VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

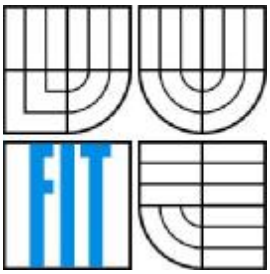
PODPORA PRO PRÁCI S XML V MODERNÍCH DATABÁZOVÝCH SYSTÉMECH

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Jan Kaňovský
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

PODPORA PRO PRÁCI S XML V MODERNÍCH DATABÁZOVÝCH SYSTÉMECH

SUPPORT FOR XML IN DATABASE SYSTEMS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Jan Kaňovský
AUTHOR

VEDOUCÍ PRÁCE — Doc. Ing. Jaroslav Zendulka, CSc.
SUPERVISOR

BRNO 2009

Abstrakt

Tato bakalářská práce je zaměřena na podporu práce s XML v moderních databázových systémech. Jako moderní databázový systém je představen databázový systém Oracle. Práce ukazuje oba nástroje, jejich spolupráci při vytváření aplikací a vzájemné propojení. V první řadě je představena technologie XML a způsob práce s touto technologií. Poté je představen systém Oracle. U představení Oracle databázového systému je velká pozornost věnována Oracle XML DB Repository. Oracle XML DB Repository je určen pro práci s XML. Bakalářská práce se soustředí na architekturu Oracle XML DB Repository a využití rysů Oracle XML DB Repository při vytváření aplikací. Teoretické znalosti jsou podloženy příklady. Součástí bakalářské práce je ukázková aplikace, která využívá teoretických poznatků získaných při studium.

Abstract

This Bachelor's thesis is oriented on support for XML in database systems. Database system Oracle is show as moderm database systems. Bachelor's thesis shows both tools XML and Oracle database system, their cooperation in creation applications and in interconnection. Primarily is show XML technologies and operating principle with this technologies. Then is show database system Oracle. When Oracle database system is showing, so the big attention is paying to Oracle XML DB Repository. Oracle XML DB Repository is determined for working with XML. Bachelor's thesis is oriented in architecture Oracle XML DB Repository and using features Oracle XML DB Repository in creation applications. Knowledge base are precedented by examples. Applications, which belong to bachelor's thesis, use knowledge base that are gets by studying.

Klíčová slova

XML, databáze, Oracle databáze 11g Release 1, SQL , XML Schema, datový typ XMLType, datové XML dokumenty, dokumentové XML dokumenty, Oracle XML DB Repository, Oracle XML DB, Oracle XML DB Content Connector, JCR, JDBC, Nativní XML databázové systémy

Keywords

XML, databases, Oracle database 11g Release 1, SQL, XML Schema, data type XMLType, data centric XML documents, documents centric XML dokument, Oracle XML DB Repository, Oracle XML DB, Oracle XML DB Content Connector, JCR, JDBC, Native XML database systém

Citace

Kaňovský Jan: Podpora pro práci s XML v moderních databázových systémech, bakalářská práce, Brno, FIT VUT v Brně, 2009

Podpora 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.

……
Jan Kaňovský
11.5.2009

Poděkování

Děkuji svému vedoucímu práce panu Doc. Ing. Jaroslava Zendulky, CSc. za poskytnutí odborných informací, cenných rad a připomínek. Dále také panu ing. Jaroslavovi Rábovi za ukázku instalace databázového serveru Oracle.
Dále mé rodině a přátelům, kteří mě v psaní této práce podporovali.
Všem těm, ještě jednou díky.

Obsah

1 Úvod

V dnešní době rozvoj v oblasti komunikačních a informačních technologií zaznamenává velký vzestup. Osobní počítače se stávají nedílnou součástí života lidí.

Softwarové firmy produkují nejrůznější programy, které mohou běžet na nejrůznějších platformách. Potřeba výměny dat mezi aplikacemi si vynutila vznik jednotného nástroje, který je nezávislý na použité platformě, a který umožňuje univerzální popis dokumentů s daty.

Vznik jazyka XML byl významným prvkem vývoje popisných jazyků. XML se stalo revolucí při popisování dokumentů. Jazyk XML se dostal do různých oblastí lidského působení. Byl přijat v obchodování (B2B), finančnictví, pojišťovnictví, justici či zdravotnictví. Přijetí XML s sebou nese potřebu uchovávání informací. S množstvím uživatelských dat a dokumentů, které jsou produkovány těmito oblastmi, souvisí databáze a jejich použití na skladování dokumentů.

Pouhé skladování XML dokumentů v databázích by však při vyvíjení uživatelsky příjemných aplikací nestačilo. V databázích vznikají rozšíření, které jsou schopna s XML dokumenty pracovat nativně (rozumí vnitřní struktuře XML dokumentu). Databáze poskytující nativní uložení lze označit za XML databáze.

Mezi databáze používající nativní uložení lze zařadit databázovým systém od firmy Oracle. Po dohodě s vedoucím práce jsem zvolil jako systém řízení báze dat (SŘBD), na který se z hlediska podpory pro praci s XML daty zaměřím, Oracle Database 11g Release 1 (11.1.0.7.0).

Konkrétně se soustředím na Oracle XML databázové úložiště (Oracle XML DB Repository), které podporuje práci s XML.

Čtenář by po přečtení této bakalářské práce měl mít přehled o technologii XML a měl by se naučit základní práce s XML. Měl by získat představu o podpoře práce s XML v databázovém systému Oracle a měl by získat hlubší znalosti v oblasti Oracle XML DB. Dále by měl získat představu o možnostech ukládání v Oracle XML DB a detailně se seznámit s některými částmi Oracle XML DB Repository. Text práce je doprovázen ukázkovými příklady, na kterých je vysvětlena práce s XML v Oracle databázi. Celý text je rozčleněn do několika kapitol.

Úvodní kapitola, označená číslem (2), je věnována technologii XML, jejímu historickému vývoji a směru jímž se XML ubírá. Kapitola je také věnována cílům, které se XML snaží naplnit a také je věnována základní syntaxi XML dokumentu.

Třetí kapitola se věnuje XML technologiím ve spojení s databázovými systémy. Ve třetí kapitole jsou vysvětleny rozdíly mezi jednotlivými typy XML dokumentů. Dále vytváří pohled na rozdělení databázových systémů, jež poskytují podporu pro XML technologie, a uvádí představitele těchto systémů. Třetí kapitola také poskytuje informaci jaké typy XMLdokumentů databázové systémy zpracovávají.

Čtvrtá kapitola se věnuje podpoře práce XML v databázovém systému Oracle. Představuje Oracle XML DB, který podporu XML zajišťuje. Dovíme se z ní o architektuře Oracle XML DB a o možnostech ukládání XML obsahu do databáze. Dále čtvrtá kapitola představuje jazyk XML Schema, který vytváří schémata pro validaci XML obsahu v databázi. V poslední části této kapitole je zmíněno programové spojení s Oracle XML DB.

Pátá kapitola se věnuje jedné částí Oracle XML DB. Konkrétně představuje přehled o Oracle XML DB Repository. Oracle XML DB Repository je úložiště databáze, které představuje obsahově orientovaný přístup k databázi. Dozvíme se o architektuře Oracle XML DB Repository a o způsobu správy dat (ukládání, přístup, mazání) v Oracle XML DB Repository. Věnuji se v ní také přístupu k databázi pomocí JCR (java content repository), který používá jedna část úložiště nazvaná Oracle XML DB Content Connector.

Šestá kapitola spojuje obě části bakalářské práce, tedy dokumentační a programovou. V této kapitole je popsán způsob řešení ukázkové aplikace s ohledem na dokumentační část. Nabízí přehled o použitých nástrojích k vývoji aplikace, o postupu spuštění aplikace a také nabízí vysvětlení jednotlivých částí aplikace.

Závěrečná kapitola shrnuje celou bakalářskou práci. Kapitola obsahuje zhodnocení dosažených výsledků a směr jakým by se ubíral další vývoj bakalářské práce. Dále pak kapitola obsahuje osobní zhodnocení přínosu.

2 XML technologie

XML, rozšiřitelný značkovací jazyk (Extensible Markup Language), je jednoduchý textový formát pocházející z jazyka SGML. XML je standardem schváleným konsorciem W3C (World Wide Web) a je určený pro značkování dokumentů, výměnu dat mezi aplikacemi a pro publikování dokumentů.

Jazyk popisuje strukturu dokumentu z hlediska věcného obsahu jednotlivých částí. Nezabývá se konkrétním vzhledem dokumentu ani jeho částí. Pomocí různých stylů můžeme provést transformaci do jiného typu dokumentu nebo do jiné struktury XML.

2.1 Historie a vývoj XML

V šedesátých letech řešila firma IBM problém ukládání velkého množství právních dokumentů [3]. Cílem jejich řešení měl být formát, který by měl dlouhou životnost, a který by nebyl závislý na používaných programech. Řešením se stal obecný značkovací jazyk. Postupným rozšiřováním se z něj v roce 1986 stal jazyk SGML (Standard Generalized Markup Language). SGML bylo velice flexibilní, aby vyhovělo různým požadavkům. Pro jazyk SGML to byla velká přednost, ale zároveň také největší slabina. Vývoj aplikací plně podporující SGML byl příliš nákladný, a proto se používal především na velkých projektech.

První masově rozšířenou aplikací SGML byl jazyk HTML. Jazyk HTML měl z počátku malou množinu tagů, které umožňovaly vytvářet hypertextové dokumenty. s vývojem prohlížečů přišly problémy. Výrobci se snažily rozšířit jazyk HTML, což vedlo k nekompatibilitě mezi prohlížeči. Řešením problému s kompatibilitou bylo vytvářet více verzí stránek pro různé prohlížeče. Konsorcium W3C se snažilo tomuto trendu zabránit. Pomocí SGML definovalo, které tagy a kde mohou webové stránky obsahovat. Vznikly tak jazyky HTML 2.0 a HTML 4.0.

Složitost SGML vedla k útlumu jeho používání, a proto skupina odborníku začala pracovat na odlehčené verzi SGML. Výsledkem byla v únoru 1998 první verze XML 1.0, která zaznamenala úspěch. Byla odstraněna složitost SGML. Flexibilita zůstala zachována, a tak XML umožňuje na rozdíl od HTML tvorbu dokumentu s vlastními tagy a s možností validace dokumentů oproti definici typu dokumentu (DTD).

XML se dočkala rozšiřujících standardů. Objevily se XML Namespaces (jmenné prostory), které definují „logické prostory“ jmen, tedy elementů a atributů v XML dokumentu. Uzlům ve stromu dávají „třetí dimenzi“. Logickému prostoru jmen odpovídá jeden globální jednoznačný identifikátor, daný URI. Práce s XML se dále rozšířila o XSL (eXtensible Stylesheet Language). Jedná se o stylové šablony aplikace umožňující transformaci XML dokumentu do tvaru zobrazitelného v internetovém

prohlížeči. Následovalo rozšíření, které umožňuje propojit XML dokument s jiným dokumentem nebo hypertextovou sítí. V tomto rozšíření hovoříme o standardech XLink, XPath a XPointer.

Vývoj XML se dočkal nástrojů DOM a SAX. DOM definuje hierarchii objektů, pomocí které jsme schopni popsat celý XML dokument. Používá se v aplikacích s opakovaným přístupem k XML dokumentu. SAX je jednoduché API pro XML. Využívá událostmi řízený model pro zpracování XML dokumentů.

2.2 Hlavní cíle XML

Jaké byly hlavní cíle tvůrců jazyka XML [1]. Cíle použití a technické vlastnosti jazyka XML shrnuli tvůrci do následujících bodů:

- *XML má podporovat co možná nejvíc aplikací*
- *XML má být kompatibilní s jazykem SGML*
- *XML má být lehce rozšiřitelný a široce použitelný na internetu*
- *Tvorba programů určených pro zpracování XML by měla být jednoduchá*
- *Množství volitelných vlastností XML redukovat na malé množství*
- *XML dokument by měl být dostatečně dobře čitelný pro uživatele*
- *XML návrh by měl splňovat rychlost, stručnost*
- *Tvorba XML dokumentů by měla být snadná*

Řada výhod, které z XML vyplívají se odráží na oblíbenosti a velké míře použití v informačních technologiích dneška.

2.3 Výhody XML

Rysy XML technologie se staly velkým přínosem pro informační technologie. Po nastudování textů k této problematice jsem došel k názoru, že výhody XML zmiňované v různých knihách a na internetu si jsou dosti podobné. Výhody a přínosy XML [2] jsou velmi praktické a mají široké uplatnění:

- *univerzální datový formát* – Relační databázové systémy jsou vhodné pro ukládání dat, které jsou strukturované a se kterými je snadná manipulace.
- *podpora internacionalizace* – podporuje standardní znakovou sadu *UNICODE* [3].
- *význam dat je nedílnou součástí dat* – XML dokumenty nepotřebují logiku, která by z venku vytvářela pohled na význam datových bloků v dokumentu. Pokud je docíleno sjednoceného formátu pro XML dokumenty v určité oblasti, pak se zvýší účinek automatického zpracování.
- *formát budoucnosti* – XML podporuje řada softwarových firem [4].

2.4 Základy syntaxe XML

XML dokument je textový soubor (obvykle s příponou `.xml`), jehož jednotlivé části (*elementy*), jsou vyznačeny značkami - *tagy*. Od ostatního textu jsou tagy (obdobně jako v HTML) odlišeny znaky **<** a **>** [5].

XML dokument by mohl vypadat takto:

```
<?xml version="1.0" encoding="windows-1250" standalone="no"?>
<!DOCTYPE faktura SYSTEM "faktura.dtd">
<faktura>
    <cislo>identifikační číslo faktury</cislo>
    <castka>celkova cena faktury</castka>
    <cisloUctu>číslo účtu</cisloUctu>
</faktura>
```

Jeho obsah je následující:

- *XML deklarace* – označuje, že se jedná o XML dokument. Deklarace je umístěna na začátek prvního řádku dokumentu. Může obsahovat vymezení tří parametrů XML dokumentu. Vymezuje verzi (`version`), kódování (`encoding`) a samostatnost (`standalone`). Uvedení verze je povinné. Přiřazované hodnoty se uzavírají do uvozovek. Pro hodnotu kódování lze použít různé standardní znakové sady. Pokud není uveden parametr `encoding`, pak dokument používá UTF-8 [1]. Parametr standalone indikuje, zda lze dokument použít i bez externí DTD. Bez použití parametru `standalone` je tato hodnota `"no"`.
- *Deklarace typu dokumentu (DOCTYPE)* - přiřazuje danému XML dokumentu použitou definici typu dokumentu *(DTD)*.
- *Obsah vyznačený XML tagy* – tagy dělí informace z XML dokument na elementy. Obsah XML dokumentu v našem případě odpovídá rozsahu od třetího řádku dále.

O pravidlech zápisů tagů, jejich atributech a o základní práci s XML dokumenty se můžete dočíst v dokumentaci k XML [1] nebo na internetových stránkách Tomáše Pitnera, věnovaným značkovacím jazykům [6].

3 XML a databázové systémy

Požadavkem na zpracování XML dat je perzistentní ukládání a následné zpracování dat [7]. XML data se ukládají do perzistentních úložišť, které jsou založena nebo využívají souborový systém, relační databáze, objektový model či nativní XML úložiště.

Popis struktury XML dokumentů má mnoho relačních rysů. Patří mezi ně jazyk XML Schema či posloupnosti prvků, datové typy, identifikátory a dědičnost. XML technologie jsou podobné databázovým technologiím (dotazovací jazyky, schémata, datové typy).

Spojení XML a databází nám umožňuje rozšířit XML technologie o databázové mechanismy, jakými jsou indexy či transakce.

3.1 Datové a dokumentové XML dokumenty

Existují dva typy XML dokumentů. Dokumenty zaměřené na data a druhým typem jsou XML dokumenty zaměřené na dokumenty. Datové dokumenty obsahují data (faktury, objednávky) a dokumentové obsahují ve svém obsahu dokumenty (knihy, manuály, dokumentace). Neexistuje přesně definovaná hranice mezi těmito dvěma typy dokumentů. Rozhodnutí, zda dokument patří do té či oné skupiny, ovlivňují různá kritéria. Jednoduchým kritériem je poměr metadat ke skutečným datům XML dokumentu. Pokud metadat (tagy, atributy) bude vzhledem ke skutečným datům malé množství, pak lze takové XML dokumenty označit za dokumentově zaměřené.

3.1.1 Datové XML dokumenty

Datově zaměřené dokumenty jsou dokumenty, které používají XML jako vhodnou formu pro zápis dat určených k přenosu. Informace z těchto dokumentů používají převážně jiné programy. Nejsou vhodné pro zpracování lidmi díky špatné čitelnosti či počtu elementů, které datové XML dokumenty obsahují. Životní cyklus dat je takový, že jsou data na počátku v nějaké aplikaci či databázi, poté jsou pro přenos zabalena do obálky XML a přenesena do jiné databáze či aplikace [8]. Význam XML formy dat je druhotný, hlavní jsou přenesená data.

3.1.2 Dokumentové XML dokumenty

Dokumentově zaměřené XML dokumenty jsou dokumenty, které jsou vytvářeny pro lidskou potřebu [10]. Například knihy, emaily. Vyznačují se méně pravidelnou nebo nepravidelnou strukturou.

3.2 Druhy databázových systémů s ohledem na XML

V dnešní době je celá řada databázových systémů, které pracují s jazykem XML. Ronald Bourret, který se věnuje databázovým systémům ve spojení s XML je v této oblasti uznávaným odborníkem. Přibližme si Bourretovo rozdělení databázových prostředků, tak jak jej definuje na svých stránkách [10]. Pokud začínáme vytvářet softwarový produkt, pak je dobré projít si následující rozdělení a vybrat si ten, který bude nejlépe odpovídat vaší specifikaci.

3.2.1 Databázový systémy s podporou XML (XML – Enabled Databases)

Databázové systémy s podporou XML obsahují rozšíření pro přenášení dat mezi XML dokumenty a jejich vlastní datovou strukturou. Používají je aplikace zaměřené na datové dokumenty. Databáze také podporují nativní XML uložení.

Rozdíl mezi XML – enabled a nativními uložením je ten, že XML – enabled uložení používá specifické schéma struktur, které musí být mapovány do dokumentů XML při návrhu. Nativní XML uložení používá obecné struktury uložení, které mohou být obsaženy v každém XML dokumentu.

Představitelé tohoto typu databází jsou například *Access 2007*, *DB2* nebo FoxPro.

3.2.2 Nativní XML databázové systémy

Databázové systémy, které ukládají XML v přirozené podobě, se nazývají nativní XML databázové systémy. Existence dokumentově zaměřených XML dokumentů zapříčinila vznik nativních XML databází. Hlavním úkolem nativních XML databází je uložit XML dokument včetně jeho logické struktury a podoby [11]. Dokument pak zpětně obdržíme přesně v takové podobě, v jaké jsme jej předali databázi.

Nativní XML databáze nám přináší možnost indexování uložených dokumentů. Zvýší se tím výkon a možnost provádění dotazů napříč sadou dokumentů umístěných v kolekci.

Ze zástupců můžeme zmínit komerční systémy *Birdstep RDM Mobile*, *Natix* či *Tamino*. Mezi volně dostupné pak řadíme *Berkeley DB XML*, *myXMLDB*.

3.2.3 Middleware

Middleware je software používaný aplikacemi, které jsou orientovány na datové XML dokumenty. Aplikace přenášejí data mezi XML dokumenty a databázemi. Middleware funguje jako

prostředník mezi dvěmi aplikacemi a databází. Zajišťuje komunikaci s databází nejčastěji pomocí *JDBC* nebo *ODBC*.

V knize *Service Orient or Be Doomed!* [12] se píše, že middleware je ve své podstatě lepidlo, které drží všechny systémy pohromadě tak, aby plnily potřeby businessu. Ale také dodávají, že lepidlo je dobré, pokud lepíte dohromady dva kusy. Pokud lepíte mnoho kousků a používáte hodně lepidla, vznikne vám nepořádek.

Mezi ně lze zařadit Oracle XML Developer's Kit (XDK), Oracle XML Query Service či ADO od firmy Microsoft.

3.2.4 Wrappery

Podle Ronalda Boureta [11] je wrapper systém, který zachází s XML dokumentem jako zdrojem relační databáze. Data XML dokumentu jsou tak reprezentována v relační formě, nejčastěji v databázových tabulkách. Nad takovými tabulkami lze provádět příkazy z jazyka SQL.

Mezi zástupce jsou řazeni *DB2* od firmy *IBM* nebo *SQL Server* od firmy *Microsoft*.

3.2.5 Systémy pro správu obsahu

Systémy pro správu obsahu (Content Management System – CMS) jsou převážně aplikace vystavěné na nativních XML databázích nebo souborových systémech pro obsahovou správu. Vyznačují se prvky jako je správce verzí, vstupní a výstupní kontrola a editování dokumentů. Podporují distribuci, publikování a zpřístupňování informací. Uživateli většinou zůstávají tyto systémy ukryty a pracují s nimi pouze na povrchu. Zde patří například objektově orientovaný *CMS* od firmy Sorman.

4 Oracle XML DB – XML v Oracle

4.1 Přehled Oracle XML DB

Oracle XML DB považuje XML dokument za přirozený datový typ databáze. Oracle XML DB není samostatný server. XML datový model zahrnuje společně strukturovaná data a nestrukturovaný obsah. Aplikace mohou používat standardní SQL a XML příkazy na generování složitých XML dokumentů z SQL dotazů a uložených XML dokumentů.

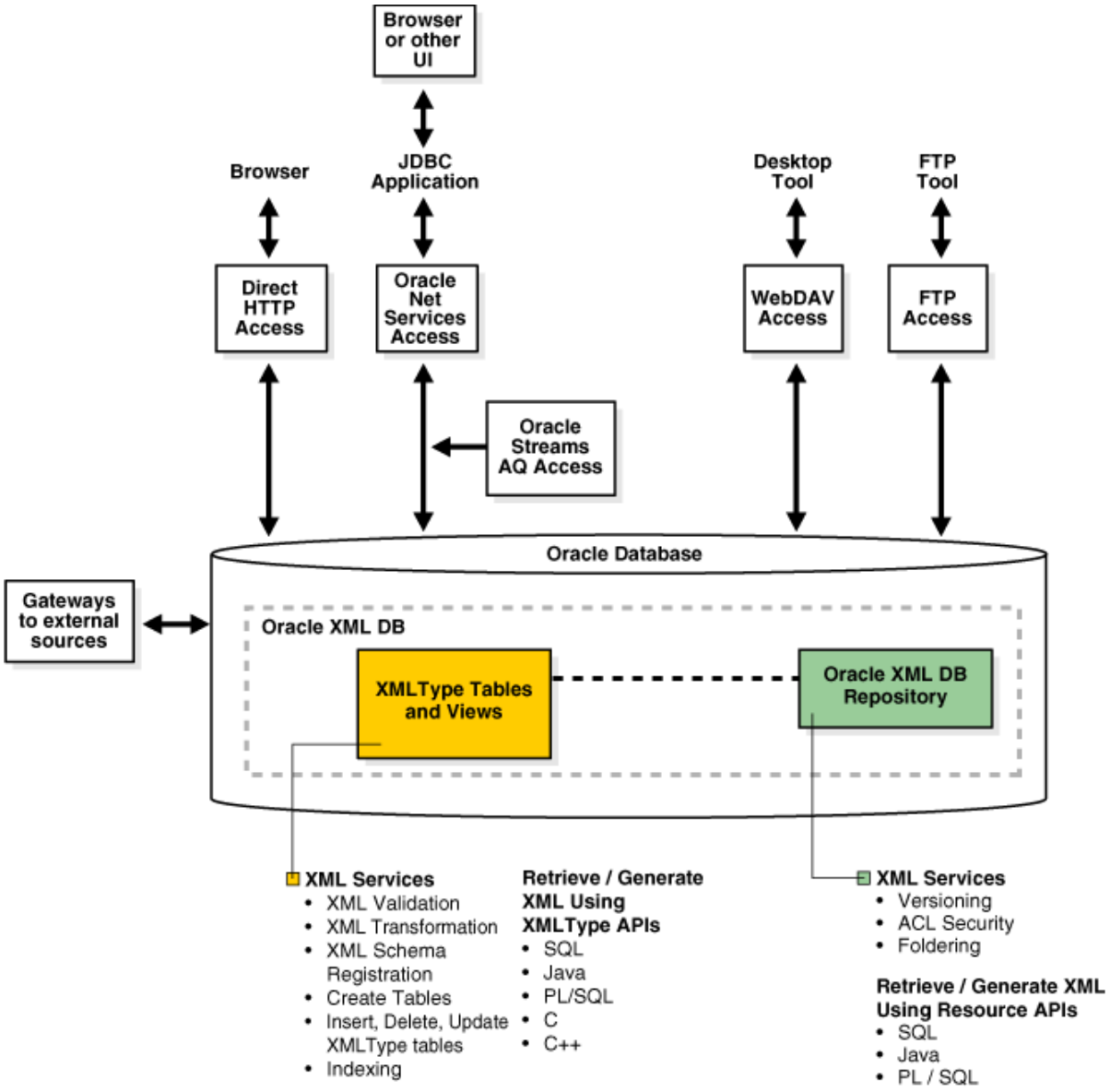
Oracle XML DB nám dává možnosti ať už dokumentově nebo datově orientovaného přístupu. Pokud se chceme dívat na XML jako na dokumentově zaměřené XML dokumenty (články, knihy), pak Oracle XML DB nabízí XML repository (úložiště). Úložiště je přístupné pomocí standardních protokolů a SQL. Mezi prostředky, pomocí nichž lze přistupovat k úložišti, patří také JCR API (Java Content repository API).

Máme-li strukturované XML dokumenty, tj. datově zaměřené XML dokumenty (adresy, objednávky, faktury), pak pro uložení použijeme XMLType. XMLType je pro databázi přirozeným datovým typem a poskytuje podporu pro XML Schema, XPath, DOM a další nástroje. Datově orientovaný přístup je více využíván a také se mu věnuje souběžná bakalářská práce pana Dvorštiaka.

Oracle XML developerr's kits (Oracle XML vývojářská výbava) obsahuje základní nástroje pro čtení, transformaci, manipulaci a zobrazení XML dokumentů. Tyto nástroje jsou dosažitelné pro programovací jazyky Java, C, C++. Díky těmto nástrojům lze efektivním způsobem vytvářet aplikace, které pro svůj běh využívají Oracle databázi.

4.2 Oracle XML DB – Architektura

Architektura Oracle XML DB má dva hlavní rysy. Jedním z nich je skladovaní v XMLType tabulkách či pohledech. Druhým je Oracle XML DB Repository. Pro větší názornost je tato architektura znázorněna na následujícím obrázku z dokumentace [9].



Obrázek 1: XMLType skladování a Oracle XML DB úložiště

4.3 Ukládání XML jako XMLType

Existuje řada způsobů jakými lze uložit XML obsah do Oracle Databáze. Pokud bychom neuvažovali Oracle XML DB, tak uložení XML dokumentu bychom mohli realizovat použitím:

- Oracle XML vývojářské výbavy (XDK), abychom rozebrali XML dokument mimo Oracle databázi a poté uložili extrahovaná data po řadě do databázových tabulek.
- CLOB (Charakter Large Object), BLOB (Binary Large Object), BFILE (Binary File) nebo VARCHAR sloupce.

Oba tyto případy ale neposkytují dostatečné techniky na zacházení s XML obsahem.
Zavedením XMLType byly vytvořeny nové techniky zacházení s XML obsahem. Usnadňují zachování stálosti XML obsahu v databázi. Lze ukládat XML dokumenty jako XMLType sloupce tabulky nebo vytvářet tabulky jejichž obsahem je XML.
XMLType může reprezentovat XML dokument, který je přístupný pomocí jazyka SQL nebo můžeme použít metody na vytváření, získávání či indexaci XML dat uložených v databázi.
SQL poskytuje funkce pracující nad celým XML dokumentem, respektive XMLType typem. Funkce `existsNode` zjišťuje přítomnost uzlu v XML dokumentu a funkce `extract` extrahuje XML dokument. Existuje mnoho dalších případů užití XMLType typu. Není však naším cílem ukázat všechny, ale pro podrobné prozkoumání lze navštívit dokumentaci k Oracle [9].

4.4 Ukládání XML obsahu do Oracle databáze

Chceme-li uložit XML obsah do databáze, tak se nám nabízí řada možností, jak tuto operaci provést. V Oracle XML DB existují dva základní přístupy. Tabulkově založený přístup a přístup, který je založen na cestě k prostředku uloženého v Oracle XML DB úložišti.

Tabulkové uložení XML obsahu do databáze můžeme realizovat za pomocí různých jazyků. Patří mezi ně SQL, PL/SQL, Java, C nebo nástroj SQL*Loader. Při výběru druhého přístupu, máme k dispozici balík DBMS_XDB z jazyka PL/SQL nebo protokoly jako je WebDAV, Windows Explorer a další.

Pro ukázku uvedu ukládání XML obsahu v jazyce Java :

```
public void Insert(Connection con, Document doc)
throws Exception
{
   String SQLTEXT = "INSERT INTO tabulka1 VALUES (?)";
   XMLType xml = null;
   xml = XMLType.createXML(con,doc);
   OraclePreparedStatement sqlStatement = null;
   sqlStatement = (OraclePreparedStatement) con.prepareStatement(SQLTEXT);
   sqlStatement.setObject(1,xml);
   sqlStatement.execute();
}
```

Funkci Insert se předá jako první parametr odkaz na spojení s databází a druhý parametr, který je získán pomocí dokumentově objektového modelu DOM. Vytvoří se XMLType, který se vloží do tabulky. Příklad předpokládá vytvoření tabulky `tabulka1`, která má první sloupec číselného typu a druhý sloupec typu XMLType.

4.5 XML Schema

XML Schema (XML Schéma) bylo vytvořeno konsorciem W3C k popisování strukturovaných XML dokumentů. XML Schéma zahrnuje úplnou definice typu dokumentů takovou, že jsme schopni převést definici na XML Schema. XML Schema je schéma definované jazykem napsaným v XML. Takové schéma může být použito k popsání XML dokumentů.

Chceme-li používat XML Schema v Oracle XML DB, měly bychom znát některé základní operace. Těmito operacemi rozumíme registrace, aktualizace a mazání schématu.

XML Schema do velké míry souvisí s používáním abstraktního datového typu XMLType. o XMLType typu víme, že usnadňuje ukládání XML dat do databáze. XML Schema nabízí dodatečné volby uložení a přístupu k XML datům. Umožňuje definovat XML elementy a atributy, jejich druhy a také datové typy, které můžeme použít. XML Schema tak může ověřit jestli XML data, respektive XMLType uchovávající XML dokument odpovídá plánované definici.

Uveďme jednoduchý příklad, na kterém ukážeme registraci a smazání zaregistrovaného schématu. Uvažme obrázek, o kterém bychom chtěli získávat různé informace. V našem případě budeme chtít uchovávat informaci o šířce, výšce a barevné hloubce obrázku. Vytvoříme si schéma `imagetechnique.xsd`:

```
<xsd:schema targetNamespace="inamespace"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xdb="http://xmlns.oracle.com/xdb" xmlns="inamespace">
    <xsd:element name="ImgTechMetadata" xdb:defaultTable="IMGTABLE">
      <xsd:complexType>
            <xsd:sequence>
                  <xsd:element name="Height"      type="xsd:float"/>
                   <xsd:element name="Width"       type="xsd:float"/>
                   <xsd:element name="ColorDepth"  type="xsd:integer"/>
            </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Takovému schématu potom odpovídá například následující XML dokument:

```
<i:ImgTechMetadata xmlns:i="inamespace"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="inamespace imagetechnique.xsd">
      <Height>1024</Height>
      <Width>640</Width>
      <ColorDepth>24</ColorDepth>
</i:ImgTechMetadata>
```

Před tím, než budeme používat XML schéma, je nutná jeho registrace v Oracle databázi. Registraci lze provést použitím balíku DBMS_XMLSCHEMA z jazyka PL/SQL. Formát SQL dotazu je následující:

```
BEGIN
      DBMS_SCHEMA.registerSchema(
            SCHEMAURL,
            SCHEMADOC,
            CSID;
      )
END;
/
```

- `SCHEMAURL` URL - schématu XML. Jednoznačný identifikátor XML schématu v rámci Oracle XML DB. Oracle XML DB nikdy neumožní přístup na tuto adresu z internetu.
- `SCHEMADOC` - zdroj schématu XML. Hodnota může být v datových typech VARCHAR, BLOB, CLOB, BFILE, XMLType nebo URIType. Nesmíme také zapomenout, aby uživateli byly přidělena práva, která ho opravňují k vytváření takovýchto datových typů v Oracle databázi.
- `CSID` - znaková sada zdroje. Nutné použít, když je zdroj schématu uložen v datovém typu BLOB nebo BFILE [9].

Naše konkrétní registrace schématu XML by poté mohla být provedena pomocí konzole SQL Plus a zadáním následujícího dotazu:

```
BEGIN
DBMS_SCHEMA.registerSchema(
 'imagetechnique.xsd',
 '<xsd:schema targetNamespace="inamespace"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xdb="http://xmlns.oracle.com/xdb" xmlns="inamespace">
     <xsd:element name="ImgTechMetadata" xdb:defaultTable="IMGTABLE">
       <xsd:complexType>
             <xsd:sequence>
              <xsd:element name="Height"      type="xsd:float"/>
              <xsd:element name="Width"       type="xsd:float"/>
              <xsd:element name="ColorDepth"  type="xsd:integer"/>
             </xsd:sequence>
       </xsd:complexType>
     </xsd:element>
 </xsd:schema>';
)
END;
/
```

Smazání XML schématu, pomocí balíku DBMS_SCHEMA by bylo provedeno zadáním SQL dotazu ve tvaru:

```
BEGIN
      DBMS_SCHEMA.deleteSchema(
            SCHEMAURL,
            DELETE_OPTION);
END;
/
```

- `SCHEMAURL` - URL schématu XML. Jednoznačný identifikátor XML schématu v rámci Oracle XML DB. Oracle XML DB nikdy neumožní přístup na tuto adresu z internetu.
- `DELETE_OPTION` - volby smazání schématu XML.

V Oracle XML DB lze najít spoustu dalších využití, která se týkají uložení a dotazování nad XML Schema. Lze vytvářet tabulky XMLType typu a sloupce založené na schématu XML. Dále je umožněno překládaní XML do mnoho různých jazyků. Není našim cílem předvést všechna tato použití, protože by byl rámec práce překročen.

4.6 Programové spojení s Oracle XML DB

Pokud chceme vyvíjet aplikace, které poskytují kvalitní uživatelské grafické rozhraní, pak si nevystačíme pouze s nástrojem SQL Plus. Všechna funkcionalita Oracle XML DB je přístupná použitím jazyků C, PL/SQL a Java. Pokud se rozhodneme vyvíjet aplikace v Java, pak se nám nabízí použití JDBC (Java Database Connectivity). JDBC je standard poskytující rozhraní pro vytvoření spojení mezi jazykem Java a relačními databázemi [13]. JDBC je definován firmou Sun Microsystems a implementován v rozhraní java.sql [14]. Oracle nabízí řadu JDBC ovladačů. Mezi ně patří Thin driver, OCI (Oracle Call Interface) a ovladače existující na straně serveru.

5 Oracle XML DB Repository

V předchozí kapitole jsme si řekli o existenci tzv. obsahově a datově orientovaný přístupu. Ve stručnosti jsme rozebrali XMLType typ a nastínili jsme možné použití v tabulkách databáze. Po dohodě s vedoucím práce jsem se zaměřil na Oracle XML DB Repository, tedy na obsahově orientovaný přístup k databázi. Tím ovšem nevylučuji použití XMLType typu v ukázkové aplikaci, kterou jsem vytvořil. Abych mohl přistoupit k prezentaci ukázkové aplikace, pak je potřeba se dovědět základy použití Oracle XML DB úložiště.

5.1 Přehled Oracle XML DB Repository

Oracle XML DB Repository (Oracle XML DB úložiště) je částí Oracle databáze. Tato část se snaží o maximální podporu obsluhy XML dat. Oracle XML DB Repository obsahuje prostředky. Pod těmito prostředky si lze představit složky nebo soubory, jež mají své vlastnosti. Každý takový prostředek je identifikován pomocí cesty a jména prostředku. Dále můžou mít vlastnosti jako je obsah, systémově definovaná popisná data tzv. systémově definovaná metadata (vlastník prostředku, datum vytvoření atd.) a uživatelsky definovaná metadata. V neposlední řadě jsou prostředky spojovány s acces control list (přístupový kontrolni seznam), který rozhoduje, kdo může přistoupit k prostředku a jaké operace nad tímto prostředkem může vykonávat.

Ačkoliv Oracle XM DB Repository je orientováno a vzniklo pro zacházení s obsahem XML, tak můžeme použít Oracle XML DB repository k ukládání jiných druhů dat, než jsou jenom XML data. V ukázkové aplikaci jsem tohoto využil. Spojuji zde binární data (obrázky a ruzné jiné soubory než XML) s XML daty.

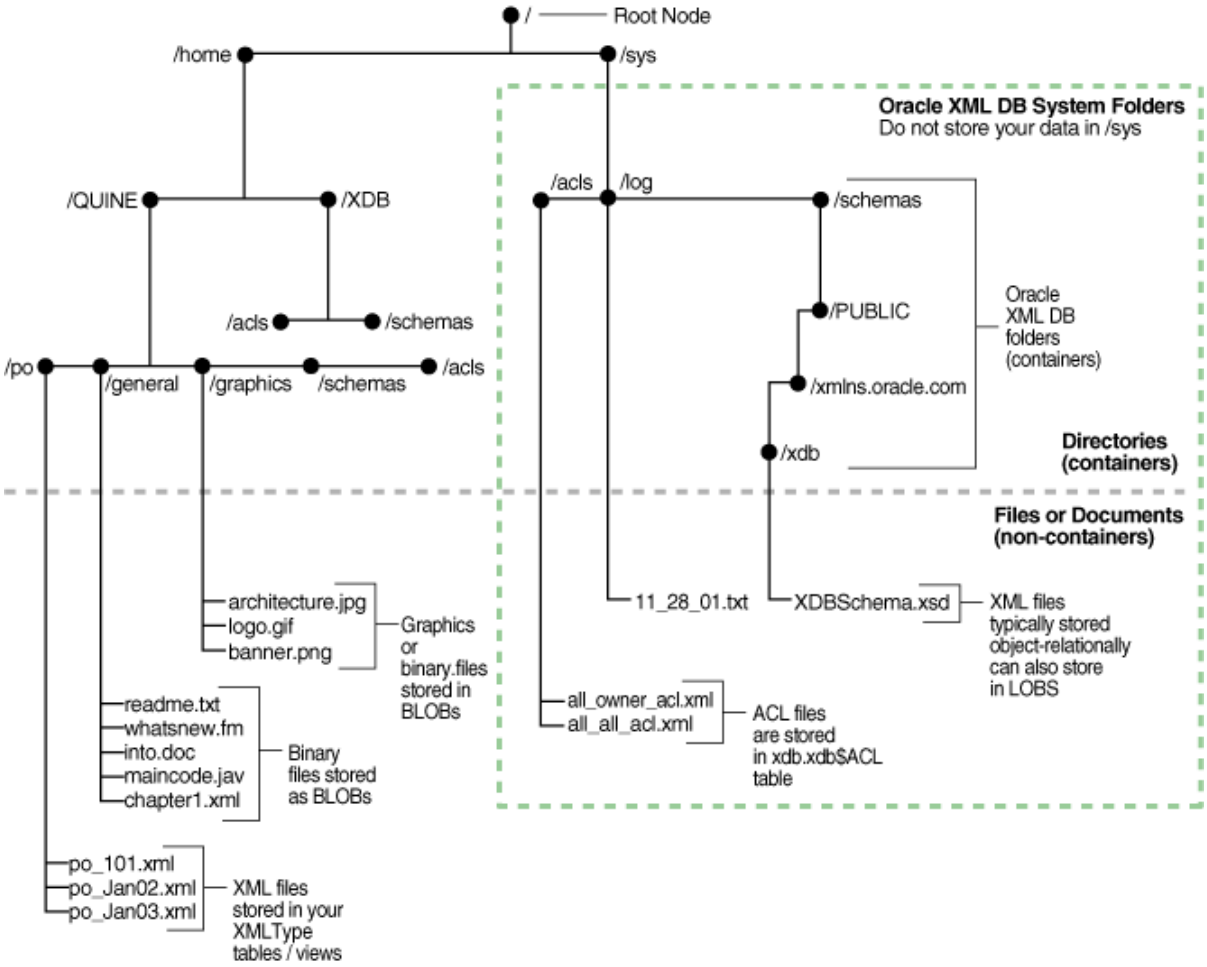
Přistup a manipulaci s prostředky Oracle XML DB repository můžeme zajistit různými cestami. Jako první se nabízí jazyk SQL se svými pohledy RESOURCE_VIEW a PATH_VIEW. Dále jazyk PL/SQL prostřednistvím DBML_XDB API nebo můžeme přístup a manipulaci s prostředky zajistit pomocí jazyka Java.

Java, jakožto významný programovací jazyk Oracle databáze, poskytuje v tomto směru bohatou podporu [15].

5.2 Oracle XML DB Foldering

Oracle XML DB Foldering by se do češtiny dalo přeložit jako využití vlastností složek a jejich hierarchie pro ukládání obsahu do databáze. V Oracle XML DB Repository prostředky ukládáme v hierarchické struktuře, na rozdíl od klasických databázových struktur.

Na následujícím obrázku můžeme vidět typický strom prostředků (složek a souborů) tvořící hierarchickou strukturu v Oracle XML DB Repository [9]. Na vrcholu tohoto stromu se nachází kořenová složka „/“ neboli „root folder“.



Obrázek 2: Strom prostředků v Oracle XML DB Repository

5.3 Uložení dat v Repository

Oracle XML DB Repository představuje množinu databázových objektů napříč všemi schématy XML a schématy databáze. Objektům můžeme dát cestu a jméno, to znamená, že je připojujeme do acyklického grafu prostředků . Tento acyklický graf má jediný kořenový uzel „/“.

Oracle XML DB ukládá data v úložišti jako množinu tabulek. Když zaregistrujeme schéma XML a vytvoříme tabulku pomocí Oracle XML DB, pak je tabulka vytvořena v našem databázovém schématu. Poté jsme schopni se na tuto tabulku dívat a modifikovat ji. Ostatním uživatelům zůstává tabulka skryta.

5.4 Link Types

Link types (typy odkazů) mohou v Oracle XML DB být buď repository links (odkazy úložiště) nebo document links (dokumentové odkazy). Odkazy úložiště jsou navigační odkazy mezi prostředky úložiště, tzn. „složky mají své děti a děti své rodiče“. Odkazy dokumentové jsou libovolné odkazy mezi dokumenty a nemusejí to být nutně odkazy mezi prostředky úložiště.

Odkazy, které se zaměřují na repository (odkazy mezi prostředky), mohou být buď Hard (tvrdé), nebo Weak (slabé). Rozdíly mezi těmito typy odkazu na první pohled nejsou zřejmé, protože mají spoustu shodných rysů. Oba typy odkazují buď na místo paměti kde jsou uložena data nebo na cestu, kde je další odkaz. Tvrdé i slabé odkazy zůstávají validní i potom, kdy dojde ke změně jejich zdrojů. To znamená, že můžeme odkazované prostředky přejmenovat nebo přemístit bez jakékoli změny odkazu.

V ukázkové aplikaci používám tvrdé odkazy, a proto se podívejme, v čem se liší od slabých odkazů. Rozdíl spočívá ve vztahu ke smazanému cílovému prostředku odkazu. Cílový prostředek je závislý na tvrdém odkazu v tom smyslu, že nemůže být smazán tak dlouho, jak dlouho existuje tvrdý odkaz. Smazáním tvrdého odkazu můžeme provést poté, kdy je tvrdý odkaz posledním tvrdým odkazem prostředku.

Slabý odkaz není tak pevný. Můžeme smazat prostředek i když je cílem slabého odkazu. Slabé odkazy se používají jako zástupci prostředků, ke kterým se často přistupuje. Negativní dopady, zapříčiněné smazáním prostředků u slabých odkazů, nejsou.

Praktické využití slabých odkazů v Oracle XML DB Repository můžeme prezentovat na krátkém příkladě. Existují dva uživatelé A a B. Uživatel A potřebuje přistoupit k prostředku X, který vlastní uživatel B. Uživatel B zjistí OID prostředku X (jednoznačný identifikátor prostředku) a pošle toto OID uživateli A. Uživatel A vytvoří slabý odkaz a připojí prostředek X do svého stromu prostředků. Výhoda takové přístupu je v tom, že uživatel A nemusí znát strukturu úložiště uživatele B.

5.5 Přistupování k datům uloženým v Repository

Existuje mnoho cest, jak lze přistoupit k datům prostředků uložených v Oracle XML DB Repository. Můžeme použít:

- internetové protokoly (HTTP(S), WebDAV nebo FTP) a protokolový server Oracle XML DB
- Oracle XML DB Content Connector a standart Content Repository API pro Javu (JCR)
- Oracle XML DB Repository API pro Javu
- Oracle XML DB Repository API pro PL/SQL

V ukázkové aplikaci přistupuji a manipuluji data v úložišti pomocí JCR (Java Content Connector).
Mezi základní operace v Oracle XML DB Repository patří vytváření, mazání a aktualizace prostředků. Dále pak vytváření Link Types, získání seznamu prostředků ze složky atd.
V ukázkové aplikaci nezůstávám jenom u použití JCR. Používám Oracle XML DB Repository API pro PL/SQL, konkrétně vytvoření tvrdého odkazu prostředku. SQL dotaz, který zajistí vytvoření odkazu, má následující tvar:

```
CALLDBMS_XDB.Link(PATH,MOUNT_PATH, NAME_RESOURCE);
```

- `PATH` cesta v úložiště ke zdrojovému prostředku
- `MOUNT_PATH` cesta v úložišti, kde bude připojen prostředek
- `NAME_RESOURCE` jméno zdrojového prostředku

SQL dotaz, který by připojil prostředek daný cestou „/public/BP1/my.txt“ na místo dané cestou „/public“ by vypadal takto:

```
CALLDBMS_XDB.Link(„/public/BP1/my.txt“, „/public“, „my.txt“);
```

5.6 RESOURCE_VIEW a PATH_VIEW

Veřejné pohledy RESOURCE_VIEW a PATH_VIEW umožňují přístup k datům v Oracle XML DB Repository. K datům uloženým v repository můžeme přistupovat pomocí jazyka SQL použitím hodnot z pohledů RESOURCE_VIEW a PATH_VIEW.

RESOURCE_VIEW se skládá z prostředků, které obsahují jméno prostředku, na které se můžeme dotazovat. Pokud je obsahem prostředek XML, pak se tento prostředek ukládá do XMLType tabulek nebo pohledů. V takovém případě pohled RESOURCE_VIEW ukazuje na řádek XMLType s uloženým obsahem. Může se stát, že obsahem prostředku není XML, pak RESOURCE_VIEW uloží prostředek jako LOB.

Hierarchický vztah mezi složkou rodiče a dítěte je nezbytný. Použitím hierarchických indexů úložiště docílíme hierarchie složek.

Podíváme-li se na definici pohledu RESOURCE_VIEW, tak zjistíme, že obsahuje jeden řádek pro každý prostředek v Oracle XML DB Repository . RESOURC_VIEW vytváří strukturu tří sloupců. Prvním je RES, nabývá datového typu XMLType a jedná se o prostředek v úložišti. Druhým sloupcem ANY_PATH se myslí cesta k prostředku. Nabývá datového typu VARCHAR2. Poslední sloupec s názvem RESID je datového typu RAW a jedná se o OID prostředku, které je jedinečné pro konkrétní prostředek.

Pohled PATH_VIEW obsahuje jeden řádek pro každou přístupovou cestu k prostředku v Oracle XML DB Repository. PATH_VIEW vytváří strukturu čtyř sloupců (PATH, RES, LINK, RESID). První je datového typu VARCHAR2 a označuje cestu k prostředku úložiště RES. RES je datového typu XMLType a odkazuje na sloupec PATH. LINK je typu XMLType a označuje, jaký charakter má odkaz (Hard Links, Weak Links).

Rozdíl spočívá v tom, že PATH_VIEW zobrazuje všechny cesty ke konkrétnímu prostředku a RESOURCE_VIEW zobrazuje jednu možnou cestu k prostředku.

RESOURCE_VIEW a PATH_VIEW využívají funkce jazyka SQL. Mezi ně patří například funkce `under_path` (vrácí jména cest, které se vyskytují v úložišti pod jménem cesty, která byla zadána jako parametr funkce). Tato funkce může vracet jména cest absolutně nebo relativně.

Příklad ukazující funkcionalitu funkce `under_path` může vypadat následovně. Představme si, že v Oracle XML DB Repository existuje tato hierarchická struktura prostředků:

```
/a/b/c/d
/a/b/d
/b/e/a
/c/d/e
```

Použitím funkce `under_path,` na získání relativní cesty pod cestou „/a/b“ v SQL dotazu,

```
SELECT path(1) FROM RESOURCE_VIEW WHERE under_path(RES, '/a/b', 1) = 1;
```

získáme následující výsledek:

```
/c/d
/d
```

Kromě dalších variant použití této funkce, existují také funkce jako `equals_path`, `path` a `depth.` Funkcionalitu lze nalézt v dokumentaci k Oraclu [9]. V ukázkové aplikaci používám aktualizaci (UPDATE) prostředků úložiště ve spojení s RESOURCE_VIEW. Toto použití bude demonstrována v kapitole o ukázkové aplikaci.

5.7 Uživatelsky definovaná metadata úložiště

Ne všechna data, která máme uložena v úložišti, nám poskytují informace, které o nich potřebujeme vědět. Vznikla potřeba spojovat přídavné informace s daty v Oracle XML DB Repository. Takové informace nejsou součástí obsahu prostředku. Potřebujeme-li se dovědět o prostředcích v databázi více informací, pak můžeme prostředky roztřídit do různých složek podle jejich typů. Více používaným přístupem je však definování uživatelských metadat k prostředkům. Pro příklad mějme sbírku digitálních fotografií. O každé fotografii potřebujeme znát její technické údaje (rozměry, barevná hloubka,…). Na druhou stranu, můžeme fotografie klasifikovat do různých kategorií (příroda, lidé, rodina, dovolená). Pokud bychom chtěli, aby obrázky spadaly do více kategorií, pak méně efektivním řešením, je vytvoření kopií obrázků v každé složce pro konkrétní kategorii. Lepší řešení nám poskytují uživatelsky definovaná metadata. Obrázek, ať již spadá do více kategorií, je za použití definovaných metadat uložen v úložišti pouze jednou. Definovaná metadata k obrázku obsahují komplexní informace včetně příslušnosti v různých kategoriích. Takovýmto způsobem lze docílit efektivní práce při správě obrázků.

Prostředek uložený v Oracle XML DB Repository můžeme chápat jako XML dokument, který obsahuje metadata a data. Prostředková data jsou obsahem prvku *Contents*. Všechny ostatní prvky v prostředku obsahují popisná data (metadata). Prostředková data mohou být buď XML data, ale není to podmínkou. Znamená to tedy, že do dat prostředku, respektive do prvku *Contents*, lze uložit například data binární.

Hlavní výhodou v tomto směru je *schopnost asociovat námi definovaná metadata s Oracle XML DB úložištěm*. Kromě uživatelsky definovaných metadat (lze je asociovat s každým prostředkem), Oracle XML DB vytváří systémově definovaná popisná data (system-defined metadata). Mezi systémově definovaná popisná data k prostředku patří: *owner* (vlastník), *displayName* (zobrazené jméno), *author* (autor), atd. Také je možné (mimo systémová metadata) určit, zda prostředková informace má být považována za data nebo metadata.

5.7.1 Druhy metadat

Mimo prostředkové informace (systémové a uživateslé metadata) je termín „metadata“ považován za schémata XML, která popisují třídu dokumentů XML nebo za značky XML (elementy a atributy), které jsou použity k popisu elementu obsahu prostředku nebo k popisu hodnot atributů.

Asociaci metadat s XML dokumentem, který je obsahem prostředku, můžeme provádět těmito způsoby:

- Přidávat přídavné elementy XML obsahující metadata do obsahu prostředku. V tomto případě bychom neměli zapomenout na charakter dat. Pokud nejsme schopni s určitostí dát odpověď, kde končí data a kde začínají metadata, pak tento způsob nepoužijeme.
- Přidávat metadata k jednotlivým prostředkům úložiště tak, že metadata jsou obsahem samostatného prostředku. V tomto případě aplikace zachází s tímto samostatným prostředkem jako s metadaty a asociuje ho s daty.

5.7.2 Uživatelsky definovaná prostředková metadata

Uživatelsky definovaná metadata k prostředku mohou být založena na schématu XML a nebo nemusí. Metadata prostředku, která jsou založena na XML schématu, jsou uložena v oddělených tabulkách. Tyto jsou spojeny s tabulkou prostředku pomocí OID prostředku.

Metadata prostředku, která nejsou založena na schématu XML, jsou uložena v CLOB sloupci v tabulce prostředku.

Použití metadat prostředku, která jsou založena na schématu XML, lze s výhodou použít. Můžeme vytvářet a registrovat schémata XML, která definují metadata ke konkrétnímu druhu prostředku. Přidávat metadata k prostředkům úložiště a aktualizovat tyto metadata. Dále je možné se dotazovat na konkrétní prostředek a najít požadovaný obsah některé vlastnosti.

Definování metadat, které chceme asociovat s prostředky, můžeme provést pomocí schématu XML. Schéma XML se dá použít jako nástroj pro definování XML dat.

Vraťme se k našemu jednoduchému příkladu s fotografiemi. Pokusíme se na něm demonstrovat použití schématu XML ve spojení s uživatelsky definovanými metadaty. O fotografie můžeme uchovávat informace o výšce, šířce, barevné hloubce, titulku a popisu. Vytvoříme si proto schéma XML, podle něhož budeme ověřovat správnost zadaných informací, a které nám popisuje technické informace o fotografii. SQL dotaz na vytvoření schématu:

```
BEGIN
  DBMS_XMLSCHEMA.registerSchema(
    'image.xsd',
    '<xsd:schema targetNamespace="inamespace"
                 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                 xmlns:xdb="http://xmlns.oracle.com/xdb"
                 xmlns="inamespace">
       <xsd:element name="ImgTechMetadata"
                    xdb:defaultTable="IMGTECHMETADATATABLE">
         <xsd:complexType>
           <xsd:sequence>
             <xsd:element name="Height"      type="xsd:float"/>
             <xsd:element name="Width"       type="xsd:float"/>
             <xsd:element name="ColorDepth"  type="xsd:integer"/>
```

```
            <xsd:element name="Title"       type="xsd:string"/>
            <xsd:element name="Description" type="xsd:string"/>
          </xsd:sequence>
         </xsd:complexType>
        </xsd:element>
      </xsd:schema>',
    enableHierarchy=>DBMS_XMLSCHEMA.ENABLE_HIERARCHY_RESMETADATA);
END;
/
```

Tímto dotazem se vytvoří a zaregistruje schéma XML image.xsd do Oracle XML DB. Použili jsme proceduru `DBMS_XMLSCHEMA.registerSchema` na zaregistrování schématu XML.

5.7.3 Přidávání, aktualizace a mazání metadat prostředků

Mazání, přidávání a aktualizace prostředkových metadat lze zařídit například použitím PL/SQL procedur z balíku DBMS_XDB (`AppendResourceMetadata`, `updateResourceMetadata`, `deleteResourceMetadata`, `purgeResourceMetadata`). Dalším způsobem je použití SQL DML příkazů (UPDATE, INSERT, DELETE) k přímé aktualizaci prostředku.

Na následujícím příkladu ukážeme použití druhého způsobu. Vložíme metadata přímo do RESOURCE_VIEW použitím SQL příkazu UPDATE. Budeme předpokládat, že náš prostředek je uložen v Oracle XML DB Repository na cestě „/public/obrazky/obr.jpg“:

```
UPDATE RESOURCE_VIEW
  SET RES =
        insertChildXML(
          RES,
          '/r:Resource',
          'i:ImgTechMetadata',
          XMLType('<i:ImgTechMetadata
                 xmlns:i="inamespace"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="inamespace image.xsd">
               <Height>680</Height>
               <Width>240</Width>
               <ColorDepth>32</ColorDepth>
               <Title>obrazek slunce</Title>
               <Description>Pořízeno za slunečného počasí</Description>
             </i:ImgTechMetadata>'),
          'xmlns:r="http://xmlns.oracle.com/xdb/XDBResource.xsd"
           xmlns:i="inamespace"
        ')
  WHERE equals_path(RES, '/public/obrazky/obr.jpg') = 1;
/
```

V příkladu se mimo přidání metadat k prostředku vyskytuje také použití schématu XML, které jsme registrovali výše. Znamená to tedy, že jsme schopni zpětně ověřit správnost námi vložených metadat podle schématu *image.xsd*. Takové skutečnosti lze v praxi s výhodou využít. Pokud metadata neodpovídají definovanému schématu, pak neumožníme vložení metadat k prostředku.

Pokud bychom se chtěli podívat na obsah vložených metadat k prostředku, pak se nám nabízí použití funkce `extract` z jazyka SQL. Extrahujeme tak vložené metadata pomocí RESOURCE_VIEW:

```
SELECT extract(RES,
       '/r:Resource/i:ImgTecMetadata',
       'xmlns:r="http://xmlns.oracle.com/xdb/XDBResource.xsd"
       xmlns:i="inamespace"
       ') AS data
FROM RESOURCE_VIEW
 WHERE equals_path(RES, '/public/obrazky/obr.jpg') = 1;
```

Tento příkaz nám extrahuje vložené metadata prostředku. V našem případě by výstup vypadal takto:

```
<i:ImgTechMetadata
          xmlns:i="inamespace"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="inamespace image.xsd">
        <Height>680</Height>
        <Width>240</Width>
        <ColorDepth>32</ColorDepth>
        <Title>obrazek slunce</Title>
        <Description>Pořízeno za slunečného počasí</Description>
</i:ImgTechMetadata>
```

Na posledním příkladu této podkapitoly ukážeme smazání metadat prostředku pomocí balíku DBMS_XDB, respektive použitím funkce z tohoto balíku. Jedná se o funkci `deleteResourceMetadata`.

```
BEGIN
       DBMS_XDB.deleteResourceMetadata('/public/obrazky/obr.jpg',
              'inamespace',
              'ImgTechMetadata');
END;
/
```

Implicitně se provedlo smazání prostředkového odkazu (REF) a tabulky metadat identifikované prostředkovým odkazem. Pomocí voleb funkce `deleteResourceMetadata` lze specifikovat volby smazání.

5.8 Oracle XML DB Content Connector

Oracle XML DB Content Connector implementuje Content Repository API pro Javu (JCR – Java Content Repository). *Oracle XML DB Content Connector umožňuje přístup do Oracle XML DB Repository použitím JCR.*

Soubory a složky v Oracle XML DB Repository jsou reprezentovány jako JCR uzly. Nad takovými uzly můžeme provádět standardní operace (jako je vytváření, získávání a aktualizace uzlů) pomocí JCR APIs.

5.8.1 JCR

JCR je standard Javy pro aplikace, které pracující s obsahem Oracle XML DB Repository. JCR vytváří data v prostoru úložiště jako strom uzlů. Každý uzel může mít jeden nebo více synovských uzlů. Každý uzel má jediný otcovský uzel, kromě kořenového uzlu. Kromě synovských uzlů, může mít uzel ještě jednu nebo více vlastností (properties). Vlastnost (property) představuje dvojice hodnota/jméno. Například uzel, který reprezentuje konkrétní složku má v obsahu úložiště vlastnost, která se jmenuje *ojcr:owner* a nabývá hodnoty jména vlastníka prostředku. Obdobně tomu je u souborů. Každá vlastnost má také svůj typ.

Dalo by se říci, že každý uzel má svůj uzlový typ. Uzel reprezentující složku má typ *nt:file*. Uzlový typ také kontroluje, jaké synovské uzly a vlastnosti musí nebo může mít uzel.

Jelikož uzly a vlastnosti mají svá jména, tak se k nim může přistupovat pomocí cesty. JCR poskytuje podporu pro relativní i absolutní cestu. Například pokud bychom chtěli zjistit čas vytvoření prostředku „obr.jpg“ z předchozí podkapitoly, pak bychom tuto informaci získaly adresováním cesty „public/obrazky/obr.jpg/jcr:created“.

Jména uzlů a vlastností mohou mít odpovídající jmenný prostor. JCR používá dvojtečku k oddělení prefixu jmenného prostoru od jména vlastnosti nebo uzlu.

5.8.2 Pohled na Oracle XML DB Repository jako na JCR

Oracle XML DB Content Connector představuje data úložiště, jako JCR uzly a vlastnosti. Soubory jsou chápány jako uzly typu *nt:file* a složky jako uzly typu *nt:folder*. Obsah a medatada těchto uzlů, můžeme chápat jako na uzly, uzlového typu *nt:resource*.

Na jednoduchém příkladě si ukážeme pohled na složku a soubor (prostředky z úložiště) v kontextu JCR.

Složka *Slozka1* je uložena v kořenové složce Oracle XML DB Repository. Složka *Slozka1* obsahuje soubor *nastaveni.xml*.

Soubor *nastaveni.xml* má následující obsah:

```
<spojeni>
 <port>1521</port>
 <SID>BP</SID>
</spojeni>
```

Oracle XML DB Content Connector ukazuje složku *Slozka1* a soubor *nastaveni.xml* jako JCR uzly a vlastnosti. Struktura zanoření uzlů a vlastností by na našem příkladě byla následující:

```
[root] (nt:folder)
      jcr:created="2009-05-13T21:00:00.000Z"
      jcr:content (nt:resources)
            jcr:data=null
            jcr:lastModified="2009-05-13T21:00:00.000Z"
            ...
            ojcr:displayName=""
      Slozka1 (nt:folder)
            jcr:created="2009-05-13T21:00:00.000Z"
            jcr:content (nt:resources)
                  jcr:data=null
                  jcr:lastModified="2009-05-13T21:00:00.000Z"
                  ...
                  ojcr:displayName="Slozka1"
            nastaveni.xml (nt:file)
                  jcr:created="2009-05-13T21:00:00.000Z"
                  jcr:content (nt:resources)
                        jcr:data=<binární reprezentace obsahu XML>
                        jcr:lastModified="2009-05-13T21:10:00.000Z"
                        ...
                        ojcr:displayName="nastaveni.xml"
                        ojcr:xmlContent (nt:unstructured)
                              spojeni
                                    port
                                          jcr:xmltext
                                                jcr:xmlcharacters="1521"
                                    SID
                                          jcr:xmltext
                                                jcr:xmlcharacters="BP"
                        ojcr:links
                        ...
```

Oracle XML DB Content Connector rozšiřuje definici uzlových typů, a tak roste jejich počet. Například uzlový typ *mix:referenceable* dovoluje, aby všechny soubory a složky byly přístupné pomocí jejich prostředkového identifikačního čísla.

JCR se tedy v podstatě snaží mapovat Oracle XML DB prostředky, vytváří se tak jakési nové obsahové úložiště, které má stromovou strukturu a hierarchii. Zmíníme-li tedy v tomto kontextu systémově či uživatelsky definovaná metadata nebo trvdé a slabé odkazy, pak můžeme říci, že i pro tyto prostředky uložené v Oracle XML DB Repository existuje mapování do JCR.

V ukázkové aplikaci je také mimo jiné předvedena podpora práce s JCR.

6 Ukázková aplikace

Součástí mé bakalářské práce je také ukázková aplikace ilustrující použití prostředků pro práci s XML v Oracle databázi. Ukázková aplikace neměla předem definovanou funkčnost. Aplikace se snaží o předvedení podpory práce s XML tak, aby mohla být použita běžnými uživateli. Při návrhu aplikace jsem se snažil o příjemné uživatelské rozhraní tak, aby uživatel intuitivně používal poskytovanou funkcionalitu. Nebylo jednoduché skloubit ukázku práce s XML s uživatelským rozhraním tak, aby tato podpora práce s XML nezanikla a nestala se pouhou součástí grafického uživatelského prostředí.

Ukázková aplikace zobrazuje použití prostředků práce s XML, které byly popsány v této dokumentaci. Snaží se o maximální možnou demonstraci těchto prostředků i přesto, že se nejedná o ryze výukovou aplikaci.

Funkční obsah realizované aplikace lze rozdělit do několika částí: Ukázka práce s hierarchickou strukturou složek, souborů a přístupovými právy k prostředkům Oracle XML DB repository. Dále práce s XML dokumenty. Poslední částí je přidávání popisných dat k prostředkům Oracle XML DB Repository, jako dat, jež mají strukturu XML dokumentu.

Aplikace je typu klient server. V Oracle XML DB Repository vytvářím uživatele jako jednotlivé složky úložiště. To znamená, že pokud uživatel existuje, tak má ve složce „/public“ v úložišti vytvořen jedinečnou složku, která uživatele identifikuje. Tato složka má shodný název jako přihlašovací jméno uživatele. Aplikace tedy spravuje složky a soubory v úložišti, respektive ho dělí na části, které jsou přístupné (nepřístupné) uživateli.

6.1 Implementační nástroje

K vývoji ukázkové aplikace jsem použil různé nástroje. Ukázková aplikace byla vyvíjena nad operačním systémem Windows XP Servis Pack 3 a nad databázovým systémem Oracle 11g Release 1 (11.1) pro Microsoft Windows (32-Bit). Při studiu funkčnosti různých SQL dotazů jsem použil konzoly SQL Plus.

Vývoj celé aplikace probíhal po většinu času v prostředí NetBeans IDE 6.5.1. Ukázková aplikace je napsána v jazyce Java SE 6, kromě části, která se stará o vytvoření nových uživatelů. Tato část je napsána v jazyce SQL a není nezbytnou součástí běhu programu. Komunikace mezi databází a ukázkovou aplikaci je zabezpečena pomocí komunikačního protokolu JDBC. JDBC s databází komunikuje implicitně na portu 1521.

Pro sestavování programu byl použit program Apache Ant 1.7.1. Ukázková aplikace je umístěna na přiloženém datovém nosiči, včetně všech nástrojů použitých na vývoj aplikace. Ukázkovou aplikaci doporučuji spustit v prostředí NetBeans IDE 6.5.1. Pro toto spuštění jsou

zdrojové soubory umístěny na přiloženém nosiči v archivu *Zdrojove_soubory_pro_NetBeans.zip.* Pro spuštění aplikace můžete ovšem využít program Apache Ant.

6.2 Spuštění aplikace

Pro spuštění aplikace je důležité, aby byl spuštěn databázový server Oracle 11 g Release 1. Každý uživatel má v úložišti složku, která jej identifikuje. Po instalaci databázového serveru Oracle se nepředpokládá existence složek uživatelů ani uživatelů samotných. Přihlásíme se pomocí konzole SQL plus s přihlašovacím jménem SYS AS SYSDBA a spustíme skript, který provede vytvoření nových uživatelů a jejich identifikujících složek. Tento skript se jmenuje *createUser.sql* a lze jej naleznout na přiloženém datovém nosiči. Svoji aplikaci jsem vyvíjel pomocí localhostu. Pokud bych chtěl pracovat se školní instalací databázového serveru, pak musí tento skript spustit správce školního serveru.

Běží-li tedy Oracle server a máme-li vytvořené uživatele, pak spustíme samotnou aplikaci. Máme dvě možnosti:

- použít zdrojových souborů pro NetBeans IDE 6.5.1. Spustíme třídu, ve které se nachází hlavní metoda *main.* Jedná se o třídu *MainH.java*.
- použít program Apache Ant a zdrojové soubory uložené v archivu *ant_zdrojove_soubory.zip*. Rozbalíme jej a pomocí příkazového řádku zadáme v rozbalené složce příkaz pro kompilaci *ant compile* a poté příkaz pro spuštění *ant bp*. Předpokládá se instalace programu Apache Ant 1.7.1.

Důležitým souborem, ve kterém lze nalézt parametry pro připojení k databázi je soubor *spojeni.xml*:

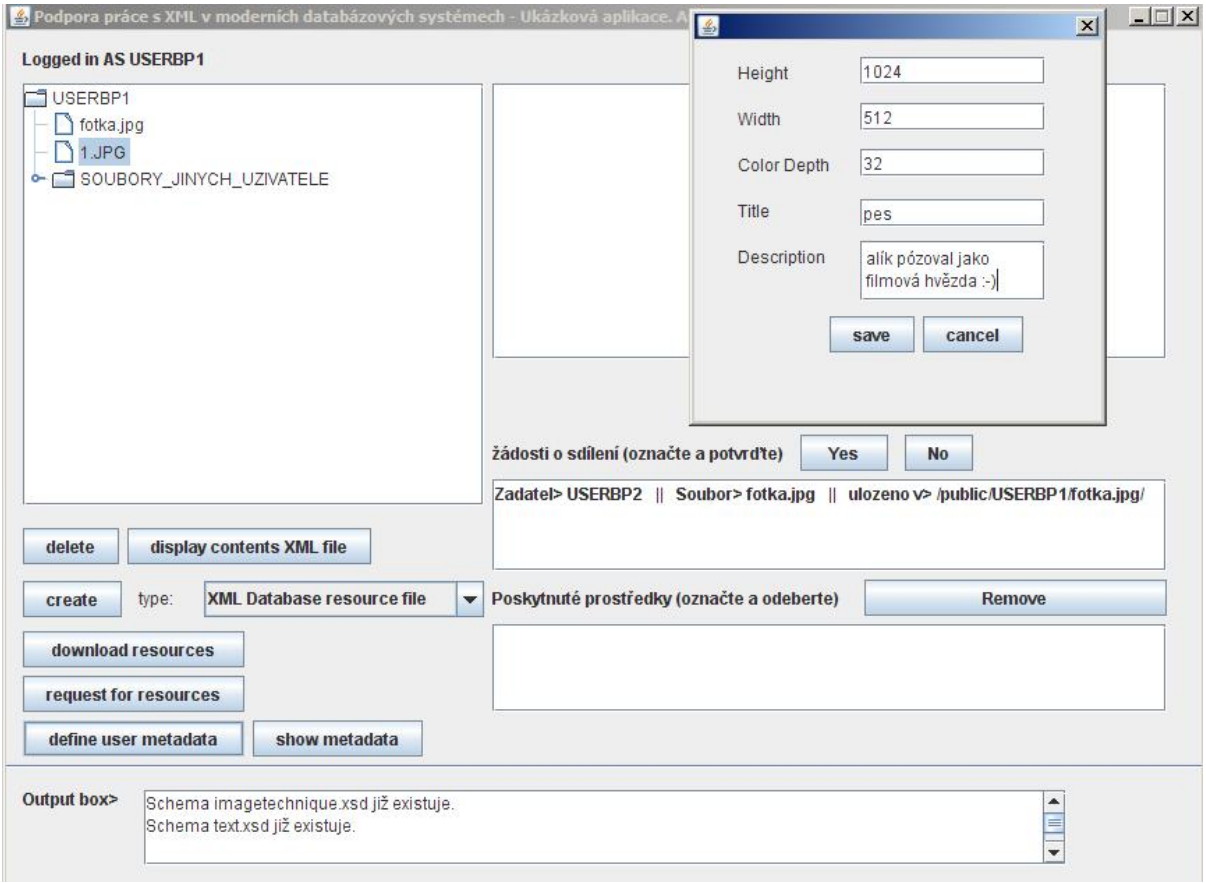
```
<?xml version="1.0"?>
<spojeni>
 <driver>THIN</driver>
 <hostname>localhost</hostname>
  <port>1521</port>
  <SID>BP</SID>
</spojeni>
```

Nalezneme jej na přiloženém datovém nosiči. Tento soubor určuje, jaký driver se má použít pro komunikační protokol JDBC. Dále určuje informaci, kde je databáze spuštěna (hostname), port a SID databáze. Pokud se chceme připojit k nějaké specifické Oracle databázi, pak musíme nastavit tyto parametry.

Nastavíme-li soubor *spojeni.xml* a spustíme aplikaci, pak se zobrazí přihlašovací obrazovka a přihlásíme se. Pokud jsme vytvořili uživatele pomocí skriptu *createUser.sql*, pak máme k dispozici uživatele „USERBP1“ a „USERBP2“. Oba dva mají heslo „12345”.

6.3 Návrh uživatelského rozhraní aplikace

Aplikace byla navrhnuta pomocí grafického uživatelského rozhraní v jazyce Java za použití knihovny *Swing*. *Swing* je knihovna uživatelských prvků, která je založena na platformě Java, a která je určena pro ovládání aplikací pomocí grafického rozhraní. Pomocí knihovny *Swing* můžeme vytvářet okna, dialogy, tlačítka a další grafické komponenty. Hlavní okno aplikace je zobrazeno na následujícím obrázku.



Obrázek 3: Hlavní okno aplikace

Pomocí tlačítek a jiných grafických komponentů je řízen běh programu. Po stisku tlačítka se vyvolá obslužná akce a provede se definovaný kód programu pro konkétní stisknuté tlačítko. V hlavním okně aplikace jsou také umístěny interní okna, které slouží pro vzájemnou interakci uživatele s jiným uživatelem. V levém horním rohu je informace o přihlášeném uživateli.

6.4 Způsob spojení s Oracle XML DB Repository

Spojení s databází je zajištěno pomocí komunikačního protokolu JDBC. Vytvoření takového spojení vypadá následovně:

```
//import potřebných knihoven
import oracle.jdbc.pool.OracleDataSource;
import java.sql.*;
//udržuje objekt databázového spojení
public Connection spojeni;
//inicializace vlastností spojení databáze
OracleDataSource ods = new OracleDataSource();
//nastaveni vlastnosti url
ods.setURL("jdbc:oracle:thin:USERBP1/12345@localhost:1521:BP");
//vytvoření spojení s databází
spojeni = ods.getConnection();
```

Takovéto spojení zajišťuje vykonávání příkazů z jazyka PL/SQL, které aplikace posílá do databáze. V ukázkové aplikaci pracuji s JCR. JCR vyžaduje vytvoření sezení (session) pro spojení s databází Oracle. Pomocí tohoto sezení aplikace vytváří prostředky Oracle XML DB Repository. Sezení se v jazyce Java vytvoří následovně:

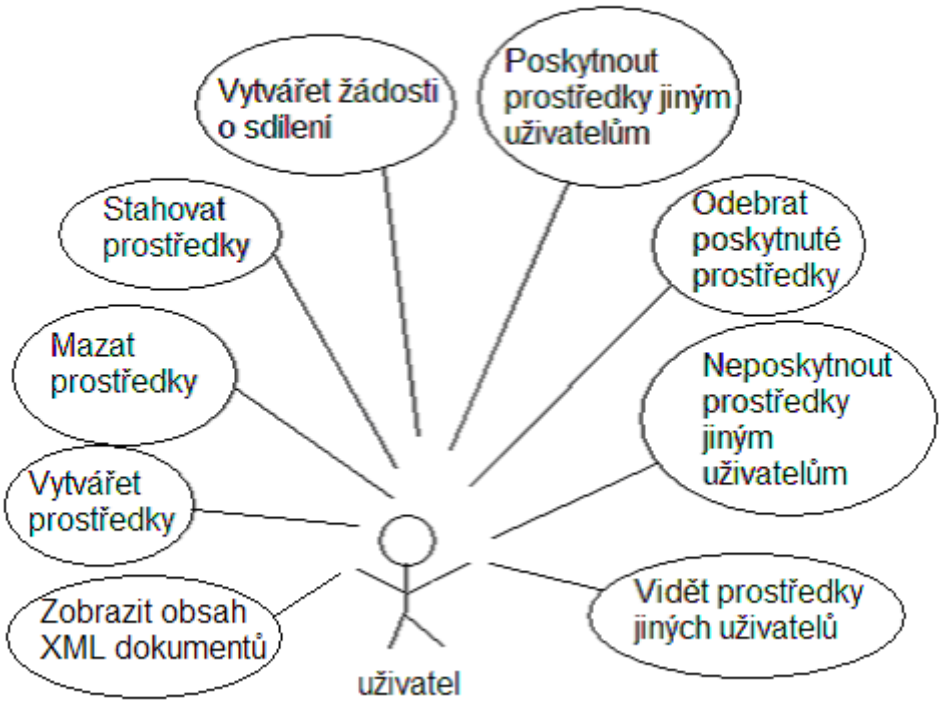
```
//import knihoven
import javax.jcr.Session;
import javax.jcr.SimpleCredentials;
import oracle.jcr.OracleRepository;
import oracle.jcr.OracleRepositoryFactory;
import oracle.jcr.xdb.XDBRepositoryConfiguration;
//udržuje sezení s databází
public Session session;
//vytvoření sezení
XDBRepositoryConfiguration configuration = null;
configuration = new XDBRepositoryConfiguration();
OracleDataSource odss = (OracleDataSource) configuration.getDataSource();
odss.setURL("jdbc:oracle:thin:USERBP1/12345@localhost:1521:BP");
OracleRepository repository =
OracleRepositoryFactory.createOracleRepository (configuration);
SimpleCredentials sc = new SimpleCredentials("USERBP1","12345");
session = repository.login(sc);
```

6.5 Práce s hierarchickou strukturou prostředků Oracle XML DB Repository

Tato část ukázkové aplikace využívá JCR k tomu, aby zjistil hierarchickou strukturu uložení složek a souborů v Oracle XML DB Repository. Aplikace poskytuje uživateli možnost vytváření, mazání a stahování složek a souborů z Oracle XML DB Repository.

Dále je možné sdílení složek a souborů mezi uživateli na základě potvrzených žádostí o sdílení.

Akce, které uživatel může v této časti aplikace provádět, znázorňuje následující diagram případů užití:

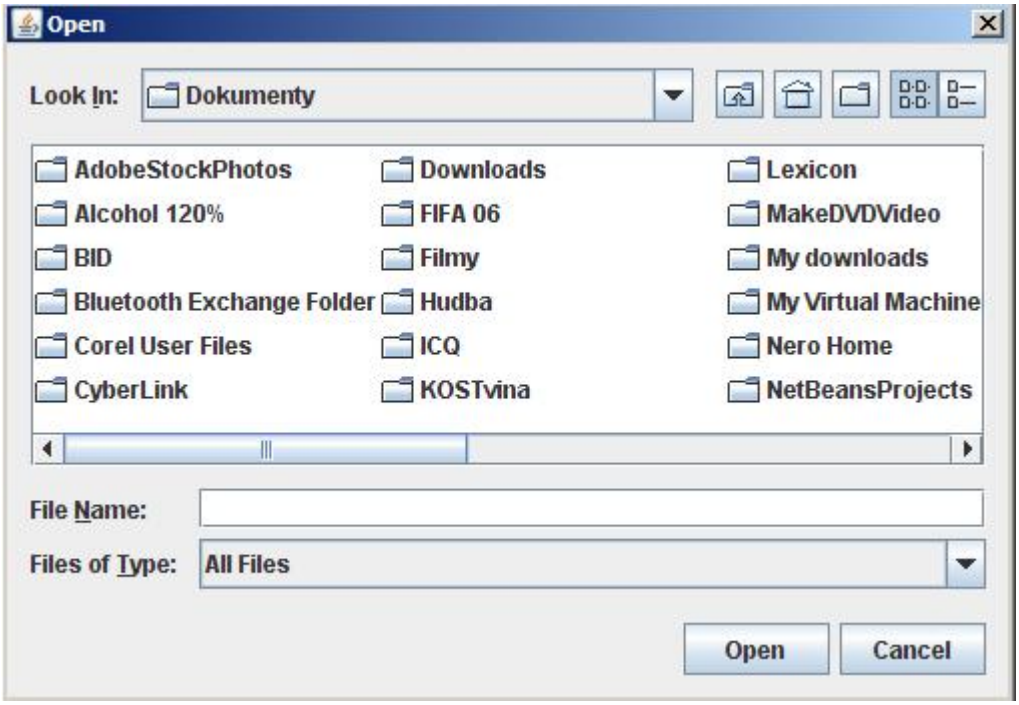


Obrázek 4: Diagram případů užití aplikace, část práce s hierarchickou strukturou

6.5.1 Popis diagramu případů užití

- *Zobrazit obsah XML dokumentu* – aplikace uživateli umožňuje zobrazení XML dokumentů, které jsou uloženy v Oracle XML DB Repository. Po stisku tlačítka *display contents XML file* se obsah označeného XML dokumentu zobrazí do zobrazovacího okna v pravém horním rohu aplikace. Nesmíme zapomenout vybrat XML dokument ve stromě prostředků v levém horním rohu aplikace.

- *Vytvářet prostředek* – umožní uživateli vytvořit databázový prostředek v Oracle XML DB Repository. Vytváří se buď složka nebo soubor. Po stisku tlačítka *create* a výběru typu prostředku *XML Database resource file* se zobrazí následující okno,



Obrázek 5: Okno pro výběr souboru ukládaného do Oracle XML DB Repository

kde vybereme soubor, který chceme uložit do úložiště. Pokud se vybere typ prostředku *XML Database resource folder*, tak se zobrazí okno pro zadání jména složky.

- *Mazat prostředek* – po stisku tlačítka *delete* se provede smazání označeného prostředku ze stromu prostředků v okně aplikace. Současně to znamená smazání prostředku z Oracle XML DB Repository. Nelze smazat prostředky jiných uživatelů.
- *Stahovat prosředek* – po stisku tlačítka *download resources* stáhneme prostředky uložené v Oracle XML DB Repository do vybraného místa na našem lokálním disku.
- *Vytvářet žádost o sdíleni prostředku* – umožňuje zasílat požadavek o prostředek skutečnému majiteli prostředku. Po stisku tlačítka *request for resources* čekáme na připojení prostředku, dokud nám vlastník prostředku požadavek neschválí.
- *Poskytnout prostředek jinému uživateli* – pokud uživatel eviduje žádost o prostředek, pak se tento prostředek zobrazí v okně žádostí o sdílení. Výběrem žádosti a stiskem tlačítka *Yes* poskytneme konkrétní prostředek žadateli o tento prostředek. Žádost o prostředek se poté eviduje jako poskytnutý prostředek v okně poskytnutých prostředků.

- *Neposkytnout prostředek jinému uživateli* - pokud uživatel eviduje žádost o prostředek, pak se tento prostředek zobrazí v okně žádostí o sdílení. Výběrem žádosti a stiskem tlačítka *No* neposkytneme konkrétní prostředek žadateli o tento prostředek.
- *Odebrat poskytnutý prostředek* – pokud se eviduje poskytnutý prostředek nějakému uživateli, pak lze tento prostředek uživateli odebrat tlačítkem *remove.*
- *Vidět prostředky jiných uživatelů* – ve stromě prostředků v levém horním rohu aplikace lze vidět prostředky jiných uživatelů. Prostředky jiných uživatelů jsou umístěny ve složce s názvem *SOUBORY_JINYCH_UZIVATELE*. O tyto prostředky můžeme pouze žádat.

6.5.2 Popis řešení práce s hierarchickou strukturou prostředků úložiště

Po spuštění aplikace dochází k uložení prostředků z Oracle XML DB Repository do stromu souborů a složek, který je umístěn na hlavní obrazovce aplikace. O plnění tohoto stromu prostředků se stará třída *StromSouboru.java*, kde dochází k rekurzivnímu zanořování ve stromě uzlů JCR. Pokud se narazí na nějaký uzel, pak se zjistí jeho uzlový typ. Podle tohoto typu se vytváří uzly ve stromě souborů a složek. Takovýmto způsobem dochází k mapování obsahu Oracle XML DB Repository do stromu souborů a složek.

Operace, které uživatel může vykonávat, jsou závislé na příslušnosti prostředku v jeho složce, která jej identifikuje. Pokud se narazí na prostředek, který není synovským k domovské složce uživatele, pak se mu sice zobrazí do stromu prostředků, ale jakékoliv jiné operace s tímto prostředkem nemůže vykonávat. Může pouze podat žádost o sdílení tohoto prostředku.

Žádosti o sdílení se ukládají do databázové tabulky. Pokud uživatel na základě hodnot z tabulky požadavků zjistí, že mu byla poslána žádost, pak ji buď potvrdí a nebo odmítne. Dále poskytovatel prostředku může odebrat prostředky, které uživatelům poskytl na sdílení.

O kontrolu tabulky žádostí se starají vlákna aplikace. Tyto vlákna zajišťují obsluhu změn v tabulce žádostí.

Na malém příkladě si ukážeme vytvoření databázového prostředku v Oracle XML DB Repository s využitím JCR. Budeme chtít vytvořit soubor *obr.jpg*. Tento soubor máme uložen na svém lokálním disku. Jsme přihlášeni jako *USERBP1* a budeme chtít uložit tento soubor do domovské složky *USERBP1*:

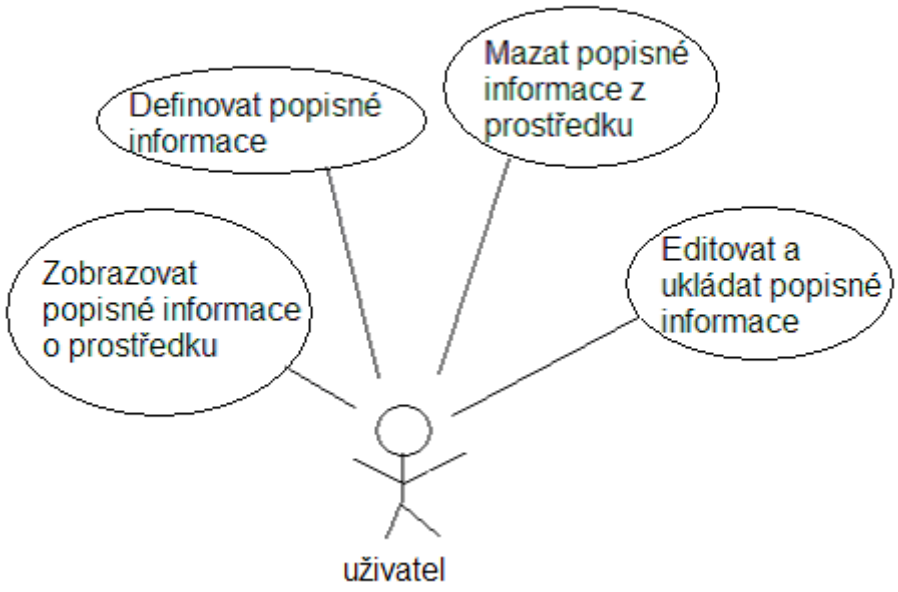
```
//přípojné místo v Oracle XML DB Repository
Node ParentNode = session.getItem("/public/USERBP1/");
//vytvoření uzlu, který reprezentuje soubor
Node node = ParentNode.addNode(file.getName(), "nt:file");
//vytvoření obsahového uzlu
Node contentNode = node.getNode("jcr:content");
//nastavení vlastníka uzlu (souboru)
contentNode.setProperty("ojcr:owner", "USERBP1");
//nastavení zobrazovaného jména souboru
contentNode.setProperty("ojcr:displayName", "obr.jpg");
//nastavení uzlu pro data souboru obr.jpg.
//inputStream je stream datsouboru obr.jpg
contentNode.setProperty("jcr:data", inputStream);
//uložení sezení; vytvoří se soubor obr.jpg v Oracle XML DB Repository
session.save();
```

Vytvoření souboru *obr.jpg* se v aplikaci provede výběrem místa uložení ve stromě souborů, který je zobrazen v hlavním okně aplikace. Dále následuje vybrání typu prostředku *XML Database resource file* a v poslední řadě kliknutí na tlačítko *create*. Poté vyskočí okno, kde se vybere soubor *obr.jpg* z lokálního disku. Posledním krokem je tlačítko *open*. Tímto způsobem lze manipulovat úložiště a vytvářet si hierarchické struktury prostředků v Oracle XML DB repository.

6.6 Přidávání popisných dat k prostředkům Oracle XML DB Repository

Kromě práce s hierarchickou strukturou prostředků z Oracle XML DB repository, nám aplikace umožňuje definování popisných informaci (metadat) ke složkám a souborům úložiště. Je zde využito příkladu o fotografiích, který již byl v této dokumentaci popsán. K obrázkům uloženým v Oracle XML DB Repository je tedy možné definovat přídavné informace, které jsou s tímto obrázkem spjaté. Aplikace se neomezuje v tomto směru pouze na obrázky. Můžeme definovat popisné informace i k jiným prostředkům, například ke složkám.

Na následujícím diagramu případů užití vidíme operace, které uživatel může s konkrétním prostředkem v této části aplikace vykonávat:



Obrázek 6: Diagram případů užití aplikace, část přidávání popisných dat k prostředku

6.6.1 Popis diagramu případů užití

- *Zobrazit popisné informace z prostředku* – po stisku tlačítka *show metadata* se zobrazí popisné informace k prostředku. Informace se zobrazí pouze v případě, kdy jsou u prostředku definovány.
- *Definovat popisné informace* – po stisku tlačítka *define user metadata* se přidávají popisné informace k prostředku. Přidání popisných informací se umožní, pokud prostředek nemá popisné informace definovány.
- *Mazat popisné informace z prostředku* – pokud jsou popisné informace definovány a pokud byly zobrazeny, tak lze popisné informace z prostředku smazat.
- *Editovat a ukládat popisné informace* – editování a uložení lze provést po zobrazení popisných informací.

6.6.2 Popis řešení přidávání popisných dat k prostředkům Oracle XML DB Repository

Prostředky uložené v Oracle XML DB Repository implicitně poskytují systémově definované informace. V ukázkové aplikaci přidávám k prostředkům další popisná data. Těmito popisnými daty jsou XML dokumenty, které jsou vytvořeny vyplněním formulářů v aplikaci. Takový XML dokument pro složky v naší aplikaci vypadá následovně:

```
<t:TextMetadata xmlns:t="tnamespace"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="tnamespace text.xsd">
    <Description>v této složce mám jenom obrázky</Description>
</t:TextMetadata>
```

Z tohoto XML dokumentu lze pak získat další popisné informace ke složce, konkrétně tedy popis složky, který je v hodnotě elementu *Description*. Aplikace by však uložením takového dokumentu, jako samostatného prostředku, nemohla spojit tuto informaci o popisu složky s konkrétní složkou. Proto je nutné takový XML dokument připojit k prostředku, tedy ke konkrétní složce. V našem případě připojíme XML dokument ke složce (prostředku) *slozka1*:

```
UPDATE RESOURCE_VIEW
 SET RES =
 insertChildXML(
       RES,
       '/r:Resource',
       't:TextMetadata',
       XMLType('
             <t:TextMetadata xmlns:t="tnamespace"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="tnamespace text.xsd">
                 <Description>v této složce mám ..</Description>
             </t:TextMetadata>
       '),
       'xmlns:r="http://xmlns.oracle.com/xdb/XDBResource.xsd"
       xmlns:t="tnamespace"
')
 WHERE equals_path(RES, '/public/USERBP1/slozka1') = 1";
```

Takovýmto příkazem jazyka SQL aplikace přidává k prostředkům XML dokumenty. XML dokumenty představují popisná data. V tomto příkazu je však něco nového, XML dokument jsme přidaly jako typ XMLType. Aplikace také podporuje zobrazení vložených XML dokumentů, které jsou datového typu XMLType:

```
SELECT
   extract(
      RES,
      '/r:Resource/t:TextMetadata',
      'xmlns:r="http://xmlns.oracle.com/xdb/XDBResource.xsd"
      xmlns:t="tnamespace"').getCLOBVal() AS metas
FROM RESOURCE_VIEW
WHERE equals_path(RES, '/public/USERBP1/slozka1') = 1;
```

Pokud tedy uživatel popisné informace k prostředku zobrazil, pak je může také editovat. XMLDokument, který představuje popisné informace, je zobrazen do hlavního okna aplikace. Po stisku tlačítka *edit metadata* může změnit obsah XML dokumentu. Po všech potřebných úpravách a po stisku tlačítka *save metadata* se upravený XML dokument převede do XMLType typu a provede se aktualizace popisných informaci prostředku. Aplikace umožňuje také kontrolu ukládaných XML dokumentů (popisné informace k prostředku) podle předem definovaného schématu XML.

6.6.3 Popis řešení ověření správnosti zadávaných popisných informací k prostředku

Vkládané popisné informace k prostředku mají předepsanou strukturu. Tato struktura je předem definována pomocí schématu XML, jež je vytvořeno a zaregistrováno při startu aplikace.

Podle tohoto schématu se ověřuje správnost vkládaného XML dokumentu, který je uložen v datového typu XMLType.

Pokud se uživatel rozhodne vložit popisné informace, pak je vytvořena databázová tabulka. Tato tabulka ve své struktuře definuje jednu položku jako XMLType typ. Aplikace vkládá XML dokument (představující popisné informace) do této položky. Poté zjistí, zda tato položka odpovídá definovanému schématu. Pokud schématu odpovídá, pak je takto ověřený XML dokument přidán k prostředku jako prostředek s popisnými informacemi. Ověření správnosti můžeme otestovat příkazem jazyka SQL,

```
select x.xmlcol.isSchemaValid('image.xsd')AS vysledek from po_tab x
where id=1;
```

kde *xmlcol* představuje položku typu XMLType a je v ní uložen XML dokument, jehož správnost chceme ověřit. Fukce *isSchemaValid* kontroluje správnost XML dokumentu. Název *po_tab* je název tabulky obsahující položku *xmlcol*. Parametr funkce *isSchemaValid* je zaregistrované schéma XML (URI), podle kterého se ověřuje správnost.

6.7 Zobrazení obsahu XML dokumentů

Aplikace umožňuje také zobrazení obsahu dokumentů XML. Zobrazený obsah se nedá editovat. Uživatel ovšem může takový prostředek uložit na svůj lokální disk, poté upravit prostředek, a znova uložit do úložiště. Takové chování jsem zvolil z důvodů sdílení prostředků. Pokud bychom poskytli ke sdílení nějaký XML dokument a žadatel by dokument upravil, pak by se změny v dokumentu provedly i poskytovateli prostředku. Toto chování by nemuselo vždy býti výhodou. Oracle XML DB na řešení takovýchto problémů poskytuje ACL (acces control list), který řeší přístupová práva k prostředkům.

7 Závěr

Úkolem bakalářské práce bylo prostudovat podporu práce s XML v moderních databázových systémech. Po dohodě s vedoucím práce jsem si vybral předvedení podpory práce s XML u databázového systému Oralce. Podporu s XML u systému Oracle jsem pak měl demonstrovat v ukázkové aplikaci.

Získané teoretické znalosti jsou prezentovány v dokumentační části a odrážejí se v ukázkové aplikaci. Teoretické znalosti jsem převážně čerpal z dokumentace k databozovému systému Oracle. Tento zdroj obsahuje nejaktuálnější informace o podpoře práce s XML u systému Oracle. Strohost dokumentace systému Oracle mě působila časové problémy. Dlouho mi trvalo, než jsem pochopil širší kontext podpory práce s XML u databázového systému Oracle.

Přínos bakalářské práce pro studenty nebo zájemce o tuto problematiku spočívá ve zpracování některých částí z Oracle XML DB Repository nebo z Oracle XML DB Content Connector a dále přínos spočívá v začlenění těchto zpracovaných částí do ukázkové aplikace.

Tato bakalářská práce byla pro mě velkou výzvou. Zkusil jsem si tvorbu rozsáhlého programu, který lze použít v praxi. Další přínosem bylo naučení se samostudia při studiu literárních pramenů. Volnost zadání bakalářské práce mi poskytlo možnost věnovat se dané problematice z různých úhlů pohledů a získat tak široký přehled o databázovém systému Oracle.

Vzhledem ke komplexni podpoře XML, kterou systém Oracle poskytuje, mi dělalo problém vymyslet návrh konkrétní ukázkové aplikace. Dlouho jsem hledal vhodnou formu demonstrace použití nástrojů, které podporují práci s XML. Doufám, že můj návrh dostatečně dobře ukazuje podporu XML u systému Oracle.

Co se týče osobního hodnocení práce, pak mohu vyjádřit se svou prací spokojenost. Povedlo se mi poznat nový databázový systém. Získal jsem nové znalosti a praktické dovednosti z oblastí databází.

Budoucí pokračování práce vidím v ještě větším začlenění podpory XML. Za pokus by stálo poskytnout aplikaci konkrétním uživatelům a na základě toho zjistit, zda by takovou aplikaci uživatelé uvítali a používali.

Dostatečným ohodnocením pro mne bude, když tato práce přiblíží podporu XML u databázového systému Oracle.

Literatura

[1] W3C. *Dokumentace k XML* [online]. c 1996-2003, poslední aktualiazace 16.4.2009 [cit 2009-03-10]. Dostupná z WWW <http://www.w3.org/XML/>.

[2] VANICKÝ, Miroslav. *XML a databáze* [online]. [cit 2009-03-01]. Diplomová práce. Praha, VŠE Praha, 2004, 109 s. Dostupné z WWW: <http://ploppy.vande.cz/dp/DP_XML_a_DB.pdf>.

[3] KOSEK, Jiří. *XML* [online]. c 1999-2009, poslední aktualizace 28.5.1999 [cit 2009-05-11]. Dostupné z WWW: <http://www.kosek.cz/clanky/xml/>.

[4] ORACLE Community. *Oracle XML Technology center* [online]. [cit 2009-05-11]. Dostupné z WWW: <http://www.oracle.com/technology/tech/xml/index.html>.

[5] KREJČÍ, Richard. *Jemný úvod do XML (II) - Základy syntaxe* [online]. c 2003, poslední aktualizace 3.10.2001 [cit 2009-05-11]. Dostupné z WWW: <http://www.grafika.cz/art/webdesign/xml2.html>.

[6] PITNER, Tomáš. *Úvod do značkovacích jazyků* [online]. c 2001-2003, [cit 2009-04-01]. Dostupné z WWW: <http://www.fi.muni.cz/~tomp/slides/pb138/1std-terminology/slides/index.html>.

[7] MLÝNKOVÁ, Irena. *Výukové materiály k předmětu pokročilé technologie XML UK MFF* [online]. Praha: poslední aktualizace 20.1.2009 [cit 2009-04-11]. Dostupné z WWW: <http://www.ksi.mff.cuni.cz/~mlynkova/prg039/slajdy/07XMLvRDBMS.pdf>.

[8] SOBOTKA, Jiří. *Podpora pro práci s XML u databázového serveru Oracle.* [cit 2009-02-2]. Diplomová práce, 2006, FIT VUT Brno.

[9] ORACLE Community. *Dokumentace systému Oracle* [online]. c 2009, [cit 2009-02-2]. Dostupné z WWW: <http://www.oracle.com/pls/db111/homepage>.

[10] BOURRET, Ronald. *XML and Databases* [online]. c 2009, [cit 2009-03-12]. Dostupné z WWW: <http://www.rpbourret.com/xml/XMLAndDatabases.htm>.

[11] ŽIŠKA, Petr. *Nativní XML databáze – nástin teorie* [online]. c 2003, poslední aktualizace 20.2.2003 [cit 2009-03-16]. Dostupné z WWW: <http://interval.cz/clanky/nativni-xml-databaze-nastin-teorie/>.

[12] BLOONBERG Jason, *SCHMELZAR Ron. Service Orient or BeDoomed!* John Wiley and Sons, 2006, 258 s. ISBN 0471768588. [cit 2009-05-12].

[13] ŠEDA, Jan. *Úvod do JDBC* [online]. c 2003, poslední aktualizace 4.3.2003 [cit 2009-05-05]. Dostupné z WWW: Dostupný na adrese <http://interval.cz/clanky/uvod-do-jdbc/>.

[14] SUN MICROSYSTEMS. *Dokumentace k Java J2SE* [online]. Dostupné z WWW: <http://java.sun.com/j2se/1.4.2/docs/api/java/sql/package-summary.html>.

[15] ORACLE Community. *Oracle and Java* [online]. c 2009, [cit 2009-05-2]. Dostupné z WWW: <http://www.oracle.com/technologies/java/index.html>.

[16] MLÝNKOVÁ, I. a kol.: XML technologie. Grada. 2008. 272 s. ISBN 978-80-247-2725-7.

Seznam příloh

Příloha 1. DVD nosič s ukázkovou aplikací, programovou dokumentací a nástroji pro spuštění.