Dissertation Summary
Storing and Querying Evolving Knowledge Graphs on the Web

Ruben Taelman

1. Motivation and Objectives

The Web has a massive and positive impact on society. It serves as a globally interconnected platform where anyone can say anything about anything, and anyone can access anything that has been said. As such, the ability to *publish* things on the Web, and the ability for others to *retrieve* them are crucial. The Semantic Web initiative was initiated as a way for humans to understand the Web in a similar way as humans do, which relies on self-descriptive and machine-processable data.

One of the foundational building blocks of the Semantic Web is the Resource Description Framework (RDF), which allows statements about things to be represented in the form of graphs. Following the Linked Data principles, such RDF data can be published on the Web, for others to consume. While RDF provides a robust foundation for representing data, one of the primary restrictions of RDF is that it is *atemporal*. This is, unfortunately, a mismatch with how data on the Web behaves in practise, which is highly volatile and continuously evolving. In order to properly handle such *evolving* data on the Web, research and engineering effort is needed for new models, storage techniques, and query algorithms for evolving data in the Semantic Web. With this, it is important to keep in mind the *decentralized* nature of the Web, where anyone should be able to publish and retrieve data. Publishing techniques for evolving data should consider affordable mechanisms for the average person, and retrieval techniques should consider the data *spread* data over multiple *heterogeneous* sources.

The goal of my research is to allow *publishing* and *retrieval* data on the Web without having to depend on costly or centralized infrastructure, with a focus on knowledge that evolves over time. This lead me to the following research question for my PhD:

"**How to store and query evolving data on the Semantic Web?**"

In my dissertation, I focus on four main challenges related to this:

1. **Experimentation requires *representative* evolving data.**
   In order to *evaluate* the performance of systems that handle *evolving* data, a flexible method for *obtaining* such data needs to be available.
2. **Indexing evolving data involves a *trade-off* between *storage efficiency* and *lookup efficiency*.**
   Indexing techniques are used to improve the efficiency of querying, but comes at the cost of increased storage space and preprocessing time. As such, it is impor-

tant to find a good *balance* between the amount of *storage* space with its indexing time, and the amount of *querying speedup*, so that evolving data can be stored in a Web-friendly way.

3. **Web interfaces are highly *heterogeneous*.**
Since data can be exposed through different types of *interfaces*, querying data from the Web involves different *algorithms* that allow retrieval from combinations of such interfaces.

4. **Publishing *evolving* data via a *queryable interface* involves *continuous* updates to clients.**
Centralized query interfaces are hard to scale for an increasing number of concurrent clients, especially when the datasets that are being queried over are continuously evolving, and clients need to be notified of data updates continuously. New kinds of interfaces and querying algorithms are needed to cope with this scalability issue.

These research challenges correspond to the four main chapters of my dissertation.

2. Evaluation Methods

I follow an experimental performance approach for evaluating my approaches based on empirical measurements on implementations. This involves defining statistical hypotheses of the expected outcomes, designing factorial experiments with factors that could potentially influence performance, executing the experiments in a fully automated manner, and validating the hypotheses via statistical validation techniques.
In order to make these experiments reproducible, and making implementations reusable for other researchers, I spend extra effort in making these implementations highly qualitative. I do this by following well-established development methods from the software industry, perform rigorous testing, and writing detailed documentation. All my implementations and experiments are available online under an open license, are being actively maintained and receive regular community contributions. As a consequence of this, these implementations are not only being used for research, but also by various companies in production environments.

3. Contributions, Approach and Results

Corresponding to the four main challenges of my dissertation, my dissertation contains four key contributions, which I elaborate on after this listing:

- **Generating Synthetic Evolving Data**: Mimicking algorithm for generating realistic synthetic evolving datasets, based on public transport networks.
- **Storing Evolving Data**: Versioned triple storage technique with highly performant streaming querying algorithms, for storing evolving datasets.
- **Querying a heterogeneous Web**: Flexible and highly modular query platform for simplifying the research and development of query interfaces and algorithms.
- **Querying Evolving Data**: Publishing and querying techniques for evolving datasets that can scale to a large number of concurrent clients.

3.1. Generating Synthetic Evolving Data

As a prerequisite to my next challenges, there was a need for realistic evolving data in different sizes and with different properties, so that storage and querying systems for evolving data can be tested properly. Ideally, real-world datasets should be used, as these can show the true performance of such systems in various circumstances. However, these real-world datasets have limited public availability, and do not allow for the required flexibility when evaluating systems.
As such, in this challenge I focused on the generation of realistic evolving data. As a target domain, I chose for public transport networks, due to it having a non-trivial data model that contains both temporal and geospatial features. The novelty of the approach is that it is based on established concepts from the domain of public transport networks design, is highly configurable (50+ parameters), and takes population distributions as input to generate realistic transport networks. The experimental results (Section 2.7) based on my implementation (PoDiGG) show that the algorithm can indeed produce datasets of a high level of realism, which is measured through a set of distance functions from real-world datasets. Furthermore, the implementation is able to produce these datasets in an efficient manner in terms of CPU and memory usage.

3.2. Storing Evolving Data

Adding a temporal dimension to RDF adds a new level of complexity to storage and querying systems. A naive way to handle this temporal dimension would be to store each dataset version as a separately materialized dataset. This can however introduce a tremendous storage overhead when consecutive versions are similar to each other. Furthermore, querying over such a naive storage method would require going through all of these versions, which does not scale well with many versions.
My goal in this challenge was to come up with a Web-friendly trade-off between storage size and lookup efficiency, so that evolving datasets can be published on the Web without requiring high-end machines. I introduce a new indexing technique for evolving data, that focuses on querying in a stream-based manner. This allows results to be sent to the client from the moment that they are found, which reduces waiting time compared to batch-based querying. Streaming is especially useful when the number of results is very large, and is memory friendly for machines with limited capabilities. Based on my implementation (OSTRICH), experiments show (Section 3.8.3) that this system achieves lower overall query execution times for the different temporal query types compared to alternative systems. This level of performance comes at the cost of a slight increase in storage size compared to other systems.

3.3. Querying a heterogeneous Web

Since the Web enables publishing data on the Web, we can find a large heterogeneity in publishing interfaces for RDF on the Web, such as data dumps, SPARQL endpoints, Linked Data subject pages, Triple Pattern Fragments, and more. While systems and algorithms exist to enable clients to query from each of these interfaces separate-

ly, it requires a large overhead from the user to handle querying over these interfaces with different systems. Furthermore, no straightforward manner was available to query over combinations of data spread over these interfaces via federated querying.
The goal in this challenge was to tackle this problem of heterogeneity in terms of interfaces and algorithms by providing a highly modular query engine architecture in which support for different interfaces and algorithms can be added *independently*, where these modules can be plugged in when they are needed. This system simplifies the research and development of new query interfaces and algorithms, as new techniques can be tested immediately in conjunction with other already existing interfaces and algorithms. Experimental results (Section 4.6) of my implementation (Comunica) show that this high modularity and flexibility does not lead to a significant reduction of performance.

3.4. Querying Evolving Data

Research has shown that publishing RDF on the Web via a queryable interface at a high availability can become hard for the data publisher. If publishers want to expose their data with a high availability publicly on the Web, they need to consider a potentially high number of clients that may concurrently execute queries. When this RDF data becomes *temporal*, this cost can even become higher, since queries may require repeated or continuous execution, as the underlying data is evolving.
As such, the goal in this challenge was to come up with a publishing and querying mechanism that reduces the cost of publishing evolving data on the Web, by moving part of the query execution to the client-side. The approach involves annotating evolving data with a description of their volatility. Based on these descriptions, clients can detect for how long parts of the knowledge graph will remain valid via a novel querying algorithm, and when new queries need to be initiated to calculate next up-to-date results. Compared to similar systems, my implementation (TPF-QS) scales significantly better to a large number of clients (Section 5.7.3).

4. Open Issues and Future Directions

In this section, I list the key opportunities for future work that I consider the most important in line with my dissertation.

Alternative Storage Trade-offs

In this work, I focused on providing a storage mechanism with useful trade-offs for Web-friendly publishing on average Web servers. However, alternative techniques for storing evolving datasets with different trade-offs will be required for different scenarios, such as low-end devices in the IoT domain.

Semantic Querying

Current storage solutions mainly focus on the syntactical querying over evolving data, but they do not yet consider the issue of *semantic querying*, which involves taking into account the meaning of things through ontology-based inferencing. Semantic

querying over evolving data is needed to enable semantic analysis over such knowledge, such as analyzing concept drift or tracking diseases in biomedical datasets over time.

Standardization

Standardization efforts will be needed to truly bring evolving data to the Web, in the form of temporal query languages, temporal models and exchange formats. For the sake of compatibility, these should be extensions or they should be representable in the existing Linked Data stack, which will mainly impact RDF and SPARQL. The W3C RDF Stream Processing community group is a first effort that aims to explore these issues.

Decentralization

More work will be needed to solve open problems with decentralized data, with the Solid ecosystem being an important driver within this effort. Concretely, new techniques and algorithms are needed to (1) intelligently navigate the the Web by following relevant links for a given query, (2) enable efficient querying over a large number of sources, (3) allow authentication-aware querying over private data, and (4) support collaborative querying between agents that handle similar queries.

Non-technical Challenges

Next to technical issues, organizational and societal changes will also be needed. For instance, the European General Data Protection Regulation places strict demands on companies that handle personal data. Decentralization efforts such as Solid are being investigated by organizations such as governments to reshape the relationship with their citizens, by giving people true ownership over their data, and making governments data consumers.

5. Conclusions

A first significance of my work are the algorithms and system architectures to store and query evolving data in a Web-friendly manner. Even though there is a large amount of evolving data on the Web and a need to handle it, this research area receives too little attention within the Semantic Web community. As such, I contributed to filling this void through my dissertation.

A second significance is the reusability of my approaches and implementations in the research domain and in industry. For instance, the Comunica query engine has been the basis for tutorials at ISWC and ESWC, which has led to its extensive usage within fundamental querying research, research projects, and companies. At the time of writing, Comunica receives an average of 1000 weekly downloads, with more than 120 third-party open-source software projects depending on it. As Comunica is the only engine in existence that is able to query over multiple heterogeneous sources from within a Web browser, it is the go-to tool for querying over decentralized Solid data pods, for which I am collaborating with Inrupt and Tim Berners-Lee to develop new query algorithms to handle the massive scale of data distribution that Solid causes. As such, Comunica has become a crucial element in the Web decentralization effort.