Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Vlada Semenova

Primerjava XML podatkovnih baz

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Dejan Lavbič

Ljubljana 2013

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00119/2013
Datum: 10.04.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **VLADA SEMENOVA**

Naslov: **PRIMERJAVA XML PODATKOVNIH BAZ**
COMPARISON OF XML DATABASES

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

Poleg relacijskih in NoSQL podatkovnih baz, pri razvoju poslovnih aplikacij, poznamo tudi XML podatkovne baze, ki podpirajo XPath in XQuery procesiranje podatkov. Poleg omenjenih funkcionalnosti poizvedovanja pa podpirajo tudi številne druge, kot je npr. podpora delovnim tokovom ipd. V okviru diplomske naloge bi bilo potrebno pregledati področje in kritično primerjati obstoječe odprtokodne in komercialne rešitve. Primerjava naj bo izvedena na izbrani problemski domeni, kjer prikažete prednosti in slabosti uporabe XML podatkovne baze. Za primerjavo določite različne scenarije ter kriterije na podlagi katerih boste primerjavo izvedli.

Mentor:

doc. dr. Dejan Lavbič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisana Vlada Semenova z vpisno številko **63090183**, sem avtor diplomskega dela z naslovom:

Primerjava XML podatkovnih baz

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom doc. dr. Dejana Lavbiča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 12. avgusta 2013 Podpis avtorja:

Zahvala

Zahvaljujem se vsem, ki ste mi pomagali pri izdelavi diplomske naloge. Posebna zahvala gre mojemu mentorju doc. dr. Dejanu Lavbiču za pomoč pri izbiri teme in za vse nasvete pri izdelavi diplomske naloge. Zahvaljujem se tudi družinskemu prijatelju Sreču Vengarju, ki mi je pomagal pri ideji za praktični del in s splošnimi nasveti za izdelavo diplomske naloge. Zahvaljujem se Nataši Bizjak in Matjažu Bertonclju za lektoriranje diplomske naloge. Zahvala pa gre tudi mojim staršem za vso podporo tekom študija in pri pisanju diplomske naloge.

Kazalo

Seznam uporabljenih kratic

XML Extensible Markup Language (razširljiv označevalni jezik)

SGML Standard Generalized Markup Language (standardni posplošeni označevalni jezik)

HTML Hyper Text Markup Language (jezik za označevanje nadbesedila)

XHTML Extensible HyperText Markup Language (označevalen jezik, ki ima namen kot HTML, vendar je usklajen s sintakso XML)

ISO International Organization for Standardization (mednarodna organizacija za standardizacijo)

CERN European Organization for Nuclear Research (evropska organizacija za jedrske raziskave)

DTD Document type definition (opredelitev tipa dokumenta)

SQL Structed Query Language (poizvedovalni jezik v relacijskih podatkovnih bazah)

XQuery Poizvedovalni jezik v XML podatkovnih bazah

XPath XML Path language (poizvedovalni jezik po elementam XML)

FLWOR FOR, LET, WHERE, ORDER BY, RETURN (izraz v poizvedovalnem jeziku XQuery)

API Aplication Programing Interface (aplikacijski programski vmesnik)

CRM Customer Relationship Management (upravljenje odnosov s strankami)

Povzetek

Današnji svet je poln XML dokumentov, ki omogočajo prenosljivost podatkov, ne glede na izbrano okolje. Vprašanje je, kako bi bilo najbolje shranjevati te dokumente ali podatkovne baze, ki so, specializirane za XML dokumente, zares potrebne. Namen diplomske naloge je preučiti področje XML podatkovnih baz, prikazati način uporabe le-teh, primerjati 3 najbolj znane predstavnike XML podatkovnih baz ter pokazati, katera podatkovna baza je najbolj primerna za naš praktični primer. Diplomska naloga je sestavljena iz dveh delov.

V prvem delu je opisan teoretični del, kjer so prikazani osnovni podatki: kaj je jezik XML, specifikacija jezika XML, kateri jeziki se uporabljajo pri XML podatkovnih bazah, kaj je XML podatkovna baza, katere lastnosti ima, katere podatkovne baze bodo primerjane med sabo ter njihove lastnosti.

Problemska domena in izbrani scenariji ter podrobnosti implementacije in testiranje so opisani v drugem delu diplomske naloge. Praktični del vsebuje implementacijo podatkovnih baz ter testiranje scenarijev, ki so podani za problemsko domeno. Podatke smo sami zgenerirali s pomočjo podatkovne baze AdventureWorks. V zaključku so predstavljeni rezultati primerjave XMl podatkovnih baz ter ugotovitev, katera podatkovna baza je najbolj primerna za problemsko domeno. Na koncu so navedene sklepne ugotovitve o XML podatkovnih bazah.

Abstract

Today's world is full of XML documents, which allow the portability of data, regardless of the selected environment. The question is, what is the most optimal way to store those necessary files or databases, which are specialized for XML documents. The purpose of the thesis is to examine the scope of XML databases, see how to use XML databases and compare the 3 most famous representative of XML databases and show which database is the most suitable for our practical example. Thesis consists of two parts.

The first part describes the theoretical part, wherein the basic information: What is XML, XML specification, which languages are used in XML databases, what is the XML database, properties of XML database, which the database will be compared with each other and their properties.

Problem domain and the selected scenarios, and details of the implementation and testing are described in the second part of the thesis. The practical part consists an implementation of databases and testing scenarios, which are given in the problem domain. The data was generated through the use of the database AdventureWorks. The results of the comparison of XML databases and findings on which database is best suited for the problem domain are presented in the conclusion. All conclusions about XML databases are listed at the end of the thesis.

Poglavje 1

Uvod

XML dokumentov je na svetovnem spletu zelo veliko. Podjetja izmenjujejo XML dokumente, do njih potrebujejo hiter dostop, zato se poraja vprašanje, kako bi XML dokumente shranili na najbolj optimalen način ter najlažje poizvedovali po teh dokumnetih. XML dokumente lahko shranjujemo tudi v relacijske podatkovne baze, vendar kasneje, v fazi povpraševanja po podatkovni bazi nastopijo težave. Z večanjem količine shranjenih podatkov se veča tudi nepreglednost podatkovnih baz. XML podatkovne baze so bile zame novo in zanimivo področje, ki sem ga želela podrobneje raziskati ter v praksi preveriti, ali so res uporabne v današnjem svetu, kakšne lastnosti imajo in kdaj jih je najprimerneje uporabiti.

Cilj diplomske naloge, ki smo si ga zastavili, je podrobnejša preučitev področja nastanka XMl dokumentov; ugotovitev zakaj bi sploh uporabili specializirane XML podatkovne baze ter katere trenutne rešitve obstajajo. Na primeru problemske domene odločiti, katera podatkovna baza je najbolj primerna za določen scenarij. Trenutno je nastalo več različnih rešitev. Obstajajo izvorne XML podatkovne baze ter podatkovne baze, ki omogočajo XML. V diplomske nalogi bomo primerjali ti dve vrsti podatkovnih baz ter na praktičnem primeru pokazali, kdaj je najbolje uporabiti katero vrsto podatkovnih baz. XML podatkovne baze so novo in zato še malo raziskano področje.

Vsebinsko je diplomska naloga sestavljena iz dveh delov: *teoretičnega* ter *praktičnega*.

Cilj prvega dela je opredeliti glavne pojme, pregledati poizvedovalne jezike, ki se uporabijo v XML podatkovnih bazah, to so XQuery ter XPath, opisati podatkovne baze, ki smo jih izbrali glede na par kriterijev. Za primer XML podatkovnih baz smo izbrali tri baze: BaseX, eXist ter MS SQL SERVER 2008 R2 EXPRESS. Prve dve bazi sta predstavnici XML izvornih podatkovnih baz, zadnja pa je predstavnica XML omogočenih podatkovnih baz. V prvem delu bodo predstavljene značilnosti teh podatkovnih baz. Prav tako bomo poskusili raziskati, zakaj sploh potrebujemo XML podatkovne baze.

Drugi del opisuje problemsko domeno s predvidenimi scenariji. Peto poglavje je*(Problemska domena)* , v kateri opisujemo našo problemsko domeno, ki je CRM, v šestem poglavju *(Testiranje)* smo predstavili testno okolje, testne podatke ter samo implementacijo testiranja. Analiza pridobljenih rezultatov je predstavljena v 7. poglavju *(Analiza rezultatov)*. Diplomsko delo zaključujejo sklepne ugotovitve, ki so predstavljene v zadnjem poglavju *(Sklepne ugotovitve)*.

Poglavje 2

XML

Zaradi razširjenosti uporabe XML dokumentov na številnih področjih, se je pojavila velika potreba po shranjevanju dokumentov v bazo. Obstajajo podatkovno naravnani ter dokumentno naravnani XML dokumenti; glede na vrsto dokumentov je smiselno izbrati pravilen pristop ter podatkovno bazo. Za shranjevanje lahko uporabimo različne vrste podatkovnih baz: relacijske podatkovne baze, objektno-relacijske, XML omogočene baze ter XML izvorne baze.

Na začetku so bili podatki shranjeni v relacijsko podatkovno bazo ter XML uporabljali kot medij za transport podatkov med aplikacijami. Ta model je bil podprt s tako imenovanimi XML omogočenimi podatkovnimi bazami, ki so dokument razgradili in nato shranili v tabele. Kasneje je velikost XML dokumentov naraščala in XML omogočene baze niso bile zmožne podpirati dokumentov tako velikih razmerij, saj je razčlenitev in reformatiranje ogromnih dokumentov hkrati pomenilo veliko število dodatnih operacij, kar je posledično vodilo do preobremenjene zmogljivosti; tako dostopa do dokumentov kot njihovega preverjanja. Zaradi tega se je pojavil nov tip XML podatkovnih baz, ki ga imenujemo izvorne XML baze, in je bil razvit za shranjevanje XML dokumentov [2].

2.1 Jezik XML

XML(eXtensible Markup Language) je enostaven, fleksibilen tekstovni format, ki temelji na SGML-ju (ISO 8879). Določen je bil leta 1989, uveljavil se je leta 2001 [3].

Leta 1996 je W3C začel iskati praktično rešitev. Za osnovo je vzel metajezik Standard Generalized Markup Language ali krajše SGML. *SGML* je razvila mednarodna organizacija ISO (International Organization for Standardization) leta 1986 pod oznako ISO 8879. Za osnovo so vzeli jezik GML (Generalized Markup Language), ki so ga že konec šestdesetih let razvijali pri IBM (Charles Goldfarb idr.). SGML je standard za defeniranje strukture ter vsebine elektronskih dokumentov. Problem SGML-ja je, da je predrag (draga orodja), presplošen in preveč kompleksen za svetovni splet. SGML danes uporabljajo v velikih industrijah in vladnih organizacijah: v letalskem prometu, avtomobilski industriji, telekomunikacijah, ameriškem ministrstvu za obrambo, v Evropski uniji in drugje (citiranje iz vira homeizum clanek o xml) [7].

HTML je razvil Tim Berners-Lee s sodelavci na inštitutu za jedrsko fiziko CERN v Ženevi leta 1990. HTML je razvil iz SGML-ja, vendar je HTML le aplikacija SGML-ja in ni metajezik. HTML ima točno določene značke, ki imajo svoj natančno določen in predpisan pomen. Čeprav je HTML najbolj uporabljen jezik opisuje le, kako naj brskalnik prikaže besedilo, s fokusom na tem, kako podatek izgleda.

XML je prav tako kot SGML metajezik, vendar je z njim enostavneje omogočiti izdelavo novih označevalnih jezikov, prilagojenih specifičnim potrebam. XML je bil razvit z namenom, da je lahko samo opisoval podatke, ki bi bili razumljivi tako človeku kot računalniku. XML tako kot SGML uporablja oznake (tags) za določanje interne strukture dokumenta. Na prvi pogled so oznake podobne tistim v HTML-ju, razlika med XML in HTML je, da pri XML uporabimo oznake, ki namesto o tem, kako naj podatek izgleda, nekaj povedo o podatku samem. V HTML so oznake fiksno določene s standardom, v XML uporabnik sam določi oznake, katerih sintaksa in pomen

sta specifična za posamezne dokumente. Podatke v XML-ju lahko primerjamo tudi s podatki v klasičnih bazah podatkov. Tudi tam metapodatki ne povedo, kaj naj z njimi naredimo, ampak kaj pomenijo [7].

2.2 Specifikacija XML

XML je enostaven standardni jezik, ki ga lahko bereta človek in računalnik ter vsakemu omogoča, da naredi nov jezik, ki služi specifičnim potrebam. Za opis informacije uporablja XML t.i. elemente. Vsak element je opremljen z začetno (start tag) in končno oznako (end tag), ki kot oklepaj oklepa njegovo vsebino, razen pri t.i. praznih elementih (empty elements), ki nimajo vsebine. Element ima lahko tudi atribute, ki ga natančneje določajo.

Postaviti moramo temeljna pravila (slovar) novega jezika, ki ga določimo z DTD-jem (Document Type Definition). DTD določa kateri elementi so lahko vključeni v dokument, katere atribute elementi lahko imajo ter vrstni red in gnezdenje elementov. DTD je lahko v isti ali v ločeni datoteki. Po definiciji je nek podatkovni objekt XML dokument, če je zapisan pravilno (well-formed), torej če se podreja pravilom za pisanje oznak. Lahko pa je tudi veljaven (valid), kar pomeni, da upošteva slovnico jezika XML-dokumenta (ki jo določimo v DTD). DTD za dokumente XML ni obvezen (v SGML-ju je obvezen). Če želimo samo objaviti dokument XML na svetovnem spletu, DTD-ja ne potrebujemo. Če pa naj bi se dokumenti XML procesirali, je programiranje mnogo lažje in učinkovitejše z DTD-jem. Prav tako je DTD smiseln tedaj, ko na dokumentih dela več ljudi.

XML shema definira najmočnejše sredstvo s katerim se določita struktura in omejitve XML dokumenta. XML sheme so za razliko od DTD XML dokumenti, ki imajo celo svoje DTD. Sheme nam zagotovijo množico osnovnih tipov. Tipi vključujejo večino programskih tipov kot integer, byte, string itd. omogočajo tudi omejitev dolžine in vrednosti vsakega elementa, kar pri DTD ni omogočeno [3].

2.3 XML povpraševanja

Za delo z XML danes uporabljajo tehnologije XPath ter XQuery.

2.3.1 XPath

XPath je rešitev, s pomočjo katere lahko najdemo podatke v XML dokumentu. Uporablja izraze za izbor vozlišča ali množice vozlišč v XML dokumentu. XPath izraz opisuje lokacijo elementa ali atributa v našem dokumentu XML. Z začetkom v korenskem elementu lahko izberete kateri koli element v dokumentu, ki skrbno ustvarja verigo otroških elementov. Vsak element je ločen s poševnico "/". XPath je bil razvit, da operira z enim dokumentom, ki izgleda kot drevesna struktura z vozlišči in vrednostmi, ki jih vrne XPath. Izrazi so običajno vozlišča. XPath ima več tipov vozlišč: element, text, koren, imenski prostor, komentar, atribut [27].

Spodnji primer nam bo izbral vse "title"elemente, ki so elementi pod "book"elementi ter so elementi pod "bookstore"elementi, ki imajo vrednost elementa "price"večjo od 30.

```
doc("books.xml")/bookstore/book[price>30]/title
```

Slika 2.1: Primer XPath izraza

2.3.2 XQuery

XQuery je jezik za poizvedovanje v XML dokumentu. Omogoča nam, da najdemo XML podatke, reorganizacijo ali preoblikovanje XML dokumenta v takšno strukturo, ki nam ustreza. Podoben je SQL-u v relacijskih bazah. Zgrajen je na osnovi XPath izrazov in ga podpirajo vse XML baze. Podpira običajno obdelavo več dokumentov (ne le enega kot pri XPath) in omogoča nam ne samo sprehajanja po dokumentu, ampak tudi izluščenje in

konstruiranje novega dokumenta. Zato je XQuery bolj uporaben za potrebe poizvedovanja v podatkovnih bazah [26].

Možnosti, ki jih omogoča XQuery:

- izbira informacije glede na posebni kriterij
- filtriranje nepotrebne informacije
- iskanje informacije v dokumentu ali v množici dokumentov
- združevanje podatkov iz več XML dokumentov ali zbirk dokumentov
- sortiranje, grupiranje in agregacija podatkov
- transformacija in prestrukturiranje XMl podatkov v drugo strukturo

XQuery izrazi spadajo med sedem različnih vrst: izrazi za pot, konstruktor za element, FLWOR izrazi, izrazi, ki vključujejo operatorje ter funkcije, pogojni izrazi ter izrazi, ki preverjajo oz. testirajo tip podatkov [25].

Uporaba XQuery:

- izluščenje informacije iz relacijske baze za uporabo v spletu
- generiranje poročil glede na podatke, ki so shranjeni v bazi za predstavo na WEB kot XHTML dokument
- preizkovanje tekstovnih dokumentov v "native"XML bazah
- pridobivanje podatkov iz baze in transformacija teh podatkov za uporabo aplikacije

  Spodnji primer nam bo izbral vse "title"elemente, ki so elementi pod "book"elementi ter so elementi pod "bookstore"elementi, ki imajo vrednost elementa "price"večjo od 30, ter uredil v padajočem vrstnem redu glede na element "title"

```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
order by $x/title
return $x/title
```

Slika 2.2: Primer XQuery izraza

2.4 Vrste XML dokumentov

Eden od najpomembnejših kriterijev pri izbiri podatkovne baze za XML dokumente je določitev tipa XML dokumentov. Ali se XML uporablja samo zato, da posreduje podatke med bazo in aplikacijo, ali je to sestavljen del XHTML. Obstajata dve vrsti XML dokumentov: *podatkovno naravnani XML dokumenti* in *dokumentno naravnani XML dokumenti*. Pri izboru podatkovne baze običajno velja pravilo, da če so dokumenti podatkovno naravnanega tipa, potem se uporablja tradicionalna podatkovna baza kot je relacijska in je dodan še en nivo v bazo. Kasneje bomo imenovali take baze XML enabled. Dokumentno naravnane XML dokumente pa običajno shranimo v Native XML database. Ta pravila niso dokončna, ampak običajno veljajo [4].

2.4.1 Podatkovno naravnani dokumenti XML

Podatkovno naravnani dokument XML je takrat, ko se XML dokumenti uporabljajo za sporočanje podatkov med podjetji ali aplikacijami, in dejstvo, da se XML uporablja kot skupni format je preprosto stvar udobja zaradi združljivosti. Običajno te dokumente, ki vsebujejo elemente ter atribute teh elemntov, ustvari računalnik.

Glavne lastnosti podatkovno naravnanih XMl dokumentov:

- enostavna in redna struktura(regular structure)
- vsa vsebina je opisana vsaj z enim XML elementom ali atributom, nič ne ostane neopisano
- vsebina je mogoče malenkost pomešana ali sploh ne

- vrstni red elementov navadno ni točno določen

Primeri podatkovno naravnanih XML dokumentov so *naročila, letalski vozni red, znanstveni podatki* [4].

```
<Memo>
    <Meeting date="23/09/2005" time="10:30AM">Finance Committee</Meeting>
    <Purpose>Discuss 2006 Budget</Purpose>
    <Location>Room 923</Location>
</Memo>
```

Slika 2.3: Primer podatkovno naravnanega dokumenta

2.4.2 Dokumentno naravnani dokumenti XML

Dokumentno naravnani dokument XML je običajno narejen, tako da ga človek lahko razume. Izgleda kot tekst, besedilo v katerega je vključen XML. XML se v takem dokumentu navdno uporabllja zato, da pojasni ključno vsebino le enega dela dokumenta. Nekateri deli dokumenta niso opisani z elementi ali atributi. Običajno se ti dokumenti pišejo na roko, namenjeni so bolj pregledu, ne pa računalniški obdelavi podatkov ter ne izvirajo iz podatkovnih baz.

Glavne lastnosti dokumentno naravnanih XML dokumentov:

- ne potrebujejo redne strukture dokumenta
- vsak najmanjši neodvisni podatek je lahko še en dokument
- vsebina dokumenta je mešana
- vrstni red elementov je običajno točno določen

Primeri dokumentno naravnanih XML dokumentov so *knjige, email, oglasi in skoraj vsi, ročno napisani XHTML* [4].

```
<Product>

   <Intro>
   The <ProductName>Turkey Wrench</ProductName> from <Developer>Full
   Fabrication Labs, Inc.</Developer>
   </Intro>

   <Description>

   <Para>The turkey wrench, which comes in <i>both right- and left-
   handed versions (skyhook optional)</i>, is made of the <b>finest
   stainless steel</b>. The Readi-grip rubberized handle quickly adapts
   to your hands, even in the greasiest situations. Adjustment is
   possible through a variety of custom dials.</Para>

   <Para>The turkey wrench costs <b>just $19.99</b> and, if you
   order now, comes with a <b>hand-crafted shrimp hammer</b> as a
   bonus gift.</Para>

   </Description>

</Product>
```

Slika 2.4: Primer dokumentno naravnanega dokumenta

2.5 Zakaj uporabljamo XML

Uporaba XML je postala aktualna šele v zadnjem času, ker nam šele zdaj tehnologija omogoča izmenjavo XML dokumentov, ker sta moč procesorja ter spomin veliko cenejša, kot je to bilo leta 1980, ter seveda razvoj interneta, ki je omogočil izmenjavo XML dokumentov.

XML nam omogoča lažje razumevanje, napisanega (tako za človeka kot za stroj), pošiljanje med aplikacijami in prenosljivost podatkov ne glede na izbrano okolje. XML omogoča hitrejšo brskanje po internetu, ker je na primer zaradi oznak kot product, cena producta itd hitrejše poizvedovanje v trgovinah na spletu. V poslovnem svetu ima XML pomembno vlogo za izmenjavo dokumentov med partnerji.

XML je tudi zelo dobra alternativa v primerjavi z EDI (Electronic Document Interchange), ki je zahteval posebno in drago programsko opremo. Tako se je pojavila velika možnost za manjša podjetja,saj je bilo pošiljanje dokumentov v XML lažje in cenejše.

Nekaj razlogov, zakaj shraniti XML v bazo:

1. Če smo se v aplikaciji že odločili za uporabo XML, potem je najbolje, da tudi shranjujemo v bazo v XML formatu, da se izognemo dodatnim obremenitvam, ki so potrebne za obnovo XML dokumenta, če bo aplikacija potrebovala branje tega dokumenta.

2. Če se podatki spreminjajo zelo pogosto, je to lahko problem pri relacijskih podatkovnih bazah. Ker, če imamo recimo povezavo 1:1, ki se spremeni v 1:m, potem je potrebno najprej normalizirati podatke, nato pa še prilagoditi vse poizvedbe, ki vsebujejo pogoj za ta element, kjer se je spremenila povezava. Če pa imamo XMl dokument shranjen v XML stolpec, potem ne potrebujemo nobenih sprememb zaradi spremembe relacije 1:1 v 1:m.

3. Dokumenti v poslovnem svetu so danes zelo zapleteni. Kompleksnost, spremenljivost podatkov ter število neobveznih podatkov lahko privede pri oblikovanju relacijske sheme prevedejo do zelo zahtevnih in obsežni količine tabel. V takih primerih je shranjevanje poslovnih dokumentov v istem tipu pogosto poenostavi zasnovo podatkov ter razvijanje aplikacije.

4. Če nadaljujemo prejšnjo misel, se izkaže, da če kompleksne poslovne objekte preslikamo ter shranimo v kompleksne ter normalizirane podatkovne baze, le ti lahko zahtevajo zapleteno združevanje SQL poizvedb, kar je časovno zelo potratno. Glede na to, da je vstavljenje dokumenta v XML baze hitrejše kot v relacijske baze [30].

Zaradi teh razlogov se vedno več podatkov obdeluje in prenaša v obliki XML in je nastala potreba po shranjevanju dokumentov v podatkovnih bazah. Shranjevanje v podatkovno bazo ni edino, kar potrebujemo, potrebovali bi tudi poizvedovalne jezike, s katerimi bi se lahko sprehajali, izločali in spreminjali podatke v XML, zato sta se pojavili tudi tehnologiji XQuery ter XPath.

Poglavje 3

XML podatkovne baze

Baza podatkov XML je sistem podatkov programske opreme, ki omogoča shranjevanje podatkov v formatu XML. Ti podatki se nato lahko poizvedujejo, izvozijo in serializirajo v željeno obliko. Baze XML so običajno povezane z dokumentno usmerjenimi podatkovnimi bazami.

3.1 Vrste XML podatkovnih baz

3.1.1 XML-omogočene podatkovne baze

XML omogočene podatkovne baze (*ang. XML enabled database*) so se prvotno uporabljale, ko smo uporabljali XML kot format za izmenjavo podatkov med podatkovnimi bazami in aplikacijami. Glavna karakteristika XML omogočenih podatkovnih baz je da uporablja za shranjevanje drugačen model od XML, ki je običajno relacijski podatkovni model. V XML omogočenih podatkovnih bazah se XML dokument uporablja za prenos podatkov v podatkovno bazo, in sicer za vhod ter izhod. Po tistem, ko so podatki vnešeni v podatkovno bazo, XML dokument ni več zanimiv za bazo. Nato baza uporablja načine poizvedovanja in jezike, ki so za bazo najbolj optimalni. Ko rabimo, da je rezultat naše poizvedbe XML dokument, se zgradi na "on-the-fly"način iz rezultata poizvedbe. Ni nobene garancije, da dokument, ki je bil prvotno dan v bazo, ne bo spremenjen. Uporaba XML-omogočenih baz je

smiselna, ko uvažamo podatke iz XML dokumentov v obstoječo podatkovno bazo.

Proces pridobivanja podatkov iz baze podatkov in gradnje dokumenta XML je znan kot založništvo (ang. publishing/composition). Obraten proces (pridobivanje podatkov iz XML dokumenta in shranjevanje v podatkovni bazi) je znan kot razpad (ang. shredding/decomposition). Ne glede na to, ali gre za sestavo ali razgradnjo, se moramo zavedati, da XML-omogočena baza ne vsebuje XML dokumenta, XML dokument je zunanji vir podatkov [31]. Ko uporabljamo XML omogočeno podatkovno bazo, je zelo pomembno, da preslikamo podatkovno shemo v XML shemo. Obstajata dva načina:

1. *Tabelsko zasnovano preslikovanje (Table-based mapping)* Ko uporabimo ta način preslikovanja, je pomembno, da XML dokument ima isto strukturo kot v relacijski bazi. To pomeni, da so podatki grupirani v vrstice ter vrstice grupirane v tabele.

```
<Database>
    <SalesOrders>
        <SalesOrder>
            <Number>123</Number>
            <OrderDate>2003-07-28</OrderDate>
            <CustomerNumber>456</CustomerNumber>
        </SalesOrder>
    </SalesOrders>
    <Items>
        <Item>
            <Number>1</Number>
            <PartNumber>XY-47</PartNumber>
            <Quantity>14</Quantity>
            <Price>16.80</Price>
            <OrderNo>123</OrderNo>
        </Item>
        <Item>
            <Number>2</Number>
            <PartNumber>B-987</PartNumber>
            <Quantity>6</Quantity>
            <Price>2.34</Price>
            <OrderNo>123</OrderNo>
        </Item>
    </Items>
</Database>
```

Slika 3.1: Primer tabelsko zasnovanega preslikovanja

2. *Objektno-relacijsko preslikovanje (Object relational mapping)* Pri tem preslikovanju je XML dokument kot množica objektov. Objekti se preslikajo v tabele, lastnosti v stolpce, relacije pa se preslikajo v primarni/tuj ključ.

```
<Database>
    <SalesOrder>
        <Number>123</Number>
        <OrderDate>2003-07-28</OrderDate>
        <CustomerNumber>456</CustomerNumber>
        <Item>
            <Number>1</Number>
            <PartNumber>XY-47</PartNumber>
            <Quantity>14</Quantity>
            <Price>16.80</Price>
        </Item>
        <Item>
            <Number>2</Number>
            <PartNumber>B-987</PartNumber>
            <Quantity>6</Quantity>
            <Price>2.34</Price>
        </Item>
    </SalesOrder>
</Database>
```

Slika 3.2: Primer objektno relacijskega preslikovanja

3.1.2 Izvorne XML podatkovne baze

Lastnost izvornih XML podatkovnih baz (*ang. Native XML database*) je, da se podatkovni model nanaša na XML; XML dokument se smatra kot glavna enota za shranjevanje. Pri tem gredo noter XML dokumenti in na izhodu dobimo tudi XMl dokumente. Pri tem tipu podatkovnih baz je XML viden bazi, ne pa tako kot pri XML-omogočenih bazah. Pozitivna stran izvornih podatkovnih baz je, da lahko shranimo cel dokument, ne glede na to, ali imamo definirano XML shemo ali ne [31].

Izvorne XML podatkovne baze omogočajo pojem zbirke(ang. Collection). Dokumente, podobne med sabo, lahko shranimo v zbirke, podobno kot pri relacijski bazi pojem tabela.

Za poizvedovanje se običajno uporabi poizvedovalni jezik XQuery. Običajno izvorni podatkovni bazi shranjujejo dokumentno usmerjene XML dokumente. Te baze se uporabljajo pri upravljanju z dokumenti in iskanju dokumentov. Izvorne podatkovne baze so uporabne tudi pri upravljanju s polstrukturiranimi podatki (ang. semi-structed data). To so podatki, ki nimajo regularne strukture, ki se lahko pogosto spreminja, pri preslikovanju v relacijsko bazo to lahko privede do velike količine null vrednosti (izguba prostora) ali pa velike količine tabel.

Še en razlog, zakaj bi shranjevali v izvorne podatkovne baze, je, da so včasih poizvedbe hitrejše. Na primer imamo sales order document. Če ga shranimo v relacijsko bazo, se bodo naredile 4 tabele, v izvorno podatkovno bazo pa lahko to shranimo v enem dokumentu, kar pomeni, da potrebuje samo eno indeksno iskanje, namesto 4 kot pri relacijski podatkovni bazi. Ampak to velja samo, če iščemo podatke v takem zaporedju, kot je zapisano na disku. V nasprotnem primeru bo hitrost slabša. Na primer: če želimo različne poglede, kot je seznam strank ter vsa naročila za posamezno stranko, bo bolj počasna poizvedba, kot če bi jo naredili v relacijskih. Podatkovne baze so še bolj primerne za shranjevanje neurejenih, globoko hierarhičnih, rekurzivnih podatkov [31].

Slabost izvornih XML podatkovnih baz je, da vrača rezultat samo v XML

formatu. Če na primer naša aplikacija potrebuje podatke v drugem formatu, potem mora aplikacija razčleniti XML preden sploh bo lahko uporabila podatke.

Poglavje 4

Primerjava XML podatkovnih baz

Za XML podatkovne baze je na trgu predstavljenih veliko različnih rešitev. Ob razvoju XML jezika sta se v podatkovnih bazah razvili dve smeri. Ena smer je podpora XML direktno v relacijske podatkovne baze; to so XML omogočene podatkovne baze. Druga smer pa je popolnomo nov razvoj na osnovi XML; to so izvorni XML podatkovne baze.

Predstavniki XML omogočenih baz so : *DB2, MySQL, MonetDB/SQL, Oracle, SQL Server* [33].

Predstavniki XML izvornih baz so: *BaseX, Berkeley DB XML, eXist, MonetDB/XQuery, Sedna XML DBMS* [33].

V diplomski nalogi smo želeli testirati obe vrsti XML podatkovnih baz. Zato smo izbrali tri podatkovne baze in sicer BaseX, eXist ter MS SQL SERVER 2008 R2 EXPRESS. Predstavnici izvornih podatkovnih baz sta BaseX ter eXist. Izbrali smo ju zato, ker trenutno se prav za ti dve bazi najde največ dokumentacije in sta tudi največkrat omenjeni. Za XML omogočeno podatkovno bazo smo izbrali MS SQL SERVER 2008 R2 EXPRESS, ker poznamo jo najbolje poznamo, saj imamo z njo največ izkušenj.

4.1 BaseX

BaseX podatkovna baza je specializirana za poizvedovanje, shranjevanje ter vizualizacijo velikih XMl dokumentov ali zbirk dokumentov. Omogoča poizvedovanje v XQuery poizvedovalnem jeziku [28].

4.1.1 Vrsta podatkovnih baz

Je izvorna podatkovna baza definira logični model za XML dokument. Model v BaseX sledi drevesni strukturi XML vozlišč. Podatke shranjuje in išče glede na ta model. BaseX ne potrebuje DTD ali XML sheme za kodiranje dokumenta. Omogoča tudi indeksiranje za vrednosti atributov ter elementov. Ti indeksi so uporabni pri poizvedbah, ki se nanašejo na tekst.

4.1.2 Za katere tipe dokumentov je najbolj primerna

Ta baza je najbolj primerna za dokumentno orientirane XMl dokumente.

4.1.3 Vmesnik API

Podpira Javo, C#, PHP, Python. Podpira formati : XML, XHTML, JSON, CSV, tekst.

4.1.4 Podpora uporabnikom

GUI uporabniku omogoča interaktivno iskanje, raziskovanje ter analiziranje podatkov ter oceno časa izvajanja XQuery poizvedb. Pri GUI je več različnih načinov vizualizacije podatkov :

- **Drevesni pogled** : Vsak element ima lahko svoje podelemente. To je običajno prikazano z jamico na seznamu. Vsak element se da razširiti, da se prikažejo podelementi.
- **Pogled tabel**: Prikaže vsa vozlišča v tabeli z vrsticami in stolpci.

- **Drevesna struktura** (ang. treemap) : Prikaže hierarhične podatke kot niz vgnezdenih pravokotnikov. Vsaka veja je podana kot pravokotnik, ki je nato razdeljen na manjše pravokotnike. Zaradi take zgradbe je zelo priročen pogled na tisoč predmetov na enem zaslonu. Na razpolago je dobra dokumentacija, najde se veliko informacij o izdelku.

4.1.5 Licenca

Je odprtokodna podatkovna baza

4.2 eXist

4.2.1 Vrsta podatkovne baze

Izvorna XML podatkovna baza je web-service orientirana. Ne potrebuje DTD ali XMl sheme. Omogoča indeksiranje podatkov. Maksimalna velikost enega XM dokumenta je samo 100 MB [24].

4.2.2 Za katere tipe dokumentov je najbolj primerna

Najbbolj je primerna za dokumentno orientirane XML dokumente.

4.2.3 Vmesnik API

Podpira Javo

4.2.4 Podpora uporabnikom

Dokumentacija je nepopolna ali pa ne daje pravih nasvetov. Zelo težko je najti primere uporabe, tako da je uporabniku potrebno kar veliko časa za uvajanje. Nima podprtega avtomatičnega indeksiranje, napisati moraš konfiguracijo za indekse, ki so potrebni.

4.2.5 Licenca

Je odptokodna podatkovna baza

4.3 MS SQL SERVER 2008 R2 EXPRESS

4.3.1 Vrsta podatkovne baze

Je XML omogočena podatkovna baza

4.3.2 Za katere tipe dokumentov je najbolj primerna

Najbolj primerno za podatkovno orientirane XML-dokumente.

4.3.3 Podpora uporabnikam

Je zelo dobra dokumentirana. Ima veliko primerov ter dokumentacije.

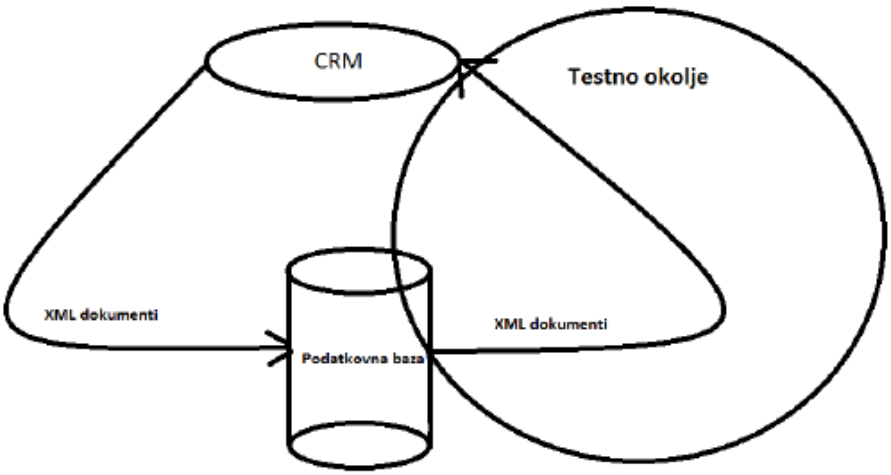
4.3.4 Licenca

Gre za odprtokodno podatkovno bazo.

Poglavje 5

Problemska domena

Imamo podjetje, ki ima lastni CRM. CRM je lastni sistem za upravljanje odnosov s strankami, ki pomaga podjetju poslovati na čim bolj učinkovit način. Podjetje ima XML dokumente, ki vsebujejo podatke o strankah, nakupih, zaposlenih ter dokumente, ki so potrebni za poslovanje. Za njih je pomembno, da pošiljajo in pridobivajo XML dokumente od drugih podružnic. Pomembno je tudi združevanje podatkov iz organizacije ter izven organizacije, da lahko podajo celostno sliko. Pri CRM je zelo pomembna hitrost pridobivanja podatkov iz podatkovne baze. Količina vhodnih podatkov v bazo, kot na primer za dodajanje novih naročil ali posodabljenje kakšnih drugih podatkov, je zelo majhna. Pri CRM so bolj pomembni izhodni podatki oz. hitrost pridobivanja podatkov iz baze. Na primer prodajalec želi na svojem zaslonu čim prej videti informacijo o kupcu, kaj in koliko je kupil, da je lahko čim bolj učinkovit pri opravljanju svojega dela. Zato bo pri testiranju baz zelo pomembno, katera baza nam najbolj hitro vrne izhodne podatke. Tako se bomo najbolj osredotočili na hitrost rezultatov select poizvedb, ki vrnejo XML dokument, in nato glede na rezultate izbrali najbolj primerno bazo za podjetje.



Slika 5.1: Skica problemske domene

Tako smo izbrali 3 scenarije v katerih bomo stestirali 3 baze :

- **BaseX**
- **eXist**
- **MS SQL SERVER 2008 R2 EXPRESS**

Za vsak scenarij bomo izvedli enostaven select, select s pogojem, select, ki vsebuje agregatne funkcije, ter select, ki bo vrnil top rezultate ter pogledal hitrost insert ter update stavkov.

1. Scenarij
   V prvem scenariju bomo testirali stavke, ki bi bili potrebne za pomoč pri analizi podatkov za podjetje. To je količina vseh prodanih izdelkov po letih, vrste najbolj prodanih izdelkov, maksimalna količina izdelkov, prodanih po regijah.

2. Scenarij
   V drugem scenariju se bomo osredotočili na stavke, ki služijo za pomoč prodajalcu. Koliko izdelkov je kupila vsaka stranka, naj vrne top 10

strank glede na zapravljeni denar, koliko kupcev ima vsaka poslovalnica glede na državo, koliko imamo strank, bodisi gre za individualne ali pa za trgovino. Preizkusili bomo tudi insert stavek, da dodamo novo naročilo.

3. Scenarij
   Pomoč vodstvu pri pregledu delovanja zaposlenih, koliko naročil je kdo sprejel, koliko prodal, s katerimi strankami se je pogovarjal, št. kontaktov pri vsakem prodajalcu ter update stavek za posamezen podatek pri zaposlenem.

Poglavje 6

Testiranje

6.1 Testno okolje

Pri testiranju smo uporabili Windows 7 operacijski sistem. Bazo smo namestili na sistem s procesorjem Intel Core 2 Duo 2GHz in 4GB glavnega pomnilnika.

6.2 Testni podatki

Da bi lahko stestirali naše scenarije, smo morali poiskati XML dokumente, ki bi bili ustrezni naši problemski domeni. Na internetu nismo našli popolnoma ustreznih podatkov za CRM smo pa našli podobno bazo AdventureWorks, ki je kot primer pri bazi MS SQL SERVER R2 2008 EXPRESS. Ta baza vsebuje podatke o fiktivnem podjetju, ki je velika multinacionalka in se ukvarja s prodajo in proizvodnjo koles. Baza vsebuje podatke o strankah, produktih, zaposlenih, podružnicah itd. Bazo smo malo preuredili, nekaterih tabel nismo uporabili in nastala je nova podatkovna baza XmlDataTest, ki smo jo nato uporabili pri testiranju zato, da bo čim bolj prilagojena naši problemski domeni CRM. Programsko smo s pomočjo SQL podatke pretvorili v XML dokumente, ki so kasneje bili dodani v bazo BaseX ter eXist. Imeli smo dva tipa XML dokumentov, eni so bili iz ene tabele, drugi tip pa je bil sestavljen

iz dveh tabel, tako da je SalesOrder vseboval tudi SalesOrderDetail. Zaradi tega, ker so v CRM bolj pomembni izhodni podatki oz. hitrost pridobivanja izhodnih podatkov, se nismo osredotočili na vhodne podatke. Zato v bazo MS SQL SERVER R2 2008 EXPRESS nismo spravili XML dokumentov, ampak pustili XmlDataTest bazo, ker smo predvidevali, da če bi dodali XML dokumente, bi se baza spet pretvorila v tabele in nastala bi podobna baza kot XmlDataTest. V ostali dve bazi pa smo dodali XML dokumente, ki so bili pridobljeni iz XmlDataTest. Velikost baze je bila 176 MB in je vsebovala 18 dokumentov.

6.3 Implementacija

Poizvedbe smo poganjali v GUI, ki je bil priložen v bazi. Na začetku smo naredili več različnih poizvedb v XQuery ter SQL za vsak scenarij, da bi našli najboljše primere, kjer bi se prikazale razlike. Nato smo glede na poizvedbe, ki so se nam zdele najbolj zanimive, stestirali 10-krat ter glede na podatke izračunali povprečni čas, ki ga potrebuje vsaka baza za izvedbo te poizvedbe.

6.3.1 BaseX

Xml dokumente, ki so bili pridobljeni iz XmlDataTest, smo shranjevali v lokalno podatkovno bazo BaseX, kjer smo tudi izvajali poizvedbe v GUI, ki je bil zraven podatkovne baze. Testirali smo vse 3 scenarije. Dokumenti so vsebovali full-text indekse ter indekse po atributih ter po elementih. Poizvedbe so bile napisane v XQuery jeziku.

6.3.2 eXist

Xml dokumente, ki so bili pridobljeni iz XmlDataTest, smo shranjevali v podatkovni bazi eXist. Testirali smo vse tri scenarije. Dokumentom smo priredili indekse najbolj pogosto uporabljenih elementov v poizvedbah. Poizvedbe so bili napisane v XQuery jeziku.

6.3.3 MS SQL SERVER 2008 R2 EXPRESS

Nismo shranjevali XMl dokumentov, ampak uporabili kar XmlDataTest, ker mislimo, da bi pri pretvorbi v tabele iz XMl dokumentov nastala ista baza. Poizvedbe smo izvajali v SQL jeziku, ker če uporabljamo bazo, jo moram uporabiti na najbolj optimiziran način, da dobimo čim hitrejše podatke. Ohranili smo indekse, ki so bili že podani v testni bazi AdventureWorks.

Poglavje 7

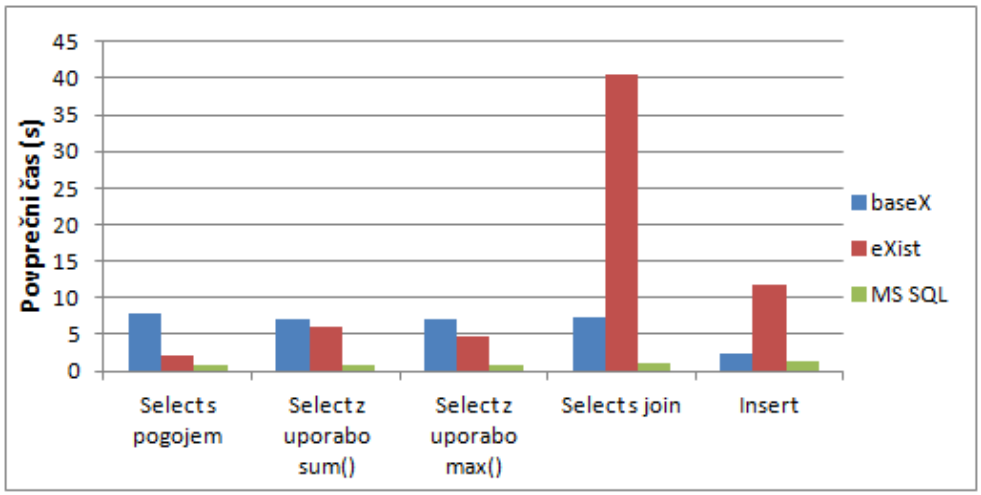
Analiza rezultatov

Poizvedbe, ki so bile uporabljenie, so v prilogi.

7.1 Scenarij 1

	baseX (s)	eXist(s)	MS SQL(s)
1.Select s pogojem	7.731	2.087	0.779
2.Select z uporabo sum	7.167	6.035	0.769
3.Select z uporabo max	7.157	4.761	0.761
4.Select s join	7.201	40.546	1.186
5.Insert	2.433	11.651	1.196

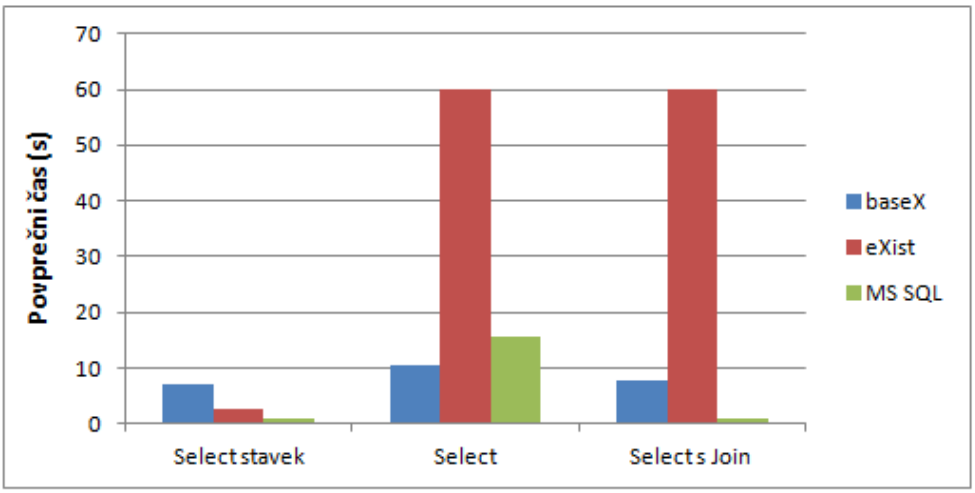
Slika 7.1: Povprečni čas izvajanja prvega scenarija



Slika 7.2: Povprečni čas izvajanja prvega scenarija

V zgornji tabeli rezultatov lahko vidimo, da je z zeleno barvo označen najboljši čas poizvedbe, z rdečo barvo pa najslabši čas poizvedbe V prvem scenariju smo izvajali select stavek s pogojem, select stavke s sum, count, max funkcijo. Testirali smo tudi uporabo join med dokumenti. Izkazalo se je, da je bil eXist najbolj počasen, ko je bilo potrebno združiti dva dokumenta. Pri vseh poizvedbah je bila najhitrejša MS SQL SERVER 2008 R2 EXPRESS podatkovna baza, običajno skoraj za 10 krat. Če pa primerjam BaseX ter eXist podatkovni bazi, je eXist skoraj 2x hitrejši od BaseX, ko gre za select stavek brez operacije join. Če imamo join operacijo (združevanje dveh dokumentov), je eXist zelo počasen. Opazili smo tudi, da če je zelo veliko podatkov, po katerih je treba poizvedovati, potem je baza eXist še počasnejša od BaseX. Operacija insert je bila najbolj počasna pri eXistu, najhitrejša pa pri MS SQL SERVER 2008 R2 EXPRESS.

7.2 Scenarij 2



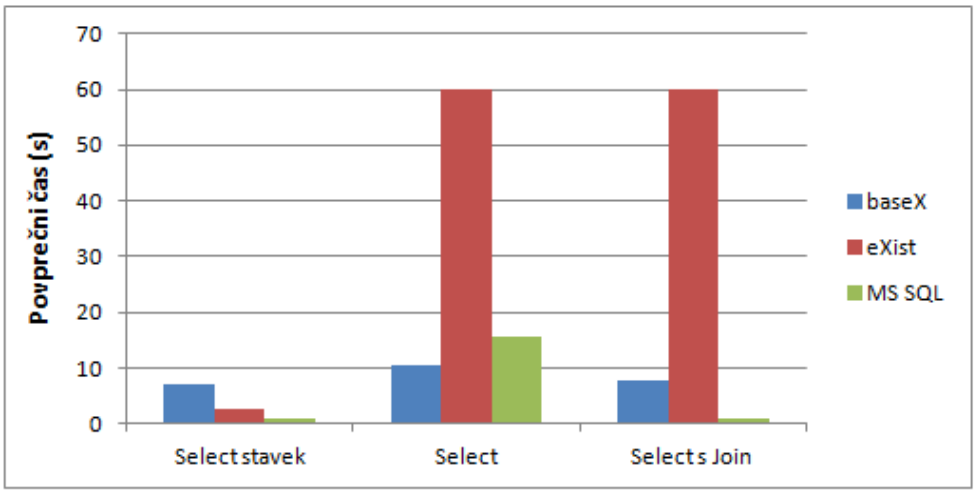
Slika 7.3: Povprečni čas izvajanja drugega scenarija

	baseX (s)	eXist (s)	MS SQL (s)
1.Select stavek	7.118	2.582	0.789
2.Select	9.471	Več kot ena minuta	15.561
3.Select s Join	7.690	Več kot ena minuta	0.804

Slika 7.4: Povprečni čas izvajanja drugega scenarija

V zgornji tabeli rezultatov lahko vidimo, da je z zeleno barvo označen najboljši čas poizvedbe, z rdečo pa označen najslabši čas poizvedbe. Testirali smo spet enostaven select, select s pogojem in join ter operacijo join. Pri tem testiranju je MS SQL SERVER 2008 R2 EXPRESS pokazal slabši rezultat kot BaseX pri poizvedbi, kjer smo imeli na izhodu veliko podatkov, ki so se nato morali pretvoriti v XML dokument. To pomeni, da če imamo na izhodu veliko podatkov, ki jih je treba pretvoriti v XML dokument, potem je MS SQL SERVER 2008 R2 EXPRESS počasnejši od XML podatkovnih baz. Ostali časi so spet kot pri prvem scenariju; če so enostavni stavki je eXist skoraj 2x hitrejši od BaseX, je pri join operacijah BaseX hitrejši od eXist.

7.3 Scenarij 3



Slika 7.5: Povprečni čas izvajanja tretjega scenarija

	baseX (s)	eXist (s)	MS SQL (s)
1.Select s pogojem	7.260	1.867	0.759
2.Select	7.125	3.748	0.680
3.Update	2.044	1.254	0.199

Slika 7.6: Povprečni čas izvajanja tretjega scenarija

V zgornji tabeli rezultatov lahko vidimo, da je z zeleno barvo označen najboljši čas poizvedbe z rdečo pa barvo označen najslabši čas poizvedbe. Select poizvedbe so pri MS SQL SERVER 2008 R2 EXPRESS spet najbolj hitre. Update stavek je hitrejši pri MS SQL SERVER 2008 R2 EXPRESS podatkovni bazi. Najslabši čas je pa pri BaseX.

7.4 Zaključek

V prvem scenariju, kje smo morali vračati kratke rezultate podatkov, je bil ne glede na vrsto select stavkov najhitrejši MS SQL SERVER 2008 R2

EXPRESS.

Pri drugem scenariju smo opazili razliko in prednost XML podatkovne baze v tem, da če imamo na izhodu več rezultatov, ki jih potem moramo pretvoriti v XML dokumente, je XML podatkovna baza za ta namen bolj uporabna.

Pri tretjem scenariju smo pa opazili, da update stavek najhitrejše opravi MS SQL SERVER 2008 R2 EXPRESS podatkovna baza.

Glede na to, da imamo pri CRM po navadi podatke, ki so združeni iz več tabel, kot recimo stranka ter naročilo, je pri združevanju dokumentov (pri operaciji join) zelo pomembna hitrost; pri našem testiranju se je najbolj obnesla baza MS SQL SERVER 2008 R2 EXPRESS. Slabša je bila edino od BaseX, ko je bila na izhodu velika količina izhodnih podatkov, ki jih je bilo treba zapisati v XML obliki. Če pa se vseeno odločimo za XML podatkovno bazo, potem se moramo odločiti za eXist, če imamo poizvedbe samo v enem dokumentu. Če moramo dobiti podatke iz več dokumentov potem je najbolje uporabiti BaseX podatkovno bazo. Najboljša rešitev za naše podjetje je, če uporabi MS SQL SERVER 2008 R2 EXPRESS .

Poglavje 8

Sklepne ugotovitve

Pri izbiri vrste podatkovne baze moramo biti pozorni na kar nekaj stvari. Če imamo aplikacijo, ki uporablja XML samo za prenos podatkov, ima strogo definirano zgradbo, nas bolj zanimajo podatki, ne pa kako so shranjeni v XML dokumentu, potem je najbolje uporabiti relacijsko podatkovno bazo.

Če pa imamo spletno stran, ki je zgrajena večinoma iz dokumentno orientiranih XML dokumentov, ter želimo zagotoviti uporabnikom iskanje po vsebini ter iskanje dokumentov, naši dokumenti imajo manj definirano zgradbo, ki se lahko pogosto spreminja ter podjetje potrebuje medsebojno izmenjavo XML dokumentov , potem je bolje uporabiti izvorno XML podatkovno bazo.

Glede na tip dokumentov običajno velja, da če imamo *podatkovno usmerjene dokumente* , je bolje uporabiti relacijske baze oz. xml omogočene podatkovne baze. Če imamo *dokumentno usmerjene dokumente* velja, da je bolje uporabiti izvorne XMl podatkovne baze.

Pri primerjavi baz smo videli, da je eXist zelo počasen, če imamo join med dokumenti ali če je potrebno sestaviti različne poglede (npr. za vse kupce, prikazati njihova naročila). Tudi čas, ki je potreben, da naložimo naše dokumente v lokalno bazo je dolg, lahko traja do 10 minut, pri tem da je cela baza samo 222 MB. Kompleksne poizvedbe se izvajajo zelo dolgo časa, edino eXist je hiter pri enostavnih poizvedbah oziroma ko se sprehajamo po enemu dokumentu. Da pospešimo poizvedbe, lahko nastavimo indekse,

ampak je slabo dokumentirano, na kašen način se da to izvesti.

BaseX je podatkovna baza, ki je pokazala, da je bolj počasna od eXist tudi pri enostavnih poizvedbah, vendar je pri join operaciji, če se ta zgodi med dvema ne prav velikima dokumentoma, razlika v času skoraj neopazna. Prednost BaseX je, da nam vrne že XMl dokument in ni potrebno pretvarjati tako kot pri MS SQL SERVER 2008 R2 EXPRESS, ki je bil zelo hiter samo, če ni bilo na izhodu več podatkov, iz katerih je bilo potrebno narediti XML dokumente.

Poglavje 9

Priloga. Poizvedbe

9.1 Scenarij 1

9.1.1 XQuery poizvedbe

1. Poizvedba

```
for $r in doc("db/xmlDocuments/SalesOrder.xml")/
    salesOrders/salesOrder[OrderDate gt "2001-01-01"
     and OrderDate lt "2003-01-01"]
let $name := $r/TerritoryID
group by $name
return
 <drzava>{$name}
        <Contact>
                <stNarocil> {count($r/OrderDate)}</
                    stNarocil>
                <kolicinaIzdelkov>{count($r/
                    salesOrderDetail/OrderQty)}</
                    kolicinaIzdelkov>
        <kolicinaDenarja>{sum($r/SubTotal)}</
            kolicinaDenarja>
```

```
    </Contact>
  </drzava>
```

2. Poizvedba

```
for $r in doc("C:/Users/vlada/Desktop/xmlDocuments/
    SalesOrder.xml")/salesOrders/salesOrder
let $country := $r/TerritoryID
group by $country
return
  <Country>{$country}
   <numberOrder> { sum($r/salesOrderDetail/OrderQty
       )}</numberOrder>
  </Country>
```

3. Poizvedba

```
for $r in doc("C:/Users/vlada/Desktop/xmlDocuments/
    SalesOrder.xml")/salesOrders/salesOrder
let $name := $r/TerritoryID
group by $name
 return
 <drzava> {$name}
   <maxKolicinaIzdelka> {max( $r/salesOrderDetail/
       OrderQty )}</maxKolicinaIzdelka>
 </drzava>
```

4. Poizvedba

```
for $r in doc("C:/Users/vlada/Desktop/xmlDocuments/
    SalesOrder.xml")/salesOrders/salesOrder,
$sellPerson in doc("C:/Users/vlada/Desktop/
    xmlDocuments/SalesPersons.xml")/salesPersons/
    salesPerson[SalesPersonID=$r/SalesPersonID]
```

```
let $country := $sellPerson/TerritoryID
group by $country
return
  <Country>{$country}
   <numberOrder> {  sum($r/salesOrderDetail/
      OrderQty)}</numberOrder>
  </Country>
```

5. Poizvedba

```
let $um := ( <salesOrder>
    <SalesOrderID>105689585</SalesOrderID>
    <SalesReasonID>10</SalesReasonID>
    <RevisionNumber>5</RevisionNumber>
    <OrderDate>2002-07-01T00:00:00</OrderDate>
    <DueDate>2001-07-13T00:00:00</DueDate>
    <ShipDate>2001-07-08T00:00:00</ShipDate>
    <Status>5</Status>
    <OnlineOrderFlag>False</OnlineOrderFlag>
    <PurchaseOrderNumber>PO522145787</
       PurchaseOrderNumber>
    <AccountNumber>10-4020-000676</AccountNumber>
    <CustomerID>11000</CustomerID>
    <ContactID>378</ContactID>
    <SalesPersonID>279</SalesPersonID>
    <TerritoryID>5</TerritoryID>
    <BillToAddressID>985</BillToAddressID>
    <ShipToAddressID>985</ShipToAddressID>
    <ShipMethodID>5</ShipMethodID>
    <CreditCardID>16281</CreditCardID>
    <CreditCardApprovalCode>105041Vi84182</
       CreditCardApprovalCode>
```

```
<SalesOrderNumber>SO43659</SalesOrderNumber>
<CurrencyRateID/>
<SubTotal>24643.9362</SubTotal>
<TaxAmt>1971.5149</TaxAmt>
<Freight>616.0984</Freight>
<TotalDue>27231.5495</TotalDue>
<SalesOrderNumber/>
<rowguid>79b65321-39ca-4115-9cba-8fe0903e12e6</
    rowguid>
<ModifiedDate>2001-07-08T00:00:00</ModifiedDate
    >
<salesOrderDetail>
  <SalesOrderDetailID>10</SalesOrderDetailID>
  <CarrierTrackingNumber>4911-403C-98</
      CarrierTrackingNumber>
  <OrderQty>10</OrderQty>
  <ProductID>709</ProductID>
  <SpecialOfferID>1</SpecialOfferID>
  <UnitPrice>8.7000</UnitPrice>
  <UnitPriceDiscount>0.0000</UnitPriceDiscount>
  <LineTotal>34.200000</LineTotal>
  <rowguid>ac769034-3c2f-495c-a5a7-3b71cdb25d4e
      </rowguid>
  <ModifiedDate>2001-07-01T00:00:00</
      ModifiedDate>
</salesOrderDetail>
<salesOrderDetail>
  <SalesOrderDetailID>12</SalesOrderDetailID>
  <CarrierTrackingNumber>4911-403C-98</
      CarrierTrackingNumber>
  <OrderQty>23</OrderQty>
```

```
        <ProductID>711</ProductID>
        <SpecialOfferID>1</SpecialOfferID>
        <UnitPrice>20.1865</UnitPrice>
        <UnitPriceDiscount>0.0000</UnitPriceDiscount>
        <LineTotal>80.746000</LineTotal>
        <rowguid>0e371ee3-253e-4bb0-b813-83cf4224f972
            </rowguid>
        <ModifiedDate>2001-07-01T00:00:00</
            ModifiedDate>
      </salesOrderDetail>
    </salesOrder>)return

insert node $um into //salesOrders
```

9.1.2 SQL poizvedbe

1. Poizvedba

```
Sales.SalesTerritory.TerritoryID AS drzava,
COUNT(DISTINCT Sales.SalesOrderHeader.SalesOrderID)
    AS steviloNarocil,
COUNT(Sales.SalesOrderDetail.ProductID) AS
    steviloProduktov,
SUM(Sales.SalesOrderDetail.LineTotal) AS
    vrednostProdaje
FROM Sales.SalesOrderHeader INNER JOIN
                        Sales.SalesOrderDetail ON
                            Sales.SalesOrderHeader.
                            SalesOrderID = Sales.
                            SalesOrderDetail.
                            SalesOrderID INNER JOIN
```

```
                        Sales.SalesTerritory ON Sales
                            .SalesOrderHeader.
                            TerritoryID = Sales.
                            SalesTerritory.TerritoryID
WHERE       (Sales.SalesOrderHeader.OrderDate >=
    '1.1.2001') AND (Sales.SalesOrderHeader.
    OrderDate <= '1.1.2003')
GROUP BY Sales.SalesTerritory.TerritoryID
order by Sales.SalesTerritory.TerritoryID
FOR XML AUTO, ELEMENTS
GO
```

2. Poizvedba

```
SELECT
Sales.SalesTerritory.TerritoryID AS drzava,
SUM(Sales.SalesOrderDetail.OrderQty) AS
    steviloProduktov
FROM Sales.SalesOrderHeader INNER JOIN
                        Sales.SalesOrderDetail ON
                            Sales.SalesOrderHeader.
                            SalesOrderID = Sales.
                            SalesOrderDetail.
                            SalesOrderID INNER JOIN
                        Sales.SalesTerritory ON Sales
                            .SalesOrderHeader.
                            TerritoryID = Sales.
                            SalesTerritory.TerritoryID
GROUP BY Sales.SalesTerritory.TerritoryID
order by Sales.SalesTerritory.TerritoryID
FOR XML AUTO, ELEMENTS
GO
```

3. Poizvedba

```
SELECT
Sales.SalesTerritory.TerritoryID AS drzava,
Max(Sales.SalesOrderDetail.OrderQty) AS maxKolicina
FROM Sales.SalesOrderHeader INNER JOIN
                        Sales.SalesOrderDetail ON
                            Sales.SalesOrderHeader.
                            SalesOrderID = Sales.
                            SalesOrderDetail.
                            SalesOrderID INNER JOIN
                        Sales.SalesTerritory ON Sales
                            .SalesOrderHeader.
                            TerritoryID = Sales.
                            SalesTerritory.TerritoryID
GROUP BY Sales.SalesTerritory.TerritoryID
order by Sales.SalesTerritory.TerritoryID
FOR XML AUTO, ELEMENTS
GO
```

4. Poizvedba

```
SELECT      Sales.SalesTerritory.TerritoryID AS
    drzava, COUNT(DISTINCT Sales.SalesOrderHeader.
    SalesOrderID) AS steviloNarocil,
                        COUNT(Sales.SalesOrderDetail.
                            ProductID) AS
                            steviloProduktov, SUM(
                            Sales.SalesOrderDetail.
                            LineTotal) AS
                            vrednostProdaje,
                        max(Sales.SalesOrderHeader.
                            SalesPersonID) as
```

```
                                    salesPerson
    FROM            Sales.SalesOrderHeader INNER JOIN
                                Sales.SalesOrderDetail ON
                                    Sales.SalesOrderHeader.
                                    SalesOrderID = Sales.
                                    SalesOrderDetail.
                                    SalesOrderID INNER JOIN
                                Sales.SalesTerritory ON Sales
                                    .SalesOrderHeader.
                                    TerritoryID = Sales.
                                    SalesTerritory.TerritoryID
    GROUP BY Sales.SalesTerritory.TerritoryID, Sales.
        SalesOrderHeader.SalesPersonID
    ORDER BY drzava
    FOR XML AUTO, ELEMENTS
    GO
```

5. Poizvedba

```
    INSERT INTO [XmlDataTest].[Sales].[SalesOrderDetail
        ]
                ([SalesOrderID]
                ,[CarrierTrackingNumber]
                ,[OrderQty]
                ,[ProductID]
                ,[SpecialOfferID]
                ,[UnitPrice]
                ,[UnitPriceDiscount]
                )
            VALUES
                (43761
                ,'4911-403Z-YY'
```

```
,22
,742
,1
,835.4587
,0)
```

9.2 Scenarij 2

9.2.1 XQuery poizvedbe

1. Poizvedba

```
for $r in doc("C:/Users/vlada/Desktop/xmlDocuments/
    SalesOrder.xml")/salesOrders/salesOrder
let $name := $r/TerritoryID
group by $name
return    <drzava>{$name}
            <bu>  {count($r/CustomerID)}</bu>
          </drzava>
```

2. Poizvedba

```
for $r in doc("C:/Users/vlada/Desktop/xmlDocuments/
    SalesOrder.xml")/salesOrders/salesOrder
let $name := $r/TerritoryID
let $date := year-from-dateTime(xs:dateTime($r/
    OrderDate))
group by $name
return
  <drzava>{$name}
   <Contact>
  {for $y in $date
  group by $y
```

```
    return
            <bu> {$y} {
            for $x in $r[year-from-dateTime(xs:
                dateTime(OrderDate)) = $y ]
                    let $nameC:=$x/CustomerID
                    group by $nameC
                    return
                    <Contact>{$nameC}
                      <numberOfOrder>{number(count($x))
                          }</numberOfOrder>
                      <numberOfProducts>{number(sum($x/
                          SubTotal))}</numberOfProducts>
                    </Contact>}
            </bu>}

     </Contact>
    </drzava>
```

3. Poizvedba

```
for $r in doc("C:/Users/vlada/Desktop/xmlDocuments/
    SalesOrder.xml")/salesOrders/salesOrder,
$indCu in doc("C:/Users/vlada/Desktop/xmlDocuments/
    StoreSales.xml")/storeSales/store[CustomerID=$r/
    CustomerID ]
let $name := $r/TerritoryID
let $date := year-from-dateTime(xs:dateTime($r/
    OrderDate))
group by $name
return
  <drzava>{$name}
     <kupci>
```

```
        {for $y in $date
        group by $y
        return
         <leto>
         {$y,
              let $prviDatumi:=( for $x in $r
          let $date := $x/OrderDate
          let $cuID:= $x/CustomerID
          group by $cuID
          return
          <cuID> {(for $i in $date return $i)[
             position() le 1]} </cuID>)
          let $vsota:= count(for $x in $prviDatumi
              where year-from-dateTime(xs:dateTime($x
                 ))= $y return $x)
                          for $i in $vsota return $i}
         </leto>}
        </kupci>
    </drzava>
```

9.2.2 SQL poizvedbe

1. Poizvedba

```
SELECT
Sales.SalesTerritory.TerritoryID AS drzava,
Sales.SalesOrderHeader.CustomerID as kupec,
COUNT(DISTINCT Sales.SalesOrderHeader.SalesOrderID)
    AS steviloNarocil
FROM Sales.SalesOrderHeader INNER JOIN
      Sales.SalesTerritory ON Sales.SalesOrderHeader
         .TerritoryID = Sales.SalesTerritory.
```

```
        TerritoryID
GROUP BY Sales.SalesTerritory.TerritoryID, Sales.
    SalesOrderHeader.CustomerID
ORDER BY drzava
FOR XML AUTO, ELEMENTS
GO
```

2. Poizvedba

```
SELECT
Sales.SalesTerritory.TerritoryID AS drzava,
Sales.SalesOrderHeader.CustomerID AS kupec,
YEAR(Sales.SalesOrderHeader.OrderDate) AS leto,
COUNT(DISTINCT(Sales.SalesOrderHeader.SalesOrderID)
    ) AS steviloNarocil,
sum(Sales.SalesOrderDetail.LineTotal) AS vrednost
FROM Sales.SalesOrderHeader
      INNER JOIN Sales.SalesTerritory ON
       Sales.SalesOrderHeader.TerritoryID = Sales.
          SalesTerritory.TerritoryID
      INNER JOIN Sales.SalesOrderDetail ON
        Sales.SalesOrderHeader.SalesOrderID = Sales.
            SalesOrderDetail.SalesOrderID
       AND  Sales.SalesOrderHeader.SalesOrderID =
            Sales.SalesOrderDetail.SalesOrderID
GROUP BY Sales.SalesTerritory.TerritoryID, Sales.
    SalesOrderHeader.CustomerID ,YEAR(Sales.
    SalesOrderHeader.OrderDate)
ORDER BY drzava, YEAR(Sales.SalesOrderHeader.
    OrderDate)
FOR XML AUTO, ELEMENTS
GO
```

3. Poizvedba

```
select drzava, leto, COUNT(kupec) as
    steviloNovihKupcev
from
(
SELECT TOP 100 PERCENT
Sales.SalesTerritory.TerritoryID AS drzava,
Sales.SalesOrderHeader.CustomerID AS kupec,
YEAR(Sales.SalesOrderHeader.OrderDate) as leto,
rowid = ROW_NUMBER() OVER (PARTITION BY  Sales.
    SalesOrderHeader.CustomerID
ORDER BY Sales.SalesTerritory.TerritoryID,Sales.
    SalesOrderHeader.CustomerID, YEAR(Sales.
    SalesOrderHeader.OrderDate))
FROM  Sales.SalesOrderHeader INNER JOIN
        Sales.SalesTerritory ON Sales.
            SalesOrderHeader.TerritoryID = Sales.
            SalesTerritory.TerritoryID
        INNER JOIN
        Sales.Customer ON Sales.SalesOrderHeader.
            CustomerID = Sales.Customer.CustomerID
         AND Sales.SalesOrderHeader.CustomerID =
             Sales.Customer.CustomerID
         AND Sales.SalesOrderHeader.CustomerID =
             Sales.Customer.CustomerID
         AND Sales.SalesOrderHeader.CustomerID =
             Sales.Customer.CustomerID
WHERE      Sales.SalesOrderHeader.CustomerID in (
    select CustomerID from Sales.Store)
GROUP BY
Sales.SalesTerritory.TerritoryID,
```

```
Sales.SalesOrderHeader.CustomerID,
YEAR(Sales.SalesOrderHeader.OrderDate)

order by
Sales.SalesTerritory.TerritoryID,
Sales.SalesOrderHeader.CustomerID,
YEAR(Sales.SalesOrderHeader.OrderDate)
) IZBOR
where rowid=1
group by drzava, leto
order by drzava, leto
FOR XML AUTO, ELEMENTS
GO
```

9.3 Scenarij 3

9.3.1 XQuery poizvedbe

1. Poizvedba

```
for $r in doc("db/xmlDocuments/SalesOrder.xml")/
    salesOrders/salesOrder[year-from-dateTime(xs:
    dateTime(OrderDate))= 2002 ]
let $name := $r/TerritoryID
group by $name
return
  <drzava>{$name}
{for $x in $r
            let $nameC:=$x/SalesPersonID
            group by $nameC
           return <Contact>{$nameC} <NumberOrder>{$x
              /SalesOrderID}</NumberOrder> </Contact
```

```
            >}

    </drzava>
```

2. Poizvedba

```
for $r in doc("C:/Users/vlada/Desktop/xmlDocuments/
    SalesOrder.xml")/salesOrders/salesOrder
let $name := $r/TerritoryID
group by $name
return
  <drzava>{$name}
  {for $c in $name
  return <Contact>
           {for $x in $r
           let $nameC:=$x/SalesPersonID
           group by $nameC
          return <Contact>{$nameC} <NumberOrder>{
             count($x)}</NumberOrder> </Contact>}
      </Contact>}
  </drzava>
```

3. Poizvedba

```
for $i in //contacts
where $i/ContactID = "1"
return
replace value of node $i/EmailAddress with '
    primer@gmail.com'
```

9.3.2 XML poizvedbe

1. Poizvedba

```
SELECT      Sales.SalesOrderHeader.SalesPersonID,
    Person.Contact.FirstName, Person.Contact.
    LastName, Sales.SalesOrderHeader.SalesOrderID,
                        Sales.SalesOrderHeader.
                            OrderDate
FROM            Sales.SalesOrderHeader INNER JOIN
                        Sales.SalesPerson ON Sales.
                            SalesOrderHeader.
                            SalesPersonID = Sales.
                            SalesPerson.SalesPersonID
                            AND
                        Sales.SalesOrderHeader.
                            SalesPersonID = Sales.
                            SalesPerson.SalesPersonID
                            AND
                        Sales.SalesOrderHeader.
                            SalesPersonID = Sales.
                            SalesPerson.SalesPersonID
                            INNER JOIN
                        HumanResources.Employee ON
                            Sales.SalesPerson.
                            SalesPersonID =
                            HumanResources.Employee.
                            EmployeeID INNER JOIN
                        Person.Contact ON
                            HumanResources.Employee.
                            ContactID = Person.Contact
                            .ContactID AND
                            HumanResources.Employee.
                            ContactID = Person.Contact
                            .ContactID
```

```
                        where YEAR(Sales.
                            SalesOrderHeader.OrderDate
                            )=2002
GROUP BY Sales.SalesOrderHeader.SalesPersonID,
    Sales.SalesOrderHeader.SalesOrderID, Person.
    Contact.FirstName, Person.Contact.LastName,
                        Sales.SalesOrderHeader.
                            OrderDate
                        order by Sales.
                            SalesOrderHeader.
                            SalesPersonID
FOR XML AUTO, ELEMENTS
GO
```

2. Poizvedba

```
SELECT
--Sales.SalesOrderHeader.SalesPersonID,
--Person.Contact.FirstName, Person.Contact.LastName
    ,
--Sales.SalesTerritory.TerritoryID as drzava,
--Count(Sales.SalesOrderHeader.SalesOrderID) as
    steviloNarocil
--FROM          Sales.SalesOrderHeader INNER JOIN
--                      Sales.SalesPerson ON Sales.
    SalesOrderHeader.SalesPersonID = Sales.
    SalesPerson.SalesPersonID AND
--                      Sales.SalesOrderHeader.
    SalesPersonID = Sales.SalesPerson.SalesPersonID
    AND
--                      Sales.SalesOrderHeader.
    SalesPersonID = Sales.SalesPerson.SalesPersonID
```

```
    INNER JOIN
--                          HumanResources.Employee ON
    Sales.SalesPerson.SalesPersonID = HumanResources
    .Employee.EmployeeID INNER JOIN
--                          Person.Contact ON
    HumanResources.Employee.ContactID = Person.
    Contact.ContactID AND
--                          HumanResources.Employee.
    ContactID = Person.Contact.ContactID INNER JOIN
--                          Sales.SalesTerritory ON
    Sales.SalesOrderHeader.TerritoryID = Sales.
    SalesTerritory.TerritoryID AND
--                          Sales.SalesOrderHeader.
    TerritoryID = Sales.SalesTerritory.TerritoryID
    AND Sales.SalesOrderHeader.TerritoryID = Sales.
    SalesTerritory.TerritoryID AND
--                          Sales.SalesOrderHeader.
    TerritoryID = Sales.SalesTerritory.TerritoryID
--GROUP BY Sales.SalesOrderHeader.SalesPersonID,
    Person.Contact.FirstName, Person.Contact.
    LastName, Sales.SalesTerritory.TerritoryID
--FOR XML AUTO, ELEMENTS
--GO
```

3. Poizvedba

```
Update Person.Contact set  EmailAddress = '
    info@email.si' where ContactID=1
```

Literatura

[1] David Hunter, Jeff Rafter, Joe Fawcett " Beggining XML, 4th Edition", 2007

[2] Članek o XML podtkovnih bazah (2013). Dostopno na: http://cot.uni-mb.si/COTL/krajnc.html

[3] XML (Wikipedia) (2013). Dostopno na: http://en.wikipedia.org/wiki/XML

[4] XML and databse (2013). Dostopno na: http://www.rpbourret.com/xml/XMLAndDatabases.html

[5] Managing XML data: Native XML databases (2013). Dostopno na: http://www.ibm.com/developerworks/xml/library/x-mxd4/index.html

[6] Comparing XML database approaches(2013). Dostopno na: http://www.ibm.com/developerworks/library/x-comparexmldb/

[7] XML (2013). Dostopno na: http://www.xml.com/pub/a/98/10/guide0.html

[8] XMl database(2013). Dostopno na: http://www.codeproject.com/Articles/370869/XML-Databases

[9] Primerjava XMl baz ter relacijskih (2013). Dostopno na: http://khpi-iip.mipk.kharkiv.edu/library/extent/dbms/viper/vip2.html

[10] XMl database (Wikipedia)(2013). Dostopno na: http://en.wikipedia.org/wiki/XML_database

[11] Comparing XML database approaches(2013). Dostopno na: http://www.ibm.com/developerworks/library/x-comparexmldb/

[12] Native XML databse (2013). Dostopno na: http://webclub.ru/736-znakomstvo-s-estestvennymi-xml-bazami-dannyh.html

[13] XQuery(2013). Dostopno na: http://msdn.microsoft.com/ru-ru/library/ms189075(v=sql.90).aspx

[14] XML ter podatkovne baze (2013). Dostopno na: http://www.osp.ru/os/2000/10/178269/

[15] XML ter podatkovne baze(2013). Dostopno na: http://www.inftech.webservis.ru/it/internet/xml/ar2.html#l4

[16] XMl database (Wikipedia)(2013). Dostopno na: http://en.wikipedia.org/wiki/XML_database

[17] XML ter podatkovne bazi(2013). Dostopno na: http://www.iso.ru/print/rus/document5909.phtml

[18] What make XML such great technology? (2013). Dostopno na: http://www.codeproject.com/Articles/20486/What-makes-XML-such-a-great-technology

[19] XMl(2013). Dostopno na: http://www.25hoursaday.com/StoringAndQueryingXML.html

[20] Članek "A study on BaseX and Benchmarking"(2013). Dostopno na: https://mailman.uni-konstanz.de/pipermail/BaseX-talk/attachments/20100822/5b4d114d/attachment-0001.pdf

[21] Managing XML data: eXist – an open source native XML database (2013). Dostopno na:
http://www.ibm.com/developerworks/library/x-mxd5/

[22] XML ter podatkovne baze(2013). Dostopno na:
http://www.iso.ru/print/rus/document5909.phtml

[23] Getting started with eXist and XQuery (2013). Dostopno na:
http://databits.lternet.edu/node/52

[24] Glavna stran eXist(2013). Dostopno na:
http://exist-db.org/exist/apps/doc/uploading-files.xml

[25] XQuery(2013). Dostopno na:
http://www.w3schools.com/xquery/

[26] XQuery(2013). Dostopno na:
http://www.lib.tpu.ru/ amili/eXist-db.html

[27] XPath(2013). Dostopno na:
http://www.w3schools.com/xpath/xpath_intro.asp

[28] Glavna stran BaseX(2013). Dostopno na:
http://BaseX.org/

[29] BaseX(2013). Dostopno na:
http://www.datadirect.com/resources/dis/flwor/index.html

[30] Članek o XML. Dostopno na:
http://nativexmldatabase.com/2010/09/28/5-reasons-for-storing-xml-in-a-database/

[31] Članek o XML. Dostopno na:
http://www.dcs.bbk.ac.uk/ sven/adm08/xmlDBs.pdf

[32] Diplomska naloga *Primerjava pristopov k shranjevanju in obdelavi XML dokumentov v podatkovni bazi ORACLE* . Dostopno na:http://bevc.org/dl/docs/diplomaMB.pdf

[33] Predstavniki XML podatkovnih baz. Dostopno na:
http://www.rpbourret.com/xml/XMLDatabaseProds.htm#categories