Kazienko P., Zgrzywa M.: *Języki wyszukiwania w dokumentach XML*.
MiSSI 2002. Multimedialne i Sieciowe Systemy Informacyjne. Materiały konferencyjne pod red. Cz. Daniłowicza. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2002, s. 391-400

XML, XPath, język wyszukiwania, XPointer, XQuery, dokument XML, XML Schema, SQL

Przemysław KAZIENKO*
Michał ZGRZYWA

JĘZYKI WYSZUKIWANIA W DOKUMENTACH XML

W referacie zaprezentowano najistotniejsze aktualnie zagadnienia związane z językami wyszukiwania w dokumentach XML. Obecnie głównym językiem wykorzystywanym do wyszukiwania jest XPath. Przedstawiono najważniejsze jego cechy: adresowanie względne i bezwzględne, mechanizmy poruszania się po drzewie węzłów dokumentu, testowanie węzłów (także testowanie wielokrotne), funkcje, połączenia kroków w ścieżki. XPath jest obecnie szeroko wykorzystywany m.in. w bazach danych XML, transformacjach XSLT, schematach XML Schema oraz odsyłaczach hipertekstowych XLink. W pracy wskazano także na najważniejsze wady i ograniczenia tego języka, jak również dokonano jego porównania z językiem SQL.

Prace nad językiem XPath są kontynuowane. W referacie zaprezentowano główne postulaty sformułowane dla języka XPath 2.0 oraz języka XQuery, w którym kolekcja dokumentów XML jest traktowana jak baza danych.

Innym kierunkiem rozwoju XPath jest nie zatwierdzony język XPointer, który rozszerza możliwości wyszukiwania, przede wszystkim dla potrzeb wskazywania.

1. WSTĘP

Od kilku lat w informatyce obserwujemy szybki rozwój technologii związanych z tekstowym, rozszerzalnym językiem znaczników XML (ang. *eXtended Markup Language*) [8]. Są one wspierane przez największe firmy produkujące oprogramowanie takie jak IBM, Microsoft, Sun, co dodatkowo przyśpiesza ten rozwój. Podstawą dokumentu XML są elementy ograniczone znacznikami (stąd jego nazwa).

Jedną z najważniejszych cech języka XML jest jego elastyczność przejawiająca się w łatwości tworzenia i modyfikacji struktury dokumentów. Powstaje jednak wtedy zasadniczy problem: w jaki sposób docierać do informacji zapisywanych w tak zróżnicowanych strukturach. Rozwiązaniem było opracowanie odpowiednich języków wyszukiwania. Działaniom tym patronowała międzynarodowa organizacja World Wide Web Consortium odpowiedzialna za rozwój systemu WWW.

* Politechnika Wrocławska, Zakład Systemów Informacyjnych, Wybrzeże Wyspiańskiego 27, 50 370 Wrocław, (e-mail: kazienko@pwr.wroc.pl, mzgrzywa@zsi.pwr.wroc.pl).

Kazienko P., Zgrzywa M.: *Języki wyszukiwania w dokumentach XML*.
MiSSI 2002. Multimedialne i Sieciowe Systemy Informacyjne. Materiały konferencyjne pod red. Cz. Daniłowicza. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2002, s. 391-400

2. JĘZYK XPath

2.1 PODSTAWOWE CECHY JĘZYKA XPath

Pierwszą próbą opracowania ujednoliconego standardu adresowania elementów dokumentu XML był język XPath 1.0 (ang. *XML Path Language*). Uzyskał on rekomendację[1] WWW Consortium już w listopadzie 1999 roku [2].

Za pomocą wyrażeń XPath można wskazywać lub wyszukiwać fragmenty dokumentów XML [5]. Wyrażenia te zwracają wartości jednego z czterech typów: zbiór węzłów **(**ang. *node-set*), wartość logiczna (ang. *boolean*), liczba (ang. *number*) i łańcuch znaków (ang. *string*). Dokument XML to hierarchia – drzewo elementów (węzłów) z jednym korzeniem. Wyrażenia XPath mogą się odnosić do węzłów drzewa w sposób bezwzględny (jako ścieżka od korzenia) lub względny — od aktualnego elementu. Przejścia pomiędzy elementami (zmiana aktualnego węzła) są dokonywane przez aplikację (lub inny język), który wykorzystuje XPath, np. transformacje XSLT.

Działanie języka XPath najłatwiej zaprezentować na przykładach. Weźmy prosty dokument XML, zawierający informacje o częściach samochodowych[2]. Dla ułatwienia analizy ponumerowano wiersze. Numery te nie są częścią dokumentu:

```
 1. <?xml version="1.0" encoding="UTF-8" ?>
 2. <Katalog Data="2002-05-01">
 3.   <NazwaKat>Katalog części</NazwaKat>
 4.   <Podkatalogi>
 5.     <Podkatalog Symbol="SLN24">
 6.       <NazwaPodkat>Silniki</NazwaPodkat>
 7.       <Podkatalog Symbol="SB1">
 8.         <NazwaPodkat>Części Silników Benz.</NazwaPodkat>
 9.         <Czesc Symbol="AD14">
10.           <NazwaCzesci>Rozdzielacz paliwa</NazwaCzesci>
11.           <Cena Waluta="PLN">75</Cena>
12.           <Gwarancja>0</Gwarancja>
13.           <ModelCzesci IdModel="Kad1.6" />
14.           <ModelCzesci IdModel="Vec2.0" />
15.         </Czesc>
16.         <Czesc Symbol="AD15">
17.           <NazwaCzesci>Wężyk wtryskowy paliwa</NazwaCzesci>
18.           <Cena Waluta="PLN">20</Cena>
19.           <Gwarancja>0</Gwarancja>
20.           <ModelCzesci IdModel="Vec2.0" />
21.         </Czesc>
22.       </Podkatalog>
23.       <Czesc Symbol="A111">
```

[1] Rekomendacja WWW Consortium jest de facto uznaniem języka jako standard.
[2] Przykład z modyfikacjami zaczerpnięto z projektu [7].

```
24.          <NazwaCzesci>Silnik benzynowy 1.6</NazwaCzesci>
25.          <Cena Waluta="PLN">2400</Cena>
26.          <Gwarancja>24</Gwarancja>
27.          <ModelCzesci IdModel="Kad1.4" />
28.        </Czesc>
29.      </Podkatalog>
30.    </Podkatalogi>
31.    <Modele>
32.      <Model Symbol="Kad1.4">
33.        <NazwaModelu>Opel Kadet 1.4</NazwaModelu>
34.        <Cena>17000</Cena>
35.      </Model>
36.      <Model Symbol="Kad1.6">
37.        <NazwaModelu>Opel Kadet 1.6</NazwaModelu>
38.        <Cena>23000</Cena>
39.      </Model>
40.      <Model Symbol="Vec2.0">
41.        <NazwaModelu>Opel Vectra 2.0</NazwaModelu>
42.        <Cena>32000</Cena>
43.      </Model>
44.    </Modele>
45. </Katalog>
```

Język XPath udostępnia szereg słów wskazujących względną lub bezwzględną pozycję w drzewie dokumentu (ang. *axes*): / — korzeń czyli odesłanie bezwzględne, `child` (dziecko), `parent` (ojciec), `self` (aktualny węzeł), `descendant` (potomek), `descendant-or-self` (potomek lub aktualny węzeł), `following` (następny węzeł w dokumencie[3]), `following-sibiling` (następny węzeł będący rodzeństwem, czyli dzieckiem tego samego rodzica), `preceding` (poprzedni węzeł w dokumencie), `preceding-sibiling` (poprzedni węzeł będący rodzeństwem), `ancestor` (przodek), `ancestor-or-self` (przodek lub aktualny węzeł), `attribute` (atrybut), `namespace` (przestrzeń nazw aktualnego węzła).

Do wymienionych słów można dodać bliższy opis szukanego fragmentu w formacie:

axis::nazwa.

Przykładowo, wyrażenie XPath: `/child::Katalog` zwróci[4] jednoelementowy zbiór węzłów zawierający cały element `Katalog`, czyli cały dokument (wiersze od 2 do 45). Pytanie: `/descendant-or-self::Czesc` da w wyniku zbiór węzłów z wszystkimi częściami (wiersze od 9 do 21 oraz od 23 do 28), natomiast `attribute::Data` zwróci łańcuch znaków — wartość atrybutu `Data` dla aktualnego węzła (jeśli węzłem tym będzie `Katalog`, otrzymamy: `2002-05-01`). Jeżeli przyjrzymy się powyższym wyrażeniom, widać, że umożliwiają one wyszukiwanie węzłów dokumentu XML poprzez nazwy elementów lub atrybutów.

[3] Każdy kolejny element, który w całości znajduje się po aktualnym elemencie

[4] Do zadawania pytań i weryfikacji odpowiedzi użyto procesora transformacji XSLT, który oczywiście obsługuje także język XPath. Jest to XP autorstwa Jamesa Clarka – współtwórcy zarówno XPath jak i XSLT.

Dla najczęściej używanych wyrażeń istnieją skróty: słowo `child::` jest domyślne i może być pominięte, `attribute::` odpowiada znakowi `@`, `self` to `.` (kropka), `parent` to `..` (dwie kropki), a `descendant-or-self` to `//` . Korzystając ze skróconej formy powyższe pytania można zapisać w postaci: `/Katalog`, `//Czesc` oraz `@Data`.

Do wyrażeń można dołączyć tzw. test węzła (ang. *node test*), za pomocą którego zapisuje się warunki ograniczające zwracane węzły. Test jest umieszczany w nawiasach kwadratowych. Można w nim stosować wszystkie przedstawione już wyrażenia (np. `child::`, `attribute::`, itd. oraz ich formy skrócone) a także operatory logiczne: `and` i `or` oraz szereg funkcji specyficznych dla różnych typów danych. Przykładowo, pytanie:

```
//Czesc[Gwarancja = 0 and Cena &lt; 100]
```

zwróci część o symbolach `AD14` i `AD15` (wiersze od 9 do 21). `Gwarancja = 0` jest tutaj skróconą wersją: `child:: Gwarancja = 0`.

W wyrażeniach testowych można korzystać z zestawu typowych operatorów: `=`, `!=` (nie równy), `>`, `>=`, `<` (mniejszy), `<=` (mniejszy równy), `and`, `or`, `+`, `-`, `*` (mnożenie), `div` (dzielenie), `mod` (modulo). Znak `<` zastępuje znak mniejszości <, ponieważ zostałby on zinterpretowany jako początek znacznika.

Dla każdego typu danych udostępnione zostały pewne funkcje zaś ich pełną listę zawiera m.in. [2, 9]. Dla zbioru węzłów mamy między innymi funkcję `position()` zwracającą aktualną pozycją w zbiorze (kolejny numer węzła jako dziecka swojego rodzica), `last()` podającą liczbę węzłów czy `id()` zwracającą węzeł posiadający atrybut typu ID o zadanej wartości. Dla wartości logicznych istnieją funkcje: `true()`, `false()`, `not()` czy przekształcająca liczby i ciągi na wartości logiczne `boolean()`. Liczby można przetwarzać za pomocą m.in. funkcji `sum()` i `round()`. Zbiór funkcji dla łańcuchów znaków jest największy i zawiera m.in. `concat()` — połączenie dwóch lub więcej łańcuchów, `contains()` sprawdzająca istnienie zadanego ciągu, `substring()` "wycinająca" odpowiedni podciąg czy `string-lengt()` zwracająca długość ciągu. Przykładowo, pytanie `//Czesc[position()=1]` zwróci wszystkie części, które wystąpią pierwsze w swoich podkatalogach (w naszym przykładzie wiersze od 9 do 15 oraz od 23 do 28), zaś `//Model[contains(NazwaModelu,'ectra')]` zwróci model zawierający w treści podelementu `NazwaModelu` ciąg „`ectra`”, czyli linie od 40 do 43.

Wyrażenia (nazywane krokami) można ze sobą łączyć, tworząc tzw. ścieżkę (ang. *path*) do węzła. Stąd nazwa języka — *XML Path Language*. Węzły zwracane przez jeden krok są wejściowe dla kroku kolejnego. Dzięki temu, można tworzyć złożone pytania, np.:

```
/Katalog/Podkatalogi//Podkatalog[@Symbol = "SB1"]/
Czesc [position()!=last() and ModelCzesci/@IdModel="Kad1.6"]
```

Odpowiedzią na to pytanie jest zbiór nieostatnich części (dzieci) z podkatalogów o symbolu SB1 przeznaczonych dla modelu `Kad1.6` (w naszym przykładzie są to linie od 9 do 15).

Poprawne jest również wielokrotne użycie testu. W kolejnym teście filtrowany jest wynik poprzedniego testu. Przykładowo, chcąc uzyskać jedną, pierwszą część tańszą niż 50 PLN (czyli w naszym dokumencie `AD15`) należy zadać pytanie: `//Czesc[number(Cena)<50][position()=1]`. Natomiast, zadając pytanie

`//Czesc[number(Cena)<50 and position()=1]` otrzymamy w wyniku wszystkie pierwsze części (pierwsze dzieci rodzica), o ile są one tańsze niż 100 zł (w naszym przypadku wynik będzie pusty).

2.2 XPath A SQL

W ciągu ostatnich dwóch lat dynamicznie rozwijane są bazy danych XML – XDBMS (ang. *XML Database Management System*), w których językiem manipulacji danych (aktualizacja, dodawanie, usuwanie) jest DOM API[5]. Przykładami takich komercyjnych baz są: X-Hive/DB firmy X-Hive Corporation, Tamino firmy Software AG, TEXTML Server firmy IXIA czy Infonyte DB firmy Infonyte GmbH. Można więc zastanowić się nad porównaniem języka XPath (używanego do budowania pytań i wskazywania węzłów w takich bazach) z językiem SQL (używanym do tworzenia pytań w bazach relacyjnych).

Najpierw zwrócić uwagę na różnicę modeli danych pomiędzy systemami relacyjnymi a opartymi na języku XML [1]. Baza relacyjna składa się z wielu osobnych tabel o ściśle określonej, sztywnej strukturze, połączonych więzami relacyjnymi. Odpowiednikiem takiej bazy w modelu XDBMS może być pojedynczy dokument XML będący hierarchią elementów. Nie ma w nim relacji a jedynie mogą istnieć odsyłacze[6] — łączniki między elementami. Informacja, którą niosą ze sobą relacje, w dokumentach XML jest zawarta w odmiennej postaci: w zależnościach hierarchicznych pomiędzy elementami.

Oba te modele mają jednak wiele cech wspólnych. Dotyczy to także ich języków wyszukiwania: SQL [12] oraz XPath. Zwróćmy uwagę na możliwość zagnieżdżania pytań języka SQL:

```
SELECT FROM (SELECT FROM ...) WHERE (SELECT FROM ...) = ...
```

W języku XPath powyższa struktura odpowiada kolejnym krokom w ścieżce. Wynik wyznaczony w pierwszym kroku jest ograniczany przez kolejne. Łatwo również dostrzec podobieństwo sekcji `WHERE` do testów występujących w języku XPath. W obu językach używa się ich do filtrowania otrzymanych wyników (w języku SQL dodatkowo określa się warunki złączeń).

Jednocześnie poprzez specyficzne otoczenie w którym działają oba języki, każdy z nich wykształcił swoje niepowtarzalne cechy. Przykładowo, język SQL, używany w ściśle zdefiniowanym środowisku szczegółowo opisanych relacji, nie pozwala na wyszukanie krotki gdy znamy jedynie wartość klucza. Konieczne jest dokładne wskazanie tabeli (relacji) oraz nazwy kolumny (atrybutu). W języku XPath zdefiniowano do tego celu funkcję `id()`, w której nie jest konieczna znajomość nazwy elementu (tutaj odpowiednik relacji) ani atrybutu identyfikatora (odpowiednik klucza).

Podobnie w SQL nie można definiować warunków dla wszystkich atrybutów relacji jednocześnie (należy podać wszystkie kolumny po kolei). Natomiast w języku XPath mamy

---

[5] DOM to obiektowy model dokumentu (ang. *Document Object Model*), w którym dokument traktowany jest jako drzewo węzłów. DOM API to interfejs umożliwiający dostęp do dokumentów XML z poziomu aplikacji, wykorzystujący standard DOM.

[6] Chodzi o atrybuty typu `IDREF` określone przez definicje typu dokumentu DTD, elementy `keyref` zawarte w schematach XML Schema oraz odsyłacze języka XLink [6].

do dyspozycji * (gwiazdkę). Przykładowo, pytanie:

```
//*[contains(text(),'24') or contains(@*,'24')]
```

zwróci nam wszystkie elementy (pierwsza gwiazdka) zawierające w treści elementu lub jakimkolwiek jego atrybucie (druga gwiazdka) wystąpienie ciągu „24", czyli cały `Podkatalog` o symbolu `SLN24` (wiersze od 5 do 29), element `Cena` (wiersz 25) oraz element `Gwarancja` (wiersz 26).

Dokumenty XML są umieszczone w jednym pliku a elementy zazwyczaj zawierają wszystkie związane z nimi informacje w swoich podelementach. Z tego powodu w języku XPath nie zdefiniowano mechanizmów służących złączeniom. Można się jednak odwoływać do innych części tego samego dokumentu. Odpowiada to odwołaniom do innych tabel w modelu relacyjnym. Przykładowo, chcąc poznać nazwy wszystkich części pasujących do modelu o nazwie „Opel Vectra 2.0" (czyli wiersze 10 i 17) musimy zadać pytanie:

```
//Czesc[ModelCzesci/@IdModel =
         (//Model[NazwaModelu="Opel Vectra 2.0"]/@Symbol)]
 /NazwaCzesci
```

Nie da się jednak uzyskać wyniku zawierającego strukturę powstałą z połączenia dwóch odrębnych poddrzew dokumentu XML. Pozostając przy naszym katalogu, chcielibyśmy przykładowo otrzymać listę części wraz z nazwami modeli do których pasują. Wymagałoby to włączenia elementu `NazwaModelu` (część poddrzewa `Modele`) jako podelementu `Czesc` (część poddrzewa `Podkatalogi`).

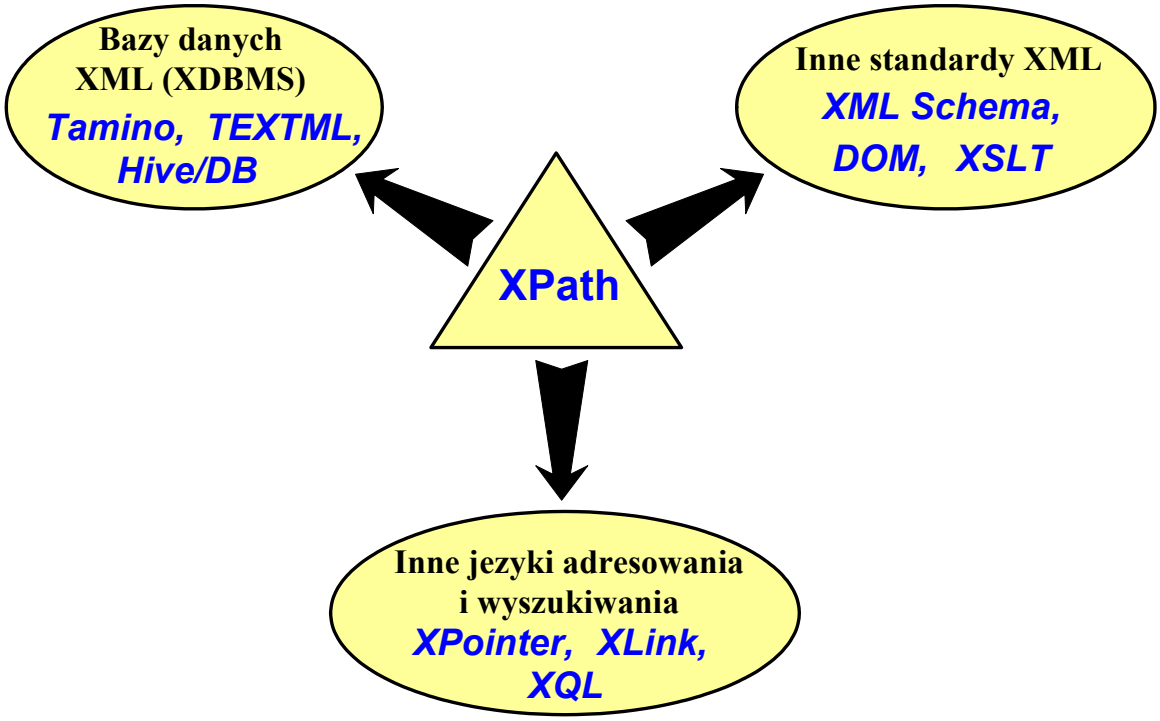
Kolejną wadą języka XPath jest brak funkcji sortujących zwracany wynik. W języku SQL służy do tego `ORDER BY`. Ponieważ zwracany przez wyrażenia XPath zbiór węzłów jest zawsze nieuporządkowany, nie istnieje także możliwość grupowania. W SQL używa się do tego wyrażenia `GROUP BY ... USING ....` Jedyną dostępną operacją przetwarzania zbioru węzłów w XPath jest sumowanie wszystkich liczb z wyniku — funkcja sum().

Mogło by się wydawać, że język XPath jest o wiele uboższy od języka SQL. Trzeba jednak pamiętać o różnicach pomiędzy relacyjnym a hierarchicznym modelem danych. W bazach danych XML często stosuje się dodatkowe (niestandardowe) mechanizmy ułatwiające przetwarzanie struktur XML, przykładowo umożliwiające złączenia. Wiele braków, np.: w sortowaniu i grupowaniu, jest niwelowanych przez transformacje XSLT.

2.3 ZASTOSOWANIA XPath

Można wyróżnić trzy zasadnicze obszary wykorzystania języka XPath (rys. 1). Pierwszy to standardy, w których niezbędne są mechanizmy umożliwiające wskazywanie poszczególnych fragmentów dokumentu XML, czyli przede wszystkim język transformacji XSLT, DOM (dopiero w niezatwierdzonej jeszcze wersji Level 3) i XML Schema (wskazania elementów kluczowych i unikatowych). Kolejną dziedziną zastosowań XPath są języki służące do adresowania dokumentów XML oraz ich części: XPointer oraz XLink.

XPath odgrywa także zasadniczą rolę w bazach danych XML. Jego wyrażenia służą tam do wyszukiwania danych [13]. W konkretnych systemach XDBMS zaimplementowany jest zwykle jedynie pewien podzbiór najważniejszych wyrażeń XPath.



Rys. 1. Zastosowania języka XPath

2.4 WADY XPATH

XPath jest prostym językiem, co oczywiście ogranicza jego możliwości. Jest to przyczyną dalszych poszukiwań i rozwoju języków wyszukiwawczych. Wskazuje się na szereg wad tego języka.

Ponieważ po zatwierdzeniu XPath powstały kolejne standardy, więc język ten nie mógł uwzględniać ich specyfiki. W konsekwencji nie uwzględnia on wbudowanych typów danych zaproponowanych dla schematów XML Schema [4] a zatwierdzonych w 2001 r. Jak wspomniano wyżej, obsługuje on łańcuchy znaków, liczby, wartości logiczne i zbiory węzłów. Schematy udostępniają wiele więcej typów. Pozwalają także na definiowanie własnych typów prostych i złożonych. Najistotniejszą niedogodnością wydaje się brak typów dla liczb rzeczywistych (*float*, *double*) oraz daty i czasu (*datetime*, *duration*, *date*, ...). Oczywiście część problemów da się rozwiązać drogą okrężną, stosując dostępne w XPath mechanizmy. Przykładowo, jeśli szukamy katalogu z datą późniejszą niż założona (2002-01-01), to znając format jej zapisu w dokumencie możemy zadać pytanie:

```
/Katalog[ (number (substring (@Data,1,4)) > 2002) or
          ((number (substring (@Data,1,4)) = 2002) and
           (number (substring-before (substring-after
                                (@Data,'/'),'/')) > 1)) or
          ((number (substring (@Data,1,4)) = 2002) and
           (number (substring-before (substring-after
                                (@Data,'/'),'/')) = 1) and
           (number (substring-after (substring-after
                                 (@Data,'/'),'/')) > 1))]
```

Jak widać pytanie jest bardzo złożone, chciało by się móc zastosować pojedynczy znak większości. Jednocześnie proszę sobie wyobrazić jak skomplikowane byłoby wyrażenie sprawdzające czy data znajduje się w zadanym przedziale czasu lub jeżeli dopuszczalnych byłoby kilka formatów.

Kolejną niedogodnością języka XPath jest mała liczba dostępnych funkcji. Dla liczb użyteczna byłaby wartość bezwzględna czy też potęgowanie. Dla ciągów znaków przydatne są: zamiana na małe bądź duże litery (wartościowa przy wyszukiwaniu ciągów, gdy nieważna bądź nieznana jest postać źródłowa dokumentu XML) a przede wszystkim zamiana ciągów (funkcja `translate()` zamienia pojedyncze znaki, ale nie całe ciągi).

W wielu przypadkach funkcjonalność brakujących operacji można uzyskać przez odpowiednie zastosowanie dostępnych wyrażeń. Efekty są jednak zazwyczaj długie i nieczytelne. Przykładowo, funkcję zamieniającą wszystkie małe litery w ciągu na wielkie można zastąpić przez wyrażenie:

```
translate (ciag, 'aąbcćdeęfghijklłmnoópqrsśtuvwxyzźż',
                 'AĄBCĆDEĘFGHIJKLŁMNOÓPQRSŚTUVWXYZŹŻ')
```

Inny przykład to wyrażenie zamiany ciągów znaków. Chcąc zamienić w opisie modelu „Opel Kadet 1.6” ciąg „Kadet” na „Vectra”, należało by użyć:

```
concat (substring-before ('Kadet', 'Opel Kadet 1.6'),
        'Vectra',
        substring-after ('Kadet', 'Opel Kadet 1.6')
```

Działa ono jednak jedynie dla pierwszego wystąpienia ciągu źródłowego.

Kolejnym istotnym ograniczeniem języka XPath jest zestaw wymienionych wyżej czterech typów wartości zwracanych przez wyrażenia. Skutkiem tego za pomocą XPath nie można uzyskać:

- fragmentów dokumentu nie będących węzłem, np. całego znacznika początkowego,
- wskazania na fragment zawartości elementu lub atrybutu; wyrażenia mogą zwrócić szukany ciąg znaków lub cały węzeł, ale nie jego fragment jako węzeł,
- ciągu znaków z węzła; wyrażenie może zwrócić wartość elementu lub cały element, ale nie cały element (ze znacznikami) jako ciąg znaków,
- wskazania na punkt, np. za znacznikiem początkowym lub po piątym znaku.

Nie można także wyszukiwać po tekstach rozciągających na kilka elementów. Ma to szczególne znaczenie w dokumentach, w których szeroko korzysta się z mieszanej zawartości elementu (tekst wraz z podelementami). Zawartość taka jest przydatna we wszelkiego rodzaju sprawozdaniach, artykułach, zgłoszeniach, dokumentacjach, itp.

3. JĘZYK XPointer

Język wskazań XPointer — nie będący jeszcze zatwierdzonym standardem [3] — jest rozszerzeniem języka XPath, tzn. wyrażenia XPath można także wykorzystywać w wyrażeniach XPointer [9]. Za pomocą XPointer wskazuje się na punkty (zbiory punktów) oraz zakresy (zbiory zakresów). Zakres jest obszarem zawartym pomiędzy dwoma punktami. Można go uzyskać podając te punkty wprost, wyszukując (np. ciąg znaków) lub uzyskując z węzła XPath. Dzięki temu można wskazywać na niemal dowolne fragmenty

dokumentu XML. Dodatkowo, XPointer udostępnia wyszukiwanie ciągów znaków z pominięciem znaczników. Przykładowo wyrażenie:

```
xpointer(string-range(//, "200"))
```

wskaże na dwa miejsca: pierwsze zacznie się w wierszu 18 a skończy w 19 (przechodzi ono przez znaczniki z dwóch różnych elementów), zaś drugie będzie zawierać fragment wiersza 42.

4. JĘZYKI XPath 2.0 ORAZ XQuery 1.0

Mając na uwadze wyżej wymienione niedogodności języka XPath, WWW Consortium wciąż prowadzi prace na nowymi językami wyszukiwania. Aktualnie w zaawansowanym stadium są języki XPath 2.0 i XQuery 1.0 (ang. *XML Query Language*). Oba są ze sobą bardzo powiązane, posiadają między innymi wspólny model danych oraz funkcje i operatory. Z pewnym przybliżeniem można stwierdzić, że XPath 2.0 jest uproszoną wersją XQuery. Jest to zrozumiałe, jeżeli weźmie się pod uwagę fakt, że pierwszy z nich jest przeznaczony dla szerszego grona użytkowników — będzie stosowany między innymi w XSLT 2.0, kolejnej wersji języka transformacji.

Najważniejszą zmianą wprowadzoną do XPath 2.0 jest obsługa typów z XML Schema [10]. Pozwala to odnosić się do danych w sposób odpowiadający ich specyfice — z uwzględnieniem ich typów. Dzięki temu można porównywać i dokonywać operacji na datach, przetwarzać identyfikatory URI (czyli także rozpoznawać wybrane fragmenty adresów URL), itd. Kolejną istotną zmianą jest wykorzystanie tzw. sekwencji zamiast dotychczasowych zbiorów węzłów. W sekwencjach uwzględniona jest kolejność węzłów oraz dopuszcza się ich powtórzenia. Otwiera to drogę do porządkowania zbioru odpowiedzi. Wprowadzono również wyrażenia warunkowe: `if then else` oraz operator pętli `for` do przetwarzania elementów sekwencji. Zestaw funkcji został znacznie poszerzony, co ułatwiło wykonywanie operacji na danych, dodano np.: `replace()` czy `to-upper()` dla ciągów znaków. Oczywiście XPath 2.0 spełnia wymagania wstecznej kompatybilności.

Język XQuery [11] ma za zadanie umożliwiać formułowanie nowych klas zapytań. Dozwolone jest w nim definiowanie własnych funkcji czy też konstruowanie nowych elementów. Dostępne są dodatkowe operacje sortowania, konwersji pomiędzy typami danych, przetwarzania sekwencji, itd. Możliwe jest także tworzenie pytań rekurencyjnych.

Zarówno w XPath 2.0 jak i w XQuery można dostrzec coraz więcej cech charakterystycznych dla języka transformacji XSLT.

Należy się spodziewać, że ze względu na szerszy obszar zastosowań język XPath 2.0 szybciej doczeka się statusu Rekomendacji WWW Consortium.

5. PODSUMOWANIE

Dynamiczny rozwój zastosowań języka XML, jak również innych standardów z nim związanych, wymusza ciągłe ulepszanie mechanizmów dostępu do informacji zawartych w dokumentach XML. Im więcej przetwarza się informacji i im bardziej złożone wykorzystuje się narzędzia, tym lepszych wymaga to mechanizmów wyszukiwania. Ma to szczególne

znaczenie wtedy, gdy użytkownik interaktywnie wpływa na zawartość i postać prezentowanych treści. Dodatkowe wymagania stawiają bazy danych XML, w których konieczna jest formalna spójność języka oraz mechanizmy złączeń nie występujące w przypadku pojedynczych dokumentów XML. Pożądane jest również wspomaganie dla narzędzi ułatwiających manipulowanie danymi. Ze względu na heterogeniczność wykorzystywanych obecnie baz danych przydatna jest pewna integracja z innymi językami wyszukiwania (np. SQL).

Rozwój języków wyszukiwania, będących podstawą każdej aplikacji przetwarzającej zewnętrzne dane, jest równocześnie motorem rozwoju wszystkich dziedzin z nim związanych.

LITERATURA

[1] ABITEBOUL S., BUNEMAN P., SUCIU D., *Dane w sieci WWW*, Mikom, Warszawa, 2001
[2] CLARK J., DEROSE S. (eds.), *XML Path Language (XPath). Version 1.0. W3C Recommendation 16 November 1999*, WWW Consortium, 1999, http://www.w3.org/TR/xpath.
[3] DEROSE S., MALER E., DANIEL R. Jr (eds.), *XML Pointer Language (XPointer) Version 1.0. W3C Candidate Recommendation 11 September* 2001, WWW Consortium, 2001, http://www.w3.org/TR/xptr/
[4] FALLSIDE D.C. (ed.), *XML Schema Part 0: Primer. W3C Recommendation, 2 May 2001*, WWW Consortium, 1999, http://www.w3.org/TR/xmlschema-0/
[5] GWIAZDA K., *Nawigacja i wyszukiwanie w XPath*, Software 2.0 nr 6 (90) czerwiec 2002, s. 28-32.
[6] GWIAZDA K., KAZIENKO P, *XLink - the future of document linking*, Information Systems Architecture and Technology ISAT 2001. Conference proceedings, Wroclaw University of Technology, 2001, pp. 132-139. ftp://ftp.zsi.pwr.wroc.pl/publications/Kazienko/ISAT2001/XLink.ps
[7] KAROLEWSKI P., WOŹNIAK P., ZGRZYWA M., *X-Service. System wspomagania sieci serwisów*, Akademia Tamino - projekt. Politechnika Wrocławska, Zakład Systemów Informacyjnych, 2002.
[8] KAZIENKO P., *Rodzina języków XML*, Software 2.0 nr 6 (90) czerwiec 2002, s. 21-27.
[9] KAZIENKO P., GWIAZDA K, *XML na poważnie*, Helion, Gliwice, 2002, http://helion.pl/ksiazki/xmlnap.htm
[10] LENZ E., *What's new in XPath 2.0*, XML.COM, 2002, http://www.xml.com/pub/a/2002/03/20/xpath2.html
[11] MARCHIORI M., *XML Query*, WWW Consortium, 2002, http://www.w3.org/XML/Query
[12] PANKOWSKI T., *Podstawy baz danych*, Wydawnictwo Naukowe PWN, Warszawa, 1992.
[13] *Tamino XML Server. White Paper*. Software AG, October 2001.

INFORMATION RETRIEVAL LANGUAGES FOR XML DOCUMENTS

XML Path Language (XPath) is nowadays the most important language for information retrieval in XML documents. In the paper were described its main features: relative and absolute addressing with various axes, navigating throughout tree of nodes, node tests, functions and paths composed form steps. XPath is now widespread used e.g. in XML databases, XSL transformations, XML Schema, XLink.

Also XPath disadvantages and restrictions were pointed and SQL was compared with this language.

Research on retrieval languages for XML are kept carrying out. Primary postulates for XPath 2.0 and XQuery (where collections of XML files are accessed like databases) were presented. Another development area is XML Pointer Language (XPointer) useful especially in hypertext links.