Querying Linked Data on the Web

Olaf Hartig
University of Waterloo

Nov. 12, 2013
The Traditional, Hypertext Web

Data exposed to the Web via HTML

Olaf Hartig - Querying Linked Data on the Web
Linked Data Publishing Principles

Data model: RDF

(CIA World Factbook: Albania, unemployment rate, 13.2%)

(MovieDB: War Child, release date, 12 July 1999)

(War Child, filming location, Albania)

(Michael Davie, directed, War Child)

(War Child,)

Albania
Linked Data Publishing Principles

Data model: **RDF**

Global identifier: **URI**

(C http://...imdb.../WarChild , release date , 12 July 1999 )
(C http://...imdb.../WarChild , filming location , http://cia.../Albania )
(C http://...imdb.../MichaelDavie , directed , http://...imdb.../WarChild )

(C http://cia.../Albania , unemployment rate , 13.2% )
Linked Data Publishing Principles

Data model: RDF
Global identifier: URI
Access mechanism: HTTP

MovieDB

CIA World Factbook

War Child

Albania

( http://...imdb.../WarChild , release date , 12 July 1999 )
( http://...imdb.../WarChild , filming location , http://cia.../Albania )
( http://...imdb.../MichaelDavie , directed , http://...imdb.../WarChild )

( http://cia.../Albania , unemployment rate , 13.2% )
Linked Data Publishing Principles

Data model: RDF
Global identifier: URI
Access mechanism: HTTP
Connection: data links

( http://...imdb.../WarChild , release date , 12 July 1999 )
( http://...imdb.../WarChild , filming location , http://cia.../Albania )
( http://...imdb.../MichaelDavie , directed , http://...imdb.../WarChild )

( http://cia.../Albania , unemployment rate , 13.2% )
A Globally Distributed Network of Data
A Huge Distributed Database
Unusual Characteristics of this DB

- Data access capabilities of any system that aims to access this DB are inherently limited
  - HTTP requests only
  - No queries (i.e., we cannot assume that data sources provide a query service)
- Number of *potential* data sources is infinite
- It is impossible to have a DB catalog that is complete or up-to-date (or even both)
Linked Data Query Processing

... is a new research field that focuses on querying this distributed DB

- **Criteria:**
  - On-line execution
  - Rely only on the Linked Data principles

- **Use cases:** live querying where freshness and discovery of results is more important than an almost instant answer
Outline

- Theoretical Foundations

- Execution of Linked Data Queries

- Conclusions
What Query Language Do We Use?

- SPARQL seems to be a natural choice
  - SPARQL is the standard query language for RDF data
  - Linked Data is based on the RDF data model
- However, the standard definition of SPARQL focuses on a more traditional query scenario
Some Terminology

- **Core of a SPARQL query is a query pattern**
  - Triple pattern: RDF triple that may contain variables; e.g., ( ?p, foaf:interest, ?i )
  - Group pattern: conjunction of query patterns ( \( P_1 \text{ AND } P_2 \) )
  - Union, filter, optional, etc.

- **Any query result is represented by a set / bag of valuations**
  \[ \Omega = \{ \mu_1, \ldots, \mu_n \} \]

- **Valuation** (a.k.a. solution mapping or substitution): partial mapping of query variables to RDF terms
  \[ \mu = \{ ?p \rightarrow \text{alice}, ?i \rightarrow \text{yoga} \} \]

- Valuations in a particular query result are called **solutions**
SPARQL Query Semantics

- **Query semantics is defined by an evaluation function**
  - i.e., \( \text{eval}( P, G ) \) defines the expected query result for evaluating a query pattern \( P \) over an RDF graph \( G \)
  - Function \( \text{eval}( P, G ) \) is defined recursively

- **Triple pattern (base case):**
  - \( \text{eval}( tp, G ) := \{ \mu | \mu[tp] \in G \text{ and } \text{dom}(\mu) = \text{vars}(tp) \} \)

- **Group pattern:**
  - \( \text{eval}( (P_1 \text{ AND } P_2), G ) := \text{Join}( \text{eval}(P_1,G), \text{eval}(P_2,G) ) \)
  - where \( \text{Join}( \Omega_1, \Omega_2 ) := \{ \mu_1 \cup \mu_2 | \mu_1 \in \Omega_1 \text{ and } \mu_2 \in \Omega_2 \text{ and } \mu_1 \text{ and } \mu_2 \text{ are compatible } \} \)
  - etc.
What Query Language Do We Use?

• SPARQL seems to be a natural choice
  • SPARQL is the standard query language for RDF data
  • Linked Data is based on the RDF data model

• However, the standard definition of SPARQL focuses on a more traditional query scenario
  • The definition captures queries over a predefined dataset (which may consist of Linked Data copied from the WWW)

• To capture queries over the Web itself we need to adjust the definition ...
Outline

- **Theoretical Foundations**
  - Data Model
  - Full-Web Query Semantics
  - Reachability-based Query Semantics

- **Execution of Linked Data Queries**

- **Conclusions**
Data Model

**Definition:** Assume a countably infinite set \( \mathcal{D} \) (LD documents). A **Web of Linked Data** is a tuple \( W = ( D, adoc, data ) \) where

- \( D \) is a proper subset of \( \mathcal{D} \),
- \( adoc \) is a partial mapping from URIs to \( D \), and
- \( data \) is a total mapping from \( D \) to finite sets of RDF triples.

**Definition:** Let \( \mathcal{W} \) denote the infinite set of all Webs of Linked Data. A **Linked Data query** \( Q \) is a total function over \( \mathcal{W} \).
Outline

- **Theoretical Foundations**
  - Data Model
  - Full-Web Query Semantics
  - Reachability-based Query Semantics

- **Execution of Linked Data Queries**

- **Conclusions**
Definition: For any Web of Linked Data $W = (D, adoc, data)$ we define

$$AllData(W) := \bigcup_{d \in D} data(d).$$

Definition: The $\text{SPARQL}_{LD}$ query $Q^P$ that uses SPARQL query pattern $P$, is a Linked Data query that, for any Web of Linked Data $W$, is defined by $Q^P(W) := \text{eval}(P, AllData(W))$.

standard SPARQL evaluation function
What about Computational Feasibility?

**Definition:** For any Web of Linked Data $W = (D, adoc, data)$ we define

$$\text{AllData}(W) := \bigcup_{d \in D} data(d).$$

**Definition:** The SPARQL\textsubscript{LD} query $Q^P$ that uses SPARQL query pattern $P$, is a Linked Data query that, for any Web of Linked Data $W$, is defined by $Q^P(W) := \text{eval}(P, \text{AllData}(W))$.

- *(Ordinary) Turing machines are unsuitable:*
  - Limited data access capabilities not properly captured

- **Web machines**
  - Abiteboul and Vianu, 1997
  - Mendelzon and Milo, 1997
LD Machine

- Multi-tape Turing machine

- Input
- Work
- Output
LD Machine

- Multi-tape Turing machine
  - Web Input: # enc(u₁) enc(adoc(u₁)) # enc(u₂) enc(adoc(u₂)) # · · ·
  - Input
  - Work
  - Output
LD Machine

- **Multi-tape Turing machine**
  - Web Input ➔ Web Input
  - Input
  - Work
  - Output

- **Access to Web Input is restricted**
  - Only by performing a particular procedure that models looking up a given HTTP-based URI on the WWW
For $Q$ exists an LD machine $M_Q$ such that for any $W$:

- $M_Q$ halts after a finite number of computation steps, and
- $M_Q$ outputs the complete result $Q(W)$
Eventually Computable LD Queries

For $Q$ exists an LD machine $M_Q$ such that for any $W$

1. Output always encodes a subset of query result $Q(W)$, and
2. Each $\mu \in Q(W)$ appears after a finite number of steps

No guarantee for termination
What about SPARQL\(_{LD}\) Queries?

**Theorem [ESWC'12]:** If a satisfiable SPARQL\(_{LD}\) query \(Q^P\) is *monotonic*, then it is eventually computable (but not finitely computable); otherwise, it is not even eventually computable.

- **Main reason:** Set of all HTTP-based URIs is infinite.
Outline

- Theoretical Foundations
  - Data Model
  - Full-Web Query Semantics
  - Reachability-based Query Semantics

- Execution of Linked Data Queries

- Conclusions
Reachability-Based Query Semantics
Reachability-Based Query Semantics

- Seed URIs $S \subseteq U$
Reachability-Based Query Semantics

- Reachability criterion $c$
  - (Turing) computable function
    \[ c: \mathcal{T} \times \mathcal{U} \times \mathcal{P} \rightarrow \{\text{true}, \text{false}\} \]

- Seed URIs $S \subset \mathcal{U}$
Reachability-Based Query Semantics

\[ Q_{c}^{P,S}(W) := \text{eval}(P, \text{AllData}(W^*)) \]
(Reachability-Based) $c_{\text{All}}$-Semantics

$$Q^{P,S}_{c_{\text{All}}} (W) := \text{eval}(P, \text{AllData}(W^*))$$

$$c_{\text{All}}(t,u,P) := \text{true for all } (t,u,P) \in \mathcal{T} \times \mathcal{U} \times \mathcal{P}$$
(Reachability-Based) $c_{\text{None}}$-Semantics

$Q_{c_{\text{None}}}^{P,S}(W) := \text{eval}(P, \text{AllData}(W^*))$

$c_{\text{None}}(t,u,P) := \text{false for all } (t,u,P) \in T \times U \times P$
(Reachability-Based) $c_{\text{Match}}$-Semantics

$$Q_{c_{\text{Match}}}^{P,S}(W) := \text{eval}(P, \text{AllData}(W^*))$$

$c_{\text{Match}}(t,u,P) := \text{true} \iff t \in T$ matches a triple pattern in $P \in \mathcal{P}$
LD Machine-Based Computability

- If we restrict\(^1\) our data model s.t. Webs of Linked Data are finite (i.e., consist of a finite number of documents), then:

  **Theorem [ESWC'12]:** For any reachability criterion c, any SPARQL based Linked Data query under c-semantics is finitely computable.

- Proof idea: specify an LD machine for computing these queries by performing two separate, consecutive phases
  - **Data retrieval phase:** recursively traverse all data links that qualify; terminates for any finite Web of Linked Data
  - **Result computation phase:** use all data retrieved in the first phase to generate the (sound and complete) query result; terminates due to the finiteness of the retrieved data

\(^1\)For results w.r.t. the unrestricted data model (i.e., Webs of Linked Data that may be infinitely large), refer to [ESWC'12].
Outline

- Theoretical Foundations
  - Data Model ✓
  - Full-Web Query Semantics ✓
  - Reachability-based Query Semantics ✓

- Execution of Linked Data Queries

- Conclusions
“Ingredients” for LD Query Execution

Query-local data
“Ingredients” for LD Query Execution

- Data retrieval approach
- Data source selection
- Data source ranking (optional, for optimization)

Query-local data

GET http://.../movie2449
“Ingredients” for LD Query Execution

- Data retrieval approach
  - Data source selection
  - Data source ranking (optional, for optimization)

- Result construction approach
  - i.e., query-local data processing

Query-local data

<table>
<thead>
<tr>
<th>?actor</th>
<th>?loc</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://mdb.../Ric">http://mdb.../Ric</a></td>
<td><a href="http://geo.../Rome">http://geo.../Rome</a></td>
</tr>
</tbody>
</table>
“Ingredients” for LD Query Execution

- Data retrieval approach
- Data source selection
- Data source ranking (optional, for optimization)

- Combining data retrieval and result construction

- Result construction approach
  - i.e., query-local data processing

Query-local data:

GET http://.../movie2449

<table>
<thead>
<tr>
<th>?actor</th>
<th>?loc</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://mdb.../Ric">http://mdb.../Ric</a></td>
<td><a href="http://geo.../Rome">http://geo.../Rome</a></td>
</tr>
</tbody>
</table>
Outline

- **Theoretical Foundations**
  - Data Model
  - Full-Web Query Semantics
  - Reachability-based Query Semantics

- **Execution of Linked Data Queries**
  - Source Selection Strategies
  - Execution Process
  - Source Ranking

- **Conclusions**
Query-Specific Relevance of URIs

- A URI is **relevant** for a query execution if a lookup of this URI gives us data that contributes to the query result.

- **Example:**
  - Conjunctive query (BGP): \( \{ (Bob, \text{lives in}, ?x) , (?y, \text{lives in}, ?x) \} \)
  - Looking up URI *Bob* gives us: \( \{ (Bob, \text{lives in}, Berlin) , ... \} \)
  - Looking up URI *Alice* gives us: \( \{ (Alice, \text{lives in}, Berlin) , ... \} \)
  - Hence, \( \mu = \{ ?x \rightarrow Berlin , ?y \rightarrow Alice \} \) is a solution
  - Thus, URIs *Bob* and *Alice* are relevant for the query

- **Simply contributing a matching triple is not sufficient:**
  - Suppose, URI *Charles* gives us \( \{ (Charles, \text{lives in}, London) , ... \} \)
  - Since the matching triple cannot be used for computing a solution, URI *Charles* is not relevant.
Objective of Source Selection

- Source selection: Given a Linked Data query, determine a set of URIs to look up

- Ideal source selection approach:
  - For any query, selects all relevant URIs
  - For any query, selects relevant URIs only

- Irrelevant URIs are not required to answer the query
  - Avoiding their lookup reduces cost of query executions significantly!

- Caveat:
  - What URIs are relevant (resp. irrelevant) is unknown before the query execution has been completed.
Idea of Index-Based Source Selection

- Use a **pre-populated index** structure to determine relevant URIs (and to avoid as many irrelevant ones as possible)

- Example: triple-pattern-based indexes

  - Key: $tp$
  - Entry: $\{ uri_1, uri_2, \ldots, uri_n \}$

- For single triple pattern queries, source selection using such an index structure is sound and complete (w.r.t. the indexed URIs)
General Properties of Lookup Indexes

• Type of index **keys** depends on the particular index structure used
  - e.g., triple patterns

• **Index entries**:  
  - Usually, a set of URIs
  - Each URI in such an entry may be paired with a cardinality (utilized for source ranking)
  - Indexed URIs may appear multiple times (i.e., associated with multiple index keys)

• **Represent a summary of the data from all indexed URIs**
Live Exploration

• General idea: Perform a recursive URI lookup process at query execution runtime
  • Start from a set of seed URIs
  • Explore the queried Web by traversing data links
• Retrieved data serves two purposes:
  (1) Discover further URIs
  (2) Construct query result
• Lookup of URIs may be constrained (i.e., not all links need be traversed)
  • Natural support of reachability-based query semantics
**Live Exploration – vs. – Index-Based**

<table>
<thead>
<tr>
<th>Possibilities for parallelized data retrieval are limited</th>
<th>Data retrieval can be fully parallelized</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Data retrieval adds to query execution time significantly</td>
<td>- Reduces the impact of data retrieval on query exec. time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usable immediately</th>
<th>Usable only after initialization phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Most suitable for “on-demand” querying scenario</td>
<td>- Depends on what has been selected for the index</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Depends on the structure of the network of data links</th>
<th>May miss new data sources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

None of both strategies is superior over the other w.r.t. result completeness (under full-Web query semantics).

- Both strategies may miss (different) solutions for a query
Hybrid Source Selection

Why not get the best of both strategies by combining them?

• Ideas:
  • Use index to obtain seed URIs for live exploration (e.g., “mixed strategy” [LT10])
  • Feed back information discovered by live exploration to update, to expand, or to reorganize the index
  • Use data summary for controlling a live exploration process (e.g., by prioritizing the URIs scheduled for lookup)
Outline

- **Theoretical Foundations**
  - Data Model ✓
  - Full-Web Query Semantics ✓
  - Reachability-based Query Semantics ✓

- **Execution of Linked Data Queries**
  - Source Selection Strategies ✓
  - Execution Process
  - Source Ranking

- **Conclusions**
Separated Execution Approaches

... clearly separate data retrieval and result construction into two consecutive phases.

1. Query-local data

2. GET http://.../movie2449

<table>
<thead>
<tr>
<th>?actor</th>
<th>?loc</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://mdb.../Ric">http://mdb.../Ric</a></td>
<td><a href="http://geo.../Rome">http://geo.../Rome</a></td>
</tr>
</tbody>
</table>
Properties of Separated Execution

- **Advantage**: straightforward to implement
  - Can be combined with any source selection strategy
  - A traditional query execution plan might then be used for constructing the result

- **Downside**: First solutions cannot be reported before data retrieval has been completed
Properties of Separated Execution

- Advantage: straightforward to implement
  - Can be combined with any source selection strategy
  - A traditional query execution plan might then be used for constructing the result

- Downside: First solutions cannot be reported before data retrieval has been completed
Integrated Execution Approaches

... intertwine data retrieval and result construction

Query-local data

GET http://.../movie2449

<table>
<thead>
<tr>
<th>?actor</th>
<th>?loc</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://mdb.../Ric">http://mdb.../Ric</a></td>
<td><a href="http://geo.../Rome">http://geo.../Rome</a></td>
</tr>
</tbody>
</table>
Integrated Execution Approaches

... intertwine data retrieval and result construction

- Implementations may report first solutions early
  - For monotonic queries
- Implementations may process data in a streaming manner
  - May require less query-local memory (b/c query-local dataset need not be materialized)
- Can also be combined with any source selection strategy
Push-Based Implementation

- Each operator runs as an asynchronous thread
- Connected via message queues
Push-Based Implementation

- Each operator runs as an asynchronous thread
  - Connected via message queues
- Data retrieval operator
  - May apply any possible source selection strategy
  - Usually, implemented via multiple lookup threads
  - Pushes any incoming matching triple to the corresponding triple pattern operator
Push-Based Implementation (Example)

Join Operator

TP Operator
( ?i, rdf:type, Book )

TP Operator
( ?p, interested in, ?i )

TP Operator
( ?p, affiliated with, orgax )

Join Operator

Data Retrieval Operator
Push-Based Implementation (Example)

TP Operator
( ?p, interested in, ?i )

TP Operator
( ?p, affiliated with, orgax )

Join Operator

Join Operator

{ ?p \rightarrow alice }

TP Operator
( ?i, rdf:type, Book )

Data Retrieval Operator
( alice, affiliated with, orgax )
Push-Based Implementation (Example)

TP Operator
( ?p, interested in, ?i )

TP Operator
( ?p, affiliated with, orgax )

Join Operator
{ ?p → alice, ?i → yoga }

Join Operator
{ ?p → alice, ?i → yoga }

Join Operator
{ ?p → alice }

TP Operator
( ?i, rdf:type, Book )

Data Retrieval Operator
( alice, interested in, yoga )
Link Traversal Based Query Execution

... is the combination of integrated execution and live exploration

(i.e., the push-based approach can be used as an implementation of LTBQE)
Outline

- **Theoretical Foundations**
  - Data Model ✓
  - Full-Web Query Semantics ✓
  - Reachability-based Query Semantics ✓

- **Execution of Linked Data Queries**
  - Source Selection Strategies ✓
  - Execution Process ✓
  - Source Ranking

- **Conclusions**
General Idea of Source Ranking

Rank the URIs resulting from source selection such that the ranking represents a priority for lookup

• Possible objectives:
  • Report first solutions as early as possible
  • Minimize time for computing the first $k$ solutions
  • Maximize the number of solutions computed in a given amount of time
Source Ranking for Live Exploration

• More challenging than for index-based source selection:
  • New URIs for lookup are discovered continuously
  • Initially, no information about (reachable) LD documents

• Idea:
  • Predict relevance of reachable LD documents using vertex scoring methods (e.g., in-degree, PageRank)
Are Vertex Scoring Methods Suitable?

- **Methodology:**
  - Execute 6 queries in different simulated Webs of Linked Data
  - For each execution, reconstruct link graph of the reachable subweb and identify relevant vertices (i.e., LD documents)
  - Compute score for each vertex, and normalize to \([0,1]\)
  - Let \(\text{avg}\) and \(\text{avg}_{rel}\) be the average of these normalized scores for all vertices and for all relevant vertices, respectively
  - Let \(\text{delta} = \text{avg}_{rel} - 0.5\) (i.e., difference to the center of \([0,1]\))

- **Scoring method is unsuitable if**
  - Distance between \(\text{avg}\) and \(\text{avg}_{rel}\) is insignificant (\(\text{dist} < 0.25\))
  - \(\text{delta}\) is insignificant (e.g., \(-0.1 < \text{delta} > 0.1\))
Is PageRank Suitable?

- Either insignificant dist or delta, or too much variation
- Same for other well-known scoring methods
  - HITS (auth. and hub)
  - betweenness centrality
  - $k$-step Markov (tested with $k=6$)
  - in-degree
New Scoring Methods

• Example:
  • For each vertex $d$, let $IN_2(d)$ be the set of vertices such that for each $d' \in IN_2(d)$ there exists a path of length $\leq 2$ to $d$
  • Let $score(d)$ be the number of relevant vertices in $IN_2(d)$
Outline

- Theoretical Foundations
  - Data Model ✓
  - Full-Web Query Semantics ✓
  - Reachability-based Query Semantics ✓

- Execution of Linked Data Queries
  - Source Selection Strategies ✓
  - Execution Process ✓
  - Source Ranking ✓

- Conclusions
Future Work

- **Source ranking for live exploration**
  - Are scoring methods still suitable for partial link graphs?
  - Use (promising) scoring method for adaptive source ranking

- **More expressive SPARQL-based Linked Data queries**
  - Property paths
  - Entailment regimes

- **Combine LTBQE with distributed query execution**
  - Some Linked Data publishers expose query processing functionality via a Web service (a.k.a. “SPARQL endpoint”)
  - Making use of such functionality is likely to improve efficiency
These slides have been created by Olaf Hartig

http://olafhartig.de

This work is licensed under a Creative Commons Attribution-Share Alike 3.0 License (http://creativecommons.org/licenses/by-sa/3.0/)