XML na poważnie

Autorzy: Przemysław Kazienko, Krzysztof Gwiazda
ISBN: 83-7197-765-4
Format: B5, stron: 444
Zawiera CD-ROM

Książka ta to przegląd głównych standardów związanych z językiem XML:

- XML 1.0.
- DTD – opis struktury dokumentu.
- XSLT – transformacje dokumentów.
- XPath – nawigacja i wyszukiwanie.
- XPointer – wskazania wewnątrz dokumentów.
- DOM (Level 1, 2, 3) – dostęp do dokumentu jako drzewa węzłów.
- SAX – zdarzeniowe przetwarzanie dokumentów.
- Namespaces – przestrzenie nazw.

a także najnowszych standardów zatwierdzonych w 2001 r.:

- XML Schema – schematy zawartości.
- XLink – odsyłacze hipertekstowe.
- XSL FO – prezentacja danych.
- XML Base – adresy odniesienia.
- SVG – grafika wektorowa.

Interesująco zapowiada się część poświęcona zastosowaniom języka XML:

- serwisy internetowe,
- elektroniczna wymiana danych (EDI),
- bazy danych XML – XDBMS,
- pseudostrukturalne dane tekstowe,
- systemy prezentacji i systemy interaktywne,
- dane specjalistyczne (np. MathML, CML).

Autorzy umieścili informacje o ograniczeniach poszczególnych standardów, jak również różnorodne zalecenia będące owocem ich przemyśleń a dotyczących np. sposobów tworzenia (modelowania) dokumentów, strategii budowy schematów XML Schema czy stosowania polskich liter.

Książka może być z powodzeniem wykorzystywana jako podręcznik języka XML dla studentów kierunków informatycznych. Przeznaczona jest także dla programistów, projektantów oraz analityków, jak również zwykłych użytkowników komputerów, posiadających jednak pewne obycie informatyczne.

„XML na poważnie” zawiera wiele przykładów, a większość z nich jest dostępna na załączonej płycie CD-ROM w postaci prawie 300 plików.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 2.

XML — pojęcia podstawowe

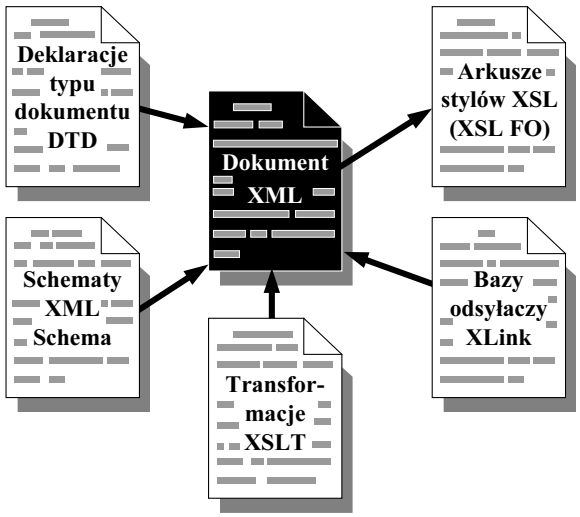
XML, ponieważ jest językiem znaczników, służy do opisywania zawartości dokumentów elektronicznych w sposób zrozumiały zarówno dla komputerów, jak i dla ludzi. Jego zaletą jest to, że nawet osoba nie mająca z nim nic wspólnego będzie w stanie w znacznym stopniu rozszyfrować informację zakodowaną w dokumencie XML. Wynika to z prostoty i harmonii tego języka.

2.1. Dokument XML

Aby rozgraniczyć niektóre pojęcia, ustalmy — dla potrzeb niniejszej książki — że **dokument XML** to dokument tekstowy (przechowywany zwykle w postaci pojedynczego pliku), w którym za pomocą elementów języka XML autor umieścił jakąś informację. Pozostałe dokumenty, np. DTD, schematy, arkusze stylów — jakkolwiek także wyrażone za pomocą języka XML — będą określane innymi terminami, np. „definicje typu dokumentu DTD", „schematy XML Schema", „arkusze stylów XSL" itd.

Definicje DTD, schematy, arkusze stylów, transformacje, a nawet bazy odsyłaczy są dodatkowymi specyfikacjami języka XML, odnoszącymi się do zasadniczej informacji zawartej w dokumencie XML — rysunek 2.1. Oczywiście często są to specyfikacje bardzo istotne dla konkretnych zastosowań. W dokumencie XML można więc wykorzystać definicje z DTD i schematów. Jednocześnie do dokumentu XML można zastosować transformacje, uzyskując np. postać gotową do wydruku — arkusze stylów. Specyfikacje XML przechowywane są zwykle w osobnych plikach, aczkolwiek mogą być także umieszczane wewnątrz dokumentu XML.

Ograniczenie znaczenia terminu „dokument XML" służy przede wszystkim łatwiejszemu zrozumieniu opisów, w których występuje więcej niż jedno pojęcie związane z językiem XML, np. „dokument XML zgodny z deklaracjami DTD" oznacza zbiór elementów XML (czyli dokument XML), który jest strukturalnie poprawny względem pewnego DTD.



Rysunek 2.1. *Dokument XML*

2.2. Podstawowe składniki dokumentu XML

Podstawowymi składnikami dokumentu XML są:

- **elementy**,
- **atrybuty**, które są umieszczane w elementach, jako dodatkowe informacje.

Elementy mogą być przy tym:

- **nie puste** — posiadające treść,
- **puste** — bez treści.

Oprócz tego w dokumencie XML można umieszczać:

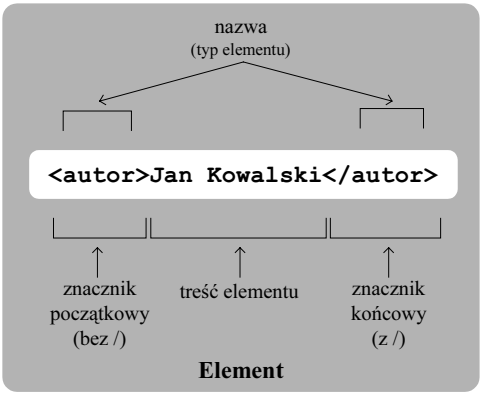
- **deklaracje**,
- **instrukcje przetwarzania**,
- **jednostki**.

Przyjrzyjmy się bliżej ww. składnikom dokumentów XML.

2.2.1. Elementy

Podstawowym budulcem dokumentu XML jest **element** (*element*). Posiada on zwykle tekstową **treść** poprzedzoną **znacznikiem początkowym** (*start-tag*) i zakończoną **znacznikiem końcowym** (*end-tag*) — rysunek 2.2. Oba znaczniki zawierają **nazwę elementu** objętą znakami „<” i „>”, przy czym w znaczniku końcowym nazwa jest zawsze poprzedzona znakiem „/”.

Rysunek 2.2.
Składowe elementu



Jeżeli element nie jest pusty (posiada jakąś treść), to w języku XML znacznik początkowy i końcowy zawsze muszą wystąpić. Brak któregoś ze znaczników oznacza, że dany element — a co za tym idzie cały dokument — są niepoprawne.

W nazwach elementów małe i duże litery są rozróżniane. Tworząc dokument XML, należy zwrócić uwagę na to, by nazwa występująca w znaczniku początkowym była taka sama jak nazwa umieszczona w znaczniku końcowym.

Element o nazwie autor jest inny niż element o nazwie `Autor`, `AUTOR` lub `aUtor`. W praktyce w nazwach używa się zwykle małych liter. Nie jest to jednak regułą i są tacy, którzy preferują duże litery, np. [125].

Poniższy element nie jest poprawny, gdyż nazwa umieszczona w jego znaczniku początkowym jest inna niż w znaczniku końcowym:

```
<autor> Jan Kowalski </AUTOR>
```

Natomiast elementy

```
<autor> Anna Nowicka </autor>
<autor> Jan Kowalski </autor>
<AUTOR> trzeci </AUTOR>
```

są poprawne. Pierwsze dwa są tego samego rodzaju, zaś trzeci jest inny — ma inną nazwę, a co za tym idzie zwykle także inne znaczenie.

Nie poleca się jednak wykorzystywać małych i dużych liter w celu tworzenia elementów o nazwach tak samo brzmiących (jak wyżej). Powoduje to zwykle więcej zamieszania, niż daje pożytku.

W nazwach elementów mogą występować polskie znaki diakrytyczne (ą, Ą, ć, Ć, ...)[1], jednak wymaga to odpowiedniego zadeklarowania i ścisłego przestrzegania kodowania znaków (patrz: punkt 2.6) we wszystkich dokumentach XML, a także dokumentach związanych, np. arkuszach stylów.

Niektóre programy przetwarzające mogą niepoprawnie interpretować polskie znaki w nazwach.

[1] Podobnie jak i inne znaki narodowe. Lista dozwolonych znaków w standardzie Unicode jest umieszczona w specyfikacji XML 1.0 pod adresem: *http://www.w3.org/TR/2000/REC-xml-20001006#NT-Letter*.

Dlatego też poprawnym elementem jest:

```
<ąĄćĆęĘłŁńŃóÓśŚżŻźŹ> Anna Nowicka </ąĄćĆęĘłŁńŃóÓśŚżŻźŹ>
```

Problemy z polskimi literami w nazwach elementów mogą wystąpić także w przypadku zastosowania dokumentów XML w kontaktach międzynarodowych. Kłopoty mogą również pojawić się przy wykorzystaniu dokumentów XML w połączeniu z bazami danych, które miewają różne standardy kodowania[2]. Z tego powodu w dużych projektach należałoby zalecić powstrzymanie się od polskich liter w nazwach elementów. W przypadku małych systemów, zwłaszcza wykorzystywanych przez zamknięte grono odbiorców, stosowanie polskich znaków może jednak zwiększyć czytelność dokumentów.

Nazwy elementów mogą zawierać cyfry, myślniki[3] i kropki, ale nie mogą się od nich zaczynać. Natomiast nazwy mogą się zaczynać od podkreśleń i dwukropków.

Poniższe elementy są poprawne[4]:

```
<Nazwa.1-2_3:ą> Treść </Nazwa.1-2_3:ą>
<_Nazwa.1-2_3:ą> Treść </_Nazwa.1-2_3:ą>
<:Nazwa.1-2_3:ą> Treść </:Nazwa.1-2_3:ą>
```

Natomiast:

```
<1Nazwa.1-2_3> Treść </1Nazwa.1-2_3>     <!-- cyfra na początku -->
<.Nazwa.1-2_3> Treść </.Nazwa.1-2_3>     <!-- kropka na początku -->
<-Nazwa.1-2_3> Treść </-Nazwa.1-2_3>     <!-- myślnik na początku -->
```

są niezgodne ze standardem XML 1.0 [22].

Nazwa elementu niesie ze sobą bardzo ważną informację. Dzięki niej można określić znaczenie treści zawartej między znacznikami.

Nazwę można więc traktować jako określenie **typu elementu**. Z tego względu należy nadawać takie nazwy, aby były one zrozumiałe dla człowieka. Nie zaleca się korzystać z jakiegokolwiek sposobu ich kodowania, tak powszechnego w przypadku nazewnictwa funkcji i zmiennych w językach programowania, np. zamiast `fktr` lepiej użyć po prostu `faktura`. Oczywiście wielu programistów preferuje zapis zwięzły, co zwykle wynika z ograniczenia długości nazwy. W języku XML jednak nie ma żadnych ograniczeń długości nazwy, więc nie ma konieczności używania skrótów. Istotne jest tutaj także to, że dokumenty XML są najczęściej przeznaczone nie dla programistów, ale dla zwykłych użytkowników komputerów, którzy wolą czytelniejsze wersje nazw.

Autorzy nie powinni się bać nadawania elementom długich, opisowych nazw.

Treścią elementu jest tekst, tzn. nawet jeżeli będzie on miał postać liczby, to i tak można ją traktować jako znakowy ciąg cyfr.

---

[2] Problem polskich liter w nazwach jest także omawiany nieco dalej, w punkcie 2.6.

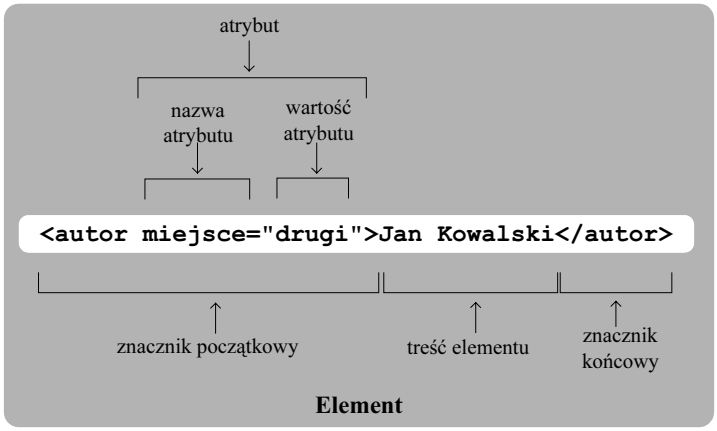
[3] Dotyczy to także podobnych znaków: półpauzy, pauzy oraz innych łączników (*Combining Characters*).

[4] Znak dwukropka rozdziela prefiks przestrzeni nazw od nazwy właściwej (patrz: punkt 3.11) i w związku z tym nie zaleca się używania go w innym celu. Przykładowo, parser XML zawarty w Microsoft Internet Explorerze nie dopuszcza stosowania znaku dwukropka na początku nazwy.

Ponieważ znaki „<” oraz „>” oznaczają początek i koniec znacznika, to nie mogą one występować w treści ujętej pomiędzy znacznikiem początkowym a końcowym (w treści elementu). Dlatego należy używać specjalnych symboli, tzw. jednostek wbudowanych (o których będzie mowa dalej), odpowiadających powyższym znakom. I tak: znak „<” jest zastępowany przez „`<`”, natomiast „>” przez „`>`”.

2.2.2. Atrybuty

Autor dokumentu XML może przekazywać informacje nie tylko przez treść i nazwę elementu. Dodatkowe informacje można również umieszczać, w postaci **atrybutów**, w znaczniku początkowym elementów — rysunek 2.3. Atrybut zawsze posiada **nazwę** i poprzedzoną znakiem równości **wartość**.



Rysunek 2.3. *Element zawierający atrybut*

Atrybuty mogą występować jedynie w znaczniku początkowym lub w znaczniku elementu pustego[5].

Podobnie jak w przypadku nazw elementów, małe i duże litery w nazwach atrybutów są rozróżniane.

W jednym elemencie nie mogą istnieć dwa atrybuty o takiej samej nazwie.

Dzięki umieszczeniu wartości atrybutu w cudzysłowie możliwe jest przechowywanie w nich dowolnych struktur tekstowych. Oczywiście — tak jak w przypadku treści elementów — ewentualne liczby traktowane są jako ciągi cyfr.

Zgodnie z powyższymi zasadami poniższy element jest poprawny:

```
<autor miejsce="pierwszy" Miejsce='drugi' MIEJSCE='trzeci'> Jan Kowalski </autor>
```

natomiast element

```
<autor miejsce="drugi" miejsce='trzeci'> Jan Kowalski </autor>
```

jest niepoprawny, gdyż posiada dwa atrybuty o tej samej nazwie.

[5] Elementy puste będą omówione w następnym podrozdziale.

W nazwach atrybutów również mogą występować polskie znaki diakrytyczne. Jednakże niektóre programy przetwarzające mogą je niepoprawnie interpretować[6].

Dlatego też poprawnym atrybutem jest:

```
<ŹźŻż ąĄćĆęĘłŁńŃóÓśŚżŻźŹ='Dziwny atrybut'> Anna Nowicka </ŹźŻż>
```

Wartości atrybutów muszą być zawsze umieszczone w cudzysłowach ("") lub apostrofach (’’).

Element:

```
<autorzy liczba_autorow='2'> Anna Nowicka, Jan Kowalski <autorzy>
```

jest poprawny, natomiast elementy:

```
<autorzy liczba_autorow=2> Anna Nowicka, Jan Kowalski <autorzy>
<autorzy liczba_autorow='2"> Anna Nowicka, Jan Kowalski <autorzy>
```

są niepoprawne, gdyż brak jest w nich takich samych znaków cudzysłowu lub apostrofów okalających wartość atrybutu.

Według standardu XML 1.0 kolejność atrybutów jest dowolna.

Może się jednak czasami zdarzyć, że program przetwarzający konkretny rodzaj dokumentów XML ma wbudowaną kolejność atrybutów lub ściśle przestrzega kolejności podanej w definicjach typu dokumentu DTD. Dobrym nawykiem jest więc zachowywanie tej samej kolejności atrybutów.

Podobnie jak znaki: „<” oraz „>” nie mogły występować w treści elementu, tak i tutaj zarówno apostrof ('), jak i cudzysłów (") nie powinny się pojawiać w polu wartości atrybutu. Dlatego należy zastąpić je odpowiednio przez „"” oraz „'”.

W praktyce wystarczy zwykle przestrzegać, aby wartość atrybutu zaczynała się i kończyła tym samym znakiem (apostrofu lub cudzysłowu), zaś w środku tej wartości może wystąpić drugi ze znaków (odpowiednio — cudzysłów lub apostrof).

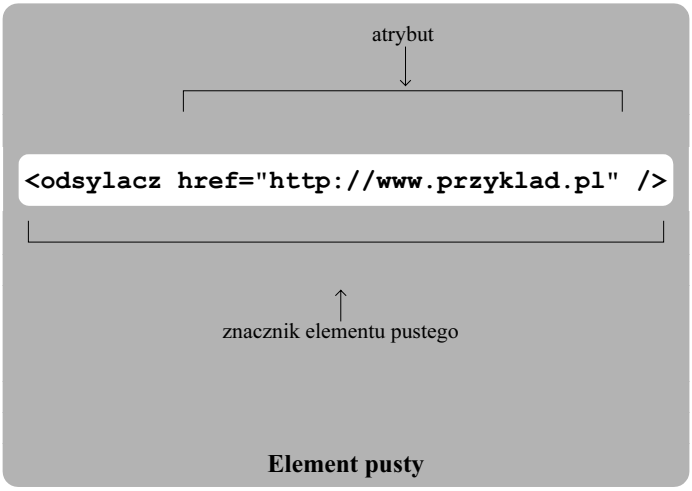
Prawidłowy jest więc element:

```
<autor pierwszy=' "Dziwny" "atrybut" ' drugi=" 'apostrof w środku' "> Anna Nowicka
</autor>
```

2.2.3. Elementy puste

W języku XML elementy mogą nie posiadać treści. W takim przypadku można zamiast dwóch znaczników — początkowego i końcowego — użyć pojedynczego, innego znacznika: **znacznika elementu pustego** (*empty-element tag*). Składa się on z nazwy elementu, po której następuje znak „/” — rysunek 2.4.

[6] Ograniczenia nałożone na nazwy atrybutów są takie same jak przy nazwach elementów, np. nazwa nie może zaczynać się od cyfry ani od kropki.



Rysunek 2.4. *Element pusty zawierający atrybut*

Znacznik elementu pustego zawsze kończy się znakiem ukośnika „/”, po którym dopiero następuje symbol „>”.

Oczywiście można także wykorzystać oba znaczniki: początkowy i końcowy i nie umieszczać między nimi żadnej treści.

Dlatego też oba poniższe przykłady są poprawne:

```
<faktura_usunieta/>
<faktura_usunieta></faktura_usunieta>
```

Naturalnie zalecaną formą jest ta pierwsza — krótsza.

Elementy puste zwykle (ale nie zawsze) zawierają atrybuty, np.:

```
<odsylacz href='http://www.onet.pl' nazwa='Portal Onet'/>
```

Zgodnie z powyższą uwagą można oczywiście także użyć:

```
<odsylacz href='http://www.onet.pl' nazwa='Portal Onet'></odsylacz>
```

Taka postać jest jednak rzadziej stosowana.

Elementy puste wykorzystuje się najczęściej wtedy, gdy zawierają one wyłącznie atrybuty.

Element pusty zazwyczaj składa się tylko ze znacznika elementu pustego, w którym można umieszczać atrybuty.

Elementy puste można także stosować wtedy, gdy potrzebne jest przekazanie dodatkowych informacji do programów przetwarzających dokument XML, np.:

```
<uruchom_mikrofon/>
```

2.3. Podelementy. Hierarchia elementów

Elementy mogą przechowywać tekst (pomiędzy znacznikiem początkowym i końcowym), mogą być puste, ale mogą także zawierać inne elementy — **podelementy** — dla których są wtedy **elementami nadrzędnymi**, np.:

```
<autorzy>
  <wspolautor> Anna Nowicka </wspolautor>
  <wspolautor> Jan Kowalski </wspolautor>
</autorzy>
```

Element `autorzy` zawiera w sobie dwa inne elementy (podelementy) — `wspolautor` — przy czym oba są tego samego rodzaju.

Oczywiście zagnieżdżanie może być bardziej złożone:

```
<sprawozdanie>
  <autorzy liczba_autorow="2">
    <wspolautor>
      <imie> Anna </imie>
      <nazwisko> Nowicka </nazwisko>
    </wspolautor>
    <wspolautor>
      <imie> Jan </imie>
      <nazwisko> Kowalski </nazwisko>
    </wspolautor>
  </autorzy>

  <tytul> Zestawienie zysków / obrotów </tytul>
</sprawozdanie>
```

Element `sprawozdanie` składa się tutaj z podelementu `autorzy` (także posiadającego podelementy), po którym następuje podelement `tytul`. Kolejność podelementów jest istotna i zamiana podelementu `autorzy` na podelement `tytul` da zupełnie inny dokument.

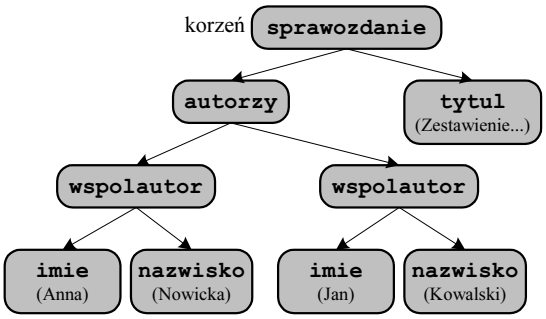
Od strony standardu języka XML nie ma żadnych ograniczeń na liczbę poziomów zagnieżdżeń.

Zagnieżdżając jedne elementy w drugich, umieszczamy w dokumencie XML informację o zależnościach występujących pomiędzy tym, co zostało zawarte w podelemencie a samym elementem, a także o zależnościach pomiędzy podelementami. Relacje te nazywane są **hierarchią elementów** — rysunek 2.5.

Podelement zawiera zwykle informację bardziej szczegółową niż element nadrzędny.

W związku z tym język XML nadaje się szczególnie dobrze do przechowywania informacji, w których występują relacje podległości (typu: nadrzędny — podrzędny), np. ogół — szczegół, całość — część, lista — element listy.

Nieco gorzej jest zbudować hierarchię dla relacji między obiektami o podobnym poziomie ogólności, np. klient — towar[7], uczeń — przedmiot[8], mąż — żona[9].



Rysunek 2.5. *Hierarchia elementów*

Zbudowanie odpowiedniej hierarchii elementów jest często najtrudniejszym problemem występującym przy tworzeniu dokumentów XML.

Zagadnienie budowy właściwej hierarchiijest zwykle rozważane podczas definiowania typu dokumentu DTD (rozdział 3.) lub schematu dokumentu (rozdział 4.).

W powyższym przykładzie widać, że `nazwisko` umieszczone jest na tym samym poziomie co `imie`, natomiast elementem najwyższym, głównym, czyli **korzeniem**, jest `sprawozdanie`.

W dokumentach XML może istnieć tylko jeden korzeń — element główny.

Cały dokument XML jest więc pojedynczą hierarchią z jednym korzeniem.

Elementy o takiej samej nazwie — nie będące korzeniem — mogą występować w różnych miejscach dokumentu, na różnych poziomach i mogą zawierać różnego rodzaju treść:

```
<dane_firmowe>
  <nazwa> Producent Jogurtów „Super-Zdrowie” S.A. </nazwa>

  <dostawa>
    <odbiorca>
      <nazwa> Hurtownia Spożywcza „Milk-XML” Sp. z o.o. </nazwa>
      <adres> ul. Niecodzienna 17, Zielona Góra </adres>
    </odbiorca>
```

[7] Relacja pomiędzy klientem a towarem wyraża się zwykle poprzez zakup lub sprzedaż. Jak jednak zbudować relację hierarchii dla dokumentu XML przechowującego informacje o miesięcznych zakupach jakiejś firmy (dane zakupu, dane klienta i informacje o towarze)? Dane o dostawcach lub towarach powielałyby się. Rozwiązaniem tutaj mogłyby się okazać odsyłacze (patrz rozdział 6.). Tradycyjne relacyjne bazy danych lepiej się sprawdzają w podobnych przypadkach.

[8] Uczeń jest w relacji z przedmiotem poprzez klasę, do której uczęszcza. Rozwiązanie podobnego problemu za pomocą odsyłaczy standardu XLink umieszczono w rozdziale 6.

[9] W relacji mąż – żona zwykle bezpieczniej nie określać stopnia podległości.

```
    <towar>
      <nazwa> Jogurt naturalny </nazwa>
      <magazyn>
        <nazwa> 5 </nazwa>
      </magazyn>
      <opakowanie>
        <nazwa> paleta </nazwa>
        <jednostka_detal> szt. </jednostka_detal>
      </opakowanie>
    </towar>
  </dostawa>
</dane_firmowe>
```

Element `nazwa` został umieszczony w kilku różnych kontekstach (nazwa firmy, nazwa dostawcy, nazwa towaru, nazwa magazynu, nazwa jednostki miary) i posiada różne rodzaje danych (tekstowe — nazwy własne firm, numeryczne — nazwa magazynu, pojedyncze słowa — nazwa jednostki miary opakowania).

Elementy mogą być zagnieżdżone, ale nie mogą na siebie zachodzić, tzn. znacznik końcowy podelementu musi wystąpić przez znacznikiem końcowym elementu nadrzędnego. Innymi słowy, każdy element (oprócz korzenia) musi być w całości otoczony przez inny element.

W związku z tym, poniższa postać jest nieprawidłowa:

```
<imie> Jan <nazwisko> Kowalski </imie></nazwisko>
```

Zamiast tego należy użyć:

```
<imie> Jan <nazwisko> Kowalski </nazwisko></imie>
```

lub:

```
<nazwisko> Kowalski <imie> Jan </imie></nazwisko>
```

Ze względu na równorzędność imienia i nazwiska najlepszym rozwiązaniem jest tutaj:

```
<imie> Jan </imie>
<nazwisko> Kowalski </nazwisko>
```

2.4. Rodzaje elementów

W przypadku, gdy element zawiera w sobie jakieś dane (tekst lub podelementy), jest nazywany pojemnikiem (*container element*).

Element — pojemnik może mieć następujące rodzaje zawartości:

1. **Zawartość elementową** (*element content*) — gdy pojemnik zawiera tylko i wyłącznie inne elementy, np.:

```
<autor>
  <opis> To jest współautor sprawozdania </opis>
  <imie> Jan </imie>
  <nazwisko> Kowalski </nazwisko>
</autor>
```

2. **Zawartość mieszaną** (*mixed content*) — gdy pojemnik zawiera zarówno tekst, jak i inne elementy, np.:

```
<autor>
  To jest współautor sprawozdania
  <imie> Jan </imie>
  <nazwisko> Kowalski </nazwisko>
</autor>
```

3. **Zawartość tekstową** (*data content*) — gdy pojemnik zawiera tylko i wyłącznie tekst, np.:

```
<autor> To jest współautor sprawozdania: Jan Kowalski </autor>
```

W przypadku zawartości mieszanej elementy mogą być nawet wplecione w tekst:

```
<sprawozdanie>
  Autorami sprawozdania są: <autor> Anna Nowicka </autor> oraz <autor> Jan Kowalski
</autor>. Innych autorów nie ma.
</sprawozdanie>
```

Zawartość mieszana nie jest jednak godna polecenia, gdyż może prowadzić do pewnego zamętu. Można ją stosować dla zwykłych tekstów, w których chcemy jedynie wydzielić pewne szczególne informacje. Jest to przydatne na etapie przejściowym, w którym tekst napisany w języku naturalnym przekształcamy w formy bardziej ustrukturalizowanej.

Dla ww. przypadku lepszym rozwiązaniem może być zawartość elementowa z dodatkowymi elementami, czyli z redundancją (powieleniem) informacji:

```
<sprawozdanie>
  <autor> Anna Nowicka </autor>
  <autor> Jan Kowalski </autor>
  <opis> Autorami sprawozdania są: Anna Nowicka oraz Jan Kowalski.
         Innych autorów nie ma. </opis>
</sprawozdanie>
```

lub (rozwiązanie gorsze):

```
<sprawozdanie>
  <opis>Autorami sprawozdania są: </opis>
  <autor> Anna Nowicka </autor>
  <opis> oraz </opis>
  <autor> Jan Kowalski </autor>
  <opis>. Innych autorów nie ma. </opis>
</sprawozdanie>
```

2.5. Postać elementów

Tworząc dokumenty XML, należy zwrócić uwagę na sposób zapisu dokumentu, czyli na postać dokumentu. Zapis:

```
<sprawozdanie><autorzy liczba_autorow="2"><autor><imie> Anna </imie> <nazwisko>
Nowicka </nazwisko></autor><autor><imie> Jan </imie> <nazwisko> Kowalski
</nazwisko></autor></autorzy><okres> 7 miesięcy </okres></sprawozdanie>
```

jest zupełnie nieczytelny! Poniższy zapis jest lepszy:

```
<sprawozdanie>
<autorzy liczba_autorow="2">
<autor>
<imie> Anna </imie>
<nazwisko> Nowicka </nazwisko>
</autor>
<autor>
<imie> Jan </imie>
<nazwisko> Kowalski </nazwisko>
</autor>
</autorzy>
<okres> 7 miesięcy </okres>
</sprawozdanie>
```

Brakuje mu jednak odpowiednich wcięć uwydatniających hierarchiczne zależności pomiędzy elementami. Jeszcze lepszą postacią zapisu jest więc:

```
<sprawozdanie>
  <autorzy liczba_autorow="2">
    <autor>
      <imie> Anna </imie>
      <nazwisko> Nowicka </nazwisko>
    </autor>
    <autor>
      <imie> Jan </imie>
      <nazwisko> Kowalski </nazwisko>
    </autor>
  </autorzy>
  <okres> 7 miesięcy </okres>
</sprawozdanie>
```

Zaważmy, że niektóre elementy zawarte są w jednej linii, zaś w innych znacznik początkowy i końcowy umieszczono w osobnych liniach. Można oczywiście znaczniki wszystkich elementów umieścić w osobnych liniach:

```
<sprawozdanie>
  <autorzy liczba_autorow="2">
    <autor>
      <imie>
        Anna
      </imie>
      <nazwisko>
        Nowicka
      </nazwisko>
    </autor>
    <autor>
      <imie>
        Jan
      </imie>
      <nazwisko>
        Kowalski
      </nazwisko>
    </autor>
  </autorzy>
  <okres>
    7 miesięcy
  </okres>
</sprawozdanie>
```

Wydaje się jednak, że to ostatnie rozwiązanie nie tyle poprawia czytelność dokumentu, ile zwiększa jego objętość, a co za tym idzie, raczej zmniejsza jego przejrzystość.

2.6. Deklaracja XML

Jak już wspomniano, podstawowym składnikiem dokumentu XML są elementy. To one zwykle zawierają zasadniczą informację, którą chcemy umieścić w dokumencie. Aby jednak dany dokument mógł być identyfikowany jako dokument XML, należy na jego początku umieścić **deklarację XML**. W deklaracji tej należy:

- określić wersję języka XML użytego w dokumencie — parametr `version`,
- określić rodzaj kodowania znaków — parametr `encoding`,
- podać informację o pominięciu (lub nie) przetwarzania zewnętrznych definicji typu dokumentu (DTD) oraz zewnętrznych jednostek — parametr `standalone` (informacja ta mówi o tym, czy dokument jest „samodzielny").

Deklaracja XML zaczyna się znakami `<?xml` i kończy sekwencją `?>`. Oto przykład deklaracji XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"  ?>
```

Parametr `version` jest obowiązkowy i zawsze powinien mieć wartość 1.0 (na razie istnieje tylko jedna wersja standardu XML). Pozostałe parametry (`encoding` oraz `standalone`) deklaracji są opcjonalne (mogą nie wystąpić).

Zalecanym rodzajem kodowania znaków w języku XML jest standard Unicode.

Standard Unicode jest obsługiwany przez większość systemów operacyjnych[10]. Każdy program przetwarzający dokumenty XML powinien potrafić odpowiednio interpretować kody Unicode w standardzie UTF-8 i UTF-16[11]. W UTF-8 wszystkie znaki zakodowane są w zmiennej liczbie bajtów (od 1 do nawet 6 bajtów), przy czym pierwsze 127 kodów ASCII (m.in. alfabet angielski, cyfry) zajmują tylko jeden bajt [71]. Polskie znaki są mniej uprzywilejowane (zajmują 2 bajty), zaś ich kody można znaleźć w [101].

Deklaracje XML wskazujące na kodowanie wg standardu Unicode wyglądają następująco:

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml version="1.0" encoding="UTF-16" ?>
```

Znaki mogą być także zakodowane za pomocą jednego ze standardów ISO, np. Latin 1 (bez polskich znaków diakrytycznych):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

[10] Obsługa ta nie zawsze jest pełna. Systemy często mają zaimplementowany ograniczony zestaw znaków Unicode, np. domyślne fonty Unicode w Windows 98 zawierają tylko 1 znak azjatycki.

[11] Wymóg ten został zapisany w standardzie XML 1.0 [22].

lub — jeżeli chcemy używać polskich znaków — Latin 2 (ISO 8859-2):

```
<?xml version="1.0" encoding="ISO-8859-2"?>
```

Dostępne są także standardy proponowane przez firmę Microsoft, znane z systemu Windows, np. dla polskich znaków może to być Windows-1250[12]:

```
<?xml version="1.0" encoding="windows-1250" ?>
```

Rodzaj kodowania zależy od tego, czy programy przetwarzające dokument będą rozumiały dany rodzaj kodowania.

Nie można stosować sposobu kodowania innego dla nazw elementów i atrybutów, a innego dla ich treści i wartości.

Możliwość stosowania tylko jednego kodowania może być pewnym ograniczeniem w przypadku zastosowania języka XML w połączeniu z bazami danych, w których często stosuje się różne standardy kodowania[13]. Dodatkowo nazwy elementów są często wykorzystywane w dokumentach związanych, np. w definicjach typu dokumentu DTD, schematach XML Schema, transformacjach XSLT czy wskazaniach XPath. Ewentualna zmiana kodowania polskich znaków (lub niezgodność z kodowaniem w dokumentach XML) spowoduje konieczność modyfikacji tych powiązanych dokumentów. Z tych powodów zwykle lepiej nie stosować polskich znaków diakrytycznych w nazwach elementów i atrybutów.

Przykładowy poprawny dokument XML, zawierający deklarację XML, może wyglądać następująco (plik *Sprawozdanie1.xml*):

```
<?xml version="1.0" encoding="ISO-8859-2"?>

<sprawozdanie>
  <autorzy liczba_autorow="2">
    <wspolautor> Anna Nowicka </wspolautor>
    <wspolautor> Jan Kowalski </wspolautor>
  </autorzy>
</sprawozdanie>
```

2.7. Poprawny dokument XML

Standard języka XML, jak już można było zauważyć, wyznacza pewne ograniczenia dotyczące zapisu elementu oraz relacji pomiędzy elementami. Przypomnijmy te ograniczenia:

1. Dokument musi się zaczynać od deklaracji XML, aby mógł być traktowany jako dokument XML.

[12] Takie kodowanie może jednak nie być akceptowane przez parsery inne niż firmy Microsoft.

[13] Wyobraźmy sobie sytuację, w której nazwy elementów w standardowej fakturze zawierają polskie znaki w standardzie Unicode, natomiast dane w bazach danych, niezbędne do wygenerowania ostatecznej wersji dokumentu przechowywane są w standardzie ISO Latin-2. Wówczas, podczas procesu wymiany faktury pomiędzy jedną a drugą bazą, niepotrzebnie należałoby dwukrotnie przekodowywać dane: najpierw ze standardu ISO Latin-2 do Unicode, a następnie odwrotnie.

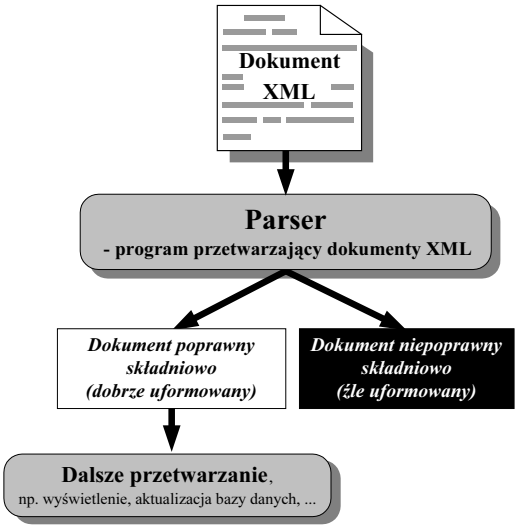
2. W dokumencie może istnieć tylko jeden unikatowy element główny — korzeń. Element o nazwie takiej jak korzeń nie może wystąpić jako podelement w tym dokumencie. W związku z tym wszystkie elementy występujące w dokumencie muszą się w całości zawierać wewnątrz korzenia.
3. Wszystkie niepuste elementy muszą posiadać znacznik początkowy i końcowy.
4. Element pusty może składać się z pojedynczego znacznika elementu pustego.
5. Elementy mogą być zagnieżdżone, jednak nie mogą na siebie zachodzić.
6. Wartości atrybutów muszą być umieszczone w cudzysłowach (lub apostrofach).

Przestrzeganie wymienionych wyżej reguł prowadzi do **dobrze uformowanego** (*well-formed*), czyli **poprawnego składniowo**, dokumentu XML. Większość programów przetwarzających dokumenty XML umożliwia sprawdzenie, czy dany dokument XML jest dobrze uformowany.

2.8. Przetwarzanie dokumentów XML

Dokumenty XML mogą być czytane przez użytkowników w zwykłych edytorach tekstowych, jednak najczęściej są one przetwarzane przez programy, zwane **parserami**. Programy te powinny w pierwszej kolejności sprawdzić poprawność składniową dokumentu, tzn. stwierdzić czy jest on dobrze uformowany — rysunek 2.6.

Rysunek 2.6. *Przetwarzanie dokumentów XML*



Następnie mogą one wykorzystać informację, zawartą w dokumencie, w jakimś specyficznym celu, np.:

- wyświetlić ją użytkownikowi w określonym formacie;
- zmienić postać informacji, np. przekształcając dokument XML na stronę w języku HTML;

- wygenerować nowy dokument XML i przekazać go innemu programowi;
- przetworzyć informację do postaci zgodnej z określonym formatem bazy danych i zaktualizować bazę danych.

Do dalszego przetwarzania dokumentu XML zwykle niezbędne są dodatkowe informacje, np. arkusze stylów, transformacje, definicje typu dokumentu DTD itd.

2.9. Inne składniki języka XML

Oprócz zwykłych elementów, zawierających informacje przydatne wprost dla użytkownika, dokument XML może zawierać także instrukcje dla programu przetwarzającego — parsera. Są to:

- deklaracje (w tym m.in. komentarze i bloki tekstu),
- instrukcje przetwarzania,
- jednostki.

2.9.1. Deklaracje

Deklaracje umieszczane są w tzw. **znacznikach deklaracji** (*markup declaration*). Różnią się nieznacznie od zwykłych znaczników. Otaczane są znakami „<!” oraz „>”.

Na początku deklaracji musi być umieszczona nazwa jednoznacznie identyfikująca daną deklarację:

```
<!NAZWA_DEKLARACJI ... >
```

Znaczniki deklaracji wykorzystywane są także do grupowania kilku innych deklaracji zamkniętych w zbiór przy użyciu znaków „[” oraz „]”, przy czym niektórych rodzajów deklaracji nie można grupować:

```
<!zbior_deklaracji [
  <!deklaracja1>
  <!deklaracja2>
]>
```

Standard XML 1.0 określa następujące deklaracje:

- deklaracje podstawowe (umieszczane w dokumentach XML),
- deklaracje wykorzystywane w definicjach typu dokumentu DTD.

Deklaracje podstawowe to:

- komentarze,
- sekcje CDATA.

2.9.1.1. Komentarze

Istnieje możliwość umieszczania w dokumencie XML **komentarzy** (*comments*). Zawierają one zwykle informację w języku naturalnym zrozumiałą jedynie dla człowieka i dlatego są przez programy przetwarzające dokument (parsery) pomijane. Komentarz zaczyna się czterema znakami „<!--", a kończy trzema „-->".

```
<!-- To są elementy opisujące autorów sprawozdania -->
<autor>Anna Nowicka</autor>   <!-- Autor pierwszy, co nie oznacza, że najważniejszy -->
<autor>Jan Kowalski</autor>  <!-- Autor drugi, wcale nie gorszy od pierwszego -->
```

Komentarze można umieszczać w dowolnym miejscu dokumentu.

Poprawny komentarz zawiera tylko dwie pary myślników występujących w części początkowej i końcowej. Nie mogą więc one wystąpić w treści zasadniczej komentarza:

```
<!-- To nie jest poprawny -- komentarz, bo ma w treści 2 myślniki -->
<!-- To jest poprawny komentarz - nie ma w treści dwóch myślników -->
```

Komentarze najczęściej występują w różnych specyfikacjach związanych z dokumentem XML (patrz rysunek 2.1), np. w definicjach typu dokumentu DTD czy schematach, gdyż tam istnieje zwykle największa potrzeba bliższego wyjaśnienia znaczenia elementów i relacji między nimi.

2.9.1.2. Sekcje CDATA — bloki tekstu

Jak już wcześniej zostało wspomniane, nie wszystkie znaki (symbole) mogą być zawarte w treści elementu, gdyż mają szczególne znaczenie dla przetwarzania dokumentu. W przypadku, gdy wymagane jest użycie znaków „<", „>" czy „&", zamiast używania kodów odpowiadających tymże znakom (`<`, `>`, `&`) można wykorzystać tzw. **blok tekstu**, czyli **sekcję CDATA**. Pozwala ona na podanie dowolnego tekstu, zaś wszystkie znaki ujęte wewnątrz deklaracji CDATA będą potraktowane jako zwykły tekst — dosłownie. Sekcja CDATA ma postać:

```
<![CDATA[tekst]]>
```

Załóżmy, że dokument XML zawiera w treściach elementów komunikaty wyświetlane przez pewien system SuperXML, w tym m.in. komunikat:

```
Naciśnij <<< Ctrl-Enter >>> aby pójść dalej
```

Odpowiedni dokument XML może mieć postać (plik *System1.xml*):


```
<?xml version="1.0" ?>

<system>
  <nazwa_systemu> SuperXML versja 2.0 </nazwa_systemu>
  <komunikat nr_komunikatu="102">
    Naciśnij &lt;&lt;&lt; Ctrl – Enter &gt;&gt;&gt; aby przejść dalej
  </komunikat>
</system>
```

Zamiast symbolu < oraz > konieczne było zastosowanie odpowiednio `<i >`. Można jednak zastosować tutaj sekcję CDATA (plik *System2.xml*):

```
<?xml version="1.0" ?>

<system>
  <nazwa_systemu> SuperXML versja 2.0 <nazwa_systemu>
  <komunikat nr_komunikatu="102">
    <![CDATA[Naciśnij <<< Ctrl-Enter >>> aby przejść dalej]]>
  </komunikat>
</system>
```

Bloki tekstu są przydatne także wtedy, gdy w treści elementu (lub jako wartość atrybutu) chcemy umieścić fragment dokumentu XML w postaci zwykłego tekstu. Przykładowo, aby umieścić tekst:

```
Złe zagnieżdżenie: <nazwisko> <imie></nazwisko></imie>
```

w treści elementu `przyklad`, należy napisać:

```
<przyklad>
  <![CDATA[Złe zagnieżdżenie: <nazwisko> <imie></nazwisko></imie>]] >
</przyklad>
```

Tekst zawarty w sekcji CDATA będzie traktowany przez parser jako ciąg znaków, a nie jako dwa niepoprawne elementy.

2.9.1.3. Deklaracje wykorzystywane w DTD

Istnieje kilka deklaracji wykorzystywanych w definicjach typu dokumentu DTD. Będą one szczegółowo omówione w rozdziale 3. W tym miejscu zostaną tylko wymienione — tabela 2.1.

Tabela 2.1. *Deklaracje wykorzystywane w DTD*

Deklaracja	Znaczenie deklaracji
`<!DOCTYPE ..... >`	Określa **typ dokumentu** i zawiera lub wskazuje definicję typu dokumentu DTD
`<!ENTITY ..... >`	Definiuje **jednostki**, dzięki którym można jeden raz zdefiniować pewien fragment dokumentu, a potem wielokrotnie się do niego odwoływać
`<!NOTATION ..... >`	Definiuje **notacje**, poprzez które można określać typy danych zewnętrznych
`<!ELEMENT ..... >`	Deklaruje element (opisuje zawartość elementu)
`<!ATTLIST ..... >`	Deklaruje listę atrybutów elementu
`<![IGNORE [ ..... ] >` `<![INCLUDE [ ..... ] >`	Umożliwia odpowiednio wyłączanie i włączanie do przetwarzania pewnych fragmentów dokumentu

Deklaracja typu dokumentu `DOCTYPE` nie zawsze musi być związana z DTD. Może ona być także wykorzystywana jedynie do nazwania dokumentu XML, np. deklaracja:

```
<?xml version="1.0" ?>
<!DOCTYPE  sprawozdanie>
<sprawozdanie>
  ...
</sprawozdanie>
```

umieszczona na początku dokumentu (po deklaracji XML) nadaje nazwę `sprawozdanie` całemu dokumentowi. Jest to jednocześnie nazwa elementu głównego — korzenia.

Takie zastosowanie deklaracji `DOCTYPE` nie ma jednak większego praktycznego sensu i dlatego też nie przez każdy parser deklaracja taka jest obsługiwana.

2.9.2. Instrukcje przetwarzania

Instrukcje przetwarzania (*processing instruction — PI*) zawierają informacje potrzebne dla programów przetwarzających dokument XML. Dla odróżnienia od zwykłego elementu instrukcje przetwarzania są umieszczane w bloku ograniczonym znakami „<?" i „?>". Instrukcje te zawierają **cel** (*PITarget*) oraz **instrukcję do wykonania**. Celem jest słowo kluczowe identyfikujące aplikację, dla której przeznaczona jest instrukcja do wykonania. Jeżeli program przetwarzający dokument poprawnie rozpozna cel, zadana instrukcja zostanie przez niego wykonana:

```
<!-- instrukcje dla robotów internetowych: zaindeksować i nie    przechodzić dalej
     po odsyłaczach -->
<?robots index="yes" follow="no" ?>

<!-- Instrukcje dla edytorów tekstowych -->
<?DTPSystem DO:page-break ?>  <!--Dla programu DTP: wstawić stronę-->

<?EdytorTXT {New Page} ?>     <!-- Edytor tekstowy ma dodać stronę-->

<!-- Jeżeli programem przetwarzającym dokument XML będzie serwer php,  to powinien
     on odpowiednio zinterpretować poniższy fragment  programu -->
<?php
  mysql_connect("baza.pwr.wroc.pl", "kowalski", "password");
  $result = mysql("STUDENCI", "SELECT Nazwisko, Imie FROM Studenci
    ORDER BY Nazwisko, Imie");
  $i = 0;
  while ($i < mysql_numrows ($result)) {
     $fields = mysql_fetch_row($result);
     echo "<student>$fields[1] $fields[0] </student>\r\n";
     $i++;
  }
  mysql_close();
?>
```

Jeżeli cel nie jest znany aplikacji przetwarzającej dokument, wtedy cała instrukcja zostanie przez nią pominięta.

Pewnym ograniczeniem jest to, że dwuznak „?>" nie może być zawarty w instrukcji do wykonania.

Instrukcje przetwarzania umożliwiają umieszczanie w dokumentach XML fragmentów innych języków, zwłaszcza języków programowania.

Gdy jako cel podana zostanie nazwa `xml`, wtedy otrzymujemy deklarację XML opisaną w punkcie 2.6. Nie jest to typowa instrukcja przetwarzania, zaś ciąg `xml` (w dowolnej kombinacji małych i dużych liter) jest zarezerwowany i nie należy używać go jako celu we własnych instrukcjach przetwarzania.

Instrukcje przetwarzania mogą być:

- **standardowe**, zdefiniowane w którymś ze standardów związanych z językiem XML, np. `<?xml ...?>`, `<?xml-stylesheet ... ?>` — wykorzystywane w języku XSLT;
- **niestandardowe** — niezwiązane ze standardami języka XML, czyli utworzone dla specjalistycznych potrzeb, np. dla skryptów PHP, konkretnego edytora tekstów czy dla potrzeb przetwarzania w procesie wymiany danych między firmą X a firmą Y.

2.9.3. Entity — jednostki

Język XML umożliwia fizyczną separację części przechowywanego dokumentu. Przykładowo, gdyby dokument XML opisywał książkę, to jej rozdziały można by umieścić w osobnych plikach. Umożliwia to sprawniejszą kontrolę, prostszą edycję czy nawet krótszy czas ładowania przez Internet. Każda taka osobna część jest nazywana **jednostką** lub **encją** (*entity*). Każda jednostka ma swoją nazwę, przez którą jest identyfikowana. Jednostka jest definiowana słowem kluczowym `ENTITY`, które wchodzi w skład **deklaracji jednostki** (*entity declaration*), a ta z kolei może być wstawiona jedynie w opisanej w następnym rozdziale deklaracji typów dokumentów (DTD). Wykorzystanie jednostki w dokumencie polega na umieszczeniu w żądanym miejscu odesłania do danej jednostki. Odesłanie składa się z nazwy jednostki poprzedzonej znakiem „`&`" oraz zakończonej znakiem średnika. W dokumencie może wystąpić wiele odesłań do tej samej jednostki.

Najczęściej jednostki stosuje się, gdy:

- ta sama informacja jest używana wielokrotnie;
- informacja jest częścią dużego dokumentu i z powodów praktycznych jest podzielona na łatwe w zarządzaniu części;
- informacja zawiera dane nie przetwarzane przez XML, lecz przez inną aplikację.

Najprostszą formą jednostki jest **jednostka wewnętrzna tekstowa** (*internal text entity*), zwana także **ogólną jednostką wewnętrzną** (*general internal entity*). Jej deklaracja znajduje się w przetwarzanym dokumencie XML; zawiera tylko i wyłącznie tekst. Stosuje się ją w przypadku częstego powtarzania fragmentu tekstu. Następująca deklaracja:

```
<!ENTITY XML "eXtensible Markup Language" >
```

oznacza, że nazwie `XML` przyporządkowano tekst „*eXtensible Markup Language*", zaś każde odwołanie do tej nazwy (czyli `&XML;`) będzie zastępowane poprzez ten tekst.

Przykładowo odwołanie:

```
<opis> To jest tekst o XML czyli o &XML;. Angielski termin "&XML;" oznacza
       rozszerzalny język znaczników</opis>
```

jest równoznaczne z:

```
<opis> To jest tekst o XML czyli o eXtensible Markup Language. Angielski termin
"eXtensible Markup Language" oznacza rozszerzalny język znaczników</opis>
```

Deklaracje jednostek powinny być umieszczane w deklaracji typu dokumentu DOCTYPE. Cały dokument zawierający powyższe opisy ma więc postać (plik *Jednostka.xml*):

```
<?xml version="1.0"?>
<!DOCTYPE opisy [
  <!ENTITY XML "eXtensible Markup Language">
]>

<opisy>

    <!-- oba opisy mają taka samą treść -->

  <opis> To jest tekst o XML czyli o &XML;. Angielski termin "&XML;"
  oznacza rozszerzalny język znaczników </opis>

  <opis> To jest tekst o XML czyli o eXtensible Markup Language.
  Angielski termin "eXtensible Markup Language" oznacza rozszerzalny
  język znaczników </opis>

</opisy>
```

Kolejnym typem jednostek jest **jednostka zewnętrzna tekstowa** (*external text entity*) zwana także **ogólną jednostką zewnętrzną** (*general external entity*). Zdefiniowanie jednostki zewnętrznej różni się od definicji jednostki wewnętrznej. W tym przypadku konieczne jest podanie, z jakiego pliku należy zaczerpnąć informację o treści danej jednostki. Deklaracja:

```
<!ENTITY XML SYSTEM "http://www.ksiazka.pl/jednostki/XML_skrot.xml">
```

oznacza, że w pliku *http://www.ksiazka.pl/jednostki/XML_skrot.xml* znajduje się angielska nazwa języka XML.

Odwołanie do ogólnej jednostki zewnętrznej jest takie samo jak do wewnętrznej:

```
<opis> To jest tekst o XML czyli o &XML;. Angielski termin "&XML;"  oznacza
       rozszerzalny język znaczników</opis>
```

Różnica między wymienionymi jednostkami jest jedynie w umiejscowieniu rozwinięcia nazwy jednostki: w jednostce wewnętrznej tekst rozwinięcia znajduje się w deklaracji, zaś w jednostce zewnętrznej — w osobnym, zewnętrznym pliku. W obu przypadkach każde odwołanie do jednostki jest zastępowane przez program przetwarzający tekst rozwinięcia.

Kolejnymi, aczkolwiek różnymi od powyżej wymienionych, są **predefiniowane jednostki ogólne** (*predefined entities*). Umożliwiają one umieszczanie w treści elementów, w wartościach atrybutów i innych miejscach dokumentu XML symboli zarezerwowanych. I tak:

- < zastępuje znak „<” (*less than*),
- > zastępuje znak „>” (*greater than*),
- & zastępuje symbol „&” (*ampersand*),
- ' zastępuje znak apostrofu ' (*apostrophe*),
- " zastępuje znak cudzysłowu " (*double quote*).

Jednostki predefiniowane nie wymagają deklarowania za pomocą ENTITY.

Jednostki predefiniowane nie podlegają żadnym zmianom. Nie istnieje także możliwość dodania własnych jednostek, tak by stały się jednostkami predefiniowanymi.

WWW Consortium dla potrzeb współpracy pomiędzy systemami wykorzystującymi język SGML oraz HTML w [84] zamieściło także szerszy zestaw jednostek predefiniowanych. Nie są to jednak jednostki standardowe dla języka XML i niekoniecznie są obsługiwane przez poszczególne parsery.

Wszystkie zaprezentowane dotąd jednostki są **jednostkami ogólnymi** (*general entity*). Można się do nich odwoływać w każdej części dokumentu, a także w deklaracjach DTD.

Istnieją także **jednostki parametryczne** (*parameter entity*), do których odwołania są dostępne jedynie w deklaracjach typu dokumentu DTD.

Jednostki zostały szerzej omówione w rozdziale 3. (patrz: punkt 3.13).

2.10. Nośniki informacji w dokumencie XML

Zasadniczym zadaniem dokumentów XML jest przechowywanie informacji. Autorzy tworzący dokumenty mogą w nich umieszczać informacje w różnych miejscach i na różny sposób (patrz rysunek 2.7), a mianowicie:

- w treściach — tekstach elementów,
- w nazwach elementów, które określają typ elementu i informują o tym, co zawiera cały element,
- w hierarchii (strukturze, układzie) elementów, która zawiera informacje o zależnościach między elementami,
- w nazwach atrybutów,
- w wartościach atrybutów.

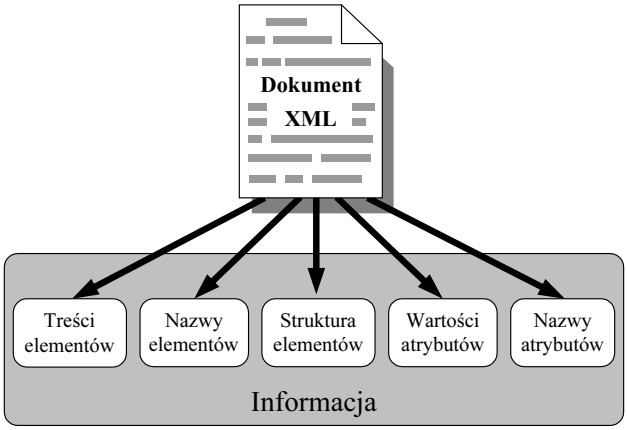
Informacje zawierające się w nazwach elementów i atrybutów są opisem zrozumiałym dla człowieka. W systemach nie wykorzystuje się ich zwykle w sposób bezpośredni, ale dopiero po przetworzeniu, zastępując je najczęściej etykietami, np. element:

```
<autor> Jan Kowalski </autor>
```

po odpowiednim przekształceniu (transformacji za pomocą XSLT) może mieć końcową postać:

```
Autor sprawozdania: Jan Kowalski
```

Znacznikowi `autor` odpowiada tutaj tekst `Autor sprawozdania:`, zaś treść elementu została podana w sposób dosłowny.



Rysunek 2.7. *Nośniki informacji w dokumencie XML*

Nazwy atrybutów często nie są wykorzystywane w sposób widoczny dla użytkownika końcowego, a jedynie stosowane są w procesie przetwarzania (do identyfikacji), np. znacznik:

```
<faktura rodzaj="korygująca" nr_faktury="12/2001" />
```

może być przekształcony do postaci:

```
Faktura korygująca numer 12/2001
```

Pominięto tutaj nazwę atrybutu `rodzaj`, a użyto jedynie jego wartości (`korygująca`).

Autor dokumentu XML musi określić, w którym miejscu umieszczać odpowiednie informacje, jak nazywać elementy i atrybuty, jaką strukturę nadać dokumentowi. Jest to bardzo ważny etap stanowiący **modelowanie rzeczywistości**. Dokonuje się tego zwykle podczas określania typu dokumentu (przy okazji tworzenia DTD) lub przy tworzeniu schematów XML Schema. W następnych rozdziałach przedstawione zostaną niektóre problemy związane z modelowaniem.

2.11. Konwersja dokumentów HTML do XML

Obecnie większość serwisów informacyjnych dostępnych w systemie WWW jest przechowywana w postaci dokumentów utworzonych w języku HTML. Wydaje się, że w przyszłości wiele z nich będzie przetwarzanych do postaci dokumentów XML. Proces takiego przekształcania można opisać za pomocą następujących kroków:

1. Zadbanie o to, by dokument był dobrze uformowany (poprawny składniowo), co nie zawsze ma miejsce w przypadku stron HTML.
2. Przekonwertowanie do języka XHTML. W tym kroku zwykle wystarcza, że dokument jest dobrze uformowany, aczkolwiek język XHTML daje więcej możliwości (np. poprzez modularyzację).

3. Zaprojektowanie struktury informacji zawartej w dokumencie, np. poprzez zdefiniowanie DTD, nawet jeżeli nie powstaną formalne deklaracje DTD. Ustalenie nazw i zawartości elementów, ich atrybutów, jak również struktury elementów (określenie hierarchii elementów).
4. Przeniesienie tekstów występujących w dokumencie HTML do zaprojektowanych elementów XML (i atrybutów). Uzyskujemy w ten sposób dokument XML odpowiadający początkowemu dokumentowi HTML.
5. Wydzielenie elementów związanych z wyświetlaniem dokumentu. Utworzenie transformacji XSLT umożliwiających przejście do odpowiedniego formatu prezentacyjnego dokumentu, np. HTML. Ewentualne utworzenie dokumentu arkuszy stylów XSL zawierających przetworzony dokument źródłowy.

Autorzy stron w języku HTML powinni już obecnie starać się o to, by ich dokumenty były dobrze uformowane (patrz: punkt 2.7).

W poprzednim rozdziale na rysunku 1.1 przedstawiono dokument HTML zawierający przykładowe sprawozdanie, umieszczone w wewnętrznym intranecie pewnej korporacji. Stosując powyższe kroki (na razie bez określania definicji DTD oraz arkuszy stylów) można ten dokument przekształcić w dokument XML. Mógłby on wyglądać następująco (plik *Sprawozdanie.xml*):

```
<?xml version='1.0' ?>
<sprawozdanie rodzaj='roczne'>
  <autorzy liczba_autorow='2'>
    <autor>
      <imie> Anna </imie>
      <nazwisko> Nowicka </nazwisko>
    </autor>
    <autor miejsce='drugi'>
      <imie> Jan </imie>
      <nazwisko> Kowalski </nazwisko>
    </autor>
  </autorzy>
  <tytul> Zestawienie zysków / obrotów </tytul>
  <dane>
    <pozycja waluta='PLN' okres='kwartal' typ_danych='zysk'>
      <nr_okresu> I </nr_okresu>
      <wartosc> 20000.00 </wartosc>
    </pozycja>
    <pozycja okres='kwartal' typ_danych='zysk'>
      <nr_okresu> II </nr_okresu>
      <wartosc> 22000.00 </wartosc>
    </pozycja>
    <pozycja okres='miesiac' typ_danych='obrot' waluta='PLN'>
      <nr_okresu> 7 </nr_okresu>
      <wartosc> 5000.00 </wartosc>
    </pozycja>
  </dane>
</sprawozdanie>
```

Deklaracje typu dokumentu DTD oraz schemat XML Schema dla powyższego dokumentu zamieszczono w dalszej części książki, w punkcie 4.19.