exploiting **Emergent Schemas** to make RDF systems more efficient

Minh-Duc Pham

Peter Boncz

CWI & VU University, Amsterdam

WWW Conference, 2015

Deriving an Emergent Relational Schema from RDF Data

Minh-Duc Pham[△] m.d.pham@vu.nl
Linnea Passing[□] passing@in.tum.de
Orri Erling[◇] oerling@openlinksw.com
Peter Boncz[⊙] boncz@cwi.nl

[△]Vrije Universiteit Amsterdam, The Netherlands
[□]Technische Universität München, Germany
[◇]OpenLink Software, UK [⊙]CWI, The Netherlands

ABSTRACT

We motivate and describe techniques that allow to detect an "emergent" *relational schema* from RDF data. We show that on a wide variety of datasets, the found structure explains well over 90% of the RDF triples. Further, we also describe technical solutions to the semantic challenge to give short names that humans find logical to these emergent tables, columns and relationships between tables. Our techniques can be exploited in many ways, e.g., to improve the efficiency of SPARQL systems, or to use existing SQL-based applications on top of any RDF dataset using a RDBMS.

O (subject, property, object) columns[1]. SQL systems tend to be more efficient than triple stores, because the latter need query plans with many self-joins – one per SPARQL triple pattern. Not only are these extra joins expensive, but because the complexity of query optimization is exponential in the amount of joins, SPARQL query optimization is much more complex than SQL query optimization. As a result, large SPARQL queries often execute with a suboptimal plan, to much performance detriment. RDBMS's can further store data efficiently e.g. using advanced techniques such as column-wise compression, table partitioning, materialized views and multi-dimensional data clustering. These

Summary

exploiting **Emergent Schemas** to make RDF systems more efficient

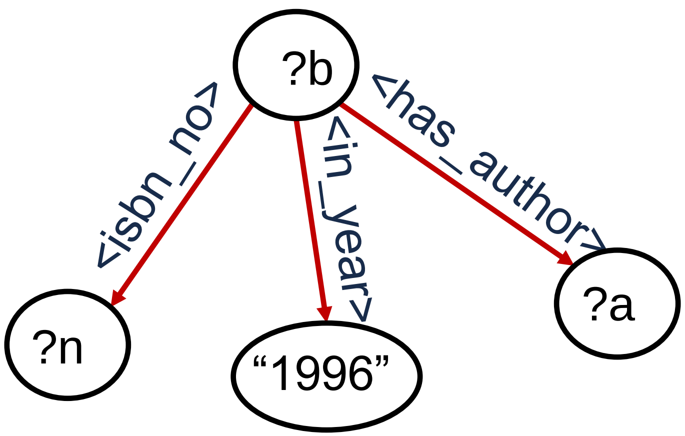
- **the 3 weaknesses of triple stores**
- “schema” confusion between DB an SW practitioners -- but both are right!
- Emergent Schemas to the rescue
- 3 technical details from the ISWC paper
  1. **structured storage is an efficient form of POS storage**
  2. **structure-aware SPARQL execution only as optimization (protects against ?P queries)**
  3. **relation plan does most work, new operator RDFscan adds the missing bindings**

Triplestore weakness #1

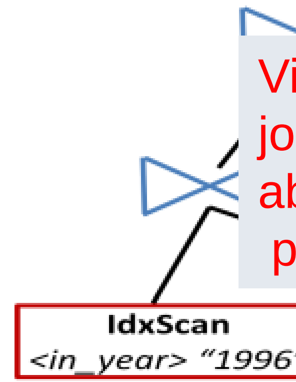
- superflous joins explode query complexity

book query:

```
SELECT ?a ?n WHERE {
  ?b    <has_author> ?a.
  ?b    <in_year>   "1996".
  ?b    <isbn_no>    ?n
}
```

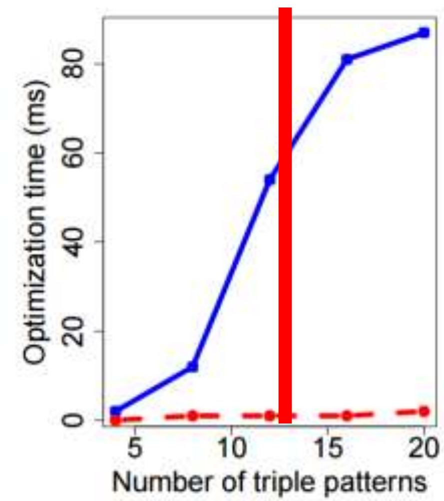


< Query graph>



< Example q

Virtuoso starts cutting join order search space above 12 patterns: best plan may be missed



(b) Virtuoso

- query has unnecessary joins
  - in a relational DB, this is retrieving a record, not a join
  - problem #1: joins are **costly at query execution time**
  - problem #2: query **optimization complexity is $O(3^N)$**

with **star patterns** size F, **exponentially worse** (3^F) optimization space coverage

CWI Centrum Wiskunde & Informatica

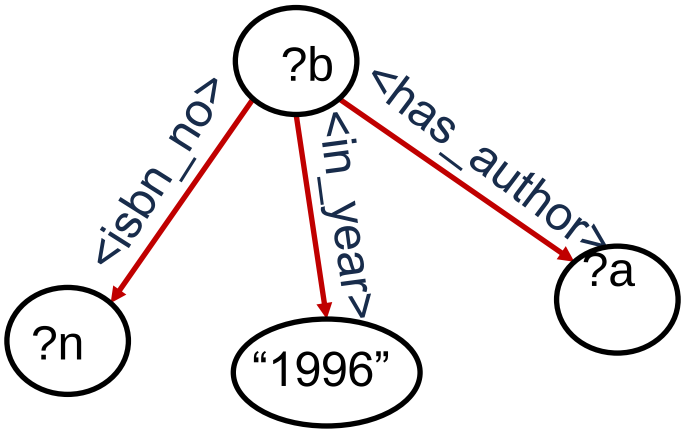
VU Vrije Universiteit Amsterdam

Triplestore weakness #2

- superflous joins explode query complexity

book query:

```
SELECT ?a ?n WHERE {
  ?b    <has_author> ?a.
  ?b    <in_year>    "1996".
  ?b    <isbn_no>     ?n
}
```



< Query graph>

- structural correlations
  - if (?b has an <isbn_no>) it's a book, and it has <in_year> and <has_author>
  - query optimizer estimates using the **independence assumption**
  - problem: result size estimates are wrong ➔ wrong query plan chosen

Query Optimization Quality of SPARQL is much worse than for SQL

Triplestore weakness #3

- RDF architect has no grip on data locality

book query:

```
SELECT ?a ?n WHERE {
  ?b     <has_author> ?a.
  ?b     <in_year>    "1996".
  ?b     <isbn_no>     ?n
}
```

O	P	S
0		
1995	year	4
1996	year	0
1996	year	6
1997	year	3
1998	year	2
ZZZZ		

⋈

P	S	O
a		
isbn	0	i1996
isbn	2	i1998
isbn	3	i1997
isbn	4	i1995
isbn	6	i1996
z		

⋈

P	S	O
a		
auth	0	a1996
auth	2	a1998
auth	3	a1997
auth	4	a1995
auth	5	foo
auth	6	a1996

hexastorage ("indexing all orders") does **not** save you from lack of locality!!

- relational **clustered index**

year	author	isbn
1975	a1995	i1995
1996	a1996	i1996
1996	a1996	i1996
1997	a1997	i1997
1998	a1998	i1998

- relational **partitioned table**

1995

author	isbn
a1995	i1995

1996

author	isbn
a1996	i1996
a1996	i1996

1997

author	isbn
a1997	i1997

1998

author	isbn
a1998	i1998

Summary

exploiting **Emergent Schemas** to make RDF systems more efficient

- the 3 weaknesses of triple stores
- **"schema" confusion between DB an SW practitioners -- but both are right!**
- Emergent Schemas to the rescue
- 3 technical details from the ISWC paper
  1. **structured storage is an efficient form of POS storage**
  2. **structure-aware SPARQL execution only as optimization (protects against ?P queries)**
  3. **relation plan does most work, new operator RDFscan adds the missing bindings**

What does “schema” mean to you?

- Relational answer: the structure definition of a database (**tables + constraints**)

- Semantic web community answer: **Ontologies and Vocabularies**

Centrum Wiskunde & Informatica

VRIJE UNIVERSITEIT AMSTERDAM

What does "schema" mean to you?

- Relational answer: the structure definition of a database (**tables + constraints**)
  - structure of one, particular, database, i.e. the structure of one dataset
  - not intended for reuse, or data integration ☹
  - give the query writer a **clear idea** of what the data looks like
  - must be declared **before** the data can be used ➔ "schema first"

- Semantic web community answer: **Ontologies and Vocabularies**
  - model a knowledge universe so current and future users can denote concepts in a universally understood way in many different contexts ➔ good for **data integration**
  - ontology classes are a poor descriptor of data structure:
    - **partial** use (DBpedia): **<30%** of ontology class attributes occur (on avg)
    - **mixed** use (Dbpedia): subjects combine **7** ontology classes (on avg)
  - symptom: SPARQL query comes back empty
  - **schema can evolve** by e.g. adding new property triples over time ➔ "schema last"

What does "schema" mean to you?

- Relational answer: the structure definition of a database (**tables + constraints**)
  - structure of one, particular, database, i.e. the structure of one dataset
  - not intended for reuse, or data integration ☹
  - give the query writer a **clear idea** of what the data looks like
  - must be declared **before** the data can be used ➔ "schema first"

- Semantic web community answer: **Ontologies and Vocabularies**

Both the DB and SW notions of "schema" are valuable!

Semantic Web applications can profit from a DB schema:

1. Users understand datasets better, no more empty SPARQL results
2. Systems can become more efficient (storage, qopt, execution) ➔ ISWC2016

Summary

exploiting **Emergent Schemas** to make RDF systems more efficient

- the 3 weaknesses of triple stores
- "schema" confusion between DB an SW practitioners -- but both are right!
- **Emergent Schemas** to the rescue
- 3 technical details from the ISWC paper
  1. **structured storage is an efficient form of POS storage**
  2. **structure-aware SPARQL execution only as optimization (protects against ?P queries)**
  3. **relation plan does most work, new operator RDFscan adds the missing bindings**

Emergent Schemas

- Detect the "DB-schema" in RDF data automatically!
  - "association rule mining" to find characteristic sets (CS) of co-occuring properties
  - merge similar CSs using multiple methods into **few tables**
  - derive human-friendly names for tables and columns (exploiting ontologies)

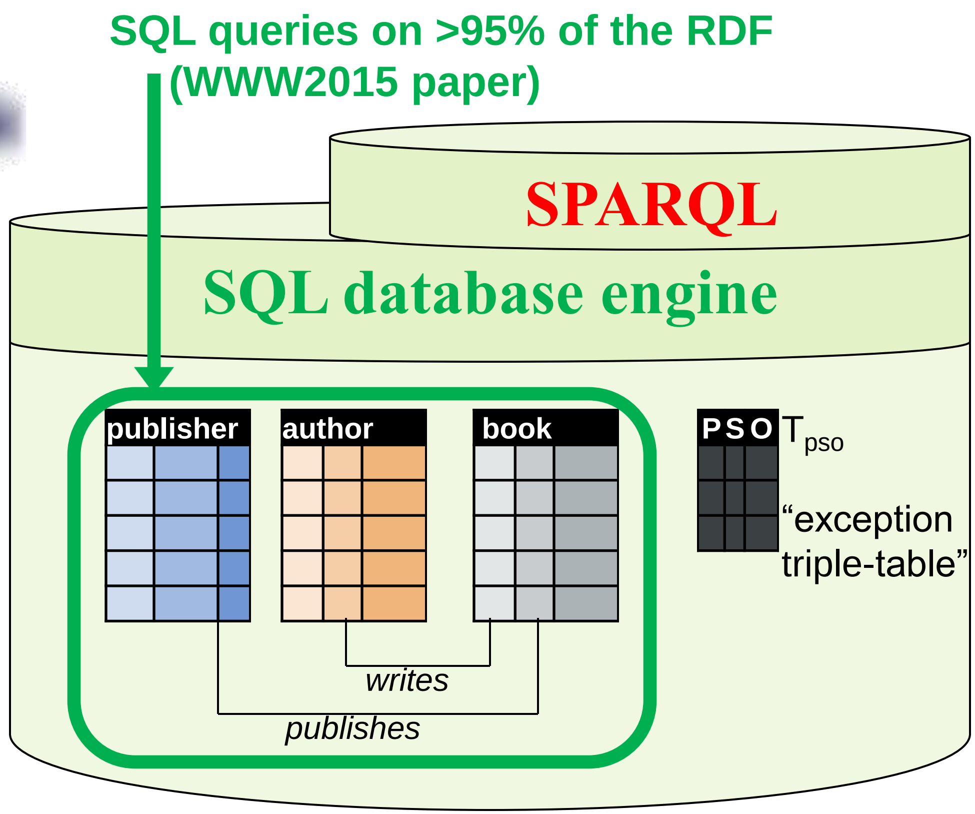
WWW Conference, 2015. *Pham, Passing, Erling, Boncz*
Deriving an Emergent Relational Schema from RDF

	Number of tables			Coverage – Metric C (%)		
Datasets	before merging	after merging	remove small tables	remove small tables	prune infreq. prop.	final schema
LUBM	17	13	12	100	100	**100.00**
BSBM	49	8	8	100	100	**100.00**
SP2B	554	13	10	99.99	99.65	**99.65**
MusicBrainz	27	12	12	100	99.9	**99.60**
EuroStat	44	10	5	99.73	99.53	**99.53**
DBLP	249	9	6	100	99.68	**99.60**
PubMed	3340	14	12	100	99.75	**99.73**
WebData.	13354	3000	253	98.17	94.37	**92.79**
DBpedia	439629	542	234	99.12	96.68	**95.82**

Table 4: #tables and metric C after merging & filtering

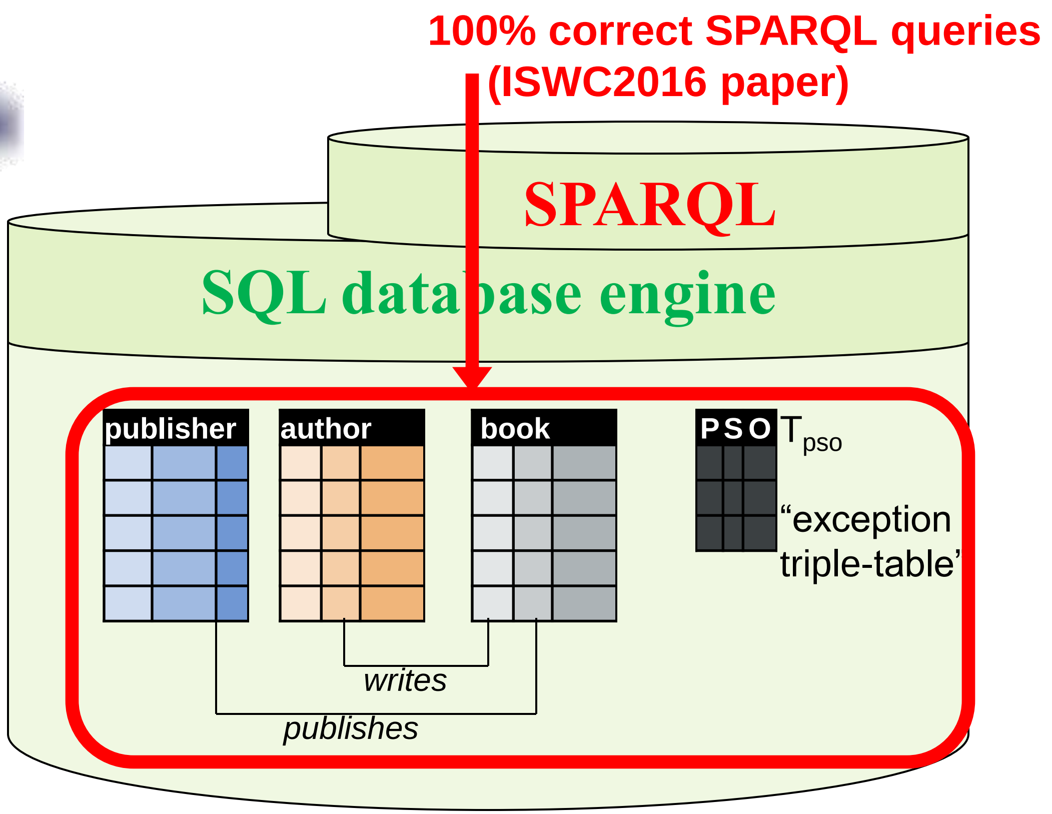
Emergent Schemas

- Detect the "DB-schema" in RDF data automatically!
  - "association rule mining" to find characteristic sets (CS) of co-occuring properties
  - merge similar CSs using multiple methods into **few tables**
  - derive human-friendly names for tables and columns (exploiting ontologies)



Emergent Schemas

- Detect the "DB-schema" in RDF data automatically!
  - "association rule mining" to find characteristic sets (CS) of co-occuring properties
  - merge similar CSs using multiple methods into **few tables**
  - derive human-friendly names for tables and columns (exploiting ontologies)



Summary

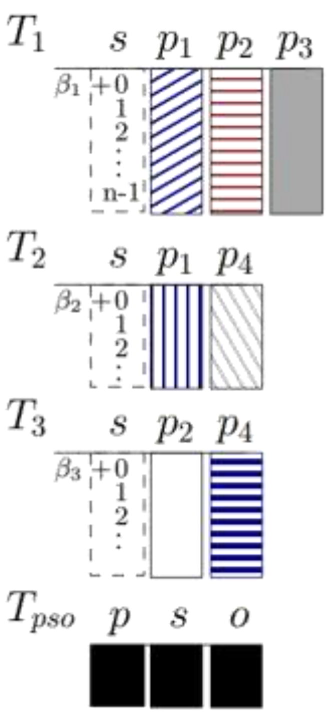
exploiting **Emergent Schemas** to make RDF systems more efficient

- the 3 weaknesses of triple stores
- "schema" confusion between DB an SW practitioners -- but both are right!
- the semantic web is mostly a bunch of tables (is it really very graphy?)
- Emergent Schemas to the rescue
- 3 technical details from **the ISWC paper**
  1. **structured storage is an efficient form of POS storage**
  2. **structure-aware SPARQL execution only as optimization (protects against ?P queries)**
  3. **relation plan does most work, new operator RDFscan adds the missing bindings**

Emergent Table – Smart Column Storage

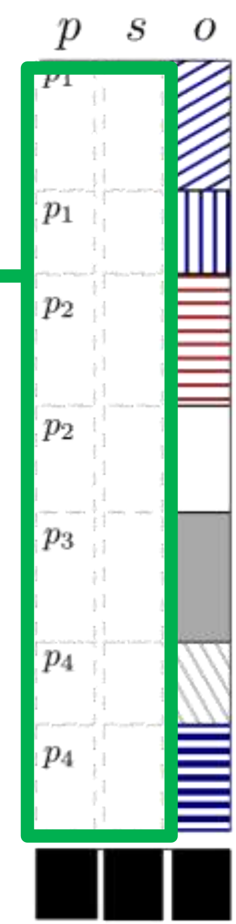
- URI→ integer OID mapping (standard technique)
  - we manipulate this dictionary during emergent schema detection
  - **subject URIs** belonging to table *i* get **dense** numbers starting at β_i
  - $\beta_i = i * 2^{40}$ (really large number

    so OIDs from different tables do not overlap)



dense sequences compress away
otherwise the s-column would be ordered integers
dense sequences allow array lookup
otherwise mergejoins would be needed
- no B-tree or HashMap needed
- makes new RDFscan operator (up next..) cheap

PSO union view



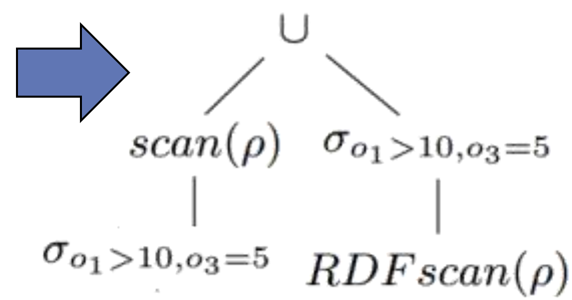
1. **structured storage is an efficient form of POS storage**
2. **structure-aware SPARQL execution only as optimization (protects against ?P queries)**

The RDFscan operator

- the strategy is to replace SPARQL **star patterns** with **table scan**(s)
  - re-using scan is good: RDBMS pushes down selections
    - it triggers index selections, partition pruning,…(goodness)
  - this would only return 95% of the data. How to generate the rest?

  ➔ the new RDFscan operator

```
SELECT ?s ?o1 ?o2 WHERE { ?s  p1  ?o1 .
                          ?s  p2  ?o2 .
                          ?s  p3    5 . FILTER (?o1 > 10) }
```



T_1

s	p_1	p_2	p_3
100	11	2	5
101	13	4	6
102	14		5
103	9	6	
104	15	8	5

T_2

s	p_1	p_3	p_4
200	11	7	1
201		5	2
202	13	9	3

T_{pso}

p	s	o
p_1	0	20
p_1	1	9
p_1	201	15
p_2	0	8
p_2	102	6
p_2	201	4
p_3	0	5
p_6	6	7

Result

s	o_1	o_2
100	11	2
104	15	8
0	20	8
102	14	6
201	15	4

RDFscan

Fig. 8: Example RDF data and expected query result.

Summary

- the 3 weaknesses of triple stores
  1. **superfluous joins explode query complexity**
  2. **structural correlations destroy estimates**
  3. **architect has no grip data locality (no data clustering, no table partitioning, …)**

- SCHEMA: confusion between DB an SW practitioners -- but both are right!

- Emergent Schemas to the rescue

  - neutralizes the 3 triple-store weaknesses!

  = DB notion of schema for RDF data

- 3 technical details from the ISWC paper
  1. **structured storage is an efficient form of POS storage**
  2. **structure-aware SPARQL execution only as optimization (protects against ?P queries)**
  3. **relation plan does most work, new operator RDFscan adds the missing bindings**

Summary

exploiting **Emergent Schemas** to make RDF systems more efficient

- the 3 weaknesses of triple stores
  1. **superfluous joins explode query complexity**
  2. **structural correlations destroy estimates**
  3. **architect has no grip data locality (no data clustering, no table partitioning, …)**

- SCHEMA: confusion between DB an SW practitioners -- but both are right!

- Er
  - n
  = D

- 3

Both the DB and SW notions of "schema" are valuable!

Semantic Web applications can profit from a DB schema:

1. Users understand datasets better, no more empty SPARQL results
2. Systems can become more efficient (storage, qopt, execution) ➔ ISWC2016

DB practitioners can also profit from a SW schema!

1. Tables, columns, constraints should have URIs, link to SW-schemas.
2. Keys could be URIs. Extend SQL! Query SQL+RDF datasets in SQL.