

MaloLuM

Table of contents

1 O diplomovej práci.....	2
1.1 Biblická konkordancia / Projekt Maiolum.....	2
1.2 History of Changes.....	2
1.3 Todo List.....	2
2 Softvérová #as#.....	3
2.1 MaIoLuM.....	3
2.2 Aktuálny stav.....	4
2.3 Analýza.....	4
2.4 Implementácia.....	10
3 Teoretická #as#.....	13
3.1 Teoretická #as#.....	13

1. O diplomovej práci

1.1. Biblická konkordancia / Projekt Maiolum

1.1.1. Vitajte

<i>Diplomant:</i>	Bc. Róbert Novotný
<i>Vedúci:</i>	RNDr. Stanislav Krajčí, PhD
<i>Seminár:</i>	Seminár z logiky informačných systémov
<i>Rok zadania:</i>	2003/2004

1.1.2. Odkazy

- [Rozpracovaná teoretická časť \[PDF\]](http://s.ics.upjs.sk/~novotnyr/dp/dp/dp.pdf) (<http://s.ics.upjs.sk/~novotnyr/dp/dp/dp.pdf>)
- [Prezentácia na SLI, 23. mája 2004](http://s.ics.upjs.sk/~novotnyr/dp/xquery-tax.pdf) (<http://s.ics.upjs.sk/~novotnyr/dp/xquery-tax.pdf>)
- [Prezentácia 12. mája 2004](http://s.ics.upjs.sk/~novotnyr/dp/dpi1b.pdf) (<http://s.ics.upjs.sk/~novotnyr/dp/dpi1b.pdf>)
- [Prezentácia 6. októbra 2004](http://s.ics.upjs.sk/~novotnyr/dp/dpi1b-2.0.pdf) (<http://s.ics.upjs.sk/~novotnyr/dp/dpi1b-2.0.pdf>)
- Prototyp projektu [Maiolum](http://158.197.31.5/maiolum) (<http://158.197.31.5/maiolum>)

1.2. History of Changes

[RSS](#)

1.2.1. Version 0.1 (unreleased)

- Initial Import (JB)

1.3. Todo List

1.3.1. high

- **[docs]** Customize this template project with your project's details. This TODO list is generated from 'status.xml'. # JB
- **[docs]** Add lots of content. XML content goes in `src/documentation/content/xdocs`, or wherever the `${project.xdocs-dir}` property (set in `forrest.properties`) points. # JB
- **[feedback]** Mail forrest-dev@xml.apache.org with feedback. # JB

2. Softvérová #as#

2.1. MaIoLuM

2.1.1. Vitajte

Projekt MaIoLuM má za cie# vytvori# nástroj umož#ujúci využíva# biblickú konkordanciu, t. j. variant full-textového vyh#adávaní v Svätom Písme s využitím morfológie a synonymie vo vyh#adávacích termoch.

2.1.2. Technológie

Za použitú platformu bola zvolená [J2EE](http://java.sun.com/j2ee/index.jsp) (<http://java.sun.com/j2ee/index.jsp>) , presnejšie sú#asti Java + XML + Servlets/JSP. Aplika#ná logika by mala by# nezávislá od „prezenta#nej“ vrstvy, t. j. užívate#ského rozhrania. Základnou úlohou je vytvori# užívate#ské rozhranie zabezpe#ované internetovým prehliada#om a nasadi# aplika#nú logiku na serveri podporujúcom Servlets/JSP. V budúcnosti sa nevylu#uje vytvorenie alternatívnych užívate#ských rozhraní.

MaIoLuM je postavený na nasledovných technológiách:

XML

Vstupné dátové súbory biblie sú vo formáte XML.

DOM

Na#ítané biblické dokumenty sú v pamäti udržiavané v reprezentácii XML zvanej Document Object Model (DOM).

XPath

Reprezentácia DOM tvorí dátový model pre adresáciu jednotlivých elementov dát XML pomocou jazyka XPath. Použitím XPath sa z dokumentov získavajú jednotlivé verše, knihy a #alšie elementy.

XSLT

Elementy získané ako výsledok transformácie možno chápa# ako fragmenty XML dokumentov. Pre použitie v prezenta#nej vrstve je potrebné ich vhodne transformova# na cie#ové formáty (typicky HTML pre internetový prehliada# klienta).

Lucene -- Full-textové vyh#adávanie

Jednou z primárnych úloh je vyh#adávanie textu v biblii. Full-textové vyh#adávanie je realizované pomocou vyh#adávacieho "motora" [Jakarta Lucene](http://jakarta.apache.org/lucene) (<http://jakarta.apache.org/lucene>) .

Servlets & JSP

Základné užívateľské rozhranie bude tvorené internetovým prehliadačom. Vytváranie HTML stránok bude realizované pomocou servletov a JSP vo vhodnom kontajneri, typicky [Apache Tomcat](http://jakarta.apache.org/tomcat) (<http://jakarta.apache.org/tomcat>) .

2.1.3. Očakávaný prínos

- Konvertovať existujúce biblické texty z rôznych nekonzistentných a neštruktúrovaných formátov do XML, primárne texty v slovenčine.
- Vhodne prepojiť existujúce technológie na vytvorenie užívateľského rozhrania podporujúceho užívateľské požiadavky.

2.2. Aktuálny stav

2.2.1. Aktuálny stav softvérovej časti

Analýza	95 %
Spracovanie vstupných súborov	80 %
Implementácia	80 %
Dokumentácia	50 %

Analýza

Analýza jadra je dokončená. Ostáva dokončiť analýzu podmodulov (konkordancia, tezaurus).

Spracovanie vstupných súborov

Prebieha spracovanie vstupných súborov Svätého písma do formátu XML. Spracovaný je Nový Zákon a tri štvrtiny Starého Zákona v slovenčine.

Implementácia

Implementácia jadra postaveného na základe natívnej XML databázy eXist je dokončená. V súčasnosti prebieha implementácia konkordancie a tezauru.

Dokumentácia

Dokumentácia je vytváraná priebežne s implementáciou projektu. Dve najdôležitejšie časti sú dokumentácia tried JavaDoc (vytváraná priebežne po implementovaní a testovaní triedy) a projektová dokumentácia reprezentovaná týmito stránkami Forrestu.

2.3. Analýza

2.3.1. Uživate#ské požiadavky

2.3.1.1. Uživate#ské požiadavky

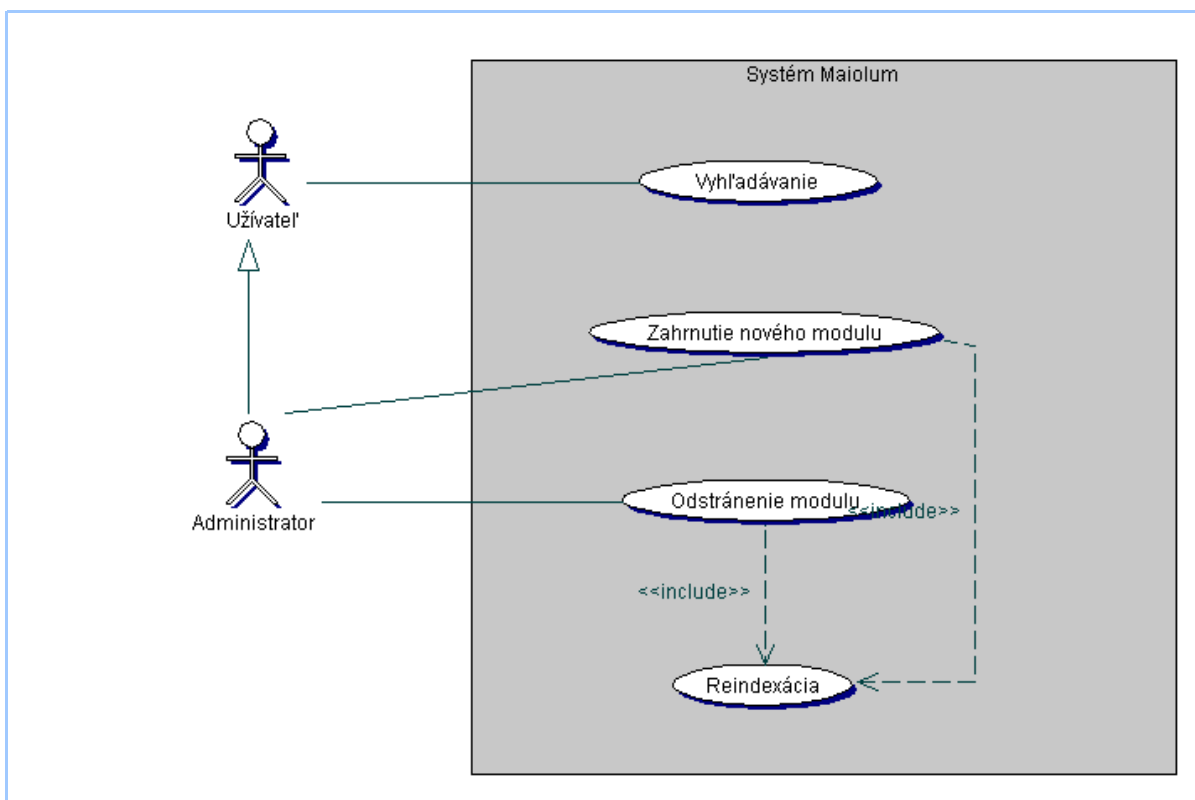
- vytvori# systém na prehliadanie a vyh#adávanie v Svätom Písme
- dodané dátové súbory sú vo formáte XML
- zavies# podporu pre viacjazy#nos#. Uživate# chce vyh#adáva# vo viacerých jazykových mutáciách.
- vyh#adávanie obohati# podporou morfológie (dodanej z externého prostredia)
- prehliadanie realizova# cez webové rozhranie

2.3.2. Analýza a návrh

2.3.2.1. Use-case -- prípady použitia

Všeobecný prípad použitia

Základný prípad použitia



Roly užívateľov

užívateľ

štandardný -- „bežný“ užívateľ systému. Jeho primárnym cieľom pri pohybe v systéme je vyhľadávať v dokumentoch podľa kľúčového slova a zobrazovať jednotlivé časti dokumentov (čo v sebe obsahuje aj navigáciu medzi nimi)

administrátor

Bežný užívateľ systému s administrátorskými právami. Administrátor má jediný právo pridávať jednotlivé moduly s dokumentami do systému, odoberá ich a iniciovať ich reindexáciu.

Popis procesov v use-case diagrame

Proces vyhľadávania

Užívateľ zadá vyhľadávací výraz. Systém vráti zoznam dokumentov, ktoré spĺňajú požiadavky užívateľa.

Reindexácia

Užívateľ spustí znovuvytvorenie indexu fulltextového vyhľadávania.

Zahrnutie modulu

Užívateľ zadá názov súboru XML pre modul, ktorý má byť zahrnutý do systému. V prípade, že súbor neexistuje alebo nemôže byť korektne načítaný, systém vypíše príslušnú správu. Ak spracovanie súboru prebehlo úspešne, systém upozorní užívateľa a vykoná reindexáciu daného modulu.

Odstránenie modulu

Užívateľ identifikuje modul, ktorý má byť odstránený zo systému. Systém odoberie príslušný modul a vykoná reindexáciu daného modulu.

2.3.3. Návrh -- koncepty

2.3.3.1. Koncepty v návrhu

Prvotný návrh vychádza z niekoľkých základných konceptov:

- *biblický fragment*. Biblické texty majú základnú pevnú štruktúru (danú „zhora“), ktorá pozostáva z niekoľkých pevných štruktúrnych komponentov. Sú nimi po poradií Zákony (Starý a Nový), knihy, kapitoly a verše. V tejto štruktúre platí, že každý z uvedených komponentov v sebe obsahuje ďalšie podkomponenty -- teda Zákon obsahuje knihy, ktoré obsahujú kapitoly atď. Navyše, každý z komponentov má svoj prirodzený jednoznačný identifikátor (napr. *Mt1, 1* je identifikátorom prvého verša prvej kapitoly knihy Evanjelium podľa Matúša).

Navyše v tejto štruktúre je pevne dané usporiadanie fragmentov a to dvomi spôsobmi: usporiadanie „predchodca -- nasledovník“ (je zrejmé, že *Mt1, 2* je identifikátorom fragmentu, ktorý nasleduje po fragmente s identifikátorom *Mt1, 1*) a usporiadanie „rodič -- potomok“ (je zrejmé, že *Mt1* je identifikátorom fragmentu, ktorý je rodičom fragmentu *Mt1, 1*). Z toho vypláva prirodzená požiadavka, aby každý biblický fragment „vedel“ o svojich nasledovníkoch, predchodcoch a prípadnom rodičovi.

Uvedenú pevne danú štruktúru, ktorá v sebe zahŕňa jednoznačné identifikátory s usporiadaním, môžeme s výhodou využiť pri návrhu.

Základné požiadavky

- biblický fragment obsahuje textové biblické dáta
- biblický fragment má jednoznačný identifikátor

- biblický fragment musí mať vedomosť o svojich predchodcoch, nasledovníkoch a rodičoch
- biblický fragment musí mať vedomosť o svojich jazykových metadátach (pozri koncept *edícia*)
- *identifikátor biblického fragmentu* je koncept navrhnutý v predošlom prípade. Štandardný zápis identifikátorov nám s výhodou umožňuje extrahovať niektoré informácie týkajúce sa usporiadania fragmentov (z daného zápisu, napr. *Mt1*, vieme extrahovať rodiča daného identifikátora, ak existuje -- napr. rodičom fragmentu s daným identifikátorom je fragment s identifikátorom *Mt*).

Predovšetkým treba brať do úvahy varianty zápisu daného identifikátora -- identifikátor *Gn1*, *1* môže mať ekvivalentné zápisy *Gn1:1*, *Gen1.1* a podobne.

Základné požiadavky

- identifikátor musí viesť spracovávať rôzne užívateľove zápisy identifikácie fragmentu
- *biblický modul*. #asto sa stáva, že jednotlivé biblické texty, prípadne ich jazykové mutácie, sú reprezentované samostatnými #asami nezávislými na ich prirodzenej štruktúre. V našom bežnom prípade sú jednotlivé texty reprezentované istým počtom XML súborov, pričom každý XML súbor obsahuje nejakú #as biblických textov. Aby sme zjednodušili administráciu týchto súborov, zavádzame koncept biblického modulu. V prípade XML súborov môže napríklad každému XML súboru prislúchať jeden modul.

Na biblické moduly sa dá istým spôsobom dívať ako na špeciálny prípad biblických fragmentov, v tomto prípade je však nutné „doplniť“ pre každý modul jednoznačné identifikátory, aby sa zachovala pevná štruktúra fragmentov.

Moduly sú voliteľnou, ale #asto používanou sú#asou nášho návrhu. V prípade implementácie, ktorá by spracovávala biblické texty uložené v rela#nej databáze nemusí byť tento koncept pri implementácii použitý.

Základné požiadavky

- biblický modul musí mať vlastný jednoznačný identifikátor
- biblický modul musí viesť, ktoré fragmenty v sebe obsahuje
- *metadáta jazykovej mutácie fragmentu -- edícia*. Spomenuli sme, že každý biblický fragment má svoj jednoznačný identifikátor. Problém nastáva v prípade viacerých jazykových verzií toho istého modulu. Určite nastane situácia, že dva biblické fragmenty majúce rôzne jazykové mutácie majú ten istý identifikátor (odhliadnuc od variantov identifikátora). Používame preto koncept edície predstavujúci metadáta o danom fragmente -- napr. jazyk danej mutácie, vydavateľ, rok vydania a podobne.

V systéme musí platiť, že dva biblické fragmenty s rovnakým identifikátorom musia mať rôzne jazykové metadáta (aby sme zabránili nejednoznačnostiam).

Základné požiadavky

- edícia musí obsahovať okrem jazykových dát metadáta o danom fragmente (vydavateľ, rok vydania atď.)

2.3.4. Návrh -- abstraktné triedy a interfaces

2.3.4.1. Ďalšie koncepty v návrhu.

Dátový zdroj (datasource)

Jedným z cieľov návrhu je možnosť pracovať nad biblickými textami, ktoré pochádzajú z rôznych dátových zdrojov. V našom prípade je dátovým zdrojom sada XML súborov, ale môže nastať situácia, keď sú texty načítavané z relačnej databázy, z webovej služby a podobne. Zavedieme preto koncept *dátového zdroja (DataSource)*.

Dátový zdroj predstavuje rozhranie k „skladu dát“ a prostriedok na získavanie a vytváranie jednotlivých biblických fragmentov. Funkcie sú nasledovné:

- prístupuje a získava biblické textové dáta (napr. načítaním XML súborov, dopytovaním sa databázy)
- poskytuje metódy na získavanie biblické fragmenty.
- vyhodnocuje informácie týkajúce sa usporiadania -- vie získavať predchodcu, nasledovníka, či rodiča daného fragmentu.

Továrňa na edície

Pri práci s edíciami predpokladáme, že ich množina je vopred daná a pre danú množinu biblických fragmentov nemenná. Množinu edícií však možno takisto získať z rôznych zdrojov. Preto zavedieme koncept *továrne na edície*. Jej úlohy sú nasledovné:

- vytvára objekty edícií
- množinu dostupných edícií vie získať z rôznych zdrojov (XML súbor, databáza)

2.3.4.2. Abstraktné triedy a interfaces

Uvedené koncepty sa pokúsime previesť na jednotlivé triedy, s ktorými budeme pracovať v implementácii.

- *biblický fragment* je reprezentovaný interfaceom *BibleFragment*

- *identifikátoru biblického fragmentu* zodpovedá abstraktná trieda *FID*
- *biblický modul* je interface *BibleModule* odvodený od interfaceu *BibleFragment*.
- *edícii* zodpovedá abstraktná trieda *Edition*
- *dátovému zdroju* zodpovedá interface *Edition*

Diagram základných tried

2.4. Implementácia

2.4.1. Implementácia softvéru

2.4.1.1. Technológie

Za použitú platformu na implementáciu softvéru bola zvolená [J2EE](http://java.sun.com/j2ee/index.jsp) (<http://java.sun.com/j2ee/index.jsp>) a ďalšie open-source projekty:

XML

Vstupné dátové súbory biblie sú vo formáte XML.

eXist

natívna XML databáza podporujúca XPath a XQuery slúži ako úložisko biblických textov.

Lucene -- full-textové vyh#adávanie

[Jakarta Lucene zabezpe#uje full-textové indexovanie a vyh#adávanie v dokumentoch. Jedným cie#om je rozšíri# Lucene pre podporu indexovania slovenských textov s použitím synonym. \(http://jakarta.apache.org/lucene\) .](http://jakarta.apache.org/lucene/)

XSLT

XML dokumenty sú pomocou XSLT transformované do podoby vhodnej pre klienta (implicitne HTML).

Jakarta Tomcat -- servlety, Jakarta Velocity -- prezenta#ná vrstva

Implicitné uživate#ské rozhranie je realizované pomocou webového prehliada#a. Samotná aplikácia je spustená na vzdialenom serveri -- konkrétne v servletovom kontajneri Jakarta Tomcat.

Vzh#ad uživate#ského rozhrania je realizovaný šablónovým motorom Jakarta Velocity.

2.4.1.2. Popis komponentov

Globálny náhľad na jednotlivé komponenty systému možno získať pomocou nasledovného obrázka:

2.4.2. Implementácia rozhraní a abstraktných tried

2.4.2.1. Implementácia maiolum.nxdb

Uvedený návrh je dostatočne flexibilný na to, aby bolo možné získať biblické fragmenty z rôznych zdrojov rôznymi spôsobmi. Teraz popíšeme jeden z možných spôsobov implementácie abstraktných tried a rozhraní -- spôsob, pri ktorom sa biblické fragmenty získavajú z natívnej XML databázy.

Dátový zdroj

Implementácia dátového zdroja maiolum.nxdb.ExistDataSource
--

V prípade tejto implementácie sa využíva natívna XML databáza eXist, v ktorej sú uložené biblické texty vo formáte XML. Dátový zdroj sa dopytuje XML databázy pomocou XQuery na XML dokumenty v nej uložené. Vrátané XML dáta použije pri vytváraní inštancií biblických fragmentov (implementácia maiolum.plainxml.PXBibleFragment)

Biblické fragmenty a moduly

Implementácia biblických fragmentov a modulov

Identifikátory fragmentov a poradie

Na určovanie usporiadania fragmentov sa využíva špeciálna trieda FIDResolver. Pri ukladaní XML dokumentu do databázy eXist sa vytvorí zvláštny index, v ktorom je uložené poradie identifikátorov v danom dokumente. FIDResolver potom poskytuje na základe tohto indexu metódy na získanie predchodcu, #i nasledovníka daného biblického fragmentu. V prípade získavania rodiča daného fragmentu tento index nie je potrebný -- keďže identifikátor rodiča možno ľahko získať zo samotného identifikátora fragmentu.

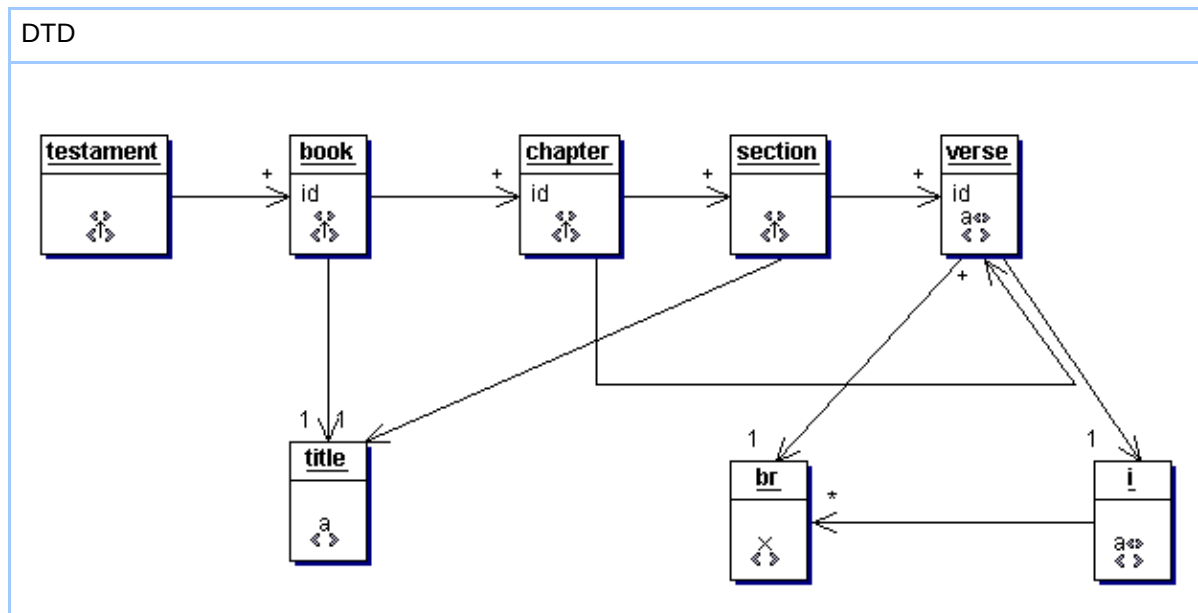
2.4.3. Analýza a návrh

2.4.3.1. Štruktúra vstupných súborov

Úvod

Vstupné súbory sú, prirodzene, vo formáte XML. Ich štruktúra „prirodzene“ vyplýva zo štruktúry Svätého písma. Jednotlivé súbory by mali prislúchať jednotlivým biblickým modulom, ktoré sú spravované Správcom modulov. (Typicky je tvorí modul XML obsahujúce Nový Zákon).

Popis štruktúry vo forme Document Type Definition (DTD)



Štruktúra je prirodzená: <testament> je všeobecný korešný element. Ten obsahuje jednu, #i viac kníh. Každá kniha má identifikátor, nadpis a obsahuje nenulový počet kapitol. Kapitola má tiež jednoznašný identifikátor, nadpis a obsahuje verše, niektoré verše sú zoskupené do sekcií. Sekcia zoskupujúca verše má nadpis. Verš -- rovnako ako kapitoly, #i knihy -- obsahuje identifikátor. Niektoré #asti veršov môžu byť zvýraznené (element <i>) a môže byť naznašené ukončenie riadka (elementom
).

Teda, výsledný návrh štruktúry DTD je nasledovný:

```

<!ELEMENT testament (book+)>
<!ELEMENT title (#PCDATA)>

<!ELEMENT book (title, chapter+)>
<!ELEMENT chapter (section+, verse+)>
<!ELEMENT section (title, verse+)>
<!ELEMENT verse (#PCDATA | br | i)*>

<!ELEMENT br EMPTY>
<!ELEMENT i (#PCDATA | br)*>
  
```

```
<!ATTLIST book id ID #REQUIRED>  
<!ATTLIST verse id ID #REQUIRED>  
<!ATTLIST chapter id ID #REQUIRED>
```

Identifikácia veršov je rovnako prevzatá z „prirodzenej“ štruktúry. Knihy sú identifikované slovenskými skratkami, napr. Evanjeliu pod# Matúša prislúcha identifikátor Mt. Kapitoly obsahujú identifikátor nadradenej knihy konkatenovaný z poradovým #íslom kapitoly: prvá kapitola Zjavenia svätého Jána má identifikátor Zj v1. Napokon, verše sú jednozna#ne ozna#ené identifikátorom kapitoly a poradovým #íslom verša, #iže štrnásty verš druhej kapitoly Jakubovho listu má ozna#enie Jk2_14.

Zdanlivá redundancia v ozna#ovaní fragmentov (verše by napr. mohli ma# len poradové #íslo) má výhodu v zna#nom zrýchlení dopytovania, pretože sa zníži zložitost# dopytov XPath. V prípade, že má element XML dokumentu atribút ID, možno ho priamo využi# v dopyte XPath pomocou funkcie id, pre ktorú sú #asto dopyty optimalizované.

Problémy pri dátových súboroch

Pri prebiehajúcej konverzii súborov je nutné vyrieši# nieko#ko problémov

- chýbajúce poznámky k Svätému Písmu (neexistencia elektronickej podoby)
- nekonzistentnos# pomocných nadpisov
- vyriešenie odkazov medzi fragmentami

Jediným spôsobom získania chýbajúcich poznámok k Svätému Písmu je ich vytvorenie, ke#že neexistuje ich elektronicná podoba.

Nekonzistentnos# pomocných nadpisov je asi najvä#ším problémom. Pomocné nadpisy plnia vysvet#ujúcu a pomocnú úlohu, podobnú ako poznámky. Štruktúra nadpisov sa líši medzi jazykovými mutáciami, dokonca aj medzi rôznymi vydaniaми tej istej jazykovej mutácie. Nadpisy sú nekonzistentné, pretože narušujú pravidelnú pološtruktúrovanos# dát, pretože nadpis (a teda koniec istej sekcie) sa môže vyskytova# na rôznych miestach, dokonca uprostred verša (pri#om by bolo ve#mi vhodné predpoklada#, že verš je elementárna štruktúrna jednotka textu Sv. Písma).

Kvôli preh#adnosti a vylepšeniu konkordancie sa o#akáva vytvorenie odkazov medzi fragmentami biblických textov. Je nutné zvoli# vhodný formát a reprezentáciu týchto odkazov (XLink?).

3. Teoretická #as#

3.1. Teoretická #as#

3.1.1. TAX: Stromová algebra pre XML

H. V. Jagadish et al. -- TAX: Tree Algebra in XML, In: *Proceedings of the 8th Biennial Workshop on Data Bases and Programming Languages (DBPL'01)*, Frascati, Rome, September 8-10, 2001.

Podrobnosti sú v druhej časti prezentácie na Seminári z logiky informačných systémov, 23. marec 2004.

3.1.2. Formálny dátový model a algebra pre XML

D. Beech, A. Malhotra, M. Rys -- A Formal Data Model and Algebra for XML.

Autori definujú dátový model pre XML v podobe grafovej reprezentácie XML stromu. Algebra definuje operátory π (slúžiaci na prechádzanie hrán spĺňajúcich dané podmienky), σ (reprezentujúci analógiu selekcie v relačnej algebre), operátory Kleeneho hviezdičky a mapovania a operátory vytvorenia hrany a vrchola. Žiaľ, nedokazujú výrazovú silu tejto algebry (minimálne ekvivalenciu so štandardnou relačnou algebrou/SQL).