RDF REPOSITORY REPLACING RELATIONAL DATABASE

[1]B.Srinivasa Rao, [2]Dr.G.Appa Rao
[1,2]Department of CSE, GITAM University
Email:[1]ugandarsrinu@gmail.com,[2]apparao_999@yahoo.com

Abstract-- This study is to propose a flexible information storage mechanism based on the principles of Semantic Web that enables information to be searched rather than queried. In this study, a prototype is developed where the focus is on the information rather than the structure. Here information is stored in a structure that is constructed on the fly. Entities in the system are connected and form a graph, similar to the web of data in the Internet. This data is persisted in a peculiar way to optimize querying on this graph of data. All information relating to a subject is persisted closely so that reqeusting any information of a subject could be handled in one call. Also, the information is maintained in triples so that the entire relationship from subject to object via the predicate is captured in one record.

Key Concepts: Entity, Subject, Predicate, Object, RDF Triple

1. Introduction

The Semantic Web is a collaborative movement led by the international standards body, the World Wide Web Consortium (W3C).[1] The semantic web is a "web of data" that enables machines to understand the semantics of information on the World Wide Web. The term "Semantic Web" is often used more specifically to refer to the formats and technologies that enable it. One such technology is RDF (Resource Description Framework)[2]. RDF is a directed, labelled graph for representing information in the Web. This can be perceived as a repository without any predefined structure

The information stored in the traditional RDBMS's requires structure to be defined upfront. On the contrary, information could be very complex to structure upfront despite the tremendous potential offered by the existing database systems. In the ever changing world, another important characteristic of information in a system that impacts its structure is the modification/enhancement to the system. This is a big concern with many software systems that exist today and there is no tidy approach to deal with the problem.

1.1. Relational databases

Relational databases need a structure to be defined upfront and this might not be feasible always with the complexities in the real world.Relational databases have immense capabilities to offer, however they do pose some serious concerns in some aspects and the most important of them are

- Enhancement to structure
    - requires a great deal of effort to update the structure of information stored
    - total understanding of the system and its dependencies

  - complex analysis to ensure the system is not impacted even after the structural change
  - focussed monitoring once the change is put in place
- Scaling
  - scaling in line with the application is not always possible because of centralized nature
  - when data set tends to grow, traditional databases cannot handle it

2. Background

2.1. RDF Model

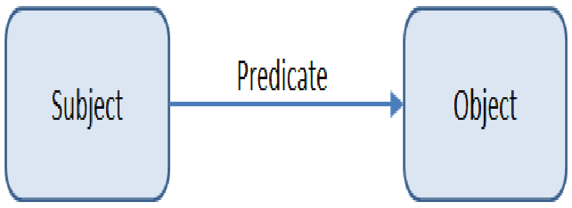


Figure 1: RDF Model

The RDF model has a triple consisting of subject, predicate and an object. The subject is the entity that is being dealt with i.e. the subject and its characteristics are being discussed. Predicate indicates one characteristic type of the subject and the object represents the characteristic.

An analogy to understand this better is to consider the {predicate, object} pair as an {attribute name, attribute value} pair for the subject. A subject can have multiple attributes, and each of subject, attribute name and attribute value is fully scoped. This ensures there is a context associated with each triplet.Figure 1: RDF Model

A subject can have any number of characteristics and an object in one context could be a subject in another context. Similarly, a predicate could also be a subject or an object in other contexts.

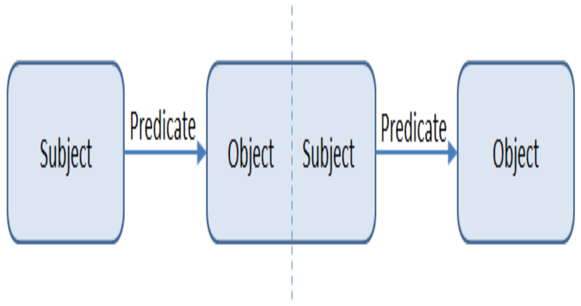


Figure 2: Generalized RDF Model

The above picture represents an entity being an object and a subject in different contexts.

2.2. Class Diagram

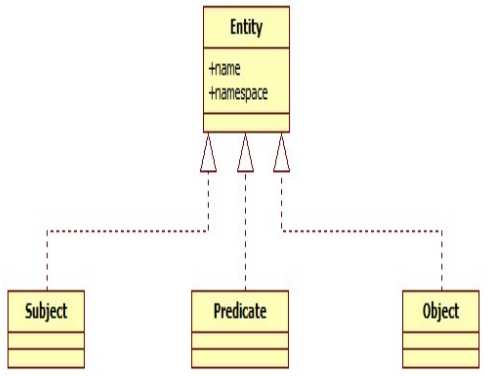


Figure 3: Class diagram

The RDF data model is similar to classic conceptual modelling approaches such as Entity-Relationship or Class diagrams,[10] as it is based upon the idea of making statements about resources (in particular Web resources) in the form of subject-predicate-object expressions. These expressions are known as *triples* in RDF terminology. The subject denotes the resource, and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and the object.

RDF is based on the idea of identifying things using Web identifiers (called *Uniform Resource Identifiers*, or *URIs[4]*), and describing resources in terms of simple properties and property values. This enables RDF to represent simple statements about resources as a *graph* of nodes and arcs representing the resources, their properties and values.

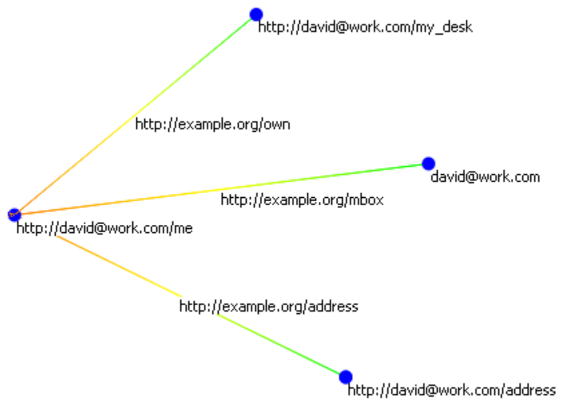

http://david@work.com/my_desk
http://example.org/own
david@work.com
http://example.org/mbox
http://david@work.com/me
http://example.org/address
http://david@work.com/address

Figure 4: An RDF Graph Describing David

Figure 4 illustrates that RDF uses URIs to identify individuals, physical objects, properties and values of properties.

This concept is extended to define relations between objects in a relational database thereby eliminating the structure. The main advantage that this approach offers is the traversal from one object to another based on the relations either directly or by hopping across objects.

2.3. Storage and querying

The RDF store can be a file system, RDBMS or any other store.

Let us consider information of individuals and their discoveries or pioneering works. Then we query the store for the mailbox and pioneering work where both of them exist. This is a simple case that shows that information can be queried from an RDF store using query languages very similar to the ones used to query traditional relational databases.

The RDF format consists of triples namely Subject, Object and Predicate. The Subject and Object are related to each other via a Predicate. This information can be maintained as simple XML as shown below

```
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:si="http://www.w3schools.com/rdf/">

  <rdf:Description rdf:about="Thomas Edison">

    <si:inventor>Electric bulb</si:inventor>

<si:mbox>tedison@example.com</si:mbox>

  </rdf:Description>

  <rdf:Description rdf:about="Albert Einstein">

    <si:discoverer>Photoelectric Effect</si:discoverer>

<si:mbox>aeinstein@example.com</si:mbox>

  </rdf:Description>

  <rdf:Description rdf:about="Max Planck">

    <si:founder>Quantum Theory</si:founder>

<si:mbox>mplank@example.com</si:mbox>

  </rdf:Description>

  <rdf:Description rdf:about="Max Planck">

    <si:age>43</si:age>

  </rdf:Description>

  <rdf:Description rdf:about="Marie Curie">

<si:mbox>mcurie@example.com</si:mbox>

  </rdf:Description>

  <rdf:Description rdf:about="Marie Curie">

    <si:pioneer>Radioactivity</si:pioneer>

  </rdf:Description>

</rdf:RDF>
```

This information is maintained in an RDF store without knowledge of how it is structured or maintained internally by RDF engines. This gives the flexibility to add new dimensions to the existing information simply executing a query. On the contrary, when a relational database has to be updated with a new dimension, a lock has to be acquired and the database is not usable until the dimension is added.

Traditional RDBMS's are queried using SQL, similarly there are querying languages like SPARQL[5]/SeRQL[6] to query information from an RDF store. SPARQL allows for a query to consist of triple patterns, conjunctions, disjunctions, and optional patterns.

In order to fetch the mailbox and pioneering work where both exist, a SPARQL query would look like

SELECT ?mbox ?pioneer where {?x <http://www.w3schools.com/rdf/mbox> ?mbox . ?x <http://www.w3schools.com/rdf/pioneer> ?pioneer}

This returns the response

"Radioactivity" mcurie@example.com

One of the core concepts of RDF is its open structure i.e. there is no need for a structure to be defined upfront. This property removes the complexity of updating the system. Information in RDF could be distributed across multiple systems like the information on the web. This distributed nature of persisting information allows the systems relying on RDF to scale without the need to do a lot of manual configuration.

This helps in overcoming the scaling problem.

2.4. MongoDB

In this section, we give a brief overview of MongoDB released with an AGPL license [7]. MongoDB (from "humongous") is a scalable, high-performance, open source NoSQL database [8]. It is part of the NoSQL family of database systems. Instead of storing data in tables as is done in a "classical" relational database, MongoDB stores structured data as JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster.

MongoDB scales horizontally using sharding [9]. The developer chooses a shard key, which determines how the data in a collection will be distributed. The data is split into ranges (based on the shard key) and distributed across multiple shards. (A shard is a master with one or more slaves.)

3. Architecture

Our system architecture consists of three major components

1. RDF Data store
2. Reactor
    a. Data de-normalizer
    b. View generator
3. Query Interface

Any external system can populate data in RDF Data store. For every such insert/update of data in the RDF Data store, a trigger begins a reaction. The reactor picks up the change and data de-normalizer component de-normalizes the data by breaking it into simple relational triples and further breaks up the data into subject, object and predicate. The view generator picks up this information and generates materialized views. These views are then persisted in MongoDB store for optimal retrieval.

The query interface is a light weight converter that lets clients query for information. It directly talks to MongoDB and converts the data back into RDF triples.

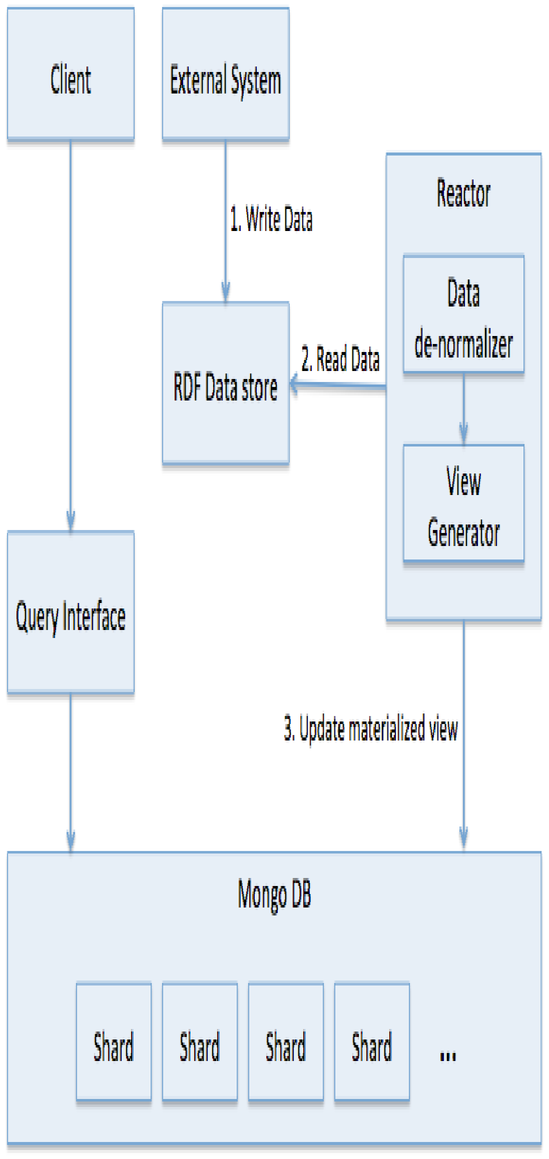


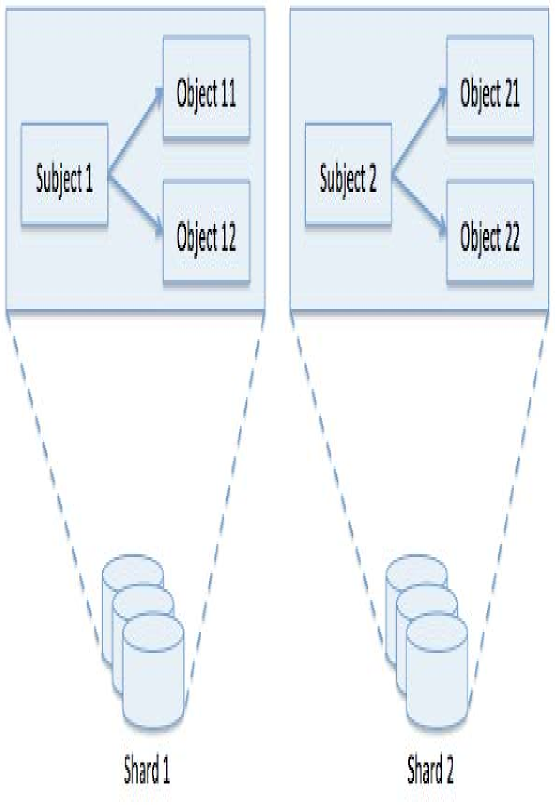
Figure 5:Architecture of MongoDB

4. Storage mechanism

Storage Definition 1. All data relating to a subject is persisted at one shard. This is to ensure that all queries related a subject can be answered in one call.

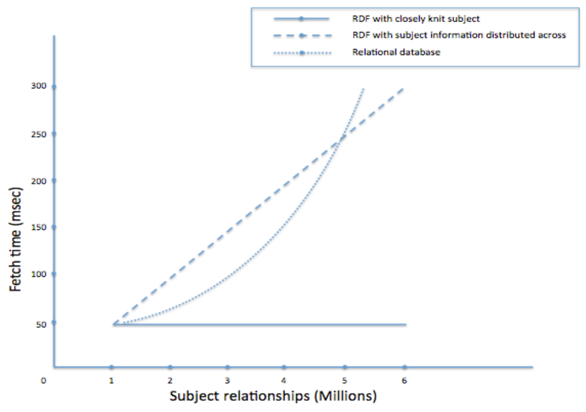
Storage Definition 2. Data for a subject is persisted as key-subkey-value triples with subject as key, predicate as subkey and object as value.

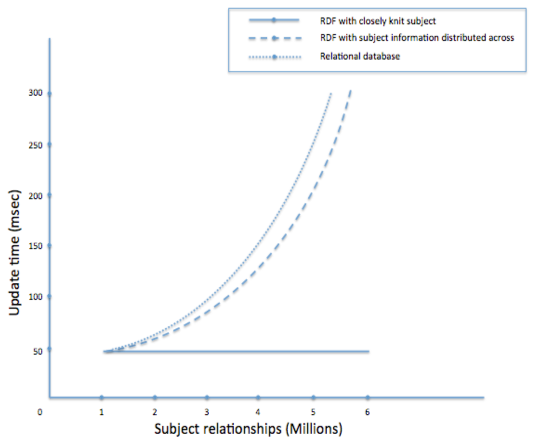
These storage definitions in combination with the power of RDF, provide a highly scalable, highly distributed and a highly available graph data store that could well form the basis for next generation storage architectures.



5. Experiment setup and results

We ran our experiments with a setup of 8 nodes.Each node is a Pentium IV 2.6 GHz processor with 2GB main memory, 320 GB disk space and operating system was RHEL 4.9. We compared the results of relational database with RDF store where information of a subject is distributed across and also with RDF store where all information of a subject is closely knit in one shard.





6. Conclusion

It is proven that Semantic repositories can replace relational databases while providing similar querying capabilities apart from providing advantages like modifying the structure/dimensions without any knowledge of the existing structure. However, it must be remembered that the information in such repositories is de-normalized and needs a lot more space as against a traditional database system. Indexes need to be built for faster querying that increase the size of the repository. Another way to maintain improve latencies is by ensuring that all the information about a subject is persisted closely. Such optimizations ensure that the advantages of traditional databases are not lost completely.

References

[1] "XML and Semantic Web W3C Standards Timeline". 2012-02-04
[2] http://www.w3.org/TR/PR-rdf-syntax/ "Resource Description Framework (RDF) Model and Syntax Specification"
[3] RDF Triple http://www.w3.org/TR/rdf-concepts/#section-triples
[4] URI http://tools.ietf.org/html/rfc3986
[5] Jim Rapoza (2 May 2006). "SPARQL Will Make the Web Shine". eWeek. Retrieved 2007-01-17
[6] SeRQL ww.openrdf.org/doc/sesame/users/ch06.html
[7] The MongoDB NoSQL Database Blog, The AGPL http://blog.mongodb.org/post/103832439/the-agpl
[8] DB-Engines Ranking - http://dbengines.com/en/ ranking

[9] "Sharding" on MongoDB Administrator's Manual - http://docs.mongodb.org/manual/sharding

[10]G. Boss, P. Malladi, D. Quan, L. Legregni, and H. Hall. Cloud computing. IBM White Paper, 2012, Retrieved 24 February2013 from http://download.boulder.ibm.com /pub/software/dw/wes/hipods/Cloud_computing_wp_final_8Oct.pdf.