UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

**Roman Oblak**

# UPORABA OBJEKTNIH TIPOV, IZDELANIH NA PODLAGI SHEM XML

DIPLOMSKO DELO NA

VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: doc. dr. Rok Rupnik

Ljubljana, 2011

University
of Ljubljana

Faculty of Computer
and Information Science

Tržaška 25
1000 Ljubljana, Slovenia
Phone: +386 1 476 84 11
Fax: +386 1 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00067/2011
Datum: 04.02.2011

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

Kandidat: **ROMAN OBLAK**

Naslov: **UPORABA OBJEKTNIH TIPOV IZDELANIH NA PODLAGI SHEM XML**
**THE USE OF XML SCHEMA BASED OBJECT TYPES**

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Prikažite različne možne načine shranjevanja in obdelave podatkov XML. Med možnimi načini preizkusite tudi hibridno shranjevanje, ki ga nadgradite z uporabo objektnih tipov. Izdelajte tudi primerjavo različnih načinov shranjevanja in obdelave podatkov.

Mentor:

doc. dr. Rok Rupnik

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

# I Z J A V A  O  A V T O R S T V U

## diplomskega dela

Spodaj podpisani Roman Oblak,

z vpisno številko 63030257,

sem avtor diplomskega dela z naslovom:

UPORABA OBJEKTNIH TIPOV, IZDELANIH NA PODLAGI SHEM XML

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom (naziv, ime in priimek)

  doc. dr. Roka Rupnika

  in somentorstvom (naziv, ime in priimek)

  /

- 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 12.5.2010 Podpis avtorja: Roman Oblak

# Zahvala

Zahvalil bi se mentorju Roku Rupniku in vsem, ki so mi stali ob strani pri pisanju diplomske naloge. Še posebej pa bi se rad zahvalil vsem, ki so mi dali znanje in voljo, ki je potrebna za uspeh.

KAZALO VSEBINE

**Seznam kratic in simbolov ..... IV**

**Povzetek ..... VI**

**Abstract ..... VII**

**1. Uvod ..... 1**

1.1 Podatkovna baza Oracle ..... 1

1.2 XML ..... 2

1.3 Shema XML ..... 3

1.4 Oracle XML DB ..... 3

**2. Predstavitev vhodnih podatkov ..... 7**

2.1 Nestrukturirano shranjevanje ..... 10

2.1.1 Uporaba podatkovnega tipa XMLType ..... 11

2.1.2 Uporaba XMLType podatkovnega tipa z registracijo sheme XML ..... 13

2.1.3 Uporaba podatkovnega tipa CLOB ..... 14

2.2 Strukturirano shranjevanje ..... 15

2.3 Binarno shranjevanje ..... 16

2.3.1 Brez registracije sheme XML ..... 17

2.3.2 Z registracijo sheme XML ..... 17

2.4 Hibridno shranjevanje ..... 18

**3. Primerjava ..... 22**

3.1 Okolje ..... 22

3.2 Primerjava hitrosti izvajanja stavkov DML ..... 23

3.2.1 Rezultati in ugotovitve ..... 24

3.3 Primerjava pretvorbe podatkov ..... 26

3.3.1 Rezultati in ugotovitve ..... 27

**4. Zaključek ..... 31**

**5. Priloge ..... 33**

5.1 Dodatek A ..... 33

**6. Viri in literatura ..... 45**

# KAZALO SLIK

Slika 1: Prikaz tabel relacijske podatkovne baze ..... 2

Slika 2: XMLType shranjevanje in Oracle XML DB Repozitorij [7]..... 5

Slika 3: Možni načini shranjevanja podatkov XML [8]..... 6

Slika 4: Grafičen prikaz sheme XML ..... 8

Slika 5: Grafičen prikaz sintakse funkcije BFILENAME [11]. ..... 9

Slika 6: Avtomatično generirani objektni tipi ..... 16

Slika 7: Avtomatično generirani objektni tipi na podlagi uporabe anotacije Oracle XML DB ..... 19

Slika 8: Grafičen prikaz primerjave hitrosti izvajanja stavka SELECT ..... 24

Slika 9: Grafičen prikaz primerjave hitrosti izvajanja stavka UPDATE..... 25

Slika 10: Grafičen prikaz primerjave hitrosti izvajanja stavka DELETE ..... 25

Slika 11: Grafičen prikaz primerjave hitrosti izvajanja stavka INSERT ..... 26

Slika 12: Grafičen prikaz primerjave porabe CPU..... 28

Slika 13: Grafičen prikaz primerjave št. prebranih blokov na diskovnem polju..... 29

Slika 14: Grafičen prikaz primerjave logičnih vhodno/izhodnih operacij ..... 30

Slika 15: Glavne prednosti in slabosti glede na različne možnosti shranjevanja [6] ..... 32

# Seznam kratic in simbolov

XML - razširljiv označevalni jezik (angl. *Extensible Markup Language*);

SQL - strukturiran povpraševalni jezik za delo s podatkovnimi bazami (angl. *Structured Query Language*) ;

ANSI - Ameriški državni inštitut za standarde (angl. *American National Standard Institute*);

PL/SQL - postopkovni programski jezik, ki razširja programski jezik SQL in se uporablja v podatkovnih bazah Oracle (angl. *Procedural Language/Standard Query Language*);

W3C - mednarodna skupnost za določanje spletnih standardev (angl. *World Wide Web Consortium*);

XPath - povpraševalni jezik, ki omogoča izbiro vozlišč določenega dokumenta XML (angl. *XML Path Language*);

XQuery - povpraševalni in funkcijski jezik za poizvedovanje po zbirkah vsebine dokumenta XML (angl. *XML Query*);

FTP - protokol, ki temelji na sistemu odjemalec-strežnik (angl. *File Transfer Protocol*);

HTTP - protokol za izmenjavo nadbesedil ter grafičnih, zvočnih in drugih večpredstavnostnih vsebin na spletu (angl. *Hypertext Transfer Protocol*);

HTTPS - protokol, ki omogoča varno internetno povezavo (angl. *Hypertext Transfer Protocol Secured scockets*);

WebDAV - skupek metod, ki temeljijo na HTTP, za izmenjavo, urejanje in nadzor nad dokumenti in datotekami shranjenih na spletu (angl. *Web-based Distributed Authoring and Versioning*);

URL - internetni naslov na katerem se nahaja vsebina (angl. *Uniform Resource Locator*);

JDBC - standard za baze podatkov pod JAVA programskim jezikom (angl. *Java Database Connectivity*);

DOM - sporazum oz. dogovor o predstavitvi in interakciji z objekti (angl. *Document Object Model*);

XSL - jezik za izražanje slogovnih datotek (angl. *Extensible Stylesheet Language*);

CLOB - zbirka znakovnih podatkov (angl. *Character Large Objects*);

RTR - register transakcijskih računov;

AJPES - Agencija Republike Slovenije za javnopravne evidence in storitve;

DML - jezik, ki omogoča vstavljanje, ažuriranje, brisanje in izbiro podatkov v podatkovni bazi (angl. *Data Manipulation Language*);

CPU - centralna procesna enota (angl. *Cenral Processing Unit*);

I/O - vhod/izhod (angl. *Input/Output*);

SAX - enostaven uporabniški vmesnik za delo z dokumenti XML (angl. *Simple API for XML*);

# Povzetek

Podatki v obliki XML niso novost in se z njimi v svetu informatike srečujemo na vsakem koraku. Novost pa prinašajo izpopolnjeni pristopi in metode zajema, branja in obdelave dokumentov XML.

Podatkovna baza Oracle nam ponuja različne pristope k shranjevanju podatkov XML. Omogoča nestrukturirano, strukturirano, binarno in hibridno shranjevanje podatkov. V diplomski nalogi je prikazanih več načinov obravnave takšnih podatkov, poudarek pa je na hibridnem shranjevanju, ki smo ga nadgradili z uporabo objektnih tipov, ki se avtomatsko kreirajo ob registraciji določene sheme XML na podatkovni bazi.

Izvedli smo dve primerjavi med omenjenimi načini shranjevanja. Prva primerjava je bila izdelana na podlagi izvajanja stavkov DML, kot so stavki SELECT, INSERT, DELETE in UPDATE. Primerjava je potrdila trditve, ki so veljavne za posamezne različne načine shranjevanja. Hkrati je potrdila tudi hitro in učinkovito izvajanje stavkov DML v primeru hibridnega načina, kakršnega smo prikazali v diplomski nalogi. Druga primerjava je bila zmogljivostna primerjava pretvorbe dokumentov XML v različne podatkovne tipe, ki se uporabljajo pri posameznem načinu shranjevanja. Čeprav je večje datoteke XML potrebno obravnavati na drugačne načine, se je izkazalo, da v primeru, da ima določena datoteka XML večjo vsebino podatkov, uporaba objektnih tipov kot način shranjevanja oz. pretvorbe podatkov ni primerna.

Diplomska naloga prikazuje možne načine shranjevanja in obdelave podatkov XML, kateri pa je primeren za uporabo, pa je predvsem odvisno od same tehnologije, ki jo imate na voljo, nato pa od vrste problema oz. poslovnega procesa, ki ga želite implementirati.

**Ključne besede**: podatkovna baza Oracle, objektni tipi, shema XML, podatki XML, metode shranjevanja

# Abstract

XML data is not new and we meet with it in every corner in the world of information technology. Advanced approaches and methods for importing, reading and processing XML documents are a novelty.

Oracle database offers different approaches of storing XML data. It offers unstructured, structured, binary and hybrid storage. Many different approaches are shown in this thesis, but the emphasis is on hybrid storage option, which we upgraded with using object types, that were created upon registering the XML schema on the database.

We performed two comparisons between mentioned different approaches. The first was based on executing DML statements, like SELECT, INSERT, DELETE and UPDATE statements. The comparison confirmed the claims that are true for each different method of storage, while it also confirmed the rapid and efficient execution of DML statements in the case of hybrid storage. The second comparison was the performance comparison of the conversion of XML documents to different data types used in each method of storage. Despite the fact that larger XML files should be dealt with in other ways, it has been shown that when a certain large XML file is used, using object types is not appropriate.

This thesis presents possible ways of storing and processing XML data, however, which one is suitable for you, mostly depends on the technology you use or the business process, which you have to implement.

**Keywords**: Oracle database, object types, schema XML, XML data, storing methods

# 1. Uvod

V računalniškem svetu so podatki v obliki XML nekaj vsakdanjega, zato se z njimi srečujemo na vsakem koraku, predvsem pri pomembni izmenjavi informacij bodisi med različnimi sistemi oz. tehnologijami, bodisi podjetji. Kako pa te podatke hranimo in obdelujemo, je odvisno predvsem od tehnologije, ki jo imamo na voljo. Osredotočili smo se na hranjenje in obdelavo podatkov XML v podatkovni bazi Oracle verzije 11g Release 2. Ker podatkovna baza Oracle omogoča hranjenje in obdelavo podatkov XML na različne načine, so predstavljeni tako primeri uporabe posameznega načina, njihove prednosti in slabosti, kot tudi dejanska meritev uporabljenih sistemskih virov pri različnih načinih.

Prikazani načini zajemajo predvsem obdelave podatkov XML, ki se lahko uporabljajo v različnih scenarijih oz. poslovnih procesih, ki jih moramo implementirati. Nekateri izmed možnih poslovnih procesov so lahko:

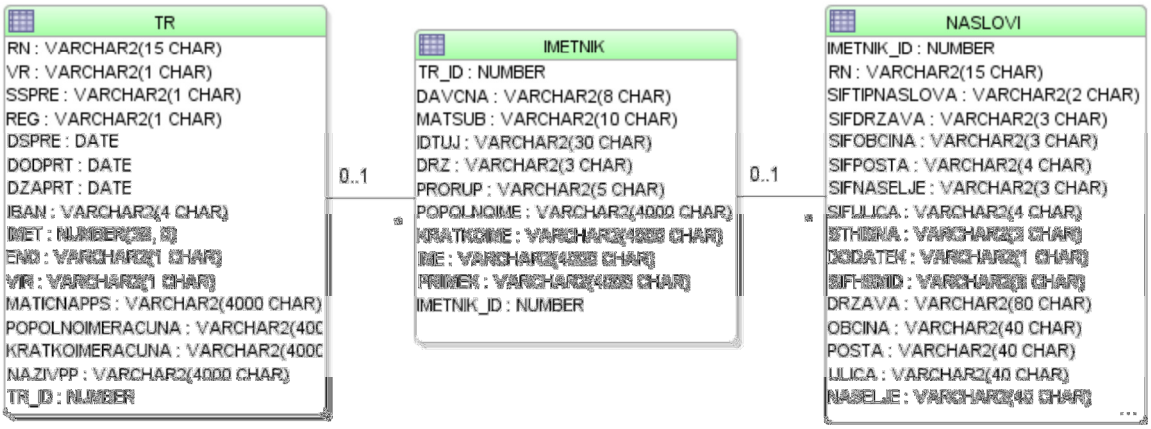
- zajem in obdelava dokumentov XML, ki predstavljajo sporočila (npr. dokument XML, ki predstavlja naročilo določenih izdelkov) in odgovore (npr. dokument XML, ki predstavlja odgovor, ki je lahko ali potrditev ali zavrnitev naročila);
- prikaz podatkov iz podatkovne baze v obliki XML (npr. podatke v relacijskih tabelah želimo prikazati v obliki XML);
- zajem in obdelava dokumentov XML, ki služijo kot izmenjava podatkov o določenih registrih in lahko predstavljajo celoten register ali dnevne spremembe registra (npr. podatki o tečajni listi, ki so dnevno objavljeni na spletu).

V našem primeru smo prikazali načine shranjevanja, ki imajo poudarek na obravnavi podatkov XML, kot del registra transakcijskih računih, spremembe katerega objavljajo dnevno.

Prikazan je poudarek na shranjevanju in obdelavi podatkov XML z uporabo objektnih tipov, zgrajenih na podlagi registrirane sheme XML, kar lahko razumemo kot hibridno shranjevanje, ki je mešanica različnih načinov shranjevanja podatkov.

## 1.1 Podatkovna baza Oracle

Podatkovna baza Oracle je relacijska podatkovna baza. Koncept relacijske podatkovne baze je prvič predstavil Dr. Edgar F. Codd leta 1970. Relacijska baza je zbirka med seboj sorodnih informacij, ki so organizirane v obliko tabel. Vsaka tabela hrani podatke v vrsticah, podatki pa so razvrščeni v stolpce. Tabele so shranjene v podatkovnih shemah (angl. *database schemas*), kjer lahko uporabniki hranijo podatke v svojih tabelah [1].




**Slika 1:** Prikaz tabel relacijske podatkovne baze

Relacijske podatkovne baze uporabljajo programski jezik SQL. ANSI standard SQL zagotavlja osnovne funkcije za delo s podatki, kontrolo nad transakcijami, in branje podatkov iz podatkovne baze. Večina poslovnih uporabnikov podatkovne baze uporablja aplikacije ali druga poslovna orodja, ki zagotavljajo vmesnike, pri katerih sta uporaba jezika SQL in njegova kompleksnost skrita [2].

Poleg jezika SQL naj omenimo še jezik PL/SQL. PL/SQL je jezik, ki je uporabljen v podatkovnih bazah Oracle in je postopkovni programski jezik, ki razširja jezik SQL in se pogosto uporablja pri implementaciji programske logike modulov aplikacije. PL/SQL se lahko uporablja za kreiranje baznih procedur in prožilcev (angl. *triggers*), kontroliranje zank (angl. *looping controls*), pogojnih stavkov, in ravnanje z napakami (angl. *error handling*). PL/SQL procedure se lahko prevede in hrani znotraj podatkovne baze. Z uporabo orodja kot je SQL*Plus, ki je interaktivno orodje zagotovljeno v vseh verzijah podatkovne baze Oracle, se lahko PL/SQL procedure izvede. PL/SQL programski deli (angl. *program units*) so lahko predprevedeni (angl. *precompiled*) [2].

## 1.2 XML

XML, razširljivi označevalni jezik, omogoča razvijalcem kreiranje svojih formatov za hranjenje in posredovanje informacij v skupno rabo. Z uporabo te svobode, so razvijalci kreirali dokumente, ki predstavljajo neverjeten razpon informacij, in XML lahko olajša marsikateri problem, ki lahko nastane pri izmenjavi informacij. Ključni del tega procesa je formalna deklaracija in dokumentacija teh formatov, ki zagotavljajo temelje na katerih lahko razvijalci razvijajo programsko opremo [3].

Primer XML dokumenta:

```
<?xml version="1.0" encoding="UTF-8"?>
<Naslov sifPosta="1358">
        <Obcina>Vrhnika</Obcina>
```

```
        <Posta>Log pri Brezovici</Posta>
        <Naselje>Bevke</Naselje>
</Naslov>
```

## 1.3 Shema XML

Shema XML predstavlja jezik, ki formalizira omejitve (angl. *constraints*), predstavljene kot pravila ali kot model strukture, ki je uporabljen v razredu dokumenta XML. V večini primerov, sheme služijo kot oblikovalsko orodje, ki vzpostavi ogrodje nad katerim se izvaja implementacija. Formalizacija je nujna za razvijajalce programske opreme, zato formalizacija omejitev in struktur vodi v zelo raznolike aplikacije. Nove aplikacije, ki uporabljajo sheme nastajajo dnevno, kljub temu pa lahko sheme razvrstimo v orodja za validacijo, dokumentacijo, poizvedovanje, povezovanje ali urejanje [3].

Primer sheme XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Obcina" type="xsd:string"/>
  <xsd:element name="Naslov">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Obcina"/>
        <xsd:element ref="Posta"/>
        <xsd:element ref="Naselje"/>
      </xsd:sequence>
      <xsd:attribute name="sifPosta" type="xsd:integer"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Naselje" type="xsd:string"/>
  <xsd:element name="Posta" type="xsd:string"/>
</xsd:schema>
```

## 1.4 Oracle XML DB

Podatkovna baza Oracle za delo z dokumenti XML od verzije 9i naprej, ponuja tehnologijo Oracle XML DB. Ta tehnologija omogoča shranjevanje, generiranje, branje, iskanje, validacijo, transformacijo, nadgrajevanje in indeksiranje XML podatkov. Funkcionalnosti in prednosti, ki jih s tem ponuja so:

- abstraktni podatkovni tip SQL, XMLType;
- zagotavljanje zanesljivosti, razpoložljivosti, nadgradljivost in varnost, poleg tega nudi še upravljanje pomnilnika (angl. *memory management*) in optimizacijo;
- standardne načine branja in ažuriranja podatkov XML. Ti standardi vključujejo SQL/XML standard in W3C XML in shema XML standarde, poleg tega pa še priporočila za uporabo XPath in XQuery. Za zapis vsebine XML v in iz podatkovne baze Oracle se lahko uporablja FTP, HTTP, HTTPS in WebDav. Standardni

programski vmesniki zagotavljajo programski dostop in delo z vsebino XML z uporabo jezikov, kot so Java, C in PL/SQL;

- načine shranjevanja, poizvedovanja, ažuriranja in transformacije podatkov XML, dostop do katerih je omogočen z uporabo jezika SQL;
- načine za izvajanje operacij XML na podatkih SQL;
- Oracle XML DB repozitorij: enostaven, lahki repozitorij, kjer lahko organiziramo in urejamo vsebino podatkovne baze, vključno z vsebino XML, z uporabo prispodob datoteka/direktorij/URL;
- načine dostopa in združevanja podatkov distribuiranih sistemov skozi prehod (angl. *gateway*), z uporabo enostavnega, skupnega podatkovnega modela. To zmanjšuje kompleksnost razvijanja aplikaciji, ki se ukvarjajo s podatki na različnih lokacijah;
- načine uporabe Oracle XML DB skupaj z Oracle XML Developer's Kit za razvoj aplikacij, ki tečejo na srednjem nivoju na ali aplikacijskem strežniku Oracle ali podatkovni bazi Oracle [4].

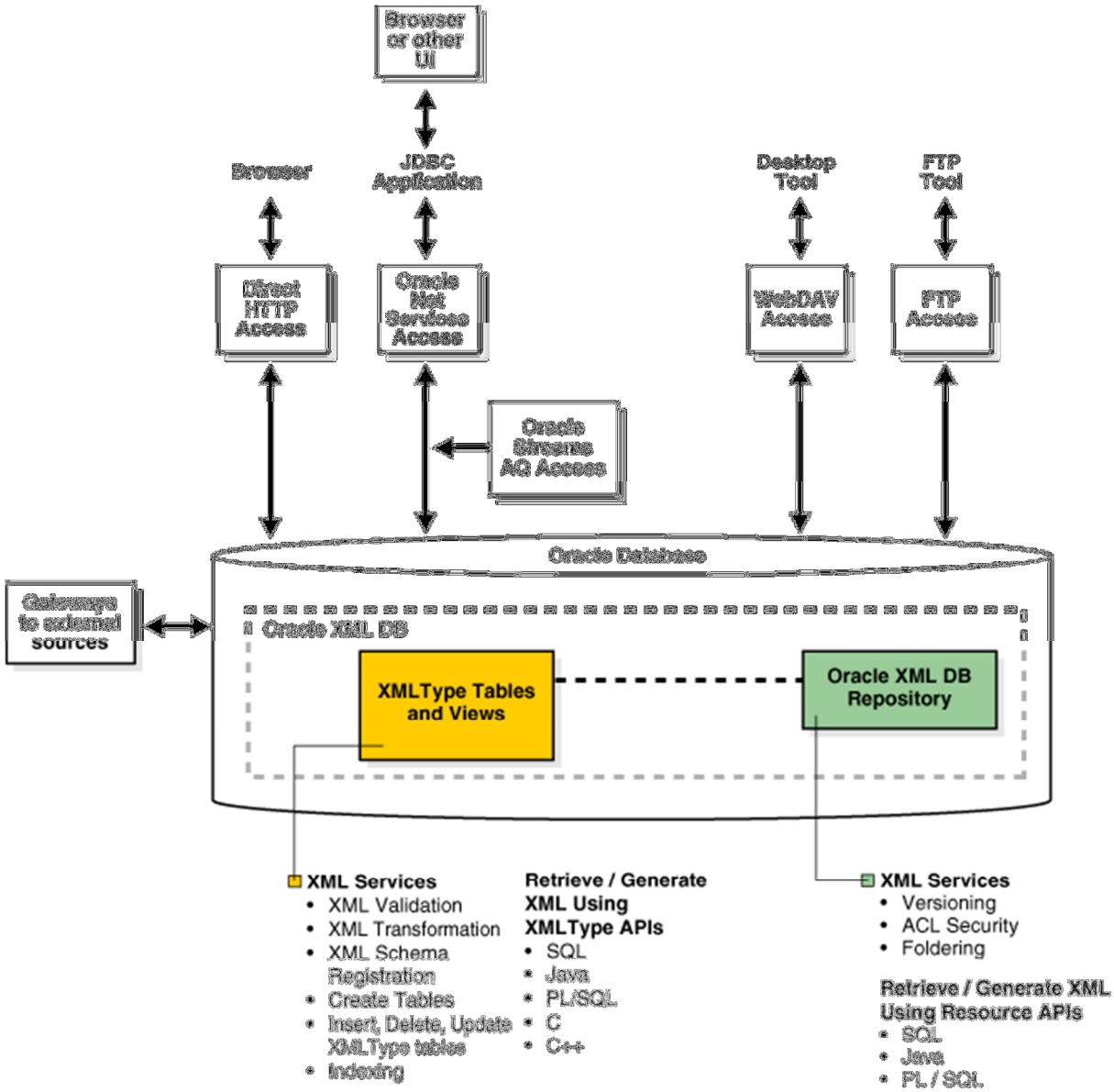
Podatkovni tip XMLType je Oracle tip, ki se uporablja za hranjenje in branje podatkov XML v podatkovni bazi. Uporablja se lahko kot argument funkcije ali kot podatkovni tip določenega stolpca v tabeli ali vpogledu [5].

Slika 2 prikazuje arhitekturo podatkovne baze Oracle XML in glavne prednosti shranjevanja podatkov XML v podatkovni tip XMLType:

- indeksiranje XMLType tabel in vpogledov z uporabo XMLIndex, B-Tree in Oracle Text indeksov,
- shranjevanje v lokalne ali oddaljene tabele.

Prikazane so tudi glavne prednosti Oracle XML DB repozitorija, ki omogoča shranjevanje kakršnihkoli dokumentov, vključno z XML dokumenti, ki temeljijo na ustrezni shemi XML registrirani v podatkovni bazi Oracle. Dostop do dokumentov v repozitoriju je mogoč na sledeče načine:

- HTTP(S), z uporabo protokola HTTP,
- WebDAV in FTP, z uporabo WebDAV in FTP protokolov,
- SQL, z uporabo Oracle Net Services in z uporabo Java Database Connectivity (JDBC) [6].




**Slika 2:** XMLType shranjevanje in Oracle XML DB Repozitorij [7]

Bistvene prednosti Oracle XML DB tehnologije so:

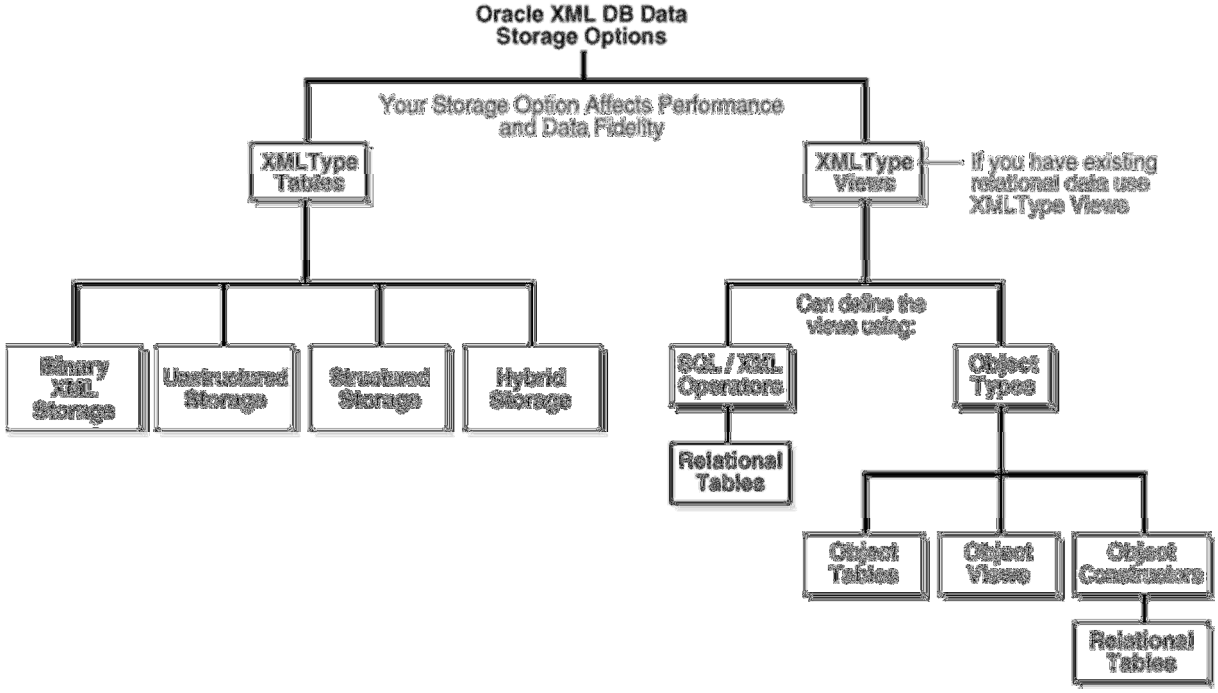
- XML/SQL dvojnost (angl. duality) – omogoča uporabo in prikaz podatkov kot stolpce v tabeli z uporabo poizvedb SQL ali kot vozlišča v dokumentu XML z uporabo tehnologij, kot so DOM in XSL transformacije,
- SQL/XML standardne funkcije – funkcije omogočajo generiranje podatkov XML na podlagi poizvedb SQL in funkcije, ki omogočajo branje in dostop do podatkov XML.

Podatkovni tip XMLType omogoča shranjevanje na tri različne načine in sicer:

- nestrukturirano ali CLOB shranjevanje – shranjevanje dokumentov v obliko CLOB,

- strukturirano ali objektno relacijsko shranjevanje – shranjevanje dokumentov kot množica objektov,
- binarno XML shranjevanje – shranjevanje dokumentov v binarnem formatu, kateri je prilagojen dokumentom XML,
- hibridno shranjevanje – uporaba različnih pristopov shranjevanja.

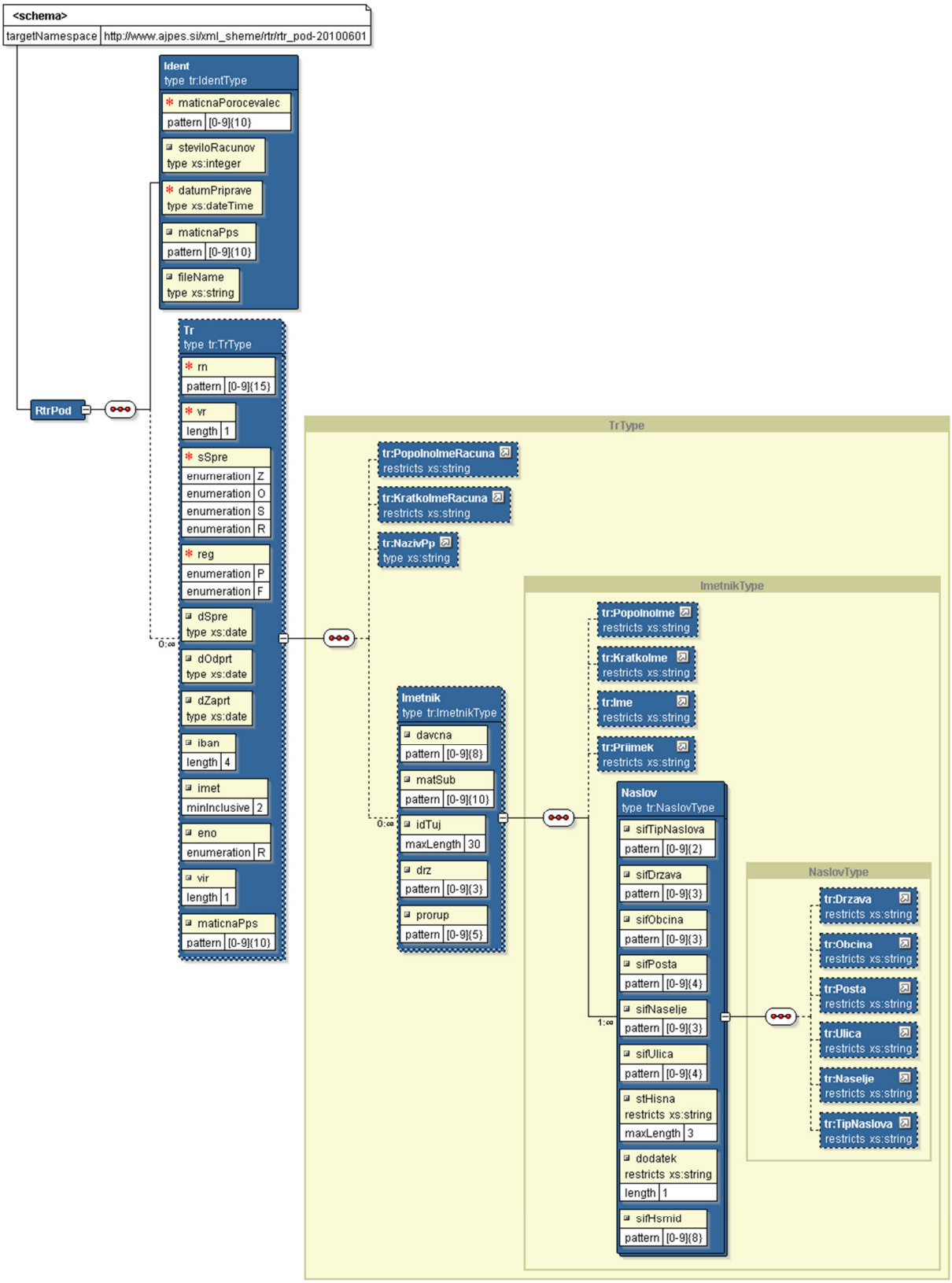
Spodnja slika grafično prikazuje možnosti shranjevanja podatkov XML:




**Slika 3:** Možni načini shranjevanja podatkov XML [8]

# 2. Predstavitev vhodnih podatkov

Primeri v naslednjih poglavjih temeljijo na zajemu in obdelavi podatkov o transakcijskih računih, dostop do katerih je omogočila Agencija Republike Slovenije za javnopravne evidence in storitve. Register transakcijskih računov (RTR) je enotna informatizirana baza podatkov o transakcijskih računih in o imetnikih transakcijskih računov [9]. AJPES omogoča javen dostop do podatkov o transakcijskih računih za pravne osebe v obliki datotek XML. Struktura datoteke XML je definirana z shemo XML »RTR_POD_V1.2.XSD« [10] in je grafično prikazana na spodnji sliki:




**Slika 4:** Grafičen prikaz sheme XML

Iz grafične predstavitve sheme XML je razvidno, da je v eni datoteki XML podatek o tem kdo je datoteko izdelal, potem pa vsebuje podatke o enem ali več transakcijskih računih. Vsak izmed njih lahko vsebuje podatke o enem ali več imetnikih transakcijskih računov, ti pa so lahko vezani na enega ali več naslovov.

Za boljšo predstavo vsebine datoteke XML je v pomoč spodnji primer, ki opisuje podatek o enem transakcijskem računu, ki je dodeljen enemu imetniku z enim naslovom:

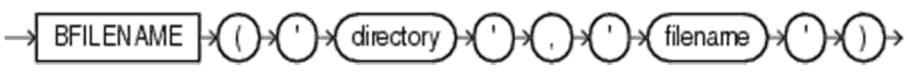
```
<RtrPod>
 <Ident datumPriprave="2010-06-18T11:18:21" maticnaPps="1234567000"
maticnaPorocevalec="1234567000" steviloRacunov="16027" />
 <Tr rn="123410113309446" vr="T" sSpre="O" reg="P" dSpre="2010-06-16"
dOdprt="2010-06-16" iban="SI56" vir="B" maticnaPps="1234567000">
  <Imetnik davcna="12345678" matSub="1234567000">
   <PopolnoIme>TESTNO POPOLNO IME</PopolnoIme>
   <KratkoIme>KRATKO IME</KratkoIme>
   <Naslov sifTipNaslova="01" sifObcina="001" sifPosta="1234" sifUlica="0000"
sifNaselje="024" stHisna="012" dodatek="A" sifHsmid="01030340">
    <Obcina>AJDOVŠČINA</Obcina>
    <Posta>AJDOVŠČINA</Posta>
    <Ulica>OTLICA</Ulica>
    <Naselje>OTLICA</Naselje>
   </Naslov>
  </Imetnik>
 </Tr>
</RtrPod>
```

V nadaljenju so prikazani primeri zajema oz. shranjevanja in prikaza podatkov glede na različne možne načine obravnave podatkov XML v podatkovni bazi Oracle. Branje vsebine datotek XML v podatkovno bazo Oracle je lahko različno in lahko zajema klicanje za to namenjene spletne storitve, uporabo različnih protokolov, kot so HTTP ali FTP, branje datotek iz same podatkovne baze iz strežniškega direktorija. Kljub temu, da se podatki o transakcijskih računih s strani AJPESa odlagajo na določen direktorij FTP, je za branje datotek XML v sledečih primerih uporabljena Oracle funkcija BFILENAME. Smisel primerov temelji na obdelavi podatkov, ne pa na samem pridobivanju podatkov.

Za uporabo funkcije moramo najprej izdelati objekt, ki bo hranil vrednost oz. pot do direktorija. To storimo z ukazom:

```
CREATE or REPLACE directory CTEMPXML  as 'C:\temp\xml';
```

Sintaksa funkcije BFILENAME je grafično prikazana s sliko:




**Slika 5:** Grafičen prikaz sintakse funkcije BFILENAME [11].

Primer uporabe z uporabo poizvedb SQL:

```
SELECT bfilename('CTEMPXML', 'TEST.XML') FROM dual;
```

Kot že omenjeno Oracle ponuja štiri različne načine shranjevanja podatkov XML:

- nestukturirano shranjevanje,
- strukturirano shranjevanje,
- binarno shranjevanje,
- hibridno shranjevanje.

Poleg omenjenih bodo v sledečih poglavjih znotraj opisa posameznega načina shranjevanja predstavljeni tudi alternativni pristopi zajema in obdelave podatkov XML.

Vsi načini so predstavljeni na enak oz. podoben način. Najprej je izdelana tabela, ki vsebuje podatke oz. vsebino ene celotne datoteke XML, nato se iz njene vsebine prebere podatek o posameznem transakcijskem računu, ki se zapiše v svojo tabelo. Nad temi podatki se nato izvajajo INSERT, SELECT, UPDATE in DELETE stavki, ki jih lahko z eno besedo poimenujemo stavki DML (angl. *Data Manipulation Language*).

## 2.1 Nestrukturirano shranjevanje

Nestrukturirano oz. CLOB shranjevanje omogoča hitrejši pretok podatkov kot strukturirano shranjevanje kadar se izvaja vpis (stavek INSERT) ali branje (stavek SELECT) celotnega dokumenta XML. Ker ni potrebe po pretvorbi podatkov, se lahko format uporabi tudi zunaj podatkovne baze. Nestrukturirano shranjevanje omogoča boljšo prilagodljivost kot strukturirano shranjevanje v strukturi dokumenta XML. Nestrukturirano shranjevanje je posebej primerno kadar dokumente XML obravnavamo kot celoto in se ne osredotočamo na njihovo strukturo. Omenjene prednosti pa se pojavijo na račun nekaterih pogledov na inteligentno procesiranje: ker ne uporabljamo indeksiranja, podatkovna baza ni zmožna optimizacije pri izvajanju poizvedb in ažuriranju podatkov XML, kadar so le ti shranjeni kot CLOB objekti. Še posebno lahko razčlenjevanje (angl. *parsing*) občutno vpliva na zmoglivost izvajanja poizvedb [6].

Nestrukturirano shranjevanje je prikazano na treh različnih primerih oz. načinih in sicer:

- shranjevanje z uporabo podatkovnega tipa XMLType,
- shranjevanje z uporabo podatkovnega tipa XMLType z registracijo sheme XML,
- shranjevanje z uporabo podatkovnega tipa CLOB.

### 2.1.1 Uporaba podatkovnega tipa XMLType

Za nestrukturirano shranjevanje podatkov smo s spodnjim ukazom izdelali tabelo TEST_1A, ki vsebuje polje tipa XMLType.

```
create table TEST_1A
(datoteka varchar2(25), racuni xmltype)
XMLTYPE COLUMN racuni  STORE AS CLOB;
```

Opcija »STORE AS CLOB« je privzeta opcija in pomeni, da so podatki shranjeni v nestukturirani oz. obliki CLOB.

Kot že omenjeno smo za branje podatkov predvideli funkcijo BFILENAME. Z spodnjim stavkom INSERT tako enostavno zapišemo podatke v tabelo TEST_1A.

```
insert into TEST_1A
  (datoteka, racuni)
select '10.xml', xmltype(bfilename('CTEMPXML', '10.xml'),nls_charset_id('AL32UTF8'))
  from dual;
```

Za namene prikaza branja in ažuriranja zapisov ne bomo obravnavali podatkov kot celoto, vendar bomo prikazali primere na nivoju podatka o enem transakcijskem računu. Za te namene smo izdelali tabelo s spodnjim ukazom:

```
create table TEST_1A_RACUNI
(datoteka varchar2(25), racun xmltype)
XMLTYPE COLUMN racun  STORE AS CLOB;
```

Sedaj lahko z ukazi SQL izvajamo stavke DML nad posameznimi zapisi v tabeli TEST_1A_RACUNI.

Za branje podatkov iz podatkovnega tipa XMLType smo uporabili dve funkciji in sicer:

- Extract() – za branje določenega nivoja podatkov (npr. sklop podatkov o imetniku)
- Extractvalue() – za branje določenega elementa (npr. podatek PopolnoIme).

Podatke bomo s pomočjo stavka INSERT in izvorne tabele TEST_1A enostavno napolnili s spodnjim ukazom:

```
insert into TEST_1A_RACUNI
  (datoteka, racun)
select t.datoteka, tr.object_value
from TEST_1A t,
     table (xmlsequence(extract(t.racuni,'//Tr',
'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"'))) tr;
```

Primer stavka SELECT, ki prikaže vse podatke iz sklopa podatkov o enem transakcijskem računu:

```
select
extractvalue(value(tr), '//@rn') as rn,
extractvalue(value(tr), '//@vr') as vr,
extractvalue(value(tr), '//@sSpre') as sSpre,
```

```
extractvalue(value(tr), '//@reg') as reg,
extractvalue(value(tr), '//@dSpre') as dSpre,
extractvalue(value(tr), '//@dOdprt') as dOdprt,
extractvalue(value(tr), '//@dZaprt') as dZaprt,
extractvalue(value(tr), '//@iban') as iban,
extractvalue(value(tr), '//@imet') as imet,
extractvalue(value(tr), '//@eno') as eno,
extractvalue(value(tr), '//@vir') as vir,
extractvalue(value(tr), '//@maticnaPps') as maticnaPps,
extractvalue(value(tr), '//PopolnoImeRacuna' ,
'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"') as PopolnoImeRacuna,
extractvalue(value(tr), '//KratkoImeRacuna' ,
'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"') as KratkoImeRacuna,
extractvalue(value(tr), '//NazivPp' , 'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-
20100601"') as NazivPp,
extractvalue(value(im), '//@davcna') as davcna,
extractvalue(value(im), '//@matSub') as matSub,
extractvalue(value(im), '//@idTuj') as idTuj,
extractvalue(value(im), '//@drz') as drz,
extractvalue(value(im), '//@prorup') as prorup,
extractvalue(value(im), '//PopolnoIme' , 'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-
20100601"') as PopolnoIme,
extractvalue(value(im), '//KratkoIme' , 'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-
20100601"') as KratkoIme,
extractvalue(value(im), '//Ime' , 'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-
20100601"') as Ime,
extractvalue(value(im), '//Priimek' , 'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-
20100601"') as Priimek,
extractvalue(value(n), '//@sifTipNaslova') as sifTipNaslova,
extractvalue(value(n), '//@sifDrzava') as sifDrzava,
extractvalue(value(n), '//@sifObcina') as sifObcina,
extractvalue(value(n), '//@sifPosta') as sifPosta,
extractvalue(value(n), '//@sifNaselje') as sifNaselje,
extractvalue(value(n), '//@sifUlica') as sifUlica,
extractvalue(value(n), '//@sifHisna') as sifHisna,
extractvalue(value(n), '//@dodatek') as dodatek,
extractvalue(value(n), '//@sifHsmid') as sifHsmid,
extractvalue(value(n), '//Drzava' , 'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-
20100601"') as Drzava,
extractvalue(value(n), '//Obcina' , 'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-
20100601"') as Obcina,
extractvalue(value(n), '//Posta' , 'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-
20100601"') as Posta,
extractvalue(value(n), '//Ulica' , 'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-
20100601"') as Ulica,
extractvalue(value(n), '//TipNaslova' , 'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-
20100601"') as TipNaslova
from test_1a_racuni a,
       table (xmlsequence(extract(a.racun,'//Tr',
'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"'))) tr,
       table (xmlsequence(extract(value(tr),'//Imetnik',
'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"'))) im,
       table (xmlsequence(extract(value(im),'//Naslov',
'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"'))) n
    where EXISTSNODE(a.racun, '/Tr[@rn="401415847510272"]'
                             ,  'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"')
= 1;
```

Primer stavka UPDATE za ažuriranje določenega sklopa podatkov na nivoju enega transakcijskega računa:

```
update TEST_1A_RACUNI t
    set racun = updateXML(racun, '//KratkoIme[1]/text()', 'TEST UPDATE'
              , 'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"')
where EXISTSNODE(t.racun, '/Tr[@rn="401415847510272"]'
                             ,  'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"')
= 1;
```

Primer stavka DELETE:

```
delete TEST_1A_RACUNI t where EXISTSNODE(t.racun, '/Tr[@rn="401415847510272"]'

                                ,   'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"')
= 1;
```

Kot lahko opazimo, je za branje podatkov na prikazan način potrebno znanje jezika XPATH in seveda dobro poznavanje strukture podatkov XML. Jezika XPATH ne bomo posebej obravnavali, naj samo omenimo, da gre za jezik, ki omogoča naslavljanje delov dokumenta XML [12].

### 2.1.2 Uporaba XMLType podatkovnega tipa z registracijo sheme XML

V kolikor želimo shranjevati podatke na takšen način, moramo najprej registrirati shemo XML v podatkovno bazo Oracle. To storimo s proceduro registerSchema, ki je del paketa DBMS_XMLSCHEMA, kateri ponuja procedure za upravljanje z shemami XML [13]. S spodnjim ukazom registriramo shemo XML, katero poimenujemo »RTR_POD_V1.2.XSD«:

```
begin
  dbms_xmlschema.registerSchema(schemaURL => 'RTR_POD_V1.2.XSD'
                               ,schemaDoc => bfilename('CTEMPXML', 'RTR_POD_V1.2.XSD')
                               ,genTypes => false
                               ,genTables => false
                               );
end;
/
```

Procedura ima sicer več vhodnih parametrov, v tem primeru pa smo uporabili zgolj štiri, in sicer parameter schemaURL, ki definira ime registrirane sheme v podatkovni bazi Oracle, schemaDoc, ki dejansko vsebuje vsebino sheme XML, ki jo preberemo kot BFILE, genTypes parameter, ki, v kolikor je nastavljen na »true«, ob registraciji generira objektne tipe na podlagi sheme XML in parameter genTables, ki določa ali se ob registraciji kreirajo tudi tabele na podlagi sheme XML.

Vse registrirane sheme XML so vidne preko sistemskega vpogleda USER_XML_SCHEMAS. Registrirane sheme s proceduro deleteSchema, ki je ravno tako del paketa DBMS_XMLSCHEMA, enostavno izbrišemo, v kolikor je ne potrebujemo več.

Sedaj lahko izdelamo tabelo, ki bo vsebovala podatkovni tip XMLType, ki bo odvisen od registrirane sheme XML. To storimo s spodnjima dvema ukazoma:

```
create table TEST_1B
(datoteka varchar2(25), racuni xmltype)
XMLTYPE COLUMN racuni  store AS CLOB
 XMLSCHEMA "RTR_POD_V1.2.XSD"
    element "RtrPod";
```

```
create table TEST_1B_RACUNI
(datoteka varchar2(25), racun xmltype)
XMLTYPE COLUMN racun  store AS CLOB
XMLSCHEMA "RTR_POD_V1.2.XSD"
    element "Tr";
```

Tabele smo izdelali na podoben način kot v poglavju 2.1.1. Dodatno smo uporabili le opcijo XMLSCHEMA in s tem definirali, da se bo v polje tipa XMLType shranjevalo podatke po strukturi registrirane sheme. Z opcijo ELEMENT smo natančno definirali kateri element iz sheme bo shranjen v tabeli oz. stolpcu tipa XMLType.

Pomembna informacija pri izdelavi tabele na zgoraj prikazan način je, da mora biti element, ki ga določimo z opcijo ELEMENT, znotraj sheme XML definiran globalno.

Podatke smo v tabelo TEST_1B zapisali z izvedbo enakega stavka INSERT kot v primeru v poglavju 2.1.1:

```
insert into TEST_1B
  (datoteka, racuni)
select '10.xml', xmltype(bfilename('CTEMPXML', '10.xml'),nls_charset_id('AL32UTF8'))
  from dual;
```

Primeri z uporabo stavkov DML so popolnoma enaki kot pri poglavju 2.1.2, kakšne pa so performančne prednosti oz. slabosti pa je prikazano v poglavju 3. Bistvena razlika v tem načinu je ta, da se ob shranjevanju oz. ažuriranju XMLTYPE podatka izvaja tudi delno preverjanje vsebine dokumenta XML glede na registrirano shemo XML.

### 2.1.3 Uporaba podatkovnega tipa CLOB

Nestrukturirano shranjevanje lahko dosežemo tudi na drugačen način in sicer ne z uporabo podatkovnega tipa XMLType, ampak s shranjevanjem neposredno v podatkovni tip CLOB. V tem primeru smo tako izdelali dve tabeli s spodaj navedenima ukazoma:

```
create table TEST_1C
(datoteka varchar2(25), racuni CLOB);
```

```
create table TEST_1C_RACUNI
(datoteka varchar2(25), racun CLOB);
```

Primeri uporabe poizvedb SQL v takem primeru se ne razlikujejo dosti od zgoraj prikazanih, pri uporabi je potrebno poskrbeti za pretvorbo med podatkovnima tipoma CLOB in XMLTYPE. Naj pretvorbo pokažemo na primeru INSERT stavka:

```
insert into test_1c_racuni
  (datoteka, racun)
select t.datoteka, tr.object_value.getclobval()
from test_1C t,
     table (xmlsequence(extract(xmltype(t.racuni),'//Tr',
'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"'))) tr;
```

Kot vidimo, smo v tem primeru uporabili konstruktor XMLType za pretvorbo objektnega tipa CLOB v XMLType in funkcijo getClobVal() za pretvorbo nazaj v podatkovni tip CLOB. Rezultati so tako enaki, performance pa seveda drugačne, kar je pokazala izvedena primerljivost. Pri izvajanju ažuriranja bi sicer lahko uporabljali pretvorbe med tipoma CLOB in XMLType, vendar to ni smiselno, zato se stavek UPDATE v takem načinu shranjevanja razlikuje, saj vedno ažuriramo celotno vsebino in ne zgolj dela vsebine dokumenta XML:

```
update TEST_1C_RACUNI t
    set racun = '<Tr>. . .</Tr>'
where EXISTSNODE(xmltype(t.racun), '/Tr[@rn="401415847510272"]'
                                 ,  'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"')
= 1;
```

Bistvena prednost takšnega načina shranjevanja se pokaže predvsem takrat, ko ne potrebujemo dostopa do podatka znotraj dokumenta XML ampak vedno dostopamo do podatkov kot celota.

## 2.2 Strukturirano shranjevanje

Strukturirano (objektno-relacijsko) shranjevanje v primerjavi z drugimi načini shranjevanja prinaša prednosti pri izvajanju poizvedb in ažuriranj, optimizirano upravljanje s spominom, zmanjšane zahteve po prostoru, t.i. »B-Tree« indeksiranje, in t.i. »in-place« ažuriranje. Te prednosti so na račun povečanega procesiranja pri izvajanju zapisovanja in branja celotne vsebine XML, in zmanjšana prilagodljivost v strukturi XML, ki jo lahko upravlja tabela ali stolpec tipa XMLType. Strukturna prilagodljivost je zmanjšana, ker so podatki in metapodatki (npr. imena stolpcev) ločeni. Primerki struktur se ne razlikujejo zlahka. Strukturirano shranjevanje je posebej primerno za visoko strukturirane podatke, katerih struktura se ne spreminja, v kolikor je povezana s kontroliranim številom podatkovnih tabel in povezav [6].

Strukturirano shranjevanje podatkov pri izdelavi tabele dosežemo z uporabo ukaza »STORE AS OBJECT RELATIONAL« namesto ukaza »STORE AS CLOB, ki smo ga uporabili v primeru nestrukturiranega shranjevanja. Razlika je očitna že pri izdelavi tabele, saj je v takem primeru nujna uporaba »XMLSchema_spec« dodatka, s katerim povemo na katero shemo in na kateri element se nanaša izdelano polje XMLType. Primer ukaza za izdelavo tabele:

```
create table TEST_2
(datoteka varchar2(25), racuni xmltype)
XMLTYPE COLUMN racuni  store AS OBJECT RELATIONAL
 XMLSCHEMA "RTR_POD_V1.2_STRUCT.XSD"
    element "RtrPod";
```

Za potrebe delovanja zgornjega ukaza smo poleg že registrirane sheme z imenom »RTR_POD_V1.2.XSD«, ponovno registrirali enako shemo XML, vendar pod drugim imenom in sicer »RTR_POD_V1.2_STRUCT.XSD«. To smo storili zato, ker smo v tem primeru uporabili drugačne vrednosti parametrov funkcije registerSchema. Parameter genTypes smo tako nastavili na vrednost »true«, saj smo ugotovili, da brez takšne nastavitve ni možno izdelati tabele glede na zgornji način. Registracijo sheme smo izvedli z naslednjim ukazom:

```
begin
  dbms_xmlschema.registerSchema(schemaURL => 'RTR_POD_V1.2_STRUCT.XSD'
                               ,schemaDoc => bfilename('CTEMPXML', 'RTR_POD_V1.2.XSD')
                               ,genTypes => TRUE
                               ,genTables => FALSE
                               );
end;
/
```

Ob izvedbi zgornjega ukaza se avtomatično izdelajo tudi objektni tipi na podlagi registrirane sheme XML (zaradi nastavitve parametra genTypes na »true«):

| TYPE_NAME | TYPECODE | ATTRIBUTES |
|---|---|---|
| RtrPod676_T | OBJECT | 3 |
| Tr678_COLL | COLLECTION | 0 |
| IdentType675_T | OBJECT | 6 |
| TrType656_T | OBJECT | 17 |
| Imetnik674_COLL | COLLECTION | 0 |
| ImetnikType660_T | OBJECT | 11 |
| NaslovType665_T | OBJECT | 16 |
| Naslov673_COLL | COLLECTION | 0 |

**Slika 6:** Avtomatično generirani objektni tipi

Podatkovna baza Oracle za shranjevanje podatkov na strukturiran način uporablja generirane objektne tipe in je zato parameter genTypes potrebno nujno nastaviti na vrednost »true«.

Ko kreiramo tabelo lahko opazimo, da podatkovna baza Oracle avtomatsko kreira sistemske tabele če shema XML vključuje več nivojev podatkov. Shema RTR_POD_V1.2.XSD, vključuje več nivojev podatkov in sicer podatke o transakcijskih računih, podatke o imetnikih in podatke o naslovu. Zato so bile ob izvedbi ukaza »CREATE TABLE« avtomatsko izdelane tri sistemske tabele za vsak nivo podatkov.

Uporaba poizvedb SQL nad podatki shranjenimi na strukturiran način je popolnoma enak načinu dela z uporabo XMLType podatkovnega tipa izdelanega na podlagi sheme XML, ki je prikazan v poglavju 2.1.2.. Prednosti kot tudi slabosti takšnega načina shranjevanja, ki so opisane na začetku tega podpoglavja, pa potrdi tudi izvedena primerljivost.

## 2.3 Binarno shranjevanje

Binarno shranjevanje podatkov XML prinaša učinkovitejše podatkovno shranjevanje, ažuriranje, indeksiranje in delno ekstrahiranje (angl. *extraction*) kot nestrukturirano shranjevanje. Prinese lahko tudi performančno boljše izvajanje poizvedb. Kot strukturirano shranjevanje, je binarno shranjevanje tudi lahko odvisno od sheme XML in njenih podatkovnih tipov in lahko prinaša prednosti pri uporabi tipov, ki pripradajo podatkovni bazi. Enako kot strukturirano shranjevanje je tudi v tem načinu možno izvajanje ažuriranj po delih oz. po sklopih podatkov XML. Ker je lahko binarna vsebina podatkov XML dostopna tudi zunaj podatkovne baze, lahko služi kot učinkovit medij za izmenjavo podatkov in lahko razbremeni podatkovno bazo in tako posledično poveča njeno zmogljivost. Kot v primeru nestrukturirano shranjenih podatkov, je tudi v primeru binarnih podatkov ohranjen vrstni red dokumentov. Za potrebe učinkovitosti, so lahko podatki in metapodatki ločeni, enako kot pri strukturiranemu shranjevanju. Z razliko od nestrukturiranega shranjevanja, binarno shranjevanje omogoča shranjevanje različnih vrst podatkov in metapodatkov, kar omogoča spreminjanje strukture primerkov. Takšno shranjevanje pripomore tudi k zmanjšanju števila podatkovnih tabel, v primeru, da se uporabljajo zelo kompleksni in raznovrstni podatki.

Binarno shranjevanje se lahko uporablja tudi v primeru, ko predhodno shema XML še ni znana. Ker lahko s takim načinom hranimo dokumente, ki pripadajo različnim shemam XML lahko izvajamo poizvedovanje po skupnih elementih [6].

Binarno shranjevanje dosežemo z uporabo opcije »STORE AS BINARY« pri kreiranju tabele. Tak način shranjevanja je, kot opis zgoraj možen na dva načina:

- brez registracije sheme XML,
- z registracijo XSD sheme.

### 2.3.1 Brez registracije sheme XML

Način shranjevanja brez registracije sheme XML omogočimo z izdelavo tabele glede na spodnji način:

```
create table TEST_3A
(datoteka varchar2(25), racuni xmltype)
XMLTYPE COLUMN racuni  store AS BINARY XML;
```

Uporaba stavkov DML je enaka kot v poglavju 2.1.1 in se razlikuje samo v uporabi »STORE AS BINARY« dodatka pri kreiranju tabele.

### 2.3.2 Z registracijo sheme XML

Preden kreiramo tabelo na omenjen način, moramo najprej pravilno registrirati shemo XML. Vsebina sheme je popolnoma enaka kot v prejšnjih poglavjih, poimenovali pa smo jo z drugim imenom »RTR_POD_V1.2_BINARY.XSD« in registrirali z dodatno opcijo »REGISTER_BINARY«, kar omogoči binarno shranjevanje dokumentov XML. Pri uporabi te opcije je potrebno obvezno nastaviti vrednost parametra genTypes na vrednost »false«. Primer izvedene registracije sheme:

```
begin
  dbms_xmlschema.registerSchema(schemaURL => 'RTR_POD_V1.2_BINARY.XSD'
                               ,schemaDoc => bfilename('CTEMPXML', 'RTR_POD_V1.2.XSD')
                               ,genTypes => FALSE
                               ,genTables => FALSE
                               ,options => dbms_xmlschema.REGISTER_BINARYXML
                               );
end;
/
```

Ko je shema XML registrirana na zgornji način, lahko izvedemo ukaz za kreiranje tabele, ki vsebuje podatek XMLType, ki hrani dokument XML v binarni obliki. To naredimo s spodnjima ukazoma »CREATE TABLE«:

```
create table TEST_3B
(datoteka varchar2(25), racuni xmltype)
XMLTYPE COLUMN racuni  store AS BINARY XML
 XMLSCHEMA "RTR_POD_V1.2_BINARY.XSD"
    element "RtrPod";

create table TEST_3B_RACUNI
```

```
(datoteka varchar2(25), racun xmltype)
XMLTYPE COLUMN racun  store AS BINARY XML
 XMLSCHEMA "RTR_POD_V1.2_BINARY.XSD"
    element "Tr";
```

Pri uporabi stavka INSERT je bila opažena manjša vendar očitna razlika. V kolikor se s pomočjo stavkov SELECT izvaja branje delov dokumenta (v našem primeru beremo posamezne podatke o transakcijskem računu znotraj podatkov XML) je potrebno uporabiti funkcijo CREATESCHEMABASEDXML. S tem poskrbimo, da se izhodni podatki pretvorijo v XMLType obliko veljavno glede na registrirano shemo XML.

```
insert into test_3b_racuni
  (datoteka, racun)
select t.datoteka, tr.object_value.CREATESCHEMABASEDXML('RTR_POD_V1.2_BINARY.XSD')
from test_3b t,
     table (xmlsequence(extract(t.racuni,'//Tr',
'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"'))) tr;
```

Uporaba SELECT, UPDATE in DELETE stavkov je v tem primeru enaka kot v dosedanjih primerih.

## 2.4 Hibridno shranjevanje

Hibridno shranjevanje pomeni objektno relacijsko shranjevanje z delnim CLOB shranjevanjem. Hibridno shranjevanje, tako vključuje več različnih pristopov omenjenih v prejšnjih poglavjih. V tem primeru, ne bomo prikazali klasičnega hibridnega shranjevanja, vendar bomo predstavili pristop, ki se je v praksi pokazal kot primeren v različnih načinih uporabe. Temelji predvsem na uporabi avtomatsko generiranih tipov pri registraciji sheme XML. V poglavju 2.2 smo že uporabili opcijo genTypes, vendar smo lahko opazili, da so nastali tipi bili avtomatsko poimenovani, kar pomeni, da bi ob brisanju sheme XML in njeni ponovni registraciji, generirani tipi pridobili vedno drugačna imena. Temu in podobnim stvarem se lahko izognemo z uporabo definiranih t.i. »Oracle XML DB« imenskih prostorov (angl. *namespaces*) [14]. Kot prvo moramo v shemo XML vključiti imenski prostor »http://xmlns.oracle.com/xdb«. Sedaj lahko uporabljamo različne anotacije. Naj naštejemo najpomembnejše in uporabljene v našem primeru:

- XDB:SQLNAME
- XDB:SQLTYPE
- XDB:SQLCOLTYPE
- XDB:MAINTAINDDOM
- XDB:MAINTAINORDER.

Uporaba anotacij je prikazana na spodnjem primeru:

```
<xs:element name="RtrPod">
 <xs:annotation>
  <xs:documentation>root</xs:documentation>
 </xs:annotation>
```

```
 <xs:complexType xdb:SQLType="RTRPOD_T" xdb:maintainDOM="false">
  <xs:sequence>
   <xs:element name="Ident" xdb:SQLName="IDENT" type="tr:IdentType"/>
   <xs:element ref="tr:Tr"  xdb:SQLName="TR" xdb:SQLCollType="TR_C"
minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Z uporabo anotacije »xdb:SQLType="RTRPOD_T"« smo določili, da se v primeru glavnega nivoja podatkov izdela objektni tip po imenu RTRPOD_T, ki ima vse podelemente elementa RtrPod. »Xdb:SQLName="TR"«, se enostavno uporablja za definiranje poimenovanja izbranih elementov. V tem primeru smo element Tr poimenovali TR in s tem ročno določili poimenovanje omenjenega elementa. Vsa imena smo poimenovali z velikimi črkami in se s tem izognil težavam pri naslavljanju tipov. »Xdb:SQLCollType« se uporablja za poimenovanje elementov, ki se večkrat ponavljajo. V našem primeru je tak element Tr in z anotacijo »xdb:SQLCollType="TR_C"« smo definiral ime objektnega tipa, ki vsebuje seznam podtipov. Zanimiva anotacija je »xdb:maintainDOM="false"«. V kolikor se te anotacije ne uporablja je njena privzeta vrednost »true«, kar pomeni zagotavljanje t.i. DOM zvestobe in tako privzeto avtomatsko generira t.i. »Positional Descriptor«, ki hrani vse metapodatke, kot so razvrstitev elementov, komentarje, navodila za procesiranje, itn.. Ker v našem primeru tega ne potrebujemo, smo uporabili »xdb:maintainDOM="false"« [15].

Celotna prenovljena vsebina sheme XML je v prilogi, kot dodatek A. Shemo smo poimenovali RTR_POD_V1.2_TYPES.XSD in jo s spodnjim ukazom registrirali pod enakim imenom:

```
begin

  dbms_xmlschema.registerSchema(schemaURL => 'RTR_POD_V1.2_TYPES.XSD'
                               ,schemaDoc => bfilename('CTEMPXML', 'RTR_POD_V1.2_TYPES.XSD')
                               ,genTypes => TRUE
                               ,genTables => FALSE
                               );
end;
/
```

Sedaj si lahko ogledamo imena avtomatsko izdelanih Oracle objektnih tipov:

| TYPE_NAME | TYPECODE | ATTRIBUTES |
|---|---|---|
| NASLOV_C | COLLECTION | 0 |
| IMETNIK_C | COLLECTION | 0 |
| TR_C | COLLECTION | 0 |
| RTRPOD_T | OBJECT | 2 |
| NASLOV_T | OBJECT | 15 |
| IMETNIK_T | OBJECT | 10 |
| TR_T | OBJECT | 16 |
| IDENT_T | OBJECT | 5 |

**Slika 7:** Avtomatično generirani objektni tipi na podlagi uporabe anotacije Oracle XML DB

Neobdelane podatke bomo hranili v obliki CLOB, podobno kot v poglavju 2.1.3. Spodnji ukaz izdela potrebno tabelo:

```
create table test_4
(datoteka varchar2(25), racuni CLOB);
```

Za hranjenje podatkov o posameznem transakcijskem računu pa bomo pri kreiranju tabele uporabili izdelane objektne tipe in sicer objektni tip TR_T:

```
create table TEST_4_RACUNI
(datoteka varchar2(25), racun TR_T);
```

Ker želimo prikazati obdelavo podatkov XML, smo izdelali relativno enostavno funkcijo, katera pretvori podatke XML iz oblike CLOB v podatkovni tip RTRPOD_T, ki je bil izdelan z registracijo sheme XML:

```
create or replace function pretvori_rtrpod(p_content in clob) return rtrpod_t is
  v_content rtrpod_t;
begin
    xmltype(p_content).toobject(v_content,'RTR_POD_V1.2_TYPES.XSD','RtrPod');
  return(v_content);
end;
```

Branje podatkov je sedaj enostavnejše z uporabo generiranih objektnih tipov, saj ni potrebno znanje strukture sheme XML, ampak enostavno uporabljamo generirane objektne tipe in njihove atribute. Pokažimo to na primeru stavka INSERT za zapis posameznih transakcijskih računov v tabelo TEST_4_RACUNI. Vsebino smo prebrali iz tabele TEST_4, jo pretvorili v objektni tip RTRPOD_T in zapisali podatke posameznega transakcijskega računa:

```
insert into test_4_racuni
  (datoteka, racun)
select t.datoteka, tr.object_value
from test_4 t, table( pretvori_rtrpod(t.racuni).TR) tr;
```

V zgornjem stavku SELECT smo uporabili funkcijo »table()«, ki je uporabniško definirana funkcija PL/SQL, katera vrne zbirko vrstic [16]. V našem primeru vrne podatke o posameznem transakcijskem računu kot vrstico v tabeli in se uporablja povsod, kjer imamo objektni tip, ki vsebuje zbirko podatkov. Branje oz. izpis podatkov je z uporabo objektnih tipov enostavnejše, saj lahko celotne podatke prikažemo z izvedbo spodnjega relativno enostavnega stavka SELECT:

```
select r.racun.RN,
       r.racun.VR,
       r.racun.SSPRE,
       r.racun.REG,
       r.racun.DSPRE,
       r.racun.DODPRT,
       r.racun.DZAPRT,
       r.racun.IBAN,
       r.racun.IMET,
       r.racun.ENO,
       r.racun.VIR,
       r.racun.MATICNAPPS,
       r.racun.POPOLNOIMERACUNA,
```

```
        r.racun.KRATKOIMERACUNA,
        r.racun.NAZIVPP,
        im.DAVCNA,
        im.MATSUB,
        im.IDTUJ,
        im.DRZ,
        im.PRORUP,
        im.POPOLNOIME,
        im.KRATKOIME,
        im.IME,
        im.PRIIMEK,
        n.SIFTIPNASLOVA,
        n.SIFDRZAVA,
        n.SIFOBCINA,
        n.SIFPOSTA,
        n.SIFNASELJE,
        n.SIFULICA,
        n.STHISNA,
        n.DODATEK,
        n.SIFHSMID,
        n.DRZAVA,
        n.OBCINA,
        n.POSTA,
        n.ULICA,
        n.NASELJE,
        n.TIPNASLOVA
from test_4_racuni r,
     table(r.racun.IMETNIK) im,
     table(im.NASLOV) n;
```

Enostavnejše je tudi ažuriranje in seveda brisanje podatkov. Kot smo v poglavju 2.1.1 pokazali primer stavka UPDATE in DELETE pri uporabi XMLType podatkov, naj sedaj pokažemo še stavek UPDATE, ki ga izvedemo v tem primeru:

```
update test_4_racuni r
set r.racun.KRATKOIMERACUNA = 'TEST UPDATE'
where r.racun.RN = '401415847510272';
```

Primer DELETE stavka pa izgleda tako:

```
delete  test_4_racuni r where r.racun.RN = '401415847510272';
```

V kolikor bi želeli imeti podatke pretvorjene nazaj v obliko XML, to storimo z uporabo konstruktorja XMLType, ki kot vhodni parameter sprejme vsebino objekta, ime registrirane sheme XML in ime elementa znotraj sheme. Z uporabo funkcije getClobVal() nato pretvorimo podatkovni tip XMLType v podatkovni tip CLOB:

```
select xmltype(r.racun,'RTR_POD_V1.2_TYPES.XSD', 'Tr').getClobVal()
 from test_4_racuni r
  where r.racun.RN = '401415847510272';
```

Iz zgornjih primerov je razvidno, da se pri takem načinu obdelave podatkov XML izognemo jeziku XPATH, uporabi extract(), extractvalue() in podobnih funkcij. Če uporabimo objektne tipe v sami kodi PL/SQL je koda lažje berljiva, saj je uporaba enaka, kot če bi uporabljali lastne izdelane objektne tipe ali objekte tipa varray ali nested table. Performančne prednosti oz. slabosti pa pokaže izvedena primerljivost vseh omenjenih načinov.

# 3. Primerjava

Izvedli smo primerjavo uporabe vseh zgoraj prikazanih različnih načinov zajema in obdelave podatkov XML.

Posamezne teste smo poimenovali na podoben način kot smo poimenovali tabele, ki smo jih izdelovali pri prikazovanju posameznih načinov in sicer:

- Test 1A – nestrukturirano shranjevanje z uporabo podatkovnega tipa XMLType,
- Test 1B – nestrukturirano shranjevanje z uporabo podatkovnega tipa XMLType, na podlagi registrirane sheme XML,
- Test 1C – nestrukturirano shranjevanje z uporabo podatkovnega tipa CLOB,
- Test 2 – strukturirano shranjevanje z uporabo podatkovnega tipa XMLType,
- Test 3A – binarno shranjevanje z uporabo podatkovnega tipa XMLType,
- Test 3B – binarno shranjevanje z uporabo podatkovnega tipa XMLType, na podlagi registrirane sheme XML,
- Test 4 – hibridno shranjevanje z uporabo objektnih tipov izdelanih na podlagi registrirane sheme XML.

Primerjava je razdeljena na dva dela in sicer:

- primerjava hitrosti izvajanja stavkov DML,
- primerjava pretvorbe podatkov v različne tipe podatkov, ki so uporabljeni pri posameznem testu.

## 3.1 Okolje

Testiranje je bilo izvedeno na razvojnem strežniku, katerega osnovne karakteristike so:

- Intel Xeon E5504 2.00 GHz
- 10 GB RAM
- Microsoft Windows Server 2003 R2 x64 Edition
- Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production

Menimo, da podrobne specifikacije strojne opreme niso ključne, saj nas predvsem zanima performančno razmerje med prikazanimi testi, ne pa dejanski rezultati, ki so lahko popolnoma različni na računalniku z drugačnimi specifikacijami.

Ker se na podatkovni bazi Oracle ves čas izvajajo sistemske obdelave, tudi ni možno natančno določiti časovne rezultate, izvajanje samih testov pa je bilo izvedeno v času najmanjše obremenitve podatkovne baze Oracle oz. celotnega strežnika, ko na sistemu ni potekala nobena večja obdelava, kot je npr. arhiviranje. Če je kateri izmed testov vidno izstopal, smo test ponovili in prejšnjega izločili, če je bilo dokazano, da je izstopal zaradi aktivnosti določene obdelave, ki se je sprožila v tistem trenutku.

## 3.2 Primerjava hitrosti izvajanja stavkov DML

Ker nas pri primerjavi hitrosti izvajanja stavkov DML zanima predvsem porabljen CPU čas in pa dejansko porabljen čas, smo za merjenje le tega uporabili ponujen Oracle paket DBMS_UTILITY in njegovo funkcijo GET_CPU_TIME. Funkcija vrne trenuten CPU čas v stotinkah sekunde natančno [17].

Testiranje je bilo izvedeno na obstoječih zapisih v tabelah. Za testne namene so bile vse tabele s podatki o transakcijskih računih (npr. tabele TEST_1A_RACUNI, TEST_1B_RACUNI,…) napolnjene s 10.000 testnimi podatki. Naključno je bilo izbranih 100 zapisov nad katerimi smo izvajali stavke DML in merili čase. Test, smo za vsak korak ponovili 10-krat in izmerili povprečen čas. Ker smo pri samem testiranju opazili, da je dejanska razlika med omenjenimi testi minimalna oz. težko izmerljiva, smo za primerjavo časov, vzeli čas, ki je potreben za obdelavo vseh 100 naključno izbranih zapisov in ne le enega. Funkcija za računanje povprečnega časa je tako enostavna:

$$t(povprečje)=\frac{\sum_{p=1}^{10}(\sum_{n=1}^{100}(t_n\ konec - t_n začetek))}{10}$$

Pred samo izvedbo testiranja so bili postavljeni tudi indeksi na vseh tabelah, katere smo neposredno uporabili pri izvajanju stavkov SELECT, UPDATE in DELETE. Ukazi, ki so bili izvedeni za postavitev indeksov so prikazani spodaj:

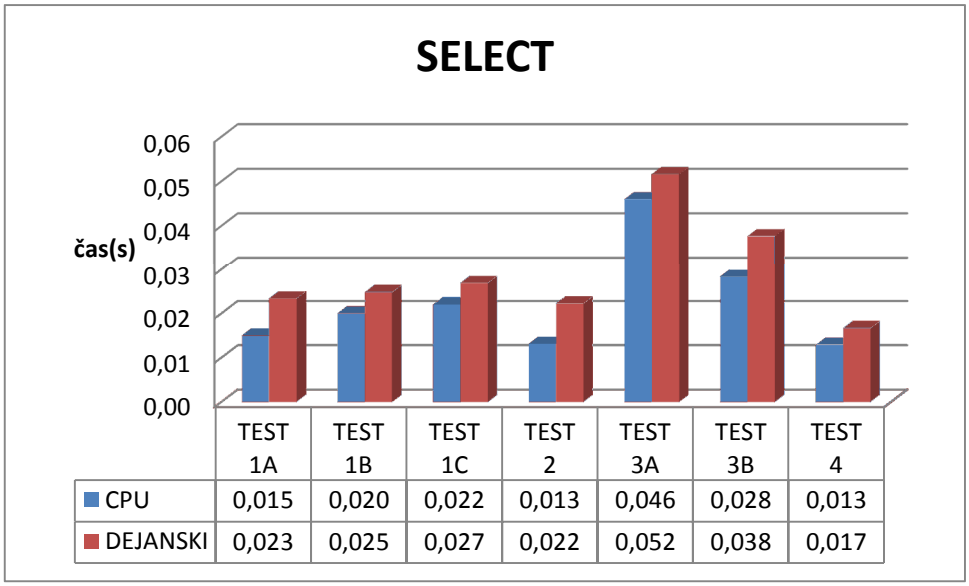
```
CREATE INDEX IND_TEST1A_RN ON test_1A_racuni
(extractValue(racun, '//@rn', 'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"'));

CREATE INDEX IND_TEST1B_RN ON test_1B_racuni
(extractValue(racun, '//@rn', 'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"'));

CREATE INDEX IND_TEST1C_RN ON test_1C_racuni
(extractValue(xmltype(racun), '//@rn', 'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-
20100601"'));

CREATE INDEX IND_TEST2_RN ON test_2_racuni
(extractValue(racun, '//@rn', 'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"'));

CREATE INDEX IND_TEST3A_RN ON test_3A_racuni
```

```
(extractValue(racun, '//@rn', 'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"'));

CREATE INDEX IND_TEST3B_RN ON test_3B_racuni
(extractValue(racun, '//@rn', 'xmlns="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"'));

CREATE INDEX IND_TEST4_RN ON test_4_racuni
(racun.RN);
```

Opazimo lahko, da se kreiranje indeksov razlikuje le v primeru tabele TEST_1C_RACUNI, kjer smo morali vsebino v obliki CLOB pretvoriti v obliko XMLType in v primeru TEST_4_RACUNI, kjer smo lahko poimensko določili polje za katerega kreiramo indeks.

### 3.2.1 Rezultati in ugotovitve

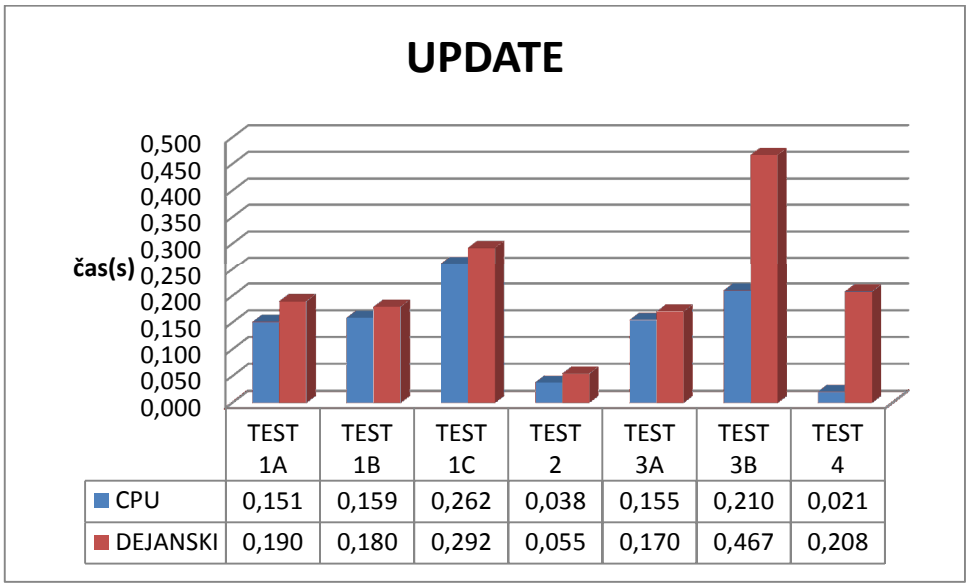
Primerjava izvedbe stavka SELECT je prikazana s spodnjim grafom. Graf nam pokaže, da so si vsi testi med seboj precej enaki, z izjemo uporabe binarnega shranjevanja, ki je približno 2 - 3 krat počasnejši od najhitrejše izvedenega, ki je pripadal testu 4.




| | TEST 1A | TEST 1B | TEST 1C | TEST 2 | TEST 3A | TEST 3B | TEST 4 |
|---|---|---|---|---|---|---|---|
| CPU | 0,015 | 0,020 | 0,022 | 0,013 | 0,046 | 0,028 | 0,013 |
| DEJANSKI | 0,023 | 0,025 | 0,027 | 0,022 | 0,052 | 0,038 | 0,017 |

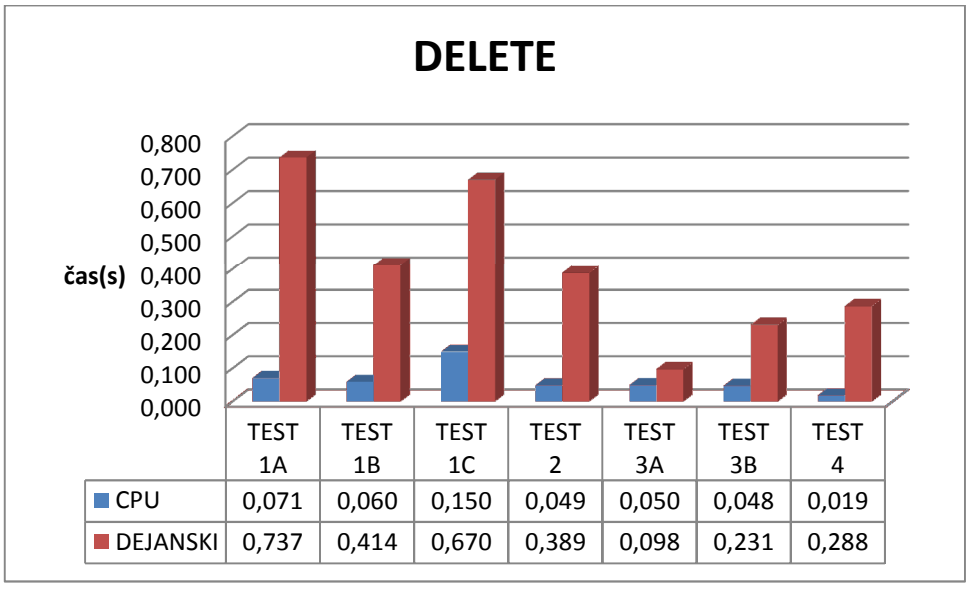
**Slika 8:** Grafičen prikaz primerjave hitrosti izvajanja stavka SELECT

Časovna razlika pri izvajanju stavka UPDATE je prikazana s spodnjim grafom. Vidimo lahko, da največ dejansko porabljenega časa zahteva binarno shranjevanje z uporabo podatkovnega tipa XMLTYPE, registriranega na podlagi sheme XML. Največja CPU poraba pa pripada shranjevanju v obliki CLOB, kar pa zahteva ažuriranje celotnega podatka, shranjenega v podatkovni tip CLOB in ne le dela podatka, kot pri ostalih primerih. Presenetljivo majhna poraba pa je opažena pri uporabi Oracle objektnih tipov generiranih na podlagi sheme XML.



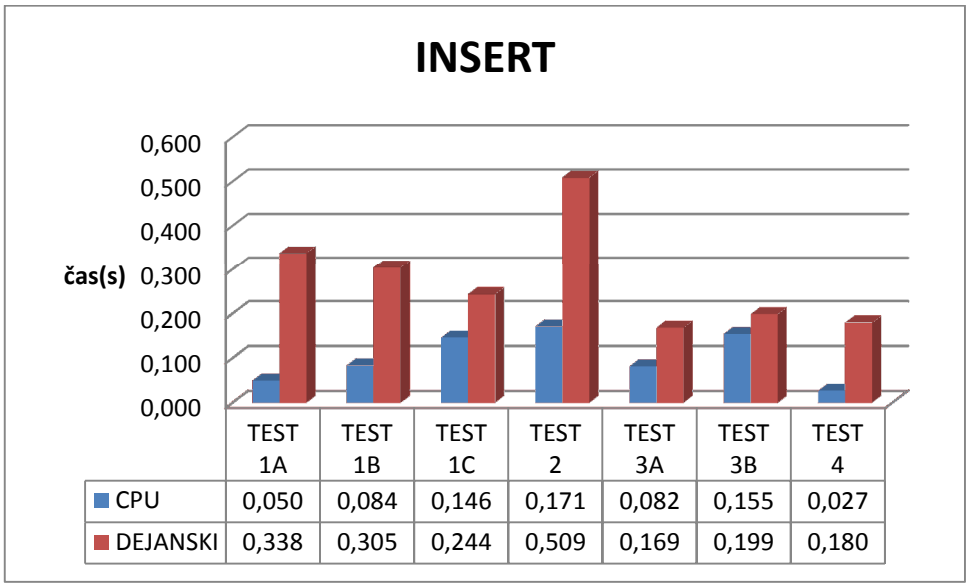

**Slika 9:** Grafičen prikaz primerjave hitrosti izvajanja stavka UPDATE

Stavek DELETE pokaže drugačne rezultate. Največja poraba CPU časa ponovno pripada nestrukturiranemu shranjevanju, medtem ko so si ostali med seboj relativno podobni in so vsi CPU časi doseženi pod 0,1 sekundo.




**Slika 10:** Grafičen prikaz primerjave hitrosti izvajanja stavka DELETE

Izvajanje INSERT stavkov je ponovno potrdilo hitrost pri uporabi objektnih tipov, kjer je bil dosežen rezultat krepko pod 0,1 sekundo in je bilo tako približno 6 krat hitrejše kot uporaba strukturiranega shranjevanja, ki je zahtevalo največ CPU časa.




| | TEST 1A | TEST 1B | TEST 1C | TEST 2 | TEST 3A | TEST 3B | TEST 4 |
|---|---|---|---|---|---|---|---|
| CPU | 0,050 | 0,084 | 0,146 | 0,171 | 0,082 | 0,155 | 0,027 |
| DEJANSKI | 0,338 | 0,305 | 0,244 | 0,509 | 0,169 | 0,199 | 0,180 |

**Slika 11:** Grafičen prikaz primerjave hitrosti izvajanja stavka INSERT

## 3.3 Primerjava pretvorbe podatkov

Primerjavo pretvorbe podatkov v različne podatkovne tipe smo izvajali na drugačen način in merili več spremenljivk. Odločili smo se za uporabo t.i. možnosti TRACE, ki jo ponuja podatkovna baza Oracle. »SQL trace facility« in »TKPROF« sta osnovna orodja za performančno diagnostificiranje aplikacij na podatkovni bazi Oracle [18].

Z uporabo TKPROF orodja je mogoče meriti naslednje statistike:

- število (angl. *COUNT*) – skupno število kolikokrat je bila izjava oz. ukaz razčlenjen, izveden, ali dosežen (angl. *fetched*),
- CPU – skupen čas CPU porabe v sekundah,
- porabljen čas (angl. *ELAPSED*) – skupno porabljen čas v sekundah,
- diskovno polje (angl. *DISK*) – skupno število fizično prebranih podatkovnih blokov,
- poizvedba (angl. *QUERY*) – skupno število pridobljenega medpomnilnika (angl. *buffer*) v konsistentnem načinu (angl. *consistent mode*),
- veljaven (angl. *CURRENT*) – skupno število pridobljenega medpomnilnika v veljavnem načinu (angl. *current mode*) [19].

Prikazali bomo le nekatere izmed njih in sicer CPU čas, diskovno polje in skupno število pridobljenega medpomnilnika kot logični vhod/izhod (angl. *logical I/O*). »COUNT« je v vseh primerih število dokumentov, ki dosega vrednost 2 zaradi tega, ker je vsak ukaz enkrat razčlenjen in enkrat izvršen. Dejansko porabljen čas ni posebej prikazan, saj nas zanima zgolj razlika porabljenega CPU časa pri posameznih primerih. Dejanski čas pa ima enako razmerje in je odvisen predvsem od obdelav, ki trenutno potekajo na sami podatkovni bazi ali strežniku.

Ker je bilo pri testiranju opaženo, da večjih razlik pri pretvorbi relativno majhnih podatkov XML ni, smo v testiranje vključili še različno velike dokumente XML, ki vsebujejo od enega do 10.000 podatkov o transakcijskih računih. Vsebino datotek smo shranili v tabelo na podatkovno bazo Oracle v obliko CLOB, tako, da so vhodni podatki za pretvorbo enostavno dostopni vsem testnim primerom. Vsebina datotek v tabeli je tako različnih velikosti in sicer:

- testna datoteka »1« – 1 KB
- testna datoteka »10« – 7 KB
- testna datoteka »100« – 65 KB
- testna datoteka »1000« – 640 KB
- testna datoteka »10000« – 6.415 KB

Posamezen test je vseboval naslednje korake:

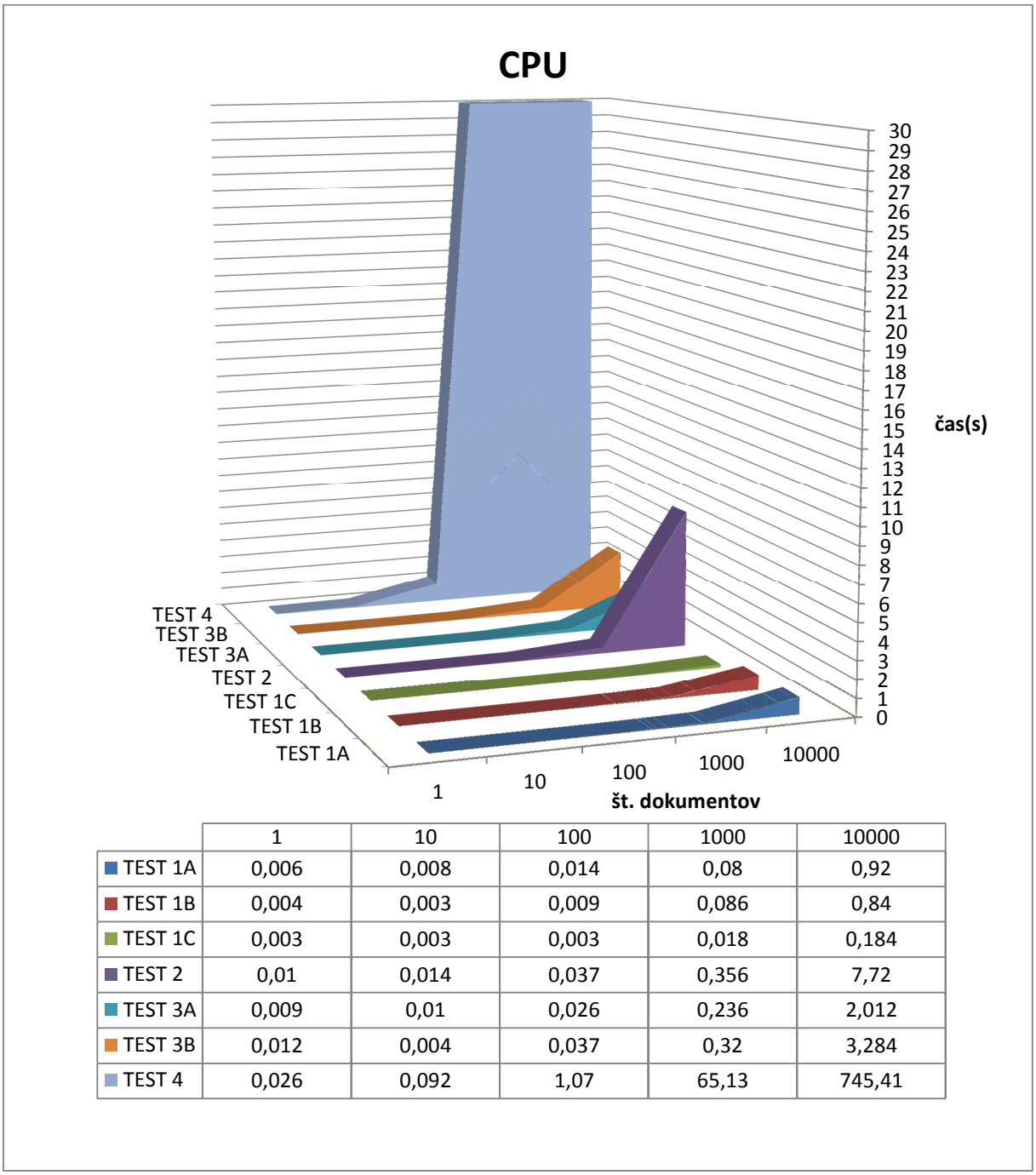
- branje podatkov iz tabele v obliki CLOB, enak korak v vseh primerih,
- pretvorbo podatkov iz podatkovnega tipa CLOB v določen tip, glede na omenjene teste (Test 1A - XMLTYPE, Test 1B – XMLTYPE na podlagi sheme XML, …),
- zapis v tabelo, glede na omenjene teste (Test 1A – zapis v tabelo TEST_1A_RACUNI, Test 1B – zapis v tabelo TEST_1B_RACUNI).

Posamezen test smo ponovili 10 krat in izračunali povprečno vrednost s pomočjo spodnje formule:

$$t(povprečje)=\frac{\sum_{p=1}^{10}(t_n\ konec - t_n začetek)}{10}$$

### 3.3.1 Rezultati in ugotovitve

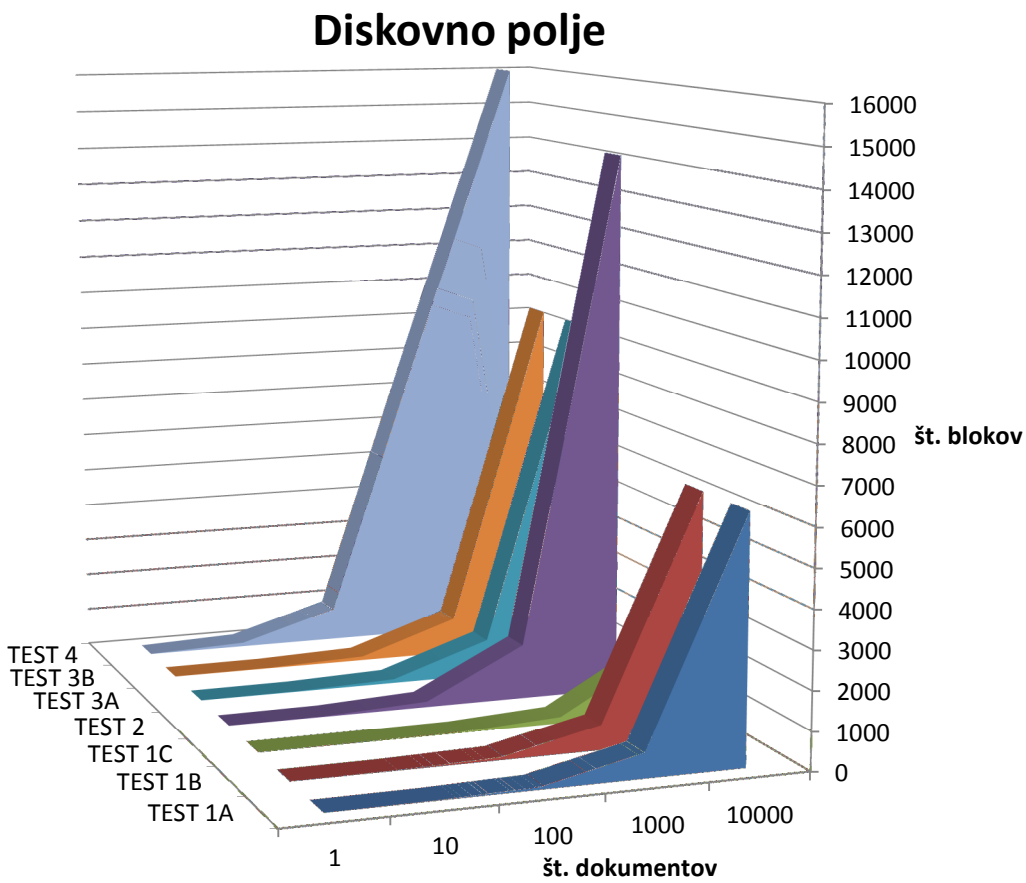
S spodnjim grafom je prikazan CPU čas, ki je potreben za pretvorbo podatkov v določene podatkovne tipe. Hitro lahko vidimo, da se uporaba objektnih tipov v primeru večjih datotek obnese bistveno slabše oz. za pretvorbo porabi veliko CPU časa. Ugotovljeno je bilo, da v primeru večjih datotek, t.j. več kot 5MB, pretvorba porabi ogromno CPU časa. V času testiranja je bilo ugotovljeno, da v primeru, ko je vsebina večja kot 10 MB postane pretvorba na tak način popolnoma neuporabna in nezanesljiva, zaradi prevelike porabe CPU časa v primerjavi z drugimi načini. Drugi najslabši rezultat pripada uporabi strukturiranega shranjevanja, kar je tudi logično, saj gre pri pretvorbi dejansko za pretvarjanje vsebine v obliki CLOB v strukturirano obliko, ki ima več nivojev (npr. struktura za podatke o imetniku, struktura za podatke o naslovu, itn.). Binarno shranjevanje je dobilo drugi najboljši rezultat, prvo mesto pa zaseda nestrukturirano shranjevanje, saj gre tukaj dejansko za pretvorbo iz CLOB v CLOB obliko oz. XMLTYPE obliko shranjeno v CLOB obliki.




| | 1 | 10 | 100 | 1000 | 10000 |
|---|---|---|---|---|---|
| TEST 1A | 0,006 | 0,008 | 0,014 | 0,08 | 0,92 |
| TEST 1B | 0,004 | 0,003 | 0,009 | 0,086 | 0,84 |
| TEST 1C | 0,003 | 0,003 | 0,003 | 0,018 | 0,184 |
| TEST 2 | 0,01 | 0,014 | 0,037 | 0,356 | 7,72 |
| TEST 3A | 0,009 | 0,01 | 0,026 | 0,236 | 2,012 |
| TEST 3B | 0,012 | 0,004 | 0,037 | 0,32 | 3,284 |
| TEST 4 | 0,026 | 0,092 | 1,07 | 65,13 | 745,41 |

**Slika 12:** Grafičen prikaz primerjave porabe CPU

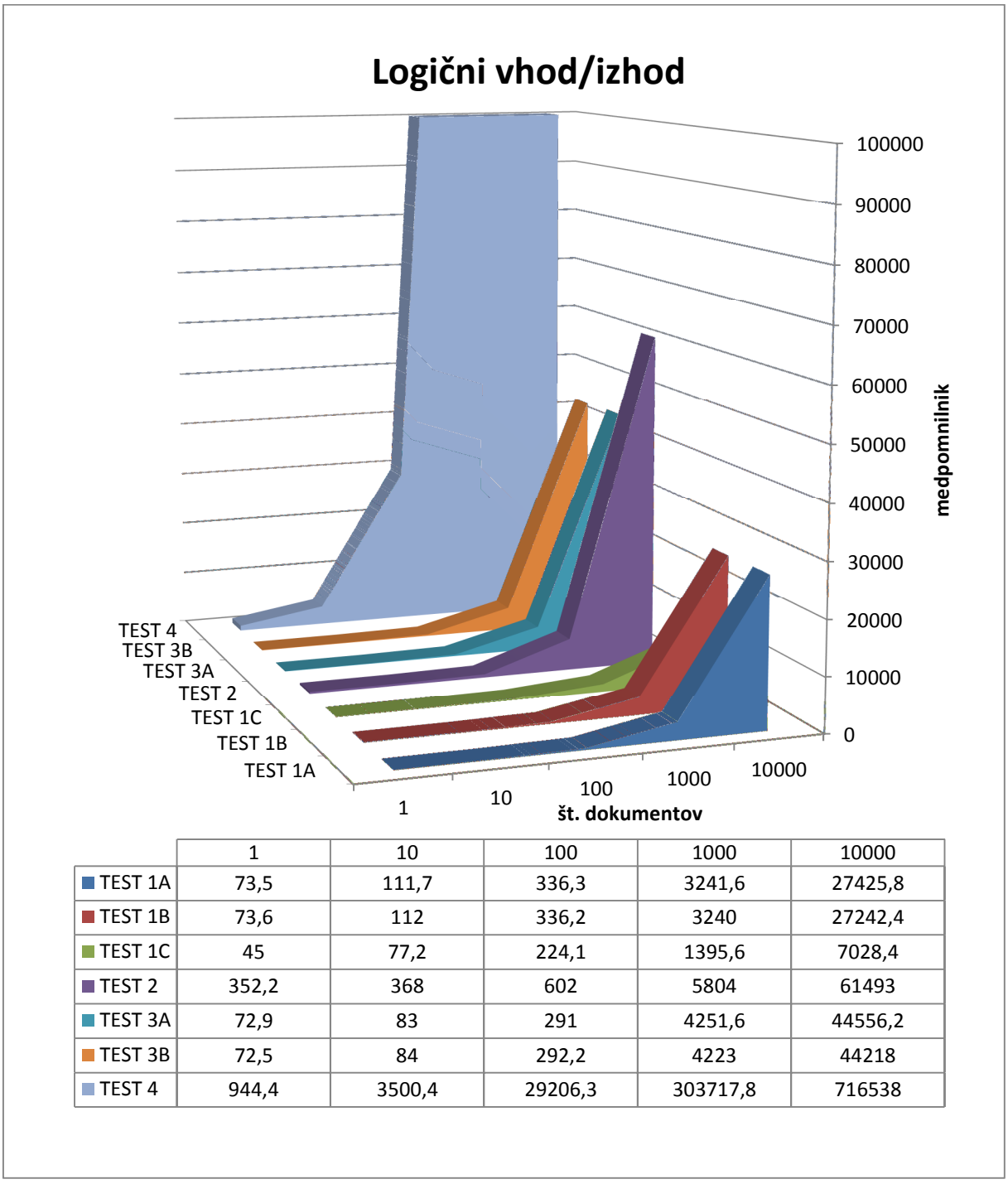
Ob pregledu primerjave števila fizično prebranih podatkovnih blokov, tudi tukaj najslabše mesto zaseda uporaba Oracle objektnih tipov. Razlika med ostalimi testi je občutno manjša. Lestvica od najboljšega do najslabšega je pričakovano podobna kot pri porabi CPU časa, le razlike med posameznimi testi so manjše.




| | 1 | 10 | 100 | 1000 | 10000 |
|---|---|---|---|---|---|
| TEST 1A | 2,1 | 12 | 68,9 | 637,8 | 6326,8 |
| TEST 1B | 2 | 13 | 69,9 | 638,8 | 6327,8 |
| TEST 1C | 2 | 6,1 | 20,9 | 164 | 1592,8 |
| TEST 2 | 2,9 | 19,9 | 145,9 | 1423,8 | 14204 |
| TEST 3A | 2,2 | 13 | 100 | 952,8 | 9472,8 |
| TEST 3B | 2 | 14 | 101 | 953,8 | 9473,8 |
| TEST 4 | 2 | 101 | 859 | 8527,8 | 16012 |

**Slika 13:** Grafičen prikaz primerjave št. prebranih blokov na diskovnem polju

Pregled števila uporabljenih logičnih vhodno/izhodnih operacij oz. seštevek celotnega števila uporabljenega medpomnilnika, pokaže skoraj identično sliko kot v primeru CPU porabe in potrdi neustreznost uporabe Oracle objektnih tipov v primeru pretvorbe večje količine podatkov.




| | 1 | 10 | 100 | 1000 | 10000 |
|---|---|---|---|---|---|
| TEST 1A | 73,5 | 111,7 | 336,3 | 3241,6 | 27425,8 |
| TEST 1B | 73,6 | 112 | 336,2 | 3240 | 27242,4 |
| TEST 1C | 45 | 77,2 | 224,1 | 1395,6 | 7028,4 |
| TEST 2 | 352,2 | 368 | 602 | 5804 | 61493 |
| TEST 3A | 72,9 | 83 | 291 | 4251,6 | 44556,2 |
| TEST 3B | 72,5 | 84 | 292,2 | 4223 | 44218 |
| TEST 4 | 944,4 | 3500,4 | 29206,3 | 303717,8 | 716538 |

**Slika 14:** Grafičen prikaz primerjave logičnih vhodno/izhodnih operacij

Večja razlika pri izvajanju testa 4, je predvsem posledica kombinacije jezika SQL in jezika PL/SQL, zaradi uporabe funkcije PRETVORI_RTRPOD pri izvajanju INSERT stavka. To bi se dalo rešiti na drugačen način, npr. z deklariranjem spremenljivk in izvajanje celotnega zajema v PLSQL programu.

# 4. Zaključek

V diplomski nalogi je bilo predstavljenih 7 različnih načinov uporabe in obdelave podatkov XML. Vsakega izmed teh načinov bi se seveda dalo še drugače prikazati oz. obravnavati in sicer z dodatnimi opcijami, ki jih ponuja Oracle, npr. uporaba XMLType indeksov, različnih opcij pri shranjevanju podatkov oz. pri ukazu »CREATE TABLE«, različne vrste optimizacij, generiranje podatkov XML iz obstoječih podatkov, itd.

Poudarek pri opisu različnih načinov je bil na uporabi objektnih tipov, kateri so generirani na podlagi sheme XML. Ugotovljeno je bilo, da se pri uporabi stavkov DML načini bistveno ne razlikujejo, in se je omenjeni način izkazal kot dober in performančno hiter. Popolnoma nasprotno je pokazala primerjava pretvorbe večje vsebine podatkov na tak način, saj se je tak način izkazal za neuporabnega. V primeru večje količine podatkov je tudi v ostalih primerih predlagan drugačen način obdelave podatkov, saj vsi zgornji primeri uporabljajo za pretvorbo oz. branje t.i. razčlenjevalnik DOM (angl. parser). Ta za pretvorbo podatkov potrebuje celoten dokument, razčlenjevalnik SAX pa lahko bere le del celotnega.

Kljub izvedenemu prikazu razlik med vsemi različnimi načini, pa naj poudarimo, da ima vsak način svoje prednosti in slabosti, katere moramo upoštevati preden se odločimo za katerikoli način. Nekatere prednosti in slabosti smo že zapisali pri prikazu posameznih načinov, glavne prednosti in slabosti pa so prikazane v spodnji tabeli:

| | **Strukturirano shranjevanje** | **Binarno shranjevanje** | **Nestrukturirano shranjevanje** |
|---|---|---|---|
| Izvajanje DML operacij | Izjemno hitro z uporabo ustreznih indeksov | Hitro | Počasno pri obravnavi večjih dokumentov |
| Prostorska učinkovitost | Zelo učinkovito | Učinkovito | Neučinkovito |
| Podatkovna fleksibilnost | Omejena fleksibilnost | Fleksibilno | Fleksibilno |
| Fleksibilnost XML sheme | Shranjeni so lahko le dokumenti, ki ustrezajo določeni shemi XML | Z uporabo dodatnih opcij, lahko shranimo XML dokumente, ki pripadajo različnim shemam XML | Dokumenti so lahko odvisni ali pa neodvisni od določene sheme XML |
| XML zvestoba (ohranjanje XML podatkov) | Delno ohranjanje | Delno ohranjanje | Ohranjanje |

| | | | |
|---|---|---|---|
| Podpora indeksiranju | B-tree, bitmap, Oracle Text, XMLIndex, in funkcijski indeksi | XMLIndex, funkcijski in Oracle Text indeksi | XMLIndex, funkcijski in Oracle Text indeksi |
| Optimizirano upravljanje s spominom | Možna optimizacija | Možna optimizacija | Ni možna, brez uporabe DOM |
| Preverjanje podatkov pri zapisovanju | Delno preverjanje | Preverjanje v celoti | Delna preverjanje – pri uporabi sheme XML |
| Razdelitev diska (angl. partitioning) | Možno | Možno na virtualnih stolpcih | Možno v primeru relacijskih stolpcev |
| Tokovna replikacija (angl. Streams-based replication) | Ni možno | Ni možno | Možno |
| Kompresija (angl. compression) | Možno | Možno z uporabo SecureFile | Ni možno |

**Slika 15:** Glavne prednosti in slabosti glede na različne možnosti shranjevanja [6]

Prednosti in slabosti hibridnega shranjevanja, kot je bil prikazan v diplomski nalogi so predvsem prednosti in slabosti strukturiranega shranjevanja. Očitno se razlikuje le v primeru preverjanja vhodnih podatkov, kjer gre za celotno preverjanje podatkov ob shranjevanju, saj je podatke potrebno pretvoriti v objektne tipe, ki so bili zgrajeni na osnovi registrirane sheme XML. Bistvena razlika pri uporabi takega tipa je tudi ta, da se nam ni potrebno več ukvarjati s strukturo dokumenta XML, saj imamo za to sedaj na voljo objektne tipe. Tako se nam ni potrebno ukvarjati s tem na katerem nivoju je nek podatek, niti s tem kakšnega tipa je ta podatek. Slabost pa se pokaže v tem, da zahteva bistveno večjo natančnost na začetku uporabe, saj je potrebno ustrezno prilagoditi shemo XML in kar je bistveno, shema XML mora biti zgrajena na način, ki jo podpira podatkovna baza Oracle.

Kot omenjeno je pri obdelavi podatkov XML potrebna najprej natančna analiza problema (kako veliki so dokumenti, ali se bodo dokumenti ažurirali, kakšna bo frekvenca branja podatkov, ali bo potrebno generiranje XML podatkov iz podatkovne baze, itn.) preden se odločimo za katerikoli način.

# 5. Priloge

## 5.1 Dodatek A

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xdb="http://xmlns.oracle.com/xdb"
xmlns:tr="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ajpes.si/xml_sheme/rtr/rtr_pod-20100601"
elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="Tr" type="tr:TrType"/>
 <xs:element name="Naslov" type="tr:NaslovType"/>
 <xs:element name="RtrPod">
                <xs:annotation>
                        <xs:documentation>root</xs:documentation>
                </xs:annotation>
                <xs:complexType xdb:SQLType="RTRPOD_T" xdb:maintainDOM="false">
                        <xs:sequence>
                                <xs:element name="Ident" xdb:SQLName="IDENT"
type="tr:IdentType"/>
                                <xs:element ref="tr:Tr"  xdb:SQLName="TR"
xdb:SQLCollType="TR_C" minOccurs="0" maxOccurs="unbounded"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
        <xs:element name="Drzava" xdb:SQLName="DRZAVA">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:maxLength value="80"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
        <xs:element name="Ime" xdb:SQLName="IME">
                <xs:annotation>
                        <xs:documentation>Ime fizicne osebe</xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                        <xs:restriction base="xs:string"/>
                </xs:simpleType>
        </xs:element>
        <xs:element name="KratkoImeRacuna" xdb:SQLName="KRATKOIMERACUNA">
                <xs:simpleType>
```

```
                    <xs:restriction base="xs:string"/>
                </xs:simpleType>
            </xs:element>
            <xs:element name="KratkoIme" xdb:SQLName="KRATKOIME">
                <xs:annotation>
                    <xs:documentation>Kratek naziv poslovnega
subjekta</xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                    <xs:restriction base="xs:string"/>
                </xs:simpleType>
            </xs:element>
            <xs:element name="NazivPp" xdb:SQLName="NAZIVPP" type="xs:string"/>
            <xs:element name="Naselje" xdb:SQLName="NASELJE">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:maxLength value="40"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="Obcina" xdb:SQLName="OBCINA">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:maxLength value="40"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="Priimek" xdb:SQLName="PRIIMEK">
                <xs:annotation>
                    <xs:documentation>Priimek fizicne osebe</xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                    <xs:restriction base="xs:string"/>
                </xs:simpleType>
            </xs:element>
            <xs:element name="PopolnoIme" xdb:SQLName="POPOLNOIME">
                <xs:annotation>
                    <xs:documentation>Naziv poslovnega subjekta</xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                    <xs:restriction base="xs:string"/>
```

```
            </xs:simpleType>
        </xs:element>
        <xs:element name="Posta" xdb:SQLName="POSTA">
            <xs:annotation>
                <xs:documentation>Kraj poste</xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:maxLength value="40"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="TipNaslova" xdb:SQLName="TIPNASLOVA">
            <xs:annotation>
                <xs:documentation>opis sifre</xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:maxLength value="50"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="Ulica" xdb:SQLName="ULICA">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:maxLength value="40"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:complexType name="IdentType" xdb:SQLType="IDENT_T"
xdb:maintainDOM="false">
            <xs:attribute name="maticnaPorocevalec"
xdb:SQLName="MATICNAPOROCEVALEC" use="required">
                <xs:annotation>
                    <xs:documentation>Identifikacijska (maticna) stevilka
porocevalca .</xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[0-9]{10}"/>
                    </xs:restriction>
```

```
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="steviloRacunov" xdb:SQLName="STEVILORACUNOV"
type="xs:integer" use="optional">
                    <xs:annotation>
                        <xs:documentation>Stevilo racunov v datoteki za
kontrolo.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="datumPriprave" xdb:SQLName="DATUMPRIPRAVE"
type="xs:dateTime" use="required"/>
                <xs:attribute name="maticnaPps" xdb:SQLName="MATICNAPPS"
use="optional">
                    <xs:annotation>
                        <xs:documentation>Maticna stevilka ponudnika placilnih
storitev.</xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="[0-9]{10}"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="fileName" xdb:SQLName="FILENAME"
type="xs:string" use="optional">
                    <xs:annotation>
                        <xs:documentation>Atribut uporablja Ajpes pri prevzemu
datotek prek ftp streznika.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
            </xs:complexType>
            <xs:complexType name="ImetnikType" xdb:SQLType="IMETNIK_T"
xdb:maintainDOM="false">
                <xs:sequence>
                    <xs:element ref="tr:PopolnoIme" xdb:SQLName="POPOLNOIME"
minOccurs="0"/>
                    <xs:element ref="tr:KratkoIme" xdb:SQLName="KRATKOIME"
minOccurs="0"/>
                    <xs:element ref="tr:Ime" xdb:SQLName="IME" minOccurs="0"/>
                    <xs:element ref="tr:Priimek" xdb:SQLName="PRIIMEK"
minOccurs="0"/>
```

```
                    <xs:element ref="tr:Naslov" xdb:SQLName="NASLOV"
xdb:SQLCollType="NASLOV_C" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="davcna" xdb:SQLName="DAVCNA" use="optional">
                    <xs:annotation>
                        <xs:documentation>Davcna stevilka
imetnika</xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="[0-9]{8}"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="matSub" xdb:SQLName="MATSUB" use="optional">
                    <xs:annotation>
                        <xs:documentation>Maticna st. poslovnega subjekta
imetnika</xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="[0-9]{10}"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="idTuj" xdb:SQLName="IDTUJ" use="optional">
                    <xs:annotation>
                        <xs:documentation>Identifikator, ko je imetnik
tujec</xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:maxLength value="30"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="drz" xdb:SQLName="DRZ" use="optional">
                    <xs:annotation>
                        <xs:documentation>Drzava imetnika</xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
```

```
                        <xs:restriction base="xs:string">
                            <xs:pattern value="[0-9]{3}"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="prorup" xdb:SQLName="PRORUP" use="optional">
                    <xs:annotation>
                        <xs:documentation>Sifra proracunskega
uporabnika.</xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="[0-9]{5}"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:complexType>
            <xs:complexType name="NaslovType" xdb:SQLType="NASLOV_T"
xdb:maintainDOM="false">
                <xs:sequence>
                    <xs:element ref="tr:Drzava" xdb:SQLName="DRZAVA"
minOccurs="0"/>
                    <xs:element ref="tr:Obcina" xdb:SQLName="OBCINA"
minOccurs="0"/>
                    <xs:element ref="tr:Posta" xdb:SQLName="POSTA" minOccurs="0"/>
                    <xs:element ref="tr:Ulica" xdb:SQLName="ULICA" minOccurs="0"/>
                    <xs:element ref="tr:Naselje" xdb:SQLName="NASELJE"
minOccurs="0"/>
                    <xs:element ref="tr:TipNaslova" xdb:SQLName="TIPNASLOVA"
minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="sifTipNaslova" xdb:SQLName="SIFTIPNASLOVA"
use="optional">
                    <xs:annotation>
                        <xs:documentation>N2; sifra tipa naslova (01-stalni, 02-
zacasni...)</xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="[0-9]{2}"/>
                        </xs:restriction>
```

```
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="sifDrzava" xdb:SQLName="SIFDRZAVA"
use="optional">
                    <xs:annotation>
                        <xs:documentation>N3 sifra drzave 705-
slovenija</xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="[0-9]{3}"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="sifObcina" xdb:SQLName="SIFOBCINA"
use="optional">
                    <xs:annotation>
                        <xs:documentation>N3 stevilka obcine</xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="[0-9]{3}"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="sifPosta" xdb:SQLName="SIFPOSTA" use="optional">
                    <xs:annotation>
                        <xs:documentation>N4 stevilka poste </xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="[0-9]{4}"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="sifNaselje" xdb:SQLName="SIFNASELJE"
use="optional">
                    <xs:annotation>
                        <xs:documentation>N3 sifra naselja iz
RPE</xs:documentation>
                    </xs:annotation>
```

```
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[0-9]{3}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="sifUlica" xdb:SQLName="SIFULICA" use="optional">
                <xs:annotation>
                    <xs:documentation>N4</xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[0-9]{4}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="stHisna" xdb:SQLName="STHISNA" use="optional">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:maxLength value="3"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="dodatek" xdb:SQLName="DODATEK" use="optional">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:length value="1"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="sifHsmid" xdb:SQLName="SIFHSMID" use="optional">
                <xs:annotation>
                    <xs:documentation>HSMID iz RPE</xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[0-9]{8}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:complexType>
```

```
        <xs:complexType name="TrType" xdb:SQLType="TR_T"
xdb:maintainDOM="false">
                <xs:sequence>
                        <xs:element ref="tr:PopolnoImeRacuna"
xdb:SQLName="POPOLNOIMERACUNA" minOccurs="0"/>
                        <xs:element ref="tr:KratkoImeRacuna"
xdb:SQLName="KRATKOIMERACUNA" minOccurs="0"/>
                        <xs:element ref="tr:NazivPp" xdb:SQLName="NAZIVPP"
minOccurs="0"/>
                        <xs:element name="Imetnik" xdb:SQLName="IMETNIK"
xdb:SQLCollType="IMETNIK_C" type="tr:ImetnikType" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="rn" xdb:SQLName="RN" use="required">
                        <xs:annotation>
                                <xs:documentation>Stevilka transakcijskega
racuna</xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                                <xs:restriction base="xs:string">
                                        <xs:pattern value="[0-9]{15}"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="vr" xdb:SQLName="VR" use="required">
                        <xs:annotation>
                                <xs:documentation>Oznaka vrste racuna</xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                                <xs:restriction base="xs:string">
                                        <xs:length value="1"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="sSpre" xdb:SQLName="SSPRE" use="required">
                        <xs:annotation>
                                <xs:documentation>Sifra vrste spremembe</xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                                <xs:restriction base="xs:string">
                                        <xs:maxLength value="1"/>
```

```
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="reg" xdb:SQLName="REG" use="required">
                    <xs:annotation>
                        <xs:documentation>Oznaka dela registra (P ali
F)</xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:maxLength value="1"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="dSpre" xdb:SQLName="DSPRE" type="xs:date"
use="optional">
                    <xs:annotation>
                        <xs:documentation>Datum veljavnosti
spremembe</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="dOdprt" xdb:SQLName="DODPRT" type="xs:date"
use="optional">
                    <xs:annotation>
                        <xs:documentation>Datum odprtja racuna</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="dZaprt" xdb:SQLName="DZAPRT" type="xs:date"
use="optional">
                    <xs:annotation>
                        <xs:documentation>Datum zaprtja racuna</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="iban" xdb:SQLName="IBAN" use="optional">
                    <xs:annotation>
                        <xs:documentation>Za Slovenijo SI56</xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:length value="4"/>
                        </xs:restriction>
```

```
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="imet" xdb:SQLName="IMET" use="optional">
                <xs:annotation>
                    <xs:documentation>Stevilo imetnikov racunov. Obvezno kadar
je razlicno od 1.</xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                    <xs:restriction base="xs:integer">
                        <xs:minInclusive value="2"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="eno" xdb:SQLName="ENO" use="optional">
                <xs:annotation>
                    <xs:documentation>Evidenca neporavnanih
obveznosti</xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:maxLength value="1"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="vir" xdb:SQLName="VIR" use="optional">
                <xs:annotation>
                    <xs:documentation>Izvor spremembe.</xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:length value="1"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="maticnaPps" xdb:SQLName="MATICNAPPS"
use="optional">
                <xs:annotation>
                    <xs:documentation>Maticna stevilka
banke.</xs:documentation>
                </xs:annotation>
                <xs:simpleType>
```

```
                        <xs:restriction base="xs:string">
                            <xs:pattern value="[0-9]{10}"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:complexType>
            <xs:element name="PopolnoImeRacuna"
xdb:SQLName="POPOLNOIMERACUNA">
                <xs:simpleType>
                    <xs:restriction base="xs:string"/>
                </xs:simpleType>
            </xs:element>
</xs:schema>
```

# 6. Viri in literatura

[1] Jason Price, Oracle Database 11g SQL, 2007, Str. 2, Poglavje 1

[2] R. Greenwald, R. Stackoviak, J. Stern - Oracle Essentials, 2007, Str. 10, Poglavje 1

[3] E. Vlist, XML Schema, 2002, Str. 13, Poglavje 1

[4] http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28369/xdb01int.htm

[5] http://download.oracle.com/docs/cd/E11882_01/server.112/e17118/sql_elements001.htm#SQLRF51012

[6] Oracle® XML DB Developer's Guide 11g Release 2 (11.2). Dostopno na: http://download.oracle.com/docs/cd/E11882_01/appdev.112/e16659/xdb01int.htm#i1047170

[7] http://download.oracle.com/docs/cd/E11882_01/appdev.112/e16659/img/adxdb017.gif

[8] http://download.oracle.com/docs/cd/E11882_01/appdev.112/e16659/img/adxdb035.gif

[9] http://www.ajpes.si/Registri/Transakcijski_racuni/Splosno?id=69

[10] http://www.ajpes.si/xml_sheme/pps/rtr_pod_v1.2.xsd

[11] http://download.oracle.com/docs/cd/E11882_01/server.112/e17118/functions019.htm#SQLRF00610

[12] http://www.w3.org/TR/xpath/

[13] http://download.oracle.com/docs/cd/E11882_01/appdev.112/e16760/d_xmlsch.htm#ARPLS377

[14] http://download.oracle.com/docs/cd/B10501_01/appdev.920/a96620/xdb03usg.htm#1037721

[15] http://download.oracle.com/docs/cd/E11882_01/appdev.112/e16659/xdb05sto.htm#ADXDB4484

[16] http://download.oracle.com/docs/cd/E11882_01/appdev.112/e17126/tuning.htm#LNPLS915

[17] http://download.oracle.com/docs/cd/E11882_01/appdev.112/e16760/d_util.htm#ARPLS73243

[18] http://download.oracle.com/docs/cd/E11882_01/server.112/e16638/sqltrace.htm#PFGRF01020

[19] http://download.oracle.com/docs/cd/E11882_01/server.112/e16638/sqltrace.htm#PFGRF01010