

# Schematy XML

Tomasz Traczyk

[ttraczyk@ia.pw.edu.pl](mailto:ttraczyk@ia.pw.edu.pl)

<http://www.ia.pw.edu.pl/~ttraczyk/>

# Plan prezentacji

- Dlaczego schematy?
- Przykład schematu
- *XML Schema* – nieco szczegółów
- Rola schematów XML

# Problemy z DTD

- Zalety DTD
  - Prostota
  - Zgodność z SGML
- DTD a przetwarzanie danych
  - Do wyświetlania zwykle wystarczy poprawność *well formed*
  - Do przetwarzania postuluje się poprawność *valid*
  - DTD wystarcza do definiowania dokumentów „tekstowych”
  - DTD nie wystarcza do definiowania złożonych struktur danych, np. w
    - EDI
    - handlu elektronicznym B2B i B2C
  - DTD nie daje możliwości sprawdzania poprawności typów danych za pomocą standardowych narzędzi
- Niedostatki DTD
  - Brak definicji typów danych dla zawartości elementów i dla atrybutów
    - możliwe ograniczenia to tylko:
      - wyliczanie dopuszczalnych wartości atrybutu
      - zasady tworzenia nazw i identyfikatorów
    - zawartość elementu: zawsze tekst
  - Ograniczone możliwości sformalizowanej rozbudowy
    - encje parametryczne
  - Brak możliwości efektywnego wyrażenia identyczności kilku części dokumentu
    - encje w DTD rozwijane przed analizą dokumentu
  - Brak dobrego wykorzystania przestrzeni nazw
    - przedrostki wpisane „na sztywno” do DTD
  - Język zapisu DTD
    - zupełnie różny od XML
    - nie daje możliwości przetwarzania narzędziami XML-owymi

# Idea schematów

- Co to jest schemat?
  - Definicja składni dokumentu XML
  - Stosowana zamiast (lub obok) DTD
  - Zapisana w XML, z użyciem przestrzeni nazw
- XML Schema
  - Definicja składni dokumentu
    - nie mająca w/w wad
    - spełniająca dodatkowe n/w postulaty
  - Standard W3C (rekomendacja)
  - Sposób definiowania dokumentów, który prawdopodobnie zastąpi DTD

# Cechy schematów

- Postulowane cechy schematu
  - Precyzyjne deklarowanie typów danych
    - z wykorzystaniem rozbudowanego słownika typów elementarnych
    - z możliwością definiowania własnych typów
  - Mechanizmy jednokrotnego definiowania powtarzających się fragmentów modelu (np. typów, grup elementów i atrybutów) i wielokrotnego odwoływania się do takich definicji
  - Możliwość definiowania zbiorów elementów, w których liczba wystąpień każdego elementu jest kontrolowana, ale kolejność jest dowolna
  - Możliwość deklarowania niepowtarzalności wybranych wartości w określonej części dokumentu, np. definiowanie kluczy i odwołań do nich
  - Możliwość deklarowania wielu elementów o takiej samej nazwie, ale innym położeniu w dokumencie i innej budowie
  - Mechanizmy pozwalające na kontrolowane rozszerzanie i uszczegóławianie modeli dokumentów; korzystanie z wielu schematów w jednym dokumencie
  - Uwzględnienie przestrzeni nazw; komponowanie nowych modeli z kilku przestrzeni nazw

# Wady schematów

- Stałe
  - Większa długość od odpowiadającego DTD
  - Znacznie bardziej skomplikowana składnia
  - Brak możliwości definiowania encji
- Przejściowe
  - Niewielka popularność
  - Niewielu ekspertów umiejących wykorzystywać schematy
  - Narzędzia nie zawierające wsparcia dla schematów

# Plan prezentacji

- Dlaczego schematy?

- Przykład schematu

- *XML Schema* – nieco szczegółów

- Rola schematów XML

# Przykład dokumentu

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<?xml-stylesheet type="text/xsl" href="konspekty.xsl"?>
<eres_konspekty
  xmlns="http://www.elka.pw.edu.pl/eres"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.elka.pw.edu.pl/eres konspekty.xsd">
  <przedmiot id="KBD2" wersja="1">
    <slowo_kluczowe>bazy danych</slowo_kluczowe>
    <slowo_kluczowe>Oracle</slowo_kluczowe>
    <konspekt>
      <czesc_konspektu id="Streszczenie">
        <P>Monograficzny przedmiot poświęcony bazie danych i narzędziom Oracle.</P>
      </czesc_konspektu>
      <czesc_konspektu id="Treść">
        <P>Omawiane są podstawowe zagadnienia związane z wykorzystaniem RDBMS
        Oracle8 i Oracle8<I>i</I>, w tym możliwości wykorzystania języka XML.</P>
        <P>Przedstawiane są także narzędzia Oracle:</P>
        <UL>
          <LI> Oracle Forms, </LI>
          <LI> Oracle Reports, </LI>
          <LI> Oracle XDK. </LI>
        </UL>
      </czesc_konspektu>
    </konspekt>
  </przedmiot>
</eres_konspekty>
```

# Przykład dokumentu, c.d.

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<?xml-stylesheet type="text/xsl" href="konspekty.xsl"?>
<eres_konspekty
  xmlns="http://www.elka.pw.edu.pl/eres"
  xmlns:xsi="http://www.
  xsi:schemaLocation="ht
<przedmiot id="KBD2" wers
  <slowo_kluczowe>bazy dan
  <slowo_kluczowe>Oracle</
  <konspekt>
    <czesc_konspektu id="Streszczenie">
      <P>Monograficzny przedmiot poświęcony bazie danych i narzędziom Oracle.</P>
    </czesc_konspektu>
    <czesc_konspektu id="Treść">
      <P>Omawiane są podstawowe zagadnienia związane z wykorzystaniem RDBMS
      Oracle8 i Oracle8<I>i</I>, w tym możliwości wykorzystania języka XML.</P>
      <P>Przedstawiane są także narzędzia Oracle:</P>
      <UL>
        <LI> Oracle Forms, </LI>
        <LI> Oracle Reports, </LI>
        <LI> Oracle XDK. </LI>
      </UL>
    </czesc_konspektu>
  </konspekt>
</przedmiot>
</eres_konspekty>
```

## Domyślna przestrzeń nazw

- Nie wymaga prefiksów w dokumencie
- Jest zgodna z przestrzenią docelową schematu

# Przykład dokumentu, c.d.

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<?xml-stylesheet type="text/xsl" href="konspekty.xsl"?>
<eres_konspekty
  xmlns="http://www.elka.pw.edu.pl/eres"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.elka.pw.edu.pl/eres_konspekty.xsd">
  <przedmiot id="KBD2" wersja="1.0">
    <slowo_kluczowe>bazy danych</slowo_kluczowe>
    <slowo_kluczowe>Oracle</slowo_kluczowe>
    <konspekt>
      <czesc_konspektu id="Streszczenie">
        <P>Monograficzny przedmiot poświęcony bazie danych i narzędziom Oracle.</P>
      </czesc_konspektu>
      <czesc_konspektu id="Treść">
        <P>Omawiane są podstawowe zagadnienia związane z wykorzystaniem RDBMS
        Oracle8 i Oracle8<I>i</I>, w tym możliwości wykorzystania języka XML.</P>
        <P>Przedstawiane są także narzędzia Oracle:</P>
        <UL>
          <LI> Oracle Forms, </LI>
          <LI> Oracle Reports, </LI>
          <LI> Oracle XDK. </LI>
        </UL>
      </czesc_konspektu>
    </konspekt>
  </przedmiot>
</eres_konspekty>
```

## Przestrzeń nazw XML Schema

- Definiuje atrybut **xsi:schemaLocation**

# Przykład dokumentu, c.d.

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<?xml-stylesheet type="text/xsl" href="konspekty.xsl"?>
<eres_konspekty
  xmlns="http://www.elka.pw.edu.pl/eres"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.elka.pw.edu.pl/eres konspekty.xsd">
  <przedmiot id="KBD2" wersja=
    <slowo_kluczowe>bazy danych</slowo_kluczowe>
    <slowo_kluczowe>Oracle</slowo_kluczowe>
    <konspekt>
      <czesc_konspektu id="Streszczenie">
        <P>Monograficzny przedmiot</P>
      </czesc_konspektu>
      <czesc_konspektu id="Treść">
        <P>Omawiane są podstawowe zagadnienia związane z wykorzystaniem RDBMS
        Oracle8 i Oracle8<I>i</I>, w tym możliwości wykorzystania języka XML.</P>
        <P>Przedstawiane są także narzędzia Oracle:</P>
        <UL>
          <LI> Oracle Forms, </LI>
          <LI> Oracle Reports, </LI>
          <LI> Oracle XDK. </LI>
        </UL>
      </czesc_konspektu>
    </konspekt>
  </przedmiot>
</eres_konspekty>
```

## Określenie schematu

- Dwa parametry:
  - nazwa URI przestrzeni nazw
  - URL lub nazwa pliku ze schematem

# Przykład schematu

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<xsd:schema      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                 targetNamespace="http://www.elka.pw.edu.pl/eres"
                 xmlns="http://www.elka.pw.edu.pl/eres"
                 elementFormDefault="qualified"
                 version="1.1">
  <xsd:include schemaLocation="teksty.xsd"/>
  <xsd:element name="eres_konspekty">
    ...
  </xsd:element>
  <xsd:element name="przedmiot">
    ...
  </xsd:element>
  <xsd:element name="czesc_konspektu">
    ...
  </xsd:element>
  <xsd:attributeGroup name="identyfikatory">
    ...
  </xsd:attributeGroup>
</xsd:schema>
```

# Przykład schematu, c.d.

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.elka.pw.edu.pl/eres"
  xmlns="http://www.elka.pw.edu.pl/eres"
  elementFormDefault="qualified"
  version="1.1">
  <xsd:include schemaLocation="teksty_xsd"/>
  <xsd:element name="..." />
  <xsd:element name="..." />
  <xsd:element name="..." />
  <xsd:attributeGroup name="identyfikatory">
    ...
  </xsd:attributeGroup>
</xsd:schema>
```

## Element główny schematu

- Odwołanie do przestrzeni nazw **xsd**
- Określenie docelowej przestrzeni nazw dla dokumentu
- Określenie domyślnej przestrzeni nazw dla schematu
- Żądanie kwalifikowania wszystkich elementów dokumentu prefiksem przestrzeni nazw

# Przykład schematu, c.d.

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.elka.pw.edu.pl/eres"
  xmlns="http://www.elka.pw.edu.pl/eres"
  elementFormDefault="qualified"
  version="1.1">
  <xsd:include schemaLocation="teksty.xsd"/>
  <xsd:element name="eres"
    ...
  </xsd:element>
  <xsd:element name="prze"
    ...
  </xsd:element>
  <xsd:element name="czes"
    ...
  </xsd:element>
  <xsd:attributeGroup name="identyfikatory">
    ...
  </xsd:attributeGroup>
</xsd:schema>
```

## Włączenie schematu

- Włącza schemat zawierający typowe deklaracje i definicje, powtarzające się w innych schematach
- Tak włączany schemat musi nie określać docelowej przestrzeni nazw lub określać ją tak samo jak schemat włączający

# Przykład schematu, c.d.

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.elka.pw.edu.pl/eres"
  xmlns="http://www.elka.pw.edu.pl/eres"
  elementFormDefault="qualified"
  version="1.1">
  <xsd:include schemaLocation="teksty.xsd"/>
  <xsd:element name="eres_konspekty">
    ...
  </xsd:element>
  <xsd:element name="prze
    ...
  </xsd:element>
  <xsd:element name="czes
    ...
  </xsd:element>
  <xsd:attributeGroup name="identyfikatory">
    ...
  </xsd:attributeGroup>
</xsd:schema>
```

## Deklaracje i definicje zawartości

- Deklaracje określają budowę dokumentu
- Definicje określają obiekty pomocnicze, np. typy, powtarzalne grupy itp. – wykorzystywane w deklaracjach

# Deklarowanie elementów

```
<xsd:element name="eres_konspekty">  
  <xsd:annotation>  
    <xsd:documentation>Przykład</xsd:documentation>  
  </xsd:annotation>
```

```
<xsd:complexType>  
  <xsd:sequence>  
    <xsd:element ref="...">  
  
  </xsd:sequence>  
</xsd:complexType>
```

## Dokumentacja

- Opisy w specjalnej strukturze
- Zwykle komentarze XML (trudniejsze do przetwarzania!)

```
<xsd:key name="id_wersji">  
  <xsd:selector xpath="przedmiot"/>  
  <xsd:field xpath="@id"/>  
  <xsd:field xpath="@wersja"/>  
</xsd:key>  
</xsd:element>
```

```
<xsd:element name="przedmiot">  
  <!-- Definicja wywoływana po nazwie -->  
  ...  
</xsd:element>
```

# Deklarowanie elementów, c.d.

```
<xsd:element name="eres_konspekty">
  <xsd:annotation>
    <xsd:documentation>Przykład</xsd:documentation>
  </xsd:annotation>
```

```
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="przedmiot"
        minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
```

```
  <xsd:key name="id_wers
    <xsd:selector xpath=
    <xsd:field xpath="@i
    <xsd:field xpath="@w
  </xsd:key>
</xsd:element>
```

## Definicje struktur

- Typy złożone służą do definiowania struktur zagnieżdżonych
- Element **sequence** pozwala definiować następstwo
- Można ściśle określić krotności

```
<xsd:element name="przedmiot">
  <!-- Definicja wywoływana po nazwie -->
  ...
</xsd:element>
```

# Deklarowanie elementów, c.d.

```
<xsd:element name="przedmiot">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="slovo_kluczowe" type="xsd:string"
                  minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="konspekt">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="czesc_konspektu"
                        minOccurs="1" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:attributeGroup ref="...">
    <xsd:attribute name="w" type="xsd:string">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxExclusive value="...">
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
  ...
</xsd:element>

<xsd:element name="czesc_konspektu">
  ...
</xsd:element>
```

## Metody deklarowania

- Bezpośrednia (*inline*)
- Z użyciem definicji i odwołań przez nazwę

# Deklarowanie atrybutów

```
<xsd:element name="przedmiot">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="slowo_kluczowe" type="xsd:string"
                  minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="konspekt">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="..." type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attributeGroup name="...">
      <xsd:attribute name="wersja">
        <xsd:simpleType>
          <xsd:restriction base="xsd:unsignedByte">
            <xsd:maxExclusive value="10"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:attributeGroup>
  </xsd:complexType>
</xsd:element>

<xsd:element name="czesc_konspektu">
  ...
</xsd:element>
```

## Definicje typów

- Do określania typów/attributów należy użyć typów prostych
- Typ prosty bazuje na jednym z typów wbudowanych
- Typ może być uściślony przez różne ograniczenia (aspekty)



# Klucze

```
<xsd:element name="przedmiot">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="slovo_kluczowe" type="string" minOccurs="0" maxOccurs="1" />
      <xsd:element name="konspekt">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="czesc_konspektu" type="string" minOccurs="0" maxOccurs="1" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attributeGroup ref="identyfikatory" />
  </xsd:complexType>
  <xsd:key name="id_czesci">
    <xsd:selector xpath="konspekt/czesc_konspektu" />
    <xsd:field xpath="@id" />
  </xsd:key>
</xsd:element>

<xsd:attributeGroup name="identyfikatory">
  <xsd:attribute name="id" type="xsd:Name" use="required" />
</xsd:attributeGroup>
```

## Definicja klucza

- Nazwa klucza (może służyć do budowy odwołań)
- Selektor: które elementy są unikalnie identyfikowane przez klucz
- Pole: składnik klucza (może być kilka)
- Używa się wyrażeń XPath
- Zakres unikalności: element nadrzędny w stosunku do deklaracji

# Plan prezentacji

- Dlaczego schematy?
- Przykład schematu
- *XML Schema* – nieco szczegółów
- Rola schematów XML

# Budowa schematu *XML Schema*

- Rodzaje elementów schematu
  - Deklaracje: określają elementy i atrybuty dokumentu
  - Definicje: określają elementy pomocnicze, używane w deklaracjach: typy, grupy itp.
- Elementy globalne
  - Elementy schematu bezpośrednio należące do elementu głównego **schema**
  - Tworzą nazwane definicje, do których można odwoływać się w definicjach i deklaracjach
- Typy
  - Służą do określania
    - nazw elementów i atrybutów, następstwa, zawierania
    - typów danych i zakresów wartości dla atrybutów i zawartości elementów
  - Rodzaje
    - proste: bez zagnieżdżeń (np. dla atrybutów)
    - złożone: zawierają zagnieżdżone elementy
- Dokumentacja
  - Elementy **annotation** służą do umieszczania dodatkowych informacji
    - element **documentation** może zawierać dokumentację w języku naturalnym
    - element **appInfo** może zawierać informacje dla aplikacji przetwarzających

# Złożone typy danych

- Budowanie złożonych typów danych
  - Element **complexType** służy do definiowania typów złożonych
  - Grupowanie elementów w ramach typu
    - **sequence** wyznacza ścisłe następstwo
    - **choice** wyznacza wybór z kilku możliwości
    - **all** wyznacza zbiór elementów i liczbę powtórzeń, ale bez określenia kolejności
    - grupowania te można zagnieżdżać
  - Liczba powtórzeń elementu lub grupy
    - ściśle określana za pomocą atrybutów **minOccurs** i **maxOccurs**
  - Elementy mieszane (tekst + znaczniki)
    - definiuje się jak zwykłe typy złożone
    - atrybut **mixed** określa, że znaczniki mogą być przeplacone tekstem
- Atrybuty
  - Deklaracje atrybutów podaje się na końcu definicji złożonego typu danych
  - Atrybut **use="required"** deklaracji wymusza obowiązkowość
  - Atrybut **default** deklaracji określa wartość domyślną
  - Atrybut **fixed** deklaracji narzuca wartość stałą

# Proste typy danych

- Typy wbudowane
  - Typy elementarne
  - Podtypy – typy pochodne
- Budowanie nowych typów prostych
  - Ograniczenie (**restriction**)
    - wzorce (wyrażenia regularne)
    - ograniczenia długości
    - określenie liczby cyfr dziesiętnych
    - ograniczenia wartości
  - Wyliczenie (**enumeration**)
  - Połączenie (**union**)
- Listy
  - Wartość atrybutu lub elementu może być listą wartości prostego typu, rozdzielonych białymi znakami
  - Do deklarowania typu lista służy konstrukcja  
`<xsd:list itemType="typ-prosty" />`

# Proste typy danych, c.d.

## Typy wbudowane (wybór)

Nazwa typu	Opis	Przykładowe podtypy
<code>string</code>	Napis	<code>normalizedString</code> , <code>language</code> , <code>Name</code>
<code>boolean</code>	Wartości <code>true</code> i <code>false</code>	
<code>decimal</code>	Liczba stałoprzecinkowa	<code>integer</code> , <code>negativeInteger</code> , <code>nonPositiveInteger</code> , <code>int</code> , <code>short</code> , <code>byte</code> , <code>unsignedInt</code>
<code>float</code> , <code>double</code>	Liczba zmiennoprzecinkowa	
<code>duration</code>	Długość okresu czasu	
<code>dateTime</code>	Data i czas	
<code>date</code> , <code>time</code>	Data, czas	

# Unikalność i klucze

- Unikalność
  - Element **unique** deklaruje niepowtarzalność
- Klucze
  - Element **key**: podobny do **unique**, definiuje klucz złożony z niepustych składników
  - Odwołania można deklarować za pomocą elementu **keyref**
- Budowa klucza
  - Element **selector** podaje jaki zbiór elementów ma zawierać niepowtarzalne klucze
  - Element(y) **field** określa(ją) składnik(i) klucza
  - Zakres niepowtarzalności określa się przez umieszczenie deklaracji **unique** lub **key** na końcu definicji odpowiedniego elementu

# Schematy a przestrzenie nazw

- Przestrzenie nazw w schematach
  - Schematy w pełni wykorzystują możliwości przestrzeni nazw (*namespaces*)
  - Schemat określa znaczniki w konkretnej przestrzeni nazw
- Docelowa przestrzeń nazw
  - Znaczniki znajdują się w docelowej przestrzeni nazw podanej w atrybucie **targetNamespace**
  - Znaczniki w dokumencie muszą być w tej samej docelowej przestrzeni nazw
- Kilka przestrzeni nazw
  - Dokument może korzystać z kilku przestrzeni nazw – każdej powiązanej z innym schematem
- Kwalifikowanie znaczników
  - Atrybut **elementFormDefault**: reguły kwalifikowania prefiksem przestrzeni nazw
    - wartość **qualified** – kwalifikować wszystkie znaczniki
    - wartość **unqualified** – kwalifikować tylko elementy globalne
      - w przestrzeni nazw znajdują się tylko elementy globalne
      - ich elementy podrzędne (lokalne) przynależą do przestrzeni pośrednio
      - można tworzyć różne elementy lokalne o tych samych nazwach w różnych miejscach schematu bez konfliktu w przestrzeni nazw
- Rozszerzanie *XML Schema*
  - Zapis schematu można rozszerzyć o własne elementy
  - Umieścić je trzeba w innej przestrzeni nazw

# Wielokrotne użycie

- Grupowanie
  - Definicja powtarzalnej grupy
    - elementów: **group**
    - atrybutów: **attributeGroup**
  - Odwołanie: po nazwie grupy
- Włączanie schematów
  - **include** włącza schemat do schematu
  - **redefine** włącza schemat, umożliwiając przedefiniowanie wybranych elementów, atrybutów i typów włączanego schematu
- Włączanie a przestrzenie nazw
  - **include** działa jeśli
    - włączany schemat nie określa przestrzeni nazw
      - znaczniki włączane przypisywane są do przestrzeni docelowej schematu włączającego
    - włączany schemat określa tę samą przestrzeń nazw co włączający
  - **import** włącza schemat z pozostawieniem jego przestrzeni nazw
    - stosowane w przypadku, gdy włączany schemat określa inną przestrzeń nazw
    - w definiowanym dokumencie pozostają dwie różne przestrzenie nazw

# Rozszerzanie schematów

- Zastępowanie (*substitution*)
  - Można określić zbiór elementów mogących zastąpić dany element
  - Definiowanie
    - atrybut **substitutionGroup** wskazuje element zastępowany (bazowy)
    - typ elementów ten sam albo pochodny od typu elementu bazowego
  - Wykorzystanie w dokumencie
    - element typu bazowego może być zastąpiony elementem typu zastępującego
    - przykład: nazwy w kilku językach
- Typy pochodne (*derived types*)
  - Dla typów prostych: ograniczenie, wyliczenie, połączenie
  - Dla typów złożonych
    - ograniczenie
      - zawężenie zakresu wartości
      - ograniczenie liczności elementów
    - rozszerzenie: dodanie nowych elementów lub atrybutów do typu bazowego
- Ograniczanie rozszerzalności
  - Specjalne atrybuty pozwalają określić dopuszczone mechanizmy wyprowadzania typów
  - Definicje elementów i typów abstrakcyjnych
    - nie mogą być bezpośrednio wykorzystane w deklaracjach
    - służą do zastępowania lub wyprowadzania nowych definicji
- Zezwolenie na rozszerzenia użytkownika
  - **any**, **anyAttribute** i **anyType** pozwalają użyć elementów/atributów/typów nie zdefiniowanych w schemacie

# XML Schema w narzędziach

- Narzędzia firmy Microsoft

- MS XML SDK 4.0 wspiera specyfikację *XML Schema* (XSD)
  - parser DOM waliduje dokumenty z użyciem XSD
  - obiekt **XMLSchemaCache** służy do ładowania schematów
- Narzędzia Microsoft zawierają także wsparcie dla firmowych specyfikacji *XML-Data* i *XML-Data Reduced*

- Narzędzia firmy Oracle

- Najnowsze wersje XDK (*XML Developer Kit*) zawierają *Oracle XML Schema Processor for Java*
- Procesor ten jest wykorzystywany przez parser typu DOM
- Walidacja na podstawie schematu
  - ładowanego automatycznie na podstawie zawartości dokumentu
  - ładowanego programowo
- Walidacja z linii komendy  
**oraxml -schema**  
*nazwa\_pliku\_XML*

# Plan prezentacji

- Dlaczego schematy?
- Przykład schematu
- *XML Schema* – nieco szczegółów

- Rola schematów XML

# Rola schematów XML

- Zastosowania schematów XML
  - W przetwarzaniu
    - walidacja na podstawie schematu pozwala uniknąć sprawdzeń poprawności struktury w czasie przetwarzania
  - W wymianie danych
    - walidacja na podstawie schematu pozwala pod razę odrzucać błędne komunikaty, bez ich szczegółowej analizy
  - W ładowaniu do baz danych
    - schemat może być użyty do automatycznego utworzenia odpowiedniej struktury danych i sterowania wczytywaniem informacji do tej struktury
  - W tworzeniu dokumentacji
    - schemat można uzupełnić o elementy opisowe i automatycznie generować dokumentację struktur XML (za pomocą XSL)
- Przyszłość schematów XML
  - Wszystko wskazuje na to, że schematy całkowicie wyprą DTD w zastosowaniu do definiowania struktur dokumentów i walidacji
  - Zastosowaniem DTD pozostanie definiowanie encji