

„Nowoczesne” zastosowania XML

XML w elektronicznej wymianie danych,
integracji aplikacji i bazach danych

Patryk Czarnik

Instytut Informatyki UW

XML i nowoczesne technologie zarządzania treścią – 2011/12

Elektroniczna wymiana danych

Wprowadzenie

Rozwiązania przed XML

XML w EDI

Integracja aplikacji

Idea

Usługi sieciowe

AJAX

XML w bezpieczeństwie

XML Signature

XML Encryption

XML w bazach danych

XML w relacyjnych bazach danych

XML-owe bazy danych

Elektroniczna wymiana danych (EDI) – motywacja

Możliwości wymiany danych między firmami/instytucjami (B2B)

- ▶ dokumenty na papierze
- ▶ elektroniczna wymiana danych

Możliwości wykorzystania standardu

- ▶ całe oprogramowanie tworzone zgodnie ze standardem,
- ▶ wewnętrzny format odmienny (np. wcześniej istniejący system), narzędzia tłumaczące do standardu w interfejsach

Możliwości ustalenia protokołu EDI

- ▶ dostawca (sprze)daje klientowi narzędzia
- ▶ partner dostosowuje się do formatu silniejszego partnera
- ▶ tworzone ad-hoc narzędzia tłumaczące
- ▶ standard

Standaryzacja EDI przed XML

ANSI Accredited Standards Committee X12 sub-group

- ▶ standard narodowy USA
- ▶ stosowany głównie w Ameryce

EDIFACT

- ▶ standard ONZ (UN/CEFACT) i ISO
- ▶ stosowany głównie w Europie i Azji
- ▶ popularny w branżach: handel, spedycja i transport

Charakterystyka EDIFACT

Format

- ▶ tekstowy
- ▶ mało czytelny
- ▶ struktura drzewiasta

Katalogi predefiniowanych struktur

- ▶ 193 typy wiadomości
- ▶ 279 segmentów
- ▶ 186 elementów

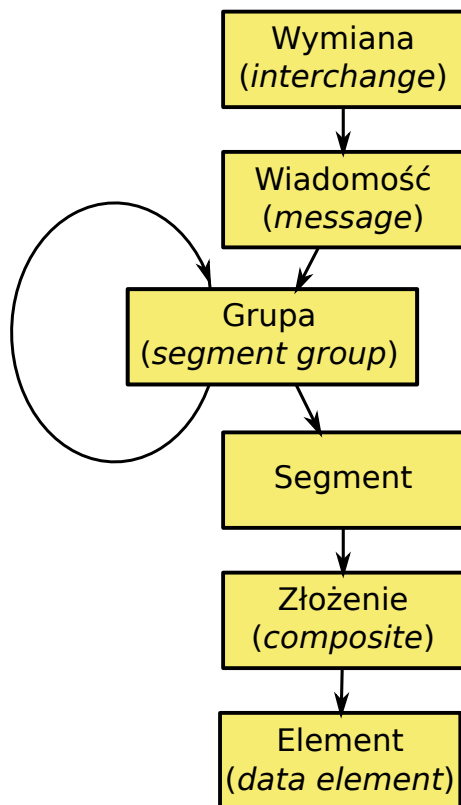
(dane dla wersji 08a z 2008 roku)

EDIFACT

EDIFACT – przykład komunikatu

```
UNB+IATB:1+6XPPC+LHPPC+940101:0950+1'  
UNH+1+PAORES:93:1:IA'  
MSG+1:45'  
IFT+3+XYZCOMPANY AVAILABILITY'  
ERC+A7V:1:AMD'  
IFT+3+NO MORE FLIGHTS'  
ODI'  
TVL+240493:1000::1220+FRA+JFK+DL+400+C'  
PDI++C:3+Y::3+F::1'  
APD+74C:0:::6+++++6X'  
TVL+240493:1740::2030+JFK+MIA+DL+081+C'  
PDI++C:4'  
APD+EM2:0:1630::6+++++DA'  
UNT+13+1'  
UNZ+1+1'
```

EDIFACT – struktura



MEA+WT+AAD+KGM:690+X5'

+KGM:690+

:690

XML EDI

Idea: zastosowanie XML jako formatu danych EDI.

Tradycyjne EDI

- ▶ Format dokumentów zapisany w specyfikacji.
- ▶ Zwięzłe komunikaty, tylko niezbędne dane.
- ▶ Scentralizowana, trudna zmiana standardu.
- ▶ Zmiany standardu pociągają uciążliwe zmiany oprogramowania.
- ▶ Narzędzia implementowane od podstaw.

XML EDI

- ▶ „Samoopisujący się” format dokumentów.
- ▶ Rozwlekłe komunikaty – narzut na „samoopisywanie się”.
- ▶ Możliwość tworzenia własnych odmian standardów.
- ▶ Większość problemów ze zmianą standardu bierze na siebie parser XML.
- ▶ Możliwość korzystania z gotowych narzędzi (parser, walidator).

Elastyczność XML EDI

- ▶ Elastyczny format komunikatów (elementy opcjonalne, wybór, zagnieżdżanie).
- ▶ Nieinwazyjne rozszerzanie formatów dzięki przestrzeniom nazw.
- ▶ XSLT jako język opisu formatowania.
- ▶ Zróżnicowane zastosowania EDI:
 - ▶ podstawowa funkcjonalność: wymiana danych między aplikacjami przedsiębiorstw,
 - ▶ nowe perspektywy: kontakt z klientami wyposażonymi tylko w przeglądarki WWW,
 - ▶ prosta integracja z Web Serwisami.

Standaryzacja XML EDI

Standaryzacja na poziomie ramowym

- ▶ możliwa wymiana informacji dowolnego typu,
- ▶ informacje jednego typu tak samo reprezentowane,
- ▶ przykład: Electronic Business XML (**ebXML**).

Standardy branżowe (konkretne zestawy komunikatów)

- ▶ SWIFT – bankowość
- ▶ RosettaNet – handel i transport (?)
- ▶ Automotive Industry Action Group – przemysł motoryzacyjny (gł. amerykański)
- ▶ Health Level Seven – ochrona zdrowia
- ▶ Open Travel Alliance – transport turystyka
- ▶ ...

ebXML – podejście do standaryzacji

- ▶ Meta-model pozwalający na opracowywanie modeli specyficznych dla zastosowań:
 - ▶ zbiór podstawowych schematów, elementów XML oraz procesów biznesowych,
 - ▶ sposób definiowania słowników danych,
 - ▶ nie definiuje zawartości konkretnych komunikatów – może ona zależeć od konkretnego zastosowania.
- ▶ Metainformacje:
 - ▶ informacje o wersjach,
 - ▶ metadane odpowiadające nagłówkom z istniejących systemów EDI.
- ▶ Ramy architektury technicznej:
 - ▶ sposoby implementacji repozytoriów, serwisów, itp.,
 - ▶ integracja z istniejącymi technologiami EDI.

XML w integracji aplikacji

- ▶ Cel: umożliwienie wymiany danych pomiędzy aplikacjami:
 - ▶ aplikacje/komponenty/moduły posługują się różnymi formatami wewnętrznymi,
 - ▶ wspólny mianownik: XML.
- ▶ Zastosowania:
 - ▶ komunikacja między klientem a serwerem,
 - ▶ komunikacja między elementami systemu rozproszonego,
 - ▶ integracja komponentów aplikacji,
 - ▶ konfigurowanie aplikacji i jej komponentów,
 - ▶ ...

Integracja aplikacji lokalnie lub globanie

Zastosowania „lokalne”

- ▶ w ramach jednego projektu / jednej instytucji
- ▶ komunikacja między różnymi modułami / aplikacjami
- ▶ być może w architekturze rozproszonej
- ▶ rozwiązania tworzone pod kątem projektu
- ▶ możliwość stosowania rozwiązań standardowych (np. JAXB)

Zastosowania „globalne”

- ▶ usługi dostępne w Internecie
- ▶ współpraca różnych partnerów
- ▶ ważna standaryzacja
- ▶ standard – Web Services

Usługi sieciowe (*Web Services*)

Definicja ogólna

- ▶ idea: „witryna WWW dla programów”
- ▶ komunikacja w wysokiej warstwie sieci
- ▶ tekstowy, strukturalny format komunikatów

Definicja konkretna

- ▶ opis interfejsu usługi: WSDL
- ▶ protokół komunikacji: SOAP (komunikaty w XML)
- ▶ rejestrowanie i wyszukiwanie usług: UDDI

Usługi sieciowe – możliwe zastosowania

- ▶ Udostępnianie/sprzedaż użytecznych danych:
 - ▶ rozkłady jazdy / lotów,
 - ▶ dane o pogodzie,
 - ▶ aktualny czas wg zegara atomowego.
- ▶ Zdalne usługi:
 - ▶ wyszukiwanie,
 - ▶ pobieranie aktualnej wersji oprogramowania,
 - ▶ e-administracja.
- ▶ Operacje biznesowe między partnerami:
 - ▶ rezerwacja biletów, miejsc w hotelach itp,
 - ▶ zamawianie towarów,
 - ▶ sprawdzanie stanu realizacji zamówienia,
 - ▶ elektroniczna wymiana danych.

Web Services – standaryzacja

- ▶ SOAP (pierwotnie *Simple Object Access Protocol*):
 - ▶ początki: 1998, Microsoft,
 - ▶ wersja 1.1: notatka W3C, 2000 (ciągle powszechnie w użyciu),
 - ▶ wersja 1.2: rekomendacje W3C, 2003 (errata: kwiecień 2007).
- ▶ Web Services Description Language:
 - ▶ wersja 1.1: notatka W3C, 2001 (ciągle powszechnie w użyciu),
 - ▶ wersja 2.0: rekomendacje W3C, 2007.
- ▶ Universal Description Discovery and Integration:
 - ▶ projekt organizacji OASIS.
- ▶ Standardy WS-* (różne organizacje: W3C, OASIS, IBM...)
 - ▶ Web Services Interoperability – ramy dla działania Web Serwisów:
WS-I Basic Profile, Simple Soap Binding Profile, ... ,
 - ▶ WS-Eventing, WS-Addressing, WS-Routing, WS-Security....
- ▶ Business Process Execution Language (OASIS) – opis semantyki i łączenie serwisów w super-serwisy.

SOAP – protokół komunikacji

- ▶ Protokół transportowy – HTTP lub inne protokoły.
- ▶ Format komunikatów – XML
 - ▶ ogólna struktura wyspecyfikowana w SOAP
 - ▶ konkretne nagłówki i komunikaty dowolne (osobne przetrzenie nazw)
- ▶ SOAP Encoding – sposób zapisywania typów danych w XML
 - ▶ obecnie niezalecany, zamiast tego kodowanie „literal” – oparte o typy danych XML Schema
- ▶ Różnice w stosunku do RPC, CORBA, DCOM itp.:
 - ▶ reprezentacja danych niezależna od platformy (tekstowa),
 - ▶ typy danych niezależne od platformy,
 - ▶ niższa efektywność.

Komunikat SOAP

- ▶ Dokument XML, jedna wiadomość:
 - ▶ przestrzeń nazw (dla wersji 1.2)
<http://www.w3.org/2001/12/soap-envelope>,
 - ▶ element główny: Envelope.
- ▶ Główne składniki dokumentu:
 - ▶ header – opcjonalny,
 - ▶ body – obowiązkowy.
- ▶ Ograniczenia:
 - ▶ brak DTD (i referencji do encji),
 - ▶ brak instrukcji przetwarzania.

Nagłówek SOAP

- ▶ actor – opcjonalny identyfikator strony komunikacji (URI),
- ▶ mustUnderstand – czy zrozumienie jest konieczne (0/1).

Przykład z W3Schools

```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Header>
    <m:Trans
      xmlns:m="http://www.w3schools.com/transaction/"
      soap:actor="http://www.w3schools.com/appml/"
      soap:mustUnderstand="1">234</m:Trans>
    </soap:Header>

    ...
  </soap:Envelope>
```

Ciało komunikatu SOAP

- ▶ wywołanie zdalnej procedury
- ▶ parametry
- ▶ encodingStyle – sposób kodowania danych (URI)

Zapytanie – przerobiony przykład z W3Schools

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope">

  <soap:Body>
    <m:GetPrice xmlns:m="http://www.w3schools.com/prices"
      soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
      <m:Item>Apples</m:Item>
      <m:Currency>PLN</m:Currency>
    </m:GetPrice>
  </soap:Body>

</soap:Envelope>
```

Ciało komunikatu SOAP

- ▶ wynik procedury
- ▶ parametry wyjściowe

Odpowiedź normalna – przerobiony przykład z W3Schools

```
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
  <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
    <m:Currency>PLN</m:Currency>
  </m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

SOAP – odpowiedź informująca o błędzie

- ▶ standardowy kod błędu
- ▶ krótki opis tekstowy
- ▶ dodatkowe dane (potencjalnie strukturalne)

Odpowiedź informująca o błędzie

```
<soap:Envelope xmlns:usos="urn:USOS"
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
  <soap:Fault>
    <soap:faultcode>Receiver</soap:faultcode>
    <soap:faultstring>Brak danych</soap:faultstring>
    <soap:faultdetail>Nie znaleziono studenta o numerze indeksu
      <usos:ind>123</usos:ind></soap:faultdetail>
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

Informacja zwrotna o błędzie

- ▶ Element(y) fault w body.
- ▶ Podelementy:
 - ▶ faultcode
 - ▶ faultstring
 - ▶ faultactor
 - ▶ detail
- ▶ Standardowe kody błędów:
 - ▶ VersionMismatch
 - ▶ MustUnderstand
 - ▶ DataEncodingUnknown
 - ▶ Sender
 - ▶ Receiver

WSDL – opis serwisu

Struktura

- ▶ dokument XML, element główny: `definitions`,
- ▶ przestrzeń nazw
`http://schemas.xmlsoap.org/wsdl/`,
- ▶ bloki definiujące pojęcia, od abstrakcyjnych do konkretnych.

Bloki definicji WSDL

- ▶ `types` – definicje typów i elementów (XML Schema),
- ▶ `message` – opis typu komunikatu,
- ▶ `portType` – typ „portu”, czyli zbiór operacji, operacje odwołują się do komunikatów,
- ▶ `binding` – powiązanie typu portu z protokołem transportowym,
- ▶ `service` – zbiór konkretnych instancji „portów” działających pod określonymi adresami.

Przykład

Początek i typy

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://example.org/kalkulator/" name="kalkulator"
  targetNamespace="http://example.org/kalkulator/">
  <wsdl:types>
    <xsd:schema targetNamespace="http://example.org/kalkulator/">
      <xsd:complexType name="two-ints">
        <xsd:sequence>
          <xsd:element name="arg1" type="xsd:int" />
          <xsd:element name="arg2" type="xsd:int" />
        </xsd:sequence>
      </xsd:complexType>
      ...
      <xsd:element name="iadd-request" type="tns:two-ints" />
      <xsd:element name="iadd-response" type="tns:one-int" />
      ...
    </wsdl:types>
```

Przykład c.d.

Komunikaty i operacje – wersja dla stylu Document

```
<wsdl:message name="iadd-req">
  <wsdl:part element="tns:iadd-request" name="parameters" />
</wsdl:message>

<wsdl:message name="iadd-resp">
  <wsdl:part element="tns:iadd-response" name="parameters" />
</wsdl:message>
...

<wsdl:portType name="kalkulator">
  <wsdl:operation name="iadd">
    <wsdl:input message="tns:iadd-req" />
    <wsdl:output message="tns:iadd-resp" />
  </wsdl:operation>
  ...
</wsdl:portType>
```

Przykład c.d.

Komunikaty i operacje – wersja dla stylu RPC

```
<wsdl:message name="iadd-req">
  <wsdl:part name="x" type="xsd:int" />
  <wsdl:part name="y" type="xsd:int" />
</wsdl:message>

<wsdl:message name="iadd-resp">
  <wsdl:part name="result" type="xsd:int" />
</wsdl:message>

...

<wsdl:portType name="kalkulator">
  <wsdl:operation name="iadd">
    <wsdl:input message="tns:iadd-req" />
    <wsdl:output message="tns:iadd-resp" />
  </wsdl:operation>
</wsdl:portType>

...
```

Przykład c.d.

- ▶ style – rpc lub document,
- ▶ transport – protokół transportowy (URI).
- ▶ soapAction – akcja SOAP (zwykle przesyłana w nagłówku HTTP) odpowiadająca operacji.

Wiązanie z protokołem SOAP (tu 1.1)

```
<binding type="kalkulator" name="kalkulator_soap">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http" />
  <operation>
    <soap:operation
      soapAction="http://example.com/iadd"/>
    <input> <soap:body use="literal"/> </input>
    <output> <soap:body use="literal"/> </output>
  </operation>
</binding>
```

Przykład c.d.

Instancja usługi

```
<wsdl:service name="kalkulator">
  <wsdl:port binding="tns:kalkulator_soap" name="kalkulatorSOAP">
    <soap:address
      location="http://localhost:8080/MyService/Kalkulator" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Wyszukiwanie usług

- ▶ Pomysł
 - ▶ dostawcy usług rejestrują je w katalogu,
 - ▶ użytkownik znajduje usługę w katalogu,
 - ▶ najlepiej: opis usługi i wyszukiwanie semantyczne.
- ▶ Universal Description Discovery and Integration (UDDI):
 - ▶ usługa katalogowa, wiele powiązanych węzłów,
 - ▶ dostępna jako Web service (poprzez SOAP),
 - ▶ opisy usług w WSDL.
- ▶ W praktyce rzadko stosowane.
- ▶ Idea publicznych rejestrów usług i globalnego wyszukiwania usług na razie się nie sprawdza.

Implementacja usług sieciowych

- ▶ Standardy
 - ▶ niezależne od platformy czy języka programowania,
 - ▶ oparte o istniejące standardy i protokoły.
- ▶ Dobrze, wysokopoziomowe wsparcie co najmniej dla Javy i .NET.
- ▶ Problemy:
 - ▶ rozmiar API i bibliotek (zrozumienie, efektywność),
 - ▶ niekompatybilne rozszerzenia dostawców.

Implementacja usług sieciowych w Javie

- ▶ Niskopoziomowo
 - ▶ serwlety, ręczna obsługa XML,
 - ▶ **SOAP with Attachments API for Java** – komunikat SOAP jako drzewo DOM.
- ▶ Wysokopoziomowo – **Java API for XML Web Services**
 - ▶ programista nie musi „widzieć” komunikacji SOAP,
 - ▶ abstrakcja zdalnego wywoływania procedur,
 - ▶ tworzenie usługi (w tym WSDL) na podstawie kodu Javy (*bottom-up*),
 - ▶ tworzenie klienta i szkieletu serwera na podstawie WSDL (*top-down*),
 - ▶ JAXB do tłumaczenia danych między Javą a XML,
 - ▶ niezbędne wsparcie implementacji, zwykle dostępne na serwerze aplikacji (JBoss, Glassfish, ...).

JAX-WS – przykład

Kawałek implementacji w JAX-WS

```
@WebService(targetNamespace = "http://example.org/kalkulator/", name = "kalkulator")
@SOAPBinding(style=SOAPBinding.Style.DOCUMENT)
    parameterStyle=SOAPBinding.ParameterStyle.BARE)
public class Kalkulator {
    @WebResult(name = "iadd-response", targetNamespace = "http://jws.javatech.pl/kalkulator")
    @WebMethod(action = "http://jws.javatech.pl/kalkulator/iadd")
    public OneInt iadd(
        @WebParam(partName = "parameters", name = "iadd-request", targetNamespace = "http://jws.javatech.pl/kalkulator")
        TwoInts parameters
    ){
        OneInt result = new OneInt();
        result.setArg(parameters.getArg1() + parameters.getArg2());
        return result;
    }
}
```

AJAX

- ▶ **AJAX – Asynchronous JavaScript and XML:**
 - ▶ strona WWW „działa” w przeglądarce użytkownika dzięki JavaScript,
 - ▶ komunikacja z serwerem po HTTP,
 - ▶ komunikaty kodowane w XML,
 - ▶ coraz szerzej stosowane (m.in. wiele usług Google).
- ▶ **Zalety:**
 - ▶ interakcja z użytkownikiem bez przeładowywania strony,
 - ▶ przesyłane tylko niezbędne dane.
- ▶ **Wady:**
 - ▶ narzut na przesłanie i uruchomienie skryptu,
 - ▶ nieintuicyjne zachowanie stron WWW (ale się przyzwyczajamy).
- ▶ **Odstępstwa w praktyce:**
 - ▶ inne „scripty”,
 - ▶ dane nie zawsze w XML (np. JSON).

XML w bezpieczeństwie

Najważniejsze aspekty bezpieczeństwa

poufność realizacja: szyfrowanie

uwierzytelnienie realizacja: podpis elektroniczny

Standardy XML związane z bezpieczeństwem

- ▶ Podpisy – XML Signature.
- ▶ Szyfrowanie – XML Encryption.
- ▶ Zastosowanie: m.in. w Web Services.

XML Signature

- ▶ Podpis dokumentu XML-owego zapisany w postaci struktury XML-owej.
- ▶ Podpis umieszczany w elemencie `Signature`:
 - ▶ w osobnym dokumencie (*detached signature*),
 - ▶ dołączonym do podpisywanego dokumentu (*enveloped signature*),
 - ▶ zawierającym podpisywane dane (*enveloping signature*).
- ▶ Możliwości XML Signature:
 - ▶ podpisywanie fragmentów dokumentu XML,
 - ▶ podpisywanie zasobów zewnętrznych (dostępnych poprzez URL),
 - ▶ podpisy wielokrotne.

XML Encryption

- ▶ Cel: zagwarantowanie poufności danych w XML.
- ▶ Możliwości: szyfrowanie zarówno całego pliku XML jak i jego części.

Klasyfikacja wsparcia dla XML-a w bazach danych

- ▶ Baza danych ze wsparciem dla XML
(zwykle relacyjna, czasem „obiektowa” bądź inna)
 - ▶ konfiguracja struktur danych jako tabel i powiązań,
 - ▶ eksport i import danych w postaci dokumentów XML,
 - ▶ struktura dokumentów XML pochodną relacyjnych struktur danych,
 - ▶ zastosowanie: integracja, wymiana danych;
- ▶ XML-owa baza danych:
 - ▶ konfiguracja struktur danych przy pomocy DTD/XML Schema,
 - ▶ wyszukiwanie z użyciem XQuery lub XPath,
 - ▶ indeksowanie elementów, atrybutów, wyrażeń XPath,
 - ▶ struktura danych – dokumenty XML,
 - ▶ możliwa optymalizacja wewnętrznego formatu danych,
 - ▶ zastosowanie: przechowywanie i przetwarzanie dokumentów strukturalnych.

XML w relacyjnych bazy danych

- ▶ Korzyści:
 - ▶ integracja aplikacji, wymiana danych,
 - ▶ łatwe transformacje danych,
 - ▶ prezentacja danych.
- ▶ Problemy:
 - ▶ czy i jak przechowywać dokumenty XML w bazie danych?
 - ▶ metody dostępu (zadawania zapytań),
 - ▶ efektywność.

XML a relacyjne bazy danych

- ▶ Przechowywanie XML-a w relacyjnych bazach danych:
 - ▶ elementy dokumentu XML jako pola tabeli bazodanowej (dokument „rozłożony na czynniki pierwsze”),
 - ▶ dokument XML w całości przechowywany w polu bazy danych.
- ▶ Sposoby wspierania XML-a przez systemy zarządzania bazami danych:
 - ▶ generowanie XML-a na podstawie zawartości bazy danych,
 - ▶ wypełnianie zawartości bazy na podstawie zawartości dokumentu XML,
 - ▶ specjalne indeksowanie pól zawierających XML,
 - ▶ zapytania XPath/XQuery na danych typu XML,
 - ▶ wbudowane parsery XML i procesory XSLT.

XML-owa baza danych

- ▶ Warstwa logiczna:
 - ▶ dokument XML jako podstawowa jednostka przechowywanych danych,
 - ▶ schemat jako definicja struktury,
 - ▶ kolekcje dokumentów.
- ▶ Warstwa fizyczna:
 - ▶ niekoniecznie tekstowo zapisane dokumenty XML,
 - ▶ indeksy.
- ▶ Funkcjonalności typowe dla baz danych:
 - ▶ interfejs do aktualizacji danych,
 - ▶ transakcje i współbieżny dostęp,
 - ▶ bezpieczeństwo.