An Approach for RDF Reconstruction and Query Processing

Anas Katib, Imrul Siddique, Suresh Yarra, Prudhvi Raj Atluri
Dr. Praveen Rao

UMKC School of Computing and Engineering

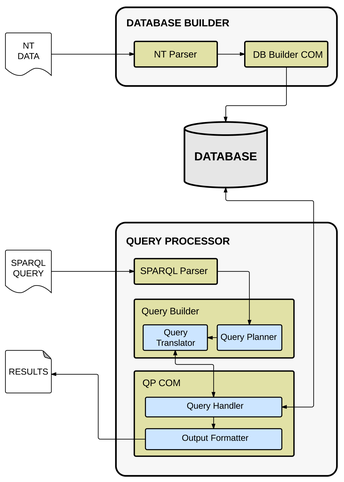
mongoDB

INTRODUCTION

- **Motivation**
  - Ever growing volume of data indicates we need better approaches to handle data.
  - Contemporary RDMS fail to scale with gigantic volume of data.
  - Distributive processing isn't seamless in RDMS.
  - New data storage model are being proposed beside RDMS; example: RDF, JSON, XML.
  - RDF is flexible with data-type/schema, allowing heterogeneous data to be encoded efficiently.
- **Challenges**
  - Finding the right data-model as well as implementation provider
  - RDF is a viable option to store large data as it is flexible with large scale data.
  - However there is no single best answer of how we should store and query RDF.
  - Several proposed approaches are :
    - Use RDMS, i.e. encode RDF into relational tables and query;
    - Devise new schema to store and query RDF (referred as native stores, e.g. Jena, RDF-3X)
    - Use No-SQL data-store (e.g. MongoDB, Hbase, Cassandra, Neo4G etc.)
- **Goal**
  - Model RDF storage and querying with **MongoDB** (No-SQL provider)
  - Compare the performance with benchmark tools (e.g. Jena, RDF-3X etc.)

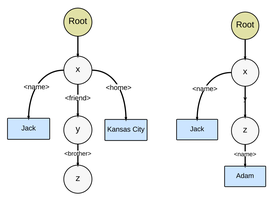
DESIGN

- Two main components
  - **Database builder**
    - responsible for converting RDF data into MongoDB documents.
    - Each RDF triple is stored as an independent document in MongoDB, because MongoDB will return the documents matched a query.
  - **Query Processor**
    - responsible for converting SPARQL queries into MongoDB queries and returning results.
    - We proposed a Data Guide(DG) graph, to find the optimal order such that triples that are independent should be processed beforehand.




```
SELECT ?x ?y ?z
WHERE {
?x <name> “Jack”.
?x <friend> ?y.
?y <brother> ?z.
?x <home> “Kansas City”.
}
```

```
SELECT ?x ?y ?z
WHERE {
?x ?y ?z.
?x <name> “Jack”.
?z <name> “Adam”.
}
```



IMPLEMENTATION

1. Database Builder: As mentioned in the Design each RDF triple stored as document in the MongoDB.

RDF	mongoDB
<sub1> <prop1> <obj1>.	{ subject: <sub1>, property: <prop1>, object: <obj1> }
<sub1> <prop2> “val”.	{ subject: <sub1>, property: <prop2>, object: “val” }
<sub2> <prop1> <obj1>.	{ subject: <sub2>, property: <prop1>, object: <obj1> }

- We have written a Java program which takes the RDF N-Triple file as input and generates the MongoDB documents and stores it into the configured database:

```
MongoClient mc=new MongoClient(Arrays.asList(new ServerAddress("localhost", 27017)));
DB db=mc.getDB("suresh");
Set<String> sc=db.getCollectionNames();
DBCollection c=db.getCollection("col");
BasicDBObject bob=new BasicDBObject("subject", q[0]) .append( "property",
q[1]).append("object", s); c.insert(bob);
```

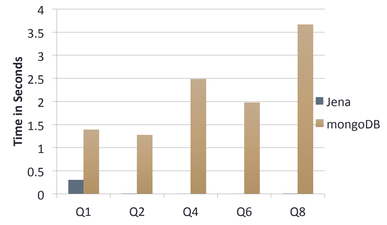
2. Query Processing:

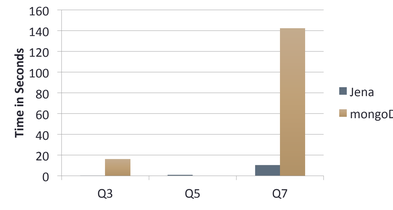
- We used Apache Jena ARQ to retrieve the output variables and triple patterns and forwards them to Query Planner.
- The task of Query Planner is to generate the Data Guide(DG) Graph.
- The query processor will traverse it, as an edge is being traversed its corresponding vertex-to-vertex relationship will be translated into MongoDB.
- Query processor execute the next relationship, which have 3 possible outcomes.
  a. no match; then delete the row form the dMat.
  b. exactly one match; then if the relationship contains a new variable insert a new column for that variable and insert its value in the current row, but do not create a column if there was no new variable.
  c. 1 < x matches; then replicate the row x-1 times and if the relationship contains a new variable insert a new column for the variable and insert its value in the current row and in the replicated rows, but do not create a column if there was no new variable.

Once all the edges have been processed by the query processor it returns the output variables.

EVALUATION

QID	SPARQL QUERY	Results
1	select ?a where { <http://uniprot.org/citations/7934828> <http://uniprot.org/author> ?a . }	17
2	select ?p ?o where { <http://purl.uniprot.org/uniprot/Q6GZX4> ?p ?o . }	29
4	select ?x ?z where { ?x <http://purl.uniprot.org/core/name> ?y . ?x <http://purl.uniprot.org/core/volume> ?z . ?x <http://purl.uniprot.org/core/pages> “176-186” . }	1
6	select ?x ?y where { ?x ?y “Israni S.” . <http://purl.uniprot.org/citations/15372022> ?y “Gomez M.” . }	48
8	select ?x ?z ?a where { ?x <http://purl.uniprot.org/core/reviewed> ?y . ?x <http://purl.uniprot.org/core/created> ?b . ?x <http://purl.uniprot.org/core/mnemonic> “003L_IIV3” . ?x <http://purl.uniprot.org/core/citation> ?z . ?z <http://purl.uniprot.org/core/author> ?a . }	8
3	select ?x ?y where { ?x <http://purl.uniprot.org/core/name> “Virology” . ?x <http://purl.uniprot.org/core/volume> ?y . }	25
5	select ?x ?y ?z where { ?x <http://purl.uniprot.org/core/name> “Science” . ?x <http://purl.uniprot.org/core/author> ?y . ?z <http://purl.uniprot.org/core/citation> ?x . }	53013
7	select ?a ?b where { ?x ?y <http://purl.uniprot.org/citations/15165820> . ?a ?b ?y . }	574692





EVALUATION SETUP

Metric	Measurement
Number of input triples	831,696
Input file size	109 MB
Database size	1.5 GB
RAM Limit	1 GB
Number of runs	2 per software
Number of queries	8

REFERENCES

1. Bornea, Mihaela A., et al. "Building an Efficient RDF Store Over a Relational Database".2013. 121-132. Print.
2. Apache Jena. The Apache Software Foundation, Web. 01 May 2014. <https://jena.apache.org/index.html>.
3. Weiss, Mark A. Florida International University. School of Computing and Information Sciences, Web. <http://users.cis.fiu.edu/~weiss/dsj2/code/Graph.java>.
4. "Map-Reduce." *MongoDB*. MongoDB, Inc., Web. 21 Feb. 2014. <http://docs.mongodb.org/manual/core/map-reduce/>.