Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Simeon Puntar

Pregled in primerjava triplestore podatkovnih baz

diplomsko delo
na univerzitetnem študiju

Mentor: doc. dr. Dejan Lavbič

Ljubljana, 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

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

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

Št. naloge: 01860/2012
Datum: 03.09.2012

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

Kandidat: **SIMEON PUNTAR**

Naslov: **PREGLED IN PRIMERJAVA TRIPLESTORE PODATKOVNIH BAZ**
REVIEW AND COMPARISON OF TRIPLESTORE DATABASES

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Na semantičnem spletu je ena ključnih nalog prihajajočih tehnologij semantična integracija podatkov. Za ta namen se uporabljajo predvsem NoSQL podatkovne baze tipa "graph-based" oz. t.i. triplestores. V okviru diplomske naloge bi bilo potrebno pregledati področje odprtokodnih in komercialnih rešitev ter jih kritično medsebojno primerjati. V primerjavo vključite reprezentativne predstavnike D2RQ, Jena SDB, Jena TDB, Neo4J, Owlim, Sesame in Virtuoso. Pri primerjavi opredelite kriterije, kjer vključite tudi performančno zmogljivost.

Mentor:

doc. dr. Dejan Lavbič

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Simeon Puntar, z vpisno številko **63040139**, sem avtor diplomskega dela z naslovom:

Pregled in primerjava triplestore podatkovnih baz

S svojim podpisom zagotavljam, da:

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

V Ljubljani, dne 14.12.2012 Podpis avtorja:

Zahvala

Zahvaljujem se mentorju doc. dr. Dejanu Lavbiču za pomoč in nasvete pri izdelavi diplomske naloge.

Zahvala gre tudi moji družini in Anji za vso podporo in spodbudo tekom študija.

Kazalo

Seznam uporabljenih kratic

ACID Atomicity, Consistency, Isolation, Durability (atomarnost, konsistentnost, izolacija, trajnost - lastnosti transakcij v relacijski podatkovni bazi)

API Application Programming Interface (aplikacijski programski vmesnik)

JDBC Java Database Connectivity (Java API za dostop do podatkovne baze)

JSON JavaScript Object Notation (format podatkov za izmenjavo strukturiranih dokumentov v spletu)

NoSQL Not Only SQL (oznaka za nerelacijske podatkovne baze)

REST Representational State Transfer (množica arhitekturnih principov za razvoj spletnih storitev)

RDF Resource Description Framework

Sail Storage And Interface Layer

SeRQL Sesame RDF Query Language (poizvedovalni jezik v Sesame podatkovnih bazah)

SPARQL SPARQL Protocol and RDF Query Language (poizvedovalni jezik v nerelacijskih podatkovnih bazah)

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

SUPB Sistem za upravljanje s podatkovno bazo

TURTLE Terse RDF Triple Language

URI Uniform Resource Identifier

Povzetek

Na semantičnem spletu je ena ključnih nalog prihajajočih tehnologij semantična integracija podatkov. Za ta namen se uporabljajo predvsem NoSQL podatkovne baze tipa "graph-based"oziroma t.i. triplestores. Namen diplomske naloge je predstavitev tega področja in medsebojna primerjava rešitev najbolj znanih ponudnikov. Vsebinsko je diplomska naloga sestavljena iz dveh delov. V prvem delu so opredeljeni kriteriji za primerjavo različnih ponudnikov triplestore podatkovnih baz, opis ter primerjava njihovih rešitev po teh kriterijih. Drugi del opisuje problemsko domeno s predvidenimi scenariji uporabe, podrobnosti implementacije v javanski aplikaciji in testiranja. Praktični del obsega implementacijo podatkovnih baz v javanski aplikaciji in testiranje scenarijev uporabe, predvidenih v problemski domeni. Skupaj s predstavitvijo rezultatov so podane tudi ugotovitve o primernosti rešitev za uporabo v podani problemski domeni. Zaključek vsebuje povzetek področja in sklepne ugotovitve o predstavljenih triplestore podatkovnih bazah.

Ključne besede:
semantični splet, NoSQL, triplestore, performančna zmogljivost

Abstract

Semantic data integration is one of the key tasks of emerging technologies on semantic web. Mainly used are graph based databases also known as triplestores. The aim of this thesis is to provide an overview of the area and to compare main triplestore solutions. Thesis consists of two parts. The first part defines criteria for comparison of selected database systems, followed by description of databases systems and their comparison based on defined criteria. The second part describes the problem domain with different use case scenarios, details of Java application implementation and testing. The practical part of thesis consists of the implementation of databases in Java application and testing of scenarios provided in the problem domain. Test results are presented and analyzed. Furthermore, the key findings gained from test results are presented. Conclusion provides summary of the area and final recommendations for use of presented triplestore databases.

Key words:
semantic web, NoSQL, triplestore, performance analysis

Poglavje 1

Uvod

Svetovni splet je sistem medsebojno povezanih dokumentov, ki vsebujejo velike količine podatkov in informacij, do katerih ljudje dostopamo preko računalnikov. Naloga računalnikov je samo dostava in predstavljanje vsebine dokumentov. Usmerjamo jih ljudje, saj računalniki sami niso sposobni interpretacije vsebine dokumentov. Semantični splet je nastal kot rezultat mednarodnega prizadevanja za predstavitev podatkov v obliki, primerni za procesiranje in interpretacijo s strani računalnikov, s čimer bi olajšali iskanje in izmenjavo podatkov med uporabniki spleta.

Na semantičnem spletu informacijo predstavlja izraz, največkrat poimenovan trojček (ang. triple), sestavljen iz treh delov: osebek, predikat in predmet. Za shranjevanje trojčkov se uporabljajo podatkovne baze tipa "graph-based" oziroma t.i. triplestores, ki poleg shranjevanja omogočajo tudi poizvedovanje in modifikacijo podatkov. V primerjavi z relacijskimi podatkovnimi bazami trenutno na področju triplestore podatkovnih baz obstaja malo standardov. To omogoča večjo svobodo pri razvoju in posledično se je razvilo veliko število odprtokodnih in komercialnih implementacij. V diplomski nalogi smo si za cilj zastavili pregled tega področja in medsebojno primerjavo rešitev najbolj znanih ponudnikov.

Vsebinsko je teoretični del diplomske naloge sestavljen iz dveh delov. Cilj prvega dela je opredelitev kriterijev za primerjavo različnih ponudnikov triple-

store NoSQL (Not Only SQL) podatkovnih baz in nato opis njihovih rešitev po teh kriterijih. Predstavlja pregled bistvenih značilnosti in funkcionalnosti implementacij podatkovnih shramb ter obsega drugo (*Kriteriji za primerjavo triplestore podatkovnih baz*) in tretje (*Triplestore podatkovne baze*) poglavje. Drugi del opisuje problemsko domeno s predvidenimi scenariji uporabe (četrto poglavje *Problemska domena*). Podrobnosti implementacije v javanski aplikaciji, opisi testnega okolja in testnih scenarijev so opisani v petem poglavju *Testiranje*. V šestem poglavju *Analiza rezultatov* so predstavljeni rezultati testiranja skupaj z analizo le teh.

Praktični del je obsegal implementacijo različnih triplestore-ov v javanski aplikaciji za testiranje scenarijev uporabe, ki jih je predvidevala problemska domena.

Diplomsko delo zaključujejo sklepne ugotovitve o predstavljenih triplestore podatkovnih bazah.

Poglavje 2

Kriteriji za primerjavo triplestore podatkovnih baz

2.1 Tip shrambe podatkov

RDF (Resource Description Framework) shrambe podatkov so sistemi, zgrajeni z namenom shranjevanja in dostopanja do poljubnih podatkov v RDF obliki. Glede na arhitekturo jih lahko razdelimo na izvirne RDF shrambe, hibridne RDF shrambe s podporo relacijskih podatkovnih baz in RDF ovojnice (ang. wrapper) [1].

2.1.1 Izvirne RDF shrambe

Izvirne RDF shrambe so optimizirane za procesiranje RDF podatkov in delujejo neodvisno od drugih SUPB (Sistem za upravljanje s podatkovno bazo). Nastale so zaradi potreb po shranjevanju in strežbi velike količine podatkov veliki množici uporabnikov. So skalabilne in običajno namenjene spletnim aplikacijam, kot so naprimer socialna omrežja. Bolj kot konsistentnost podatkov sta v njih pomembna skalabilnost in visoka razpoložljivost. Omogočajo prilagodljivo podatkovno shemo, zato so primerne za aplikacije s pogostimi spremembami v podatkovnem modelu. V takih aplikacijah bi nas toga shema relacijskih podatkovnih baz ovirala [2].

Prinašajo številne prednosti, vendar niso univerzalna rešitev. V nekaterih sistemih, kot so bančni ali borzni, so relacije in ACID (Atomicity, Consistency, Isolation, Durability) transakcije še vedno potrebne. Podatki morajo biti vedno razpoložljivi in pravilni [3]. NoSQL podatkovne baze pa s sprejetjem možne začasne nekonsistentnosti pridobijo pri skalabilnosti in prilagodljivosti.

2.1.2 Hibridne RDF shrambe - s podporo SUPB

Hibridne RDF shrambe uporabljajo za shranjevanje in dostop do podatkov funkcionalnosti obstoječega, ponavadi relacijskega, SUPB. Delimo jih na sisteme z generično ali ontološko specifično shemo. Sisteme z generično shemo lahko naprej delimo na:

- enotabelni model (ang. single database table design), kjer se trojčki shranjujejo v tabeli s tremi stolpci - osebek, predikat in predmet;
- model tabele lastnosti (ang. property table design), kjer so v stolpcih denormalizirane RDF tabele shranjeni osebek in njegove lastnosti;
- normaliziran model, kjer se v ločenih tabelah shranjujejo URI-ji (Uniform Resource Identifier) in literali;
- hibridni pristop, ki združuje prednosti enotabelnih in normaliziranih modelov.

Ontološko-specifične sheme ne shranjujejo trojčkov v enotno tabelo, temveč uporabljajo pri shranjevanju shemo, ki posnema strukturne lastnosti ontologije [1]. Vsaka sprememba ontologije se prenese na strukturo podatkovnih tabel.

2.1.3 RDF ovojnice

RDF ovojnice so programske komponente, nameščene nad obstoječim izvorom podatkov. Omogočajo dostop do podatkov v RDF obliki brez spreminjanja obstoječega ogrodja (ang. framework). Njihova najpomembnejša

značilnost je, da so podatki namenjeni samo branju, zato operacije spreminjanja podatkov niso možne. Primer take shrambe je D2RQ platforma, ki jo bomo podrobneje opisali v nadaljevanju.

2.2 Serializacija

RDF grafi sicer omogočajo predstavitev podatkov v ljudem razumljivi obliki, vendar je ta neprimerna za izmenjavo podatkov med aplikacijami. Serializacija je proces pretvarjanja stanja objekta v format, ki je primeren za shranjevanje (npr. v datoteko ali podatkovno bazo) ali prenos po omrežju. Komplementaren proces je deserializacija, s pomočjo katere iz serializirane oblike dobimo prvotno stanje oziroma objekt. S kombinacijo obeh procesov dosežemo enostavno prenašanje in shranjevanje podatkov. Poznamo več enakovrednih serializacijskih oblik, med katerimi so TURTLE (Terse RDF Triple Language), N-Triples, RDFa, RDF/JSON in RDF/XML, ki so tudi podrobneje opisane v nadaljevanju.

2.2.1 TURTLE

TURTLE je nastal kot del N3 [4] zasnovan posebej za RDF z enostavnejšo in uporabniku lažje razumljivo sintakso. Format trojk je enostaven: osebek, predikat in predmet so zapisani v vrstici, ločeni s presledkom in zaključeni s piko. S @prefix znakom lahko predefiniramo resurse v obliki:

@prefix oseba: `<http:/www.test.com/oseba/>` .
@prefix predpona: `<http:/www.test.com/predpona>` .

in v celotnem dokumentu uporabljamo samo okrajšavo `oseba` namesto celotnega URI-ja. Primer celotnega izraza z uporabo okrajšave:

```
oseba:Peter predpona:pozna oseba:Janez .
```

V tej okrajšavi je `oseba:Peter` osebek, `predpona:pozna` predikat in `oseba:Janez` predmet, trditev zaključuje pika. Predikat in predmet v izrazih s skupnim

osebkom ločimo s podpičjem, s čimer dobimo strnjeno obliko, kot je vidno na spodnjem primeru:

```
oseba:Peter
    predpona:pozna oseba:Janez;
    predpona:priimek "Novak".
```

Podoben princip velja za izreke s skupnim osebkom in predikatom, kjer zapišemo najprej osebek in predikat, na koncu pa dodamo še predmete, ločene z vejicami. Primer:

```
oseba:Peter predpona:pozna oseba:Janez, oseba:Marko, oseba:Matej.
```

Komentarji so določeni na začetku vrstice z znakom #, prevajalnik ignorira besedilo od tega znaka do konca vrstice [5].

2.2.2 N-Triples

N-Triples je poenostavljena verzija TURTLE-ja, ki ne podpira okrajšave s @prefix in združevanja izrekov s podpičjem in vejicami. Izrek je definiran v eni vrstici, ki vsebuje osebek, predikat in predmet, zaključeni s piko [6]. Osebek je lahko v obliki URI-ja ali v obliki praznega vozlišča kot alfanumerično besedilo, pred katerim sta znaka _:. Predikat je lahko samo URI, predmet pa je lahko URI, prazno vozlišče (ang. node) ali literal (ASCII besedilo v dvojnih narekovajih) [7]. Primer:

```
<http://www.test.com/osebek> <http://www.test.com/predikat>
"predmet".
```

2.2.3 RDFa

RDFa je način dodajanja komentarjev v RDF obliki k XML dokumentom za razjasnjevanje pomena obstoječe vsebine. Osebek se nastavi z atributom `about`, ki je v obliki URI. Predikat nastavimo na tri načine:

- z `rel` izrazimo razmerje med dvema resursoma;

- s `property` razmerje med resursom in literalom;
- z `rev` obratno razmerje med dvema resursoma.

Predmet se nastavlja na štiri načine:

- s `content` določimo literal;
- s `href` URI vir v obliki povezave znotraj vrstice;
- s `src` URI vir, vdelan znotraj vrstice;
- z `resource` URI, kjer predmet ni viden na strani [6].

Med predmetom in predikatom obstaja direktna povezanost. Za predmet v obliki literala določimo predikat s `property`. V primeru, da je predmet vir, uporabimo atributa `rev` ali `rel`.
Primer trojčka v formatu RDFa:

<span about="http://www.test.com/oseba/Janez"
property="http://www.test.com/imenik#telefon"
content="012345678"\>

2.2.4 RDF-JSON

RDF/JSON format predstavlja množico RDF trojčkov kot zaporedje vgnezdenih podatkovnih struktur. Trojček osebek (S), predikat (P) in predmet (O) je predstavljen v obliki:

"S" : "P" : [O]

Predmet trojčka je predstavljen kot JSON objekt z naslednjimi vrednostmi:

- `type` je lahko URI, literal ali prazno vozlišče (podatek je obvezen);
- `value` predstavlja vrednost objekta, ki je lahko v obliki URI ali literala (podatek je obvezen);
- `lang` določa jezik literala (podatek ni obvezen);

- `datatype` določa podatkovni tip literala (podatek ni obvezen).

Vrednosti `lang` in `datatype` se uporabljata samo v primeru, da je vrednost `value` literal [8].
Trojček:

```
<http:/www.test.com/oseba/Peter> <http:/www.test.com/predpona/pozna>
"Janez".
```

lahko v formatu RDF/JSON zapišemo kot:

```
{
    "http:/www.test.com/oseba/Peter":
        {
            "http:/www.test.com/predpona/pozna":
            [ {
                "type":  "literal", "value":  "Janez"
            } ]
        }
}
```

2.2.5 RDF/XML

RDF/XML omogoča predstavitev RDF trojčkov v obliki XML sintakse in ga podpira večina aplikacij na semantičnem spletu. V `rdf:RDF` znački je več `rdf:Description` oz. opisnih elementov, ki skupaj sestavljajo pot skozi RDF graf. Element vsebuje enega ali več izrazov v obliki:

```
<rdf:Description rdf:about="osebek">
    <predikat rdf:resource="predmet"/>
    <predikat>literal</predikat>
</rdf:Description>
```

Z `rdf:about` atributom določimo osebek vsem izrazom v opisnem elementu. Predmet je predstavljen kot vir z `rdf:resource` značko ali kot literal znotraj

oznak predikata. Komentarji se kot v ostalih XML dokumentih začnejo z `<!--` in končajo z `-->` [5].

2.3 Način dostopa do podatkov

Posebna zgradba RDF shramb zahteva tudi drugačne mehanizme za dostop in operacije nad podatki kot pri relacijskih podatkovnih bazah. Obstaja več različnih RDF poizvedovalnih jezikov, najbolj znani med njimi so SPARQL (SPARQL Protocol and RDF Query Language), SeRQL (Sesame RDF Query Language), Gremlin, Cypher in RQL.

2.3.1 SPARQL

SPARQL je, podobno kot SQL pri relacijskih podatkovnih bazah, standardizirani poizvedovalni jezik za RDF shrambe. Omogoča štiri vrste poizvedb: SELECT, CONSTRUCT, ASK in DESCRIBE, ki so natančneje opisani v nadaljevanju.

- `SELECT` poizvedbo sestavljata dva osnovna dela: SELECT, ki določa nabor spremenljivk v rezultatu, in WHERE pogoj, ki določa vzorec v grafu za primerjavo z RDF podatki. Primer SELECT poizvedbe:

```
PREFIX imenik: <http:/www.test.com/imenik#>
SELECT ?oseba ?p ?o
WHERE
{
        ?oseba imenik:ime "Janez";
                imenik:primek "Novak";
                ?p ?o .
}
```

  Poizvedba vrača osebek, predikat in predmet vseh trojčkov, ki imajo skupen osebek z imenom Janez in priimkom Novak.

- `CONSTRUCT` vrača graf oziroma množico trojčkov. Lahko jih nespremenjene dobimo iz baze podatkov ali pa jih sestavimo iz pridobljenih podatkov. To omogoča kreiranje, kopiranje in pretvorbo enega RDF grafa v drugega. Primer CONSTRUCT poizvedbe:

```
PREFIX imenik: <http:/www.test.com/imenik#>
CONSTRUCT
{ ?oseba ?p ?o . }
WHERE
{
        ?oseba imenik:ime "Janez";
                imenik:primek "Novak";
                ?p ?o .
}
```

  Poizvedba vrača osebek, predikat in predmet vseh trojčkov, ki imajo skupen osebek z imenom Janez in priimkom Novak.

- `ASK` vrača vrednosti `true` ali `false`, če je podani vzorec grafa v rezultatu poizvedbe ali ne. Primer ASK poizvedbe, ki preveri, če v grafu obstaja osebek z imenom Janez in priimkom Novak:

```
PREFIX imenik: <http:/www.test.com/imenik#>
ASK WHERE
{
        ?oseba imenik:ime “Janez”;
                imenik:primek “Novak”.
}
```

- `DESCRIBE` se lahko uporablja na enak način kot SELECT, vendar ne vrača množice podatkov, temveč poizkuša podati informacije o virih. Pomagal naj bi pri razumevanju konteksta vrnjenih virov, vendar je koristnost rezultatov nepredvidljiva [9]. Rezultati se lahko razlikujejo

tudi pri uporabi različnih procesorjev poizvedb (ang. query processor).
Primer DESCRIBE poizvedbe o viru `<http:/www.test.com/oseba/id1#>`:

DESCRIBE `<http:/www.test.com/oseba/id1#>`

Poizvedba vrne vse trojčke, kjer je iskani vir objekt ali predmet.

Spreminjanje podatkov v triplestore podatkovni bazi omogoča razširitev SPARQL jezika SPARUL, imenovan tudi SPARQL/Update. Ta omogoča vstavljanje, brisanje in posodobitev podatkov. Zadnja verzija se imenuje SPARQL/Update 1.1, v kateri so tudi napisani primeri poizvedb.
Primer poizvedbe INSERT za vstavitev trojčka:

```
PREFIX oseba: <http:/www.test.com/oseba/>
PREFIX imenik: <http:/www.test.com/imenik#>
INSERT DATA
{
    oseba:id2 imenik:ime "Peter".
}
```

Primer poizvedbe DELETE za brisanje trojčka:

```
PREFIX oseba: <http:/www.test.com/oseba/>
PREFIX imenik: <http:/www.test.com/imenik#>
DELETE DATA
{
    oseba:id1 imenik:ime "Janez".
}
```

Posodobitev podatkov je rešena z operacijo INSERT/DELETE. Trojček se najprej izbriše, nato se v podatkovno bazo zapiše željena vrednost:

```
PREFIX oseba: <http:/www.test.com/oseba/>
PREFIX imenik: <http:/www.test.com/imenik#>
DELETE
{
```

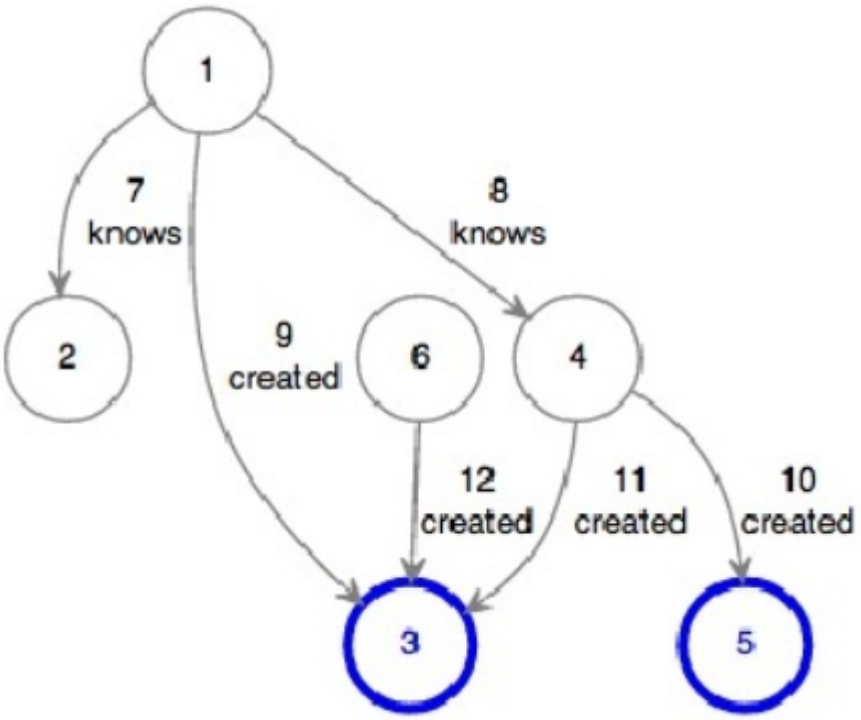
```
        oseba:id1 imenik:ime "Janez".
}
INSERT
{
        oseba:id1 imenik:ime "Peter".
}
```

2.3.2 Gremlin

Gremlin je zasnovan za poizvedovanje, analizo in spreminjanje usmerjenih multirelacijskh grafov tipa ključ/vrednost, ki imajo naslednje iz imena razvidne značilnosti:

- graf je usmerjen, kar pomeni, da ima vsaka povezava določen izvor in ponor;
- obstaja več različnih vrst vozlišč in med njimi lahko obstajajo različne povezave;
- izvori in ponori grafa imajo lahko poljubno število lastnosti.

Omogoča enostavno dodajanje vozlišč, povezav in iskanje zapletenih poti v grafu [10]. Primer enostavne poizvedbe v Gremlinu, ki poišče vsa vozlišča, do katerih lahko pridemo iz začetnega vozlišča po dveh korakih obhoda v grafu, je viden na sliki 2.1:



Slika 2.1: Rezultat Gremlin poizvedbe '`g.v(1).out.out`' sta z modro označeni vozlišči 3 in 5 (vir: `http://nosql.mypopescu.com/post/23916353997/short-intro-to-graph-databases-manipulating-and`)

2.3.3 Cypher

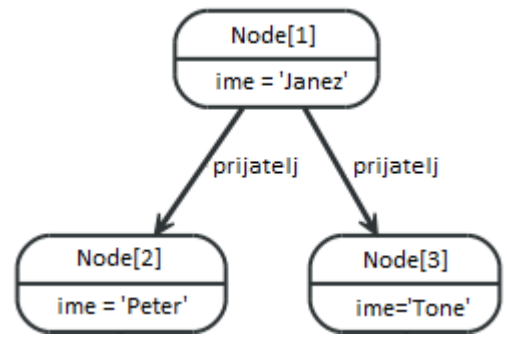
Cypher je poizvedovalni jezik, ki omogoča učinkovito pregledovanje in spreminjanje Neo4j graf podatkovne baze. Združuje različne pristope in prakse iz poizvedovalnih jezikov SQL (Structured Query Language) - ključne besede, kot sta WHERE in ORDER BY, in SPARQL - iskanje vzorcev v grafu. Uporablja naslednje izraze:

- `START` določa začetne točke v grafu, pridobljene preko poizvedb ali preko ID elementov;
- `MATCH` določa iskani vzorec v grafu, ki je povezan z začetnimi točkami grafa;

- `WHERE` določa kriterije filtriranja;
- `RETURN` določa izhodne vrednosti;
- `CREATE` za izdelavo vozlišč in povezav med njimi;
- `DELETE` za odstranjevanje vozlišč, povezav in lastnosti;
- `SET` za nastavljanje vrednosti lastnostim;
- `FOREACH` izvede spremembe enkrat za vsak element;
- `WITH` razdeli poizvedbo na več različnih delov [11].

Primer Cypher poizvedbe nad grafom nad sliki 2.2, ki išče prijatelje osebe z imenom Janez:

```
START oseba=node(1)
MATCH oseba-[prijatelj]->prijatelj
RETURN oseba, prijatelj
```



Slika 2.2: Graf prijateljev osebe Janez

Rezultat poizvedbe sta para v tabeli 2.1:

Tabela 2.1: Rezultati Cypher poizvedbe

oseba	prijatelj
Node[1]{ime->"Janez"}	Node[2]{ime->"Peter"}
Node[1]{ime->"Janez"}	Node[3]{ime->"Tone"}

2.4 Vmesnik API

Vmesnik API (Application Programming Interface) je zelo pomemben del vsake izmed implementacij NoSQL podatkovnih baz, saj so enostavna namestitev, uporaba in funkcionalnost zelo pomembni pri odločitvi za določeno implementacijo. Ob pomanjkanju orodij, ki so sicer prisotni v relacijskih podatkovnih bazah, poteka preko vmesnikov večina operacij nad NoSQL podatkovno bazo. Med programskimi jeziki je najbolj razširjena in podprta Java. Večinoma so poleg uradno podprtih jezikov za večino rešitev razvite tudi ovojnice za druge programske jezike.

2.5 Licenca

Licence se delijo na odprtokodne in komercialne. Odprtokodne licence večinoma ponujajo vse potrebno za normalno delo s podatkovnimi bazami. Omejitve se pojavljajo pri količini podatkov in številu uporabnikov, ki so jih sposobne postreči. Komercialne licence vsebujejo dodatne funkcionalnosti za zahtevnejšo uporabo in lahko ponujajo tudi boljše performančne lastnosti.

2.6 Skalabilnost

Triplestore podatkovne baze so se razvile kot alternativa relacijskim podatkovnim bazam, ko njihove zmogljivosti niso več zadoščale potrebam spletnih

aplikacij po obdelavi več terabajtov podatkov in strežbe velikega števila uporabnikov. Kot rešitev teh problemov nudijo skalabilnost v horizontalni smeri s porazdelitvijo podatkov preko gruč računalnikov.

2.7 Podpora uporabnikom

Pri novih tehnologijah, kar NOSQL podatkovne baze zagotovo so, je zelo pomembna izbira ponudnika z dobro dokumentirano podatkovno bazo, stabilnim delovanjem in učinkovito podporo uporabnikom. Večina rešitev izhaja iz odprtokodnih projektov, kjer običajno ni veliko sredstev namenjenih podpori, za razliko od uveljavljenih ponudnikov relacijskih podatkovnih baz. Vsaj zaenkrat je za namestitev NoSQL podatkovnih baz potreben višji nivo znanja kot pri relacijskih podatkovnih bazah. Posledično je pričakovati tudi daljši čas za uvajanje uporabnikov in integracijo v aplikacije.

Poglavje 3

Triplestore podatkovne baze

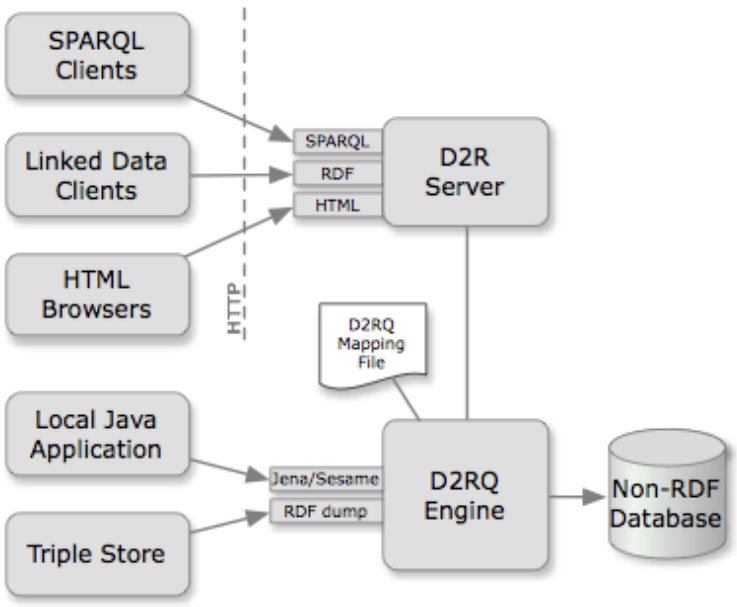
3.1 D2RQ

D2RQ platforma omogoča predstavitev podatkov relacijskih podatkovnih baz v obliki RDF grafov z uporabo preslikovanja. Sestavljajo jo tri glavne komponente (slika 3.1).

- `D2RQ Mapping Language` je deklarativni jezik za opis relacije med ontologijo in podatkovnim modelom, ki preslika podatke relacijske podatkovne baze v navidezni RDF graf. Graf lahko primerjamo s pregledi v relacijskih podatkovnih bazah, le da je namesto navidezne tabele RDF graf. Replikacija v RDF podatkovno bazo ni potrebna, graf se shranjuje v preslikovalno datoteko.

- `D2RQ Engine` je vtičnik za Jena in Sesame ogrodji, ki pretvori klice njunih vmesnikov API v SQL poizvedbe ali pa izvaja neposredne SPARQL poizvedbe nad prej preslikanim RDF grafom.

- `D2R Server` omogoča objavo podatkov relacijske podatkovne baze na semantičnem spletu.



Slika 3.1: D2RQ platforma (vir: `http://d2rq.org/d2r-server`)

Operacije dodajanja (Create), brisanja (Delete) in posodabljanja (Update) zaradi preslikave v virtualni graf niso podprte, saj D2RQ omogoča samo branje.

3.1.1 Tip shrambe podatkov

D2RQ podpira uporabo naslednjih podatkovnih baz: Oracle, MySql, PostgreSQL, Microsoft SQL server, HSQLDB in Interbase/Firebird. V preslikovalni datoteki se definira potrebne parametre za dostop do podatkovne baze in določi preslikavo tabel in njihovih stolpcev iz podatkovne baze v ustrezne RDF ekvivalente.

3.1.2 Serializacija

Podatki so običajno predstavljeni v N3 ter v TURTLE ali RDF/XML serializacijskem formatu.

3.1.3 Način dostopa do podatkov

Dostop do preslikanih podatkov je, odvisno od namena uporabe, mogoč na več načinov:

- z izvozom preslikanih podatkov v RDF/XML in N-Triples formatu preko D2RQ Engine-a,
- z vdelavo D2RQ v javanske aplikacije preko Jena ali Sesame API-ja, ki omogočata uporaba SPARQL poizvedb,
- z D2R strežnikom, ki omogoča uporabo oddaljenega dostopa do preslikane baze preko SPARQL protokola [21].

3.1.4 Vmesnik API

D2RQ API je namenjen uporabi v Jena ali Sesame ogrodju. Nekatere izmed pomembnejših funkcij API-ja so:

- pretvorba SPARQL poizvedb v SQL poizvedbe in prikaz rezultatov v obliki RDF grafa,
- pretvorba preslikanega D2RQ grafa v javanske objekte,
- ModelD2RQ razred za pretvorbo preslikanega grafa v Jena Model,
- GraphD2RQ omogoča pretvorbo preslikanega grafa v Jena Graph API.

3.1.5 Licenca

D2RQ platforma je odprtokodna rešitev, na voljo pod Apache License V2 licenco, ki dovoljuje distribucijo in spreminjanje programske kode.

3.1.6 Skalabilnost

D2RQ naj bi ponujal zadovoljive performanse na podatkovnih bazah z do nekaj deset milijonov zapisov.

3.1.7 Podpora uporabnikom

Dokumentacija z opisi orodij, preslikovalnega jezika in začetnimi navodili je dostopna na spletu na naslovu `http://d2rq.org/`. Mailing lista na naslovu `https://lists.sourceforge.net/lists/listinfo/d2rq-map-devel` je na voljo uporabnikom za podporo in diskusijo o razvoju D2RQ. Na naslovu `https://github.com/d2rq/d2rq/issues` je objavljen seznam vseh težav in hroščev v D2RQ. Izvorna koda je na voljo na naslovu `http://github.com/d2rq/d2rq`.

3.2 Jena SDB

Jena SDB je komponenta Apache Jena ogrodja, ki za shranjevanje in poizvedovanje uporablja relacijske podatkovne baze.

3.2.1 Tip shrambe podatkov

Trenutno so podprte naslednje relacijske podatkovne baze: PostgreSQL, MySQL, Microsoft SQL server, Oracle 10g, Apache Derby, H2, HSQLDB in DB2 9 [20]. Podatki se shranjujejo v tabele vozlišč, predpon, trojčkov in četverčkov (ang. quad).

3.2.2 Serializacija

Jena SDB podpira podatkovne formate RDF/XML, N-Triples in Turtle.

3.2.3 Način dostopa do podatkov

Dostop do podatkovne shrambe in upravljanje z njo je mogoč preko ukazne vrstice ali javanskega Jena API-ja. SPARQL poizvedbe se s preslikavo preslikajo v SQL poizvedbe, ki se nato lahko izvedejo nad relacijsko podatkovno bazo.

Uporaba Joseki SPARQL strežnika omogoča dostop do podatkov preko HTTP z uporabo SPARQL protokola.

3.2.4 Vmesnik API

Jena SDB je del javanskega Apache Jena ogrodja in izrablja vse njegove funkcionalnosti za povezavo s podatkovno bazo in spreminjanje podatkov v njej. Parametre povezave s podatkovno bazo določimo preko JDBC (Java Database Connectivity) gonilnika. Ta je lahko definiran v programski kodi ali pa se prebere iz konfiguracijske datoteke, kar omogoča enostavno spremembo v aplikaciji uporabljene podatkovne baze.

3.2.5 Licenca

Jena SDB je odprtokodna rešitev, na voljo pod Apache License V2 licenco, ki dovoljuje distribucijo in spreminjanje programske kode.

3.2.6 Skalabilnost

Uradnih podatkov o omejitvah podatkovne shrambe ni, so pa v Jena SDB podatkovno bazo s podporo PostreSQL uspešno shranili 650 milijonov trojčkov [23].

3.2.7 Podpora uporabnikom

Uporabnikom je na voljo podporna mailing lista na naslovu `users@jena.apache.org`, kjer lahko postavljajo vprašanja o razvoju Jena aplikacij in reševanju težav pri uporabi. Razvijalcem je na voljo še dodatna lista `dev@jena.apache.org`, kjer poteka diskusija o razvoju Jena platforme in njenih komponent. Nekateri razvijalci Jene so občasno na voljo na `#jena` kanalu na `irc.freenode.net`. Veliko vprašanj in odgovorov v povezavi z Jeno je tudi na spletnih straneh `stackoverflow.com` in `answers.semanticweb.com`.

3.3 Jena TDB

Apache Jena je ogrodje, napisano v Javi, ki se uporablja za gradnjo aplikacij na semantičnem spletu. Njena komponenta Jena TDB je namenjena shranjevanju in preiskovanju RDF podatkov.

3.3.1 Tip shrambe podatkov

Jena TDB je izvirna shramba podatkov, kjer so podatki lahko shranjeni v glavnem pomnilniku ali na disku. Na disku so podatki shranjeni v eni datoteki za shranjevanje ene množice RDF podatkov.

3.3.2 Serializacija

Jena TDB podpira podatkovne formate RDF/XML, N-Triples in Turtle.

3.3.3 Način dostopa do podatkov

Dostop do lokalne baze podatkov je mogoč preko ukazne vrstice ali preko Jena API-ja. Preko ukazne vrstice so mogoči različni ukazi nad podatkovno bazo in SPARQL poizvedbe. Verjetno najbolj uporaben med ukazi je `tdbloader` za hitro vstavljanje večjega števila podatkov, ki ni mogoč preko Jena API-ja. Uporaba Fuseki SPARQL strežnika omogoča dostop do podatkov preko HTTP z uporabo SPARQL protokola.

3.3.4 Vmesnik API

Jena je javanski API za kreacijo in manipulacijo RDF grafov. Ima razrede za predstavitev grafov, virov, lastnosti in literalov. Graf se v Jeni imenuje model in je implementiran z vmesnikom Model. Jena TDB uporablja vse funkcionalnosti Jena API-ja, ki vključuje metode za kreacijo in povezavo do podatkovne baze iz aplikacije ter dostop do podatkov v njej. Jena TDB podpira transakcije, kar je tudi priporočljiv način dela, saj mora v nasprotnem primeru aplikacija sama skrbeti za sinhronizacijo podatkov [19].

3.3.5 Licenca

Jena TDB je odprtokodna rešitev, na voljo pod Apache License V2 licenco, ki dovoljuje distribucijo in spreminjanje programske kode.

3.3.6 Skalabilnost

Uradnih podatkov s strani razvijalcev o omejitvah pri shranjevanju podatkov ni, so pa objavljeni podatki o delujoči podatkovni bazi z 1.7 milijarde trojčkov [23].

3.3.7 Podpora uporabnikom

Uporabnikom je na voljo podporna mailing lista na naslovu `users@jena.apache.org`, kjer lahko postavljajo vprašanja o razvoju Jena aplikacij in reševanju težav pri uporabi. Razvijalcem je na voljo še dodatna lista `dev@jena.apache.org`, kjer poteka diskusija o razvoju Jena platforme in njenih komponent. Nekateri razvijalci Jene so občasno na voljo na `#jena` kanalu na `irc.freenode.net`. Veliko vprašanj in odgovorov v povezavi z Jeno je tudi na spletnih straneh `stackoverflow.com` in `answers.semanticweb.com`.

3.4 Neo4j

Neo4j je robustna, skalabilna in visoko performančna podatkovna baza, implementirana v Javi. Primerna je tako za uporabo v manjših aplikacijah kot tudi pri velikih komercialnih projektih z velikim številom uporabnikov in podatkov. Omogoča: ACID transakcije, visoko razpoložljivost, skaliranje do milijard vozlišč in relacij ter visoko hitrost preiskovanja podatkov. Uporablja se lahko kot vdelana podatkovna baza ali pa kot samostojni strežnik v PHP, .NET in Javascript spletnih aplikacijah, ki se jim prilagaja po velikosti tekom njihovega razvoja. V Javansko aplikacijo se vdela z uporabo Neo4j Java API-ja. Razvite so še neuradne ovojnice za JRuby/Ruby, Python, Clojure, Scala in druge programske jezike [12]. En sam strežnik lahko zadosti milijardam

vozlišč in relacij, po potrebi pa se lahko obremenitev porazdeli tudi med več strežniki v visoko dostopni konfiguraciji.

3.4.1 Tip shrambe podatkov

Neo4j je izvirna shramba podatkov, kjer je celoten podatkovni model shranjen kot graf na disk. Graf sestavljajo vozlišča, relacije in lastnosti, ki so neposredno povezani med seboj s kazalci, kar odpravi potrebo po dupliciranju podatkov [11].

3.4.2 Serializacija

Neo4j je podatkovna baza, ki temelji na grafih. Preko vmesnika Neo4j API omogoča vstavljanje podatkov v obliki vozlišč in povezav med njimi. Preko vmesnika Sail (Storage And Interface Layer) lahko v Neo4j vstavimo tudi podatke v vseh najpogostejših RDF formatih, kot so RDF/XML, Turtle in N-Triples.

3.4.3 Način dostopa do podatkov

Do podatkov v Neo4j se lahko dostopa na več načinov. Najenostavnejši način je preko Neo4j Java API-ja, kompleksnejši je Neo4j Traversal API, ki implementira dodatne vmesnike, kar pa doda k zahtevnosti uporabe. Omogoča tudi direktno uporabo poizvedovalnega jezika SPARQL preko dodatnih komponent (npr. Sail). Obstajata še vtičnika za Gremlin in od verzije 1.4 naprej še za Cypher. Do podatkov na strežniku se dostopa preko REST (Representational State Transfer) API-ja. Spletni vmesnik Neo4j Web Administration poleg operacij in pregleda podatkov dopušča še pregled stanja Neo4j strežnika in interakcijo z bazo preko različnih konzol.

3.4.4 Vmesnik API

Neo4j API vsebuje vse potrebno za uporabo Neo4j podatkovne baze, in sicer povezavo s podatkovnim strežnikom, kreacijo, spreminjanje in brisanje grafov.
Neo4j Traversal API dopolnjuje osnovni API. Ima dodatne vmesnike za izdelavo in natančno določanje pravil obhodov grafa, glavni med njimi so `TraversalDescription, Evaluator, Traverser in Uniqueness`. Določajo, katera vozlišča naj bodo v končnem rezultatu, katere relacije naj se pri obhodu ne upoštevajo, ali se lahko ista vozlišča obišče večkrat in še mnoge druge funkcionalnosti [11].
Neo4j REST API uporablja HTTP zahteve in JSON (JavaScript Object Notation), v katerem so odgovori na zahteve in podatki, poslani z zahtevami. V glavnem se uporabljajo zahteve GET, POST, PUT in DELETE za različne operacije, naprimer kreiranje vozlišč, lastnosti, relacij in preiskovanje grafa.

3.4.5 Licenca

Neo4j ponuja tri vrste rešitev: Neo4j Community je odprtokodna rešitev, na voljo pod GPLv3 Community edition licenco. Neo4j Advanced in Neo4j Enterprise različici sta na voljo pod dvojno licenco: odprtokodno (A)GPLv3 in Neo Technology Commercial License (NTCL). Community različica je zastonj, letna uporabnina za Advanced verzijo je 6.000 USD\EUR, za Enterprise pa 24.000 USD\EUR [13]. Za testiranje so na voljo vse tri rešitve pod odprtokodno licenco. Za poslovne spletne aplikacije sta primerni Advanced ali Enteprise različici.

3.4.6 Skalabilnost

Količina podatkov, ki jo lahko sprejme Neo4j podatkovna baza, je omejena z naslovnim prostorom primarnih ključev vozlišč, relacij, lastnosti in tipov relacij. V podatkovno bazo je trenutno mogoče naenkrat shraniti primarne ključe:

- 2^{35} ($\sim$ 34 milijard) vozlišč v grafu;
- 2^{35} ($\sim$ 34 milijard) relacij med dvema vozliščema v grafu;
- 2^{36} ($\sim$ 68 milijard) lastnosti vozlišč;
- 2^{15} ($\sim$ 32.000) tipov relacij.

Vsaka relacija je sestavljena iz začetnega vozlišča, končnega vozlišča in tipa relacije. Primer relacije:

```
( Janez ) -> POZNA -> ( Peter )
```

"Janez" je začetno, "Peter" končno vozlišče, povezuje ju tip relacije "POZNA".

3.4.7 Podpora uporabnikom

Podpora uporabnikom je odvisna od verzije.

- Community nima uradne podpore razvijalcev, vendar je zelo razvita spletna uporabniška pomoč. Na voljo so Googlove skupine, Tweeter in uporabniški blogi.
- Advanced vključuje uradno podporo preko elektronske pošte deset ur dnevno, pet dni v tednu.
- Enterprise vključuje neprekinjeno uradno telefonsko pomoč.

Na spletni strani je mogoč tudi ogled in prenos dokumentacije.

3.5 OWLIM

OWLIM ponuja visoko performančne semantične repozitorije. Implementiran je v Javi, kar omogoča prenosljivost in enostavno namestitev. Ponuja različne rešitve, primerne za najširši spekter uporabnikov.

- OWLIM-Lite je po izjavah avtorjev najhitrejši semantični repozitorij na svetu. Implementiran je v Javi za uporabo kot Sail vmesnik za Sesame openRDF ogrodje. Obdelava poizvedb poteka v glavnem pommnilniku [26].
- OWLIM-SE je po izjavah avtorjev najskalabilnejši semantični repozitorij na svetu, tako glede količine RDF podatkov, ki jih lahko shrani, kot tudi hitrosti, s katero lahko izvaja operacije nad podatki [27].
- OWLIM-Enterprise je zasnovan na OWLIM-SE in uporablja gruče za paralelno izvajanje poizvedb z enakomerno delitvijo obremenitve na računalnike v gruči [28].

3.5.1 Tip shrambe podatkov

Owlim je izvirna shramba podatkov, kjer je podatkovni model lahko shranjen v glavnem pomnilniku ali na disku.

3.5.2 Serializacija

OWLIM podpira uvoz in izvoz širokega nabora RDF podatkovnih formatov preko vmesnika Sesame API, med njimi so RDF/XML, N3, N-Triples in TURTLE.

3.5.3 Način dostopa do podatkov

OWLIM zaradi odvisnosti od podatkovnih in poizvedovalnih standardov Sesame uporablja poizvedovalne jezike SeRQL, SPARQL, RQL in RDQL, ki jih podpira Sesame.

3.5.4 Vmesnik API

Owlim nima razvitega lastnega vmesnika API. Za integracijo in uporabo repozitorijev uporablja že razvite API, med drugimi Sesame openRDF API,

ogrodje Apache Jena in ogrodje ORDI [25]. Najprimernejši in tudi najučinkovitejši je Sesame openRDF API, ker je OWLIM zgrajen okoli Sesamovega RDF podatkovnega modela. Sesame vmesnik Sail omogoča lokalni ali preko `Sesame HTTP Server/Client` komponent še oddaljen dostop do repozitorija. Lastnega uporabniškega vmesnika nima, priporočena je uporaba Sesame spletnega vmesnika.

3.5.5 Licenca

OWLIM-Lite ni odprtokodna programska oprema, saj ne dovoljuje nikakršnih posegov v programski paket, vendar pa je njegova uporaba brezplačna. OWLIM-SE in OWLIM-Enterprise sta na voljo pod komercialno licenco z neomejenim rokom trajanja. Cena je odvisna od zmogljivosti strežnika oziroma od števila strežnikovih CPU jeder, na katerega bo OWLIM nameščen. Za raziskovalne in evaluacijske potrebe dovoljujejo tudi brezplačno uporabo obeh komercialnih rešitev.

3.5.6 Skalabilnost

Količina podatkov, ki jih lahko sprejmejo Owlim podatkovne baze, je odvisna od različice.

- OWLIM-Lite podpira do sto milijonov trojčkov na povprečni računalniški konfiguraciji. Dostop do podatkov je hiter tudi pri veliki količini podatkov, performančne težave povzroča počasno brisanje podatkov [26].
- OWLIM-SE podpira na običajnih namiznih računalnikih več milijard trojčkov, na zmogljivejših strežniških konfiguracijah pa več deset milijard trojčkov [27].
- OWLIM-Enterprise je zasnovan na OWLIM-SE in omogoča poleg obdelave velikih količin podatkov še paralelno obdelovanje poizvedb z delitvijo bremena na računalnike v gruči [28].

3.5.7 Podpora uporabnikom

Tehnična podpora v procesu razvoja, optimizacije in vzdrževanja je kupcem na voljo za vse različice OWLIM-a. Ločujejo tri nivoje resnosti napak, od katerih sta odvisna tudi odzivni čas in čas odprave napake. Prikazani so v tabeli 3.1. Napake najvišje stopnje lahko povzročijo izpad sistema in možnost okvare ali izgube podatkov. Napake visoke stopnje lahko zelo poslabšajo učinkovitost ter funkcionalnost sistema. Napake najnižje stopnje povzročajo le manjše performančne težave in ne ogrožajo stabilnosti sistema.

Tabela 3.1: Nivoji resnosti napak v OWLIM podatkovnih bazah

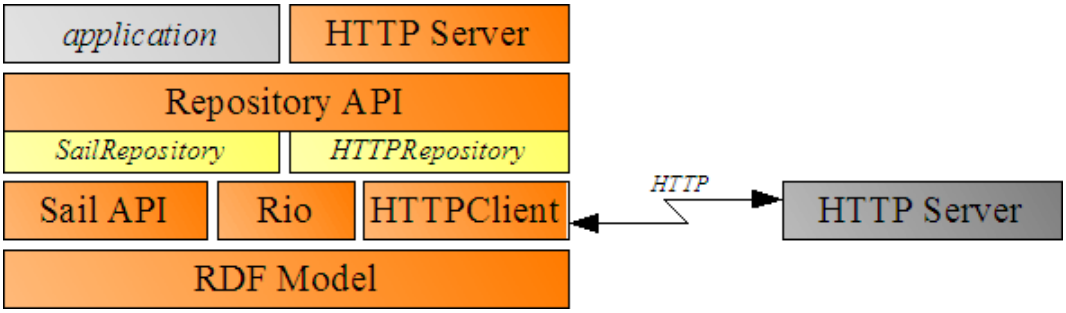
Resnost napake	Odzivni čas	Čas odprave napake
Najvišja	2 uri	24 ur
Visoka	24 ur	5 dni
Nizka	5 dni	15 dni

Naročnina na tehnično podporo je mogoča preko paketov podpore s predplačniško letno naročnino. Vsebina paketa je določena z mesečno omejitvijo števila odprav napak najvišje in visoke stopnje, ter s številom inženirjev, ki te napake odpravljajo [29]. Dokumentacija je za vse tri različice na voljo na spletni strani `http://www.ontotext.com/owlim/documentation`. Dokumentacija se deli na štiri različne dokumente: osnovne podatke o programu, vodič za hitri start, opis funkcionalnosti novih različic in daljša uporabniška navodila za instalacijo, konfiguracijo ter uporabo podatkovnih baz.

3.6 Sesame

Sesame je odprtokodno javansko ogrodje za shranjevanje in operacije nad RDF podatki. Razvilo ga je nizozemsko podjetje Aduna. Omogoča poljubno razširljivost in izbiro shranjevalnih mehanizmov, RDF podatkovnih formatov, rezultatov poizvedovanj in poizvedovalnih jezikov. Ogrodje sestavlja več

komponent, najpomembnejše med njimi so na sliki 3.2.



Slika 3.2: Komponente ogrodja Sesame (vir: `http://www.openrdf.org/doc/sesame2/2.0.1/users/ch03.html`)

`RDF Model` predstavlja osnovo, od katere so odvisni vsi deli ogrodja. Med drugim določa vmesnike in implementacijo vseh osnovnih RDF entitet, kot so URI, prazno vozlišče, literal in izrek [15]. Vmesne komponente omogočajo uporabo različnih RDF datotečnih formatov, tipov shramb podatkov ter preko `Repository API` metod za delo z RDF podatki. Najvišja komponenta `HTTP Server` je sestavljena iz več javanskih metod za dostop do podatkov preko HTTP.

3.6.1 Tip shrambe podatkov

Sesame je izvirna shramba podatkov, ki za shranjevanje RDF podatkov uporablja repozitorije, nad katerimi se tudi izvajajo vse operacije. Obstaja sedem vrst repozitorijev:

- `memory` je RDF repozitorij, kjer so podatki shranjeni v pomnilniku;
- `memory-rdfs` je RDF repozitorij, kjer so podatki shranjeni v pomnilniku z RDF shemo;
- `memory-rdfs-dt` je RDF repozitorij, kjer so podatki shranjeni v pomnilniku z RDF shemo in z neposrednim sklepanjem (ang. inference)

o hierarhiji tipa podatkov;

- `native` je izvirni RDF repozitorij s podatkovno strukturo na disku;
- `native-rdfs` je izvirni RDF repozitorij, kjer so podatki shranjeni na disku in z RDF shemo;
- `native-rdfs-dt` je izvirni RDF repozitorij z RDF shemo in z neposrednim sklepanjem o hierarhiji tipa podatkov;
- `remote` je repozitorij, ki služi kot nadomestek za repozitorij na Sesame strežniku;
- `pgsql` je repozitorij, ki podatke shranjuje v Postgre podatkovno bazo;
- `mysql` je repozitorij, ki podatke shranjuje v MySQL podatkovno bazo.

Vse vrste pomnilniško hranjenih repozitorijev lahko svoje stanje shranijo v datoteko ob koncu seje in ga ob ponovni inicializaciji preberejo iz nje. V nasprotnem primeru so ob inicializaciji pomnilniški repozitoriji vedno prazni. Izvirni repozitoriji, shranjeni na disku, v primerjavi s pomnilniškimi bolje skalirajo in niso omejeni z velikostjo razpoložljivega pomnilnika. Počasnejši so zaradi časa dostopa do diska.
Preko relacijskega SUPB Sail vmesnika se lahko RDF podatki shranijo v relacijske podatkovne baze. Trenutno podprte so PostgreSQL, MySQL, Microsoft SQL Server in Oracle podatkovne baze. Performančni rezultati so slabši kot pri shranjevanju v izvirne RDF repozitorije.

3.6.2 Serializacija

Sesame podpira vse glavne RDF podatkovne formate, kot so RDF/XML, Turtle, N-Triples in drugi.

3.6.3 Način dostopa do podatkov

Sesame podpira dva poizvedovalna jezika: s strani Sesame razvitega SeRQL in standardiziranega SPARQL. SeRQL je nov poizvedovalni jezik s podobnimi funkcijami kot SPARQL, vendar z drugačno sintakso, v okviru Sesame frameworka.

3.6.4 Vmesnik API

`Repository API` ponuja različne metode za kreiranje repozitorijev in izvedbo poizvedb nad podatki v repozitorijih. Trije različni tipi poizvedb so tuple, graf in boolean poizvedbe, ki se razlikujejo v rezultatu poizvedbe. Rezultat tuple poizvedbe je množica trojčkov. Vsaka izmed množic predstavlja rešitev poizvedbe in se uporablja za pridobitev točnih vrednosti iz RDF podatkov. Graf poizvedba vrne RDF graf, ki lahko predstavlja podmnožico podatkov za nadaljno obdelavo ali serializacijo v RDF dokument. Boolean poizvedba vrača samo true ali false in običajno služi za preverjanje, če repozitorij vsebuje določeno informacijo.
Z uporabo metod v komponenti `HTTP Server` je omogočen dostop do oddaljenih repozitorijev in izvajanje vseh tipov poizvedb.
Preko Sesame API-ja je mogoča uporaba tudi drugih triplestore podatkovnih shramb, kot so AllegroGraph, Owlim, Virtuoso in drugi.

3.6.5 Licenca

Sesame je odprtokodna rešitev, izdana pod licenco BSD. Omogoča prosto spreminjanje programske kode in neomejeno uporabo tudi v komercialnih aplikacijah [16]. To omogoča hitrejše odpravljanje napak in prost razvoj s sodelovanjem različnih razvijalcev.

3.6.6 Skalabilnost

Pomnilniški repozitoriji so omejeni z velikostjo glavnega pomnilnika. Sesame izvirni repozitoriji so namenjeni uporabi s srednje velikimi količinami podatkov, to je med 100 in 150 milijoni trojčkov, odvisno od strojne opreme in značilnosti podatkov. Pri uporabi večjih količin podatkov je priporočena delitev na več izvirnih repozitorijev.

3.6.7 Podpora uporabnikom

Uporabnikom je na voljo mailing lista, preko katere lahko postavljajo vprašanja in izmenjujejo mnenja. Odziv na zastavljena vprašanja je hiter, večinoma še v istem ali naslednjem dnevu. Obstaja še posebna mailing lista, na kateri so oznanjene nove verzije programov in različni dogodki. Dodajanje novih objav je omejeno na projektne administratorje. Odkrite hrošče lahko uporabniki objavijo preko vmesnika za sporočanje težav JIRA [17], kjer jih administratorji pregledajo in razrešijo.

3.7 Virtuoso

Virtuoso Universal Server je platforma, ki združuje funkcionalnosti upravljanja z relacijskimi, RDF in XML podatki, podatkovnega in spletnega strežnika. Brezplačna različica za razliko od komercialne ne omogoča uporabe gruč, visoke razpoložljivosti in replikacije podatkov. Kljub temu pa zadostuje večini potreb, ki se pojavljajo v sodobnih aplikacijah.

3.7.1 Tip shrambe podatkov

Virtuoso je izvirna shramba podatkov, kjer je podatkovni model shranjen v glavnem pomnilniku ali na disku.
Poleg uporabe izvirne RDF shrambe omogoča preko komponente `RDF views` tudi preslikavo podatkov iz relacijskih podatkovnih baz v RDF obliko. Preslikava je dinamična, zato se vsaka sprememba v podatkovni bazi prenese

tudi na RDF predstavitev podatkov. Osnovna funkcionalnost je preslikava rezultata SQL poizvedbe v množico trojčkov.

3.7.2 Serializacija

Virtuoso podpira N3, N-Triples in RDF/XML serializacijske podatkovne formate.

3.7.3 Način dostopa do podatkov

Dostop do podatkov v Virtuosu je mogoč na tri načine:

- preko SPARQL vmesnika, ki obdeluje zahteve preko SPARQL protokola,
- preko SQL vmesnikov do Virtuosa (ODBC, JDBC, OLEDB, ADO.NET in XMLA),
- preko shranjenih Virtuosovih procedur in funkcij.

3.7.4 Vmesnik API

Virtuoso distribucija v odprtokodni in komercialni različici vsebuje Virtuoso Conductor, ki je celovit sistem za nadzor in operacije nad podatkovnimi bazami preko spletnega brskalnika. Za uporabo v javanskih aplikacijah niso razvili lastnega vmesnika API, ampak so izdelali samo podporo za dostop do Virtuoso podatkovne baze preko Sesame in Jena API-ja. Podprt je poln dostop do vseh funkcionalnosti njunih API-jev.

3.7.5 Licenca

Virtuoso Open-Source je na voljo pod GPLv2 licenco. Njegova uporaba je brezplačna, na voljo je tudi celotna izvorna koda.
OpenLink Virtuoso Universal Server ponuja več tipov komercialnih licenc. Cena licence je odvisna od vrste strežnika, števila niti, pri strežnikih v gruči

pa od števila vozlišč. Za testiranje je na voljo 15-dnevna brezplačna licenca [24].

3.7.6 Skalabilnost

Uradnih podatkov s strani razvijalcev o omejitvah Virtuosa pri shranjevanju podatkov ni, so pa objavljeni podatki o delujoči Virtuoso gruči, ki vsebuje več kot 15 milijard trojčkov.

3.7.7 Podpora uporabnikom

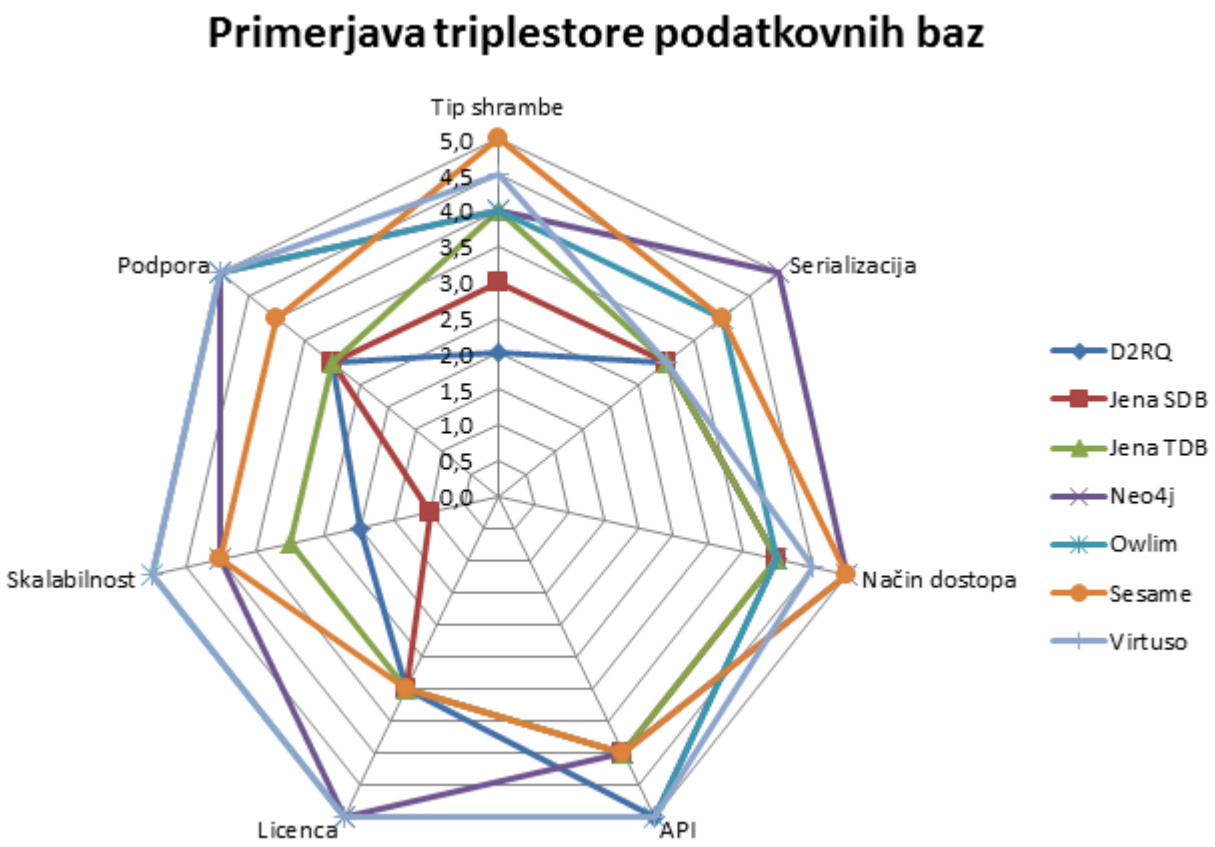
S spletne strani je mogoč brezplačen prenos zelo obsežne dokumentacije, ki obsega preko 3900 strani. Primernejša je za komercialne uporabnike, saj opisuje tudi funkcionalnosti, ki jih v brezplačni različici ni. Ločena dokumentacija samo za uporabnike brezplačne verzije Virtuoso Open-Source je dostopna na spletu na naslovu `http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/`. Tem uporabnikom so na voljo tudi tri mailing liste:

- `virtuoso-announce`, kjer so objavljene novice o novih različicah in o resnejših hroščih;
- `virtuoso-users` je namenjena uporabniški diskusiji o implementaciji, uporabi, optimizaciji...;
- `virtuoso-devel` je namenjena uporabnikom za sporočanje hroščev in diskusiji o težavah z razvijalci.

Spletna podpora je na voljo vsem uporabnikom komercialne različice Virtuosa, nova vprašanja lahko vnašajo preko spletnega vmesnika `http://support.openlinksw.com/support/online-support.vsp`. Uporabniki z aktivnimi licencami lahko dokupijo vzdrževalno podporni paket, ki ponuja visoko prioritetno podporo in popuste na posodobitve programa.

3.8 Primerjava triplestore podatkovnih baz

Primerjavo lastnosti podatkovnih baz z našimi ocenami najboljše prikazuje radarski graf na sliki 3.3. Lastnosti podatkovnih baz smo ovrednotili po lestvici od 0-5, pri čemer je 0 najslabša ocena, 5 pa najboljša. Pri oceni smo upoštevali prednosti vsake izmed podatkovnih baz pred drugimi.
Glede na tip shrambe smo najvišje ocenili Sesame, ker ponuja različne vrste repozitorijev, od pomnilniških, izvirnih in tudi shranjevanje v relacijskih podatkovnih bazah. Najnižje smo ocenili D2RQ, saj ponuja samo preslikavo relacijskih podatkov v RDF format, ne pa tudi spreminjanja podatkov.
Neo4j podpira vse najpogostejše RDF podatkovne formate, omogoča pa tudi uvoz vozlišč v JSON formatu.
Neo4j, Sesame in Virtuoso ponujajo dostop do podatkov z uporabo SPARQL poizvedb preko API vmesnikov. Na voljo je tudi spletni vmesnik za pregled podatkovnih baz in izvajanje poizvedb. Neo4j in Sesame pa omogočata tudi uporabo dodatnih poizvedovalnih jezikov.
Pri ocenjevanju API vmesnikov smo najvišjo oceno namenili D2RQ, Virtuoso-u in Owlim-u, saj vsi omogočajo uporabo tako Jena API kot Sesame API.
Neo4j, Owlim in Virtuoso ponujajo odprtokodne in komercialne rešitve, s čimer zadovoljijo potrebe najširšega kroga uporabnikov.
Delujoče implementacije Virtuoso in Owlim podatkovnih baz vsebujejo več milijard trojčkov, s čimer sta najbolj skalabilna med primerjanimi podatkovnimi bazami.
Podpora uporabnikom je najboljša pri komercialnih različicah rešitev. Najboljše so zato ocenjeni Neo4j, Owlim in Virtuoso, ki poleg odprtokodnih ponujajo tudi komercialne rešitve.



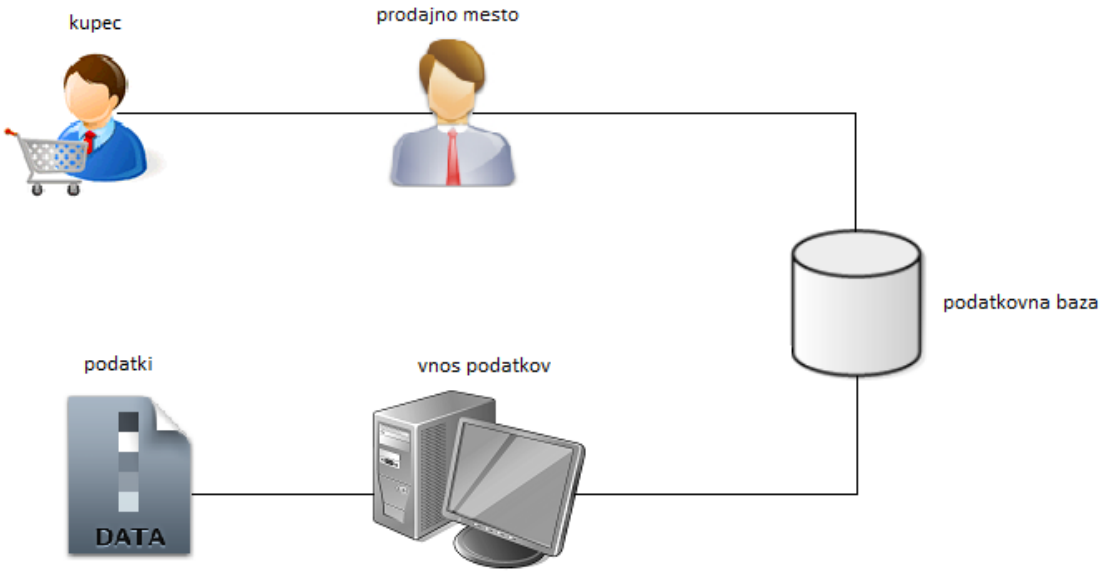
Slika 3.3: Primerjava triplestore podatkovnih baz

Poglavje 4

Problemska domena

Manjša trgovina se ukvarja s prodajo najrazličnejšega blaga. Zaradi prostorske stiske so njihove skladiščne zmožnosti zelo omejene, zato blago večinoma sproti naročajo preko katalogov pri dobaviteljih. Cene izdelkov se med dobavitelji razlikujejo in tudi konstantno spreminjajo, kar je že večkrat privedlo do večjih nabavnih stroškov, kot je bilo potrebno. Dobavitelji omogočajo tudi pošiljanje podatkov v RDF formatu ali v obliki SQL skript. Zato so se v trgovini odločili za prehod na informacijski sistem, kjer bi se v enotni podatkovni bazi zbirali vsi podatki o izdelkih. Shema problemske domene je prikazana na sliki 4.1.
Podjetju, ki jim bo zasnovalo informacijski sistem, so podali štiri predvidene scenarije uporabe:

- kupcem na prodajnem mestu na podlagi njihovih želja poiščejo primeren produkt;
- tekom dneva poleg običajnega dela s kupci poteka še vnos podatkov, ki jih pošljejo novi dobavitelji;
- sistem je uravnoteženo obremenjen z enako količino dostopanja do podatkov in dodajanjem ter brisanjem vnosov;
- posodobitve podatkovne baze so predvidene čez noč, ko se pobrišejo nepotrebni vnosi in dodajo novi.



Slika 4.1: Shema problemske domene

Poglavje 5

Testiranje

5.1 Testno okolje

Pri testiranju smo uporabili operacijski sistem Ubuntu Linux 32-bit verzija 12.10, ki smo ga poganjali iz VmWare Workstation-a s 3GB pomnilnika. Vsaka izmed rešitev je imela ločen navidezni stroj, s čimer smo preprečili morebitne medsebojne vplive. Navidezni stroj smo naselili na sistem s procesorjem Intel Core 2 Duo 2GHz in 4GB glavnega pomnilnika.

5.2 Testni podatki

Podatke za testiranje smo pridobili z uporabo podatkovnega generatorja Berlin SPARQL Benchmark Data Generator-ja [30], ki omogoča kreiranje poljubne količine podatkov v več RDF formatih in SQL skriptah.

Slika 5.1: Podatkovni model podatkovnega generatorja (vir: `http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/Bizer-Schultz-Berlin-SPARQL-Benchmark-IJSWIS.pdf`)

Podatkovni model sestavljajo naslednji razredi (Slika 5.1): izdelki (Product), tipi izdelkov (ProductType), lastnosti izdelkov (ProductFeature), proizvajalci (Producer), prodajalci (Vendor), ponudbe (Offer), opisi (Review) in opisovalci (Person). Izdelki so različnih tipov z različnimi lastnostmi. Vsak izdelek je proizveden s strani določenega proizvajalca. Prodajajo jih prodajalci iz več držav. Prodajalci ponujajo izdelke po datumsko omejenih ponudbah z različnimi časi dobave. Izdelki so opisani s strani različnih opisovalcev [31].

Izbrali smo RDF format N-Triples in izdelali SQL skripte z enakimi podatki. Testirali smo z naslednjimi količinami podatkov: 10.000, 100.000, 1.000.000 in 5.000.000. Merili smo čas izvedbe začetnega vstavljanja podatkov v podatkovno bazo za vsako izmed rešitev.

5.3 Testni scenariji

Štiri predvidene scenarije uporabe smo pretvorili v štiri testne scenarije, s pomočjo katerih smo poskušali ugotoviti, katera rešitev je najustreznejša za podano problemsko domeno. Vsak scenarij je obsegal tisoč operacij, ki se, odvisno od scenarija, delijo na branje, pisanje in brisanje. Za operacijo branja smo sestavili pet različnih SPARQL SELECT poizvedb in rezultat omejili na 10 trojčkov. V vsakem scenariju, ki vključuje branje, so se poizvedbe večkrat ponovile v naključnem zaporedju. Vse spremembe podatkov so potekale preko operacij iz SPARQL INSERT/UPDATE 1.1. Podatke smo preko SPARQL vstavljali z operacijo INSERT, brisali pa preko operacije DELETE. Izjema je le Neo4j, ki ne podpira vstavljanja ali brisanja podatkov preko SPARQL, zato smo pri njem za spremembe podatkov uporabili neposredne operacije nad grafom. Vsaka operacija je obsegala vnos oziroma brisanje enega trojčka.

5.3.1 Scenarij 1

V prvem scenariju se podatkovna baza uporablja samo za branje. Razdelitev operacij je sledeča:

- branj 100%
- pisanj 0%
- brisanj 0%

Merili smo celoten čas izvajanja in čas izvedbe posamezne operacije branja.

5.3.2 Scenarij 2

V drugem scenariju se podatkovna baza uporablja za branje in za vnašanje podatkov, predvideli smo uravnoteženo porazdelitev obeh operacij:

- branj 50%

- pisanj 50%
- brisanj 0%

Merili smo celoten čas izvajanja, čas izvajanja samo branj/pisanj in povprečen čas izvedbe posamezne operacije branja/pisanja.

5.3.3 Scenarij 3

V tretjem scenariju se nad podatkovno bazo izvajajo vse operacije, predvideli smo njihovo uravnoteženo porazdelitev:

- branj 33%
- pisanj 33%
- brisanj 33%

Merili smo celoten čas izvajanja, čas izvajanja samo branj/pisanj/brisanj in povprečen čas izvedbe posamezne operacije branja/pisanja/brisanja.

5.3.4 Scenarij 4

V četrtem scenariju smo predvideli večjo količino vstavljanja novih podatkov in manjši del brisanj že obstoječih podatkov. Razdelitev operacij je sledeča:

- branj 0%
- pisanj 80%
- brisanj 20%

Merili smo celoten čas izvajanja, čas izvajanja samo pisanj/brisanj in povprečen čas izvedbe posamezne operacije pisanja/brisanja.

5.4 Implementacija

Testiranje smo izvajali za vsako izmed količin podatkov posebej. Vsak scenarij smo izvedli petkrat in izmed pridobljenih časov izvajanja izločili najslabši in najboljši rezultat. Iz preostalih treh rezultatov smo izračunali povprečje, ki smo ga upoštevali kot končni čas, potreben za izvedbo vsakega izmed scenarijev.

5.4.1 D2RQ

Podatke smo shranjevali v lokalno podatkovno bazo MySQL, za dostop in operacije nad podatki smo uporabili Jena API. Testirali smo samo prvi scenarij, saj D2RQ ne podpira operacij vstavljanja in brisanja, ki so prisotni v naslednjih treh scenarijih. Tabele niso vsebovale sekundarnih indeksov. Privzetih nastavitev D2RQ strežnika nismo spreminjali.

5.4.2 Jena SDB

Podatke smo shranjevali v lokalno podatkovno bazo MySQL. Za dostop do podatkov smo uporabili SPARQL poizvedbe preko Jena API-ja. Uporabili smo privzete nastavitve Jena SDB, kjer se trojčki shranjujejo v tabelo s tremi stolpci: objekt (S), predikat (P) in predmet (O). Primarni ključ tabele je sestavljen iz vseh treh stolpcev tabele. Sekundarna indeksa sta sestavljena iz stolpcev P in O ter O in S.

5.4.3 Jena TDB

Podatke smo shranjevali v izvirno RDF podatkovno bazo na trdem disku. Za dostop do podatkov smo uporabili SPARQL poizvedbe preko Jena API-ja. Uporabili smo privzete nastavitve optimiziranja poizvedb.

5.4.4 Neo4j

Podatkovni model je bil shranjen na trdem disku, do podatkov smo dostopali preko Neo4j Java API-ja. Operacijo branja podatkov smo izvajali preko SPARQL poizvedb, vstavljanje in brisanje pa preko neposrednih operacij nad grafom, saj Neo4j ne podpira SPARQL UPDATE/DELETE. Vse operacije spreminjanja podatkov so se izvajale v transakciji. Uporabili smo privzete nastavitve podatkovne baze.

5.4.5 Owlim

Owlim sloni na Sesamovem podatkovnem modelu, zato smo se za dostop in operacije nad lokalnim repozitorijem odločili uporabiti Sesame openRDF API. Testirali smo Owlim LITE verzijo z lokalnim izvirnim repozitorijem. Uporabili smo privzete nastavitve z izklopljenim mehanizmom sklepanja, ker lahko ta zelo upočasni operacijo brisanja. Vzrok počasnega brisanja je v ponovitvi sklepanja za vse izpeljane trojčke po vsakem brisanju.

5.4.6 Sesame

Pri testiranju smo uporabili izvirni repozitorij, ki je bil shranjen na trdem disku. Do podatkov smo dostopali z uporabo SPARQL poizvedb preko Sesame API-ja. Dodali smo indekse `spoc`, `posc` in `cosp`, kjer s pomeni osebek, p predikat, o predmet in c kontekst.

5.4.7 Virtuoso

Virtuoso omogoča uporabo Sesame ali Jena API-ja. Odločili smo se za uporabo Jena API-ja za direktno performančno primerjavo podatkovnih baz z Jena TDB. Podatkovna baza je bila shranjena na lokalnem disku, do podatkov smo dostopali preko SPARQL poizvedb. Parameter `NumberOfBuffers` določa količino glavnega pomnilnika, ki je na voljo Virtuosu za shranjevanje podatkov, prebranih iz podatkovne baze. `MaxDirtyBuffers` parameter

določa maksimalno število spremenjenih blokov, preden jih je potrebno zapisati na disk. Povečali smo vrednosti parametrov `NumberOfBuffers` na 170.000 in `MaxDirtyBuffers` na 130.000, da sta ustrezala razpoložljivemu pomnilniškemu prostoru.

Poglavje 6

Analiza rezultatov

6.1 Čas vstavljanja podatkov

Začetno vstavljanje podatkov je daleč najhitreje opravil D2RQ preko MySQL podatkovne baze. Pri majhnih količinah podatkov je Jena TDB lahko še sledila hitrosti D2RQ, z večanjem količine podatkov pa se je razlika v porabljenem času vstavljanja povečevala od 2-krat pri 1.000.000 do skoraj 4-krat pri 5.000.000 trojčkov (Tabela 6.1). Jena SDB tako kot D2RQ uporablja MySQL podatkovno bazo, vendar je vstavljanje trojčkov skoraj 100-krat počasnejše pri Jena SDB. To nakazuje, da je za shranjevanje RDF podatkov primernejša namenska podatkovna baza. Za majhne količine podatkov, to je do 100.000 vnosov, so Neo4j, Owlim in Sesame dosegli podobne rezultate. Pri večjih količinah podatkov je Neo4j začel precej zaostajati in dosegel nekajkrat daljše čase vstavljanja.

Tabela 6.1: Časi vstavljanja podatkov v sekundah

	D2RQ	JenaSDB	JenaTDB	Neo4j	Owlim	Sesame	Virtuoso
10.000	1,100	3,673	1,045	13,239	6,920	8,358	100,054
100.000	2,800	44,597	4,998	58,480	63,302	59,370	963,969
1.000.000	23,500	1359,185	44,148	2222,800	578,833	680,320	9568,958
5.000.000	153,000	17319,072	595,167	34037,544	2969,284	4300,746	/

Iz grafa na sliki 6.1 lahko sklepamo, da bi se z večanjem količine podatkov razlika v času vstavljanja samo stopnjevala. OWLIM in Sesame, ki oba uporabljata Sesame OpenRDF API, sta dosegala približno enake rezultate do 1.000.000 trojčkov, nato je prevladala hitrost OWLIM repozitorijev. Vnos v Virtuosovo bazo je bil že pri majhnih količinah podatkov daleč najpočasnejši, pri 5.000.000 podatkov pa vnos sploh ni bil uspešno dokončan, saj je pri vnosu porabil celoten razpoložljiv pomnilniški prostor.

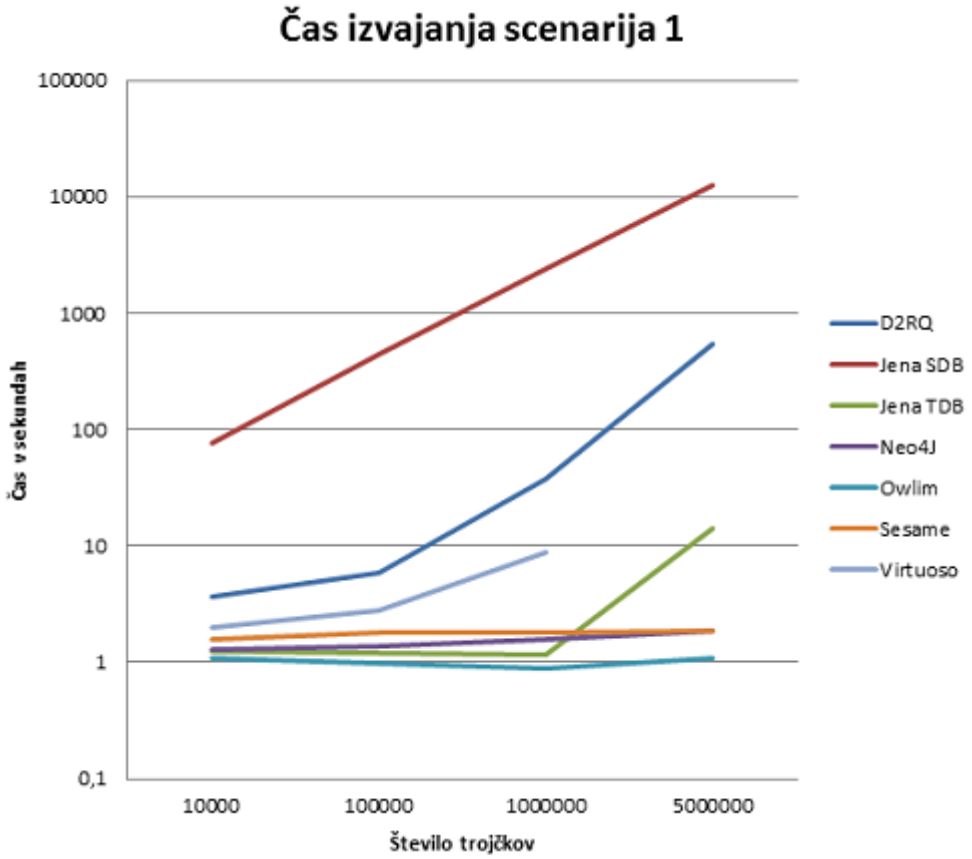
Slika 6.1: Časi vstavljanja trojčkov v podatkovne baze

6.2 Scenarij 1

V prvem scenariju smo izvajali samo operacije branja podatkov. Merili smo celoten čas izvajanja 1000 branj in povprečen čas izvedbe posamezne operacije branja.

D2RQ in Jena SDB, ki za shranjevanje podatkov uporabljata MySQL podatkovno bazo, sta za izvajanje poizvedb porabila precej daljši čas od ostalih rešitev. Skupni čas izvajanja se je pri teh dveh podatkovnih bazah najhitreje povečeval z naraščanjem količine podatkov (slika 6.2). Owlim je dosegel najboljše rezultate pri vseh testiranih količinah podatkov. Skupni čas izvajanja poizvedb preko Owlim-a se z naraščanjem količine podatkov bistveno ni podaljševal in je ostal okoli sekunde. Najslabše rezultate od rešitev z izvirnimi repozitoriji je dosegel Virtuoso, in sicer skorajda 10-krat počasnejši čas izvajanja od najhitrejšega Owlima že pri 1.000.000 podatkov (Tabela 6.4). Jena SDB je s povprečnim časom branja več kot 12 sekund pri 5.000.000 vnosih daleč najpočasnejša in neprimerna za uporabo v poslovni aplikaciji (Tabela 6.5). Owlim, Sesame, Neo4j in Jena TDB z zelo nizkimi povprečnimi časi na poizvedbo zadostujejo tudi potrebam poslovnih aplikacij.

V testu Berlin SPARQL Benchmark V2 so testirali Jena TDB, Sesame, Virtuoso in D2RQ podatkovne baze. Tako kot pri našem testiranju je med njimi najpočasneje s scenarijem opravil D2RQ, Sesame je bil najhitrejši. V nasprotju z našimi rezultati, kjer je bil Virtuoso počasnejši od Jena TDB, je pri njihovem testu Virtuoso hitreje izvedel poizvedbe [32].



Slika 6.2: Čas izvajanja scenarija 1

Tabela 6.2: Časi izvedbe scenarija 1 za 10.000 podatkov v sekundah

	D2RQ	Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
Čas izvajanja	3,640	76,322	1,229	1,274	1,095	1,594	1,980
Povpr. čas br.	0,004	0,076	0,001	0,001	0,001	0,002	0,002

Tabela 6.3: Časi izvedbe scenarija 1 za 100.000 podatkov v sekundah

	D2RQ	Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
Čas izvajanja	5,895	436,902	1,198	1,371	0,984	1,779	2,766
Povpr. čas br.	0,006	0,437	0,001	0,001	0,001	0,002	0,003

Tabela 6.4: Časi izvedbe scenarija 1 za 1.000.000 podatkov v sekundah

	D2RQ	Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
Čas izvajanja	37,503	2435,792	1,162	1,583	0,889	1,829	8,842
Povpr. čas br.	0,038	2,436	0,001	0,002	0,001	0,002	0,009

Tabela 6.5: Časi izvedbe scenarija 1 za 5.000.000 podatkov v sekundah

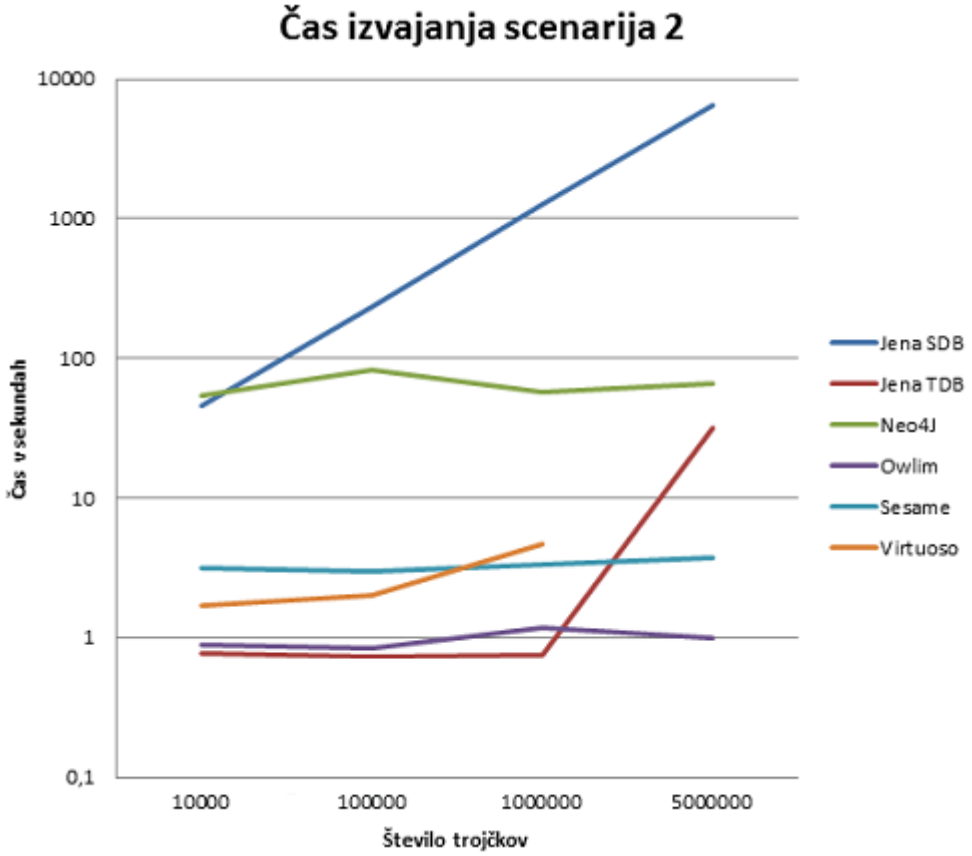
	D2RQ	Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
Čas izvajanja	536,094	12390,580	14,278	1,887	1,085	1,864	/
Povpr. čas br.	0,536	12,3906	0,014	0,002	0,001	0,002	/

6.3 Scenarij 2

Scenarij je vseboval uravnoteženo porazdelitev operacij branja in pisanja. Izvedli smo 500 branj in 500 vstavljanj podatkov. Merili smo celoten čas izvajanja, čas izvajanja samo branj/pisanj in povprečen čas izvedbe posamezne operacije branja/pisanja.

Najboljše rezultate sta pri 10.000 trojčkih zabeležila Jena TDB in Owlim, zaradi hitrejšega branja je boljši rezultat dosegla Jena TDB (Tabela 6.6). Daleč najpočasnejša sta bila Neo4j in Jena SDB. Neo4j zaradi izredno počasnega časa vstavljanja, ki je več kot 10-krat daljši kot pri ostalih, Jena SDB pa zaradi počasnega branja. Pri 100.000 in 1.000.000 trojčkov sta Jena TDB in Owlim beležila približno enake skupne čase izvajanja, približno dve sekundi počasneje je s scenarijem opravil Sesame (tabeli 6.7 in 6.8). S povečevanjem količine podatkov se je čas branja v Jena SDB hitro povečeval in pri 5.000.000 podatkov je bil povprečni čas enega branja skorajda enak celotnemu času vstavljanja (Tabela 6.9). Čas izvajanja scenarija v Jena TDB se je glede na 1.000.000 podatkov pri 5.000.000 skorajda kot 40-krat podaljšal, kar je vidno kot strm vzpon premice na grafu 6.3. Vzrok je predvsem v počasnejšem branju, ki je bilo pri Jena TDB skorajda 3-krat počasnejše od vstavljanja. Sesame je kot drugi najhitrejši pri 5.000.000 podatkov dosegel približno 4-krat

počasnejši rezultat kot Owlim. Vzrok je v 4-krat počasnejšem vstavljanju v primerjavi z branjem. Owlim je pokazal najboljše razmerje med hitrostjo branja in vstavljanja in je s scenarijem opravil najhitreje.



Slika 6.3: Čas izvajanja scenarija 2

Tabela 6.6: Časi izvedbe scenarija 2 za 10.000 podatkov v sekundah

	Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
Čas izvajanja	45,095	0,771	54,660	0,900	3,132	1,712
Čas branja	38,396	0,433	0,964	0,567	0,658	0,780
Čas vstavlj.	6,699	0,337	53,696	0,333	2,475	0,932
Povpr. čas br.	0,077	0,001	0,002	0,001	0,001	0,002
Povpr. čas vst.	0,013	0,001	0,107	0,001	0,005	0,002

Tabela 6.7: Časi izvedbe scenarija 2 za 100.000 podatkov v sekundah

	Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
Čas izvajanja	235,267	0,727	83,060	0,832	2,960	2,012
Čas branja	228,882	0,418	0,490	0,433	0,671	1,083
Čas vstavlj.	6,385	0,309	82,570	0,398	2,289	0,928
Povpr. čas br.	0,458	0,001	0,001	0,001	0,001	0,002
Povpr. čas vst.	0,013	0,001	0,165	0,001	0,005	0,002

Tabela 6.8: Časi izvedbe scenarija 2 za 1.000.000 podatkov v sekundah

	Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
Čas izvajanja	1277,367	0,760	57,076	1,173	3,297	4,663
Čas branja	1270,262	0,442	0,616	0,538	0,780	3,730
Čas vstavlj.	7,105	0,319	56,460	0,635	2,518	0,932
Povpr. čas br.	2,541	0,001	0,001	0,001	0,002	0,007
Povpr. čas vst.	0,014	0,001	0,113	0,001	0,005	0,002

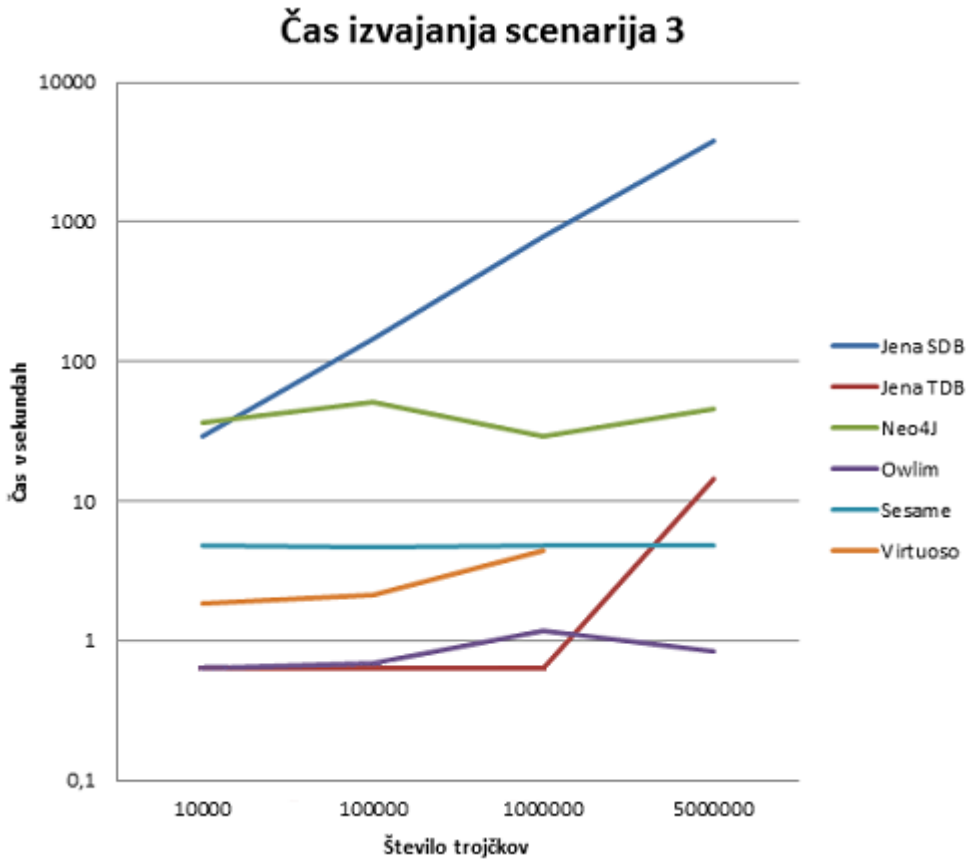
Tabela 6.9: Časi izvedbe scenarija 2 za 5.000.000 podatkov v sekundah

	Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
Čas izvajanja	6456,452	31,784	66,763	0,985	3,687	/
Čas branja	6443,575	22,317	0,701	0,456	0,709	/
Čas vstavlj.	12,876	9,467	66,062	0,528	2,978	/
Povpr. čas br.	12,887	0,045	0,001	0,001	0,001	/
Povpr. čas vst.	0,026	0,019	0,132	0,001	0,006	/

6.4 Scenarij 3

Scenarij je vseboval uravnoteženo porazdelitev operacij branja, pisanja in brisanja. Izvedli smo 333 branj, 333 pisanj in 333 brisanj. Merili smo celoten čas izvajanja, čas izvajanja samo branj/pisanj/brisanj in povprečen čas izvedbe posamezne operacije branja/pisanja/brisanja.

Najboljše rezultate sta pri 10.000 podatkih zabeležila Jena TDB in Owlim, sledi jima Virtuoso (Tabela 6.10). Najslabše rezultate je dosegel Neo4j zaradi počasnega vstavljanja. Pri 100.000 podatkih je Jena SDB že zaostala za Neo4j (Tabela 6.11) in od 1.000.000 podatkov naprej se je razlika v hitrosti samo povečevala. Vzrok je branje podatkov, ki je pri Jena SDB tudi do 1000-krat počasnejše kot pri ostalih. Celoten čas izvajanja scenarija se pri Jena SDB s količino podatkov linearno povečuje. Jena TDB je do vključno 1.000.000 podatkov z izvajanjem scenarija opravila najhitreje. Zaradi 10-krat počasnejšega branja in brisanja v primerjavi z vstavljanjem je nato precej zaostala (Tabeli 6.12 in 6.13). Owlim in Sesame sta dosegala pri vseh količinah podatkov približno enake rezultate (Slika 6.4).



Slika 6.4: Čas izvajanja scenarija 3

Tabela 6.10: Časi izvedbe scenarija 3 za 10.000 podatkov v sekundah

	Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
Čas izvajanja	29,169	0,627	36,893	0,640	4,812	1,852
Čas branja	23,037	0,296	0,279	0,299	0,367	0,496
Čas vstavljanja	3,603	0,161	35,314	0,198	1,597	0,614
Čas brisanj	2,529	0,170	1,301	0,143	2,848	0,742
Povpr. čas br.	0,069	0,001	0,001	0,001	0,001	0,001
Povpr. čas vst.	0,011	0,001	0,106	0,001	0,005	0,002
Povpr. čas bris.	0,008	0,001	0,004	0,001	0,009	0,002

Tabela 6.11: Časi izvedbe scenarija 3 za 100.000 podatkov v sekundah

	Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
Čas izvajanja	143,062	0,628	50,553	0,698	4,738	2,135
Čas branja	137,030	0,289	0,318	0,329	0,393	0,799
Čas vstavljanja	3,763	0,174	47,234	0,190	1,689	0,598
Čas brisanj	2,269	0,165	3,001	0,179	2,656	0,738
Povpr. čas br.	0,412	0,001	0,001	0,001	0,001	0,002
Povpr. čas vst.	0,011	0,001	0,142	0,001	0,005	0,002
Povpr. čas bris.	0,007	0,001	0,009	0,001	0,008	0,002

Tabela 6.12: Časi izvedbe scenarija 3 za 1.000.000 podatkov v sekundah

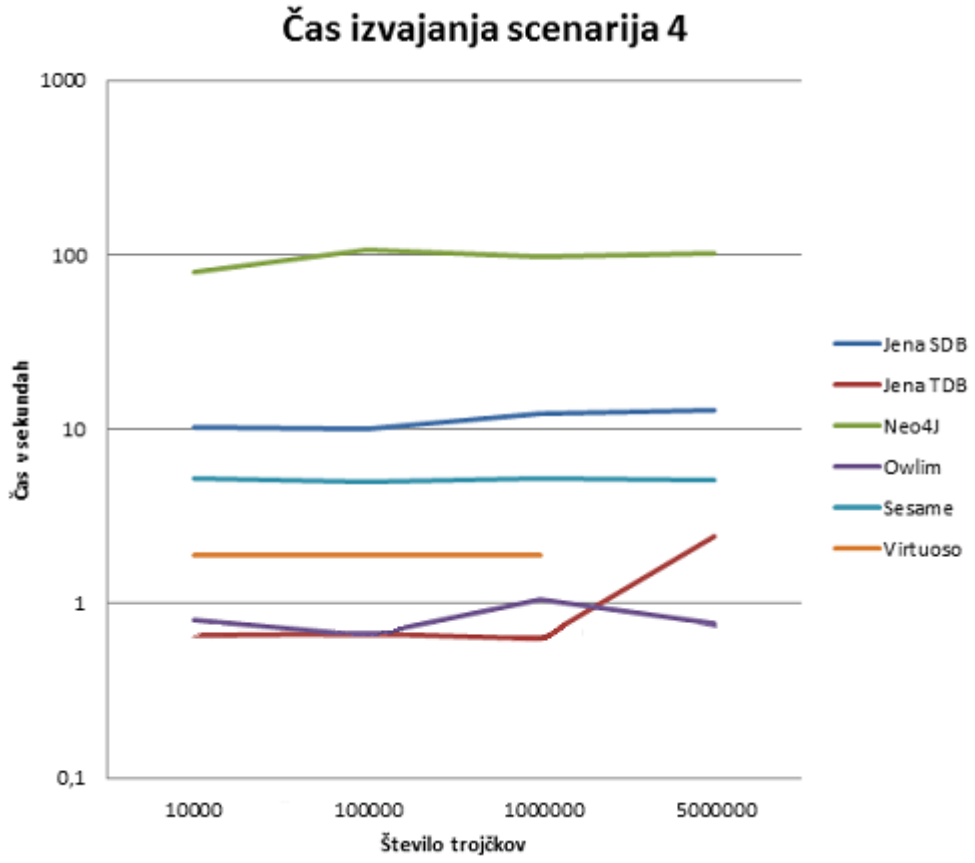
	Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
Čas izvajanja	794,032	0,640	28,867	1,173	4,832	4,439
Čas branja	785,084	0,299	0,386	0,365	0,507	3,108
Čas vstavljanja	5,215	0,160	26,375	0,382	1,553	0,596
Čas brisanj	3,733	0,182	2,106	0,426	2,772	0,736
Povpr. čas br.	2,358	0,001	0,001	0,001	0,002	0,009
Povpr. čas vst.	0,016	0,001	0,079	0,001	0,005	0,002
Povpr. čas bris.	0,011	0,001	0,006	0,001	0,008	0,002

Tabela 6.13: Časi izvedbe scenarija 3 za 5.000.000 podatkov v sekundah

	Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
Čas izvajanja	3802,987	14,384	46,340	0,849	4,870	/
Čas branja	3795,300	7,241	0,482	0,395	0,476	/
Čas vstavljanja	4,828	0,846	40,582	0,224	1,610	/
Čas brisanj	2,859	6,297	5,276	0,230	2,785	/
Povpr. čas br.	11,397	0,022	0,001	0,001	0,001	/
Povpr. čas vst.	0,014	0,003	0,122	0,001	0,005	/
Povpr. čas bris.	0,009	0,019	0,016	0,001	0,008	/

6.5 Scenarij 4

Scenarij je vseboval večji delež pisanj in manjši delež brisanj. Izvedli smo 800 operacij pisanja in 200 operacij brisanja podatkov. Merili smo celoten čas izvajanja, čas izvajanja samo pisanj/brisanj in povprečen čas izvedbe posamezne operacije pisanja/brisanja.
Najhitreje sta s scenarijem opravila Jena TDB in Owlim, nekoliko hitrejša je bila do 1.000.000 podatkov Jena TDB s krajšim časom vstavljanja podatkov. Najpočasneje je scenarij izvedel Neo4j zaradi počasnega vstavljanja podatkov (Tabele 6.14, 6.15, 6.16 in 6.17). Pri 100.000 podatkih je Owlim za malenkost prehitel Jena TDB, pri 5.000.000 podatkov je bil Owlim približno 3-krat hitrejši od Jena TDB. Sesame je preko celega scenarija zaostajal za najhitrejšima zaradi nekajkrat počasnejšega vstavljanja podatkov. Povprečni čas izvedbe ene operacije brisanja je zelo majhen pri vseh rešitvah, nekoliko večji je čas vstavljanja pri Neo4j. Sesame in Virtuoso sta skozi celoten scenarij imela konstanten čas izvajanja (Slika 6.5).



Slika 6.5: Čas izvajanja scenarija 4

Tabela 6.14: Časi izvedbe scenarija 4 za 10.000 podatkov v sekundah

	Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
Čas izvajanja	10,196	0,654	79,818	0,803	5,203	1,910
Čas vstavljanja	8,823	0,510	78,841	0,680	3,550	1,437
Čas brisanja	1,372	0,143	0,977	0,122	1,653	0,437
Povpr. čas vst.	0,011	0,001	0,099	0,001	0,004	0,002
Povpr. čas bris.	0,007	0,001	0,005	0,001	0,008	0,002

Tabela 6.15: Časi izvedbe scenarija 4 za 100.000 podatkov v sekundah

	Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
Čas izvajanja	9,954	0,676	105,452	0,653	4,976	1,890
Čas vstavljanja	8,526	0,506	102,624	0,521	3,417	1,422
Čas brisanja	1,428	0,170	2,828	0,132	1,559	0,469
Povpr. čas vst.	0,011	0,001	0,128	0,001	0,004	0,002
Povpr. čas bris.	0,007	0,001	0,014	0,001	0,008	0,002

Tabela 6.16: Časi izvedbe scenarija 4 za 1.000.000 podatkov v sekundah

	Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
Čas izvajanja	12,200	0,634	97,724	1,062	5,169	1,896
Čas vstavljanja	10,267	0,457	94,872	0,707	3,557	1,421
Čas brisanja	1,933	0,176	2,852	0,355	1,612	0,474
Povpr. čas vst.	0,013	0,001	0,119	0,001	0,004	0,002
Povpr. čas bris.	0,010	0,001	0,014	0,002	0,008	0,002

Tabela 6.17: Časi izvedbe scenarija 4 za 5.000.000 podatkov v sekundah

	Jena SDB	Jena TDB	Neo4j	Owlim	Sesame	Virtuoso
Čas izvajanja	12,748	2,415	103,048	0,761	5,121	/
Čas vstavljanja	10,948	2,229	100,075	0,497	3,569	/
Čas brisanja	1,800	0,186	2,973	0,264	1,551	/
Povpr. čas vst.	0,014	0,003	0,125	0,001	0,004	/
Povpr. čas bris.	0,009	0,001	0,015	0,001	0,008	/

6.6 Zaključki

Prvi scenarij obsega samo branje podatkov in najbolj primerni za take operacije so Neo4j, Owlim in Sesame. Vsi imajo izredno nizek povprečni čas branja, le nekaj tisočink sekunde. To je za uporabnika aplikacije neopazno in omogoča odlično uporabniško izkušnjo. Jena SDB in D2RQ pa sta z dolgimi povprečnimi časi branja neprimerna za uporabo v poslovni aplikaciji, kjer je odzivnost zelo pomembna.

V drugem scenariju, z enako količino branja in vstavljanja, sta se za najprimernejšo izbiro izkazala Owlim in Sesame. Oba imata nizek povprečni čas branja in vstavljanja, kar dovoljuje nemoteno delovanje aplikacije. Jena SDB pa je z nekaj sekund dolgim časom branja neprimerna za uporabo v poslovni aplikaciji.

Na podlagi rezultatov izvedbe tretjega scenarija lahko zaključimo, da je pri enakomerni obremenitvi podatkovne baze najprimernejši za uporabo Owlim. Ponuja izredno nizke čase branja, vstavljanja in brisanja. Dobro uporabniško izkušnjo omogoča tudi Sesame s časom izvajanja posamezne operacije le nekaj tisočink sekunde. Jena SDB zaradi zelo počasnega branja ne more zadostiti potrebam, ki naj bi jih aplikacija zadovoljevala.

V aplikacijah, kjer je veliko spreminjanja podatkov, je glede na rezultate četrtega scenarija za uporabo najprimernejši Owlim. Konstantna časa vstavljanja in brisanja pri vseh količinah podatkov omogočata odlično delovanje aplikacije. Primerna za uporabo sta tudi malenkost počasnejša Jena TDB in Sesame, saj je povprečen čas posamezne operacije le nekaj tisočink sekunde. Neo4j zaradi počasnega vstavljanja podatkov ne more zagotoviti dobre uporabniške izkušnje in ni primeren za uporabo v aplikaciji.

Poglavje 7

Sklepne ugotovitve

NoSQL podatkovne baze so razmeroma novo in hitro se razvijajoče področje z malo standardizacije. To sicer omogoča večjo svobodo pri implementaciji, vendar pa prinaša tudi določene slabosti. Namestitev in tudi uporaba je bolj zapletena kot pri relacijskih podatkovnih bazah in se večinoma od ponudnika do ponudnika zelo razlikuje. Orodja so slabše razvita in ponujajo precej manj funkcionalnosti za razliko od ustaljenih in standardiziranih relacijskih podatkovnih baz.

V diplomski nalogi smo preverili obnašanje triplestore-ov pri razmeroma majhni količini podatkov glede na predvidene sposobnosti takih podatkovnih baz, vendar so se že tu pojavile velike razlike v hitrosti delovanja. Ugotovili smo, da rešitve, ki uporabljajo za shranjevanje relacijske podatkovne baze, performančno ne morejo konkurirati namenskim NoSQL bazam. Med NoSQL podatkovnimi bazami se je za najbolj vsestranskega izkazal Owlim, pri katerem so časi branja, vstavljanja in brisanja približno enako dolgi.

Zaradi velike raznolikosti implementacij triplestore podatkovnih baz je težko izbrati univerzalno rešitev, saj imajo vse tako prednosti kot slabosti. Pred odločitvijo je zato najpametneje preizkusiti nekaj rešitev, ki po funkcionalnostih najbolj odgovarjajo našim potrebam. Pri odločitvi naj bodo v pomoč kriteriji, določeni v diplomskem delu. Rezultati, pridobljeni v okviru diplomske naloge, so nastali na podlagi testiranja na specifični strojni konfigu-

raciji za točno določeno problemsko domeno. Posledično naj služijo predvsem kot opora pri iskanju primerne implementacije podatkovne baze, za končno odločitev pa priporočamo lastno testiranje na ciljni strojni konfiguraciji.

Slike

Tabele

Literatura

[1] B. Haslhofer, E. Momeni, B. Schandl, S. Zander. (2011, Mar.). Europeana RDF Store Report. University of Vienna. Vienna. Dostopno na: http://eprints.cs.univie.ac.at/2833/.

[2] R. Žlender. (2011). Primerjava zmogljivosti nerelacijskih podatkovnih baz. Faktulteta za računalništvo in informatiko. Ljubljana.

[3] (2012) Podatkovne baze NoSQL . Dostopno na: http://lovro.lpt.fri.uni-lj.si/resources/research/papers/nosql.pdf.

[4] (2012) Notation3. Dostopno na: http://en.wikipedia.org/wiki/Notation3.

[5] J. Hebeler, M.Fisher, R. Blace, A. Perez-Lopez, *Semantic web programming*, Indianapolis: Wiley Publishing, 2009. pogl. 3.

[6] T. Segaran, C. Evans, J.Taylor, *Programming the symantic web*, Sebastopol: O'Reilly, 2009. pogl. 4.

[7] (2012) N-Triples. Dostopno na: http://en.wikipedia.org/wiki/N-Triples.

[8] (2011) RDF JSON. Dostopno na: http://docs.api.talis.com/platform-api/output-types/rdf-json.

[9] (2012) SPARQL. Dostopno na: http://en.wikipedia.org/wiki/SPARQL.

[10] (2012) Defining-a-Property-Graph. Dostopno na: `https://github.com/tinkerpop/gremlin/wiki/Defining-a-Property-Graph`.

[11] (2012) Neo4j manual. Dostopno na: `http://docs.neo4j.org/`.

[12] (2012) Graph Databases, NOSQL and Neo4j. Dostopno na: `http://www.infoq.com/articles/graph-nosql-neo4j`.

[13] (2012) Price list. Dostopno na: `http://www.neotechnology.com/price-list/`.

[14] (2012) Capacity. Dostopno na: `http://docs.neo4j.org/chunked/stable/capabilities-capacity.html`.

[15] (2009) Short introduction to Sesame's components. Dostopno na: `http://www.openrdf.org/doc/sesame2/2.0.1/users/ch03.html`.

[16] (2012) Licenca BSD. Dostopno na: `http://opensource.org/licenses/BSD-3-Clause`.

[17] (2012) Jira. Dostopno na: `http://www.atlassian.com/software/jira/overview/`.

[18] (2012) Jena TDB arhitecture. Dostopno na: `http://jena.apache.org/documentation/tdb/architecture.html`.

[19] (2012) TDB Java API. Dostopno na: `http://jena.apache.org/documentation/tdb/java_api.html`.

[20] (2012) SDB Databases Supported. Dostopno na: `http://jena.apache.org/documentation/sdb/databases_supported.html`.

[21] (2007) D2RQ — Lessons Learned. Dostopno na: `http://www.w3.org/2007/03/RdfRDB/papers/d2rq-positionpaper/`.

[22] (2012) RDF Triple Store FAQ. Dostopno na: `http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSRDFFAQ`.

[23] (2011) Large Triple Stores. Dostopno na: `http://www.w3.org/wiki/LargeTripleStores`.

[24] (2012) Virtuoso Pricing. Dostopno na: `http://virtuoso.openlinksw.com/pricing/`.

[25] (2012) ORDI SG. Dostopno na: `http://www.ontotext.com/ordi`.

[26] (2012) Owlim Lite Dostopno na: `http://owlim.ontotext.com/display/OWLIMv51/OWLIM-Lite+Fact+Sheet`.

[27] (2012) Owlim SE Dostopno na: `http://owlim.ontotext.com/display/OWLIMv51/OWLIM-SE+Fact+Sheet`.

[28] (2012) Owlim Enterprise. Dostopno na: `http://owlim.ontotext.com/display/OWLIMv51/OWLIM-Enterprise+Fact+Sheet`.

[29] (2012) OWLIM Licensing and Technical Support. Dostopno na: `http://www.ontotext.com/owlim/pricelist`.

[30] (2010) Data Generator and Test Driver. Dostopno na: `http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/spec/BenchmarkRules/index.html#datagenerator`.

[31] (2009) The Berlin SPARQL Benchmark. Dostopno na: `http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/Bizer-Schultz-Berlin-SPARQL-Benchmark-IJSWIS.pdf` .

[32] (2008) Berlin SPARQL Benchmark V2. Dostopno na: `http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/V2/results/index.html#results` .