UNIWERSYTET MIKOŁAJA KOPERNIKA
W TORUNIU
WYDZIAŁ MATEMATYKI I INFORMATYKI

KIERUNEK: Informatyka
SPECJALNOŚĆ: Programowanie i przetwarzanie informacji

PIOTR WICENTY

NATYWNY TYP XML W RELACYJNYCH BAZACH DANYCH

Praca magisterska
napisana pod kierunkiem
dr hab. Krzysztofa Stencla prof. UMK

Toruń 2009

Spis treści

Rozdział I
Wprowadzenie

1. Wprowadzenie do XML

1.1. Czym jest XML

Rozszerzalny język znaczników, czyli XML (Extensible Markup Language) to prosty, szybki i elastyczny format tekstowy wywodzący się z SGML (Standard Generalized Markup Language). XML jest otwartym, wolnym od opłat licencyjnych standardem rekomendowanym przez organizację World Wide Web Consortium (W3C). Mimo swojej nazwy nie językiem lecz raczej metajęzykiem – służącym do konstruowania innych języków oraz opisującym zasady ich tworzenia. Został zaprojektowany jako niezależny od platformy format przechowywania, wymiany oraz reprezentacji danych.

1.2. Rozwój i znaczenie formatu XML

Początki XML sięgają roku 1996. Wtedy to pojawiał się pierwsza, robocza wersja specyfikacji. Prace nad nią trwały jeszcze 2 lata, aż w końcu, 10 lutego 1998 roku XML w wersji 1.0 został oficjalnie zarekomendowany przez organizację W3C. Od tego czasu jego znaczenie stale rośnie - technologia XML stała się wszechobecna. Ze względu na swoją wszechstronność i zdolność wymiany danych między różnymi systemami, aplikacjami oraz urządzeniami znajduje zastosowanie w niemal każdej branży. XML dzięki swojej samo opisującej budowie oraz zdolności przechowywania danych o różnej strukturze stał się uniwersalnym standardem wymiany informacji.

Wraz z rozwojem Internetu nastąpił również wyraźny wzrost znaczenia XML. Stał się on podstawowym formatem wymiany bądź opisu danych dla wielu nowych języków, protokołów oraz technologii internetowych, takich jak AJAX, XHTML, RSS, XUL, SOAP, WSDL.

O tym jak elastycznym i uniwersalnym formatem danych jest XML może świadczyć choćby fakt pojawienia się w ostatnich latach bardzo wielu różnorodnych aplikacji języka XML. Niemal w każdej dziedzinie nauki znajdziemy aplikację opisującą właściwe dla niej dane. Wśród najpopularniejszych wymienia się VoiceXML, MathXML, XBRL, FpML, HL7 DockBook, BPEL, NewsML. Wzrost popularności XML spowodował, że na rynku pojawiło się wiele nowych narzędzi wspomagających pracę z tym formatem danych. Oprócz dużych pakietów narzędziowych takich jak Altova XML Spy, Liquid XML Studio czy Stylus Studio dostępnych jest także szereg darmowych programów pozwalających na różny sposób

przetwarzać dokumenty XML.

Szybko rosnąca popularność XML wymusiła od czołowych producentów systemów zarządzania bazami danych (SZDB) zaaplikowanie do swoich produktów mechanizmów wspierających ten język. Początkowo wsparcie to miało postać funkcji wbudowanych, później natomiast pojawił się natywny typ XML pozwalający na przechowywanie i przetwarzanie w relacyjnych bazach danych dokumentów XML.

1.3. Tradycyjne metody zarządzanie dokumentami XML

Zanim SZBD wzbogacone zostały o typ XML zarządzanie danymi w tym formacie było znacznie utrudnione. Brak natywnego typu oznacza konieczność przechowywania dokumentów poza relacyjną bazą danych. Można to robić na kilka sposobów – na przykład umieszczając je w systemie plików. Rozwiązanie to jednak cechuje szereg istotnych wad. Przede wszystkim jest mało efektywne, gdy mamy do czynienia z dużą liczbą dokumentów, nie daje możliwości równoległego dostępu do danych oraz wygodnego zarządzania ich bezpieczeństwem. Archiwizacja i odtwarzanie danych z systemu plików ...

Innym sposobem jest wykorzystanie natywnej bazy danych XML – przeznaczonej do przechowywania danych w tym formacie. W ostatnich latach pojawiło się wiele implementacji i bazy takie zdobywają coraz większą popularność. Zalet płynących z zastosowania takich rozwiązań jest kilka. Bazy XML-owe są zoptymalizowane pod obsługę danych o strukturze XML i dobrze radzą sobie z dużą liczbą dokumentów. Są jednak wciąż młode i w dużych projektach informatycznych często nie znajdują zastosowania ze względu na wątpliwą stabilność i małe w porównaniu z relacyjnymi bazami doświadczenie programistów w pracy z takimi systemami.

Jeśli potraktować dokument XML jak zwykły plik , wówczas jego przechowywanie nie stanowi nowego wyzwania dla bazy danych. Można go umieścić w tabeli z kolumną typu znakowego (VARCHAR, TEXT) bądź typu obiekt (BLOB, CLOB). Rozwiązanie jest właściwe i skuteczne tak długo jak zależy nam jedynie na przechowywaniu i odzyskiwaniu danych XML. Uniemożliwia jednak efektywne odpytywanie bazy danych o zawartość dokumentu, wyświetlanie jego fragmentów, konkretnych elementów czy atrybutów.

Innym dość często wykorzystywanym mechanizmem przechowywania danych XML jest ich dekompozycja i umieszczenie w relacyjnych strukturach bazy danych. Dokument jest dzielony na elementy oraz atrybuty a następnie umieszczany w wielu oddzielnych tabelach i

kolumnach. Odtwarzanie takiego dokumentu wymaga wówczas wielu złączeń tabel i dość skomplikowanych zapytań. Takie podejście pozbawia dodatkowo dane XML ich wrodzonej elastyczności i powoduje, że dokumenty stają się nieczytelne.

Wszystkie wymienione powyżej rozwiązania posiadają istotne wady. Dlatego też uzasadnione i konieczne stało się rozszerzenie relacyjnych systemów baz danych o nowy typ: natywny XML.

2. XBRL jako aplikacja XML

2.1. Czym jest XBRL

Extensible Business Reporting Language czyli XBRL jest standardem stworzonym dla potrzeb wymiany, interpretacji, analizy i prezentacji szeroko pojętych sprawozdań gospodarczych – zarówno finansowych jak i tych o charakterze niefinansowym. Podobnie jak XML jest otwarty, wolny od opłat licencyjnych oraz niezależny od systemu operacyjnego. Za rozwój formatu odpowiada XBRL International – międzynarodowa organizacja non-profit skupiająca ponad 500 firm, organizacji i agencji rządowych. To ona tworzy i rozwija specyfikację XBRL. Aktualny dokument – w wersji 2.1 można znaleźć na oficjalnej stronie internetowej organizacji[1].

Dokumenty XBRL wykorzystywane są głównie w bankowości, finansach. Standard wprowadzono, aby rozwiązać problem wielu rozmaitych formatów danych, w których tworzone były sprawozdania finansowe. XBRL jest już obowiązującym standardem w wielu krajach Europy oraz w USA. W Polsce został przyjęty m. in. przez Narodowy Bank Polski. W najbliższym czasie stanie się także obowiązującym na Giełdzie Papierów Wartościowych i w wielu innych instytucjach finansowych.

2.2. Budowa dokumentów XBRL

Technologię XBRL podzielić można na dwie rodziny dokumentów: taksonomie oraz instancje. Taksonomie XBRL to zbiór dokumentów XSD oraz XML opisujących i definiujących raportowane dane. Pliki XSD zawierają informację o nazwach raportowanych faktów, ich typie (prosty lub złożony) oraz typie raportowanych w ramach elementu danych (monetarne, całkowitoliczbowe, zmiennoprzecinkowe, tekstowe). Pliki XML zawierają natomiast definicje warstw (ang. XBRL linkbases) pozwalających ujmować dane w różnych kontekstach.
Specyfikacja XBRL 2.1 wyróżnia 5 warstw:

- warstwa etykiet (ang. label linkbase) – pozwala nadawać wiele etykiet dla tego samego elementu, dzięki czemu możliwa jest m. in. realizacja wielojęzyczności
- warstwa referencji (ang. reference linkbase) – pozwala łączyć elementy z

1 http://www.xbrl.org - oficjalna strona internetowa XBRL International

dodatkowymi, zewnętrznymi informacjami np. właściwymi organami, przepisami prawnymi dotyczącymi elementu

- warstwa prezentacji (ang. presentation linkbase) – określa sposób prezentacji danych użytkownikowi
- warstwa kalkulacji (ang. calculation linkbase) – wprowadza reguły matematyczne, pozwalające weryfikować poprawność danych
- warstwa definicji (ang. definition linkbase) – określa relacje między elementami np. jeden do wielu, wiele do wielu itp.

Taki zestaw dokumentów zapewnia jednoznaczną definicję i klasyfikację pojęć występujących w raportach – instancjach XBRL.

Przykład:

```
<link:loc xlink:type="locator" xlink:href="finrep.xsd#ifrs-gp_AssetsTotal" xlink:label="AssetsTotal"/>
<link:label xlink:type="resource" xlink:label="pl_AssetsTotal"  xml:lang="pl" id="pl_AssetsTotal">
Aktywa, Razem</link:label>
<link:labelArc xlink:type="arc"  xlink:from="AssetsTotal" xlink:to="pl_AssetsTotal"/>
```

Przykład 1: Fragment pliku warstwy etykiet, nadający elementowi o nazwie „AssetsTotal" polską etykietę - „Aktywa, Razem"

Instancja XBRL to bazujący na określonej taksonomii elektroniczny dokument zawierający dane. Fakty wykazane w instancji są raportowane w kontekstach, których dokładna definicja znajduje się w taksonomiach XBRL. Plik instancji dopiero z taksonomią opisującą raportowane fakty stanowi całość sprawozdania i tylko w obecności taksonomii możliwa jest walidacja jego poprawności.

Z reguły instancje XBRL zawierają od kilkuset do kilkudziesięciu tysięcy raportowanych faktów. Instytucje finansowe sprawujące kontrolę nad daną dziedziną gospodarki nierzadko muszą weryfikować sprawozdania napływający od wielu instytucji w tym samym czasie, stąd też poszukuje się wydajnych narzędzi do przetwarzania tego rodzaju dokumentów.

Przykład:

```
<context id="C1">
    <entity>
    <identifier scheme="http://sis.nbp.pl/bank132">132</identifier>
        <segment>
            <xbrldi:explicitMember dimension="Pozycje">KredytyLokaty</xbrldi:explicitMember>
            <xbrldi:explicitMember dimension="Waluty">Pln</xbrldi:explicitMember>
        </segment>
    </entity>
     <period>
        <instant>2009-12-31</instant>
    </period>
</context>
<finrep:Kredyty contextRef="C1" unitRef="PLN" decimals="0">18000</finrep:Kredyty>
<finrep:Lokaty contextRef="C1" unitRef="PLN" decimals="0">10000</finrep:Lokaty>
<finrep:Inne contextRef="C2" unitRef="PLN" decimals="0">120000</finrep:Inne>
```

Przykład 2: Fragment pliku instancji zawierający informacje o wysokości lokat i kredytów

2.3. Narzędzia obsługujące format XBRL

Rosnąca popularność standardu XBRL spowodowała, że na rynku zaczęło pojawić się co raz więcej narzędzi pomagających w pracy z tym formatem danych. Są to głównie aplikacje typu stand-alone nakierowane na obsługę XBRL oraz dodatki do większych pakietów narzędziowych XML oferujące dodatkowe funkcje użyteczne w procesie generowania i przetwarzania dokumentów XBRL. O tym jak znaczącą rolę odgrywa ten standard sprawozdawczości może świadczyć choćby fakt zaangażowania w rozrój narzędzi do jego wsparcia takich firma jak Fujitsu czy Microsoft. Obecnie na rynku dostępnych jest już kilkadziesiąt aplikacji umożliwiających zarówno tworzenie sprawozdań i taksonomii XBRL jak również pozwalających na przeglądanie ich zawartości oraz walidację poprawności.

3. Cel pracy

3.1. Porównanie możliwości poszczególnych systemów zarządzania bazami danych w zakresie wsparcia dla formatu XML

Głównym celem pracy jest porównanie możliwości poszczególnych systemów zarządzania bazami danych pod kątem obsługi standardów XML. Analizie w szczególności poddaną zostaną kwestie związane z przechowywaniem, przetwarzaniem, indeksowaniem, przeszukiwaniem oraz walidacją poprawności dokumentów XML. Oprócz wydajności SZBD w danym zakresie podjęta zostanie również próba nakreślenia zakresu w jakim dany system jest zgodny z obowiązującymi specyfikacjami. Praca traktować będzie o czterech różnych SZBD: IBM DB2, Microsoft SQL Server. Oracle Database oraz PostgerSQL.

3.2. Obsługa aplikacji XML po stronie bazy danych na przykładzie XBRL

XBRL oparty jest na wielu technologiach związanych z XML, m. in. na schematach XSD, języku arkuszy stylów XSL, języku ścieżek Xpath. Do pracy z dokumentami XBRL wykorzystuje się m. in. przekształcenia arkuszy stylów XSLT oraz język zapytań Xquery. To sprawia, że obsługa tego standardu wymaga dość skomplikowanych narzędzi i stanowi dobry przykład do sprawdzenia mechanizmów wsparcia dla formatu XML w relacyjnych bazach danych. Jednym z celów tej pracy jest odpowiedź na pytanie: na ile obsługę wybranej aplikacji XML można przenieść na stronę bazy danych korzystając z natywnego typu XML wbudowanego w SZBD. W wyniku tej części pracy powstanie szereg procedur składowanych realizujących wybrane operacje na dokumentach XBRL.

Rozdział II
XML w relacyjnych bazach danych

1. Systemy zarządzania bazami danych wspierające format XML

1.1. IBM DB2

Rosnąca popularność standardu XML skłoniła firmę IBM do podjęcia prac nad wprowadzeniem do swego produktu mechanizmów pozwalających obsłużyć nowy typ danych. Pięć lat pracy kilkuset deweloperów, architektów oraz inżynierów przyniosło efekt w postaci pierwszego na rynku hybrydowego serwera bazy danych. Nowy produkt – IBM DB2 w wersji 9 ujrzał światło dzienne w lipcu 2006 roku. Spośród innych wyróżniało go to, że pozwalał na efektywne przechowywanie zarówno danych relacyjnych jak i hierarchicznych. Podczas, gdy inne serwery baz danych oraz wcześniejsze wersje DB2 potrafiły przechowywać dokumenty XML jedynie w strukturach relacyjnych, IBM zaproponował rozwiązanie pozwalające na umieszczanie danych również w zhierarchizowanej formie w postaci drzew. Nowa technologia zyskała nazwę pureXML. Zapytania do serwera można wysyłać zarówno w postaci SQL/XML[2] jak i w języku zapytań XML – Xquery.

W 2007 roku IBM wydał kolejną, ostatnią jak dotąd stabilną wersję produktu - DB2 9.5 zawierającą szereg poprawek oraz usprawnień w stosunku do poprzedniczki. IBM DB2 z technologią pureXML może pracować pod kontrolą systemów operacyjnych z rodziny Linux, Windows, Solaris, AIX oraz HP-UX.

1.2. Microsoft SQL Server

Microsoft pierwsze usprawnienia dla XML wprowadził w SQL Server 2000. Polegały one na dodaniu do słownika serwera słów kluczowych FOR XML oraz OPENXML. FOR XML jest rozszerzeniem w składni SELECT pozwalającym na zwracanie wyników zapytania w postaci strumienia XML. Funkcja OPENXML działa w przeciwny sposób – wynik zapytania dla otrzymanego dokumentu XML przedstawia w standardowym, wierszowym zestawieniu.

Przykład:

2 SQL/XML jest rozszerzeniem SQL i stanowi część specyfikacji ANSI/ISO SQL 2003 (formalnie: ISO/ IEC 9075-14:2003)

```
SELECT ID, Nazwa
FROM Produkty Produkt
FOR XML AUTO

<Produkt ID=”1” Nazwa=”szafa”/>
<Produkt ID=”2” Nazwa=”komoda”/>
```

Przykład 3: Zapytanie SQL oraz rezultat w formacie XML, serwer SQL Server

Przykład:

```
<Zamowienie ZamowienieID = "65002">
    <Produkt ProduktID="1" Liczba="3"/>
    <Produkt ProduktID="2" Liczba="16"/>
</Zamowienie>

SELECT * FROM
OPENXML (@xmlDoc, 'Zamowienie/Produkt', 1)
WITH
(ZamowienieID integer '../@ZamowienieID', ProduktID integer, Liczba integer)
```

ZamowienieID	ProduktID	Liczba
65002	1	3
65002	2	16

Przykład 4: Zapytanie danych XML zwracające wynik w postaci SQL, serwer SQL Server

W SQL Server 2005 wprowadzono szereg poprawek oraz usprawnień. Nowa wersja umożliwiała m. in. zagnieżdżanie FOR XML, dzięki czemu zapytania mogły zwracać bardziej złożone wyniki. Największą nowością w wersji 2005 był jednak nowy typ danych – natywny XML pozwalający na tworzenie zmiennych oraz kolumn dla danych XML. Wraz z nowym typem pojawiało się również repozytorium dokumentów XSD oraz funkcja walidacji. Dodano obsługę zapytań Xquery oraz indeksowanie XML.

Przykład:

```
CREATE TABLE Zamowienia (
    ID integer PRIMARY KEY,
    Data datetime,
    KlientID integer,
    Uwagi xml (UwagiZamowieniaXSD))
```

Przykład 5: Tworzenie tabeli z kolumną typu XML z nałożonym schematem XSD

SQL Server 2008 oferuje dodatkowe możliwości walidacji XSD, rozbudowane wsparcie dla zapytań Xquery oraz rozszerzoną funkcjonalność ułatwiającą operacje modyfikacji danych XML. Po tym, bazuje głównie na rozwiązaniach zaproponowanych wersji 2005.

1.3. Oracle Database

Oracle jako pierwszy wprowadził do swoich produktów mechanizmy pozwalające na pracę z danymi XML. W wydanej w 1999 roku wersji 8i pojawiła się możliwość generowania XML bezpośrednio z bazy przy pomocy specjalnych zapytań, a także wypełniania bazy na podstawie zawartości dokumentu XML. Oracle dostarczył również XDK (ang. XML Developer's Kit) zawierające m. in. parsery pozwalające na wykorzystanie XML w aplikacjach napisanych w języku PL-SQL, Java oraz C/C++.

Wśród komponentów Oracle Database 9i pojawił się XMLType – typ danych do przechowywania dokumentów XML, a wraz z nim szereg nowych funkcji SQL do ich generowania i przetwarzania. XMLType powstał w oparciu o typ dużych obiektów znakowych (CLOB). Kolejne odsłony – wersje 9.2 oraz 10.1 rozbudowują możliwości bazy w zakresie generowania, modyfikacji, importu oraz eksportu danych XML.

Obecnie Oracle Database pozwala na przechowywanie dokumentów XML w modelu obiektowo-relacyjnym, w postaci obiektów znakowych oraz w postaci binarnej. Umożliwia ich indeksowanie, przeszukiwanie i modyfikację. Zapewnia również szereg funkcjonalności związanych z transformacją i walidacją danych XML.

Zbiór technologii Oracle Database związanych z XML nosi nazwę Oracle XML DB.

1.4. PostgreSQL

Twórcy serwera PostgreSQL nieustannie starają się skrócić dystans dzielący ich produkt od komercyjnych rozwiązań takich firm jak IBM, Microsoft czy Oracle. Co kilka miesięcy wydawane są kolejne wersje systemu a jego możliwości stale rosną. Wciąż jednak istnieje spory wachlarz funkcjonalności, które oferują już bazy DB2 czy SQL Server a których nie posiada PostgreSQL. Podobnie rzecz się ma jeśli chodzi o oferowane przez ten system wsparcie dla danych XML.

Pierwszy, znaczący pakiet funkcji XML pojawił się w wydanej w grudniu 2006 roku wersji oznaczonej numerem 8.2. Wówczas serwer oferował m. in. walidację poprawności dokumentu, proste indeksowanie, przeszukiwanie za pomocą Xpath. Dokumenty przechowywane były jednak wciąż jako dane typu tekst. Znaczące zmiany nastąpiły wraz z pojawieniem się w lutym 2008 roku wersji 8.3. Najważniejsze z nich do implementacja wsparcia dla standardu SQL/XML oraz dodanie nowego typu danych – xml. Choć obecnie PostgreSQL zawiera całkiem pokaźny zestaw narzędzi do pracy z danymi XML to twórcy serwera wciąż pracują nad kolejnymi. Wśród najważniejszych celów dla kolejnych wersji wymienia się implementację walidacji zgodności dokumentów z XSD, usprawnienie działania zapytań Xpath, rozbudowa mechanizmu indeksowania, dostosowanie niektórych funkcji do standardu SQL/XML:2006[3] oraz implementacja obsługi zapytań Xquery.

1.5. Pozostałe

Możliwość operowania danymi XML nie jest wyłącznie domeną omawianych w tej pracy SZBZ. Pewne możliwości w tej dziedzinie posiadają także np. bazy danych Sybase ASE[4], FrontBase[5] czy MySQL[6]. Cieszący się dużą popularnością serwer MySQL oferuje jednak tylko funkcje do odpytywania oraz modyfikacji łańcucha znaków reprezentującego dokument XML. Wykorzystuje przy tym język Xpath do nawigacji po dokumencie. Funkcje dostępne są od wersji 5.1.

Przykład:

3 SQL/XML:2006 jest nowszą wersją standardu SQL/XML:2003, formalnie ISO/ IEC 9075-14:2006

4 Relacyjny serwer baz danych – produkt firmy Sybase Inc. Oficjalna witryna firmy: www.sybase.com

5 Relacyjny serwer baz danych, produkt komercyjny. Oficjalna witryna projektu: www.frontbase.com

6 Relacyjny SZBD – dostępny na licencji GPL oraz komercyjnej. Oficjalna witryna projektu: www.mysql.com

```
<Osoba>
    <Imie>Tomasz</Imie>
    <Imie>Krzysztof</Imie>
    <Nazwisko>Nowak</Nazwisko>
</Osoba>
```

SELECT ExtractValue(@xmlDoc, '//Imie[1]')

Imie
Tomasz
Krzysztof

SELECT UPDATEXML (@xmlDoc, '/Osoba/Nazwisko', '<ID>11093</ID>') **AS** NowyXML

NowyXML: '<Osoba><Imie>Tomasz</Imie><Imie>Krzysztof</Imie><ID>11093</ID></Osoba>'

Przykład 6: Wykorzystanie funkcji ExtractValue oraz UpdateXML, serwer MySQL

Znacznie więcej możliwości daje serwer Sybase ASE – w wersji 15.0 zaimplementowano m.in. obsługę zapytań Xpath i Xquery, indeksowanie dokumentów XML oraz konwersję danych relacyjnych do XML. Jednak podobnie jak w przypadku MySQL dane XML są wciąż reprezentowane przez dane typu tekst.

Inne SZBD w większości nie posiadają wparcia dla danych XML lub wspierają je w bardzo ograniczonym stopniu. Twórcy niektórych z nich – jak choćby serwera Firebird[7] zapowiedzieli dopiero implementację wybranych funkcji w kolejnych wersjach.

7 Wolnodostępny serwer relacyjnych baz danych. Oficjalna witryna projektu: www.firebirdsql.org

2. Architektura danych XML w SZBD

2.1. Reprezentacja danych typu XML

Dane XML mogą być reprezentowane na kilka sposobów. W każdym z omawianych SZBD implementacja typu XML jest inna, ale można podzielić je na dwie grupy. Pierwszą z nich stanowią implementacje oparte o dane znakowe: VARCHAR, CLOB itp. W rozwiązaniu tym typ XML jest szczególnym przypadkiem typu znakowego – na typ znakowy nałożona jest dodatkowo kontrola poprawności składni XML. Dane przechowywane w bazie są wówczas dokładną kopią danych wprowadzonych do bazy – zachowane są m. in. białe znaki, kolejność atrybutów, pamiętany jest sposób domknięcia znaczników itp. W niektórych przypadkach może być to zaleta – np. gdy zależy nam na zachowaniu dokumentu (licencji, umowy) w oryginalnej formie. Rozwiązanie takie zastosowano m. in. w serwerze PostgreSQL.

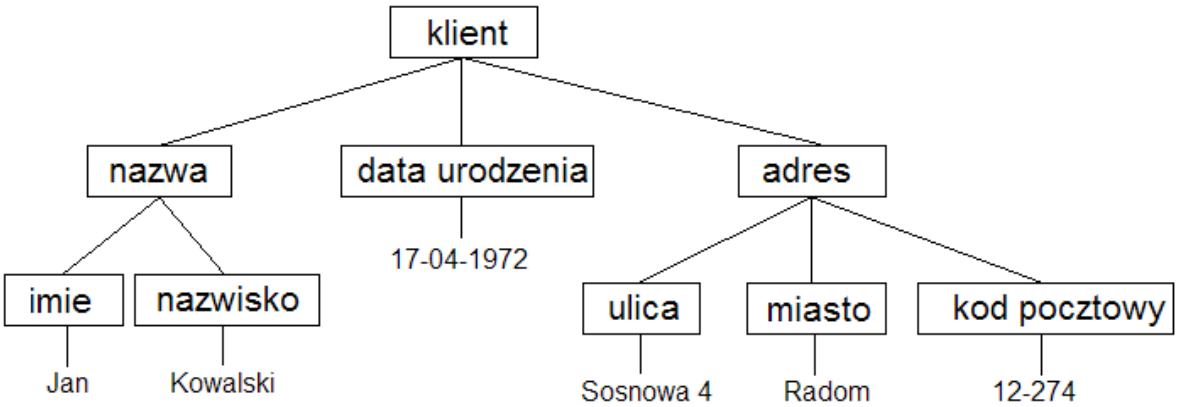
Reprezentacja w postaci tekstu niesie jednak za sobą wiele istotnych wad – o ile operacje wstawiania czy usuwania całych dokumentów są dość efektywne i proste o tyle manipulacja częścią dokumentu jest już znacznie utrudniona. W takim modelu przede wszystkim nie ma możliwości efektywnego indeksowania zawartości, co uniemożliwia sprawne przeszukiwanie i modyfikację wybranych fragmentów dokumentu. Z punktu widzenia składu danych typ XML w serwerze PostgreSQL nie różni się od typów znakowych. W serwerze tym typ XML, podobnie jak inne mogące służyć do przechowywania znacznych porcji danych, podlega mechanizmowi o nazwie TOAST (The Oversized-Arrtibute Storage Technique). Mechanizm polega na kompresji oraz dzieleniu porcji danych w przypadku, gdy ta przekracza rozmiar strony (zwykle 8 kB). Stąd też, dokumenty XML o znacznych rozmiarach, przed umieszczeniem w bazie danych są kompresowane, a następnie w razie konieczności dzielne na kawałki o wielkości rozmiaru strony. Każdy z takich kawałków posiada 32 bitowy nagłówek, z którego 2 bity zarezerwowane są na potrzeby mechanizmu TOAST, natomiast kolejne 30 bitów stanowi całkowitą długość danych wyrażoną w bajtach. Stąd też ograniczenie na wielkość danych XML w serwerze PostgreSQL wynosi 1 GB ($2^{30} – 1$ bajtów). Cały proces przebiega niezauważalnie dla użytkownika systemu. Zastosowanie takiego mechanizmu ma swoje wady i zalety. Z jednej strony znacznie redukuje rozmiar danych XML w bazie danych. Wynika to z faktu, że dane XML reprezentowane są w postaci tekstu, dają się zatem dobrze kompresować. Wadą mechanizmu jest wydłużony czas pracy z danymi, które każdorazowo trzeba dekompresować oraz łączyć w całość. Wyniki testów własnych dotyczących szybkości oraz stopnia kompresji danych zaprezentowano w kolejnych

podrozdziałach.

Całkiem inne rozwiązanie zastosowano w serwerze SQL Server. system ten jest „świadomy” hierarchicznej struktury dokumentów i dane typu XML przechowuje w hierarchicznej formie. W tym celu wykorzystywany jest standard XML Information Set[8], będący zaleceniem konsorcjum W3C. Standard opisuje abstrakcyjną reprezentację dokumentu i stanowi próbę zdefiniowania, jakie informacje w dokumencie XML są informacjami znaczącymi.

Przykład:

```
<klient>
    <nazwa>
        <imie>Jan</imie>
        <nazwisko>Kowalski</nazwisko>
    </nazwa>
    <data urodzenia>17-04-1972</data urodzenia>
    <adres>
        <ulica>Sosnowa 4</ulica>
        <miasto>Radom</miasto>
        <kod pocztowy>12-274</kod pocztowy>
    </adres>
</klient>
```



Ilustracja 1: Dokument XML w hierarchicznej formie

8 XML Information Set (XML InfoSet) – standard organizacji W3C opisujący abstrakcyjną reprezentację dokumentu XML. Oficjalna witryna standardu: http://www.w3.org/TR/xml-infoset

Przed zasileniem bazy danych dokument jest parsowany pod kątem poprawności składni XML a następnie przekształcany do wewnętrznej reprezentacji serwera. W tym przypadku zawartość bazy danych nie jest dokładną kopią danych źródłowych. W bazie danych przechowywana jest struktura drzewiasta, a zatem niektóre, nieistotne informacje o dokumencie zostają utracone. W szczególności nie jest rozróżniany sposób domknięcia pustego elementu (<element></element> jest tym samym co <element/>) oraz pomijane są niektóre białe znaki. Zbiór informacji nieistotnych z punktu widzenia reprezentacji danych XML w postaci hierarchicznej opisany jest przez standard XML InfoSet.

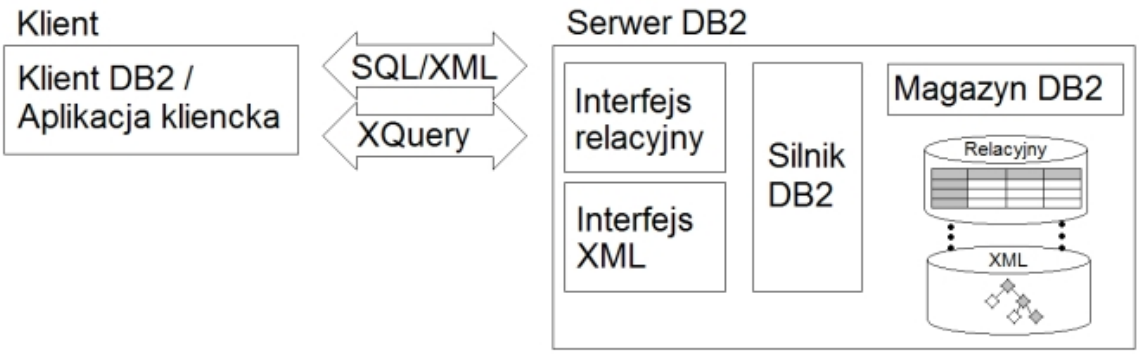
Przykład:

```
INSERT INTO docs VALUES
('<?xml version="1.0"?><magazyn>   <towar id="1"></towar>       <towar id="2"/>   </magazyn>' )

SELECT doc FROM docs
doc
----------------------
<magazyn>
  <towar id="1" />
  <towar id="2" />
</magazyn>
```

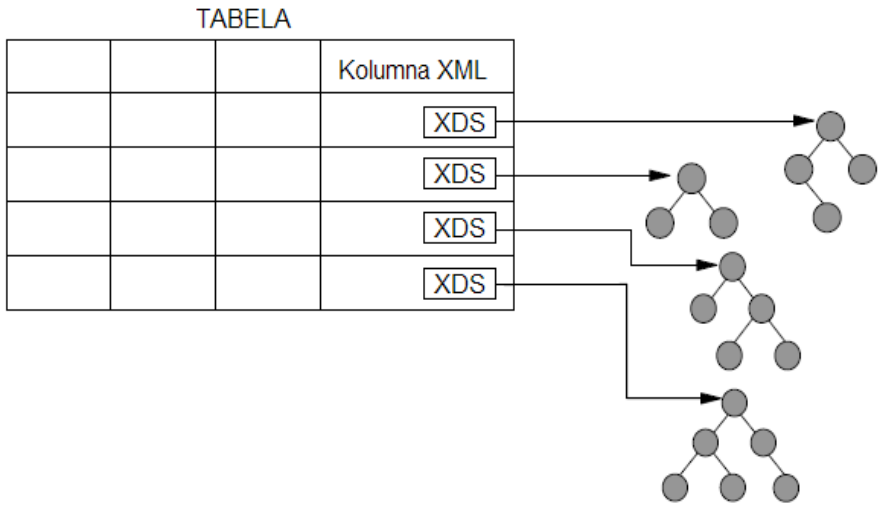
Przykład 7: Zasilenie oraz odtworzenie dokumentu XML na podstawie danych z InfoSet, serwer SQL Server

Podobne, choć nieco inne rozwiązania zastosowano w serwerze firmy IBM. DB2 łączy w sobie mechanizmy zarządzania danymi relacyjnymi oraz danymi XML. Dane te składowane i przetwarzane są osobno – stąd też DB2 nazywany jest serwerem hybrydowym. Z punktu widzenia aplikacji klienckich dostęp do danych jest jednakowy – to serwer zarządza i integruje dane XML z danymi relacyjnymi.



Ilustracja 2: Uproszczona architektura serwera IBM DB2 9

Podobnie jak SQL Server także DB2 podczas wprowadzania danych do bazy parsuje je a następnie przekształca do wewnętrznej, hierarchicznej struktury. Oznacza to, że każdy dokument musi być poprawnie sformatowany zanim zostanie umieszczony w bazie. Każda próba zaimportowania lub zasilenia tabeli danymi, które nie stanowią poprawnego dokumentu XML zakończy się błędem. W przypadku powodzenia każdy wiersz tabeli w kolumnie typu XML zawiera jedynie wskaźnik XDS[9] – właściwe dane składowane są w odrębnych strukturach bazy.



Ilustracja 3: Relacje pomiędzy kolumną typu XML a danymi XML

Serwer Oracle oferuje typ XMLType, który daje możliwość składowania danych na

9 XDS (XML Data Specifier) – wskaźnik zawierający informacje o lokalizacji danych XML na dysku

dwa sposoby. W zależności od wybranej opcji dokument XML może być reprezentowany w bazie danych jako duży obiekt znakowy (CLOB) lub w postaci binarnej. Wybór pierwszej z opcji upodabnia XMLType do implementacji typu XML serwera PostgreSQL – XMLType jest wówczas typem znakowym z nałożoną kontrolą poprawności dokumentu a zawartość bazy danych jest kopią źródłowego ciągu znaków. Druga opcja pozwala na składowanie danych XML w wewnętrznych, dedykowanych strukturach – podobnie jak ma to miejsce w serwerze DB2. O tym w jaki sposób składować dane XML użytkownik musi zdecydować w momencie tworzenia tabeli z kolumną XMLType (domyślnie jest typem znakowym).

Przykład:

```
CREATE TABLE XmlDoc (id INTEGER, doc XMLTYPE)
    XMLTYPE COLUMN doc STORE AS CLOB;

CREATE TABLE XmlDoc (id INTEGER, doc XMLTYPE)
    XMLTYPE COLUMN doc STORE AS BINARY XML;
```

Przykład 8: Tworzenie tabeli z kolumną typu XMLType w postaci tekstowej oraz binarnej, serwer Oracle

Typ binarny jest jedną z nowości systemu Oracle 11g. Jedną z jego głównych zalet jest sposób w jaki dane reprezentowane są w systemie. Dotychczas dane XML wymagały parsowania oraz serializacji za każdym razem, gdy były wymieniane pomiędzy różnymi warstwami aplikacji lub przed zapisem na dysk. Binarny XMLType stanowi reprezentację jednakową dla danych składowanych na dysku, w pamięci systemu czy wykorzystywaną w trakcie transferu danych. Dzięki takiemu ujednoliceniu formatu zredukowano narzut na pracę procesora oraz wymaganą pamięć.

Przekształcanie danych XML do binarnego typu XMLType polega m. in. na tokenizacji tagów (podział tagów na tokeny, grupowanie oraz mapowanie na identyfikatory). Proces ten tym skuteczniej redukuje rozmiar dokumentu im bardziej powtarzalna jest jego struktura. Ponadto, tekstowe wartości atrybutów i węzłów przekształcane są do natywnej postaci – np. dane całkowitoliczbowe reprezentowane są jako typ INTEGER. Według zapewnień producenta, operacje te znacznie zmniejszają wielkość dokumentów XML w bazie danych (wyniki testów własnych zaprezentowano w kolejnych podrozdziałach).

2.2. Generowanie zawartości XML przy pomocy funkcji SQL/XML

SQL/XML jest standardem ANSI oraz ISO opisującym możliwości i sposoby przetwarzania danych XML w bazach danych SQL. Został stworzony na potrzeby integracji danych relacyjnych z danymi XML. Standard definiuje szereg funkcji pozwalających na tworzenie i modyfikację zawartości typu XML. Za ich pomocą możliwa jest m. in. konwersja łańcuchów znaków na typ XML, tworzenie elementów, atrybutów, komentarzy XML, łączenie małych fragmentów danych w większe itp.

Jedną z podstawowych funkcji do tworzenia zawartości XML jest XMLPARSE. Przekształca on dane z postaci tekstowej do natywnego typu XML, pozwala określić czy łańcuch znaków stanowi kompletny dokument XML czy jedynie fragment oraz w jaki sposób traktować białe znaki. Standardowo funkcja ma postać:

XMLPARSE ({DOCUMENT | CONTENT} łańcuch_znaków [<opcje>]),
gdzie **<opcje> := { PRESERVE | STRIP } WHITESPACES**

Ze względu na sposób w jaki serwer przechowuje dane XML poszczególne implementacje różnią się. Np. w serwerze PostgreSQL funkcja nie posiada opcji pozwalających na wybór sposobu w jaki postępować z białymi znakami, z kolei wersja Oracle posiada dodatkową flagę 'WELLFORMED' informującą serwer, że dane są poprawnie sformatowane i nie ma potrzeby sprawdzania poprawności składni XML. W serwerze Oracle podobne rozszerzenia standardu SQL/XML zastosowano także w przypadku wielu innych funkcji. Funkcją odwrotną do XMLPARSE jest XMLSERIALIZE.

Przykład:

```
INSERT INTO docs VALUES( Xmlparse( Document
'<?xml version="1.0"?><magazyn>    <towar id="1"></towar>       <towar id="2"/> </magazyn>' ));

SELECT Xmlserialize(Document (SELECT doc FROM docs) AS VARCHAR(100));

                                        xmlserialize
-----------------------------------------------------------------------------------------------------
<?xml version="1.0"?><magazyn>    <towar id="1"></towar>       <towar id="2"/> </magazyn>
```

Przykład 9: Wykorzystanie funkcji XMLSERIALIZE do pobrania z dokumentu XML w oryginalnej formie, serwer PostgreSQL

Pozostałe funkcje SQL/XML pozwalające na tworzenie zawartości XML to m. in.:

XMLELEMENT	- tworzy element XML z nazwą, atrybutami oraz zawartością
XMLATTRIBUTES	- tworzy atrybut elementu XML
XMLFOREST	- tworzy sekwencję elementów XML
XMLCONCAT	- łączy listę pojedynczych wartości XML
XMLCOMMENT	- tworzy komentarz XML
XMLPI	- tworzy instrukcję przetwarzania XML
XMLAGG	- funkcja agregująca, agreguje wartości XML
XMLROOT	- tworzy element ROOT w dokumencie XML

Przykład:

```
SELECT Xmlelement(name zamowienie, Xmlattributes(146 as id), Xmlelement(name data, current_date));

                xmlelement
-----------------------------------------------------------
<zamowienie id="146"><data>2009-08-11</data></zamowienie>
```

Przykład 10: Wykorzystanie funkcji XMLELEMENT oraz XMLATTRIBUTES do stworzenia fragmentu dokumentu XML, serwer PostgreSQL

Przykład:

```
VALUES( Xmlelement( name "osoba", Xmlforest( 'Jan' as "imie", 'Nowicki' as "nazwisko", 'M' as "plec")));
```

wynik:

```
<osoba>
    <imie>Jan</imie>
    <nazwisko>Nowicki</nazwisko>
    <plec>M</plec>
</osoba>
```

Przykład 11: Wykorzystanie funkcji XMLFOREST do stworzenia sekwencji elementów, serwer DB2

Opisane powyżej funkcje to tylko niektóre spośród zdefiniowanych przez standard ISO/IEC 9075:14. Wyszczególniono głównie te, które przeznaczone są do tworzenia zawartości typu XML. Są one przydatne w procesie automatycznego generowania dokumentów XML i zwalniają programistę bazy danych z obowiązku wprowadzania danych XML do bazy w postaci tekstu. Ponadto, umożliwiają generowanie XML na podstawie danych relacyjnych z tabel i widoków. Zaimplementowane zostały w serwerach PostgreSQL, Oracle oraz DB2, pominięte natomiast w Sql Server. W przypadku serwera firmy Microsoft większość z opisanej funkcjonalności uzyskać można przy pomocy klauzuli FOR XML (opisanej dokładniej w kolejnych podrozdziałach). Ponadto, zadania realizowane przez niektóre z funkcji SQL/XML są w Sql Server wykonywane niejawnie – tak jest w przypadku operacji parsowania tekstu oraz serializacji danych XML (XMLPARSE oraz XMLSERIALIZE).

++ - funkcja zaimplementowana, zgodna ze standardem SQL/XML
+ - funkcja zaimplementowana, nieznacznie odbiega od standardu
- - funkcja niezaimplementowana
* - funkcja zawiera rozszerzoną składnię i dodatkowe opcje

Zestawienie 1: Wybrane funkcje standardu SQL/XML na tle omawianych SZBD

2.3. Tworzenie XML ze struktur relacyjnych

Jednym z podstawowych zadań jakim sprostać powinien system zarządzania bazami danych z natywnym typem XML jest zdolność do integracji i swobodnego przekształcania danych relacyjnych na dane hierarchiczne i na odwrót. Konwersja danych pomiędzy modelem relacyjnym i hierarchicznym rodzi potrzebę zdefiniowania schematu opisu struktury danych oraz typu i zakresu prezentowanych informacji, zgodnych ze schematem modelu relacyjnego. Proces wymiany danych pomiędzy strukturą XML, a relacyjną powinien zostać poprzedzony wcześniejszym zdefiniowaniem schematu opisującego charakter przechowywanych wartości. Można to uczynić dokonując odwzorowania struktury bazy relacyjnej do standardu XML Schema Definition (XSD). Specyfikacja języka XSD dostarcza zestawu narzędzi dla poprawnego odwzorowania elementów modelu relacyjnego (tabele, kolumny) wraz z zachowaniem występujących zależności oraz integralności istniejących danych. Realizowane jest to z jednej strony poprzez możliwość zastosowania określonego zestawu elementów oraz atrybutów języka XSD, umożliwiających wymuszenie np. unikalności wartości elementu (klucz główny w modelu relacyjnym) lub liczebności występowania elementów (klauzula NOT NULL itp. w modelu relacyjnym), z drugiej zaś strony poprzez określenie relacji

zależności pomiędzy elementami schematu hierarchicznego, odzwierciedlającymi związki występujące w modelu relacyjnym.

Każdy z omawianych SZBD posiada własny, specyficzny mechanizm pozwalający generować schematy XML na podstawie modelu relacyjnego. Mechanizmy te w różnym stopniu odzwierciedlają strukturę danych relacyjnych. Jakość generowanych schematów pozostawia często wiele do życzenia. Z przeprowadzonej analizy wynika, że zdarzają się przypadki np. niepoprawnie rozpoznawanych typów danych, wymaganej liczebności czy unikalności elementów. Często definicja typu elementu ma charakter ogólny i konieczne jest jej ręczne doprecyzowanie.

Przykład:

Kolumna	**Typ**	**Modyfikatory**
id	integer	**not null**
imie	character varying(20)	
nazwisko	character varying(20)	

PRIMARY KEY, btree (id)

```
SELECT Table_To_Xmlschema('table_1', false, false, 'testns');
 <xsd:schema   xmlns:xsd="http://www.w3.org/2001/XMLSchema"  targetNamespace="testns" … >
    <xsd:simpleType name="INTEGER"> ... </xsd:simpleType>
    <xsd:simpleType name="VARCHAR"> ... </xsd:simpleType>
   <xsd:complexType name="RowType.testdb.public.table_1">
      <xsd:sequence>
         <xsd:element name="id" type="INTEGER" minOccurs="0"></xsd:element>
         <xsd:element name="imie" type="VARCHAR" minOccurs="0"></xsd:element>
         <xsd:element name="nazwisko" type="VARCHAR" minOccurs="0"></xsd:element>
       </xsd:sequence>
   </xsd:complexType>
   ...
 </xsd:schema>
```

Przykład 12: Generowanie schematu XML przy pomocy funkcji TABLE_TO_XMLSCHEMA, wyróżniono błędnie zdefiniowany element „id”, serwer PostgreSQL

Generowanie zawartości XML na podstawie danych składowanych w modelu relacyjnym odbywać się może na kilka sposobów. Wykorzystać do tego celu można funkcje generujące standardu SQL/XML(Oracle, DB2 oraz PostgreSQL) jak również szereg

systemowych funkcji, właściwych dla danego SZBD. Poszczególne implementacje różnią się zarówno sposobem jak i treścią generowanych danych. Np. odwzorowanie elementów modelu relacyjnego realizowane może być zarówno poprzez zastosowanie deklaracji elementów, jak i zbioru atrybutów w dokumencie XML. Pierwsze rozwiązanie zastosowano w serwerach DB2, Oracle oraz PostgreSQL, drugie natomiast w SQL Server.

Przykład:

Osoba:

Id	Imię	Nazwisko
1	Jan	Kowalski

Zbiór atrybutów:

```
<Osoba Id="1" Imię="Jan" Nazwisko="Kowalski" />
```

Zbiór elementów:

```
<Osoba>
   <Id>1</Id>
   <Imię>Jan</Imię>
   <Nazwisko>Kowalski</Nazwisko>
</Osoba>
```

Przykład 13: Prezentacja elementów w postaci zbioru atrybutów oraz zbioru elementów

/* podrozdział będzie jeszcze rozwijany*/

2.4. Import danych XML z zewnętrznych plików

Niewielkie dokumenty XML mogą być wprowadzane do bazy danych w postaci jawnie podawanego tekstu za pośrednictwem np. funkcji SQL/XML. W przypadku większych dokumentów jest to bardzo kłopotliwe lub niemożliwe. Niektóre aplikacje klienckie baz danych posiadają ograniczenia na długość ciągu wejściowego stąd też, konieczne jest stosowanie innych rozwiązań. W przypadku dokumentów XML składowanych na dysku w postaci plików wykorzystać można systemowe funkcje importu oferowane przez każdy z omawianych w tej pracy serwerów.

Poniżej zaprezentowano przykłady użycia funkcji importu, specyficznych dla każdego z SZBD oraz wyniki testów mających na celu sprawdzenie szybkości z jaką serwery baz danych konwertują dokumenty XML do wewnętrznej, natywnej postaci. Funkcje importu

testowano na plikach o różnych rozmiarach (od kilku kB do kilkudziesięciu MB). Na wykresie przedstawiono uśrednione wyniki w podziale na pliki o rozmiarze do 1 MB oraz powyżej 1 MB.

Przykład:

```
INSERT INTO tab(id,doc) VALUES(1, CAST(PG_READ_FILE('f.xml', 0, 10000000) AS XML))
```

Przykład 14: Wykorzystanie funkcji PG_READFILE do importu danych XML z zewnętrznego pliku, serwer PostgreSQL

```
IMPORT FROM 'xmldir\files.del' OF DEL XML FROM 'xmldir' INSERT INTO tab
```

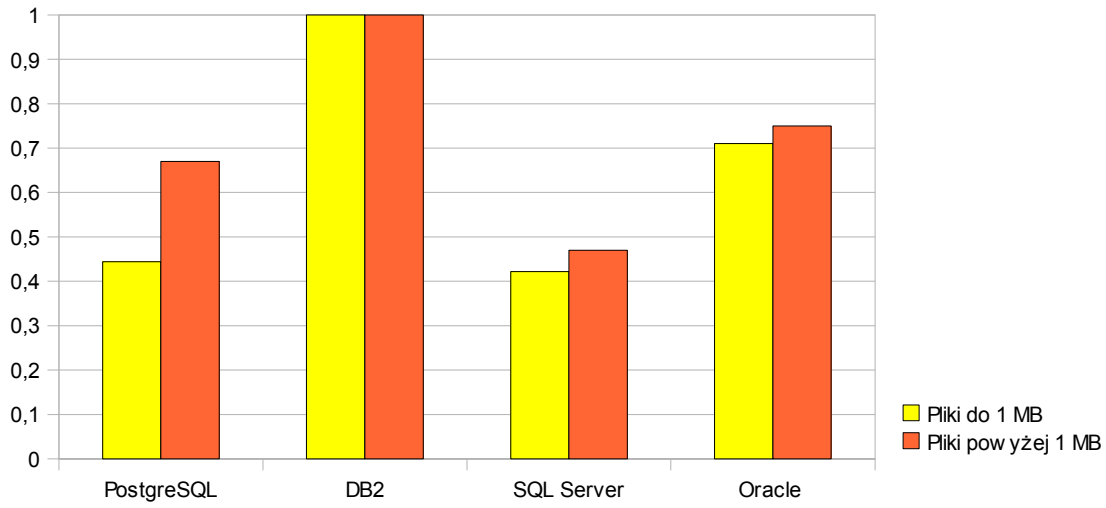
Przykład 15: Wykorzystanie funkcji IMPORT do importu danych XML z zewnętrznych plików (wyszczególnionych w pliku importu .del), serwer DB2

```
INSERT INTO tab(id,doc) SELECT 1, x.* FROM OPENROWSET( BULK 'f.xml', SINGLE_BLOB) AS x
```

Przykład 16: Wykorzystanie funkcji OPENROWSET do importu danych XML z zewnętrznego pliku (konwersja do XML odbywa się niejawnie), serwer SQL Server

```
INSERT INTO tab VALUES (1, XMLType(BFILENAME('DIR', 'f.xml'), nls_charset_id('UTF8')))
```

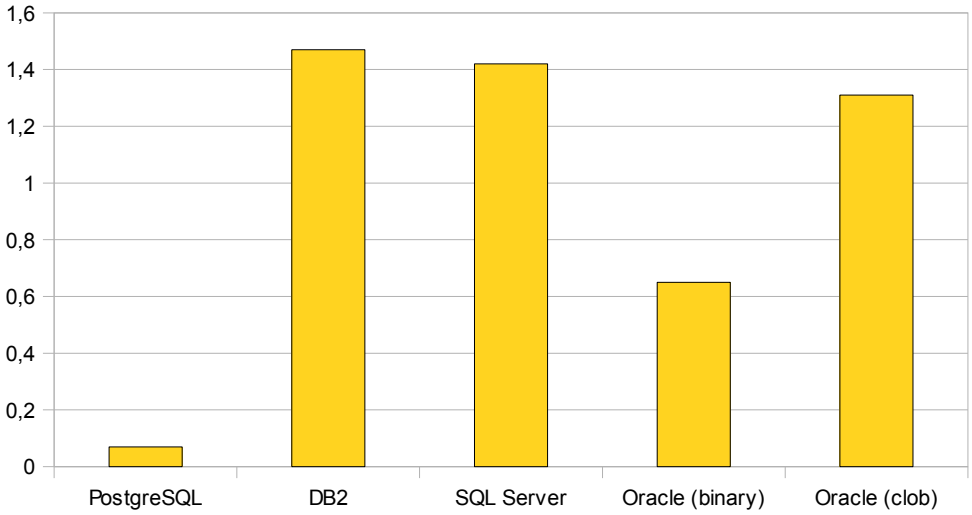
Przykład 17: Wykorzystanie funkcji BFILENAME do importu danych XML z zewnętrznego pliku, serwer Oracle



Zestawienie 2: Porównanie szybkości konwersji danych tekstowych do natywnego typu XML, w osi pionowej wektor czasu w znormalizowanej postaci

Zestawienie 2 pokazuje, że czas konwersji na typ XML jest najkrótszy w przypadku serwera SQL Server i nieco dłuższy dla PostgreSQL'a oraz Oracle'a. Najdłużej trwa konwersja w serwerze DB2 – zarówno dla małych jak i większych dokumentów. Wynika to z konieczności przekonwertowania danych XML do wewnętrznej reprezentacji serwera oraz umieszczenia ich w osobnym, przeznaczonym dla danych XML magazynie. Podobną operację przed umieszczeniem w bazie danych przeprowadza serwer Oralce, jednak w tym przypadku proces trawa o ok 20-30 % krócej. W przypadku serwera PostgreSQL dłuższy czas pracy z większymi dokumentami związany jest z działaniem mechanizmu dzielenia i kompresji danych (TOAST).

Poniżej zaprezentowano zestawienie zawierające porównanie przestrzeni dyskowej zajmowanej przez dokumenty XML składowane w poszczególnych bazach danych. Na potrzeby testu w bazach składowano dokumenty o znacznych rozmiarach (powyżej 5 MB). Wyniki dla serwera Oracle zaprezentowano w podziale na typ binarny oraz znakowy.



Zestawienie 3: Porównanie wymaganej przestrzeni dyskowej dla dokumentów XML, w osi pionowej stosunek rozmiaru dokumentu XML w bazie danych do rozmiaru pliku na dysku w systemie plików Windows (NTFS)

Serwer PostgreSQL jako jedyny w wymienionych stosuje klasyczną kompresję danych – stąd tak znaczące różnice w wielkości danych w bazie. Zastosowany algorytm charakteryzuje się dużą szybkością działania przy zachowaniu stosunkowo wysokiego stopnia kompresji danych (testy wykazały, że średni stopień kompresji dla dokumentów XML o rozmiarach 5-10 MB wynosi ok 17, przy czym stopień kompresji dla jednakowych dokumentów poddanych kompresji przy pomocy popularnych algorytmów (rar, zip, bz2) wyniósł ok 25-40). Znaczną redukcję wymaganej przestrzeni dyskowej zapewnia również typ binarny XMLType serwera Oracle. Testy wykazały, że zastosowany proces tokenizacji tagów oraz konwersji wartości atrybutów i węzłów na inny niż tekstowy typ danych daje ok dwukrotną redukcję wielkości danych w stosunku to typu XMLType CLOB (według twórców serwera Oracle może być to nawet pięciokrotny zysk). W pozostałych przypadkach dane w bazie przechowywane są w nieskompresowanej formie, we właściwej dla danego serwera wewnętrznej postaci stąd też ich rozmiar przekracza rozmiar importowanego dokumentu XML.

2.5. Ograniczenia dotyczące danych typu XML

2.5.1. PostgreSQL

Maksymalny rozmiar danych typu XML, podobnie jak pozostałych łańcuchów

znaków wynosi 1 GB. Pamiętać jednak należy, że duże dokumenty XML przechowywane są w bazie w skompresowanej formie. Oznacza to, że pierwotny rozmiar pliku XML importowanego do bazy danych może być znacznie większy niż 1GB.

2.5.2. SQL Server

W SQL Server ograniczenie na wielkość pola typu XML wynosi 2 GB. Na rozmiar ten składa się właściwy dokument XML oraz jego indeksy: podstawowy (primary index - tworzony automatycznie) oraz dodatkowe (secondary index).

2.5.3. DB2

W przeciwieństwie do typu VARCHAR czy CLOB typ XML nie posiada skojarzonej ze sobą długości. Magazyn XML oraz architektura przetwarzania w DB2 nie nakłada limitu na rozmiar dokumentu XML. Obecnie jedynie protokół komunikacyjny klient-serwer ogranicza wielkość dokumentu do 2 GB.

Magazyn XML przystosowany jest jedynie do składowanie poprawnie sformatowanych dokumentów XML. Oznacza to, że nie ma możliwości umieszczania fragmentów dokumentów (bez elementu ROOT).

2.5.4. Oracle

Maksymalny rozmiar danych typu XMLType wynosi 4 GB. Wielkość ta ma jednak inne znaczenie w przypadku wersji binarnej a inne w wersji znakowej. Dla typu XMLType CLOB oznacza, że dokument może zawierać co najwyżej 2^{31} -1 znaków (2 bajty na znak). Dokumenty składowane jako XMLType binary podlegają konwersji na typ binarny i mają w bazie danych rozmiar znacznie mniejszy niż pierwotnie, w postaci tekstu. Stąd też, możliwe jest umieszczanie w systemie dokumentów znacznie przekraczających rozmiar 4 GB.

2.5.5. Podsumowanie

Maks. rozmiar	1 GB	dowolny *	4 GB **	2 GB

* protokół komunikacyjny klient-serwer nakłada ograniczenie do 2 GB na dokument

** zarówno dla dokumentów przechowywanych jako CLOB jak i binarnych

Zestawienie 4: Ograniczenie na wielkość dokumentu XML w bazie danych

/* pozostałe ograniczenia opiszę podczas realizacji kolejnych podrozdziałów */

3. Przeszukiwanie i modyfikacja dokumentów XML

3.1. Zapytania SQL/XML

XMLEXISTS

XMLTABLE

XMLQUERY

3.2. Zapytania XPATH

3.3. Zapytania XQUERY

3.4. Inne metody dostępu do danych XML

4. Indeksowanie dokumentów XML

5. Walidacja poprawności oraz zgodności z schematem XSD

5.1. Weryfikacja poprawności dokumentów

5.2. Weryfikacja zgodności ze schematem XSD

6. Inne mechanizmy pomocne w pracy z dokumentami XML

6.1. Transformacje XSLT

6.2. Optymalizacja zapytań do XML

6.3. Inne

BIBLIOGRAFIA

1. Michael Coles. *Pro SQL Server 2008 XML*. Apress, 2008.
2. Cynthia M. Saracco, Don Chamberlin, Rav Ahuja. *DB2 9: pureXML Overview and Fast Start*. IBM Redbooks, 2006.
3. Whei-Jen Chen, Art Sammartino, Dobromir Goutev, Felicity Hendricks, Ippei Komi, Ming-Pang Wei, Rav Ahuja, Matthias Nicola. *DB2 9 pureXML Guide*. IBM Redbooks, 2007
4. Drew Adams i inni. *Oracle XML DB Developer's Guide, 11g Release 1 (11.1)*. Oracle Corporation, 2008