Towards distributed RDF querying

Vdovjak, R.; Houben, G.J.P.M.; Stuckenschmidt, H.

Published: 01/01/2004

Document Version
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the author’s version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 31. May. 2018
Towards Distributed RDF Querying

Richard Vdovjak (TU Eindhoven)
Geert-Jan Houben (TU Eindhoven)
Heiner Stuckenschmidt (VU Amsterdam)

Introduction / motivation
- Hera
- Integration model
- Query Processing
- Optimization

Motivation
- Demand for combining distributed data on the Web
  - Comparative Shopping
  - Virtual Museums
  - Digital Libraries
  - Web Portals
- User:
  - formulate his query
  - split the query
  - assemble results

The RDF data model
- statements are \langle Subject, Predicate, Object \rangle triples:
  - \langle http://wwwis.win.tue.nl/~houben\rangle: dc:creator: "Geert-Jan"
- statements describe properties of resources
  - a resource is everything that has an identifier: URI
  - Directed Labelled Multi-graph

Sem. Web / RDF(S) to the Rescue (?)
- RDFS
  - The Pivot Language of the Semantic Web
  - Solves the problem of syntactic heterogeneity of different sources
  - Provides basic modelling primitives and reasoning techniques for conceptual knowledge
  - Built on an extremely simple data model that avoids complications of other object oriented formalisms
  - And (most importantly): It is a standard!

... (still unhappy)
### Possible Solution: Data Warehousing

- Gather All Data Locally
- Watch for Updates
  - Insert New Data
  - Delete Old Data
- Evaluate (locally) User Query

### Data Warehousing: Problems

- Performance Bottleneck
- Freshness
- Copyright / Data Ownership issues

### Another Solution: Virtual Repository

- Split the User Query into Sub-Queries
- Translate Sub-Queries to Source Schemata
- Distribute Sub-Queries
- Assemble Results

### Challenges / Requirements for Integration of RDF(S) Sources

- Unified Interface to data sources
  - Usually only a part of a source is of interest
  - "Freshness" of data must be guaranteed
  - Sources are many, autonomous, and volatile
- Semantic Heterogeneity
  - Schema heterogeneity
  - Designation heterogeneity
- Distributed Query processing
  - Complex/Join queries (SesRQL language)
  - Flexibility w.r.t. user needs/queries
  - (the order of importance changes)
  - Correctness, Completeness
  - Performance

### Hera Design Methodology

- Data retrieval is just a beginning
- Navigational Structure
  - Presentation Rendering
- Adapation / Adaptivity

### The Semantic Layer

#### Hera Back-end

- RDF Source
- Repository
- SQL Interface
- Integration Model

#### Hera Front-end

- User Interface
- Presentation
- User Query
**Conceptual Model (CM)**

- Interface between data retrieval and presentation generation (via SeRQL)
- CM exists on its own (even without instances)
  - Made by a system designer
  - Top down approach
- Specifies the application’s semantics
- What is the information system about
- Expressed in RDF(S)
- Populated on demand with data from several heterogeneous information sources
- Challenge: Map the Sources to the defined CM

**Integration Model (IM)**

- A generic framework for describing, integrating and relating concepts from sources to their CM counterparts
  - View Definition/Translation Language +
  - Object reconciliation language +
  - Language for programming the mediator +
- Expressed in RDFS
- Instantiated by the integration designer into IMIs: program the mediator to overcome the semantic heterogeneity between the sources and the CM

**Schema heterogeneity**

- Sources are autonomous and can therefore differ a lot from each other
- Mappings are formed through the notion of Path Expressions (PE) which form articulations
  - An articulation is a pair of two PEs, one in an external source, one in the CM
  - consists of a link between the start-classes and a link between the ending-edges

**Integration Model in RDF(S)**

**Schema heterogeneity**

- Design the CM like source 1:

**Integration Model**

- Introduction / motivation
- Hera
- Integration model
- Query Processing
- Optimization
**Schema heterogeneity**

- Sometimes a value should be mapped to a list of values
- A transformer is needed for the necessary action
- Denoted in the IM as “obtained by list”

**Designation heterogeneity**

- Different sources may have different ways to uniquely identify instances
- Need to define the identifying properties “primary key” of objects in every source
- Consolidate them into the CM so that a join can be performed across multiple sources

**Designation heterogeneity**

- IM offers three kinds of data-identification:
  1. idByUri → every object has a unique URI
  2. idByValue → a value is unique for an object
  3. idByProperty → a “super resource” defines the uniqueness of an object

**Examples:**

**Designation heterogeneity : idByUri**

- An object is uniquely defined by its URI
- Can be used/imposed within closed communities, e.g. corporate IS
- Does not work World Wide

**Designation heterogeneity : idByValue**

- Value based (similarly to the relational model)
- A object is uniquely defined by one(or more) of its own properties

**Designation heterogeneity : idByProperty**

- idBy-information provided by a super-resource
- Recursive path until either idByUri or idByValue is encountered
Join Example
- If the primary key is the same data is joined

The two idBy-paths are different but joining is still possible if the end-values for the primary key are the same.

Different sources - different qualities
- Sources can “get points” for certain qualities:
  - Data reliability
  - Data quality: e.g. picture quality for a Photo database
  - Reachability
  - Speed
- Sources can be consulted in different order based on the current user preferences
- Decorations
  - making background knowledge explicit
  - “exported” into the CM (extending the Schema)

Extra IM features
- Process Instructions:
  - (Need to compare (primary) keys
    - Transformers/Comparators
    - Conversion functions (date of birth -> Personal number)
    - Look-up table value translations
    - Unit conversions (km->mile)
    - Formal conversion (itl -> jg)
- Direct translation
  - In case of homogeneous Schemas/sources
  - Lists of classes with identical outgoing edges

Extra IM features
- Direct translation
  - In case of homogeneous Schemas/sources
  - Lists of classes with identical outgoing edges

Layout
- Introduction / motivation
- Hera
- Integration model
- Query Processing
- Optimization

CM Instance (CMI)
- The Hera presentation engine needs more data than that resulted from a “bare” user query
- The user query is extended to retrieve literal values (the real content)
- An RDF graph is constructed out of the “flat” SeRQL output

CMI generation
- Given the CM:
  - User Query:
    - Retrieve all writers
    - Extended query:
      - Retrieve also the name, age, hasPortrait P4,P5
Introduction / motivation
Integration model
Query Processing
Optimization

Optimization

Initial Performance experiments
- Queries against small-size applications (e.g., the comics database or virtual museum) answered within ms
- Medium Test Set: RDF version of Wordnet
  - Naturally split in four parts: Nouns (10MB), Glossary (15MB), Similar-To Definitions (2MB), Hyperonyms (8MB)
- Test Setting:
  - Three local stores: Vrije Universiteit Amsterdam, CWI (Amsterdam) and Eindhoven Technical University, Mediator at Vrije Universiteit

Results
When installed locally Performance of distributed system is between 50 and 200% of the original sesame (200-1000 ms ec.)
- The performance drops with the size of the result set due to extensive joining and communication overhead

Optimization: Initial Performance

Where the Time Goes?
Depends on many factors
- Data Size / Source Processing Speed
- Query (complexity, result size ...)
- Connection Speed
- Local Processing
- Communication
- Result Joining

Small Result Set
Large Result Set

Optimization: Initial Performance

Improving the Performance
- Large applications (hundreds of MB) require sophisticated optimization techniques:
- Currently
  - Schema/path Indexing
  - Join ordering
  - Algebraic optimizations
- Work in progress
  - Reducing the transferred data
  - requires an architecture change (similar to P2P)

Optimization: Initial Performance

Schema/Path Index Hierarchy
- Central place to store the translatable paths from the articulations
- The main idea: pushing long paths to sources is more efficient than joining many small paths at the mediator
- fewer joins
- smaller data traffic
- The index is constructed out of a pool of articulations
- Articulations can be added/deleted or modified on the fly: flexible source management
- Index is able to infer “new” paths of its own:

Schema/Path Index Hierarchy

Inference rules:
- Inclusion of “super paths”
  - Given: A → B and B → C
  - Insert A → C
- Transitive closure of subClassOf / subPropertyOf
  - Given: A → B, C → D and B → subClassOf C
  - Insert A → D

Optimization: Initial Performance
Schema/Path Index Hierarchy
- Key = path (sequence of properties)
- Value = list of sources to which this path can be translated

Optimization: Path Indexing

Performance Results: Full Index vs 1-path Index

Optimization: Join Ordering

From a RDF Path to Relational Tables

Optimization: Join Ordering

A Cost Model for RDF Querying
- Data Access Costs
  - Initializing the transmission
  - Transferring the data
- Join Costs
  - Nested Loop Join
  - Hash Join (potentially faster but less flexible w.r.t. determining object identity)
- Costs of a Query Plan:
  - Transmission costs for all relations
  - Join costs for the chosen footprint

Optimization: Join ordering

The Problem of Join Ordering
- Determine the optimal order of join execution in a (chain) query
- Different strategies have different execution costs due to different selectivity of the join operations
- Problem is NP-hard: we consider heuristic solutions

Optimization: Join Ordering

Join Ordering Heuristics
- Complexity of Task demands heuristic approaches, performance of heuristics depends on class of queries.
  - For chain queries the following performs best [Steinbrunn et al., 1997]
- 1) Iterative Improvement
  - Start with random solutions
  - Improve solution using a greedy heuristic
  - Result: local optima
- 2) Simulated Annealing
  - Further improve solutions allowing for increasing costs with a certain probability
  - Helps to get out of local optima and converge towards better solutions

Optimization: Join ordering
**Optimizing Communication: Collaborating Network of Mediators**

- One coordinator + Set of cooperating nodes
- The coordinator generates the (global) query plan for other nodes
- instructions:
  - obtain data
  - receive a table
  - use your local data
  - join obtained data in a given order
  - ship data to another node

**Conclusions**

- Virtual Repositories are a viable solution for building distributed WIS
- Challenges:
  - Semantic heterogeneity
  - Source schemas can differ
  - URI is not enough for joining
  - Flexible Evaluation
  - Performance / Scalability
    - Path Indexing
    - Join Ordering
    - Collaborating Mediators

**Questions**