WaterFowl: a Compact, Self-indexed and Inference-enabled immutable RDF Store

ESWC 2014

Olivier Curé[a], Guillaume Blin[a], Dominique Revuz[a], David Faye[b]

[a]: UPEM, LIGM UMR CNRS 8049, France
[b]: Univ. Gaston Berger de Saint Louis, LANI, sénégal

Wednesday, May 28th, 2014

Outline

- Motivation
- WaterFowl system
    - Presentation
    - Architecture
    - Two-layer storage approach
    - Dictionary encoding
    - Query processing
    - Inference
- Evaluation
- Conclusion

Motivation

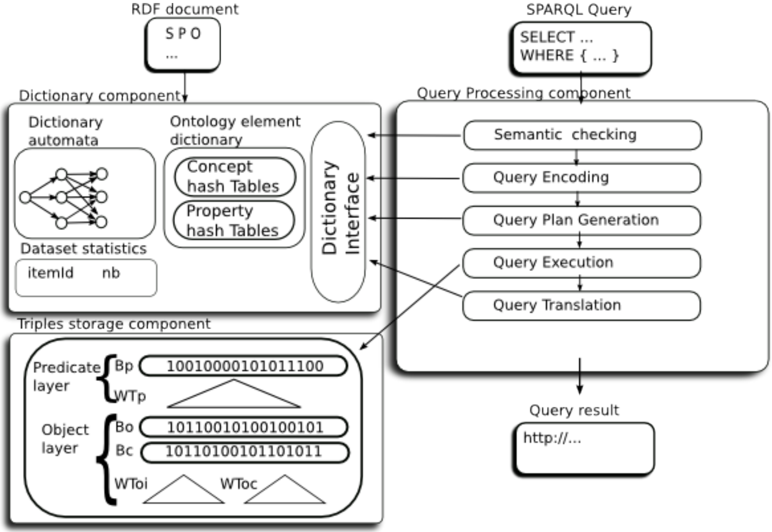
- Facing increasing volumes of RDF triples
- The management issues (load balancing, SPOF) are usually addressed by distributing the workload (Virtuoso, OWLIM, bigdata, etc.)
- Requires a lot of work on replication.
- Few systems address high level compression, e.g., BitMat[1], TripleBit[2]
- There is a need for highly compressed and distributed RDF Stores

[1]Medha Atre et al (2010).Matrix "Bit" loaded: a scalable lightweight join query processor for RDF data, WWW, pp 41-50

[2]Yuan, P.et al (2013). Triplebit: a fast and compact system for large scale rdf data. PVLDB,6(7):517528

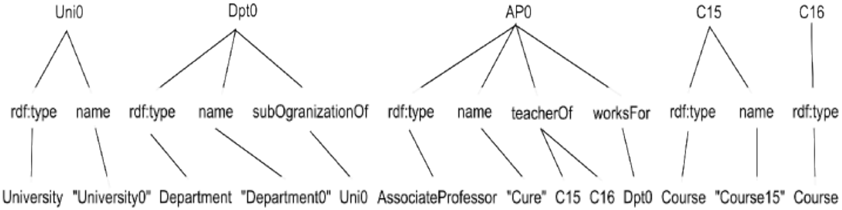
- Like HDT (Head Dictionary Triples)[3], WaterFowl aims at compressing the data using Succinct Data Structures (SDS), limits I/O by accessing the structures in directly in RAM.
- Our contributions :
  - even more compression (using SDS only) and experimenting with different implementations : compact, unique self-indexing
  - propose algorithms for query processing and optimization
  - clever encoding of dictionaries to support RDFS entailment regime

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


RDF document
S P O
...
SPARQL Query
SELECT ...
WHERE { ... }
Dictionary component
Dictionary automata
Ontology element dictionary
Concept hash Tables
Property hash Tables
Dictionary Interface
Dataset statistics
itemId nb
Query Processing component
Semantic checking
Query Encoding
Query Plan Generation
Query Execution
Query Translation
Triples storage component
Predicate layer
Bp
100100001010111100
WTp
Object layer
Bo
101100101001001 01
Bc
101101001011010 11
WToi
WToc
Query result
http://...

Two-layer storage approach

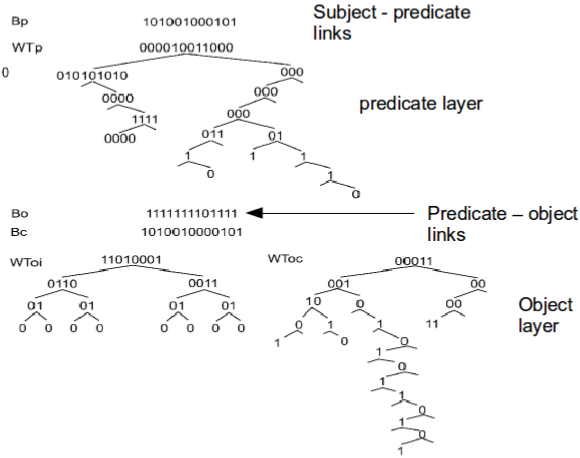
- Uses two forms of SDS: Bitmaps and Wavelet trees
- Wavelet trees are binary balanced trees and enable to encode large alphabet.
- SDS are known for using an amount of space that is close to the information-theoretic lower bound.
- Operations on SDS:
  - $access(i)$: returns the elements stored at the i^{th} position.
  - $rank_a(i)$: returns the number of occurences of a in the binary substring ranging from position 0 to i.
  - $select_a(i)$: returns the position of the i^{th} occurence of a.

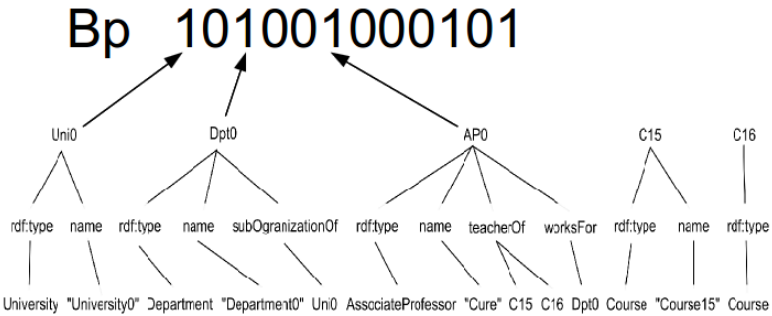

Uni0
Dpt0
AP0
C15
C16
rdf:type
name
rdf:type
name
subOgranizationOf
rdf:type
name
teacherOf
worksFor
rdf:type
name
rdf:type
University
"University0"
Department
"Department0"
Uni0
AssociateProfessor
"Cure"
C15
C16
Dpt0
Course
"Course15"
Course

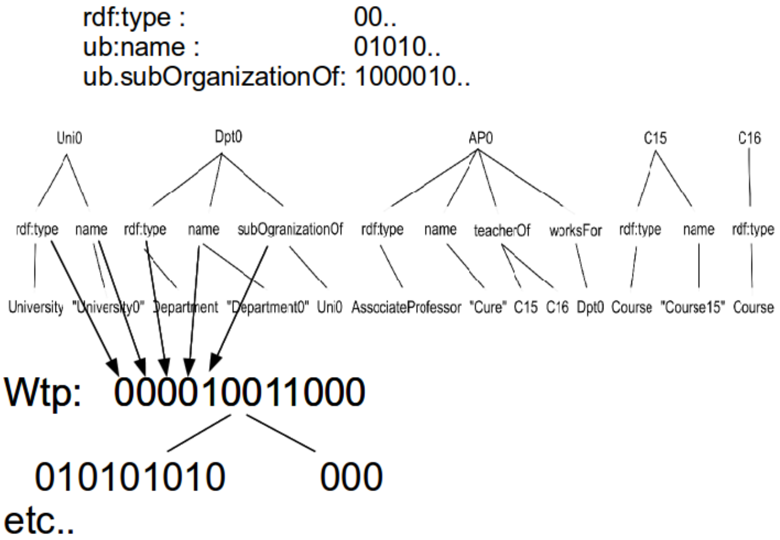
Concept Encoding
ub:University = 00 110
ub:Department = 00 001
ub:AssociateProfessor = 01 010 10 11 010
ub:Course = 10 01

Property Encoding
rdf:type = 00
ub:name = 01 010
ub:subOrganizationOf = 10 00010
ub:teacherOf = 10 00101
ub:worksFor = 10 00111 1 0

Encoding of the rest
Uni0 = 0000
Dp0 = 0010
AP0 = 0011
C15 = 0100
C16 = 0110
"University0" = 1000
"Department0" = 1010
"Cure" = 1100
"Course15" = 1110



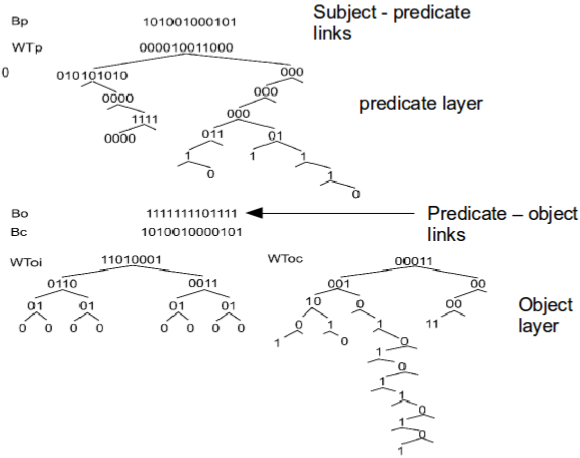

Bp 101001000101
Uni0
Dpt0
AP0
C15
C16
rdf:type
name
rdf:type
name
subOgranizationOf
rdf:type
name
teacherOf
worksFor
rdf:type
name
rdf:type
University
"University0"
Department
"Department0"
Uni0
AssociateProfessor
"Cure"
C15
C16
Dpt0
Course
"Course15"
Course


rdf:type : 00..
ub:name : 01010..
ub.subOrganizationOf: 1000010..
Uni0
Dpt0
AP0
C15
C16
rdf:type
name
rdf:type
name
subOgranizationOf
rdf:type
name
teacherOf
worksFor
rdf:type
name
rdf:type
University
"University0"
Department
"Department0"
Uni0
AssociateProfessor
"Cure"
C15
C16
Dpt0
Course
"Course15"
Course
Wtp: 000010011000
010101010
000
etc..

Concept Encoding
ub:University = 00 110
ub:Department = 00 001
ub:AssociateProfessor = 01 010 10 11 010
ub:Course = 10 01

Property Encoding
rdf:type = 00
ub:name = 01 010
ub:subOrganizationOf = 10 00010
ub:teacherOf = 10 00101
ub:worksFor = 10 00111 1 0

Encoding of the rest
Uni0 = 0000
Dp0 = 0010
AP0 = 0011
C15 = 0100
C16 = 0110
"University0" = 1000
"Department0" = 1010
"Cure" = 1100
"Course15" = 1110



Dictionary encoding

- Distinction between ABox and TBox
- Automata-based data structure for the ABox
- Hash table-based for the TBox
  - two structures: one for the concept another one for the properties
  - key of the hash tables are binary strings representing the subsumption relationships
- Hence, all entries entries of a hierarchy have the same prefix.

(a) Concept hierarchy encoding

- 00 Organization
  - 000 self
  - 001 Department
  - 010 Institue
  - 011 Program
  - 100 ResearchGroup
  - 101 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 Work
  - 00 self
  - 01 Course
- ...

(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 subOrganizationOf
  - 00011 degreeFrom
    - 00 self
    - 01 doctoralDegreeFrom
    - 10 mastersDegreeFrom
    - 11 undergraduateDegreeFrom
  - 00100 hasAlumnus
  - 00101 teacherOf
  - 00110 member
  - 00111 memberOf
    - 0 self
    - 1 worksFor
      - 0 self
      - 1 headOf

- SPARQL queries can be translated with a set of rank, select and access operations.
- We have designed a query optimization approach based on the:
  - identification of the cost of Basic Graph Pattern (BGP) in terms of access, rank and select
  - storing basic statistics about triple element distribution
  - design of heuristics based on these costs and statistics

- `subClassOf` and `subPropertyOf` inferences are performed by using prefix versions of rank and select operations.
- We support multiple inheritance using a mapping table storing the different identifiers of a concept.
- For the `domain` and `range` inferences:
    - adopt a materialization approach (types of objects and subjects)
    - reduce of size of the materialization but storing the most general concept

Table 1. Size of database serialization (MB) and Time to prepare datasets

	Size in MB			Time in sec		
	univ100	univ1000	Yago	univ100	univ1000	Yago
RDF-3X	831,717	7,795,458	2,189,735	240	3050	1090
BigOWLIM	2,411,260	22,600,088	6,348,338	838	10640	3708
Jena TDB	1,492,057	13,984,467	3,928,271	1285	16332	5837
WaterFowl Mode 1	91,539	922,106	271,616	168	2134	768
WaterFowl Mode 2	71,064	720,396	210,556	119	1515	545
WaterFowl Mode 3	77,351	798,829	203,728	107	1488	513

Table 2. Query answering times (sec) on univ1000

	LUBM QR#1	LUBM QR#2	LUBM QR#14
RDF-3X	1.65	14.88	1640
BigOWLIM	138	5.7	3320
Jena TDB	3.52	2.18	2998
WaterFowl Mode 2	1.80	10.18	1710
WaterFowl Mode 3	1.75	10.13	1680

Table 3. Inference-based query answering times (sec) on univ100

	QR#4	QR#5	QR#6	QR#7	QR#10
RDF-3X	4.2	2.5	15.3	1.4	1.6
OWLIM-SE	705	16771	72	1708	3.65
Jena TDB	4.85	6.3	30.7	207	1.55
WaterFowl Mode 2	3.66	2.3	13.4	1.2	1.4

Conclusion

- On-going work on:
  - efficient distribution of triples and processing
  - parallelism in computing the dictionaries, query processing
  - Updates at the schema and data levels.
- Extending reasoning capabilities to OWL profiles

- Thank you.
- Questions ?