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 $<$subject, predicate, object$>$ triples:
  - http://www.win.tue.nl/~houben/
  - dc:creator
  - 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 (?)
- RDF(S)
  - 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!
Possible Solution: Data Warehousing

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

Introduction: Data Warehousing

Data Warehousing: Problems

- Performance
- Bottleneck
- Freshness
- Copyright / Data Ownership issues

Introduction: Data Warehousing

Another Solution: Virtual Repository

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

Introduction: Virtual Repository

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 (SeRQL language)
  - Flexibility w.r.t. user needs/query
  - (the order of importance changes)
  - Correctness, Completeness
  - Performance

Introduction: Requirements

Hera Design Methodology

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

The Semantic Layer

Hera Back-end

Hera Front-end

Source Clusters

Repository

Hera Semantic Layer

Introduction: HERA Suite
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

Integration Model in RDF(S)
- 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

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
**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. id By Uri  
     - Every object has a unique URI
  2. id By Value  
     - A value is unique for an object
  3. id By Property  
     - A “super resource” defines the uniqueness of an object

**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)
    - Format conversion (flf -> jgg)
  - 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
- Query Processing: CMI definition
- Query Processing: CMI generation example
Introduction / motivation
Integration model
Query Processing
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), Hypernyms (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/sec.)
  - The performance drops with the size of the result set due to extensive joining and communication overhead

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

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: Approaches

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

- 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

From a RDF Path to Relational Tables

The Problem of Join Ordering

A Cost Model for RDF Querying

Join Ordering Heuristics
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