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.

0 (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

- superfluous joins explode query complexity
  - 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

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

Virtuoso starts cutting join order search space above 12 patterns: best plan may be missed
Triplestore weakness #2

- superfluous joins explode query complexity

- 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  \(\Rightarrow\) 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

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

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
```

- relational partitioned table

```
author  isbn
1995    a1995  i1995
1996    a1996  i1996
1996    a1996  i1996
1997    a1997  i1997
```
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
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 \((\text{tables} + \text{constraints})\)
  - 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-occurring 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

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Number of tables before merging</th>
<th>after merging</th>
<th>remove small tables</th>
<th>remove small tables</th>
<th>prune infreq. prop.</th>
<th>final schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUBM</td>
<td>17</td>
<td>13</td>
<td>12</td>
<td>100</td>
<td>100</td>
<td>100.00</td>
</tr>
<tr>
<td>BSBM</td>
<td>49</td>
<td>8</td>
<td>8</td>
<td>100</td>
<td>100</td>
<td>100.00</td>
</tr>
<tr>
<td>SP2B</td>
<td>554</td>
<td>13</td>
<td>10</td>
<td>99.99</td>
<td>99.65</td>
<td>99.65</td>
</tr>
<tr>
<td>MusicBrainz</td>
<td>27</td>
<td>12</td>
<td>12</td>
<td>100</td>
<td>99.9</td>
<td>99.60</td>
</tr>
<tr>
<td>EuroStat</td>
<td>44</td>
<td>10</td>
<td>5</td>
<td>99.73</td>
<td>99.53</td>
<td>99.53</td>
</tr>
<tr>
<td>DBLP</td>
<td>249</td>
<td>9</td>
<td>6</td>
<td>100</td>
<td>99.68</td>
<td>99.60</td>
</tr>
<tr>
<td>PubMed</td>
<td>3340</td>
<td>14</td>
<td>12</td>
<td>100</td>
<td>99.75</td>
<td>99.73</td>
</tr>
<tr>
<td>WebData.</td>
<td>13354</td>
<td>3000</td>
<td>253</td>
<td>98.17</td>
<td>94.37</td>
<td>92.79</td>
</tr>
<tr>
<td>DBpedia</td>
<td>439629</td>
<td>542</td>
<td>234</td>
<td>99.12</td>
<td>96.68</td>
<td>95.82</td>
</tr>
</tbody>
</table>

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-occurring properties
  - merge similar CSs using multiple methods into **few tables**
  - derive human-friendly names for tables and columns (exploiting ontologies)

SQL queries on >95% of the RDF
(WWW2015 paper)
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)

100% correct SPARQL queries
(ISWC2016 paper)
Summary exploiting Emergent Schemas to make RDF systems more efficient

• the 3 weaknesses of triple stores

• “schema” confusion between DB and 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 \( \beta_i \)
  - \( \beta_i = i \times 2^{40} \) (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

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

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

![Table and RDFscan operator diagram]

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

- 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!

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.