyniło się do popularności tego języka w dobie Internetu. XML jest standardem rekomendowanym oraz specyfikowanym przez organizację W3C. Koncepcja języka XML polega na znacznym uproszczeniu zawiłego języka SGML, co pozwoliło na budowę prostszych parserów (procesorów XML). XML jest podzbiorem standardu SGML eliminując jego zbyt złożony charakter tam, gdzie się tylko dało. Większość dokumentów XML jest zgodna z SGML, ale nie na odwrót. Istnieją dokumenty XML, które nie są poprawnymi dokumentami SGML.

3 TECHNOLOGIA MS SQL SERVER 2008 I SQL SERVER MENAGEMENT STUDIO

3.1 WPROWADZENIE DO TECHNOLOGII MS SQL SERVER 2008

Dotychczasowe rozważania prowadziliśmy w oderwaniu od technologii, czyli od sposu realizacji. Koncentrowaliśmy się na teorii, a teraz przyszła pora na praktykę. W tym celu zdefiniujemy nowe pojęcie:

Systemem Zarządzania Bazami Danych(SZBD) nazywamy specjalistyczne oprogramowanie umożliwiające tworzenie baz danych oraz ich eksploatację. Wydaje się oczywiste, że tworzenie i działanie baz danych musi być wspierane przez specjalistyczne oprogramowanie, które powinno umożliwiać realizację pewnych zadań:

- definiowanie obiektów bazy danych,
- manipulowanie danymi,
- generowanie zapytań,
- zapewnienie spójności i integralności danych.

Zadania te brzmią bardzo ogólnie, obejmują jednak większość potrzeb w zakresie tworzenia i eksploatacji baz danych. Dla przybliżenia pojęcia SZBD można podać kilka nazw handlowych, pod jakimi te produkty można spotkać na rynku i w zastosowaniach: MS SQL Server 2008, Oracle, MySQL, Access, DB2 i wiele innych mniej lub bardziej popularnych.

Jednym z najważniejszych zadań stojących przed SZBD jest zapewnienie spójności i integralności danych, czyli dostarczenie mechanizmów zapewniających przestrzeganie określonych reguł przez dane. SZBD dostarczają mechanizmów służących do zapewnienia spójności i integralności danych, czyli mówiąc innymi słowami, zapewnienia logicznej poprawności danych zapisanych w bazie. Podstawowe mechanizmy, realizujące te zadania to :

- deklaracja typu,
- definicje kluczy,

- reguły poprawności dla kolumny,
- reguły poprawności dla wiersza,
- reguły integralności referencyjnej

W ramach kursu będziemy wykorzystywać technologię MS SQL Server 2008 Express. Jest to jeden z najpopularniejszych serwerów baz danych. Edycja SQL Server Express, z której będziemy korzystać, jest wersją darmową z możliwością wykorzystania jej w celach komercyjnych. Technologia SQL Server 2008 zawiera następujące podsystemy:

- Serwer bazy danych (*Database Engine*) – podsystem odpowiedzialny za zarządzanie bazami danych (definiowanie, eksploatacja i administracja baz danych).
- Serwer raportowania (*Reporting Services*) – podsystem umożliwiający zarządzanie procesem tworzenia i dystrybucji raportów generowanych na podstawie danych z różnych źródeł (bazy danych, pliki Excel, pliki tekstowe, dokumenty XML).
- Serwer usług analitycznych (*Analysis Services*) – podsystem wspomagający organizację hurtowni danych, wielowymiarowych kostek analitycznych, tworzenie pulpitów menadżerskich oraz realizację algorytmów wyszukiwania złożonych zależności (*Data Mining*).
- Serwer usług integracyjnych (*Integration Services*) – podsystem realizujący zadania integracji danych, polegające, w dużym uproszczeniu, na pobieraniu danych z pewnych źródeł danych, poddanie ich procesowi przetwarzania (sprawdzanie poprawności, eliminowanie błędów itp.) a następnie zapisanie przetworzonych danych w docelowej lokalizacji. Zadania te są określane w teorii jako platforma ET&L (*Extract, Transform and Load*).

W ramach kursu będziemy wykorzystywać jedynie serwer baz danych (*Database Engine*), który zawiera wiele dodatkowych technologii :

- Usługi asynchronicznego przetwarzania (*Service Broker*) – uożliwiają realizację asynchronicznego przetwarzania z wykorzystanie kolejek.
- Usługi replikacji danych – umożliwiają konfigurowanie zadań związanych z odtwarzaniem części zasobów bazy danych w innych lokalizacjach.
- Usługi wyszukiwania pełno tekstowego – umożliwiają wyszukiwanie fragmentów tekstu niezależnie od ich lokalizacji w bazie danych (klasyczne zapytanie SQL wymagają określenie tabel z których dane są pobierane).

Wymienione zostały niektóre elementy technologii MS SQL Server 2008 co i tak pokazuje, że jest to bardzo rozległy i złożony system umożliwiający realizacje różnych zadań związanych z bazami danych. Nasz kurs należy traktować jako pierwszy krok w złożony i bardzo ciekawy świat technologii MS SQL Server 2008, a szczególnie na wykorzystanie nowych możliwości, jakie wprowadza typ danych XML.

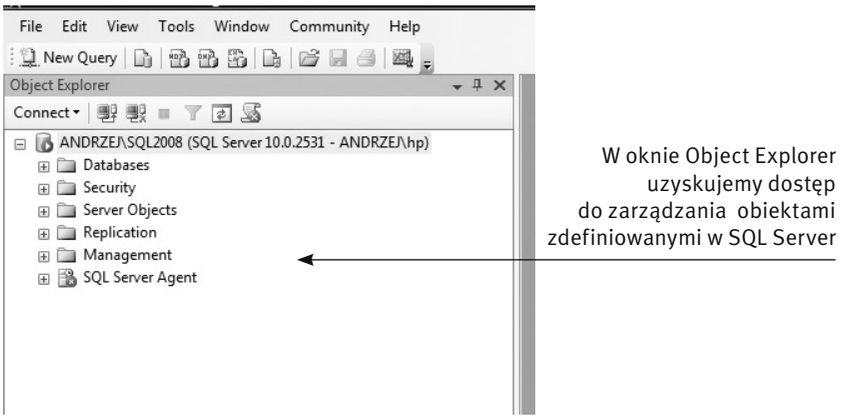
3.2 ĆWICZENIE 1 – ZAPOZNANIE ZE ŚRODOWISKIEM SQL MENAGEMENT STUDIO

Wspólnie z prowadzącym kurs rozpoznajemy środowisko SZBD MS SQL Server 2008. Korzystanie z tego systemu umożliwia specjalne oprogramowanie SQL Server Menagement Studio.

1. Uruchamiamy SQL Server Menagement Studio (lokalizację programu poda prowadzący kurs).
2. Po uruchomieniu programu przechodzimy do logowania się do SQL Servera, na ekranie pojawi się okienko do logowania, w którym wpisujemy w pola wartości takie, jak podane na rysunku 1 (w polu Server name wybieramy nazwę serwera, który został zainstalowany.

Rysunek 1.
Okienko logowania do SQL Server

3. Po poprawnym wpisaniu powyższych wartości uruchomione zostanie oprogramowanie i pojawia się okno SQL Server Menagement Studio.
4. Wspólnie z prowadzącym kurs poznajemy wybrane elementy środowiska SQL Servera.



Rysunek 2.
Widok Object Explorer

4 TWORZENIE DOKUMENTÓW XML ZA POMOCĄ POLECENIA SELECT SQL Z KLAUZULĄ FOR XML

Pierwszym zagadnieniem dotyczącym użycia języka XML w bazach danych jest zwracanie wyników zapytań w postaci dokumentów XML. Jest to wygodny mechanizm, szczególnie gdy aplikacja korzysta z danych o bardziej złożonej strukturze. Zbudowanie dokumentu XML umożliwia utworzenie struktury adekwatnej do wymagań i pozwala uniknąć np. wielokrotnego komunikowania się z bazą danych w celu pobierania różnych fragmentów potrzebnych danych. Baza w odpowiedzi na jedno zapytanie zwraca odpowiedni dokument XML, a aplikacja zajmuje się jego przetworzeniem, co zwykle odbywa się z wykorzystaniem gotowych narzędzi i rozwiązań (np. jako transformacja dokumentu XML do postaci strony HTML lub do dokumentu PDF czy XLS).

SQL Server począwszy od wersji 2000 dopuszcza zwracanie rezultatu wykonania polecenia SELECT w postaci fragmentu lub kompletnego dokumentu XML. Żeby skorzystać z tej możliwości, należy dobrać odpowiedni wariant klauzuli FOR XML, które występuje w czterech wariantach:

- RAW
- AUTO
- EXPLICIT
- PATH

W prostszych przypadkach mogą to być tryby RAW albo AUTO. Służą one do tworzenia prostej reprezentacji zbioru wynikowego jako fragmentu dokumentu XML z opcją ujęcia go w zdefiniowany element główny (root). W przypadku istnienia wymagań generowania bardziej złożonej struktury dokumentu XML, mamy do dyspozycji klauzulę FOR XML w wariantach EXPLICIT oraz PATH. Wariant PATH, podobnie jak RAW czy AUTO, można stosować do większości zapytań, gdyż bardzo łatwo jest dostosować zapytanie do postaci wymaganej przez ten wariant klauzuli FOR XML. Inaczej wygląda sytuacja w przypadku wariantu EXPLICIT, który ma ściśle określone wymagania co do postaci zbioru wynikowego, który ma być przekształcony na postać XML, tzw. **tablica uniwersalna**. Taka tablica jest zwykle generowana za pomocą wieku zapytań łączonych klauzulą UNION. Poszczególne zapytania zwracają wartości różnych kolumn zbiorczego wyniku zapytania.

Najprostszym do zastosowania jest tryb RAW. Jego działanie sprowadza się do wygenerowania dla każdego wiersza ze zbioru wynikowego jednego elementu XML o domyślnej nazwie row. Wartości poszczególnych kolumn dla wiersza stają się wartościami atrybutów elementu row. Można także określić własną nazwę

dla elementu row oraz zamiast atrybutów wybrać generowanie wartości z kolumn jako elementów XML o takiej nazwie jak nazwa kolumny. W ramach trybu RAW można stosować dodatkowe opcje:

- Opcja ROOT powoduje dodanie do rezultatu zapytania elementu głównego o domyślnej nazwie root, bądź dowolnej innej określonej w ramach opcji.
- Korzystanie z opcji ROOT jest powszechną praktyką, gdyż tylko wtedy zwrócony dokument XML spełnia reguły składni i jest poprawnie sformułowany, co umożliwia jego dalsze przetwarzanie.
- Opcja ELEMENTS powoduje, że zamiast atrybutów, do umieszczenia zawartości kolumn są wykorzystane elementy o nazwach takich, jak poszczególne kolumny.

W tym momencie nasuwa się pytanie: Czy stosować elementy czy atrybuty? Jest ono jednym z częściej zadawanych pytań dotyczących planowania struktury dokumentów. Niestety nie ma na nie jednoznacznej odpowiedzi, warto natomiast zdawać sobie sprawę, że korzystanie z atrybutów:

- zmniejsza rozmiar wynikowego dokumentu,
- uniemożliwia nadawanie struktury zawartości atrybutu,
- ogranicza liczebność wystąpień atrybutów o tej samej nazwie w ramach elementu do jednego.

Korzystanie z elementów natomiast:

- zwiększa rozmiar wynikowego dokumentu,
- umożliwia w razie potrzeby nadawanie struktury wartościom elementu.

Dla przykładu weźmy dwa elementy:
‹Dane osoba='Jan Nowak'/› i ‹Dane›‹Osoba›Jan Nowak‹/Osoba›‹/Dane›

Widać, że pierwszy wariant jest bardziej zwięzły. Nie można natomiast wykonać w nim dodania drugiej „osoby" do elementu dane, chyba, że w karkołomny sposób:
‹Dane osoba='Jan Nowak' osoba2='Tomasz Kowalski' /›

W drugim wariancie nie ma takiego problemu:

```
‹Dane›
 ‹Osoba›Jan Nowak‹/Osoba›
 ‹Osoba›Tomasz Kowalski‹/Osoba›
‹/Dane›
```

Podobnie, gdy chcemy nadać zawartości atrybutu czy elementu osoba jakąś strukturę, to pierwszy wariant skutecznie to uniemożliwia. Drugi wariant umożliwia natomiast swobodne wykonanie:

```
‹Dane›
  ‹Osoba›
    ‹Imie›Jan‹/Imie›
    ‹Nazwisko›Nowak‹/Nazwisko›
  ‹/Osoba›
‹/Dane›
```

Kolejnym trybem dostępnym w ramach klauzuli FOR XML jest tryb AUTO. Ma on możliwości zbliżone do RAW z tą różnicą, że potrafi budować proste hierarchie w dokumencie XML. Proces budowania hierarchii jest oparty o **heurystyki**. Analizowane są kolejne wiersze i wartości kolumn. Umożliwia to, przy odpowiednim skonstruowaniu zapytania (sortowanie, kolejność złączeń), sterowanie postacią wyjściowej hierarchii dokumentu XML a także (podobnie jak RAW) osadzenie w wyjściowym dokumencie XML wygenerowanego dla niego dokumentu XML Schema, opisującego postać wyjściowego dokumentu XML.

W przypadku pojawienia się w wyniku zapytania pól binarnych, są one domyślnie kodowane metodą URL Encode. Jeżeli nie jest to odpowiednie rozwiązanie, można skorzystać z opcji BINARY BASE64.

Klauzula FOR XML w wariancie EXPLICIT ma największe możliwości, ale jest też najbardziej złożona i trudna w stosowaniu. Nie nadaje się w przeciwieństwie do RAW i AUTO do zastosowania w dowolnym zapytaniu. Żeby skorzystać z trybu EXPLICIT, tabela wejściowa musi być tzw. **tabelą uniwersalną**, składającą się z kolumn o ustalonej kolejności, nazwach i znaczeniu:

Tabela 1.
Przykład tabeli uniwersalnej

Tag	Parent	Customer!1!cid	Customer!1!name	Order!2!id	Order!2!date	OrderDetail!3!id!id	OrderDetail!3!pid!idref
1	NULL	C1	"Janine"	NULL	NULL	NULL	NULL
2	1	C1	NULL	01	1/20/1996	NULL	NULL
3	2	C1	NULL	01	NULL	OD1	P1
3	2	C1	NULL	01	NULL	OD2	P2
2	1	C1	NULL	02	3/29/1997	NULL	NULL

Przykładowo (patrz tab. 1), pierwszą kolumną musi być kolumna Tag, która zawiera unikatowy numer dla każdego elementu, który będzie generowany. Druga kolumna ma nazwę Parent i określa wartość Tag dla rodzica elementu. Kolejne kolumny definiują składowe elementów XML. Nazwy tych kolumn są tworzone w oparciu o schemat *nazwaElementu!Tag!NazwaAtrybutu!Dyrektywa*. Umożliwia to dokładne sterowanie procesem definiowania struktury elementów i atrybutami. Dokładne omówienie mechanizmu działania trybu EXPLICIT wykracza poza zakres niniejszego opracowania. Klauzula FOR XML EXPLICIT pomimo dużych możliwości definiowania struktury wynikowego dokumentu XML jest obarczona wadami związanymi z trudnościami w przygotowaniu odpowiedniego zapytania. Szczególnie nieprzyjemne jest modyfikowanie już istniejącej struktury. Jest to proces żmudny i podatny na błędy, często skutkujący pisaniem zapytania od nowa.

Wraz z SQL Server 2005 pojawił się tryb PATH. Jest on rozsądnym kompromisem pomiędzy możliwościami w zakresie definiowania struktury XML, a łatwością zapisu tych reguł. W uproszczeniu, tryb PATH bazuje na nazwach nadawanych kolumnom zapytania. Mają one postać zbliżoną do wyrażeń języka XPath, wskazujących elementy bądź atrybuty. Na ich podstawie jest budowana wyjściowa struktura XML. Można ją dodatkowo komplikować poprzez zagnieżdżanie zapytań. Pamiętać należy jedynie o odpowiedniej kolejności występowania kolumn w zapytaniu – te definiujące atrybuty elementu muszą wystąpić przed definiującymi kolejne lub zagnieżdżone elementy.

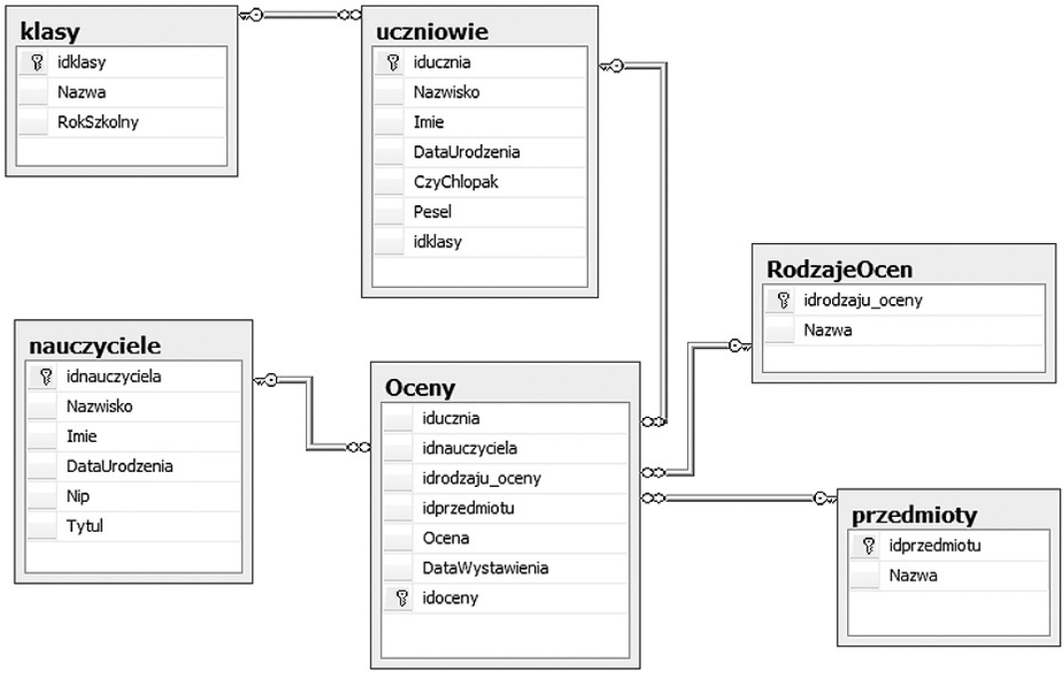
Jeżeli kolumna w wyniku zapytania ma być elementem, to domyślną jego nazwą będzie nazwa kolumny. Jeżeli ma być atrybutem – należy nadać kolumnie alias zaczynający się od znaku @. Alias może zawierać także znaki ukośnika, które określają kolejne poziomy zagnieżdżenia. Z kolei znak * powoduje, że w przypadku kolumny typu XML, jej zawartość będzie osadzona w wyniku zapytania wprost. Dla kolumn innych typów wstawiony będzie węzeł tekstowy z zawartością.

Wszystkie opisane warianty klauzuli FOR XML mają jeszcze kilka opcji, z którymi warto się zapoznać. Jako przykład można podać opcję TYPE powodującą, że zwrócona wartość jest traktowana jak zmienna typu XML, co umożliwia wygodne jej przetwarzanie. Równie ciekawa jest opcja XSNIL, która umożliwia umieszczenie w wynikowym dokumencie XML elementów, które mają w wejściowym zapytaniu wartość null i domyślnie nie byłyby umieszczone w wyniku. Jest to szczególnie istotne przy przetwarzaniu zwróconych danych w aplikacji, która niekoniecznie ma skąd wziąć pełną listę dopuszczalnych elementów, które można w ramach przetwarzania uzupełnić. Oczywiście można próbować ten problem rozwiązać za pomocą osadzania XML Schema, ale jest to bardziej złożone i pracochłonne.

Opisane pokrótce możliwości SQL Server 2008 w zakresie zwracania wyników zapytań w postaci XML nie wyczerpuje wszystkich możliwości istniejących w tym narzędziu, sygnalizują jedynie podstawowe mechanizmy i ich stosowanie.

4.1 ĆWICZENIE 2 – WYKORZYSTANIE KLAUZULI FOR XML RAW

1. W folderze Databases wybrać bazę danych ElektronicznyDziennikOcen.
2. Omówić wspólnie z prowadzącym schemat bazy danych ElektronicznyDziennikOcen.
   Ponieważ wiele ćwiczeń realizowanych będzie z wykorzystaniem przykładowej bazy danych ElektronicznyDziennikOcen, należy dobrze zapoznać się z jej schematem i zrozumieć logikę tej bazy danych. W ogólnym ujęciu jest to fragment bazy danych, która mogłaby być wykorzystana do gromadzenia danych o wystawionych ocenach w obrębie danej szkoły. Schemat bazy danych jest pokazany na rysunku 3.



Rysunek 3.
Schemat przykładowej bazy danych ElektronicznyDziennikOcen

3. Przejść do edytora zapytań naciskając przycisk New Query.
4. Napisać w oknie edycyjnym i wykonać następujące zapytanie (wciskając klawisz F5):
   SELECT **Nazwa**,
   RokSzkolny
   FROM **Klasy**
   Klasyczne zapytanie języka SQL zwraca dane w postaci tabeli:

Nazwa	RokSzkolny
Ia	2008/2009
IIIC	2008/2009
Ib	2008/2009
IIc	2008/2009

Rysunek 4.
Wynik zapytania

5. Dodanie klauzuli FOR XML spowoduje zwrócenie wyniku w postaci dokumentu XML. W tym celu modyfikujemy pierwotną postać zapytania :
   SELECT **Nazwa**,
   RokSzkolny
   FROM **Klasy**
   FOR XML RAW
   Jest to najprostsza postać przekształcenia wyniku zapytania do postaci XML. Poniżej otrzymany wynik:
   ‹row Nazwa=”Ia” RokSzkolny=”2008/2009” /›
   ‹row Nazwa=”IIIC” RokSzkolny=”2008/2009” /›
   ‹row Nazwa=”Ib” RokSzkolny=”2008/2009” /›
   ‹row Nazwa=”IIc” RokSzkolny=”2008/2009” /›
   Jak widać, dla każdego wiersza wyniku zapytania wygenerowany został element o nazwie ***row*** zawierajacy atrybuty o nazwach kolumn otrzymanych w wyniku zapytania. Należy zwrócić uwagę, że zaprezentowany wynik nie jest poprawnym dokumentem XML, ponieważ nie zawiera elementu głównego.

6. Dokonujemy modyfikacji zapytania do poniższej postaci:
   SELECT **Nazwa**,
   RokSzkolny
   FROM **Klasy**
   FOR XML RAW(‘Klasa’),ELEMENTS, ROOT(‘Klasy’)
   Dokonane zmiany wymuszają następujące działanie:

a. Zmiana nazwy elementu generowanego dla każdego wiersza wyniku zapytania – RAW(‘Klasa’).
b. Wykorzystanie opcji ELEMENTS, powoduje, że wynikowy dokument XML zbudowany jest z elementów a nie atrybutów.
c. Dodanie elementu głównego (root) o nazwie Klasy.
   Po dokonanych zmianach wynik zapytania prezentuje się następująco:

```
‹Klasy›
 ‹Klasa›
  ‹Nazwa›Ia‹/Nazwa›
  ‹RokSzkolny›2008/2009‹/RokSzkolny›
 ‹/Klasa›
 ‹Klasa›
  ‹Nazwa›IIIC‹/Nazwa›
  ‹RokSzkolny›2008/2009‹/RokSzkolny›
 ‹/Klasa›
 ‹Klasa›
  ‹Nazwa›Ib‹/Nazwa›
  ‹RokSzkolny›2008/2009‹/RokSzkolny›
 ‹/Klasa›
 ‹Klasa›
  ‹Nazwa›IIc‹/Nazwa›
  ‹RokSzkolny›2008/2009‹/RokSzkolny›
 ‹/Klasa›
‹/Klasy›
```

7. Zmianę nazw elementów składowy dokumentu XML realizuje się poprzez nadanie nowej nazwy kolumnie wynikowe zapytania z wykorzystaniem opcji AS, jak w poniższym przykładzie:
   SELECT **Nazwa** AS **NazwaKlasy**,
   RokSzkolny
   FROM **Klasy**
   FOR XML RAW(‘Klasa’),ELEMENTS, ROOT(‘Klasy’) ,
   Otrzymany wynik ma nastepującą postać:

```
‹Klasy›
 ‹Klasa›
  ‹NazwaKlasy›Ia‹/NazwaKlasy›
  ‹RokSzkolny›2008/2009‹/RokSzkolny›
 ‹/Klasa›
 ‹Klasa›
  ‹NazwaKlasy›IIIC‹/NazwaKlasy›
  ‹RokSzkolny›2008/2009‹/RokSzkolny›
 ‹/Klasa›
 ‹Klasa›
  ‹NazwaKlasy›Ib‹/NazwaKlasy›
  ‹RokSzkolny›2008/2009‹/RokSzkolny›
 ‹/Klasa›
 ‹Klasa›
  ‹NazwaKlasy›IIc‹/NazwaKlasy›
  ‹RokSzkolny›2008/2009‹/RokSzkolny›
 ‹/Klasa›
‹/Klasy›
```

Podsumowując, klauzula FOR XML z opcją RAW umożliwia formatowanie wyniku zapytania do dokumentu XML, tworząc dla każdego wiersza element. Opcja RAW można użyć do tworzenia bardziej złożonych struktur dokumentów XML.

8. Zadanie do samodzielnego wykonania.
   Napisać zapytanie, które przygotuje dokument XML na postawie zawartości tabeli Uczniowie.

4.2 ĆWICZENIE 3 – WYKORZYSTANIE KLAUZULI FOR XML AUTO

Opcja AUTO klauzuli FOR XML umożliwia budowanie złożonych struktur dokumentów XML na podstawie operacji złączenia tabel w zapytaniu. Działanie opcji AUTO zostanie pokazane ramach tego ćwiczenia.

1. W folderze Databases wybrać bazę danych ElektronicznyDziennikOcen.
2. Przejść do edytora zapytań naciskając przycisk New Query.
3. Napisać w oknie edycyjnym i wykonać następujące zapytanie (naciskając klawisz F5):

```
SELECT Nazwa ,
       RokSzkolny,
       Nazwisko,
       Imie,
      Pesel
FROM Klasy as Klasa join Uczniowie as Uczen
ON Klasa.idklasy=Uczen.idklasy
ORDER BY NazwaKlasy
FOR XML AUTO,ELEMENTS, ROOT('Klasy')
```

4. Otrzymany wynik prezentuje się następująco :

```
⊟ <Klasy>
  ⊟ <Klasa>
      <NazwaKlasy>Ia</NazwaKlasy>
      <RokSzkolny>2008/2009</RokSzkolny>
    ⊟ <Uczen>
        <Nazwisko>Gąska</Nazwisko>
        <Imie>Wacek</Imie>
        <Pesel>91031199123</Pesel>
      </Uczen>
    ⊞ <Uczen>...
    ⊞ <Uczen>...
    ⊞ <Uczen>...
    ⊞ <Uczen>...
    ⊞ <Uczen>...
    ⊞ <Uczen>...
    </Klasa>
  ⊞ <Klasa>...
  ⊞ <Klasa>...
  ⊞ <Klasa>...
  </Klasy>
```

Proszę zwrócić uwagę jak zmieni się wynik tego zapytania jeżeli zmienimy klauzulę ORDR BY na następującą: ORDER BY Uczen.iducznia. Wynik przedyskutować z prowadzącym kurs.

Korzystając z opcji AUTO klauzuli FOR XML można tworzyć zagnieżdżone dokumenty XML – w naszym przykładzie w elemencie ‹Klasa› zawarty jest zbiór elementów ‹Uczen› (uczniowie danej klasy).

5. Zadanie do samodzielnego wykonania.
   Napisać zapytanie, które przygotuje dokument XML na postawie zawartości tabeli Nauczyciele, Oceny i Przedmioty.

4.3 ĆWICZENIE 4 – WYKORZYSTANIE KLAUZULI FOR XML PATH

Wraz z SQL Server 2005 pojawił się tryb PATH. Jest on rozsądnym kompromisem pomiędzy możliwościami w zakresie definiowania struktury XML, a łatwością zapisu tych reguł. W uproszczeniu, tryb PATH bazuje na nazwach nadawanych kolumnom zapytania. Mają one postać zbliżoną do wyrażeń języka XPath, wskazujących elementy bądź atrybuty. Na ich podstawie jest budowana wyjściowa struktura XML. Można ją dodatko-

wo komplikować poprzez zagnieżdżanie zapytań. Pamiętać należy jedynie o odpowiedniej kolejności występowania kolumn w zapytaniu – te definiujące atrybuty elementu muszą wystąpić przed definiującymi kolejne lub zagnieżdżone elementy.

Jeżeli kolumna w wyniku zapytania ma być elementem, to domyślną jego nazwą będzie nazwa kolumny. Jeżeli ma być atrybutem – należy nadać kolumnie alias zaczynający się od znaku @. Alias może zawierać także znaki ukośnika, które określają kolejne poziomy zagnieżdżenia. Z kolei znak * powoduje, że w przypadku kolumny typu XML, jej zawartość będzie osadzona w wyniku zapytania wprost. Dla kolumn innych typów wstawiony będzie węzeł tekstowy z zawartością.

1. W folderze Databases wybrać bazę danych ElektronicznyDziennikOcen.
2. Przejść do edytora zapytań naciskając przycisk New Query.
3. Napisać w oknie edycyjnym i wykonać następujące zapytanie (wciskając klawisz F5):

```
SELECT Pesel as '@Pesel',
        Nazwisko ,
        Imie,
       CASE CzyChlopak
        WHEN 1 THEN 'M  czyzna'
        ELSE 'Kobieta'
       END AS Plec
FROM Uczniowie
WHERE idklasy=1
FOR XML PATH('Uczen'), ELEMENTS, ROOT('Uczniowie')
```

4. Otrzymany wynik prezentuje się następująco :

```
‹Uczniowie›
 ‹Uczen Pesel="91031199123"›
  ‹Nazwisko›Gąska‹/Nazwisko›
  ‹Imie›Wacek‹/Imie›
  ‹Plec›Mężczyzna‹/Plec›
 ‹/Uczen›
 ‹Uczen Pesel="92051577646"›
  ‹Nazwisko›Krówka‹/Nazwisko›
  ‹Imie›Rysio‹/Imie›
  ‹Plec›Mężczyzna‹/Plec›
 ‹/Uczen›
 ‹Uczen Pesel="93030399846"›
  ‹Nazwisko›Zebra‹/Nazwisko›
  ‹Imie›Wojtek‹/Imie›
  ‹Plec›Mężczyzna‹/Plec›
 ‹/Uczen›
‹/Uczniowie›
```

Jak widać, za pomocą opcji PATH można budować struktury dokumentów XML złożone z elementów i atrybutów. Wynik zapytania i jego postać należy omówić z prowadzącym kurs.

5. Zadanie do samodzielnego wykonania.
   Napisać zapytanie, które przygotuje dokument XML na postawie zawartości tabeli Nauczyciele, Oceny i Przedmioty – wykorzystując opcję PATH klauzuli FOR XML.

4.4 ĆWICZENIE 5 – TWORZENIE DOKUMENTU XML Z WYKORZYSTANIEM ZAPYTAŃ SKORELOWANYCH

Stosowanie różnych opcji klauzuli FOR XML daje duże możliwości w zakresie tworzenia dokumentów XML, jednak w wielu przypadkach trzeba skorzystać z możliwości języka SQL w zakresie zagnieżdżania zapytań.

1. W folderze Databases wybrać bazę danych ElektronicznyDziennikOcen.
2. Przejść do edytora zapytań naciskając przycisk New Query.
3. Napisać w oknie edycyjnym i wykonać następujące zapytanie (naciskając klawisz F5):

```
SELECT TOP 3
    nazwa AS NazwaKlasy,
    RokSzkolny,
```

```
(SELECT Nazwisko AS Nazwisko,
        Imie,
        DataUrodzenia,
        Pesel,
       CASE CzyChlopak
        WHEN 1 THEN 'Mężczyzna'
        ELSE 'Kobieta'
       END AS Plec,
       (SELECT TOP 5 przedmiot AS Przedmiot,
            ImieNauczyciela+' '+NazwiskoNauczyciela AS Nauczyciel,
            RodzajOceny AS RodzajOceny,
            Ocena AS Ocena
       FROM v_oceny
       WHERE iducznia=uczniowie.iducznia
       FOR XML RAW('Ocena'),ELEMENTS,ROOT('Oceny'),TYPE )
  FROM Uczniowie
  WHERE idklasy=klasy.idklasy
  FOR XML RAW('Uczen'),ELEMENTS,ROOT('Uczniowie'),TYPE)
FROM Klasy
FOR XML RAW('Klasa'),ELEMENTS,ROOT('Klasy')
```

4. Wynik zapytania jest następujący (zaprezentowano tylko jego fragment):

```
‹Klasy›
 ‹Klasa›
  ‹NazwaKlasy›Ia‹/NazwaKlasy›
  ‹RokSzkolny›2008/2009‹/RokSzkolny›
  ‹Uczniowie›
   ‹Uczen›
    ‹Nazwisko›Gąska‹/Nazwisko›
    ‹Imie›Wacek‹/Imie›
    ‹DataUrodzenia›1991-03-11‹/DataUrodzenia›
    ‹Pesel›91031199123‹/Pesel›
    ‹Plec›Mężczyzna‹/Plec›
    ‹Oceny›
     ‹Ocena›
      ‹Przedmiot›Geografia‹/Przedmiot›
      ‹Nauczyciel›Hala Gepard‹/Nauczyciel›
      ‹RodzajOceny›Odpowiedź‹/RodzajOceny›
      ‹Ocena›2.00‹/Ocena›
     ‹/Ocena›
     ‹Ocena›
      ‹Przedmiot›Fizyka‹/Przedmiot›
      ‹Nauczyciel›Saba Pantera‹/Nauczyciel›
      ‹RodzajOceny›Odpowiedź‹/RodzajOceny›
      ‹Ocena›3.00‹/Ocena›
     ‹/Ocena›
```

UWAGA: Aby obejrzeć cały wynik zapytania należy zapisać go w pliku a następnie otworzyć ten plik w dowolnej przegladarce internetowej. Szczegóły zapytania i otrzymanego wyniku należy przedyskutować z prowadzącym kurs.

5. Zadanie do samodzielnego wykonania.

Napisać zapytanie, które przygotuje dokument XML w postaci przedstawionej na rys. 4

```
<Klasy>
  <Klasa>
    <NazwaKlasy>Ia</NazwaKlasy>
    <RokSzkolny>2008/2009</RokSzkolny>
    <Uczniowie>
      <Uczen>...
      <Uczen>...
      <Uczen>...
      <Uczen>...
      <Uczen>...
      <Uczen>...
      <Uczen>
        <Nazwisko>Lisek</Nazwisko>
        <Imie>Olenka</Imie>
        <DataUrodzenia>1998-09-08</DataUrodzenia>
        <Pesel>76767689876</Pesel>
        <Plec>Kobieta</Plec>
      </Uczen>
    </Uczniowie>
  </Klasa>
  <Klasa>...
  <Klasa>...
</Klasy>
```

Rysunek 4.
Postać przykładowego dokumentu XML

5. TYP DANYCH XML

5.1 CHARAKTERYSTYKA TYPU DANYCH XML

Konsekwencją wzrostu popularności języka XML było wprowadzenie do standardu SQL typu danych XML. Całkowicie to 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.

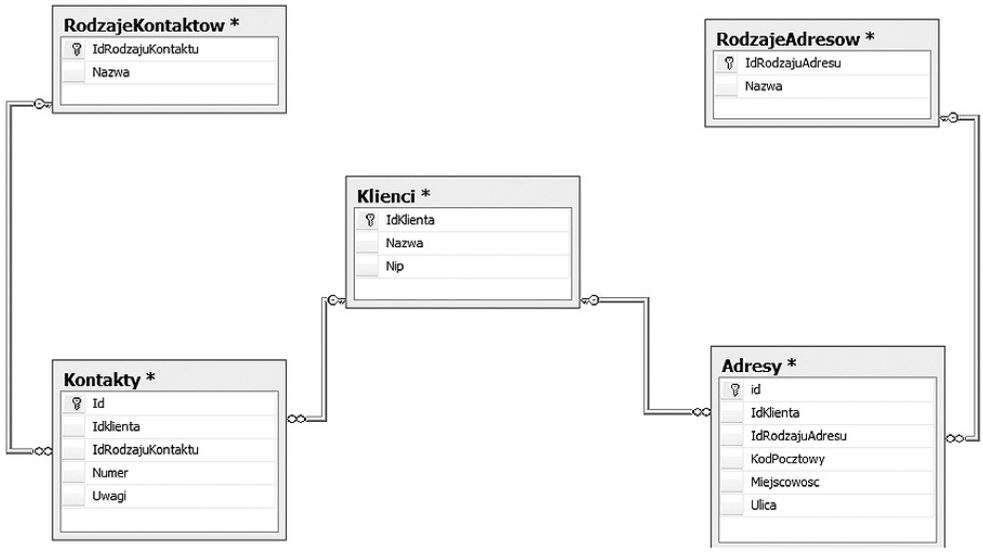
Zanim skorzysta się z typu danych XML warto zapoznać się z jego dokumentacją. Szczególnie chodzi tu o sposób przechowywania dokumentu oraz o ograniczenia związane z samym typem danych. Warto również zastanowić się nad korzystaniem z kolekcji XML Schema oraz indeksów XML. Istotną informacja jest to, że 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. 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 two-

rząc 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.

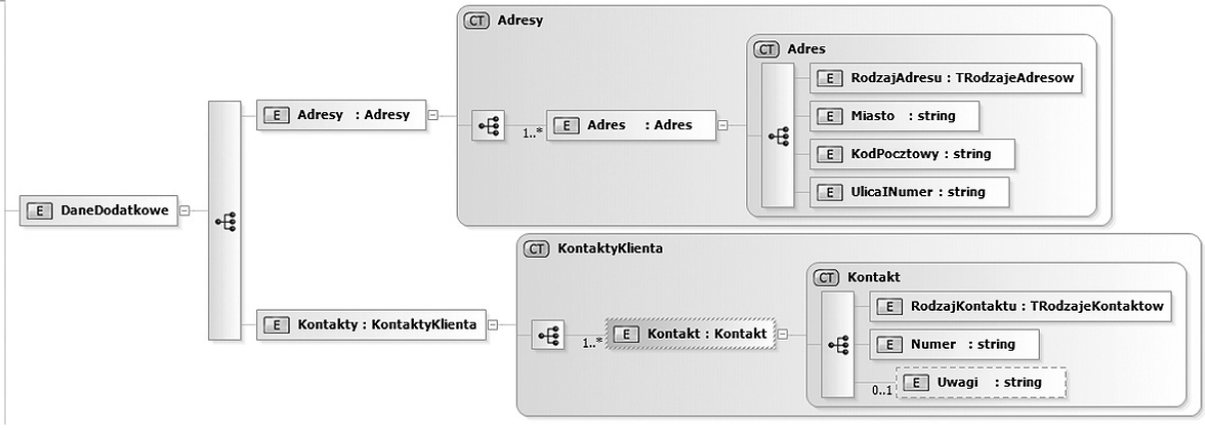
5.2 MOŻLIWOŚCI WYKORZYSTANIA TYPU XML DO UPROSZCZENIA SCHEMATU BAZY DANYCH

Jedną z podstawowych korzyści wykorzystania typu danych XML w relacyjnych bazach danych jest możliwość znacznego uproszczenia schematu bazy danych dzięki umieszczeniu części danych w dokumentach XML. Istotę tego procesu omówimy na następującym przykładzie. Zaprezentowany na rysunku 5 przykład bazy danych składa się z pięciu tabel i jest jak najbardziej poprawny z punktu widzenia relacyjnej bazy danych. Jednak rozproszenie danych opisujących klienta w wielu tabelach sprawia problemy w trakcie korzystania z takiej bazy danych.



Rysunek 5.
Schemat przykładowej bazy danych KursXMLPrzyklady

Dla danych zapisywanych w tabelach RodzajeKontaktow, RodzajeAdresow, Adresy i Kontakty można wyobrazić sobie schemat dokumentu XML przedstawiony na rysunku 6.



Rysunek 6.
Przykład schematu dokumentu XML dla danych dodatkowych opisujących klienta

W dalszych ćwiczeniach powrócimy do tego schematu dokumentu XML. Przykładowy dokument XML zgodny z tym schematem mógłby wyglądać następująco:

```
‹DaneDodatkowe›
 ‹Adresy›
  ‹Adres›
   ‹RodzajAdresu›Korespondencyjny‹/RodzajAdresu›
   ‹KodPocztowy›04-765‹/KodPocztowy›
   ‹Miejscowosc›Opole‹/Miejscowosc›
   ‹Ulica›ul.Nowa 13‹/Ulica›
  ‹/Adres›
  ‹Adres›
   ‹RodzajAdresu›NaFakture‹/RodzajAdresu›
   ‹KodPocztowy›45-324‹/KodPocztowy›
   ‹Miejscowosc›Sopot‹/Miejscowosc›
   ‹Ulica›ul.Morska 124‹/Ulica›
  ‹/Adres›
 ‹/Adresy›
 ‹Kontakty›
  ‹Kontakt›
   ‹RodzajKontaktu›Email‹/RodzajKontaktu›
   ‹Numer›KapitanNemo@wp.pl‹/Numer›
  ‹/Kontakt›
  ‹Kontakt›
   ‹RodzajKontaktu›Email‹/RodzajKontaktu›
   ‹Numer›81 8863412‹/Numer›
   ‹Uwagi›wew.123‹/Uwagi›
  ‹/Kontakt›
 ‹/Kontakty›
‹/DaneDodatkowe›
```

Ponieważ przykładowy dokument XML zawiera wszystkie niezbędne dane, które w schemacie relacyjnym były rozpisane w czterech tabelach, można zmodyfikowac wyjściowy schemat bazy danych wykorzystując kolumnę typu XML. Zamiast pięciu tabel możemy zaprojektować jedną tabelę zawierajacą kolumnę typu XML zawierajacą dodatkowe dane opisujace klienta.

Nowa propozycja schematu bazy danych z rysunku 5 może wyglądać tak, jak na rysunku 7.

Column Name	Data Type	Allow Nulls
idklienta	int	☐
Nazwa	varchar(256)	☐
Nip	varchar(10)	☐
DaneDodatkowe	xml	☐

Rysunek 7.
Schemat tabeli KlienciXML

Na tym prostym przykładzie widać, że typ XML można wykorzystać na etapie projektowania do uproszczenia końcowego schematu bazy danych. W naszym przykładzie zamiast pięciu tabel mamy jedną tabelę i zachowujemy wszystkie dane. W praktyce można osiągnąć jeszcze bardzie spektakularne uproszczenia projektowanych baz danych dzięki zastosowaniu typu XML.

5.3 ĆWICZENIE 6 – WYKONANIE PROJEKTU BAZY DANYCH Z WYKORZYSTANIEM TYPU XML

W ramach ćwiczenia każdy uczestnik przygotuje projekt bazy danych składający się z około 5 tabel. W projekcie należy wykorzystać kolumny typu XML. Dziedzina problemu dowolna. Po wykonaniu projektu należy skonsultować go z prowadzącym ćwiczenia. Projekt powinien zawierać :

- Nazwy tabel.
- Nazwy kolumn i określenie typu wartości.
- Zaznaczenie kluczy podstawowych o obcych.

Dla zaprojektowanych kolumn typu XML przygotować, w edytorze tekstu, przykładową postać dokumentu XML.

6 METODY TYPU XML

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.

6.1 ĆWICZENIE 7 – POBIERANIE DANYCH Z DOKUMENTU XML Z WYKORZYSTANIEM METODY VALUE

Metoda ***value*** zastosowana do danych typu XML zwraca wartość skalarną, która jest zawartością wskazanego elementu lub atrybutu.

1. Zapisać w swoim komputerze przykładowy plik DziennikOcen.xml – prowadzący wskaże lokalizację której należy ten plik pobrać.
2. Otworzyć przykładowy plik w przeglądarce internetowej – zapoznać się z jego strukturą – przedyskutować wątpliwości z prowadzącym kurs.
3. Przejść do edytora zapytań naciskając przycisk New Query

4. W edytorze zapytań napisać następujący skrypt:

```
DECLARE @dane XML
SELECT @dane=K
FROM OPENROWSET(BULK N'D:\DziennikOcen.xml', SINGLE_BLOB) AS T(k)
SELECT @dane.value('DziennikOcen[1]/Klasa[3]/NazwaKlasy[1]', 'varchar(128)')
```

Tu wpisać lokalizację przykładowego pliku

5. Wykonać skrypt poprzez naciśnięcie klawisza F5.
6. W wyniku zapytania powinniśmy otrzymać wartość Ib.
7. Omówić poszczególne polecenia skryptu z prowadzącym kurs.

6.2 ĆWICZENIE 8 – POBIERANIE DANYCH Z DOKUMENTU XML Z WYKORZYSTANIEM METODY QUERY

Metoda ***query*** zastosowana do danych typu XML zwraca dokument XML, który jest zlokalizowany we wskazanym miejscu.

1. Przejść do edytora zapytań naciskając przycisk New Query
2. W edytorze zapytań napisać następujący skrypt:

```
DECLARE @dane XML
SELECT @dane=K
FROM OPENROWSET(BULK N'D:\DziennikOcen.xml', SINGLE_BLOB) AS T(k)
SELECT @dane.query('DziennikOcen[1]/Klasa[3]', 'varchar(128)')
```

Tu wpisać lokalizację przykładowego pliku

3. Wykonać skrypt poprzez naciśnięcie klawisza F5.
4. W wyniku zapytania powinniśmy otrzymać dokument XML o następującej postaci:

```
‹Klasa›
 ‹NazwaKlasy›Ib‹/NazwaKlasy›
 ‹RokSzkolny›2008/2009‹/RokSzkolny›
 ‹Uczniowie›
  ‹Uczen›
   ‹NazwiskoImie›Jasio Wilczek‹/NazwiskoImie›
   ‹DataUrodzenia›1999-12-12‹/DataUrodzenia›
   ‹Pesel›99121209876‹/Pesel›
   ‹Oceny›
    ‹Ocena›
     ‹Nauczyciel›Saba Pantera‹/Nauczyciel›
     ‹Przedmiot›Informatyka‹/Przedmiot›
     ‹RodzajOceny›Praca domowa‹/RodzajOceny›
     ‹DataWystawienia›2009-02-09‹/DataWystawienia›
     ‹Ocena›4.00‹/Ocena›
    ‹/Ocena›
   ‹/Oceny›
  ‹/Uczen›
 ‹/Uczniowie›
‹/Klasa›
```

5. Omówić poszczególne polecenia skryptu z prowadzącym kurs.

UWAGA: Możliwości metody ***query*** są dużo większe niż zademonstrowany przykład, ale te szczegóły wykraczają poza zakres kursu.

6.3 ĆWICZENIE 9 – POBIERANIE DANYCH Z DOKUMENTU XML Z WYKORZYSTANIEM METODY NODES.

Metoda ***nodes*** zastosowana do danych typu XML zwraca wynik w postaci tabeli i z tego powodu jest wykorzystywana do budowania zapytań, w których można wykorzystywać dane z tabel relacyjnych i dokumentów XML.

1. Przejść do edytora zapytań naciskając przycisk New Query
2. W edytorze zapytań napisać następujący skrypt:

```
DECLARE @dane XML
SELECT @dane=K
FROM OPENROWSET(BULK N'D:\DziennikOcen.xml', SINGLE_BLOB) AS T(k);
WITH CTE as
(
```

Tu wpisać lokalizację przykładowego pliku

```
SELECT k.value('Przedmiot[1]','varchar(128)') as Przedmiot,
       k.value('Ocena[1]','numeric(5,2)') as Ocena
FROM @dane.nodes('DziennikOcen/Klasa/Uczniowie/Uczen/Oceny/Ocena') t(k)
)
SELECT Przedmiot,
       AVG(ocena) as Srednia
FROM CTE
GROUP BY Przedmiot
```

3. Wykonać skrypt poprzez naciśnięcie klawisza F5.
4. W wyniku zapytania powinniśmy otrzymać tabelę o następującej postaci:

Przedmiot	Srednia
Fizyka	2.891891
Geografia	3.058823
Informatyka	2.862745
Literatura	3.125000
Matematyka	3.024590

5. Omówić poszczególne polecenia skryptu z prowadzącym kurs.
6. Zadanie do samodzielnego wykonania.
   Napisać zapytanie, które z przykładowego dokumentu XML zwróci tabelę zawierającą; nazwisko, imię i numer pesel oraz średnią ocen danego ucznia z matematyki.

6.4 OMÓWIENIE OPERATORA CROSS APPLY

Klasyczny operator łączenia tabel JOIN łączy dwie tabele, które w momencie łączenia mają skończoną postać. Istotą operatora CROSS APPLY, wykorzystywanego do łączenia tabel w klauzuli FROM polecenia SELECT jest to, że dla każdego wiersza tabeli wymienionej przed operatorem APPLY jest tworzona dynamicznie tabela, z którą ten wiersz łączymy (jeden wiersz tabeli wyjściowej łączymy z każdym wierszem w tabeli drugiej). Często operator APPLY jest wykorzystywany do zapytań, które muszą przeglądać wiele dokumentów XML zapisanych w tabeli.

6.5 ĆWICZENIE 10 – WYKONANIE ZAPYTANIA WYKORZYSTUJĄCEGO OPERATOR CROSS APPLY

1. W folderze Databases wybrać bazę danych ElektronicznyDziennikOcen.
2. W bazie danych ElektronicznyDziennikOcen znajduje się tabela o nazwie UczniowieXML o następującej strukturze przedstawionej na rysunku 8.

iducznia	int	☐
Nazwisko	varchar(50)	☐
Imie	varchar(50)	☐
DataUrodzenia	date	☑
CzyChlopak	bit	☐
Pesel	varchar(11)	☑
idklasy	int	☐
Oceny	xml	☑

Rysunek 8.
Struktura tabeli UczniowieXML

3. W kolumnie Oceny (typu XML) zapisane są oceny uzyskane przez danego ucznia w następującej postaci :
   ‹Oceny›

```
<Ocena>
  <Przedmiot>Geografia</Przedmiot>
  <Nauczyciel>Hala Gepard</Nauczyciel>
  <RodzajOceny>Odpowiedź</RodzajOceny>
  <Ocena>2.00</Ocena>
</Ocena>
<Ocena>
  <Przedmiot>Fizyka</Przedmiot>
  <Nauczyciel>Saba Pantera</Nauczyciel>
  <RodzajOceny>Odpowiedź</RodzajOceny>
  <Ocena>3.00</Ocena>
</Ocena>
<Ocena>
  <Przedmiot>Geografia</Przedmiot>
  <Nauczyciel>Tadeusz Łoś</Nauczyciel>
  <RodzajOceny>Odpowiedź</RodzajOceny>
  <Ocena>2.00</Ocena>
</Ocena>
<Ocena>
  <Przedmiot>Geografia</Przedmiot>
  <Nauczyciel>Hala Gepard</Nauczyciel>
  <RodzajOceny>Odpowiedź</RodzajOceny>
  <Ocena>2.00</Ocena>
</Ocena>
<Ocena>
  <Przedmiot>Matematyka</Przedmiot>
  <Nauczyciel>Wojciech Lew</Nauczyciel>
  <RodzajOceny>Odpowiedź</RodzajOceny>
  <Ocena>1.00</Ocena>
</Ocena>
</Oceny>
```

4. Przejść do edytora zapytań naciskając przycisk New Query
5. W edytorze napisać zapytanie, które dla każdego ucznia policzy średnia jego ocen:

```
SELECT Nazwisko,
       Imie,
       Pesel,
       (
         SELECT AVG(k.value('Ocena[1]','numeric(5,2)'))
         FROM Oceny.nodes('Oceny/Ocena') t(k)
       ) AS Srednia
FROM UczniowieXML
   WHERE  (
            SELECT AVG(k.value('Ocena[1]','numeric(5,2)'))
            FROM Oceny.nodes('Oceny/Ocena') t(k)
       )  IS NOT NULL
```

W powyższym przykładzie nie korzystamy z operatora APPLY, ponieważ dla każdego wiersza tabeli UczniowieXML wykonywane jest zapytanie, które z kolumny Oceny(typu XML) oblicza średnia ocen danego ucznia.

6. Otrzymany wynik ma postać:

Nazwisko	Imie	Pesel	Srednia
Kotek	Kasia	92031275446	3.000000
Piesek	Jan	92051587746	2.980000
Lisek	Kasia	92022277654	3.229508
Kurka	Jola	92060288788	3.030769
Gąska	Wacek	91031199123	2.857142
Krówka	Rysio	92051577646	2.781818
Zebra	Wojtek	93030399846	3.075757
Gazela	Basia	92111177446	2.932432
Sarenka	Rysio	92121278766	2.563636
Konik	Kasia	93031275446	2.686274
Ryba	Jan	93051587746	3.123076
Kura	Kasia	93022277654	3.035087
Łoś	Jola	93060288788	3.019607
Miś	Wacek	93031199123	2.875000
Okoń	Rysio	93051577646	3.035714
Płotka	Wojtek	93030399846	2.907692
Różycz...	Basia	93111177446	2.842857
Wilczek	Jasio	99121209876	4.000000
Foka	Jola	92060223454	4.333333

7. W edytorze zapisać zapytanie, które zwróci tabelę zawierającą nazwę przedmiotu oraz średnia ocen uzyskaną przez wszystkich uczniów z danego przedmiotu;

```
SELECT Przedmiot,
        AVG(Ocena) as srednia
FROM UczniowieXML CROSS APPLY (
                        SELECT k.value('Ocena[1]','numeric(5,2)') AS Ocena,
                                k.value('Przedmiot[1]','varchar(64)') AS Przedmiot
                        FROM Oceny.nodes('Oceny/Ocena') t(k)
                      )     AS A
GROUP BY Przedmiot
```

8. Otrzymamy wynik w następującej postaci:

Przedmiot	srednia
Informatyka	2.862745
Literatura	3.125000
Fizyka	2.891891
Geografia	3.058823
Matematyka	3.024590

9. Omówić z prowadzącym kurs szczegóły podanego zapytania.
10. Zadanie do samodzielnego wykonania.
    Napisać zapytanie, które zwróci nazwisko nauczyciela i średnia ocen wystawiona przez danego nauczyciela z odpowiedzi.

7. WALIDACJA DANYCH XML – SCHEMATY XSD

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, które zapisuje się jedna pod drugą. 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 schematu XML jest proste i ogranicza się do wykonania jednego polecenia. Od tego momentu można używać zdefiniowanej w ten sposób kolekcji przy deklaracjach kolumn typu XML

7.1 ĆWICZENIE 11 – DEFINIOWANIE SCHEMATU XSD JAKO OBIEKTU BAZY DANYCH.

Przykład schematu XML, pokazany na rysunku 7, może zostać zdefiniowany jako obiekt bazy danych. Pokazanej na rysunku 6 graficznej prezentacji schematu odpowiada odpowiedni dokument XSD w następującej postaci:

```
‹xs:schema xmlns:xs=”http://www.w3.org/2001/XMLSchema” elementFormDefault=”qualified” attributeFormDefault=”unqualified”›
 ‹xs:element name=”DaneDodatkowe”›
  ‹xs:complexType›
   ‹xs:sequence›
    ‹xs:element name=”Adresy” type=”Adresy”/›
    ‹xs:element name=”Kontakty” type=”KontaktyKlienta”/›
   ‹/xs:sequence›
  ‹/xs:complexType›
 ‹/xs:element›
 ‹xs:complexType name=”Adresy”›
  ‹xs:sequence›
   ‹xs:element name=”Adres” type=”Adres” maxOccurs=”unbounded”/›
  ‹/xs:sequence›
 ‹/xs:complexType›
 ‹xs:complexType name=”KontaktyKlienta”›
  ‹xs:sequence›
   ‹xs:element name=”Kontakt” type=”Kontakt” maxOccurs=”unbounded”/›
  ‹/xs:sequence›
 ‹/xs:complexType›
 ‹xs:complexType name=”Adres”›
  ‹xs:sequence›
   ‹xs:element name=”RodzajAdresu” type=”TRodzajeAdresow”/›
   ‹xs:element name=”Miasto” type=”xs:string”/›
   ‹xs:element name=”KodPocztowy”›
    ‹xs:simpleType›
     ‹xs:restriction base=”xs:string”›
      ‹xs:minLength value=”6”/›
      ‹xs:maxLength value=”6”/›
     ‹/xs:restriction›
    ‹/xs:simpleType›
   ‹/xs:element›
   ‹xs:element name=”UlicaINumer” type=”xs:string”/›
  ‹/xs:sequence›
 ‹/xs:complexType›
 ‹xs:complexType name=”Kontakt”›
  ‹xs:sequence›
   ‹xs:element name=”RodzajKontaktu” type=”TRodzajeKontaktow”/›
   ‹xs:element name=”Numer” type=”xs:string”/›
```

```
    <xs:element name="Uwagi" type="xs:string" minOccurs="0"/>
   </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="TRodzajeKontaktow">
   <xs:annotation>
    <xs:documentation>Typ wyliczeniowy dla rodzajów kontaktów</xs:documentation>
   </xs:annotation>
   <xs:restriction base="xs:string">
    <xs:enumeration value="Email"/>
    <xs:enumeration value="TelefonKomorkowy"/>
    <xs:enumeration value="TelefonStacjonarny"/>
   </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="TRodzajeAdresow">
   <xs:annotation>
    <xs:documentation>Typ wyliczeniowy dla rodzajów adresów</xs:documentation>
   </xs:annotation>
   <xs:restriction base="xs:string">
    <xs:enumeration value="Korespondencyjny"/>
    <xs:enumeration value="NaFakture"/>
   </xs:restriction>
  </xs:simpleType>
 </xs:schema>
```

1. Omówić z prowadzącym kurs podstawowe elementy tego dokumentu.
2. W folderze Databases wybrać bazę danych KursXMLPrzyklady.
3. Przejść do edytora zapytań naciskając przycisk New Query.
4. Zapisać na swoim komputerze przykładowy plik DziennikOcen.xsd – prowadzący wskaże lokalizację której należy ten plik pobrać.
5. Otworzyć przykładowy plik w przeglądarce internetowej – zapoznać się z jego strukturą – przedyskutować wątpliwości z prowadzącym kurs.
6. Przejść do edytora zapytań naciskając przycisk New Query
7. W edytorze zapytań napisać następujący skrypt:

   Tu wpisać lokalizację przykładowego pliku

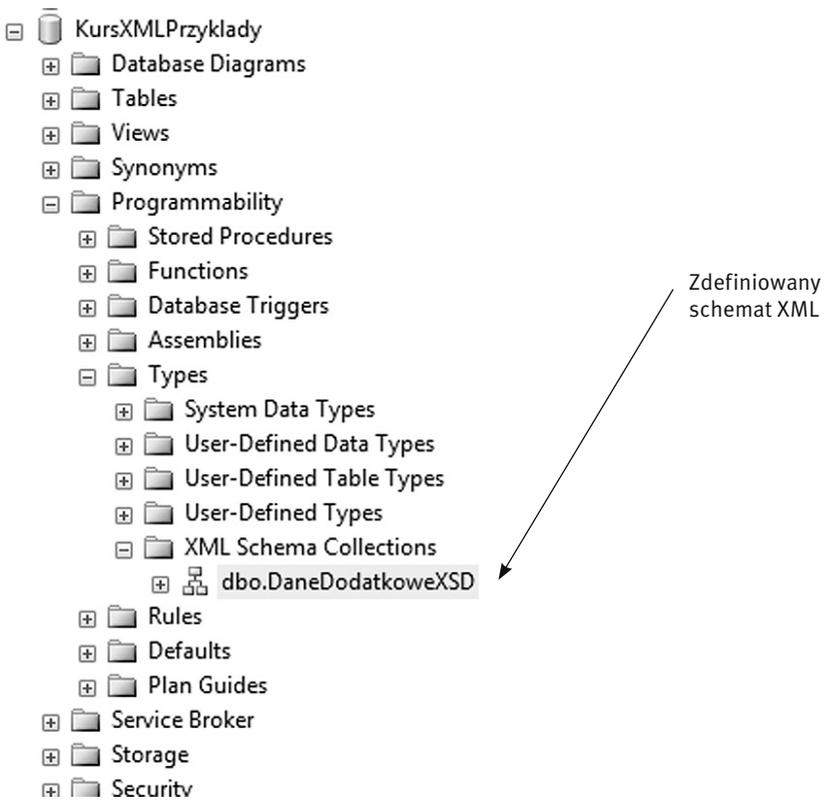
   ```
   DECLARE @dane XML
   SELECT @dane=K
   FROM OPENROWSET(BULK N'D:\DaneDodatkowe.xsd', SINGLE_BLOB) AS T(k)
   CREATE XML SCHEMA COLLECTION DaneDodatkoweXSD
   As
   @dane
   ```
8. Wykonać skrypt poprzez naciśnięcie klawisza F5.
9. W wyniku wykonania tego polecenia w bazie danych zapisany zostanie obiekt o nazwie DaneDodatkowe-XSD, który od tego momentu będzie można wykorzystywać do walidowania danych typu XML. Na rysunku 9 przedstawiono lokalizację definicji schematu XML w oknie Object Explorer:

7.2 ĆWICZENIE 12 – WYKORZYSTANIE ZDEFINIOWANEGO W BAZIE DANYCH SCHEMATU XSD DO WALIDACJI DOKUMENTU XML.

1. W folderze Databases wybrać bazę danych KursXMLPrzyklady.
2. Przejść do edytora zapytań naciskając przycisk New Query.
3. W edytorze napisać następujący skrypt:

   ```
   DECLARE @dane XML(DaneDodatkoweXSD)
   SET @dane='<DaneDodatkowe>
    <Adresy>
     <Adres>
      <RodzajAdresu>AdresKorespondencyjny</RodzajAdresu>
      <Miasto>Opole</Miasto>
   ```

```
    ‹KodPocztowy›04-765‹/KodPocztowy›
    ‹UlicaINumer›ul.Nowa 13‹/UlicaINumer›
   ‹/Adres›
   ‹Adres›
    ‹RodzajAdresu›AdresKorespondencyjny‹/RodzajAdresu›
    ‹Miasto›Opole‹/Miasto›
    ‹KodPocztowy›04-765‹/KodPocztowy›
    ‹UlicaINumer›ul.Nowa 13‹/UlicaINumer›
   ‹/Adres›
  ‹/Adresy›
  ‹Kontakty›
   ‹Kontakt›
    ‹RodzajKontaktu›Email‹/RodzajKontaktu›
    ‹Numer›KapitanNemo@wp.pl‹/Numer›
   ‹/Kontakt›
   ‹Kontakt›
    ‹RodzajKontaktu›Email‹/RodzajKontaktu›
    ‹Numer›81 8863412‹/Numer›
    ‹Uwagi›wew.123‹/Uwagi›
   ‹/Kontakt›
  ‹/Kontakty›
‹/DaneDodatkowe›'
```



Rysunek 9.
Lokalizacji definicji kolekcji schematów XML

4. Wykonać skrypt naciskając klawisz F5. Skrypt powinien wykonać się poprawnie
5. Dokonać modyfikacji dokumenty XML i wykonać skrypt.
6. Przećwiczyć i omówić z prowadzącym różne przypadki zmian i błędów generowanych przez walida tor XSD .

Przykładowo, jeżeli zmienimy RodzajKontaktu Email na E-mail powinien pojawić się następujący komunikat walidatora;

XML Validation: Invalid simple type value: ‚E-mail'. Location: /*:DaneDodatkowe[1]/*:Kontakty[1]/*:Kontakt[2]/*:RodzajKontaktu[1]

8 PRZYKŁADY WYKORZYSTANIA XML W PROCEDURACH I WYZWALACZACH

8.1 ĆWICZENIE 13 – WYKONANIE PROCEDURY SKŁADOWANEJ PRZYJMUJĄCEJ JAKO PARAMETR DANE TYPU XML

W ramach tego ćwiczenia chcemy napisać procedurę składowaną, która zwraca listę uczniów wraz z ich średnia ocen – dla tych uczniów, których numery pesel zostały przekazane jako parametr procedury.

1. W folderze Databases wybrać bazę danych ElektronicznyDziennikOcen.
2. Przejść do edytora zapytań naciskając przycisk New Query.
3. W edytorze napisać następujące polecenie:

```
CREATE PROCEDURE SrednieListyUczniow
@ListaPeseli XML
AS
SELECT Nazwisko,
       Imie,
       Pesel,
       AVG(Ocena) as Srednia
FROM Uczniowie JOIN Oceny
   ON Uczniowie.iducznia=Oceny.iducznia
WHERE Pesel IN (
                  SELECT k.value('text()[1]','varchar(10)')
                  FROM @ListaPeseli.nodes('ListaPeseli/Pesel') t(k)
                )
 GROUP BY Nazwisko, Imie,Pesel
```

4. Wykonać polecenie naciskając klawisz F5. Wykonanie tego polecenia spowoduje zdefiniowanie w bazie danych procedury składowanej o nazwie **SrednieListyUczniow.**
5. W celu przetestowania procedury w edytorze zapytań piszmy nastepujace polecenie:

```
EXEC SrednieListyUczniow ' ‹ListaPeseli›
                           ‹Pesel›92031275446‹/Pesel›
                           ‹Pesel›92060288788‹/Pesel›
                           ‹Pesel›92051577646‹/Pesel›
                         ‹/ListaPeseli›'
```

6. Wykonanie procedury powinno zwrócić następujący wynik :

Nazwisko	Imie	Pesel	Srednia
Kotek	Kasia	92031275446	3.000000
Krówka	Rysio	92051577646	2.781818
Kurka	Jola	92060288788	3.030769

7. Omówić z prowadzącym kod procedury i znaczenie takiego rozwiązana.

8.2 ĆWICZENIE 14 – WYKONANIE WYZWALACZA DDL Z WYKORZYSTANIEM FUNKCJI EVENTDATA() ZWRACAJĄCEJ DOKUMENT XML

W ramach ćwiczenia zdefiniowany zostanie wyzwalacz typu DDL, który ma uniemożliwić zdefiniowanie widoku jeżeli jego nazwa nie będzie zaczynała się od litery V.

1. W folderze Databases wybrać bazę danych ElektronicznyDziennikOcen.
2. Przejść do edytora zapytań naciskając przycisk New Query.
3. W edytorze napisać następujące polecenie:

```
CREATE TRIGGER TR_NazwyWidokow
ON DATABASE
```

```
AFTER Create_View
AS
DECLARE @dane XML=EVENTDATA()
IF @dane.value('EVENT_INSTANCE[1]/ObjectName[1]', 'varchar(128)') NOT LIKE 'V%'
BEGIN
PRINT 'Nazwa widoku musi zaczyna sie od litery V'
ROLLBACK
END
```

4. Wykonać polecenie naciskając klawisz F5. Efektem tego polecenia będzie zdefiniowanie w bazie danych wyzwalacza, który powinien anulować polecenie definicji widoku, jeżeli jego nazwa nie będzie zaczynała się od litery V.
5. W edytorze zapytań napisać następujące polecenie:

```
CREATE VIEW NoweKlasy
AS
SELECT Nazwa AS NazwaKlasy,
       RokSzzkolny
FROM Klasy
```

6. Wykonać polecenie naciskając klawisz F5. Efektem tego polecenia powinien być następujący komunikat:

```
Nazwa widoku musi zaczynać sie od litery V
Msg 3609, Level 16, State 2, Procedure NoweKlasy, Line 3
The transaction ended in the trigger. The batch has been aborted.
```

7. Omówić z prowadzącym kurs kod wyzwalacza i znaczenie takiego rozwiązania.

8.3 ĆWICZENIE 15 – WYKONANIE WYZWALACZA DML ZAPISUJĄCEGO ZMIANY W WYBRANEJ TABELI W POSTACI DOKUMENTU XML.

W ramach tego ćwiczenia zdefiniowany zostanie wyzwalacz dla tabeli Klasy, który będzie rejestrował zmiany danych tej tabeli wraz z informacją dodatkową identyfikującą parametry modyfikacji.

1. W folderze Databases wybrać bazę danych ElektronicznyDziennikOcen.
2. W bazie danych znajduje się tabela SledzenieZmian o strukturze przedstawionej na rysunku 10.

Column Name	Data Type	Allow Nulls
id	int	☐
kto	varchar(128)	☐
komputer	varchar(128)	☐
kiedy	datetime2(7)	☐
tabela	varchar(128)	☐
operacja	char(1)	☐
zmiany	xml	☐

Rysunek 10.
Struktura tabeli SledzenieZmian.

3. Przejść do edytora zapytań naciskając przycisk New Query.
4. W edytorze napisać następujące polecenie:

```
CREATE TRIGGER TR_SledzenieZmian
ON Klasy
AFTER INSERT, UPDATE, DELETE
AS
INSERT INTO SledzenieZmian(kto,komputer,kiedy,tabela,operacja,zmiany)
SELECT system_user,
       host_name(),
       sysdatetime(),
       'Klasy',
            CASE
             WHEN NOT EXISTS  (SELECT * FROM  inserted)
              AND NOT EXISTS (SELECT  * FROM deleted) THEN 'P'
```

```
            WHEN  EXISTS  (SELECT * FROM  inserted)
             AND NOT EXISTS (SELECT  * FROM deleted) THEN 'I'
             WHEN NOT EXISTS  (SELECT * FROM  inserted)
             AND  EXISTS (SELECT  * FROM deleted) THEN 'D'
            ELSE 'U'
           END,
      (SELECT coalesce(i.idklasy,d.idklasy) AS Idklasy,
          CASE
            WHEN  i.rok_szkolny=d.rok_szkolny THEN  null
            ELSE coalesce(d.rok_szkolny,i.rok_szkolny)
          END AS RokSzkolny,
          CASE
            WHEN  i.nazwa=d.nazwa THEN null
           ELSE coalesce(d.nazwa,i.nazwa)
          END AS NazwaKlasy
       FROM  inserted AS I FULL JOIN deleted  AS D
        ON i.idklasy=d.idklasy
       FOR XML RAW('Wiersz'),ELEMENTS, ROOT('Zmiany') ,TYPE )
```

5. Wykonać polecenie naciskając klawisz F5.
6. Omówić z prowadzącym kurs kod wyzwalacza oraz znaczenie takiego rozwiązania.
7. W edytorze zapytań napisać następujące polecenie polecenie;
   UPDATE **Klasy** SET
   RokSzkolny='2009/2010'
   WHERE **idklasy**›2
8. Po wykonaniu polecenia sprawdzamy zawartość tabeli SledzenieZmian – w tabeli pojawił się wiersz opisujący wykonana operację:

id	kto	komputer	kiedy	tabela	operacja	zmiany
3	ANDRZEJ\hp	ANDRZEJ	2010-06-14 03:49:47.1223228	Klasy	U	<Zmiany><Wiersz><Idklasy>5</Idklasy><RokSzkolny>...

9. Dokument XML zapisany w kolumnie Zmiany ma następująca postać:

```
‹Zmiany›
 ‹Wiersz›
  ‹Idklasy›5‹/Idklasy›
  ‹RokSzkolny›2008/2010‹/RokSzkolny›
 ‹/Wiersz›
 ‹Wiersz›
  ‹Idklasy›3‹/Idklasy›
  ‹RokSzkolny›2008/2010‹/RokSzkolny›
 ‹/Wiersz›
‹/Zmiany›
```

10. Wykonać inne polecenia zmieniające dane w tabeli i sprawdzic zawartość tabeli SledzenieZmian.
11. Omówić z prowadzącym kurs otrzymywane wyniki.

LITERATURA

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

W projekcie **Informatyka +**, poza wykładami i warsztatami, przewidziano następujące działania:

- 24-godzinne kursy dla uczniów w ramach modułów tematycznych
- 24-godzinne kursy metodyczne dla nauczycieli, przygotowujące do pracy z uczniem zdolnym
- nagrania 60 wykładów informatycznych, prowadzonych przez wybitnych specjalistów i nauczycieli akademickich
- konkursy dla uczniów, trzy w ciągu roku
- udział uczniów w pracach kół naukowych
- udział uczniów w konferencjach naukowych
- obozy wypoczynkowo-naukowe.

Szczegółowe informacje znajdują się na stronie projektu

www.informatykaplus.edu.pl