VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DEMONŠTRÁCIA PODPORY PRE PRÁCU S XML V MODERNÝCH DATABÁZOVÝCH SYSTÉMOCH

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MATEJ DVORŠTIAK
AUTHOR

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DEMONSTRACE PODPORY PRO PRÁCI S XML V MODERNÍCH DATABÁZOVÝCH SYSTÉMECH

DEMONSTRATION OF SUPPORT FOR XML IN DATABASE SYSTEMS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MATEJ DVORŠTIAK
AUTHOR

VEDOUCÍ PRÁCE DOC. ING. JAROSLAV ZENDULKA, CSC.
SUPERVISOR

BRNO 2008

Abstrakt

Tato bakalářská práce je zaměřena na XML technologie a XML podporu u databázových systémů. Konkrétně u systému Oracle. Jeho XML podpora je podrobně rozebrána v práci. Detailní popis je hlavně zaměřen na architekturu Oracle XML DB, datový typ XMLType, na práci XML schématem, možnosti ukládání a dotazování XML dat v databázi Oracle. Praktická část popisuje aplikaci, která demonstruje XML podporu pro výukové účely. Teoretická a praktická část je doplněna příklady a obrázky pro lepší ilustraci problematiky.

Abstract

This thesis is concerned with XML technologies and XML support of database systems. Especialy with Oracle database. Support is descripted in this work. Desription is aimed to an Oracle XML DB architecture, XML data type, work with XML schema, storing and quering options of XML data in Oracle database. Practical part describes aplication, demonstrates XML support for learning proposes. Theoretical and practical parts are appended with examples and pictures for better understanding of described problematic.

Klíčová slova

XML, XML podpora, značkovací jazyk, DTD, XML schéma, dokumentové XML dokumenty, datové XML dokumenty, XML jmenovité prostory, XSL, XSLT, XSL-FO, DOM, SAX, XML analyzátory, nativní datový typ, XPath, XQuery, XLink, XPointer, Microsoft SQL Server, Adaptive Server Enterprise, Oracle, Oracle XML DB, Oracle XML Developer `s Kit, Oracle XML DB Repository, datový typ XMLType, Indexy, SQL / XML

Keywords

XML, XML support, markup language, DTD, XML schema, documental XML document, data XML document, XML Namespaces, XSL, XSLT, XSL-FO, DOM, SAX, XML parsers, native data type, XPath, XQuery, XLink, XPointer, Microsoft SQL Server, Adaptive Server Enterprise, Oracle, Oracle XML DB, XML Developers` Kit, Oracle XML DB Repository, XMLType data type, Indexes, SQL/XML

Citace

Dvorštiak Matej: Demonstrace podpory pro práci s XML v moderních databázových systémech, bakalářská práce, Brno, FIT VUT v Brne, 2009

Demonstrace podpory pro práci s XML v moderních databázových systémech

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Doc. Ing. Jaroslava Zendulky, CSc. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Matej Dvorštiak
10. května 2008

Poděkování

Rád bych poděkoval Doc. Ing. Jaroslavu Zendulkovi, CSc., vedoucímu semestrálnej a bakalářskej práce, za konzultace a cenné rady, které mi během tvorby práce poskytl a také za trpělivost, kterou se mnou mal. Dále děkuji rodičům a přátelům za podporu během studia.

© Matej Dvorštiak, 2009
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů..

Obsah

Obsah 1
1 Úvod 3
2 XML 4
2.1 Úvod do XML 4
2.2 História jazyka XML 4
2.3 Syntax jazyka XML 4
2.3.1 Základné syntaktické pravidlá jazyka XML 5
2.4 Menné priestory 6
2.5 DTD 7
2.6 XML schéma 7
2.7 XML analyzátory 8
2.7.1 DOM (objektový model dokumentu) 8
2.7.2 SAX (jednoduché API pre XML) 8
2.7.3 Hlavné rozdiely medzi DOM a SAX [1] 8
2.8 XPath 9
2.9 XQuery 9
2.10 XPointer 9
2.11 XLink 10
2.12 XSL 10
2.13 Typy XML dokumentov 11
3 Databázové systémy a XML 13
3.1 Microsoft SQL Server 13
3.1.1 XML podpora u servera Microsoft SQL Server 13
3.2 Adaptive Server Enterprise 17
3.2.1 Adaptive Server Enterprise a podpora XML 18
3.3 Oracle 20
3.3.1 Podpora XML u databázového systému Oracle 20
3.4 Záver k databázovým produktom 25
4 Oracle XML DB 26
4.1 Dátový typ XMLType 26
4.2 Modely XMLType úložiska 27
4.2.1 Štruktúrované XMLType úložisko 27
4.2.2 Neštruktúrované XMLType úložisko 28
4.2.3 Binárne XMLType úložisko 28
4.3 XML schéma 28
4.3.1 Registrácia XML schémy 28
4.3.2 Anotácia XML schémy 30
4.3.3 Mapovanie XML schémy do úložiska XML DB 31
4.4 Indexy 32
4.4.1 B-tree index 33
4.4.2 XMLIndex 33
4.4.3 Funkčne založené indexy 33
4.4.4 Textové indexy 33

4.5 DML operácie nad XML dátami 34
4.5.1 SQL príkazy s XMLType 34
4.5.2 DBMS_XMLSTORE 34
4.5.3 SQL funkcie 34
4.6 Dotazovanie XML dát 35
4.6.1 Zobrazovanie XML dát XMLType metódami 35
4.6.2 SQL funkcie pre dotazovanie 36
4.6.3 Ďalšie spôsoby ako dotazovať XML dáta 37
4.7 Generovanie XML dát z databázy 37
4.7.1 Generovanie XML funkciami štandardu SQL/XML 38
4.7.2 Generovanie XML pomocou DBURIType 38
4.7.3 Generovanie XML pomocou DBMS_XMLGEN 39
4.8 Ďalšie možnosti balíka Oracle XML DB 39
4.8.1 XSLT Transformácia 39
4.8.2 Pohľady (views) 40
4.8.3 Prepisovanie XQuery a XPath výrazov 40
5 Ukážková aplikácia 41
5.1 Desktopová aplikácia 41
5.1.1 Architektúra aplikácie 42
5.1.2 Pripojenie sa k databáze 43
5.1.3 Práca so skriptami 43
5.1.4 Prístup k nápovede 45
5.2 Ukážkové skripty 46
5.3 Nápoveda 48
5.4 Implementačné nástroje 48
5.5 Požiadavky aplikácie 49
6 Záver 50
Literatura 51
Príloha A – Obsah priloženého CD 52
Príloha B – XML dokument 53
Príloha C – XML schéma 54
Príloha D – XSL šablóna 56

1 Úvod

V dnešnej dobe komunikačných technológií sa kladie dôraz na dorozumievanie sa, a to nielen medzi ľuďmi, ale aj medzi aplikáciami.

U aplikácií tento problém z určitého pohľadu rieši XML jazyk, ktorý sa stal najpopulárnejším jazykom pre popisovanie a prenos dát. Do popredia ho vyzdvihla jeho jednoduchosť a platformová nezávislosť. Jazyk dopĺňa rodina XML technológií, ktorá jeho použitie rozširuje o kontrolu štruktúry, možnosti práce a zobrazenia. Tejto problematike sa venuje celá druhá kapitola, v ktorej je okrajovo spomenutá syntax jazyka XML. Kapitola sa ďalej venuje už spomenutej rodine XML technológií. Záver kapitoly je zameraný na jednotlivé typy XML dokumentov. Toto delenie zohráva dôležitú úlohu v rámci XML podpory databázových systémov.

Jedným z cieľov bakalárskej práce je oboznámenie sa s XML podporou u populárnych databázových produktov. Touto problematikou sa zaoberá tretia kapitola. V nej som rozobral XML podporu vybraných databázových systémov. Zameral som sa na dôležité prvky XML podpory a porovnal ich u jednotlivých produktov.

Na základe konzultácie s vedúcim práce som vybral jeden z troch produktov popísaných v tretej kapitole, ktorému sa podrobne venujem. Jedná sa o produkt Oracle, hlavne jeho časti, ktorá sa týka úložiska dátového typu XMLType. Presne touto problematikou sa zaoberá štvrtá kapitola. V nej sa venujem teórii ohľadom natívneho dátového typu XMLType a práce s ním. V kapitole ďalej popisujem aj kľúčovú úlohu XML podpory servera Oracle a tou je XML schéma. Okrem nej spomeniem aj modely uloženia XMLType, indexovanie nad XML dátami, ale aj DML operácie a dotazovanie XML dát.

Piata kapitola je venovaná praktickej časti bakalárskej práce. Popisujem v nej spôsob, akým bola riešená aplikácia, jej zameranie a funkčnosť, ale aj problémy, s ktorými som sa pri jej vytváraní stretol. Ďalej sú v kapitole zmienené technológie a požiadavky potrebné pre beh aplikácie.

2 XML

2.1 Úvod do XML

XML (Extensible Markup Language) je rozšíriteľný značkovací jazyk, určený pre značkovanie dokumentov. Je schválený konzorciom W3C (World Wide Web). Jazyk XML je meta-značkovací jazyk, stanovený pre textové dokumenty, kde predstavuje nositeľa uložených dát. Jazyk je definovaný množinou pravidiel, resp. syntaxou, ktorá sa používa pre označovanie dát. Jednoduchá, striktná a obecná syntax, spolu s platformovou nezávislosťou, vytvárajú z jazyka flexibilný a široko uplatniteľný nástroj. Nie je určený pre trvalé ukladanie dát, ale predovšetkým pre výmenu údajov medzi aplikáciami a na tvorbu dokumentov.

2.2 História jazyka XML

História jazyka XML siaha až do roku 1986, kedy bol vyvinutý obecný značkovací jazyk SGML (Standard Generalized Markup Language), a bol prijatý za štandard ISO. Nevýhodou tohto jazyka bola jeho zložitosť a ťažkopádnosť. Pre potreby webu bol potrebnejší jednoduchší jazyk s voľnejšou syntaxou ako SGML.

Za týmto účelom bol z jazyka SGML vyvinutý jazyk XML. Prvá verzia XML 1.0 bola konzorciom W3C prijatá za štandard 10. februára 1998. Počas písania bakalárskej práce, je najaktuálnejšia verzia jazyka XML 1.0 piate vydanie z 26. novembra 2008.

2.3 Syntax jazyka XML

XML dokument je textový dokument, ktorý sa skladá z kombinácie značiek a textového obsahu (tzv. znakové dáta). Pričom značky stanovujú štruktúru a v správne vytvorenom XML dokumente tiež popisujú sémantiku dokumentu. Jazyk XML určuje, akú syntax značky musia dodržiavať (napríklad ktoré značky platia, kde sa majú nachádzať, v akom tvare atď.). Značky v XML dokumente vyzerajú podobne ako v HTML dokumentoch, až na niektoré rozdiely. Hlavným rozdielom od jazyka HTML je, že XML má ľubovoľný počet značiek. Tie sa totiž vytvárajú užívateľmi.

XML dokument sa skladá z elementov. Element pozostáva z dvojice značiek (počiatočnej a koncovej značky) a textového obsahu (textový reťazec s bielymi znakmi). Elementy môžeme zanoriť do seba a tým vytvoriť stromovú štruktúru XML dokumentu. Vlastnosti jednotlivých elementov sa popisujú pomocou atribútov.

Popis elementu [1] :

<**meno_elementu atribút =** “hodnota atribútu“> <!--počiatočná značka elementu-->
obsah elementu <!--znakové dáta-->
</**meno_elementu**> <!--koncová značka elementu-->

```
<zákazník identifikačné_číslo = "152">Matej Dvorštiak</zákazník>
```

Príklad č. 1 Element XML dokumentu

2.3.1 Základné syntaktické pravidlá jazyka XML

Základné syntaktické pravidlá, ktoré musí spĺňať každý XML dokument :

- Každý element musí mať uzatváraciu značku (*tag*) *<element>obsah</element>*, okrem prázdneho elementu (element, ktorý nemá obsah), ten sa môže zapísať pomocou skrátenej notácie *</element>.*

- XML jazyk rozlišuje malé a veľké písmená (jazyk je *CASE SENSITIVE*).
  <**Zlý príklad**>obsah</**zlý príklad**>
  <**Dobrý príklad**>obsah</**Dobrý príklad**>

- Elementy sa nesmú prekrývať (musia byť korektne zanorené).
  <**zlý príklad**><**b**></**zlý príklad**></**b**>
  <**dobrý príklad**><**b**></**b**></**dobrý príklad**>

- XML dokument musí obsahovať koreňový element tzv. *ROOT ELEMENT.* Tento element je špecifický tým, že sa nachádza prvý v dokumente a nemá rodičov. Všetky ostatné elementy sú v ňom zanorené.

- Hodnota atribútu musí byť v úvodzovkách a to buď jednoduchých, alebo dvojitých.
  <element atribút = ‘hodnota atribútu‘>
  <element atribút = “hodnota atribútu“>

- Dokument môže byť doplnený o komentáre, ktoré sa uvádzajú nasledovným spôsobom :
  <!--komentar**-->**

Ďalším prvkom XML dokumentu je deklarácia dokumentu, ktorá nie je povinná. Je ale dobrým zvykom ju uvádzať a uvádza sa na začiatku dokumentu. Deklarácia sa skladá z nasledovných parametrov :

- parameter deklarácie *version* informuje o verzii XML dokumentu
- *encoding* informuje o použitej znakovej sade
- *standalone parameter* určuje či dokument pracuje sám alebo s nejakými inými súbormi (napríklad so XML schémou)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

Príklad č. 2 Deklarácia XML dokumentu

Všetky XML dokumenty splňujúce syntaktické pravidla označujeme ako *dobre vytvorené* (anglicky well formed).

```
<?xml version="1.0" encoding="UTF-8"?>
<zákazník ID="123"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <meno>Matej</meno>
     <priezvysko>Dvorštiak</priezvysko>
     <adresa>Kolejní 2</adresa>
     <telefónne_číslo>123456789</telefónne_číslo>
</zákazník>
```

Príklad č. 3 XML dokument

2.4 Menné priestory

Menné priestory (anglicky namespaces) patria tiež medzi štandardy XML. V XML dokumentoch sa používajú k dvom účelom [1] :

- *Rozlíšeniu* – rozlišujú elementy a atribúty, ktoré majú síce rovnaký názov, ale pochádzajú z rôznych XML dokumentov.
- *Združujú* – združujú všetky súvisiace elementy a atribúty z jednej aplikácie XML tak, aby s nimi software mohol ľahko pracovať.

Menné priestory sa implementujú tak, že pred každý názov elementu, či atribútu sa uvedie predpona – *prefix*. Každý prefix je asociovaný s určitou URI. Tým sa zaistí jednoznačnosť elementov a atribútov a nedôjde k duplicite názvov značiek.

```
xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

Príklad č. 4 Deklarácia menného priestoru

2.5 DTD

DTD (Document Type Definition) slovensky definícia typu dokumentu, je jazyk, ktorý stanovuje akú štruktúru XML, prípadne SGML, dokumenty musia mať. Tým sa obmedzí množina prípustných dokumentov spadajúcich do daného typu, alebo triedy DTD. Syntax DTD popisuje, ktoré elementy sa v dokumente môžu objaviť, s akým obsahom a atribútmi. Akceptované dokumenty sa potom označujú za *validné* dokumenty. Porovnávanie dokumentov s ich DTD popisom uskutočňujú overovacie analyzátory.

2.6 XML schéma

Od roku 2004 je konzorciom W3C odporúčaná druhá edícia XML schémy. XML schéma je jazyk, ktorý slúži na opis štruktúry XML dokumentov. Jazyk je úspešnejšou XML alternatívou DTD. Oproti DTD podporuje menné priestory, dátové typy, XSLT transformáciu, XML analyzátor atď. Ak XML dokument odpovedá XML schéme, je dokument označený podobne ako u DTD za *validný*.

XML schéma definuje :

- miesto, kde sa môžu v dokumente vyskytovať rôzne elementy
- atribúty
- potomkov elementov
- poradie a počet elementov
- obsah elementov
- dátové typy elementov a ich atribútov
- hodnoty elementov a atribútov

Kvôli názornosti uvádzam aj jednoduchú XML schému, ktorá odpovedá XML dokumentu z príkladu číslo 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="zákazník" type="typ_zákazník"/>
    <xs:complexType name="typ_zákazník">
      <xs:sequence>
        <xs:element name="meno" type="xs:string"/>
        <xs:element name="priezvysko" type="xs:string"/>
        <xs:element name="adresa" type="xs:string"/>
        <xs:element name="telefónne_číslo" type="xs:decimal"/>
      </xs:sequence>
    <xs:attribute name="ID" type="xs:integer" use="required"/>
  </xs:complexType>
</xs:schema>
```

Príklad č. 5 XML schéma

2.7 XML analyzátory

XML analyzátory tvoria API (Application programming interface) rozhranie pre manipuláciu so XML dokumentmi. Najpopulárnejšími XML API pre prácu so XML dokumentmi sú dnes Simple API for XML (SAX) a Document Object Model (DOM). Pre obe rozhrania existujú implementácie v najpoužívanejších programovacích jazykoch a rozhraniach.

2.7.1 DOM (objektový model dokumentu)

DOM model je založený na načítaní celého XML dokumentu, jeho spracovaním a následným vložením do hierarchickej štruktúry objektov, ku ktorej potom môžu pristupovať klientske aplikácie.

2.7.2 SAX (jednoduché API pre XML)

Model SAX je udalosťami riadené aplikačné rozhranie. Čo znamená, že aplikácia môže pristúpiť iba k tým častiam (štruktúram) dokumentu, ktoré sú analyzátorom pribežne spracované. SAX funguje na tzv. princípe „Push“, oproti stromovo orientovaným rozhraniam, ktoré fungujú na princípe „Pull“. Princíp „Push“ znamená, že prenos, či spracovanie je iniciované zdrojom dát a „Pull“ princíp znamená, že prenos, či spracovanie iniciuje konzumenta dát.

2.7.3 Hlavné rozdiely medzi DOM a SAX [1]

DOM	**SAX**
Vytvára kópiu stromu dokumentu do pamäte.	Neukladá dáta z dokumentu XML počas spracovania.
Strom XML dokumentu, uložený v pamäti je možné modifikovať.	Neobsahuje žiadnu podporu pre modifikáciu, alebo zapisovanie do dát v dokumente XML.
Celý dokument sa musí najprv spracovať a potom je strom dokumentu prístupný klientskej aplikácii.	Dáta dokumentu sa sprístupnia, už počas jeho spracovania analyzátorom.

Tabuľka č. 1 Rozdiely medzi DOM a SAX

2.8 XPath

XPath (XML Path language) verzia 1.0 definuje organizáciou W3C schválenú syntax pre popis adries častí XML dokumentov. Adresovanie častí dokumentov funguje tak, že sa vyhodnotia XPath výrazy podľa logického stromu XML dokumentu. XPath sa na uzly odkazuje na základe ich pozície, relatívnej pozície, typu, obsahu a iných ďalších kritérií. Jazyk XPath je základom pre ďalšie jazyky ako XQuery a XPointer.

```
/zákazník/@ID
```

Príklad č. 6 XPath výraz

2.9 XQuery

Jazyk XQuery je dotazovací jazyk od konzorcia W3C. Spolupracuje s jazykom XPath na vyhľadávaní a vyberaní elementov a atribútov zo XML dokumentov. Organizácia W3C prirovnala jazyk XQuery k jazyku SQL, ale iba v tom význame, že XQuery je pre XML, to isté ako SQL pre databázy.

Ako názornú ukážku uvádzam funkciu XQuery doc() s XPath výrazom. Kde doc() funkcia otvorí daný dokument a XPath výraz zobrazí element „adresa“.

```
doc("zákazníci.xml")/zákazník/adresa
```

Príklad č. 7 XQuery a XPath

2.10 XPointer

XPointer sa používa k identifikácií častí XML dokumentov. Pripája sa na koniec adresy URI ako jej súčasť a vyznačuje nejakú konkrétnu časť dokumentu XML. XPointer na viac rozširuje XPath o operácie, pomocou ktorých je možné vyberať nielen konkrétne miesta, ale aj rozsahy v XML dokumente. Pre predstavu prikladám aj ukážku (príklad číslo 8) použitia XPointer. V ukážke XPointer výraz “`#xpointer(ID('123'))`“ odkazuje na element s atribútom ID a hodnotou atribútu “123“.

2.11 XLink

Xlink predstavuje syntax založenú na atribútoch, pre pripojovanie odkazov k dokumentom XML. XLink reprezentuje ako jednoduché odkazy (z bodu A do bodu B), tak aj obojsmerné odkazy a tiež aj viacsmerové (mnoho rôznych ciest medzi ľubovoľným počtom dokumentov). Odkaz môže byť umiestnený v XML a odkazovať na iné dokumenty, ktoré môžu i nemusia byť XML dokumentmi.

```
xlink:href="http://www.stud.fit.vutbr.cz/~xdvors07/zákazníci.xml#xpo
                        inter(ID('123'))"
```

Príklad č. 8 XPointer a XLink

2.12 XSL

XSL (eXtensible Stylesheet Language) je jazyk pre transformáciu a formátovanie XML dokumentov.
XSL sa skladá z dvoch častí :

1. XSLT (XSL Transformations) – jazyk pre transformáciu XML dokumentov. Táto technológia prevádza XML dokument z jedného formátu do iného. Ako napríklad do HTML, alebo XHTML formátu. Pri transformácii sa používa výraz XPath, na označenie elementov a atribútov, ktoré sa majú zmeniť podľa štýlových šablón. Šablóna je vytvorená na základe pravidiel jazyka XSL.

2. XSL-FO (XSL Formatting objects) – jazyk pre formátovanie dokumentov XML. Prevedie významové značkovanie na formátové. XSL-FO popisuje presné usporiadanie dokumentu na stránke. Elementy XSL-FO reprezentujú jednotlivé stránky, bloky textu na stránkach, grafiku alebo oddeľovače. XSL-FO dokumenty sa netvoria priamo, ale za použitia štýlových šablón v XSLT. Tie prevedú značkovanie. Dokument potom zobrazí aplikácia, ktorá prečíta XSL-FO.

V Príklade číslo 9 uvádzam XSLT šablónu pomocou, ktorej XSLT transformácia prevedie XML dokument uvedený v príklade číslo 3 do HTML formátu, pričom elementy XML dokumentu naplnia HTML tabuľku.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <body>
        <table border="1">
          <tbody>
            <tr bgcolor="green">
              <th>ID</th>
              <th>Meno</th>
              <th>Priezvysko</th>
              <th>Adresa</th>
              <th>Telefónne číslo</th>
            </tr>
            <xsl:for-each select="/zákazník">
            <tr>
              <td><xsl:value-of elect="@ID"/></td>
              <td><xsl:value-of select="meno"/></td>
              <td><xsl:value-of select="priezvysko"/></td>
              <td><xsl:value-of select="adresa"/></td>
              <td><xsl:value-of select="telefónne_číslo"/></td>
            </tr>
            </xsl:for-each>
          </tbody>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Príklad č. 9 XSLT šablóna

2.13 Typy XML dokumentov

XML dokumenty môžeme rozdeliť do dvoch skupín [1]. Toto rozdelenie má hlavne význam pri databázovej podpore XML a aplikačnom využití.

Dátové XML dokumenty (anglicky data based) sú dokumenty s obsahovým zameraním na dáta a plnia funkciu samopopisnej obálky, slúžiacu pre prenos či manipuláciu s dátami. Dokument je silno štruktúrovaný a pravidelný.

Dátový XML dokument môže predstavovať objednávku, zoznam zamestnancov, výpis bankového účtu, vedecké dáta (napr. seizmické záznamy) atď.

```
<?xml version="1.0" encoding="UTF-8"?>
<zamestnanci>
      <zamestnanec id="1">
            <meno>Jano Hluchý</meno>
            <oddelenie>Personálne</oddelenie>
            <plat>25000</plat>
      </zamestnanec>
      <zamestnanec id="2">
            <meno>Martin Dlhý</meno>
            <oddelenie>IT</oddelenie>
            <plat>30000</plat>
      </zamestnanec>
</zamestnanci>
```

Príklad č. 10 Zoznam zamestnancov ako príklad dátového XML dokumentu

Dokumentové XML dokumenty (anglicky document based) majú obsah zameraný dokumentovo. Štruktúra XML dokumentu je premenlivá so zmiešaným obsahom. Skôr sú určené pre čítanie a spracovanie ľuďmi.

Príkladom dokumentových XML dokumentov môže byť napr. email, životopis, rôzne formuláre, webová stránka atď.

```
<?xml version="1.0" encoding="UTF-8"?>
<email>
      <dátum>Pondelok <čas>20:00</čas>
            5 mája 2008
      </dátum>
      <od>Fero Hora</od>
      <komu>Mne</komu>
      <predmet>Pozdrav z dovolenky</predmet>
      <obsah>obsah emailu</obsah>
</email>
```

Príklad č. 11 Email ako príklad dokumentového XML dokumentu

3 Databázové systémy a XML

Rozmach internetu spôsobil vzrastajúcu popularitu jazyka XML, ktorého hlavné zameranie sa sústredí na prácu s dátami. Na čo museli zareagovať a stále aj reagujú najväčší výrobcovia softwarových produktov, ktoré s dátami pracujú. Týmito produktmi sú databázové systémy. Či už sa jedná o komerčné (napr. Microsoft SQL server, Oracle database, alebo IBM DB2), alebo nekomerčné databázové produkty (napr. MySQL, alebo PostgreSQL). Obidve skupiny sa snažia, čo najviac sprístupniť a zefektívniť podporu XML dokumentov. V rámci zadania som vybrali tri databázové produkty a stručne popísal ich podporu XML dokumentov. Popis som sústredil na natívnu podporu (Microsoft SQL Server od firmy Microsoft, Adaptive Server Enterprise od spoločnosti SyBase a na Oracle 11g od korporácie Oracle Corporation).

S natívnou podporou súvisí aj štandard SQL/XML, ktorý je veľa databázami podporovaný. Jedná sa o štandard definovaný International Committee for Information Technology Standards, čo je komisia zaoberajúca sa štandardmi syntaxe a sémantiky databázových jazykov, včetne SQL. Štandard SQL/XML rozširuje SQL príkazy o funkcie, ktoré pracujú s XML dátami.

3.1 Microsoft SQL Server

Súčasnou verziou je Microsoft SQL server 2008. Je to relačný databázový systém s dotazovacími jazykmi T-SQL a ANSI SQL od spoločnosti Microsoft. Spoločnosť Microsoft sa snaží posunúť spravovanie dát na úroveň, na ktorej sa dáta sami ladia, udržujú a organizujú. Ďalej Microsoft SQL Server 2008 podporuje štruktúrované a čiastočne štruktúrované dáta, tiež aj multimediálne dáta (obrázky, audio, video atď.), ktoré môžu byť uložené ako BLOB (Binary Large Objects). Server poskytuje úložisko pre rôzne dáta ako XML, emaily, súbory, dokumenty, súradnice a operácie nad nimi ako vyhľadávanie, dotazovanie, analyzovanie, zdieľanie a synchronizáciu medzi dátami. Informácie uvedené v tejto kapitole som čerpal z literatúry [3] a [2i].

3.1.1 XML podpora u servera Microsoft SQL Server

Microsoft SQL Server 2008 ponúka štyri možnosti pre uloženie XML dokumentu. Prvou možnosťou je rozloženie XML dokumentu a následné vloženie vybraných dát do databáze. Druhý spôsob ponúka uložiť XML dokument vo forme textu. Tento spôsob je však náročný pre prácu s XML dokumentom. Ak je dokument príliš veľký, tak sa naskytuje tretia možnosť a to uložiť dokument ako veľký objekt (Large Object Storage). Pričom sa uloží identická kópia XML dokumentu. Posledným variantom je natívne uloženie XML dokumentu ako inštanciu dátového typu XML.

Natívna XML podpora servera sa sústredí na dátový typ XML. Jej najvýznamnejšie časti som rozpísal v nasledujúcich statiach.

3.1.1.1 Dátový typ XML

Dátový typ XML ukladá XML v jeho natívnom formáte a zároveň sprístupňuje dáta XML dokumentu pre ľahké a efektívne spracovanie. Dátový typ XML je možné použiť ako premennú, parameter uloženej procedúry či užívateľom definovanej funkcie, návratovú hodnotu, alebo ako dátový typ stĺpca v tabuľke. Pričom je obmedzená veľkosť XML dokumentu do 2GB. Interná reprezentácia uchováva štruktúru dokumentu, poradie prvkov dokumentu, elementy a atribúty. Ďalšími dôležitými obmedzeniami pri použití dátového typu XML ako stĺpec v tabuľke je, že ho nemôžeme použiť ako primárny alebo cudzí kľúč, ani naň nemôžeme uplatniť obmedzenie UNIQUE. Použitie dátového typu XML nám poskytuje aj podporu iných XML technológií ako napríklad menné priestory, XML schému, XQuery výrazy atď.

```
CREATE TABLE objednavky_tab (
     ID_objednavky    INT   PRIMARY KEY,
     Objednavka_doc   XML   NOT NULL)
```

Príklad č. 12 Vytvorenie tabuľky s dátovým typom XML

3.1.1.2 XML schémy (XSD)

Microsoft SQL server poskytuje tiež podporu XML schém. Tie sa využívajú pre validáciu XML dokumentov. XML schémy je možné uložiť do databázy ako objekty a tie asociovať s premennou alebo stĺpcom dátového typu XML. Pred vytvorením XML dátového typu, musí byť schéma zaregistrovaná, až potom môže byť asociovaná s premennou, parametrom, alebo stĺpcom dátového typu XML.

XML schémy sú ukladané do kolekcie XML schém. Čo je jednotka, resp. metadátový objekt (pre prirovnanie, niečo ako tabuľka v databáze), ktorá uchováva a spravuje XML schémy. Kolekcia XML schém nám ponúka vytváranie, modifikovanie či mazanie XML schém a to iba tých schém, ktoré sú ňou definované. Pred použitím DDL operácií na XML schémy sa najprv musia tieto operácie overiť v nastaveniach kolekcie XML schém.

Ďalšími významnými operáciami nad XML schémami, ktoré ponúka Microsoft SQL server je napríklad generovanie XML schém alebo možnosť dotazovať XML schémy. Asociácia XML schémy s inštanciou XML dokumentu nám ponúka viacero výhod. Najvýznamnejšou je automatická kontrola validácie vkladaného, alebo modifikovaného XML dokumentu proti XML schéme. Čo má za následok zaistenie integrity dát. Ak dátový typ XML uchováva inštanciu asociovanú s XML schémou, tak tento dátový typ označujeme za *typovaný* (anglicky typed*)*. V opačnom prípade, kedy inštancia nemá priradenú XML schému, tak dátový typ XML je označený za *netypovaný* (untyped*)*.

```
CREATE TABLE objednavky_tab (
     ID_objednavky   INT   PRIMARY KEY,
     Objednavka_doc  XML(objednavka_xsd))
```

Príklad č. 13 Asociovanie XML schémy s XML dokumentom

3.1.1.3 Indexy

SQL server nám poskytuje možnosť vytvárať indexy nad dátovým typom XML a tým zefektívniť časté dotazy nad dátami XML dokumentu. Indexy u Microsoft SQL servera delíme na dva hlavné typy indexov, a to primárne a sekundárne indexy. Indexy, ako aj iné databázové objekty, môžeme vytvárať, mazať, či modifikovať.

Primárny XML index sa mierne odlišuje od klasického indexu SQL servera a to hlavne zameraním na XML dáta. Primárny index spojuje každý uzol stĺpca dátového typu XML so stĺpcom primárneho kľúča SQL. Z toho vyplýva obmedzenie, resp. že primárny index sa dá použiť iba nad tabuľkou, ktorá má primárny kľúč.

```
CREATE Primary XML INDEX
     ON objednavka_tab (Objednavka_doc)
```

Príklad č. 14 Vytvorenie primárneho indexu nad tabuľkou objednávka_tab

Sekundárne indexy sú vytvárané na základe primárneho indexu. Sekundárne indexy delíme na tri typy a to Path (cesta), Value (hodnota) a Property (vlastnosť) indexy. Podmienkou je, že môžu byť implementované až po vytvorení primárneho indexu. Ak potrebujeme urýchliť výrazy XPath a XQuery, ktoré odkazujú na určitý uzol s explicitnou hodnotou, je potrebné vytvoriť Path index pomocou FOR PATH. Avšak ak potrebujeme dotazovať hodnoty uzlov, je možné vytvoriť Value index pomocou FOR VALUE. Posledným typom sekundárnych indexov sú Property indexy, ktoré slúžia na optimalizáciu hierarchie elementov a atribútov a vytvárajú sa pomocou FOR PROPERTY. Za každým vytvoreným sekundárnym indexom musíme pomocou USING XML INDEX špecifikovať, že používame primárny index.

```
--Vytvorenie Path indexu
CREATE XML INDEX index_P
     ON objednavka_tab (Objednavka_doc)
USING XML INDEX index_1 FOR PATH
--Vytvorenie Value indexu
CREATE XML INDEX index_V
     ON objednavka_tab (Objednavka_doc)
USING XML INDEX index_1 FOR VALUE
--Vytvorenie Property indexu
CREATE XML INDEX index_Pro
     ON objednavka_tab (Objednavka_doc)
USING XML INDEX index_1 FOR PROPERTY
```

Príklad č. 15 Vytvorenie sekundárnych indexov

Za poznámku stojí ešte uviesť, že SQL server podporuje aj tzv. FULL-TEXTové indexy.

3.1.1.4 Vkladanie a vyberanie dát

Operácie pre vkladanie a vyberanie XML dát do a z databáze sú hlavne podporované v rámci XML funkcií. Tieto funkcie rozširujú syntax T-SQL. Vkladať je možné iba správne formátované dokumenty (dobre vytvorené), inak pokus o vloženie vyhodí výnimku.

- Funkcia *FOR XML* prevádza výsledky SQL dotazov na formát XML. Používa sa spolu s voľbami RAW, AUTO, EXPLICIT (tie určujú tvar výsledného XML dokumentu) a výrazom PATH na stanovenie formátu výstupu dotazu. Laicky povedané, funkcia vytvára XML z riadkov dát.

  Nasledujúci príklad vypíše hodnoty pre *ID*, *meno* a *adresu* z tabuľky *zákaznici* uloženej v databáze *názov_databázy* v XML formáte podľa voľby AUTO.

```
USE nazov_databaze
GO
SELECT ID, meno, adresu
FROM zakaznici
FOR XML AUTO
```

Príklad č. 16 Použitie funkcie FOR XML

- Funkcia *OPENXML* je systémová funkcia, a funguje presne opačne ako funkcia FOR XML, čiže rozloží XML dokument do sady riadkov. Táto interná reprezentácia sa tvári ako databázový pohľad a dá sa s ňou pracovať ako s ostatnými relačnými dátami.

- *XML Bulk Load* je funkcia na načítanie XML dokumentov zo súboru do dátového typu XML, či už na strane klienta, alebo servera. Používa sa spolu so systémovou tabuľkovou funkciou OPENROWSET, ktorá stanoví používateľa Bulk.

  Nasledujúci príklad demonštruje použitie funkcie XML Bulk Load. Príkazom *INSERT* sa vložia všetky dáta (resp. celý obsah dokumentu *objednavka.xml*) načítané zo súboru pomocou Bulk a OPENROWSET do tabuľky *objednavky_tab.*

```
INSERT INTO objednavky_tab(1, xmldata)
       SELECT * FROM OPENROWSET
       (Bulk 'C:objednavky.xml',
        SINGLE_CLOB) as xmldata
```

Príklad č. 17 Použitie funkcie XML Bulk Load

- Funkcie *CAST* a *CONVERT* prevádzajú stringové dátové typy do dátového typu XML.
- Dáta z databázy môžu byť vybrané tiež pomocou XML pohľadov (views).

3.1.1.5 Dotazovanie XML dát

Microsoft SQL Server podporuje XPath a XQuery výrazy, slúžiace na dotazovanie XML dát. Okrem toho v servery je implementovaný štandard XQuery a to piatimi funkciami u dátového typu XML :

- *xml.exist()* - metóda, ktorá zisťuje existenciu elementu, určeného výrazom XQuery, ktorý je vstupom metódy.

- *xml.value()* – metóda vracia skalárny typ (hodnotu) elementu, určeného výrazom XQuery.

- *xml.query()* – metóda funguje rovnako ako metóda xml.value(), až na to že vracia časť XML dokumentu, určenú na základe výrazu XQuery, ako dátový typ XML.

- *xml.nodes()* – metóda na základe vstupu, ten je určeným výrazom XQuery, vráti rozložený XML dokument ako dátový typ XML.

- *xml.modify()* – metóda umožňuje manipuláciu s uzlami XML inštancie (ako vkladanie, odstraňovanie, či upravovanie) pomocou výrazu XQuery, ktorý určí uzol.

3.2 Adaptive Server Enterprise

Adaptive Server Enterprise je relačný databázový systém s platformovým zameraním hlavne na operačný systém UNIX, avšak tiež dostupný aj pre operačný systém Windows. Dnes je k dispozícii verzia 15.0. Databázový systém je navrhnutý tak, aby podporoval náročné požiadavky OLTP aplikácií a aplikácie analytického charakteru. Hlavné zameranie serveru je na oblasť obchodovania cez internet (e-Business), správu a podporu charakteristických dát pre túto oblasť a s tým aj súvisiacu optimalizáciu výkonu a bezpečnosti. Údaje o tejto kapitole som naštudoval z literatúry [3i].

3.2.1 Adaptive Server Enterprise a podpora XML

Adaptive Server Enterprise (ďalej už len ASE) podporuje XML dáta, vo forme XML služieb. Tento mechanizmus nám umožňuje uložiť XML dokument buď, ako znakové dáta (napr. char, varchar, text atď.), alebo ako rozložený (anglicky parsed) XML dokument uložený v dátovom type *image*. Možnosti XML podpory u servera ASE môžeme rozdeliť na okruhy :

- Indexovanie XML dát
- Vkladanie a vyberanie XML dát
- DTD popis a XML schéma
- XQuery, XPath a menné priestory

3.2.1.1 Indexovanie XML dát

Pre rýchlejší spôsob vykonávania častých dotazov a operácií nad XML dátami, nám ASE poskytuje vstavanú funkciu *xmlparse()*. Funkcia rozloží XML dokument, ktorý bol vložený ako parameter funkcie, a vráti jeho rozloženú podobu v dátovom type *image*, pričom sa vytvoria indexy na rozložené časti XML dokumentu. Vývojári ASE odporúčajú používať funkciu *xmlparse()* kvôli efektívnejšiemu a rýchlejšiemu spracovávaniu XML dokumentov.

Nasledujúca ukážka zmení hodnotu stĺpca *image_doc* tabuľky *tab_example* o rozložený XML dokument zo stĺpca *text_doc*.

```
UPDATE tab_example SET image_doc = xmlparse(text_doc)
```

Príklad č. 18 Použitie funkcie xmlparse()

3.2.1.2 Vkladanie a vyberanie XML dát

V rámci XML podpory ASE server poskytuje aj funkcie pre vkladanie a vyberanie XML dát do a z databázy. Tieto funkcie sa delia na dve skupiny. V jednej skupine je funkcia, ktorá rozširuje SQL príkazy a do druhej skupiny radíme funkcie založené na jazyku Java.

Funkcia z prvej skupiny, rozširuje SQL príkazy o klauzulu FOR XML. Toto rozšírenie SQL príkazu vracia výsledok reprezentovaný XML formou. Pričom použitie je jednoduché, stačí pridať klauzulu FOR XML na koniec SQL príkazu.

Názorná ukážka vyberie hodnoty *ID* a *meno* z tabuľky *tab_example* v XML podobe.

```
SELECT ID, meno FROM tab_example FOR XML
```

Príklad č. 19 Použitie rozširujúcej klauzuly FOR XML

Funkcie založené na jazyku Java nám ponúkajú širšiu funkcionalitu. Nevýhodou týchto funkcií je, že nie sú priamo súčasťou servera a pred použitím sa musia doinštalovať. Tieto funkcie môžeme opäť rozdeliť do dvoch skupín, a to podľa toho, či vyberajú dáta v XML podobe, alebo XML dáta vkladajú do databázy v relačnej podobe.
Medzi funkcie, čo vyberajú dáta z databázy v XML podobe, patria *forxmlj()*, *forxmldtdj()*, *forxmlschemaj()* a tiež aj *forxmlallj()*. Tieto funkcie majú spoločnú vec, všetky vrátia výsledok SQL dotazu v podobe XML. Posledné tri funkcie však okrem toho rozširujú funkcionalitu o prvky DTD popisu XML dát a XML schémy. Preto sú podrobnejšie popísané v stati o DTD a XML schémach.
K funkciám slúžiacim pre vkladanie XML dát do databáze radíme funkcie *forsqlcreatej()*, *forsqlinsertj()* a *forsqlscriptj()*. Funkcia *forsqlcreatej()* na základe XML schémy ako parametra funkcie spustí SQL skript, ktorý vygeneruje tabuľku. *Forsqlinsertj()* sa používa na naplnenie tabuľky XML dátami, ako dátového typu varchar. Posledná z uvedených funkcií *forsqlscriptj()* je kombináciou oboch predchádzajúcich funkcií. Takže najprv vytvorí tabuľku a následne ju naplní XML dátami.

ASE server podporuje aj dve funkcie orientované na klientské aplikácie :

- *ForXmlTree* – funkcia, ktorá vloží výsledok SQL príkazu do stromovej štruktúry XML dokumentu.
- *OpenXML* – funkcia je v podstate inverzná k predchádzajúcej funkcii a na základe SQL príkazu vloží XML dáta z XML dokumentu do tabuľky.

3.2.1.3 DTD a XML schémy

Integrovaný XML dátový manažment servera ASE podporuje aj DTD popis XML dokumentu a XML schémy, ktoré nám umožňujú validovať dokumenty a tým udržiavať integritu dát. Validné dokumenty sú vyžadované napr. pri práci s funkciami založenými na jazyku Java. Validovať dokument proti schéme, či DTD, môžeme pomocou vstavanej funkcie *xmlvalidate()*. Ak je potrebné získať XML schému, alebo DTD popis ASE, server nám ponúka niekoľko ciest ako to docieliť.

Jednou je použitie klauzuly *for xml all*. Táto klauzula rozširuje SQL príkaz SELECT, ktorý potom vráti výsledok SQL príkazu v XML podobe, DTD popis výsledku a XML schému súčasne.

Ekvivalentom klauzuly *for xml all* je funkcia založená na jazyku Java *forxmlaljl(),* ktorá taktiež vracia výsledky príkazu v XML podobe, DTD popis a XML schému. Ak potrebujeme vrátiť iba DTD popis XML dát alebo iba XML schému, máme k dispozícii funkcie *forxmldtdj()* a *forxmlschemaj().*

3.2.1.4 XQuery, XPath a menné priestory

Server ASE okrem štandardu XML podporuje aj iné štandardy pre prácu s XML dokumentmi, ako XQuery, XPath a menné priestory. Podpora XPath disponuje okrem podpory entít (resp. špeciálnych znakov) aj funkciami pre znakovú úpravu (toupper, tolower, normalize-space a concat).

XQuery má podporu v rámci funkcií pre dotazy :

- *xmlextract* – je vstavaná funkcia, ktorá slúži na vyberanie elementov zo XML dokumentu na základe výrazu XQuery.
  Názorný príklad vráti hodnotu elementu *meno*, ak hodnota elementu *ID* je rovná 1.

```
SELECT xmlextract('/zakaznici [ID=1] / meno')
FROM tab_example
```

Príklad č. 20 Použitie funkcie xmlextract

- *xmltest* – funkcia, ktorá testuje existenciu uzla (ako napr. elementu) podľa XPath výrazu.
  Nasledujúci príklad vypíše všetky hodnoty *ID* z tabuľky *tab_example*, kde hodnota elementu *meno* sa zhoduje s reťazcom *MATEJ*.

```
SELECT ID FROM tab_example
WHERE '/zakaznici [meno="MATEJ"]' xmltest image_doc
```

Príklad č. 21 Použitie funkcie xmltest

- *xmlrepresentation* – posledná vstavaná funkcia zistí či je XML dokument rozložený, alebo nie.

3.3 Oracle

Oracle je moderný multiplatformový databázový systém riadenia báze dát (Oracle database managment system - DBMS). Je vyvíjaný spoločnosťou Oracle Corporation. Aktuálna verzia je Oracle Database 11g. Jedná sa o relačno-objektový databázový systém, ktorý okrem podpory štandardného relačného dopytovacieho jazyka SQL, podporuje aj procedurálny jazyk PL/SQL. Ten rozširuje možnosti jazyka SQL. Podporované sú aj programátorské jazyky ako C/C++ (prístup skrz ODBC), PHP, alebo Java (prístup skrz JDBC). Oracle disponuje aj širokou ponukou rôznych druhov a typov dát. Od bežných dát, cez multimediálne, priestorové, časové až po XML dokumenty. Pričom spracovanie dát je umožnené s veľmi pokročilými možnosťami, vysokým výkonom a jednoduchou škálovateľnosťou. Podklady k tejto kapitole som čerpal z literatúry [4] a [5].

3.3.1 Podpora XML u databázového systému Oracle

Podpora pre prácu s XML dokumentmi u databázového systému Oracle pozostáva z dvoch balíkov. Jedná sa o balík Oracle XML Developer`s Kit (skrátene XDK) a o balík Oracle XML DB.

3.3.1.1 Oracle XML Developer`s Kit

Balík XDK nezahrňuje vstavanú podporu, ale skôr sa jedná o všestrannú sadu nástrojov pre vytváranie aplikácií, ktoré pracujú s XML formátom. Tento set komponent podporuje programovacie jazyky Java, C a C++. Nasledujúca tabuľka prevzatá z literatúry [6], popisuje komponenty sady XDK.

Komponenta	Popis komponenty	Programovací jazyk
XML Parser	Vytvára a rozložuje XML s rozhraniami DOM a SAX.	Java, C a C++
XML Compressor	Binárna kompresia a dekompresia XML dokumentov. Kompresor je súčasťou XML Parseru pre jazyk Java.	Java
Java API for XML Processing (JAXP)	Umožňuje použiť SAX, DOM, XML Schema procesor, XSLT procesor a prípadne alternatívny procesor pre aplikácie napísane v jazyku Java.	Java
XSLT Processor	Transformácia XML formátu na iný textový formát, ako napríklad HTML.	Java, C a C++
XML Schema Processor	Validácia XML dokumentov	Java, C a C++
XML Class Generator	Generuje triedy jazyka Java a C++ z DTD popisu a z XML schém pre posielanie XML dát z aplikácií.	Java a C++
XML Pipeline Processor	Používa XML procesy špecifikované v deklaratívnom XML Pipeline dokumente.	Java
XML Java Beans	Poskytuje sadu zapuzdrených XDK komponent pre jednoduchšie použitie vo vývojovom prostredí.	Java
XML SQL Utility (XSU)	Generovanie XML dokumentov, DTD popisu a XML schém z SQL príkazov.	Java, PL/SQL
TransX Utility	Načíta preložené počiatočné dáta a správy do databázy pomocou XML.	Java
XSQL servlet	Kombinácia technológií XML, SQL a XSLT pre server s dynamickým webovým obsahom	Java
Oracle SOAP Server	Poskytuje odľahčený SOAP protokol pre posielanie a prijímanie dotazov a odpovedí skrz internet	C

XSLT Virtual Machine (XVM)	Poskytuje vysoko výkonný XSLT transformačný mechanizmus, ktorý podporuje kompilované XSLT šablóny.	C, C++

Tabuľka č. 2 Komponenty balíka XDK

Sadu nástrojov XDK je možné použiť spolu s balíkom Oracle XML DB, k vytvoreniu aplikácie bežiacej v databáze Oracle. Avšak komponenty XDK sa dajú použiť tiež nezávisle.

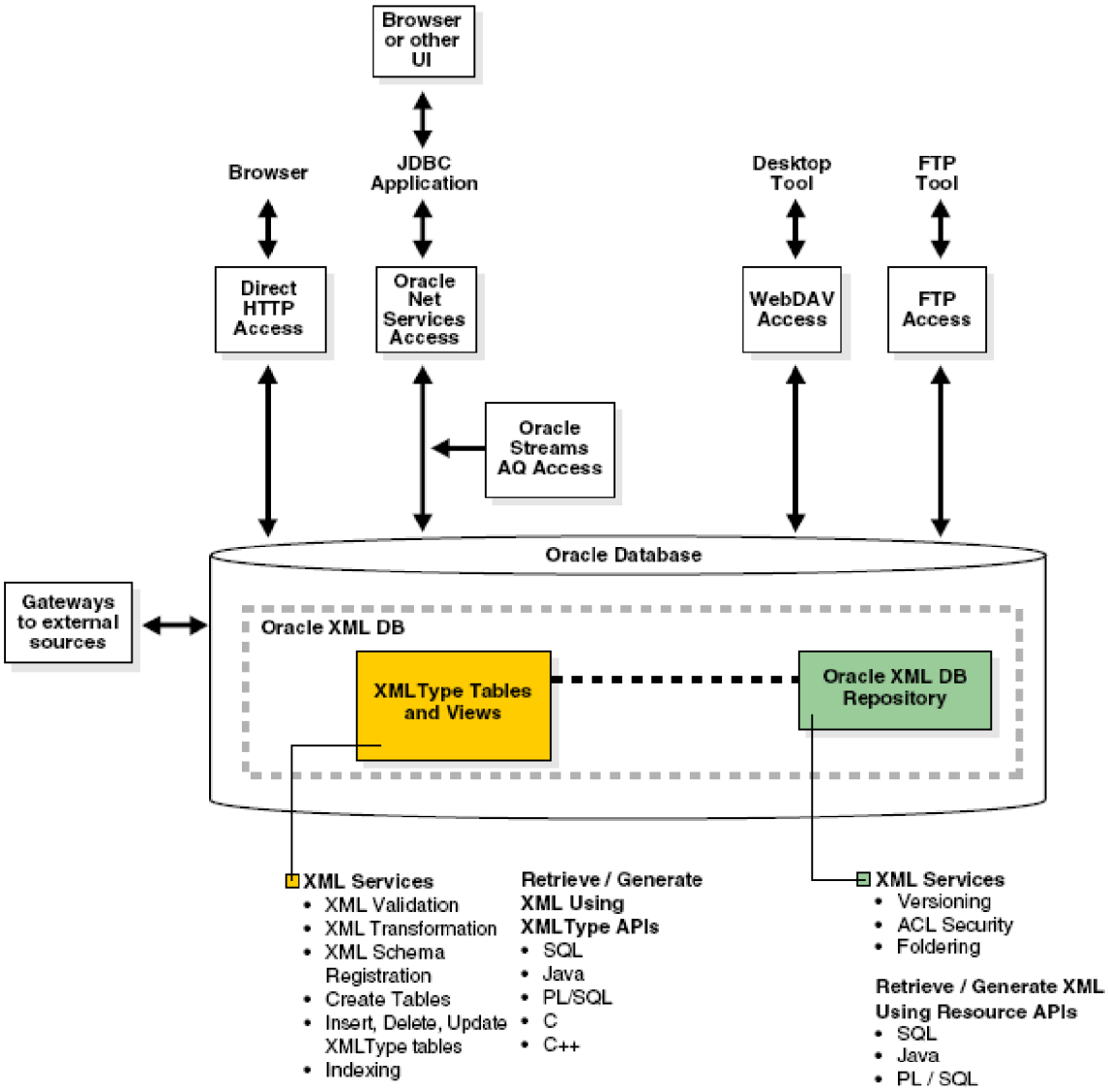
3.3.1.2 Oracle XML DB

Oracle XML DB zoskupuje množinu technológií určených pre vysoko výkonné a efektívne ukladanie a vyberanie XML dokumentov. Tento balík tak rozširuje relačno-objektovú databázu o XML technológie, čím vytvára z databázového systému Oracle plnohodnotný nástroj pre ukladanie a spracovanie XML dokumentov a to aj v natívnej podobe.

Možnosti, ktoré ponúka balík Oracle XML DB :

- Nativný dátový typ XMLType určený pre ukladanie a spracovanie XML dokumentov.
- Široká podpora štandardu XML schém.
- XML/SQL dualita, ktorá dovoľuje SQL operácie nad XML dátami a opačne XML operácie nad SQL dátami.
- SQL funkcie definované štandardom SQL/XML.
- Podpora štandardov ako napríklad XPath, XQuery, menné priestory, DOM, XSLT a iné.
- Možnosť ukladať, spravovať, organizovať a pristupovať k XML dátam v repozitáre XML DB Repository
- Prístup k dátam pomocou komunikačných protokolov FTP, WebDAV a http(s).

Vnútorná architektúra balíka Oracle XML DB sa skladá z dvoch častí, a to z úložiska tabuliek a pohľadov dátového typu XMLType a z repozitáru XML DB Repository. Nasledujúci obrázok prevzatý z literatúry [5] ilustruje dané rozdelenie vnútornej architektúry, spolu s prístupovými komunikačnými protokolmi, XML službami a podporovanými programovacími jazykmi.



Obrázok č. 1 Vnútorná štruktúra architektúry Oracle XML DB

Úložisko dátového typu XMLType

Táto časť balíka Oracle XML DB je zameraná na natívny dátový typ XMLType. S týmto dátovým typom súvisia aj všetky štandardy a technológie, ktoré tvoria XML podporu servera Oracle. Najvýznamnejšie prvky a mechanizmy XML podpory sú podrobnejšie popísané v kapitole číslo 4 Oracle XML DB.

Oracle XML DB Repository

XML DB Repository je komponenta Oracle XML DB, ktorá slúži k organizovaniu uchovávaných dát. Napriek tomu, že repozitár je navrhnutý pre XML dokumenty a XML schémy, môže byť použitý aj pre ostatné dáta uložené v databáze. Repozitár si môžeme

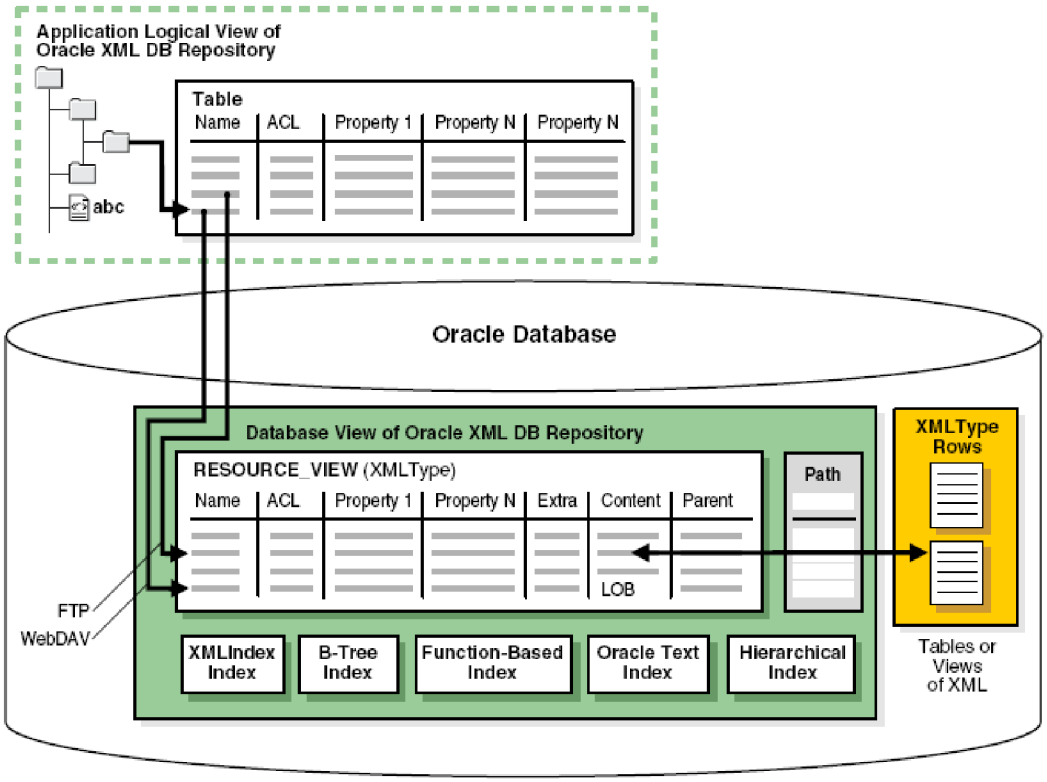
predstaviť ako virtuálny súborový systém, ktorý obsahuje položky, resp. zdroje. Tento hierarchický systém v skutočnosti pozostáva z tabuliek, pohľadov a indexov. Pod pojmom zdroje sa nachádzajú súbory, alebo adresáre, ale nie samostatné dáta. Každý zdroj obsahuje nasledujúce informácie :

- Je identifikovaný menom a prístupovou cestou.
- Má obsah, či už XML dáta, alebo iné.
- Systémom definované meta-dáta (napr. vlastníka, dátum vyvorenia).
- Užívateľom definované meta-dáta.
- Zoznam prístupových oprávnení.

Oracle XML DB Repository poskytuje nasledujúce služby :

- Drží informácie o verzii zdroja.
- ACL (Access control lists) bezpečnosť je bezpečnosť založená na zoznamoch prístupových oprávnení. Tento zoznam obsahuje každý zdroj.
- Správa súborov a adresárov, ktorá udržuje hierarchickú štruktúru zdrojov.
- Hierarchické indexovanie, pre rýchlejšie prehľadávanie dát uložených v Oracle XML DB.
- Prístup k zdrojom repozitáru pomocou http, FTP, WebDAV, tiež pomocou balíčka DBMS_XDB jazyka PL/SQL a SQL skrz RESOURCE_VIEW a PATH_VIEW.

Nasledujúci obrázok popisuje architektúru repozitára XML DB Repository, spolu aj s prístupom a službami.



Obrázok č. 2 Architektúra Oracle XML DB Repository [5]

3.4 Záver k databázovým produktom

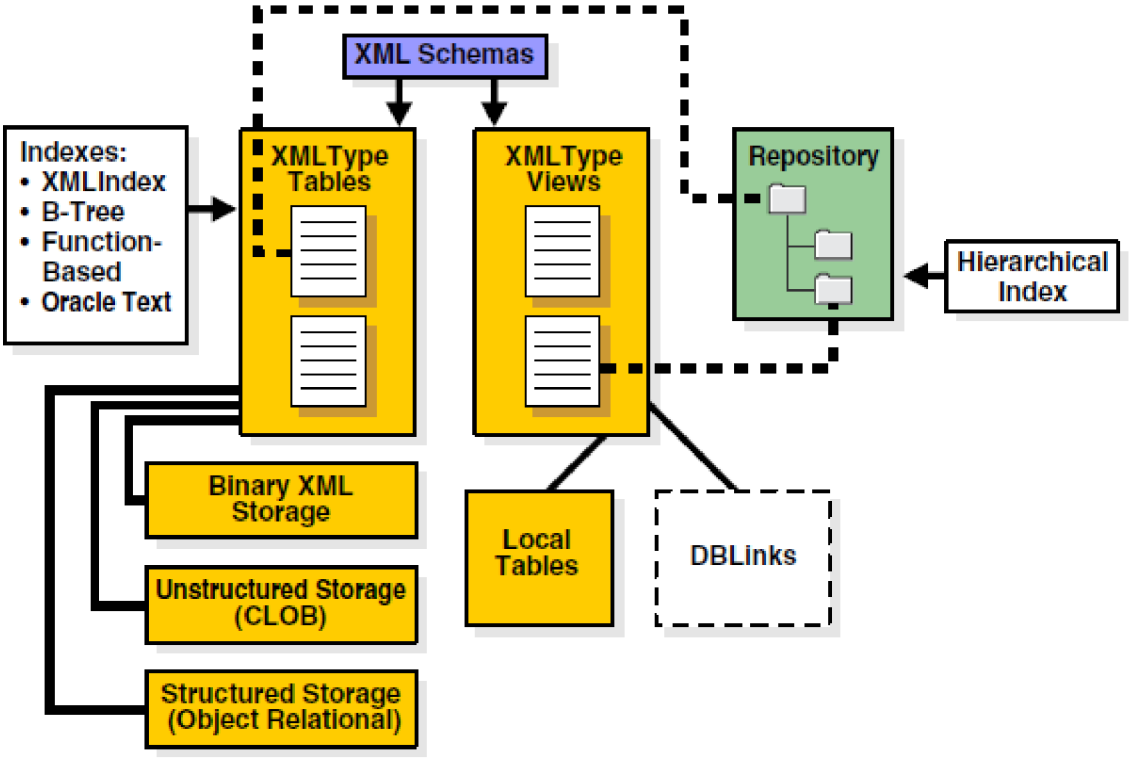
Každý z uvedených databázových systémov poskytuje základné možnosti, operácie a funkcie pre prácu s XML dokumentmi. Ako validáciu XML dokumentu proti XML schéme, indexovanie XML dát, vkladanie a vyberanie XML dokumentov do a z databázy a tiež aj dotazovanie pomocou XQuery a XPath. Rozdiely medzi XML podporou jednotlivých produktov sa líšia v rozsiahlosti, jednoduchosti práce s XML podporou a v zameraní XML podpory. Nasledujúca tabuľka zhŕňa jednotlivé časti XML podpory u vybraných databázových produktov, kde znamienko „+“ znamená, že daný prvok je podporovaný a znamienko „-“ práve naopak.

	Databázový produkt	**Microsoft SQL Server 2008**	**Oracle 11g**	**Adaptive Server Enterprise 15.0**
XML technológie	**DTD**	-	+	+
	XML schéma	+	+	+
	XQuery	+	+	-
	XPath	+	+	+
	XSL	+	+	+
Ukladanie XML dokumentov ako	**CLOB**	-	+	-
	BLOB	+	+	-
	VARCHAR	-	+	+
	Natívny dátový typ	+	+	+
Validovanie XML dokumentov	**Validácia oproti DTD**	-	+	+
	Validácia oproti XML schéme	+	+	+
	XML mapovanie	+	+	+
	Štandard SQL/XML	-	+	+
	XML Indexy	+	+	+
	XML Repository	-	+	-

Tabuľka č. 3 XML podpora u vybraných databázových produktov

4 Oracle XML DB

V tejto kapitole je popísaná časť balíka Oracle XML DB, ktorá je zameraná na úložisko natívneho dátového typu XMLType. Tento je určený pre správu a manipuláciu s databázovými objektmi (tabuľky, pohľady) obsahujúcimi dátový typ XMLType. Architektúru tejto časti ilustruje nasledujúci obrázok[5].



Obrázok č. 3 Architektúra XMLType úložiska

4.1 Dátový typ XMLType

XML dokument môžeme uložiť do databázy dvoma spôsobmi :

- Rozložením XML dokumentu mimo databázy, napríklad pomocou setu nástrojov XML Developer`s Kit, a extrahované dáta vložiť ako riadky do tabuliek.
- Druhou možnosťou je uložiť XML dokument ako CLOB (Character Large Object), BLOB (Binary Large Object), BFILE(Binary File), alebo ako VARCHAR hodnotu stĺpca.

Databáza však ani v jednom prípade nepozná či pracuje s XML obsahom. Tento problém rieši dátový typ XMLType, čo je natívny dátový typ databázy Oracle, ktorý slúži na

perzistentné ukladanie XML dokumentov do databázy. XMLType môžeme použiť pre objektovú tabuľku, či stĺpcový objekt, prípadne vytvoriť pohľad nad týmto dátovým typom. Dátový typ XMLType je dátový typ ako ostatné typy v databáze, avšak so svojimi vlastnými technikami pre manipuláciu. Jedná sa o objektový dátový typ, ktorý disponuje konštruktorom a metódami. Konštruktor nám umožňuje vytvoriť XMLType inštanciu z CLOB, BLOB BFILE a VARCHAR hodnôt. Väčšina metód inštancie XMLType je ekvivalentom SQL funkcií štandardu SQL/XML.

```
CREATE TABLE table OF XMLType
```

Príklad č. 22 Vytvorenie tabuľky typu XMLType

```
CREATE TABLE table_with_XML_as_column(
ID NUMBER(5) PRIMARY KEY,
XML_DOC XMLType )
```

Príklad č. 23 Vytvorenie tabuľky so stĺpcom typu XMLType

```
CREATE TABLE cdshop_as_table OF XMLTYPE
XMLTYPE STORE AS BINARY XML
XMLSCHEMA
"http://www.stud.fit.vutbr.cz/~xdvors07/xml/CDshop.xsd"
ELEMENT "customers"
```

Príklad č. 24 Vytvorenie tabuľky typu XMLType založenej na XML schéme

4.2 Modely XMLType úložiska

Dátový typ XMLType môžeme považovať za abstraktný dátový typ, z dôvodu výberu jedného z troch modelov úložiska. Model úložiska si vyberáme na základe obsahu XML dokumentu pri vytváraní XMLType tabuliek, alebo stĺpcov. Táto flexibilita slúži k zvýšeniu efektivity práce s XML dokumentom. XML dokumenty podľa obsahu rozdeľujeme na dátové, alebo dokumentové XML dokumenty (viď kapitola číslo 2.13 Typy XML dokumentov), prípadne na hybridný XML dokument, ktorý je variantom vyššie spomenutých typov. Na základe typu XML dokumentu si zvolíme jeden z modelov XMLType úložiska.

4.2.1 Štruktúrované XMLType úložisko

Tiež nazývané aj relačno-objektové úložisko. XML dokument je uložený ako sada objektov. K tomuto modelu radíme hlavne dátové, prípadne hybridné XML dokumenty. Dokumenty sa vyznačujú vysokou, ale zato drobnou štruktúrovanosťou. Napríklad ako záznamy zamestnancov.

4.2.2 Neštruktúrované XMLType úložisko

Nazývané aj textovo založené úložisko, ukladá dokumenty ako CLOB hodnoty. XML dokumenty sú ukladané vcelku, bez akejkoľvek zmeny. Sem patria prevažne dokumentové XML dokumenty, čiže dokumenty s nepravidelnou a hrubou štruktúrou. Príkladom sú webové stránky alebo články.

4.2.3 Binárne XMLType úložisko

Binárne úložisko ukladá XML dokumenty do špeciálneho binárneho XML formátu. Toto úložisko je efektívnejšie ako neštruktúrované. Binárne úložisko je zamerané na podobné typy dokumentov ako neštruktúrované.

4.3 XML schéma

XML schéma sa pokladá za kľúčový prvok XML podpory u servera Oracle, pretože okrem validácie (pomocou XMLType metód *isSchemaValid()* a *schemaValidate()*), ktorou obmedzuje obsah XML dokumentov (resp. špecifikuje štruktúru dokumentu) a tým udržuje integritu dát, poskytuje mnoho iných možností.

Ale hlavným dôvodom označenia XML schémy za kľúčový prvok XML podpory je ten, že špecifikuje to ako bude XML dokument uložený v databáze. Na jej základe budú namapované (priradené) dáta XML dokumentu do príslušných dátových typov. Určenie podoby XML dokumentu v databáze závisí na viacerých faktoroch. Ako napríklad od nastavenia parametrov pri registrácii schémy, alebo od anotácie schémy.

XML dokumenty rozdeľujeme na dve skupiny, čo sa týka asociácie s XML schémou :

- XML dokumenty, ktoré sú asociované s XML schémou (schémovo založené – *schema based*)
- XML dokumenty, ktoré XML schému nemajú priradenú (neschémovo založené – *nonschema based*)

4.3.1 Registrácia XML schémy

Pred akýmkoľvek použitím XML schémy sa musí schéma zaregistrovať, preto aby databáza o nej vedela a mohla ju naplno používať. XML schéma sa registruje veľmi jednoducho, a to pomocou balíka DBMS_XMLSCHEMA jazyka PL/SQL. Tento balík obsahuje procedúru DBMS_XMLSCHEMA.registerSchema, ktorá zaregistruje XML schému. Napriek tomu, že

registrácia XML schémy nie je transakčná operácia ako všetky DDL operácie. Registrácia XML schémy je atomická, to znamená, že ak pri registrácii nastane chyba, všetky operácie súvisiace s registráciou a stav databázy sú vrátené do situácie pred registráciou schémy.

Najhlavnejšie parametre procedúry DBMS_XMLSCHEMA.registerSchema :

- *schemaurl* URL identifikátor schémy. Podľa tohto parametru je identifikovaná schéma v celej databáze.
- *schemadoc* samotný dokument XML schémy, ktorý však musí byť validný. Hodnota parametru môže byť typu VARCHAR, CLOB, BLOB, BFILE, XMLType, alebo URIType.
- *local* parameter určuje či bude schéma zaregistrovaná ako lokálna, alebo ako globálna, resp. či ako súkromná (dostupná iba pre vlastníka schémy), alebo verejná (aj pre ostatných užívateľov).
- *gentypes* parameter, na základe ktorého sa budú generovať objektové typy alebo nie.
- *gentables* tento parameter dá kompilátoru vedieť či sa majú, alebo nemajú vygenerovať defaultné tabuľky. Defaultné tabuľky sú vygenerované na základe globálnych elementov.
- *owner* parameter určuje vlastníka XML schémy, implicitne je nastavený užívateľ, ktorý registruje schému.
- *options* ďalšie možnosti nastavenia registrácie schémy, ale najdôležitejšia možnosť je REGISTER_BINARYXML, čím povieme procedúre, že chceme XML schému registrovať pre binárne úložisko. Schéma potom už nemôže byť použitá pre ostatné úložiskové modely.

Nasledujúci príklad registruje XML schému *CDShop_ant.xsd* pomocou procedúry *DBMS_XMLSCHEMA.registerSchema*, pričom schéma bude prístupná iba pre vlastníka *SCOTT* a pri registrácii sa vytvoria defaultné tabuľky s objektovými typmi.

```
BEGIN
  DBMS_XMLSCHEMA.REGISTERSCHEMA(
    'http://www.stud.fit.vutbr.cz/~xdvors07/xml/CDshop_ant.xsd',
     sys.UriFactory.getUri(
       'http://www.stud.fit.vutbr.cz/~xdvors07/xml/CDshop_ant.xsd'),
     local => TRUE,
     gentypes => TRUE,
     genbean => FALSE,
     gentables => TRUE,
     force => FALSE,
     owner => 'SCOTT');
END;
```

Príklad č. 24 Registrácia XML schémy

Okrem registrácie schémy balíček DBMS_XMLSCHEMA poskytuje aj procedúry pre mazanie a upravenie (update) XML schémy.

4.3.2 Anotácia XML schémy

Pod pojmom anotácia XML schémy si môžeme prestaviť parametre, ktoré rozširujú XML schému. Tieto parametre pridané v elementoch a atribútoch XML schémy určujú, aké mená (názvy) a dátové typy budú vygenerované pri registrácii schémy pre tabuľky a objektové typy. Pričom vytvorené tabuľky odpovedajú globálnym elementom a objektové typy zase odpovedajú ostatným elementom, atribútom a kolekciám stanovených v XML schéme. Za kolekciu považujeme každý element, ktorého výskyt v XML dokumente je väčší ako jedna, resp. jeho parameter *maxOccurs* je väčší ako jedna. Ak anotácia nie je manuálne vložená v XML schéme, tak sa mená vytvoria automaticky, dedením názvov z elementov schémy.

Najviac používané parametre anotácie v XML schéme :

- *defaultTable* určuje názov východzej tabuľky, generovanej pre každý globálny element v XML schéme.
- *SQLName* určuje názov SQL objektu, ktorý odpovedá každému elementu alebo atribútu definovanému v XML schéme.
- *SQLType* špecifikuje u elementu typu complexType, že sa jedná o SQL objektový typu a u elementu typu simpleType určuje typ SQL atribútu.
- *SQLCollType* používa sa na špecifikáciu VARRAY typu, ktorý predstavuje kolekciu elementov.
- *storeVarrayAsTable* parameter sa nachádza v ROOTovskom elemente XML schémy. Ak je nastavený na TRUE, tak všetky kolekcie budú uložené a manažované ako tabuľky so zoradenou kolekciou.

Pre predstavu uvádzame príklad XML schémy s anotáciou. Anotácia je zvýraznená **hrubo** a určuje, že XML dokument odpovedajúci schéme bude uložený do tabuľky *CUSTOMERS*. Tiež definuje mená a hodnoty ostatných elementov a atribútov. XML schéma nie je v príklade uvedená celá z dôvodu veľkosti schémy. Kompletná XML schéma je uvedená v prílohe.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xdb="http://xmlns.oracle.com/xdb" version="1.0"
xdb:storeVarrayAsTable="true">
  <xs:element name="customers" type="customersType"
xdb:defaultTable="CUSTOMERS"/>
      <xs:complexType name="customersType" xdb:SQLType="CUSTOMERS_T">
            <xs:sequence>
                  <xs:element name="customer" type="customerType"
maxOccurs="unbounded"  xdb:SQLCollType="CUSTOMER_V"
xdb:SQLName="CUSTOMER_VARRAY"/>
            </xs:sequence>
      </xs:complexType>
......
</xs:schema>
```

Príklad č. 25 XML schéma s anotáciou

4.3.3 Mapovanie XML schémy do úložiska XML DB

Mapovaním sa myslí priradenie schémou definovaných elementov a atribútov do tabuliek, objektových typov a SQL atribútov vytvorených v úložisku. Na základe mapovania sa pri vkladaní dokumentu uložia hodnoty elementov a atribútov dokumentu na príslušné miesta. Mapovanie sa vytvorí buď pri registrácii XML schémy, alebo ak už je schéma zaregistrovaná, tak pri vytváraní tabuliek, kde XMLType je asociovaný s XML schémou.

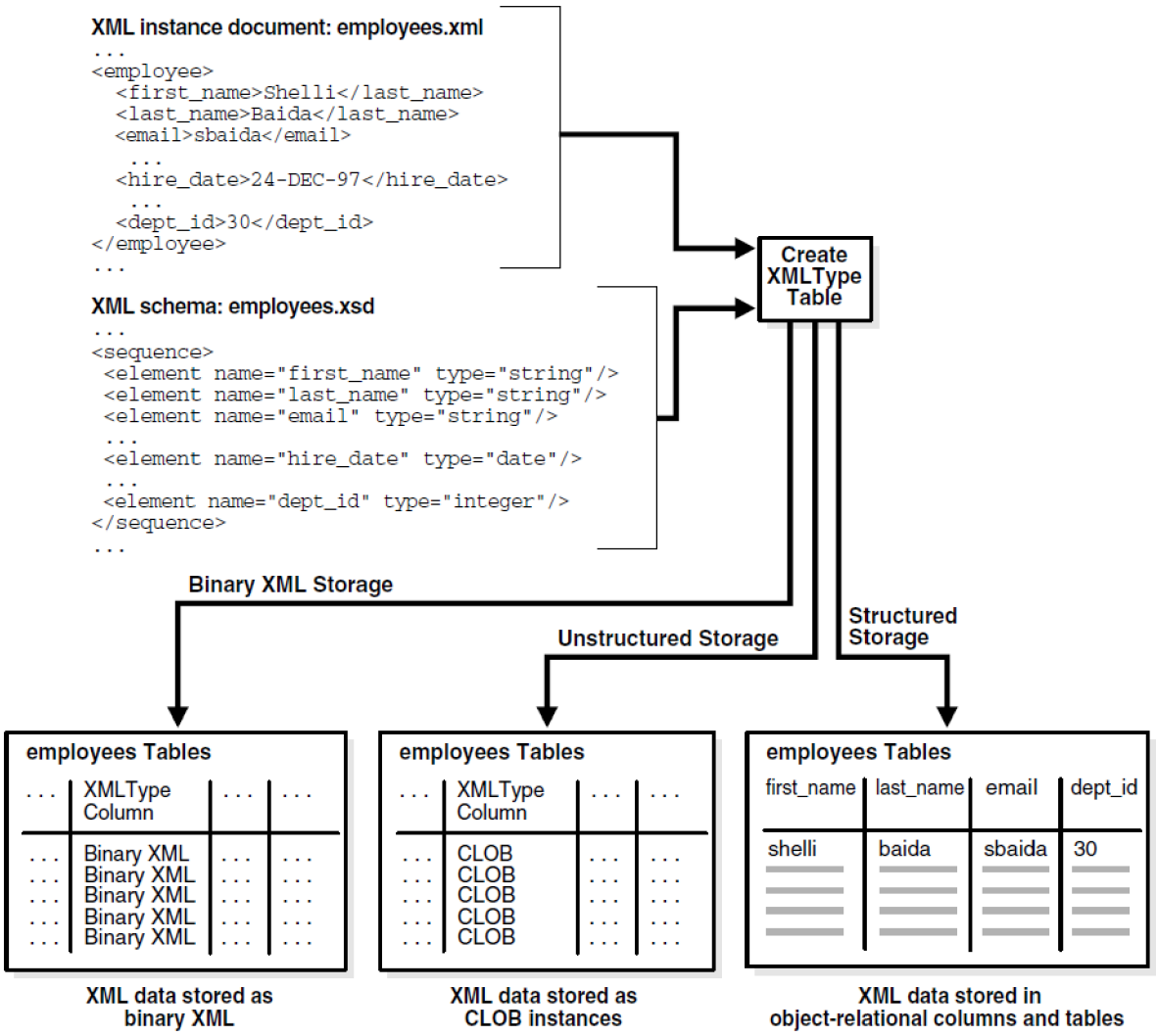
Mapovanie je rôzne pri každom modely úložiska :

- Štruktúrované úložisko dátové typy XML schémy sú priradené k SQL dátovým typom.

- Neštruktúrované úložisko mapovanie je veľmi jednoduché, a to preto lebo všetky XML dáta sú priradené spoločne do jedného dátového typu CLOB.

- Binárne úložisko dátové typy XML schémy sú priradené zakódovaným binárnym XML typom. Zakódovanie závisí od toho, či dáta sú závislé na XML schéme, alebo nie a tiež aj podľa toho, aká schéma je priradená dátam.

Nasledujúci obrázok [5] znázorňuje, ako Oracle XML DB vytvára XMLType tabuľky na základe XML schémy podľa špecifikácie modelu úložiska. Tabuľky špecifikované binárnym, alebo neštruktúrovaným úložiskom ukladajú XML dokumenty do jedného XMLType stĺpca, pričom tabuľka so štruktúrovaným úložiskom ukladá XML dáta do viacerých stĺpcov.



Obrázok č. 4 Ako Oracle XML DB vytvára tabuľky typu XMLType na základe XML schémy

Okrem zmienených možností podpory XML schém, Oracle XML DB poskytuje aj generovanie XML schémy z SQL objektových typov, alebo dotazovanie XML schémy.

Oracle vrátane XML schém podporuje aj DTD popis XML dokumentov a to ako interné DTD (spolu s XML dokumentov), tak aj externé. Podpora DTD popisu však nie je moc rozšírená a preto sa doporučuje previesť DTD popis na XML schému.

4.4 Indexy

Indexy sa vytvárajú z dôvodu zvýšenia výkonu často dotazovanej časti XML dokumentu. Zvýšenie výkonu sa dosiahne zrýchlením prístupu k dátam. Oracle XML DB ponúka niekoľko možností ako vytvoriť indexy nad XML dátami, či už schémovo alebo neschémovo založených a nad všetkými modelmi úložiska. Tieto techniky väčšinou používajú

k pristupovaniu XML dát *OBJECT_VALUE*. OBJECT_VALUE je pseudo-stĺpec, ktorý predstavuje alias stĺpca so XML dátami.

4.4.1 B-tree index

B-tree index je určený pre štruktúrovaný model úložiska. A to z dôvodu dosiahnutia väčšej efektivity indexovania XML dokumentu s drobnou štruktúrou. B-tree index sa vytvára nad tabuľkami a stĺpcami, ktoré odpovedajú elementom a atribútom XML dokumentu.

4.4.2 XMLIndex

XMLIndex je špeciálny index navrhnutý pre XML dáta. Vyznačuje sa tým, že indexuje vnútornú štruktúru XML dokumentu. To je docielené vďaka indexovaniu XML značiek v dokumente a identifikácii fragmentov dokumentu pomocou XPath výrazov. Táto vlastnosť predurčuje index hlavne pre dokumenty uložené ako CLOB. Index sa používa spolu s SQL funkciami.

```
CREATE INDEX xmlindex ON table_CDshop (OBJECT_VALUE)
INDEXTYPE IS XDB.XMLIndex
```

Príklad č. 27 Vytvorenie XMLIndexu nad celým XML dokumentom

4.4.3 Funkčne založené indexy

Ako už názov napovedá jedná sa o indexy vytvorené z funkcií. Funkčne založené indexy sa používajú nad všetkými modelmi úložiska, ale iba v prípadoch, keď XPath výraz funkcie označuje iba jeden uzol. V niektorých prípadoch bývajú funkčne založené indexy aj rýchlejšie ako XMLIndex.

```
CREATE UNIQUE INDEX function_index ON table_CDshop
(extractValue(OBJECT_VALUE, '/customers/customer/@cus_ID'))
```

Príklad č. 28 Vytvorenie funkčne založeného indexu s funkciou extractValue()

4.4.4 Textové indexy

Textové indexy sa vyznačujú rýchlym prístupom k textovému obsahu uzlov. Rýchly prístup sa dosiahne použitím, buď SQL funkcie *contains*, alebo ekvivalentu v podobe funkcie XPath *ora:contains*. Obe funkcie slúžia k tzv. full-textovému vyhľadávaniu nad celým XML dokumentom. Nasledujúci príklad vráti hodnotu uzla *street* z XML dokumentu, iba v tom prípade ak dokument obsahuje znakový reťazec *Juro*.

```
SELECT extractValue(OBJECT_VALUE,
'/customers/customer/adress/street')
FROM table_CDshop
WHERE contains(OBJECT_VALUE, 'Juro')>0
```

Príklad č. 29 Full-textové vyhľadávania pomocou SQL funkcie contains

4.5 DML operácie nad XML dátami

Oracle XML DB v rámci XML podpory zahŕňa aj DML operácie nad XML dátami. Tie zohrávajú dôležitú úlohu pri manipulácii s XML dokumentmi. Operácie, ktoré vkladajú, modifikujú a mažú obsah XML dát, sú podporené viacerými formami.

4.5.1 SQL príkazy s XMLType

SQL príkazy INSERT, UPDATE a DELETE spolu s inštanciou XMLType, môžu manipulovať iba s celým XML dokumentom. Tento spôsob z dôvodu efektivity je určený hlavne pre XML dokumenty, uložené v neštruktúrovanom XMLType úložisku ako CLOB. V uvedenom príklade XML dokument načítaný zo súboru prepíše dokument uložený v tabuľke *customers*, ktorý je identifikovaný v klauzule WHERE.

```
UPDATE customers SET OBJECT_VALUE = XMLType(bfilename('XMLD',
'CDshop_ant.xml'), nls_charset_id('AL32UTF8'))
WHERE existsNode(OBJECT_VALUE,
'/customers/customer[@cus_ID=3]')=1
```

Príklad č. 30 Modifikovanie celého XML dokumentu príkazom SQL UPDATE

4.5.2 DBMS_XMLSTORE

Jazyk PL/SQL podporuje DML operácie balíkom funkcií DBMS_XMLSTORE. Tento balík obsahuje funkcie insertXML, updateXML a deleteXML, ktoré vkladajú, menia a mažú dáta z XML dokumentu uloženom v objektovo-relačnej tabuľke.

4.5.3 SQL funkcie

Najvhodnejšou cestou ako vykonávať DML operácie nad XML dátami je použitie SQL funkcií spolu s príkazom SQL UPDATE. Pomocou SQL funkcií môžeme manipulovať aj s časťami XML dokumentu, čím dosiahneme zvýšenie výkonu u často vykonávaných operáciách. K zmene dát je vytvorená funkcia updateXML(), ktorá môže byť použitá aj na vkladanie XML dát. Kvôli efektívnosti sa však doporučuje použiť SQL funkcie pre vkladanie

insertChildXML(), insertChildXMLbefore(), insertChildXMLafter(), insertXMLbefore(), insertXMLafter() a appendChildXML(). Možnosť mazania dát vo vnútri XML dokumentu poskytuje funkcia deleteXML(). Nasledujúce tri príklady modifikujú obsah XML dokumentu, určeného klauzulou WHERE.

```
UPDATE customers
SET OBJECT_VALUE = updateXML(OBJECT_VALUE,
'/customers/customer/adress/houseNumber/text()','45',
'/customers/customer/adress/street/text()','Povazska',
'/customers/customer/adress/city/text()','Nove Mesto nad Vahom',
'/customers/customer/adress/zipCode/text()','91501',
'/customers/customer/adress/state/text()','Slovenska Republika')
WHERE existsNode(OBJECT_VALUE, '/customers/customer[@cus_ID=2]')=1
```

Príklad č. 31 Modifikácia viacerých uzlov použitím funkcie updateXML()

```
UPDATE customers
SET OBJECT_VALUE = insertXMLBefore(OBJECT_VALUE,
'/customers/customer/items/item[1]',
                XMLType('<item item_ID="3">
                           <title>Hey Stupid</title>
                           <artist>Alice Cooper</artist>
                           <category>Rock</category>
                           <releaseDate>1980-08-21</releaseDate>
                           <price>421</price>
                       </item>'))
WHERE existsNode(OBJECT_VALUE, '/customers/customer[@cus_ID=5]')=1
```

Príklad č. 32 Pridanie celého uzlu za prvým uzlom použitím funkcie insertXMLBefore()

```
UPDATE customers
SET OBJECT_VALUE = deleteXML(OBJECT_VALUE,
'/customers/customer/items/item[3]')
WHERE existsNode(OBJECT_VALUE, '/customers/customer[@cus_ID=5]')=1
```

Príklad č. 33 Vymazanie tretieho uzlu item použitím deleteXML()

4.6 Dotazovanie XML dát

Dotazovanie dát je veľmi dôležitou súčasťou operácií vykonávaných nad XML dátami a zvyšuje efektivitu pri práci s XML dátami. Široká podpora zo strany Oracle XML DB nám opäť ponúka viac riešení ako dotazovať XML dáta uložené v databáze. Ponúkané riešenia môžeme začleniť do viacerých skupín.

4.6.1 Zobrazovanie XML dát XMLType metódami

Tento spôsob slúži skôr z zobrazeniu celého XML dokumentu, uloženého v tabuľke, či stĺpci. Kombinuje SQL príkaz SELECT s jednou z ponúkaných metód inštancie XMLType. Metódy

inštancie iba určujú dátový typ výsledku SQL príkazu. Metódy getStringVal(), getClobVal(), getBlobVal() a getNumberVal() vracajú XML dáta ako typ VARCHAR, CLOB, BLOB, alebo ako NUMBER.

```
SELECT c.OBJECT_VALUE.getStringVal() FROM customers c
```

Príklad č. 34 SQL SELECT s metódou getStringVal()

4.6.2 SQL funkcie pre dotazovanie

Ďalší spôsob dotazovania XML dát je najefektívnejší a najflexibilnejší zo zmienených spôsobov. Klauzuly SELECT, FROM, alebo WHERE SQL príkazu sú doplnené o SQL funkcie štandardu SQL/XML. Parametrami funkcií sú výrazy XQuery a XPath, ktoré identifikujú ciele dotazu. Pre objasnenie tohto spôsobu sú uvedené jednotlivé funkcie spolu s príkladmi použitia.

XMLexists je SQL funkcia definovaná štandardom SQL/XML, ktorá vyhodnocuje existenciu daného uzlu na základe výsledku XQuery výrazu. V prípade že uzol existuje, výsledkom je sekvencia XQuery, v opačnom prípade je výsledok prázdny. Hlavným obmedzením funkcie je, že ju môžeme použiť iba v klauzule WHERE.

```
SELECT c.OBJECT_VALUE.getStringVal()
FROM CUSTOMERS c
WHERE XMLExists('/customers/customer[login="king"]'
PASSING OBJECT_VALUE)
```

Príklad č. 35 SQL funkcia XMLExists

existsNode funkcia je ekvivalentná k predchádzajúcej funkcii XMLexists. Existencia uzla sa však overuje na základe XPath výrazu. V prípade existencie uzla funkcia vracia 1, v opačnom prípade vracia 0. SQL funkcia má aj ekvivalent v podobe XMLType metódy existsNode().

```
SELECT c.OBJECT_VALUE.getStringVal()
FROM CUSTOMERS c
WHERE existsNode(OBJECT_VALUE,
'/customers/customer[login="superman"]')
= 1
```

Príklad č. 36 SQL funkcia existsNode

extract je ďalšou SQL funkciou. Funkcia používa XPath výraz, k určeniu jedného alebo viacerých uzlov, ktoré vráti ako XMLType inštanciu. Extract funkcia má tiež ekvivalent v podobe metódy XMLType extract().

```
SELECT extract(OBJECT_VALUE,
'/customers/customer/items/item').getStringVal()
FROM CUSTOMERS
```

Príklad č. 37 SQL funkcia extract

`extractValue` je XMLType metódou, ktorá je schopná vrátiť len jednu hodnotu listového uzla, identifikovaného XPath výrazom. Ekvivalentom je metóda XMLType extractValue().

```
SELECT extractValue(OBJECT_VALUE,
'/customers/customer/name')
FROM customers
```

Príklad č. 38 SQL funkcia extractValue

XMLCast je SQL funkcia tiež definovaná štandardom SQL/XML. Podobá sa metóde extractValue, len s tým rozdielom, že táto funkcia dovoľuje určiť návratový typ výsledku dotazu.

```
SELECT
XMLCast(XMLQuery('/customers/customer/items/item/title'
PASSING OBJECT_VALUE RETURNING CONTENT)
AS VARCHAR2(100))
FROM customers
```

Príklad č. 39 SQL funkcia XMLCast

Štandard SQL/XML ponúka ešte dve ďalšie SQL funkcie, ktoré sa môžu použiť na dotazovanie XML dát. Keďže funkcie sa na bežné dotazovanie často nevyužívajú, spomeniem iba ich názvy. Jedná sa o funkcie XMLQuery a XMLTable.

4.6.3 Ďalšie spôsoby ako dotazovať XML dáta

Okrem už spomenutých možností dotazovania obsahu XML dokumentov, existuje ešte veľa iných spôsobov. V tejto časti skôr iba naznačíme ďalšie, niektoré skôr alternatívne spôsoby, ako využiť Oracle XML DB k dotazovaniu dát. Jednou z možností je použiť Full-textové vyhľadávanie k dotazovaniu dát. Ďalšou možnosťou je vytvoriť relačný pohľad nad XML dátami a tým vybrať potrebné z nich. Posledným spôsobom je spojiť jazyky XSLT a XPath a tým vytvoriť dávku dotazov.

4.7 Generovanie XML dát z databázy

Generovanie XML dát z databázy je dosť užitočný spôsob ako získať dáta v XML formáte. Oracle XML DB podporuje generovanie hneď niekoľkými spôsobmi. Dáta môžeme generovať z obyčajných objektovo-relačných dát, ale aj z XMLType dát. Zmienim sa o troch najčastejších technikách generovania XML dát z databázy.

4.7.1 Generovanie XML funkciami štandardu SQL/XML

Tento spôsob generovania XML dát je najlepšou variantou, pretože je to najvýkonnejší a najflexibilnejší spôsob z uvedených. Umožňuje nám generovať jeden a viac XML dokumentov, pričom nám dovoľuje určiť výsledný tvar, resp. štruktúru dokumentu. Tieto SQL funkcie sú priamo použité v príkaze SELECT. SQL/XML štandard poskytuje nasledovné funkcie, určené pre generovanie XML dát :

- *XMLElement* vytvorenie elementu
- *XMLAttributes* vloží atribúty do elementu
- *XMLForest* vytvorí "les" elementov
- *XMLAgg* vytvorí jeden element z kolekcie elementov

Uvedený príklad vygeneruje XML dokument z tabuliek *customers* a *items*.

```
SELECT XMLElement("customers",
XMLAttributes(c.id AS "customers_ID"),
XMLForest(c.name AS "name",
          i.title AS "title"))
FROM customers c, items i
WHERE c.id = i.id;
```

Príklad č. 40 Generovanie XML dát funkciami štandardu SQL/XML

4.7.2 Generovanie XML pomocou DBURIType

DBURIType je funkcia SQL, ktorá automaticky vytvorí XML dokument z relačných dát. Možné je vytvorenie iba jedného XML dokumentu a to z riadkov tabuliek, alebo pohľadov. Pričom názov ROOTovského elementu sa dedí z názvu tabuľky, alebo pohľadu. Synovské elementy ROOTovského elementu sú vytvorené z riadkov tabuľky, prípadne pohľadu a majú názov *ROW*. Tieto elementy opäť obsahujú ďalšie synovské elementy, ktoré sú už vytvorené z jednotlivých stĺpcov tabuliek, či pohľadov. Každý z týchto elementov obsahuje textovú hodnotu uzla, ktorá odpovedá hodnote v stĺpci. Tento spôsob je jednoduchý na použitie, avšak neumožňuje ovplyvňovať štruktúru dokumentu.

Nasledujúci príklad prezentuje použitie SQL funkcie DBURIType na generovanie XML dát z tabuľky názov_tabuľky. Použitá metóda getXML vracia dokument ako XMLType inštanciu.

```
SELECT DBURIType('/SCOTT/CUSTOMERS_TAB').getXML() FROM dual
```

Príklad č. 41 Generovanie XML dát pomocou DBURIType

4.7.3 Generovanie XML pomocou DBMS_XMLGEN

Posledný spôsob generovania XML dát z databázy používa balíček DBMS_XMLGEN jazyka PL/SQL, k vytvoreniu XML dokumentu z výsledku SQL príkazu. Funkcie a procedúry obsiahnuté v balíčku umožňujú vytvoriť XML dokument ako hodnotu CLOB, alebo XMLType. Veľké množstvo funkcií a procedúr ponúka veľa možností pri vytváraní dokumentu. Ako napríklad určiť maximálny počet vrátených riadkov, alebo určiť, ktoré riadky sa majú vynechať.

4.8 Ďalšie možnosti balíka Oracle XML DB

Okrem už zmienených možností pre prácu s XML, Oracle XML DB poskytuje ešte omnoho viac. Stručne však spomenieme už iba niektoré popredné technológie z dôvodu, že popis všetkých technológií a možností podpory XML dát je dosť rozsiahly a presahoval by rámec bakalárskej práce.

Menné priestory majú plnú podporu Oracle XML DB, pretože všetky XMLType metódy a SQL funkcie pracujú s XPath, ktorý môže obsahovať prefixy menných priestorov.

Na udržovanie integrity dát Oracle XML DB okrem XML schém ponúka aj možnosť vytvoriť obmedzenia a triggre nad XML dátami.

4.8.1 XSLT Transformácia

Transformovanie obsahu XML dokumentu do iného formátu je vykonané v rámci Oracle XML DB pomocou XSLT procesoru. Ten umožňuje transformáciu priamo v databáze, čím redukuje pamäťové nároky.

Transformovať XML dokument môžeme troma spôsobmi. A to pomocou SQL funkcie XMLtransform, XMLType metódy transform(), alebo pomocou balíka DBMS_XSLPROCESSOR jazyka PL/SQL. Zo všetkých týchto možností je najvýkonnejším spôsobom použitie balíka DBMS_XSLPROCESSOR, a to z dôvodu použitia štýlovej šablóny iba raz. Každá z metód vyžaduje na vstupe XML dokument, ktorý sa má transformovať a štýlovú šablónu, podľa ktorej bude dokument transformovaný. Výsledkom metód XMLtransform a transform() je dokument, ktorý nemusí spĺňať syntax jazyka XML. Typickým príkladom je HTML stránka. Avšak pri použití balíka DBMS_XSLPROCESSOR výsledkom transformácie je vždy dobre vytvorený dokument, čo pri transformácii XML dokumentu na HTML znamená, že výsledkom bude XHTML dokument.

Transformácia uvedená v príklade č. 42 transformuje XML dokumenty z tabuľky *customers* podľa šablóny *oders.xsl* na HTML dokument pomocou SQL funkcie transform.

```
SELECT c.transform(XMLType(bfilename('XMLD', 'orders.xsl'),
nls_charset_id('AL32UTF8'))) FROM customers c
```

Príklad č. 42 XSLT transformácia XML dokumentu na HTML dokument

4.8.2 Pohľady (views)

Oracle XML DB poskytuje dve možnosti pri práci s pohľadmi, ktorých najpodstatnejšie využitie je, že zabaľujú dáta a tým vytvárajú určitú abstrakciu prístupu k dátam.

Jednou možnosťou je vytvorenie klasického relačného pohľadu nad XML dátami. Toto zabalenie XML dát do relačného pohľadu, slúži k relačnému prístupu k XML dátam. Táto možnosť tak sprístupňuje manipuláciu s obsahom XML dokumentu aj bez akýchkoľvek znalostí XML. Relačný pohľad používa výrazy XPath a SQL funkcie k definícii mapovania medzi stĺpcami pohľadu a uzlami XML dokumentu.

Druhou možnosťou je zabaliť objektovo-relačné, alebo XML dáta do XMLType pohľadu. XMLType pohľad je podobný klasickému pohľadu. Každý riadok XMLType pohľadu odpovedá XMLType inštancii. Riadky v pohľade identifikujeme pomocou funkcie extract, použitou spolu s metódou getNumberVal() a aplikovanou na XMLType výsledok. XMLType pohľady rozdeľujeme podľa toho či odpovedajú XML schéme na schémovo a neschémovo založené XMLType pohľady.

Uvedený príklad vytvorí schémovo založený pohľad cus_view typu XMLType z dokumentu, identifikovaného klauzulou WHERE.

```
CREATE OR REPLACE VIEW cus_view OF XMLType
XMLSCHEMA "http://www.stud.fit.vutbr.cz/~xdvors07/xml/orders.xsd"
ELEMENT "customers"
WITH OBJECT ID (extract(OBJECT_VALUE,
'/customers/customer/@cus_ID').getNumberVal())
AS SELECT xml_doc FROM customers
WHERE extractValue(xml_doc, '/customers/customer/login')='mato'
```

Príklad č. 43 Vytvorenie pohľadu typu XMLType

4.8.3 Prepisovanie XQuery a XPath výrazov

Oracle XML DB poskytuje mechanizmus pre prepisovanie XQuery a XPath výrazov za účelom zvýšenia optimalizácie dotazovania XML dát. Tento mechanizmus sa snaží prepísať daný výraz na kód, ktorý sa vykoná priamo nad daným objektom. Optimalizácia dotazov pomocou tohto mechanizmu sa odohráva nad každým modelom úložiska rozdielne.

Najznámejší proces prepisovania XPath výrazov sa používa len pre štruktúrované úložisko a označuje sa ako XPath Rewrite. Tento proces vykonávaný XML DB sa pokúša nahradiť XPath výraz obsiahnutý v SQL/XML funkciách ekvivalentným SQL dotazom.

5 Ukážková aplikácia

Táto kapitola sa zaoberá programovou časťou bakalárskej práce. Programová časť je prezentovaná ukážkovou aplikáciou, ktorú som vytvoril. Zadanie bakalárskej práce nám dalo voľnú ruku nad funkčným zameraním aplikácie. Jediným požiadavkom bolo vytvoriť aplikáciu, ktorá by ilustrovala databázovú podporu XML, formou výukovej aplikácie.

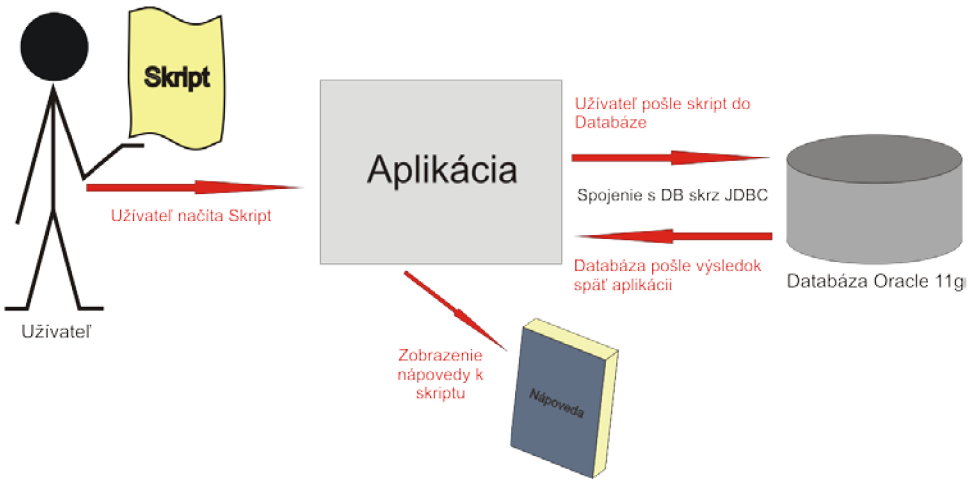
Túto požiadavku sa mi podarilo splniť. Vytvoril som aplikáciu, ktorej použitie v praxi je zamerané práve na výukové účely. Účelom aplikácie je pomôcť užívateľovi pochopiť XML podporu databázového servera Oracle, pričom je demonštrovaná XML podpora popísaná v bakalárskej práci.

Ukážkovú aplikáciu je možné rozdeliť na tri časti : Desktopovú aplikáciu, Ukážkové skripty a Nápovedu.

Pomôcť užívateľovi pochopiť XML podporu servera Oracle je docielené jednotlivými ukážkovými skriptami. Tie tvoria podstatnú časť ukážkovej aplikácie. Skripty sú vytvorené z SQL príkazov a z procedúr jazyka PL/SQL, ktoré demonštrujú prvky XML podpory. Pohodlnú prácu so skriptami poskytuje desktopová aplikácia. Tá tvorí užívateľské rozhranie pre prácu, resp. pre posielanie skriptov do databázy a prijímanie výsledkov z databázy. Aplikácia je doplnená o nápovedu, ktorá pomáha užívateľovi pochopiť význam a funkciu jednotlivých skriptov

5.1 Desktopová aplikácia

Desktopová aplikácia tvorí užívateľské prostredie pre prácu s ukážkovými skriptami. Aplikácia bola vytvorená v jazyku Java. Toto prostredie si môžeme predstaviť ako vzdialenú klientskú konzolu, ktorá sa pripája na databázový server Oracle pomocou JDBC ovládačov. Po pripojení k serveru je možné skripty posielať do databázy. Tá po spracovaní skriptov pošle naspäť výsledok, ktorý sa zobrazí. Tým je užívateľovi demonštrovaná funkčnosť skriptov.



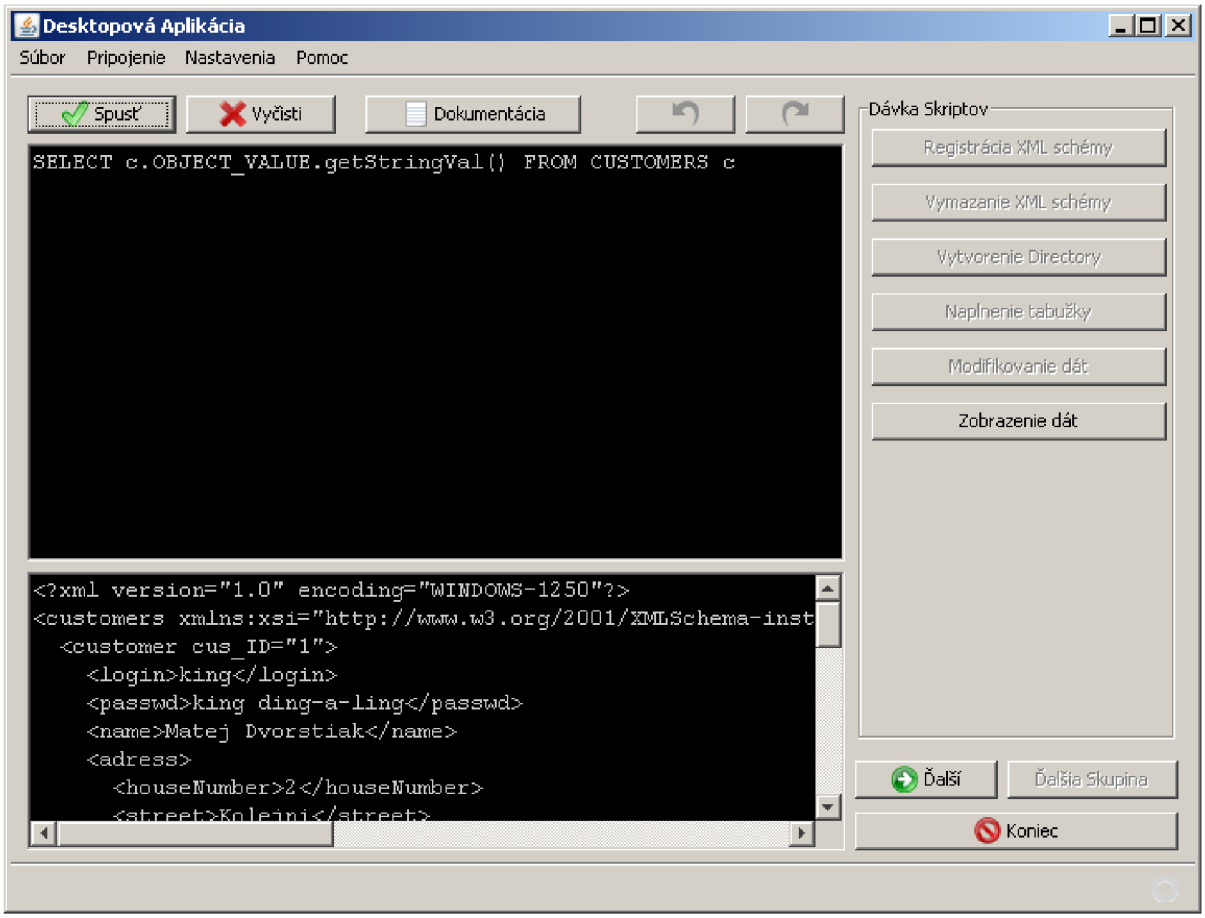
Obrázok č. 5 Princíp funkčnosti aplikácie

5.1.1 Architektúra aplikácie

Keďže sa jedná o desktopovú aplikáciu, je aplikácia vytvorená z okien. Tie sú vytvorené ako *JFrame objekty*. Aplikácia pozostáva z jedného hlavného okna a štyroch vedľajších okien. K tým sa pristupuje výhradne iba z hlavného okna.

Vedľajšie okná boli vytvorené za účelom, odľahčenia hlavného okna od možností a nastavení, ktoré užívateľ tak často nevykonáva, ale aj z dôvodu zlepšenia prehľadnosti aplikácie. Patrí k nim napríklad okno, do ktorého sa vkladajú parametre pripojenia, okno na vytváranie dávok skriptov a tiež aj okno s nastaveniami dávky skriptov.

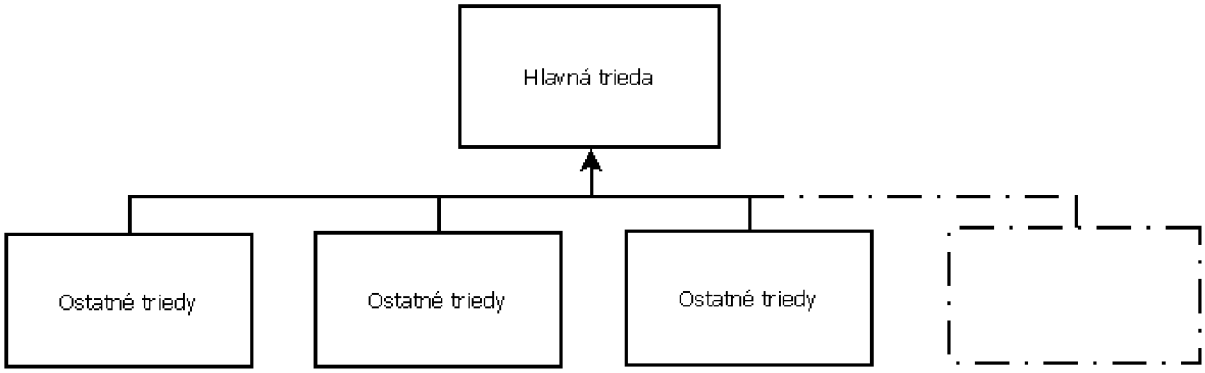
Hlavné okno predstavuje najdôležitejšiu časť aplikácie, pretože práve v ňom sa odohráva práca so skriptami. V rámci zamerania aplikácie, môžem rozdeliť hlavné okno na tri časti a tlačítka pre obsluhu. Dve časti predstavujú textové plochy. Do jednej z nich je načítaný skript a v druhej textovej ploche sa zobrazujú výsledky skriptov poslaných do databázy. Poslednú časť tvorí panel nazvaný *Dávka Skriptov*. V tomto panely sa pri načítaní dávky skriptov zobrazia tlačítka predstavujúce jednotlivé skripty dávky. Nasledujúci obrázok prezentuje hlavné okno aplikácie.



Obrázok č. 6 Hlavné okno aplikácie

V architektúre aplikácie je uplatnený návrhový vzor Singleton – existuje tak jediná inštancia hlavnej triedy aplikácie a z nej sa spúšťa hlavné okno aplikácie. V tejto triede sú deklarované a vytvorené objekty ostatných tried, čím sa stávajú viditeľné aj v iných triedach.

Dôvodom použitia tejto architektúry je prístupnosť vytvoreného databázového spojenia aj v ostatných triedach. Štruktúru a previazanosť tried aplikácie zobrazuje nasledujúci obrázok.



Obrázok č. 7 Štruktúra aplikácie na úrovni tried

5.1.2 Pripojenie sa k databáze

Aby sme mohli pracovať s databázou musíme sa k nej najprv pripojiť. Pripojenie sa vytvorí na základe vložených parametrov pripojenia. Okrem mena a hesla užívateľského účtu je potrebné vložiť aj IP adresu servera, na ktorom beží databáza. Tiež aj unikátny identifikátor databázy (SID) spolu s číslom portu. To je implicitne nastavené na 1521. Pripojenie je vytvorené do tej doby, pokiaľ sa neukončí, alebo pokiaľ neukončíme aplikáciu.

```
/**Nastavenie ovládačov spojenia*/
Class.forName("oracle.jdbc.driver.OracleDriver");
/**vytvorenie spojenia na základe URL adresy, loginu a hesla
užívateľa*/
this.setConn(DriverManager.getConnection(url, login, passwd));
```

Zdrojový kód č. 1 Vytvorenie spojenia

5.1.3 Práca so skriptami

Hlavnou časťou aplikácie je práca so skriptami, a tú môžeme uskutočniť dvojakým spôsobom. Prvý spôsob nám umožňuje prácu s jednotlivými skriptami a druhý so skupinkou skriptov tzv. dávka skriptov.

5.1.3.1 Jednotlivé skripty

Jednotlivé skripty sú uložené ako textové súbory, ktoré sa načítaním zobrazia v aplikácii a sú pripravené k použitiu. Načítané skripty môžeme editovať, alebo si môžeme vytvoriť vlastný skript. Obsah skriptov nie je vôbec kontrolovaný a správnu funkčnosť je možné overiť iba jeho vykonaním. V prípade neúspechu vykonania skriptu, dôjde k vyhodeniu výnimky, ktorá bude zobrazená užívateľovi. V opačnom prípade bude vypísaný vrátený

výsledok. U skriptov ktoré nevracajú výsledky, ako napríklad vytvorenie tabuľky, o úspešnom vykonaní informuje výpis. Po spustení skriptu je daný skript uložený dočasne do pamäti a tým je vytvorená história skriptov. Skripty sú uchovávané v zozname a sú pridávané vždy na koniec zoznamu. Históriu skritov je možné prechádzať. Každý z implicitne vytvorených skriptov je doplnený o príslušnú nápovedu, ktorá sa zobrazí pre načítaný skript stlačením tlačítka *Dokumentácia*.

5.1.3.2 Dávky skriptov

Dávky slúžia k vytvoreniu skupinky skriptov. Tým sa môže prezentovať sled skriptov demonštrujúci určitú časť XML podpory. Napríklad ako ukážka podpory generovania XML dát. Ukážka sa skladá z troch skriptov. Najprv musíme vytvoriť relačné tabuľky, tie následne naplniť dátami. A ako posledné vygenerovať XML dáta z tabuliek.

Dávka skriptov sa skladá z 1 a viac skriptov. Pričom každý skript obsahuje : názov skriptu, vlastný obsah a číslo skupiny. Pomocou čísla skupiny môžeme zgrupovať skripty. Táto možnosť je vytvorená za účelom, aby sa vykonávali jednotlivé skupiny skriptov v určenom poradí. Tým sa dá zamedziť napríklad situácii, keď by sa mali vykonať príkazy naplňujúce tabuľku skôr, ako príkazy, ktoré vytvoria tabuľku. Určenie čísla skupiny skriptu je maximálne flexibilné a záleží na užívateľovi, ako s tým naloží. Nastavenia aplikácie nám však aj povoľujú vypnúť rozdelenie skriptov v dávke na skupiny.

Po načítaní dávky sú jednotlivé skripty zobrazené ako tlačítka v bočnom panely s názvom *Dávka Skriptov*. Ak je nastavené vypnutie rozdelenia skriptov na skupiny, tak sú prístupné všetky tlačítka a môžeme ich prechádzať ľubovoľne. V opačnom prípade, teda ak je zapnuté rozdelenie skriptov na základe čísla skupiny, sú prístupné iba tlačítka reprezentujúce skripty s aktuálnym číslom skupiny. Hneď po načítaní dávky skriptov je aktuálne číslo skupiny najmenšie zo zadaných. Prejdenie na najbližšiu vyššiu úroveň skupín sa dá iba pomocou tlačítka *Ďalšia Skupina*. Čím sa sprístupní iná skupinka skriptov a tá predchádzajúca sa zablokuje. Skupinky skriptov sa dajú prechádzať iba vzostupne. Ukončenie práce s dávkami, vykonáme tlačítkom *Koniec,* pričom sa aplikácia vráti do stavu pred načítaním dávky skriptov.

Dávku môžeme okrem načítania aj editovať, alebo vytvoriť. Dávka so skriptami je uložená ako XML súbor, ktorého štruktúru prezentuje nasledujúci obrázok.

```
  <?xml version="1.0" encoding="ISO8859-2" ?>
- <root>
  - <script>
      <group>0</group>
      <name>Vytvorenie relačnej tabuľky</name>
      <path>X:\Script\\Create Table.txt</path>
    </script>
  - <script>
      <group>1</group>
      <name>Naplnenie tabuľky</name>
      <path>X:\Script\\Insert.txt</path>
    </script>
  - <script>
      <group>2</group>
      <name>Generovanie XML dát z tabuľky</name>
      <path>X:\Script\\GenerateXML.txt</path>
    </script>
  </root>
```

Obrázok č. 8 Štruktúra súboru s dávkou skriptov

5.1.3.3 Spustenie skriptu

Mechanizmus spustenia skriptu je jednoduchý. Po načítaní skriptu stačí stlačiť tlačítko *Spusť*. Skript z textovej plochy sa použije ako parameter metódy *executeQuery()*,ktorá pošle skript databáze a vráti výsledok. Výsledok je vrátený formou objektu *ResultSet*, ktorý skôr prezentuje tabuľkovú podobu. Z stĺpcovej a riadkovej formy výsledku vyberieme dáta a tie zobrazíme. Pri riešení tohto mechanizmu som sa stretol s problémom, ktorý zabral dosť času a úsilia. Nepodarilo sa zobraziť, resp. z výsledku poslaného databázou vybrať XML dáta. Daný problém som nakoniec vyriešili tak, že je potrebné ku každému dotazu, ktorý má vrátiť XML dáta, pripojiť jednu z XMLType metód getStringVal() alebo getCLobVal().
Tým dáme vedieť databáze v akom dátovom type požadujeme výsledné dáta. So zobrazením textového reťazca alebo CLOB už žiadny problém nenastáva.

5.1.4 Prístup k nápovede

Prístup k nápovede je vytvorený prostredníctvom tlačítka *Dokumentácia*. Po stlačení sa otvorí v defaultnom webovom prehliadači nápoveda, resp. tá časť nápovedy, ktorá popisuje načítaný skript. Mechanizmus otvárania nápovedy povoľuje otvoriť nápovedu iba pre skripty, ktoré sme vytvorili, pretože každý súbor, ktorý predstavuje jeden skript, obsahuje okrem samotného skriptu aj časť, ktorá slúži k identifikácii konkrétnej časti nápovedy. Táto časť súboru obsahuje hlavičku a samotný odkaz na nápovedu a je umiestnená v súbore samostatne na prvom riadku. Pomocou hlavičky identifikujeme, či daný skript obsahuje odkaz na nápovedu. Odkaz pozostáva z www adresy, ktorá odkazuje na časť nápovedy zaoberajúcou sa daným skriptom. Pri načítaní skriptu sa tento odkaz načíta zvlášť do premennej. K tejto premennej pristupuje potom mechanizmus otvárania nápovedy. Za identifikátorom nápovedy nasleduje na ďalších riadkoch už samostatný skript.

V prípade, že užívateľ bude chcieť rozšíriť otváranie nápovedy aj pre skripty, ktoré sám vytvoril, bude musieť pridať do súboru so skriptom na prvý riadok hlavičku „url:“ a následne za ňou adresu, ktorá odkazuje na časť dokumentácie. Identifikačná časť súboru môže vyzerať napríklad takto :

```
url:http://www.stud.fit.vutbr.cz/~xdvors07/index.html
```

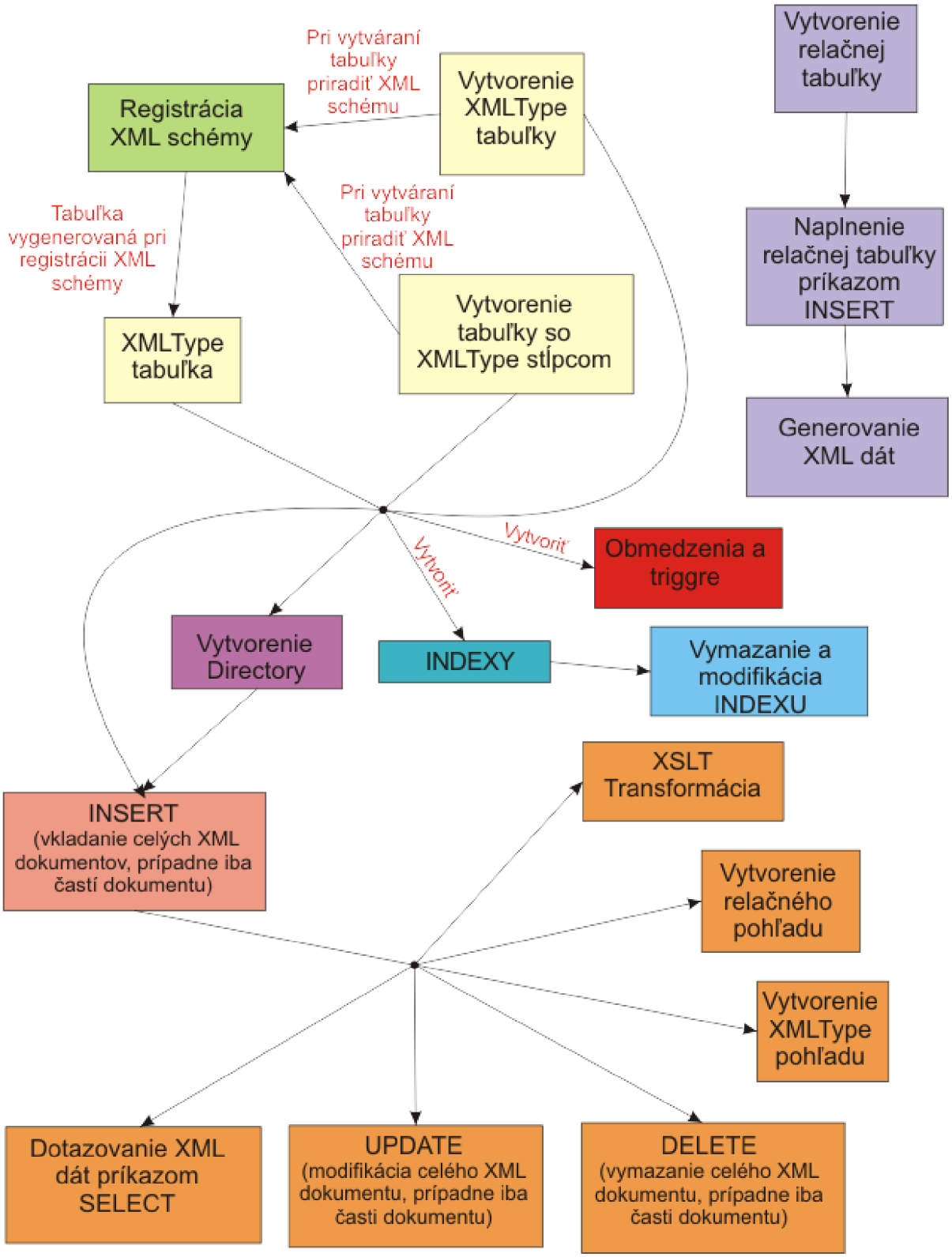
Príklad č. 42 Odkaz na nápovedu umiestnený v súbore

5.2 Ukážkové skripty

Ukážkové skripty sú najdôležitejšou súčasťou programovej časti bakalárskej práce, pretože demonštrujú XML podporu databázového servera Oracle. Pojmom skript označujem SQL príkazy a PL/SQL procedúry a funkcie. Sú zamerané hlavne na podporu popísanú v teoretickej časti práce, kde som ich využil ako príklady. Skripty sú vytvorené pre všetky tri modely úložiska, avšak najviac demonštrujú XML podporu štruktúrovaného úložiska XMLType, pretože štruktúrované úložisko poskytuje najviac pestrých možností ako predviesť XML podporu. Jednotlivé skripty sú uložené ako textové súbory, aby sa mohli jednoducho vytvoriť a editovať v akomkoľvek textovom editore. Názvy súborov som sa snažil vytvoriť tak, aby sa v nich mohol užívateľ jednoducho orientovať. Všetky skripty sú súčasťou CD prílohy a sú rozdelené do troch adresárov nazvaných podľa modelu úložiska.

Každý skript predstavuje iba časť podpory a sám o sebe väčšinou nemá žiadnu výpovednú hodnotu. Skripty boli takto vytvorené z dôvodu, aby sa dali univerzálne použiť (avšak iba v rámci nejakej postupnosti a zoskupenia). Zoskupením myslím jednotlivé modely úložiska. Univerzálne použitie umožňuje užívateľovi vytvoriť kombinácie skriptov a tak nasimulovať rôzne situácie XML podpory.

Nasledujúci obrázok zhruba popisuje, ako by mohol užívateľ použiť skripty, resp., ako ich navzájom nakombinovať, pričom segmenty (obdĺžniky) predstavujú jednotlivé skripty a šípky znázorňujú v akom poradí majú byť skripty použité. Farbou naznačujem, ako by mohli byť priradené skripty do skupín pri použití dávok skriptov.



Obrázok č. 9 Možnosti kombinácie skriptov

Všetky skripty pracujú s XML dokumentom, XML schémou a XSL šablónou uvedenými v prílohe.

5.3 Nápoveda

Nápoveda rozširuje aplikáciu o teoretické poznatky, ktoré som nadobudol pri štúdiu XML podpory. Bola hlavne vytvorená za účelom, aby som užívateľovi priblížili princípy funkčnosti jednotlivých skriptov. Nápoveda je vytvorená ako webová stránka a je dostupná priamo z aplikácie, prostredníctvom tlačítka *Dokumentácia*. Môžeme ju rozdeliť na tri časti :

- Návod k aplikácie
- Teória XML podpory databázového servera Oracle
- Skripty

Časť nápovedy nazvaná *Návod k aplikácii* obsahuje stručnú inštruktáž k aplikácii. Tá oboznámi užívateľa s možnosťami ako pracovať s aplikáciou. Tento sprievodca aplikáciou je doplnený obrázkami pre lepšie vysvetlenie práce s aplikáciou.

Ďalšia časť nápovedy slúži k teoretickému oboznámeniu užívateľa s XML podporou databázového servera Oracle. Informácie k tejto časti som čerpal z teoretickej časti bakalárskej práce.

Posledná, a najdôležitejšia časť nápovedy, sa zaoberá jednotlivými skriptami, pričom okrem samotných skriptov obsahuje aj popis ich fungovania spolu s výstupmi.

5.4 Implementačné nástroje

Na vytvorenie ukážkovej aplikácie som použil viacero implementačných nástrojov. Samotnú desktopovú aplikáciu som napísal v jazyku Java 1.6.0_13 a aplikácia bola vytváraná vo vývojovom prostredí NetBeans IDE 6.5.1., ktoré mi uľahčilo prácu hlavne prepracovaným a dobre užívateľsky prístupným Debuggerom, ale aj rôznymi inými maličkosťami. Komunikácia medzi aplikáciou a databázou je sprostredkovaná ovládačmi JDBC vo verzii thin. Tie sú po registrácii voľne prístupné na stránkach spoločnosti Oracle. Ukážkové skripty som testoval na databáze Oracle 11g, ktorú som si pre pohodlnejšie účely nainštaloval doma. Z produktov databázy Oracle som najviac využil konzolu SQL Plus na overovanie správnosti skriptov, prípadne pre drobné nastavenia a ako napríklad nastavenie práv užívateľov som využil Oracle Enterprise Manager. Pre vytvorenie XML dokumentov, XML schémy a XSLT šablóny som použil editor XMLSpy Enterprise Edition 2008 release 2.

5.5 Požiadavky aplikácie

Pre správny beh a funkčnosť ukážkovej aplikácie musia byť splnené určité požiadavky.

Prvou z nich je výber operačného systému. Aplikácia je zameraná na platformy Windows a Linux. Testovaná bola konkrétne na systéme Windows XP Professional x64 Edition Version 2003 Service Pack 2 a na Linuxovej distribúcii Ubuntu 9.04 Destop Edition.

Ďalším požiadavkom je databázový systém Oracle 11g Release 1 (počas písania tejto práce je to najnovšia verzia) spolu s ORACLE XML DB. Daná verzia je požadovaná z dôvodu použitia niektorých technológií a funkcií, ktoré nie sú obsiahnuté v predchádzajúcich verziách. Aplikáciu som testoval na systéme Oracle 11g Release 1 Enterprise Edition.

Keďže aplikácia vyžaduje pripojenie k databáze, je nutné mať tiež vytvorený užívateľský účet spolu s príslušnými právami v databáze. Ja som využil už implicitne vytvorený účet SCOTT s prístupovým heslom TIGER. Užívateľ tiež musí vedieť IP adresu servera, na ktorom beží databáza Oracle, SID databáza (unikátny identifikátor databázy) a číslo portu, ktoré je implicitne nastavené na 1521.

Súčasťou aplikácie je aj nápoveda, ktorá je vytvorená formou webovej stránky a je nutné ju umiestniť na webový server. My sme uložili nápovedu na školský webový server a adresa nápovedy je http://www.stud.fit.vutbr.cz/~xdvors07/ . Pri akejkoľvek manipulácii s nápovedou, či už pri zmene adresy, alebo pridaním častí nápovedy, je nutné aj pozmeniť jednotlivé odkazy uložené v príslušných skriptových súboroch, napríklad pomocou textového editora.

6 Záver

Bakalárska práca mala za cieľ zoznámiť sa XML technológiou a s podrobnou XML podporou jedného z populárnych databázových systémov. Na základe získaných poznatkov vytvoriť ukážkovú aplikáciu, ktorá by demonštrovala XML podporu pre výukové účely.

Napriek vzrastajúcej popularite jazyka XML a podpory tohto jazyka u databázových systémov, nie je táto oblasť dostatočne spracovaná. Toto poznanie hodnotím na základe dostupnej literatúry a internetových zdrojov, ktorých je v globálnej miere dosť, ale pri riešení konkrétnych princípov a problémov sa mi príslušná literatúra zdá pomerne stručná. Napriek tomu som sa snažil uviesť získané vedomosti čo najzrozumiteľnejšie a najvýstižnejšie v teoretickej časti tejto práce. Uvedené znalosti som doplnil príkladmi z praktickej časti a obrázkami. Najviac poznatkov som nadobudol z príručky, resp. manuálu k Oracle XML DB.

Okrem nadobudnutých teoretických znalostí, som získal aj veľa skúseností pri práci s praktickou časťou bakalárskej práce. Tú predstavuje aplikácia, ktorá ilustruje XML podporu databázového servera Oracle. Najviac prínosnou bola práca s databázou Oracle. Vypracovaním bakalárskej práce som získal veľa znalostí a skúseností, či už o XML technológiách, prehľade XML podpory u komerčných aj nekomerčných databázových systémoch a hlavne XML podpory u databázového produktu Oracle.

Pokračovanie, prípadne rozšírenie vidím vo vytvorení viac interaktívnej aplikácie, ktorá by s užívateľom komunikovala a tým ho nútila viac porozumieť XML podpore.

Literatura

[1] Harold, Elliotte R. & Means, W. Scott: XML v kostce, 1, Praha: Computer Press, 2002, s. 439

[2] Skonnard, Aaron, Gudgin, Martin: XML – pohotová referenční příručka, 1, Praha: Grada Publishing, 2006

[3] Brust, Andrew J., Forte, Stephen: Mistrovství v programování SQL Serveru 2005, 1, Praha: Computer Press, 2007

[4] Sobotka, Jiří: Podpora pro práci s XML u databázového serveru Oracle, Diplomová práca, FIT VUT, Brno, 2006

[5] Adams, Drew: Oracle Developer`s Guide, 11g Release 1, 1, Oracle Corporation, 2008

[6] Ashdown, Lance, Greenberg, Janis, Melnick, Jack: Oracle Developer`s Kit Programmer`s Guide, 11g Release 1, 1, Oracle Corporation, 2008

Internetové zdroje

[1i] W3C Consortium, Extensible Markup Language (XML)
http://www.w3.org/XML/

[2i] Microsoft MSDN Library
http://msdn.microsoft.com/sk-sk/library/bb418432(en-us,SQL.10).aspx

[3i] SyBase Documentation
http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.ase_15.0/title.htm

Príloha A – Obsah priloženého CD

Súčasťou bakalárskej práce je priložený CD-ROM s nasledujúcim adresárovým obsahom :

- Aplikácia adresár s programovou časťou bakalárskej práce.
- Dokumentácia písomná správa bakalárskej práce.
- Obrázky obrázky použité v písomnej správe.

Príloha B – XML dokument

Časť príkladov uvedená v písomnej správe pracuje s priloženým XML dokumentom.

```
<?xml version="1.0" encoding="UTF-8"?>
<customers xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.stud.fit.vutbr.cz/~xdvors0
7/xml/orders.xsd">
	<customer cus_ID="1">
		<login>mato</login>
		<passwd>heslo</passwd>
		<name>Matej Dvorstiak</name>
		<adress>
			<houseNumber>2</houseNumber>
			<street>Kolejni</street>
			<city>Brno</city>
			<zipCode>61200</zipCode>
			<state>Ceska Republika</state>
		</adress>
		<email>Dvorstiak.Matej@seznam.cz</email>
		<phone>123456789</phone>
		<items>
			<item item_ID="1254">
				<title>Knock on the wood</title>
				<artist>Young Gods</artist>
				<category>Industrial</category>
				<releaseDate>2008-10-05</releaseDate>
				<price>550</price>
			</item>
			<item item_ID="562">
				<title>Split</title>
				<artist>Nine Inch Nails</artist>
				<category>Industrial</category>
				<releaseDate>2007-07-12</releaseDate>
				<price>400</price>
			</item>
		</items>
	</customer>
</customers>
```

Príloha C – XML schéma

Uvedená XML schéma patrí k XML dokumentu z prílohy B.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xdb="http://xmlns.oracle.com/xdb" version="1.0"
xdb:storeVarrayAsTable="true">
  <xs:element name="customers" type="customersType"
xdb:defaultTable="CUSTOMERS"/>
  <xs:complexType name="customersType" xdb:SQLType="CUSTOMERS_T">
    <xs:sequence>
      <xs:element name="customer" type="customerType"
maxOccurs="unbounded" xdb:SQLCollType="CUSTOMER_V"
xdb:SQLName="CUSTOMER_VARRAY"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="customerType" xdb:SQLType="CUSTOMER_T">
    <xs:sequence>
      <xs:element name="login" type="stringType" xdb:SQLName="LOGIN"
xdb:SQLType="VARCHAR2"/>
      <xs:element name="passwd" type="stringType"
xdb:SQLName="PASSWD" xdb:SQLType="VARCHAR2"/>
      <xs:element name="name" type="stringType" xdb:SQLName="NAME"
xdb:SQLType="VARCHAR2"/>
      <xs:element name="adress" type="adressType"
xdb:SQLName="ADRESS"/>
      <xs:element name="email" type="stringType" xdb:SQLName="EMAIL"
xdb:SQLType="VARCHAR2"/>
      <xs:element name="phone" type="integerType"
xdb:SQLName="PHONE" xdb:SQLType="NUMBER"/>
      <xs:element name="items" type="itemsType"
xdb:SQLName="ITEMS"/>
    </xs:sequence>
    <xs:attribute name="cus_ID" type="xs:integer" use="required"
xdb:SQLName="CUS_ID" xdb:SQLType="NUMBER"/>
  </xs:complexType>
  <xs:complexType name="adressType" xdb:SQLType="ADRESS_T">
    <xs:sequence>
      <xs:element name="houseNumber" type="integerType"
xdb:SQLName="HOUSE" xdb:SQLType="NUMBER"/>
      <xs:element name="street" type="stringType"
xdb:SQLName="STREET" xdb:SQLType="VARCHAR2"/>
      <xs:element name="city" type="stringType" xdb:SQLName="CITY"
xdb:SQLType="VARCHAR2"/>
      <xs:element name="zipCode" type="integerType"
xdb:SQLName="ZIPCODE" xdb:SQLType="NUMBER"/>
      <xs:element name="state" type="stringType" xdb:SQLName="STATE"
xdb:SQLType="VARCHAR2"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="itemsType" xdb:SQLType="ITEMS_T">
```

```
    <xs:sequence>
      <xs:element name="item" type="itemType" maxOccurs="unbounded"
xdb:SQLCollType="ITEM_V" xdb:SQLName="ITEM_VARRAY"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="itemType" xdb:SQLType="ITEM_T">
    <xs:sequence>
      <xs:element name="title" type="stringType" xdb:SQLName="TITLE"
xdb:SQLType="VARCHAR2"/>
      <xs:element name="artist" type="stringType"
xdb:SQLName="ARTIST" xdb:SQLType="VARCHAR2"/>
      <xs:element name="category" type="stringType"
xdb:SQLName="CATEGORY" xdb:SQLType="VARCHAR2"/>
      <xs:element name="releaseDate" type="dateType"
xdb:SQLName="RDATE" xdb:SQLType="DATE"/>
      <xs:element name="price" type="decimalType"
xdb:SQLName="PRICE" xdb:SQLType="NUMBER"/>
    </xs:sequence>
    <xs:attribute name="item_ID" type="xs:integer" use="required"
xdb:SQLName="ITEM_ID" xdb:SQLType="NUMBER"/>
  </xs:complexType>
  <xs:simpleType name="stringType">
    <xs:restriction base="xs:string">
      <xs:maxLength value="100"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="integerType">
    <xs:restriction base="xs:positiveInteger"/>
  </xs:simpleType>
  <xs:simpleType name="dateType">
    <xs:restriction base="xs:date"/>
  </xs:simpleType>
  <xs:simpleType name="decimalType">
    <xs:restriction base="xs:decimal"/>
  </xs:simpleType>
</xs:schema>
```

Príloha D – XSL šablóna

Nasledujúca šablóna bola použitá v príklade číslo 42.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <meta http-equiv="content-type" content="text/html;
charset=windows-1250"/>
        <title>Zakaznici</title>
      </head>
      <body>
        <h1 align="center">Zoznam Zakaznikov</h1>
        <table border="2" align="center">
          <tbody>
            <tr bgcolor="green" align="center">
              <th colspan="11">Zakaznici</th>
            </tr>
            <tr bgcolor="blue">
              <th>ID</th>
              <th>Login</th>
              <th>Password</th>
              <th>Name</th>
              <th>House Number</th>
              <th>Street</th>
              <th>City</th>
              <th>Zip Code</th>
              <th>State</th>
              <th>Email</th>
              <th>Phone</th>
            </tr>
            <xsl:for-each select="/customers/customer">
              <xsl:sort select="name"/>
              <tr>
                <td><xsl:value-of select="@cus_ID"/></td>
                <td><xsl:value-of select="login"/></td>
                <td><xsl:value-of select="passwd"/></td>
                <td><xsl:value-of select="name"/></td>
                <td><xsl:value-of select="adress/houseNumber"/></td>
                <td><xsl:value-of select="adress/street"/></td>
                <td><xsl:value-of select="adress/city"/></td>
                <td><xsl:value-of select="adress/zipCode"/></td>
                <td><xsl:value-of select="adress/state"/></td>
                <td><xsl:value-of select="email"/></td>
                <td><xsl:value-of select="phone"/></td>
              </tr>
            </xsl:for-each>
          </tbody>
        </table>
```

```
        <table border="2" align="center">
          <tbody>
            <tr bgcolor="yellow">
              <th colspan="6">Polozky</th>
            </tr>
            <tr bgcolor="red">
              <th>ID</th>
              <th>Title</th>
              <th>Artist</th>
              <th>Category</th>
              <th>Release Date</th>
              <th>Price</th>
            </tr>
            <xsl:for-each select="/customers/customer/items/item">
              <xsl:choose>
                <xsl:when test="price &gt; 500">
                  <tr bgcolor="pink">
                    <td><xsl:value-of select="@item_ID"/></td>
                    <td><xsl:value-of select="title"/></td>
                    <td><xsl:value-of select="artist"/></td>
                    <td><xsl:value-of select="category"/></td>
                    <td><xsl:value-of select="releaseDate"/></td>
                    <td><xsl:value-of select="price"/></td>
                  </tr>
                </xsl:when>
                <xsl:otherwise>
                  <tr>
                    <td><xsl:value-of select="@item_ID"/></td>
                    <td><xsl:value-of select="title"/></td>
                    <td><xsl:value-of select="artist"/></td>
                    <td><xsl:value-of select="category"/></td>
                    <td><xsl:value-of select="releaseDate"/></td>
                    <td><xsl:value-of select="price"/></td>
                  </tr>
                </xsl:otherwise>
              </xsl:choose>
            </xsl:for-each>
          </tbody>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```