XML Schema

Irena Mlýnková

Karlova Univerzita
Matematicko-fyzikální fakulta
Katedra softwarového inženýrství
Malostranské náměstí 25
118 00 Praha 1

irena.mlynkova@mff.cuni.cz

Anotace. *Jedním z nejpoužívanějších nástrojů pro popis přípustné struktury XML dokumentů je jazyk DTD. Jeho obliba spočívá především v jeho jednoduchosti a pro většinu existujících aplikací postačující vyjadřovací síle. Pro složitější aplikace, nebo při nutnosti vyjádřit přípustnou strukturu XML dokumentů podrobněji, je již třeba použít nástroj podstatně silnější, např. jazyk XML Schema navržený konsorciem W3C. Specifikace zapsané v XML Schema (nazývané XML schémata) poskytují více informací pro zpracování XML dat v XML databázi než DTD. Článek obsahuje přehled nejdůležitějších prvků jazyka XML Schema, jejich základních vlastností a příkladů možných způsobů jejich použití.*

1 Úvod

Výměna a techniky efektivního zpracování informací měly vždy strategický význam a určovaly chod dějin. V dnešní době tomu není jinak. Pro vzájemnou komunikaci dvou (a více) subjektů je nezbytné, aby oba znaly strukturu předávaných dat a byly schopny pokud možno pružně reagovat na její případné změny. Především splnění druhého požadavku bývá obvykle problematické, zvláště pak v dnešní době elektronického zpracování dat.

V reakci na tento problém se právě jazyk XML (eXtensible Markup Language) [BT04] velmi rychle stává klíčovým standardem pro reprezentaci dat. Důvodem této tendence je zejména fakt, že XML není pouze jazyk pro popis dat samotných, ale zahrnuje i nástroje pro popis jejich struktury. A právě díky této vlastnosti se zpracování XML dat zásadním způsobem zjednodušuje.

Jedním z nejpoužívanějších nástrojů pro popis struktury XML dokumentů je jazyk DTD (Document Type Definition) [BT04]. Jeho velká obliba spočívá především v jeho jednoduchosti a pro většinu existujících aplikací postačující vyjadřovací síle. Ovšem pro složitější aplikace, nebo při potřebě vyjádřit strukturu XML dat přesněji, je již třeba použít nástroj podstatně silnější. Jedním z nich může být např. jazyk XML Schema [FDC01] [THS01] [BPV01] navržený konsorciem W3C[1] právě pro tyto účely. Specifikace v jazyce XML Schema (tzv. XML schémata[2]) poskytují podstatně více informací o struktuře XML dat než je možné získat z popisu v jazyce DTD. Objektově-orientované rysy tohoto jazyka (jako např. dědičnost, substituovatelnost atd.) navíc přináší výhody především při vytváření XML schémat, tj. při modelování reality, pro něž je objektově-orientovaný přístup poměrně přirozený. Naopak díky tomu, že jazyk XML Schema vychází z jazyka XML (tj. každé XML schéma popsané v jazyce XML Schema je současně XML dokumentem), je možné při zpracování XML schémat využít všech nástrojů vytvořených pro práci s XML dokumenty.

Tento článek obsahuje přehled nejdůležitějších prvků jazyka XML Schema, popis jejich základních vlastností a příkladů možných způsobů jejich použití. Cílem článku není vytvořit podrobné shrnutí specifikace jazyka, nýbrž nabídnout stručný přehled toho, co jazyk XML Schema nabízí a umožňuje.

---

[1] http://www.w3.org

[2] Je třeba rozlišovat mezi pojmy „XML schéma“ (tj. schéma struktury XML dat vyjádřené v libovolném jazyce, např. DTD, XML Schema atd.) a „XML Schema“ (tj. název jednoho z jazyků).

Článek je strukturován takto: Kapitola 2 stručně připomíná základní vlastnosti jazyka DTD a zmiňuje hlavní výhody a rozšíření jazyka XML Schema právě oproti DTD. V kapitole 3 jsou popsány základní prvky jazyka XML Schema, se kterými se zřejmě setká každý jeho běžný uživatel. Kapitola 4 pak obsahuje přehled pokročilejších prvků, díky nimž se XML Schema stává opravdu silným nástrojem pro popis struktury XML dat. Kapitola 5 obsahuje stručné zhodnocení popsaných prvků a závěr.

2 Motivace

Jak již bylo řečeno, jazyk DTD se stále těší velké oblibě, a to především po svoji jednoduchost a snadné použití. Umožňuje definovat:

- elementy a atributy, které se smí v XML dokumentu vyskytovat,
- vztahy element-podelement a element-atribut,
- pořadí a počet výskytů podelementů v rámci nadelementu,
- přípustný obsah elementu (prázdný, textový, elementový, smíšený...),
- datové typy, povinnost výskytu a implicitní hodnoty atributů.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT zamestnanci (osoba)+>

<!ELEMENT osoba (jmeno,email*,vztahy?)>
<!ATTLIST osoba id ID #REQUIRED>
<!ATTLIST osoba poznamka CDATA #IMPLIED>
<!ATTLIST osoba dovolena (ano|ne) "ne">

<!ELEMENT jmeno ((krestni,prijmeni)|(prijmeni,krestni))>

<!ELEMENT krestni (#PCDATA)>

<!ELEMENT prijmeni (#PCDATA)>

<!ELEMENT email (#PCDATA)>

<!ELEMENT vztahy EMPTY>
<!ATTLIST vztahy nadrizeny IDREF #IMPLIED>
<!ATTLIST vztahy podrizeni IDREFS #IMPLIED>
```

V příkladu je definován element `zamestnanci` s elementovým obsahem sestávajícím z alespoň jednoho elementu `osoba`. Element `osoba` má také elementový obsah, který sestává z právě jednoho elementu `jmeno`, libovolného počtu elementů `email` a nepovinného elementu `vztahy`, které se musí vyskytovat právě v tomto pořadí. Element `osoba` má dále tři atributy – v rámci dokumentu jednoznačný a povinný identifikátor `id`, nepovinný atribut `poznamka` s textovým obsahem a nepovinný atribut `dovolena` s přípustnými hodnotami `ano` nebo `ne` a s implicitní hodnotou `ne`. Element `jmeno` má elementový obsah a obsahuje právě dvojici elementů `krestni` a `prijmeni` v libovolném pořadí. Elementy `krestni`, `prijmeni` a `email` mají textový obsah. Element `vztahy` má prázdný obsah a dva nepovinné atributy – atribut `nadrizeny` obsahující odkaz (tj.

hodnotu atributu `id`) na některý z elementů `osoba` (tj. na nadřízeného dané osoby) a atribut `podrizeni` obsahující seznam odkazů na elementy osoba (tj. na podřízené dané osoby).[3]

Příklad XML dokumentu vyhovujícího tomuto DTD by mohl vypadat takto:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE zamestnanci SYSTEM "zamestnanci.dtd">
<zamestnanci>
  <osoba id="nacelnik" poznamka="Nemecky Cech" dovolena="ano">
    <jmeno>
      <krestni>Karel</krestni>
      <prijmeni>Nemec</prijmeni>
    </jmeno>
    <email>karel.nemec@cimr.cz</email>
    <vztahy podrizeni="lekarnik ucitel nosic"/>
  </osoba>

  <osoba id="lekarnik" poznamka="Domu! Do Prahy! Do Podoli!">
    <jmeno>
      <krestni>Vojtech</krestni>
      <prijmeni>Sofr</prijmeni>
    </jmeno>
    <email>vojtech.sofr@cimr.cz</email>
    <vztahy nadrizeny="nacelnik"/>
  </osoba>

  <osoba id="ucitel">
    <jmeno>
      <krestni>Vaclav</krestni>
      <prijmeni>Poustka</prijmeni>
    </jmeno>
    <email>vaclav.poustka@cimr.cz</email>
    <vztahy nadrizeny="nacelnik"/>
  </osoba>

  <osoba id="nosic" poznamka="Ja bych sned’ i Kratochvila...">
    <jmeno>
      <prijmeni>Fristensky</prijmeni>
      <krestni>Varel</krestni>
    </jmeno>
    <vztahy nadrizeny="nacelnik"/>
  </osoba>
</zamestnanci>
```

Na první pohled se zdá, že specifikace obsahu XML dokumentu pomocí DTD je postačující a jasně modeluje danou skutečnost. Při bližším pohledu je ale už i v tomto jednoduchém příkladě možné nalézt několik problémů. Například pomocí DTD není možné specifikovat jak by měl vypadat správný tvar e-mailové adresy (tj. obsahem elementu `email` může být zcela libovolný text). Jiný problém by nastal, pokud bychom chtěli specifikovat, že uživatel smí mít maximálně pět e-mailových adres. Tuto skutečnost by šlo v DTD vyjádřit poněkud „neohrabaně“ příslušným zopakováním elementu `email` v rámci specifikace elementu `osoba`. Toto řešení by však již bylo neúnosné pro vyjádření např. maximálního počtu tisíce osob.

[3] Podrobnější výklad jazyka DTD a vysvětlení použitých pojmů uvádí související příspěvek [RK04].

Jak je tedy vidět, pro přesnější vyjádření struktury XML dokumentu jazyk DTD nestačí a je třeba využít nástroj silnější, v našem případě jazyk XML Schema.

2.1 Přínosy jazyka XML Schema

Některé z přínosů a výhod jazyka XML Schema už byly zmíněny v předchozích kapitolách. Stručný přehled těch nejdůležitějších a nejzajímavějších z nich by mohl vypadat takto:

1. Narozdíl od jazyka DTD nevyžaduje speciální syntaxi – XML schémata v jazyce XML Schema jsou opět XML dokumenty s pevně danou strukturou. Uživatel tedy není nucen umět dva různé jazyky a pro zpracování XML schémat je možné využít všech nástrojů, které již existují pro práci s XML dokumenty.
2. Má silnou podporu datových typů. Obsahuje rozsáhlou sadu vestavěných typů (např. `string`, `boolean`, `date`, ...) a současně umožňuje specifikovat vlastní uživatelsky definované typy. Užitečným prvkem jsou bezesporu také vícehodnotové datové typy, v podstatě jakási obdoba pole hodnot.
3. Při modelování je možné opakovaně využívat již definované prvky, čímž se specifikace schématu značně zjednodušuje a urychluje.
4. Umožňuje přesně (tj. v rámci daného rozmezí) vyjádřit přípustné počty výskytů elementů v rámci nadelementu.
5. Striktně rozlišuje mezi posloupností a množinou elementů, tj. umožňuje snadno vyjádřit libovolné pořadí podelementů v rámci nadelementu. Tuto vlastnost je v DTD možné vyjádřit pouze explicitním vyjmenováním všech možných pořadí podelementů, pro složitější elementy tedy poměrně komplikovaně.
6. Při modelování reality je využíváno mnoho objektově-orientovaných prvků, např. dědičnost, substituovatelnost apod., které jsou pro tyto účely poměrně přirozené.
7. Umožňuje definovat elementy se stejným názvem, ale různým obsahem.
8. Umožňuje definovat nejen elementy s prázdným obsahem, ale i elementy, které se mohou vyskytovat bez obsahu.
9. Umožňuje specifikovat unikátnost obsahu elementu, hodnoty atributu nebo jejich kombinace v rámci požadované části XML dokumentu. V jazyce DTD je možné specifikovat pouze unikátnost hodnot atributů v rámci celého XML dokumentu.
10. Zachovává většinu prvků jazyka DTD.

Další vlastností, která je výhodná zejména pro uživatele, je možnost definovat tutéž věc několika způsoby. Na jedné straně tak sice dává autorovi XML schématu do rukou další silný nástroj, na druhé straně je ovšem zpracování XML schématu o to komplikovanější.

3 Základy jazyka XML Schema

Jak už bylo řečeno, každé XML schéma[4] je současně XML dokumentem, který obsahuje pouze speciálně definované elementy. Na začátku každého XML schématu se tedy nachází obvyklá XML deklarace a celé XML schéma je uzavřeno do kořenového elementu `schema`:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema ...>
  ... <!-- definice XML schématu --> ...
</schema>
```

Jednotlivé prvky jazyka XML Schema jsou v podstatě elementy, prostřednictvím jejichž atributů a podelementů požadované XML schéma, tj. přípustnou strukturu XML dokumentů, specifikujeme. Tyto elementy jsou definovány ve *jmenném prostoru* jazyka XML Schema. Přestože je princip jmenných prostorů pro jazyk XML Schema klíčový, není třeba znát jej detailně. Pro porozumění dalšímu textu stačí vědět, že jmenný prostor je prostor jmen, v němž jsou jména elementů a v kontextu elementu i jména atributů unikátní a který je jednoznačně identifikován svým URI[5].

Jmenný prostor (nebo prostory) pro element a jeho podelementy určíme prostřednictvím atributu `xmlns:`<<*prefix*>>, jehož hodnotou je URI daného jmenného prostoru. Při použití prvku z daného jmenného prostoru pak přidáme před jeho název námi definovaný prefix, tj. <<*prefix*>>`:`<<*název prvku*>>. Pro zjednodušení je navíc možné pro jeden z jmenných prostorů (tzv. *implicitní jmenný prostor*) při specifikaci i v následném použití jeho prvků prefix vynechat.

Správně by měl tedy předchozí příklad vypadat takto:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.mff.cuni.cz/MojeSchema"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.mff.cuni.cz/MojeSchema">
  ... <!-- definice XML schématu --> ...
</xs:schema>
```

Atribut `xmlns:xs` říká, že element `schema` i jeho podelementy s prefixem `xs` patří do jmenného prostoru jazyka XML Schema (`http://www.w3.org/2001/XMLSchema`), tj. že se jedná o specifikaci XML schématu. Atributem `targetNamespace`, který je již jedním z možných atributů elementu `schema`, naopak určujeme URI jmenného prostoru, který vytvářeným XML schématem definujeme. Ten je současně (atributem `xmlns`) specifikován jako implicitní jmenný prostor, abychom se mohli na jeho prvky odkazovat bez prefixu.

Podobného principu je využíváno i při připojení XML schématu ke XML dokumentu:

```
<?xml version="1.0" encoding="UTF-8"?>
<KorenovyElement
    xmlns="http://www.mff.cuni.cz/MojeSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.mff.cuni.cz/MojeSchema
                        schema1.xsd">
  ...
</KorenovyElement>
```

[4] Pokud nebude řečeno jinak, je v této kapitole vždy myšleno XML schéma vyjádřené v jazyce XML Schema.

[5] Uniform Resource Identifier – obecný jednoznačný identifikátor informačního zdroje

Atribut `xmlns:xsi` určuje jmenný prostor instancí XML schémat (tj. XML dokumentů) vyhovujících určitému XML schématu (`http://www.w3.org/2001/XMLSchema-instance`). O které schéma se konkrétně jedná určuje atribut `schemaLocation`, který pochází právě z tohoto jmenného prostoru a obsahuje URI jmenného prostoru XML schématu a URL[6] souboru, v němž je schéma uloženo. Atribut `xmlns` navíc určuje, že implicitním jmenným prostorem bude opět jmenný prostor XML schématu našeho dokumentu, a tedy zajišťuje, že názvy elementů můžeme psát bez prefixu.

Pokud pro dané XML schéma nebyl specifikován jmenný prostor (tj. nebyl specifikován atribut `targetNamespace` elementu `schema`), určujeme XML schéma XML dokumentu prostřednictvím atributu `noNamespaceSchemaLocation`, který obsahuje pouze URL souboru, v němž je schéma uloženo. Názvy elementů v tomto případě píšeme bez prefixu implicitně:

```
<?xml version="1.0"?>
<KorenovyElement
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="schema2.xsd">
  ...
</KorenovyElement>
```

Zbývá objasnit odkud se v předchozím příkladu vzal element `KorenovyElement`. Kořenovým elementem XML dokumentu odpovídajícího určitému XML schématu může být libovolný *globálně definovaný element*. Globálně definované prvky XML schématu jsou všechny prvky, které jsou přímými podelementy elementu `schema`. Globálně definované prvky mají v XML schématu speciální postavení, které bude objasněno později. Pro globálně definované elementy je to (mimo jiné) právě vlastnost „být kořenovým elementem XML dokumentu“.

Samotná specifikace XML schématu spočívá v definici datových typů a jejich následném přiřazení elementům nebo atributům. Datové typy se dělí na *jednoduché*, které mohou být přiřazeny elementům i atributům a *složené*, které mohou přiřazeny pouze elementům. Základními prvky jazyka XML Schema, kterým jsou věnovány následující podkapitoly, jsou tedy:

- jednoduchý datový typ,
- složený datový typ,
- element,
- atribut a
- skupina atributů.

Poznamenejme ještě, že v následujících příkladech bude pro zkrácení zápisu vynechávána XML deklarace i kořenový element `schema`. Ve všech příkladech se implicitně předpokládá jejich definice tak, jak byla uvedena v předchozím textu.

3.1 Jednoduché datové typy

Nejjednoduššími prvky jazyka XML Schema jsou jednoduché datové typy. Obsahem elementů nebo hodnotami atributů v XML dokumentu je sice vždy text, nicméně prostřednictvím jednoduchých datových typů lze specifikovat omezení, která na textové hodnoty klademe. Jednoduchým datovým typem je tedy možné popsat omezení řetězce na množinu přípustných hodnot.

V jazyce XML Schema rozlišujeme dva druhy jednoduchých typů – *vestavěné*, tj. předdefinované a *uživatelsky definované*, tj. typy, které si může uživatel sám nadefinovat *odvozením* z jiných (vestavěných i uživatelsky definovaných) typů.

[6] Uniform Resource Locator – jednoznačný identifikátor umístění souboru

3.1.1 Vestavěné typy

Množinu vestavěných datových typů je možné dále dělit na *základní* (viz. tabulka 1.) a *odvozené* (viz. tabulka 2. a 3.).

Tabulka 1. Základní vestavěné datové typy	
Název:	**Význam:**
`string`	Řetězec znaků
`boolean`	Logické hodnoty `true` a `false`, popř. `1` a `0`
`decimal`	Kladné nebo záporné reálné číslo (např. `-1.23`, `1267.5433`, `210`)
`float`	32-bitové kladné nebo záporné reálné číslo vyjádřené pomocí mantisy a exponentu (např. `-1E4`, `1267.43233E12`, `12`, popř. speciální hodnoty `0`, `-0`, `INF`, `-INF` nebo `NaN`)
`double`	64-bitové číslo se stejnými vlastnostmi jako `float`
`duration`	Časový úsek ve tvaru `PnYnMnDTnHnMnS`, kde `P` a `T` jsou oddělovače, `nY` znamená `n` let, `nM` znamená `n` měsíců atd. (např. `-P13Y7M`, `P2Y1MT2H`)
`dateTime`	Datum a čas ve tvaru `YYYY-MM-DDThh:mm:ss.ss`, kde `T` je oddělovač
`time`	Čas ve tvaru `hh:mm:ss.ss`
`date`	Datum ve tvaru `YYYY-MM-DD`
`gYearMonth`	Měsíc v roce ve tvaru `YYYY-MM`
`gYear`	Rok ve tvaru `YYYY`
`gMonthDay`	Den v měsíci ve tvaru `MM-DD`
`gMonth`	Měsíc ve tvaru `MM`
`gDay`	Den ve tvaru `DD`
`hexBinary`	Hexadecimální číslo
`base64Binary`	Binární data s kódováním *Base64*
`anyURI`	Absolutní nebo relativní URI
`QName`	*XML Qualified Name*, tj. řetězec ve tvaru <<*prefix*>>:<<*místní část*>>, kde <<*prefix*>> je označení jmenného prostoru a <<*místní část*>> je název prvku patřícího do tohoto jmenného prostoru
`NOTATION`	Odkaz na notaci

Tabulka 2. Vestavěné datové typy odvozené od typu `string`	
Název:	**Význam:**
`normalizedString`	`string`, který neobsahuje znaky CR, LF a tabulátor
`token`	`normalizedString`, který nemá mezery na začátku ani na konci a neobsahuje posloupnost mezer delší než jedna
`language`	Identifikátor jazyka (např. `en`, `en-GB`)
`Name`	*XML Name*, tj. řetězec, který obsahuje písmena, číslice nebo znaky '`-`', '`_`', '`:`' a '`.`'
`NCName`	XML Name neobsahující znak '`:`'
`NMTOKEN`	Jednoslovná hodnota z písmen, číslic a příp. dalších znaků
`NMTOKENS`	Seznam jednoslovných hodnot
`ID`	Hodnota jednoznačná v rámci celého XML dokumentu
`IDREF`	Odkaz na hodnotu typu `ID`
`IDREFS`	Seznam odkazů na hodnoty typu `ID`
`ENTITY`	Odkaz na entitu
`ENTITIES`	Seznam odkazů na entity

Tabulka 3. Vestavěné datové typy odvozené od typu `decimal`	
Název:	**Význam:**
`integer`	Celé číslo
`positiveInteger`	Kladné celé číslo
`negativeInteger`	Záporné celé číslo
`nonPositiveInteger`	Nekladné celé číslo
`nonNegativeInteger`	Nezáporné celé číslo
`long`	Celé číslo z intervalu $< -2^{63}, 2^{63}-1 >$
`int`	Celé číslo z intervalu $< -2^{31}, 2^{31}-1 >$
`short`	Celé číslo z intervalu $< -2^{15}, 2^{15}-1 >$
`byte`	Celé číslo z intervalu $< -2^{7}, 2^{7}-1 >$
`unsignedLong`	Nezáporné číslo menší než 2^{64}
`unsignedInt`	Nezáporné číslo menší než 2^{32}
`unsignedShort`	Nezáporné číslo menší než 2^{16}
`unsignedByte`	Nezáporné číslo menší než 2^{8}

Datové typy, jejichž názvy jsou velkými písmeny (např. `ID`, `NMTOKENS` atd.) pochází z jazyka DTD, tj. mají stejný význam (viz. [RK04]) a mohou být přiřazeny pouze atributům. Ostatní datové typy mohou být přiřazeny elementům i atributům.

3.1.2 Uživatelsky definované typy

Zcela novým nástrojem pro specifikaci XML schématu je možnost definovat vlastní jednoduchý datový typ. Definici provádíme odvozením nového typu z jiného již existujícího (vestavěného nebo uživatelsky definovaného) typu jedním ze tří možných způsobů, a to:

- restrikcí,
- seznamem nebo
- sjednocením.

Uživatelsky definovaný jednoduchý typ definujeme pomocí elementu `simpleType`, zvolený typ odvození jeho podelementem `restriction`, `list` nebo `union`. Pokud je jednoduchý datový typ definován jako globální, musí být navíc prostřednictvím atributu `name` pojmenován. Dále je prostřednictvím atributu `final` možné pro daný datový typ zakázat další odvozování restrikcí (`restriction`), seznamem (`list`), sjednocením (`union`) nebo libovolným způsobem (`#all`).

Odvození restrikcí znamená omezení hodnot původního datového typu dle daného pravidla:

```
<xs:simpleType name="Porty">
  <xs:restriction base="xs:integer">
    <xs:enumeration value="111"/>
    <xs:enumeration value="21"/>
    <xs:enumeration value="80"/>
  </xs:restriction>
</xs:simpleType>
```

```
<xs:simpleType name="NeprazdnyRetezec" final="#all">
  <xs:restriction base="xs:string">
    <xs:minLength value="1"/>
    <xs:maxLength value="100"/>
  </xs:restriction>
</xs:simpleType>
```

V prvním příkladě je definován jednoduchý datový typ s názvem `Porty`, který je odvozen restrikcí z typu `integer` omezením jeho přípustných hodnot na `111`, `21` a `80`. Druhý příklad definuje jednoduchý datový typ s názvem `NeprazdnyRetezec`, který je odvozen restrikcí z datového typu `string`, omezením minimální délky na `1` a maximální délky na `100`, z něhož již navíc není možné odvozovat žádné další typy.

Všechna možná pravidla pro omezení restrikcí jsou uvedena v tabulce 4. Princip jejich použití je stejný jako v předchozích příkladech.

Tabulka 4. Pravidla pro odvozování restrikcí

Název:	**Význam:**
`length`	Počet jednotek daného typu (např. znaků v řetězci)
`minLength`	Minimální počet jednotek daného typu
`maxLength`	Maximální počet jednotek daného typu
`pattern`	Regulární výraz, kterému musí hodnoty vyhovovat
`enumeration`	Explicitně vyjmenovaná množina povolených hodnot daného typu
`whiteSpace`	Způsob zpracování bílých znaků v řetězci – `preserve` (žádné změny), `replace` (znaky CR, LF a tabulátor jsou nahrazeny mezerou), `collapse` (navíc jsou odstraněny mezery na začátku a na konci řetězce a posloupnosti mezer nahrazeny jednou mezerou)
`maxInclusive`	Hodnoty datového typu musí být <= zadané hodnotě
`minInclusive`	Hodnoty datového typu musí být >= zadané hodnotě
`maxExclusive`	Hodnoty datového typu musí být < zadaná hodnota
`minExclusive`	Hodnoty datového typu musí být > zadaná hodnota
`totalDigits`	Maximální počet cifer
`fractionDigits`	Maximální počet cifer za desetinnou čárkou

Odvozením seznamem je naopak možné vytvořit datový typ odpovídající seznamu hodnot původního typu oddělených mezerou:

```
<xs:simpleType name="SeznamRealnychCisel">
  <xs:list itemType="xs:float"/>
</xs:simpleType>
```

Odvození seznamem není přirozeně možné provádět z datových typů, které již byly odvozeny seznamem. Všimněme si dále, že mezi vestavěnými datovými typy již existují tři typy odvozené seznamem – `NMTOKENS`, `IDREFS` a `ENTITIES`.

Konečně odvozením sjednocením vzniká datový typ odpovídající sjednocení přípustných hodnot všech sjednocovaných typů:

```
<xs:simpleType name="NenulovaCelaCisla">
  <xs:union
      memberTypes="xs:positiveInteger xs:negativeInteger"/>
</xs:simpleType>
```

Jednoduchý datový typ může být v XML schématu definován globálně nebo lokálně. V prvním případě musí být přímým podelementem elementu `schema` a musí mít název, jehož prostřednictvím se na něj lze (opakovaně) odkazovat. Ve druhém případě je umístěn v rámci definice elementu, atributu, složeného typu nebo jiného jednoduchého typu, pojmenován není a tudíž je určen pouze pro dané konkrétní použití:

```
<xs:simpleType name="SjednoceníTypu">
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="xs:positiveInteger">
        <xs:minInclusive value="8"/>
        <xs:maxInclusive value="72"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="male"/>
        <xs:enumeration value="velke"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
```

3.2 Atributy

Atributy definujeme prostřednictvím elementu `attribute`. Každý atribut musí mít název, který specifikujeme atributem `name` a datový typ. Datový typ atributu může být pouze jednoduchý (vestavěný nebo uživatelsky definovaný) a může být určen buď atributem `type` nebo podelementem `simpleType` elementu `attribute`. V prvním případě lze atributu přiřazovat globálně definované (tj. pojmenované) nebo vestavěné typy, ve druhém případě přiřazujeme lokálně definovaný typ určený pouze pro daný atribut.

```
<xs:attribute name="Vek" type="xs:positiveInteger"/>
```

```
<xs:attribute name="Nazev" type="NeprazdnyRetezec"/>
```

```
<xs:attribute name="TelefonniCislo">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="10"/>
      <xs:pattern value="\d{3}-\d{6}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
```

Prostřednictvím dalších atributů elementu `attribute` je možné specifikovat:

- implicitní hodnotu atributu, není-li v XML dokumentu uveden (hodnotou atributu `default`),
- konstantní hodnotu atributu (hodnotou atributu `fixed`) nebo
- (ne)povinnost výskytu atributu v XML dokumentu (atributem `use` a jeho hodnotami `optional`, `required` a `prohibited`).

Atribut může být v XML schématu také definován globálně nebo lokálně, ovšem v obou případech musí být pojmenován. V prvním případě musí být přímým podelementem elementu `schema` a lze se na něj (opakovaně) odkazovat, ve druhém případě je umístěn v rámci definice složeného typu nebo skupiny atributů a je určen pouze pro dané konkrétní použití. Obojí bude demonstrováno v následujících kapitolách.

3.3 Elementy

Elementy definujeme prostřednictvím elementu `element`. Podobně jako v případě atributů, musí mít každý element název, který specifikujeme atributem `name` a datový typ. Datový typ elementu může být jednoduchý nebo složený a může být určen buď atributem `type` nebo podelementem `simpleType` nebo `complexType`. V prvním případě lze elementu přiřazovat globálně definované nebo vestavěné typy, ve druhém případě přiřazujeme lokálně definovaný typ určený pouze pro daný element.

Pokud elementu přiřadíme jednoduchý datový typ znamená to, že obsahem elementu mohou být pouze hodnoty daného typu. Takový element navíc nemá atributy:

```
<xs:element name="Jmeno" type="xs:string"/>
```

```
<xs:element name="Prijmeni">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Složitější elementy, tj. elementy, které mají atributy, popř. elementový nebo smíšený obsah je možné vytvořit pouze pomocí složených datových typů (viz. dále).

Prostřednictvím dalších atributů elementu `element` je možné specifikovat:

- že se element smí v dokumentu vyskytovat bez svého obsahu (atributem `nillable` a jeho hodnotami `true` nebo `false`),
- implicitní hodnotu elementu s jednoduchým typem, není-li v XML dokumentu uveden (hodnotou atributu `default`) nebo
- konstantní hodnotu elementu s jednoduchým typem (hodnotou atributu `fixed`).

Pro globální a lokální definici elementů platí podobná pravidla jako pro atributy – element může být definován globálně, jakožto přímý podelement elementu `schema`, nebo lokálně v rámci definice složeného typu. Navíc právě globálně definované elementy mohou být kořenovými elementy instancí XML dokumentů daného XML schématu.

3.4 Složené datové typy

Jak již bylo naznačeno, složené datové typy slouží pro definici složitějších typů elementů. Pomocí složených datových typů je možné určit vztahy element-podelement a element-atribut a vyjádřit počty a pořadí výskytu elementů v rámci nadelementu. Každý složený typ se tedy skládá ze specifikace obsahu složeného typu a z množiny atributů. Pokud je množina atributů prázdná, jedná se o definici složeného typu elementu bez atributů, pokud je prázdná specifikace obsahu, jedná se o definici složeného typu prázdného elementu.

```
<xs:complexType name="Adresa">
  <xs:sequence>
    <xs:element name="Ulice" type="xs:string"/>
    <xs:element name="CDomu" type="xs:integer"/>
    <xs:element name="Mesto" type="xs:string"/>
  </xs:sequence>
  <xs:attribute name="Zeme"  type="xs:NMTOKEN" default="CZ"/>
</xs:complexType>
```

Složené typy definujeme prostřednictvím elementu `complexType`. Název složeného typu určíme atributem `name`, množinu atributů vyjádříme prostřednictvím podelementů `attribute`. Pro specifikaci obsahu složeného typu existuje šest podelementů odpovídajících šesti druhům obsahů složených typů:

- složený typ s jednoduchým obsahem (`simpleContent`),
- posloupnost elementů (`sequence`),
- výběr z elementů (`choice`),
- množina elementů (`all`),
- modelová skupina (`group`) a
- složený typ se složeným obsahem (`complexContent`).

Stejně jako v případě jednoduchých datových typů je možné složené typy definovat globálně i lokálně. Princip jejich použití je stejný:

```
<xs:element name="MojeAdresa" type="Adresa"/>
```

```
<xs:element name="Osoba">
  <xs:complexType>
    <xs:all>
      <xs:element name="Jmeno"      type="xs:string"/>
      <xs:element name="Vek"        type="xs:positiveInteger"/>
      <xs:element name="AdresaDomu" type="Adresa"/>
    </xs:all>
    <xs:attribute name="Id" type="xs:ID"/>
  </xs:complexType>
</xs:element>
```

Významným atributem elementu `complexType` je atribut `mixed` (s hodnotami `true` a `false`) jakožto příznak *smíšeného obsahu*. Smíšený obsah elementu znamená, že jeho obsahem může být kromě specifikovaných podelementů také libovolný další text.

3.4.1 Složený typ s jednoduchým obsahem

Složený typ s jednoduchým obsahem slouží pro vytváření elementů, jejichž obsahem je pouze jednoduchý datový typ (tj. nemají žádné podelementy, pouze textový obsah) ale mají alespoň jeden atribut. Element `simpleContent` obsahuje buď rozšíření (`extension`) jednoduchého datového typu (určeného atributem `base`) o atributy nebo restrikci (`restriction`) jiného složeného typu s jednoduchým obsahem, tj. restrikci jeho obsahu, tak jak byla popsána v předchozím textu.

```
<xs:complexType name="Typ">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="Podtyp" type="xs:string"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

```
<xs:complexType name="TypAuta">
  <xs:simpleContent>
    <xs:restriction base="Typ">
      <xs:enumeration value="Audi"/>
      <xs:enumeration value="Golf"/>
      <xs:enumeration value="BMW"/>
      <xs:attribute name="Podtyp" type="xs:string"/>
    </xs:restriction>
  </xs:simpleContent>
</xs:complexType>
```

3.4.2 Posloupnost elementů

Složený datový typ s obsahem typu posloupnost elementů (`sequence`) slouží pro vytváření elementů, jejichž obsahem jsou specifikované prvky (elementy, jiné posloupnosti elementů, výběry z elementů, množiny elementů nebo modelové skupiny) v daném pořadí.

Pro každý prvek posloupnosti i pro posloupnost samotnou je možné prostřednictvím atributů `minOccurs` a `maxOccurs` specifikovat minimální a maximální počet výskytů daného prvku. Implicitní hodnoty těchto atributů jsou rovny `1`, tj. pokud nejsou uvedeny, musí se daný prvek vyskytovat právě jednou. Nepovinný výskyt prvku vyjádříme nastavením atributu `minOccurs` na `0`, pro neomezený maximální počet výskytů je definována konstanta `unbounded`.

```
<xs:complexType name="Osoba">
  <xs:sequence>
    <xs:element name="Jmeno"    type="xs:string" maxOccurs="5"/>
    <xs:element name="Prijmeni" type="xs:string"/>
    <xs:element name="DatumNar" type="xs:date"/>
    <xs:element name="Poznamka" type="xs:string" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="Id" type="xs:ID"/>
</xs:complexType>
```

3.4.3 Výběr z elementů

Složený datový typ s obsahem typu výběr z elementů (`choice`) slouží pro vytváření elementů, jejichž obsahem je jeden ze specifikovaných prvků výběru. Prvky výběru mohou být tytéž jako v případě posloupnosti, pro jejich počty výskytů platí tatáž pravidla.

```
<xs:complexType name="DopravniProstredek">
  <xs:choice>
    <xs:element name="Auto"    type="xs:string"/>
    <xs:element name="Vlak"    type="xs:string"/>
    <xs:element name="Letadlo" type="xs:string"/>
  </xs:choice>
</xs:complexType>
```

3.4.4 Množina elementů

Složený datový typ s obsahem typu množina elementů (`all`) slouží pro vytváření elementů, jejichž obsahem jsou specifikované prvky v libovolném pořadí. Prvky množiny mohou být ovšem pouze elementy s maximálním počtem výskytů rovným `1`.

```
<xs:complexType name="Kniha">
  <xs:all>
    <xs:element name="Nazev"  type="xs:string"/>
    <xs:element name="Autor"  type="xs:string"/>
    <xs:element name="Vydani" type="xs:date"/>
    <xs:element name="ISBN"   type="xs:string"/>
  </xs:all>
</xs:complexType>
```

3.4.5 Modelová skupina

Modelová skupina (`group`) může jako podelement obsahovat posloupnost, množinu nebo výběr z elementů a musí být vždy deklarována globálně. Slouží k tomu, aby mohly být tyto tři typy obsahu definovány globálně (tj. s pojmenováním) a tudíž využívány opakovaně.

Opakované využití globálně definované modelové skupiny je zajištěno prostřednictvím tzv. *referencí*. Referenci na (jakýkoli) globálně definovaný prvek deklarujeme stejným elementem jakým je deklarován daný globální prvek (v případě modelové skupiny tedy elementem `group`) a jeho atributem `ref`. Reference v podstatně znamená „vložení“ odkazovaného prvku na dané místo:

```
<xs:group name="ElementyProPublikaci">
  <xs:sequence>
    <xs:element name="Nazev" type="xs:string"/>
    <xs:element name="Autor" type="xs:string"
                             maxOccurs="unbounded"/>
    <xs:element name="Datum" type="xs:date"/>
  </xs:sequence>
</xs:group>

<xs:complexType name="Kniha">
  <xs:sequence>
    <xs:group   ref="ElementyProPublikaci"/>
    <xs:element name="ISBN"      type="xs:string"/>
    <xs:element name="Vydavatel" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="CD">
  <xs:sequence>
    <xs:group   ref="ElementyProPublikaci"/>
    <xs:element name="NahravaciStudio" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Stejného principu je možné využívat také pro globálně definované elementy:

```
<xs:element name="Nazev">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:complexType name="Kniha">
  <xs:sequence>
    <xs:element ref="Nazev"/>
    <xs:element name="Vydavatel" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="CD">
  <xs:sequence>
    <xs:element ref="Nazev"/>
    <xs:element name="NahravaciStudio" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

3.4.6 Složený typ se složeným obsahem

Složený typ se složeným obsahem slouží pro odvozování nových složených typů z již existujících. Pro jeho definici slouží element `complexContent` obsahující buď restrikci (`restriction`) nebo rozšíření (`extension`) již existujícího typu, který určíme prostřednictvím atributu `base`.

Restrikce znamená vytvoření nového složeného typu, který je podmnožinou typu původního (např. omezením počtu výskytů elementů, omezením přípustných hodnot použitých jednoduchých typů apod.). Jelikož je restrikcí možné také vypouštět elementy nebo atributy, je třeba při definici nového typu zopakovat definici všech prvků, které v něm chceme zachovat:

```
<xs:complexType name="Publikace">
  <xs:sequence>
    <xs:element name="Název" type="xs:string"/>
    <xs:element name="Autor" type="xs:string"
                             maxOccurs="unbounded"/>
    <xs:element name="Datum" type="xs:gYear"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="PublikaceSJednimAutorem">
  <xs:complexContent>
    <xs:restriction base="Publikace">
      <xs:sequence>
        <xs:element name="Nazev" type="xs:string"/>
        <xs:element name="Autor" type="xs:string"/>
        <xs:element name="Datum" type="xs:gYear"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

Naproti tomu rozšířením vznikne složený typ, který obsahuje původní i nový typ (tj. jeho prvky) v tomto pořadí. Při definici nového typu tedy specifikujeme pouze ty prvky, o které má být původní typ rozšířen. Je to tedy jakási obdoba dědičnosti známé z objektově-orientovaných programovacích jazyků.

```
<xs:complexType name="Osoba">
  <xs:sequence>
    <xs:element name="Jmeno"    type="xs:string"/>
    <xs:element name="Prijmeni" type="xs:string"/>
    <xs:element name="Poznamka" type="xs:string" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="Id" type="xs:ID"/>
</xs:complexType>

<xs:complexType name="Student">
  <xs:complexContent>
    <xs:extension base="Osoba">
      <xs:sequence>
        <xs:element name="Obor"   type="xs:string"/>
        <xs:element name="Rocnik" type="xs:positiveInteger"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Se složenými datovými typy se složeným obsahem souvisí další dva atributy elementu `complexType`:

- atribut `abstract` (s hodnotami `true` nebo `false`) umožňující vytvořit tzv. *abstraktní datový typ*, který nelze přiřadit žádnému elementu a
- atribut `final`, který pro daný typ zakazuje další odvozování restrikcí (`restriction`), rozšířením (`extension`) nebo oběma způsoby (`#all`).

3.5 Skupiny atributů

Skupina atributů má podobný význam pro atributy jako má modelová skupina pro elementy, tj. umožňuje opakované využívání celé množiny atributů. Skupinu atributů definujeme prostřednictvím elementu `attributeGroup` obsahujícím množinu deklarací atributů. Skupina musí být vždy definována globálně a pojmenována prostřednictvím atributu `name`. Její opakované využití je stejně jako v případě modelové skupiny zajištěno prostřednictvím referencí:

```
<xs:attributeGroup name="SpolecneAtributy">
  <xs:attribute name="Vypujcen" type="xs:boolean"/>
  <xs:attribute name="Id"       type="xs:ID"/>
</xs:attributeGroup>

<xs:complexType name="Kniha">
  <xs:sequence>
    <xs:element name="Nazev"     type="xs:string"/>
    <xs:element name="Vydavatel" type="xs:string"/>
  </xs:sequence>
  <xs:attributeGroup ref="SpolecneAtributy"/>
</xs:complexType>

<xs:complexType name="CD">
  <xs:sequence>
    <xs:element name="Nazev"           type="xs:string"/>
    <xs:element name="NahravaciStudio" type="xs:string"/>
  </xs:sequence>
  <xs:attributeGroup ref="SpolecneAtributy"/>
</xs:complexType>
```

Stejný princip je možné použít také pro globálně definované atributy:

```
<xs:attribute name="Vypujcen">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="ano"/>
      <xs:enumeration value="ne"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>

<xs:complexType name="Kniha">
  <xs:sequence>
    <xs:element name="Nazev"     type="xs:string"/>
    <xs:element name="Vydavatel" type="xs:string"/>
  </xs:sequence>
  <xs:attribute ref="Vypujcen"/>
</xs:complexType>

<xs:complexType name="CD">
  <xs:sequence>
    <xs:element name="Nazev"           type="xs:string"/>
    <xs:element name="NahravaciStudio" type="xs:string"/>
  </xs:sequence>
  <xs:attribute ref="Vypujcen"/>
</xs:complexType>
```

3.6 Shrnutí

Pro úplnost ještě uveďme příklad XML schématu v jazyce XML Schema, kterému by vyhovoval XML dokument z kapitoly 2:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.mff.cuni.cz/MojeSchema">
  <xs:element name="zamestnanci">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="osoba" type="TypOsoba"
                                 maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="TypOsoba">
    <xs:sequence>
      <xs:element name="jmeno" type="TypJmeno"/>
      <xs:element name="email" type="xs:string"
                               minOccurs="0"
                               maxOccurs="unbounded"/>
      <xs:element name="vztahy" minOccurs="0">
        <xs:complexType>
          <xs:attribute name="nadrizeny" type="xs:IDREF"/>
          <xs:attribute name="podrizeni" type="xs:IDREFS"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID"
                            use="required"/>
    <xs:attribute name="poznamka" type="xs:string"/>
    <xs:attribute name="dovolena" type="TypAnoNe"
                                  default="ne"/>
  </xs:complexType>

  <xs:complexType name="TypJmeno">
    <xs:all>
      <xs:element name="krestni"  type="xs:string"/>
      <xs:element name="prijmeni" type="xs:string"/>
    </xs:all>
  </xs:complexType>

  <xs:simpleType name="TypAnoNe">
    <xs:restriction base="xs:string">
      <xs:enumeration value="ano"/>
      <xs:enumeration value="ne"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Toto XML schéma je pouze jednou z možných verzí daného XML schématu. V jiné by mohly být např. některé z globálně definovaných typů definovány lokálně, naopak některé z lokálně definovaných elementů (respektive atributů) definovány globálně, popř. v rámci globálně definované modelové skupiny (respektive skupiny atributů) atd. Vzhledem k tomu, že jediným globálně definovaným elementem je element `zamestnanci`, je to také jediný element, který se může stát kořenovým elementem instance XML dokumentu.

4 Pokročilé prvky jazyka XML Schema

V předchozí kapitole byly vysvětleny základní principy specifikace XML schémat – definice jednoduchých a složených datových typů, elementů, atributů, skupin atributů a princip využívání globálně a lokálně definovaných prvků. Jazyk XML Schema obsahuje ještě několik dalších prvků, které dále rozšiřují jeho možnosti a které by bylo možné označit jako pokročilé. Jejich stručný přehled a nástin jejich funkcí je uveden následujících podkapitolách.

4.1 Omezení identity

Z jazyka DTD jsou známy tři datové typy (vyhrazené pro atributy) umožňující vyjádřit obdobu databázových klíčů a cizích klíčů – `ID`, `IDREF` a `IDREFS`, které byly zachovány i v jazyce XML Schema. Ten byl navíc rozšířen o další tři prvky pro omezení identity – tzv. omezení na klíč (`key`), omezení na unikátnost (`unique`) a omezení na cizí klíč (`keyref`), které umožňují:

- rozlišovat mezi pojmy unikátní a klíčový,
- definovat také obsah elementu jako unikátní nebo klíčový,
- definovat unikátní neklíčové atributy,
- definovat kombinaci elementu a atributu jako unikátní nebo klíčovou a
- definovat pouze určitou část XML dokumentu, v rámci níž je něco unikátní nebo klíčem.

Klíčovou hodnotou je hodnota, která je v XML dokumentu vždy obsažena, je nenulová a v rámci dané oblasti jednoznačná. Pro *unikátní* hodnoty nevyžadujeme podmínku na nutnou přítomnost v XML dokumentu, tj. hodnota musí být nenulová a jednoznačná pouze tehdy, je-li v dokumentu obsažena. Pro určování prvků, jejichž identitu omezujeme a oblastí, v rámci nichž omezení platí, je využíván jazyk XPath [CJ99].

Omezení na klíč umožňuje vybrat prvky (elementy, atributy nebo jejich kombinaci), jejichž hodnoty musí být v rámci dané oblasti klíčové. Omezení na klíč specifikujeme prostřednictvím elementu `key` a jeho podelementů `selector` a `field`. Omezení může být umístěno pouze na konci definice elementu, XPath dotazy potom začínají právě v kontextu tohoto elementu. Elementem `selector` (respektive jeho atributem `xpath`), který se musí vyskytovat právě jednou, specifikujeme množinu elementů v rámci níž omezení platí. Alespoň jedním elementem `field` specifikujeme element nebo atribut, který má být v rámci této oblasti klíčem.

```
<xs:element name="Knihovna">
  ...
  <xs:element name="Kniha" maxOccurs="unbounded">
    ...
    <xs:element name="ISBN" type="xs:string"/>
    ...
  </xs:element>

  <xs:key name="PrimarniKlic">
    <xs:selector xpath="./Kniha"/>
    <xs:field    xpath="./ISBN"/>
  </xs:key>
</xs:element>
```

V příkladu je definován element `Knihovna`, který obsahuje elementy `Kniha`, z nichž každý má podelement `ISBN`. Omezení `PrimarniKlic` specifikuje, že v rámci všech elementů `Kniha`, které jsou podelementy aktuálního elementu (tj. elementu `Knihovna`) je obsah elementu `ISBN` klíčem.

Stejným způsobem (pouze místo elementu `key` by byl použit element `unique`) by bylo možné definovat omezení na unikátnost.

Oproti tomu omezení na cizí klíč umožňuje (opět velmi podobným principem) specifikovat tzv. *cizí klíče*, tj. odkazy na klíčové nebo unikátní hodnoty. Pomocí elementů `selector` a `field` je opět zvolena oblast a v rámci ní prvky, které mají obsahovat hodnoty cizích klíčů. Navíc je třeba hodnotou atributu `refer` specifikovat omezení na unikátnost nebo klíč, k němuž se cizí klíč vztahuje.

```
<xs:element name="Knihovna">
  ...
  <xs:element name="Kniha" maxOccurs="unbounded">
    ...
    <xs:element name="ISBN" type="xs:string"/>
    ...
  </xs:element>
  <xs:element name="Autor" maxOccurs="unbounded">
    ...
    <xs:element name="NejlepsiKniha">
      ...
      <xs:element name="ISBN" type="xs:string"/>
      ...
    </xs:element>
    ...
  </xs:element>

  <xs:key name="PrimarniKlic">
    <xs:selector xpath="./Kniha"/>
    <xs:field    xpath="./ISBN"/>
  </xs:key>

  <xs:keyref name="CiziKlic" refer="PrimarniKlic">
    <xs:selector xpath="./Autor/NejlepsiKniha"/>
    <xs:field    xpath="./ISBN"/>
  </xs:keyref>
</xs:element>
```

Omezení `CiziKlic` v příkladu specifikuje, že v rámci elementů `NejlepsiKniha`, které jsou podelementy elementu `Autor` je hodnota elementu `ISBN` cizím klíčem, který se odkazuje na klíčové hodnoty specifikované omezením `PrimarniKlic`.

4.2 Substituční skupiny

Jedním z vůbec nejsilnějších nástrojů v jazyce XML Schema je mechanismus substitučních skupin, který je v podstatě obdobou substituovatelnosti známé z objektově-orientovaných programovacích jazyků. Tento mechanismus jednoduše zajišťuje, aby bylo možné v instancích XML dokumentů substituovat za daný element jiné elementy.

Prostřednictvím atributu `substitutionGroup` elementu `element` specifikujeme pro daný element tzv. *vedoucí element*, do jehož substituční skupiny má patřit. Tím zajistíme, že se na všech místech v XML dokumentu, na kterých se vyskytuje vedoucí element, může vyskytovat i libovolný element z jeho substituční skupiny. Pro elementy v substituční skupině musí navíc platit, že mají stejný datový typ jako vedoucí element, nebo typ z něj odvozený (restrikcí nebo rozšířením).

```
<xs:element name="Publikace" type="TypPublikace"/>
<xs:element name="Kniha"     type="TypKniha"
            substitutionGroup="Publikace"/>
<xs:element name="Casopis"   type="TypCasopis"
            substitutionGroup="Publikace"/>

<xs:element name="Knihovna">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Publikace" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

V příkladu jsou definovány tři elementy – `Kniha`, `Casopis` a `Publikace` s datovými typy `TypKniha`, `TypCasopis` a `TypPublikace` o nichž předpokládáme, že první dva byly nějakým způsobem odvozeny z třetího. Dále jsou elementy `Kniha` a `Casopis` začleněny do substituční skupiny vedoucího elementu `Publikace`. Díky této vlastnosti, může v instancích XML dokumentů element `Knihovna` obsahovat nejen elementy `Publikace`, ale i `Kniha` nebo `Casopis`.

Se substitučními skupinami souvisí další dva atributy elementu `element`:

- atribut `abstract` (s hodnotami `true` nebo `false`) umožňující vytvořit tzv. *abstraktní vedoucí element*, za který musí být v XML dokumentu vždy substituován některý z elementů v jeho substituční skupině a
- atribut `final`, který zajišťuje, že se v substituční skupině vedoucího elementu nesmí vykytovat elementy odvozené restrikcí (`restriction`), rozšířením (`extension`) nebo oběma způsoby (`#all`).

Za zmínku ještě stojí fakt, že relace „být v substituční skupině“ je tranzitivní.

4.3 Zástupci

Mechanismus zástupců zajišťuje, aby bylo možné na určité místo v XML dokumentu vložit v podstatě libovolný element (`any`) nebo atribut (`anyAttribute`). Pro určení nakolik libovolný daný prvek může být je opět využíváno jmenných prostorů, tj. pro každý prvek specifikujeme z jakého jmenného prostoru nebo prostorů musí pocházet. K tomuto účelu je určen atribut `namespace`, kterým specifikujeme buď přímo URI konkrétního jmenného prostoru nebo můžeme vyžít pomocné konstanty `##any` (libovolný známý jmenný prostor), `##targetNamespace` (cílový jmenný prostor aktuálního schématu), `##other` (prostor jiný než cílový jmenný prostor aktuálního schématu) nebo `##local` (prvky bez prefixu jmenného prostoru). Naproti tomu atributem `processContents` specifikujeme způsob validace vkládaných prvků – přísná validace vůči jmennému prostoru (`strict`), validace pouze vůči známým schématům jmenných prostorů (`lax`) nebo žádná validace (`skip`). Pro elementy je navíc možné prostřednictvím atributů `minOccurs` a `maxOccurs` specifikovat minimální a maximální počet jejich výskytů.

```
<xs:complexType name="LibovolnyHtmlText">
  <xs:sequence>
    <xs:any namespace="http://www.w3.org/1999/xhtml"
            minOccurs="1" maxOccurs="unbounded"
            processContents="lax"/>
  </xs:sequence>
</xs:complexType>
```

4.4 Další prvky

Zbývající dva prvky jazyka XML Schema už budou pouze ve zkratce zmíněny. Jedná se o prvky, které již nijak zásadně nemění strukturu XML schématu, ale slouží pouze pro jeho uživatelsky příjemnější specifikaci.

Prvním z nich je možnost využívat při specifikaci aktuálního XML schématu již existujících tzv. *externích* schémat, popř. jejich prvky předefinovat. Pro odkazování na externí schémata slouží elementy `include` a `import`, pro předefinování prvků z externích schémat element `redefine`.

Naproti tomu druhý prvek umožňuje vytvářené XML schéma dokumentovat. V podstatě na libovolné (ovšem vzhledem k vlastnostem XML dokumentů) korektní místo v XML schématu je možné vložit element `annotation`, který obsahuje informace o aktuální části XML schématu určené buď pro čtenáře (`documentation`) nebo pro aplikaci (`appinfo`) zpracovávající dané schéma, popř. může obsahovat pouze odkaz na dokument, v němž jsou tyto informace uloženy.

5 Závěr

Cílem tohoto článku bylo nabídnout stručný přehled toho, co jazyk XML Schema v současné době nabízí a umožňuje, a to především prostřednictvím sady ukázkových příkladů použití jeho jednotlivých prvků. Vzhledem k tomu, jak obsáhlá je specifikace tohoto jazyka, nebylo samozřejmě možné postihnout všechny detaily ani vysvětlovat zmíněné prvky příliš podrobně. K tomuto účelu je již třeba nahlédnout přímo do jeho specifikací [FDC01] [THS01] [BPV01].

Přes zjevnou hlavní nevýhodu jazyka XML Schema, kterou je délka zápisu jednotlivých konstruktů, má tento jazyk především velké množství výhod. Jak už bylo řečeno jsou to zejména jeho objektově-orientované rysy, které je možné využít jak při modelování reality, jakožto přirozený nástroj pro tento účel, tak při zpracování XML schémat v objektově-orientovaných programovacích jazycích nebo prostřednictvím objektových nebo objektově-relačních databázích. Nicméně přestože je jazyk XML Schema v současné době využíván stále častěji, na plné využití všech svých rysů stále čeká.

Reference

[BPV01] Biron, P. V. – Malhotra, A.: *XML Schema Part 2: Datatypes*. W3C Recommendation, 2001. `www.w3.org/TR/xmlschema-2/`.

[BT04] Bray, T. – Paoli, J. et al.: *Extensible Markup Language (XML) 1.0 (Third Edition)*. W3C Recommendation, 2004. `www.w3.org/TR/REC-xml/`.

[CJ99] Clark, J. – DeRose, S.: *XML Path Language (XPath) Version 1.0*. W3C Recommendation, 1999. `www.w3.org/TR/xpath/`.

[FDC01] Fallside, D. C.: *XML Schema Part 0: Primer*. W3C Recommendation, 2001. `www.w3.org/TR/xmlschema-0/`.

[RK04] Richta, K.: *Rodina formátů XML*. Sborník z XXIV. konference EurOpen.CZ, Sněžník, Dolní Morava, 2004.

[THS01] Thompson, H. S. – Beech, D. – Maloney, M. – Mendelsohn, N.: *XML Schema Part 1: Structures*. W3C Recommendation, 2001. `www.w3.org/TR/xmlschema-1/`.