Zrozumienie XML. Podejście do opracowywania oprogramowania

1. Wstęp

Zarządzanie, przetwarzanie, wymiana, manipulowanie i prezentowanie danych to niektóre z kluczowych działań w kontekście technologii informacyjnej i informatyki. Rzeczywiście, w pierwszym starciu z obliczaniem, niezależnie od tego, jakie jest specjalne pole, zwykle wprowadzamy pojęcie "input, process, output", a następnie powiedziano nam, że dane wejściowe mogą mieć dowolną postać "danych". Następnie zapoznajemy się z ważnością danych, co to jest, dlaczego jest tak ważna i jak należy ją traktować. Jeśli jesteś w informatyce lub inżynierii oprogramowania i takich dziedzinach, to dowiedziałeś się również o strukturze danych i bazach danych. Niektórzy z was mogli nawet usłyszeć o "dużych danych", "hurtowni danych" i "eksploracji danych". Ma to jedynie przypominać, jak daleko posunąć się może koncepcja manipulacji danymi i danymi. Od początku przetwarzania danych jako nauki manipulacja danymi i danymi przebiegała na różnych etapach. Przedstawiono wiele technologii, z których niektóre trwają krótko, a niektóre są na przykład od lat 70. XX wieku. Aby to zilustrować, relacyjne koncepcje danych i relacyjne bazy danych rozpoczęły się w latach 70. XX wieku i nadal należą do dominujących, zarówno w przemyśle, jak i na uczelniach wyższych. Koncepcja nawet teoretyzowana w formacie Relational Calculus. Poszukiwanie odpowiednich sposobów gromadzenia, zachowywania, manipulowania i prezentowania danych było ciągłą podróżą w dziedzinie informatyki. Jednak w trakcie tego żądania było kilka przypadków niektóre długotrwałe owoce. XML jest prawdopodobnie jednym z nich. Termin XML jest tak ważny, że jestem pewien, że większość z was, którzy zaczęli czytać tę książkę, albo dlatego, że często o niej słyszeliście, albo powinniście ją używać w jednym z waszych projektów / zadań, albo doradzono wam naukę i dowiedz się, jak możesz go zastosować. Ale dlaczego XML jest tak ważny, że stał się de facto standardem komunikacji danych w branży oprogramowania? Pomysł pochodzi z faktu, że pomimo naszego pierwszego zrozumienia danych jako uporządkowanej, wstępnie zdefiniowanej, ściśle sformatowanej rzeczy, nie dotyczy to większości obiektów na świecie. Oznacza to, że dane wokół nas, przez większość czasu dla zaskoczenia, nie mogą być łatwo łączone w sztywne formaty, które zapewniają zwykłe bazy danych, takie jak relacyjne bazy danych. Koncepcja niestrukturalnych lub semistrukturalnych danych jest znana od dawna. Niezależnie od struktury i formatu danych, rosnące zapotrzebowanie na wymianę danych między firmami z jednej strony, a korzystanie z Internetu z drugiej strony, zachęciło naukowców, naukowców i technologów do ulepszenia technologii gromadzenia, wymiany, prezentacja i manipulacja danymi. W poniższej sekcji przedstawiono charakterystykę półstrukturalnych i niestrukturalnych danych za pomocą kilku przykładów.

1.1 Struktura danych

Kiedy myślimy o danych, zwykle myślimy o danych, które mają wstępnie zdefiniowaną strukturę. Być może większość z nas zna już dane relacyjne i relacyjną bazę danych. Mogliśmy również wykorzystać struktury jako format uporządkowanych danych i zastosowaliśmy ten pomysł w programowaniu i tworzeniu oprogramowania. W systemach baz danych powiedziano nam, aby zaprojektował nasz model danych, który zapewnia solidną i głęboką bazę docelowego systemu. W większości aplikacji baz danych stały model danych. Chociaż istnieją różne sposoby, aby uczynić system elastycznym, takim jak architektura ANSI, trójwarstwowa architektura programowania, MVC, koncepcja "widoków", dynamiczna modyfikacja schematów i tak dalej, ale podstawowy system jest solidny. Jednak w wielu przypadkach struktura danych zmienia się częściej. Poza tym sposób, w jaki zmienia się struktura danych, nie jest łatwy do przewidzenia. Ta sytuacja prowadzi do koncepcji danych niestrukturalnych i semistrukturalnych. Spójrzmy na kilka przykładów. Być może wcześniej widziałeś różne wersje tych przykładów.

Przykład 1:

Adres może składać się z różnych elementów, które z kolei tworzą różne architektury, jak poniżej:

• Kod pocztowy, miasto, kraj

• Numer domu, ulica, miasto, województwo, kraj

• Numer mieszkania, numer domu, aleja, ulica, miasto, województwo, kraj

• ...

Przykład 2:

Książka może mieć kilka architektur, jak poniżej:

• Spis treści, wstęp, rozdziały

• Spis treści, Przedmowa, Rozdziały, Ćwiczenia, Bibliografia, Indeks

• Spis treści, Przedmowa, Rozdział (sekcja, ...), Rozdział Odsyłacz, Ćwiczenia, Słownik, Indeks

• ...

Przykład 3:

Wpis odniesienia (lub element biblioteki) ma różne formaty cytowania i zawiera różne dane w zależności od typu:

• Książka: autorzy, rok, tytuł, wydawca, miejsce, strona (y), do której odnosi się strona.

• Artykuł: Autorzy, rok, tytuł, dziennik, tom, strony z odniesieniami, wydawca, miejsce.

• Film: tytuł, rok, reżyser, kraj, firma produkcyjna.

• ...

Przykład 4:

Zdanie angielskie może mieć następujące struktury:

• Temat, czasownik

• Temat, czasownik, przedmiot

• Temat, czasownik, przymiotnik

• ...

Przykład 5:

Dokument może być:

• Faktura.

• Artykuł.

• Instrukcja użytkownika.

• Książka.

• ...

Przyjrzyjmy się uważnie tym przykładom, ujawniając między nimi kilka typowych tematów:

• Są elastyczne (nie sztywne) pod względem struktury.

• Można je przedstawić za pomocą wykresów.

• Mają strukturę przypominającą drzewo.

Dane z powyższych przykładów nazywane są danymi "semistrukturalnymi" w celu odróżnienia ich od danych strukturalnych. Być może zajmowałeś się powyższymi przykładami, szczególnie przykładami 1 i 3, w którym to przypadku najprawdopodobniej użyłeś modelu relacyjnego, a następnie relacyjnej bazy danych do obsługi danych. Chociaż twoje podejście jest całkowicie poprawne i akceptowalne, nie jest to jedyny sposób na uporządkowanie tych danych. Na przykład, być może odkryłeś, że jest to skomplikowane, gdy stworzyłeś relacyjną bazę danych dla biblioteki. Poza tym, jeśli chcesz udostępnić swoje dane innej bibliotece, musisz wiedzieć o projekcie bazy danych, lub musisz wyodrębnić swoje dane i powiedzieć, jak mogą je "zinterpretować". Można argumentować, że w tej sytuacji może pomóc relacyjno-relacyjne bazy danych lub obiektowe modelowanie danych. Nie chcę zagłębiać się w szczegóły i dryfować głównej ścieżki książki, ale mogę powiedzieć, że niestety nawet tego rodzaju modele danych nie pomogłyby przy "semistrukturach", a ich złożoność nadal stanowiłaby problem. Dane mogą być również "nieustrukturyzowane". W rzeczywistości filmy, wiadomości e-mail i zwykłe teksty są traktowane jako dane niestrukturalne. Ale w tej książce skupiamy się na półstrukturalnych danych, ponieważ XML jest bardzo potężną technologią do obsługi tego typu danych. Nie oznacza to, że XML nie jest w stanie obsłużyć danych strukturalnych lub niestrukturalnych. W rzeczywistości język XML jako język znaczników został zaprojektowany do obsługi wszystkich typów danych. Powodem wyłączenia danych strukturalnych jest to, że w przypadku tego typu danych istnieją inne zaawansowane technologie i modele, które działają prawidłowo. Jeśli chodzi o dane nieustrukturyzowane, pojęcie to należy do innej kategorii manipulacji danymi, co wykracza poza zakres tej książki, jednak ten rodzaj danych zostałby lekko omówiony również w odpowiednich sekcjach.

1.2 Definicja danych i wymiana danych

Dane powinny być modelowane, aby mogły być właściwie obsługiwane. Modelowanie danych to faza inżynierii oprogramowania, podczas której modeluje się sposób gromadzenia, rozpowszechniania, aktualizacji i prawdopodobnie wycofywania danych. Istnieją różne metody, techniki, diagramy i podejścia stosowane w tym procesie. W zależności od sytuacji ważne jest, aby dane mogły być łatwo udostępniane różnym użytkownikom. W większości podejść do manipulowania danymi definicja danych jest odrębną częścią modelu, który definiuje struktura danych. Czasami ta część nosi nazwę "meta danych", co oznacza "dane o danych", czyli dane, za pomocą których opisujemy dane. Definicja danych pozwala programistom umieścić solidną podstawę, która definiuje dane i zachowuje ich integralność. Każda zmiana definicji danych będzie miała konsekwencje dla systemu. Dlatego modelowanie danych jest kluczową fazą inżynierii oprogramowania. Aby dokonać analogii, można ją porównać do fundamentu budynku. Jeśli pójdzie źle, wiele innych rzeczy prawdopodobnie pójdzie źle. To zmusiło informatyków i naukowców do poszukiwania pewnych metod i technologii do budowy odpowiednich narzędzi, aby nieuniknione zmiany były możliwe do zrealizowania. Jednak udostępnianie i dostępność danych w różnych środowiskach nadal pozostaje problemem. Powodem jest to, że zwykle narzędzia te są używane w zintegrowanym środowisku, co widać w systemach zarządzania bazami danych (DBMS), na przykład Oracle, MS SQL Server i MySQL. Chociaż te narzędzia

i systemy zapewniają kilka sposobów migracji danych, eksportu i importu danych, wciąż wymagają dużego nakładu pracy i specjalności, aby obsłużyć żądania wymiany danych między różnymi środowiskami. Ale XML zapewnia elastyczną, a jednocześnie łatwą do zrozumienia metodę obsługi danych. Główne cechy tej metody można podsumować poniżej:

• Pozwala na "samoopisanie" danych. Oznacza to, że dane i ich opis są ze sobą powiązane.

• Dane można budować za pomocą prostego edytora tekstu.

• Narzędzia programistyczne mogą korzystać z podstawowych operacji wprowadzania i operacji na plikach w celu odczytu danych.

• Narzędzia programistyczne mogą wykorzystywać podstawowe operacje wyjściowe i operacje na plikach w celu zapisywania i aktualizowania danych.

• Dane mogą być udostępniane niezależnie od platformy i środowiska. Nazywa się to niezależnością od platformy.

• Dane można przeglądać za pomocą większości przeglądarek internetowych lub prostych edytorów tekstu.

• Można je odczytać zarówno przez komputery (tj. Komputery), jak i przez ludzi.

1.3 XML i tworzenie oprogramowania

Rosnąca popularność XML i jego elastyczność sprawiły, że prawie wszyscy ważni dostawcy DBMS włączyli technologię XML do swoich produktów, w taki czy inny sposób. Zobaczysz kilka przykładów narzędzi i udogodnień oferowanych przez tych dostawców w całym tekście. Podobnie, większość dostawców w dziedzinie tworzenia oprogramowania opracowało gotowe narzędzia do obsługi manipulacji danymi XML. Na przykład, PHP i Microsoft Visual Studio dostarczyły czytniki i programy piszące XML, które mogą ułatwić obsługę danych XML w systemach, które tworzysz. XML jest rozwijany i utrzymywany przez W3C (World Wide Web Consortium). W3C zaleca standardy, których należy przestrzegać podczas użytkowania XML. Dzięki temu XML jest niezależną od platformy technologią kodowania i formatowania dokumentów. Co więcej, główni gracze w branży komputerowej przyczyniają się do rozwoju XML. Podsumowując, XML jest technologią, która nadal będzie odgrywać dużą rolę w informatyce w nadchodzących latach. Niektórzy eksperci twierdzą nawet, że byłoby to dominujące narzędzie w wymianie danych i manipulacji. Dlatego uczenie się i zdobywanie doświadczenia w zakresie XML jest niezbędne każdemu, kto pracuje w branży komputerowej. Poniższe rozdziały mają na celu wypełnienie części tej konieczności. W tym celu próba ćwiczenia na końcu każdej części i próba zastosowania XML w różnych projektach jest kluczowym czynnikiem, aby skutecznie uczyć się i stać się ekspertem w tej prostej, pięknej, a jednocześnie niesamowicie wydajnej technologii

1.4 Alternatywne metody wymiany danych

XML nie jest ani pierwszą metodą wymiany danych, ani ostatnią. Koncepcja wymiany danych, w nowoczesnym sensie, sięga lat czterdziestych. W związku z tym wzrost wymogów w zakresie elektronicznej wymiany danych doprowadził do pojawienia się bardziej wyrafinowanych metod w latach siedemdziesiątych i osiemdziesiątych. EDI (Electronic Data Interchange) została zasugerowana w 1996 roku i wkrótce stała się międzynarodowym standardem wymiany danych. EDI jest szeroko stosowaną metodą wymiany danych między partnerami biznesowymi na całym świecie. Ma wiele zalet i zalet, takich jak bezpieczeństwo i prostota. Więc jedno pytanie może być, dlaczego potrzebujemy innej metody? Odpowiedź jest poza celem tej książki, ale żeby nie pozostawić cię bez odpowiedzi, porównaj ją z różnymi językami programowania. Chociaż, z naukowego punktu widzenia, możemy

zaprogramować komputer, aby rozwiązać każdy rozwiązywany obliczeniowo problem z jednym z kilkuset (w rzeczywistości, tysięcy!) Języków programowania, niektóre z nich mogą być bardziej wydajne, szybsze i łatwiejsze w użyciu dla określonych rodzajów problemów a niektóre za inne problemy. Jednak widzieliśmy, że niektóre z języków zastąpiły pozostałe. Czy EDI może zastąpić XML? Nie chcę przewidzieć. Wolę widzieć zarówno współżycie, jak i więcej opcji dla użytkowników. Ponownie, w czasie, gdy zaczęło pojawiać się EDI i podczas gdy XML wchodziło w grę, na rynku wymiany danych pojawiła się inna technologia: JSON (JavaScript Object Notation). Jest to otwarty standard wymiany danych, który stał się bardzo popularny, szczególnie w związku ze wzrostem mobilne przetwarzanie danych i komunikacja. Używa formatu tekstowego i jest niezależny od języka. Więcej informacji można znaleźć na stronie http://json.org/. Jednak dla tych, którzy już znają JSON, mogą pojawić się pewne pytania. Czy powinniśmy używać JSON zamiast XML? Czy JSON zastąpi XML? Ponownie, z punktu widzenia użytkowników, przydatność do użycia jest głównym czynnikiem. Dla wielu celów JSON może być bardzo przydatnym wyborem do wymiany danych. Ale nie powinieneś ograniczać się tylko do jednego narzędzia. Deweloper powinien korzystać z narzędzi zgodnie ze środowiskiem i problemami, które są w zasięgu ręki. Powinieneś wiedzieć, że JSON, przynajmniej obecnie, nie jest kontenerem danych i nie posiada różnorodnych technologii wspomagających, które XML zapewnia do różnych celów poza wymianą danych. Nie ma bazy danych. Jednak, jak wspomniano, w wielu przypadkach, szczególnie w aplikacjach mobilnych, jest to bardzo dobry wybór jako lekkie narzędzie do wymiany danych. JSON szybko się rozwija, choć wydaje się, że nie ma zamiaru zastępować XML. Chociaż walka o rynek zawsze trwa, jeśli JSON chce zastąpić XML, powinien rozwinąć wszystkie powiązane technologie, które zapewnia XML, w którym to przypadku nie jest już "lekkim" JSON.

2 Czym jest XML?

XML oznacza eXtensible Markup Language. Oznacza to, że jest to zasadniczo język znaczników. Ale czym jest język znaczników?

Język znaczników

- Markup Language to system do oznaczania lub "oznaczania" dokumentu w sposób pokazujący logiczną obecność dokumentu.

- Zawiera instrukcje dotyczące sposobu wyświetlania dokumentu w formie elektronicznej.

- Innymi słowy, jest to zbiór instrukcji, który jest wprowadzany w formacie "tagów", które zostałyby włączone do tekstu dokumentu i pozwala dowolnemu medium, które jest w stanie zinterpretować tagi, aby pokazać dokument w sposób, w jaki projektanci zamierzali go przedstawić.

- HTML (Hyper Text Markup Language) to najbardziej znany język znaczników. Jest to format, na podstawie którego prezentowana jest większość internetowych dokumentów tekstowych.

- HTML ewoluował przez lata. W rezultacie powstało kilka wersji, z których wszystkie współistnieją obecnie razem, np. HTML, DHTML i HTML5.

- Głównym celem języka znaczników jest umożliwienie wyświetlania różnych dokumentów użytkownikom w ten sam sposób, niezależnie od używanego medium. W związku z tym podczas przeglądania strony na komputerze stacjonarnym, laptopie lub smartfonie (w przeglądarce Firefox, IE, Safari, Chrome, Opera lub dowolnej innej przeglądarce) widać to samo.

- Pomimo faktu, że XML jest językiem znaczników, jego głównym celem nie jest renderowanie (pokazywanie) danych. Jest to jedna z różnic w porównaniu z innymi językami znaczników, takimi jak HTML.

- Zamiast tego głównym celem XML jest opisywanie danych. Opisywanie danych oznacza, że XML poprzez swoje znaczniki pozwala czytnikowi XML zrozumieć, w jaki sposób dane zostały podzielone na różne części i co zawiera każda część

Większość wcześniejszych języków znaczników używała zestawu znaczników zdefiniowanych, podczas gdy w XML definiujecie własne znaczniki. Dlatego nazywa się eXtensible. W ostatnich latach niektóre z języków znaczników, w szczególności HTML, zostały przekształcone i zrekonfigurowane, co zwiększyło ich możliwości umożliwienia użytkownikom definiowania nowych znaczników. Ale, co ciekawe, zostały one przekształcone w oparciu o XML!

<sampleDic>

<word>

<head> ethic </head>

<definition>

rules of behavior based on ideas about what is morally good and

bad.

</definition>

</word>

</sampleDic>

Możesz łatwo zrozumieć (lub przynajmniej domyślać się), że jest to dokument zawierający słowo ("etyka") i jego definicję ("reguły zachowania oparte na ideach dotyczących tego, co jest dobre i złe moralnie"). To właśnie ma oznaczać, że dokument XML jest samoopisowy. Dowiesz się o strukturze i składni powyższych danych XML w dalszej części tej części

2.1 Zalety XML

XML jest uważany za niezwykle użyteczną technologię komputerową, ponieważ został wynaleziony pod koniec ubiegłego wieku. W odpowiedzi na pytanie "jakie są zalety tej technologii?", Można uzyskać długą listę korzyści z używania XML. Aplikacja XML stale rośnie od czasu jej utworzenia. Duża ilość dokumentów jest tworzona i przechowywana w formacie XML. Dokumenty te mogą być wymieniane i udostępniane w Internecie lub wykorzystywane do celów wewnętrznych w przedsiębiorstwach i organizacjach. Ale czym jest XML? Zanim zaczniemy mówić o XML, ważne jest, aby zrozumieć pewne pojęcia, które mogą nam pomóc w realizacji konieczności czegoś takiego jak XML. Poniżej znajduje się podsumowanie korzyści XML.

Zalety XML

- XML jest samoopisowy. Oznacza to, że XML, w zasadzie, nie potrzebuje żadnych dodatkowych dokumentów do jego interpretacji.

- XML jest zarówno czytelny dla komputera, jak i dla człowieka. W przeciwieństwie do wielu zakodowanych dokumentów, dokument XML jest czytelny dla ludzi, a także może być sprawnie przetwarzany przez komputery.

- Łatwe do skonstruowania (komponowania). Zbudowanie dokumentu XML nie wymaga specjalnego narzędzia. Do tego celu można użyć dowolnego edytora trywialnego (np. TextEdit, Notepad i gedit).

- Dokument XML może być łatwo udostępniany różnym użytkownikom.

- Dane XML można łatwo transportować. Nie ma potrzeby zgodności aplikacji źródłowej i docelowej.

- XML jest niezależny od platformy. Bez względu na to, jakiego środowiska operacyjnego używasz (opartego na własnościach, na przykład Windows / Mac lub open source, np. Ubuntu), XML może być zrozumiany w ten sam sposób.

- XML może być używany jako kontener danych, w którym to przypadku nie potrzebujesz żadnego specjalnego DBMS do zarządzania twoimi danymi.

- Jest obsługiwany i standaryzowany przez W3C.

- Obsługuje prawie wszystkie języki ludzkie dzięki obsłudze Unicode

Jako przykład, który będziemy śledzić w tej książce, rozważmy dane słownikowe. Wybrałem tę sprawę z dwóch powodów. Po pierwsze, jego dane mogą być bardzo proste i mogą stać się bardziej złożone w oparciu o nasz wybór i zastosowanie. Po drugie, jego dane spełniają jeden z wcześniej wymienionych warunków, które były danymi przeznaczonymi głównie do odczytu i nieaktualizowania, stąd też właściwy kandydat do używania XML jako kontenera danych.

2.2 Elementy XML

XML składa się z elementów. Element XML jest podstawową strukturą dokumentów XML. Dlatego dokument XML można uznać za las elementów.

Element XML

- Element XML to nazwa podana konkretnej części dokumentu XML.

- Musi być przedstawiony w następujący sposób:

o <elementname> dowolny tekst </ elementname>

o <elementname> nazywa się tagiem startowym.

o </ elementname> nazywa się tagiem końcowym.

o "<elementname>" rozpoczyna definicję elementu, a "</ elementname>" kończy definicję elementu.

- Nazwa elementu:

o Może składać się ze znaków alfanumerycznych i innych.

o Nie można rozpocząć od liczby lub znaku interpunkcyjnego (np.?%%, ").

o Nie ma spacji między znakami.

o Nie może rozpoczynać się od żadnych liter XML (xml, xMl, itp.).

- Przykład:

o Valid XML element: <word> Ethic </ word>

o Nieprawidłowy element XML: <1word> dowolny tekst </ 1word>

2.2.1 Przewodnik po nazwach elementów

Prostą radą nazywania jest podążanie za samoopisowym pojęciem XML. Przypisywanie nazw do elementów opartych na najlepszych praktykach nazewnictwa w komputerach. Postępuj zgodnie z jedną konwencją w całym dokumencie. XML nie jest ogólnie przydatny do kodowania i szyfrowania. Dlatego unikaj tajemniczego podejścia, chyba że jest to naprawdę konieczne (jeśli w ogóle jest!). Ogólne wytyczne dotyczące programowania mają również zastosowanie tutaj, choć z dużym naciskiem na zrozumiałość i samoopisowość.

2.3 Atrybuty XML

Atrybuty są parami nazwa-wartość, które zawierają opisowe informacje o elemencie. Atrybut umieszczany jest wewnątrz znacznika początkowego po odpowiedniej nazwie elementu. Wartość atrybutu musi być ujęta w cudzysłów. Atrybut może wystąpić tylko raz w tagu, podczas gdy elementy zagnieżdżone (które czasami są nazywane podelementami) w obrębie tego samego znacznika mogą być powtarzane. Rysunek 3 pokazuje poprzedni przykład zmieniony za pomocą atrybutu. W tym przykładzie do słowa "etyka" dodano nowy element "definicja", aby pokazać wspomniany przypadek powtarzalności.

<sampleDic>

<word head = “ethic” >

<definition>

rules of behavior based on ideas about what is morally good and bad.

</definition>

<definition>

a set of moral principles, especially ones relating to or affirming a

specified group, field, or form of conduct.

</definition>

</word>

</sampleDic>

2.4 Deklaracja XML

Jest to opcjonalna instrukcja na początku dokumentu XML. Może mieć prosty format lub używać parametrów.

Deklaracja XML

- Format ogólny:

<? xml version = '1.0' encoding = 'kodowanie znaków' standalone = 'yes | no'?>

- Przykład 1: <? Xml version = '1.0'>

- Przykład 2: <? Xml version = '1.0' encoding = 'UTF-8'?>

- Przykład 3: <? Xml version = "1.0" encoding = 'ISO-8859-1??>

- Przykład 4: <? Xml version = '1.0' standalone = 'yes'?>

Nie 0 - Jeśli użyto deklaracji XML, musi ona pojawić się jako pierwsza linia dokumentu.

Nie 1 - Domyślnym kodowaniem w Internecie jest UTF-8, dlatego też przykłady 1 i 2 deklarują to samo.

Nie 1 - Kolejność parametrów deklaracji jest ważna. Oznacza to, że "samodzielny" musi pojawić się jako ostatni. Cel "samodzielny" zostanie wyjaśniony w następnym rozdziale

2.5 Komentarze XML

Komentarze w dokumentach XML są zawarte w <! i -> tagi.

Komentarze XML

<! To jest przykład komentarza XML. ->

2.6 Zamawianie XML

Jak już wspomniano, dokumenty XML mają strukturę podobną do drzewa. Mają root i elementy, które są uporządkowane. Kolejność zależy od dwóch czynników: celu dokumentu i sposobu jego przetwarzania. Aby to zilustrować, na rysunkach 4 i 5 pokazano dwa dokumenty XML, które nie są identyczne.

<sampleDic>

<słowo>

<definicja>

zasady postępowania oparte na ideach dotyczących tego, co moralnie dobre i złe.

</ definition>

<źródło>

Merriam-Webster, online

</ source>

<użycie>

Etyka jest jego wybranym kierunkiem studiów.

</ usage>

</ word>

</ sampleDic>

<sampleDic>

<słowo>

<źródło>

Merriam-Webster, online

</ source>

<definicja>

zasady postępowania oparte na ideach dotyczących tego, co moralnie dobre i złe.

</ definition>

<użycie>

Etyka jest jego wybranym kierunkiem studiów.

</ usage>

</ word>

</ sampleDic>

Jednak dla atrybutów kolejność nie ma znaczenia.

2.7 Dobrze sformułowane

Ważne jest, aby dostarczyć dokument XML, który jest strukturalnie "dobrze uformowany". Dobrze sformułowane, w kontekście XML, oznacza, że jego struktura jest poprawna pod względem składniowym. Innymi słowy, możemy powiedzieć, że dokument XML jest dobrze sformułowany tylko wtedy, gdy jego składnia jest poprawna. Aby architektura była poprawna pod względem składni, każdy element musi być dobrze sformułowany. Można to uznać za pełne wyjaśnienie, ale ponieważ powoduje to zamieszanie dla początkujących, starałem się wyjaśnić różnicę poprzez rozróżnienie między "poprawnością składniową" i "poprawnością strukturalną". Istnieje jeszcze jeden aspekt XML, który nazywa się validity, który zajmuje się składnią w bardziej rozsądny sposób, który zostanie wyjaśniony w następnej części.

XML dobrze sformułowane

- Dokument XML jest dobrze sformułowany, jeśli jego architektura jest poprawna pod względem składniowym.

- Architektura XML jest poprawna pod względem składni wtedy i tylko wtedy, gdy ma element główny, a architektura wszystkich jego elementów jest poprawna pod względem składniowym.

- Element główny musi zawierać wszystkie pozostałe elementy.

- Architektura elementu jest poprawna pod względem składni, jeśli:

o Jest zawarty w początkowej i końcowej parze tagów.

o Nazwy elementów są zgodne z zasadami nazewnictwa.

- Jeśli element nie ma żadnych danych, nazywa się go pustym elementem.

- Puste elementy mogą być wyświetlane na dwa sposoby:

o <elementname> </ elementname>

o <elementname />

2.7.1 Dobrze sformatowany XML

Rysunek 6 pokazuje dobrze sformułowany XML

<sampleDic>

<word>

<head> ethic </head>

<definition>

rules of behavior based on ideas about what is morally good and bad.

</definition>

<source>

Merriam-Webster, online

</source>

<usage>

Ethics is his chosen field of study.

</usage>

</word>

<word>

<head> moral </head>

<definition>

concerned with the principles of right and wrong behavior and the

goodness or badness of human character.

</definition>

<source>

The Internet, online

</source>

<usage>

the moral dimensions of medical intervention

</usage>

</word>

</sampleDic>

Rysunek 7 pokazuje XML, który nie jest dobrze uformowany.

<sampleDic>

<word>

<head> ethic </head>

<definition>

rules of behavior based on ideas about what is morally good and bad.

</definition>

<source>

Merriam-Webster, online

</source>

<usage>

Ethics is his chosen field of study.

</usage>

</sampleDic>

<word>

<head> moral </head>

<definition>

concerned with the principles of right and wrong behavior and the

goodness or badness of human character.

</definition>

<source>

The Internet, on-line

</source>

<usage>

the moral dimensions of medical intervention

</usage>

</word>

Są dwa problemy z dokumentem. Pierwszym problemem jest to, że nie ma elementu root. Drugą kwestią jest to, że element <słowo> jodły nie ma zamkniętego znacznika.

2.8 Wady XML

Wady XML w porównaniu do jego zalet nie są znaczne. Jednak podobnie jak inne narzędzia i technologie, niezależnie od ich użyteczności, można zidentyfikować pewne wady. Po pierwsze, renderowanie XML nadal nie jest czymś w rodzaju dokumentów HTML. Chociaż większość przeglądarek dostępnych na rynku może renderować dokument XML w dobrze skonstruowanym, drzewiastym kształcie, niektóre mogą tego nie robić poprawnie. Tak więc narzędzia i techniki wciąż rosną wokół

XML. Listę przeglądarek internetowych obsługujących dokumenty oparte na formacie XML można znaleźć na stronie http://www.xml.com/pub/rg/XML_Browsers. Jednak najlepszym sposobem zrozumienia poziomu wsparcia oferowanego przez każdą przeglądarkę jest przetestowanie dokumentu XML za pomocą przeglądarki docelowej. Ponadto należy pamiętać, że większość przeglądarek jest regularnie aktualizowana. W rezultacie ich funkcja obsługi XML może się odpowiednio zmienić. Po drugie, XML jest łatwy, jeśli używa się go na prostym poziomie. To staje się nieco skomplikowane, jeśli jest używane w pełni rozwinięty sposób. Inne technologie związane z XML, takie jak arkusz stylów XML, schemat XML, kwerenda XML i narzędzia do pisania i czytania XML w aplikacjach, mogą być dość długo zrozumiane. Last but not least, bezpieczeństwo XML nadal stanowi problem. Chociaż istnieją narzędzia i techniki, które pozwalają przezwyciężyć ten problem, to dodatkowo zwiększa złożoność technologii.

2.9 Kiedy używać XML?

XML może być używany do różnych celów. Zakres zastosowania zmienia się od celów naukowych / technicznych do zwykłych aplikacji biznesowych. Jednak XML nie powinien być traktowany jako ostateczna odpowiedź na wszystko. W niektórych przypadkach może działać bardzo dobrze, a słabo w innych. Poniżej znajduje się pewna reguła, w której można wybrać XML jako kontener danych

Kiedy wybrać XML jako kontener danych?

- XML działa dobrze z danymi o następujących cechach:

o Dane niezmienne - Dane niezmienione (np. słowniki).

o Przeznaczony do częstego czytania, ale rzadko aktualizowany (np. katalogi biblioteczne).

o Ma bardzo złożony format (np. pliki konfiguracyjne komputerów / aplikacji).

Wskazówki dotyczące korzystania z XML

- XML jest głównie technologią służącą do organizowania / standaryzacji wymiany danych

- Rozważ użycie XML jako kontenera danych, jeśli:

o Twoje dane są częściowo strukturalne lub nieustrukturyzowane

o Nie masz ciężkiej aktualizacji bazy danych

o Bezpieczeństwo nie jest głównym problemem

o Nie zajmujesz się dużą ilością danych

- Zignoruj wszystkie powyższe dane i użyj XML, jeśli nie jesteś zobowiązany do korzystania z tradycyjnej bazy danych! Jest to okazja, aby ćwiczyć niesamowitą technologię oprogramowania.

3 Ważność XML

W poprzednim rozdziale "poprawność" dokumentu XML przedstawiono jako koncepcję, która odnosi się do poprawności składni dokumentu. Wskazano jednak, że dobrze ukształtowany dokument bada tylko dokument, biorąc pod uwagę jego strukturę. Oznacza to, że poza formatem strukturalnym nie sprawdza żadnych innych warunków, które mógłby być zainteresowany kompozytor XML w odniesieniu do niektórych lub wszystkich elementów. Na przykład, jak możemy powiedzieć, że dokument słownikowy musi mieć co najmniej jeden element <słowo>? Albo jak możemy określić, jakie dane może zaakceptować dany element? Tego rodzaju kontrole są ważne, nie tylko dlatego, że pomagają autorowi stworzyć odpowiedni dokument, ale także w udostępnianiu danych. W tym

ostatnim przypadku różne strony, które dzielą dane, mogą uzgodnić określony format, na podstawie którego mogą kontrolować, czy otrzymały i / lub wygenerowały prawidłowe (ważne) dane. Wynalazcy XML wprowadzili strukturę towarzyszącą, aby umożliwić wykonanie wspomnianych kontroli. Następnie nazwał tę strukturę Definicja typu dokumentu (DTD).

3.1 Definicja typu dokumentu (DTD)

DTD pozwala XML-owi na zdefiniowanie poprawnej składni dokumentu XML. Może być zawarty w dokumencie XML lub zapisany jako osobny plik. W tym drugim przypadku rozszerzenie pliku powinno być "dtd".

Definicja typu dokumentu (DTD)

- DTD pomaga zdefiniować następujące elementy:

o Lista nazw elementów, które mogą występować w dokumencie XML

o Sposób, w jaki elementy mogą pojawiać się w połączeniu z innymi elementami

o Struktura elementów gniazdujących

o Nazwa atrybutów, które są dostępne dla każdego typu elementu

Uwaga - Termin "słownictwo" czasami używany w odniesieniu do elementów używanych w konkretnej aplikacji. DTD używa gramatyki EBNF (Extended Backus-Naur Form), a nie XML. Ta gramatyka jest gramatyką bezkontekstową. Tematyka gramatyk bezkontekstowych wykracza poza zakres tej książki. Konieczne jest tutaj zrozumienie, w jaki sposób jesteśmy w stanie skonstruować poprawny DTD i jak możemy sprawdzić ważność dokumentu XML względem DTD. Jak już wspomniano, dokument XML ma być samoopisowy, dlatego posiadanie DTD jest opcjonalne. Jeśli jednak planujesz udostępnić dokument i chcesz się upewnić, że dokument jest zgodny z określoną składnią, lepiej jest skomponować odpowiedni DTD, który będzie towarzyszył Twojemu dokumentowi XML. W ten sposób wszystkie strony, które korzystają z dokumentu, mają środek, dzięki któremu mogą sprawdzić, czy dokument spełnia określone zasady składni, czy nie.

Jak zbudować DTD?

- Ogólna forma DTD:

o <! ELEMENT root (element1, element2, ...)>

o <! ELEMENT element1 (# ELEMEN TYPE)>

o <! ELEMENT element2 (# ELEMEN TYPE)>

o ...

- Rozpoczyna się od rootu <! DOCTYPE

- Reszta definicji jest zawarta w [].

(Opracowywanie DTD SGML - od tekstu, do modelu, do znaczników) autorstwa Eve Maler i Jeanne El Andaloussi zawiera pełne odniesienie do DTD. Można go znaleźć pod adresem http://www.xmlgrrl.com/publications/DSDTD/index.html

3.2 Deklaracja typu elementu

Kiedy definiujesz element w DTD, jego typ musi być zadeklarowany. Zgłoszenie tego typu oznacza, że pewna reguła musi być zastosowana, gdy element jest przedstawiony w dokumencie XML. Jak wspomniano, element można powtórzyć w dokumencie XML. W rzeczywistości jest to sposób, w jaki prezentowane są wielokrotne wystąpienia elementu. Deklaracja typu elementu pozwala określić reguły, których należy przestrzegać w przypadku powtarzania elementu.

3.2.1 Reguły powtarzania elementów

Tabela 1 pokazuje reguły powtarzania elementów, odpowiednie symbole, które muszą być użyte w deklaracji, oraz przykład ich użycia

Reguły powtórzeń	Symbol	Przykład
Zero lub więcej	*	<! ELEMENT sampleDic (word) *>
Jedno lub więcej słów	+	<! ELEMENT (head, definition +)>
Zero lub dokładnie jeden	?	<! ELEMENT słowo (głowa, definicja +, źródło?)>
Dokładnie jedno słowo		<! ELEMENT (głowa, definicja)>
Elementy muszą być w		
określonej kolejności	,	<! ELEMENT (head, definition +)>
Kolejność występowania nie		
Ma znaczenia	Space	<! ELEMENT word (head definition +)>
Albo / Or |	Słowo	<! ELEMENT word (head, (definition | sence) +)>

3.2.2 Rodzaje elementów

Rodzaje danych elementów XML można zdefiniować w odpowiednim DTD. Tabela 2 pokazuje różne typy, które można przypisać do elementów.

Typ	Opis	Działanie analizatora składni XML
#PCDATA	Parsowane dane postaci	Dane będą analizowane.
#CDATA	Character Data	Nie parsowane.
ANY	Dowolna kombinacja	
	danych parsowalnych	Dowolny element z dowolną zawartością.
EMPTY	Pusty element	Dozwolony jest pusty element.

Poniżej wykorzystanie tych typów wyjaśniono na przykładzie

<!ELEMENT sampleDic (letter*)>

<!ELEMENT letter (word*)>

<!ELEMENT word (head, definition*, synonym*, antonym*)>

<!ELEMENT definition (sense+, description*)>

<!ELEMENT sense (sorder, stext)>

<!ELEMENT description (dtext, usage+)>

<!ELEMENT usage (utext, source+)>

<!ELEMENT synonym (synorder, syntext)>

<!ELEMENT antonym (antorder, anttext)>

<!ELEMENT head (#PCDATA)>

<!ELEMENT sorder (#PCDATA)>

<!ELEMENT stext (#PCDATA)>

<!ELEMENT dtext (#PCDATA)>

<!ELEMENT utext (#CDATA)>

<!ELEMENT source (#CDATA)>

<!ELEMENT synorder (#PCDATA)>

<!ELEMENT syntext (#PCDATA)>

<!ELEMENT antorder (#PCDATA)>

<!ELEMENT anttext (#PCDATA)>

W powyższym przykładzie niektóre typy elementów zostały zadeklarowane jako #PCDATA, a niektóre jako #CDATA. Powodem jest na przykład, że chcę pod-elementy XML "zamówić" i "tekst", aby "sens" był przetwarzany przez XML i aby nie zawierał żadnych nielegalnych symboli takich jak "<" i "&". Jednakże, jako przykłady "użycia" "zmysłu" mogą obejmować dowolną postać, w tym wspomniane symbole, które są interpretowane przez XML parser w inny sposób, zostały one zadeklarowane jako #CDATA, w którym to przypadku cokolwiek zostanie zapisane jako wartość tych elementów, zostanie wysłane na wyjście nienaruszone (niezapisane). Innymi słowy, CDATA instruuje procesor XML, aby ignorował znaki znaczników i przekazywał załączony tekst bezpośrednio do aplikacji. Zostanie to pokazane na przykładach w kolejnych rozdziałach.

3.3 Deklaracja listy atrybutów

Jak już wspomniano, atrybutami są informacje pojawiające się w elemencie. Aby zachować ważność dokumentu XML, atrybuty, które mają być używane w dokumencie, powinny być deklarowane za pomocą "deklaracji listy atrybutów". Atrybut może zostać zadeklarowany przy użyciu następującej składni:

<!ATTLIST nazwa-elementu nazwa-atrybutu-typu wartość-domyślna>

Składnia ma następujące części:

• element-name - jest to element, w którym pojawia się atrybut.

• attribute-name - jest to nazwa przypisana do atrybutu.

• atrybut-type - jest to typ atrybutu.

• wartość domyślna - jest to domyślna wartość atrybutu.

Atrybut default-type może być jednym z elementów pokazanych w tabeli 3.

Typ	Opis
CDATA	Wartością jest danych znakowych
ENTITY	Wartością jest Entity, czyli skrót lub odniesienie do lokalizacji zewnętrznej.
ENTITIES	Wartością może być wiele encji, które są oddzielone spacjami białymi.
(en1 | en2 | ...)	Lista wyliczeniowa; wartość atrybutu musi być jedną pozycją listy.
ID	Atrybut ma stałą wartość, której nie można zmienić w dokumencie XML.
IDREF	Atrybut jest odniesieniem do innego elementu.
IDREFS	Atrybut zawiera kilka odniesień, każde oddzielone białymi spacjami, do kilku innych elementów.
NMTOKEN	Wartość musi być poprawną nazwą XML.
NMTOKENS	Lista NMTOKEN oddzielonych białymi znakami.
NOTYFIKACJA	Wartością jest nazwa notacji.
xml:	Wrtość jest predefiniowaną wartością XML.

Atrybut wartości domyślnej może być jednym z elementów, które są pokazane w tabeli 4.

Typ	Opis
#DEFAULT	Atrybut ma wartość domyślną
#FIXED	Wartość Atrybut ma stałą wartość
#IMPLIED	Podanie atrybutu jest opcjonalne
# REQUIRED	Atrybut musi być dostarczony wraz z elementem

DEFAULT - Jeśli jest używany, oznacza to, że istnieje domyślna wartość dla atrybutu, w takim przypadku nawet jeśli zostanie zapomniana w XML, otrzyma tę domyślną wartość przez "dtd".

FIXED - W przypadku użycia oznacza to, że atrybutowi przypisano stałą wartość, taką, która jest podana w ATTLIST, i nie można jej zmienić w dokumencie XML.

IMPLIED - W przypadku użycia oznacza to, że atrybut nie ma wartości domyślnej. Oznacza to również, że uwzględnienie tego atrybutu nie jest obowiązkowe.

WYMAGANE - w przypadku użycia oznacza to, że atrybut nie ma wartości domyślnej, ale jego obecność w elemencie jest obowiązkowa.

3.4 Kiedy używać atrybutów?

Ogólna sugestia korzystania z atrybutów jest następująca:

"Unikaj używania atrybutów, dopóki nie jest to absolutnie konieczne!"

Innymi słowy, używaj atrybutów, gdy dodają one więcej czytelności do twojego dokumentu i unikaj ich używania, gdy powodują zamieszanie. Za tą sugestią kryje się kilka powodów. Po pierwsze, możesz zrobić wszystko, co musisz zrobić z normalną kompilacją dokumentów XML, bez dodawania więcej zamieszania. Po drugie, atrybuty nie mogą być rozwijane i nie mogą mieć struktury drzewa. Wreszcie, atrybuty nie mogą zawierać wielu wartości. Możesz wypowiedzieć się na temat tego ostatniego powodu, zwracając się do NMTOKENS, IDREFS i Enumerated. Rzeczywiście, we wszystkich tych przypadkach atrybut może na koniec przybierać jedną wartość spośród wielu. Dlatego warto ją przyjąć.

<!ELEMENT sampleDic (letter*)>

<!ELEMENT letter (word*)>

<!ELEMENT word (head, definition*, synonym*, antonym*)>

<!ELEMENT definition (sense+, description*)>

<!ELEMENT sense (sorder, stext)>

<!ELEMENT description (dtext, usage+)>

<!ELEMENT usage (utext, source+)>

<!ELEMENT synonym (synorder, syntext)>

<!ELEMENT antonym (antorder, anttext)>

<!ELEMENT head (#PCDATA)>

<!ELEMENT sorder (#PCDATA)>

<!ELEMENT stext (#PCDATA)>

<!ELEMENT dtext (#PCDATA)>

<!ELEMENT utext (#CDATA)>

<!ELEMENT source (#CDATA)>

<!ELEMENT synorder (#PCDATA)>

<!ELEMENT syntext (#PCDATA)>

<!ELEMENT antorder (#PCDATA)>

<!ELEMENT anttext (#PCDATA)>

<!ATTLIST head part-of-speech #PCDATA #REQUIRED>

3.5 Ważność dokumentu

Gdy wprowadziliśmy pojęcie poprawności formalnej w poprzednich sekcjach, dokumenty XML można zweryfikować w celu zapewnienia, że dokument nie tylko przestrzega zasad generowania dokumentów XML, ale także jest zgodny z uzgodnionym formatem dla jego elementów. Ważność ta jest ważną częścią wymiany i udostępniania danych. Oznacza to, że przygotowuje DTD, który działa jako protokół sprawdzania zgodności wymienianych (udostępnianie) dokumentów XML. Pomaga również kompozytorom XML uzyskać wcześniejszą wiedzę na temat sposobu kompilowania dokumentów. Dokument XML uznaje się za ważny, jeśli jego definicja jest zgodna z przedstawionymi zasadami w powiązanym DTD.

3.6 Korzystanie z DTD

DTD może być używany w dwóch formach, jako część plików XML lub jako osobny plik. Obie metody wyjaśniono w poniższych sekcjach.

3.6.1 Używanie DTD wewnątrz dokumentu XML

DTD może być używany jako część dokumentu XML. Rysunek 10 pokazuje tę metodę. Jednak użycie DTD w dokumencie XML ogranicza jego użycie tylko do tego konkretnego dokumentu. Jest to w pewnym stopniu sprzeczne z celami DTD, które definiują podstawę do sprawdzania poprawności dokumentów XML i służą jako wytyczna do kompilowania i generowania różnych dokumentów

<?xml version=“1.0“ encoding=“UTF-8“?>

<!DOCTYPE TVSCHEDULE [

<!ELEMENT sampleDic (letter*)>

<!ELEMENT letter (word*)>

<!ELEMENT word (head, definition*, synonym*, antonym*)>

<!ELEMENT definition (sense+, description*)>

<!ELEMENT sense (sorder, stext)>

<!ELEMENT description (dtext, usage+)>

<!ELEMENT usage (utext, source+)>

<!ELEMENT synonym (synorder, syntext)>

<!ELEMENT antonym (antorder, anttext)>

<!ELEMENT head (#PCDATA)>

<!ELEMENT sorder (#PCDATA)>

<!ELEMENT stext (#PCDATA)>

<!ELEMENT dtext (#PCDATA)>

<!ELEMENT utext (#CDATA)>

<!ELEMENT source (#CDATA)>

<!ELEMENT synorder (#PCDATA)>

<!ELEMENT syntext (#PCDATA)>

<!ELEMENT antorder (#PCDATA)>

<!ELEMENT anttext (#PCDATA)>

]>

<sampleDic>

<word>

<head> ethic </head>

<definition>

rules of behavior based on ideas about what is morally good and bad.

</definition>

<source>Merriam-Webster, online </source>

<usage>Ethics is his chosen field of study.</usage>

</word>

<word>

<head> moral </head>

<definition>

concerned with the principles of right and wrong behavior and the

goodness or badness of human character.

</definition>

<source>The Internet, online</source>

<usage>

the moral dimensions of medical intervention

</usage>

</word>

</sampleDic>

3.7 Używanie DTD jako oddzielnego pliku

DTD może być użyty jako osobny dokument, w którym to przypadku musi być odwołany w dokumencie XML. W ten sposób każdy dokument XML, który ma być zgodny z tymi samymi regułami dla ich generowania, może odnosić się do niego jako podstawy do sprawdzania poprawności. Rysunek 11 pokazuje tę metodę, zakładając, że DTD zostało zapisane w osobnym pliku o nazwie "SampleDic.dtd". Jeśli planujesz mieć kilka dokumentów XML, które są zgodne z tym samym DTD, z pewnością ta metoda byłaby bardziej wydajna niż poprzednia. Jednak poprzednia metoda ma swoją własną aplikację, gdy pożądane jest, aby DTD było blisko dokumentu XML.

<?xml version=“1.0“ encoding=“UTF-8“?>

<!DOCTYPE sampleDic SYSTEM “SampleDic.dtd“>

<sampleDic>

<word>

<head> ethic </head>

<definition>

rules of behavior based on ideas about what is morally good and bad.

</definition>

<source>Merriam-Webster, online </source>

<usage>Ethics is his chosen field of study.</usage>

</word>

<word>

<head> moral </head>

<definition>

concerned with the principles of right and wrong behavior and the

goodness or badness of human character.

</definition>

<source>The Internet, online</source>

<usage>

the moral dimensions of medical intervention

</usage>

</word>

</sampleDic>

Jeśli jest sprawdzany pod względem ważności, dokument XML na Rysunku 11 jest uważany za ważny dokument, ponieważ jest zgodny z podanym DTD. Sprawdzenie nie jest łatwe do wykonania ręcznie. Istnieją różne narzędzia, które można wykorzystać zarówno do poprawności formalnej, jak i poprawności. Poniżej znajdują się dwa przydatne narzędzia, które można wykorzystać. Oba narzędzia są otwarte i umożliwiają sprawdzenie poprawności i poprawności dokumentów XML:

• XML Copy Editor - proste narzędzie open source, które pozwala tworzyć dokumenty XML, DTD, schematy XML i więcej.

• NetBeans - IDE, zasadniczo dla programisty Java, który jest w stanie wykonać wiele różnych operacji

4 Korzystanie z XML

Wokół XML opracowano różne technologie. Każda z tych technologii służy innym celom. Przedstawimy najbardziej popularne w tej i następnych częściach. Aby korzystać z XML w systemie oprogramowania, należy użyć interfejsu programowania aplikacji (API). Powodem jest to, że dokument XML, jak już

wspomnieliśmy, jest czytelny dla człowieka, potrzebuje narzędzia, które umożliwi jego zrozumienie przez maszyny. To narzędzie jest "analizatorem składni", który może przechodzić przez strukturę drzewa

Dokument XML i przedstaw prezentację, którą może obsłużyć oprogramowanie. Te narzędzia (interfejsy API) są podzielone na dwie formy:

• Drzewo

• Zdarzenie napędzane

Poniższe sekcje wyjaśniają te metody.

4.1 API XML oparty na drzewach

Jak sama nazwa wskazuje, ten typ API rekonstruuje drzewo XML, ponieważ jest prezentowane w pliku XML. Oznacza to, że kompletna drzewna reprezentacja XML jest wbudowana w pamięć. W3C dostarczyło standard przechodzenia dokumentów, takich jak XML i HTML. Ten standard nosi nazwę "Dynamiczny model obiektowy" lub DOM. DOM ma trzy poziomy: Core DOM, XML DOM i HTML DOM.

4.1.1 XML DOM (Dynamic Object Model)

XML DOM jest standardowym modelem obiektowym i API dla XML, który jest niezależny od platformy i languageneutral. Wykonuje się jak poniżej:

• Tworzy XML jako strukturę drzewa w pamięci.

• Drzewo jest postrzegane jako złożona struktura węzłów.

• Interfejs obiektowy pozwala na przejście i przetworzenie drzewa.

• Działa na podstawie koncepcji węzła.

• Węzeł może być:

• Dokument XML - węzeł dokumentu

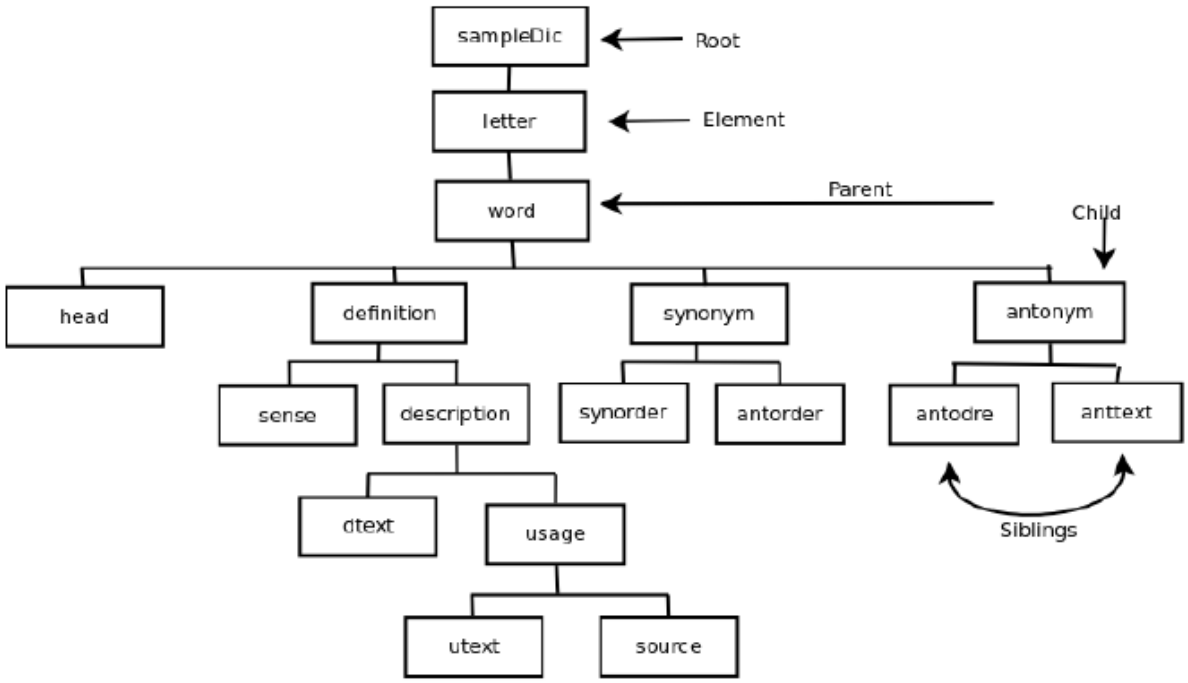
• Każdy element - węzeł elementu

• Teksty w elemencie - węzeł tekstowy

• Atrybuty - węzeł atrybutu

• Komentarze - węzeł komentarza

Rysunek 12 pokazuje drzewo DOM XML i definicje jego węzłów.



Utworzone drzewo według XML DOM jest nazywane drzewem węzłowym. Jak pokazano na rysunku 12, węzły podążają za rodzajem relacji "rodzinnej". To jest:

• Istnieje root, który jest ostatnim rodzicem wszystkich węzłów.

• Węzeł może być rodzicem dla niektórych dzieci.

• Każde dziecko ma rodzica (tylko jeden rodzic).

• Root nie jest dzieckiem, więc nie ma rodzica.

• Dzieci jednego rodzica nazywane są rodzeństwem.

• Węzeł bez dzieci nazywa się liściem.

Jak już wspomniano, istnieje interfejs API oparty na XML DOM, który zapewnia interfejs, za pomocą którego można przenosić i przetwarzać drzewo XML. Ponieważ interfejs ten opiera się na podejściu obiektowym, składa się z zestawu atrybutów i zestawu metod. Atrybuty przechowują informacje o drzewie węzłów, a metody udostępniają funkcje, takie jak dołączanie i usuwanie, które można wykonać na atrybutach. W rzeczywistości, gdy zamierza się przetwarzać dokument XML za pomocą XML DOM, należy postępować zgodnie z tradycyjnym sposobem programowania obiektowego. To jest stworzenie obiektu DOM i radzenie sobie z nim w sposób, który robią w podejściu obiektowym. Tworzenie dużych obiektów może być kosztowne, zarówno pod względem wielkości pamięci, jak i czasu do utworzenia. Dlatego XML DOM, niezależnie od tego, czy potrafi obsłużyć strukturalny widok dokumentu XML, w wielu przypadkach może nie być najlepszym wyborem dla programistów. Aby rozwiązać ten problem, opracowano interfejs API sterowany zdarzeniami. W poniższej sekcji omówiono to podejście.

4.2 Interfejs XML oparty na zdarzeniach

Jak sama nazwa wskazuje, ten interfejs przechodzi przez dokument XML oparty na serii zdarzeń, zamiast tworzyć całe drzewo w pamięci. Ta metoda nie została opracowana przez W3C. Został opracowany przez społeczność programistów XML. OIMIS international, konsorcjum non-profit,

obsługuje xml. org, który jest odpowiedzialny za "postępy w stosowaniu otwartych standardów poprzez dostarczanie informacji edukacyjnych, wspólne zasoby i obszary dyskusji ". Utrzymuje "listę dyskusyjną XML-DEV", która była odpowiedzialna za rozwój API opartego na zdarzeniach dla procesów XML (http://www.xml.org/xml-dev). Przyjrzymy się temu API w poniższej sekcji

4.2.1 Proste API dla XML (SAX)

"Lista dyskusyjna XML-DEV" opracowała prosty interfejs API dla XML (SAX). SAX jest powszechnie akceptowanym standardem de facto dla analizatorów składni XML. SAX to parser sterowany zdarzeniami. Zamiast tworzyć całe drzewo XML w pamięci, zachowuje minimalne wymagane dane, które są niezbędne do przetwarzania i przetwarzania. Ta ilość pamięci jest znacznie mniejsza niż ilość wymagana przez interfejsy API DOM. Pierwsza wersja SAX, SAX 1.0, została wydana w 1998 roku. W związku z tym SAX 2.0.1 został wydany w 2001 roku, który jest wciąż aktualną wersją. SAX używa wywołań zwrotnych do raportowania zdarzeń. Funkcja oddzwaniania to funkcja wywoływana przez wskaźnik. Oznacza to, że jeśli ktoś wysyła adres funkcji jako argument do innej funkcji, która z kolei wywołuje funkcję ukierunkowana funkcja za pośrednictwem przekazanego wskaźnika, następuje wywołanie zwrotne. Funkcja, która jest wywoływana za pośrednictwem przekazanego wskaźnika, jest funkcją wywołania zwrotnego. Wydarzenie to coś, co się dzieje. W przypadku analizowania XML zdarzenia są określonymi sytuacjami, które napotkał analizator składni, takimi jak wykrywanie "początku dokumentu", "początku elementu", "końca elementu" i "znaków" (lub nieprzetworzonego tekstu). Gdy takie zdarzenie miało miejsce, system wywołań zwrotnych analizatora składni wywołałby funkcję opracowaną przez użytkownika API. Istnieje inna implementacja SAX.

```
import javax.xml.parsers.SAXParser;

import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.Attributes;

import org.xml.sax.SAXException;

import org.xml.sax.helpers.DefaultHandler;

public class ReadXMLFile {

public static void main(String argv[]) {

try {

SAXParserFactory factory = SAXParserFactory.newInstance();

SAXParser saxParser = factory.newSAXParser();

DefaultHandler handler = new DefaultHandler() {

boolean bletter = false;

boolean bword = false;

boolean bhead = false;

boolean bsorder = false;

boolean bstext = false;

public void startElement(String uri, String localName,String xName,
```

```
Attributes attributes) throws SAXException {
System.out.println(“Start Element:” + xName);
if (xName.equalsIgnoreCase(“letter”)){
bletter = true;
}
if (xName.equalsIgnoreCase(“word”)){
bword = true;
}
if (xName.equalsIgnoreCase(“head”)){
bhead = true;
}
if (xName.equalsIgnoreCase(“sorder”)){
bsorder = true;
}
if (xName.equalsIgnoreCase(“stext”)){
bstext = true;
}
}
public void endElement(String uri, String localName,
String xName) throws SAXException {
System.out.println(“End Element:” + xName);
}
public void characters(char ch[], int start, int length) throws SAXException {
if (bletter) {
System.out.println(“Letter:” + new String(ch, start, length));
bletter = false;
}
if (bword) {
System.out.println(“Word:” + new String(ch, start, length));
bword = false;
}
```

```
if (bword) {
System.out.println(“Word:” + new String(ch, start, length));
bword = false;
}
if (bsorder) {
System.out.println(“Order:” + new String(ch, start, length));
bsorder = false;
}
}
};
saxParser.parse(“SampleDic-3.xml”, handler);
} catch (Exception e) {
e.printStackTrace();
}
}
}
```

Jak widać z kodu na Rysunku 13, Java SAX API wykorzystuje następujące metody:

• startDocument () - ta metoda jest wywoływana po wykryciu początku dla dokumentu XML.

• endDocument () - ta metoda jest wywoływana po wykryciu końca dokumentu XML.

• startElement () - ta metoda jest wywoływana po wykryciu początku elementu.

• endElement () - ta metoda jest wywoływana po wykryciu końca elementu.

• characters () - ta metoda jest wywoływana z przekazywaniem tekstu znajdującego się między tagiem start i tagiem końcowym jako parametrem wejściowym.

Uwaga: Można zauważyć, że była ona dużo czytelna, jeśli powyższy kod został napisany przy użyciu instrukcji "switch". Ponieważ instrukcja "switch" nie obsługuje zmiennych "string" jako zmiennej warunkowej w Javie przed 1.7, wolałem użyć ogólnej instrukcji "if then else", aby umożliwić kompilację kodu i uruchamianie go w różnych wersjach Javy

4.3 DOM kontra SAX

DOM lub SAX? Który powinien być naszym wyborem? Decyzja zależy od kilku parametrów, jak poniżej:

• Rozmiar dokumentów XML

• Liczba dokumentów XML

• Zasoby komputerowe, w szczególności pamięć

• Oczekiwana szybkość pobierania danych

• Charakter dokumentu XML: złożony a prosty

• Charakter procesu obliczeniowego: aktualizacja w porównaniu do odczytu

Poniżej znajdują się wskazówki dotyczące wyboru jednego z interfejsów API. Jednak nikt nie może zabronić ci korzystania z obu, jeśli analiza nakazuje ci to zrobić.

DOM lub SAX

Wybierz DOM, jeśli:

• Twoja struktura XML jest złożona, a system komputerowy ma dużą ilość pamięci.

• Musisz zaktualizować XML (w pamięci).

• Musisz zajmować się węzłami XML jako obiektami.

Wybierz SAX, jeśli:

• Początkowe pobieranie danych nie jest ważne.

• Nie trzeba aktualizować danych XML w pamięci.

• Nie musisz mieć całej struktury w pamięci.

• Twój system komputerowy ma ograniczone zasoby pamięci.

• Twoja aplikacja musi radzić sobie z danymi na podstawie ich występowania i zdarzeń, które wywołują podczas analizy.

Wybierz kombinację, jeśli:

• Masz kombinację powyższych sytuacji.

4.4 Przestrzeń nazw

Jak już wspomniano, dokumenty XML są łatwo opracowywane dla różnych aplikacji za pomocą prostego edytora. Powszechność XML sprawia, że dokumenty XML są bardzo podatne na konflikty nazw. Innymi słowy, ponieważ możliwe jest użycie tej samej nazwy w różnych kontekstach, całkiem możliwe jest, aby dokumenty XML używały nazwy elementu do określania różnych rzeczy. Spowodowałoby to konflikt nazwy, gdy chce się scalić te dokumenty. Rysunki 14 i 15 pokazują konflikt nazw.

<word>

<head> ethic </ head>

<definition>

zasady postępowania oparte na ideach dotyczących tego, co moralnie dobre i złe.

</ definition>

</ word>

<word>

<filename> test.doc </ filename>

<catefory> Raport techniczny 1 </ category>

</ word>

Przyczyną konfliktu jest to, że element "słowo" został zdefiniowany na dwa różne sposoby dokumenty, w których ani ich nazwy podelementów, ani ich zawartość nie pasują do siebie. Jeśli spróbujesz się połączyć tych dwóch dokumentów, otrzymają błąd wskazujący na to, że wystąpił konflikt nazw nie może być rozwiązane. W celu rozwiązania tego rodzaju konfliktów, XML używa mechanizmu nazywanego przestrzenią nazw. Programiści w niektórych językach programowania, takich jak C ++, C # i Python są już zaznajomieni z mechanizmem. W niektórych innych językach, na przykład Java, mechanizm ten jest stosowany poprzez koncepcję modułów. Przestrzeń nazw jest po prostu etykietą środowiska, w tym adresami, nazwami modułów, nazwami plików, lokalizacjami plików i takimi, w których "nazwa" jest unikalna. Przestrzeń nazw pozwala programistom upewnić się, że niektóre nazwy są unikalne w całej aplikacji, a nie tylko w pojedynczym programie. Innymi słowy, przestrzeń nazw umożliwia programistom zdefiniowanie zestawu nazw bez żadnych niejednoznaczności. W języku programowania służy również jako rozdzielczość zakresu. Poniższe sekcje wyjaśniają mechanizm w kontekście XML. Ale zanim przejdziemy do tematu przestrzeni nazw XML, przyjrzyjmy się metodzie rozwiązywania konfliktu o podwyższonej nazwie.

4.4.1 Rozwiązywanie konfliktów nazw za pomocą prefiksu

Prostą metodą rozwiązania konfliktu nazw jest użycie przedrostka do nazw. Na przykład możemy użyć dwóch prefiksów dla poprzednich przykładów, aby uniknąć konfliktu nazw. Ta metoda została przedstawiona na rysunkach 16, 17.

<dic:word>

<head> ethic </head>

<definition>

rules of behavior based on ideas about what is morally good and bad.

</definition>

</dic:word>

<proc:word>

<filename> test.doc </filename>

<category> Technical Report 1 </category>

</proc:word>

Prefiksy "dic" i "proc" teraz kwalifikują słowa, podając dwa nowe nazwy, które różnią się od siebie, dlatego nie mają już konfliktu. Musimy jednak określić, w jaki sposób parser może znaleźć "dic" i "proc".

4.4.2 Przestrzeń nazw XML

Z technicznego punktu widzenia powinniśmy określić przestrzeń nazw, w której można znaleźć te prefiksy. W XML przestrzeń nazw jest definiowana za pomocą atrybutu o nazwie xmlns (aby go zapamiętać, można go zobaczyć jako krótką nazwę dla "przestrzeni nazw XML"). Poniżej znajdują się główne zasady definicji XML namespce:

• Wartość atrybutu określa lokalizację, w której można znaleźć obszar nazw.

• Atrybut przestrzeni nazw może pojawić się w znaczniku początkowym elementu, w którym to przypadku wszystkie elementy podrzędne tego elementu będą powiązane z tą samą przestrzenią nazw.

• Paczka nazw może pojawić się w głównym elemencie, w którym to przypadku cały dokument będzie powiązany ze zdefiniowaną przestrzenią nazw.

• Ponieważ xmlns jest atrybutem, może zawierać kilka prefiksów, dzięki czemu każdy z nich jest unikalny, dlatego możliwe jest zdefiniowanie kilku przestrzeni nazw wewnątrz znacznika.

• Dzieci (podelementy) powinny być poprzedzone zdefiniowanym przedrostkiem

XML Namespace

Syntax:

xmlns:prefix=”URI”

prefix: is any valid string.

URI: is an abbreviation for “Uniform Resource Identifier”, which must be a valid Internet Resource. It is called URN – Universal Resource Name – as well.

Example:

xmlns=http://www.w3.org/HTML/1998/html4

4.4.3 Domyślny obszar nazw

Przydatnym sposobem uniknięcia powtarzania przed każdym elementem potomnym jest zdefiniowanie domyślnej przestrzeni nazw. Domyślna przestrzeń nazw nie ma żadnego prefiksu.

Domyślny obszar nazw XML

Składnia:

xmlns: = "URI"

Przykład:

xmlns = "http //www.w3.org/1999/xhtml"

W powyższym przykładzie wszystkie dzieci niejawnie używają definicji namepace podanej w znaczniku początkowym elementu XML. Rysunek 18 pokazuje przykład aplikacji przestrzeni nazw.

<dic: word>

<head> ethic </ head>

<definition>

zasady postępowania oparte na ideach dotyczących tego, co moralnie dobre i złe.

</ definition>

</ dic: word>

5 XSL - eXtensible Stylesheet Language

Większość czytelników, którzy już opracowali aplikacje internetowe, zna pojęcie CSS (Cascading Style Sheet). CSS to przydatny sposób predefiniowania niektórych tagów, które pozwalają dokumentom HTML na równomierne wyświetlanie zamierzonego materiału. Jednak HTML wykorzystuje predefiniowane znaczniki, które są interpretowane przez przeglądarki w ten sam sposób. CSS pomaga zunifikować kolory, czcionki i inne właściwości. W przeciwieństwie do HTML, XML nie ma predefiniowanych tagów. W rzeczywistości jednym z głównych celów XML jest umożliwienie użytkownikom definiowania własnych znaczników. Odpowiednio, XML nie ma predefiniowanego sposobu renderowania różnych znaczników. Dlatego w większości przeglądarek dokumenty XML byłyby wyświetlane tak, jak są zdefiniowane w dokumencie tekstowym, chociaż może istnieć "luksus" znaków "zwiń-rozwinięcie", które zwykle są oznaczone za pomocą "+" i "-", aby pozwalają rozwinąć lub zwinąć wybrane elementy lub cały dokument. W rezultacie W3C opracował arkusz stylów dla dokumentów XML o nazwie eXtensible Stylesheet Language, w skrócie XSL. XSL nie jest pojedynczym narzędziem ani językiem. Składa się z kilku części, które pozwalają użytkownikom zdefiniować sposób, w jaki chcą, aby ich dokumenty XML były renderowane i wyświetlane. Poniższe sekcje wyjaśniają XSL i jego różne części.

5.1 XSLT - Transformacja XSL

Jak już wspomnieliśmy, XML nie zapewnia żadnego konkretnego sposobu renderowania i wyświetlania informacji. Dlatego podstawowa część XSL została opracowana w celu zaspokojenia tej konieczności. XSL Transformation (XSLT for brief) to język, który pozwala programistom XML zdefiniować sposób, w jaki ich dokument XML może zostać "przekształcony" w inny format, taki jak HTML, rozpoznawalny przez przeglądarki. XSLT może również przekształcić dokument XML w inny dokument XML. XSLT oferuje kilka funkcji, takich jak:

• Dodaj elementy do pliku wyjściowego.

• Usuń elementy z pliku wyjściowego.

• Sortuj elementy.

• Testowanie elementów i podejmowanie decyzji (np. Pokazywanie / ukrywanie niektórych elementów).

Zanim jednak zajmiemy się dalszym XSL, należy wyjaśnić narzędzie i pojęcie: XPath, który pozwala użytkownikom (w tym przypadku XSLT) pobierać elementy dokumentu XML.

5.1.1 XPath

XPath został opracowany na podstawie wyrażeń ścieżki. Wyrażenie ścieżki to metoda w językach zapytań, która umożliwia nawigację do obiektu. Najczęstszym przykładem ścieżki jest to, co zobaczyłeś, określając lokalizację pliku na dysku komputera.

Na przykład:

• "/home/user1/temp/xmlsample.xml" w środowisku UNIX, lub

• "C: \ home \ user1 \ temp \ xmlsample.xml" w środowisku Microsoft Windows.

Ta sama idea służy do definiowania ścieżek w narzędziach XSL. XPath udostępnia różne funkcje. Oferuje ponad 100 wbudowanych funkcji. Funkcje są podzielone na różne kategorie, jak poniżej:

• Różne rodzaje porównań wartości

• Wartość logiczna

• Manipulowanie sekwencjami

• ...

XPath ma własną terminologię, którą można znaleźć poniżej.

Terminologia XPath

• Węzeł - ogólny termin dla następujących obiektów:

• element

• atrybut

• tekst

• przestrzeń nazw

• instrukcja przetwarzania

• komentarz

• Węzły dokumentów

• Atomic Value - Węzeł bez rodzica lub dzieci.

• Pozycja - Węzeł lub wartość atomowa.

• Parent - Element lub atrybut ma jednego rodzica.

• Dzieci - element może mieć zero, jedno lub więcej dzieci.

• Rodzeństwo - Węzły z tym samym rodzicem.

• Przodkowie - rodzic węzła, rodzic rodzica i tak dalej.

• Potomkowie - dzieci węzła, dzieci dzieci i tak dalej.

Aby pobrać informacje, XPath używa wyrażeń ścieżki, jak pokazano w Tabeli 5.

Wyrażenie	Opis
nodename	Wybierz wszystkie węzły o nazwie nodename
/	• Wybierz główny węzeł • Działa również jako separator między różnymi częściami wyrażenia ścieżki
//	Wybierz

.	• Wybierz węzeł kontekstu • Wybierz bieżący węzeł
..	Wybierz rodzica bieżącego węzła
@	Wybierz atrybut
*	Wybierz nieznany element (dopasowuje dowolny węzeł elementu)
@*	Wybierz nieznany atrybut (pasuje do dowolnego węzła atrybutu)
node ()	Wybierz nieznany węzeł (dopasowania i rodzaj węzła)
[predicate]	Znajduje węzeł, który spełnia określony warunek, zgodnie z predykatem.

Niektóre funkcje, takie jak last () i position (), mogą być używane w predykatach

5.1.2 Osie XPath

XPath ma inną zaawansowaną funkcję, która pozwala użytkownikom na pobranie zestawu węzłów na podstawie ich relacji z bieżącym węzłem. Ta funkcja nazywa się oś. W3C definiuje oś a zestawu węzłów względem bieżącego węzła. Tabela 6 pokazuje osie XPath.

Oś	Wynik Zestaw węzłów
self	Bieżący węzeł
	Rodzic bieżącego węzła
child	Wszystkie dzieci bieżącego węzła
atrybut	Wszystkie atrybuty bieżącego węzła
potomek	Wszystkie dzieci i dzieci dzieci i tak dalej w obecnym węźle
	następujące Wszystkie węzły po znaczniku zamykającym bieżącego węzła
previous	Wszystkie węzły przed bieżącym węzłem, z wyjątkiem przodków, atrybutów i przestrzeni nazw.
przodek	elementy nadrzędne, rodzice nadrzędnego itd. bieżącego węzła
potomek-lub-własny	Potomek + obecny węzeł
przodek-lub-siebie	Przodek + bieżący węzeł
follow-sibling	Wszystkie rodzeństwo po bieżącym węźle
previous-sibling	Wszystkie rodzeństwo przed bieżącym węzłem
namespace	Wszystkie węzły przestrzeni nazw bieżącego węzła

Zauważ różnicę między potomkiem a podążaniem. Pierwszy zapewnia dzieci i dzieci dzieci i tak dalej, ale nie te węzły, które nie "zstępują" od bieżącego węzła. Podczas gdy ta ostatnia zapewnia wszystko, co przychodzi po tagu zamykającym bieżącego węzła. Różnicę tę pokazano poniżej.

5.1.3 Operatory XPath

XPath obsługuje kilka operatorów, które umożliwiają tworzenie złożonych wyrażeń.

Operatory XPath

+, -, *, div, =,! =, <, <=,>,> =, lub, i, mod, |

Operatorzy nie wymagają objaśnień. Jedynym, który może wymagać wyjaśnienia, jest "|", która pobiera (oblicza) dwa zestawy węzłów. Podsumowując, XPath jest uważany za język deklaratywny, z perspektywy języków programowania. Oznacza to, że przy użyciu XPath pytamy komputer, co chcemy zrobić z dokumentem XML, ale bez określania, jak to zrobić. XPath odgrywa kluczową rolę w wielu technologiach XML takich jak XPointer, XSL, XML Schema i XQuery. Rola ta zostanie omówiona w kolejnych częściach.

5.2 XPointer

XPointer to kolejne narzędzie wspierające XML. Ułatwia także pobieranie informacji XML. Zasadniczo używa XPath do wyszukiwania informacji. W rzeczywistości XPointer został opracowany na XPath. Ma jednak dodatkowe możliwości, co daje większą prostotę w praktyce. Osiąga się to głównie poprzez wykorzystanie URI, o których pisaliśmy w sekcji 4.4.2. Po prostu XPointer można uznać za XPath wyrażenie, które pojawia się w URI. XPointer pozwala użytkownikowi łączyć się z sekcjami tekstu, wybierać poszczególne elementy lub atrybuty i poruszać się po elementach. Za pomocą XPointer można również wybrać dane zawarte w więcej niż jednym zestawie węzłów, czego nie można zrobić przy pomocy XPath.

5.3 XLink

Prawdopodobnie znasz pojęcie "łącza" w dokumentach HTML lub, jeśli nie w HTML, w innych dokumentach tekstowych, dzięki czemu można utworzyć "link" do źródła znajdującego się na lokalnym komputerze lub gdzie indziej, w Internecie, lub twoją prywatną sieć. XLink wykorzystał wspomnianą koncepcję i uogólnił ją do poziomu, który służy do obsługi bardziej złożonych i zaawansowanych linków, nie tylko do hipertekstów, ale także do dokumentów XML w ogóle. XLink, używa XPointer do adresowania i dodaje więcej możliwości obliczeniowych, które pozwalają na wstawianie elementów do dokumentów XML i tworzenie "łączy" pomiędzy różnymi zasobami. XLink, jak to jest ogólne podejście z prawie wszystkimi technologiami XML, używa składni XML do tworzenia linków. Istnieją dwa typy XLink:

• Simple XLink - łączy źródło ze źródłem docelowym.

• Rozszerzony XLink - łączy dowolną liczbę zasobów

6 Schemat XML

W części 3 wprowadziliśmy koncepcję walidacji XML poprzez definicję typu dokumentu (DTD). Pomimo prostoty DTD były uważane za silne narzędzie do sprawdzania poprawności XML i odgrywały wielką rolę w utrzymaniu jednolitości wśród różnych stron, które zamierzały udostępniać informacje za pomocą dokumentów XML. Jednak używanie DTD nakłada na użytkownika kilka ograniczeń, jak poniżej:

• DTD ma składnię inną niż XML.

• Jego typy danych są bardzo ograniczone.

• Nie obsługuje przestrzeni nazw.

W3C opracowało Schemat XML, aby zapewnić bardziej wyrafinowaną i wszechstronną technologię, dzięki której zastosowania są w stanie opisać strukturę dokumentów XML. W3C polecił tę technologię pod nazwą XML Schema Definition (XSD). Jak sama nazwa wskazuje, można spojrzeć na schematy XML jako odpowiednik schematów w kontekście projektowania bazy danych. Jak wiadomo, w kontekście bazy danych powszechnie stosowano schematy. W tym kontekście schemat jest formalnym sposobem definiowania bazy danych i podstawą, na podstawie której można utworzyć bazę danych. Schemat relacji to także formalna definicja relacji (inaczej mówiąc, tabela), poprzez którą można utworzyć relację i zdefiniować jej charakterystykę. Schematy odgrywają kluczową rolę w bazach danych i stanowią podstawę do modyfikacji, manipulacji i wyszukiwania informacji w bazach danych. Schemat XML, analogicznie, odgrywa podobną rolę w technologii XML. Charakterystyka XML Schema i jej zdolność do wyrafinowanych definicji struktur XML pozwalają aplikacjom na dzielenie się i wymianę danych XML w dobrze zarządzany sposób. Podstawowe role schematu XML dla określonego dokumentu XML to:

• Aby zdefiniować organizację struktury XML. Na przykład dotyczy to sposobu, w jaki elementy, dzieci i atrybuty muszą pojawiać się w dokumencie. Również, czy element może być pusty.

• Aby zdefiniować typy danych, które mają być używane w dokumencie XML.

• Aby zdefiniować elementy, które mogą być używane w dokumencie XML.

• Aby określić typ danych, które musi posiadać każdy element.

• Aby określić dowolne stałe i możliwe wartości domyślne dla elementów i atrybutów.

• Aby zdefiniować inne kryteria, takie jak wielkość liter w nazwach elementów.

Podobnie jak prawie wszystkie technologie XML, sama Schema XML jest dokumentem XML, dlatego może być tworzona, edytowana i przetwarzana w ten sam sposób i za pomocą tych samych narzędzi, które są używane do tworzenia, edycji i przetwarzania dokumentów XML. Istnieją jednak krytyczne opinie na temat schematu XML, wskazując na jego złożoność i techniczne problemy (na przykład: http://www.informit.com/articles/article.aspx?p=367637&seqNum=2).

W poniższych sekcjach wyjaśnione zostaną różne części schematu XML.

6.1 Element <schema>

Każdy schemat XML Schema z elementem <schema> jako jego root. W razie potrzeby może mieć pewne atrybuty.

6.2 Proste elementy i proste typy

Prosty element wskazuje, że element nie może zawierać niczego oprócz tekstu, gdy pojawia się w dokumencie XML. Oznacza to, że element nie może zawierać żadnych innych elementów ani atrybutów. Jednakże oznacza to, że zawiera on tylko "tekst" w swoim tradycyjnym znaczeniu. W rzeczywistości powinieneś zachować ostrożność przy zrozumieniu pojęcia, ponieważ prosty element może być łańcuchem, cyfrą po przecinku, liczbą całkowitą, datą, czas, wartość logiczna i takie. W dalszej części tej części zobaczysz, że możesz dodać pewne ograniczenia i wzorce, dzięki którym definicja prostych typów jest bardzo solidna i wyrafinowana. Elementy i atrybuty są niektórych typów.

Podstawowe typy schematów XML to: simpleType i complexType. Użytkownicy mogą również definiować własne typy na podstawie podstawowych typów. Proste elementy mogą przyjmować tylko simpleType.

Schemat XML - prosty element

Składnia:

<xs: nazwa elementu = "nazwa" type = "xs: simpleType" />

Atrybuty muszą być zdefiniowane jako ostatnie:

<xs: nazwa atrybutu = "atrybut" type = "xs: simpleType" />

simpleType:

• Numeryczne:

-- strunowy

- dziesiętny

-- pływak

- podwójne

- ...

• Strunowy

-- strunowy

- token

- NMTOKEN

- ...

• Trwanie

- czas

- data

- datetime

- ....

• Inne

- boolean

-- ID

- IDREFS

- ...

http://www.w3.org/TR/xmlschema-2/i http://www.xmlschemareference.com/builtInDatatype.html, aby dowiedzieć się więcej na temat wbudowanych typów danych XML Schema.

6.3 Złożone elementy i złożone typy

W przeciwieństwie do prostych elementów, złożone elementy są elementami, które zawierają inne elementy i / lub atrybuty. Są definiowane za pomocą elementu complexType. Lista dzieci typu złożonego opisana jest przez element sekwencji.

Schemat XML - złożony element

<xs: nazwa elementu = "nazwa">

<xs: complexType>

<xs: sequence>

<! - zdefiniowane tutaj dzieci ->

</ xs: sequence>

</ xs: complexType>

</ xs: element>

6.4 Wskaźniki schematu XML

Ogólna składnia XSD zawiera opcję zwaną wskaźnikiem. Istnieją trzy kategorie wskaźników, jak poniżej:

• Wskaźniki zamówień

• Wskaźniki zdarzeń

• Wskaźniki grupowe

Wskaźniki występowania są bardzo powszechne w XML Shemas. Tego typu wskaźniki zostaną wyjaśnione w następnej części.

6.4.1 Kardynalność

Ponieważ relacje między elementami można zdefiniować w schemacie XML, technologia oferuje koncepcję podobną do koncepcji, która jest używana w schemacie bazy danych, do definiowania liczności elementu. Kardynalność jest definiowana za pomocą wskaźników występowania. Wskaźniki zdarzeń służą do definiowania kardynalności elementów za pomocą następujących atrybutów:

• minOccurs

• maxOccurs

Schemat XML - Element Kardynalność

• minOccurs = "0" - Aby reprezentować opcjonalny element.

• maxOccurs = "bez ograniczeń" - Aby wskazać, że nie ma maksymalnej liczby

wystąpień.

6.5 Referencje

W niektórych przypadkach może zajść potrzeba odwołania się do innego elementu w schemacie XML. Można to osiągnąć za pomocą atrybutu ref. Możemy używać odwołań do elementów i definicji atrybutów

Schemat XML - Element Kardynalność

Składnia:

<xs: element ref = "element odniesiony" />

Przykład:

<xs: nazwa elementu = "sorder" typ = "xs: decimal" />

...

<xs: element ref = "sorder" />

6.6 Nowa definicja typu

Jak wspomniano, XSD pozwala użytkownikowi definiować nowe typy. Nowy typ został zdefiniowany jako ograniczenie ciągu znaków (do maksymalnej długości 9 znaków).

XML Schema – New Type Definition

Syntax:

<xs:simpleType name = “Mobile”>

<xs:restriction base = “xs:string”>

<xs:maxLength value = “9”/>

</xs:restriction>

</xs:simpleType>

Example:

<xs:simpleType name = “Mobile”>

<xs:restriction base = “xs:string”>

<xs:maxLength value = “9”/>

</xs:restriction>

</xs:simpleType>

6.7 Grupy

Elementy można pogrupować. To samo dotyczy atrybutów. Mimo że grupa nie jest typem danych, odgrywa rolę kontenera do przechowywania zestawu elementów lub atrybutów.

XML Schema – Group

Syntax:

<xs:group name=”group-name”>

<xs:sequence>

<xs:element name=”name” type=”type”/>

</xs:sequence>

</xs:group>

Example:

<xs:group name = “synonyms”>

<xs:sequence>

<xs:element name=“synonym” type=“string”/>

</xs:sequence>

</xs:group>

6.8 Ograniczenia

Schemat XML udostępnia funkcje oparte na XPath w celu określenia ograniczeń unikalności i odpowiednich ograniczeń referencyjnych, które są utrzymywane w określonym zakresie.

XML Schema – Constraints

Syntax:

<xs:unique name=”constraint_name”>

<xs:selector xpath = “selector_name”/>

<xs:field xpath = “field_name1”/>

<xs:field xpath = “field_name2”/>

….

</xs:unique>

Example:

<xs:unique name = “unique_”>

<xs:selector xpath = “guest”/>

<xs:field xpath = “Name/Surname”/>

<xs:field xpath = “DoB”/>

</xs:unique>

6.9 Kluczowe ograniczenia

Podobne do ograniczenia unikalności, z wyjątkiem tego, że wartość musi mieć wartość inną niż null. Pozwala również na odwołanie do klucza.

XML Schema – Constraints

Syntax:

<xs:key name=”key_name”/>

Example:

<xs:key name = “NAMEDOBUNIQUE”>

<xs:selector xpath = “guest”/>

<xs:field xpath = “Name/Surname”/>

<xs:field xpath = “DoB”/>

</xs:key>

7 Języki zapytań XML

W części 5 przedstawiono niektóre metody wyszukiwania informacji za pomocą XPath. Chociaż, jak wspomniano, XPath jest prostym i potężnym narzędziem do pobierania informacji z dokumentu XML, nie jest tak potężny, jak jest oczekiwany w przetwarzaniu zapytań w kontekście manipulacji danymi. Koncepcja kwerendy ma długą historię w systemie baz danych, niezależnie od ich architektury. Operacje takie jak wyodrębnianie danych, transformacja i integracja opierają się głównie na przetwarzaniu zapytań. Przetwarzanie zapytań odbywa się za pomocą języków zapytań. Standardowy język zapytań (SQL) i język zapytań obiektowych (OQL) to dwa dobrze znane języki zapytań, które są używane odpowiednio w relacyjnych i obiektowych bazach danych. Jednak te języki zapytań nie mają zastosowania, tak jak są, w kontekście XML. Dzieje się tak, ponieważ te języki zakładają dane strukturalne lub obiektowe, a dane XML nie są zwykłymi danymi. Jednak, jak widzieliśmy w poprzednich rozdziałach, dane XML są podobne do danych semistrukturalnych. Na szczęście opracowano języki zapytań dla semistruktur, które mają zastosowanie do dokumentów XML. Opracowano różne adaptacje tego rodzaju języków zapytań, na przykład:

• XML-QL (XML Query Language) został zgłoszony do W3C w 1998 roku (http://www.w3.org/TR/NOTE-xml-ql/).

• UnQL (Unstructured Query Language), wymawia się Uncle

• XQL

Wszystkie te języki zapytań mają pojęcie wyrażenia ścieżki do nawigacji w zagnieżdżonej strukturze XML.

7.1 XQuery

XQuery jest szeroko stosowanym językiem zapytań XML. Został wyprowadzony z Quilt, innego języka zapytań XML. Kołdra wykorzystywała funkcje różnych technologii, takich jak XPath, XML-QL, SQL, OQL, Lorel (kolejny język zapytań dla semistrukturalnych danych), XQL i YATL (Yet Another Transformation Language). XQuery jest językiem funkcjonalnym, w którym zapytanie jest reprezentowane jako wyrażenie. XQuery obsługuje kilka rodzajów wyrażeń, które można zagnieżdżać (wspierające pojęcie podkwerendy).

7.1.1 Wyrażenie ścieżki

Wyrażenie ścieżki jest podstawową częścią XQuery. Path Expression używa składni XPath. W XQuery:

• Wynikiem wyrażenia ścieżki jest uporządkowana lista węzłów.

• Lista wyników zawiera również węzły potomne.

• Listy są sortowane według:

• ich pozycja w pierwotnej hierarchii

• z góry na dół

• od lewej do prawej

• Wynik może zawierać zduplikowane wartości.

• Każdy krok w wyrażeniu ścieżki reprezentuje ruch w dokumencie w określonym kierunku.

• Każdy krok może wyeliminować węzły, stosując jeden lub więcej predykatów.

• Wynik każdego kroku jest listą węzłów, która służy jako punkt wyjścia do następnego kroku.

• Wyrażenie ścieżki może rozpoczynać się od wyrażenia identyfikującego określony węzeł, takiego jak funkcja doc (string), który zwraca węzeł główny wymienionego dokumentu.

• Zapytanie może również zawierać wyrażenie ścieżki rozpoczynające się od "/" lub "//", które reprezentuje niejawny węzeł główny określony przez środowisko, w którym wykonywane jest zapytanie.

Przykład:

Znajdź wszystkie zastosowania pierwszego słowa dokumentu XML z rysunku 6.

doc ("simpleDic.xml") / letters / word [1] // usage

To zapytanie zostanie wykonane w czterech następujących krokach:

• Najpierw otwiera "simpleDic.xml" i zwraca węzeł dokumentu.

• Używa "/ letters", aby wybrać element "letter" u góry.

• Lokalizuje pierwszy element "słowo", który jest dzieckiem elementu głównego.

• Znajduje element "użytkowania" występujący gdziekolwiek w obrębie tego "słowa" elementu.

7.1.2 Wyrażenia FLOWR

XQuery zdefiniował specjalną składnię, która nazywa się FLWOR (wymawia się "kwiat"). FLOW oznacza klauzule FOR, LET, WHERE, ORDER BY, RETURN. Są one używane w wyrażeniach do konstruowania żądanych zapytań. Aby skonstruować wyrażenie FLOWR:

• Zacznij od jednej lub więcej klauzul FOR lub LET w dowolnej kolejności

• Użyj klauzuli WHERE (opcjonalnie)

• Użyj klauzuli ORDER BY (opcjonalnie)

• Użyj klauzuli RETURN.

Cel każdej klauzuli jest następujący:

• Klauzula FOR

• Powiązanie wartości z jedną lub większą liczbą zmiennych za pomocą wyrażeń, takich jak wyrażenia ścieżki.

• Służy do iteracji, asocjacji każdej określonej zmiennej z wyrażeniem zwracającym listę węzłów.

• Może być interpretowany jako iteracja przez węzły zwracane przez odpowiednie wyrażenie.

• Klauzula LET

• Powiązanie wartości z jedną lub większą liczbą zmiennych za pomocą wyrażeń, takich jak wyrażenia ścieżki.

• W związku z tym nie ma iteracji, więc wiąże pojedyncze dla każdej zmiennej.

• Klauzula WHERE

• Określa jeden lub więcej warunków, aby ograniczyć krotki wygenerowane przez FOR i LET.

• Klauzula ORDER BY

• Określa kolejność strumienia krotek.

• Określony strumień krotek określa kolejność, w której klauzula RETURN jest oceniana przy użyciu zmiennych powiązań w odpowiednich krotkach.

• klauzula RETURN

• Oceniana raz dla każdej krotki w strumieniu krotek, a wyniki łączone w celu uzyskania wyniku.

Przykład:

Wymień "słowa", których źródłem jest "Miriam-Wbseter".

LET $ source: = "Miriam-Wbseter"

RETURN doc ("simpleDic.xml") // słowo [źródło = źródło $]

Zauważ, że warunek (predykat) wydaje się porównywać element (źródło) z wartością ($ source). Jednak w tym przypadku operator "równości" najpierw zwraca wartość ciągu, wyodrębniając wpisaną wartość elementu, a następnie porównuje ten ciąg z $ source. W rzeczywistości operator "równości" (=) jest operatorem ogólnego porównania, który działa w ten sposób. XQuery definiuje również operatorów porównywania wartości, jak poniżej:

• 'eq' - sprawdza, czy dwa węzły atomowe są równe.

• "ne" - sprawdza, czy dwa węzły atomowe nie są równe.

• "lt" - sprawdza, czy jedna wartość atomowa jest mniejsza od drugiej.

• "le" - sprawdza, czy jedna wartość atomowa jest mniejsza lub równa drugiej.

• 'gt' - sprawdza, czy jedna wartość atomowa jest większa od drugiej.

• "ge" - sprawdza, czy jedna wartość atomowa jest większa lub równa drugiej.

We wszystkich powyższych przypadkach, jeśli dowolny operand jest węzłem, operacja wykorzystuje proces atomizacji do przekształcenia go w wartość atomową. Jeśli FLOWR spróbuje porównać wartość atomową z wyrażeniem zwracającym wiele węzłów, to operator porównania porówna wartość true, jeśli dowolna wartość spełnia warunek, ale operator porównania wartości uruchomi wyjątek (podniesie błąd).

7.2 Zestaw informacji XML (XML Infoset)

Zbiór danych XML (Infoset) był rekomendowany przez W3C. Zapewnia spójny zestaw definicji umożliwiających użytkownikom odwoływanie się do informacji o dokumencie XML. Dokument XML zawiera informacje tylko wtedy, gdy jest dobrze sformułowany i spełnia wymagania przestrzeni nazw. Oznacza to, że nie ma potrzeby ważności dokumentu. Infoset zawiera informacje o niektórych lub wszystkich z następujących elementów:

• Dokument

• Elementy

• Atrybut

• Instrukcja przetwarzania

• Nieekspandowane odwołanie do jednostki

• Postacie

• Komentarze

• DTD

• Nieprzetworzone jednostki

• Notacje

• Przestrzenie nazw

8 XML i bazy danych

Mimo że XML można uznać za kontener danych, nie jest on uważany za bazę danych. Jednak istnieje również baza danych XML, która zostanie omówiona w dalszej części tego rozdziału. Ale wcześniej przyjrzyjmy się relacji między dokumentami XML a tradycyjnymi relacyjnymi bazami danych. Po pierwsze, jaki byłby powód przechowywania XML w bazie danych, w szczególności w relacyjnej bazie danych? Czy nie jest to sprzeczne z celem bazy danych relacji i jej podstaw, algebry relacyjnej? Istnieją różne opinie na ten temat. Niektórzy profesjonaliści są całkowicie przeciwni tej idei, niektórzy nie odrzucają jej w szczególnych przypadkach, a niektórzy wspierają ją w większości przypadków. Stoję gdzieś pośrodku. Uważam, że XML ma pełną moc w przechowywaniu danych półstrukturalnych. Z jednej strony relacyjna baza danych dowiodła swojej przydatności w wielu przypadkach, szczególnie w tych przypadkach, w których schemat można uznać za solidny, a zmiany są rzadkie. Z drugiej strony są przypadki, że ta sytuacja nie jest prawidłowa. Można spotkać się z systemami, które używają rodzaju danych, które bardzo często zmieniają się strukturalnie. Ci, którzy byli w dziedzinie bazy danych są dość obeznani z trudnością utrzymania takich baz danych i powiązanych z nimi schematów. Znów zdarzają się sytuacje, w których system wymienia ogromne ilości danych z różnymi stronami, które mają różne platformy i różne typy baz danych i kontenerów danych. We wspomnianych przypadkach przechowywanie dokumentu XML w bazie danych może być uważane za wybór i wierzę, że w większości takich sytuacji może to być dobry wybór. Jeśli podjęto decyzję o przechowywaniu dokumentu XML w relacyjnej bazie danych, istnieją różne podejścia, aby to zrobić, jak poniżej:

• Aby przechowywać XML jako wartość jakiegoś atrybutu w rekordzie (krotce).

• Fragmentacja kodu XML w kilku polach (atrybutach) i tabelach (relacje).

• Aby przechowywać XML w niezależnym schemacie.

• Aby przechowywać XML w analizowanym formularzu; tj. przekonwertuj plik XML na format wewnętrzny, taki jak reprezentacja Infoset lub PSVI, i zapisz tę reprezentację.

8.1 Przechowywanie XML w atrybucie

W tym podejściu kod XML będzie przechowywany w polu, którego typem danych jest typ obiektu dużej wielkości (CLOB). Jednak niektóre systemy zarządzania bazami danych (DBMS), takie jak MS SQL i Oracle, zapewniają natywny typ danych XML, który może być używany zamiast użycia funkcji CLOB. W tym podejściu, ponieważ surowy XML jest przechowywany w postaci szeregowej, efektywne byłoby wstawianie dokumentów do bazy danych i pobieranie ich w oryginalnej postaci. To sprawia, że wdrażanie i korzystanie z indeksowania pełnotekstowego jest łatwiejsze, co z kolei sprawia, że wyszukiwanie kontekstowe i istotne informacje są bardziej wydajne. Jednak niektóre problemy nadal pozostają. Na przykład, jaka byłaby wydajność, gdy dokument XML musi być parsowany w locie? Ponadto w tym przypadku nie można częściowo zaktualizować dokumentu XML. W rzeczywistości każda aktualizacja oznacza zastąpienie całego dokumentu.

8.2 Przechowywanie XML w kilku atrybutach i relacjach

W tym podejściu, które jest również nazywane postacią rozdrobnioną, XML rozkłada się na elementy składowe i dane rozłożone na liczbę atrybutów w jednym lub większej liczbie relacji. Przechowywanie rozdrobnionych dokumentów ma tę zaletę, że łatwiej jest indeksować te elementy, które zostałyby umieszczone w osobnych atrybutach relacji, w porównaniu z poprzednią metodą, pod warunkiem, że elementy te zostaną umieszczone w ich własnych atrybutach. Co więcej, to podejście pozwala twórcy dodawać dodatkowe dane, które umożliwiają zapisanie hierarchicznego formatu dokumentu XML w bazie danych, który z kolei dostarcza niezbędne dane do rekonstrukcji dokumentu XML, tak jak pierwotnie był. Takie podejście wymaga jednak pewnych wysiłków w celu zaprojektowania odpowiedniego schematu bazy danych, który powinien spełniać wspomniane wymagania.

8.3 Niezależne przedstawienie schematu

Inne podejście nazywa się niezależną od schematu reprezentacją, aby podkreślić ideę, że dane będą przechowywane w reprezentacji relacyjnej, która jest niezależna od jej schematu XML. Projektowanie reprezentacji relacyjnej jest zwykle oparte na koncepcji DOM. Oznacza to, że DOM jest strukturą opartą na drzewach, każdy węzeł ma tylko jednego rodzica.

8.4 SQL / XML

SQL / XML jest, podobnie jak SQL, językiem deklaratywnym jako rozszerzenie ANSI / ISO SQL 2003. Zapewnia funkcje publikowania XML, dzięki którym użytkownicy mogą tworzyć dokumenty XML. SQL / XML ma:

• Natywny typ danych XML, który pozwala użytkownikom traktować elementy XML jako wartości relacyjne w kolumnach tabel.

• Zestaw operatorów operujących na typie danych XML.

• Niejawny zestaw mapowań z danych relacyjnych do XML.

Jednak standard nie definiuje żadnych zasad dekompozycji dokumentu XML w formacie SQL. Microsoft SQL Server używa technologii o nazwie SQLXML i musi być odróżniony od SQL / XML. Ten pierwszy nie jest standardem, a raczej właściwością Microsoft, która umożliwia jego bazie danych obsługę dokumentów XML

```
CREATE TABLE simpleDic (
letter CHAR(1), word XML,
PRIMARY KEY letter, word );
INSERT INTO simpleDic VALUES ('E', 'ethic',
XML('<word>
<head> ethic </head>
<definition> rules of behavior based on ideas about what is
morally good and bad.
</definition>
<source>
Merriam-Webster, online
</source>
<usage>
Ethics is his chosen field of study.
</usage>
</word>) );
```

8.4.1 Operatory SQL / XML

SQL / XML oferuje kilka operatorów do obsługi typu danych XML, jak pokazano w Tabeli 8.

Operator	Działanie
XMLELEMENT	Aby wygenerować wartość XML z pojedynczym elementem jako element potomny elementu głównego. XMLATTRIBUTES może być użyty jako podrozdział, aby określić atrybuty elementów, jeśli je posiadają.
XMLFOREST	Aby wygenerować wartość XML z listą elementów jako element potomny elementu głównego.
XMLCONCAT	Aby połączyć listę wartości XML.
XMLPARSE	Aby wykonać niepotwierdzoną analizę ciągu znaków w celu wygenerowania wartości XML.
XMLROOT	Aby utworzyć wartość XML, modyfikując właściwości głównego elementu innej wartości XML.
XMLCOMMENT	Aby wygenerować komentarz XML.
XMLPI	Aby wygenerować instrukcję przetwarzania XML.
XMLSERIALIZE	Aby wygenerować znak lub łańcuch binarny z wartości XML.

XMLAGG	Funkcja agregująca, do generowania lasu elementów z kolekcji elementów.

8.4.2 Funkcje mapowania SQL / XML

Jak wspomniano, SQL / XML pozwala użytkownikom mapować relacje (tabele) danych do postaci dokumentów XML. Źródłem mapowania może być relacja, cały schemat lub wszystkie relacje w katalogu. SQL / XML udostępnia funkcje odwzorowania do wykonywania tych mapowań. Jednak standard nie określa żadnej konkretnej składni mapowania

8.4.3 Mapowanie typów danych SQL na schemat XML

SQL / XML mapuje każdy typ danych SQL na najbliżej dopasowany w schemacie XML. Specyfikacje XML związane z XML / IEC 9075-14: 2008 Część 14 (SQL / XML) definiują mapowanie z typów danych SQL na typy danych schematów XML

8.5 Załaduj XML do MySQL

Dokumenty XML można importować do baz danych MySQL. Można to zrobić za pomocą polecenia LOAD XML. Ogólne formy tego polecenia są następujące:

Załaduj XML do MySQL

LOADAL LOILAL LOILAL LOVE "file_name"

W TABELI [nazwa_bazy.] Nazwa_tabeli

Przykład:

LOADAL LOILAL XML LOKALNA LOKALIZACJA "mySimpleDic.xml"

INTO TABEL langDB.enSimpleDic

8.6 Natywne bazy danych XML

W poprzednich rozdziałach omówiliśmy koncepcję przechowywania dokumentów XML w relacyjnych bazach danych. Można to zrobić w dowolnej relacyjnej bazie danych za pomocą jednej z przedstawionych metod. Jednak, jak wspomniano, niektóre bazy danych oferują urządzenia do przechowywania danych XML. Ten rodzaj bazy danych nazywa się "XMLenabled" w niektórych kontekstach. Istnieje inny typ bazy danych, który umożliwia użytkownikowi określenie formatu XML danych. Niektóre nawet przechowują dane w formacie XML. Ten rodzaj bazy danych nazywa się Natywną bazą danych XML (NXD), a czasami po prostu bazą danych XML. Większość NXD może odpowiadać na zapytania i przekształcać dane w inne formaty. Ponieważ te bazy danych dotyczą głównie dokumentów XML, są one kategoryzowane w dokumentowych bazach danych. Są one również nazywane NoSQL (nie (tylko) SQL), ponieważ nie używają SQL jako głównego języka DDL (Data Definition Language) i DML (język manipulacji danymi). Jednak termin NoSQL odnosi się teraz do szerszego obszaru, obejmującego bazy danych, które są w stanie przechowywać różne formaty danych, których XML jest tylko jednym z nich. NXD powinien zdefiniować logiczny model danych oparty na technologii XML. Model powinien przynajmniej zapewniać obsługę elementów, atrybutów, typu PCDATA i definicji porządku dokumentów. Zasadniczo są one podzielone na dwie kategorie:

• Oparte na tekście: przechowujące XML jako tekst, np. jako plik w systemie plików lub jako obiekt CLOB w RDBMS

• Oparty na modelu: który przechowuje XML w niektórych wewnętrznych reprezentacjach drzewa, np. Infoset, PSVI lub reprezentację, możliwie z tokenami znaczników.

Dobre odpowiedzi na często zadawane pytania dotyczące korzystania z Native XML Databases można znaleźć tutaj: http://www.rpbourret.com/xml/UseCases.htm

Poniższe sekcje przedstawiają niektóre NXD używane obecnie w branży.

8.6.1 Oracle XML DB

Firma Oracle zaczęła wprowadzać Oracle XML DB po wydaniu Oracle 9i Release2, bardzo wydajnej relacyjnej bazy danych, obsługującej systemy na dużą skalę. Miało to na celu "zapewnienie wysokowydajnego, natywnego przechowywania danych, pobierania i zarządzania danymi XML". Firma Oracle udoskonaliła tę technologię wraz z wydaniem swojej nowej bazy danych 11g, w której dodała kilka nowych funkcji do swojej bazy danych XML, aby uprościć zadania programistów architektury XML i architektury zorientowanej na usługi (SOA). Oracle XML DB obsługuje teraz:

• Wiele natywnych modeli przechowywania danych XML i schematów indeksowania XML

• Standardowe operacje SQL / XML

• Standardowy model danych XQuery W3C i języki XQuery / XPath

• Natywne usługi sieciowe bazujące na bazach danych

• Wydajne publikowanie w formacie XML

• Repozytorium XML DB

• Wersja XML

• Kontrola dostępu XML

8.6.2 BaseX

BaseX to lekka baza danych XML o otwartym kodzie źródłowym. Oficjalna strona internetowa mówi: "jest to wydajny i skalowalny aparat bazy danych XML oraz procesor XPath / XQuery 3.1, który obejmuje pełną obsługę rozszerzeń W3C i rozszerzeń pełnotekstowych."

8.6.3 Sedna

Senda to natywna baza danych XML. Zapewnia pełny zakres podstawowych usług baz danych, w tym:

• Pamięć trwała

• Transakcje ACID

• Bezpieczeństwo

• Wskaźniki

• Gorąca kopia zapasowa.

• Elastyczne możliwości przetwarzania XML obejmują:

• Wdrożenie W3C XQuery

• Ścisła integracja XQuery z funkcjami wyszukiwania pełnotekstowego

• Język aktualizacji na poziomie węzła

Sedna jest dostępna za darmo w formie open source w ramach Apache License 2.0.

8.6.4 Apache Xindice

Apache Xindice (wymawiane jako zeen-dee-chay) to inna natywna baza danych XML, zaprojektowana pierwotnie do obsługi danych XML. Programiści Xindice apelują o prawidłowe wypowiadanie go! Xindice mówi, że ma zamiar zmienić to podejście do przechowywania danych w ogóle, są zainteresowani jedynie lepszym rozwiązaniem do obsługi danych XML. Mówią: "Jeśli nie masz żadnych danych XML, nie chcesz żadnych danych XML lub uważasz, że XML jest najbardziej przesadzoną technologią nowego tysiąclecia, to Xindice nie jest dla ciebie. "Zamiast tego bardzo często zachęcają tych, którzy napotykają dane XML, aby wypróbować Xindice. Xindice jest lepiej wykorzystywane do małych i średnich danych XML. Dzięki programistom Xindice mogą:

• Mapuj dane XML do innych struktur danych.

• Aby wstawić i pobrać dane jako XML.

• Aby skorzystać z elastyczności częściowo strukturyzowanej natury XML.

• Aby mieć niezależny model schematu.

8.6.5 eXistdb

eXistdb to kolejna baza danych NoSQL. Jest to Natywna Baza danych XML, opracowana przy użyciu Javy, która jest również uważana za "bazę danych dokumentów". Wykorzystuje XQuery i XSLT jako technologię przetwarzania zapytań. Baza danych została opracowana w oparciu o technologię XML. Jest dostępny dla trzech głównych platform (Windows, Linux i Mac).

8.7 XML i Big Data

Rozwój danych, szczególnie w związku z rozwojem danych pojawiających się w Internecie, skłonił badaczy do zaproponowania metod i narzędzi, które sprawią, że aplikacje będą w stanie obsłużyć tego typu dane. Pojęcie Big Data odnosi się do zbiorów danych, które stały się tak ogromne, że tradycyjne aplikacje nie mogą sobie z nimi poradzić. W rzeczywistości XML stał się jednym z głównych źródeł Big Data. Dlatego niektórzy z głównych graczy w tej dziedzinie zaproponowali narzędzia do przetwarzania XML w środowisku Big Data. Na przykład projekt Hadoop (https://hadoop.apache.org/#What+Is+Apache+Hadoop%3F), który został zainicjowany w celu obsługi dużych zbiorów danych i przetwarzania rozproszonego, dostarczył narzędzie do analizy danych XML. Ponownie, firma Google dostarczyła BigQuery (https://cloud.google.com/bigquery/what-is-bigquery) do przeprowadzania zapytań na ogromnych danych. Jednakże, ponieważ obsługa złożonych danych nie jest łatwa w oprogramowaniu BiqQuery, zapewnia pewne scenariusze przekształcania zawartości XML w formaty, które można wydajniej przetwarzać.

9 XML i platformy programistyczne

Ze względu na szybki wzrost wykorzystania XML i jego popularność, a także różnorodność użytkowania, prawie wszystkie ważne platformy programistyczne opracowały różne narzędzia wspierające tworzenie aplikacji opartych na XML. Poniżej przedstawiono niektóre z tych narzędzi.

9.1 XMLWriter z PHP

PHP ma klasę o nazwie XMLWriter, która pozwala programistom tworzyć dokumenty XML. Rysunek 20 pokazuje prosty przykład użycia XMLWriter w PHP. Istnieją inne sposoby na napisanie kodu, używając

niektórych funkcji PHP, takich jak iteratory (na przykład dla każdego z nich) i funkcje rekurencyjne, które mogą wydawać się bardziej inteligentne i wydajne dla niektórych czytelników. Jednak w wielu przypadkach przyjęcie takiego podejścia może być żmudnym wysiłkiem bez rzeczywistej nagrody.

```
<?php

$writer = new XMLWriter();

$writer->openURI('\home\user\sampleDic.xml'); // Output file

$writer->startDocument('1.0', 'UTF-8');

$writer->setIndent(4);

$writer->writeRaw('<?xml-stylesheet href= "sortedguest.xsl" type= "text/xsl" ?>');

$writer->startElement('letter');

$writer->startElement('word');

$writer->startElement('head');

$writer->writeElement('head', 'Ethic');

$writer->endElement(); // end head

$writer->startElement('definition');

$writer->startElement('sense');

$writer->startElement('sorder');

$writer->writeElement('sorder', '1');

$writer->endElement(); // end sorder

$writer->writeElement('stext', 'rules of behavior based on ideas about what is morally good and bad.');

$writer->endElement(); // end stext

$writer->endElement(); // end sense

$writer->startElement('description');

$writer->startElement('dtext');

$writer->endElement(); // end dtext

$writer->startElement('usage');

$writer->startElement('utext');

$writer->endElement(); // end utext

$writer->startElement('source');

$writer->endElement(); // end source

$writer->endElement(); // end usage

$writer->endElement(); // end description
```

```
$writer->endElement(); // end definition

$writer->endElement(); // end word

$writer->endElement(); // end letter

$writer->endDocument();

$writer->flush();

?>
```

9.2 Używanie XML w Pythonie

Python jest potężny dla wielu aplikacji. Ponieważ główny przykład powstał wokół słownika opartego na XML, właściwe jest, jeśli przyjrzymy się używaniu XML w Pythonie, który jest dobrze znanym językiem do przetwarzania języka naturalnego (NLP). Python ma pakiet XML, który obejmuje zarówno obsługę DOM, jak i SAX. Poniżej znajdują się dwie próbki kodów dla obu przypadków. Kody są proste i pokazują część możliwości każdego interfejsu API. Zauważysz również różnice między podejściem opartym na drzewach i zdarzeniami w przypadku każdego interfejsu API i uzyskasz lepsze zrozumienie tego, jak działają w prostym programie. Rysunek 21 pokazuje kod Pythona, który używa DOM do przetworzenia i wydrukowania SampleDic.xml, którego użyliśmy do naszej dyskusji w poprzednich rozdziałach. Rysunek 22 pokazuje proces wykorzystujący SAX API dla Pythona z analizy importu xml.dom.minidom

```
import xml.dom.minidom

DOMTree = xml.dom.minidom.parse(“SampleDic.xml”)

collection = DOMTree.documentElement

entries = collection.getElementsByTagName(“letter”)

for entry in entries:

print “Letter:”

if entry.hasAttribute(“letter”):

print “Letter: %s“ % movie.getAttribute(“letter”)

word = entry.getElementsByTagName(‘word’)[0]

print “Word: %s“ % word.childNodes[0].data

head = entry.getElementsByTagName(‘head’)[0]

print “Head: %s“ % head.childNodes[0].data

sorder = entry.getElementsByTagName(‘sorder’)[0]

print “ %s“ % stext.childNodes[0].data

sense = entry.getElementsByTagName(‘sense’)[0]

print “Sense: %s“ % stext.childNodes[0].data

utext = entry.getElementsByTagName(‘utext’)[0]
```

```
print “Usage: %s“ % utext.childNodes[0].data
source = entry.getElementsByTagName(‘source’)[0]
print “Source: %s“ % source.childNodes[0].data
description = entry.getElementsByTagName(‘description’)[0]
print “Description: %s“ % description.childNodes[0].data

import xml.sax
class sampleDicHandler( xml.sax.ContentHandler ):
def __init__(self ):
self.CurrentData = ““
self.head = ““
self.word = ““
self.sorder = ““
self.stext = ““
self.utext = ““
self.usage = ““
# Call when an element starts
def startElement(self, tag, attributes):
self.CurrentData = tag
if tag == “letter“:
print “*****Letter*****“
# Call when an elements ends
def endElement(self, tag):
if self.CurrentData == “letter“:
print “Letter:”, self.letter
elif self.CurrentData == “head“:
print “Head:”, self.head
elif self.CurrentData == “sense“:
print “Sense:”, self.sense
elif self.CurrentData == “definition“:
print “Definition:”, self.definiton
```

```
elif self.CurrentData == “usage“:
print “usage:”, self.usage
elif self.CurrentData == “description“:
print “Description:”, self.description
self.CurrentData = ““
# Call when a character is read
def characters(self, content):
if self.CurrentData == “head“:
self.head = content
elif self.CurrentData == “sorder“:
self.sorder = content
elif self.CurrentData == “stext“:
self.stext = content
elif self.CurrentData == “utext“:
self.utext = content
elif self.CurrentData == “source“:
self.source = content
elif self.CurrentData == “description“:
self.description = content
print content
# create an XMLReader
parser = xml.sax.make_parser()
# turn off namepsaces
parser.setFeature(xml.sax.handler.feature_namespaces, 0)
# override the default ContextHandler
handler = sampleDicHandler()
parser.setContentHandler( handler )
parser.parse(“SampleDic.xml”)
```

9.3 XMLWriter w MS Visual Studio

MS Visual Studio udostępnia klasę o nazwie XMLWriter. Można go używać w podobny sposób, jak wyjaśniono w poprzedniej sekcji na temat używania XML w PHP. Klasa może być używana w różnych językach obsługiwanych przez MS Visual Studio, takich jak C # i VB .NET. Ponadto klasa ma wiele publicznych metod obsługiwanych w aplikacjach .NET dla Sklepu Windows. Ta funkcja pozwala użytkownikom wygodnie tworzyć aplikacje mobilne, które mogą korzystać z technologii XML.

9.4 Języki specjalistyczne XML

XML znalazł swoje zastosowanie w wielu dziedzinach nauki. W niektórych dziedzinach opracowano wyspecjalizowany format XML, który służy specyfice danych terenowych. Poniżej przedstawiono niektóre z tych wyspecjalizowanych formularzy. Próbki są właśnie wybrane, aby pokazać różnorodność i rozszerzenia prób specjalizacji i nie wykazują żadnych preferencji ani szczególnego znaczenia.

9.4.1 Chemiczny język znaczników (CML)

"CML zapewnia wsparcie dla większości chemii, szczególnie cząsteczek, związków, reakcji, widm, kryształów i chemii obliczeniowej (compchem)."

9.4.2 Matematyczny język znaczników (MathMLTM)

"MathML to aplikacja XML do opisywania notacji matematycznych i przechwytywania zarówno jej struktury, jak i zawartości."

9.4.3 Język znaczników dziurkowanych

Pierwotnie opracowany przez Google język znaczników oparty na XML koncentruje się na wizualizacji geograficznej i jest używany w systemie GIS (Geographic Information System).

9.4.4 Office Open XML (OOXML)

"Office Open XML (OpenXML) to proponowany otwarty standard dla dokumentów, prezentacji i arkuszy kalkulacyjnych, które mogą być swobodnie implementowane przez wiele aplikacji na wielu platformach."

9.5 XHTML - eXtensible HTML i HTML5

HTML został wprowadzony przez poważne ulepszenia i ewolucje od czasu jego wprowadzenia. HTML4 był świetnym krokiem wśród innych. Jednak nie spełniało ono wszystkich oczekiwań deweloperów. XHTML został wprowadzony jako rekonstrukcja HTML 4 w formacie XML 1.0. Miała to być następna generacja HTML. Ale nawet ten został zastąpiony przez HTML5, który od października 2014 stał się piątą kompletną i ostateczną wersją HTML zalecaną przez W3C. Jest to w zasadzie bardziej restrykcyjna i czystsza wersja HTML. Jednak ta książka nie zamierza omawiać tematu HTML. Temat jest znacznie szerszy. Zamiast tego chciałem po raz kolejny podkreślić wagę XML-a i wspomnieć o szkielecie przeglądania sieci, które jest teraz również oparte na tej technologii. Zainteresowani użytkownicy mogą odwoływać się do dokumentów W3C i wielu innych dostępnych zasobów w Internecie.

9.6 Bezpieczeństwo w XML

Kwestia bezpieczeństwa w odniesieniu do XML jest jednym z głównych problemów specjalistów zajmujących się danymi. W tej sekcji problem ten zostanie krótko omówiony .W3C zapewnił bezpieczne środowisko do manipulowania danymi za pomocą XML. Podstawowymi technologiami, które tworzą to środowisko są podpis XML, szyfrowanie XML i XKMS (XML Key Management Specification). Podpis XML zajmuje się procesami integralności i uwierzytelniania. Szyfrowanie XML zajmuje się proces szyfrowania całości lub części dokumentu XML. XKMS stanowi uzupełnienie dla XML Signature i

szyfrowania XML, które udostępnia protokoły do dystrybucji i rejestracji kluczy publicznych, które będą wykorzystywane w wymianie danych opartej na XML. Te technologie wspólnie rozwiązują problem bezpieczeństwa w odniesieniu do dokumentów XML. Tam standardy zostały ulepszone i zmienione w ciągu ostatniej dekady

9.7 XML i NLP (przetwarzanie języka naturalnego)

Stworzyliśmy nasz podstawowy przykład omawiania XML, wokół koncepcji zarządzania danymi dla słownika, w całej tej książce. Tak więc byłoby właściwe, aby zakończyć książkę, przeglądając użycie XML w kontekście przetwarzania tekstu i przetwarzania języka naturalnego (NLP). W tej sekcji podano kilka przykładów takich przypadków. XML został wykorzystany przez różne grupy badawcze, które są aktywne w NLP i polach przetwarzania tekstu. Na przykład Stanford Natural Language Processing Group na Uniwersytecie Stanforda (http: //nlp.stanford.edu/ software/ corenlp.shtml # About) dostarczył zestaw narzędzi, z których niektóre zapewniają prezentacje XML tekstów wejściowych, przy czym użytkownik może przetworzyć różne aspekty prezentowanego tekstu (patrz: http: // nlp. stanford.edu:8080/corenlp/ dla wersji demonstracyjnej). NLTK (Natural Language Toolkit), dobrze znany zestaw narzędzi do NLP, oparty na Pythonie, wykorzystywał również XML w różnych kontekstach, takich jak czytniki Corpus. Corpus jest jednym z głównych filarów NLP, który posiada dużą ilość tekstów, które zostały zebrane z różnych źródeł. Czytniki korpusu NLTK udostępniają różne klasy, które można wykorzystać do uzyskania dostępu do różnych zestawów korpusów. Również GATE (ogólna architektura inżynierii tekstu), obsługiwana głównie przez University of Sheffield, wykorzystała XML w NLP i przetwarzaniu tekstu (https://gate.ac.uk/). Podsumowując, XML jest uważany za odpowiednie środowisko do przetwarzania tekstu i języka naturalnego.