RDF Triple Stores and Indexation

CrEDIBLE thematic working days
October 2-4, 2013)

Olivier Curé

UPEM , LIGM UMR CNRS 8049, France

October 4, 2013

Observations

- RDF is a logical data model and does not propose a physical storage approach.
- Many physical storage solutions have been proposed, mainly based on existing database management systems, e.g., RDBMS.
- But an RDF data management systems (RDFMS) also has to handle inferences related to some entailment regimes, e.g., RDFS.
- It is still an open problem to define an efficient, robust, scalable RDFMS.

Overview

- RDBMS-based solutions
- NoSQL-based solutions
- Trishaped project
- Conclusion

RDBMS-based solutions

- Many solutions are based on a RDBMS (Jena, Sesame, 3Store, Redland, swStore, etc.)
- They benefit from the more than 30 years of RDBMS history.
- SPARQL queries are translated into SQL queries and benefit from RDBMS query optimization

Triple table store : One big table with three columns to store all triples.

subject	Property	Object

Clustered Property table: clusters of properties that tend to appear together in one table.

Property table

subject	Prop1.	Prop.2	...

Left-over table

subject	Property	Object

Property-class table : Create one table for each subject type.

subject	Property	Object

Vertical partitioning: one table 2-column (subject-object) for each property.

Property: ..

subject	Property

DB2RDF : One table with a column subject, and a set of (property/object) plus a spill (for overflow) column. Plus one table for multivalued properties.

Direct Primary Hash

subject	spill	prop1	obj1	prop2	obj2	...
			id1			

Direct Secondary Hash

id	object
id1	..
id1	..

- In an OBDA context, a legacy RDBMS instance is mapped to ontology (OWLQL)
- The mapping is used to translate SPARQL to SQL queries
- A dedicated reasoner ensures to return all answers to a given query, e.g., Quest [1].

[1] M. Rodriguez-Muro and D. Calvanese, High Performance Query Answering over DL-Lite Ontologies, KR12

Limits of the RDBMS approach

- Distribution of data on several machines: manual 'sharding', latencies due to replications in an ACID context.
- Parallelization of some processing: hard to integrate MapReduce like approaches $\rightarrow$ Hadapt, Oracle and Cloudera, Vertica.
- Index maintenance when multiple indexes are being used.

Distributed approaches

- [2] distributes triples over a cluster of machines using a graph partitioning approach.
- Each machine stores its own subgraph on an RDF-3X instance.
- Subgraphs share some RDF triples to prevent high inter-machine communication.
- Some queries are being executed in parallel using the MapReduce framework.
- Other distributed approaches (mainly hash-based): YARS2, SHARD, Virtuoso

[2] Huand et al: Scalable SPARQL Querying of Large RDF Graphs. VLDB 2011

NoSQL presentation

- Not Only SQL, a set of database systems addressing the following issues:
  - data deluge
  - heterogeneity of structures of stored documents.
  - data connectivity
  - architecture of applications using these data.

NoSQL ecosystem

Key Value stores

Redis riak MEMBASE.ORG

Voldemort
Dynomite
TokyoCabinet
Scalaris
Amazon Dynamo & SimpleDB

Graph databases

Neo4j the graph database
InfiniteGraph
HyperGraphDB
AllegroGraph
INFOGRID

Document databases

Terrastore
OrientDB
CouchDB
mongoDB
RAVEN DB

Column family databases

Google BigTable
HYPERTABLE
Cassandra
HBASE

- [3] adapts the Hexastore (6 indexes) approach to HBase.
- Rya is an implementation on Accumulo (Column store family).
- Amada is designed on Cassandra and supports RDF storage on the cloud, i.e., AWS
- Trinity.RDF (Microsoft) uses the key/value approach of Trinity to store RDF triples.
- Oracle on its Oracle NoSQL database.
- CumulusRDF with Cassandra.
- Also some approaches using CouchDB and MongoDB.

[3] Jianling Sun, Qiang Jin, Scalable RDF Store Based on HBase and MapReduce

- TRIple Store HAute PErformance Distribué
- Like HDT (Head Dictionary Triples)[4], Trishaped aims to compress the data (using Succinct Data Structures) limits I/O by storing the structures in RAM.
- Our contributions :
  - clever encoding of dictionaries to support RDFS entailment regime
  - even more compression
  - propose algorithms for query processing and optimization

[4] Miguel A. Martnez-Prieto, Mario Arias Gallego, Javier D. Fernndez: Exchange and Consumption of Huge RDF Data. ESWC 2012

Dictionary encoding

- Most systems encode (usually with integers) the URIs, blank nodes and literals found in triples.
- A standard approach is to create 2 structures (e.g., hash maps)
  - one from identifier to value: to translate a query answer
  - one from value to identifier: to encode the triples and the SPARQL query,
- different forms of compression can be used to reduce the size of the structures (e.g., delta compression).
- In Trishaped, we use a special encoding for ontology concepts and properties.

(a) Concept hierarchy encoding

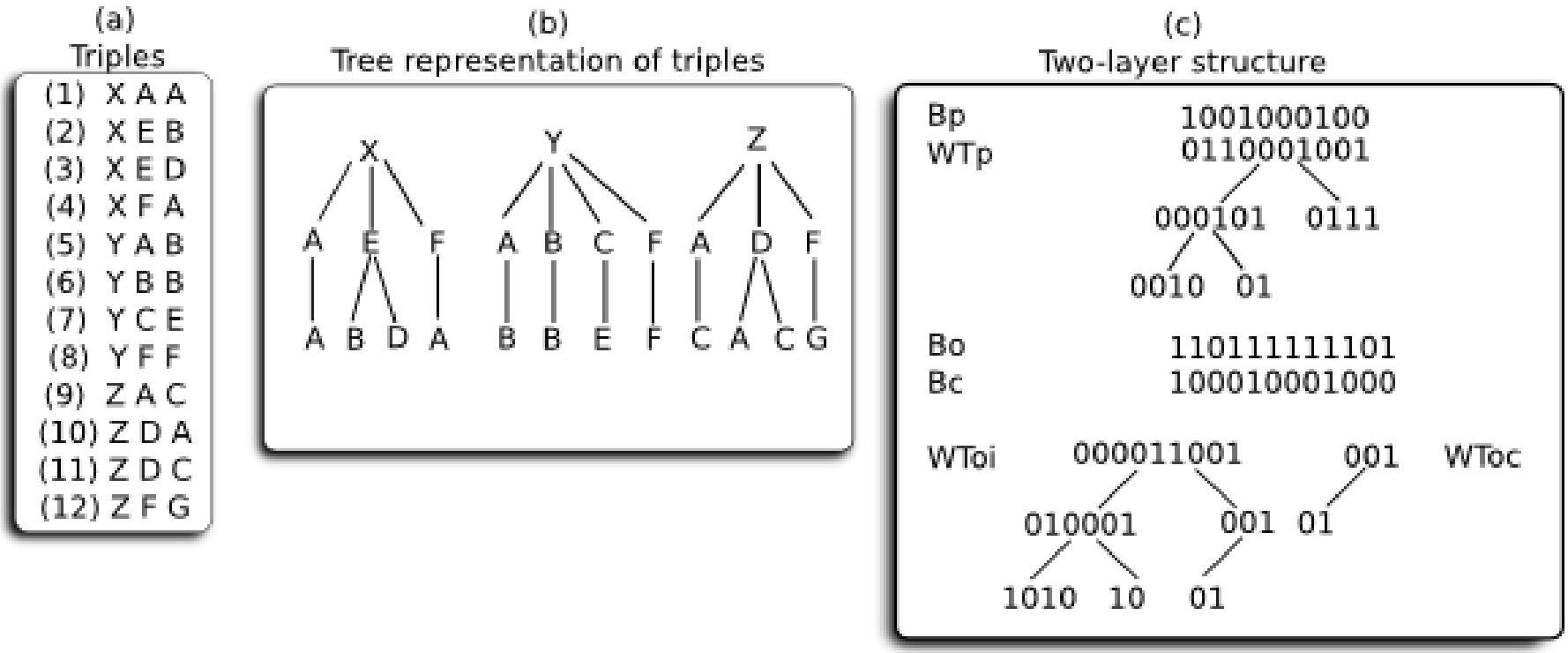
```
00 Organization
    000 self
    001 Department
    010 Institue
    100 Program
    101 ResearchGroup
    110 University
01 Person
    000 self
    001 Director
    010 Employee
       00 self
       01 AdministrativeStaff
          00 self
          01 ClericalStaff
          10 SystemsStaff
       10 Faculty
          00 self
          01 Lecturer
          10 PostDoc
          11 Professor
            000 self
            001 AssistantProfessor
            010 AssociateProfessor
            011 Chair
            100 Dean
            101 FullProfessor
            110 VisitingProfessor
...
10 Publication
...
```

(b) Property hierarchy encoding

```
00 rdf:type
01 datatype property
  000 age
  001 emailAdress
  010 name
  011 officeNumber
  100 researchInterest
  101 telephone
  110 title
10 Object Property
  00000 advisor
  00001 affiliateOf
  00010 affiliateOrganizationOf
  00011 degreeFrom
        00 self
        01 doctoralDegreeFrom
        10 mastersDegreeFrom
        11 undergraduateDegreeFrom
  00100 hasAlumnus
  00101 listedCourse
  00110 member
  00111 memberOf
        0 self
        1 worksFor
          0 self
          1 headOf
...
```

Indexing

- Efficiency of query processing is largely due to the existence of indexes.
- Many solutions are using a multiple indexing approach (Hexastore, Kowari, Virtuoso, RDF-3X, YARS, etc.)
- Hexastore uses 6 triple indexes : pso, pos, spo, sop, ops et osp; 6 binary indexes (sp,so,ps,po,os,op) et 3 unaries (s,p,o)
- Trishaped uses a self-index approach, i.e., one can seek and retrieve any portion of the data without accessing the original data.


(a)
Triples
(1) X A A
(2) X E B
(3) X E D
(4) X F A
(5) Y A B
(6) Y B B
(7) Y C E
(8) Y F F
(9) Z A C
(10) Z D A
(11) Z D C
(12) Z F G
(b)
Tree representation of triples
X
Y
Z
A E F
A B C F
A D F
A B D A
B B E F
C A C G
(c)
Two-layer structure
Bp
WTp
1001000100
0110001001
000101 0111
0010 01
Bo
Bc
110111111101
100010001000
WTOi
000011001
001
WTOc
010001
001 01
1010 10
01

Two-layer structure

- Uses two forms of SDS: Bitmaps and Wavelet trees
- Operations on SDS:
  - $access(i)$: returns the elements stored at the i^{th} position.
  - $rank_a(i)$: returns the number of occurences of i a in the binary substring ranging from position 0 to i.
  - $select_a(i)$: returns the position of the i^{th} position of a.
- Wavelet trees enable to encode large alphabet.

Wavelet tree encoding example

- 6 properties so 3 bits are required.
- A: 000, B: 001, C:010, D: 011, E: 100, F:110
- Property sequence is AEFABCFADF

Wavelet tree encoding example

- 6 properties so 3 bits are required.
- A: 000, B: 001, C:010, D: 011, E: 100, F:101
- Property sequence is AEFABCFADF
- We create a first sequence with the first bit of each character: 0110001001

Wavelet tree encoding example

- 6 properties so 3 bits are required.
- A: 000, B: 001, C:010, D: 011, E: 100, F:101
- Property sequence is AEFABCFADF
- We create a first sequence with the first bit of each character: 0110001001
- For the second sequence we consider 2 alphabets {ABCD} and {EF} where A, B and E are 0 and the rest are 1.
- 0001010111. Note that since we have 6 '0' in the first layer we can identify the boundary at the 2nd sequence.
- and so one the property layer, and the second layer.

- All SPARQL queries can be translated with a set of rank, select and access operations.
- We have designed a query optimization approach based on the cost of these operations and some data set statistics.
- Dictionaries are used both the query encoding and answer set decoding.

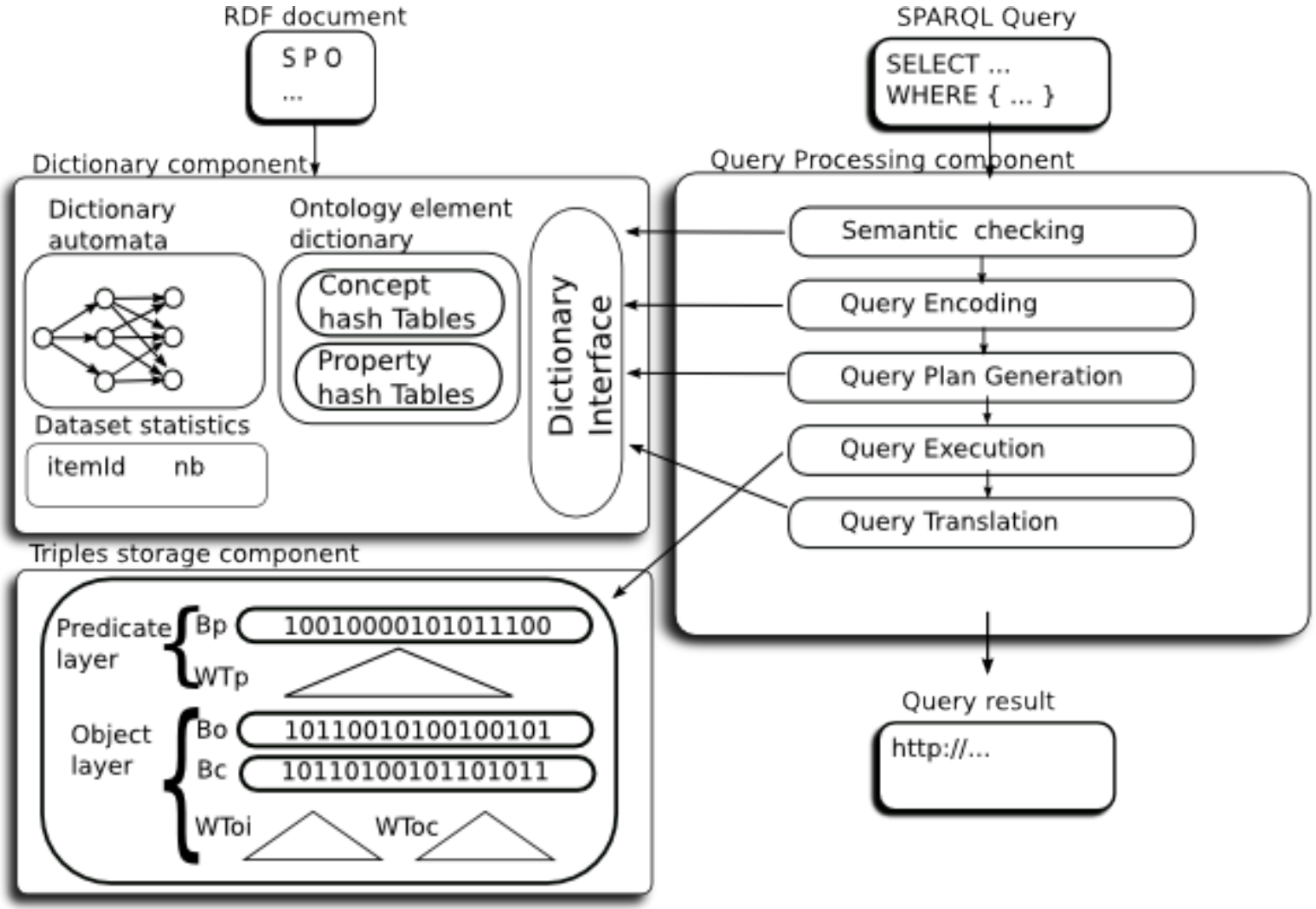

RDF document
S P O
...
SPARQL Query
SELECT ...
WHERE { ... }
Dictionary component
Dictionary automata
Ontology element dictionary
Concept hash Tables
Property hash Tables
Dataset statistics
itemId nb
Dictionary Interface
Query Processing component
Semantic checking
Query Encoding
Query Plan Generation
Query Execution
Query Translation
Triples storage component
Predicate layer
Bp
WTp
10010000101011100
Object layer
Bo
Bc
WToi
WToc
10110010100100101
10110100101101011
Query result
http://...

Table: Description of the datasets

Dataset	Triples (Million)	Size (MB)
LUBM100	13.4	1125
LUBM1000	133.5	11307
Yago2	37.5	5325

Table: Size of database serialization (MB) and Time to prepare datasets

	Size in MB			Time in sec		
	u100	u1000	Yago	u100	u1000	Yago
RDF3X	831,717	7,795,458	2,189,735	240	3050	1090
WF M1	73,231	737,685	217,293	168	2134	768
WF M2	56,851	576,317	168,445	119	1515	545

Table: Inference-based query answering times (sec) on univ100

	QR#4	QR#5	QR#6	QR#7	QR#10
RDF3X	4.2	2.5	15.3	1.4	1.6
WF M2	2.66	1.3	13.1	1.2	1.4

Conclusion

- Many issues to be solved:
  - efficient distribution of triples and processing
  - parallelism in computing the dictionaries, query processing
  - Updates at the schema and data levels.
- NoSQL systems are attractive but RDBMS are adapting rapidly, i.e., NewSQL.
- Motivation to adopt a solution must be based on our usage.
- More invesgations are needed on indexing (e.g. 'index cracking'), taking into account the ontology in many aspects (e.g., partitioning), **polyglot persistance**, etc.

- Thank you for your attention.
- Questions ?