UMKC

School of Computing
and Engineering

INTE

An Approach fo

RODUCTION

or RDF Reconstr

Anas Katib, Imrul Siddique, Su

Dr. Prav

IMPLEME

ruction and Que

uresh Yarra, Prudhvi Raj Atluri
een Rao

ENTATION

ery Processing

EVALUAT

mongoDB

ION

INT

Motivation

- Ever growing volume of data ir
- Contemporary RDMS fail to sca
- Distributive processing isn't sea
- New data storage model are be
- RDF is flexible with data-type/s
  efficiently.

Challenges

- Finding the right data-model as
- RDF is a viable option to store l
- However there is no single best
- Several proposed approaches
  - Use RDMS, i.e. encode RDF into re
  - Devise new schema to store and c
  - Use No-SQL data-store (e.g. Mong

Goal

- Model RDF storage and queryin
- Compare the performance with

RODUCTION

ndicates we need better approaches to handle data.

ale with gigantic volume of data.

amless in RDMS.

eing proposed beside RDMS; example: RDF, JSON, XML.

schema, allowing heterogeneous data to be encoded

s well as implementation provider

large data as it is flexible with large scale data.

t answer of how we should store and query RDF.

are :

elational tables and query;

query RDF (referred as native stores, e.g. Jena, RDF-3X)

goDB, Hbase, Cassandra, Neo4G etc.)

ng with **MongoDB** (No-SQL provider)

n benchmark tools (e.g. Jena, RDF-3X etc.)

IMPLEME

1. Database Builder: As mentioned in the De
the MongoDB.

RDF
<sub1> <prop1> <obj1>.
<sub1> <prop2> “val”.
<sub2> <prop1> <obj1>.

- We have written a Java program which ta
generates the MongoDB documents and s

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

ENTATION

esign each RDF triple stored as document in

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

kes the RDF N-Triple file as input and
stores it into the configured database:

```
List(new ServerAddress("localhost", 27017)));



ject", q[0]) .append( "property",
;
```

EVALUAT

QID	SPARQL QUERY	Results

1	select ?a
	where
	{
	<http://uniprot.org/citations/7934828> <http://uniprot.org/author> ?a .
	}
		17

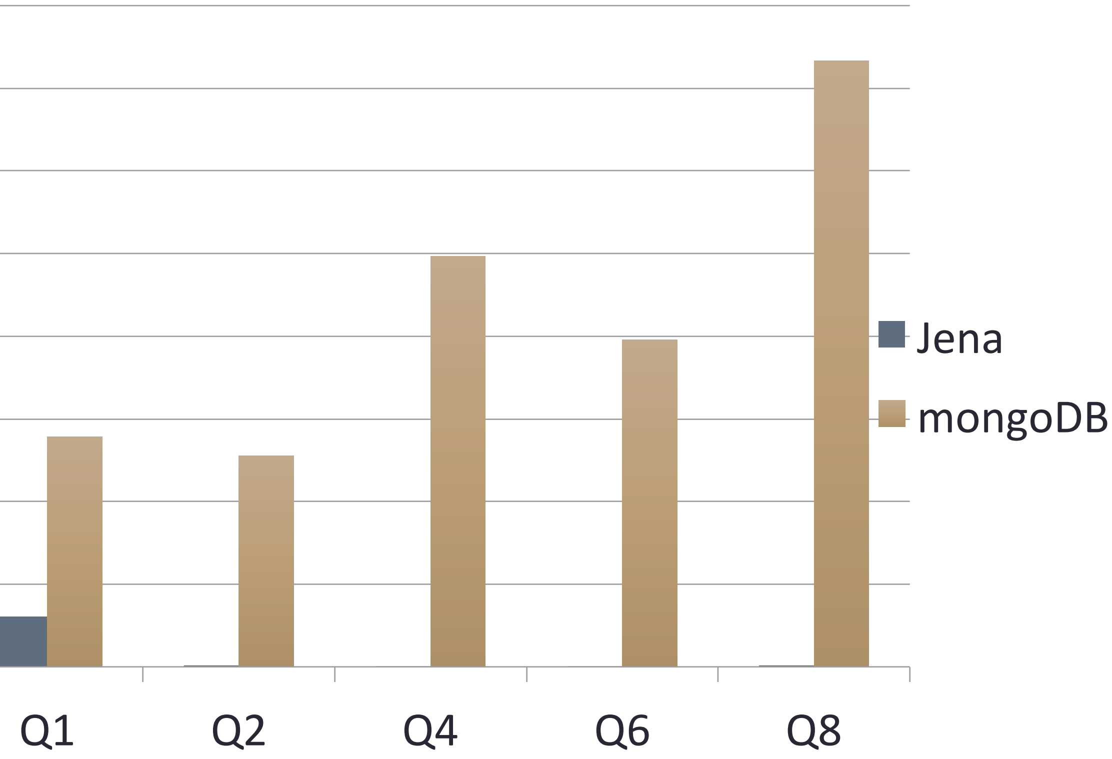
2	select ?p ?o
	where
	{
	<http://purl.uniprot.org/uniprot/Q6GZX4> ?p ?o .
	}
		29

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

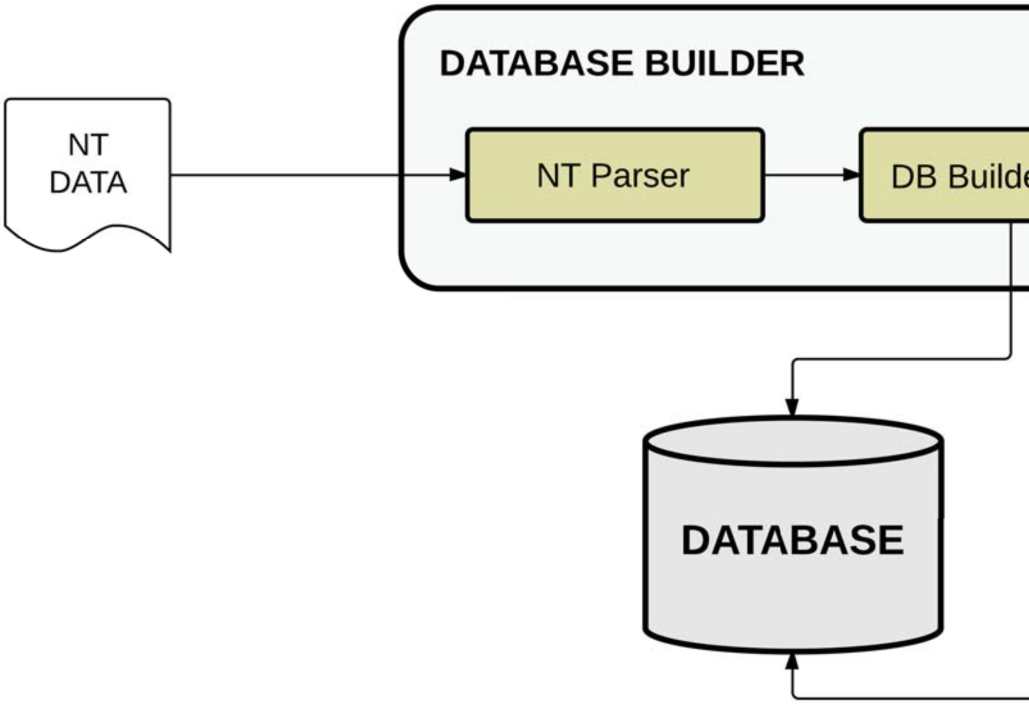
6	select ?x ?y
	where
	{
	?x ?y "Israni S." .
	<http://purl.uniprot.org/citations/15372022> ?y "Gomez M." .
	}
		48

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

ION



- Two main components
  - **Database builder**
    - responsible for converting R
    - Each RDF triple is stored as a will return the documents m
  - **Query Processor**
    - responsible for converting S
    - We proposed a Data Guide(are independent should be



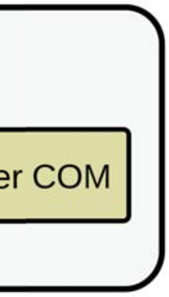
DESIGN

RDF data into MongoDB documents.

an independent document in MongoDB, because MongoDB natched a query.

SPARQL queries into MongoDB queries and returning results.

DG) graph, to find the optimal order such that triples that processed beforehand.



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

2. Query Processing:

- We used Apache Jena ARQ to retrieve the outp Query Planner.
- The task of Query Planner is to generate the Data
- The query processor will traverse it, as an edge relationship will be translated into MongoDB.
- Query processor execute the next relationship, w
  a. no match; then delete the row form the dMa
  b. exactly one match; then if the relationship co variable and insert its value in the current ro
  c. 1 < x matches; then replicate the row x-1 times an column for the variable and insert its value in the column if there was no new variable.

Once all the edges have been processed by the query

EVALUATI

Metric

ut variables and triple patterns and forwards them to

Guide(DG) Graph.

e is being traversed its corresponding vertex-to-vertex

hich have 3 possible outcomes.

t.

ntains a new variable insert a new column for that
w, but do not create a column if there was no new variable.

d if the relationship contains a new variable insert a new
current row and in the replicated rows, but do not create a

processor it returns the output variables.

ON SETUP

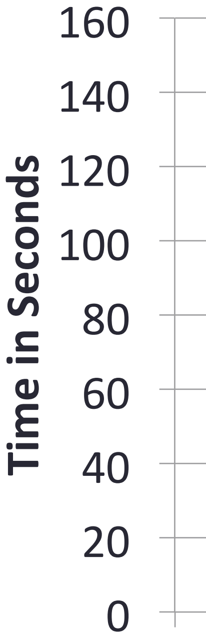
Measurement

QID	SPARQL QUERY	Results

3	select ?x ?y
	where
	{
	?x <http://purl.uniprot.org/core/name> "Virology" .
	?x <http://purl.uniprot.org/core/volume> ?y .
	}
		25

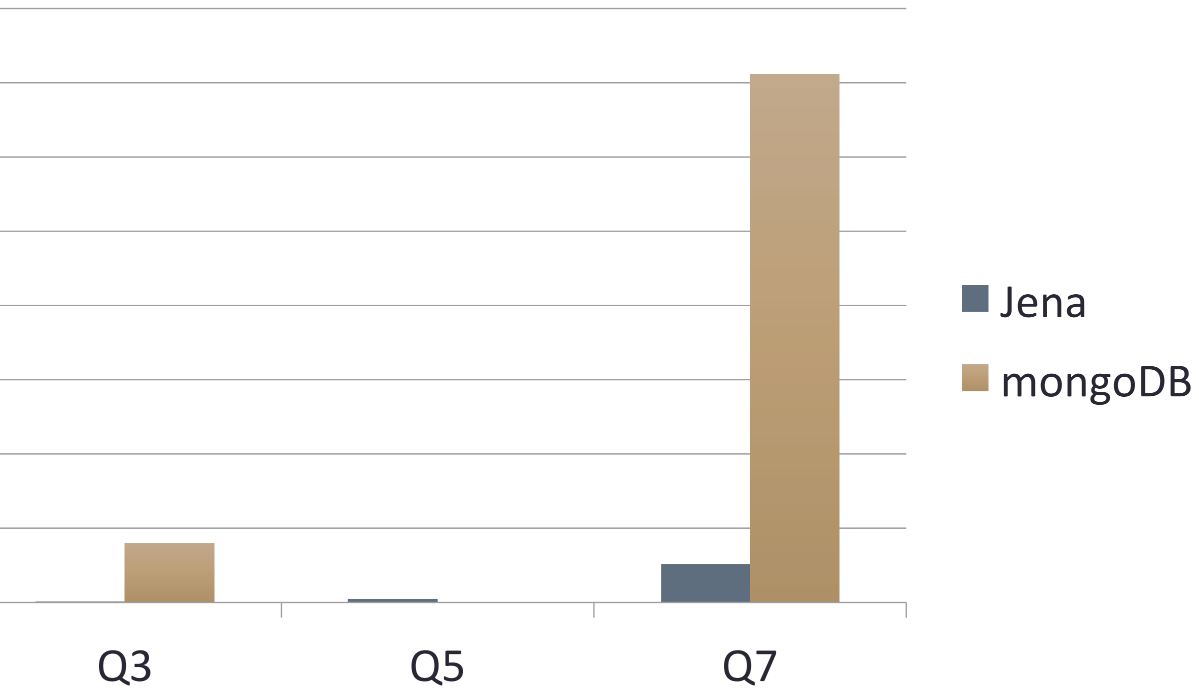
5	select ?x ?y ?z
	where
	{
	?x <http://purl.uniprot.org/core/name> "Science" .
	?x <http://purl.uniprot.org/core/author> ?y .
	?z <http://purl.uniprot.org/core/citation> ?x .
	}
		53013

7	select ?a ?b
	where
	{
	?x ?y <http://purl.uniprot.org/citations/15165820> .
	?a ?b ?y .
	}
		574692



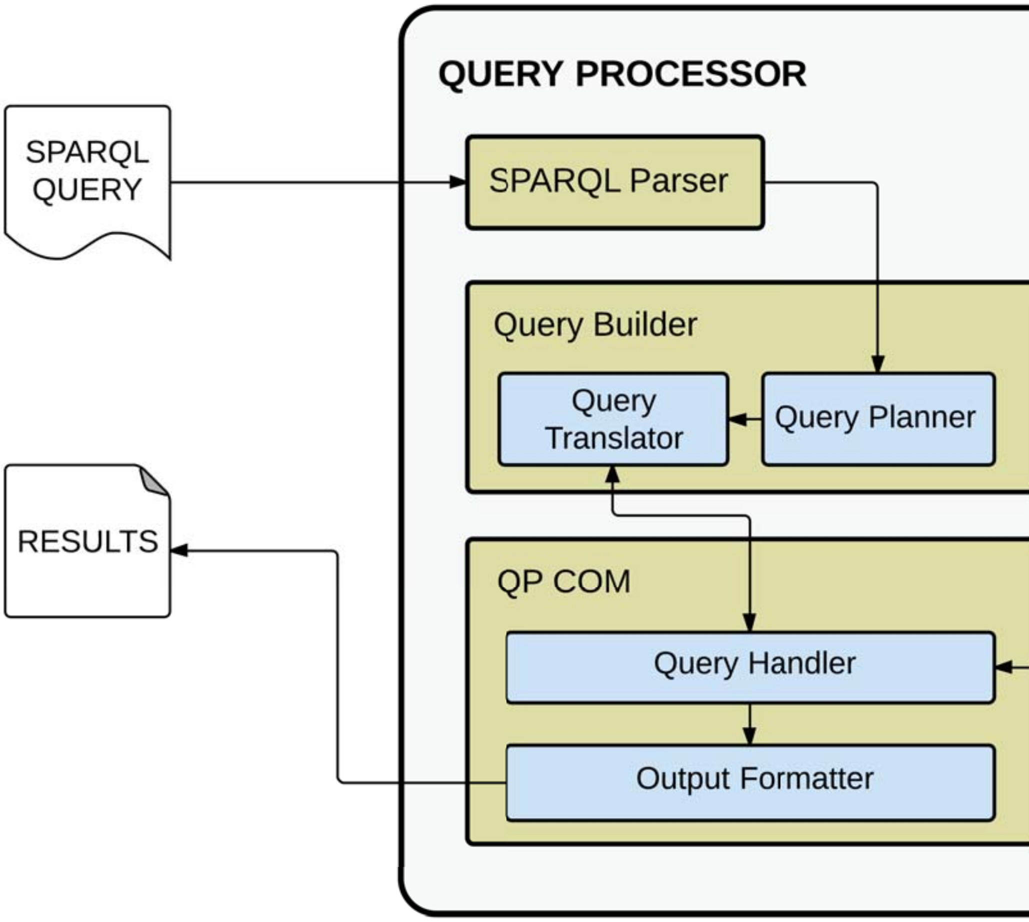
REFEREN

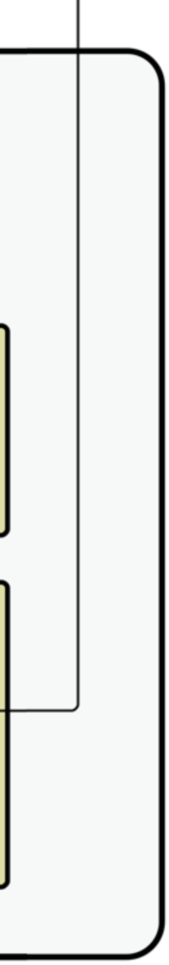
1. Bornea, Mihaela A., et al. "Building an Efficient
Database".2013. 121-132. Print.

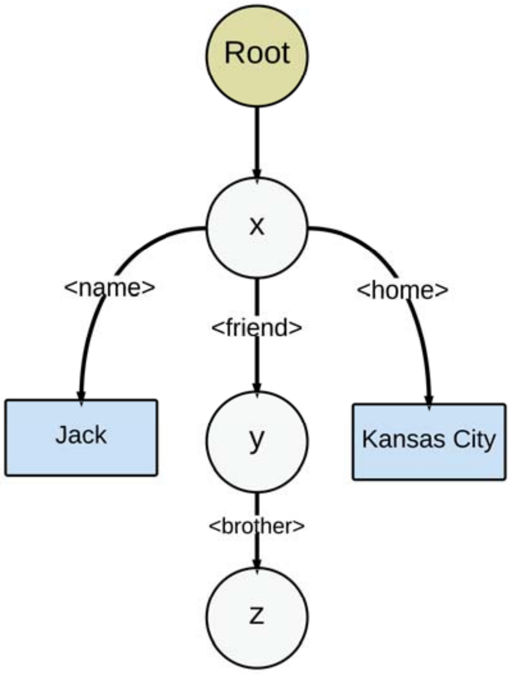


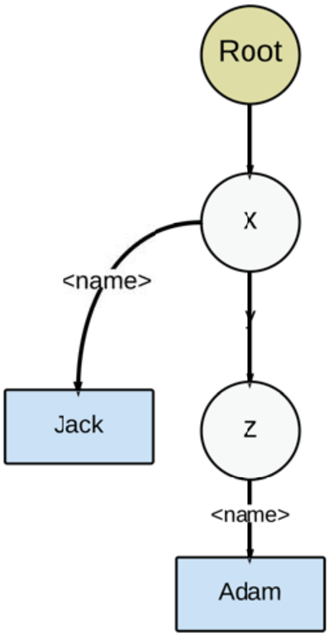
CES

RDF Store Over a Relational


QUERY PROCESSOR
SPARQL QUERY
SPARQL Parser
Query Builder
Query Translator
Query Planner
RESULTS
QP COM
Query Handler
Output Formatter




Root
x
<name>
<friend>
<home>
Jack
y
Kansas City
<brother>
z


Root
x
<name>
y
Jack
z
<name>
Adam

Number of input triples
Input file size
Database size
RAM Limit
Number of runs
Number of queries

831,696
109 MB
1.5 GB
1 GB
2 per software
8

2. Apache Jena. The Apache Software Foundation
jena.apache.org/index.html>.

3. Weiss, Mark A. Florida International University
Information Sciences, Web. <http://users.cis.fi
Graph.java>.

4. "Map-Reduce." *MongoDB*. MongoDB, Inc., We
docs.mongodb.org/manual/core/map-reduce/

n, Web. 01 May 2014. <https://

y. School of Computing and
u.edu/~weiss/dsj2/code/

b. 21 Feb. 2014. <http://
/>.