Merging RDF Characteristic Sets to Optimize SPARQL Queries

Marios Meimaris and George Papastefanatos

INSTITUTE for the MANAGEMENT of INFORMATION SYSTEMS

Preliminaries

- RDF (Resource Description Framework)
  - Abstract Data model for Linked Data
  - Based on *Triples*: Subject-Predicate-Object
  - RDF datasets are *Directed Labelled Graphs*
- Characteristic Set (CS)
  - A CS is a set of properties with the same subject as source node
  - An RDF dataset can be described as a set of unique CSs
  - Each CS is an *implicit resource type*

ATHENA INSTITUTE for the MANAGEMENT of INFORMATION SYSTEMS

Motivation

Web of Data

› Large Volumes
› Schema Diversity
› Loose Structure
› Complex analysis needs

*The need for fast and efficient indexing and **complex** query processing methods on voluminous and diverse data arises…*

State of the Art

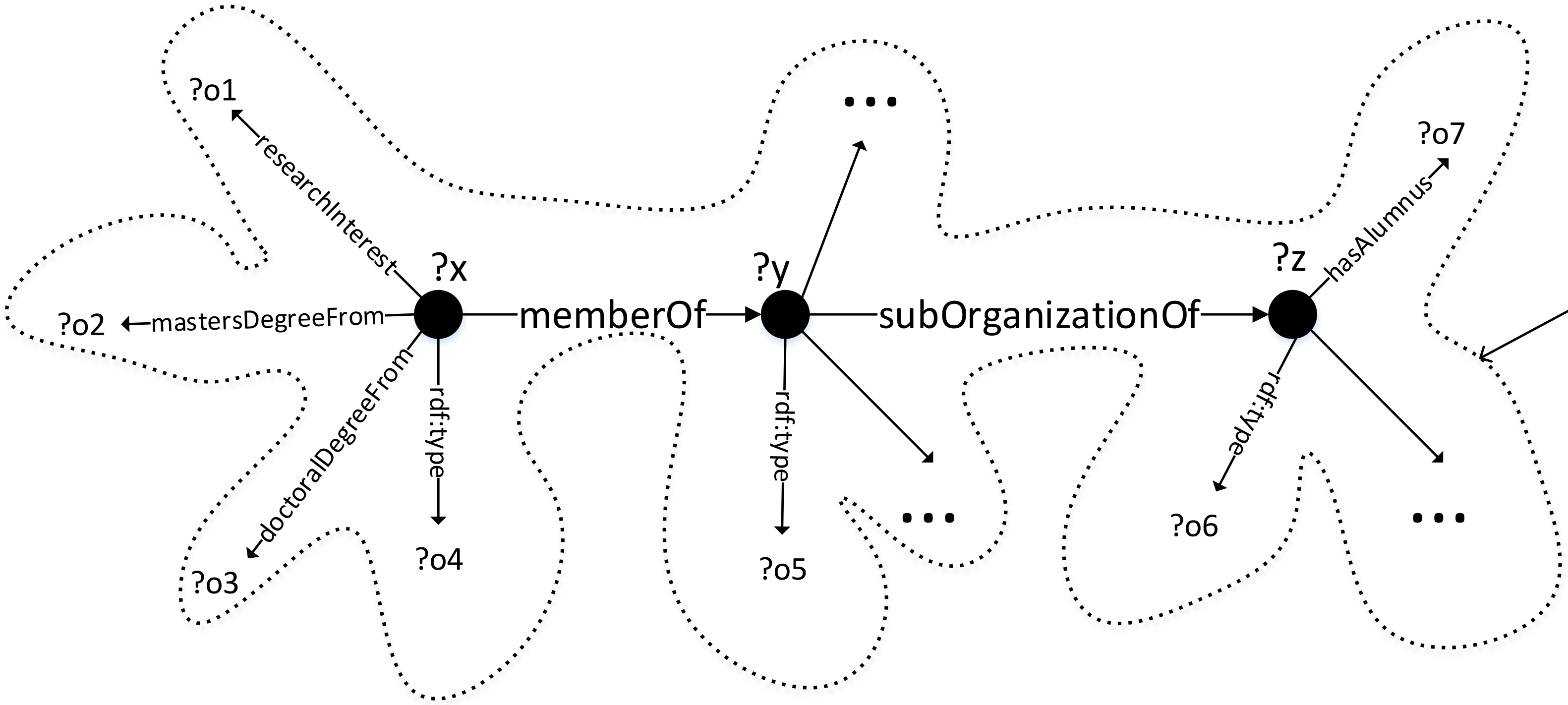
- Well-established approaches:
  - Exhaustive permutation indexing
    - (SPO, PSO, OPS, etc.)
    - Partial Permutated Indexes (SO, OS etc.)
    - Facilitation of merge joins (where possible)
  - Property tables
    - Each node type becomes a relational table
    - Fast grouping and retrieval by concept type
  - Vertical partitioning
    - Each property becomes a relational table
    - Very simple design

Challenge

We target efficient SPARQL query processing:

- Query Characteristics:
  - long chain patterns (object-subject joins)
  - descriptive star patterns (subject-subject joins)
- Data Characteristics:
  - large volume
  - semi-structured (loosely-defined schema)

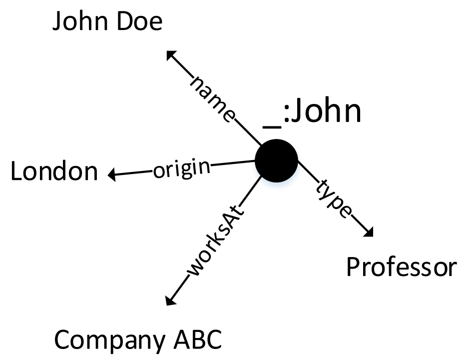
Challenge

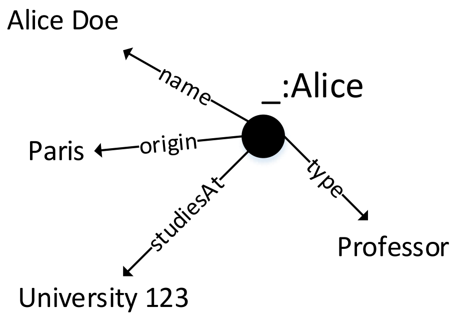



```
SELECT DISTINCT * WHERE {
        ?x lubm:researchInterest ?o1 .
        ?x lubm:mastersDegreeFrom ?o2 .
        ?x lubm:doctoralDegreeFrom ?o3 .
        ?x lubm:memberOf  ?y .
        ?x rdf:type ?o4 .
        ?y rdf:type ?o5 .
        ?y lubm:subOrganizationOf ?z .
        ?z rdf:type ?o6  .
        ?z lubm:hasAlumnus ?o7
}
```

Approach

- Use Characteristic Sets (CSs) and their links in order to store and index triples
- ***Characteristic Sets*** (Neumann & Moerkotte, ICDE 2011)
  - A *Characteristic Set* (CS) S_c of a node x is defined as the set of properties emitting from x (i.e., x as subject)





S_c(John) = {name, origin, worksAt, type}

S_c(Alice) = {name, origin, studiesAt, type}

Approach

- Derive a relational representation of an RDF dataset
- Use CSs as tables and links between CSs as relationships
  - SS joins (star patterns) are very easy to compute
  - SO / OS joins (chain patterns) become simple semi-joins between tables
  - Combined SS star/chain patterns are answered efficiently
- Problems
  - many CSs with small numbers of triples
  - few CSs with large numbers of triples
  - huge schema overlaps between them

Observations

- Based on previous findings:
  - CS number is generally low but exhibits skewed distribution
    - E.g., many CSs with very few (<10) subjects
  - CS number affects number of joins
- Merging closely related CSs helps storage & querying
  - Less CSs means less joins
  - Less CSs means less I/O costs in disk-based systems
  - Compact schema easier to understand and maintain
- CSs are hierarchical, i.e., their property sets can be super/subsets of each other
- Challenge: *exploit the hierarchical structure in order to merge together closely related CSs*

ATHENA

INSTITUTE for the MANAGEMENT of INFORMATION SYSTEMS

Challenge

- Each CS defines a relational table $(s, p_1, p_2, \ldots, p_k)$
- Merging of CS tables results in NULL values for non-shared attributes
- Challenge: merge CSs and reduce NULL value effect

e.g.:

c_0 = {name, age}
c_1 = {name, age, marriedTo}
c_2 = {name, age, marriedTo, worksAt}

s	p_a	p_b
s_1	o_1	o_2
s_2	o_3	o_4

c_0

s	p_a	p_b	p_c
s_3	o_5	o_6	o_7
s_4	o_8	o_9	o_{10}

c_1

s	p_a	p_b	p_c	p_d
s_5	o_{11}	o_{12}	o_{13}	o_{14}
s_6	o_{15}	o_{16}	o_{17}	o_{18}

c_2

s	p_a	p_b	p_c	p_d
s_1	o_1	o_2	NULL	NULL
s_2	o_3	o_4	NULL	NULL
s_3	o_5	o_6	o_7	NULL
s_4	o_8	o_9	o_{10}	NULL
s_5	o_{11}	o_{12}	o_{13}	o_{14}
s_6	o_{15}	o_{16}	o_{17}	o_{18}

c_{merged}

Approach

- Use a dense child table and merge its parents into it
  - **Why dense? ->** # of NULLs is proportional to # of records of table to be merged
  - **Why child? ->** more specialized, thus will contain columns of parents
- Identify dense CSs
    - if $|c_i| > m \ x \ |c_{max}|$ parameter => c_i is dense
  - Every resulting (merged) table will contain **exactly one dense node** (and several non-dense)
- Find optimal merging of ancestors to dense child CSs

e.g.
c_1: {name, age, address}, c_2: {name, age}: c_1 child of c_2

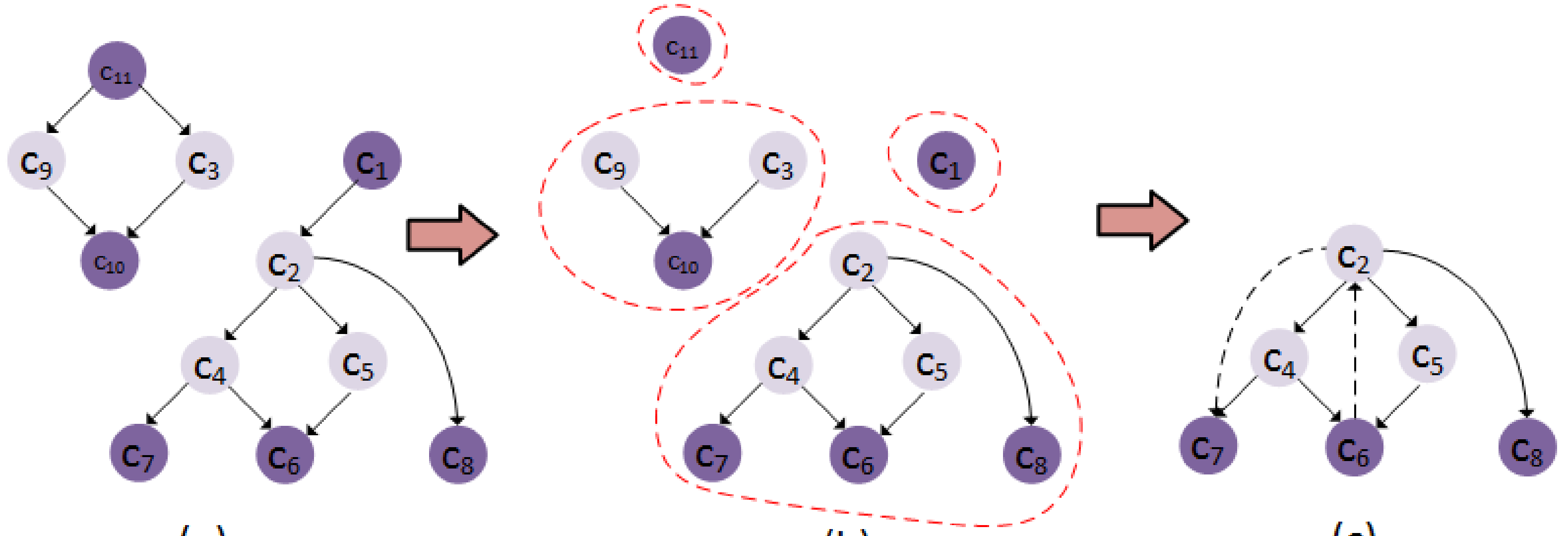
hier_merge(c_1, c_2) = c_{12}

c_{12}: {name, age, address)

ATHENA

INSTITUTE for the MANAGEMENT of INFORMATION SYSTEMS

Approach - Example



(a) (b) (c)

c_2	c_4	c_5
c_7	c_7	c_6
c_7	c_6	c_6
c_6	c_7	c_6
c_6	c_6	c_6
c_8	c_7	c_6
c_8	c_6	c_6

(d)

Approach – Loading and Merging

- Finding the optimal solution is equivalent to enumerating all possible sub-graphs -> *exponential*
- Greedy approximation
  - At each step, merge parent CS and dense child CS that minimize objective cost function
  - Cost function minimizes the number of NULL values introduced by the merge
- Tuning of *m* parameter

Approach – Querying

- Parse incoming SPARQL queries
  - Identify query CSs that match merged CSs in the dataset
  - Rewrite query as an SQL statement with UNIONs between matched CSs
  - In case of SO/OS joins, prune off CSs that are not linked
- Pass final query to relational optimizer
- Build and output results

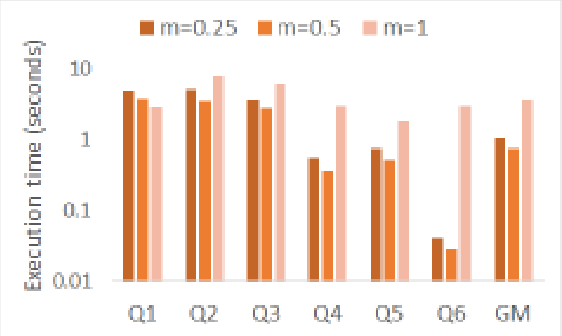
Implementation & Evaluation

- Implemented *raxonDB* on top of relational backbone
- Evaluated on real & synthetic data:
  - Geonames (~150m triples)
  - Reactome (~15m triples)
  - LUBM (up to 350m triples)
  - WatDiv (up to 100m triples)
- Measured Loading Time, Disk Size, Query Processing
- Compared with plain ECS indexing and state of the art competitors

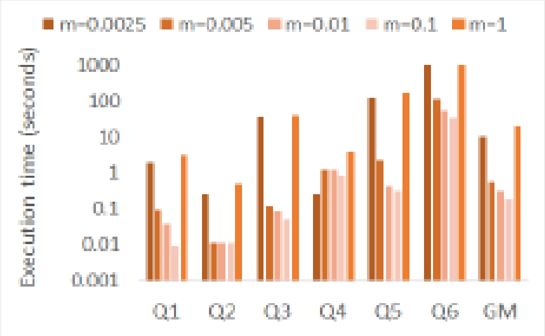
Implementation & Evaluation (Loading)

Dataset	Size (MB)	Time	# Tables (CSs)	# of ECSs	Dense CS Coverage
Reactome Simple	781	3min	112	346	100%
Reactome (m=0.05)	675	4min	35	252	97%
Reactome (m=0.25)	865	4min	14	73	77%
Geonames Simple	4991	69min	851	12136	100%
Geonames (m=0.0025)	4999	70min	82	2455	97%
Geonames (m=0.05)	5093	91min	19	76	87%
Geonames (m=0.1)	5104	92min	6	28	83%
LUBM Simple	591	3min	14	68	100%
LUBM (m=0.25)	610	3min	6	21	90%
LUBM (m=0.5)	620	3min	3	6	58%
WatDiv Simple	4910	97min	5667	802	100%
WatDiv (m=0.01)	5094	75min	67	99	77%
WatDiv (m=0.1)	5250	75min	25	23	63%
WatDiv (m=0.5)	5250	77min	16	19	55%

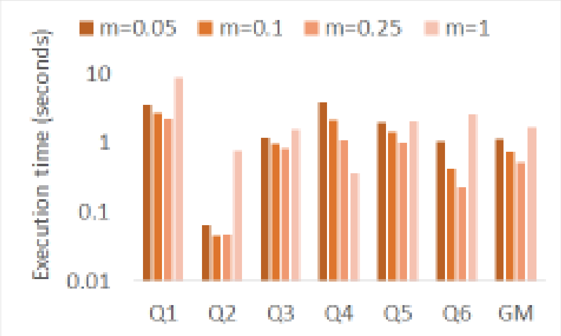
Implementation & Evaluation (Querying)



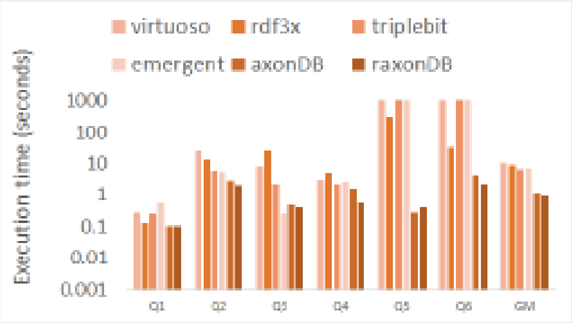
(a) Execution time (seconds) for LUBM



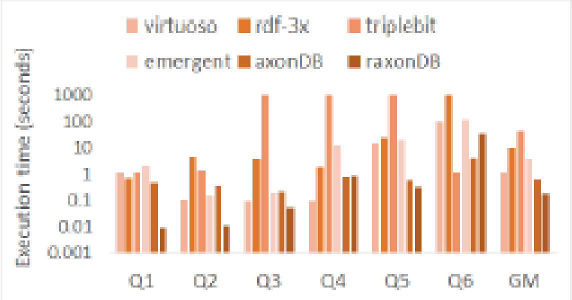
(b) Execution time (seconds) for Geonames



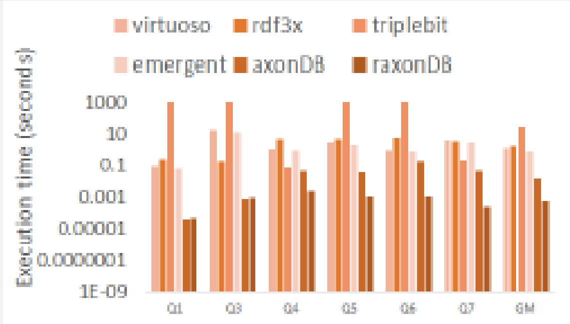
(c) Execution time (seconds) for Reactome



(a) Execution time (seconds) for LUBM2000



(b) Execution time (seconds) for Geonames



(c) Execution time (seconds) for Reactome

Papers and more info

- Ongoing (technical report):
  - Meimaris, Marios, and George Papastefanatos. "Hierarchical Characteristic Set Merging for Optimizing SPARQL Queries in Heterogeneous RDF." *arXiv preprint arXiv:1809.02345* (2018).
- Previous works:
  - Meimaris, Marios, and George Papastefanatos. "Double Chain-Star: an RDF indexing scheme for fast processing of SPARQL joins." *EDBT*. 2016.
  - Meimaris, Marios, et al. "Extended characteristic sets: graph indexing for SPARQL query optimization." *ICDE.* 2017.

ATHENA

INSTITUTE for the MANAGEMENT of INFORMATION SYSTEMS

Future Work

- Optimal merging in polynomial time
- Better cost functions
- Distributed version of raxonDB
  - Exploit well-established relational backbones
    - Impala (SQL Engine) over Hive or Kudu

Thank you

Questions?

H.F.R.I. Hellenic Foundation for Research & Innovation

This research is funded by the project VisualFacts (#1614) - 1st Call of the Hellenic Foundation for Research and Innovation Research Projects for the support of post-doctoral researchers.

ATHENA

INSTITUTE for the MANAGEMENT of INFORMATION SYSTEMS