By considering the WWW as the ultimate platform, we have advanced the state of the art of decentralized data management with a focus on *discoverability, interoperability, and automation*

These aspects are a strength to be pursued further.
Reflections on Linked Data Querying and other Related Topics

Olaf Hartig
Dept. of Computer and Information Science, Linköping University, Sweden
Amazon Neptune, AWS

http://olafhartig.de  @olafhartig
Terminology

“Linked Data”?
Data on the Web

➢ Traditionally published in HTML documents that are designed for human consumption
Linked Data Publishing Principles

Goal: publishing data on the WWW in a structured form that is standardized and machine-readable

- data model: RDF
- global identifiers: URIs
- access protocol: HTTP

Movie Database

CIA World Factbook Database
The Web of Linked Data

Three types of graphs

- RDF graphs within each such Linked Data document
- Graph of links between these documents
- Graph of links between datasets

( http://...imdb.../WarChild , http://.../releaseDate , 12 July 1999 )
( http://...imdb.../WarChild , http://.../director , http://...imdb.../MichaelDavie )

( http://cia.../Albania , http://.../unemploymentRate , 13.2% )
The Web of Linked Data

Three types of graphs

➢ RDF graphs within each such Linked Data document
➢ Graph of links between these documents
➢ Graph of links between datasets

Image source: https://lod-cloud.net/
Can we enable applications to query such a Web of interlinked data in a declarative manner?
Research Question

Can we enable applications to query such a Web of interlinked data in a declarative manner, without relying on a centralized copy?
Research Question

Can we enable applications to query such a Web of interlinked data in a declarative manner, without relying on a centralized copy?

Why not relying on a centralized copy?

- **Legal issues** may prevent storing copies of some of the data
- Hypothesis: massive adoption of the Linked Data principles, and not only for static datasets
  - creating and maintaining a centralized copy would become *impractical*, in particular for small organizations and individuals
  - copied data becomes *outdated*
  - new data sources *not captured*

Image source: https://lod-cloud.net/
Research Question

Can we enable applications to query such a Web of interlinked data in a declarative manner, without relying on a centralized copy, and without focusing only on a predetermined set of known data sources?
Linked Data Querying
Ingredients of Query Execution Processes

1. URI lookup component
   - retrieves data used for producing the query result
   - decides which URIs to look up (“source selection”)
   - ... and in which order (“source ranking”)

2. Result construction component
   - produces and combines intermediate solutions of the query result incrementally, based on the retrieved data

• Both components may be fused into a combined process
  - e.g., URI lookup decisions may depend on intermediate results
Ingredients of Query Execution Processes

1. URI lookup component
   - retrieves data used for producing the query result
   - decides which URIs to look up (“source selection”)
   - ... and in which order (“source ranking”)

2. Result construction component
   - produces and combines intermediate solutions of the query result incrementally, based on the retrieved data

• Both components may be fused into a combined process
  - e.g., URI lookup decisions may depend on intermediate results

• Traversal of data links can be integrated into this process to actively discover potentially relevant URIs to look up
  - traversal may be selective and query specific
  - “traversal-based query execution”
Traversal-Based Query Execution (Example)

- Traversal of data links can be integrated into this process to actively discover potentially relevant URIs to look up
  - traversal may be selective and query specific
  - “traversal-based query execution”
Traversal-Based Query Execution (Example)

- Traversal of data links can be integrated into this process to actively discover potentially relevant URIs to look up
  - traversal may be selective and query specific
  - “traversal-based query execution”
Traversals of data links can be integrated into this process to actively discover potentially relevant URIs to look up:

- traversal may be selective and query specific
- "traversal-based query execution"
Traversal-Based Query Execution (Example)

- Traversal of data links can be integrated into this process to actively discover potentially relevant URIs to look up
  - traversal may be selective and query specific
  - “traversal-based query execution”
**Traversal-Based Query Execution (Example)**

- Traversal of data links can be integrated into this process to actively discover potentially relevant URIs to look up
  - traversal may be selective and query specific
  - “traversal-based query execution”
Reflections on Linked Data Querying and other Related Topics

Olaf Hartig – @olafhartig

1. URI lookup component
- retrieves data used for producing the query result
- decides which URIs to look up (“source selection”)
- … and in which order (“source ranking”)

2. Result construction component
- produces and combines intermediate solutions of the query result incrementally, based on the retrieved data

Both components may be fused into a combined process – e.g., URI lookup decisions may depend on intermediate results

Observe, retrieved data serves two purposes:
1. produce the query result
2. discover further URIs
Techniques for Traversal-Based Query Engines

• Link-traversing iterators
  − URI lookups are a side effect of calling such an iterator
  − Extension of the iterator paradigm to enable an adaptive execution that avoids blocking when waiting for lookups
  − Different join orders are not semantically equivalent
  − Heuristics-based plan selection

• Fully adaptive approach based on a tuple-routing operator

• Prioritizing URI lookups to reduce the response times

O Hartig, C Bizer, and J-C Freytag: Executing SPARQL Queries over the Web of Linked Data. ISWC 2009.
O Hartig and T Özsu: Walking without a Map: Ranking-Based Traversal for Querying Linked Data. ISWC 2016
Linked Data Query Processing Approaches

- index-based systems
  - Some form of **complete** catalog of data sources assumed (e.g., data summaries, statistics)

- traversal-based systems
  - No information assumed prior to query execution

---

A Harth, K Hose, M Karnstedt, A Polleres, K-U Sattler, and J Umbrich: *Data Summaries for On-Demand Queries over Linked Data*. WWW 2010.

J Umbrich, K Hose, M Karnstedt, A Harth, and A Polleres: *Comparing Data Summaries for Processing Live Queries over Linked Data*. WWWJ 14(5-6), 2011.


E Paret, W Van Woensel, S Casteleyn, B Signer, and O De Troyer: *Efficient Querying of Distributed RDF Sources in Mobile Settings based on a Source Index Model*. ANT/MobiWIS 2011.


O Hartig, C Bizer, and J-C Freytag: *Executing SPARQL Queries over the Web of Linked Data*. ISWC 2009.


O Hartig and T Özsu: *Walking without a Map: Ranking-Based Traversal for Querying Linked Data*. ISWC 2016


Linked Data Query Processing Approaches

index-based systems

Some form of complete catalog of data sources assumed (e.g., data summaries, statistics)

hybrids

No information assumed prior to query execution

traversal-based systems

A Harth, K Hose, M Karnstedt, A Polleres, K-U Sattler, and J Umbrich: Data Summaries for On-Demand Queries over Linked Data. WWW 2010.

J Umbrich, K Hose, M Karnstedt, A Harth, and A Polleres: Comparing Data Summaries for Processing Live Queries over Linked Data. WWWJ 14(5-6), 2011.

Y Tian, J Umbrich, and Y Yu: Enhancing Source Selection for Live Queries over Linked Data via Query Log Mining. JIST 2011.

E Paret, W Van Woensel, S Casteleyn, B Signer, and O De Troyer: Efficient Querying of Distributed RDF Sources in Mobile Settings based on a Source Index Model. ANT/MobiWIS 2011.


O Hartig, C Bizer, and J-C Freytag: Executing SPARQL Queries over the Web of Linked Data. ISWC 2009.


O Hartig and T Özsu: Walking without a Map: Ranking-Based Traversal for Querying Linked Data. ISWC 2016


“Completeness!”?

Index-based systems

Some form of complete catalog of data sources assumed (e.g., data summaries, statistics)

Hybrids

No information assumed prior to query execution

Traversal-based systems

A Harth, K Hose, M Karnstedt, A Polleres, K-U Sattler, and J Umbrich: Data Summaries for On-Demand Queries over Linked Data. WWW 2010.

J Umbrich, K Hose, M Karnstedt, A Harth, and A Polleres: Comparing Data Summaries for Processing Live Queries over Linked Data. WWWJ 14(5-6), 2011.

Y Tian, J Umbrich, and Y Yu: Enhancing Source Selection for Live Queries over Linked Data via Query Log Mining. JIST 2011.

E Paret, W Van Woensel, S Casteleyn, B Signer, and O De Troyer: Efficient Querying of Distributed RDF Sources in Mobile Settings based on a Source Index Model. ANT/MobiWIS 2011.


O Hartig, C Bizer, and J-C Freytag: Executing SPARQL Queries over the Web of Linked Data. ISWC 2009.


Foundations of Linked Data Queries

- Model of a Web of Linked Data
- Different evaluation semantics for different fragments of SPARQL over this model
- Computational properties of such queries
- LDQL, a query language specifically designed for queries over Linked Data

<table>
<thead>
<tr>
<th>SPARQL_{LD}</th>
<th>SPARQL_{LD(R)} (reachability criterion does not ensure finiteness)</th>
<th>SPARQL_{LD(R)} (reachability criterion ensures finiteness)</th>
</tr>
</thead>
<tbody>
<tr>
<td>- unsatisfiable</td>
<td>all</td>
<td>all</td>
</tr>
<tr>
<td>- satisfiable, monotonic</td>
<td>all</td>
<td>some</td>
</tr>
<tr>
<td>- non-monotonic</td>
<td>all</td>
<td>some</td>
</tr>
<tr>
<td>- unsatisfiable</td>
<td>all</td>
<td>some</td>
</tr>
<tr>
<td>- satisfiable, monotonic</td>
<td>some</td>
<td>some</td>
</tr>
<tr>
<td>- non-monotonic</td>
<td>all</td>
<td>all</td>
</tr>
<tr>
<td>- unsatisfiable</td>
<td>all</td>
<td>all</td>
</tr>
<tr>
<td>- satisfiable, monotonic</td>
<td>all</td>
<td>all</td>
</tr>
<tr>
<td>- non-monotonic</td>
<td>all</td>
<td>all</td>
</tr>
</tbody>
</table>

Table 4.1.: Correlation between (LD-machine-based) computability and basic properties for all SPARQL-based Linked Data queries considered in this dissertation.


Main Innovation

- Discovery of data sources as an integral part of the runtime processes of a query engine
- Automatic utilization of such discovered data sources within the query execution process

<table>
<thead>
<tr>
<th>SPARQL&lt;sub&gt;LD&lt;/sub&gt;</th>
<th>finitely computable</th>
<th>event. computable</th>
<th>not even event. comp.</th>
<th>Corresponding result</th>
</tr>
</thead>
<tbody>
<tr>
<td>- unsatisfiable</td>
<td>all</td>
<td>all</td>
<td>all</td>
<td>Proposition 2.3, p.29, Theorem 3.2, p.53</td>
</tr>
<tr>
<td>- satisfiable, monotonic</td>
<td></td>
<td></td>
<td></td>
<td>Theorem 4.3, p.91</td>
</tr>
<tr>
<td>- non-monotonic</td>
<td></td>
<td></td>
<td></td>
<td>Theorem 4.3, p.91</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SPARQL&lt;sub&gt;LD&lt;/sub&gt;(R) (reachability criterion does not ensure finiteness)</th>
<th>finitely computable</th>
<th>event. computable</th>
<th>not even event. comp.</th>
<th>Corresponding result</th>
</tr>
</thead>
<tbody>
<tr>
<td>- unsatisfiable</td>
<td>all</td>
<td>some</td>
<td>some</td>
<td>Proposition 2.3, p.29</td>
</tr>
<tr>
<td>- satisfiable, monotonic</td>
<td></td>
<td>some</td>
<td>some</td>
<td>Theorem 4.3, p.91</td>
</tr>
<tr>
<td>- non-monotonic</td>
<td></td>
<td>some</td>
<td>some</td>
<td>Theorem 4.3, p.91</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SPARQL&lt;sub&gt;LD&lt;/sub&gt;(R) (reachability criterion ensures finiteness)</th>
<th>finitely computable</th>
<th>event. computable</th>
<th>not even event. comp.</th>
<th>Corresponding result</th>
</tr>
</thead>
<tbody>
<tr>
<td>- unsatisfiable</td>
<td>all</td>
<td>all</td>
<td>all</td>
<td>Proposition 2.3, p.29</td>
</tr>
<tr>
<td>- satisfiable, monotonic</td>
<td>all</td>
<td>all</td>
<td>all</td>
<td>Proposition 4.12, p.91</td>
</tr>
<tr>
<td>- non-monotonic</td>
<td>all</td>
<td>all</td>
<td>all</td>
<td>Proposition 4.12, p.91</td>
</tr>
</tbody>
</table>
Reflections and Outlook
No Adoption in Practice – Why?

1. Massive amounts of data links? No 😞
   - links between datasets have been sparse
   - hardly any bidirectional linking between datasets
   - little reuse of URIs across datasets

2. Behavior of Linked Data servers under-specified
   - what exactly can I receive when looking up a URI?

3. Query execution times not competitive for interactive applications
   - options for parallel fetching inherently limited
   - over-fetching and under-fetching
Should we give up on this line of research?

1. Massive amounts of data links? No 😞
   - links between datasets have been sparse
   - hardly any bidirectional linking between datasets
   - little reuse of URIs across datasets

2. Behavior of Linked Data servers under-specified
   - what *exactly* can I receive when looking up a URI?

3. Query execution times not competitive for interactive applications
   - options for parallel fetching inherently limited
   - over-fetching and under-fetching

• No: capability to actively discover initially unknown data sources and to automatically interoperate with them is valuable

• Yes: give up trying to rely on the LD principles as a basis for developing some form of federated query processors
Landscape of RDF Federation Approaches

- Index-based systems
- Traversal-based systems
- Hybrids

Some form of complete catalog of data sources assumed (e.g., data summaries, statistics)

- DARQ
- FedX, Lusail

No information assumed prior to query execution

Linked Data

SPARQL endpoints

ANAPSID, SPLENDID, SemaGrow, etc.
Landscape of RDF Federation Approaches

Linked Data

traversal-based systems

index-based systems

hybrids

Some form of complete catalog of data sources assumed (e.g., data summaries, statistics)

No information assumed prior to query execution

TPF          brTPF          SPF       …

SmartKG      WiseKG

ANAPSID, SPLENDID, SemaGrow, etc.

FedX, Lusail

DARQ

SPARQL endpoints

Reflections on Linked Data Querying and other Related Topics
Olaf Hartig – @olafhartig
Query engines should become more flexible!

- Flexibility in the horizontal dimension
  - i.e., capability to interact with multiple types of (query-based) data access interfaces
  - perhaps even a capability to adapt to new types of interfaces at runtime

- Flexibility in terms of new federation members
  - i.e., integrate new federation members at runtime
  - not only passively, but actively (i.e., adapt traversal-based execution, not to produce query results but to discover new federation members at runtime)
Query engines should become more flexible!

- Flexibility also the vertical dimension
  - i.e., different types of summaries/stats in the catalog, depending on type of federation member
  - capability to operate with an incomplete catalog
  - capability to populate and update the catalog at runtime (e.g., as a side effect of query processing)

- All of this in a well-defined manner!
Further Aspects for Future Work

- More attention to joins on literals, including similarity joins
- Schema heterogeneity
- Provenance of query results
- Access control
- Optimization criteria other than total query execution time
- Suitable methods for evaluation and comparison
- Efficiency of servers
- Read & write with transactional guarantees
Concluding Remarks

• Learned that some ideas did not work out
  – assumptions were too optimistic
  – technical limitations

• Better understanding of problems and solutions for decentralized data management
  – discoverability, interoperability, and automation

• Continue challenging the assumption that we can start with (or ever achieve) a complete catalog in a decentralized data architecture

• Advertisement: ISWC 2022 DC, deadline May 13
These slides have been created by Olaf Hartig. Feel free to reuse them under the terms of the Creative Commons BY SA license.

https://creativecommons.org/licenses/by-sa/4.0/

www.liu.se