VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

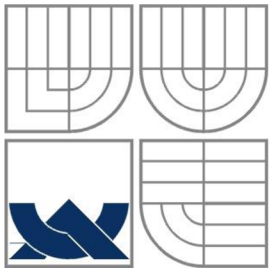
PERZISTENCE XML V RELAČNÍ DATABÁZI

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

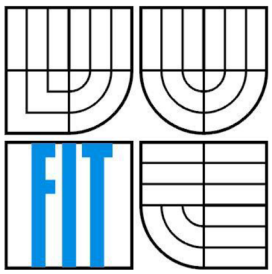
AUTOR PRÁCE
AUTHOR

Bc. Martin Boháč

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PERZISTENCE XML V RELAČNÍ DATABÁZI

XML PERSISTENCE IN RELATIONAL DATABASES

SEMESTRÁLNÍ PROJEKT
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MARTIN BOHÁČ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR CHMELAŘ

BRNO 2010

Abstrakt

Cílem této diplomové práce je vytvoření klienta xDB databáze se schopností vizualizace a správy XML dokumentů a schémat. Úvodní část se věnuje seznámení s jazykem XML, schématy XML (DTD, XML Schema, RelaxNG aj.) a se souvisejícími technologiemi. Poté se práce zabývá problémem perzistence XML a zaměřuje se na techniky mapování nutné pro efektivní ukládání do relační databáze. Hlavní část je věnovaná návrhu a implementaci klientské aplikace XML Admin, která je naprogramovaná v jazyce Java. Aplikace používá rozhraní XML:DB pro komunikaci s databází xDB. Podporuje ukládání XML dokumentů do kolekcí a jazyk XPath pro dotazování. Závěrečná část se věnuje výkonnostním testům aplikace a porovnání s existující nativní databází eXist.

Abstract

The aim of this thesis is to create a client xDB database, which supports visualization and management of XML documents and schemas. The first part deals with the introduction of XML, XML schemas (DTD, XML Schema, RelaxNG, etc.) and contextual technologies. After that the thesis deals with the problem of the XML persistence and it focuses on mapping techniques necessary for an efficient storage in a relational database. The main part is devoted to the design and implementation of client application XML Admin, which is programmed in Java. The application uses the XML:DB interface to communicate with the xDB database. It supports storing XML documents to a collection and the XPath language for querying them. The final section is devoted to application performance testing and comparison with existing native database eXist.

Klíčová slova

XML, relační databáze, nativní XML databáze, databáze s podporou XML, XPath, XQuery, DTD, XML Schema, Relax NG.

Keywords

XML, relational databases, nativ XML database, XML-enabled database, XPath, XQuery, DTD, XML Schema, Relax NG.

Citace

Boháč Martin: Perzistence XML v relační databázi, semestrální projekt, Brno, FIT VUT v Brně, 2010

Perzistence XML v relační databázi

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Petra Chmelaře. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Martin Boháč
6. ledna 2009

Poděkování

Velmi rád bych na tomto místě poděkoval Ing. Petru Chmelaři za poskytnutou pomoc a konzultace při tvorbě této práce.

Citace

Boháč Martin: Perzistence XML v relační databázi, semestrální projekt, Brno, FIT VUT v Brně, 2010

Obsah

1 Úvod

Efektivní šíření a uchování informace bylo pro lidstvo vždy důležité. S rozvojem věd a učení přibývala nutnost vědomosti uchovat a šířit mezi lidmi. V historii se informace šířily zejména pomocí řeči, později pomocí knih, tiskovin a s příchodem techniky například pomocí rozhlasu či televize. V dnešní době díky rozvoji výpočetní techniky se stal největším zdrojem informací internet. Informatika jako věda se zabývá strukturou, správou, uchováváním, získáváním, šířením a přenosem informací [16]. Pro uchování dat ze kterých lze čerpat potřebné informace, vzniklo v počítačovém světě mnoho formátů elektronických dokumentů např. pdf, xls, doc aj. Tyto formáty mají jednu společnou nevýhodu potřeby speciálního softwaru pro zpracování dat uložených v tomto formátu. Proto vznikl XML (Extensible Markup Language - rozšiřitelný značkovací jazyk), jednoduchý otevřený formát pro ukládání strukturovaného a semi-strukturovaného textu, jež se stal standardním formátem pro výměnu informací. S rozšířením XML vznikl problém jak tyto dokumenty efektivně uchovávat a proto kromě databází s přidanou podporou XML vznikají nové nativní XML databáze.

Tato diplomová práce se zaměřuje na dokumenty XML a zabývá se problémem perzistence, vizualizace a transformace XML.

1.1 Cíl práce

Cílem této práce je vytvoření klientské aplikace pro vizualizaci, uchování a transformaci XML dokumentů a schémat v běžné objektově relační databázi. Diplomová práce navazuje na práci skupiny zabývající se XML vizualizací a perzistencí, v jejímž rámci jsou na FIT VUT v Brně řešeny problémy spojené s vizualizací a efektivním uchováním XML. Skupina zahájila svoji činnost v roce 2006. První výsledky vizualizace XML jsou shrnuty v publikaci Interactive Visualization of Data-Oriented XML Documents [30].

Konkrétně tato práce navazuje na aplikace vytvořené v bakalářské práci Michala Filuse Interaktvína vizualizácia XML [40], Martina Seka Interaktivní vizualizace XML [19] a diplomové práci Radima Hernycha Transformace a perzistence XML v relační databázi [18]. K ukládání XML dat do databází bude využit databázový systém xDB a pro vizualizaci budou použity některé části programů XML Spark a XML visualizer. Praktický výstup této diplomové práce je součástí autorizovaného SW vyvíjeného na FIT XML Visualization vyvíjeném od roku 2008, jejichž autory jsou Boháč Martin, Chmelař Petr, Filus Michal a Pospěch Ladislav.

1.2 Struktura práce

První část této práce přibližuje strukturu jazyka XML a do jakých kategorií se dělí. Poté částečně vysvětluje, co jsou to XML schémata a jaké druhy existují. Také zmiňuje existující dotazovací jazyky v souvislosti s XML, jako jsou jazyk XPath a XQuery.

Druhá kapitola je zaměřena na databáze podporující XML a jeho nativní databáze. Je poukázáno na výhody a nevýhody obou druhů databází a jsou zmíněny některé jejich zástupce.

Následující kapitola se věnuje způsobům mapování XML do relační databáze a na problémy s tím související. Zmiňuje druhy technik, které lze použít, pokud máme k dispozici schéma nebo pokud máme jen samotný XML dokument.

Čtvrtá kapitola pojednává o návrhu aplikace. Popisuje již vyvinuté systémy a jejich části, které lze využít pro vytvoření výsledné aplikace a konkretizuje funkce a případy užití, jaké by měla splňovat.

Další kapitola se zaměřuje na samotnou implementaci klientské aplikace XML Admin. Popisuje funkčnost a možnosti výsledné aplikace.

Předposlední kapitola je věnována testování databáze xDB a je provedeno srovnání s databázovým systémem eXist.

Závěrečná kapitola shrnuje výsledky práce a obsahuje zamyšlení nad budoucím směrem vývoje a možnostmi rozšíření vyvinutého systému.

V příloze následuje uživatelská příručka k programu XML Admin , obsah přiloženého CD, seznam použitých zkratek a detailní výsledky provedených testů.

2 Úvod do problematiky XML

Úspěch jazyka XML v posledních letech plyne z obecného formátu pro reprezentaci dat, jeho nezávislosti na platformě a použitelnosti v různých oblastech informačních technologií. Díky schopnosti XML uchovávat strukturovaná data, která jsou dobře čitelná lidmi, i dobře zpracovatelná počítačem se stal tento formát velice oblíbený. V této kapitole se blíže seznámíme s jazykem XML a zmíníme si oblasti informačních technologií, ve kterých se používá.

2.1 Jazyk XML

Rozšiřitelný značkovací jazyk (XML - eXtensible Markup Language) ve verzi 1.0 vydán konsorciem W3C v roce 1998 [1]. Navazuje na principy jazyků SGML a HTML a je určen pro ukládání, zpracování a šíření informací. Jazyk SGML, jež byl schválen organizací ISO o dvanáct let dříve, byl navržen velice robustně a ne všechny jeho vlastnosti se zcela využily. Aplikace s plnou podporou dokumentů SGML byly náročné na vývoj a proto vznikali spíše proprietární řešení. Naproti tomu jazyk HTML rozšířený zejména v oblasti internetu, který také vychází z jazyka SGML, neposkytuje možnost rozšířit jazyk pomocí vlastních tagů. Bylo zapotřebí jazyka, jenž by zjednodušoval jazyk SGML a rozšiřoval možnosti jazyka HTML. Jazyk XML tedy umožňuje snadné vytváření značkovacích jazyků pro různé typy aplikací a různé účely. V dnešní době již existuje celá řada nástrojů pro zpracování XML dokumentů a je podporována u většiny programovacích jazyků [17].

Dokumenty XML nejčastěji rozdělujeme do dvou hlavních kategorií: datově zaměřené a dokumentově zaměřené. Datově zaměřené XML dokumenty jsou zejména používány v oblasti pro přenos dat. Například obchodní objednávky, záznam pacienta nebo systematická data. Mají pravidelnou strukturu, nezáleží na pořadí a není povolen smíšený obsah (text a elementy). Dokumentově zaměřené jsou charakterizovány méně pravidelnou strukturou, významem pořadí elementů a povolují smíšený obsah. Jsou to například uživatelské manuály, statické webové stránky aj.

Dokumenty XML musí splňovat určité formátování. Musí mít kořenový element zahrnující celý dokument. Elementy se nesmějí překrývat a všechny tagy musí být párové, tedy každý počáteční tag `<tag>` musí ukončovat koncový tag `</tag>`. U elementů, které mají prázdný obsah je možno toto pravidlo obejít zkráceným zápisem, kdy počáteční tag obsahuje na konci lomítko `<tag />`. Rozlišují se malá a velká písmena ve jménech značek a atributů. Atributy musí být vždy uzavřeny do uvozovek nebo apostrofů [1].

2.1.1 Kategorie XML produktů

Možnosti využití XML je celá řada. Pro lepší zorientování zde uvádím možné rozdělení dle Ronalda Bourreta [5].

Middleware

Je software, tvořící mezivrstvu mezi programy. Z hlediska XML přenáší data mezi XML dokumenty a XML databázemi.

Vývojová prostředí a editory (IDEs and Editors)

Softwary pro psaní XML aplikací nebo editaci XML dokumentů.

Software pro integraci dat

Servery navrženy pro přenos dat mezi zdroji dat.

XML-podporující databáze

Jedná se o (objektově) relační databázové systémy s rozšířenou funkcionalitou pro převážně datově zaměřené XML dokumenty. Těmto databázím a technikám mapování XML dokumentů do relační databáze se budeme věnovat podrobněji v jedné z následujících kapitol.

Nativní XML databáze

Jedná se o databázové systémy ukládající XML dokumenty v nativní formě.

Aplikační servery

Jedná se o XML servery poskytující data ve formátu XML. Nejznámější je Coxoon od Apache Software Foundation, Cold Fusion od Adobe a další.

Wrappery

Jedná se o software, který zpracuje XML dokument jako zdroj relačních dat. Následně se lze nad dokumentem dotazovat pomocí SQL.

Systémy pro správu obsahu

Aplikace vestavěné nad nativními XML databázemi nebo nad systémy souborů.

XML dotazovací nástroje

Nejznámějšími dotazovací jazyky jsou XPath a XQuery, které jsou standardizovány konsorciem W3C.

2.1.2 Jmenné prostory

Pokud chceme v jednom dokumentu použít více sad značek, které jsou popsány odlišnými schématy, může dojít ke konfliktu názvů elementů nebo atributů. Pro rozlišení elementů a atributů se stejným názvem vznikly jmenné prostory [1]. Ty se definují nejprve pomocí atributu s názvem ve tvaru xmlns:prefix, jehož obsahem je URI adresa k nějakému schématu a kde prefix lze nahradit libovolným názvem, který pak dále ve jménech elementů a atributů toto URI zastupuje.

Příklad 2.1: Ukázka jmenných prostorů v XML dokumentu

```
<Q:kniha
xmlns: Q="file: /dtd/kniha.dtd"xmlns:X="http://www.w3c.org/TR/REC-html40">
  <Q:odst>Běžný odstavec</Q:odst>
  <X:td>Buňka tabulky v html</X:td>
</Q: kniha >
```

2.2 Schémata XML dokumentu

Volnost, kterou nám XML dovoluje, nemusí být a není vždy žádoucí. Jazyky pro definování schématu dokumentu formálně definují, jaké elementy lze v dokumentu použít, jaké mohou mít tyto elementy atributy a obsah. Díky schématu dokumentu může během čtení probíhat validace, tedy kontrola platnosti XML dokumentu.

Nejstarší a zároveň nejznámější jazyk definující schéma je DTD (Document Type Definition [15]), který je již pro současné aplikace nedostačující, zejména z důvodu chybějící podpory jmenných prostorů a datových typů.

Dnes je nejpoužívanějším jazykem XML Schema [6], které konsorcium W3C standardizovalo a nazvalo nástupcem DTD. Podporuje jmenné prostory a datové typy (včetně definování vlastních), definuje jaké elementy se mohou v dokumentu vyskytovat a jaké mohou být jejich atributy, pořadí a počet elementů atd. Nezaměřuje se pouze na validaci struktury, ale také na validaci obsahu textových uzlů a atributů a kontrolu integrity. Výhodou XML Schema je, že používá XML syntaxi.

Další možností jak definovat schéma XML dokumentu je použití jazyka Relax NG (REgular LAnguage for XML Next Generation), které je postaveno na vzorech. Tedy oproti výše zmíněným schématům, které popisují každý element samostatně, popisuje elementy ve struktuře množiny uzlů jaké jsou v dokumentu povoleny. Je primárně zaměřen na validaci struktury dokumentu a nemá tak široký záběr jako XML Schema. Díky tomu je mnohem jednodušší a snadno pochopitelný [32].

Převod mezi jednotlivými schématy lze provést např. pomocí nástroje Trang. Je naprogramován v jazyce Java a podpora konverze schémat vychází z objektovém modelu Relax NG.

Jsou podporovány všechny výše zmíněné schémata, kromě W3C XML Schema, které je podporováno pouze jako výstupní formát [33].

2.2.1 Ukázky jazyků schémat

Nejprve si uvedeme jednoduchý XML dokument (příklad 2.2) na kterém budeme demonstrovat možné formáty schémat. Dokument obsahuje informace vždy o jednom zaměstnanci, který má svůj identifikátor, jméno, příjmení, datum narození a plat. Schémata byla vygenerována pomocí programu XML Admin.

Příklad 2.2: Ukázka XML odpovídá níže definovaným schématům(převzato z [9]):

```
<zamestnanec id="101">
  <jmeno>Jan</jmeno>
  <prijmeni>Novák</prijmeni>
  <plat>25000</plat>
  <narozen>1965-12-24</narozen>
</zamestnanec>
```

Příklad 2.3: DTD

```
<!ELEMENT zamestnanec       (jmeno, prijmeni, plat, narozen)>
<!ATTLIST zamestnanec          id            CDATA   #REQUIRED>
<!ELEMENT jmeno         (#PCDATA)>
<!ELEMENT prijmeni      (#PCDATA)>
<!ELEMENT plat          (#PCDATA)>
<!ELEMENT narozen       (#PCDATA)>
```

U elementů se může specifikovat počet výskytu vnořených elementů, případně atributů, pomocí operátorů uvedené v tabulce 2.1. Pro jednoduchost takové elementy náš zaměstnanec nemá, ale pokud by například byly zaznamenány i informace o dětech, přibyl by element `dite` s operátorem `*` uvádějící žádný a více výskytů.

Tabulka 2.1: Operátory jazyka DTD a Relax NG compact.

Operátor	význam
*	žádný a více výskytů
+	jeden a více výskytů
?	Nepovinný, tzn. žádný nebo jeden
\| | A\|B element a nebo B |

...

Příklad 2.4: XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                    xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="zamestnanec">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="jmeno"/>
        <xs:element ref="prijmeni"/>
        <xs:element ref="plat"/>
        <xs:element ref="narozen"/>
      </xs:sequence>
      <xs:attribute name="id" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="jmeno" type="xs:string"/>
  <xs:element name="prijmeni" type="xs:string"/>
  <xs:element name="plat" type="xs:string"/>
  <xs:element name="narozen" type="xs:string"/>
</xs:schema>
dsafd
```

Syntaxe XML Schema je oproti ostatním složitější a na první pohled hůře pochopitelná.

Příklad 2.5: Relax NG

```
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
  <start>
    <element name="zamestnanec">
      <group>
        <attribute name="id">
          <text/>
        </attribute>
        <group>
          <element name="jmeno">
            <text/>
          </element>
          <element name="prijmeni">
            <text/>
          </element>
          <element name="plat">
            <text/>
          </element>
          <element name="narozen">
            <text/>
          </element>
        </group>
      </group>
    </element>
  </start>
</grammar>
```

Výhodou syntaxe Relax NG, stejně jako XML Schema, je formát XML. Schéma lze uložit do XML databáze a pracovat jako s klasickým XML dokumentem. Pro specifikaci vícenásobného

výskytu dětského elementu se dětský element vloží do nového elementu, jehož název určí počet. Příkladem může být element s názvem `zeroOrMore`, `oneOrMore` pro vícenásobný výskyt nebo `optional` pro volitelný element [10].

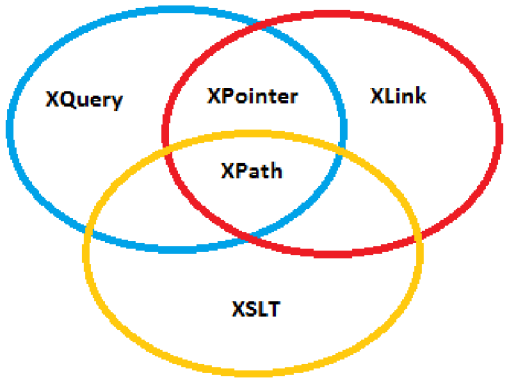
Příklad 2.6: Relax NG compact

```
element zamestnanec {
  attribute id { text },
  element jmeno { text },
  element prijmeni { text },
  element plat { text },
  element narozen { text }
}
```

Pomocí kompaktnější syntaxe RNC lze zapsat schéma na minimum řádku a přitom lze snadno a rychle schématu porozumět. Pro vícenásobně vnořené elementy se využívají stejné operátory jako v DTD uvedené v tabulce 2.2. Operátory specifikující počet výskytů se píší za uzavírací závorku elementu, tedy v případě údajích o dítěti by do schématu přibyl řádek ve tvaru `element dite { text }*`.

2.3 Dotazovací jazyky

Jelikož se XML nepoužívá jen pro přenos dat, ale také pro uložení strukturovaných dat, je potřeba prohledávat a vybírat z nich jen požadované údaje. K tomu nám slouží níže popsané dotazovací jazyky XQuery, XPath.



Obrázek 2.1: Ukázka souvislostí jazyků (překresleno dle [7]).

XPath

Je jazykem pro nalezení informací v XML dokumentu a je W3C doporučeným dotazovacím jazykem [7]. K navigaci používá výrazy vyjadřující cestu k výběru uzlu nebo množiny uzlů. Tvar výrazu je podobný tradičnímu souborovému systému. Jazyk XPath zahrnuje přes 100 vestavěných funkcí, zejména pro porovnání různých datových typů jako řetězce (string), numerické hodnoty, datum a čas. Tento jazyk byl navržen pro použití XSLT, XPointer a dalším tzv. parsing software.

Jazyk XPath rozlišuje sedm druhů uzlů: element, atribut, text, jmenný prostor, instrukce, komentář a dokumentový uzel. Datový model XML má stromovou strukturu ve které se lze pohybovat po tzv. osách. Jazyk XPath jich rozlišuje celkem třináct (předek, rodič, dítě, následující a předchozí sourozenec atd.) [7]. Vybírání uzlů probíhá pomocí zadání cesty k požadovanému uzlu, kterou lze specifikovat pomocí osy a predikátu. Rozlišuje dva typy cest, nazvané relativní a absolutní [1]. Pokud známe přesné umístění požadovaného elementu můžeme zvolit typ absolutní cesty. Absolutní cesty se vyznačují začínajícím lomítkem indikující kořen dokumentu. Relativní cesty na rozdíl od absolutních cest začínají od aktuálního uzlu. Nejběžnější používané výrazy jsou uvedené v tabulce 2.2.

Tabulka 2.2: Nejpoužívanější výrazy jazyka XPath [7].

Výraz	Nezkrácený výraz	Popis
název_uzlu	child::nazev_uzlu	Vybírá potomky uzlu podle jména
/		Vybírá kořenový uzel
//	descendant-or-self::	Vybírá uzly v dokumentu od aktuálního uzlu, v libovolném zanoření
.	self::	Vybírá aktuální uzel
..	parent::	Vybírá rodičovský uzel od aktuálního uzlu
@	attribute::	Pomocí toho výrazu se vybírají atributy

Tabulka 2.3: Zástupné znaky (tzv. wildcards) jazyka XPath [7].

Výraz	Popis
*	Vybírá libovolný element
@*	Vybírá libovolný atribut
node()	Vybírá všechny druhy uzlů

Dále můžeme pomocí zástupných znaků (tabulka 2.3) vyhledávat neznámé elementy, atributy resp. uzly libovolného typu. Lze použít predikáty, které jsou vždy umístěny v hranatých závorkách, a představují podmínku, jakou musí splňovat vybrané elementy. Ve výrazech lze použít porovnávací, logické a aritmetické operátory. Příklady několika výrazů cest a jejich výsledek aplikovaný na XML dokument z přikladu 2.2 uvedený v předchozí podkapitole jsou uvedeny v tabulce 0.

Tabulka 2.4: Ukázka výrazů cest jazyka XPath.

Výraz	Popis
//zamestnanec[@id='101']	Vybere zaměstnance s hodnotou id rovnající se 101
//zamestnanec/jmeno | //zamestnanec/plat	Vybere všechny jména a plat ze všech elementů `zamestnanec`
//zamestnanec[@*]	Vybere všechny elementy `zamestnanec`, které mají nějaký atribut
/zamestnanci/zamestnanec[1]	Vybere prvního zaměstnance
zamestnanec/jmeno	Vybírá jména všech zaměstnanců

XQuery

Dle W3C [8], jehož je XQuery také doporučením, je založený na XPath a dokonce XQuery 1.0 zahrnuje XPath 2.0. Podporuje stejné funkce a operátory, každý dotaz v jazyce XPath 2.0 je zároveň dotazem XQuery 1.0. Integrovaná funkcionalita jazyka XPath slouží pro navigaci po stromové struktuře dokumentu. Dále k funkčnosti XPath přidává možnost agregace dat výsledku dotazů a jejich filtrování, zavádí FLWOR výrazy, výrazy konstruující nové elementy, možnost vytváření uživatelsky definovaných funkcí aj.

Struktura FLWOR (čte se „flaur“) výrazu (převzato z [12]):

- FOR – výběr posloupnosti uzlů k dalšímu zpracování
- LET – přiřazení proměnných pro každý prvek posloupnosti
- WHERE – filtrování uzlů v posloupnosti
- ORDER BY – seřazení vybraných a odfiltrovaných uzlů
- RETURN – specifikace výstupu pro každý vybraný a odfiltrovaný uzel

Praktickou ukázku flower výrazu vidíme na příkladu 2.7, kde z dokumentu obsahující informace o zaměstnancích vyberu jen ty, které mají plat větší jak 15000. Následně jsou ještě seřazeny podle jména.

Příklad 2.7: Ukázka dotazu pomocí FLWOR výrazu.

```
for $x in doc("zamestnanci.xml")/zamestnanci/zamestnanec
where $x/plat>15000
order by $x/jmeno
return $x/jmeno
```

Proměnné jsou označeny znakem `$`, řetězce mohou být v jednoduchých nebo ve dvojitých uvozovkách. V XQuery jsou také povoleny podmíněné výrazy (If-Then-Else), kde může být využito obecných operátorů porovnání (`<, <=, =>, >, =, !=`) nebo nově zavedených operátorů `lt, le, gt, ge, eq, ne` pro tzv. hodnotové porovnání.

SQL/XML

Standardním a nejrozšířenějším jazykem pro dotazování v relačních databázových systémech je jazyk SQL (Structured Query Language). S rozšířením integrace XML do relačních databází bylo zapotřebí rozšířit jazyk SQL o práci s XML daty. Část SQL 2003 standardu nazvanou "XML-related Specifications" se věnuje standardizaci interakce a integrace SQL a XML. Potencionální oblasti použití je prezentace SQL dat ve formě XML, integrace XML dat do SQL dat, využití XML pro výměnu SQL dat a pohledy nad SQL daty ve formě XML. Rozšíření zahrnuje datový typ XML, funkce pro práci s XML a mapování SQL dat do struktury XML [36]. Ukázku použití datového typu XML vidíme na příkladě Příklad 2.8, ve kterém se nachází příkaz pro vytvoření jednoduché tabulky zaměstnanců. Tabulka obsahuje sloupec pro uložení životopisu zaměstnance ve formátu XML.

Příklad 2.8: Vytvoření tabulky pomocí SQL/XML se sloupcem typu XML.

```
CREATE TABLE zamestnanci (
id integer(11),
jmeno varchar(30),
prijmeni varchar(30),
zivotopis XML)
```

Tabulka 2.5: Tabulka některých SQL/XML funkcí pro generování struktury XML [36].

Funkce	Popis
XMLELEMENT	Vytváří element určitého jména.
XMLATTRIBUTES *(uvnitř XMLELEMENT)*	Vyváří atributy elementu. Musí být uvedena přímo za jménem elementu, případně za deklarací jmenného prostoru.
XMLNAMESPACES *(uvnitř XMLELEMENT)*	Deklarace jmenného prostoru. Musí být uveden přímo za jménem elementu.
XMLCONCAT	Spojí hodnoty více výrazů XML do jedné.
XMLFOREST	Vytvoří posloupnost XML elementů dané výrazem v argumentu. Jména elementů lze explicitně měnit.
XMLAGG	Agreguje hodnoty podobně jako SUM, AVG aj.
XMLCOMMENT	Vytvoření XML komentáře.
XMLPI	Vytváří XML instrukce pro zpracování.
XMLCAST	Konvertuje předefinované typy SQL na atomické typy XQuery a naopak.
XMLQUERY	Vyvolání výrazu v jazyce XQuery nebo XPath.
XMLTABLE	Transformuje data XML do formátu tabulky.
XMLVALIDATE	Zajišťuje validaci XML vůči určitému schématu XML.

Dotazování v jazyce SQL/XML je rozšířeno o tzv. konstrukční funkce (tabulka 2.5), které umožňují získávat výsledek dotazu ve formě XML. Použití těchto funkcí je velice snadné jak

můžeme vidět v ukázce SQL/XML dotazu (příkladu Příklad 2.9) jehož potencionální výsledek je uveden v tabulce Tabulka 2.6.

Příklad 2.9: Ukázka dotazu v jazyce SQL/XML.

```
SELECT XMLELEMENT(NAME "zamestnanci",
XMLNAMESPACES(DEFAULT 'uri'),
     XMLCOMMENT('Můj komentář'),
     XMLCONCAT(XMLELEMENT(NAME "zamestnanec",
               XMLATTRIBUTES(z.id AS "identifikator"),
               XMLELEMENT(NAME "jmeno", z.jmeno),
               XMLELEMENT(NAME "prijmeni", z.prijmeni),
               XMLELEMENT(NAME "bydliste",
               XMLATTRIBUTES(z.stat AS "stat"),c.adresa),
               XMLELEMENT(NAME "telefon",c.tel))
FROM Zamestnanci z
```

Tabulka 2.6: Ukázka možného výsledku SQL/XML dotazu z příkladu 2.9.

Zamestnanci
<zamestnanci> <zamestnanec identifikator=""> <jmeno>Jan</jmeno> <prijmeni>Novák</prijmeni> <bydliste stat="Česká Republika"> Brno, Námořní ulice 1234 </bydliste> <telefon>608 111 111<telefon> <zamestnanec> <zamestnanec identifikator=""> <jmeno>Josef</jmeno> <prijmeni>Nadaný</prijmeni> <bydliste stat="Česká Republika"> Brno, Polní ulice 1234 </bydliste> <telefon>608 222 222<telefon> <zamestnanec> </zamestnanci>

XLink, XPointer

Podobně jako hypertextové odkazy v HTML poskytuje jazyk XLink[1] jednoduché odkazy (simple link), kdy jednotlivé dokumenty jsou propojeny pomocí uri (Uniform Resource Identifikator). Navíc umožňuje odkazování mezi více zdroji, doplnění odkazů o metainformace (způsob zobrazení, okamžik aktivace odkazu aj. [11]) nebo odkazování na části dokumentu za pomocí jazyka XPointer,

který je součástí jazyka XLink. Na elementy mající identifikátor se v dokumentu lze odkazovat přímo. Elementy ovšem nemusí mít vždy definovaný svůj identifikátor a odkazování na ně může být problém. To řeší jazyk XPointer, jež za pomocí jazyka XPath umožňuje odkazovat i na zdroje podle kontextu, ve kterém se nacházejí. Informaci o zdroji přidává na konci URL. Za oddělovacím znakem # je výraz v jazyce XPath, kterým specifikujeme požadovanou část dokumentu.

Příklad 2.10: Způsob odkazování pomocí jazyka XPointer

```
../dokument.xml#kapitola2
../dokument.xml#xpointer(…)
../dokument.xml#xpointer(/zamestnanec/jmeno)
../dokument.xml#xpointer(/uvod/nadpis)
```

Můžeme tak odkazovat například na druhý odstavec v druhé kapitole, na třetí dětský element první kapitoly nebo podle výskytu nějaké fráze např. druhý výskyt fráze "autor článku".

2.4 Transformace

Samotný XML dokument má sice samo-popisující charakter dat, které obsahuje, ovšem nenese žádné informace o jeho formátování. Definovat styl pro vizualizaci dokumentu lze pomocí stylového jazyka. Mezi nejrozšířenější dnes patří kaskádový styl (CSS) a XSL (XML Stylesheet Language). Na rozdíl od CSS sloužil jazyk XSL nejen pro definici vzhledu, ale také pro transformaci XML dokumentů. Proto byl rozdělen na dvě části, jazyk XSLT (XSL Transformation) pro transformaci a XSL FO (Formatting Objects) pro formátování. XSL FO je W3C doporučením a formálně je nazýván právě XSL [34]. Jazyk XSLT je W3C standard pro transformaci struktur XML od roku 1999. Pomocí XSLT lze definovat jakým způsobem se má XML dokument transformovat do jiného formátu např. HTML, textového souboru nebo opět dokumentu XML s odlišnou strukturou [35].

Obrázek 2.2: Ukázka principu transformace (inspirováno podle[1]).

2.5 Shrnutí

Jazyk XML zaplnil díru, která tu před ním byla a od jeho uvedení jako standardu se rozšířil do mnoha oblastí počítačového světa. XML je ve skutečnosti metajazyk a tedy slouží k popisu dalších jazyků [1]. S kombinací schémat pro definici povolené struktury, lze XML vhodně využít pro různé účely např. konfigurační soubory, protokoly, uchování dat aj. Data uložená ve formátu XML mohou být prezentována v různých podobách a vhodných formátech díky stylovým a transformačním jazykům. Data lze pak pohodlně prohlížet na internetu díky převodu do HTML nebo vytisknout jako přehledně naformátovaný text. Pomocí jazyků XPath a XQuery lze efektivně procházet a vybírat jen potřebné množiny uzlů, které vyhovují požadovaným kritériím. S přibývajícím počtem informací uložených ve formátu XML nastává nutnost jejich efektivního uchování a s tím související otázka XML databáze, které se budeme věnovat v nadcházející kapitole.

3 XML a Databáze

Definice databáze [23]:
Systém řízení báze dat (SŘBD) je kolekce souvisejících dat a množina programů umožňující k datům přistupovat. Kolekce dat, obvykle označována jako databáze, obsahuje informace týkající se podniku. Primárním cílem SŘBD je poskytnutí metody pro uložení a znovuzískání databázových informací, která je jednak pohodlná a zároveň efektivní.

Dokument XML by ve smyslu striktně brané definice databáze mohl být považován za databázi. V mnoha směrech se XML dokument nijak neliší od běžného souboru. Na druhou stranu obsahuje kolekci dat, je samo-popisující, přenositelný a data mohou být popsána pomocí grafu nebo stromové struktury. Kdyby jsme zahrnuli i okolní technologie pro práci s XML dalo by se více méně hovořit o databázi. Stejně jako databáze XML poskytuje úložiště ve formě XML dokumentů, má schémata (DTD, XML Schema, Relax NG aj.), dotazovací jazyky (XPath, XQuery, XQL, XML-QL aj.), programové rozhraní (SAX, DOM, JDOM) atd. Naproti tomu postrádá mnoho věcí, které lze nalézt u skutečných databází například efektivní úložiště, indexace, bezpečnost, transakce a integrita dat, víceuživatelský přístup, dotazy nad více dokumenty atd. Tedy použití XML dokumentu či dokumentů jako databázi je možné v prostředí s malým množstvím dat, málo uživateli a nenáročnými nároky na výkon [21].

V ostatních případech je nutno zvolit mezi některou z databází s podporou XML nebo z nativních XML databází. Ty si probereme v následujících podkapitolách.

3.1 Databáze s podporou XML

Definice databáze s podporou XML (XML-enabled database) [29]:
Databáze s podporou XML (XED z anglického XML-enabled database) mohou být definovány jako tradiční databáze s podporou ukládání a manipulace XML dat ve formě dokumentů. Většina XML podporujících databází byla původně vyvinuta jako relační databáze.

Dle formální definice XML:DB databáze s podporou XML (přeloženo z [22]):
Databáze s podporou XML (XEDB)- databáze, která má přidanou XML mapovací vrstvu poskytnutou buďto dodavatelem databáze nebo třetí stranou.Tato mapovací vrstva řídí ukládání a načítání XML dat. Data, která jsou mapována do databáze jsou mapována do formátu specifických pro danou

aplikaci a původní meta-data XML a struktury mohou být ztracena. U dat opětovně získaných jako XML nemusí být zaručena původní podoba. Manipulace s daty může být prostřednictvím konkrétní technologie (např. XPath, XSL-T, DOM nebo SAX) nebo jiných databázových technologií (např. SQL). Základní jednotka skladování v XEDB závisí na implementaci.

Dle definice se tedy může jednat o klasickou relační, objektově relační nebo hierarchickou databázi. Základem každé relační databáze je tabulka, která obsahuje data. Na rozdíl od dokumentů XML, kde jsou vztahy mezi daty vyjádřeny úrovní zanoření, se v relačních databázích data ukládají do tabulek a vztahy mezi nimi jsou provázány pomocí klíčů. Onou podporou je myšlena schopnost ukládání dat XML dokumentů do tabulek v databázi. Ta může být zaručena přímo samotnou databází nebo mezivrstvou (tzv. middlewarem) zajišťující mapování dat do tabulek a opětovné získání dat ve formě XML. Techniky mapování XML dokumentů do relačních databází probereme podrobněji v příští kapitole.

Nevýhodou těchto databází bývá zpětná rekonstrukce dokumentů (round-tripping), kdy oproti původnímu dokumentu se nezachovají komentáře, programové instrukce aj. Proto je toto databázové řešení vhodné spíše pro uložení datově zaměřených XML dokumentů a je určeno spíše pro použití ze strany profesionálních softwarových vývojářů nebo databázových administrátorů, než u koncových uživatelů.

3.1.1 Současné databáze s podporou XML

V současné době existuje celá řada databází s podporou XML. Podle výčtu Ronnalda Burreta [5], jež byl naposledy aktualizován 17. března 2010 lze v současné době k této oblasti zařadit kolem 23 databází. V nadcházející podkapitole si zmíníme některé z nich.

MySQL

Databáze rozšířená zejména v oblasti webových aplikací jako řešení v podobě AMP (LAMP pro Linux) společně s webovým serverem Apache a skriptovacím jazykem PHP. Jedná se o relační databázový systém, který od verze 5.0 již podporuje transakce, triggery, pohledy aj. Díky své jednoduchosti je velice oblíbený a snadno naučitelný. Podpora XML spočívá v podobě mysql a mysqldump utilitách, které vrací data uložená v databázi ve formátu XML vytvořená pomocí tabulkového mapování (viz. kapitola 4.1.1). Data XML se do databáze ukládají jako řetězce. Utility také využívá tzv. ExtratValue a UpdateXML funkce v SQL. ExtractValue akceptuje hodnoty XML a výrazy jazyka XPath. Vrací text uzlu nalezeným výrazem XPath nebo konkatenuje texty oddělené mezerami u více nalezených uzlů. Aktualizace je možná pomocí UpdateXML, který v původním

XML pomocí jazyka XPath nalezne jeden uzel jehož hodnotu nahradí novou hodnotou. V případě nalezení žádného nebo více uzlů se vrací původní XML [27].

Využití je spíše pro zálohování databáze nebo pro přenos mezi databázemi. Pro efektivnější uchování XML dat je tato databáze vhodná v kombinací s middleware softwarem třetí strany.

PostgreSQL

Vyslovuje se "Post Gres Q L". Je to objektově-relační databázový systém. Je podporován celou řadou programovacích jazyků (PHP, Perl, Python, Java(JDBC) aj.). Je rychlejší oproti MySQL ve vykonávání složitějších dotazů a při víceuživatelském přístupu. Od verze 8.3 přibyla podpora SQL/XML. XML může být uloženo pouze jako text do znakových sloupců (char, varchar, CLOB, atd.). Poskytuje několik funkcí pro vykonání XPath dotazu (`xpath_string`, `xpath_bool`, `xpath_number` aj.) a zpracování XML pomocí XSLT [27]. Přesto se stejně jako MySQL hodí i pro kombinaci s middleware softwarem třetí strany.

Oracle

Podle definice XML:DB by se tento databázový systém dal zařadit do kategorie tzv. hybridních XML databází (HXD). Oracle 11g považuje XML jako datový typ XMLType, který může být mapován do relační databáze nebo uložen v nativní formě. Přesněji podporuje tři základní přístupy jak mohou být data ve sloupcích typu XMLType uložena. Prvním z nich je tedy rozkouskování dokumentu do sloupců jedné nebo více tabulek. Mapování XML dokumentu do objektově-relačních tabulek je za pomocí schématu XML Schema, které pokud uživatel nemá k dispozici může být vygenerováno automaticky.

Další možností je uložení dokumentu v podobě velkého binárního objektu BLOB (Binary Large Object) což znamená, že do databáze je uložena binární reprezentace dokument XML. Třetí možností je CLOB (Charakter Large Object), která ukládá dokument v textové formě jako celek a při opětovném získání dokumentu se vrací ve stejné podobě jako byl uložen. Má tedy nejvyšší úroveň tzv. Round tripingu (rekonstrukce dokumentu) oproti předcházejícím dvou metodám, kde je na úrovni DOM. Naproti tomu první dvě zmíněné metody mohou aktualizovat i na úrovni uzlů, kdežto u uložení ve formě CLOB pouze na úrovni dokumentu. Pro všechny zmíněné možnosti skladování je stejný aplikační kód. Přechod z jednoho typu úložiště na jiné lze pouze pomocí exportu a importu databáze. Sloupce typu XMLType mohou být indexovány čtyřmi způsoby:

- Sloupce u objektově-relačního úložiště na které jsou elementy a atributy mapovány lze indexovat pomocí B-stromu.

- Podle XML hodnot uložených v BLOB nebo CLOB indexuje tzv. XMLIndex, který je implicitně nastaven na indexaci celého dokumentu, ale může být omezen na určitou množinu cest nebo indexovat celý dokument s výjimkou určité sady cest.
- Funkčně založené indexy mohou být použity pro všechny typy XML hodnot. Využívá předdefinovaných XPath dotazů k identifikaci hodnot v dokumentu XML.
- Textové indexy poskytující full-textové indexování hodnot elementů a atributů.

Dotazovat se lze pomocí jazyka XQuery nebo fulltextově vyhledávat pomocí jazyka XQuery ora. Aktualizace jsou prováděny přes proprietární SQL funkce umožňující vkládání, přidávání a mazání uzlů. Lze také aktualizovat hodnotu uzlu, který je identifikován pomocí jazyka XPath. Objekty uložené v databázi lze prohlížet pomocí souborového systému XML DB repository, který je primárně navržen pro data XML, ale může být využit i pro jiné druhy dat. K datům lze přistupovat pomocí protokolu HTTP, WebDAV a FTP, stejně tak jako přes JDBC, PL/SQL aj.

Oracle 11g je komerční verze, ale existuje Oreacle 10g Express Edition, která je zdarma a zahrnuje plnou podporu XML. Hlavní vlastností, kterou verze 10g Express Edition oproti novější 11g nepodporuje, je možnost ukládání typu XMLType jako BLOB [27].

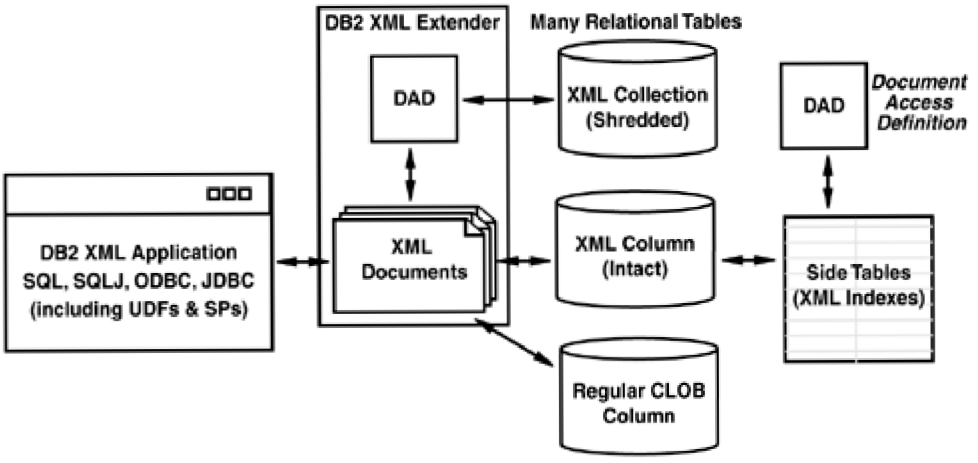
DB2

Skupina produktů DB2 Universal Database (DB2) od IBM poskytuje relační a objektově relační správu dat s podporou pro XML aplikace. Podporu XML lze rozdělit do tří oblastí [2]:

- DB2 XML Extender
- IBM objektově-relační "datablade" poskytující přístup a možnosti ukládání
- Podpora transformace skrze uživatelsky definovaných funkcí a uložených procedur

Pomocí rozšířeného SQL lze vytvářet XML fragmenty z relačních dat použitím podmnožiny SQL/XML funkcí.

Důležitou součástí architektury (Obr. 3.1) je tedy XML Extender, který poskytuje funkční podporu pro typické požadavky aplikací jako je ukládání, přistupování a získávání XML dat. Pro ukládání XML dat nabízí dvě možnosti. První možností je ukládání XML dat do relačních sloupců a druhou ukládání do kolekcí (XML Columns nebo XML Collections). V případě ukládání do sloupců jsou XML data uložena v podobě jednoho ze tří speciálních datových typů: XMLVarchar, XMLCLOB a XMLFile. XMLVarchar je určen pro ukládání malých XML dokumentů, XMLCLOB pro ukládání objemných XML dokumentů a XMLFile pro asociace a spojování dat relační databáze s XML dokumenty uloženými mimo databázi. Ukládání do kolekcí je použito pokud chceme uložit surová data z XML dokumentů do množiny malých relačních tabulek. Také je vhodné při aktualizaci nevelkých sekcí dokumentů [2].



Obrázek 3.1: Možnosti uložení v DB2 (převzato z [2]).

3.2 Nativní XML databáze

Pro lepší pochopení co si pod pojmem nativní XML databáze představit zde zmiňuji některé definice.

Definice (přeloženo z [39]):
U nativních XML databáze není XML fragmentován, ale spíše uložen jako celek do nativní XML databáze.To znamená, že dokumenty jsou uloženy, indexovány a jsou získatelné v jejich originální podobě s celým svým obsahem, tagy, atributy, odkazy na entity a se zachovaným pořadím. Touto technikou je XML databáze navržena pro práci s XML v nativním formátu a tím urychlení přístupu k datům z důvodu odstranění potřeby transformovat data do řádků a sloupců standardní relační databáze.

Dle formální definice XML:DB nativní XML databáze (NXD) (přeloženo z [22]):

- *Definuje logický model XML dokumentu a umožňuje ukládat a získávat dokumenty odpovídající tomuto modelu. Minimálně musí model obsahovat podporu alespoň pro elementy, atributy, PCDATA a to s ohledem na pořadí. Příkladem takových modelů je datový model XPath, XML infoset a modely vycházející z DOM a událostí ze SAX 1.0.*
- *Považuje XML dokument za základní (logickou) jednotku databáze, stejně jako je základní jednotkou řádek v tabulce u relačních databází.*
- *Nemusí implementovat konkrétní model fyzického uložení dat. Ten tak může být postaven např. na relační, hierarchické nebo objektově orientované databázi, případně používat vlastnický formát pro ukládání dat jako např. indexovaný komprimovaný soubor.*

Definice nám udává základní představu o nativních XML databázích, nyní si zmíníme některé jejich typické vlastnosti, které lze u většiny nalézt. Jednou z nich jsou kolekce. Tyto kolekce mají podobný význam jako tabulky v relačních databází nebo jako adresář v souborovém systému. Tedy např. pro ukládání knih zvolíme kolekci knihy. Následné vyhledávání určitých knih lze omezit na tuto kolekci. Mezi další nutnou vlastností je podpora dotazovacích jazyků. Dnes nejpopulárnější je jazyk XPath a XQuery nebo bývá vytvořen nějaký proprietární dotazovací jazyk. Mnoho proprietárních řešení je také v oblasti editace a mazání, přestože jsou již standardy v podobě jazyka XUpdate od XML:DB Initiative a množina rozšíření pro jazyk XQuery od grupy z W3C [21].

Nativní XML databáze jsou vhodné pro datově i dokumentově zaměřené XML dokumenty. Ukládají XML dokumenty ve své nativní podobě tedy přímo jako dokument XML a proto při

opětovném získání dokumentu (tzv. Round-Triping) nechybí žádné informace v dokumentu uložených jako jsou komentáře, instrukce pro zpracování aj.

Největší slabinou těchto databází je editace. Pokud nevyužívají jazyk XUpdate, který je schopen editovat XML dokument přímo v databázi, musí XML dokument nejprve načíst z databáze, provést změny a následně znovu uložit. Pro zrychlení operací provádějí nativní XML databáze indexaci ukládaných dat. Indexují jména elementů a atributů čímž se snadněji vyhledává, ovšem je to zátěž navíc zejména pro aktualizaci.

3.2.1 Současné Nativní XML databáze

V současné době existuje celá řada nativních XML databází. Podle výčtu Ronnalda Burreta [5], jež byl naposledy aktualizován 17. března 2010 je nativních XML databází celkem čtyřicet. V nadcházející podkapitole si zmíníme některé z nich.

Berkeley DB XML

Oproti ostatním zde zmíněným databázím se jedná o tzv. embeded databázi, která je v současnosti vyvíjena společností Oracle. Není tedy samostatný databázový server, ale XML databáze zabudovatelná do vlastního softwaru. Ukládá XML dokumenty do logických skupin, nazývaných kontejnery (odpovídá kolekcím v jiných XML nativních databázích), které jsou indexovány na základě jejich obsahu. K datům přistupuje pomocí jazyka XQuery. Poskytuje aplikační rozhraní pro aktualizaci dokumentů za použití jazyka XQuery pro identifikaci množiny uzlů, které se mají aktualizovat. Umožňuje uživatelům přidat cílovému uzlu nový dětský uzel nebo vložit uzel před nebo za cílový uzel. Také lze cílový uzel odstranit, přejmenovat nebo změnit jeho hodnotu. Podporuje transakce, zálohování, kompresy a šifrování dat na disku pomocí AES. Má programové rozhraní pro jazyky C++, Java, Tcl, Perl, Python a PHP, pomocí kterého lze rozšiřovat funkčnost XQuery [25]. Berkeley DB XML poskytuje rychlou, spolehlivou a škálovatelnou perzistenci pro aplikace vyžadující správu XML dat [26].

dbXML

Nativní databáze vytvořená skupinou dbXML Group podporuje čtyři rozdílná uložení dat. První možností je uložení využívající B stromu. Druhou je uložení do paměti, pro dočasné použití. Třetí možností je souborový systém a čtvrtou je mapování do relační databáze (není známo jaké mapování je použito).

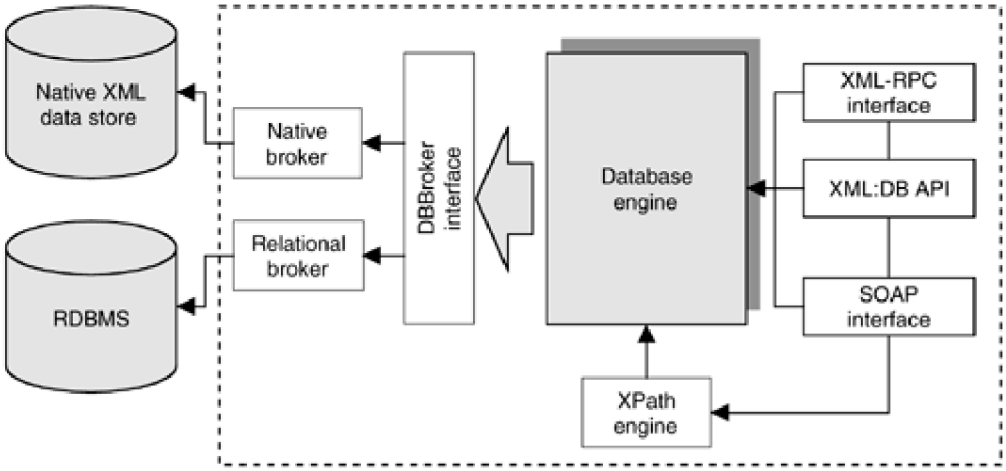
Adresáře má jako kolekce. Kolekce mohou být vloženy a mohou ukládat dokumenty odpovídající libovolnému XML schématu, ale v každé kolekci mohou být dokumenty odpovídající pouze jedinému XML schématu. Databáze dbXML podporuje XPath, XSLT, XUpdate a fulltextové

vyhledávání. Podporuje čtyři aplikační rozhraní: přímé API, klientské API, XML:DB a Webové služby.

eXist

Jedná se o Open Source XML nativní databázový systém, kompletně napsaný v jazyce Java. Může běžet jako samostatný server, ve spojení se servlet kontejnerem nebo lze snadno integrovat do vlastní aplikace. Má mnoho podobností s databázovým systémem Xindice, ale každý je určen pro jiné typy aplikací a založen na rozdílné architektuře. Databázový systém eXist klade důraz na indexově založené zpracování dotazů. Poskytuje rychlé zpracování XPath dotazů užitím indexů pro všechny elementy, texty a atributy. Využitím speciálních algoritmů (path join algorithms), může být mnoho dotazů vykonáno pouze pomocí indexů bez nutnosti načítat jednotlivé uzly XML dokumentu. Poskytuje rozšířenou syntaxi jazyka XPath, díky které se lze dotazovat na určité části kolekce nebo nad všemi dokumenty obsažené v databázi. Dále rozšiřuje možnosti jazyka XPath o fulltextové vyhledávání pro podporu dokumentově zaměřených XML dokumentů [2].

Pro vývojáře je poskytnut přístup přes XML-RPC, SOAP, WebDAV a Cocoon pro webové aplikace. Aplikace Java mohou použít rozhraní XML:DB API, které je běžné pro nativní a XML podporující databáze. eXist umožňuje ukládání i dokumentů bez schématu do hierarchických kolekcí. Dokumenty lze ukládat do interního nativního XML úložiště nebo externího relačního systému (MySQL nebo PostreSQL). Nativní úložiště bylo přidáno ve verzi 0.6 a ve většině aplikací dosahuje lepšího výkonu. Architekturu databázového systému eXist můžeme vidět na obrázku 3.1.



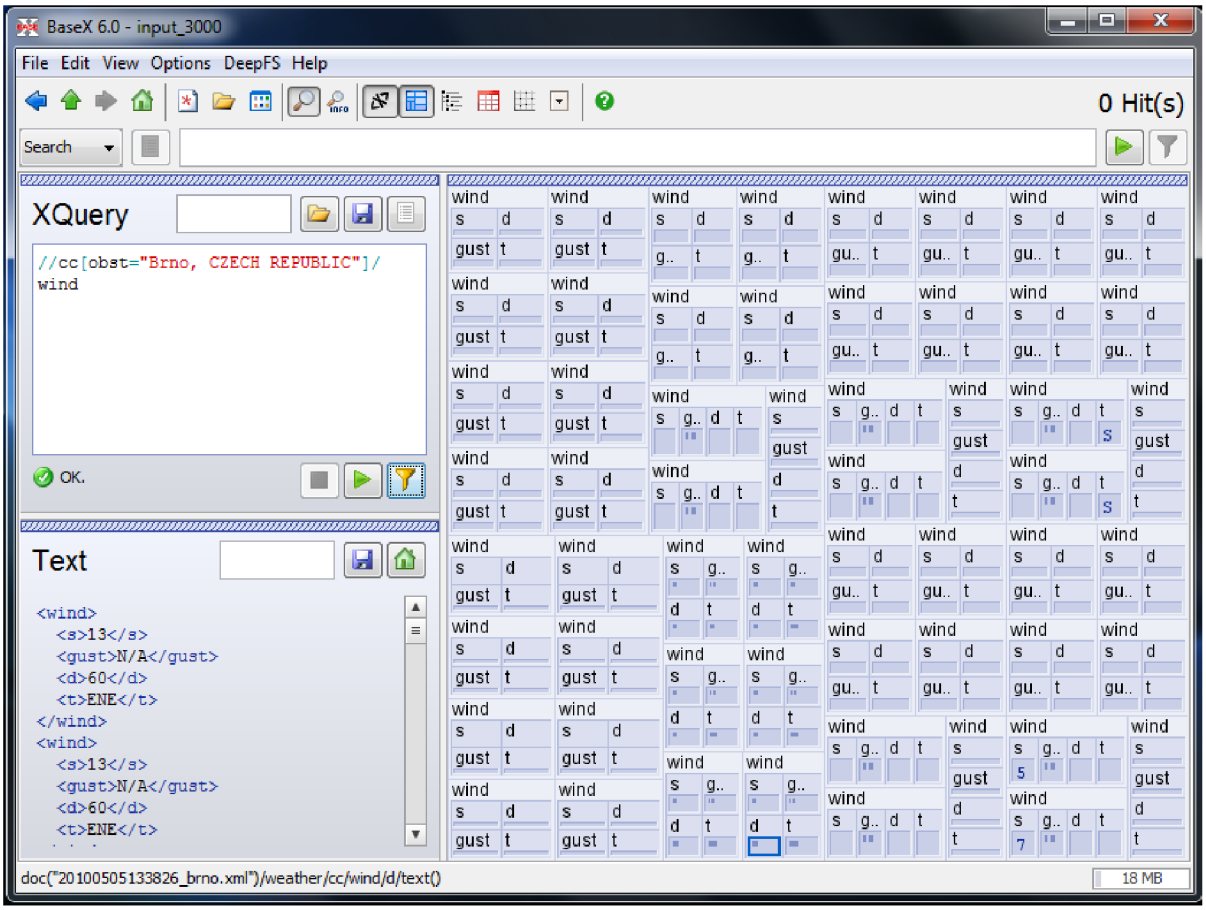
Obrázek 3.2: Architektura eXist (převzato z [2]).

Sedna XML DBMS

Multiplatformní databázový systém vyvinutý na Ruské akademie věd (instituce systémového programování). Jedná se o Open Source XML nativní databázový systém naprogramovaný v jazyce C/C++. Byla navržena s cíly implementovat všechny vlastnosti tradičních databází jako jsou aktualizace, transakce, indexace, zabezpečení apod. Dokumenty se ukládají samostatně nebo do kolekcí. Podporovaným dotazovacím jazykem je XQuery a pro aktualizaci je použit proprietární jazyk vycházející z XQuery Update. Jazyk XQuery je rozšířen o podporu spojení s SQL databází, triggerů a možnosti definovat externí XQuery funkce v jazycích C a C++. Dotazování lze provádět nad jednotlivými dokumenty nebo nad celou kolekcí. Podporuje zálohování včetně záloh za běhu a inkrementálního režimu [24]. Databázový systém umožňuje přístup k několika různým databázím současně a pro každou zvlášť je možnost nastavit uživatelská práva. Také lze nastavit přístupová práva pro celou kolekci nebo dokument. Aplikační programové rozhraní má podporu pro celou řadu programovací jazyků např. Java, C, PHP, Python, Ruby, Perl, Delphi, C# aj. Aplikace Java mohou použít rozhraní XML:DB a XQJ, které je momentálně dostupné jako betaverze [28].

BaseX

Kompaktní nekomerční XML nativní databázový systém s vysoce interaktivní vizualizací, který byl vyvinut na univerzitě Konstanz v Německu [36]. Podporuje rychlé a efektivní zpracování dotazů jazyka XPath a XQuery včetně jeho W3C rozšíření pro fulltextové vyhledávaní a aktualizaci. Doporučení W3C k XQuery Full Text (XQFT) povoluje použití bodovacího modelu a hodnot v rámci dotazů, čehož BaseX využívá. Nabízí efektivní interní bodovací model jež je snadno rozšiřitelný podle použití aplikací. BaseX poskytuje tři bodovací typy, jejíž hodnoty navíc dovoluje ukládat v rámci struktury fulltextových indexů. Způsob bodování může být upraven pomocí nastavení vlastnosti SCORING. Kromě W3C XQFT doporučení podporuje BaseX také tzv. fuzzy vyhledávání, kdy vrací výsledky s použitím fuzzy. To znamená, že výsledkem dotazu mohou být i data s částečnou shodou. Kompilátor dotazů se snaží optimalizovat a urychlit vyhledávání aplikací struktury fulltextových indexů kdykoli je to možné. K dispozici jsou tři vyhodnocovací strategie: standardní sekvenční skenování databáze, vyhodnocování založené na struktuře fulltextových indexů a hybridní, který kombinuje obě předchozí strategie. Od verze 6.0 nabízí plnou implementaci XQuery Update. Všechny aktualizace jsou prováděny nad databázovými uzly a proto se změna nepromítne na původních vstupních datech. Jsou však k dispozici příkazy a XQuery funkce pro vytvoření aktualizovaného XML souboru. BaseX se zaměřuje především na efektivitu, údržba indexů pomocí příkazu OPTIMIZE je ponechána na uživateli. Podporuje práci s velkým XML dokumenty a nabízí vysoce interaktivní front-end. Podporovaná aplikační rozhraní jsou REST/JAX-RX, XQJ a XML:DB.



Obrázek 3.3: Vysoce interaktivní Front-end nativní databáze BaseX [36].

3.3 Porovnání

Které databáze jsou lepší nelze jednoznačně rozhodnout. Srovnání výkonu bylo provedeno v knize [2] na základě testování operací vkládání, mazání, změn, hledání záznamů dle indexu a dalších pro databáze s počtem od 100 do 50 000 záznamu. Na rozdíl od XML-podporujících databází zabírají nativní databáze větší prostor na disku, který u počtu 50 000 záznamů může být více jak 150 krát větší. U operací vkládání, mazání a aktualizování jednoho záznamu mají navrch XML-podporující databáze, které jsou v těchto operacích zhruba o 30% výkonnější. U hromadných operací nad více záznamy zase vykazují lepší výsledky nativní XML databáze.

4 Perzistence XML v relační databázi

V této kapitole se budeme zabývat problémem mapování logické struktury XML dokumentů. Pro řešení tohoto problému se používají techniky, které se mohou rozdělit do dvou hlavních tříd. První z nich je *schématem neřízené mapování* (z anglického schema-oblivious XML storage), které využívá stejné relační schéma pro ukládání různých dokumentů. Druhou třídou je *schématem řízené mapování* (z anglického schema-based XML storage), které vybírá/vytváří různá relační schémata podle schéma definované pro daný XML dokument [14].

4.1 Schématem neřízené mapování

Pokud nemáme k danému XML dokumentu schéma, nezbývá nám než analyzovat příslušný dokument a relační schéma vytvořit za pomocí nějaké obecné metody. V následujících podkapitolách zmiňuji některé z nich.

4.1.1 Tabulkové mapování

Metoda [13] zobrazující dokument jako jednu nebo více tabulek. Výhodou této metody je rychlé pochopení, protože je velmi jednoduchá a lehce naprogramovatelná. Může být použita například pro přenos tabulky mezi databázemi.

Dokument XML se podle úrovně zanoření převádí do jedné nebo více tabulek. Data ve sloupcích mohou být reprezentována v elementech, které jsou definovány jako PCDATA, tedy mohou obsahovat pouze text, nebo jako atributy elementů, které jsou mapovány jako řádky tabulky. Vhodnou strukturu XML dokumentu pro mapování do tabulek vidíme na příkladu 4.1.

Příklad 4.1: Vhodná struktura XML Dokumentu pro Table-based mapping

```
<tabulky>
  <tabulka1>
      <radek>
        <sloupec1>text prvního řádku prvního sloupce</sloupec1>
        <sloupec2>text prvního řádku druhého sloupce </sloupec2>
      </radek>
      <radek>
        <sloupec1>text druhého řádku prvního sloupce </sloupec1>
        <sloupec2>text druhého řádku druhého sloupce </sloupec2>
      </radek>
  </tabulka1>
  <tabulka2>
      <radek sloupec1="text..." sloupec2="text..." />
      <radek sloupec1="text..." sloupec2="text..." />
```

```
  </tabulka2>
</tabulky>
```

Tabulka 4.1: Ukázka Table-based mapping.

sloupec1	Sloupec2
text prvního řádku prvního sloupce	text prvního řádku druhého sloupce
text prvního řádku druhého sloupce	text druhého řádku druhého sloupce

Tato metoda má několik nevýhod. Největší z nich je, že je schopna pracovat pouze s malou podmnožinou XML dokumentů. Neuchovává fyzickou strukturu, komentáře nebo programové instrukce.

Navzdory nevýhodám, které *tabulkové mapování* má, je tato metoda široce používaná například *middleware* programy pro přenos mezi XML dokumentem a relační databází, webovými servery či XML podporujícími databázemi při vracení výsledků.

4.1.2 Objektově-relační mapování

Z důvodu nevýhody tabulkově založeného mapování, které je schopno pracovat pouze s malou podmnožinou XML dokumentů používá většina relačních a objektově-relačních databází s podporou XML více sofistikovanější mapování nazývané objektově-relační mapování. Princip této metody je vcelku jednoduchý. Proces mapování má dvě fáze. Nejprve se data z XML dokumentu namodelují jako strom objektů a poté se tyto objekty mapují do databáze. Podobným způsobem lze vytvářet relační schémata ze schémat XML [13].

Pří použití této techniky na XML dokumentu (příklad 4.2) bylo zapotřebí vytvoření tří objektů (příklad 4.3) a dvou tabulek (tabulka 4.2 a 4.3).

Příklad 4.2: Ukázka XML dokumentu.

```
<zamestnanci>
  <oddeleni>Obchod a prodej</oddeleni>
  <zamestnanec id="11">
    <jmeno>Jan</jmeno>
    <prijmeni>Novák</prijmeni>
    <plat>25000</plat>
    <narozen>1965-12-24</narozen>
  </zamestnanec>
  <zamestnanec id="12">
    <jmeno>Josef</jmeno>
    <prijmeni>Nadany</prijmeni>
    <plat>22000</plat>
    <narozen>1985-1-18</narozen>
  </zamestnanec>
</zamestnanci>
```

Příklad 4.3: Ukázka objektů pro příklad 4.2.

```
object zamestnanci {
      oddeleni = "Obchod a prodej"
      zamestnanecKolekce = {ukazatele na objekty zamestnanec}
}

object zamestnanec {                    object zamestnanec {
      id = 11                                 id = 12
      jmeno = "Jan";                          jmeno = "Josef"
      prijmeni = "Novák";                     prijmeni = "Nadaný"
      plat = 25000                            plat = 22000
      narozen = 1965-12-24                    narozen = 1985-1-18
}                                       }
```

Tabulka 4.2: Ukázka tabulky zamestnanci vytvořená objektově-relačním mapováním.

zamestnanci	
id	oddeleni
1	Obchod a prodej

Tabulka 4.3: Ukázka tabulky osoba vytvořená objektově-relačním mapováním.

	zamestnanec				
oddeleni	id	jmeno	prijmeni	plat	narozen
1	11	Jan	Novák	25000	1965-12-24
1	12	Josef	Nadaný	22000	1985-1-18

4.1.3 Stromové mapování

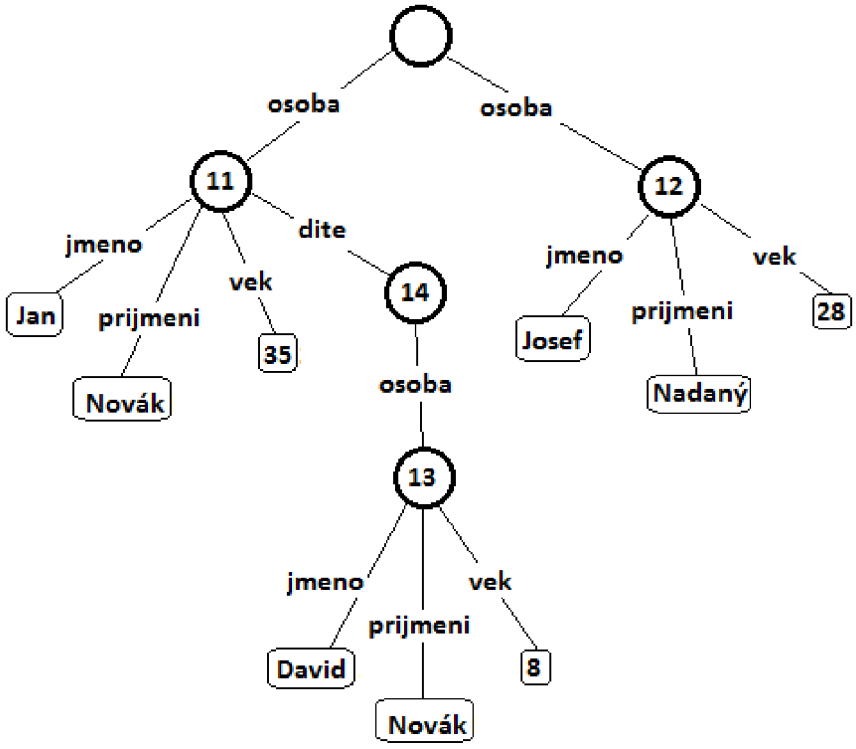
Níže popsané metody budeme pro lepší pochopení demonstrovat na jednoduchém dokumentu XML (příklad 4.3), který je přizpůsobený pro vhodnou demonstraci níže uvedených technik.

Příklad 4.4: Ukázka XML dokumentu.

```
<osoba id="11" vek="35">
  <jmeno>Jan</jmeno>
  <prijmeni>Novák</prijmeni>
  <dite>
    <osoba id="13" vek="8">
      <jmeno>David</jmeno>
      <prijmeni>Novák</prijmeni>
    </osoba>
  </dite>
</osoba>
<osoba id="12" vek="28">
  <jmeno>Josef</jmeno>
  <prijmeni>Nadaný</prijmeni>
</osoba>
```

XML dokument reprezentujeme jako orientovaný stromový graf. Každý element je reprezentován uzlem grafu označeným identifikátorem elementu *id*. Pokud element nemá svůj identifikátor, systém automaticky nějaký vygeneruje. Vztahy mezi rodičovským a dětským elementem resp. atributem jsou reprezentovány pomocí hran v grafu a jsou nazvané jménem dětských elementů resp. atributů. Hodnoty XML dokumentu jsou v listech toho grafu.

Reprezentace XML dat jako grafu je zjednodušení, které může vést ke ztrátě některých informací. Například se nerozlišuje, zda je hodnota uvedena v atributu nebo podelementu. Podíváme se na metody [3], které graf (Obr. 4.1) reprezentující XML dokument z příkladu 4.4, převádí do tabulek relační databáze.



Obrázek 4.1: XML dokument jako orientovaný stromový graf [3].

Hranové mapování

Jedná se o nejjednodušší způsob jak uložit graf reprezentující XML dokument do relační databáze. Stačí pouze jedna tabulka *Hrana* pro uložení všech hran grafu a případně ještě tabulky pro hodnoty listových uzlů (jedna pro numerické hodnoty viz. tabulka 4.5, druhá pro řetězce viz. tabulka 4.6). Do tabulky *Hrana* (tabulka 4.4) se ukládá identifikátor zdroje, identifikátor cíle, jméno hrany, příznak zda se jedná o hranu spojující rodičovský s dětským uzlem nebo zda vede k listu a pořadí, jelikož se jedná o uspořádaný graf [3].

Tabulka 4.4: Ukázka tabulky vytvořené hranovou metodou.

Hrana

IDzdroje	poradi	jmeno	priznak	IDcile
11	1	jmeno	string	v1
11	2	prijmeni	string	v2
11	3	vek	int	v3
11	4	dite	ref	14
12	1	jmeno	string	v4

...

Primárním klíčem v tabulce *Hrana* je zdroj a pořadí.

Tabulka 4.5: Tabulka číselných hodnot.

CiselneHodnoty

IDlistu	hodnota
v3	35
v6	28
v9	8

Tabulka 4.6: Tabulka řetězců listových uzlů.

RetezceHodnoty

IDlistu	hodnota
v1	Jan
v2	Novák
v4	Josef
v5	Nadaný
v6	David

...

Binární mapování

Toto mapovací schéma seskupuje všechny hrany se stejným jménem a vytváří pro ně svou vlastní tabulku. Ta již neobsahuje sloupce se jménem. Pro lepší představu řekněme, že tabulka Hrana z předchozího příkladu se rozdělí na množství tabulek odpovídající počtu rozdílných jmen hran. Primární klič tabulky zůstává stejný [3].

Tabulka 4.7: Ukázka první tabulky vytvořené binární metodou.

HranaJmeno			
IDzdroje	poradi	priznak	IDcile
11	1	string	v1
12	1	string	v4
13	1	string	v7
14	1	string	v10

...

Tabulka 4.8: Ukázka první tabulky vytvořené binární metodou.

HranaVek			
IDzdroje	poradi	priznak	IDcile
11	3	int	v3
12	3	int	v6
13	3	int	v9
14	3	int	v12

...

Univerzální mapování

Dalším přístupem, nazvané *Univerzální,* je generování jedné univerzální tabulky pro všechny hrany. V podstatě se jedná o spojení všech tabulek přecházející metody do jedné univerzální. Pokud tedy máme hrany se jmény n_1 až n_k výsledná tabulka bude [3]:

$$\texttt{Univerzalni(zdroj, poradi}_{n1}\texttt{, priznak}_{n1}\texttt{ ,cíl}_{n1}\texttt{, ... poradi}_{n1}\texttt{, priznak}_{nk}\texttt{ ,cíl}_{nk}\texttt{)}$$

Taková tabulka má na každém řádku velké množství hodnot null ve sloupcích a také obsahuje velké množství redundantních dat. Například pokud osoba Jan Novák by měl dvě děti, bude se v tabulce vyskytovat ve dvou záznamech.

4.1.4 Zhodnocení technik

Poslední tří výše popsané metody mapování XML dokumentů mohou alternativně vkládat (inline) hodnoty přímo do tabulky hran místo do zvláštních tabulek. Z hlediska výkonu u dotazování si vedou dobře, jako např. vybíraní uzlů podle id, podle jména aj. . Nejlépe si vede Binární metoda s inlinováním hodnot. O mnoho horší je to u rekonstrukce XML dokumentu, kdy jsou tyto metody velice pomalé a náročné na paměť. Jedná se tedy o velice jednoduché metody, které nejsou úplně nejlepší volbou pro ukládání XML dat [3].

4.2 Schématem řízené mapování

Mapování řízené schématem odvozuje relační schéma ze schématu dokumentu XML. Díky znalosti elementů, možných atributů a způsobu jakým se mohou elementy do sebe zanořovat lze navrhnout optimální databázové schéma. Základní myšlenkou mapovacích technik je vytváření samostatných tabulek pro opakující se elementy a vložení neopakujících se do tabulky svého rodičovského elementu. Volitelné elementy jsou řešeny nulovými údaji. Alternativní uzly jsou reprezentovány více tabulkami nebo univerzální tabulkou s nulovými údaji. Struktura dokumentu a pořadí uzlů jsou

zaznamenány ve zvláštním sloupci tabulky. Dokumenty XML se smíšeným obsahem nejsou podporovány [20]. Vytvořené relační schéma obsahuje optimální počet tabulek.

4.2.1 Fixní metody mapování

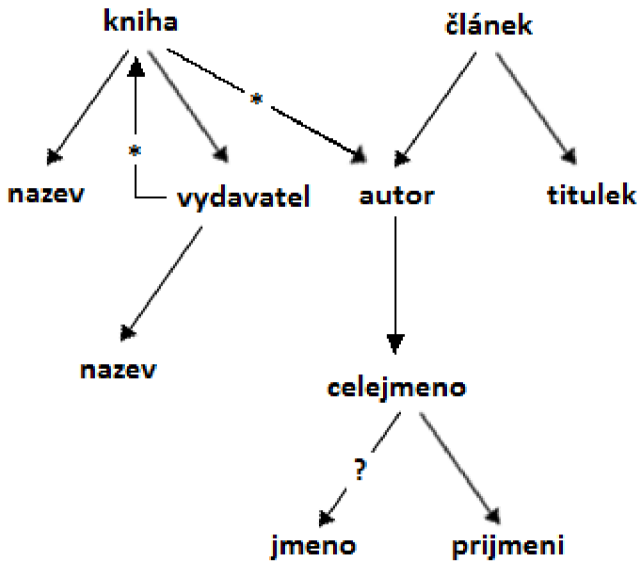
Přímé mapování elementů na relační tabulky vede k nadměrné fragmentaci dokumentu. Proto vznikají různé techniky založené na vkládání (inlining) hodnot dětských uzlů do tabulek svých rodičů. Inlining je typická schématem řízená technika pro ukládání XML dokumentů. Obecně se rozlišují tři druhy inlining technik [4]: *Basic*, *Shared* a *Hybrid*. Záměrně jejich názvy nepřekládám, protože jsou známy pod tímto anglickým názvem. Základní myšlenka těchto technik je vkládání dětských elementů nevyskytujících se často do relačních tabulek svých rodičovských elementů. U elementů, které se mohou objevovat vícekrát, se vytváří vlastní relační tabulka. Efektivnost spočívá také v zacházení s nulovými prvky, které jsou ukládány do zvláštních tabulek nebo je vytvořena jedna univerzální, pro ukládání nulových atributů. Vztahy rodičovského elementu se svým potomkem jsou mapovány pomoci primárních a cizích klíčů.

Pro demonstraci fixních metod mapování je zapotřebí nejprve vytvořit schéma dokumentu XML, zvolíme schéma DTD na kterém byly techniky založeny (příklad 4.5).

Příklad 4.5: Ukázka DTD pro demonstraci inlining metod.

```
<!ELEMENT kniha (nazev, autor, vydavatel)
<!ELEMENT clanek (titulek, autor*)>
<!ELEMENT vydavatel (kniha*)>
<!ATTLIST vydavatel nazev CDATA #REQUIRED>
<!ELEMENT autor (celejmeno)>
<!ATTLIST author id ID #REQUIRED>
<!ELEMENT celejmeno (jmeno?, prijmeni)>
<!ELEMENT jmeno (#PCDATA)>
<!ELEMENT prijmeni (#PCDATA)>
<!ELEMENT nazev (#PCDATA)>
<!ELEMENT titulek (#PCDATA)>
```

Vytvořené schéma DTD (příklad 4.5) lze pro lepší orientaci zobrazit v orientovaném grafu jako můžeme vidět na obrázku 4.2.



Obrázek 4.2: DTD graf reprezentující strukturu schématu DTD [4].

Basic inlining

Tato technika řeší problém fragmentace vkládáním potomků elementu do jedné relační tabulky. Nicméně vytváří relační tabulku pro každý element, který dle schématu může být kořenový. Problémem je, že každý element definovaný ve schématu DTD může být zároveň elementem kořenovým. Například element *autor* bude mapován do své vlastní relační tabulky se sloupci jméno, příjmení a adresa, ale také tyto sloupce budou vloženy v tabulce vytvořené pro element *článek*, který obsahuje právě jednoho autora. Tedy informace o daném autorovi se budou v databázi nacházet duplicitně [4]. Pokud má navíc nějaký element např. *kniha* více výskytů svého potomka např. *autor*, vytvoří se opět další tabulka *kniha.autor a* vzniká tak zbytečně velký počet tabulek v relační databázi. Ukázky tabulek vygenerované touto technikou podle zvoleného DTD z příkladu 4.3 vidíme v tabulce 4.9 až 4.14.

Tabulka 4.9: Ukázka tabulky Kniha vygenerovanou technikou Basic.

Kniha					
kniha_id	kniha. nazev	kniha. vydavatel.id	kniha. vydavatel.nazev	kniha.autor. celejmeno.jmeno	kniha.autor. celejmeno.prijmeni

Tabulka 4.10: Tabulka autorů knihy, kterých může být více, jsou v samostatné tabulce.

kniha.autor			
kniha.autor_id	kniha.autor.id_rodice	kniha.autor.celejmeno.jmeno	knih.autor.celejmeno.prijmeni

Tabulka 4.11: Tabulka Clanek opět obsahuje informace o autorovi.

Clanek			
clanek_id	clanek.titulek	clanek.autor.celejmeno.jmeno	clanek.autor.celejmeno.prijmeni

Tabulka 4.12: Ukázka tabulky Vydavatel.

Vydavatel		
vydavatel_id	vydavatel.id_rodice	vydavatel.nazev

Tabulka 4.13: Infromace o autorovi je uložena i v samostatné tabulce.

Autor		
autor_id	autor.celejmeno.jmeno	autor.celejmeno.prijmeni

Tabulka 4.14: Pro každý element je vytvořena vlastní tabulka.

Titulek	
id	titulek

Nazev	
id	nazev

Jmeno	
id	jmeno

Prijmeni	
id	prijmeni

Teoreticky může být použití této techniky vhodné pro určité typy dotazů jako je *seznam všech autorů knih, seznam všech autorů článků*, ale pro ostatní druhy dotazů je příliš nevhodná. Například pro dotaz typu *všichni autoři, jmenující se Martin*, se musí vykonávat přes několik tabulek.

Shared inlining

Technika nazvaná *Shared* eliminuje nevýhody techniky *Basic* jedním mapováním každého elementu. Identifikuje dětské elementy, které mohou mít více rodičovských elementů, a pro ně vytvoří jednu vlastní tabulku. Pro uzly vyskytující se více než jednou, i ty co se nemusí vůbec vyskytovat (0..n), se také vytváří zvláštní tabulka. U navzájem rekurzivních elementů se zvolí některý z nich pro nějž se vytvoří tabulka, a ostatní jsou inlinovány do této tabulky. Problémem nastane, pokud je kořenový element inlinován do jiné tabulky. Z tohoto důvodu se pro každý kořenový element, který není v samostatné tabulce, přidává sloupec obsahující příznak je_koren (isRoot).

Jedná se tedy o velice solidní techniku vytvářející relační schéma s rozumným počtem tabulek a tedy i vhodnou pro dotazy různého typu.

Tabulka 4.15: Ukázka tabulky Kniha vygenerovanou technikou Shared.

Kniha							
kniha_id	kniha. id_rodice	kniha. kod_rodice	kniha. nazev.je_koren	kniha. nazev	kniha. vydavatel.je_koren	kniha. vydavatel.id	kniha. vydavatel.nazev

Tabulka 4.16: Tabulka Clanek opět obsahuje informace o autorovi.

Clanek		
clanek_id	clanek.titulek.je_koren	clanek.titulek

Tabulka 4.17: Infromace o autorovi je uložena i v samostatné tabulce.

Autor							
autor_id	autor. id_rodice	autor. kod_rodice	autor. celejmeno. je_koren	autor. celejmeno. jmeno.je_koren	autor. celejmeno. jmeno	autor. celejmeno. prijmeni.je_koren	autor. celejmeno. prijmeni

I kdyby knihy nemohly mít více než jednoho autora, vytvořila by se pro autora samostatná tabulka, aby mohla být sdílena (odtud Shared) s tabulkou Clanek.

Hybrid inlining

Jedná se o techniku velice podobnou předchozí technice *Shared*, která se snaží inlinovat více elementů. Navíc vkládá (inline) dětské elementy, které se mohou vyskytovat vícekrát než jednou, pokud nejsou rekurzivní nebo podelementem operátoru *. Tento způsob redukuje počet spojování tabulek při vykonávání některých SQL dotazů. Tedy na rozdíl od metody *Shared* má článek ve své tabulce navíc vloženo jméno a příjmení autora (tabulka 4.19). Článek totiž na rozdíl od knihy může mít pouze jednoho autora.

Tabulka 4.18: Ukázka tabulky Kinha vygenerovanou technikou Hybrid.

Kniha					
kniha_id	kniha.id_rodice	kniha. kod_rodice	kniha. nazev.je_koren	kniha.nazev	...

Kniha			
...	kniha.vydavatel.je_koren	kniha.vydavatel.id	kniha.vydavatel.nazev

Tabulka 4.19: Tabulka "článek" opět obsahuje informace o autorovi.

Clanek				
clanek_id	clanek.titulek.je_koren	clanek.titulek	autor.celejmeno. jmeno	autor.celejmeno. prijmeni

Tabulka 4.20: Infromace o autorovi je uložena i v samostatné tabulce.

Autor				
autor_id	autor.id_rodice	autor.kod_rodice	autor.celejmeno.je_koren	...

Autor				
...	autor.celejmeno. jmeno.je_koren	autor.celejmeno. jmeno	autor.celejmeno. prijmeni.je_koren	autor.celejmeno. prijmeni

4.2.2 Zhodnocení technik

Technika *Basic* je dle jména skutečně pouze základní mapovací technikou. Vytváří mnoho relačních tabulek a u větších DTD schémat je toto množství neúnosné a z důvodu omezené virtuální paměti i nemožné. Naproti tomu se metody *Shared* a *Hybrid* jeví jako efektivní a použitelné. Obě vytváří přiměřené množství tabulek a jsou vhodné pro všechny typy dotazu. U některých typů dotazů je *Hybrid* efektivnější, ovšem na úkor duplicitně uložených informací.

4.3 Shrnutí

Pro řešení problému uložení XML dokumentu do relační databáze existuje v dnešní době celá řada technik. Základní metody, které by napadly programátora jako první při potřebě provést mapování, jsou buďto neefektivní nebo jsou použitelné jen pro určitou skupinu XML dokumentů. Tím mám například na mysli výše zmíněné tabulkově založené mapování nebo u schématem řízeného mapování technika Basic. Na druhou stranu existuje i řada sofistikovanějších metod jako je Shared a Hybrid, využívající k mapování schémata a jsou použitelné pro všechny typy dokumentů XML.

5 Návrh Implementace

Obsahem této kapitoly je koncepce klientské aplikace využívající databázi xDB. Zmiňuje požadavky na funkčnost jaké by měla aplikace splňovat. Také obsahuje stručný popis některých softwarů pro vizualizaci XML, které byly vyvinuty na fakultě FIT. Další existující řešení byly zmíněny v kapitolách 3.1 a 3.2.

5.1 Neformální specifikace požadavků

Je zapotřebí vyvinout systém, který umožňuje správu XML dokumentů a schémat s možností efektivního ukládání XML dokumentů do relační databáze. Systém musí poskytovat možnost dotazování nad uloženými daty pomocí jazyka XPath nebo XQuery. Aplikace by měla být schopna vizualizace pro lepší porozumění spravovaných XML dokumentů. Primárně by měla být zaměřena na datově orientované XML dokumenty.

V jednotlivých bodech by systém měl splňovat následující požadavky:

- Umožnit nastavení připojení k databázi.
- Zobrazení stromové struktury XML dokumentu.
- Možnost úpravy XML dokumentů a jejich uložení.
- Možnost úpravy XML schémat a jejich uložení.
- Vizualizaci XML dokumentů bez schématu.
- Vizualizaci XML dokumentů do tabulek s využitím XML schémat.
- Generování XML schémat z XML dokumentu.
- Ukládání dokumentu XML do databáze.
- Hromadné ukládání XML dokumentů.
- Dotazování pomocí jazyka XPath.

5.2 Případy použití

Podle neformální specifikace požadavků byl sestaven diagram případu použití (viz. obrázek 5.1). Aktéry jsou dva typy uživatelů klientské aplikace XML Admin. Prvním z aktérů je nepřihlášený uživatel, který má možnost správy XML dokumentů a XML schémat. Pokud je dokument XML otevřený, má uživatel možnost jeho vizualizace. Druhým aktérem je přihlášený uživatel, který má

možnost řady dalších užití. Má možnost vytvářet a mazat kolekce. Do kolekcí může ukládat dokumenty, mazat je nebo je z nich načítat. Přihlášený uživatel má navíc možnost vizualizovat i dokument načtený z databáze. Také se může dotazovat v jazyce XPath nad XML daty uloženými v databázi provádět dotazy v jazyce XPath.

Obrázek 5.1: Use case programu XML Admin.

5.2.1 Detaily případu užití

UC1. Připojení ke kolekci

Případ použití:	Připojení ke kolekci
ID:	1
Popis:	Uživatel se přihlašuje na základě přihlašovacího jména a hesla k databázi určené v url.
Aktéři:	Uživatel
Hlavní tok:	Uživatel zvolí tlačítko Connect nebo zvolí položku Connect v menu XDB. Systém zobrazí formulář pro přihlášení. Uživatel zvolí tlačítko connect. Systém nalezne kolekci a připojí se k ní.
Vstupní podmínky:	Existující databáze Existující uživatel s právy přístupu ke zvolené databázi.
Výstupní podmínky:	Uživatel je připojen ke kolekci
Alternativní toky:	Kolekce daného jména neexistuje. Špatně zadané přihlašovací údaje. Zvolena neexistující databáze.
Výjimky:	Selhání systému
Poznámky:	

UC2. Dotaz v jazyce XPath

Případ použití:	Dotaz v jazyce XPath
ID:	2
Popis:	Uživatel se přihlašuje na základě přihlašovacího jména a hesla k databázi určené v url.
Aktéři:	Uživatel
Hlavní tok:	Uživatel zobrazí záložku s názvem XPath Uživatel zadá do Textového pole "Query" dotaz v jazyce XPath. Systém vykoná dotaz a průběžně vypisuje výsledky dotazu a informuje uživatele o počtu výsledků. Při vykonávání dotazu je pozadí výsledkové pole zbarveno do oranžova. Po vykonání dotazu jsou všechny výsledky zobrazeny v textovém poli "Result".
Vstupní podmínky:	Systém je připojen ke kolekci
Výstupní podmínky:	Systém zobrazuje výsledek dotazu
Alternativní toky:	Dotaz XPath není validní zobrazí se chybová hláška.
Výjimky:	Selhání systému
Poznámky:	

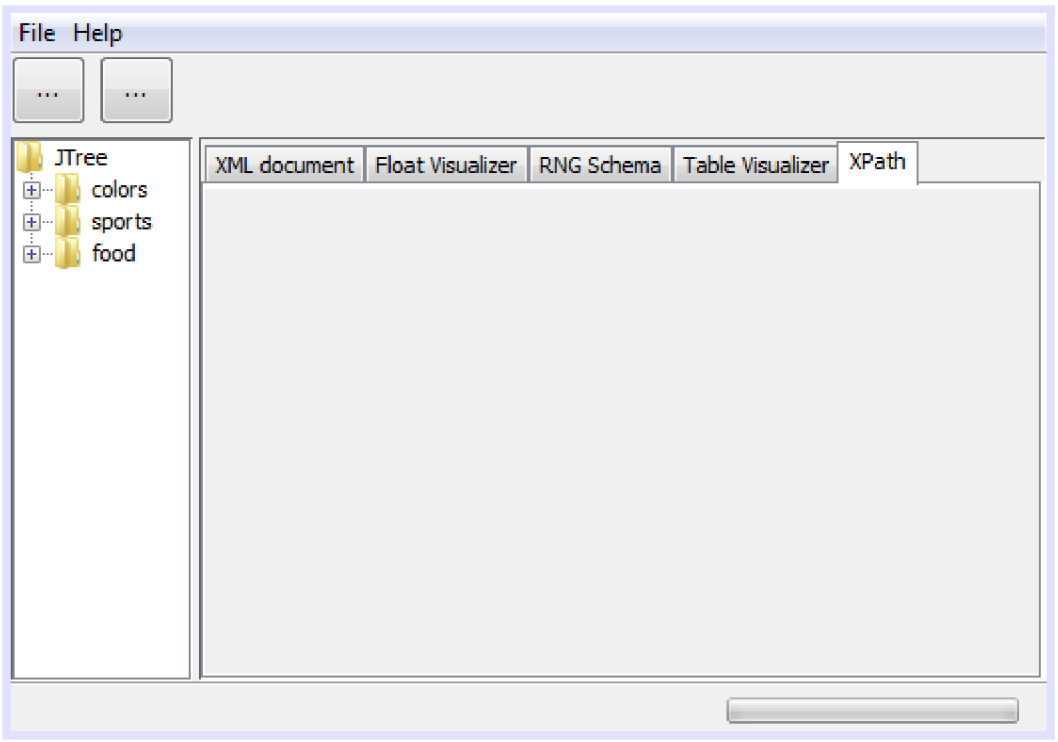
5.3 Návrh GUI

Je potřeba aby byl klient schopen spravovat dokument XML a zároveň jeho schéma. U datově orientovaných XML dokumentů se v textové formě hůře orientuje, proto bude užitečné mít současně

možnost procházet dokument ve vizualizované formě. Rychle by se mělo dát přejít k obrazovce pro vyhledávání v databázi pomocí jazyka XPath. Důležitá je přehlednost a intuitivní ovládání.

Základní koncept vidíme na obrázku 5.5. Na horním panelu by měly být ikony zpřístupňující nejčastější funkce aplikace. Na levé straně hlavního okna aplikace bude neustále zobrazena vizualizovaná stromová struktura otevřeného XML dokumentu s možností změny šířky pro dobré zobrazení většího množství zanořených elementů. Na pravé straně hlavního okna aplikace by měl být systém záložek, které lze pojmenovávat a přehledně tak informovat uživatele co se na dané záložce bude nacházet. Ve spodní části využijeme statusbar pro informování uživatele o právě probíhající operacích.

Jedná se o podobné rozložení jaké bylo zvoleno v programu Spark [19], kde bylo GUI zvoleno vhodně a vzhledem k předpokladu využití implementovaných funkcí z této aplikace bude výhodné podobné rozhraní.



Obrázek 5.2: Návrh aplikace XML Admin.

5.4 xDB

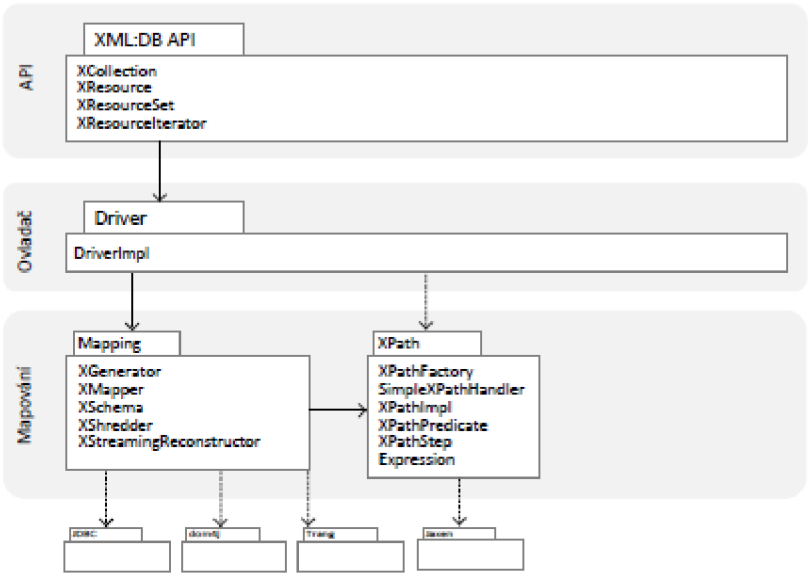
Databáze xDB implementoval v Javě student Ing. Radim Hernych [18]. Umožňuje připojení k databázi PostgreSQL, vkládání XML dokumentů, sdružování dokumentů do kolekcí a dotazování

pomocí jazyka XPath. Systém poskytuje aplikační rozhraní XML:DB jež je podporováno i jinými databázemi např. eXist, Apache Xindacne, Tamino, BaseX, Sedna aj., což otevírá možnost použití klientských aplikací i pro jiné databázové systémy.

Systém v databázi ukládá kromě dat také metainformace o dokumentech, jejich schématech a kolekcích dokumentů. Pro metainformace jsou v databázi implicitně vytvořeny tři tabulky `xcollection`, `xdocument` a `xtable`. Tabulka kolekcí uchovává, kromě jména kolekce, URI definující XML schéma, samotné schéma dokumentu a jméno kořenové tabulky. Tabulka dokumentů obsahuje mj. jméno a obsah dokumentu. V tabulce `xtable` se uchovávají metainformace o vytvořených tabulkách mapující XML schémata a některé další informace k elementu, ke kterému se tabulka vztahuje.

Databáze xDB primárně určena pro datově orientované dokumenty. Pro ukládání XML dokumentů implementuje schématem řízenou metodu mapování na relační tabulky databáze. Jedná se o proprietární řešení metody Hybrid, kdy na rozdíl od užití schématu DTD (pro které byl Hybrid navržen) využívá výhradně schéma formátu XML Schema. Metoda pro elementy vytváří tabulky nebo je inlinuje do tabulky rodiče podle četnosti jejich výskytu, které se v XML Schema definují pomocí atributů *minOccurs* a *maxOccurs*. Současná implementace vytváří tabulky pro všechny komplexní elementy (tj. elementy obsahující jiné elementy a atributy) s výskytem větším než jedna. Elementy s obsahem jednoduchého typu (tj. string, decimal, boolean, date aj.) a atributy jsou vždy mapovány jako sloupce v tabulce svého rodiče. Mapovací vrstva je implementována v balíčcích `org.xdb.map` a `org.xdb.xpath`. Podpora generování XML Schema z XML dokumentu je implementována ve třídě `XGenerator` využívající knihovnu Trang. Zpětná rekonstrukce XML dokumentů z databáze včetně podpory dotazovacího jazyka XPath je implementována ve třídě *XStreamingReconstuctor*. Při vykonání XPath dotazu jsou výsledky vráceny postupně a systém má tedy dobrou odezvu i při velkém počtu výsledků.

Systém je sestaven ze tří vrstev tj. aplikační vrstvy, vrstvy ovladače a mapovací vrstvy (viz. obrázek 5.3). Nejnižší vrstva implementuje zmíněné mapování XML schémat a XML dokumentů do databáze, podporu rekonstrukce a dotazování v jazyce XPath. Vrstva ovladače je tvořena balíčkem `org.xdb.driver`, který obsahuje implementaci ovladače pro připojení k databázi PostreSQL a poskytuje rozhraní pro manipulaci s kolekcí a dokumenty. Horní vrstva je vrstva aplikačního rozhraní XML:DB (XAPI), které je podporováno i u řady jiných XML databází. Aplikační rozhraní je implementováno v balíčku `org.xdb.api`.

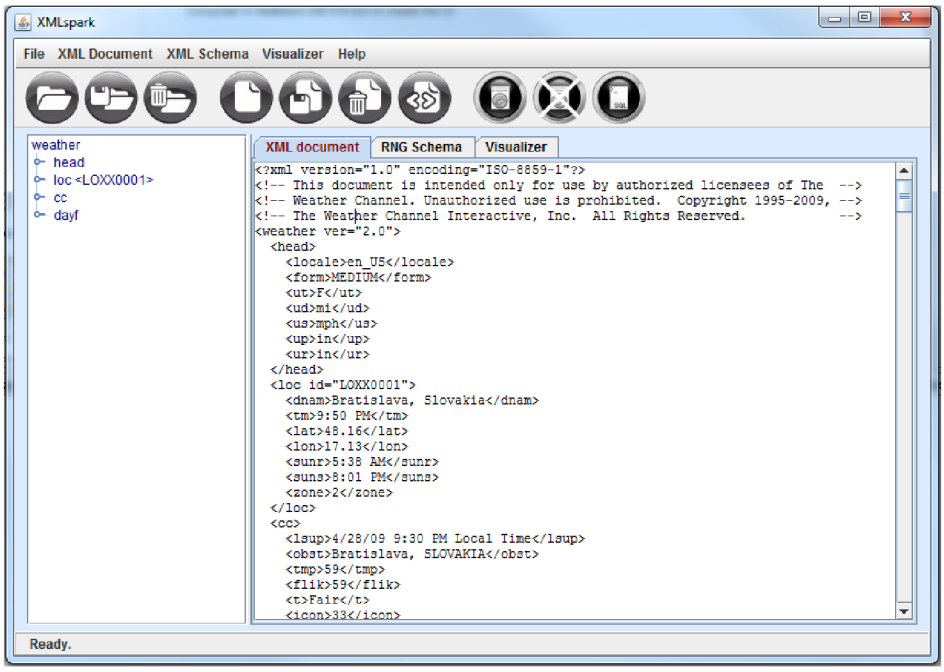


Obrázek 5.3: Zjednodušený diagram balíčků systému s naznačenými vrstvami (převzato z [18]).

5.5 XML Spark

V loňském roce byla na naší fakultě vytvořena aplikace pro interaktivní vizualizaci XML. Vytvořil ji student Martin Seko [19]. Program je implementovaný v jazyce Java pomocí Swing GUI a balíčků JDOM, RelaNG, DOM4j a Apache-commons. Tento program umožňuje prohlížení XML dokumentů, XML schémat a je schopen jejich vizualizace. Vizualizace je podporována ve dvou podobách, první z nich je ve formě stromové struktury a druhá ve formě tabulek. Po otevření XML dokumentu je aplikace v tzv. modu *XML and Schema*, kdy je aktivní pouze záložka se zobrazeným obsahem XML dokumentu, případně i záložka s obsahem XML schématu. Dalšími podporovanými mody jsou *Schema only* a *Visualization only*. Pokud je XML dokument i schéma otevřené, lze stiskem tlačítka *Visualize* dokument vizualizovat. Poté se na levé straně aplikace zobrazí stromová struktura XML dokumentu (viz. levá část obrázku 5.4) a zpřístupní se záložka *Visualizer*, na které je XML dokument zobrazen ve formě tabulek. Ve stromové vizualizaci se lze libovolně zanořovat a zobrazovat jen potřebné detaily včetně hodnot atributů a elementů. Pro procházení stromové struktury XML dokumentu je použita knihovna JDOM. Pro vizualizaci ve formě tabulek je zapotřebí schéma, které lze načíst ze souboru nebo pokud schéma nemáme k dispozici je možné ho vygenerovat. Schéma se vygeneruje podle právě otevřeného XML dokumentu a podle nastaveného typu. Podporované typy jsou Relax NG, Relax NG compact, Documet Type Definition a XML Schema Definition. Mapovací technikou použitou pro vizualizaci tabulek je *Hybrid* (přesný popis nalezneme zde [29]). Samotné mapování do tabulek probíhá pomocí schématu ve formátu Relax NG. Pokud je otevřen nebo

vygenerován jiný typ schématu, převede se automaticky do formátu RNG a uloží do dočasného souboru. Vizualizace ve formě tabulek je limitována na maximálně 30 tabulek. V horním menu jako poslední rychlá ikona je *Export to SQL*, bohužel její funkce není implementována.



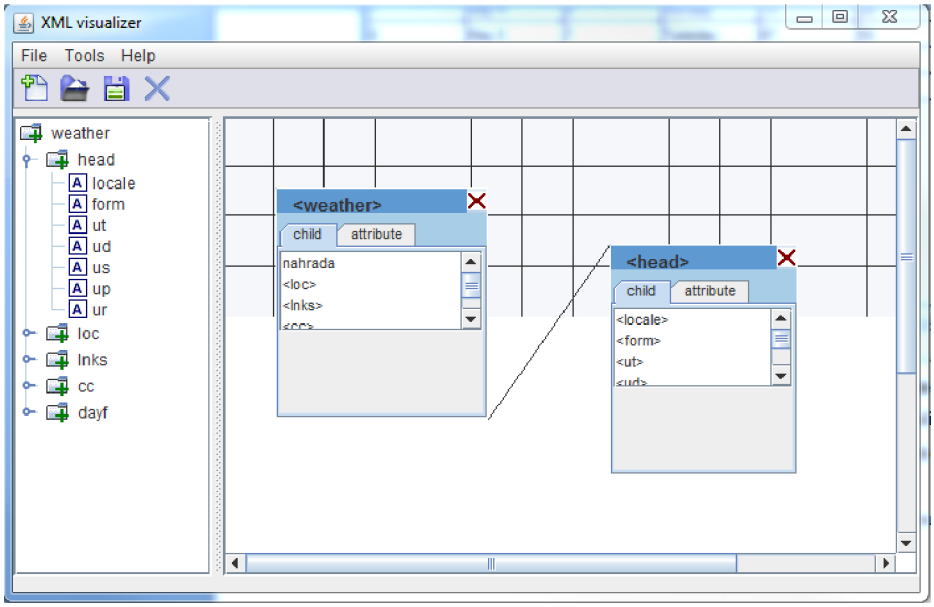
Obrázek 5.4: Ukázka programu Spark [19].

Projekt programu obsahuje 3 balíky *base*, *convert* a *xmltorng*. Balíky *conver* a *xmltorng* jsou využity pro generování schémat. Balík *base* obsahuje grafické uživatelské rozhraní a třídy potřebné pro vizualizaci [19]. Bohužel veškeré grafické komponenty jsou naprogramovány v jediné třídě MainWindow a provázanost grafických komponent a aplikační logiky je velmi vysoká. Přesto program Spark poskytuje mnoho užitečných funkcí, které jsou vhodné i pro klienta xDB databáze.

5.6 XML visualizer

Aplikace XML visualizer (Obr. 5.5) byla také vyvinuta na naší fakultě FIT [40]. Je naprogramovaná v jazyce Java. Pársování XML dokumentů provádí pomocí parser JDOM a pro manipulaci s XML dokumenty využívá rozhraní DOM. Podporuje dva typy vizualizace. První z nich je vizualizace v podobě stromové struktury (viz. levá část obrázku 5.5). Po kliknutí pravým tlačítkem myši na libovolný element ve stromové struktuře se zobrazí menu pro manipulaci s daným elementem jako je přejmenování, mazání, přidání atributu nebo dětského elementu. Bohužel funkce tohoto menu nebyly

implementovány. Druhým způsobem vizualizace umožňuje zobrazovat jednotlivé elementy v samostatných panelech. Každý element zobrazuje ve svém panelu seznam dětských uzlů a atributy, které jsou pro daný element definované. Informace o elementu jsou pro přehlednost rozděleny do záložek. Při dvojkliku na jméno dětského uzlu se zobrazí jeho vlastní panel a je vykreslena spojnice mezi oběma panely jako vztah rodičovského uzlu a jeho potomka. V seznamu dětských uzlů se u uzlů jež mají zobrazen vlastní panel vypisuje slovo "nahrada" namísto původního názvu uzlu. Tuto vlastnost nepovažuji za příliš povedenou, protože při zobrazení více dětských uzlů ve vlastních panelech se v rodičovském panelu ztrácí přehlednost. Původní záměr aplikace byl zřejmě mít vizualizační nástroj s možností pohodlného přidávání, mazání a přejmenování uzlů a atributů. Bohužel tyto funkce se nestihlo implementovat, přesto se jedná o zajímavý vizualizační nástroj, který by mohl být zabudován do klienta XML Admin.



Obrázek 5.5: Aplikace XML visualizer [40].

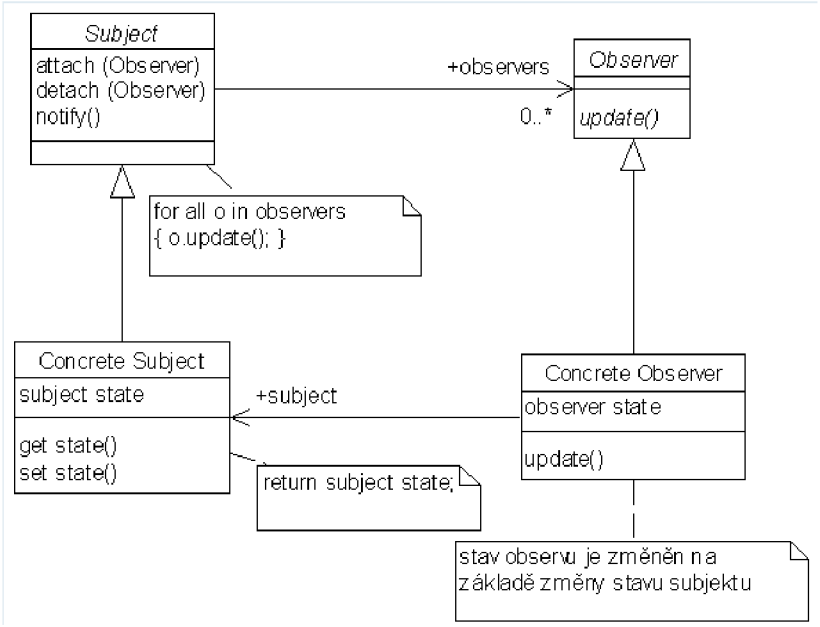
6 Implementace programu XML Admin

Vzhledem k situaci kdy mezivrstva xDB je implementována v Javě, a také možnosti využití některých již vyvinutých vizualizačních nástrojů, byl jako implementační jazyk zvolen Java. Jako vývojové prostředí bylo vybráno NetBeans IDE 6.8 a na počítači instalováno JDK 6 update 20.

Program vycházel z již vytvořeného program Spark a XML Visualizer. Výsledný projekt obsahuje čtyři hlavní balíky `base`, `gui`, `controllers` a `spark`. V balíčku `spark` jsou zahrnuty zdrojové kódy z balíčku `base` projektu Spark kromě tříd definující grafické komponenty. Balíčky `xmltorng` a `convert` z programu Spark pro generování schémat byly upraveny tak, aby nebyly závislé na uživatelském rozhraní a byly z nich vytvořeny knihovny (class library) zabalené v archivu jar. V balíčku `contollers` jsou umístěny ovladače umožňující přístup k programovému rozhraní xDB nebo řídící logiku aplikace. Všechny kontroléry jsou implementované podle návrhového vzoru Singleton (Jedináček) a tedy je k ním globální přístup. O změnách ovladače informují GUI a jiné závislé komponenty pomocí návrhového vzoru Observer (Pozorovatel).

6.1 Grafické uživatelské rozhraní

Výsledný vzhled GUI vidíme například na obrázku 6.3. V projektu jsou veškeré grafické komponenty implementovány v balíčku `gui`, ve kterém jsou obsaženy další tři balíčky: `toolbars`, `dialogs` a `visualization`. Balíček `toolbars` obsahuje všechny lišty s nástroji (pro XML dokument, schéma a databázi). Balíček `dialogs` obsahuje všechna dialogová okna používana v aplikaci.V posledním zmiňovaném balíčku `visualization` se nacházejí ještě balíček `spark` a `xmlvisualizer`. Jsou pojmenované dle programů ze kterých byly komponenty v nich obsažené přeprogramovány. Hlavní změnou bylo oddělení uživatelského rozhraní od aplikační logiky s využitím návrhového vzoru Observer (viz. obrázek 6.1).

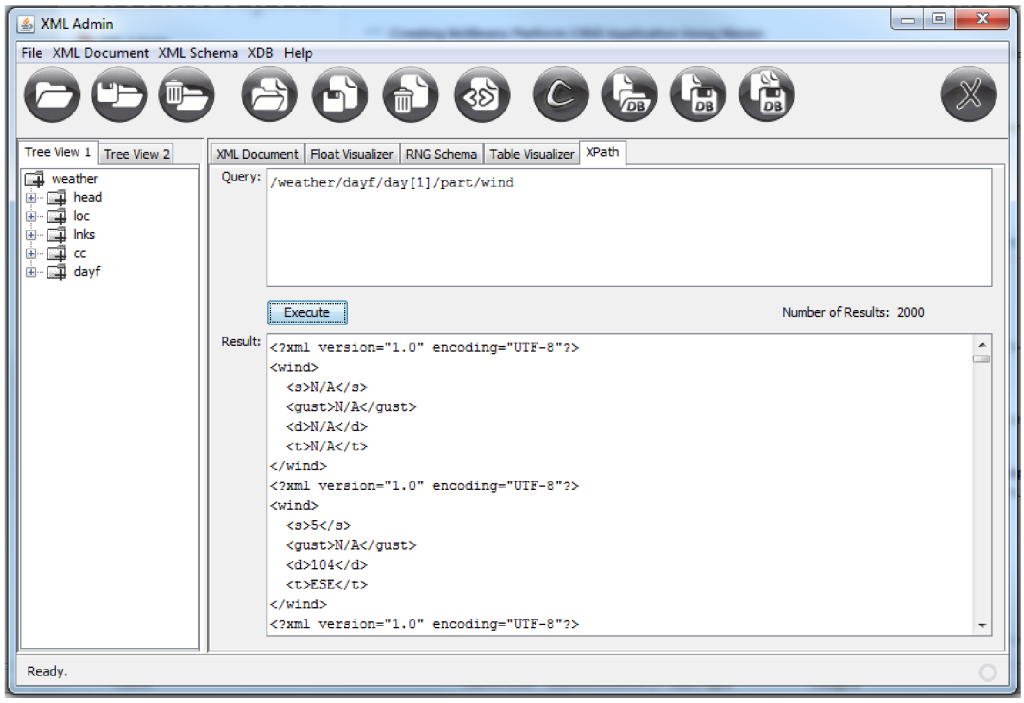


Obrázek 6.1: Návrhový vzor Observer (převzato z [31]).

Přestože se jedná o zcela novou aplikaci, provedu v následujícím textu srovnání s aplikací Spark, ze které se vycházelo.

Oproti programu Spark přibyla možnost volby mezi dvěma vizualizacemi stromové struktury XML dokumentu na levé straně aplikace. První typ stromu je zobrazen ve formě jaké byl v programu XML Visualizer, druhý dle programu Spark. Oba byly graficky sladěny s aplikací a byla upravena jejich funkčnost. Ikony pro ovládací panel dokumentu a schématu byly převzaty z aplikace Spark. Ikony pro práci s databází byly vytvořeny tak, aby ladily s ostatními. Také byl vypuštěn ovládací panel pro vizualizaci, protože vizualizace lze provádět automaticky podle aktuálně otevřeného XML dokumentu a jeho schématu. V horním ovládacím panelu (Obr. 6.2) přibyla nástrojová lišta pro databázi a tlačítko pro rychlé zavření všech dokumentů a odpojení od databáze (viz. vyznačená čtvrtá sekce na obrázku 6.2). V hlavním okně přibyly záložky Float Visualizer a XPath popsané v následujících podkapitolách.

Obrázek 6.2: Horní ovládací panel s vyznačenými funkčními sekcemi.



Obrázek 6.3: Ukázka výsledné aplikace XML admin.

6.1.1 Správa XML dokumentů

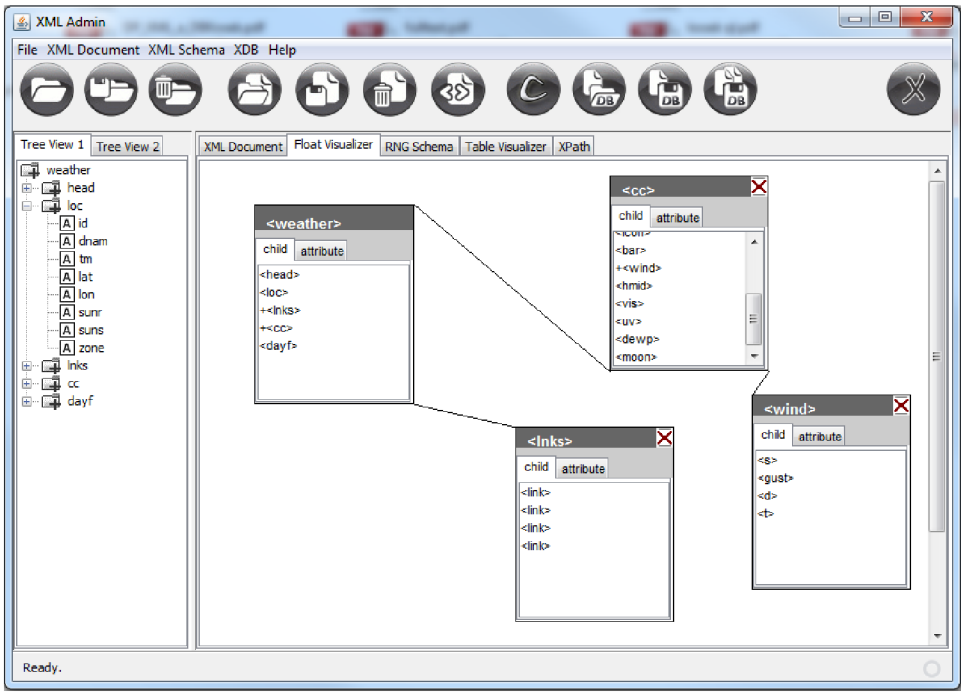
Pro rychlejší správu XML dokumentů slouží první část ovládacího panelu (viz. obrázek 6.2), která umožňuje otevření, uložení a uzavření XML dokumentu. Pokud je XML dokument otevřen, lze ho procházet a upravovat v integrovaném textové editoru umístěným pod záložkou *XML Dokument* nebo procházet pomocí Float Visualizéru (viz. obrázek 6.4).

Textový editor podporuje pouze základní procházení s možností editace včetně změn názvů elementů, atributů a jejich hodnot. Změněný dokument pak lze uložit.

Float visualizer je původní XML visualizer s pozměněnou funkčností a upraveným vzhledem. Zobrazuje element v tzv. element panelu, který obsahuje v záhlaví jeho název a dvě záložky. První záložka slouží pro zobrazení dětských uzlů, ve druhé záložce jsou zobrazeny atributy k příslušnému elementu. Dvojím kliknutím na dětský uzel se zobrazí další element panel obsahující informace k tomuto uzlu. Pokud má dětský uzel zobrazen svůj "element panel", je v rodičovském uzlu rozlišen znakem plus.

Všechny funkce z ovládacího menu jsou přístupné i pod klávesovou zkratkou. O funkčnost ovládacího panelu se stará třída `XDocumentController`, která poskytuje tři hlavní metody `openXMLDocumet()`, `saveXMLDocument()` a `closeXMLDocument()`. Informace o dokumentu

včetně jeho textové reprezentace se ukládají do třídy `XMLhandler`. Tato třída byla přepracována z balíku `base` programu spark a byla odstraněna její závislost na GUI. Pro zpracování XML dokumentu je využit `SAXBuilder` z balíku `org.jdom.input`, který používá parser třetí strany Xerces Java Parser.



Obrázek 6.4: Vizualizace XML dokumentu pomocí Float Visualizer.

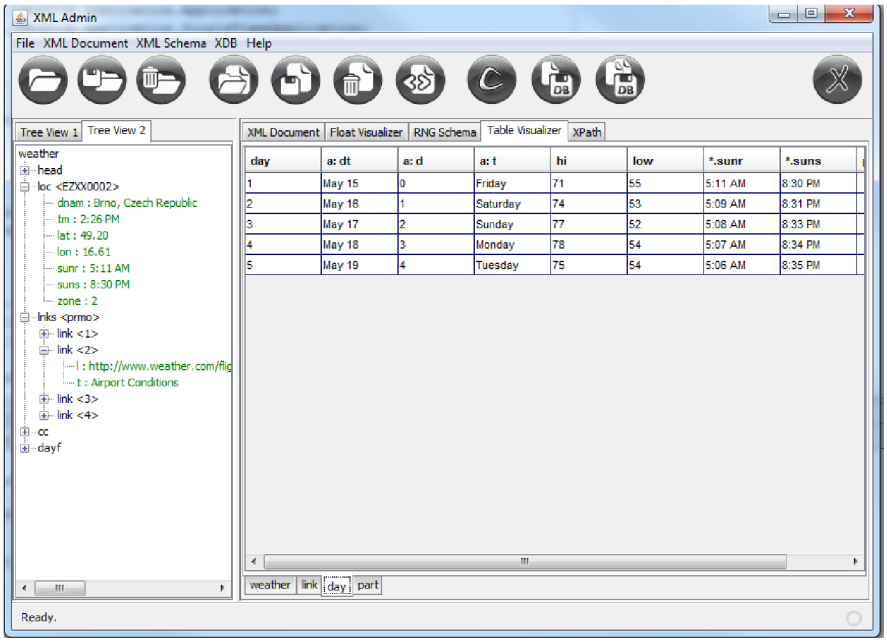
6.1.2 Správa XML schémat

Ovládací panel pro XML schéma (viz. druhá skupina ikon na obrázku Obrázek 6.2) obsahuje kromě otevření, uložení a zavření schématu také jeho vygenerování z XML dokumentu. Tato funkčnost byla implementována podobně jako v programu Spark s využitím některých zdrojových kódů. Změnou bylo opět přesunutí funkčnosti do kontroléru nazvaném `XSchemaController`, ve které jsou obsaženy čtyři hlavní metody `openXSchemaDocument()`, `seveXSchemaDocument()`, `closeXSchemaDocument()` a `Generate()` volané z ovládacího panelu a příslušného menu.

Informace o schématu se ukládají do třídy `XSDHandler` v balíku `spark.base` většinou jako statické proměnné. Pokud je otevřen XML dokument spolu s příslušným schématem, nebo je schéma z něj vygenerováno, automaticky se provede vizualizace ve formě tabulek v záložce `Table Visualizer`.

Generování schémat, jak již bylo řečeno, bylo implementováno v balíčcích `convert` a `xmltorng`. Původně byly zdrojové kódy v nich obsažené součástí open source programu InstanceToSchema [19]. Nastavení typu schématu mezi RNG, RNC, XSD a DTD je možno nastavit v *Schema Preferences*, které je přístupné z *XML Schema* menu.

Table visualizer zobrazuje informace z XML dokumentu v přehledných tabulkách. Dětské uzly které mají maximálně jeden výskyt se inlinují do tabulky rodiče s upraveným názevem *rodič.element*. Pro elementy mající vícenásobný výskyt se vytvoří vlastní tabulka. Na jednotlivé tabulky lze odkazovat pomocí záložek po nich pojmenovaných umístěných ve spodní části pravého hlavního okna (viz. obrázek 6.5). O vytváření tabulek se stará třída `TableCreator`, která byla již naprogramována v programu Spark [19], proto byla umístěna do balíčku `spark.table`.



Obrázek 6.5: Vizualizace XML dokumentu ve formě tabulek.

Vizualizaci aplikace XML Admin na obrázku 6.4 a 6.5 můžeme porovnat v současné době s nejlepším známým způsobem vizualizace obecného XML na obrázku 3.3.

6.1.3 Správa databáze

Hlavním cílem aplikace XML Admin je pohodlná práce s xDB databází. Ovládací panel pro databázi obsahuje čtyři tlačítka (viz. třetí sekce ikon na obrázku 6.2). První tlačítko vyvolá dialog pro připojení k databázi, ve kterém lze vyplnit url databáze, název kolekce, uživatelské jméno a heslo. Po stisku tlačítka připojit se volá metoda `connect()`, která dle zadaných údajů zavolá metodu

`setDatabase()` v kontroléru nazvaném `ConnectionDBController`. Pokud není zadaná kolekce, program se připojí do databáze a díky předpokládanému schématu relační databáze načte všechny kolekce v databázi a vyplní je do rozbalovacího seznamu v dialogu připojení. Poté může uživatel zvolit libovolnou kolekci ke které chce připojit. Další možností, která se v tomto dialogu nachází je možnost odstranění a vytvoření kolekce. Pro odstranění slouží samostatné tlačítko, vytvoření je prováděno automaticky při připojení. Tedy pokud uživatel zadá název kolekce jež se v databázi nenachází, dojde k jejímu vytvoření po potvrzení uživatelem. Pro připojení klienta k databázi bylo nutné zvolit ovladač xDB (viz. příklad Příklad 6.1) a poté pomocí `DatabaseManager` získat přístup ke kolekci.

Databázové schéma xDB se vytvoří na základě XML schématu, které musí být uvedeno v atributu `schemaLocation v` kořenovém elementu XML dokumentu. Pokud uvedeno není, xDB schéma jej vygeneruje.

Po připojení k databázi lze také v záložce XPath (viz. obrázek 6.3) vykonávat dotazy pomocí stejnojmenného jazyka. O funkčnost se starají zejména metody implementované ve třídě `XPathController`. Metoda `query()` zpracovává dotazy jazyka XPath, které jsou zpracovávány ve vlastním vlákně. Jiné vlákno je zodpovědné za průběžné zobrazování výsledků dotazu v aplikaci.

Další dvě tlačítka ovládacího panelu slouží pro vkládání XML dokumentů do databáze. První z nich vkládá dokument, který je právě otevřený v XML Admin. Druhé tlačítko umožňuje vkládání i více dokumentů nebo celých složek.

Příklad 6.1: Zdrojový kód pro připojení klienta k xDB databázi (převzato z [18])

```
// výběr ovladače
String driver = "org.xdb.driver.DriverImpl";
Class c = Class.forName(driver);

// uri databaze
String URI = "xmldb:xdb://localhost/xdb/";
Database database = (Database) c.newInstance();
DatabaseManager.registerDatabase(database);
// získáme referenci na vybranou kolekci
collection = DatabaseManager.getCollection(URI);
```

7 Testování

Kapitola se zabývá srovnávacími testy mezi databází xDB a existující nativní databází eXist. Pro testovací účely byla využita konzolová aplikace xdbamin vytvořená současně s databází xDB pro testovací účely [19]. Díky podpoře rozhraní XML:DB bylo možno využít tohoto klienta pro testování obou databází. U některých testů bude společně s databázemi testována i aplikace XML Admin.

7.1 Příprava k testování

Pro testování byl zvolen můj osobní počítač na kterém byly nainstalovány obě databáze.

Konfigurace PC

AMD Athlon(tm) 64 X2 Dual Core Processor 5000+ 2.60 GHz
4 GB RAM
Windows 7 Professional 64 bit

Testovací data

Prvními testovacími daty byl již hotový datový set obsahující 1000 XML dokumentů o velikosti 6MB. Další testovací data byla získána pomocí vytvořeného skriptu v jazyce Python. Tento skript stahoval záznamy počasí z různých světových měst z portálu na `xoap.weather.com`. Záznamy jsou ve formátu XML, proto je skript mohl rovnou ukládat jako samostatné XML soubory. Takto skript vytvořil datový set obsahující 5000 XML dokumentů i velikosti 27,6 MB.

7.1.1 Instalace databáze xDB

Pro zprovoznění xDB databáze bylo zapotřebí nainstalovat objektově relační databázi PostreSQL verze 8.4. Pro vytvoření databáze a uživatele s právy přístupu byl využit nástroj pgAdmin III. Databáze standardně běží na localhostu portu 5432.

7.1.2 Instalace databáze eXist

K instalaci databáze eXist byla zvolena nová verze 1.4. Při instalaci bylo zvoleno uživatelské jméno a heslo pro administrátora, poté proběhla instalace automaticky bez problémů. V administrátorském webovém rozhraní běžící na adrese http://localhost:8080/exist, je možnost pohodlného procházení kolekcí, jejich vytváření a mazání. Do zvolené kolekce lze snadno vkládat dokumenty XML ovšem

pouze po jednom. Z tohoto důvodu byla pomocí webového rozhraní vytvořena pouze kolekce a dokumenty se do ní vkládaly pomocí konzolové aplikace xDBadmin [18].

7.2 Test vkládání

V prvním testu se porovnával čas potřebný k uložení většího množství XML dokumentů do databáze. Vkládaly se datové sety o velikosti 1000, 3000 a 5000 XML dokumentů obsahující informace o počasí.

Tabulka 7.1: Časy potřebné k uložení dokumentů.

Počet. dokumentů	xDB	eXist
	čas celkem(ms)	čas celkem(ms)
1000	34109	17518
3000	102001	65983
5000	160316	94071

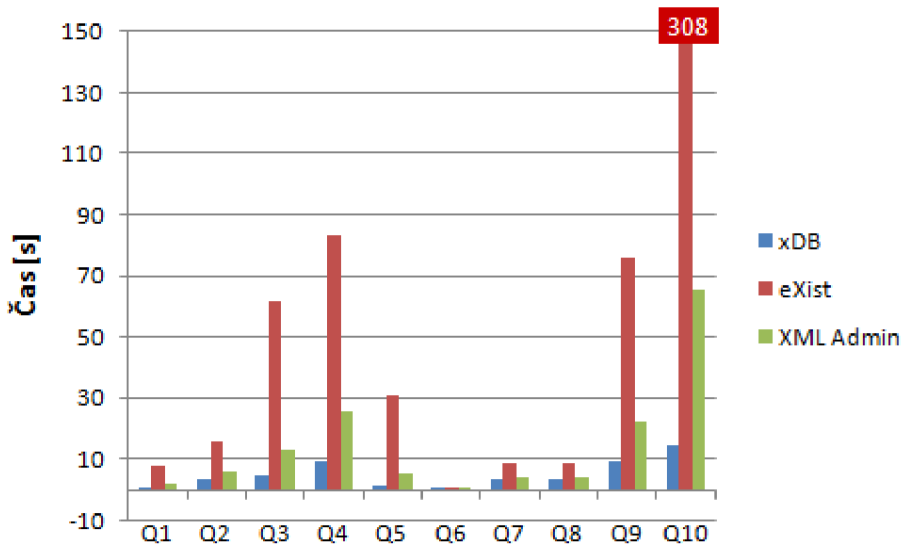
Z výsledků (viz. tabulka 7.1) je patrné, že mapování XML dokumentů do relační databáze je náročnější než-li ukládání v jejich nativní formě. Čas potřebný k uložení stejného množství dokumentů byl u databáze xDB téměř dvojnásobný než-li u databáze eXist. S větším množstvím vkládaných dokumentů přibývá čas potřebný k uložení lineárně.

7.3 Test dotazování 1

Ve druhém testu šlo o ověření výsledků, které byly zveřejněny v publikaci NAXD [37] a k získání výkonnostních hodnot se kterými by mohla být aplikace XML Admin objektivně porovnána. Bylo provedeno deset XPath dotazů (viz. tabulka 7.2) nad stejným datovým setem, tedy 1000 XML dokumentů o velikosti 6MB. Test byl proveden třikrát a poté zprůměrován nejlepší a nejhorší výsledek a zapsán do tabulky 7.3. Poté byly stejné dotazy vykonány v aplikaci XML Admin, kde jsou výsledky dotazu průběžně vypisovány do grafického uživatelského rozhraní a tedy potřeba větší režie.

Tabulka 7.2: XPath dotazy pro druhý test (převzato z [19])

ID	XPath	Popis
Q1	`/weather/head/locale`	absolutní cesta
Q2	`/weather/dayf/day[1]/part/wind`	absolutní cesta s indexem elementu
Q3	`/weather/dayf/day[1]/part/wind/*`	absolutní cesta s indexem elementu a výběrem všech podřízených elementů
Q4	`//wind`	výběr elementů `wind` bez ohledu na umístění v dokumentu
Q5	`//cc/wind/*`	výběr všech
Q6	`//cc[obst="Brno, CZECH REPUBLIC"]/wind`	predikát s testem na rovnost hodnoty elementu
Q7	`//day[@t="Saturday"]/part[1]`	predikát s testem atributu a index podřízeného elementu
Q8	`//day[@t="Saturday"]/part[@p="n"]`	predikáty ve více krocích
Q9	`//part/wind`	výběr věch elementů `wind`, které jsou přímými potomky elementu `part`
Q10	`//part/wind/*`	výběr všech přímých potomků elementu `wind`, které jsou přímými potomky elementu `part`



Obrázek 7.1: Graf s celkovými časy pro zpracování dotazů v jazyce XPath provedené v testu 2.

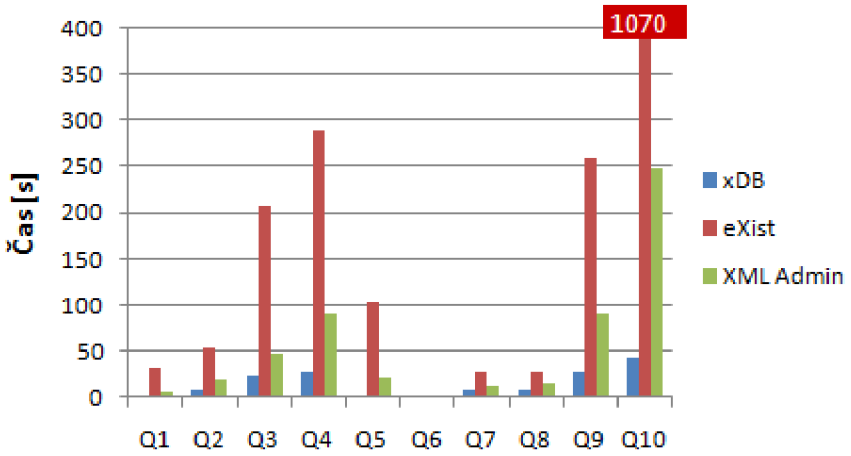
Výsledky testu

Výsledky testu dotazů XPath potvrdily velký rozdíl v rychlosti mezi databází xDB a eXist. I ve srovnání s novou verzí 1.4 databáze eXist rychlostně značně zaostává za databází xDB. Z naměřených časů uvedených v tabulce (viz. příloha tabulka 7.3) a grafu (Obr. 7.1) je databáze xDB několikanásobně rychlejší. Rozdíl se zejména projevoval u dotazů vracejících velký počet výsledků.

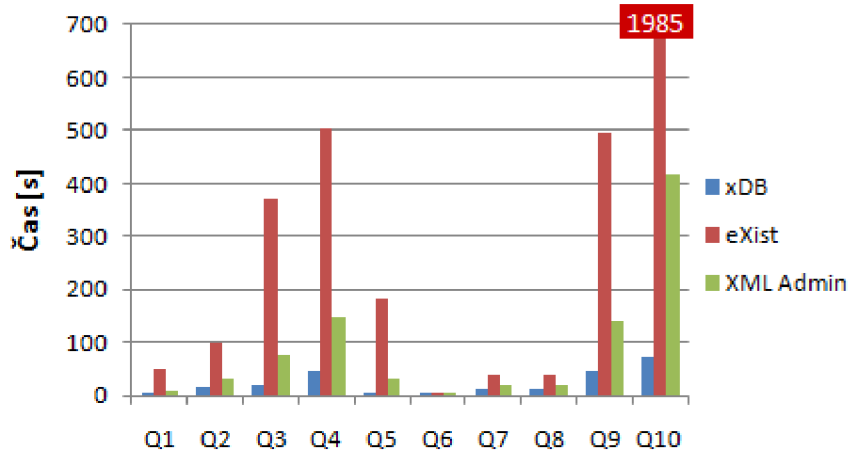
Z časů naměřených při testování klienta XML Admin napojeného na databázi xDB vyplývá určité zpomalení. Průběžné vypisování výsledků v grafickém rozhraní vedlo až k několikanásobnému zpomalení a bude určitě předmětem budoucí optimalizace.

7.4 Test dotazování 2

V posledním testu šlo spíše o zátěžový test s použitím několikanásobně větších datových setů. Byly zvoleny sety o velikosti 3000 XML dokumentů a 5000 XML dokumentů s informacemi o počasí. Vykonány byly stejné dotazy jazyka XPath jako v předchozím testu (tabulka 7.2).



Obrázek 7.2: Graf s celkovými časy pro zpracování dotazů u datového setu 3000 záznamů.



Obrázek 7.3: Graf s celkovými časy pro zpracování dotazů u datového setu 5000 záznamů.

Výsledky testu

Naměřené výsledky jsou uvedeny tabulkách 7.4 a 7.5 jejichž hodnoty jsou vyneseny do grafu 7.2 a 7.3. Z grafů je patrné, že obě databáze si u větších datových setů vedou ve vzájemném srovnání stejně jako bylo u menšího datového setu v předcházejícím testu. O to víc jsou zarážející výsledky databáze eXist, kdy se v testu s 5000 dokumenty u více než poloviny prováděných dotazů muselo čekat na výsledky v rámci několika minut a u dotazu Q10 dokonce více než půl hodiny.

7.5 Shrnutí

Při prvním pohledu na výsledky testů je zřejmá delší doba potřebná pro uložení dat u databáze xDB oproti eXist a naopak méně času pro vykonání XPath dotazů. Ovšem při bližším prozkoumání je nutné poukázat na skutečnost, že zhruba dvojnásobný čas potřebný k uložení dat značně kompenzuje několikanásobně rychlejší vykonání XPath dotazů. Zejména pak u rozsáhlejších databází obsahující několik tisíc uložených dokumentů je pro uživatele jistě rozdíl, zda bude na výsledek dotazu čekat 70 sekund v případě databáze xDB nebo půl hodiny v případě databáze eXist.

8 Závěr

Při řešení této diplomové práce jsem musel značně rozšířit své znalosti z oblasti XML. Seznámit se ze způsoby formátování XML dokumentů a různými druhy schémat pro jejich validaci. Bylo potřeba nastudovat databáze s podporu XML a principy nativních XML databázích.

Mým cílem v rámci diplomové práce bylo navázání na práci předchozích studentů a realizovat systém pro vizualizaci, uchování a transformaci XML dokumentů a schémat. Bylo nutné se seznámit s aplikačním rozhraním XML:DB implementovaném u databáze xDB a nastudování zdrojových kódů aplikací pro vizualizaci XML dokumentů. Během programování aplikace XML Admin byly nacházeny chyby v kódech využívaných aplikací, které bylo nutné prostudovat a opravit, případně doladit nebo zcela změnit jejich funkčnost.

Závěrečná fáze diplomové práce byla věnována výkonnostním testům jak vytvořené aplikace, tak testu samotné databáze xDB. Databáze xDB i s použitím klienta XML Admin byla o mnoho rychlejší ve srovnání s řešením v podobě nativní databáze eXist. Na druhou stranu testy odkryly některé ne zcela optimální skutečnosti, na které se bude muset soustředit další vývoj.

Současně s klientem XML Admin se na naší fakultě vyvíjí nová databáze NeXD, který by již měl mít lepší podporu jazyka XPath a dokonce by měl mít i zabudovanou podporu jazyka XQuery. S podporou jazyka XQuery by se aplikace mohla podrobit řadě existujících aplikačních testů (tzv. benchmarků) a více porovnat s nativními XML databázemi, které v drtivé většině tento jazyk podporují. Možnosti rozšíření funkcionality klienta XML Admin je mnoho. Jednou z možností je rozšíření v oblasti vizualizace. Z hlediska manipulace s daty uloženými v databázi, může být klient tak mocný jako je samotná databáze a naopak. Proto s nadcházejícím příchodem nové verze databáze xDB bude nutné rozšířit i klienta o podporu nově implementovaných funkcích. Dále by bylo vhodné aplikaci opatřit instalátorem, který by během instalace automaticky vytvořil potřebné databázové schéma a umožnil by jednoduše zvolit jméno a heslo uživatele. Prozatím je tento problém řešen předpřipravenými skripty.

Literatura

[1] BRADLEY, Neil. XML : kompletní průvodce. Praha : Grada Publishing, 2000. 536 s. ISBN 80-7169-949-7.

[2] CHAUDRI, Akmal B. XML Data Management : Native XML and XML-Enabled Database Systems. Boston : Addison-Wesley, 2003. 641 s. ISBN 0-201-84452-4.

[3] FLORESCU, D., KOSSMANN, D. Storing and Querying XML Data using an RDMBS. Bulletin of the Technical Committee on Data Engineering . 1999. Dostupné na URL: <http://www.soe.ucsc.edu/classes/cmps290s/Spring03/fk.pdf>.

[4] SHANMUGASUNDARAM, J., et al. Relational Databases for Querying XML Documents: Limitations and Opportunities. 1999.

[5] BOURRET, R. XML Database Products [online]. 2000-2009 [cit. 2009-12-19]. Dostupné na URL: <http://www.rpbourret.com/xml/XMLDatabaseProds.htm#categories>.

[6] Introduction to XML Schema [online]. 1999-2009 [cit. 2009-09-20]. Dostupné na URL: <http://www.w3schools.com/schema/schema_intro.asp>.

[7] XPath Introduction [online]. 1999-2009 [cit. 2009-09-21]. Dostupné na URL: <http://www.w3schools.com/xpath/xpath_intro.asp>.

[8] Introduction to XQuery [online]. 1999-2009 [cit. 2009-09-21]. Dostupné na URL: <http://www.w3schools.com/xquery/xquery_intro.asp>.

[9] KOSEK, J. Kapitola 4. Relax NG [online]. 2004 [cit. 2009-09-21]. Dostupné na URL: <http://www.kosek.cz/xml/schema/rng.html>.

[10] Extensible Markup Language (XML) [online]. 1999-2009 [cit. 2009-09-19]. Dostupné na URL: <http://www.w3.org/XML/>.

[11] KOSEK, J. XLink a XPointer [online]. 1999 [cit. 2009-09-21]. Dostupné na URL: <http://www.kosek.cz/clanky/xml/xml-uvod.html >.

[12] KOSEK, J. XQuery [online]. 2005 [cit. 2009-09-21]. Dostupné na URL: <http://www.kosek.cz/xml/2005devcon/>.

[13] BOURRET, R.. Mapping DTDs to Databases [online]. 2000-2005 [cit. 2010-01-05]. Dostupný z WWW: <http://www.rpbourret.com/xml/DTDToDatabase.htm>.

[14] MLÝNKOVÁ, I.. Relační databáze s XML rozšířením, SQL/XML [online]. 2009 [cit. 2010-01-05]. Dostupné na URL: <http://www.ksi.mff.cuni.cz/~mlynkova/Y36XML/slajdy/Y36XML_relXML.pdf>.

[15] DTD Tutorial [online]. 1999-2009 [cit. 2009-09-20]. Dostupné na URL: <http://www.w3schools.com/dtd/default.asp>.

[16] Informatika [online]. Dostupné na URL: <http://cs.wikipedia.org/wiki/Informatika >.

[17] Extensible Markup Language [online]. [cit. 2010-04-28]. Dostupné na URL: <http://cs.wikipedia.org/wiki/XML >.

[18] HERNYCH, R. Transformace a perzistence XML v relační databázi, diplomová práce, Brno, FIT VUT v Brně, 2009

[19] SEKO, M. Interaktívna vizualizácia XML, bakalárska práca, Brno, FIT VUT v Brně, 2009

[20] AMER-YAHIA, S., FERNANDEZ, M. Overview of Existing XML Storage Techniques. AT&T Labs, Cambridge, UK., 2001

[21] BOURRET, R.. XML and Databases [online]. 2000-2005 [cit. 2010-05-08]. Dostupné na URL: <http://www.rpbourret.com/xml/XMLAndDatabases.htm>.

[22] The XML:DB Initiative: Frequently Asked Questions About XML:DB. 2000-2003, [Online]. 2000-2003 [cit. 2010-05-08]. Dostupné na URL: <http://xmldb-org.sourceforge.net/faqs.html#faq-1>.

[23] SILBERSCHATZ, A., KORTH, H., SUNDERSHAN, S. Database System Concepts. NY: McGraw-Hill Higher Education, 2001. 1043 s. ISBN 0-07-228363-7.

[24] BUKATOVIČ, M. Srovnání dostupných implementací XML databází, bakalářská práce, Brno, Fakulta Informatiky Masarykova univerzita v Brně, 2010

[25] BOURRET, R. XML Database Products: Native XML Databases [online]. 2000-2010 [cit. 2010-05-09]. Dostupné na URL: <http://www.rpbourret.com/xml/ProdsNative.htm>.

[26] Oracle Berkeley DB XML [online]. [cit. 2010-05-09]. Dostupné na URL: <http://www.oracle.com/technology/products/berkeley-db/xml/index.html>.

[27] XML Database Products: XML-Enabled Databases [online]. [cit. 2010-05-09]. Dostupný na URL: <http://www.rpbourret.com/xml/ProdsXMLEnabled.htm>

[28] About Sedna [Online]. [cit. 2010-05-08]. Dostupné na URL: <http://modis.ispras.ru/sedna/index.html>.

[29] PARDEDE, E., RAHAYU, W. SQL/XML Hierarchical Query Performance Analysis in an XML-Enabled Database System, Melbourne, Australia

[30] CHMELAR, P., HERNYCH, R. a KUBICEK, D. Interactive Visualization of Data-Oriented XML Documents. Advances in Computer and Information Sciences and Engineering. Springer Netherlands, 2008. s. 390-393.

[31] DVOŘÁK, M. Návrhové vzory (design patterns) [online]. 2003. [cit. 2010-05-06]. Dostupné na URL: <http://objekty.vse.cz/Objekty/Vzory>.

[32] VAN DER VLIST, Eric. Relax NG. Sebastopol : O'Reilly & Associates, 2004. 473 s. Dostupné z URL: <http://books.xmlschemata.org/relaxng/page2.html>. ISBN 0596004214.

[33] Trang, [online]. [cit. 2010-05-05]. Dostupné z URL: < http://www.thaiopensource.com/relaxng/trang.html >.

[34] XSL-FO Tutorial, [online cit. 2010-05-06]. Dostupné z URL<http://www.w3schools.com/xslfo/default.asp>

[35] KOSEK, Jiří. XSLT v příkladech [online]. 2004 [cit. 2010-05-14]. Dostupné z URL: <http://www.kosek.cz/xml/xslt/>.

[36] BaseX, Dostupné z URL: <http://basex.org/>.

[37] MICHELS, J. Data Management and Interchange - SQL Standard and XML Functionality, Dostupné z URL: <http://www.wiscorp.com/H2-2005-197-SC32N1293-WG3_Presentation_for_SC32_20050418.pdf>.

[38] PIWKO, Karel, et al. NAXD: Native XML Interface for a Relational Database. In XML Prague Conference Proceedings, s. 307-316. Charles University in Prague, 2010. Dostupné na URL: <http://www.xmlprague.cz/2010/files/XMLPrague_2010_Proceedings.pdf>. ISBN 978-80-7378-115-6.

[39] NATU, S., MENDONCA, J. Digital Asset Management Using A Native XML Database Implementation. Proceedings of the 4th conference on Information technology curriculum, s. 237–241, New York, NY, USA, 2003.

[40] MICHAL, F. Interaktívna vizualizácia XML, bakalárska práca, Brno, FIT VUT v Brně, 2008.

Přílohy

A Seznam použitého software

B Obsah přiloženého CD

C Uživatelská příručka k XML Admin

D Naměřené hodnoty z testů

Příloha A

Seznam použitých zkratek a symbolů

API - Application Programming Interface

BLOB - Binary Large OBject

CDATA - Character Data

CLOB - Character Large OBject

CSS - Cascading Style Sheets

DOM - Document Object Model

DTD - Document Type De_nition

FLWOR - For, Let, Where, Order By, and Return v XQuery

HTML - HyperText Markup Language

RNG - Relax Next Generation

RNC - Relax Next Generation Compact

SQL - Structured Query Language

SGML - Standard Generalised Markup Language

URI - Uniform Resource Identifier

W3C - World Wide Web Consortium

XML - eXtensible Markup Language

XML:DB API - XML Database API

XPath - XML Path Language

XQuery - XML Query Language

XSD - XML Schema Defnition

XSL - eXtensible Stylesheet Language

XSLT - eXtensible Stylesheet Language Transformations

Příloha B

Obsah přiloženého CD

CD\db\Perizistence XML v relační databázi.pdf	- technická zpráva diplomové práce ve formátu PDF
CD\db\Perizistence XML v relační databázi.docx	- zdrojový tvar technické zprávy
CD\db\Desky DP Martin Boháč.pdf	- desky k technické zprávě ve formátu PDF
CD\XMLAdmin	- adresář programu XMLAdmin
CD\XMLAdmin\doc	- dokumentace k programu
CD\XMLAdmin\src	
CD\XMLAdmin\input	- vstupní data pro testování
CD\XMLAdmin\scripts	- skripty použité pro tvorbu datových setů při testování
CD\XMLAdmin\dist	- zkompilovaná verze programu
CD\XMLAdmin\dist\lib	- potřebné knihovny
CD\install\	- přiložen instalační soubor Java Development Kit 6 Update 20 a PostreSQL 8.4.4
README	- informace o aplikaci
INSTALL	- instalační návod
LICENCE	- licence programu XML Admin

Příloha C

Uživatelská příručka k XML Admin

O autorovi

autor: Martin Boháč

e-mail: xbohac06@stud.fit.vutbr.cz

Diplomová práce

O XML Admin

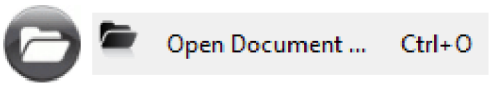
Aplikace XML Admin je databázový klient komunikující s databází pomocí aplikačního rozhraní XML:DB. Program implementuje následující funkce:

- Editace XML dokumentů.
- Editace XML schémat.
- Vizualizace XML pomocí zlepšeného Float Visualizéru.
- Vizualizace XML do tabulek.
- Generování schémat XML dokumentů (XML Schema, DTD, RNG a RNC).
- Ukládání a mapování XML dokumentů do databází pomocí xDB.
- Hromadné ukládání XML dokumentů nebo adresářů s XML dokumenty.

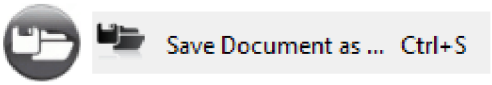
Ovládací prvky

Menu: XML dokument

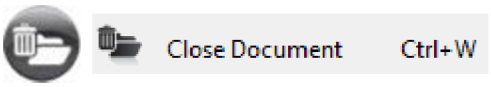
Část ovládacího menu pro XML dokument zůstala vizuálně stejná jako v programu SPARK. Funkce které zpřístupňuje se mírně liší z hlediska vizualizace.



Otevírá zvolený XML dokument. Po otevření se provede vizualizace ve formě stromové struktury dokumentu ve dvou různých zobrazení. Také se automaticky provede tzv. Float vizualizace (viz. obrázek 6.4).



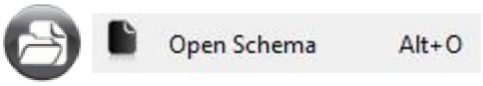
Uloží otevřený XML dokument.



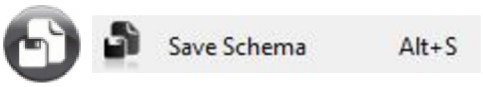
Zavře otevřený XML dokument. Ponechá však otevřené/vygenerované schéma dokument.

Menu: XML schéma

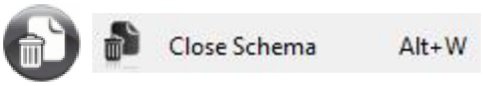
Část ovládacího menu pro XML schéma zůstala také vizuálně stejná jako v programu SPARK. V menu přibyla položka pro nastavení formátu XML schéma.



Otevře zvolené schéma dokumentu.



Uloží otevřené schéma dokumentu.



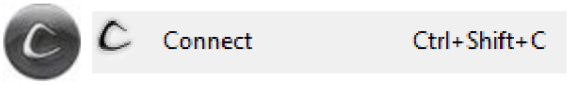
Zavře otevřené schéma. Ponechá však dokument XML otevřený.

Generuje schéma dokumentu podle otevřeného XML dokumentu. Zároveň se provede vizualizace ve formě tabulek.

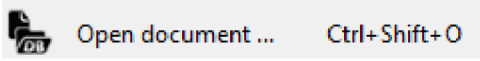
Umožňuje nastavit formát XML schématu.

Menu: XDB databáze

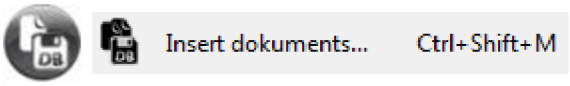
Ovládací prvky pro práci s databází.



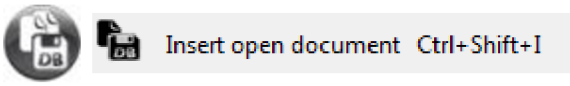
Odkáže na dialog pro připojení klienta ke kolekci v databázi.



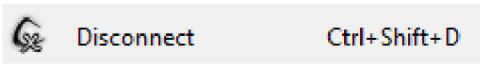
Otevře dialog pro načtení dokumentu XML z databáze.



Uloží otevřený XML dokument do zvolené kolekce v databázi, ke které je klient připojen.



Hromadné uložení XML dokumentů do zvolené kolekce v databázi. Umožňuje výběr více dokumentů nebo celé složky.



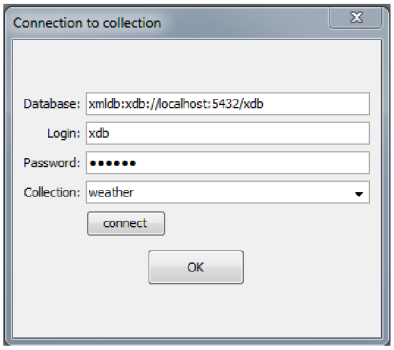
Odpojení od databáze.

Zavřít vše

Zavře otevřený dokument spolu s otevřeným nebo vygenerovaným XML schématem.

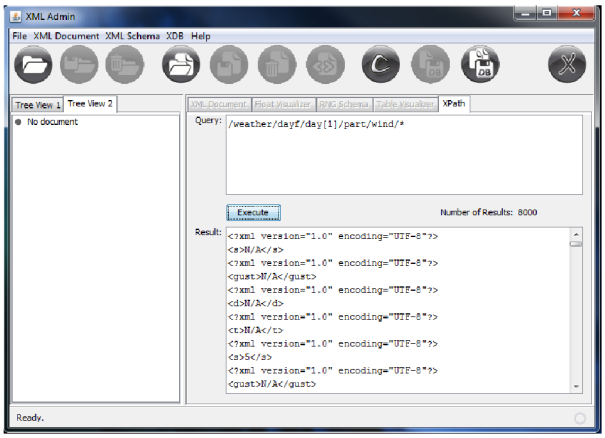
Ukázka aplikace

Připojení k databázi a dotazování pomocí jazyka XPath



Pokud se chceme připojit k databázi, po stisku tlačítka connect se nám objeví dialog, ve kterém lze zadat všechny potřebné údaje jako je url databáze, uživatelské jméno a heslo pro přístup do databáze a jméno kolekce se kterou chceme pracovat. Pokud neznáme jméno kolekce, která již existuje v databázi a se kterou chceme pracovat, necháme políčko se jménem kolekce prázdné a připojíme se. Program automaticky zvolí pro připojení kořenové schéma, kde jsou uloženy informace o všech uložených kolekcích a načte je do rozbalovacího menu. Poté lze zvolit požadovanou kolekci výběrem položky v rozbalovacím menu.

Po připojení do databáze se lze v záložce XPath dotazovat nad kolekcí, ke které jsme připojeni. Pro dotazy slouží textové pole Query, kde lze pokládat dotazy v jazyce XPath. Po stisku tlačítka *execute* se provede dotaz a výsledky jsou průběžně vypisovány do textového pole Result. V pravém horním rohu jsme informování o počtu vrácených výsledků.



Příloha D

Naměřené výsledky testů

Tabulka 7.3: Výsledky testu dotazování 1 (1000 XML), časy jsou uvedené v milisekundách.

		xDB		eXist		XML Admin s xDB	
dotaz	počet výsledků	čas dotazu	čas celkem	čas dotazu	čas celkem	čas dotazu	čas celkem
Q1	1000	237	754	70	7822	101	1809
Q2	2000	161	3573	199	16081	143	5866
Q3	8000	91	4502	397	61358	54	13413
Q4	11000	91	9374	291	83013	38	25361
Q5	4000	88	1151	184	30577	26	5625
Q6	56	80	182	77	771	118	225
Q7	1000	95	3332	297	8765	67	4082
Q8	1000	83	3320	220	8754	36	4104
Q9	10000	91	9122	273	75748	34	22576
Q10	40000	99	14391	1110	308804	38	65330

Tabulka 7.4: Výsledky testu dotazování 2 (3000 XML), časy jsou uvedené v milisekundách.

		xDB		eXist		XML Admin s xDB	
dotaz	počet výsledků	čas dotazu	čas celkem	čas dotazu	čas celkem	čas dotazu	čas celkem
Q1	3000	171	1244	376	32515	86	6314
Q2	6000	169	9411	2370	53298	154	19004
Q3	24000	189	24000	2323	206215	81	48343
Q4	33000	152	28006	862	289268	116	91303
Q5	12000	141	2623	613	103529	79	22343
Q6	57	137	267	276	905	87	225
Q7	3000	183	9484	2203	28479	81	13812
Q8	3000	154	9244	622	27727	84	15100
Q9	30000	149	27042	823	258242	115	89667
Q10	120000	154	42485	4130	1069241	749	248721

Tabulka 7.5: Výsledky testu dotazování 2 (5000 XML), časy jsou uvedené v milisekundách.

		xDB		eXist		XML Admin s xDB	
dotaz	počet výsledků	čas dotazu	čas celkem	čas dotazu	čas celkem	čas dotazu	čas celkem
Q1	5000	270	1812	705	49433	331	8422
Q2	10000	403,5	16583	7154	99007	199	31338
Q3	40000	249	20631	7137	371870	125	76709
Q4	55000	236,5	45726	1367	504966	158	149450
Q5	20000	209	4033	952	183207	129	31248
Q6	94	208,5	388	471	2130	248	541
Q7	3711	207,5	13050	3845	38764	324	19543
Q8	3711	198	13173	896	36955	130	18543
Q9	50000	223,5	45839	1273	494750	143	142348
Q10	200000	270,5	73357	6174	1985399	181	414617