Graph Data Management & The Semantic Web

Prof. Dr. Philippe Cudré-Mauroux
Director, eXascale Infolab
University of Fribourg, Switzerland

GDM Workshop, Washington DC, April 5, 2012
The Semantic Web Vision

- Provide interoperable, machine-processable data on the Web
  - Machines cannot meaningfully parse HTML pages
  - Relational DBMSs were designed to be centralized (implicit schema, local data, outside of the Web)
  - XML is document-oriented, localized

- Pushed by Sir Tim Berners-Lee
- Standardized by the W3C (RDF/S, OWL)
- Now promoted by major vendors & actors
  - Google, Yahoo!, Facebook, Oracle
  - Wikipedia
  - Pharmaceuticals, Governments, News industry, etc.
Linked Open Data

- Proposed five years ago by TBL
- Three principles
  1. Use URIs to identify things
  2. Use dereferenceable HTTP URIs and add metadata about things (e.g., in RDF)
  3. Include links to related URIs

http://data.semanticweb.org/person/philippe-cudre-mauroux
Linked Open Data

- For the first time, a giant, open Web of Data is emerging!
- Growing exponentially!

Scale
- Hundreds of data sets
- 30B+ tuples
Decentralized DBMSs!

- Ozone level visualization
  - DATA.gov
  - EPA.gov

- 2 data sets
  - Clean air status [data.gov]
  - Castnet site information [epa.gov]

- 2 SPARQL queries
Structured data
- Resource Description Framework (RDF) [Manola04]

Triples!
1: subject, 2: predicate, 3: object

ex.: philippe, made, idmesh_paper:
1: http://data.semanticweb.org/person/philippe-cudre-mauroux
2: http://xmlns.com/foaf/0.1/made
Web of Data: Data Model (2)

- Naturally forms (distributed) **graphs**

- **Nodes**
  - URIs [subjects]
  - URIs / literals [objects]

- **Edges**
  - URIs [predicates]
  - Directed

![Diagram of a graph with nodes and edges labeled Philippe made Idmesh paper](image-url)
Web of Data: Data Model (3)

- Example

```
<http://www.w3.org/2010/Talkes/1107-rdf-sandro/slides>
```

```
db:Boston dbo:nickname "Beantown".
db:Boston dbo:population "642109"^^xsd:integer.
db:Boston dbo:inState db:Massachusetts.
db:Massachusetts dbo:capital db:Boston.
db:Massachusetts dbo:nickname "The Bay State".
```

Graphs © Sandro Hawke
RDF Schemas (RDFS) [Brickley04]

- Classes, inheritance
  - *Class, Property, SubClass, SubProperty*

- Constraints on structure
  - Constraints on subjects (*Domain*)
  - Constraints on objects (*Range*)

- Collections
  - *List, Bag*

- Reification

Schemas can be **reused, mixed**
RDFS Example
Ontologies (OWL) [W3C OWL 2009]

- Very expressive schemas (ontologies)
- Based on Description Logics
  - Exists in different flavors
- Example: OWL 2 EL axioms:
  - class inclusion (SubClassOf)
  - class equivalence (EquivalentClasses)
  - class disjointness (DisjointClasses)
  - object property inclusion (SubObjectPropertyOf) with or without property chains, and data property inclusion (SubDataPropertyOf)
  - property equivalence (EquivalentObjectProperties and EquivalentDataProperties),
  - transitive object properties (TransitiveObjectProperty)
  - reflexive object properties (ReflexiveObjectProperty)
  - domain restrictions (ObjectPropertyDomain and DataPropertyDomain)
  - range restrictions (ObjectPropertyRange and DataPropertyRange)
  - assertions (SameIndividual, DifferentIndividuals, ClassAssertion, ObjectPropertyAssertion, DataPropertyAssertion, NegativeObjectPropertyAssertion, and NegativeDataPropertyAssertion)
  - functional data properties (FunctionalDataProperty)
  - keys (HasKey)

- Inference! ex.: TransitiveObjectProperty(hasAncestor)
  hasAncestor(x, y) ∧ hasAncestor(y, z) → hasAncestor(x, z)
A few GDM Problems in that space

- Storage
- Query Processing
RDF Storage (1)

XML/JSON **Serialization**
- Exchange format
  - Not meant for humans (ugly)
  - Not meant for DBMSs (verbose)

**Example:**

```xml
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:db="http://dbpedia.org/resource/">
  <rdf:Description rdf:about="http://dbpedia.org/resource/Massachusetts">
    <db:Governor>
    </db:Governor>
    <db:Nickname>Bay State</db:Nickname>
    <db:Capital>
      <rdf:Description rdf:about="http://dbpedia.org/resource/Boston">
        <db:Nickname>Beantown</db:Nickname>
      </rdf:Description>
    </db:Capital>
  </rdf:Description>
</rdf:RDF>
```

13
RDF Storage (2)

RDFa

- Embedding RDF information in HTML pages
- Supported by Google, Yahoo, etc
- Example:

```html
<body>
  <div about="http://dbpedia.org/resource/Massachusetts">The Massachusetts governor is
    <span rel="db:Governor">
      <span about="http://dbpedia.org/resource/Deval_Patrick">Deval Patrick
    </span>,
  </span>
  the nickname is "<span property="db:Nickname">Bay State</span>",
  and the capital
  <span rel="db:Capital">
    <span about="http://dbpedia.org/resource/Boston">
      has the nickname "<span property="db:Nickname">Beantown</span>".
    </span>
  </span>
</div>
</body>
```
RDF Storage (3)

- Various internal formats for DBMSs
  - Giant **triple table** (*triple stores*)
    - |subject|predicate|object|
  - Property tables
    - |subject|property1|property2|property3|…|
    - Decomposition Storage Model

→ JOINS!

- Sub-graphs
  - K-means?
RDF Storage in dipLODocus

- Materialize the joins!
- Dense-pack the values
- Provide new indices

- Co-locate
- Co-locate
- Co-locate
Main Idea - data structures
Template Matching

\[(\text{Student032, FirstName, "Joe"}) \xrightarrow{\text{match}}\]

\[\text{insert}\]

\[\text{Hash-Table}\]

\[\text{hash("Joe")} \rightarrow \text{TID5: (cluster032)}\]

\[\text{Cluster032} \quad \text{Template List 5}\]

\[\text{Schema Template and Template IDs (TIDs)}\]

\[\text{Student Class} \quad \text{TID: 2}\]

\[\text{Takes} \quad \text{TID: 3}\]

\[\text{Is_a} \quad \text{TID: 1}\]

\[\text{StudentID} \quad \text{TID: 6}\]

\[\text{FirstName} \quad \text{TID: 5}\]

\[\text{LastName} \quad \text{TID: 4}\]
Other Interesting Storage Problems

- Beyond partitioning...
  - Directed VS Undirected
  - Object VS DataType properties
  - Strongly connected components (data integration)
  - Subgraph isomorphism(entity linking)
  - Blank nodes
  - Replication
  - New Indices

- Various workloads (analytics, updates)

- Graph analysis of LoD (centrality? etc.)
SPARQL

- Declarative query language for SW data
- SPJ combinations of *triple patterns*
  - E.g., “Retrieve all students who live in Seattle and take a graduate course”
  - Select ?s Where {
      ?s is_a Student
      ?s lives_in Seattle
      ?s takes ?c
      ?c is_a GraduateCourse }
SPARQL Query Execution

- Typically start from bound variables and performs **self-joins** on giant triple table
  - Select ?s Where {
    - ?s is_a Student
    - ?s lives_in Seattle
    - ?s takes ?c
    - ?c is_a GraduateCourse }

- \( \prod_s \sigma_p = \text{is}_a \land o= \text{Student} \)
- \( \bowtie \prod_s \sigma_p = \text{lives}_in \land o= \text{Seattle} \)
- \( \bowtie \prod_s (\sigma_p = \text{takes} \land o= \text{GraduateCourse}) \)
Beyond conjunctions of triple patterns
- Named graphs
- Disjunctions
  - UNION
  - OPTIONAL (semi-structured data model)
- Predicate filters
  - FILTER (?price < 30)
- Duplicate handling (*bag semantics*)
  - DISTINCT, REDUCED
- Wildcards
- *Negation as failure*
  
  ```sparql
  WHERE { ?x foaf:givenName ?name .
  OPTIONAL { ?x dc:date ?date } .
  FILTER (!bound(?date)) }
  ```
Candidate recommendation

Adds a whole new set of [graph] beasts

- Aggregates
- Subqueries
- Filters
  - EXISTS, NOT EXISTS

Property paths (? + * ^ / | )

Inference queries
- Entailment regimes[Glimm11]
Sub-graph Queries (1)

- **Molecule** queries
  - Star-shape sub-queries
  - Combining properties of a given entity
  - E.g., “Retrieve the first name, last name and full address of all students”

![Diagram of a molecule query](image)
Sub-graph Queries (2)

- **Scope** queries
  - Retrieve all triples within a certain scope from a given root node (typically for visualization purposes)
  - E.g., scope 1 from “Student23”
**Path Queries (1)**

- **Property path** queries
  - Queries on series of predicates
  - E.g., “find all professors who supervise students following courses”
Various sub-flavors

- Frequent path queries
  - For optimization or visualization purposes
  - E.g., “Find the most frequent paths of length 2”

Regular expressions for properties (SPARQL 1.1)

- ? + * ^ / |
  - E.g., “find reachable friends through 2 different paths”
    - SELECT * WHERE {
      :John (foaf:friendOf|urn:friend)+ ?friend. }

SELECT * WHERE {
  :John (foaf:friendOf|urn:friend)+ ?friend. }
Inference Queries

- Additional data can be inferred using various sets of logical rules
- Specify which ones to use by entailment regimes [Glimm11]
  - **RDF Schema** has 14 entailment rules
    - E.g., \((p, \text{rdfs:domain}, x) \land (u, p, y)\) => \((u \text{ rdf:type } x)\)
  - **OWL 2 RL** has 70+ entailment rules.
    - E.g., \((p, \text{rdf:type}, \text{owl:FunctionalProperty}) \land (x, p, y1) \land (x, p, y2)\) => \((y1, \text{owl:sameAs}, y2)\)
Federated Queries

- Retrieve data by joining distributed, heterogeneous SPARQL end-points
  - Distributed graph with various latencies attached to nodes
  - Optimum branching (Edmonds’ algorithms) Minimum-Spanning-Tree algorithms, etc.
Thanks for your attention!

- VLDB tutorial (w/ Sameh) last year