Towards Using NOSQL Database Systems for Semantic Web Data

By

Mahek Hanfi

NUST201362761MSEECS61413F

Supervisor

Dr. Khalid Latif

Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree of Masters in Computer Science (MS CS)

In

School of Electrical Engineering and Computer Science,

National University of Sciences and Technology (NUST),

Islamabad, Pakistan.

(July 2015)

Approval

It is certified that the contents and form of the thesis entitled "**Towards Using NOSQL Database Systems for Semantic Web Data**" submitted by **Mahek Hanfi** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Khalid Latif**

Signature: _____

Date: _____

Committee Member 1: **Dr. Hamid Mukhtar**

Signature: _____

Date: _____

Committee Member 2: **Dr. Farooq Ahmad**

Signature: _____

Date: _____

Committee Member 3: **Dr. Asad Anwar**

Signature: _____

Date: _____

Dedication

With affection and gratitude, I would like to dedicate this thesis to my family and teachers who have been a continuous source of inspiration and motivaion for me and supported me all the way throughout my education.

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Mahek Hanfi**

Signature: _____

Acknowledgment

In the name of Allah the beneficent and merciful, on whom we all are dependent for eventual support and guidance.

I would like to express my immense gratitude to my supervisor Dr. Khalid Latif, who has guided and supported me throughout my thesis work and allow me to work in my own way and polish my skills. I highly appreciate his help in technical writing.His mentorship was predominant in sustaining versatile experience in my long term career goals.

I would also like to thank my committee members, Dr. Hamid Mukhtar, Dr. Farooq Ahmad and Dr. Asad Anwar who continuously guided me in my thesis, and provided their valuable suggestions and encouragement.

Finally I would like to thanks my siblings,friends and colleagues who helped and motivated me in completing my thesis.

Table of Contents

List of Tables

List of Figures

Abstract

The Semantic Web allows data to be shared and reused through a mechanism of URIs and resource description. RDF, as a standard for describing resources and data, models information in the form of subject-predicate-object triples. This model can be viewed as a graph; subjects and objects are vertices or nodes and are connected through predicates as edges of the graph. Relational database systems are not efficient in processing Semantic Web data. With the massive increase in graph data, attributed to the growth of social networks, many native graph database systems have surfaced to replace Relational databases. Neo4j is the leading and very scalable graph database system that can store up to millions of nodes and relationships. Storage of Semantic Web data in Neo4J requires mapping of RDF constructs to Neo4j. Henceforth, retrieval requires mapping of the SPARQL query language for Semantic Web, with the Cypher query language for Neo4j. Logical inference or reasoning is a fundamental building block of the Semantic Web. A complementary standard in the Semantic Web architecture, known as RDF Schema or RDF/S for short, provides a set of inference rules that define the mechanism for discovering and generating new relationships based on exist-

ing data. An efficient and scalable implementation of RDF/S inference rules is also missing in most of the graph database systems. This paper contributes three pieces of the jigsaw to achieve a scalable RDF storage system. First a mechanism for mapping RDF constructs with Neo4J is outlined. Secondly, SPARQL query language is mapped with Cypher for efficient retrieval. And finally, RDF/S inference rules are implemented to realize logical inference capability.

Chapter 1

Introduction

This chapter gives the basic idea of the concepts involved in this research. It also presents the background and motivation for this study. Moreover, it provides an idea of expected results, and methodology to get and evaluate the results. Finally, it presents the structure of this thesis document.

1.1 Introduction

Graphs represent entities as nodes and the ways these entities relate to the world as relationships. A graph database; uses graph structures with nodes and edges to represent and store data. It is capable of elegantly representing any kind of data in a highly accessible way. With the massive increase in graph data many native graph database systems are surfaced to

replace relational databases [1] . Neo4j is one of the leading and highly scalable graph databases that can store upto billions of nodes and relationships (edges). It stores data in form of nodes and relationships rather than tables. Neo4j follow the structure of property graph. Neo4j implements property graph model. In property graph, each node is identified with a unique identifier. Each edge is identified with a unique identifier and a labeled string. Nodes and edges in a property graph can have collection of key/value properties [2]. Neo4j data is accessed through cypher query language which is based on patterns. Patterns are used to match desired graph structure. Semantic Web on the other is a set of standards that allows us to share and reuse data across the internet. These standards promote standard data models (also known as ontologies) on the Web. RDF (Resource Description Framework) is the most fundamental semantic web standard that represents data in the form of triples ¡subject, predicate, object¿ where predicate is edge between subject and object [3]. RDF graphs are queried using SPARQL query language. RDF data can be viewed as graph with subject and object as nodes and predicates as edges of the graph [4]. Moreover, there are other standards in semantic web that allow us to define RDF data models. RDFS is one of such standards that provide a vocabulary to define RDF data. It also provides set of inference or reasoning rules that are used to determine what additional facts can be implied if other facts are known. Jena is a Semantic Web framework for Java that can store, modify and retrieve data from RDF graphs. The framework also supports SPARQL (RDF query language) and inference over RDF data. It has built in support for RDFS and OWL reasoners [5]. In this paper, we have proposed and implemented a framework that

maps RDF data to property graph and store it in neo4j. Our work focuses on RDF triple mapping to Neo4j graph and vice versa. The framework includes the implementation of RDFS reasoner in neo4j and SPARQL query mapping to neo4j's cypher queries.

1.2 Motivation

Attributed to the growth of social networks, the need to store and process graph data has increased many folds in last few years. Every enterprise needs a scalable system to manage the data overflow that too in intuitive way based on emerging industry standards and mature tools. Many enterprises are using linked data for knowledge representation. Graph databases on the other hand can model real world problems and are scalable. This present us the need of integration of semantic web data with the graph databases.

1.3 Objective

The objective of this thesis is to :

- Map RDF constructs with the graph database.
- Implement inference engine to add inferred RDF statements based on RDFS rules

- Map SPARQL query language constructs with the graph database query language constructs.
- Efficient storage and retrieval of RDF for millions of statements.

1.4 Problem Statement

RDF data cannot be directly stored in graph stores. So while storing RDF data in graph stores we have following challenges:

- RDF data is in triple form (subject, predicate, object) while graph stores data in form of nodes and relationships.
- RDF data is queries using SPARQL query language while graph stores have their own query languages which are based on graph patterns.
- Many graph stores doesnot support inference or reasoning over data.

1.5 Contribution

For the integration of semantic web data with graph stores following contributions are made:

- Mapped and stored RDF data (subject-predicate-object) triple to neo4j constructs (nodes and edges).

- Retrieve data using SPARQL query language - mapped SPARQL to cypher query language.
- Mapped neo4j constructs to RDF
- Implemented RDF/S rules for semantic reasoning in neo4j.

1.6 Evaluation

The evaluation is based on following perspectives.

- To evaluate the loading and querying performance of data access layer five different datasets were generated for Lehigh University Benchmark(LUBM)
- Inference performance is evaluated using positive and negative entailment tests.

1.7 Methodology

As per methodology, firstly RDF data (subject, predicate, object) is mapped and stored in neo4j (nodes and relationships) and then retrieved. For retrieval of RDF from neo4j, SPARQL query is mapped with cypher query language. Lastly, RDFS inference rules are applied on neo4j data.

1.8 Expected Results

As per proposed methodology, the data access layer should store RDF data in neo4j and query using SPARQL query languagae. And it should also infer new facts from existing data using RDFS inference rules.

1.9 Structure

Rest of the thesis is structured as follows:

- Chapter 2: Semantic Web and Graph databases
- Chapter 3: Related Work
- Chapter 4: Mapping of RDF constructs with Neo4j
- Chapter 5: System Architecture and Implementation is followed.
- Chapter 6: Evaluation and Results
- Chapter 7: Conclusion.

Chapter 2

Semantic Web and Graph databases

2.1 Semantic Web

Semantic web also known as Web 3.0 is **Web of Data**. The main goal of Web 3.0 is to enable sharing and supporting trusted interactions over the internet. Web 3.0 is an extension of web through standards of World Wide Web Consortium (W3C). Semantic Web is W3C's recommendation for linked data. For handling of data, semantic web allows to create data that can be stored on web, build vocabularies and write rules for inferring new facts from existing data. The semantic web is the collection of standard technologies to realize Web of data. The technology stack of semantic web data is shown if figure. In semantic web data is shared and reused through URIs

and resource description. Knowledge in semantic web is represented using different knowledge representation standards. Some semantic web standards are described below.

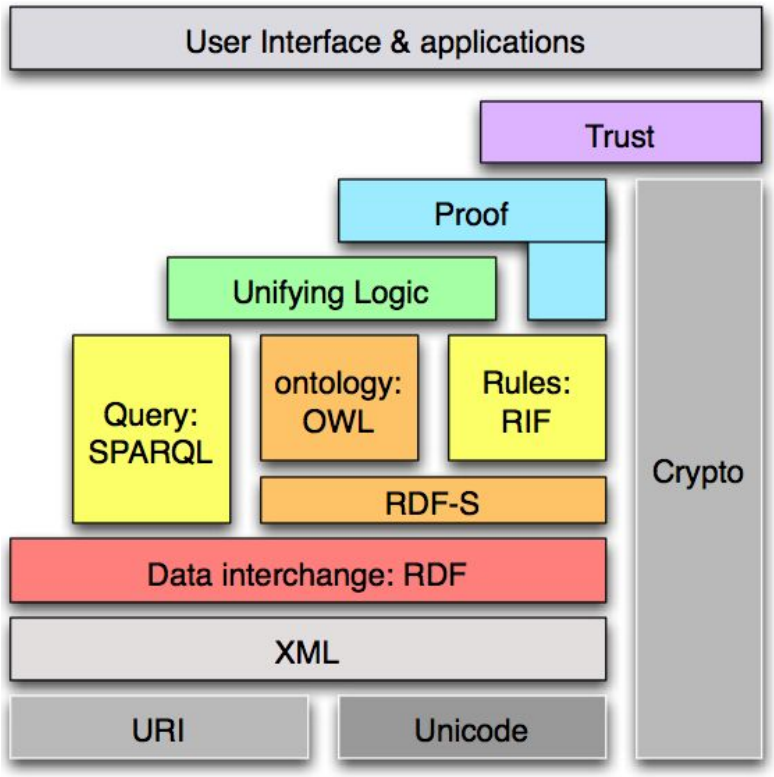

Figure 2.1: Semantic Web Architecture

2.2 RDF

Resource Description Framework (RDF) is a W3C recommendation for describing information contained in the web. It is a data model for describing meta data (data about data). RDF represents knowledge in form of triple (subject, predicate, object). The subject-predicate-object triple form

a RDF statement. The basic building blocks of RDF are resources and properties. **Resources** are always uniquely identified using URIs. **Blank nodes** don't have a unique identifier and can't be accessed externally. **Literal values** are atomic values. **Properties(predicates)** are the relationship between resoucres and they are uniquely identified using URI. A simple RDF statement is represented in following figure.

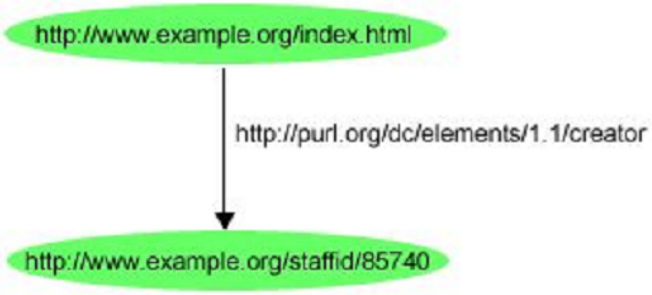

Figure 2.2: RDF Statement

Subject and predicate are always resources (URIs and blank nodes) while object can be a resource or literal. Blank nodes can be introduced to represent multi valued relationships with dependent nodes. Blank nodes are for resources that don't need to be referred externally. Literals are atomic values and consist of three parts; the actual value, data type and language of value.

2.3 RDFS

RDF Schema is an ontology language that describes specific domains . RDF Schema is described by:

- Classes and properties
- Class hierarchies and inheritence
- Property hierarchies

Classes are set of objects. Members of classes are called the *instances*.Forexample Person is a class and *John* is its instance. Classes are resources and uniquely identified through URI. The relationship between instances and classes are defined using **rdf:type**.

:John rdf:type :Person

Classes are organized in hierarchies.

- A is a subclass of B if every instance of A is also an instance of B.
- A class may have multiple super classes.

:Student rdfs:subClassOf :Person

Properties are the relationship between subject and object resources. For example in the above triple ***rdfs:subClassOf*** is a property that relate Student class with Person class. RDF properties are defined globally.

Declaration:

:studiedAt rdf:type rdf:Property

Use:

:John :studiedAt :MIT

Domain of a property declares the class of *subject* and **Range** of a property declares the class of *object* in a triple whose second component is the *predicate*.

:studiedAt rdfs:domain :Person

:studiedAt rdfs:range :University

2.4 Inference

Inference is the ability to infer new facts from the existing facts. Reasoning on knowledge to discover new facts is an important process on

Table 2.1: RDFS Inference Rules

Rule	If KB contains	Add
1	s p o (if o is a literal)	:n rdf:type rdfs:Literal
2	p rdfs:domain x & s p o	s rdf:type x
3	p rdfs:range x & s p o	o rdf:type x
4a	s p o	s rdf:type rdfs:Resource
4b	s p o	o rdf:type rdfs:Resource
5	p rdfs:subPropertyOf q & q rdfs:subPropertyOf r	p rdfs:subPropertyOf r
6	p rdf:type rdf:Property	p rdfs:subPropertyOf p
7	s p o & p rdfs:subPropertyOf q	s q o
8	s rdf:type rdfs:Class	s rdfs:subClassOf rdfs:Resource
9	s rdf:type x & x rdfs:subClassOf y	s rdf:type y
10	s rdf:type rdfs:Class	s rdfs:subClassOf s
11	x rdfs:subClassOf y & y rdfs:subClassof z	x rdfs:subClassOf z
12	p rdf:type rdfs:ContainerMembershipProperty	p rdfs:subPropertyOf rdfs:member
13	o rdf:type rdfs:Datatype	o rdfs:subClassOf rdfs:Literal

the semantic web. Common queries, such as "what are the (direct and indirect) sub-classes and instances of class" and "which instances have a certain relationship with a given instance", may all involve inference. The fact generation are based on some set of rules. RDFS provides rules for both Tbox (schema) and Abox (instance) level rules. The set of RDFS 14 rules are described in following table:

2.5 Graph Databases

Graph is a collection of nodes(vertices) and edges. Graph represent entities as nodes and the way these nodes connect with each other as relationships. Data is increasing exponentially and more connected year after year making it difficult for relational databases to handle as it requires a lot of

joins. The best way to handle big and connected data is by using **Graphs**. A graph database stores data in form of nodes and edges. Processing big graphs is also a big challenge. The property graph model is a kind of graph model and has following characteristics:

- Contains nodes and relationships
- Nodes have properties (key-value pair)
- Relationships are directed and contains properties.

2.6 Neo4j

Neo4j is the leading graph database and massively scalable that can store up to millions of nodes and relationships. Neo4j is the property graph, nodes and edges can have properties [2]. **Labels** in neo4j assign roles to the nodes. A single node can have multiple labels on it. For example, Ali can be have a label Person as well as Student.
Relationships in neo4j are always directed and must have a start and end node.

Chapter 3

Related Work

Many RDF management systems are developed using NOSQL databases [6]. Each NOSQL databases has its own performance. The performance of these NOSQL databases (HBase, Couchbase, and Cassandra) is compared with native triple store (4store). RDF triples are first stored on these systems and then queried using SPARQL. The evaluation determine the performance profiles of these system on data size, cluster size and query characteristics. The evaluation was done on Berlin and DBpedia benchmark. On the basic of the evaluation following conclusion was drawn:

- Simple SPARQL queries having lookups execute more efficiently on NOSQL databases than native triple store.
- Complex SPARQL queries having complex filters and a lot of data execute more efficiently on native triple stores than NOSQL databases.

- RDF loading time depends on the indexing approached used.
- When triples are stored in large graph structured, finding data is inefficient because of the expensive join operations.

For property graph model, tinkerpop blueprint provides implementation, interfaces and ouplementations [7]. Blueprint is the JDBC for graph databases. Graph databases backend can easily be plug and play using the interfaces provided by blueprint. It is implemented using open RDF (sesame), an open source framework for querying and analyzing RDF [8]. This RDF can be viewed as graph with classes as nodes and properties as edges of the graph. It is possible to model RDF quads (subject, predicate, object and named graph) as a property graph. The subjects in triples are URIs or blank nodes, predicates are URIs and objects are URIs, blank nodes or literals. To map RDF to property graph, all nodes in property graph have a kind property (key-value pair) which can be a uri, literal or a blank node. The nodes which have kind literal, can further have three properties (key value pair). These properties are language, datatype and value of literal [9].

Chapter 4

Mapping of RDF constructs with Neo4j

4.1 Mapping of RDF to Neo4j

Resource Description Framework (RDF) is the W3C standard for semantic data modeling. RDF statement consists of subject-predicate-object triple. Subject must be a URI or a blank node, predicate must be a URI and object must be a URI, blank node or a literal. URI is the unique identifier of resource. Blank nodes don't have a unique identifier and can't be referred externally. Literals are atomic values and consist of three parts; the actual value, data type and language of value. On the other hand neo4j represents data as nodes, relationships and properties (key -value pairs) of nodes and edges. RDF URIs and blank nodes are mapped as neo4j nodes. This neo4j

node has a property which stores the URI of resource. In case of blank node the URI is the hash value of node. Literal values can be stored as subject node properties. For example an RDF triple: ex:Peter foaf:nick "Peter". The neo4j node Peter has property (key: foaf:nick, value: Peter). But the problem occurs when a person has two nick names. Another problem in that mapping is it could not store literal data type and language. When mapping RDF to neo4j:

- RDF resources (URI) are mapped as neo4j nodes with label 'uri' and a property
  - key: uri ; value: the URI of resource.
- RDF predicate are mapped as neo4j relationships.
- RDF literals are mapped as neo4j nodes with label 'literal' and three properties
  - Key: value ; value: the actual value of literal.
  - Key: datatype ; value: datatype of value.
  - Key: language ; value: language of value
- Blank nodes are mapped as neo4j nodes with label 'bnode' and a property
  - Key: uri ; value: hash of blanknode

Example 1: RDF Statement

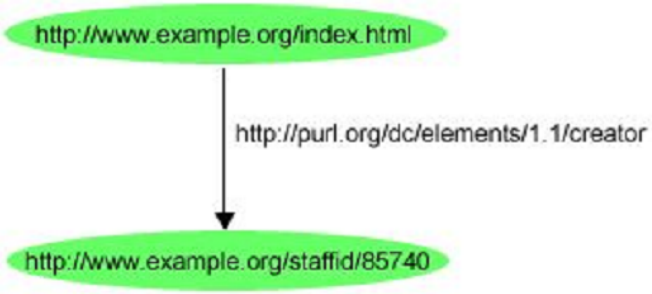

Figure 4.1: RDF Statement

This RDF statement has:

- a subject http://www.example.org/index.html
- a predicate http://purl.org/dc/elements/1.1/creator
- and an object http://www.example.org/staffid/85740

The subject and object are resources so while mapping them neo4j node has label uri. The following figure shows how this statement is represented in Neo4j. (In mapping prefixes are used)

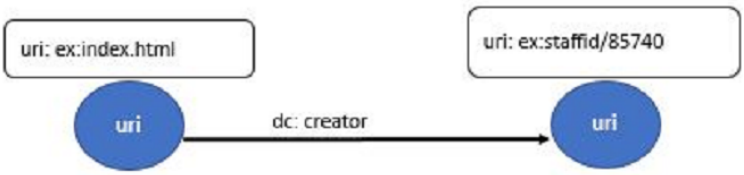

Figure 4.2: RDF Statement

Example 2: RDF Statement with Literal

This RDF statement has:

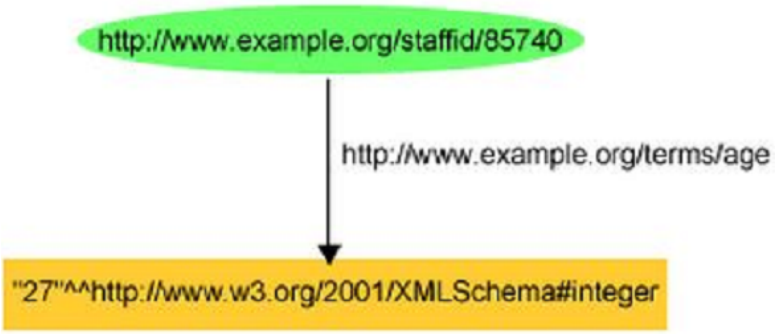

Figure 4.3: RDF Statement with Literal

- a subject http://www.example.org/staffid/85740
- a predicate http://example.org/terms/age
- and an object "27" which is a literal value.

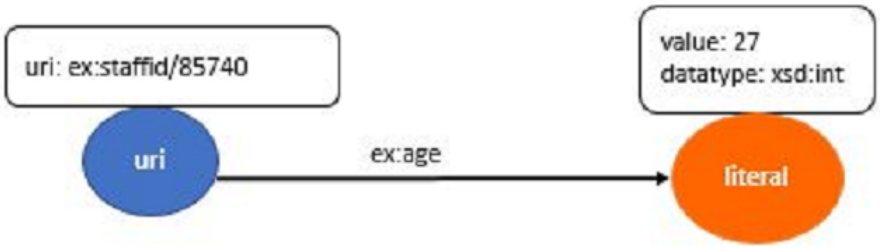

Figure 4.4: RDF Statement with Literal

The subject is a resource so while mapping them neo4j node has label uri.Literal values can have their datatype and language as well.

Example 3: Multiple Statements including Literal values

Subjects and objects when mapped in neo4j node have label uri while literal values have label literal. The properties of uri and literals are different.

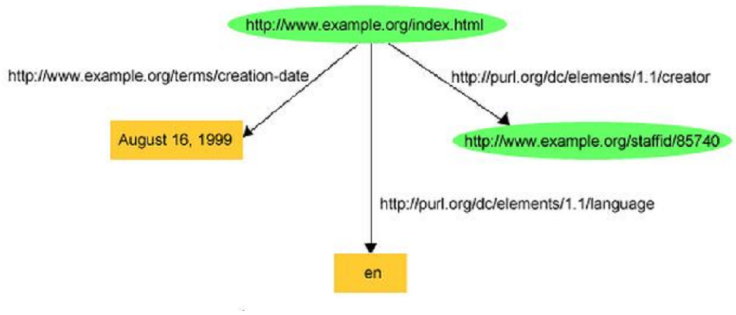

Figure 4.5: Multiple Statements including Literal values

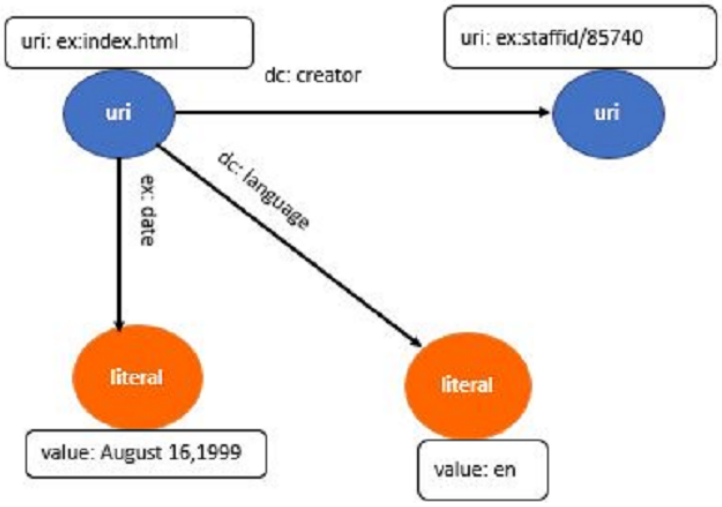

Figure 4.6: Multiple Statements including Literal values

Example 4:Statement with Blank nodes

While mapping blank node the neo4j node has label bnode.

4.2 Mapping SPARQL to Cypher

SPARQL is the RDF query language and cypher is the neo4j query language. To retrieve RDF from neo4j, SPARQL query is converted to cypher

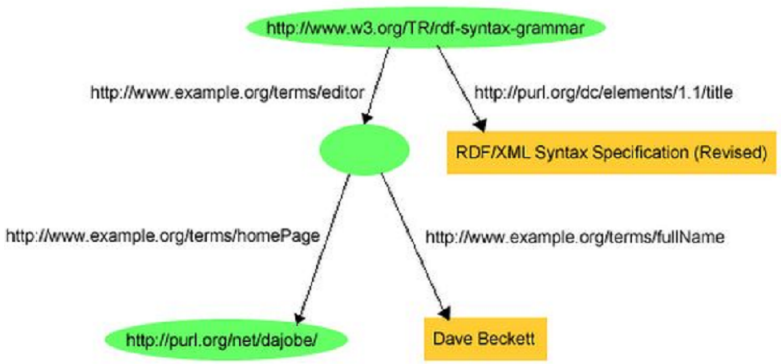

Figure 4.7: RDF Statement

query and executes in neo4j.

Table 4.1: SPARQL to Cypher Mapping

SPARQL	CYPHER
Select	Return
Where	Match
Describe	Match
Insert	Create
Update	Delete,Create
Delete	Delete
Limit	Limit

4.3 Inference

Reasoning on knowledge to discover implicit information is an important process on the semantic web. Common queries, such as "what are the (direct and indirect) sub-classes and instances of class" and "which instances have a certain relationship with a given instance", may all involve inference. Neo4j doesn't support inference. RDF(S) rules are implemented to

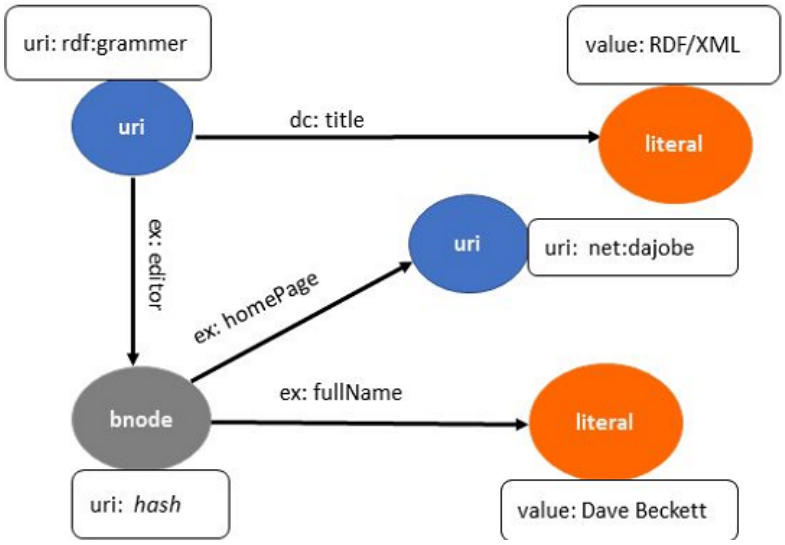

Figure 4.8: Statement with Blank nodes

support inference in neo4j. Terminological box (Tbox) describes controlled vocabularies in term of concepts (classes and properties) and concept hierarchies. Assertion box (ABox) describes ground sentences stating where in the hierarchy the individual belongs. Both TBox and ABox are converted to cypher query and then executes in neo4j. RDFS reasoner consists of set of 14 rules. There are certain rules that produce trivial output that cannot be used for further derivation. These rules are rules number 1, 4a, 4b, 6, 8, 10. We haven't implemented these rules in neo4j.

TBox Rules

Rule 5:

$p\ \ rdfs:subPropertyOf\ \ q\ \ \&\ \ q\ \ rdfs:subPropertyOf\ \ r$

$\Longrightarrow\ p\ \ rdfs:subPropertyOf\ \ r$

The rule states that if p is subproperty of q and q is subproperty of

r, a new triple will be inferred which implies that p is also the subproperty of r.

Cypher conversion:

MATCH (p)-[:'rdfs:subPropertyOf']->(q)

MATCH (q)-[:'rdfs:subPropertyOf']->(r)

MERGE (p)-[:'rdfs:subPropertyOf']->(r)

Match construct finds the triples. Merge is the combination of match and create, it creates a new triple only if it doesnt exist otherwise it finds the triple.

Rule 11:

$x \ \ rdfs:subClassOf \ \ y \ \ \& \ \ y \ \ rdfs:subClassOf \ \ z$

$\implies x \ \ rdfs:subClassOf \ \ z$

The rule states that if x is subclass of y and y is subclass of z, a new triple will be inferred which implies that x is also the subclass of z

Cypher conversion:

MATCH (x)-[:'rdfs:subClassOf']->(y)

MATCH (y)-[:'rdfs:subClassOf']->(z)

MERGE (x)-[:'rdfs:subClassOf']->(z)

ABox Rules

Rule 7:

$s \ \ p \ \ o \ \ \& \ \ p \ \ rdfs:subPropertyOf \ \ q \implies s \ \ q \ \ o$

The rule states that if there exist a triple whose property (p) is subproperty of another property (q), a new triple will be inferred with the property q.

Cypher conversion:

MATCH (p)-[:'rdfs:subPropertyOf']->(q)

MATCH (s)-[:p]->(o)

MERGE (s)-[:q]->(o)

Rule 2:

$p\ \ rdfs:domain\ \ x\ \ \&\ \ s\ \ p\ \ o\ \implies\ s\ \ rdf:type\ \ x$

The rule states that if there exist a triple with property p and p has domain x that implies the subject of triple is of type x.

Cypher conversion:

MATCH (p)-[:'rdfs:domain']->(x)

MATCH (s)-[:p]->(o)

MERGE (s)-[:'rdf:type']->(x)

Rule 3:

$p\ \ rdfs:range\ \ x\ \ \&\ \ s\ \ p\ \ o\ \implies\ o\ \ rdf:type\ \ x$

The rule states that if there exist a triple with property p and p has range x that implies the object of triple is of type x.

Cypher conversion:

MATCH (p)-[:'rdfs:range']->(x)

MATCH (s)-[:p]->(o)

MERGE (o)-[:'rdf:type']->(x)

Rule 9:

$s\ \ rdf:type\ \ x\ \ \&\ \ x\ \ rdfs:subClassOf\ \ y \implies s\ \ rdf:type\ \ y$

The rule states that if the type of subject of triple is x and x is subclass of y that implies subject of triple is also of type y.

Cypher conversion:

MATCH (s)-[:'rdf:type']->(x)

MATCH (x)-[:'rdfs:subClassOf']->(y)

MERGE (s)-[:'rdf:type']->(y)

Rule execution order:

Rule execution order is important as one rule output can be another rule input and so on. The Tbox rules (5, 11) are recursive i-e they can repeatedly call themseleves. Rule 2, 3 and 9 outputs data triple that defines type of some particular resource. The output of rule 7 can be of any nature. Tbox rules are executed first than abox rules are executed. In abox rules, rule 7 is executed first as its output is of any nature. After that rule 2 and 3 are executed. And lastly rule 9 is executed.

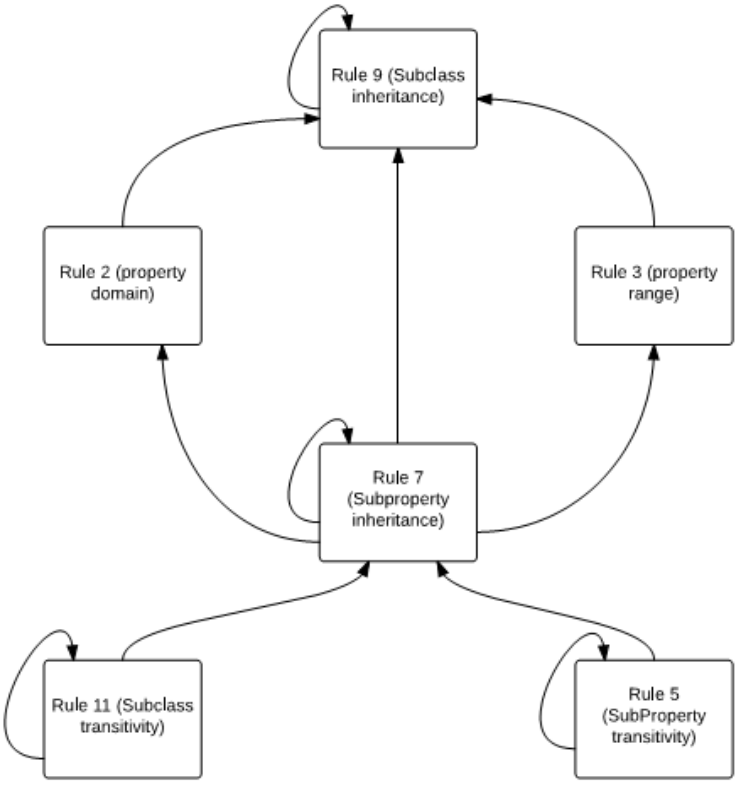

Figure 4.9: RDFS Rule execution order

Chapter 5

System Architecture and Implementation

This thesis proposes a framework neo4jena. This framework maps RDF data with property graph. Two implementations are provided to load RDF in neo4j

- Triple by triple storing in neo4j
- Bulk load

First RDF triples are converted to neo4j using RDF-Neo mapper. After this they can be stored in neo4j either triple by triple or by bulk load. RDF is queried using SPARQL query language. SPARQL query language is converted to cypher query language using SPARQL- cypher mapper. Cypher

query is then executed in neo4j and return results in form of neo4j nodes and relationships. These neo4j nodes and relationships are then converted to triples. Inference rules are then executed in neo4j.

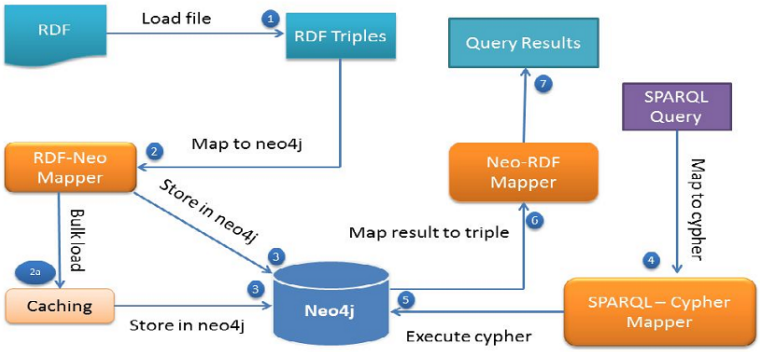

Figure 5.1: Neo4jena architecture

Chapter 6

Evaluation and Results

6.1 Benchmarks

6.1.1 Lehigh University Benchmark (LUBM)

LUBM is used to evaluate Semantic Web repositories in a standard and systematic way. The benchmark contains university data which can have different departments, professors, students, courses and the relationship between them.Five datasets were generated for this benchmark:

- 15,144 triples
- 95,280 triples
- 230,062 triples

- 477,785 triples
- 1,001,419 triples

6.1.2 Computational Environment

All evaluation was done on CentOS with 4 GB of RAM and 115 GB of storage.

6.2 Results

6.2.1 Loading

The loading performance of RDF in neo4j is compared with TDB and tinkerpop. The following graph shows time to load single triple in different datasets.

Neo4jena loads 6-30 triples in a millisecond while TDB loads 5-11 triples per millisecond and tinkerpop loads 2-4 triples in a millisecond.

6.2.2 Inference

Inference is performed on wordnet dataset. The following graph shows the time taken to infer the triples.

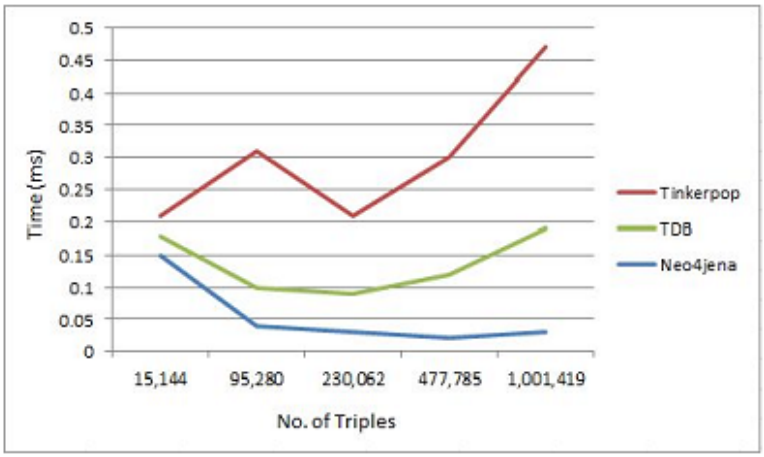

Figure 6.1: Load time for lehigh benchmark

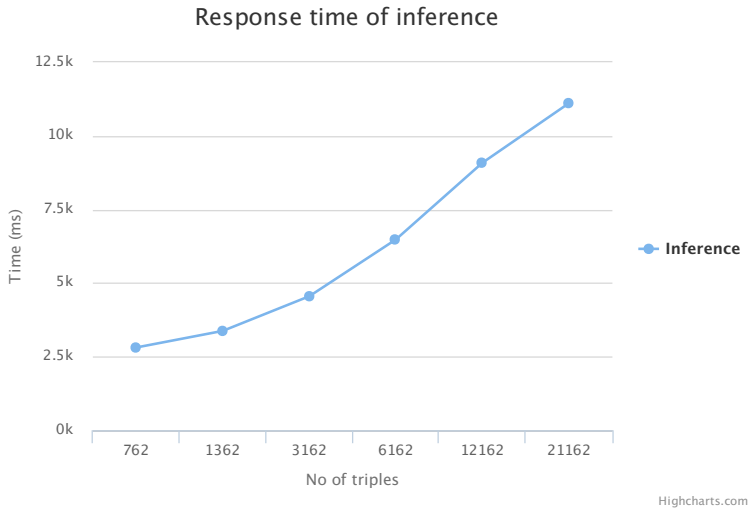

Figure 6.2: Response time for inference

Inference rules are also applied on different schemas to test Tbox rules. Following graph shows the time taken for Tbox rules (subclass and subpropertyof) to infer triples.

Rule 5:

$p\ rdfs:subPropertyOf\ q\ \&\ q\ rdfs:subPropertyOf\ r \implies p\ rdfs:$

$subPropertyOf \;\; r$

Rule 11:

$x \;\; rdfs:subClassOf \;\; y \;\; \& \;\; y \;\; rdfs:subClassOf \;\; z \implies x \;\; rdfs:subClassOf \;\; z$

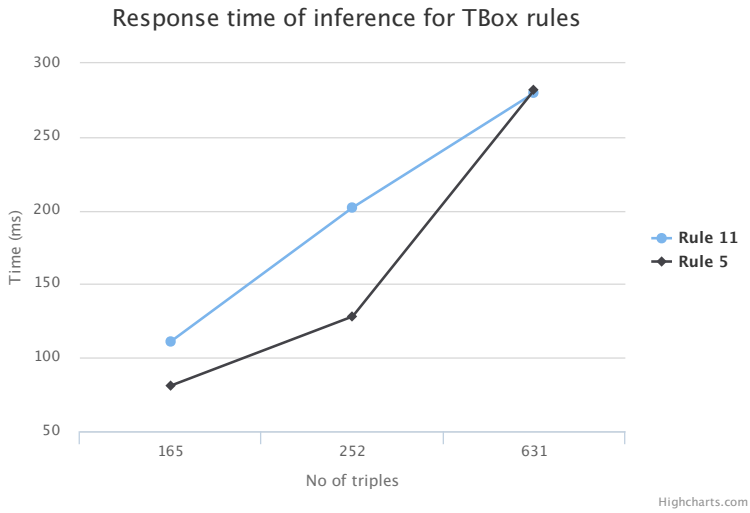

Figure 6.3: Response time of inference for Tbox rules

Positive and negative entailment test are taken to test the implementation.

Positive Entailment:

Positive entailment consist of two ontologies in which one is semantically entail by the other.

Negative entailment:

Negative entailment consist of two ontologies in which one is explicitly not to be entail by the other.

Table 6.1: Positive Entailment

Input	Output
ex: c1 rdfs:subClassOf ex:c2 ex:c2 rdfs:subClassOf ex:c3	ex : c1 rdfs:subClassOf ex:c3
ex: p1 rdfs:subPropertyOf ex:p2 ex:p2 rdfs:subPropertyOf ex:p3	ex : p1 rdfs:subPropertyOf ex:p3
ex: p rdfs:domain ex:c ex:u ex:p ex:v	ex : u rdf:type ex:c

Table 6.2: Negative entailment

Rule name	Premises	Conclusion
subClass	ex:foo rdf:type,rdf:Property ex:bar rdfs:subClassOf ex:foo.	FALSE
subProperty	ex:foo rdf:type rdfs:Class ex:bar rdfs:subPropertyOf ex:foo.	FALSE
domain	ex:p rdfs:domain ex:c ex:u ex:q ex:v .	FALSE
range	ex:p rdfs:range ex:c ex:u ex:q ex:v .	FALSE

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis proposes a mapping of RDF construct with neo4j.By using the framework (neo4jena) defined in this thesis, RDF data can be easily stored in a property graph. SPARQL is the W3C recommendation for querying RDF data. To query RDF data from neo4j, SPARQL query language constructs are mapped with cypher query language constructs. Inference is also applied on neo4j data to infer new facts from the existing ones. The load time of neo4jena is compared with TDB and tinkerpop on LUBM dataset. The performance of neo4jena is better than TDB and tinkerpop. On conclusion, this thesis resolve three problems to achieve a scalable RDF storage system. Firstly RDF data is successfully mapped with neo4j. Secondly for retrieval of RDF from neo4j SPARQL query constructs are suc-

cessfully mapped with cypher query language. Lastly, an RDFS reasoner is implemented in neo4j.

7.2 Future Work

Querying time of neo4jena can be optimized to achieve high scalability. Moreover other reasoner e.g OWL-DL can also be implemented in neo4j to perform advance inference operations.

References

[1] Michael Ovelgnne, Noseong Park, V.S. Subrahmanian, ElizabethK. Bowman, and KirkA. Ogaard. Personalized best answer computation in graph databases. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, JosianeXavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web ISWC 2013*, volume 8218 of *Lecture Notes in Computer Science*, pages 478–493. Springer Berlin Heidelberg, 2013.

[2] Inc. Neo Technology. Neo4j, the world leading graph database, 2015. http://neo4j.com/.

[3] RDF Working Group. Rdf, 2014. http://www.w3.org/RDF/.

[4] Mohsen Taheriyan, CraigA. Knoblock, Pedro Szekely, and JosLuis Ambite. A graph-based approach to learn semantic descriptions of data sources. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, JosianeXavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web ISWC 2013*, vol-

ume 8218 of *Lecture Notes in Computer Science*, pages 607–623. Springer Berlin Heidelberg, 2013.

[5] Apache Jena. Apache jena, 2011-2015. https://jena.apache.org/.

[6] Philippe Cudr-Mauroux, Iliya Enchev, Sever Fundatureanu, Paul Groth, Albert Haque, Andreas Harth, FelixLeif Keppmann, Daniel Miranker, JuanF. Sequeda, and Marcin Wylot. Nosql databases for rdf: An empirical evaluation. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, JosianeXavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web ISWC 2013*, volume 8219 of *Lecture Notes in Computer Science*, pages 310–325. Springer Berlin Heidelberg, 2013.

[7] Apache. Apache tinkerpop. http://tinkerpop.incubator.apache.org/.

[8] Apache. Blueprints, 2014. https://github.com/tinkerpop/blueprints/wiki.

[9] Apache. Sail implementation, 2012. https://github.com/tinkerpop/blueprints/wiki/Sail-Implementation.