UNIVERZA V MARIBORU

FAKULTETA ZA ELEKTROTEHNIKO,
RAČUNALNIŠTVO IN INFORMATIKO

Marko Bevc

PRIMERJAVA PRISTOPOV K SHRANJEVANJU IN OBDELAVI XML DOKUMENTOV V PODATKOVNI BAZI ORACLE

Diplomska naloga

Maribor, januar 2007

UNIVERZA V MARIBORU

FAKULTETA ZA ELEKTROTEHNIKO,
RAČUNALNIŠTVO IN INFORMATIKO
2000 Maribor, Smetanova ul. 17

Diplomska naloga univerzitetnega študijskega programa

PRIMERJAVA PRISTOPOV K SHRANJEVANJU IN OBDELAVI XML DOKUMENTOV V PODATKOVNI BAZI ORACLE

Študent: Marko BEVC
Študijski program: univerzitetni, Računalništvo in informatika
Smer: Informatika

Mentor: izred. prof. dr. Marjan Heričko
Somentor: doc. dr. Aleš Živkovič
Lektor(ica): Suzana REJAK

Maribor, januar 2007

UNIVERZA V MARIBORU

FAKULTETA ZA ELEKTROTEHNIKO,
RAČUNALNIŠTVO IN INFORMATIKO
2000 Maribor, Smetanova ul. 17

Številka: RI-389

Datum: 20. 12. 2006

SKLEP O DIPLOMSKI NALOGI

1. **Marko BEVC**, absolvent univerzitetnega študijskega programa računalništvo in informatika, smer informatika, izpolnjuje pogoje, zato se mu dovoljuje diplomsko delo.

2. **Tema diplomske naloge** je s področja Inštituta za informatiko pri predmetu **GRADNJA INFORMACIJSKIH SISTEMOV**.

   MENTOR: izred. prof. dr. Marjan HERIČKO
   SOMENTOR: doc. dr. Aleš Živkovič

3. **Naslov diplomskega dela:**
   PRIMERJAVA PRISTOPOV K SHRANJEVANJU IN OBDELAVI XML DOKUMENTOV V PODATKOVNI BAZI ORACLE

4. **Vsebina diplomskega dela:**
   Raziščite in opišite podporo tehnologijam XML v podatkovnih bazah. Primerjajte različne pristope k shranjevanju XML dokumentov v podatkovni bazi Oracle.

5. Diplomsko delo je potrebno izdelati skladno z "Navodili za izdelavo diplomskega dela" in ga oddati v treh izvodih do 20.12.2007 v referatu za študijske zadeve.

PREDSTOJNIK INŠTITUTA — DEKAN

red. prof. dr. Ivan Rozman — red. prof. dr. Igor Tičar

MENTOR

izred. prof. dr. Marjan Heričko

ZAHVALA

Zahvaljujem se mentorju dr. Marjanu Heričku za pomoč in vodenje pri opravljanju diplomske naloge. Prav tako se zahvaljujem celotnima ekipama tako podjetja Actual I.T. kot Inštituta za informatiko, FERI Maribor.

Posebna zahvala velja staršem, ki so mi omogočili študij in me pri njem podpirali ter vzpodbujali. Brez njihove pomoči ne bi uspel uresničiti zastavljenih ciljev.

PRIMERJAVA PRISTOPOV K SHRANJEVANJU IN OBDELAVI XML DOKUMENTOV V PODATKOVNI BAZI ORACLE

Ključne besede: **podatkovne baze, XML dokumenti, Oracle, XML shema, Java, storitvena arhitektura, primerjava različnih pristopov shranjevanja, zanesljivost sistemov**

UDK: 007:004.738.5(043.2)

Povzetek

Diplomska naloga opiše in predstavi novosti v podatkovni bazi Oracle10g R2 glede podpore tehnologijam XML. Predstavljene so osnovne možnosti in prednosti uporabe mehanizmov za shranjevanje in obdelavo XML dokumentov ter povezava s podatkovno bazo. Prav tako so predstavljeni rezultati primerjave shranjevanja XML dokumentov v relacijskih tabelah in XML tabelah z uporabo sheme ali brez.

Podrobneje so predstavljene novosti, ki jih prinaša tehnologija XML v podatkovno bazo Oracle in možnosti uporabe ter migracije obstoječih sistemov. Predstavljene so tudi osnovne tehnologije povezane s pojavom jezika XML in njegovo integracijo v podatkovne baze. Izdelana je tudi primerjava obstoječih rešitev z novimi tehnologijami na primeru sheme eSlog. Analiza lahko služi kot referenčna primerjava za migracijo obstoječih implementacij rešitev informacijskih sistemov na nove tehnologije.

Praktični del zajema načrtovanje in implementacijo testne aplikacije informacijskih rešitev ter izvedbo meritev in primerjavo rezultatov. Predstavljeni so rezultati meritev odzivnosti posameznih pristopov k shranjevanju XML dokumentov v Oracle podatkovni bazi. Poizkušali smo poiskati in locirati vzroke za posamezne pristope ter izpostaviti prednosti in slabosti posameznih pristopov k hranjenju XML dokumentov v podatkovni bazi Oracle. Opisani so tudi primeri uporabe shranjevanja XML dokumentov v podatkovni bazi.

COMPARING STORING APPROACHES AND MANAGING XML DOCUMENTS IN ORACLE DATABASE

Key words: **databases, XML documents, Oracle, XML Schema, Java, service oriented architecture, comparison of different approaches, system performance**

UDK: 007:004.738.5(043.2)

Abstract

My diploma work describes and introduces changes in Oracle10g R2 database in the aspect of XML technology. Basic options and mechanisms for saving and processing XML documents are introduced. Analysis for comparing conventional-table and XML data is also introduced.

Features introduced with XML technology in Oracle database and new possibilities and usage and migrations of existing systems are also described in detail. There is also a comparison between existing solutions and integration with new technologies based on XML has been made. We have made comparison of existing solutions with new technologies using eSlog data scheme for data storage. It can be used as reference comparison for migration of existing implementation of information systems on new technologies.

The practical part of my diploma consists of planning and implementation of test application, measurements, analysis and comparison of analyzed results. Results gathered and measured in this diploma work are also analyzed and statistically compared. Measurements are made on Oracle database. We tried and locate reasons for each approach for saving XML documents in Oracle database. We also have shown basic use cases for saving and manipulating XML documents in Oracle database.

VSEBINA

UPORABLJENE KRATICE

FERI - Fakulteta za elektrotehniko, računalništvo in informatiko

XSLT - eXtensible Stylesheet Language Transformations

EAN - European Article Numbering

XML - eXtended Markup Language

SGML - Standard Generalized Markup Language

ER - Entitetno Relacijski

XSD - eXtended Scheme Definition

XLS - Extensible Stylesheet Language

DTD - Data Type Definition

DB - DataBase

CLOB - Character Large OBject

BLOB - Binary Large OBject

HTTP - Hyper Text Transfer Protocol

FTP - File Transfer Protocol

WebDAV - Web Distributed Authoring and Versioning

RIP (EDI - Electronic Data Interchange) - Računalniška Izmenjava Podatkov

SQL - Structured Query Language

CPU - Central Processing Unit

API - Application Programming Interface

BLOB - Binary Large Object

CLOB - Character Large Object

OS - Operating System

VM - Virtual Machine

GC - Garbage Collector

SOAP - Simple Object Access Protocol

RPC – Remote Procedure Call

1 UVOD

1.1 Področje in namen raziskav

Razvijalci, ki razvijajo rešitve na osnovi sodobnih XML tehnologij, se velikokrat srečujejo s težavami ob hranjenju podatkov in dokumentov v podatkovnih bazah. Načeloma lahko XML dokumente shranimo tudi v klasične entitetno relacijske tabele, a težave nastopijo predvsem v kasnejši fazi pri povpraševanju po podatkovni bazi in uporabi podatkov. Predvsem se kot težavno, poleg samega shranjevanja dokumentov izkaže tudi vzdrževanje in iskanje po zbirki shranjenih dokumentov. Z večanjem količine shranjenih podatkov prihaja do nepreglednosti in težav ob iskanju. XML dokumenti imajo urejeno in določeno strukturo (XML sheme). Vnašanje in iskanje le-teh po relacijski bazi, je lahko brez zavedanja o njihovi strukturi zelo zamudno in redundantno opravilo. Tako se izkaže, da bi bila optimalna rešitev podatkovna baza, v kateri je mogoče izkoristiti izrazno moč in prednosti XML tehnologij že ob samem shranjevanju in uporabi podatkov. V diplomski nalogi se bomo omejili na področje primerjave shranjevanja XML dokumentov v podatkovni bazi Oracle 10g R2, tako v klasični relacijski obliki kot vse bolj razširjeni XML obliki. Opisane bodo osnovne značilnosti obeh pristopov in podani njihovi osnovi primeri uporabe s slabostmi in prednostmi takšne uporabe. Podana bo analitična metoda meritev in primerjava obeh pristopov k shranjevanju in obdelavi XML dokumentov. Rezultati bodo podrobneje opisani skozi analizo izvedenih meritev.

V današnjem času vsesplošne informatizacije, razvoja in širitve Interneta ter globalne povezljivosti je potrebno izbrati optimalno tehnologijo za shranjevanje podatkov in zagotoviti čim bolj učinkovito shranjevanje XML dokumentov v podatkovni bazi in sicer v obliki, primerni za elektronsko izmenjavo dokumentov. Pomembno je optimizirati vse dele informacijskega sistema in s tem zagotoviti potrebno konkurenčno prednost pred ostalimi podjetji. Glede na veliko razširjenost in popularnost XML tehnologij je potrebno dodatno zagotoviti tudi učinkovit način hrambe in upravljanja teh podatkov – XML dokumentov. Prav tako se vse bolj uveljavlja tudi razpršeno procesiranje in shranjevanje podatkov. [1]

Namen diplomskega dela je primerjati različne pristope k shranjevanju in obdelavi XML dokumentov v podatkovni bazi in ugotoviti njihove prednosti ter slabosti.

1.2 Struktura dela

Diplomsko delo je sestavljeno iz dveh delov in sicer iz teoretičnega in praktičnega dela. Teoretični del sestavljajo poglavja 2, 3 in 4.

V drugem poglavju opišemo stanje trenutnih rešitev in tehnologij povezanih z jezikom XML. Predstavljeni bodo trenutne smernice v industriji kot tudi načini shranjevanja XML dokumentov. Sledi razlaga pomena tehnologij XML v podatkovni bazi skozi njihov razvoj, motivacija za njihovo vpeljavo in primeri uporabe v praksi – Oracle 10g R2. Četrto poglavje je namenjeno predstavitvi podatkovne baze Oracle in podpori XML tehnologij v njej. Predstavljena bo osnovna arhitektura podpornega mehanizma za XML tehnologijo v podatkovni bazi Oracle 10g R2, njegove lastnosti in struktura podatkovnega tipa XML Type, ki je na voljo za hranjenje dokumentov. Ogledali si bomo njegove prednosti ter primerjali strukturirano in nestrukturirano shranjevanje dokumentov. Na kratko bo predstavljena podpora XML shemam in njihovi uporabi. Kot ključna funkcionalnost pa bo prikazana SQL/XML dvojnost, s pomočjo katere lahko poljubno mešamo funkcionalnost »klasičnih« relacijskih tabel in povpraševanj z novimi tehnologijami XML. Tako ne samo, da zelo olajšamo prehod na uporabo sodobnih tehnologij, pač pa združimo izrazno moč obeh tehnologij in te uporabimo v optimalnih primerih. Predstavljen bo XML repozitorij in njegova organizacija za upravljanje z XML dokumenti in njihovo validacijo. Na kratko bomo predstavili povpraševanje po podatkih (XQuery, XPath) v podatkovni bazi. Na koncu bo predstavljena še programska podpora za delo s podatki in dokumenti v sodobnih programskih jezikih.

Teoretični del nato v nadaljevanju nadgradimo in podkrepimo še s praktičnimi primeri uporabe in analizo rezultatov meritev posameznih načinov shranjevanja dokumentov v Oracle XML DB. Kot podatkovni model je uporabljena shema elektronskega računa (eSlog). Iz teh dokumentov bodo kreirane normalizirane tabele, ki bodo uporabljene za referenčno primerjavo z XML shranjevanjem dokumentov. V šestem poglavju bo predstavljena analiza rezultatov meritev. Le-ta bo zajemala primerjavo funkcionalno identičnih podatkov, shranjenih v podatkovni bazi. Predstavljene bodo možnosti aplikativne uporabe in načini dostopa, ki so na voljo preko raznih programskih vmesnikov (API-jev) in knjižnic. Podanih bo tudi nekaj primerov uporabe teh vmesnikov.

2 POMEN XML DOKUMENTOV

2.1 Trendi v industriji

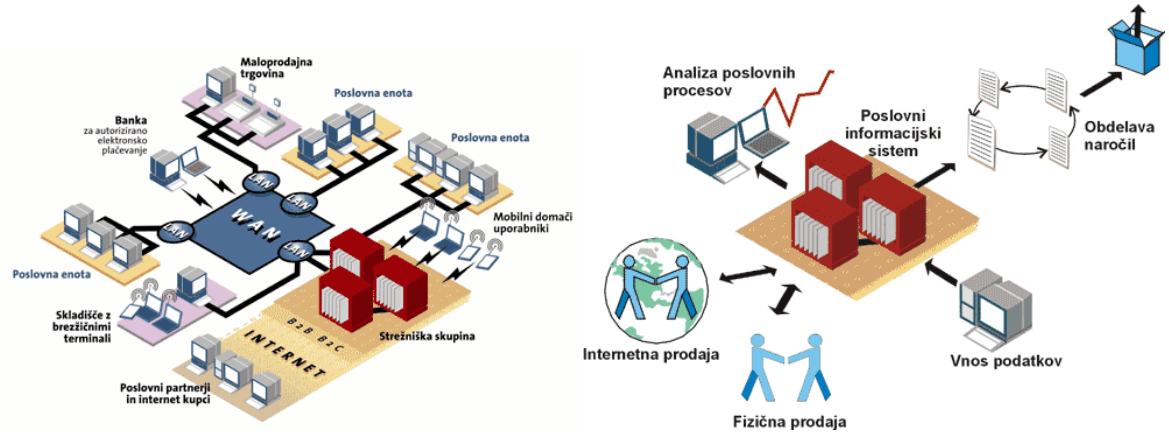
Tehnologija XML (eXtensible Markup Language) [1] spada med najpomembnejše tehnologije, ki so se pojavile v zadnjem času. Omogoča prenosljivost podatkov ne glede na izbrano okolje. Lahko rečemo, da se vedno več podatkov pojavlja v obliki XML. Uporaba tehnologije pa se bo verjetno še širila, predvsem na področjih elektronske izmenjave podatkov in elektronskega trgovanja, povezovanja poslovnih sistemov, objavljanja, shranjevanja informacij ter razvoja programske opreme.

S širitvijo uporabe tehnologije XML raste tudi potreba po shranjevanju XML dokumentov v podatkovnih bazah. Ker obstajajo podatkovno naravnani in dokumentno naravnani XML dokumenti, je smiselno pri ravnanju z njimi uporabiti različne pristope. Dandanes se vedno več podatkov oblikuje, obdeluje in prenaša v obliki XML. Prav tako je opaziti porast uporabe aplikacij s strukturiranimi podatki in semi-strukturiranimi dokumenti. Ob tem pa prihaja do vedno večje standardizacije besednjakov, ki je potrebna za učinkovito medsebojno izmenjavo dokumentov in sporočil. V vse večji poplavi podatkov in informacij se dodatno pojavlja vedno večja potreba po enotnem upravljanju vseh vrst podatkov. Tehnologije, ki so povezane z jezikom XML, tako v vedno večjem obsegu zasledimo tudi v sklopu sistemov za upravljanje podatkovnih baz. [2]

Na tržišču je že mogoče zaslediti produkte, ki omogočajo shranjevanje XML dokumentov v podatkovnih bazah. Enega od večjih problemov predstavlja pomanjkanje standardiziranega povpraševalnega jezika za XML. Prednost Oracle XML DB se pokaže v dualnosti načina vračanja in obravnave podatkov (SQL/XML). Čeprav "XML podatkovne baze" verjetno še nekaj časa ne bodo popolnoma "zrele", lahko predvsem zaradi uporabljenih **odprtih standardov** predstavljajo eno izmed pomembnejših smeri razvoja na področju podatkovnih baz. [28,29]

Zaradi možne velike razširjenosti uporabe XML tehnologij na številnih področjih se je pojavila potreba po shranjevanju XML dokumentov v podatkovnih bazah. Za shranjevanje lahko uporabimo različne vrste podatkovnih baz, od relacijskih, objektnih, objektno-relacijskih do raznih drugih. Če se ozremo na razpoložljive podatkovne baze na tržišču, ugotovimo, da v splošnem nobena od njih ni popolnoma primerna za shranjevanje XML dokumentov v osnovni obliki. Pretvorba XML dokumentov v objekte lahko povzroči

precej preglavic, še večje težave pa lahko imamo pri pretvorbi XML dokumentov v relacijske modele. [3]



Slika 1: Povezovanja s pomočjo XML in odprtih standardov [2]

Čeprav so nekateri mnenja, da bi lahko bili XML dokumenti shranjeni tudi v obliki datotek, to velikokrat ne zadostuje. Podatkovne baze poleg večje hitrosti delovanja namreč ponujajo še upravljanje nad podatki, učinkovite povpraševalne jezike (SQL, OQL itd.), boljše varnostne mehanizme, možnost postavljanja različnih omejitev za posamezne podatke in boljšo integracijo podatkov.

Klasične relacijske podatkovne baze resda ne nudijo velike podpore XML dokumentom, vendar pa s svojimi razširitvami in s pomočjo novih standardov (SQL3, DOM, SQLJ itd.) nudijo funkcionalnost, potrebno za shranjevanje tovrstnih dokumentov [4, 5]. Podobno velja tudi za objektne podatkovne baze, pri katerih pa shranjevanje XML dokumentov za razliko od relacijskih podatkovnih baz temelji na uporabi objektov.

In prav o tem, kako je najbolje shranjevati XML dokumente v podatkovnih bazah, bomo govorili v diplomskem delu. Na splošno poznamo podatkovno naravnane in dokumentno naravnane XML dokumente. Obe vrsti dokumentov uporabljata različne koncepte za shranjevanje in pripravo podatkov. Na tržišču obstaja že kar nekaj produktov s področja XML in podatkovnih baz. Najdemo lahko dodatke za podatkovne baze, razširitve podatkovnih baz, XML strežnike in sisteme za upravljanje z vsebinami. Na koncu poglavja so opisani tudi različni povpraševalni jeziki za XML, ki se uporabljajo za iskanje, pridobivanje in manipulacijo podatkov. So zelo podobni nekaterim obstoječim povpraševalnim jezikom. Na primer XQuery s SQL jezikom.

2.2 Jezik XML

Za razliko od programskih jezikov, ki obravnavajo predvsem postopke, opisni jeziki opisujejo podatke. Paradigma opisovanja podatkov je označevanje, kar pomeni, da nek del podatkov označimo z določeno značko ali oznako (*tag*). Oznake predstavljajo tipe, ki jih lahko tudi gnezdimo. Na primer, v najbrž najbolj znanem jeziku za označevanje – HTML (HyperText Markup Language), oznaka <*p*> pomeni začetek, </*p*> pa konec odstavka, podatki med oznakama pa so besedilo odstavka.

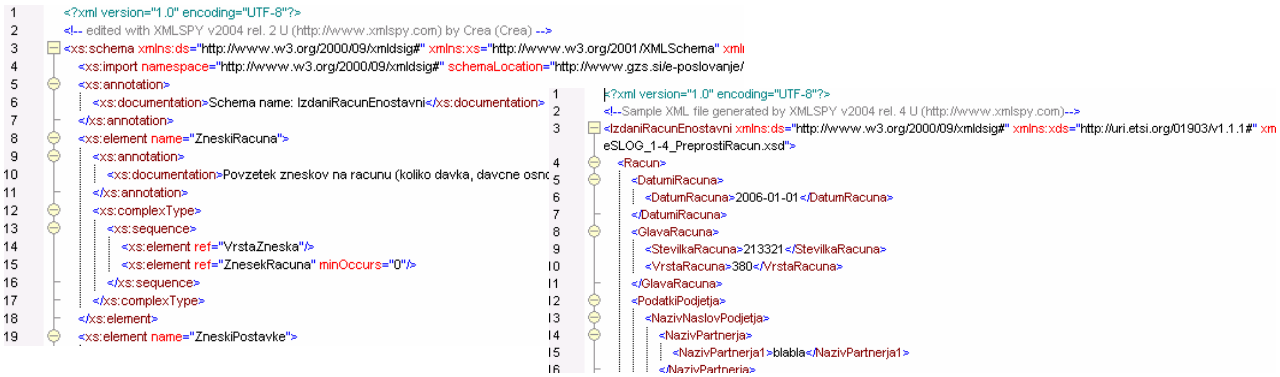
XML (Extensible Markup Language) je poskus, da bi ustvarili standardni način shranjevanja poljubnih tipov strukturiranih podatkov, ki je tekstnem formatu, razumljivem človeku in računalniku. Pomembna je predvsem vizija, da bi dejansko skoraj vse podatke istega tipa shranjevali na enak strukturiran način, kar bi omogočilo izmenjavo podatkov med različnimi programskimi paketi, aplikacijami ter operacijskimi sistemi. To bi v spletu omogočilo uporabo strukturiranih podatkovnih baz in omogočilo nova področja uporabe, kot recimo avtomatsko obdelavo. Ob tem je zelo važno opozoriti, da je jezik XML le okolje za označevanje besedila in sam po sebi se ne prinaša kake proceduralne funkcionalnosti. Na primer, dve podatkovni bazi lahko izmenjujeta podatke kot XML datoteke oz. tok podatkov (stream).

Zanimiva je tudi primerjava jezikov XML in SGML (Standard Generalized Markup Language), katerega podmnožica je XML. Vsak pravilno strukturiran dokument v XML je v skladu s standardom SGML. SGML je namreč zelo kompleksen standard in ta kompleksnost je tudi ovira pri prenosu v uporabo. Sam XML pa poskuša ohraniti pogosto uporabljene del funkcionalnosti, manj uporabljene lastnosti pa zaradi ohranjanja enostavnosti niso vključene.

XML je meta-format, kar pomeni, da določa le strukturo, ne pa tudi vsebine predstavitve nekega tipa informacij. Torej, sintaksa je standardna, semantika pa ostaja nestandardna. Na primer, HTML je specifičen format za shranjevanje podatkov o povezanem besedilu (hypertext), kjer sta natančno določeni tako semantika kot sintaksa. Prav tako vsebuje kot meta-format tudi opisne informacije o podatkih, kar predstavlja dodatno nalaganje »odvečnih« informacij kot tudi strukturiranost dokumentov. XML

omogoča nadgradnjo funkcionalnosti samega HTML, na primer z uporabo XSL (Extensible Stylesheet Language) lahko definiramo obliko prikaza nekega besedila tako, da definiramo, kako se prikaže nek tip podatkov, definiran z določeno oznako (na primer naslov, tabela, pa tudi poljubni kompleksni tipi). To omogoča učinkovit prikaz nekega tipa podatkov z uporabo splošne programske opreme za prikaz podatkov.

Bistvo uporabnosti jezika XML je torej definicija lastnih tipov in posplošenje shranjevanja le-teh. To naredimo tako, da definiramo sintaktične lastnosti posameznih oznak z jezikom DTD (Document Type Definition), ki je lahko v isti ali v ločeni datoteki. Alternativo na višjem nivoju z bolj kompleksnimi tipi pa predstavlja XML shema, kjer lahko podatkom znotraj neke oznake pripišemo splošnejše tipe. XML shema definira nekaj osnovnih elementov ter z mnogo možnosti razširjanja, kot so sekvenca (sequence), element (element), enostavni tip (simpleType), kompleksni tip (complexType), omejitev (restriction), niz (string), decimalno število (decimal). Na tak način definicija tipov vsebuje tudi elemente semantike, enostaven primer tega so omejitve intervalov za števila. Nekaj primerov uporabe sheme in XML dokumenta lahko vidimo na Sliki 2.



Slika 2: Primer XML sheme in z njo skladnega dokumenta

2.2.1 Tehnologije, povezane z jezikom XML

Jezik XML bi brez podpornih tehnologij za njegovo manipulacijo in delo z njim bil težje obvladljiv in manj uporaben. Ravno ti podporni mehanizmi ga postavijo v ospredje

današnje uporabe v izmenjavi podatkov, kot tudi njihovi obdelavi in izmenjavi. Za delo z jezikom XML se uporabljajo tehnologije povpraševanja XPath in XQuery, transformacije XSLT in programsko manipulacijo podatkov z modeloma DOM in SAX. Ravno v transformaciji in povpraševanju po podatkih se pokaže glavna prednost uporabe jezika XML.

Najpogostejši **XPath** izraz, od kod izvira tudi njegovo ime, je t.i. izraz za pot. Napisan je kot sekvenca korakov za pot od začetnega (ponavadi trenutnega) do končnega vozlišča. Koraki so ločeni z znaki »/«. Vsak korak ima tri komponente:

- os,
- test vozlišča,
- predikat.

Definirani sta dve notaciji. Prva znana kot *skrajšana oblika* je bolj kompaktna in dovoljuje lažje branje ter bolj intuitivno razumevanje izraza in konstruktov. *Popolna sintaksa* je mnogo bolj obširna in dovoljuje več opcij dostopa, ter je bolj opisna. Skrajšana oblika bi bila:

```
/A/B/C
```

Izberemo element C, ki je otrok B elementa, ki je otrok A-ja v XML dokumentu. XPath sintaksa je namenjena za URI (Uniform Resource Identifier) sintakso in pot za datoteke.

Bolj zapletena oblika bi bila z uporabo osi namesto privzete »otrok« sintakse, uporabo testa vozlišča namesto imena ali predikata, ki se navede v oglatih oklepajih za vsakim korakom. Na primer:

```
A//B/*[1]
```

Izberemo prvi element ('[1]'), kateregakoli imena ('*'), ki je otrok ('/') elementa B, ki je sam otrok nekje globlje ('//') elementa A, ki je otrok trenutnega vozlišča (nismo pričeli z korenskim vozliščem '/').

XQuery je povpraševalni jezik (z nekaterimi lastnostmi programskega jezika) za povpraševanje po XML dokumentih. Semantično je zelo podoben jeziku SQL. XQuery 1.0 je bil razvit v okviru XML Query delovne skupine znotraj konzorcija W3C. Razvoj je bil

tesno povezan s razvojem tehnologije XSLT 2.0, katero je razvijala delovna skupina XSL Working Group. Skupini si delita odgovornost za nastanek XPath 2.0, ki je razširitev XQuery 1.0. XQuery 1.0 je nastal pod okriljem W3C Proposed Recommendation 21 novembra 2006.

XQuery ponuja sredstva za pridobivanje in manipulacijo podatkov iz XML dokumentov ali poljubnih XML podatkovnih virov (podatkovne baze ali office dokumenti). XQuery uporablja XPath izraze za doseganje določenih delov XML dokumenta. Uporablja SQL FLWOR (For-Let-Where-OrderBy-Return) izraze za izvajanje združitev podatkov. Jezik prav tako omogoča gradnjo novih XML dokumentov. Kjer poznamo obstoječe elemente in atribute jih lahko uporabimo, v drugih primerih uporabimo dinamične konstrukte, ki jih poljubno lahko gnezdimo. Jezik temelji na drevesni strukturi in vsebuje sedem vrst vozlišč: dokumentna vozlišča, elementi, atributi, tekstovna vozlišča, komentarji, procesne inštrukcije in naslovna področja. Vse vrednosti so tipizirane kot sekvence (singleton vrednost je sekvenca dolžine ena). Vrednosti sekvence so lahko vozlišča ali atomične vrednosti (števila, nizi, logične vrednosti in ostali elementi definirani v XML shemi).

XQuery 1.0 ne vsebuje funkcionalnosti za spreminjanje XML dokumentov ali podatkovnih baz, kot to uporablja XUpdate. Prav tako nima polne podpore za t.i. »full text search« iskanje. Te funkcionalnosti bodo na voljo v različici 2.0.

XQuery lahko izvaja XML transformacije s sledečimi lastnostmi:

- neodvisno od podatkov,
- deklerativno,
- na visokem nivoju,
- brez stranskih učinkov,
- strogo tipizirano.

V naslednjem primeru izberemo unikatne primerke podjetij iz eRačuna:

```
<html><head/><body>
{
```

```
    for $act in doc("eRacun.xml")//Racun
    let $speakers := distinct-values($racun//Podjetje)
    return
      <span>
        <h1>{ $racun/NAZIV/text() }</h1>
        <ul>
        {
          for $naziv in $nazivi
          return <li>{ $naziv }</li>
        }
        </ul>
      </span>
  }
  </body></html>
```

XQuery je funkcijski jezik, ki je sestavljen samo iz izrazov brez stavkov. Za kreiranje funkcije treba prav tako napisati izraz, ki se ovrednoti:

```
  declare function local:doubler($x) { $x * 2 }
```

Nekaj primerov aplikacij kjer se lahko uporabi XQuery:

- ekstrahiranje informacij iz podatkovne baze ali spletne storitve,
- generiranje poročil iz XML podatkovne baze,
- iskanje tekstovnih dokumentov na spletu,
- izbira in pretvorba XML podatkov v obliko XHTML za objavo na spletu,
- splošna transformacija XML dokumentov v poljubno obliko.

Kljub temu, da je XQuery narejen kot povpraševalni jezik omogoča tudi transformacije nad zbirkami XML dokumentov. Kot tak se prekriva z funkcionalnostmi XSLT jezika, ki je bil razvit samo z namenom transformacije XML dokumentov v poljubni format. Standarda sta bila razvita posebej in kljub temu imata enak podatkovni model, tipski sistem, zbirko knjižnic in oba vsebujeta XPath kot podjezik.

XSLT se primarno uporablja kot jezik za oblikovanje XML dokumentov, XQuery pa kot povpraševalni jezik podoben jeziku SQL. Tako je XQuery boljši za XML v XML transformacije podatkov, povpraševanja in pridobivanje podatkov. Kot boljši se XSLT izkaže za XML v ne-XML (kot E-text ali HTML) transformacije podatkov, obravnavo pol

strukturiranih dokumentov, dogodkovno ali vzorčno naravnanih algoritmov ter filtrirnih aplikacij.

SAX je sklop programskih vmesnikov in deluje kot serijski pregledovalnik (parser) . je mehanizem za zaporedno branje podatkov iz XML dokumenta. Je popularna in preprostejša alternativa Document Object Model (DOM) modelu. Ime je izpeljanka iz »Simple API for XML«.

Pregledovalnik, ki implementira SAX, obravnava XML kot enostaven serijski tok podatkov. Tok je enosmeren, da bi dobili kak prejšnji podatek je ponovno treba pričeti od začetka. Porabi definitivno manj pomnilniške kapacitete kot DOM, kjer imamo vedno v pomnilniku celotno drevo. Je dogodkovno naravnan model v katerem programer zagotovi metode, ki se izvedejo med prehodom branja XML dokumenta.

DOM je nastal pod okriljem World Wide Web konzorcija (W3C) kot odgovor na pojav različnih novih modelov za HTML, uporabljenih v internetnih brskalnikih. Obstoječi vmesniki podjetij so bili združeni v vmesne DOM-e. Document Object Model (DOM) je platofrma in jezik, ki je neodvisen za standardno predstavitev objektnega modela HTML ali XML kot drevo. Pravzaprav gre celo za izdelek kot tak – DOM.

Zaradi podpore povezav nazaj (*parent* in *previous* zlogovna ukaza) ter dovoljuje poljubne modifikacije v drevesu, mora implementacija hraniti ves prebran dokument do sedaj, oz. vsaj neko prebrano verzijo. Tako je DOM najpogosteje uporabljen za aplikacije kjer mora biti dokument naključno dostopen in ne sekvenčno. Če potrebujemo sekvenčni in verjetno eno-prehodni dostop je SAX model gotovo boljša, hitrejša in manj pomnilniško potratna varianta.

2.3 Vrste XML dokumentov

Pri XML dokumentih smo velikokrat v dilemi, saj ne vemo, ali gre zgolj za podatke ali gre v bistvu za dokumente. Običajno jih delimo na dokumente podatkovne narave in tiste, ki so dokumentne narave. Čeprav je ločnico med njimi težko potegniti, lahko rečemo, da je pri prvih pomembna predvsem struktura, pri drugih pa sta lahko v ospredju npr. tudi zaporedje in hierarhija elementov.

Ali imamo opravka s podatki ali dokumenti, je pomembno pri izbiri podatkovne baze, saj so za shranjevanje podatkov prav gotovo najprimernejše klasične relacijske oz. objektno-relacijske podatkovne baze, medtem ko so za dokumente boljše druge rešitve, kot npr. sistemi za upravljanje z vsebinami (t. i. content management systems). Čeprav je v klasičnih podatkovnih bazah možno shranjevati tudi dokumente, se temu izogibamo zaradi kasnejših težav ob povpraševanjih in iskanjih po takšni vrsti shranjenih podatkov. Naletimo pa tudi na slabše odzivne čase povpraševanja in povečano procesiranje na baznem strežniku. Podobno pa tudi sistemov za upravljanje z vsebinami ne uporabljamo za shranjevanje navadnih objektno-relacijskih podatkov.

Dokumenti XML se delijo na dve osnovni vrsti:

- podatkovno naravnani dokumenti XML (vsebujejo podatke, pri čemer je vsak podatek opisan z meta podatki; npr. naročilo),
- dokumentno naravnani dokumenti XML (vsebujejo neločljivo vsebino, ki je zgolj po delih ni moč opisati z meta podatki; npr. članek, poglavje ali zapisnik sestanka).

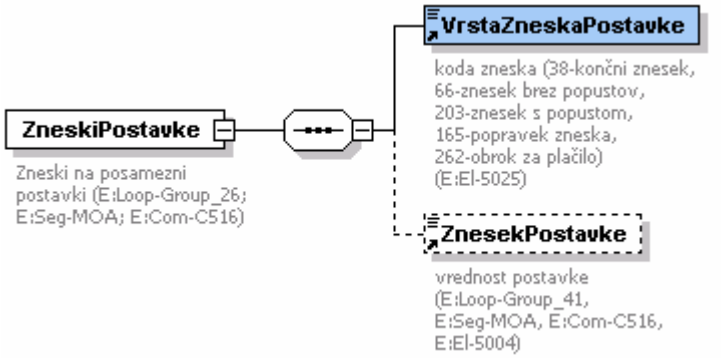
Shema XML služi kot osnova za definiranje in preverjanje pravilnosti dokumentov XML in je zapisana v jeziku XML. Ko uporabnik izdela dokument XML, ga najprej obdela tolmač (parser), ki prepozna osnovne napake (nedovoljeni znaki, ipd.). Nato potrjevalec (validator) glede na pripadajočo shemo poišče semantične, oziroma vsebinske napake. Če je dokument ustrezen oz. skladen s shemo, se obdelava nadaljuje in podatki se npr. lahko začno uporabljati. V nasprotnem primeru program sporoči napako in obdelava se neuspešno konča.

Podatkovno naravnani XML dokumenti

Za podatkovno naravnane dokumente je značilno, da poleg drugih elementov vsebujejo predvsem podatke, zapisane v obliki elementov s postavljenimi vrednostmi atributov. Imajo dokaj enostavno strukturo, pri čemer zaporedje pojavljanja posameznih elementov največkrat ni pomembno. Na dokaj enostaven način jih je možno pretvoriti v relacijski oz. objektni model.

Zaradi pretvorb XML dokumentov v relacijske oz. objektne modele dostopamo do njih s pomočjo standardnih povpraševalnih jezikov SQL, OQL itd.

Tipični primeri podatkovno naravnanih dokumentov so naročila, ceniki, vozni redi ipd., ki si jih danes številna podjetja že izmenjujejo v obliki XML. Za njih v splošnem velja, da so bolj kot predstavitvi namenjeni nadaljnji računalniški obdelavi. Poenostavljen primer naročila prikazuje Slika 3. Račun eSlog-a je dokumentno naravnan a kot primer podatkovne naravnanosti je prikazan primer postavk.



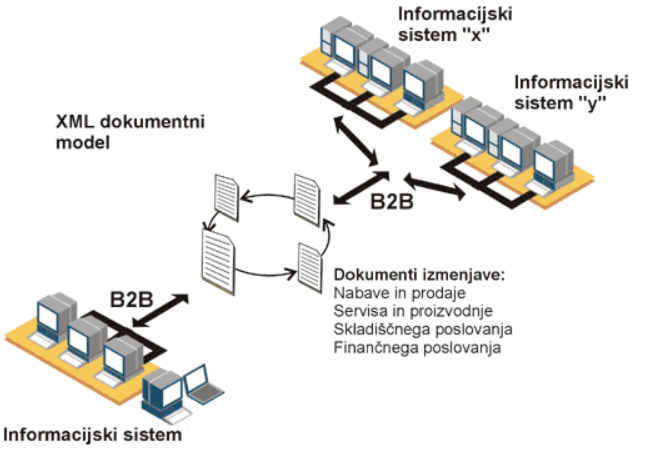
Slika 3: Primer podatkovno naravnanega XML dokumenta

Ker lahko na internetu v različnih katalogih najdemo obširne opise produktov, se na prvi pogled zdi, da so ti opisi dokumentno naravnani. V resnici pa gre v večini primerov za podatkovno naravnane opise, sestavljene iz vnaprej oblikovanih delov, ki si sledijo v točno določenem zaporedju.

Dokumentno naravnani XML dokumenti

Za dokumentno naravnane XML dokumente je značilno, da opisujejo predvsem besedilo ali kakšen drug dokument z dodanimi oznakami, ki dajejo delom besedila poseben pomen. Kljub podatkovni naravnanosti oznak ima celoten dokument dokumentno naravo.

Za dokumente je značilno, da je pogosto treba upoštevati hierarhijo elementov, zaporedje elementov ipd. Dokumentno naravnani XML dokumenti, kot npr. knjige, elektronska pošta, oglasi itd., so bolj kot nadaljnji računalniški obdelavi namenjeni predstavitvi. Poenostavljen primer potovanja in izmenjave dokumenta prikazuje Slika 4, kjer je razvidno, da pri dokumentnih sistemih prav tako obstaja definiran tok izmenjave in uporabe dokumentov.



Slika 4: Primer dokumentno naravnanih XML dokumentov

2.4 Standardizacija XML dokumentov in tehnologij

Večina standardov temelji na XML, ki ga določa W3C. Za XML se lahko definirajo poljubni tipi, ki so shranjeni v repozitorijih. XML dokument in njegova struktura je definirana preko t.i. sheme shem, ki definira samo shemo, ta pa potem XML dokument. OASIS/XML.org in BizTalk.org sta konkurenčna repozitorija. Ogrodje (framework) predstavlja usklajeno skupnost različnih semantičnih definicij tipov, predstavljenih v DTD ali XML sheme. Taka ogrodja so eCo, ebXML in BizTalk. V ogrodja pa se vklapljajo vsebinski tipi, ki imajo določeno semantiko. Ti so XML/EDI, Simpl-EDI, cXML, OAGI, Requisite, FINXML, CBL, HL7 in drugi. Ogrodja pri tem tudi določajo proces sprejemanja vsebinskih tipov. Rosetta je od ostalih pristopov širša, saj pokriva obe stopnji, tako ogrodja kot vsebine, njena arhitektura pa ne temelji na repozitorijih. [30]

RosettaNet je nastal kot eden izmed najnatančnejših predlogov, ne standardizira le sporočil, temveč tudi procese ter protokole. Zato ne potrebuje niti ogrodja niti repozitorija, saj jih implicitno pokriva, čemur pravimo vertikalni standard. Sporočila v RosettaNet temeljijo na definiciji slovarjev (dictionaries), ki opisujejo lastnosti (recimo: ime računa, ulica, mesto, datum dobave, ipd.), ali entitete (entities), sestavljene iz lastnosti (properties) (recimo: poročilo o prodaji, opis poročila, naročilo, podatki o področnem davku, ipd.). Slovarji vsebujejo tudi instance entitet (entity instances), ki opisujejo množice (na primer: valuta, jezik, tip kreditne kartice), ter osnovne podatkovne entitete (fundamental data entities), ki opisujejo tipe (na primer: številka bančnega računa, številka čeka, časovni žig,

ipd.). Določeni so splošni slovarji za določene dejavnosti in področja dela, pa tudi specifični slovarji. **Protokoli** v RosettaNet pokrivajo 8 nivojev nad transportnim nivojem omrežja, od spodaj navzgor: *varnostni* (security), *prenosni* (transfer), *sporočilni* (message handling), *agentni* (agent), *storitveni* (service), *procesni* (process), *transakcijski* (transaction), *akcijski* (action). **Procesi** RosettaNet so razdeljeni v kompleksno hierarhijo s tremi nivoji, ki omogoča zelo kompleksne operacije. Na primer, v grozdu (cluster) o upravljanju z zalogami (inventory management), v segmentu o sodelovanju pri napovedovanju (collaborative forecasting), je proces (Partner Interface Process) o obveščanju prejemnika o napovedih naročil (notify of order forecast), podrobnosti katerega določa pogodba o trgovanju (Trading Partner Agreement). [27]

Microsoftov **BizTalk** predstavlja ovojnico za podatke XML dokumentov. To omogoča, da lahko nek poljuben dokument v XML prilagodimo dogovoru, ki omogoča obravnavo znotraj strežnika BizTalk. Microsoft ponuja tudi repozitorij za podatkovne tipe, BizTalk.org. [5]

Ogrodje **eCo** je ustvarila neprofitna organizacija CommerceNet. eCo temelji na semantični integraciji tipov, registraciji ponudb in uporabi agentov pri kupovanju. eCo daje velik poudarek na podporo velikemu številu uporabnikov trga. eCo sestavljajo naslednji nivoji: podatkovni element, dokument, interakcija, storitev, poslovni nivo, trg; najvišji nivo pa je Commerce Network. Na najvišjem nivoju lahko izvajamo poizvedbe o lastnostih posameznih trgov. V eCo se vklapljajo XML DTD in specializirane standardne XML sheme, na primer Common Business Language, Catalog Information Specification, Channel Definition Format, Internet Content Exchange, Open Buying on the Internet, Open Financial Exchange, Open Trading Protocol, Simpl-EDI in drugi. [5, 10]

ebXML je zgrajen na podlagi ovojnic za sporočila, ki se jih lahko prenaša po različnih protokolih (HTTP, SMTP, FTP), pri čemer so natančne specifikacije podatkov shranjene v repozitorijih. Svoj lasten predlog protokola za komunikacijo je organizacija zavrgla in sprejela SOAP. Od podjetij ga podpira predvsem Sun, od organizacij pa OASIS in UN/CEFACT, ustanovljen je bil avgusta 2000, zasnovo infrastrukture pa so predstavili marca 2001. Projekt ebXML želi na temeljih XML standardizirati izmenjavo poslovnih

dokumentov. Specifikacija poslovnega dokumenta XML je sestavljena iz opisa vsebine dokumenta XML, iz opisa struktura dokumenta XML in iz opisa relacijskega modela podatkov v dokumentu XML. [9]

CBL je zbirka z vnaprej definiranimi podatkovnimi tipi za XML. Namenjena je izmenjavi poslovnih dokumentov, na primer opisov izdelkov, naročil, računov in dobavnih urnikov. Temelji torej predvsem na komponentah. Glavni pokrovitelj tega ogrodja je CommerceOne, ki je CBL nadgradil v ogrodje eCo. Internet Content Exchange (ICE) v organizaciji CNet in drugih določa protokole in sheme za dostavljanje informacij (na primer novic) uporabnikom. Iniciativo Open Buying on the Internet (OBI) so začele American Express in druge organizacije z namenom avtomatizacije nabav pisarniških in vzdrževalnih potrebščin za velika podjetja, kar je sicer trg podjetja Ariba. [23]

cXML je razvil konzorcij pod vodstvom podjetja Ariba. Temeljil je na enostavnih podatkovnih tipih, namenjenih izmenjavi informacij o ponudbi in transakcije, na primer naročila, spremembe naročil, potrditve, plačila, ipd. Definiral je tudi protokole, ki so lahko enosmerni ali tipa zahteva/odgovor. Microsoft in Ariba sta ga nameravala vgraditi v BizTalk vendar se zadeva ni obnesla in je navezava zamrla. [7]

XML/EDI se nanaša na prepisovanje obstoječih ANSI X12 RIP standardov v XML brez kakšnih večjih sprememb. Simpl-EDI je bil poenostavljen prepis specifikacij UN/CEFACT v XML. OAGI je vključen v BizTalk in ponuja zbirko približno 150 sporočil, ki opisujejo naročanje, nabavo, dobavo, logistiko, ipd. [11]

Dokaj specifični **UDDI** je namenjen iskanju storitev na spletu. Predvsem gre za to, da podjetja omogočijo dostop do informacij o njihovih izdelkih in storitvah, oziroma najdejo poslovne partnerje in stranke. [29] Programski vmesnik (API) omogoča iskanje določenih storitev, dobivanje informacij o njih, omogoča pa tudi dodajanje in odstranjevanje teh podatkov. Komunikacija temelji na protokolu SOAP, podatki in komunikacija pa so v XML. UDDI je razdeljen na tri vrste vsebin: bele, rumene in zelene strani. Bele strani vsebujejo imena podjetij ter njihove opise v smislu področja dela ter storitve, ki jih posamezno podjetje nudi oz. zagotavlja, in protokole, ki jih podjetje podpira. Rumene

strani opisujejo podjetja z njihovo geografsko lokacijo ter oznakami poslovnih funkcij, kot jih določajo vlade in mednarodni dogovori. Zelene strani vsebujejo specifične informacije o tipih dokumentov, ki jih podjetje lahko sprejme, o vstopnih točkah do podjetja ter o tehnologiji, ki jo podjetje uporablja in podpira.

Nekatere organizacije so se odrekle pripravi lastnih standardov. Na primer, konzorcij BIC nima namena določati standardov, temveč te le priporočati. Nastal pa je kot odziv podjetij s področja programske opreme (npr., Microsoft, SAP, CA). Na iniciativo RosettaNet, za katero so predvsem podjetja iz področja elektronike. Ustanovljen je bil konec novembra, 2000.

Precej starejših standardov za računalniško izmenjavo podatkov (RIP) (EDI - *Electronic Data Interchange*) že obstaja, na primer ANSI ASC X12 je standardiziral 300 tipov poslovnih dokumentov. Podobno ima tudi UN/EDIFACT javno dostopno zbirko 200 standardnih sporočil (United Nations Standard Message - UNSM), kar je tudi ISO standard 9735, sprejet leta 1987. [11] V delovnem telesu UN/ECE WP.4, ki pripravlja UN/EDIFACT, je prisotnih preko 60 držav ter različne organizacije: EC, IATA, International Chamber of Commerce, ISO, International Chamber of Shipping, EAN, itn. V Sloveniji imamo interesno skupino GS1 (EAN) v okviru GZS. GS1 prevzema vodilno vlogo pri ustanavljanju globalnega multiindustrijskega sistema za identifikacijo in komunikacijo proizvodov, storitev in lokacij, ki temelji na mednarodno sprejetih in v poslovnem svetu vodilnih standardih. Sistem GS1 je zbir standardov, ki omogočajo učinkovito upravljanje preskrbovalne verige z edinstvenim označevanjem proizvodov, transportnih enot, lokacij in storitev. Sistem GS1 pospešuje procese elektronske trgovine, sledenja in izsledovanja. EAN pa predlaga način implementacije in razlago standarda UN/EDIFACT, kasneje pa je tudi predlagal svoje lastne XML standarde, na primer za podporo poslovnih procesov naročanja, razpošiljanja ter fakturiranja. [8]

Standardna sporočila EDIFACT se lahko deli v sporočila z osnovnimi podatki (mere proizvodov, imena), sporočila s podatki o partnerju, sporočila z informacijami o proizvodih, sporočila poslovnih transakcij ponudbe (rok dobave, plačilni pogoji, cene, popusti, takse), naročila (količine, datumi, kraj dostave), logistika in transport, računi in

nakazila), planiranje (predvidevanja o dostavi, prodaji in zalogah) ter poročila o zgradbi in načinu obdelave sporočila (potrditve prejema, ipd.).

Trenutno poteka več iniciativ, da bi svoje standarde prilagodili XML. Novejši standardi v večji meri temeljijo na XML. Evropska skupnost si prizadeva podpreti odprte standarde v čim večji meri in tako zagotoviti interoperabilnost in čim večjo povezljivost med posameznimi podjetji. Prav tako se poizkušajo oblikovati standardni dokumenti za izmenjavo med svetovnimi podjetji in ponudniki storitev. Cilj je doseči višji nivo orkestracije spletnih storitev in medsebojno avtomatsko sodelovanje. Osnovo pa predstavljajo prav infrastrukturni gradniki kot so podatkovne baze.

2.5 Komercialne rešitve elektronske izmenjave dokumentov

Po poročilu Jupiter MMXI Europe je v Evropi okrog 500 B2B portalov, od katerih naj bi jih preživelo le 100. [12] Trenutno se preko njih trguje z 185 milijoni EUR, kar pa je do leta 2006 naraslo že na 3.7 milijarde EUR. Uspešni trgi imajo podporo ostalih podjetij, veliko prometa in podporo uporabnikom. Najuspešnejših 10 podjetij na tem področju v Evropi, po mnenju Jupiter MMXI, so: angleški Acequote (pisarniške potrebščine), Buildonline (gradnja), Band-X (trg komunikacijskih kapacitet), nizozemski EumediX (medicinske potrebščine), švedski Phonetrade (telefoni) ter Eu-supply.com (gradnja), nemški Goodex (industrijska oprema), Mondus (poslovne potrebščine), irski IngredientsNet.com (prehrambene surovine) ter belgijski PEFA.com (ribe). Obstajajo tudi splošni trgi, na primer cc-markets, ki so ga ustanovili BASF, Degussa-Huels, Henkel in SAP, in Integrated Business Exchange (IBE), ki je namenjen vsem podjetjem z območja Skandinavije in Finske. [18, 19]

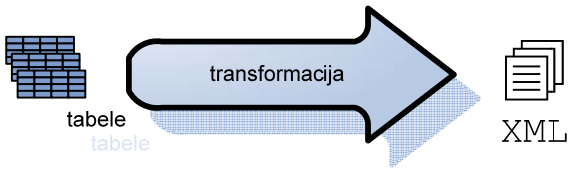
Orodja, ki omogočajo izgradnjo elektronskih trgov, so MarketSet, ki je nastal v sodelovanju CommerceOne in SAP, ki omogočajo ustanovitev splošnih trgov, torej takih, kjer nobeno podjetje nima izrazito dominantne vloge. Drugi elektronski trgi iz CommerceOne so namenjeni predvsem velikim podjetjem in vsebujejo več odprtih storitev, na primer: dražbe, primerjalno iskanje po katalogih, nabavo, upravljanje z dokumentacijo, analitika dobaviteljev, načrtovanje proizvodnje (potrebno za SCM). Posamezni elektronski trgi iz CommerceOne se med seboj povezujejo v Global Trading Web, vsi pa temeljijo na XML. [12] Je tudi precej konkurence: IBM ponuja vzorčno

realizacijo e-Marketplace v okviru WebSphere Commerce Suite. Oracle ima e-Business Suite s funkcionalnostjo elektronskih trgov. HP je ponujal e-Speak, ki bi omogočal pripravo odprtih storitev (za poljubne transakcije) in je bil združljiv z BizTalk, CommerceNet in RosettaNet, povrh tega pa je na voljo skupaj z izvorno kodo. Žal pa zadeva ni bila dovolj dobro zastavljena in ni zaživela v željenem obsegu.[20, 21]

2.6 Upravljanje s podatki

Podatki v XML dokumentih morajo biti za kasnejšo uporabo in lažjo manipulacijo znotraj podatkovne baze organizirani tako, da je možna pretvorba kot v tabele. V praksi sta najpogosteje uporabljena dva pristopa. Pri prvem pristopu, ki se najpogosteje uporablja pri relacijskih podatkovnih bazah, modeliramo XML dokument kot množico tabel z vrsticami in stolpci.

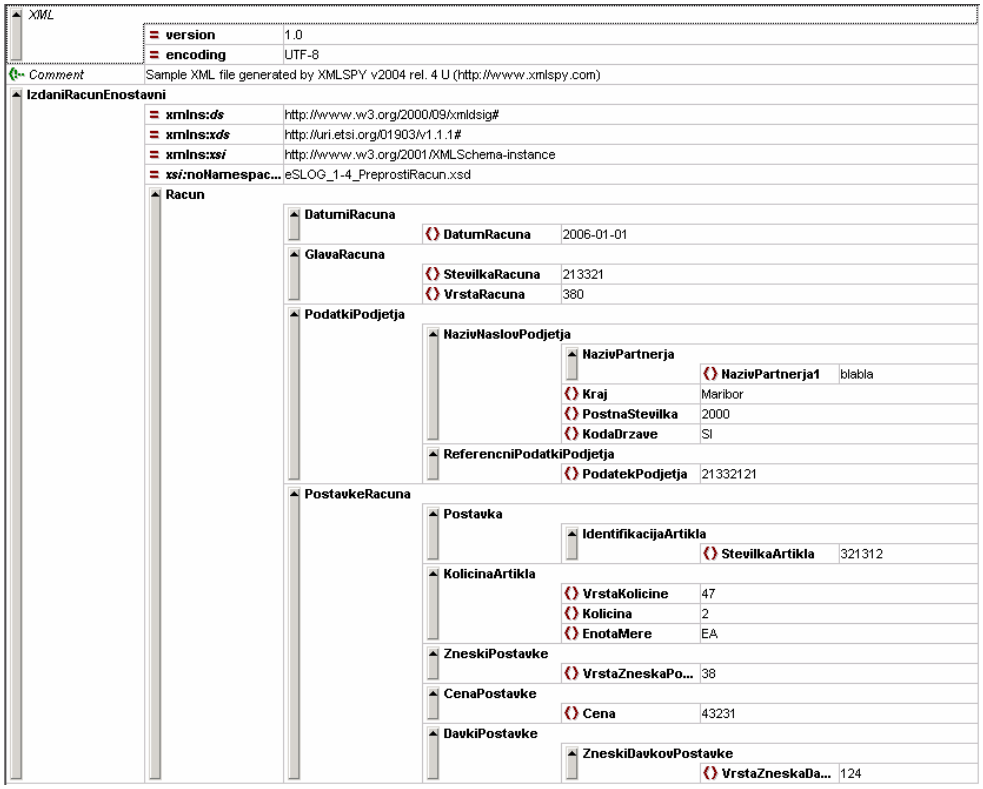
Model preslikave relacijskih tabel v dokument prikazuje Slika 5.



Slika 5: Model preslikave relacijskih tabel v XML

Osnovna ideja drugega pristopa je pretvorba podatkov v drevo objektov. Koncept, pri katerem v XML dokumentih uporabljamo drevesa, je še posebej primeren za objektne podatkovne baze. Uporaben pa je tudi pri relacijskih podatkovnih bazah in to z uporabo objektno relacijskih razširitev oz. objektnih lastnosti, ki jih prinaša standard SQL3.

Primer drevesne sheme je predstavljen na Sliki 6. Tako že sama drevesna struktura nakazuje na dokumentno usmerjenost podatkov.



Slika 6: Drevesna struktura XML dokumenta

Ko pretvarjamo podatke iz podatkovne baze v XML dokumente in obratno, naletimo na določene težave, saj pretvorba ni zmeraj enostavna. Pri tem lahko težave predstavljajo predvsem:

- *podatkovni tipi*

  V XML dokumentih so trenutno vsi podatki predstavljeni kot niz znakov, klasičnega tipiziranja podatkov kot v programskih jezikih ni. To lahko povzroči določene težave pri pretvorbi besedila v določen tip, ker je treba vnaprej poznati vse možne formate, v katerih se lahko pojavljajo določeni tipi. Še posebej to velja za datumske in številske tipe, kjer se format izpisa v besedilu lahko razlikuje glede na države. Znano je, da Američani namesto vejic uporabljajo pike (npr. 3.14 namesto 3,14).

- *binarni podatki*

  Če želimo v XML dokumente vključevati tudi binarne podatke, jih moramo na ustrezen način kodirati (npr. Base64). Ker standardnega načina, kako nakazati, da element vsebuje kodiran objekt, še ni, orodja težko ugotovijo, da v XML dokumentu obstaja kodiran podatek.

- *podatki brez vrednosti (null)*

  Če ima nek atribut v podatkovni bazi vrednost "null", to pomeni, da atribut nima vrednosti. Velikokrat ima lahko vrednost "null" popolnoma drugačen pomen kot prazno besedilo ali 0 pri številskih vrednostih. Zato moramo pri pretvorbi v XML dokumente in iz njih paziti in posebej obravnavati primere, ko se pojavi vrednost "null". Najpogosteje se odločimo, da v takšnem primeru atributa v dokumentu sploh ne navedemo, to pa pomeni, da bo po pretvorbi v podatkovni model vrednost tega atributa "null".
- *znakovni nabor*

  V skladu s standardom se lahko v XML dokumentih pojavljajo kakršnikoli znaki, ki so del standarda Unicode. Zaradi še nezadostne podpore tega v nekaterih zbirkah podatkov in aplikacijah, lahko naletimo na določene težave, če uporabljamo znake, ki ne sodijo v nabor ASCII.

2.7 Pomen shranjevanja XML dokumentov v storitvenih arhitekturah

Shranjevanje XML dokumentov v podatkovno bazo je prav tako zelo pomembno iz vidika storitvene arhitekture in njene povezave z razmahom XML jezika ter spletnih storitev. Medtem, ko se je nepričakovano povečevalo zanimanje in popularnost spletnih storitev so pričeli glavni razvijalci programske opreme agresivno razvijati vrsto razširitev za prvo generacijo WS-* standardov. Te specifikacije so naslavljale specifična področja funkcionalnosti s skupnim ciljem povzdigniti tehnologijo spletnih storitev na nivo »enterprise« platforme. Tako se sedaj izmenjujejo predvsem XML dokumenti in sporočila namesto klasičnih načinov aplikacijske komunikacije ter se tudi s tega vidika povečuje pomen podpore XML jeziku v podatkovni bazi.

Ni trajalo dolgo preden so organizacije pričele kmalu le spoznavati, da spletne storitve predstavljajo osnovo popolnoma nove in drugačne platforme. Ta bi lahko pokrila delitev celotnega poslovnega spektra s serijo avtonomnih storitev. Tako je nastala storitvena arhitektura, ki je pomladila celotno tehnologijo storitev in jo preoblikovala v obliko za katero je bila mišljena. Storitvena arhitektura je tako nastala kot premišljena in načrtovana platforma ter je plod sodelovanja standarizacijskih organizacij in večjih proizvajalcev programske opreme.

2.8 Motivacija za uporabo jezika XML v podatkovnih bazah

Dandanes se vedno več podatkov oblikuje, obdeluje in prenaša v obliki XML. XML se je pokazal kot učinkovit način zapisa podatkov in njihovega prenašanja preko omrežja. Kljub prenekaterim pomanjkljivostim tehnologije so v ospredju predvsem njene prednosti, ki jo postavljajo v sam vrh uporabnosti. Poleg preglednosti nad podatki, gre predvsem za enostavne obdelave podatkov, njihovo zajemanje iz sporočil in zaznavanje napak.

Razlog zakaj shranjujemo XML dokumente v podatkovnih bazah je v težki obvladljivosti velike količine dokumentov, ki se danes nabirajo na datotečnih sistemih. Bilo bi zelo težavno učinkovito dostopati in manipulirati s takšnimi zbirkami podatkov. Tudi rešitev pretvorbe v objektno-relacijsko obliko in shranjevanje v relacijske podatkovne baze ni idealna. Tukaj se izkaže kot glavna pomanjkljivost predvsem večkratna transformacija podatkov – najprej v objektno in nato še v relacijsko obliko podatkovnih baz. Potrebno je prav tako zagotoviti enostavne in učinkovite operacije iskanja, spreminjanja in izmenjave dokumentov.

Tehnologije XML so kot trend današnjega časa tako prodrle tudi v podatkovne baze – ki so jedro shranjevanja podatkov in njihove obdelave. Najprej se je pričelo s shranjevanjem kot tekstovno polje. Vendar se je to dokaj hitro izkazalo za zelo neučinkovito in potratno početje. Tako je bilo potrebno vpeljati naprednejše mehanizme shranjevanja in poizvedbe po podatkih v obliki XML že v podatkovni bazi. Potrebno je bilo podpreti tako XPath in XQuery kot optimizacijo in mehanizme dostopa do podatkov v obliki XML. Le-ta omogoča mnogo boljši nadzor in kompleksnejšo obdelavo podatkov kot to omogočajo preprosta tekstovna polja. Že sama tehnologija XML ima vsebovanih veliko odvečnih podatkov, tako da bi neučinkovit način shranjevanja in manipulacije le-teh še dodatno onemogočil učinkovito in konkurenčno uporabe te tehnologije v aplikacijah. Različni pristopi shranjevanja XML dokumentov v podatkovni bazi bodo opisani v naslednjem poglavju.

3 PODPORA XML TEHNOLOGIJAM V PODATKOVNIH BAZAH

3.1 Zgodovinski razvoj

Sama zgodovina podpore XML tehnologijam v podatkovni bazi se je pojavila precej časa za pojavom XML tehnologije. Razvoj se je pričel nekje ob pojavu spletnih storitev leta 2000, ko je pod okriljem W3C nastal standard SOAP (Simple Object Access Protocol). Ta se je pojavil kot unifikacija RPC (Remote Procedure Call) komunikacije in kasneje ponekod celo kot njene zamenjave. Podpora XML-a v podatkovnih bazah se je tako pojavila pred komaj petimi leti (januarja 2001 je prvič nastal predlog, in komaj leta 2002 se je pričel razvoj tehnologije) in je kljub nepričakovani rasti uporabe v celotnem spektru IT (informacijskih tehnologij) še vedno v polnem razmahu nastajanja. Že od samega začetka se je porajalo vprašanje, ali je jezik XML dovolj izrazno močen in uspešen za zamenjavo obstoječih, že preizkušenih ter uveljavljenih podatkovnih struktur in načina izmenjave podatkov. Pravzaprav je bilo bolj vprašanje ali lahko izpolni praznino in pomanjkljivost obstoječih tehnologij ter se obenem uspešno postavi ob bok obstoječim sistemom z obstoječimi aplikativnimi rešitvami in trenutnimi performančnimi zahtevami uporabnikov.

Sam XML je podobno kot Java mogoče prišel na trg pred svojim časom in je šele kasneje dozorel. Ob prihodu je bil dokaj okoren, z veliko »overhead-a« in zelo počasen. Dandanes ga rešujejo dobri programski vmesniki in ogrodja, vse bolj zmogljivi računalniški sistemi in racionalizacija uporabe. Prav tako je prinesel drugačno razmišljanje in strategijo uporabe. Le-te se je bilo treba navaditi in pravilno uporabiti – v začetnih projektih je bil XML prepogosto »zlorabljen« in napačno uporabljen. [19]

Tako se je ob razvoju in dozorevanju jezika XML pojavila iniciativa po uporabi te tehnologije tudi v podatkovnih bazah. Razvila sta se dve glavni smeri razvoja; podpora jezika XML direktno v že obstoječih relacijskih in objektno-relacijskih podatkovnih bazah ter popolnoma nov razvoj podatkovne baze na temelju tehnologije XML. Vsaka smer ima svoje prednosti in slabosti. V diplomskem delu bomo podrobneje raziskali prvi primer. Slednji žanje veliko simpatij v krogih, kjer se uporabljajo spletne storitve in elektronsko izmenjevanje podatkov (odprti standardi) in transakcij – elektronsko poslovanje. Le-to se ob podpori XML v podatkovni bazi zelo dobro informacijsko podpre tudi z vidika zagotavljanja trajnosti dokumentov. Prav tako veliko bremena iz programskega dela preide na podatkovno bazo, ker je večina zadev že avtomatiziranih in »pravilno« implementiranih

(optimizirano in v skladu s smernicami dobrega načrtovanja). V primeru XML podatkov in sheme lahko izkoristimo strukturiranost podatkov in lažje vršimo samo manipulacijo in iskanje po podatkih. V primeru, da ne uporabimo sheme, ki je vezana na XMLType, naletimo na podobne »težave« kot pri shranjevanju v tekstovno polje.

Ali gredo XML podatkovne baze po poti objektno orientiranih podatkovnih baz bo pokazal čas. Te so po začetnem navdušenju in obljubah doživele počasen, a zanesljiv propad. Zaenkrat se je XML pričel dobro uveljavljati in pojavlja se vse več programskih vmesnikov za različne programske jezike. Prav tako se izkaže kot odlična podlaga za mnoge preproste zbirke podatkov in konfiguracije aplikacij – vendar ne kot na začetku promoviran človeku prijazen, pač pa stroju »prijazni« formi.

3.2 Vrste XML podatkovnih baz

Kljub novim možnostim shranjevanja XML podatkov v podatkovni bazi, še vedno obstaja primarni način shranjevanja, ki je na voljo od vsega začetka – tekstovno polje. Ključni problem takšnega načina je gotovo sama obdelava podatkov in njihova manipulacija, ki je procesorsko in pomnilniško zelo zahtevna.

Kot alternativa so se pojavile t.i. »native« XML podatkovne baze, ki definirajo (logični) model XML dokumenta – kot podpora za podatke, ki se nahajajo v njem ter njihovo shranjevanje in povpraševanje glede na model. Model mora vsebovati vsaj elemente, atribute, PCDATA in zaporedje dokumenta. Primeri takšnih modelov so recimo XPath, XML Infoset, ter modeli, izpeljani iz DOM in SAX dogodkov [14]. Nekaj primerov »native« podatkovnih baz so: Berkely XML DB, eXist, dbXML in ostale. Kot »native« pomeni, da lahko omogočajo shranjevanje relacijskih podatkov, mnoge pa prav tako same zase ne morejo obstajati.

Obstajata dve vrsti XML podatkovnih baz:

- XML-omogočene – te preslikajo vse XML podatke v relacijske podatke, ter sprejemajo XML kot vhod in ga dajejo na izhod – to implicira, da baza dela pretvorbo sama (glede na spodaj ležeč middleware).
- »native« XML (NXD) – notranji model je odvisen od XML-a in uporablja XML dokumente kot osnovno enoto shranjevanja.

XML dokument je definiran kot osnovna logična enota, kot je pri relacijskih bazah osnovna enota vrstica v tabeli. Ni pa nujno, da obstaja kak specifičen spodaj ležeč fizični

podatkovni model. Na primer, lahko je enostavno nadgradnja relacijske, hierarhične ali objektne podatkovne baze ali pa uporablja kakšnega izmed osnovnih primitivnih formatov shranjevanja kot so recimo indeksirane in kompresirane datoteke. Baza je specializirana za shranjevanje XML podatkov in shranjuje vse komponente XML-la nespremenjeno – dokumenti nastopajo kot vhodni in izhodni tok. Niti ni nujno, da obstaja v obliki samostojne podatkovne baze. Poizvedbe nad takšno podatkovno bazo so razširjene s funkcionalnostmi za delo z XML jezikom in njegovo obdelavo.

3.3 Različni načini shranjevanja XML dokumentov

Uporaba standarda SQL3 omogoča shranjevanje velikih objektov, v našem primeru celotnih XML dokumentov; tako lahko v podatkovnih bazah poleg podatkov shranjujemo tudi celotne dokumente. Dokumenti so shranjeni kot vrednosti atributa vrste BLOB ("binary large object") ali CLOB ("character large object") oz. kot »native« objekti (XMLType), kadar uporabimo objektne razširitve podatkovnih baz. [16] XML tip se tako pokaže kot zanimivo področje in nov način shranjevanja in manipulacije podatkov v podatkovnih bazah. Potrebno se je prilagoditi in pravilno uporabiti dano tehnologijo in tako izkoristiti kar se da največ iz nje – kljub temu, da ima prav tako svoje pomanjkljivosti.

Omeniti velja, da večina vodilnih podatkovnih baz ponuja številne nove možnosti v zvezi s povpraševanjem, kot npr. iskanje po polnem besedilu (full-text search), iskanje podobnih besed, sinonimov itd. To lahko s pridom izkoriščamo tudi pri shranjevanju XML dokumentov. Obstajajo pa tudi povpraševalni jeziki za iskanje in manipulacijo XML dokumentov, kot recimo XQuery in XPath s pomočjo katerih pridobimo širok spekter dodatnih funkcionalnosti. XML dokumente lahko v relacijskih podatkovnih bazah shranimo na tri osnovne načine:

1. **Pretvorba XML v objektno-relacijski podatkovni model**

Elementi XML dokumenta se preslikajo v tabele, vrstice in stolpce. Povpraševanje po podatkih je v tem primeru najhitrejše, se pa porabi nekaj več časa za shranjevanje in pripravo.

2. **Shranjevanje XML dokumentov v stolpce**

   Vsak XML dokument se shrani v en stolpec. Priprava in shranjevanje nista zahtevni opravili, zato pa povpraševanje prek celotnih XML dokumentov zahteva več časa.

3. **Mešan (hibridni) način**

   Mešan način poskuša uveljaviti dobre lastnosti obeh prejšnjih načinov. Pri tem načinu se vsak XML dokument shrani v en stolpec, določeni atributi iz tega istega dokumenta pa v različne stolpce. S tem pride v podatkovni bazi do določene redundance, vendar lahko dodatne stolpce učinkovito uporabimo pri povpraševanjih.

3.4 Podpora XML v komercialnih podatkovnih bazah

Vsi večji proizvajalci so že napovedali in tudi implementirali podporo pretvarjanju podatkov v XML in obratno. XML je splošno podprt tip in vedno bolj razširjen ter ga tako lahko najdemo v večini uveljavljenih podatkovnih baz. Opaziti je predvsem razliko podpore tipu XML kot takem, saj je večinoma podprt kot tekstovno polje in mogoče samo označen kot XML za možnost nadaljnje obdelave in razširitve.

V podatkovni bazi **Oracle** je od verzije 8i vključen XML Developer's Kit (XDK), ki vsebuje komponente za branje, manipulacijo, pretvarjanje in vpogled v XML dokumente. Oracle Intermedia XML Search predstavlja zmogljivo orodje za iskanje po dokumentih, saj med drugim omogoča tudi iskanje po hierarhiji in posameznih XML atributih, išče besede, ki podobno zvenijo, besede, ki so si podobne itd. [26] Nekaj primerov podpore je prikazano v Tabeli 1.

Tabela 1: Primerjava podpore XML tipa v podatkovnih bazah

podatkovna baza/podpora	podpora tipu XML	manipulacija podatkov znotraj PB	podprti operacijski sistemi	podprti programski vmesniki
Oracle	od 9i (delno kot razširitev v 8i), popolna 10g R2	da	Linux, Windows, HP-UNIX, SUN OS, MacOS, Java zmožen	SQL, PL/SQL, Java, C, C++
MSSQL	od 2005	da	MS Windows	C#, C++, C, Java*
DB2	od verzije 9	ne	Windows, Linux/Unix	C#, C/C++, Java
Berkley DB	2.0.x, popolna od 2.2.x	da	Windows, Linux, BSD UNIX, Mac OS/X in katerikoli POSIX kompatabilni	C++, Java, Perl, Python, PHP, Tcl, Ruby
Informix	pričetek z IDX 8.x	ne	Linux, Windows	Java, COBOL, C++, PHP, ODBC
Sybase SQL	ASE 12.x	da	Linux, Windows	Java, ODBC, C++

Podpora XML podatkovnemu tipu je v podatkovni bazi Oracle zelo temeljita, funkcionalna in uporabna. Naletimo lahko le na manjše težave glede zahtevnosti določenih operacij nad XML podatki. [25] Žal pa zaenkrat ni zaslediti obstoječega standarda oz. specifikacije kriterijev XML podatkovne baze. Vse temelji na lastni promociji posameznega proizvajalca glede podprte funkcionalnosti za delo z XML dokumenti.

Microsoft že od začetka podpira shranjevanje XML dokumentov v podatkovni bazi MS SQL in je naredil v zadnjem obdobju velik korak naprej na področju shranjevanja in obdelave XML dokumentov v podatkovni bazi. XML se je kot vodilna tehnologija v komercialnem svetu dobro uveljavil in tako zagotovil vso podporo s strani Microsofta. V strežniku **MS SQL** je od različice 2005 zagotovljena zelo dobra podpora za obdelavo dokumentov tipa XML. Prednostno je ob razvoju in uporabi Microsoft-ovih razvojnih orodij uporabiti njihovo podatkovno bazo – zelo velika mera integracije ter specializiranih komponent za ta tip podatkovne baze. Prav tako je visok nivo podpore XML podatkovnemu tipu in mešanju ter enostavni pretvorbi – (de)serializaciji med XML in objekti.

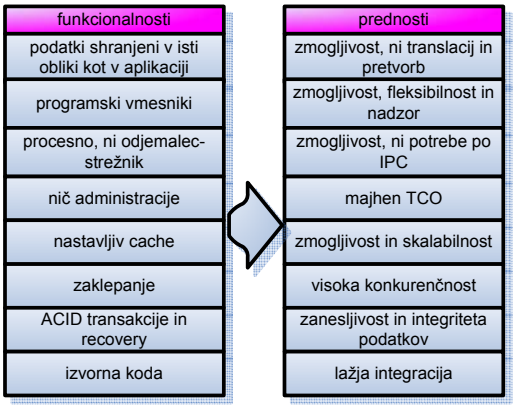
V podatkovni bazi **DB2** je podpora XML še dokaj šibka in na nižjem nivoju, ter uporabniku(razvijalcu) manj blizu in enostavno za uporabo. DB2 XML Extender je

razširitev podatkovne baze DB2, ki jo uporabljamo za pretvorbo relacijskega modela v XML ter nazaj. DB2 Text Extender omogoča zmogljivejše iskanje po XML dokumentih, saj med drugim išče sinonime, besede, ki so si podobne, podpira iskanje po atributih XML itd.

Berkley DB je ravno med pisanjem diplome kupilo podjetje Oracle kot nov produkt v liniji visoko zmogljivih vgrajenih in odprtokodnih podatkovnih baz. Le-ta že vsebuje podporo XML podatkovnega tipa in je ob združitvi z gigantom Oracle tako postal del skupne vizije in napredka družbe.

V podatkovni bazi **Informix** (Informix) modul Informix Web DataBlade podpira XML, produkt Hierarchical XML Data Storage pa omogoča uporabnikom, da uvažajo, izvažajo in shranjujejo dokumente ter iščejo po XML dokumentih v njihovem naravnem formatu.

Sybase SQL Server (Sybase) se s svojim produktom Adaptive Server 12.0 postavlja ob bok drugim podatkovnim bazam, ki omogočajo shranjevanje, pripravo in integracijo podatkov XML. Večina orodij, tudi XML tolmači (parsers), so napisani v javi.



Slika 7: Zahtevane in priporočene funkcionalnosti za NXD [4]

V XML podatkovni bazi si želimo vsaj takšne ali celo boljše funkcionalnosti kot so na volj v objektno-relacijskih podatkovnih bazah (Slika 7). Iz njih pa izhajajo prednosti, ki jih le-te prinašajo, kot so recimo visoka zmogljivost in skalabilnost, majhni stroški vzdrževanja in lažja integracija.

4 PODATKOVNA BAZA ORACLE IN PODPORA XML TEHNOLOGIJAM

Oracle 10g XML DB

Različica 10g je kombinacija relacijske in XML podatkovne baze, saj združuje tako klasične metode dostopa in manipulacije podatkov kot nov podatkovni model XML (XMLType) ter ustrezne programske vmesnike za dostop in različne operacije nad podatki XML. [23, 24]

Podprte so naslednje operacije nad podatki XML:

- shranjevanje,
- povpraševanje,
- ažuriranje,
- transformiranje in
- obdelava podatkov XML.

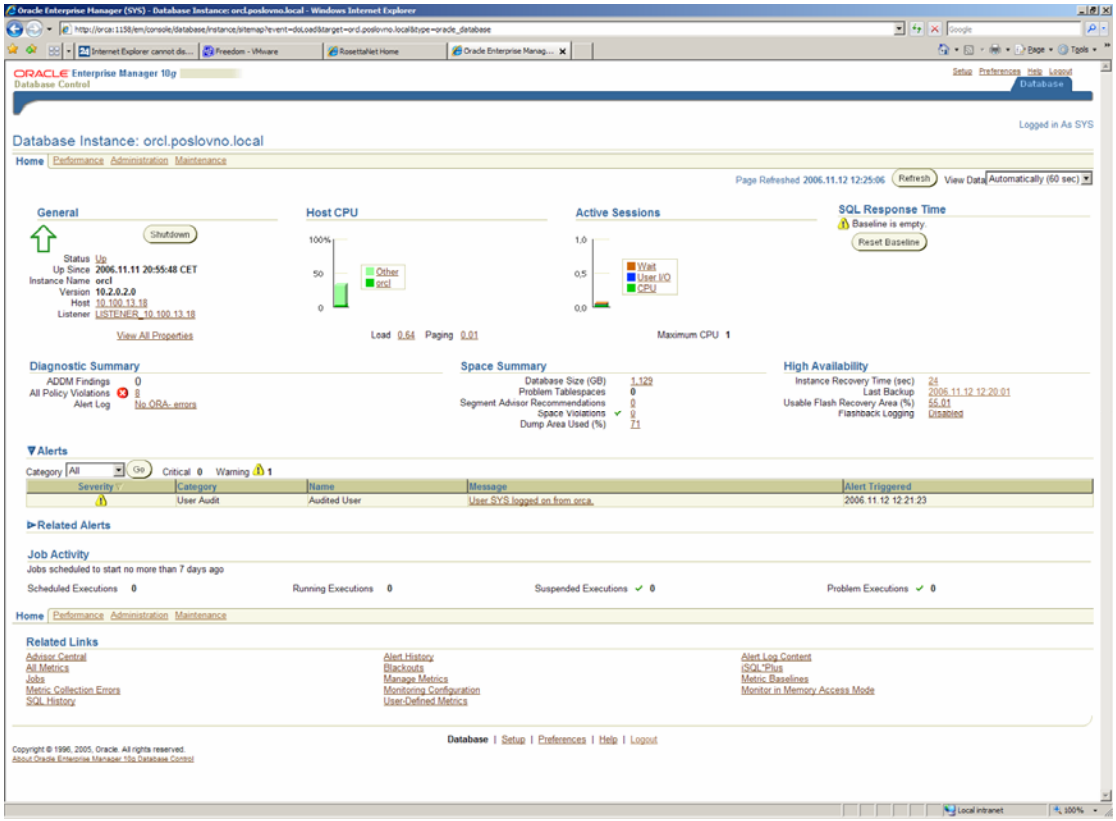
XML repozitorij je dostopen preko standardnih internetnih protokolov (datotečni sistem, FTP, HTTP(s), WebDAV). Oracle DB 10g R2 infrastruktura je prav tako neodvisna od načina shranjevanja, vsebine in uporabljenih programskih jezikov.

Novosti podpore za XML v različici 10g R2

Oracle je že od različice 9i (8i delno v okviru razširitve XDK) v svoji podatkovni bazi podpiral tehnologije XML. [13] Vendar zaradi začetka razmaha in nadaljnje negotovosti razširjenosti in uporabnosti tehnologije še ni bila zagotovljena popolna podpora vseh funkcionalnosti. Prav tako je implementacija le-te pomenila novo prelomnico v nadaljnjem razvoju same baze ter potrebne dodatne implementacije ter podporo uporabnikov. [18] V različici podatkovne baze Oracle 10g R2 so bile dodane oz. izpopolnjenje sledeče funkcionalnosti in tehnologije, ki podprejo samo povpraševanje in jih nadziramo tudi preko nove spletne konzole (Slika 8):

- XQuery;
- nove SQL funkcije za delo s podatki XML;
- *UpdateXML()* omogoča delo z vsebino nad obstoječimi vozlišči v dokumentu – ne omogoča dodajanja, ali odstranjevanja vozlišč;

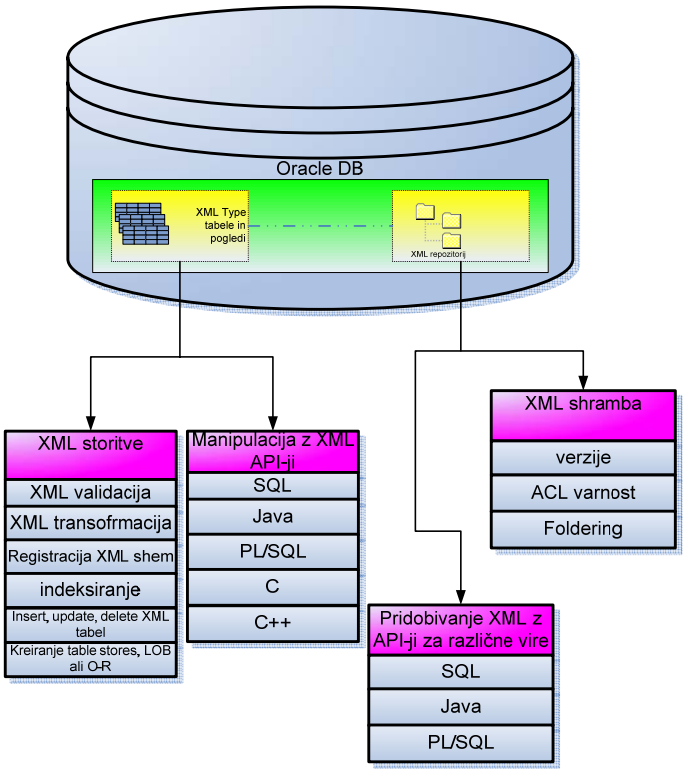
- *InsertXML()*, *AppendChildXML()*, *InsertXMLBefore()* in *DeleteXML()*
- podpora SQL/XML standardu 2005;
- dodane funkcije: XMLPI, XMLComment, XMLRoot, XMLSerialize, XMLCDATA, XMLPARSE;
- na XML shemi temelječi metapodatki;
- dodajanje/urejanje metapodatkov (virov v Oracle XML DB repozitoriju), ki temeljijo na XML shemi;
- izboljšano povpraševanje XPath;
- zamenjava Oracle XDK PL/SQL paketov;
- XMLDOM, XMLPARSER in XSL_PROCESSOR;
- DBMS_XMLDOM, DBMS_XMLPARSER in DBMX_XSLPROCESSOR;
- podpora XSLT 2.0 in XPath v Javi;
- dodatne funkcije upravljanja Oracle XML PB v konzoli Enterprise Manager;
- konfiguracijski parametri, viri v rezpozitoriju, seznami ACL, sheme XML, XMLType tabele in stolpci.



Slika 8: Spletna konzola Oracle 10g R2

4.1 Arhitektura podatkovne baze Oracle

Slika 9 prikazuje osnovno arhitekturo podatkovne baze Oracle. Jedro tvori stroj (engine) za procesiranje XML podatkov in dokumentov. Meta podatki se nahajajo v XML repozitoriju, kjer lahko najdemo vse vrste podatkov o podatkih, ki se nahajajo v podatkovni bazi ter njihovi opisi. Do repozitorija lahko dostopamo preko različnih protokolov. V povezavi z Oracle podatkovno bazo lahko prav tako izpostavimo XML storitve, manipuliramo z XML dokumenti in hranimo XML dokumente.



Slika 9: Oracle XML DB arhitektura

Arhitektura XML DB v podatkovni bazi Oracle je razdeljena na XML repozitorij, kjer se nahajajo vse entitete in objekti baze, in objekt XMLType, s katerim operiramo v repozitoriju. Repozitorij predstavlja shrambo XML dokumentov, kot je prikazano na Sliki 9. V tej je omogočeno verzioniranje, ACL(Access Control Lists) varnostni mehanizmi ter delitev in interna organiziranost objektov [18]. Dokumente in objekte pridobivamo iz repozitorija preko programskih vmesnikov za različne vire. Tip XMLType se uporablja za vse vrste storitev (XML validacija, transformacija, registracija shem, indeksiranje, itd.) do vračanja objektov in manipuliranje podatkov preko programskih vmesnikov. [18]

Repozitorij zagotavlja skupen prostor za odlaganje in manipulacijo objektov/dokumentov v Oracle XML DB shrambi. Hkrati pa predstavlja centralno točko dostopa in uporabe funkcionalnosti XML podatkovne baze. Preko administracijskega vmesnika ponuja tudi napredne funkcije administracije dostopa, manipulacije in vzdrževanja podatkov v njem. Podatke lahko pridobivamo v poljubni obliki in jih kot takšne tudi vstavljamo v repozitorij. Pravzaprav je to ključni element in skupek vseh novih funkcionalnosti in prednosti uporabe XML DB podatkovne baze.

4.2 XML DB lastnosti

XMLType je natančno določen s XML Schema dokumentom. [25] Na osnovi sheme je tako omogočeno takojšnje preverjanje pravilnosti dokumentov že ob zapisovanju XML dokumentov v podatkovno bazo. Omogočeno je tako strukturirano kot nestrukturirano shranjevanje objektov – dokumentov. V tem primeru na XML dokument ne vežemo sheme, pač pa ga shranimo kot CLOB polje v podatkovno bazo. Prav tako je možno hkrati uporabljati XML in SQL povpraševanja – to imenujemo **XML/SQL dvojnost ali dualnost**! Pri obeh tipih povpraševanj so na voljo standardni operatorji za SQL in XML povpraševanja. Na voljo je tako PL/SQL kot tudi XML jezik za izvedbo povpraševanj in tudi pridobivanje rezultatov. Popolnoma sta podprta tako XPath 1.0 in XQuery 1.0 za uporabo XML povpraševanj. Tako je možno izvesti povpraševanje nad XML podatki in dobiti nazaj vrstice tabele. Tudi obratno, mogoče je uporabiti SQL povpraševanje za pridobitev XML podatkov. Tako nismo več omejeni s samo obliko izhodnih podatkov povpraševanja po podatkovni bazi in se določene nadaljnje obdelave v informacijskem sistemu bistveno poenostavijo. Primer kombinacije SQL povpraševanja z XPath:

```
SELECT extractValue(OBJECT_VALUE,'predstavitev/naslov') "Naslov diplome"
2 FROM DIPLOMA_TABELA
3 WHERE existsNode(OBJECT_VALUE,'predstavitev/diploma[@id="28"]') = 1;
```

Prav tako je omogočena boljša podpora odprtim standardom in izpostavitvi spletnih storitev in podatkov v poljubni obliki (XSLT). [24]

XMLType se uporablja za hrambo (storage) tabel in pogledov. Vsi podatki se nahajajo v Oracle XML DB repozitorij-u, kjer jih lahko relativno enostavno pridobimo s pomočjo

tako SQL kot XML povpraševanja. Kljub uvedbi podatkovnega tipa XMLType je v ozadju shranjevanje XML dokumentov v tabele – kot object-relational oz. CLOB tip.

4.3 XML podatkovni tip XMLType

XML Type je vgrajen podatkovni tip (od 9.0.1), ki ga lahko uporabimo kot podatkovni tip stolpcev ali celotnih tabel. Podatke lahko shranjujemo nestrukturirano (CLOB), native XML obliki (object-relational) ali kombinirano. Prav tako je omogočena deklaracija PL/SQL spremenljivke ter kot tip v Javi (JDBC) in jeziku C/C++ (ODP.NET). Ob uporabi lahko definiramo in kličemo tako PL/SQL procedure in funkcije, kot tudi XPath in XQuery povpraševanja. Vse skupaj pa je dostopno preko globalnega repozitorija v podatkovni bazi. [28]

Podatkovni tip XMLType in ustrezni programski vmesniki pa omogočajo:

- povpraševanje in pridobivanje podatkov (XPath, XQuery),
- SQL operacije nad XML vsebino / XML operacije nad SQL vsebino,
- SQL povpraševanje po delu ali celotnem XML dokumentu (npr. funkciji *existsNode()*, *extract()* nudita SQL povpraševanje nad dokumenti XML),
- preverjanje skladnosti podatkov XML z določeno shemo XML in
- transformacije podatkov XML (XSLT).

4.3.1 Uporaba podatkovnega tipa XMLType

Tip **XMLType** lahko uporabimo v SQL stavkih kot kombinacijo z drugimi stolpci in podatkovnimi tipi. Tako lahko povprašujemo na primer po stolpcih XMLType in združimo rezultat z relacijskim stolpcem. Indeksiranje je izvedeno s pomočjo BTREE indeksov nad objektno-relacijskimi tabelami, ki so namenjene strukturiranemu shranjevanju XMLType tabel in stolpcev. Na voljo imamo indekse, ki temeljijo na funkcijah, in indekse, ki temeljijo na tekstu. [13]

Prav tako se lahko uporabi katerikoli drug poljuben podatkovni tip – definicija stolpca v tabeli, deklaracija PL/SQL spremenljivke, Java, C (JDBC, in ODP.NET) tip, definicija in klic PL/SQL procedure ali funkcije. Podatke v tabelah in stolpcih tipa XMLType lahko shranjujemo v:

- CLOB (primerno za pridobivanje celotnih dokumentov in dokumente, kjer ni potrebe po ažuriranju XML podatkov),
- "native" obliki XML (object-relational),
- kombinirano.

Za vse registrirane XML sheme se elementi XML za tabele, stolpce in poglede tipa XMLType direktno preslikajo v tabele v podatkovni bazi. Do le-teh dostopamo v bazi preko repozitorija XML. Podatki v pogledih XMLType so lahko shranjeni v lokalnih ali oddaljenih tabelah (DBLinks). Podatki XML, ki niso vezani na shemo, se shranijo kot CLOB stolpci.

Vnos podatkov v Oracle XML DB

XML podatkovni tip je kot že omenjeno posamezen atribut ali pa celotna tabela v podatkovni bazi. Vezan je na XML shemo, kjer se tudi izvaja validacija vnosov dokumentov in povpraševanj nad njimi. Vnos se vrši preko programskih vmesnikov SQL ali PL/SQL, Java, C, SQL*Loader. Dostop in vstavljanje pa je možno direktno preko repozitorija in ustreznih protokolov (WebDAV, FTP, HTTP). [18]

Primer uporabe XMLType (CLOB)

Tabela:

```
create table dokumenti of XMLType;
```

Stolpec:

```
create table dokumenti1 (
kljuc VARCHAR2(10) primary key,
dokument XMLType);
```

Primer uporabe ukaza SQL za vnos:

```
insert into dokument values (
XMLType('<?xml version="1.0" encoding="UTF-8"?>
<dokument>
<avtor>Marko Bevc</avtor>
<naslov>Podpora tehnologijam XML v Oracle-u</naslov>
<povzetek>Tehnologije, povezane s tehnologijo XML, v
vedno večjem obsegu zasledimo v sklopu sistemov
za upravljanje podatkovnih baz.
Podpora....</povzetek>
</dokument>')
);
```

Primer uporabe ukaza SQL ali PL/SQL za vnos XML dokumenta iz datoteke:

```
CREATE DIRECTORY xml_direktorij AS 'C:\TEMP';
GRANT READ,write ON DIRECTORY xml_direktorij TO MARKO;
insert into dokument values (
XMLType ( bfilename('XML_DIREKTORIJ', 'dokument.xml'),
nls_charset_id('AL32UTF8') )
);
```

Tabela 2: Primerjava strukturiranega in nestrukturiranega shranjevanja

nestrukturirano	**strukturirano**
Dobra fleksibilnost (zamenjava sheme)	Omejena fleksibilnost (ob zamenjavi sheme – podobno kot pri ALTER TABLE)
Ohranja originalno vsebino XML (pomembno za določene aplikacije)	Vsebina podatkov se zapiše iz DOM predstavitve – izguba podatkov, kot so npr. presledki
Dostop do nekaterih SQL funkcionalnosti in lastnosti	Večja možnost uporabe SQL funkcionalnosti in lastnosti
Zasede veliko prostora	Zasede manj prostora za shranjevanje (XMLSchema)

Strukturirano shranjevanje nudi določene prednosti pri upravljanju dokumentov XML, saj so za različne primere uporabe na voljo različni načini shranjevanja zbirk podatkov XML (Collections). Shranjevanje temelji na dekompoziciji vsebine dokumenta XML na množico objektov SQL, ki temeljijo na standardu SQL 1999! Za že registrirane sheme pa se SQL podatkovni tipi definirajo na podlagi XML sheme. Vsebina XMLType

tipa se shrani kot množica SQL tipov. Oracle XML DB avtomatično preslika 47 skalarnih tipov (ki jih podpira XML Shema) v 19 skalarnih podatkovnih tipov (Oracle). Pri preslikavi se opravijo pretvorbe v ekvivalentne tipe in poenostavitve nekaterih neobstoječih tipov. Posledično lahko naletimo na izgubo natančnosti ali pretvorbe med tipi. Prav tako lahko definiramo nove preslikave. Kljub vsemu pa zahteva več procesiranja med vstavljanjem/pridobivanjem podatkov (XML shema) kot klasične in alternativne metode shranjevanja. Upoštevati je potrebno dejstvo, da se vsak vstavljen dokument preslika v entitetno relacijske tabele in ob povpraševanju ne dobimo istih podatkov! Dobimo le iste regenerirane vrednosti, kar lahko povzroči določene težave in preglavice z raznimi posebnimi znaki (presledki). [18]

Nestrukturirano shranjevanje lahko drugače poimenujemo kot "dokumentno orientirani" podatki XML, saj je XML dokument shranjen kot celota v CLOB. Izvorni dokument se ohrani tak, kot smo ga vpisali v podatkovno bazo (presledki, komentarji, inštrukcije za procesiranje, itd.). Prav tako se kot dodatna težava pojavi pri povpraševanju po velikih dokumentih, saj to pomeni veliko obremenitev pomnilniških in procesorskih moči. [25] Nestrukturirano shranjevanje uporabimo, kadar je zahtevano shranjevanje/pridobivanje dokumentov v celoti in ni potrebe po spreminjanju dokumentov XML ali posameznih delov dokumenta XML.

4.4 Podpora XML shemam

Pred samo uporabo shem v podatkovni bazi Oracle je potrebno sheme registrirati z ukazom *registerSchema()*. Sheme lahko vežemo na podatke (stolpec, tabelo) in nato validiramo XML dokumente ob uporabi. Po shemah lahko tudi povprašujemo ter pregledujemo njihovo strukturo. V primeru, da shem ne potrebujemo več, jih lahko tudi brišemo – *deleteSchema()*.

Sheme lahko registriramo tako lokalno kot tudi globalno – odvisno od uporabe in želene dosegljivosti shem. Generiramo jih lahko iz že obstoječih objektnih tipov. Prav tako je omogočeno referenciranje na sheme, katerih lastniki so drugi uporabniki. Potrebno pa je eksplicitno referenciranje na globalno shemo, ko obstaja lokalna shema z istim imenom. Lahko pa uporabimo tudi označeno shemo ("Annotated" schema), katera ima dodatne

atribute (metapodatki) – Oracle atributi uporabljajo imensko področje http://xmlns.oracle.com/xdb. Le-ti so sledeči: *defaultTable*, *SQLName*, *SQLType*, *SQLCollType*, *MaintainOrder* in ostali. Omogočena je tudi uporaba zunanjih shem *DBLinks*, na katere se vežemo preko URI naslova.

DBMS_XMLSCHEMA – PL/SQL paket

Funkcija *registerSchema()*:

- razpozna in validira XML shemo,
- kreira množico definicij SQL objektov, ki temeljijo na complexType v XML shemi

Parametri:

- schemaURL – niz, ki unikatno določa shemo v Oracle XML DB
- schemaDoc – XML shema izvorni dokument
- LOCAL (true, false) – registracija lokalne ali globalne sheme
- GENTYPES (true, false) – generiranje objektnih tipov
- GENTABLES (true, false) – generiranje privzetih tabel
- force (true, false) – registrira shemo tudi ob morebitnih napakah
- owner – ime lastnika. Privzeto je ime uporabnika, ki izvaja funkcijo za registracijo sheme
- CSID – nabor znakov

Primer za registracijo sheme:

```
begin
dbms_xmlschema.registerSchema (
'http://www.feri.uni-mb.si/marko2006',
xdbURIType('/public/../Diploma.xsd').getClob(),
TRUE,TRUE,FALSE,TRUE
);
end;
/
```

```
SQL> SELECT table_name FROM user_xml_tables WHERE
2 XMLSCHEMA = 'http://www.feri.uni-mb.si/marko2006';

TABLE_NAME
```

```
------------------------------
diploma447_TAB
```

Generirana tabela:

```
TABLE Diploma447_TAB
OF XMLTYPE XMLSCHEMA
"http://www.feri.uni-mb.si/marko2006"
ELEMENT "diploma"
STORE AS OBJECT-RELATIONAL;

SQL> desc "diploma447_TAB";

Name Null? Type
---------------------------------------- -------- ----------------------
------
TABLE of SYS.XMLTYPE(XMLSchema "http://www.feri.uni-mb.si/marko2006"
Element "diploma")
STORAGE Object-Relational
SQL>
SQL> desc "IzdaniRacunEnostvni470_TAB";
TABLE of SYS.XMLTYPE(XMLSchema "http://www.gzs.si/e-
poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd" Element
"IzdaniRacunEnostavni") STORAGE Object-relational TYPE
"IzdaniRacunEnostavni325_T"
```

Generirani tipi:

TABLE_NAME	XMLSCHEMA	SCHEMA_OWNER	ELEMENT_NAME	STORAGE_TYPE
XML_TEST				CLOB
IzdaniRacunEnostavni470_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	IzdaniRacunEnostavni	OBJECT-RELATIONAL
Racun465_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	Racun	OBJECT-RELATIONAL
PovzetekZneskovRacuna463_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	PovzetekZneskovRacuna	OBJECT-RELATIONAL
SklicZaPlacilo462_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	SklicZaPlacilo	OBJECT-RELATIONAL
StevilkaSklica461_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	StevilkaSklica	CLOB
SklicPlacila460_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	SklicPlacila	OBJECT-RELATIONAL
ZneskiRacuna458_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	ZneskiRacuna	OBJECT-RELATIONAL
ZnesekRacuna457_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	ZnesekRacuna	CLOB
VrstaZneska456_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	VrstaZneska	OBJECT-RELATIONAL
PovzetekDavkovRacuna452_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	PovzetekDavkovRacuna	OBJECT-RELATIONAL
ZneskiDavkov450_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	ZneskiDavkov	OBJECT-RELATIONAL
ZnesekDavka449_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	ZnesekDavka	CLOB
VrstaZneskaDavka448_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	VrstaZneskaDavka	OBJECT-RELATIONAL
PostavkeRacuna444_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	PostavkeRacuna	OBJECT-RELATIONAL
OdstotkiPostavk442_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	OdstotkiPostavk	OBJECT-RELATIONAL
ZnesekOdstotka441_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	ZnesekOdstotka	CLOB
VrstaZneskaOdstotka440_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	VrstaZneskaOdstotka	OBJECT-RELATIONAL
OdstotekPostavke439_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	OdstotekPostavke	CLOB
VrstaOdstotkaPostavke438_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	VrstaOdstotkaPostavke	OBJECT-RELATIONAL
ReferencniDokumentiPo434_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	ReferencniDokumentiPostavke	OBJECT-RELATIONAL
StevilkaDokumentaPost433_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	StevilkaDokumentaPostavke	CLOB
VrstaDokumentaPostavk432_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	VrstaDokumentaPostavke	OBJECT-RELATIONAL
ZneskiPostavke428_TAB	http://www.gzs.si/e-poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd	MARKO	ZneskiPostavke	OBJECT-RELATIONAL

Slika 10: Primer generiranih tipov iz sheme

- TYPE IzdaniRacunEnostavni470_TAB AS(racun racun286_COLL);
- TYPE racun286_COLL AS VARRAY()OF RacunTip281_T
- TYPE RacunTip281_T AS (id_ VARCHAR2(4000),
  postavke postavke282_T,
  zneski ZneskiTip284_T,
  SklicZaPlacilo VARCHAR2(4000),
  PovzetekZneskov VARCHAR2(4000));
- TYPE podjetje282_T AS (podjetje podjetje283_COLL);
- TYPE podjetje283_COLL AS VARRAY() OFVARCHAR2(4000 CHAR)
- TYPE DatumTip284_T AS(ura VARCHAR2(40),
  dan VARCHAR2(40),
  mesec VARCHAR2(40),
  leto VARCHAR2(4));

```
SQL> desc "IzdaniRacunEnostavni470_TAB";
Name Null? Type
--------------------------------- -------- -------------------
TABLE of SYS.XMLTYPE(XMLSchema "http://www.gzs.si/e-
poslovanje/sheme/eSlog_1-4_PreprostiRacun.xsd" Element
"IzdaniRacunEnostavni") STORAGE Object-relational TYPE
"IzdaniRacunEnostavni325_T"
SQL> desc " GlavaRacuna327_COLL";
"GlavaRacuna327_COLL" VARRAY(2147483647) OF GlavaRacuna312_T
SQL> desc "PredavanjeTip281_T";
Name Null? Type
--------------------------------- -------- -------------------
SYS_XDBPD$ XDB.XDB$RAW_LIST_T
 Id VARCHAR2(4000 CHAR)
 GlavaRacuna  GlavaRacuna327_COLL
 DatumiRacuna DatumiRacuna328_COLL
 Lokacije  Lokacije334_COLL
 PoljubnoBesedilo  PoljubnoBesedilo338_COLL
 ReferencniDokumenti ReferencniDokumenti342_COLL
 PodatkiPodjetja PodatkiPodjetja381_COLL
 Valuta Valuta386_COLL
 PlacilniPogoji PlacilniPogoji405_COLL
 PostavkeRacuna PostavkeRacuna445_COLL
 PovzetekDavkovRacuna PovzetekDavkovRacuna453_COLL
 PovzetekZneskovRacuna PovzetekZneskovRacun464_COLL
SQL> desc "PodatkiPodjetja381_COLL";
Name Null? Type
--------------------------------- -------- -------------------
"PodatkiPodjetja381_COLL" VARRAY(2147483647) OF PodatkiPodjetja343_T
```

Doseg shem:

Lokalne sheme (prednost):

- dostopne samo lastniku

  /sys/schemas/**MARKO**/ www.feri.uni-mb.si /shema/shema_diplome.xsd

Globalne sheme(ob ustreznih pravicah uporabnika)

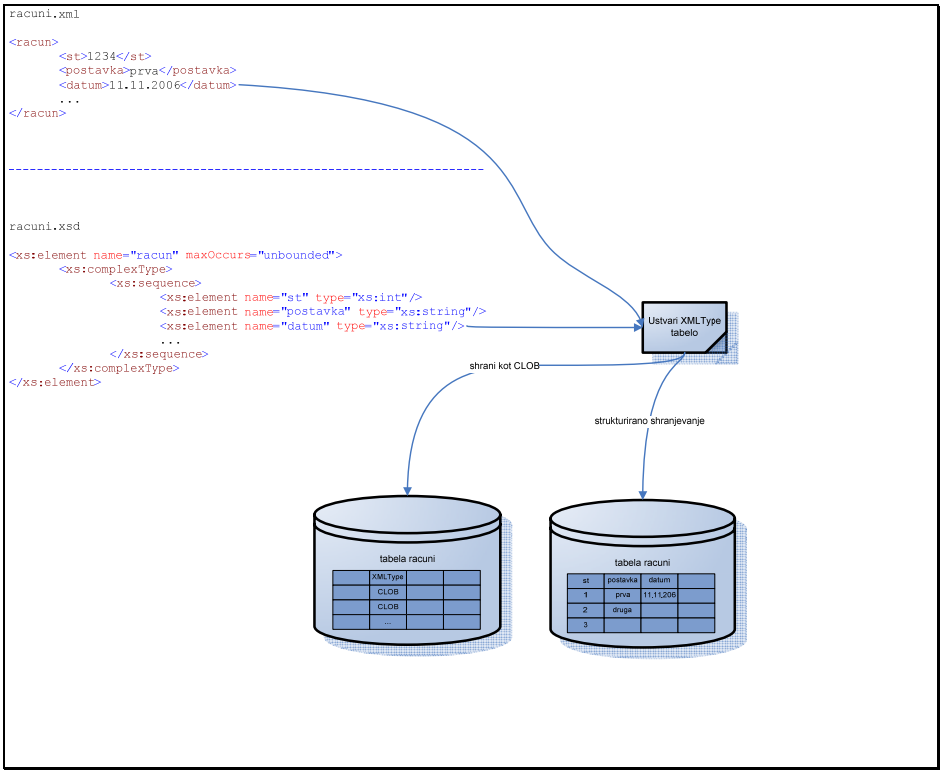
- vidne vsem uporabnikom PB

  /sys/schemas/**PUBLIC**/ www.feri.uni-mb.si /shema/shema_diplome.xsd

```
BEGIN
DBMS_XMLSCHEMA.registerSchema(
SCHEMAURL => '...',
SCHEMADOC => ...,
LOCAL => TRUE | FALSE,
GENTYPES => TRUE,
GENTABLES => FALSE,
CSID => ...);
END;
/
```

Preslikava tipov XML sheme v SQL tipe

Po registraciji sheme v podatkovni bazi se zgenerirajo pripadajoče tabele in polja. V vsakem primeru – z uporabo sheme ali brez, se v ozadju vse preslika v relacijske tabele. Osnovna razlika je v logični preslikavi strukture. Pri strukturiranem shranjevanju se izvede transformacija XML tipov v ustrezne SQL tipe in avtomatsko kreira logično identična struktura. V primeru nestrukturiranega shranjevanja (CLOB oblika) se kreira tabelo z poljem, ki vsebuje CLOB vsebino celotnega XML dokumenta in je prisoten kot polje vrstice. [20]



Slika 11: Preslikava XML dokumenta v tabele

Preslikava XML tipov sheme se vrši direktno v klasične in že poznane SQL tipe in podatkovne strukture. Privzeto se večina tipov pretvori oz. je v XML dokumentu predstavljenih s podatkovnim tipom **xs:*string***, kar je pri shranjevanju podatkov dokaj neugodno zaradi optimizacije, iskanja in manipulacije shranjenih podatkov. Pri pretvorbi je možno izbirati tip shranjevanja oz. pretvorbe podatkov. Ob shranjevanju v obliki CLOB – veliki objekti, prav tako naletimo na slabšo odzivnost ob samem povpraševanju po podatkih zaradi procesiranja podatkov in strukture v času povpraševanja.

Uporaba XML shem v XMLType

XMLType tabele in stolpce lahko vežemo na registrirane sheme, kjer se vrši avtomatska validacija podatkov že ob vstavljanju v podatkovno bazo in tako skrajšamo čas poizvedb po že vstavljenih podatkih.

```
CREATE TABLE DIPLOMA_TABELA OF XMLTYPE
XMLSCHEMA "http://www.feri.uni-mb.si/marko2006"
ELEMENT "diploma";
CREATE TABLE DIPLOMA_STOLPEC(
xmlstolpec XMLType)
XMLType COLUMN xmlstolpec ELEMENT
"http://www.feri.uni-mb.si/marko2006#diploma";
```

Pri kreiranju tabele tipa XMLType je potrebno navesti XML shemo, na katero se veže tabela, in iz katere se bodo nato generirale tabele – identična relacijska struktura. Vneseni podatki se bodo transformirali glede na shemo, kateri pripada tabela, in se validirali ob vsakem vnosu v tabele. Podatki bodo tako ostali logično enaki, a ne identični, kar je posledica transformacij.

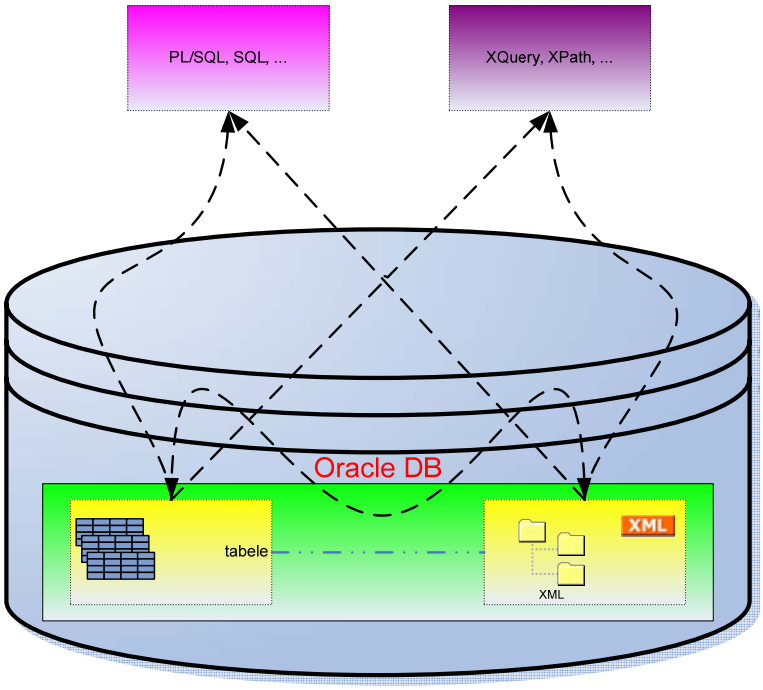
Označena (»Annotated«) shema je XML shema z dodatnimi atributi (metapodatki), ki so dodani shemi – Oracle atributi uporabljajo imensko področje http://xmlns.oracle.com/xdb. Primeri metapodatkov so sledeči: *defaultTable*, *SQLName*, *SQLType*, *SQLCollType*, *MaintainOrder* in ostali. [20] Označene sheme pa omogočajo med drugim nadzorovano shranjevanje XML v podatkovno bazo, poimenovanje SQL objektov in atributov pri preslikavi elementov, v XML shemi definirane podatkovne tipe, shranjevanje zbirk elementov v bazi (maxOccurs>1), podrobno shranjevanje tipa CLOB ter shranjevanje variabilnih tipov kot:

- VARRAY kot LOB,
- VARRAY kot gnezdeno tabelo,
- VARRAY of REF XMLType kot LOB,
- VARRAY of REF XMLType kot gnezdeno tabelo.

Kot slabost se izkaže dodatni »overhead« nad podatki; že XML je dokaj potraten s prostorom; s tem pa še dodatno razširimo velikost s dodanimi metapodatki. Potrebno jih je pazljivo dodajati, saj sčasoma naletimo na težavo glede odzivnosti in zmogljivosti rešitev.

4.5 XML/SQL dvojnost

Kot zelo uporabna se izkaže nova funkcionalnost v Oracle 10g R2 XML DB, kjer lahko hkrati uporabljamo in mešamo tako sintakso XML in SQL. Prekop in uporaba obeh načinov dostopa in manipulacije podatkov zelo olajša razvoj kasnejših rešitev in povečuje njihovo skalabilnost/razširljivost. Omogoča pa lažje pretvorbe med oblikami podatkov in podpori odprtih standardov. Kot slabost se pokaže velika procesorska zahtevnost te fleksibilnosti in nenehne obdelave podatkov. Potrebno pa je poznati več sintakse. [18]

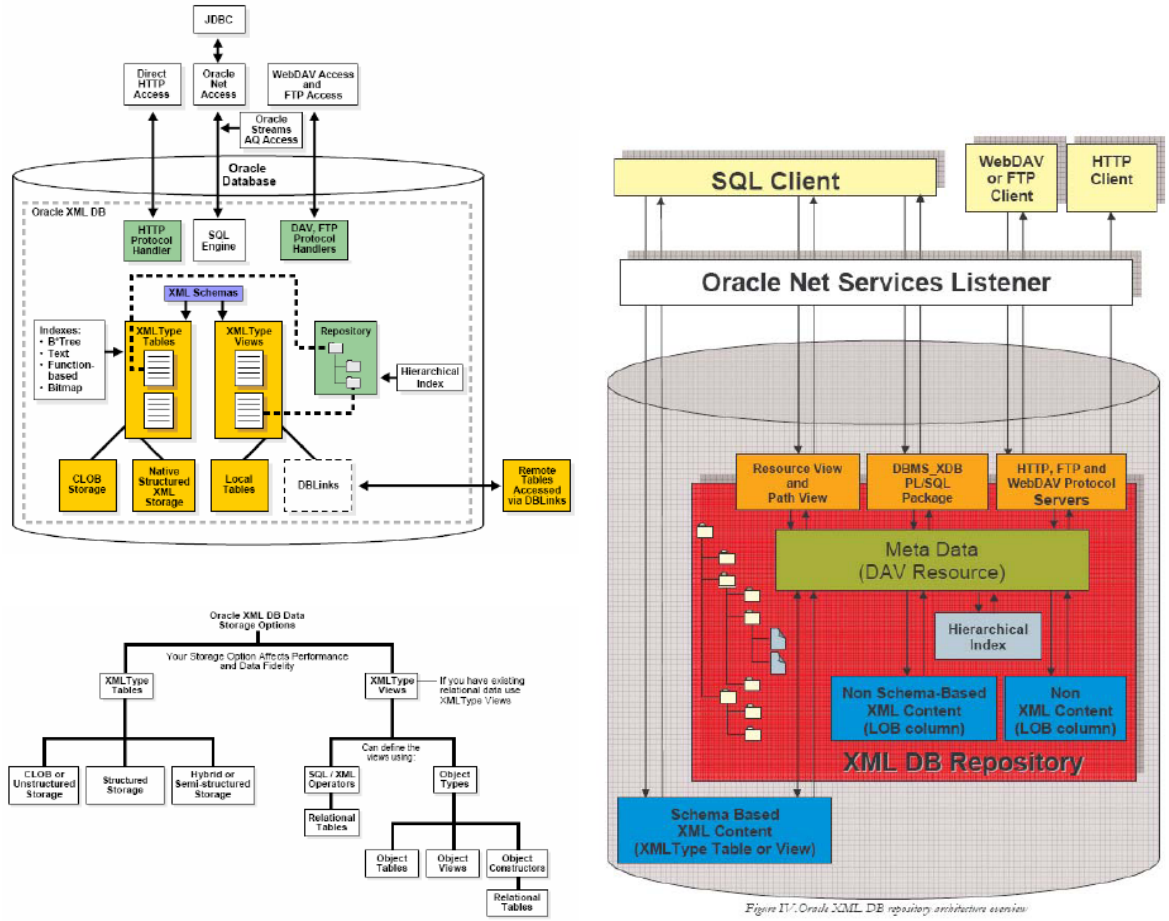


Slika 12: Dvojnost v Oracle XML DB

Dvojnost omogoča predvsem uporabo in enostavno pretvorbo med XML obliko in klasično PL/SQL sintakso. Tako lahko povprašujemo po SQL podatkih in dobimo kot rezultat XML dokument in obratno. Predstavlja pomemben mejnik v razvoju podatkovnih baz.

4.6 XML repozitorij

Oracle XML DB repozitorij predstavlja celoten repozitorij podatkov XML in je posebej optimiran za delo z XML podatki. Vsebino predstavljajo viri (resources), katere predstavljajo vsebniki (containers) in datoteke (files). Identificirajo se z imenom poti in imajo veliko, razširljivo opisno množico lastnosti (metapodatkov), kot so npr. lastnik, datum kreiranja. Prenos v in iz repozitorija se opravi preko standardnih protokolov (FTP, WebDAV, HTTP). [25]



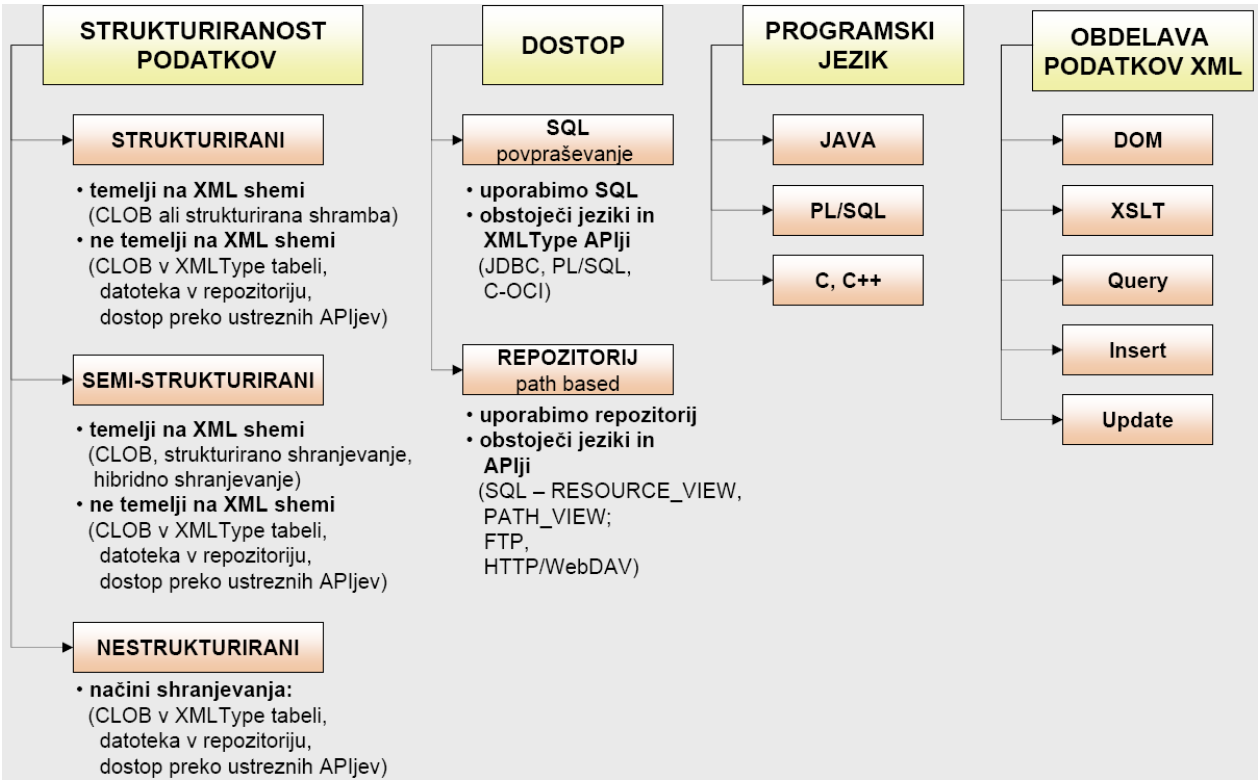
Slika 13: Oracle XML DB repozitorij [18]

Oracle XML DB repozitorij je centralna točka nadzora in dostopa do dokumentov in meta-podatkov določene podatkovne baze. Z zunanjim okoljem je povezan preko Oracle Net Listener-ja, ki služi kot nekak krmilnik namestnik (proxy) vsem zahtevam po dostopu in manipulacijo s podatki. Le-ta je dostopna točka za odjemalce preko vseh podprtih protokolov (WebDAV, FTP, HTTP).

Skrbi tudi za vmesnik med okolico in shrambo XML dokumentov in omogoča njihovo manipulacijo ter obdelavo in shranjevanje.

4.7 Povpraševanje po podatkih z uporabo XQuery in XPath

Povpraševanje po XML dokumentih poteka ali po klasični SQL poti (SELECT, UPDATE, DELETE) ali po predvideni poti za XML dokumente – XQuery in XPath. V primeru Oracle XML DB lahko uporabimo oba načina. Ne samo to, lahko ju tudi izmenično uporabljamo, kar je posledica dvojnosti XML/SQL! [20]



Slika 14: Vidiki načrtovanja rešitev [20]

Načini povpraševanja v Oracle XML DB

- *XQuery* je omogočen z implementacijo SQL/XML funkcij,
- *XMLQuery* – omogoča konstrukcijo/povpraševanje podatkov XML,
- *XMLTable* - omogoča razporeditev ("shreding") rezultirajočih podatkov v relacijske vrstice in stolpce nove - navidezne tabele,
- V *SQL*Plus* je omogočena ukazna vrstica XQUERY in je možno povpraševanje PL/SQL,
- *Oracle XML DB* dodaja nove XQuery funkcije (ki jih ne definira W3C standard): ora:contains, ora:matches, ora:replace, ora:sqrt, ora:view.

Primer za uporabo XQuery-ja v Oracle podatkovni bazi:

```
SELECT XMLQuery(
'for $d in fn:doc("/public/dokument18.xml")/dokument/naslov
let $e := fn:doc("/public/dokument18.xml")/dokument/povzetek
return
<dokument>
{$d} {$e}
</dokument>'
RETURNING CONTENT) FROM DUAL;
SELECT XMLQuery(‘
for $d in fn:doc("/public/dokument18.xml")/dokument/naslov/text()
let $e := fn:doc("/public/dokument18.xml")/dokument/povzetek/text()
return
<dokument>
<naslov>{$d}</naslov>
<abstract>{$e}</abstract>
</dokument>'
RETURNING CONTENT) FROM DUAL;
SELECT XMLQuery(‘
for $d in fn:doc("/public/predstavitev.xml")/diploma/predstavitev
where $d/@id="28" or $d/@id="45"
return <diploma>{$d}</diploma>'
RETURNING CONTENT) FROM DUAL;
```

V XQuery primeru smo iz *dokument18.xml* poiskali vse naslove in povzetke, ter jih vrnili kot XHTML dokument. Na koncu smo še iz *predstavitev.xml* izbrali vse iz ID-jem 28 in 45 ter prav tako rezultate vrnili kot XHTML dokument.

Povpraševanje po XML (XPath)

- operatorji temeljijo na tehnologiji XPath,
- *existsNode()* – preveri če obstaja vozlišče, definirano z XPath,
- *extract()* - vrne vozlišče ali del dokumenta, ki je definirano z XPath,
- *extractValue()* - vrne vrednost elementa ali atributa, definiranega z XPath,
- XQuery se uporablja za povpraševanje.

Primer uporabe XPath jezika v podatkovni bazi Oracle:

```
SQL> SELECT * FROM DIPLOMA_TABELA WHERE
2   existsNode(OBJECT_VALUE,   'diploma/predstavitev/avtorji/avtor="Marko
Bevc"')=1;
<?xml version="1.0" encoding="ISO-8859-2"?>
<predstavitev       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.actual.si/diploma2006 "
xsi:schemaLocation="http://www.actual.si/diploma2006
http://www.actual.si/diploma2006">
<predstavitev id="28">
<avtorji>
<avtor>Marko Bevc</avtor>
...
</avtorji>
<termin>
<datum>22.10.2006</datum>
<ura>15:45</ura>
<prostor>alfa</prostor>
</termin>
<naslov>Podpora tehnologijam XML v podatkovni bazi Oracle</naslov>
...
</predstavitev>
```

S pomočjo izraza XPath smo iz podatkovne baze izbrali celoten dokument iz tabele *diploma*, katerega je avtor »Marko Bevc«. Rezultat je vrnjen in izpisan kot niz.

Primer uporabe funkcije *extract()*:

```
SQL> SELECT extract(OBJECT_VALUE, 'diploma/naslov')
2 from DIPLOMA_TABELA;
EXTRACT(OBJECT_VALUE,'diploma/NASLOV')
--------------------------------------------------------------------------
-------
<naslov xmlns="http://www.actual.si/diploma2006">Podpora tehnologijam XML
v podatkovni bazi Oracle</naslov>
```

S pomočjo funkcije *extract()* smo izpisali samo vrednost vozlišča NASLOV iz dokumenta *diploma*, ki je shranjen v podatkovni bazi.

Primer uporabe funkcije *extractValue()*:

```
SQL> SELECT extractValue(OBJECT_VALUE, 'predstavitev/naslov')
2 from DIPLOMA_TABELA;
EXTRACTVALUE(OBJECT_VALUE,'diploma/NASLOV')
--------------------------------------------------------------------------
-------
Podpora tehnologijam XML v podatkovni bazi Oracle
```

V primeru funkcije *extractValue()* smo naredili podobno, le da smo vrnili rezultat kot niz znakov brez opisa vozlišča – samo vrednost podatkov.

Uporabe kombinacije funkcij *extractValue()* in *existsNode()* v WHERE stavku:

```
SQL>  SELECT  extractValue(OBJECT_VALUE,'predstavitev/naslov')  "Naslov
predstavitve"
2 FROM DIPLOMA_TABELA
3 WHERE existsNode(OBJECT_VALUE,'predstavitev/diploma[@id="28"]') = 1;
Naslov predstavitve
--------------------------------------------------------------------------
-------
Podpora tehnologijam XML v podatkovni bazi Oracle
```

Funkcijo *exsistNode()* lahko prav tako uporabimo v WHERE stavku, kjer določa iskalni pogoj dokumentom v podatkovni bazi.

Relacijski dostop do XMLType s pogledi v Oracle XML DB omogoča definiranje "relacijskih pogledov" s pomočjo vsebin XMLType. S kombinacijo XPath izrazov in funkcij (*extractValue*) lahko definiramo povezavo med stolpci v pogledu in elementi v dokumentu XML.

```
SQL> CREATE OR REPLACE view DIPLOMA_REL_VIEW
2 (ID_PREDSTAVITVE, NASLOV, AVTOR, DATUM, URA, PROSTOR)
3 AS SELECT
4 extractValue(OBJECT_VALUE,'diploma/@id'),
5 extractValue(OBJECT_VALUE,'diploma/naslov'),
6 extractValue(OBJECT_VALUE,'diploma/avtorji/avtor[1]'),
7 extractValue(OBJECT_VALUE,'diploma/termin/datum'),
8 extractValue(OBJECT_VALUE,'diploma/termin/ura'),
9 extractValue(OBJECT_VALUE,'diploma/termin/prostor')
10 FROM DIPLOMA_TABELA;
View created.
```

```
SQL> select naslov, avtor from DIPLOMA_REL_VIEW;
NASLOV
--------------------------------------------------------------------------
-------
AVTOR
--------------------------------------------------------------------------
-------
Podpora tehnologijam XML v podatkovni bazi Oracle
Janez Novak
```

Ažuriranje vsebine XMLType

Za ažuriranje vsebine stolpca tipa XMLType lahko uporabimo sledeče metode:

- uporabimo operator *updateXML()*,
- del vsebine ali celotno vsebino XMLType,
- XPath definira, katere dele je potrebno ažurirati,
- ažuriranje dokumentov, ki so zapisani v CLOB,
- Oracle XML DB zgradi model DOM,
- za ažuriranje posameznih vozlišč uporabi Oracle XML DB metode iz DOM API,
- ažuriranje dokumentov, shranjenih objektno-relacijsko,
- Oracle XML DB izvede SQL stavke za ažuriranje ustreznih objektov.

```
SQL> UPDATE XML_TABELA
2 SET OBJECT_VALUE =
3   updateXML(OBJECT_VALUE,'diploma/naslov/text()','Podpora   XML   v   PB
Oracle')
4 WHERE existsNode(OBJECT_VALUE,'diploma[@id="28"]') = 1;
1 row updated.
```

XMLType pogledi predstavljajo ovoj nad obstoječimi relacijskimi in objektno-relacijskimi podatki v obliki XML. Generiramo lahko podatke XML, ki temeljijo na shemi. Za generiranje XML vsebine se uporabljajo standardni SQL/XML operatorji, kot npr.:

- *XMLElement()*,
- *XMLAttributes()*,
- *XMLForest()*,
- *XMLSequence()*.

```
SQL> CREATE OR REPLACE VIEW XMLPOGLED OF XMLTYPE with object id
(extract(sys_nc_rowinfo$, '//@id').getnumberval())
```

```
2 AS SELECT
3 XMLElement("racun",
4 XMLForest(
5 id as "id",
6 povzetek as "povzetek",
7 datum as "datum",
8 ura as "ura",
9 postavka as "postavka")
10 )
11 FROM PREDSTAVITEV WHERE PREDSTAVITEV.ID='28';
View created.
```

XMLType pogledi so osnova za dvojnost znotraj podatkovne baze Oracle saj omogočajo ovijanje določenih povpraševanj in vračanje poljubne oblike podatkov, ne glede na vhod v povpraševanje in jih lahko kasneje uporabimo v poljubnih povpraševanjih.

Transformacija podatkov XMLType

Podatke XMLType lahko transformiramo na več načinov, kot npr.:

- *XMLTransform()* – kot parametra vzame XMLType in XSLT predlogo,
- *XMLType.transform()*,
- XDK transformacijski mehanizmi na vmesnem sloju.

Te metode služijo kot pomoč pri transformaciji in izmenjavi podatkov v različnih oblikah. Prav tako so osnovni element pri zagotavljanju dualnosti med SQL-om in XML-om!

```
select XMLTRANSFORM(OBJECT_VALUE ,
xdbUriType('/home/MARKO/diploma/xsl/eRacun-14.xslt').getXML())
FROM RACUNI_TABELA where
existsNode(OBJECT_VALUE,'Racun/postavka[@id_postavke="55"]') = 1;
```

Primer prikazuje uporabo mešane sintakse XPath in PL/SQL, kjer izberemo XSLT transformacijo eRačuna in iz njegove XML oblike poiščemo postavko z identifikacijsko številko 55.

4.8 Programska podpora XML dokumentom (Java, C#, itd.)

Programski vmesniki za tip XMLType

Obstajajo sledeči programski vmesniki (API) in knjižnice, ki se uporabljajo za razvoj aplikacij v navezavi z Oracle XML DB:

- PL/SQL API,
- DBMS_XMLSTORE,
- Java API for XMLType,
- C API,
- ODP.NET (ADO.NET r2).

Programski vmesniki za tip XMLType so na voljo za različne programske jezike kot tudi skriptne jezike. Omogočajo lažji razvoj in predvsem služijo kot paket skupnih procedur in funkcij, uporabljenih pri večini najpogostejših programskih operacij. Omogočajo večino najpogostejših načinov dostopa in manipulacije podatkov v Oracle XML DB. S tem se poenostavi uporaba in migracija na nove tehnologije in pristope k razvoju programske opreme. Vgrajen in najstarejši vmesnik komunikacije s podatkovno bazo Oracle je definitivno PL/SQL. S prihodom novih tehnologij je doživel nekaj sprememb in dodane funkcionalnosti ter razširitev izrazne moči.

PL/SQL za XMLType

- XMLType APIji,
- PL/SQL Document Object Model API (DBMS_XMLDOM),
- mehanizmi za delo tako z dokumenti, ki ne temeljijo na shemi XML, in dokumenti, ki temeljijo na shemi XML,
- XML Parser API (DBMS_XMLPARSER),
- XSLT Processor (DBMS_XSLPROCESSOR),
- API nudi funkcije za:
    - kreiranje XMLType tabel, stolpcev in pogledov,
    - pridobivanje podatkov XMLType,
    - manipuliranje s podatki XMLType.

Java programski vmesnik za tip XMLType omogoča več načinov dela v Java programskem vidiku. Lahko delamo preko DOM ali SAX vmesnikov. Slednji je slabše

obvladljiv v smislu načina razmišljanja in obravnave dogodkov. DOM je bolj »naraven« in lažje obvladljiv s stališča človeškega razmišljanja. XML strukturo prečeše (parse) in razdeli ter sestavi v drevesno strukturo. Po njej se lahko pomikamo, jo do neke mere spreminjamo in opravljamo osnovne operacije združevanja, rezanja, transformiranja. Smo pa pri tem načinu dostopa do podatkov in njihove obdelave omejeni z količino pomnilnika, saj so vse podatkovne strukture ves čas shranjene v dinamičnem pomnilniku, kjer se kasneje vršijo tudi same njihova obdelava.

V sklopu oraclovih Java knjižnic in XDB paketa lahko najdemo sledeče pakete, ki pomagajo pri razvoju aplikacij v Javi. Primer dostopa do podatkovne baze preko Java programskih vmensikov lahko vidimo na Sliki 15. Uporabljajo se sledeči paketi knjižnic:

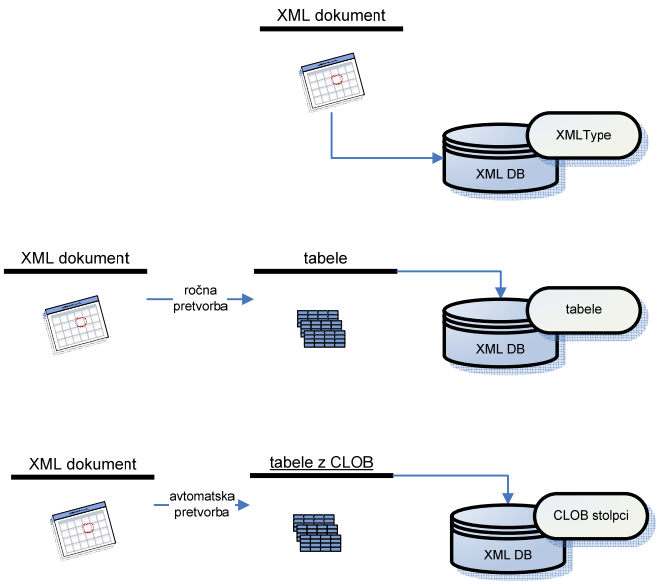
- Java DOM API (oracle.xdb.dom),
- Java DOM API za XMLType razrede (oracle.xdb.XMLType).

```
String sql = "INSERT INTO PRIMER2 VALUES (?)";
XMLType xml = null;
try {
        DriverManager.registerDriver(new OracleDriver());
        XmlDocument xmlDoc =
        XmlDocument.createXmlDocument("imedatoteke");
        OraclePreparedStatement statement = null;
        xml = XMLType.createXML(povezava, xmlDoc.getDocumentElement().toString());
        statement = (OraclePreparedStatement)povezava.getConnection().prepareStatement(sql);
        statement.setObject(1, xml);
        statement.execute();
}catch (Exception e) {
        e.printStackTrace();
}
```

Slika 15: Primer uporabe Java API za Oracle XML DB

5 Predstavitev metode meritev

Eden izmed ciljev in namen izvedenih meritev je primerjati rezultate zmogljivosti različnih pristopov shranjevanja v Oracle 10g R2 podatkovni bazi, ki se trenutno uvršča v sam vrh hranjenja in obdelave podatkov v svetu.



Slika 16: Testirani pristopi (XMLType, tabele in CLOB stolpec)

Že pred meritvami je bilo potrebno podrobno preučiti tehnologije, ki so bile objekt meritev in sicer zaradi postavitve algoritma testiranja in kasnejše medsebojne primerljivosti. Napačno postavljen algoritem bi lahko prinesel neprimerljive rezultate in posledično napačne končne ugotovitve. Na Sliki 16 lahko vidimo vse pristope, ki smo jih izbrali in primerjali v diplomskem delu, ter katere je mogoče uporabiti v Oracle podatkovni bazi.

5.1 Opis postopka meritev

Metoda je zastavljena tako, da izloči čim večje število zunanjih faktorjev, ki bi lahko vplivali na meritev in rezultate meritve. Upoštevani so bili tako faktorji strojne opreme (zasedenost virov z drugimi opravili in procesi, zakasnitve omrežja, virtualizacijski nivo in ostali) kot tudi programski (režija OS, GC, napaka meritev/izračuna). Za čim bolj realno oceno so bile vse meritve opravljene štiri krat in je bilo upoštevano geometrijsko povprečje meritev.

Meritve so bile razdeljene na tri logične sklope meritev. V prvem smo merili odzivne čase relacijskih, v drugem XML CLOB in v tretjem XML object-relational način obravnave podatkov. V vsakem sklopu smo merili sledeče zadeve: vstavljanje (INSERT), povpraševanje (SELECT) in izbris (DELETE) XML dokumentov. Osnovni princip meritev je bil sledeč, kot vhod vedno nastopa XML dokument, ki ga vstavljamo v podatkovno bazo – ga želimo shraniti, do njega dostopati. Pri shranjevanju XML dokumentov v relacijske tabele je pred shranjevanjem bilo potrebno poskrbeti za pretvorbo v objekte – upoštevani so bili tako pretvorbo iz in v XML dokumente/obliko. Operacija popravljanja podatkov ni bila predmet diplomske naloge saj je časovno približno ekvivalentna povpraševanju in vstavljanju. Prav tako se zelo poredko v praktičnih primerih uporablja za spremembo XML dokumentov. Meritev ene operacije je postavljena s formulo, prikazano na Sliki 17. Za vsak sklop meritev določene operacije je vsebovala število ponovitev in znotraj tega še število iteracij posamezne ponovitve. Tako je rezultat vsake ponovitve dobljen s povprečjem iteracij ponovitve. Kot rezultat celotne meritve je pa uporabljeno povprečje vseh ponovitev.

$$t_{sk} = \frac{\sum_{i=1}^{i=N_{repeat}} \frac{t_{end,i} - t_{start,i}}{N_{iter}}}{N_{repeat}}$$

Slika 17: Formula meritve ene operacije sklopa

Določiti je bilo treba smiselne vrednosti za število ponovitev, kot tudi število iteracij v posamezni ponovitvi. Potrebno je bilo upoštevati natančnost podatkovnega tipa *float* in *ure sistemskega časa,* s katero smo merili posamezno operacijo. Prav tako nastane težava ob večanju števila ponovitev kot tudi iteracij in pride do ekspotencialne rasti časov meritev in tako do neprimerno dolgih časov. Tako smo s pomočjo testov in primerjav določili najbolj ugoden razpon rezultatov s sledečimi nastavitvami konstant ponovitev, kot je to razvidno iz Slike 18. v primeru večanja ponovitev ali števila iteracij se je zelo povečal čas celotnih meritev kar zelo oteži njihovo nadaljnje izvajanje.

```
private static int NO_ITER = 1000;
private static int REPEAT = 10;
…
```

Slika 18: Smiselne vrednosti ponovitev

Pri večanju število ponovitev je v primeru XMLType in CLOB prišlo do zelo velikih časov na posamezno fazo meritev. To je posledično pomenilo vedno večjo razliko med najkrajšimi in najdaljšimi časi meritev, kar pa onemogoča dobro primerjavo med posameznimi meritvami. Želimo imeti čim bolj enakovredne oz. primerljive rezultate, ki nekaj povedo. Morajo pa definitivno biti v področju zmožnosti meritve – je možno zajeti razlike s pomočjo merilnih sistemov, ki so na voljo.

5.2 Predstavitev vzorčne sheme in dokumenta XML

Kot vzorčna shema za izvedbo meritev je bila izbrana shema eRačuna standarda eSlog. Pri izbiri je bil odločilen predvsem faktor uporabnosti in razširjenosti uporabe. Prav za eSlog je značilno, da s prihodom XML tehnologij, izmenjave podatkov in odprtih standardov doživlja svoj preporod in dobiva zagon in podporo širšega kroga uporabnikov. [15]

Na Sliki 19 je prikazan izsek osnovne sheme eRačuna in začetek definicij. V času nastajanja tega diplomskega dela je doživela tudi manjšo prenovo iz različice 1.4 na 1.5. Predstavlja nadaljnjo pot razvoja standarda elektronske izmenjave dokumentov.

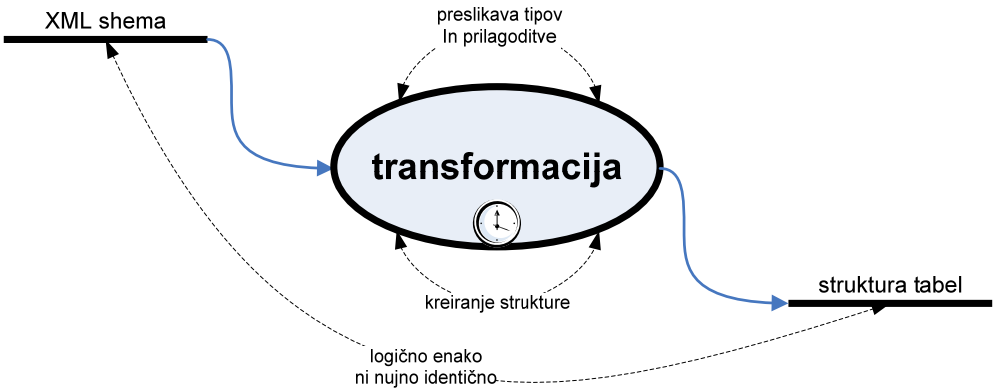
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xds="http://uri.etsi.org/01903/v1.1.1#">
 <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.gzs.si/e-poslovanje/sheme/eSlog_1-
4_PreprostiRacun_signature.xsd"/>
 <xs:annotation>
  <xs:documentation>Schema name: IzdaniRacunEnostavni</xs:documentation>
 </xs:annotation>
 <xs:element name="ZneskiRacuna">
  <xs:annotation>
   <xs:documentation>Povzetek zneskov na racunu (koliko davka, davcne osnove, popustov,...)
(E:Seg-MOA; E:Com-C516_3)</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="VrstaZneska"/>
    <xs:element ref="ZnesekRacuna" minOccurs="0"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="ZneskiPostavke">
  <xs:annotation>
   <xs:documentation>Zneski na posamezni postavki (E:Loop-Group_26; E:Seg-MOA; E:Com-
C516)</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="VrstaZneskaPostavke"/>
    <xs:element ref="ZnesekPostavke" minOccurs="0"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="ZneskiDavkovPostavke">
  <xs:annotation>
   <xs:documentation>znesek davka/trosarine na postavki (E:Seg-MOA; E:Com-
C516)</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="VrstaZneskaDavkaPostavke"/>
    <xs:element ref="Znesek" minOccurs="0"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="ZneskiDavkov">
  <xs:annotation>
   <xs:documentation>znesek davka na racunu (E:Seg-MOA; E:Com-C516)</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="VrstaZneskaDavka"/>
    <xs:element ref="ZnesekDavka" minOccurs="0"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="ZnesekRacuna">
  <xs:annotation>
   <xs:documentation>znesek racuna (E:Loop-Group_41, E:Seg-MOA, E:Com-C516, E:El-
5004)</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
   <xs:restriction base="xs:float">
    <xs:pattern value="([0-9]|\+|\-|E|e|\.){0,18}"/>
   </xs:restriction>
  </xs:simpleType>
 </xs:element>
       <xs:element name="ZnesekPostavke">
      ...
```

Slika 19: Shema eRačuna

Poleg vzorca sheme lahko vidimo (Slika 20) tudi izsek testnega primera XML dokumenta (eRačun), ki je bil uporabljen pri meritvah zmogljivosti in odzivnosti Oracle 10g R2 XML DB. Le-ta je bil testno izpolnjen s pravilnimi podatki kot ekvivalent testnih podatkov ob uporabi takšne sheme. Tako smo tudi izločili tipične načine testiranja osnovnih podatkov, ki niso realni in iz kakšnega praktičnega primera.

```
<?xml version="1.0" encoding="UTF-8"?>
<IzdaniRacunEnostavni xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.gzs.si/e-poslovanje/sheme/eSlog_1-
4_PreprostiRacun.xsd">
 <Racun Id="data">
  <GlavaRacuna>
   <VrstaRacuna>380</VrstaRacuna>
   <StevilkaRacuna>90-1600</StevilkaRacuna>
   <FunkcijaRacuna>9</FunkcijaRacuna>
  </GlavaRacuna>
  <DatumiRacuna>
   <VrstaDatuma>35</VrstaDatuma>
   <DatumRacuna>20040505</DatumRacuna>
  </DatumiRacuna>
  <Lokacije>
   <VrstaLokacije>91</VrstaLokacije>
   <NazivLokacije>Maribor</NazivLokacije>
  </Lokacije>
  <PoljubnoBesedilo>
   <VrstaBesedila>AAI</VrstaBesedila>
   <Besedilo>
    <Tekst1>Na podlagi vašega naročila vam zaračunavamo opravljene storitve:</Tekst1>
    <Tekst2/>
    <Tekst3/>
    <Tekst4/>
   </Besedilo>
  </PoljubnoBesedilo>
 <PodatkiPodjetja>
 <NazivNaslovPodjetja>
  <VrstaPartnerja>BY</VrstaPartnerja>
  <IdentifikacijaPartnerja />
  <NazivPartnerja>
   <NazivPartnerja1>Gospodarska zbornica Slovenije</NazivPartnerja1>
 </NazivPartnerja>
  <Ulica>
   <Ulica1>Dimičeva 13</Ulica1>
 </Ulica>
 <Kraj>Ljubljana</Kraj>
 <NazivDrzave>Slovenija</NazivDrzave>
 <PostnaStevilka>1000</PostnaStevilka>
 <KodaDrzave>SI</KodaDrzave>
 </NazivNaslovPodjetja>
<FinancniPodatkiPodjetja>
<TipInstitucije>RB</TipInstitucije>
<BancniRacun>
 <StevilkaBancnegaRacuna>02924-0017841495</StevilkaBancnegaRacuna>
 <NazivBanke1>Nova Ljubljanska banka d.d.</NazivBanke1>
</BancniRacun>
...
```

Slika 20: Primer XML računa



Slika 21: Pretvorba iz XML sheme v tabele

Pri shranjevanju je potrebno paziti na vsebino XML podatkov v Oracle XML DB podatkovni bazi, zaradi načina shranjevanja. Ob uporabi XMLType se je potrebno zavedati transformacije podatkov – podatki se pretvorijo v objektno-relacijske povezave in se pri povpraševanju rekonstruirajo v ekvivalentno obliko (Slika 21)! Tako lahko pri povpraševanju naletimo na težave, saj dobimo logično iste podatki, ki pa niso točno enaki. To ne velja pri shranjevanju XML dokumenta kot CLOB polje.

5.3 Normalizacija in pretvorba v tabele

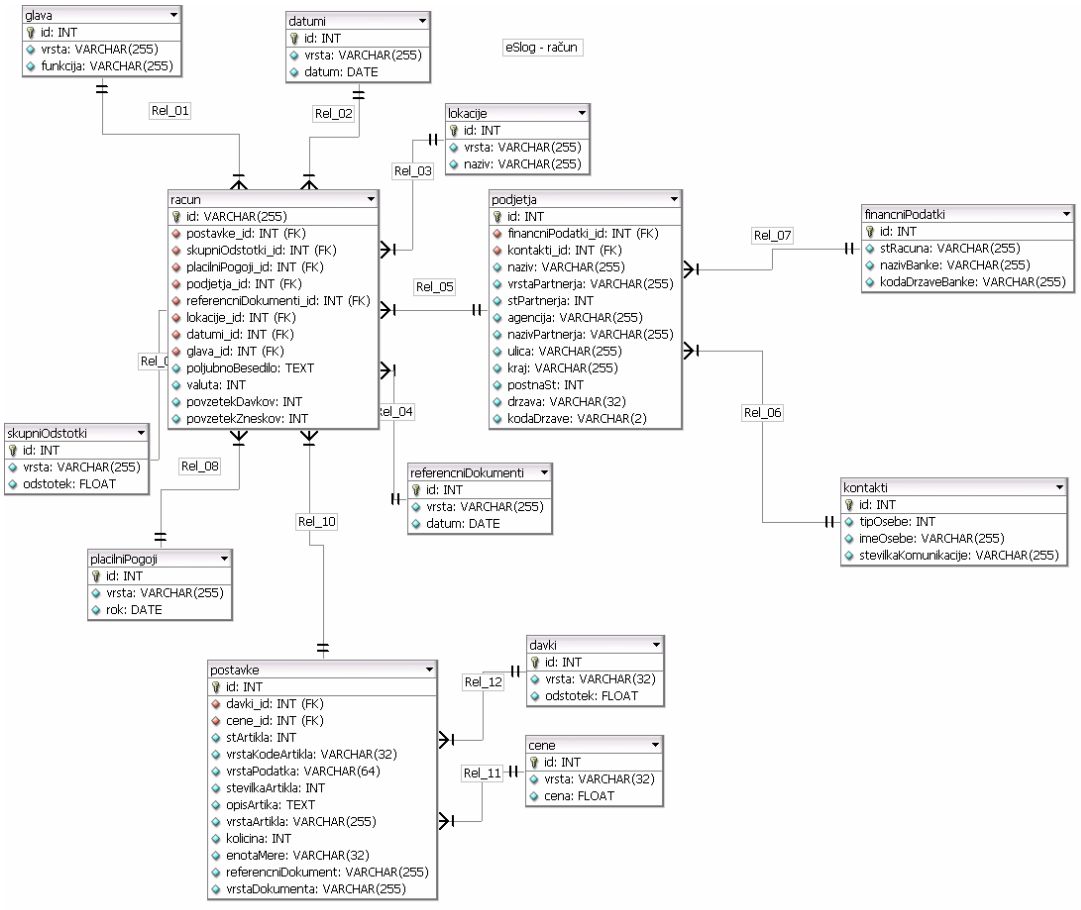
Zaradi primerljivosti relacijskih tabel in XML dokumenta je bil ta pretvorjen v relacijske tabele in normaliziran. Na Sliki 22 vidimo izsek E-R diagrama normaliziranih tabel iz sheme XML eRačuna. Zaradi narave diplomskega dela, časa in področja raziskave so bili določeni deli v podatkovni bazi odstranjeni, tako da se je ohranila funkcionalna ekvivalenca.

Za uporabo relacijskih tabel smo se odločili predvsem iz dveh razlogov – zato, ker so mogoče v obstoječem sistemu, kamor je treba pripeljat tudi podatke iz prejetih XML dokumentov, že prisotne in seveda zaradi primerljivosti z obstoječimi sistemi. Prav tako je bila zanimiva primerjava med lastno »optimizacijo« in »strojno«, ki je prisotna ob internih pretvorbah XML dokumenta v podatkovni bazi Oracle.

Vprašamo se lahko glede smiselnosti pretvorbe, če že ob samem kreiranju tabel tipa XMLType Oracle XML DB pretvori shemo v relacijske tabele. Res je, da uporablja svojo logiko pretvorbe in optimizacij. Pretvorba ne deluje kot čisto relacijska. Nekaj stolpcev

ohrani kot CLOB polja in tuje ključe ter omejitve (constrains) pa ustvari umetno. Takšna pretvorba tako ni berljiva, razumljiva in človeku prijazna. Prav tako ne omogoča nadaljnjega razvoja modela in je ne moremo uporabljati kot orodje pretvorbe XML shem v relacijske tabele. To je definitivno prednost ob uporabi že obstoječih podatkovnih baz in aplikacij, kjer je potrebno izvesti migracijo ali celo hkrati uporabljati oba sistema.

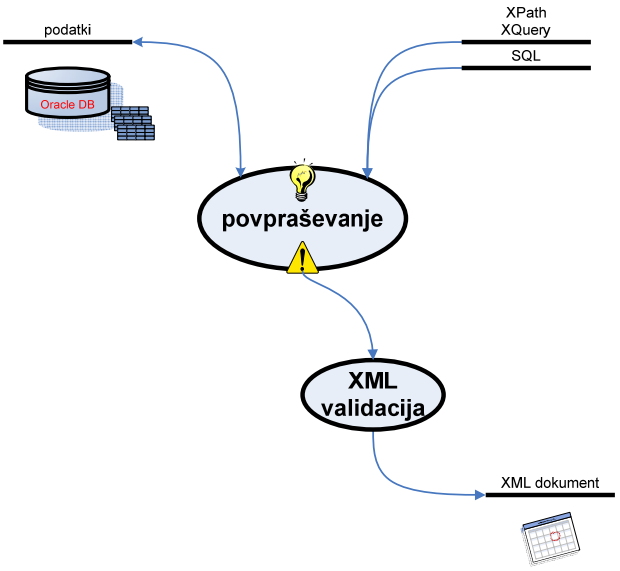
Pri pretvorbi v relacijske tabele smo smiselno pretvorili shemo eSloga za preprost račun v 12 relacijskih tabel. Večina le-teh predstavljajo entitete, ki se relativno malo spreminjajo. To so tabele podjetij, kontaktov, naslovov in podobne. Pri meritvah se je tako v tem primeru realno spreminjalo nekje med 6 do 8 tabel. Pri ostalih je šlo večinoma za začetne inicializacije in kasnejšo uporabo tujih ključev na njihove indekse. Na drugi strani pa imamo optimizacijo že v podatkovni bazi Oracle, kjer se prav tako ne spreminjajo vse zadeve ob vstavljanju in manipulaciji XML dokumentov v XML DB podatkovni bazi. Pri uporabi predhodne začetne polnitve testne podatkovne baze z večjo količino podatkov – XML dokumentov, razlike niso bile bistvene.



Slika 22: Tabele v katere shranimo eRačun

5.4 Opis pretvorb uporabljenih pri meritvah

Pri pretvorbi podatkov smo upoštevali podobnost tipov (INT-DECIMAL, STRING – VARCHAR()) in njihove kardinalnosti. Prav tako je bilo potrebno upoštevati povezave med tipi in omejitve. Pri snovanju primera relacijskih tabel je bila XML shema eSlog računa poenostavljena, normalizirana in racionalizirana, saj vsebuje veliko dodatnih atributov, ki so za namen tega diplomskega dela nepomembni in izven področja raziskovanja. Tako je bil v prvem sklopu meritev zasnovan entitetno-relacijski model sheme XML dokumenta eRačuna po standardu eSlog [15]. V primeru shranjevanja kot XMLType tabela vrši podatkovna baza sama transformacijo v relacijske tabele. Ob uporabi XML CLOB shranjevanja dokumentov pa se le-ti shranijo kot CLOB polje in se ne transformirajo.



Slika 23: Dualnost pri povpraševanjih po podatkih

Kot je razvidno iz Slike 23 se pri kasnejšem povpraševanju in manipulaciji nad XML podatki vrši validacija in nadaljnja transformacija dokumenta. XML DB pa omogoča še eno dodatno funkcionalnost, ki pravzaprav povzdigne uporabo XMLType tipa nad ostala dva pristopa. Gre za t.i. dualnost podatkov – vsi podatki, shranjeni v XMLType tabelah, se lahko uporabljajo kot bi bili XML ali navadni podatki v relacijskih tabelah. Izvajamo lahko mešane operacije povpraševanj in dobljenih rezultatov – XQuery/XPath in dobimo vrstice, ter PL/SQL za XML rezultat povpraševanja.

6 Meritve in analiza rezultatov

6.1 Okolje meritev

Pred meritvami je bilo potrebno vzpostaviti delovno okolje, v katerem so se opravljale meritve. Priprava okolja je potekala po sledečih korakih:

1. vzpostavitev infrastrukture:
    a. postavitev VM,
    b. namestitev operacijskega sistema (RHEL 4 update 4),
    c. namestitev Oracle 10g R2 (zadnji patchset iz MetaLink-a),
    d. konfiguracija XML DB podatkovnega stroja.
2. analiza problema testiranj (priprava testnih primerov).
3. razvoj in namestitev aplikacije.
4. izvedba meritev.
5. zajem rezultatov.
6. analiza rezultatov in ugotovitve.

Okolje meritev:

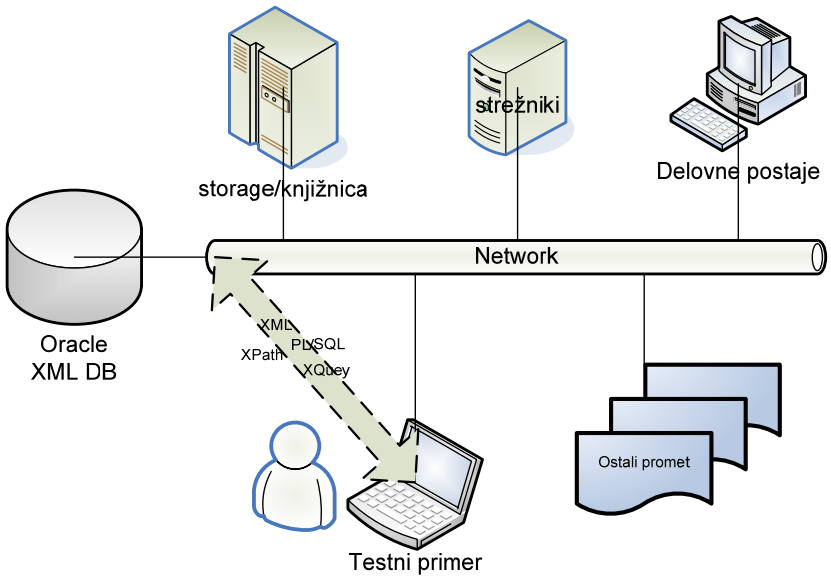
Strojna konfiguracija	Programska konfiguracija

• IBM x346
  o Xeon 3,6 GHz
  o 16 GB RAM DDR II ECC
• IBM DS4800 storage
• Cisco stikala
• Gbps ethernet mrežne povezave
• Qlogic storage host bus adapter
• fiber channel povezave
	• VM (esx 3.0.1)
	• Oracle 10g R2 podatkovna baza
	• Java 1.5.0_09 JDK
	• Linux RHEL 4 Update 4
	• Oracle JDeveloper 10.1.3.1.0

6.2 Potek meritev

Po pripravi infrastrukture, vzpostavitve delovnega in testnega okolja ter analizi zastavljenih ciljev je bilo potrebno izvesti meritve. Zagotoviti je bilo potrebno kar se da optimalne razmere za opravljanje meritev – enakovredne za vse tri sklope meritev. Tako

smo morebitne zunanje vplive (ostali procesi) poizkušali izključiti s pomočjo povprečenja rezultatov in več ponovitvami meritev. Prav tako so se meritve izvajale v času manjše obremenitve virov (strežnikov, omrežja). Kljub poskušanju zagotovitve optimalnih pogojev se je potrebno zavedati morebitnih moteči faktorjev realnega okolja, ki utegnejo nastopiti pri realni uporabi.



Slika 24: Shema okolja meritev

Shema okolja meritev je prikazana na Sliki 24. Okolje je zagotavljajo kar se enakovredne pogoje za vse sklope meritev (relacijske tabele, XML z in brez sheme). Način dostopa do podatkovne baze je bil preko Oracle-ovih »native« javanskih knjižnic za dostop do podatkovne baze (`oracle.jdbc.driver.*, oracle.xdb.*`).

Kot omenjeno so se celotne meritve ponovile štiri krat. Potekale so v treh sklopih; meritve relacijskih tabel, XML podatkov brez preverjanja sheme (CLOB) in s preverjanjem sheme (object-relational transformacija). Vsak sklop je bil sestavljen iz sledečih faz meritvenega algoritma:

1. Inicializacija (vzpostavitev povezave na bazo in inicializacija).
2. Pričetek polnjenja podatkov (INSERT).
3. Povpraševanje (SELECT).
4. Brisanje (DELETE).
5. Zaključek (zapiranje povezave in sprostitev virov).

Ob *inicializaciji* so vse vzpostavile potrebne povezave na bazo in objekti, s katerimi kasneje izvajamo same meritve. Tako smo izločili faktor vsakokratnega kreiranja novih objektov in izgube časa za le-te. Zaradi reference na objekte, ki jih uporabljamo, prav tako preprečimo GC-ju (garbage collector), da po nepotrebnem uniči objekte med izvajanjem meritev.

Pri *polnjenju podatkov* (INSERT) smo merili čase, potrebne za vstavitev XML dokumenta v podatkovno bazo. Vnašali smo enostaven primer XML dokumenta; pri čemer je bil v primeru uporabe relacijskih tabel normaliziran. Pri XML vnosu smo enkrat uporabili za validacijo shemo eSlog-a, drugič pa smo vnašali XML dokument kot CLOB objekt. Vhod je bil XML dokument katerega je bilo potrebno v primeru relacijskih tabel prej pretvorit (deserializirat) iz XML oblike.

Ob *povpraševanju* (SELECT) po podatkih smo iskali določen zapis izmed množice že vstavljenih. Uporabili smo že prej vnesene podatke, ki smo jih vnašali ob fazi merjenja polnjenja podatkov. Iskali smo določen račun s pomočjo iskalnega pogoja po primarnem ključu (oz. indeksiranem atributu dokumenta). Uporabili bi lahko poljuben ključ, saj so le-ti poindeksirani in ponavadi iščemo ravno po njih. Vhod je bil XML dokument katerega je bilo potrebno v primeru relacijskih tabel prej pretvorit (deserializirat) iz XML oblike. Prav tako je bilo potrebno pa rezultat serializirati v XML obliko. Ta proces lahko tudi opravimo znotraj podatkovne baze Oracle s pomočjo dvojnosti SQL/XML.

Brisanje (DELETE) je bila zadnja faza vsakega sklopa, kjer smo izbrisali zapise v podatkovni bazi. Merili smo čase, potrebne za odstranitev vstavljenih zapisov. Za samo merjenje ažuriranja (UPDATE) se nismo odločili, saj ne spada v področje diplomskega dela. Prav tako v praksi po izdaji računa ne spreminjamo in ne dodajamo ali brišemo vnesenih postavk ter tako ne spada v področje te naloge in meritev. Brisanje podatkov se je presenetljivo izkazalo kot zelo potratna operacija v vseh sklopih. Brisali smo dokument glede na indeksiran atribut. Vhod je bil XML dokument katerega je bilo potrebno v primeru relacijskih tabel prej pretvorit (deserializirat) iz XML oblike.

V vsakem sklopu je bila v vseh fazah meritev upoštevana že omenjena formula za izračun rezultatov (povprečen čas meritev):

$$t_{sk} = \frac{\sum_{i=1}^{i=N_{repeat}} \frac{t_{end,i} - t_{start,i}}{N_{iter}}}{N_{repeat}}$$

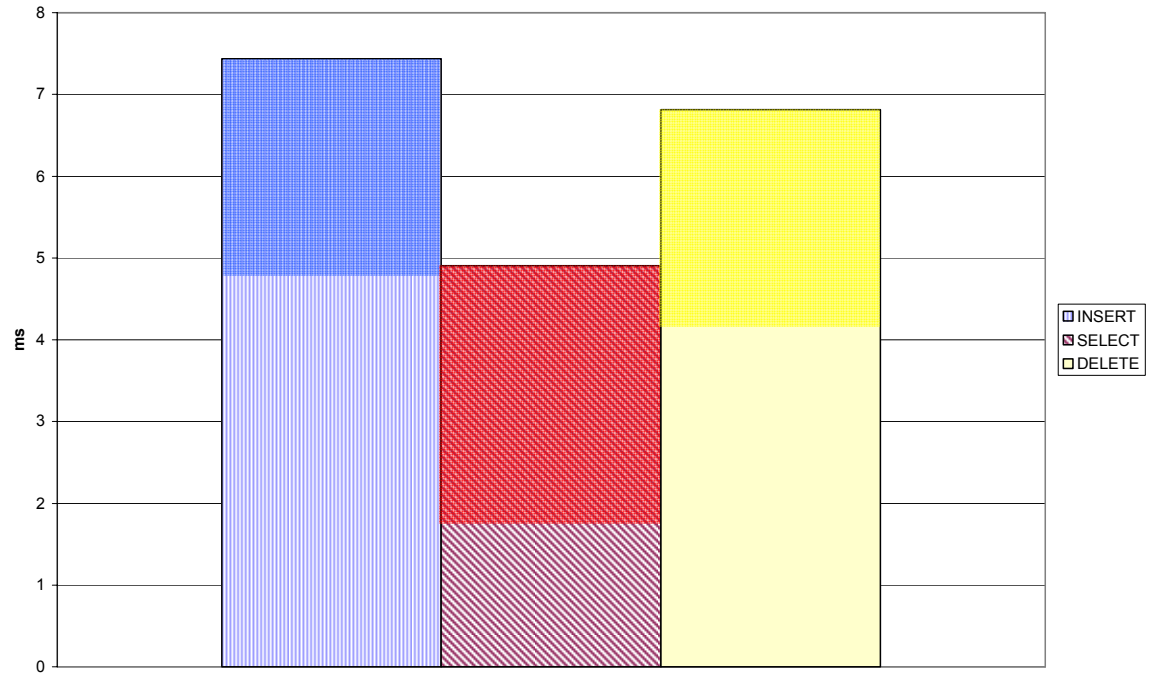
Vsaka meritev je ponovljena za N_{iter} iteracij in glede na število le-teh je izračunano povprečje ene operacije. Vse to se ponovi še N_{repeat} krat, kar predstavlja število iteracij meritve, ki jih nato analiziramo in primerjamo. Zaradi izničevanja različnih zunanjih vplivov (natančnost meritev in podatkovnih tipov ter zasedenosti ostalih virov). Potrebno je bilo eksperimentalno poiskati najbolj optimalne konstante števila ponovitev in iteracij znotraj le-teh. Uskladiti je bilo potrebno natančnost meritve (enota je milisekunda) in dolžino časa najdaljše operacije zaradi medsebojne primerjave. Seveda je prišlo do velikih razponov in je bilo potrebno zaradi lažje in bolj pregledne primerjave uporabiti žal celo logaritemsko skalo.

6.3 Predstavitev rezultatov in analiza

Rezultati dobljeni z meritvami, so potrdili pričakovanja; tabele so še vedno glavni način shranjevanja podatkov in tudi performančno najbolj optimalen. Kljub vsemu se tudi ob hranjenju podatkov v XML DB podatkovni bazi zadeve transformirajo v relacijske tabele glede na uporabljeno shemo. Razen v primeru shranjevanja brez sheme se shranijo XML dokumenti v isti obliki kot CLOB objekti. Upoštevati je prav tako potrebno pretvorbo v objekte oz. iz njih v XML dokumente pri shranjevanje v relacijske tabele. Vedno smo izhajali iz stališča, da je vhodni podatek XML dokument, katerega je potrebno vstaviti ali obdelati v podatkovni bazi. Pri primerjavi zmogljivosti in odzivnosti med XML metodama pa je gotovo prednost vstavljanja brez sheme, saj se le tako ne vrši validacija in transformacija XML dokumentov – vstavljeni so isti dokumenti in se ne spreminjajo. Kar pa prinese kot ceno hitrosti je slabša odzivnost, kasnejše procesiranje in zmogljivost ob iskanju po takšni bazi podatkov kljub temu, da imamo shranjene dejanske dokumente, kot smo jih vstavili v podatkovno bazo – kar je lahko velikokrat zahteva informacijskega sistema ter naša prednost, da se izognemo kasnejšim težava v razvoju. Le-to pa ne velja za shranjevanje preko sheme, kajti tam se zadeve preverijo preko pravil sheme in se shranijo ekvivalentno transformirani podatki! Le-ti so logično identični a regenerirani/transformirani. V tem primeru lahko kvečjemu pričakujemo olajšano

povpraševanje in kasnejše transformacije podatkov v različne oblike in formate, saj nam je na voljo kup orodij in sintaks za delo s pravilno validiranimi XML podatki. Prav tako je takrat lažja objava podatkov preko spleta in njihova izmenjava – odprti standardi.

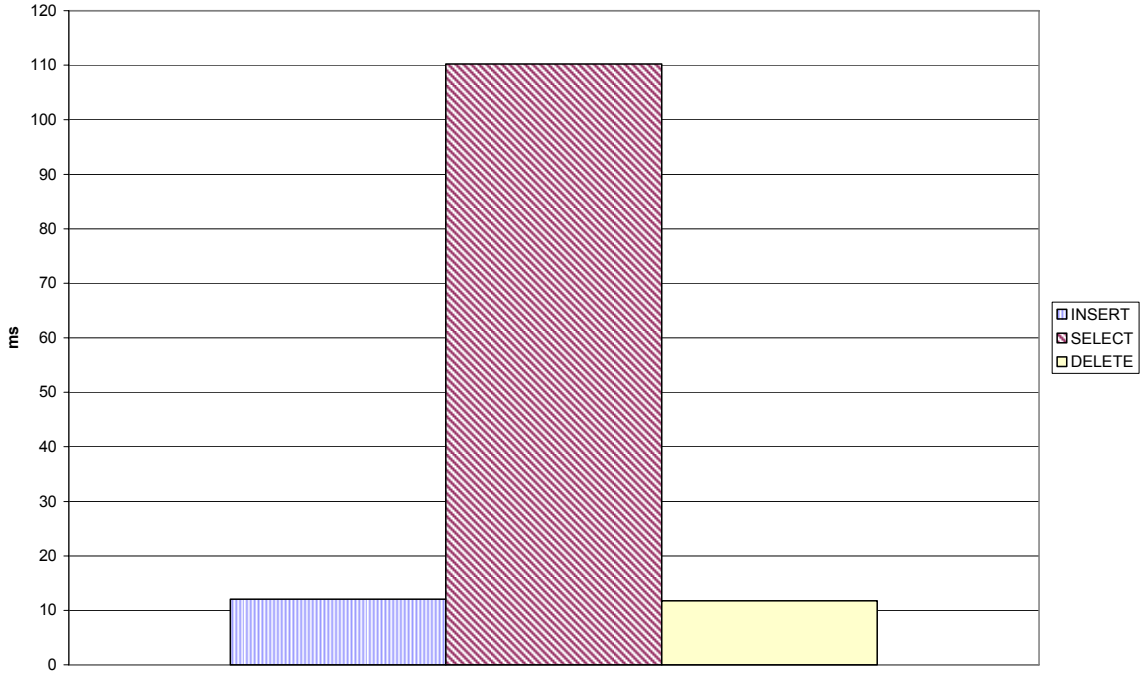
V prvem sklopu smo merili odzivnost klasičnih relacijskih tabel. Le-te so se izkazale kot pričakovano najhitrejši pristop shranjevanja podatkov v podatkovnih bazah – tudi v Oracle 10g R2. Ne glede na rezultate pa v primeru uporabe XML dokumentov pomeni več dela v programski kodi. Kot je vidno iz Slike 25 je časovno najbolj zahtevna operacija vstavljanja podatkov v podatkovno bazo, kjer so časi nekje med 6,6 do 10 ms na posamezno operacijo. Sledi ji operacija brisanja v relacijskih tabelah, kjer se gibljejo časi operacije dosti bolj konstanto med 6,5 in 11 ms. Kot najmanj zahtevna se izkaže operacija povpraševanja podatkov kjer so bili časi med 4,8 in 5,1 ms – kar je še vedno vsaj en krat hitreje od brisanja ali vstavljanja. Opaziti je bilo zamik prvih operacij, kar je posledica kreiranja vseh objektov pred povpraševanji. Upoštevan je bil tudi čas potreben za serializacijo oz. deserializacijo XML dokumenta. Ta je bil za vstavljanje in brisanje nekje 2,6 ms, ter za izbiro XML dokumenta do 3,3 ms (Slika 25).



Slika 25: Rezultati meritev za pristop »relacijske tabele«

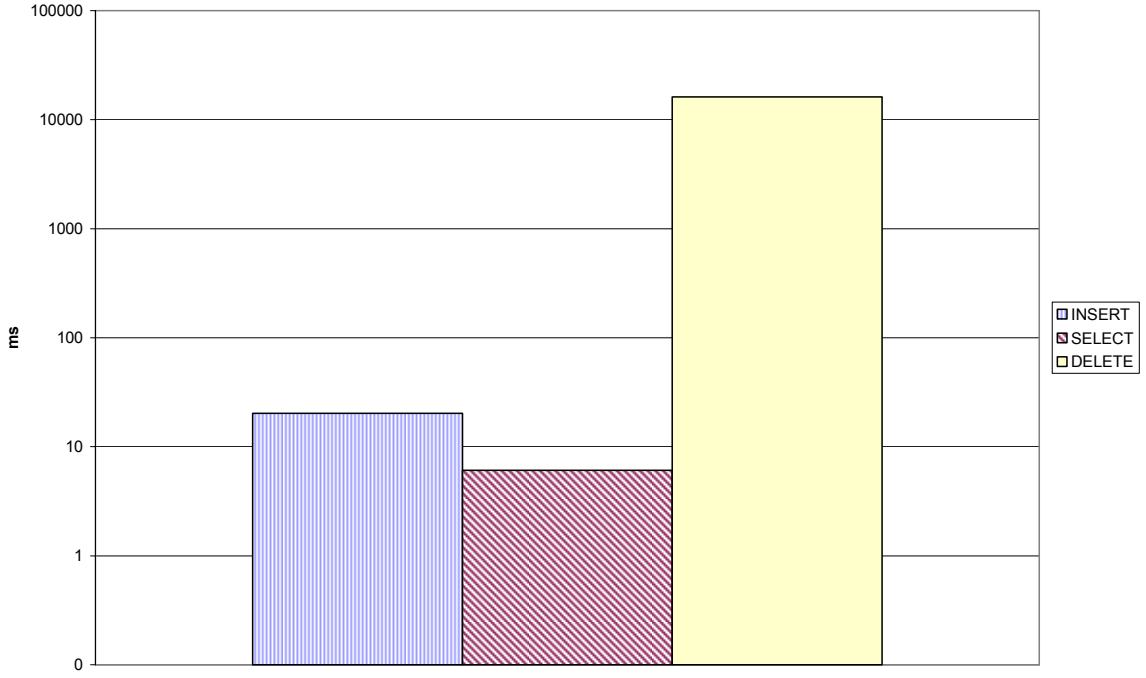
Časi potrebni za serializacijo/deserializacijo XML dokumenta v objekte in nato vstavljanje njihovih vrednosti v podatkovno bazo so bili med 1,2 ms in 2,6 ms. Upoštevali smo prav tako prisotnost obeh operacij, serializacije in nato deserializacije, pri operaciji povpraševanja.

Sledil je sklop merjenja hitrosti operacij v primeru shranjevanja podatkov v obliki XML CLOB – kot direkten vpis v bazo brez preverjanja sheme. Način bi lahko delno primerjali z shranjevanjem XML dokumentov kot navadna TEXT ali BLOB polja v tabelo, saj se vsebina dokumenta shrani v BLOB polje tabele. Kot takšna se tudi obnašajo pri meritvah, kar je razvidno iz Slike 26. Najbolj potratna operacija je tako iskanje po tabeli (SELECT) – kar je posledica iskanja po tekstovnem polju v tabeli. Prav tako je videti naraščajoči trend kar nakazuje na velike strežniške obremenitve zaradi prenosa celotnega dokumenta in njegovo shranjevanje v stolpec tabele. Časi se gibljejo od 32 in vse do 252 ms na povpraševanje. Operaciji sledita operaciji vstavljanja (INSERT) in brisanja (DELETE), ki sta dokaj primerljivi s prejšnjim sklopom relacijskih tabel. Pri vstavljanju so časi med 10 in 15 ms, ter pri brisanju med 10 in 22 ms.



Slika 26: Rezultati meritev za pristop »XML CLOB«

Kot zadnji sklop je bilo na vrsti shranjevanje XML dokumentov kot tabel tipa XMLType, ki so vezane na določeno shemo. Tukaj se XML dokumenti prav tako v ozadju transformirajo v relacijske tabele s standardnimi podatkovnimi tipi in nekaj atributov kot CLOB oblika stolpcev. Podatki so vezani na shemo tabel in njihova pravilnost se validira ob samem vhodu v podatkovno bazo. Tako smo opazili manjšo porast časov ob samem vstavljanju (INSERT) v podatkovno bazo, kjer so bili časi med 18 in 29 ms na operacijo. Tudi časi povpraševanja so malce višji kot pri relacijskih tabelah, a kljub vsemu manjši od vstavljanja, kjer se opravi validacija. Gibljejo se med 5,7 in 8,6 ms. V tem primeru je olajšana predvsem nadaljnja manipulacija nad podatki, medtem ko je brisanje podatkov prineslo presenetljivo zamudo. Časi so bili dokaj porazni in v rangu 1000 krat večji od ostalih operacij vseh sklopov, kar je razvidno iz Slike 27. Razlaga gre predvsem v smeri, da je potrebno veliko preverjanja omejitev (constrains) in medsebojnih povezav generiranih tabel kot tudi shem.

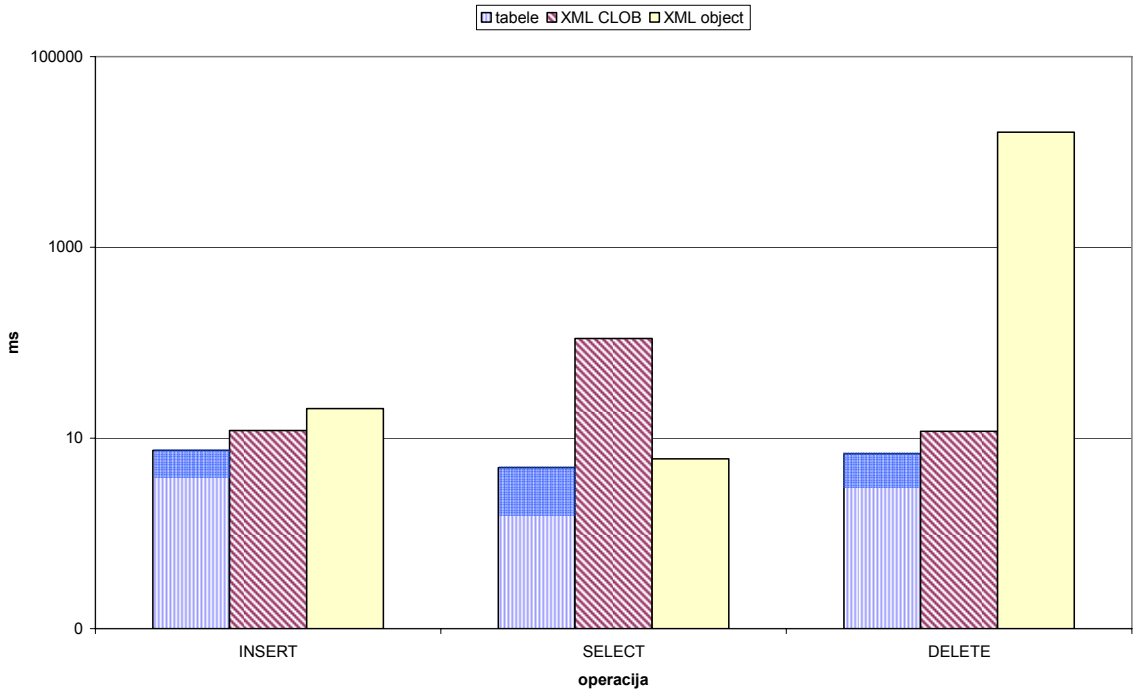


Slika 27: Rezultati meritev za pristop »object XML«

Pri primerjavi med posameznimi sklopi lahko, kot je vidno iz Slike 28, ugotovimo da je razen brisanja iz XMLType tabel zadeva dokaj izenačena. Izstopa predvsem brisanje v primeru XMLType tabel, saj je potrebno pobrisati vse relacije in razdrobljene podatke po

tabelah – tudi upoštevati omejitve (tuji ključi). V ostalih primerih je vseeno povsod v manjši prednosti še shranjevanje v relacijske tabele. Med XMLType tabelami in klasičnim shranjevanjem pa je zelo majhna razlika. V primeru vstavljanja (INSERT) je najslabši XMLType, zaradi validacije glede na shemo – ni pa velikih razlik. Pri povpraševanju po podatkovni bazi se kot enakopravna izkažeta oba pristopa, tako relacijske tabele kot XMLType. CLOB polje pa izstopa, saj je operacija iskanja zelo obremenjujoča za strežnik.

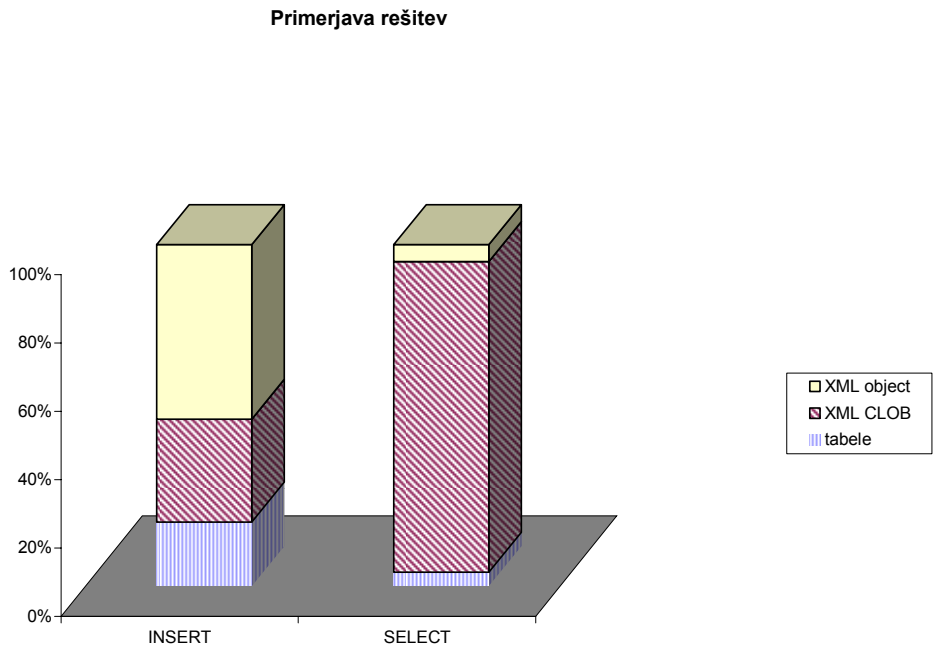
V medsebojni primerjavi posameznih načinov shranjevanja se izkažejo vsi trije načini dokaj enakovredni (Slika 28). Pri vstavljanju (INSERT) za malenkost vodi relacijski način shranjevanja XML dokumentov z 7,22 ms. Pri povpraševanju (SELECT) pa se kot enakovredna pokažeta tako relacijski način kot XMLType. Izstopa le način XML CLOB pri operaciji povpraševanja in XMLType pri operaciji brisanja (DELETE), kjer so rezultati dobljeni slabši za faktor 1000. Pri izvedbi meritev smo upoštevali kot vstopno točko XML dokument, ki ga je potrebno vstaviti, poiskati ali izbrisati iz podatkovne baze. Tako je bilo potrebno pri relacijskem načinu dodati kos kode za pretvorbo XML dokumenta v Java objekte, katerih vrednosti smo nato shranili v tabele. Za pretvorbo je bilo dodanih okoli 100 vrstic kode. Za delo z XML dokumenti v Javi je bila uporabljena Sun-ova knjižnica JAXB.



Slika 28: Primerjava meritev

Na Sliki 28 je prav tako prikazan delež časov potrebnih za lastno pretvorbo XML dokumentov v primeru uporabe relacijskih tabel. Le-ta, kot lahko vidimo na grafu, ni zanemarljiv in je pomemben tako za izvajanje aplikacij, kot tudi pri razvoju rešitev lastne pretvorbe in normalizacije XML sheme. Predvsem pa ne smemo prezreti dejstva, da je prednost tehnologije XML v nadaljnji najenostavnejši manipulaciji nad podatki in ne samem shranjevanju v podatkovno bazo ter njihovem brisanju. Kljub temu pa lahko rečemo, da je prišlo do napredka tudi v tej smeri, saj so se zadeve zelo približale zmogljivostim klasičnega shranjevanja in še bolj težijo k uporabi novih tehnologij. Le-te imajo tako še bolj široko odprta vrata k masovnem prodoru in širši uporabi v širokem spektru aplikacij kot tudi v podatkovnih bazah prihodnosti. Presenetili so le nekoliko rezultati brisanja pri XMLType tabelah. Ostalo je bilo nekako pričakovano in logično povezano z zahtevami in omejitvami tehnologij. Pomembna je tudi vpeljava XML jezika v podatkovne baze iz vidika storitveno orientirane arhitekture, ki vedno bolj pridobiva na pomenu in se uveljavlja.

V globalni primerjavi časa, potrebnega za uporabo posamezne tehnologije v enakomerni razporeditvi operacij, bi lahko rekli, da gre za zelo izenačene rezultate, kot lahko vidimo iz Slike 29. Opaziti je zelo majhne razlike med posameznimi pristopi shranjevanja v podatkovno bazo. Izstopajo določene situacije, ki narekujejo pogoje za uporabo teh pristopov v določenih primerih. Tukaj smo primerjali seštevke povprečnih časov vseh operacij. Iz tega je razvidna primerjava med povprečno porabljenim časi vseh operacij skupno na posameznem načinu shranjevanja XML dokumentov v podatkovni bazi. Medtem, ko so relacijske tabele še zmerom globalno v manjši prednosti (23% časa) XMLType (XML object) ne zaostaja dosti (30%). Kot najslabša alternativa se pokaže XML CLOB s 47%. V kolikor lahko v aplikacijski rešitvi 2 ms nista ključnega pomena za izvajanje procesov (realni čas) in želimo podatke nadalje oblikovati, jih posredovati v poljubni obliki ter podpreti novo prihajajoče odprte standarde, se lahko skoraj brez težav odločimo za uporabo pristopa XMLType. Je gotovo pot, ki zagotavlja kar se da enostavno integracijo s prihajajočimi standardi in tehnologijami, ter smernicami sodobnega razvoja programske opreme.



Slika 29: Primerjava časov za operaciji INSERT in SELECT

Poleg vseh naštetih zadev je potrebno paziti še pri izbiri strojne opreme, saj v primeru manj zmogljive opreme pride do drastičnega povečanja odzivnih časov dostopa in manipulacije nad podatki. Uporaba jezika XML je zaradi dodatnih validacij in transformacij precej procesorsko in pomnilniško zahtevna, kar lahko odvrne prehod na novo tehnologijo marsikatero podjetje z neustrezno ali pomanjkljivo načrtovano infrastrukturo. Kljub vsemu je mnogo bolj obetavna in naravnana v prihodnost ter zelo primerljiva s tem kar smo imeli do sedaj. Kaže, da bo to naslednja pot pri razvoju rešitev shranjevanja podatkov in se bo dokaj uvajala še v prihodnosti.

Tabela 3: Rezultati meritev

čas v ms	tabele	XML CLOB	XML object	odklon:	tabele	XML CLOB	XML object
INSERT	7,4368	11,9984	20,3272		13,17%	27,31%	16,94%
SELECT	4,9072	110,2506	6,0588		2,50%	67,29%	15,19%
DELETE	6,8123	11,7584	16194,7435		22,34%	35,59%	0,00%
geo. Povprečje							
					meritve:		
tabelce					9,9892	15,5012	29,3588
INSERT	SELECT	DELETE			7,2693	12,8397	18,5037
9,9892	2,3608	11,2551			7,1936	11,8359	18,2990
7,2693	2,2528	6,4194			7,1731	16,3665	20,4779
7,1936	2,2253	6,4307			6,8713	17,0445	19,1522
7,1731	2,2131	6,4390			7,1063	13,2619	19,7801
6,8713	2,2079	6,4462			8,2979	11,7964	23,2962
7,1063	2,1997	6,4543			6,7623	8,2821	18,5835
8,2979	2,1850	6,4544			7,3679	7,6081	18,2609
6,7623	2,1925	6,4349			6,8408	9,5457	19,7554
7,3679	2,1777	6,4468					
6,8408	2,1710	6,4582			2,3608	33,2139	8,6949
					2,2528	50,7644	5,9631
XML CLOB					2,2253	66,7121	5,6579
INSERT	SELECT	DELETE			2,2131	89,3149	5,8209
15,5012	33,2139	24,0496			2,2079	112,9422	5,6271
12,8397	50,7644	10,3066			2,1997	135,5900	5,7819
11,8359	66,7121	10,8217			2,1850	163,1823	6,1492
16,3665	89,3149	10,7316			2,1925	188,7146	5,8549
17,0445	112,9422	10,3661			2,1777	222,4075	5,7836
13,2619	135,5900	11,4440			2,1710	251,8205	5,7623
11,7964	163,1823	10,9329					
8,2821	188,7146	10,8540			11,2551	24,0496	16194,7435
7,6081	222,4075	11,4085			6,4194	10,3066	16194,7435
9,5457	251,8205	10,9287			6,4307	10,8217	16194,7435
					6,4390	10,7316	16194,7435
XML object					6,4462	10,3661	16194,7435
INSERT	SELECT	DELETE			6,4543	11,4440	16194,7435
29,3588	8,6949	16194,7435			6,4544	10,9329	16194,7435
18,5037	5,9631	16194,7435			6,4349	10,8540	16194,7435
18,2990	5,6579	16194,7435			6,4468	11,4085	16194,7435
20,4779	5,8209	16194,7435			6,4582	10,9287	16194,7435
19,1522	5,6271	16194,7435					
19,7801	5,7819	16194,7435					
23,2962	6,1492	16194,7435		connect	444,0650		
18,5835	5,8549	16194,7435					
18,2609	5,7836	16194,7435					
19,7554	5,7623	16194,7435					

V tabeli 3 lahko vidimo, da je čas povezave na bazo, ki je bil prav tako izveden 4 krat in povprečen z geometrijskim povprečjem, nekje 444 ms. To predstavlja velik delež glede na same kasnejše operacije in tako potrjuje tudi nadaljnjo uporabo persistentnih povezav oz. tehnologij bazenov povezav (*connection pool*). Tako lahko poleg rezultatov meritev – ki so že statistično obdelani z geometrijskim povprečjem štirih meritev, med seboj primerjamo tudi geometrijsko povprečje vseh ponovitev posameznih sklopov. Izračunan in prikazan je tudi standardni odklon posameznih sklopov in faz znotraj le-teh. Največji odklon je opaziti ravno pri XMLType načinu, kar je posledica odstopanja od ostalih rezultatov. Po fazah znotraj posameznega sklopa pa opazimo povečanje odklona pri brisanju podatkov (DELETE) glede na ostale operacije sklopov. Kar pomeni, da je operacija brisanja zelo občutljiva in je njena odzivnost zelo odvisna od trenutno razpoložljivih virov in sposobnosti strežnika ter komunikacijskih poti do odjemalcev.

Tabela 4: Razlike obravnavanih pristopov

lastnosti\pristopi	tabele	CLOB XML	object-relational XML
INSERT indeks	1	1,5	2,7
SELECT indeks	1	22	1,2
DELETE indeks	1	2	1000
kje je shranjen dokument	v tabelah	v kolonah	preslikane tabele
kdo ga validira	preverjanje na strani implementatorja	odgovornost na strani odjemalca	Oracle DB glede na shemo
kaj je potrebno v programski kodi	pretvorba iz oz. v XML dokument	ni dodatne implementacije	ni dodatne implementacije
prednosti (+)	kompatibilnost nazaj, performančno boljša izbira	uporaba klasičnih tabel (kompatibilnost) in shranjevanje ter podpora XML shranjevanju	popolna preslikava dokumentov v podatkovno bazo z popolno podporo
slabosti (-)	ročna pretvorba XML dokumentov – zamudno in rutinsko opravilo	slaba podpora manipulaciji in povpraševanju po XML dokumentih	slabše performančno od tabel a bolje kot CLOB

Kot lahko vidimo v Tabeli 4 se pokaže cena hitrosti nekaterih pristopov shranjevanja kot delegiranje nekaterih operacij, kot je recimo preverjanje pravilnosti vhodnih podatkov, na stran implementatorja in njihov čas izvajanja ter razvoja prav tako ni zanemarljiv. Prav

tako je iz indeksa operacije razvidno relativno razmerje in primerjava med posameznimi operacijami različnih pristopov shranjevanja XML dokumentov v podatkovni bazi Oracle. Prav tako je potrebno ob uporabi relacijskih tabel zagotoviti ustrezno pretvorbo oz. serializacijo v XML obliko ter iz nje – vhodni podatek je XML dokument! To prinese dodatno kodo in seveda čas procesiranja, ki je drugače opravljeno v podatkovni bazi. Ni tudi zanemarljiva razlika optimizacije normalizacije in pretvorbe XML dokumentov v primeru »človeške« in »strojne« transformacije – pomembno vlogo igra kompleksnost modeliranega podatkovnega modela.

Iz razvijalskega vidika je v primeru uporabe pristopa relacijskih tabel razvoj malo otežen in zahteva razvoj dodatne kode (nekje okoli 100 vrstic) za pretvorbo v oz. iz XML dokumentov. Prav tako je uporaba klasičnega načina dostopa (JDBC, ODBC in podobni programski vmesniki) do podatkovne baze in repozitorija XML dokumentov okorna, ter nepregledna za razvijalce (prisotnost XML oznak v programski kodi). Ob uporabi pristopa XMLType pa se kot prednost in možnost agilnega ter hitrega razvoja programske opreme izkaže možnost uporabe standardnih protokolov dostopa in uporabe repozitorija XML dokumentov. Do njih dostopamo in jih uporabljamo kot navadne datoteke na »oddaljeni« lokaciji, do katere lahko dostopamo preko protokolov kot so HTTP, FTP in WebDAV. Kljub enostavnosti uporabe XML repozitorija je za razvoj kompleksnejših rešitev in za nadaljnji razvoj gotovo bolje uporabiti že znane dostope do podatkovnih baz.

Rezultati so bili nato nekako delno pričakovani v tej smeri. Presenetili so le rezultati brisanja XML dokumentov z uporabo XMLType tipa, saj so bili bistveno slabši od relacijskega in CLOB načina shranjevanja dokumentov. Razlika je dober pokazatelj koliko je dejansko procesiranja in dodatnega preverjanja pri transformaciji in validaciji XML podatkov pred dejanskim kasnejšim relacijskim shranjevanjem. Pokaže se tudi koliko je dobra optimizacija v podatkovni bazi proti »ročni« normalizaciji in optimizaciji shranjevanja XML dokumentov – kar je gotovo slabša izbira, že zaradi zamudnosti opravila.

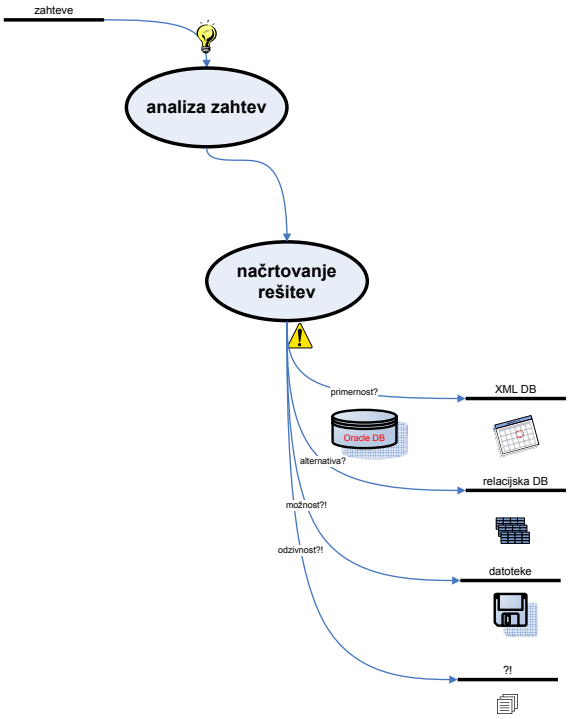
7 Možnosti aplikativne uporabe in integracija v obstoječe rešitve

7.1 Načrtovanje rešitev

Vse implementacije je potrebno že načrtovati z mislijo na modeliranje XML dokumentov in kasnejše uporabe teh tehnologij. Niso pa vse implementacije primerne za XML – primernost je treba upoštevati glede na nadaljnjo uporabo in manipulacijo podatkov; napačne uporabe in ob nepoznavanju ozadja posameznih tehnologij lahko privede do zavajajočih rešitev in slabega delovanja ob uporabi dobrih tehnologij.

Kot vsaka zadeva imajo tudi tehnologije svoj namen uporabe. Potrebno je upoštevati vse smernice razvoja in namen uporabe le-teh ter ciljno področje uporabe. V primeru napačne uporabe ni mogoče pričakovati optimalnih rezultatov. Zato si je potrebno ob vsakem načrtovanju informacijskih rešitev treba pretehtati sledeča vprašanja (Slika 30):

- primernost rešitve/tehnologije,
- možnost alternativ,
- možnost same uporabe glede na platformo in želje stranke,
- pričakovana/zahtevana odzivnost v danem primeru na uporabnih podatkih.



Slika 30: Izbira tehnologije za implementacijo rešitve

Za načrtovanje informacijskih rešitev so ključne prve faze razvoja programske opreme z vidika kasnejšega odpravljanja napak. Kasneje ko odkrijemo napako, več sredstev, ki smo jih vložili v sam razvoj, smo porabili zaman. Prav tako zelo narastejo stroški razvoja – kar je v današnjem času zelo neugodno glede na kratke roke in omejena sredstva razvoja! Tako je ključnega pomena, da se v začetku postavijo dobre smernice in primerni ter realni cilji, katere je možno dosegati v časovnih okvirjih projektnega plana in v zmožnostih dodeljenih virov in sredstev na sam projekt.

Pri uporabi XML tehnologij je potrebno upoštevati tip podatkov, način obdelave, hranjenja, kasnejših manipulacij in povpraševanja po podatkih. S temi parametri lahko dokaj natančno določimo, ali je XML tehnologija primerna za naš primer rešitve obdelave/hranjenja podatkov ali ne. Prav tako je treba upoštevati strojno zahtevnost rešitve z uporabo XML tehnologij – procesorska zahtevnost.

Kot prednost uporabe implementacij z uporabo XML tehnologij je izkaže možnost odprtosti izmenjave podatkov ter spletnih storitev. Po drugi strani plačamo to s precej prekomerne (overhead) uporabe procesorske moči podatkovnega strežnika in diskovnega prostora ob fizičnem hranjenju podatkov.

7.2 Primerjava implementacij

Posamezne implementacije obstoječih rešitev s pomočjo XML tehnologij se razlikujejo predvsem po uporabi različnih prednosti XML tehnologije in z različnim namenom. XML tehnologija se uporablja iz sledečih razlogov in olajšave razvoja/obdelave podatkov: [24]

- avtomatsko preverjanje pravilnosti podatkov – validacija;
- izpostavljanje storitev navzven – spletne storitve, povezovanje, odprtost;
- lažja manipulacija podatkov;
- uporaba API-jev za transformacijo in povpraševanje po podatkih;
- sledenje trendom razvoja programske opreme in optimizacija razvoja.

Med obema tipoma načrtovanja programske opreme obstajajo manjše razlike. Potrebno je upoštevati specifike posameznih tehnologij in njihove značilnosti – treba poznati tudi kanček delovanja iz ozadja. Predvsem gre za razlike med načrtovanjem

podatkovnih struktur in njihovih kasnejših obdelav. Pri klasičnem načrtovanju moramo tipično kasneje poskrbeti za nadzor nad pravilnostjo podatkov, njihovo konsistenco in formatiranje. Nasprotno lahko dosežemo z XML tehnologijo, saj imamo tukaj na voljo veliko pripomočkov, orodij in programskih vmesnikov kadar gre za samo validacijo podatkov (sheme), povpraševanje po njih (XPath, XQuery) in njihovo kasnejšo manipulacijo (DOM, SAX). Ključnega pomena je predvsem upoštevanje prednosti in slabosti posameznih tehnologij in pravilna ter predvsem primerna uporaba le-teh glede na potrebe in zahteve s strani uporabnika/naročnika!

Pomembne lastnosti podatkovne baze Oracle XML DB se izražajo predvsem v funkcionalnostih, kot so npr. hibridna baza, kar omogoča tako relacijsko vstavljanje kot tudi uporaba XML jezika, uporaba XML podatkovnega modela (s tipom XMLType), navezava z XML shemo, DOM, XPath, XQuery ter XML operacije: shranjevanje, povpraševanje, ažuriranje, transformiranje in druge načine obdelave podatkov XML. Ključna prednost dvojnega (SQL/XML) načina delovanja podatkovne baze so XML operacije nad relacijskimi podatki (XML pogledi) in obratno. Repozitorij XML omogoča dostop preko standardnih protokolov kot so datotečni sistem, FTP, HTTP in WebDAV. Hranjenje XML dokumentov v podatkovni bazi omogoča in prinaša sledeče funkcionalnosti ter prednosti:

- obdelava XML bližje podatkom (večja skalabilnost in zmogljivost aplikacij),
- generiranje dokumentov XML, transformacija, povpraševanje, ažuriranje, upravljanje z velikimi dokumenti XML,
- eliminacija nivojev, namenjenih obdelavi dokumentov XML,
- sprejemanje XML podatkov in njihovo začasno shranjevanje XML dokumenti,
- procesiranje XML-a, ekstrahiranje relacijskih podatkov in shranjevanje v relacijsko shemo – dualnost XML in relacijsko.

Centraliziran nadzor in repozitorij dokumentov pa omogočata kreiranje XML pogledov iz relacijskih shem, učinkovito pretvorbo in manipulacijo XML dokumentov ter odprtost, dostop (kreiranih iz XML pogledov) preko spletnega vmesnika in izmenjavo dokumentov preko odprtih standardov.

Kdaj uporabiti Oracle XML DB

Nekaj primerov kdaj bi si želeli oz. bi bilo najbolj optimalno uporabiti XML tehnologije v podatkovni bazi Oracle (XML DB), da dosežemo optimalni izkoristek in produktivnejši razvoj programske opreme. Oracle XML DB uporabimo, kadar želimo procesirati veliko število dokumentov XML, kadar so potrebni zmogljivi mehanizmi iskanja (znotraj dokumenta, čez zbirko dokumentov), v primeru izmenjave dokumentov - strukturiranih dokumentov (poslovni dokumenti – naročila, računi) in nestrukturiranih dokumentov (dokumenti, sporočila) ter je dokumentna vsebina dobro strukturirana. XMLType tip pa omogoča ohranitev paradigme XML in hkrati izkoristek prednosti relacijske podatkovne baze.

Tipi aplikacij, za katere je PB Oracle XML DB primerna:

- integracija (B2B, B2C, EAI),
- aplikacije in storitve, dostopne preko standardnih internetnih protokolov,
- aplikacije za upravljanje z vsebinami.

Pred samo vpeljavo in uporabo XML podpore v podatkovni bazi se vsekakor treba zavedati in si odgovoriti na vprašanja, ki si jih zastavimo, ko se odločamo za/proti različnim pristopov shranjevanja (XMLType, CLOB ali tabele) dokumentov v podatkovni bazi Oracle:

- ali že obstajajo tabele obstoječih aplikacij,
- kdo jamči za veljavnost vstavljenega dokumenta,
- ali mora dokument ohraniti popolno identičnost,
- zmogljivosti strojne opreme,
- zahtevan nivo odzivnosti aplikacij in s tem povezano odzivnostjo podatkovne baze,
- nenazadnje tudi znanje in sposobnosti sodelujočih!

8 SKLEP

V diplomskem delu je bila prikazana primerjava med shranjevanjem XML dokumentov v relacijskih tabelah, kot CLOB stolpec v tabelah in XMLType podatkovnim tipom v podatkovni bazi Oracle 10g R2 (XML DB). Prav tako je bila predstavljena primerjava med uporabo sheme za hranjenje podatkov v XML obliki in brez njene uporabe. Najprej je bil predstavljen vidik uporabe posameznih tehnologij in njihova primerjava; nato je bila opisana novejša izmed njih – XMLDB ter možnosti uporabe s shemo in brez nje.

XML format je trenutno najpogosteje uporabljen za izmenjavo in shranjevanje dokumentov [28]. Pri shranjevanju podatkov je relacijska baza fleksibilnejša in performančno boljša rešitev kot pa XML shramba. Standardne operacije v podatkovni bazi so uveljavljene funkcionalnosti znotraj jedra relacijske podatkovne baze in so razširljive do več terabajtov podatkov. Hranjenje XML dokumentov v podatkovni bazi je relevantno dokler dokumenti ne postanejo podatki. Kadar moramo generirati več dokumentov, migracijo iz »native« bazne aplikacije v kakšno drugo, novo heterogeno aplikacijo potrebujemo predvsem učinkovito XML shranjevanje. V kolikor pravilnost XML dokumenta jamči že pošiljatelj načeloma ni težav – a se v produkcijskih okoljih na to žal ne moremo zanašati.

Podatkovna baza Oracle je vodilna na področju sistemov za upravljanje s podatki že vrsto let. S celovito podporo tehnologije XML in njenih derivatov ter omogočanjem polnega upravljanja vsebin XML (ustvarjanje, povpraševanje, brisanje, ažuriranje, preoblikovanje, itd.) ustvari zelo pomemben člen v verigi komponent, ki so potrebne za razvoj sodobnih informacijskih rešitev.

Zastavili smo si analizirati in primerjati različne pristope shranjevanja XML dokumentov v podatkovni bazi Oracle. Zaradi obsega dela smo se v diplomskem delu omejili na tri osnovne pristope shranjevanja in jih medsebojno primerjali glede na čas potreben za osnovne operacije. Prav tako je bilo zastavljeno, da primerjamo tri osnovne operacije dela s podatki – INSERT, SELECT, DELETE. Osrednja primerjava je bila opravljena med podatkovnim tipom XMLType in klasičnimi relacijskimi tabelami. Tip XMLType se na koncu prav tako pretvori v tabele, po definirani transformaciji. Meritve so pokazale, da manjše spremembe lahko bistveno vplivajo na rezultate meritev.

Meritve so v večini primerov potrdile pričakovane rezultate. Nekoliko pa so presenetili rezultati shranjevanja XML dokumentov kot CLOB polje in brisanje pri pristopu shranjevanja kot XMLType. Razlika je dober pokazatelj koliko je dejansko procesiranja in dodatnega preverjanja pri transformaciji in validaciji XML podatkov preden se le-ti dejansko shranijo v relacijski model. Pokaže se tudi koliko je dobra optimizacija v podatkovni bazi proti »ročni« normalizaciji in optimizaciji shranjevanja XML dokumentov – kar je gotovo slabša izbira, že zaradi zamudnosti opravila. Prav tako je potrebno ob uporabi relacijskih tabel zagotoviti ustrezno pretvorbo oz. serializacijo v XML obliko ter iz nje. Izhajali smo zmeraj iz izhodišča, da je vhodni podatek XML dokument. To prinese dodatno kodo in seveda čas procesiranja, ki je drugače opravljeno v podatkovni bazi. Zanemarljiva ni niti razlika optimizacije normalizacije in pretvorbe XML dokumentov v primeru »človeške« in »strojne« transformacije – pomembno vlogo igra kompleksnost modeliranega podatkovnega modela.

Uporabnost XML tehnologij v podatkovnih bazah se kaže v prednosti pri uporabi in povezovanju z »okolico« preko odprtih standardov in spletnih storitev. Izkaže se, da sta pristopa z uporabo relacijskih tabel in tipa XMLType približno enakovredna in je možno uporabiti XMLType kot zamenjavo za relacijske tabele. Upoštevati je potrebno le omejitve pristopa, kot sta recimo infrastrukturna zahtevnost in daljši časi brisanja dokumentov iz podatkovne baze. Performančno sta v manjši prednosti uporaba klasičnih relacijskih tabel, katera je pa omejena z dobrim načrtovanjem in ročno optimizacijo že med samo analizo in načrtovanjem rešitve ter shranjevanje XML dokumentov v XMLType tabelo. Pričakovano sledi uporaba XML CLOB načina, ki shrani XML dokumente v tekstovni stolpec tabele.

Meritve so pokazale, da se pojavljajo opazne razlike med uporabo sheme ob shranjevanju v podatkovno bazo ali shranjevanjem brez sheme. Pri uporabi le-te naletimo na nekaj počasnejše vnašanje v podatkovno bazo, saj se ob vsakem vnosu podatki validirajo z uporabo sheme, ter se preveri njihova skladnost s shemo. V kolikor vneseni podatki ne ustrezajo shemi, se vnos ne izvede. Ob veliki frekvenci vnesenih podatkov v podatkovno bazo se pojavijo velike časovne zakasnitve, saj je sam postopek validacije procesorsko precej zahteven. Po drugi strani se rešimo problema preverjanja podatkov pred vnosom. V primeru, da sheme ne uporabimo, so vnosi skoraj dva krat hitrejši, vendar je preverjanje pravilnosti vnesenih podatkov potrebno opraviti ločeno. Prav tako je iskanje in manipulacija podatkov v podatkovni bazi v tem primeru počasnejša, kot pa pri uporabi

sheme, kjer so podatki že validirani in je struktura le-teh znana in je tako sama manipulacija enostavna in procesorsko manj zahtevna. Brez uporabe sheme se validacija prenese v fazo manipulacije – nastanejo zakasnitve v obdelavi.

Zaradi vedno večje uporabe XML dokumentov za podatkovni prenos in izmenjavo sporočil se izkaže shranjevanje XML dokumentov v podatkovni bazi prava rešitev. Tudi vse več podatkov iz podatkovnih baz transformiramo v XML dokumente in nazaj v objektno relacijsko obliko. Prav tako obstaja težka obvladljivosti velike količine dokumentov, ki se danes nabirajo na velikih datotečnih sistemih. Bilo bi zelo težavno učinkovito dostopati in manipulirati s takšnimi zbirkami podatkov. Rešitev, s pretvorbo najprej v objektno-relacijsko obliko in nato shranjevanje le-te v relacijske podatkovne baze, je neprimerna. Kot glavna pomanjkljivost se izkaže predvsem večkratna transformacija podatkov – najprej v objektno, in nato še v relacijsko obliko podatkovnih baz, ki še dodatno upočasni povpraševanja. Tako je potrebno zagotoviti enostavne in učinkovite operacije iskanja, spreminjanja in izmenjave dokumentov, ki jih ponujajo ravno XML podatkovne baze.

Iz analize meritev lahko povzamemo, da je uporaba tipa XMLType v podatkovni bazi Oracle dolgoročno najučinkovitejša rešitev shranjevanja XML dokumentov. Kljub strojni zahtevnosti in slabši odzivnosti pri brisanju dokumentov je v večini primerov zelo dobra izbira za shranjevanje podatkov v podatkovni bazi Oracle. S podporo XML dvojnosti v podatkovni bazi zagotavlja povezljivost in združljivost z vsemi preteklimi kot prihajajočimi načini shranjevanja podatkov v podatkovni bazi. Prav tako omogoča dostop preko najpogosteje uporabljenih protokolov (HTTP, FTP in WebDAV). Tako lahko najbolje izkoristimo vse prednosti, ki jih prinaša jezik XML in smo najbolje pripravljeni na nove tehnološke izzive, ki jih prinaša s seboj.

VIRI, LITERATURA

[1] Aida Kamišalić, TEHNOLOGIJA ZA PODPORO RAZPRŠENEMU RAČUNALNIŠTVU, GRID TECHNOLOGY, diplomska naloga, september 2003

[2] Aleks Jakulin, Elektronski trgi, [http://kt.ijs.si/aleks/B2B/b2b.htm] , september 2001

[3] Andrej Krajnc, XML in podatkovne baze, OTS' 2000, [http://cot.uni-mb.si/COTL/krajnc.htm], obiskano avgust 2006

[4] BerkleyDB, [http://sleepycat2.inetu.net/products/bdbxml.html] , januar 2006

[5] BizTalk, [http://www.biztalk.org] , avgust 2006

[6] CommerceNet, [http://www.commerce.net] , avgust 2006

[7] cXML, [http://cxml.org] , obiskano september 2006

[8] EAN Slovenija, [http://www.ean.atnet.si] , obiskano avgust 2006

[9] ebXML, [http://ebxml.org] , obiskano avgust 2006

[10] eCo, [http://eco.commerce.net] , obiskano avgust 2006

[11] EDI/XML, [http://www.xmledi-group.org] , obiskano julij 2006

[12] Electronic Commerce Times, [http://www.ecommercetimes.com] , obiskano avgust 2006

[13] Geoffrey E. Bock, Mainstreaming XML-Based Enterprise Applications, Using Oracle XML DB to Manage, Financial Information within a Global, Banking System, Patricia Seybold Group, november 2003

[14] Going Native: Making the Case for XML Databases, Ronald Bourret , [http://www.xml.com/pub/a/2005/03/30/native.html in http://www.rpbourret.com/xml/XMLDatabaseProds.htm], marec 2005

[15] Gospodarska zbornica Slovenije, [http://www.gzs.si] , obiskano julij 2006

[16] Introduction to Native XML Databases, Kimbro Staken, online article, [http://www.xml.com/pub/a/2001/10/31/nativexmldb.html] , oktober 2001

[18] Oracle Database 10g Release 2: A Revolution in Database Technology , An Oracle White Paper, [http://www.oracle.com/technology/products/database/oracle10g/pdf/BWP_Overview_10gR2_060205.pdf], maj 2005

[18] Oracle XML DB tehnology library, [http://www.oracle.com/technology/tech/xml/xmldb/index.html] , povzeto junij 2006

[19] OTS' 2006, [http://cot.uni-mb.si/ots2006/], obiskano julij 2006

[20] SIOUG, Boštjan Šumak, Luka Pavlič, Maja Pušnik, Podpora tehnologijam XML v PB ORACLE, [http://www.sioug.si], obiskano junij 2006

[21] OAGI, [http://www.openapplications.org] , obiskano julij 2006

[22] Oasis, [http://www.oasis-open.org] , obiskano julij 2006

[23] OBI, [http://www.openbuy.org] , obiskano julij 2006

[24] Oracle XML DB vs. IBM DB2 XML Extender, An Oracle Technical White Paper, Februar 2003, [http://www.oracle.com/technology/tech/xml/xmldb/Current/IBMDB2.PDF] , povzeto november 2006

[25] Oracle documentation library, [http://www.oracle.com/pls/db102/homepage] , obiskano avgust 2006

[26] Oracle Slovenija, [http://www.oracle.si] , obiskano oktober 2006

[27] RosettaNet, [http://www.rosettanet.org] , obiskano december 2006

[28] Shouvik Basu, Managing XML using Oracle's XMLDB, [http://orafaq.com/node/508], povzeto oktober 2006

[29] UN/CEFACT, [http://www.unece.org/cefact] , obiskano september 2006

[30] W3C, [http://w3c.org] , obiskano november 2006