Information Extraction from Ontology using Distributed Environment

Submitted By

Nikhil S Musale

15MCEN13

DEPARTMENT OF INFORMATION TECHNOLOGY

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2017

Information Extraction from Ontology using Distributed Environment

Major Project

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering

(Networking Technologies)

Submitted By

Nikhil S Musale

(15MCEN13)

Guided By

Prof. Sapan H Mankad

DEPARTMENT OF INFORMATION TECHNOLOGY

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2017

Certificate

This is to certify that the major project entitled **"Information Extraction from Ontology using Distributed Environment"** submitted by **Nikhil S Musale (Roll No: 15MCEN13)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering (Networking Technologies) of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this Major project part-II, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. Sapan H Mankad
Guide & Assistant Professor,
IT Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Gaurang Raval
Associate Professor,
Coordinator M.Tech (CSE-NT)
Institute of Technology,
Nirma University, Ahmedabad

Dr. Madhuri Bhavsar
Professor and Head,
IT Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr Alka Mahajan
Director,
Institute of Technology,
Nirma University, Ahmedabad

Statement of Originality

I, **Nikhil S Musale**, Roll. No. **15MCEN13**, give undertaking that the Major Project entitled **“Information Extraction from Ontology using Distributed Environment"** submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science and Engineering (Networking Technologies)** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made.It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Signature of Student

Date:

Place:

Endorsed by

Prof. Sapan H Mankad

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Prof. Sapan H Mankad**, Assistant Professor, Information Technology Department, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout this work. The appreciation and continual support he has imparted has been a great motivation to me in reaching a higher goal. His guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank **Dr. Madhuri Bhavsar**, Hon'ble Head of Information Technology Department, Institute of Technology, Nirma University, Ahmedabad for her kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr Alka Mahajan**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation she has extended throughout course of this work.

I would also thank the Institution, all faculty members of Computer Engineering and Information Technology Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

- Nikhil S Musale
15MCEN13

Abstract

Today we live in era of information. Every kind of information is easily available. It is because of Internet. Digitization of data help us to store very large amount of information. To store these large amount of data, we use system like database and server. From last two decades the data growth increases rapidly. Today every day and every moment, data is increasing on Internet. Conventional database system like SQL is quit inefficient for this type of data growth. To overcome this problem, we need other type of data storage system. Semantic Web comes into picture here. Semantic web uses an ontology to store data. Ontology is type of graph database. Ontology uses RDF, which shows the relationship between web resources. Web resources are presented in the form of URI. SPARQL is query language to extract the information from Ontology. Semantic Web gives facility of single store database like Apache Jena but it has limit of storage. Query processing is also challenge. Due to these reasons distributed environment comes into picture. RDF data can be converted to N-Triple format. This N-Triple data-set can be loaded to Apache Hive. Hive runs on top of the Hadoop. After loading this N-Triple format, Hive query is executed. Hive internally convert query in to map-reduce and gives the results. We have executed different queries on Apache Hive and Apache Jena. We found that Apache Hive performs better than Apache Jena.

Abbreviations

SW	Semantic Web
RDF	Resource Description Framework
OWL	Ontology Web Language
GOT	Graph of Things
OAEI	Ontology Alignment Evaluation Initiative
QpS	Query per Second
SQL	standardized query language
SPARQL	Simple Protocol and RDF Query Language
HQL	Hive Query Language
CQELS	Continuous Query Evaluation over Linked Streams
NLP	Natural Language Processing

–

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 What is Semantic Web?

In very famous cartoon series Arabian nights, one character is very famous call Jinn. He completes all wishes of his possessor. Likewise if we have such 'Jin' then. . .

We can fulfill our all wishes!!!

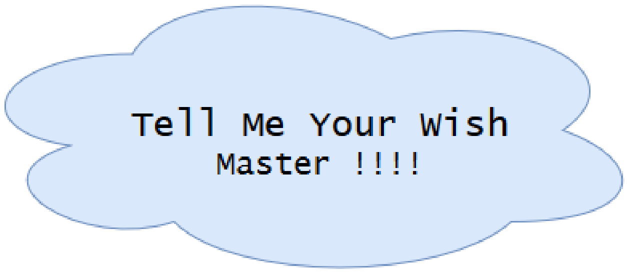


Figure 1.1: Tagline of Jinn

We saw many scientific movies like Star-wars, Iron Man in which one super computer plays exact role like Jinn. This computer has every information on earth, you have to just ask a question and it will give you an answer.

How this is possible? But now a day this is happening. Which was imagination in past, today it is reality. Let take one example suppose you shifted to new city and you want to join sports club nearby your house within a five kilometer radius. You give this finding task to your agent and put some constrain like distance from his house, yearly fee should be less than 15000, minimum four star ratting etc... and your agent search for this task and give you a result of matching sports club which fulfill your requirement.

This technology is possible due to Semantic web and other similar technologies. Semantic web is built on existing structure of world wide web [7]. The conventional search engines are work on concept of inverted index [8] which presents to users the web pages which contains most repeated word which mention in query.

Semantic web uses an other concept. It works with URI(Universal Resource Identifier). Semantic web searches for this URI with the help of SPARQL query language. It gives us a very accurate result.

1.2 What is Ontology?

Ontology is unanimous part of semantic web. Ontology word originate from ancient Greek. The meaning of ontology word is study of relation between the objects which are exiting in the world. You can study the objects by observing it and find relation between them. Ontology covers the vast area of studies like philosophy, science and medical. The concept of Ontology is existed in ancient India, known as 'Tatva Mimhasa'[1].

Current Computer science utilize the word ontology and use it for it's own. In computer science ontology is study of relation between the objects which are reside on the world wide web or internet. These all objects has unique address across the world known as URI(Universal Resource Identifier). There is a difference between the URI and URL. URL is a sub class of URI. URI identifies the physical resource and URL shows the physical address of it[2].

In Ontology we use entities(class of objects) and show relationship between them. This relationship may describe by property, hierarchy relationship, value restriction, disjoint etc...

Here this entities are describe as triples where subject and object may be classes and predicate shows the relationship between them. In above figure The Monalisa painting is created by Lionardo da Vinci. Here we can clearly see that, the both objects are described by URI, www.painting.com/mon.htm and www.pariceart.com/lio.htm and pan:created by is a prefix of some URI, here it used to describe Predicate.

Here one common question arises, why we use prefix? Because when user work with

[1]https://en.wikipedia.org/wiki/Ontology
[2]https://danielmiessler.com/study/url-uri/

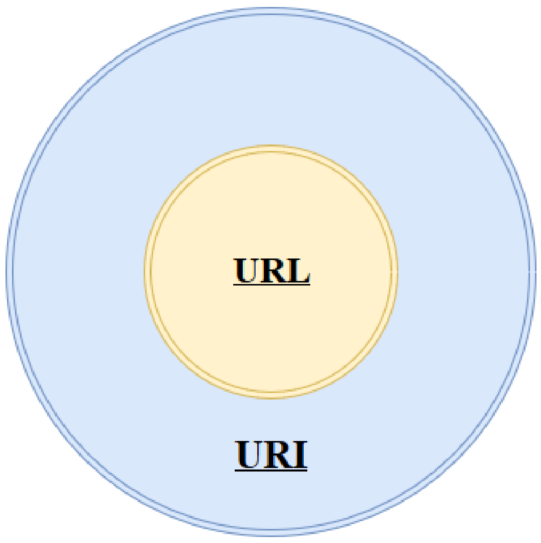


Figure 1.2: URI and URL

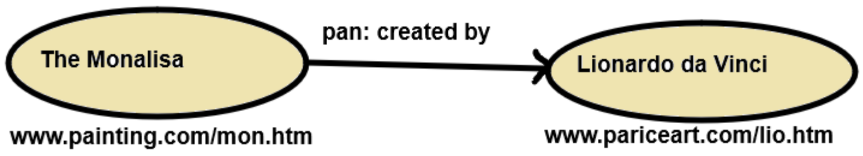


Figure 1.3: RDF Triple

ontology, it is very insufficient that every time user mention entire URI. Prefix is a short form of URI. Concept of URI utilize in XML schema, RDF schema and SPARQL.

Ontology improves the accuracy of web search because it always provides a class or object which is prior to user. It helps to quickly navigate around web sites.

1.3 What is RDF?

RDF(Resource Description Framework) is a method of showing an ontology structure. RDF is built on top of the XML structure. We can built an ontology with the help of RDF. RDF is always deals with URI. RDF provides a facility to create class hierarchy, call RDFS(RDF schema). Here user can describe the sub class of other class, means describe class hierarchy similar to Object oriented concept. Let us take an example of

RDFS.

RDF is understand by both human and computers. Here almost all entities are web resource and web recourse are describe with URI.

Let us take a small example of RDF

```
 <?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xml:base="http://www.iamcomputer.com/computing/Computer.rdfs">
            <rdfs:Class rdf:ID="Computer">
           </rdfs:Class>
           <rdfs:Class rdf:ID="Byers">
           </rdfs:Class>
           <rdfs:Class rdf:ID="Screen">
                   <rdfs:subClassOf rdf:resource="#Computer"/>
           </rdfs:Class>
           <rdfs:Class rdf:ID="RAM">
                   <rdfs:subClassOf rdf:resource="#Computer"/>
           </rdfs:Class>
           <rdfs:Class rdf:ID="Debit␣card">
                   <rdfs:subClassOf rdf:resource="#Byersl"/>
           </rdfs:Class>
           </rdfs:Class>
 </rdf:RDF>
```

Above example is shows the RDF listing for computers. Example shows two RDF class Computer and Byers. Here screen and RAM is describe as subclass of Computer and Debit card is subclass of Byers. User can also describe an RDF graphically.

1.4 What is OWL?

Owl(Ontology Web Language) is more richer language then RDF and it is built on top of RDF. OWL provides more features [9].

- Local scope of property: it defines restriction property for some of class.
- Disjointedness of classes: two classes totally different from them. In rdf we can only highlight class hierarchy.
- Cardinally restriction: we want to restrict the value that object property have. In rdf it is impossible. For example, a person have exactly two parent.
- Special character property: here it means transitive("grater then"), unique("mother of"), inverse of("not sleep"). These properties is impossible in rdf.

1.5 SPARQL

SPARQL stands for Simple Protocol and RDF Query Language. SPARQL is a query language for RDF graph database to retrieve the information. RDF basically structure of URI and SPARQL helps to retrieve this URI information. Let us take an example.

Insert Query

```
PREFIX cr : <http://cricket/>
insert data{
    cr :tilak cr :age cr :23.
    cr :meet cr :age cr :22.
    cr :gaurang cr :playsin cr :nirmacup.
    cr :nirmacup cr :organizeby cr :ITNU.
    cr :binaka cr :learn cr :judo.
        }
```

Above SPARQL query inserts the information in to graph. Here PREFIX cr is very important. User need to put cr front of every object. Here cr is short form of http://cricket/ .

Simple Select

```
PREFIX : <http://cricket/>
select * where{
        cr:gaurang cr:playsin ?a.
        }
```

Above query explains simple select from data we have inserted. Here query is "In which tournament gaurang plays?". The answer will be in this form http://cricket/nirmacup.

1.6 Layered Architecture

As per shown in figure. At bottom layer XML and XML Schema is presented. Above it RDF and RDF schema is there which is use find relation between web resources. On top of that Ontology vocabulary reside, which puts some more constrain on data. On top of it logic will help for finding the relevant information with the help of SPARQL. After this result we get our result as proof and trust.

Simply below figure show that RDF and OWL script can be embedded with XML.

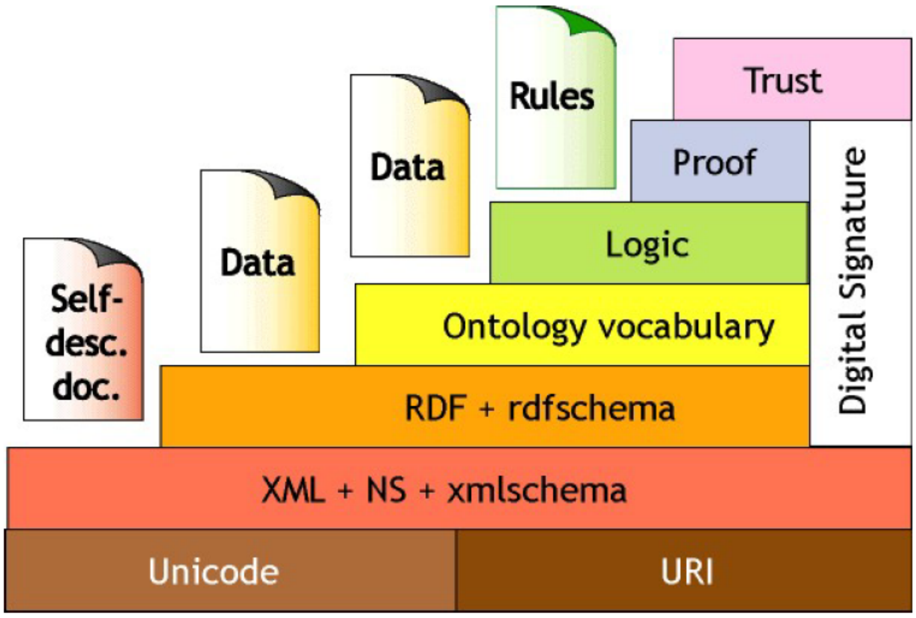


Figure 1.4: Layered Architecture

Chapter 2

Literature Survey

2.1 Ontology Mapping

We discuss till now what is semantic web and component of it. Semantic web covers a concept of ontology and ontology means studding and finding the relationship between various entities on web.

How we can find relationship between ontology? Various ways available to find relationship between ontology, one of them is finding hierarchy of classes which is on the web. Finding hierarchy in terms of generalization, relationship between classes [10,11]. Let us take very simple example, we all know objected oriented programming concept. Let say one class is 'Vehicle' and one class is 'bike'. Which class is more general? Obviously 'Vehicle' class is more general. Through this concept we find the relationship 'r' between two ontology.

Now question arises is which type of relationship we achieve here. Three types of relationship we get 1. More general (as discussed above) 2. Equivalent 3. Disjoint.

Equivalent relationship shows exact similarity between two classes and disjoint means no similarity between classes [10]. Here we can talk about one more thing call similarity index 's'. From similarity index we find statically that how much both class is similar to each other [2,10].

In figure two ontology are given. One is product ontology O1 and other is Monograph

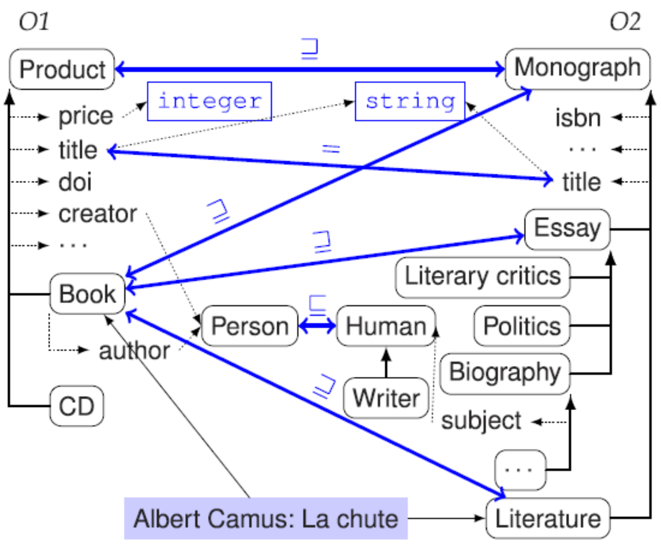


Figure 2.1: Ontology Alignment [2]

ontology O2. Product class's properties are given price, title, creator etc... Every class has its own property. From ontology we can observe that 'product' class is more general than 'monograph'. Book class is also more general than Monograph, Essay, Literature, while comparing classes we actually comparing their property. In figure title property is same in both classes 'Product' and 'Monograph'. From property of classes we can find index's'. Whose value can be for example 0.6587. We obtain this value by comparing property. In figure class 'book' is compare with 'Monograph', 'Essay', 'Literature'. Here concept of ontology alignment used.

Ontology alignment can be of four types, which are...

- 1:1(one to one)
- 1:m(one to many)
- m:1(many to one)
- m:m(many to many)

Book class is example of one to many.

Ontology mapping is complex task itself. For ontology mapping neural network is used [2]. Here ontology is given as input to neural network and matching parameter is defined by user or may not. Here parameter means which property of class is going to compare and which is not.

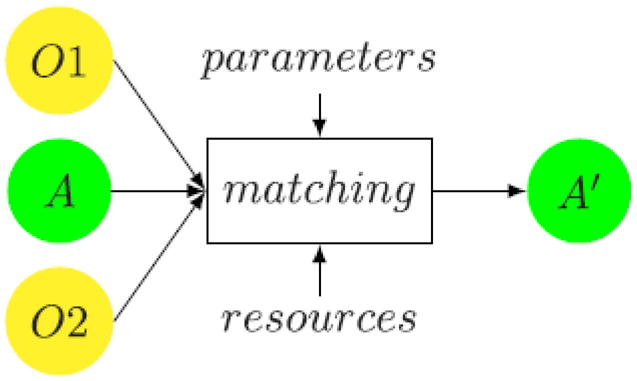


Figure 2.2: Working of Neural Network [3]

In figure two ontology are compare in neural network and similarity index A' is achieved as output.

This ontology mapping also helps us in question answering system, where user always asks a question in their abstract term [12–14]. These questions can generate for ask something or extract information from picture.

Here we process the query using NLP(natural language processing). After that parsed query as input enters in to a system and through keyword it will find an answer of your question. Here keywords match with classes.

If you are not getting an answer from your query, you can expand the query. Query expansion can be done from what feedback is given from your system. If your query is very near or resemble of getting an answer. In this case system will give feedback and user can expand the query [3].

2.2 Visual Query Answering with the help of Background Knowledge

Getting an answer from picture is call visual query answering. Visual query answering with the help of semantic web is very efficient technique to find answers which are not present in a picture.

Consider the figure. The picture is of beach, we do image processing. Here we decide attributes of picture. In figure attributes are umbrella, mountain, human, ocean, house

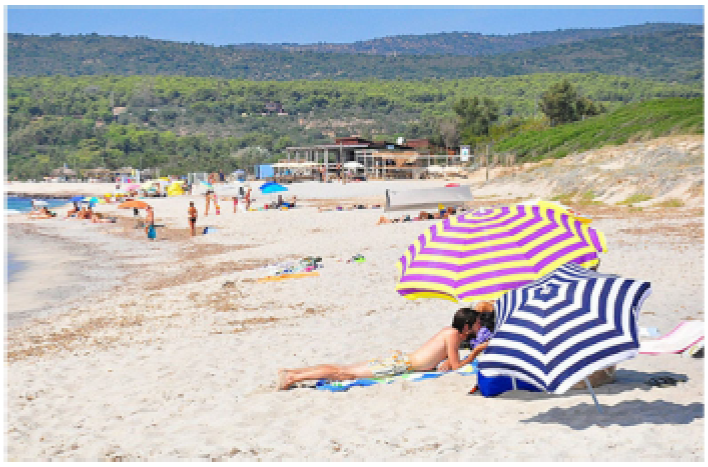


Figure 2.3: Example Image for Visual Question Answering [4]

etc. . . Now normally when user asks a question, it provides an answer which is present in a picture.

Semantic web breaks this constrain. For example user may ask, "How many types of umbrella available in the world?" Certainly the answer is not present in figure but attribute umbrella is present. Here answer come from knowledge base like DBpedia [4, 13]. Attribute umbrella searches in to DBpedia. DBpedia is structured information of Wikipedia.

Here system searches an attribute in to DBpedia find it and extract information require in question [2]. Answer is provided to user.

DBpedia is knowledge base, means DBpedia is structured information extracted from Wekipedia. Structured information means information about every verb or attribute in Wikipedia, provided by DBpedia.

2.3 Information extraction from IOT and Social Media with SPA-RQL

Question answering is an important side of semantic web. When some question arises by user, System redirected to the DBpedia and find an answer. This concept call getting answer from background knowledge. Most important question is how user can query to the ontology system or DBpedia? An answer is through SPARQL [15,16]. SPARQL is a query language for graph database like RDF graph database. Through SPARQL query user can query to the graph database and get answer.

User can query two types of ontology 1. Domain dependent 2. Domain independent [3]. Domain dependent ontology is restricted to one domain. To execute SPARQL query domain dependent ontology is very sufficient and user can get answer easily.

Ontology can be made with RDF or OWL. We can say RDF and OWL is one type of tool that uses to create ontology. RDF graph is made of subject, predicate and object. For example 'Nikhil' has a friend 'Parth'. Here has friend shows a relationship between two classes, when you want to find the friend of 'Nikhil'. It is so easy with SPARQL [17].

We can use ontology in very effective recommend er system, but how this system works? With the help of IOT(Internet of Things). Today IOT is top technology and it covers most areas of technology. IOT sensors are everywhere in near feature. IOT generates very large amount of data everyday and billions of user using it [5].

Let us take one scenario, a person named 'Bhavik' visits new area and he gets a notification of temperature on his mobile phone. After this he searches for hotel and number of hotel as his preference is displayed. Here one major difference is user is not searching for on Google. User simply asks a question and system answer it how this is happened? Answer is RDF converter, all data generated by IOT. We convert it all in RDF graph with the help of RDF converter [5].

With the help of RDF converter, data converted in to RDF graph. Same this can be applied to social media website. We can extracts knowledge from social media website like twitter. Let us take an example. Suppose one tweet, "I am in Great Britain right now. I am in London and enjoying Queen's special dish at famous Test of England restaurant near Thems River." In this tweet attributes are 'Great Britain', 'London', 'Queen's Special Dish', 'Test of England'. From this attribute we can extract knowledge from DBpedia and YAGO ontology. Architecture is give below.

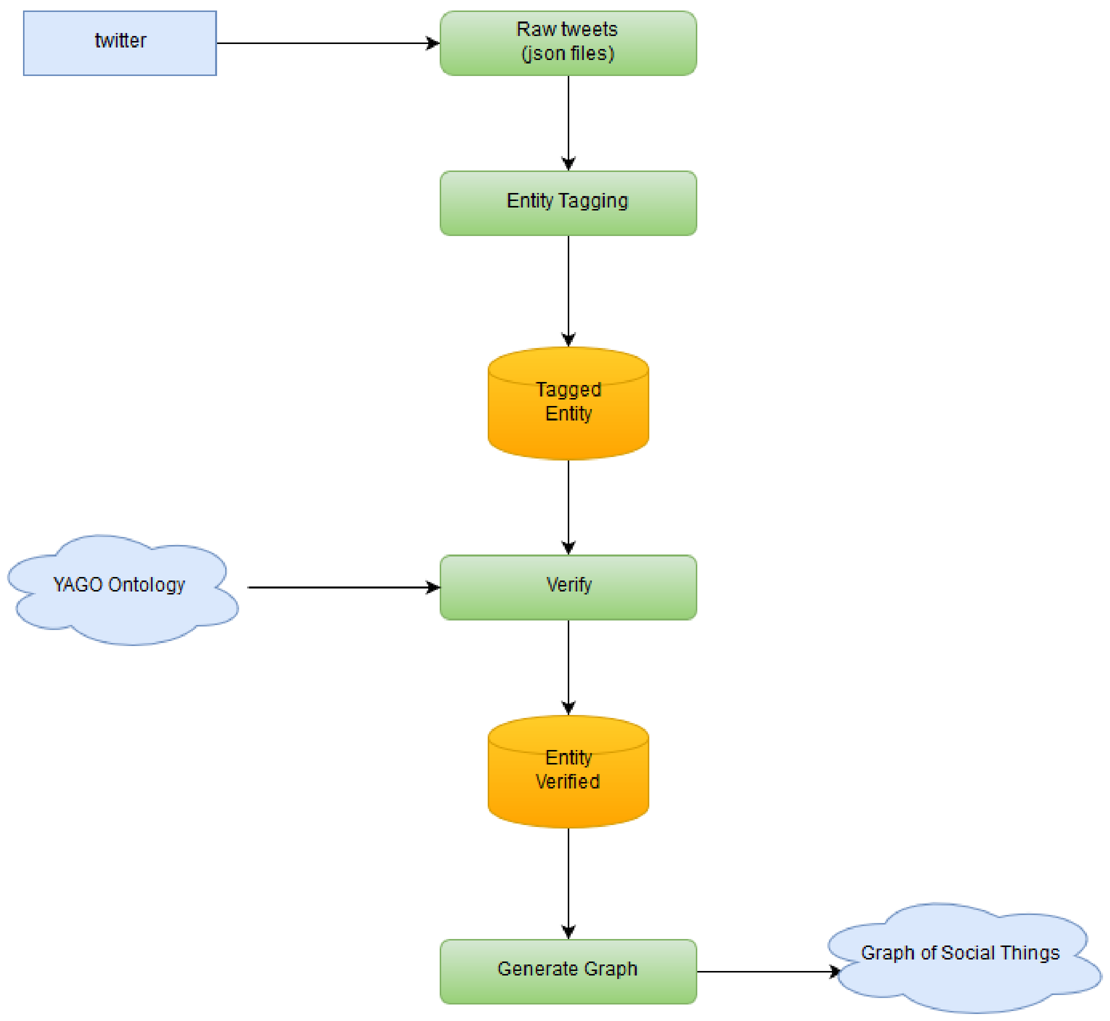


Figure 2.4: Architecture for Data generated from Social Media [5]

Here Twitter data comes in raw format (json files), after tagging it will compare with YAGO ontology [5]. If information related to that entity is available, it will tag it in to existing graph.

IOT device continuously generates the data time to time, example of this device is whether sensor. This data are continuously generated time to time. This generated data connected to RDF graph and this graph is extended. Now user can query this graph via SPARQL engine.

Let us discuss one scenario, a person call Raj wants to catch a flight from surat to delhi but at the airport he discovered that flight is two hour late now what should he do? He thinks to go to the Punjabi restaurant so he ask his phone to search Punjabi restaurant nearby airport. Phone gives all list of Punjabi restaurant and it's ratting, his phone also suggest that his old friend lives in surat so he can meet him, after having dinner with his friend he finds that flight is late for three more hours and subsequently his phone gives him suggestion that there is rock night is organized and lead singer is arjit singh, so Raj visits the rock show and enjoy the night.

So from above scenario we can say that data from iot is always generated time to time and update is received by your device.

This is all possible due to information extracted from IOT sensors. This information comes in form of raw data. Our rdf parser convert this data in to rdf triple format and link it with existing rdf graph, so like this rdf graph can be expanded. Now we are going to see some architecture.

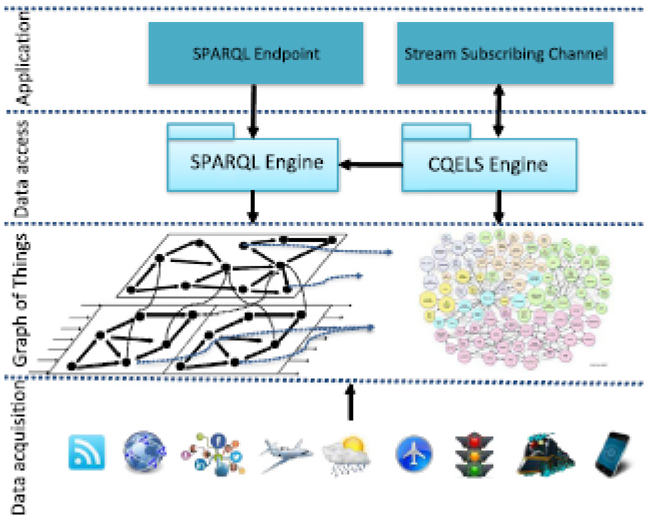


Figure 2.5: Layered Architecture for generated Data [5]

Figure 2.5 is divided in to four layers. The bottom layer is data acquisition layer, from this layer data is generated. The sources of this data are traffic update, flight update, whether update.

The second layer is graph of things layer. At this layer graph is generated. The stream data comes in and connects the rdf graph. Firstly this data is converted in to rdf triple. Likewise a graph is expanded.

This data is in very large amount and it continuously increases so we need very large database for this one can use jena tdb. Here in figure we can see the SPARQL engine this engine converts the user query in to SPARQL format. CQELS engine continuously triggers an update query to SPARQL engine, so user can get automatic update [5]. This query is system generated. For example you are travelling when you rich to the new area your mobile rings and notify you that you are in new area and area's name and information about area. How this is happens? It is due to CQELS engine because it is continuously triggers an query about an area.

IOT device generates two kinds of data, first spatial and second temporal. Spatial data means this data is related to geography. Temporal data means this data is generated at fix time interval.

2.4 SPARQL Benchmark

Benchmark shows some standard points against which we can compare other reference point. In computer science benchmark describes performance of various computer programs relatively [1]. In general benchmark is a word for comparison, for example A student who achieved highest marks, other student's marks calculated with reference to his mark. This is the concept of bench-marking. SPARQL benchmark compares the results of query written in **SPARQL and SQL** on large data-sets. Comparison of these results on the basis of QpS(Query per Second).

QpS means **one query can execute number of result in one second** [1]. QpS may vary from query to query. It is totally depends on complexity of query.

$$QpS = \frac{NumberofTriplesinDatabase}{TotalExecutionTime} \tag{2.1}$$

Let us take an example, suppose we have a database of 10000000 entries and we run some query and it gives the result in 200 seconds. So the QpS of Database is $\frac{10000000}{200}$.

Here we will discuss the result of The Berlin SPARQL Benchmark, which compares the result of SPARQL and SQL in terms of QpS. They have large data-set of products and it's features. Below two SPARQL query give and results of it is given respective table. They run SQL query on same data-set and compare the results of SPARQL and SQL in terms of QpS.

Query 1: **Finds products for a given set of generic features.**

```
SELECT DISTINCT ?product ?label
WHERE {
    ?product rdfs:label ?label .
    ?product rdf:type  : ProductType  .
    ?product bsbm:productFeature  : ProductFeature1 .
    ?product bsbm:productFeature  : ProductFeature2 .
    ?product bsbm:productPropertyNumeric1 ?value1 .
FILTER (?value1 > x)}
ORDER BY ?label
LIMIT 10
```

SPARQL Query1 Finds products for a given set of generic features.

Query 2: **Find products having some specific features and not having one feature**

```
SELECT ?product ?label
WHERE {
     ?product rdfs:label ?label .
     ?product rdf:type : ProductType .
     ?product bsbm:productFeature : ProductFeature1 .
     ?product bsbm:productPropertyNumeric1 ?p1 .
FILTER ( ?p1 > x )
     ?product bsbm:productPropertyNumeric3 ?p3 .
FILTER (?p3 < y )
OPTIONAL {
     ?product bsbm:productFeature : ProductFeature2 .
     ?product rdfs:label ?testVar }
FILTER (!bound(?testVar)) }
ORDER BY ?label
LIMIT 10
```

SPARQL Query2 Finds products having some specific features and not having one

RDF Stores	1M	25M	100M
Sesame	00:02:59	12:17:05	3:06:27:35
Jena TDB	00:00:49	00:16:53	01:34:14
Virtuoso RV	00:00:34	00:17:51	01:03:53
D2R Server	00:00:06	00:02:03	00:11:45
MySQL	00:00:06	00:02:03	00:11:45

Table 2.1: Loading time for different RDF data stores [1]

RDF Stores	1M	25M	100M
Sesame	662	200	15
Jena TDB	494	165	35
Virtuoso RV	199	173	122
D2R Server	328	236	79
MYSQL	3,021	955	476

Table 2.2: Find Product for given set of generic features [1]

feature. The Responce time of query is given as below.

Table 2.2 shows the result of query1 in terms of QpS for 1M, 25M and 100M data-set. A response time of SQL query is more better than SPARQL query.

Table 2.3 shows the result of query2 in terms of QpS for 1M, 25M and 100M data-set. A response time of SQL query is more better than SPARQL query. Here we can conclude that response time of SQL is more preferable than SPARQL. That is why we can use Apache Hive for Information Extraction

2.5 RDF Processing in Distributed Hadoop Environment

Distributed environment is better idea to achieve goal, when your job is very large. In distributed computing computers are connected through high speed network, communicate with each other and work for achieve common goal.

RDF Stores	1M	25M	100M
Sesame	505	140	13
Jena TDB	451	141	28
Virtuoso RV	182	167	129
D2R Server	226	115	56
MYSQL	2,833	919	459

Table 2.3: Find Product having some specific features and not having one feature. [1]

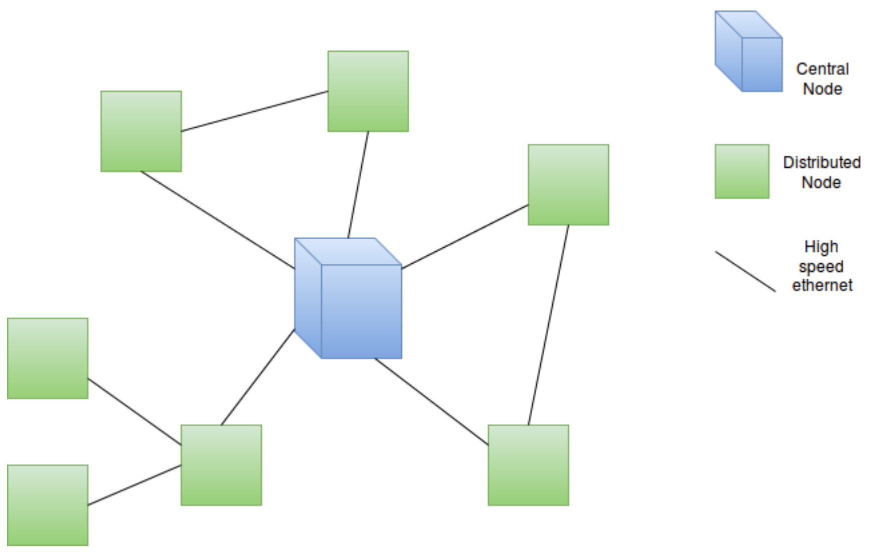


Figure 2.6: Distributed Environment

2.5.1 Overview of Apache technologies

For setting up a cluster we use Apache Hadoop and hive technology. Hadoop uses a google's very famous algorithm Map-reduce. Hadoop is open source distributed framework for data storage. As mention in above section Hadoop distributed the task to the data node, which call mapping task. Collecting all result to the name node is call Reducing phase.

The scenario of today's world is that most of big-data technology running on mapreduce. Technology like Apache Spark, Hbase, Pig and Apache hive. For all this technology to run installation of Apache Hadoop is mandatory.

2.5.2 Hadoop Technologies

Hadoop is invented by Dough Cutting and Mike Cafarella in 2005. Hadoop is a distributed computing environment in which jobs are divided in to chunks. One chunk has maximum size of 64MB. In Haddop we can analyze every type of data whether it is csv file, log file or audio and video file.

Hadoop cluster have mainly two components. (1) Name Node (2) Data Nodes. Name Node stores the meta data about data stored in data node. For Data Integrity and availability every datanode replicate their the content to other datanode, this is concept of replication. This replication is used in case of data loss or in language of security it is call Data Integrity and availability .

Hadoop is kind of Master Slave Model. We can say that name node is a master and data node is a slave. There can be many data nodes but name node is only one. User submit data to hadoop from name node. Task of Hadoop is described as follow.

Hadoop Flowchart:-

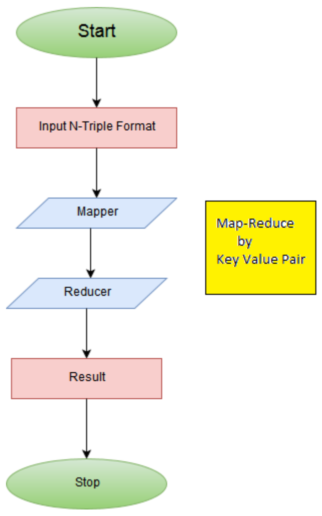


Figure 2.7: Hadoop Processing Flowchart [6]

- Mapper : Here name node divided the task between data nodes. This process is known as mapper task. Data is divided in to chunks and delivered to data nodes.

- Shuffling : This is intermediate stage of mapper and reducer. In this stage intermediate result is sorted.

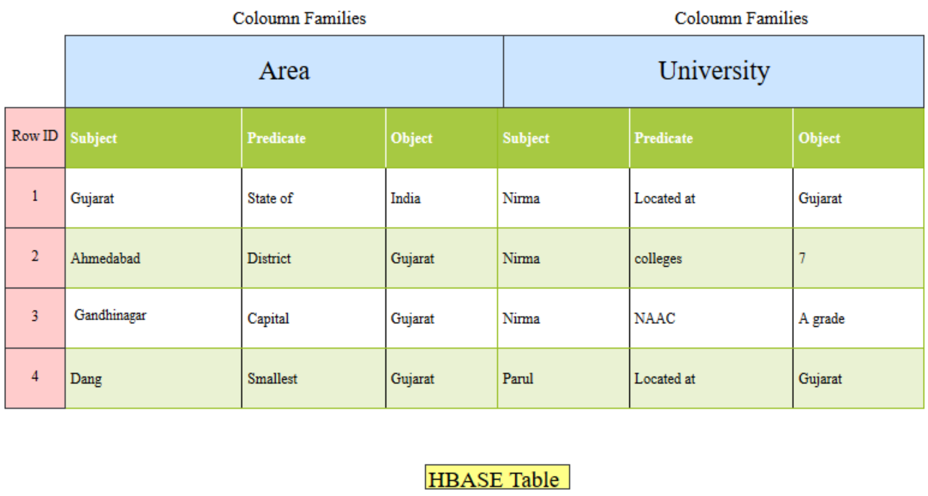
- Reducer : In this phase result from all nodes reduce to the name node.

Here we can process the RDF data with the help of hadoop mapreduce. An algorithm for mapreduce of RDF.

Apache Hbase :

Apache Hbase is distributed table, model after Google's Big-table and is written in java. Apache Hbase is column oriented database works on Hadoop [18]. Apache Hbase use concept of table. Different from SQL it has main feature column, while SQL has row. Figure given below explain architecture of Hbase.

Hbase Architecture in Fig. 2.7 shows complete storage mechanism of Apache Hbase. Two column families Area and University is presented in architecture. Every column family contains three columns subject, predicate and object. Every row has unique row ID. Result can derived from mapping between different column families. Hbase table can store thousand TB of data and distributed it on large clusters.



	Coloumn Families			Coloumn Families		
	Area			University		
Row ID	Subject	Predicate	Object	Subject	Predicate	Object
1	Gujarat	State of	India	Nirma	Located at	Gujarat
2	Ahmedabad	District	Gujarat	Nirma	colleges	7
3	Gandhinagar	Capital	Gujarat	Nirma	NAAC	A grade
4	Dang	Smallest	Gujarat	Parul	Located at	Gujarat

HBASE Table

Figure 2.8: Hadoop Architecture

Let us take one simple example. . .

Create table in Hbase:

create ’university’, ’student data’, professor data’

Here University is table name and student data and professor data is two column families.

Data entry in table:

put ’university’,’1’,’student data:name’,’raju’

put ’university’,’1’,’student data:roll_no’,’44’

put ’university’,’1’,’professor data:name’,’vikas’

Above line shows that we put data in two column families whose row ID is ’1’. Student data has two columns ’name’ and ’roll_no’, while professor has only one column ’name’.

Appache Hive

Apache Hive is developed by facebook. Apache hive has similar query syntax like SQL calls HQL(Hive query language). Hive is installed on Hadoop distributed environment. Some times it is very difficult to analyze data with map-reduce code because it is difficult to write map-reduce code for large application. HQL converts the query in to map-reduce code internally and gives desire output[1].

[1]https://www.tutorialspoint.com/hive/

Chapter 3

Proposed Architecture

3.1 Problem Statement

Storing a large amount of RDF data generated everyday and required time to information extraction from that data.

3.1.1 Description of Problem Statement

21st century is an era of information. Information generated very rapidly now a days. By 2020, 40 zeta-byte of data will exist on this earth and Cisco has predicted that by 2020 [5]. Globally IP traffic will reached up to 2.3 Zeta-byte[1]. These are very large amount of data. The main problem is data is generated every day and every moment. Here Organizations faces two problems (1) Processing Data (2) Storing Data. Second problem solved by cloud provider organizations like Amazon EC2. First problem processing the data is huge problem. Today most of the organization turn their faces to distributed processing.

Berlin SPARQL benchmark proves that SQL is much more efficient than SPARQL [1]. Here we can use similar query language like SQL to extract information. Distributed computing is good approch for storing and processing. Hadoop can process the data with map-reduce code. One problem with hadoop is, it needs map-reduce code for analyze the data every time. This is very tedious process to get an result. That is why Apache hive is useful.

[1]http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html

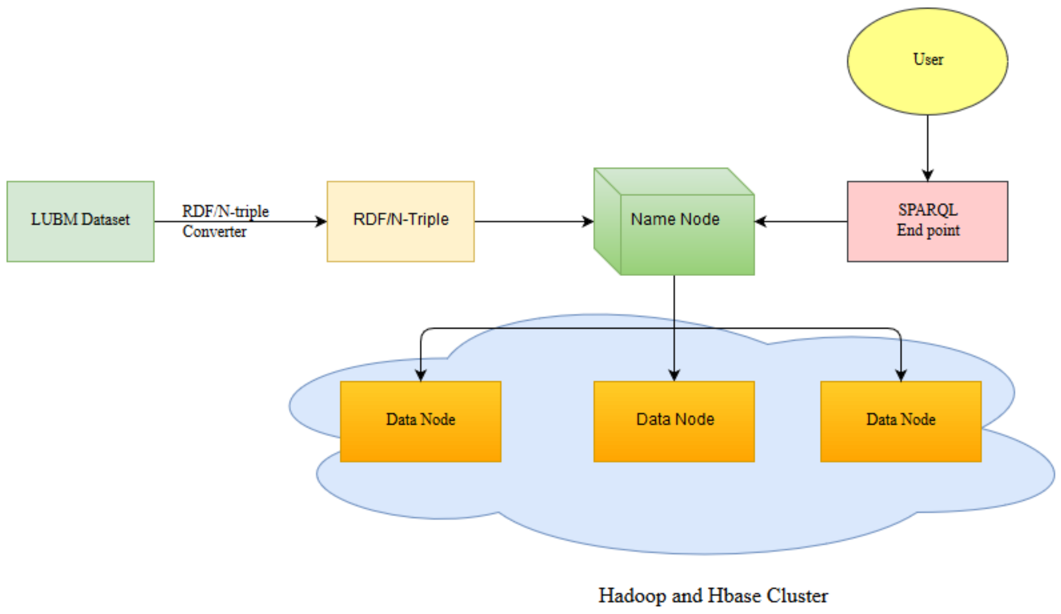


Figure 3.1: Proposed Architecture

3.2 Our Approach with Apache Hive

To implement our approach with Apache Hive, we have downlodede data-set from official DBpedia resource[2]. These dataset are available in form of N-Triple format and ttl format. We use N-Triple format to load data on Hive.

RDF is a graph database which form of triples; Subject, Predicate, Object. RDF have many form in which it can be represented like N-Triple, Turtle, Json, RDF/XML, OWL etc. We can convert RDF to CSV(comma separated value) format with the help of some online tools. Convert RDF or OWL to N-Triple format with easy RDF converter.

Let us look at N-Triple format.

```
<: Iphone> <: ownedBy> <: Apple> .
<: Apple> <: hasWorth> <: 800 Billion> .
<: Safari> <: madeBy> <: Apple> .
```

Above is a simple example of triple. For simplisity we use name. Normaly N-Triple format contains full URI.

[2]http://oldwiki.dbpedia.org/Downloads2014

Chapter 4

Implementation

4.1 Implementation

4.1.1 Component Definition

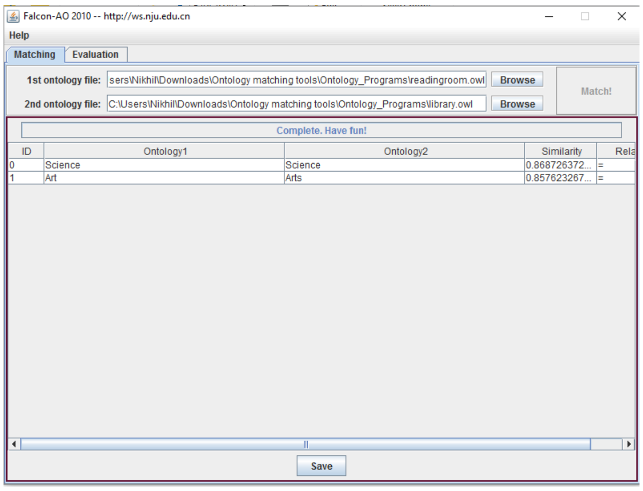
Here discussed about databases on which rdf triple can be store. Jena TDB is more reliable resource for storing rdf triple

- Jena TDB: Jena TDB is use to store RDF triples. Using Jena TDB we store large number of RDF Triple on single machine. It uses JVM for execute. We use Jena with Eclipse.
- Hbase: It is a Hadoop database. It is distributed among various machine. It is very large scale database.
- Open TSDB: It is built on Hbase and Hadoop. It is use to store data generated continuous with time.
- Elastic Search: It is search engine based on java. Use schema free JSON document. Develop in java.

4.2 Ontology Mapping using Falcon

Falcon is an Ontology alignment tool. It works by comparing properties of two ontology. Ontology contains a various classes and every class has property. Falcon compares these properties and gives similarity index **s**.

As shown in figure we compare two ontology reading room and library. We get the result of similarity index and relation.



Falcon-AO 2010 -- http://ws.nju.edu.cn

1st ontology file: sers\Nikhil\Downloads\Ontology matching tools\Ontology_Programs\readingroom.owl

2nd ontology file: C:\Users\Nikhil\Downloads\Ontology matching tools\Ontology_Programs\library.owl

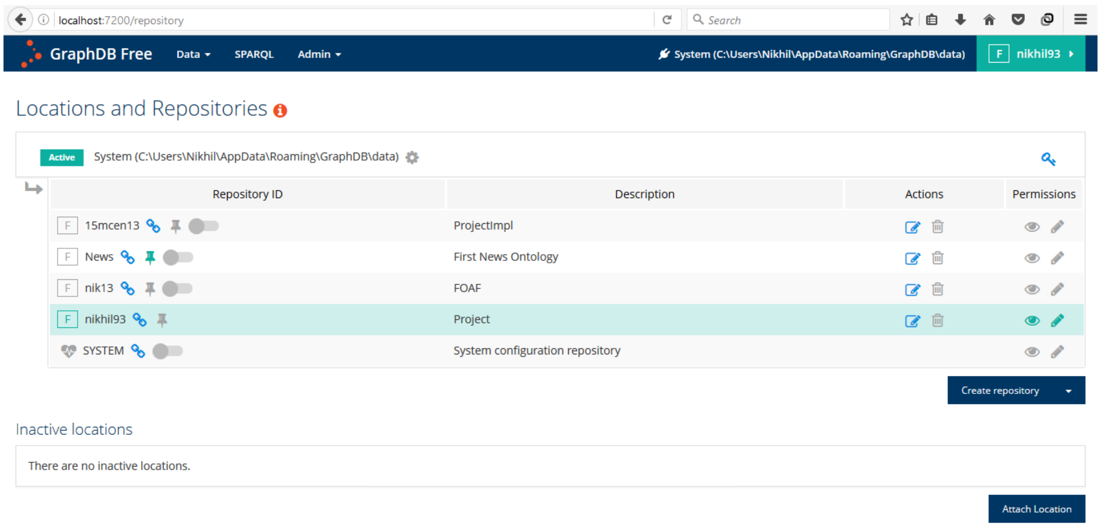
Complete. Have fun!

ID	Ontology1	Ontology2	Similarity	Rela
0	Science	Science	0.868726372...	=
1	Art	Arts	0.857623267...	=

Figure 4.1: Ontology Mapping with Falcon

4.3 Ontotext GraphDB

GraphDB is a platform where you can create and import and manipulate rdf graph. Graphdb has mainly three buttons on tab Data, SPARQL and Admin. In admin section there is Location and Repository where you can create new repository. Here you provide repository ID and name. After the creating repository you can import the file in to it or export from it.



localhost:7200/repository

Locations and Repositories

Active System (C:\Users\Nikhil\AppData\Roaming\GraphDB\data)

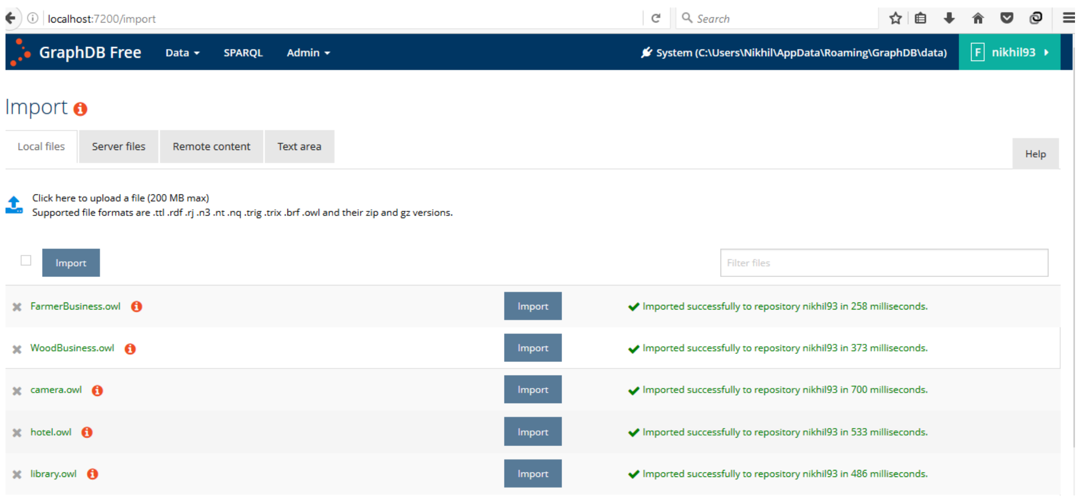
Repository ID	Description	Actions	Permissions
15mcen13	ProjectImpl		
News	First News Ontology		
nik13	FOAF		
nikhil93	Project		
SYSTEM	System configuration repository		

Inactive locations

There are no inactive locations.

Figure 4.2: Ontotext GraphDB Repository

From data section select import option to import rdf file. Here system supports various types of file like .ttl, .rdf, .owl, .nq, .trix, .trig etc. You can import files in four various ways 1. Import from system 2. Server files 3. Remote content 4. Text Area.



Click here to upload a file (200 MB max)
Supported file formats are .ttl .rdf .rj .n3 .nt .nq .trig .trix .brf .owl and their zip and gz versions.

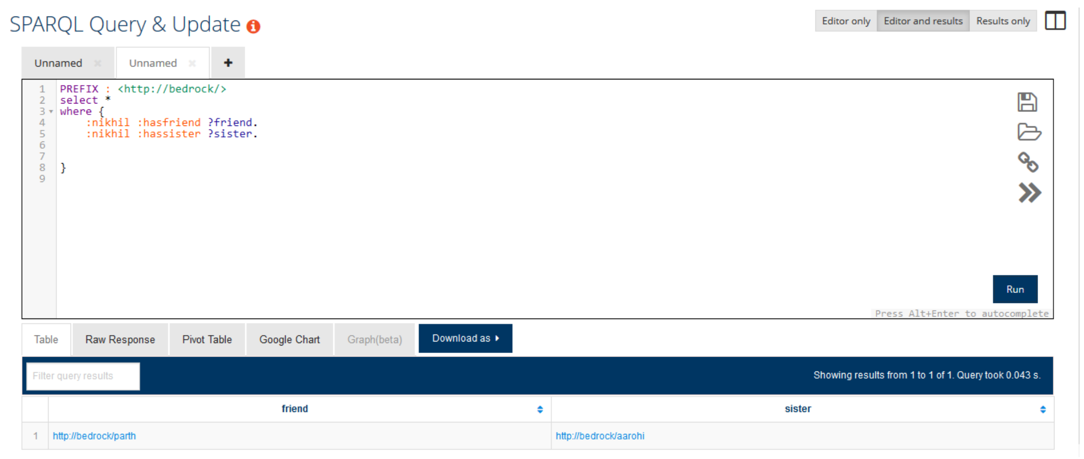
FarmerBusiness.owl	Import	✓ Imported successfully to repository nikhil93 in 258 milliseconds.
WoodBusiness.owl	Import	✓ Imported successfully to repository nikhil93 in 373 milliseconds.
camera.owl	Import	✓ Imported successfully to repository nikhil93 in 700 milliseconds.
hotel.owl	Import	✓ Imported successfully to repository nikhil93 in 533 milliseconds.
library.owl	Import	✓ Imported successfully to repository nikhil93 in 486 milliseconds.

Figure 4.3: Import data in GraphDB Repository

- import from system: In this type you are going to create a rdf or owl file on your system and import it in to your system.
- . Server File: Here put your content on server and access your file through server. Here you put your files in to your graphdb import folder.
- Remote content: Here provide data URL for operation on data. Here all data is on the web and it extract it online.
- Area: Create your own code here.

SPARQL

With the help of SPARQL you can create your own ontology and you can query on your database created by your ontology.



```
PREFIX : <http://bedrock/>
select *
where {
    :nikhil :hasfriend ?friend.
    :nikhil :hassister ?sister.


}
```

Showing results from 1 to 1 of 1. Query took 0.043 s.

	friend	sister
1	http://bedrock/parth	http://bedrock/aarohi

Figure 4.4: SPARQL End Point for Ontotext GraphDB

Class Hierarchy

Here class hierarchy is made from imported class.

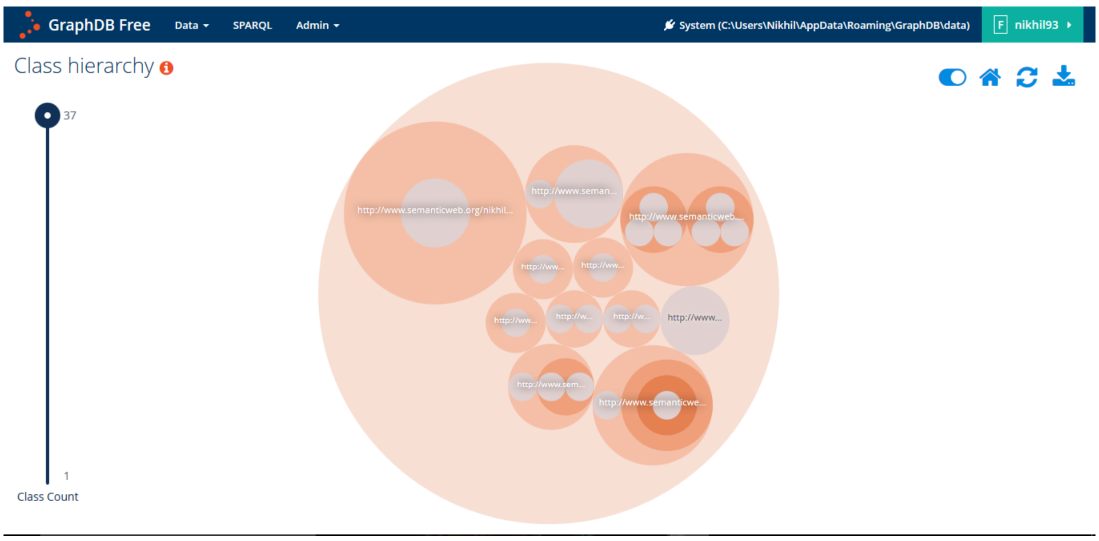


Figure 4.5: Class hierarchy of Imported Data

Domain Range Graph.

Domain Range graph shows the boundary of particular class. Domain is base class of give class and range is value. For example Airlift is directed by Niraj Pande. Here Airlift is domain of director and Niraj Pande is value means range.

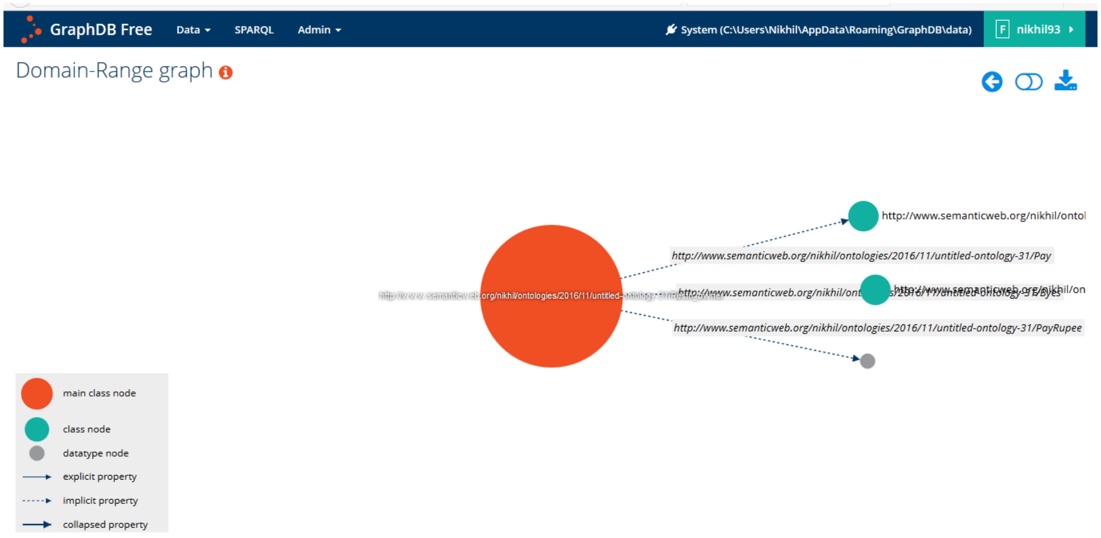

GraphDB Free
Data
SPARQL
Admin
System (C:\Users\Nikhil\AppData\Roaming\GraphDB\data)
nikhil93
Domain-Range graph
http://www.semanticweb.org/nikhil/ontologies/2016/11/untitled-ontology-31/Resto_owner
http://www.semanticweb.org/nikhil/ontol
http://www.semanticweb.org/nikhil/ontologies/2016/11/untitled-ontology-31/Pay
http://www.semanticweb.org/nikhil/ontologies/2016/11/untitled-ontology-31/Byes
http://www.semanticweb.org/nikhil/on
http://www.semanticweb.org/nikhil/ontologies/2016/11/untitled-ontology-31/PayRupee
main class node
class node
datatype node
explicit property
implicit property
collapsed property

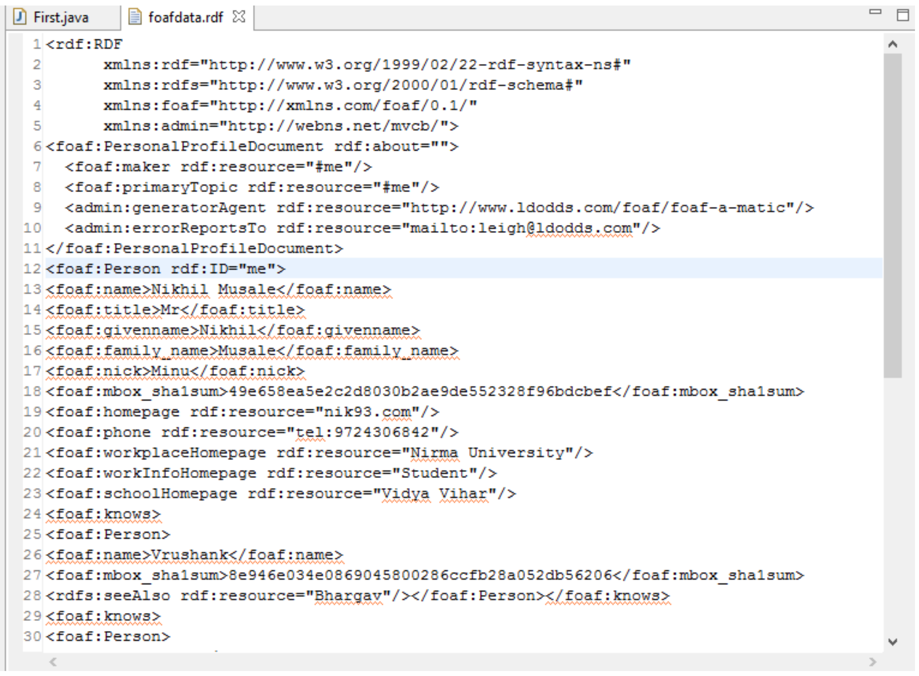
Figure 4.6: Domain Range Graph

4.4 Jena TDB

TDB is part of Jena API which is use for store RDF data or RDF triples, with the help of jena API you can store the large number of RDF triples on a single machine.

A TDB get direct access to JVM. We can use Jena TDB with Eclipse. For that install Jena TDB latest version and import all jar files of Jena lib folder in to Eclipse project. This is how we can use Jena TDB with Eclipse.

Jena Fuseki provides SPARQL server, it is very similar to Ontotext GraphDB.



```
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:foaf="http://xmlns.com/foaf/0.1/"
        xmlns:admin="http://webns.net/mvcb/">
<foaf:PersonalProfileDocument rdf:about="">
  <foaf:maker rdf:resource="#me"/>
  <foaf:primaryTopic rdf:resource="#me"/>
  <admin:generatorAgent rdf:resource="http://www.ldodds.com/foaf/foaf-a-matic"/>
  <admin:errorReportsTo rdf:resource="mailto:leigh@ldodds.com"/>
</foaf:PersonalProfileDocument>
<foaf:Person rdf:ID="me">
<foaf:name>Nikhil Musale</foaf:name>
<foaf:title>Mr</foaf:title>
<foaf:givenname>Nikhil</foaf:givenname>
<foaf:family_name>Musale</foaf:family_name>
<foaf:nick>Minu</foaf:nick>
<foaf:mbox_sha1sum>49e658ea5e2c2d8030b2ae9de552328f96bdcbef</foaf:mbox_sha1sum>
<foaf:homepage rdf:resource="nik93.com"/>
<foaf:phone rdf:resource="tel:9724306842"/>
<foaf:workplaceHomepage rdf:resource="Nirma University"/>
<foaf:workInfoHomepage rdf:resource="Student"/>
<foaf:schoolHomepage rdf:resource="Vidya Vihar"/>
<foaf:knows>
<foaf:Person>
<foaf:name>Vrushank</foaf:name>
<foaf:mbox_sha1sum>8e946e034e0869045800286ccfb28a052db56206</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="Bhargav"/></foaf:Person></foaf:knows>
<foaf:knows>
<foaf:Person>
```

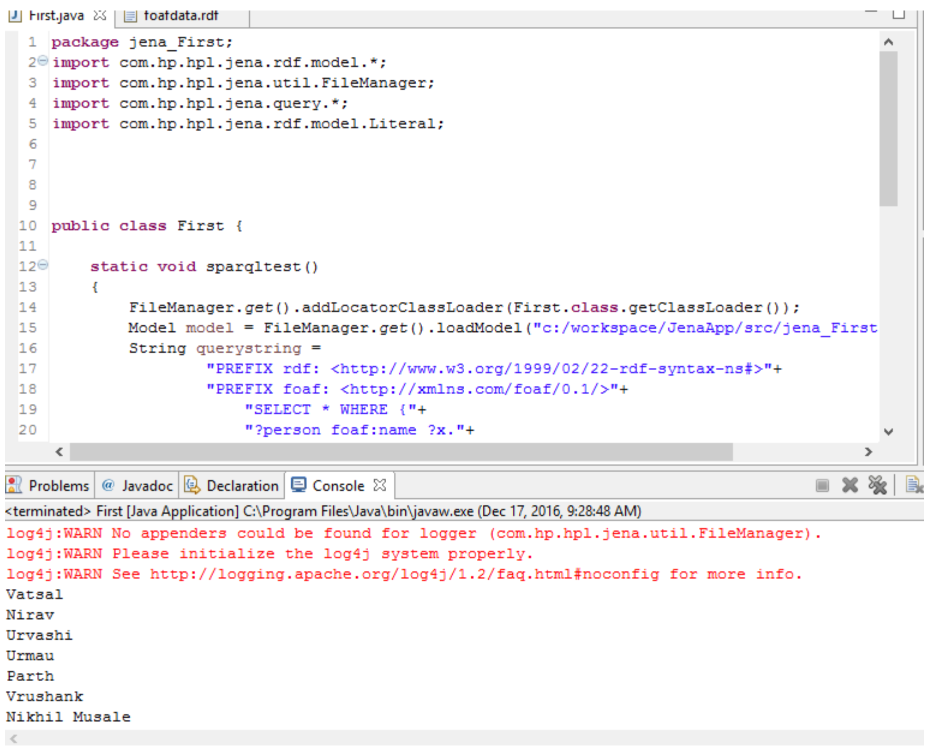
Figure 4.7: FOAF RDF file

4.5 Apache Hive

Apache Hive is developed by facebook. Apache hive has similar query syntax like SQL calls HQL(Hive query language). Hive is installed on Hadoop distributed environment. Some times it is very difficult to analyze data with map-reduce code because it is difficult to write map-reduce code for large application. HQL converts the query in to map-reduce code internally and gives desire output.

4.5.1 Results

The given Result set we derived for processing N-Triple format of RDF data on Jena TDB(Table-1) and Apache Hive(Table-2).



```
package jena_First;
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.util.FileManager;
import com.hp.hpl.jena.query.*;
import com.hp.hpl.jena.rdf.model.Literal;




public class First {

    static void sparqltest()
    {
        FileManager.get().addLocatorClassLoader(First.class.getClassLoader());
        Model model = FileManager.get().loadModel("c:/workspace/JenaApp/src/jena_First
        String querystring =
                "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"+
                "PREFIX foaf: <http://xmlns.com/foaf/0.1/>"+
                    "SELECT * WHERE {"+
                    "?person foaf:name ?x."+
```

```
<terminated> First [Java Application] C:\Program Files\Java\bin\javaw.exe (Dec 17, 2016, 9:28:48 AM)
log4j:WARN No appenders could be found for logger (com.hp.hpl.jena.util.FileManager).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Vatsal
Nirav
Urvashi
Urmau
Parth
Vrushank
Nikhil Musale
```

Figure 4.8: Jena Query Results in Eclipse

Triples	Time (milli seconds)
277000	8064
554000	12614
831000	17063
1108000	exception

Table 4.1: Result Table Jena TDB(Time is given in second)

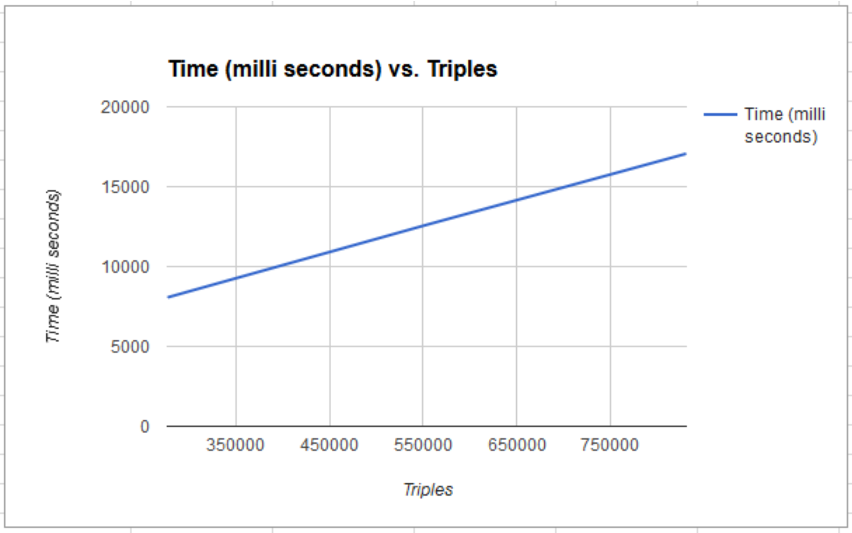


Figure 4.9: Result Graph of Simple Select

Apache provides a Jena API. Graph in Fig 4.9 shows that execution time is increase with increase of data. This implementation is done on Eclipse IDE. We install all depedency of Jena API and upload RDF file which contains triple as shown in table 4.1. It takes the time to execute and it is increase with data increase.

Time in Seconds		
Operations	**1 Lakh Triple**	**2 Lakh Triple**
loding Time	1.623	1.824
Select all(complete display time)	15	40
Simple Select	0.242	0.408
Select one triple	0.493	0.866
Select Distinct (map-reduce time included)	34	45

Table 4.2: Result Table on hive.(Time is given in second)

Table 4.2 shows the Result of various select for Apache hive. It includes Loading time, Select all, Simple Select etc. By Analyzing the table we can see that Apache Hive requires very less time to generate results.

Time in Seconds		
Dataset	**Apache Jena**	**Apache Hive**
300000	8	0.408
600000	12	0.566
900000	17	0.705
1000000	25	1.215

Table 4.3: Results of Simple Select

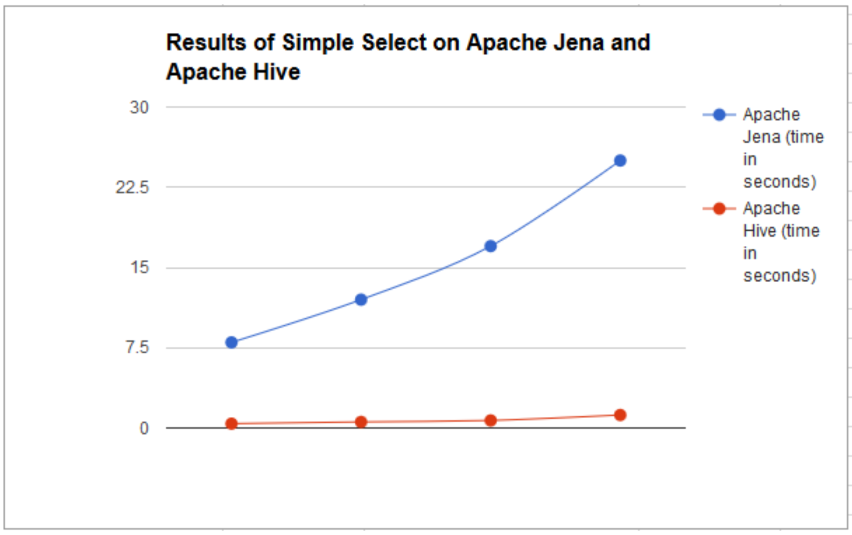


Figure 4.10: Result Graph of Simple Select

Graph in Fig 4.10 shows the comperision between Jena API and Apache Hive. From Table 4.3 and Fig 4.10 We can conclude that Apache hive has much more better result then Jena TDB.

Time in Seconds		
Triples(Size)	**Subquery**	**Join query**
2 lakh(22 MB)	20	21
10 lakh(113.1 MB)	23	21
20 lakh(206.4 MB)	31	29
30 lakh(299 MB)	71	73
50 lakh (586.5 MB)	145	137

Table 4.4: Comparison between query type

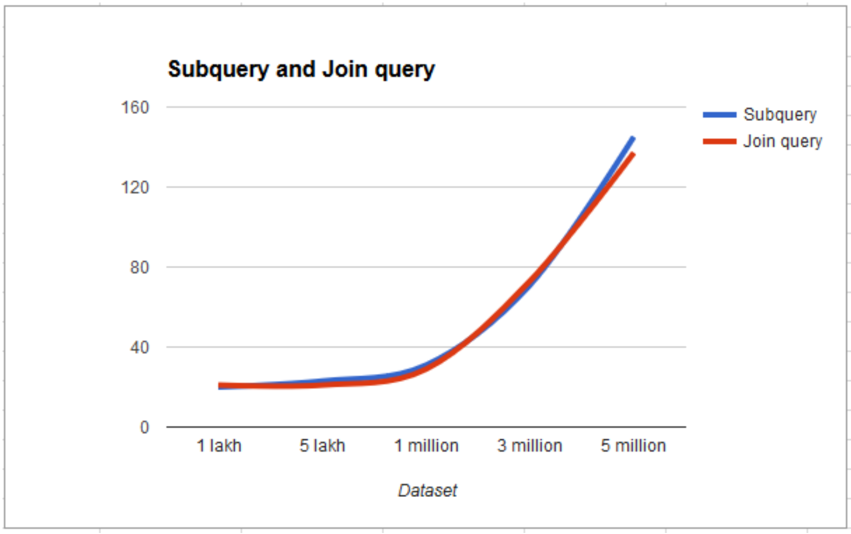


Figure 4.11: Comparison between query type

Table 4.4 Shows the comparison between two kind of query. (1)Subquery means nested query and (2) Join query. It shows the result for given data-set. From Table 4.4 and Fig 4.11 We can say that, results from both query has no more difference. Both type of query is same efficient.

Results in Query per Second			
Query Type	**Apache Hive**	**SQL**	**Sesame**
Join	15453	4525	662
Union	12658	3021	550
Distinct	13901	4145	750

Table 4.5: Comparison between Apache Hive SQL and Sesame in terms of QpS.

Table 4.5 and Fig 4.12 show the comparison between Apache Hive, SQL and sesame for various query type in terms of Query per Second. From results we can say that Apache Hive performs best.

In table 4.5 Only Apache Hive is implemented. Result for SQL and Sesame taken from paper The Berlin SPARQL Benchmark [1].

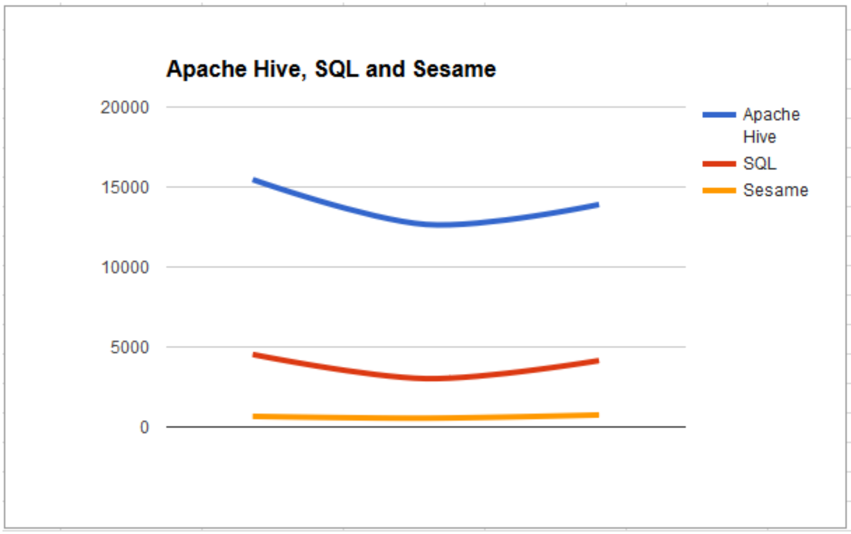


Figure 4.12: Comparison between Apache Hive SQL and Sesame in terms of QpS.

Hive gives better results on small data-set as well as on large data-set because Hive is built on top of Hadoop. When user writes query on Apache Hive, it will internaly converted in to map-reduce code and run on to HDFS file system. For every query there will be map-reduce process activated. Process will run on cluster. For These reson Apache Hive is better option.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

We explore the storage and retrieval of information from ontology. Usually Ontology is generated from RDF. With the help of Ontology mapping, we can find similarity between RDF classes in terms of similarity index. Ontology mapping help us to find most relevant information. Semantic web covers area like IOT. Data generated by IOT can be converted into RDF graph. We can apply queries to RDF graph through SPARQL. Some times knowledge base like DBpedia is used to give answer of query by mapping the information with data generated from IOT. The Berlin SPARQL benchmark proves that SQL is more faster than SPARQL in retrieving result.It is challange to store and retrieve data from single RDF data store like Apache Jena and Seasame. To solve this problem, we need distributed environment. Results can be improved with Apache Hive. Hive works on top of Hadoop. Hive query language is similar to SQL. N-Triple format of RDF is lodead into Hive. HQL internally converted into map-reduce code and give results. Generated results are better then the result of SPARQL and SQL.

Our results clearly shows that Apache Hive is more better than any other data storage system. Table 4.3 shows the comparison between Apache Jena and Hive for triples of 300000 Apache Jena takes 8 seconds and Apache Hive takes 0.408 seconds. We also compare the result of types of HQL query like sub-query and join query, both type of query have similar result. For 1 lakh data-set sub-query takes 20 seconds and join query takes 21 seconds. In third result we compare Apache Hive with SQL and Sesame. Results are in QpS. Here for join query, Apache hive has QpS 15453, SQL has 4525 and Sesame has

662. From result we can easily conclude that Distributed Computing for RDF processing is better for query processing in terms of time.

Future Work

In the future work, we can collect data generated from various IOT devices and convert it in to N-Triple format. Load this data onto Hadoop cluster and apply various Apache Hive queries. For the semantic question answering system we can convert user question in to Apache query with the help on NLP(Natural Language Processing) and pass it to the system. This gives very accurate answer to the question of user. By using these techniques we can create Semantic question answering system.

References

[1] C. Bizer and A. Schultz, "The berlin sparql benchmark," 2009.

[2] P. Shvaiko and J. Euzenat, "Ontology matching: state of the art and future challenges," *IEEE Transactions on knowledge and data engineering*, vol. 25, no. 1, pp. 158–176, 2013.

[3] J. Bhogal, A. Macfarlane, and P. Smith, "A review of ontology based query expansion," *Information processing & management*, vol. 43, no. 4, pp. 866–886, 2007.

[4] Q. Wu, P. Wang, C. Shen, A. v. d. Hengel, and A. Dick, "Ask me anything: Free-form visual question answering based on knowledge from external sources," *arXiv preprint arXiv:1511.06973*, 2015.

[5] D. Le-Phuoc, H. N. M. Quoc, H. N. Quoc, T. T. Nhat, and M. Hauswirth, "The graph of things: A step towards the live knowledge graph of connected things," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 37, pp. 25–35, 2016.

[6] J. Kawises and W. Vatanawood, "A development of rdf data transfer and query on hadoop framework," in *Computer and Information Science (ICIS), 2016 IEEE/ACIS 15th International Conference on*, pp. 1–4, IEEE, 2016.

[7] T. Berners-Lee, J. Hendler, O. Lassila, *et al.*, "The semantic web," *Scientific american*, vol. 284, no. 5, pp. 28–37, 2001.

[8] L. A. Barroso, J. Dean, and U. Holzle, "Web search for a planet: The google cluster architecture," *IEEE micro*, vol. 23, no. 2, pp. 22–28, 2003.

[9] G. Antoniou and F. Van Harmelen, *A semantic web primer.* MIT press, 2004.

[10] M. Mao, Y. Peng, and M. Spring, "An adaptive ontology mapping approach with neural network based constraint satisfaction," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 8, no. 1, pp. 14–25, 2010.

[11] B. Glimm, I. Horrocks, B. Motik, R. Shearer, and G. Stoilos, "A novel approach to ontology classification," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 14, pp. 84–101, 2012.

[12] R. Usbeck, A.-C. N. Ngomo, L. Bühmann, and C. Unger, "Hawk–hybrid question answering using linked data," in *European Semantic Web Conference*, pp. 353–368, Springer, 2015.

[13] M.-C. Yang, D.-G. Lee, S.-Y. Park, and H.-C. Rim, "Knowledge-based question answering using the semantic embedding space," *Expert Systems with Applications*, vol. 42, no. 23, pp. 9086–9104, 2015.

[14] M. Vargas-Vera, E. Motta, and J. Domingue, "Aqua: An ontology-driven question answering system.," in *New Directions in Question Answering*, pp. 53–57, 2003.

[15] A. Loizou, R. Angles, and P. Groth, "On the formulation of performant sparql queries," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 31, pp. 1–26, 2015.

[16] W.-D. Fang, L. Zhang, Y.-X. Wang, and S.-B. Dong, "Toward a semantic search engine based on ontologies," in *2005 International Conference on Machine Learning and Cybernetics*, vol. 3, pp. 1913–1918, IEEE, 2005.

[17] M. Arenas, B. C. Grau, E. Kharlamov, Š. Marciuška, and D. Zheleznyakov, "Faceted search over rdf-based knowledge graphs," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 37, pp. 55–74, 2016.

[18] V. Khadilkar, M. Kantarcioglu, B. Thuraisingham, and P. Castagna, "Jena-hbase: A distributed, scalable and efficient rdf triple store," in *Proceedings of the 2012th International Conference on Posters & Demonstrations Track-Volume 914*, pp. 85–88, CEUR-WS. org, 2012.