On Combining RDF Streams and Remotely Stored Background Data

Soheila Dehghanzadeh[1], Daniele Dell'Aglio[2], Shen Gao[3], Emanuele Della Valle[2], Alessandra Mileo[1], Abraham Bernstein[3]

[1]INSIGHT Research Center, NUI Galway, Ireland
[2]DEIB, Politecnico of Milano, Italy
[3]Department of Informatics, University of Zurich, Switzerland

To perform complex tasks, continuous queries in RDF Stream Processing can combine RDF streams and quasi-static background data, as depicted in Figure 1. While the former is pushed in the query engine, the latter is pulled from the repository. In the following, we focus on the cases where background data is stored remotely and can be accessed through the `SERVICE` clause. That means, background data are retrieved from SPARQL endpoints, or from Web APIs through OBDA techniques.

Existing engines usually retrieve the background data at each new iteration. But as soon as the background data volume increases, the cost to retrieve them increases as well, and consequently engines are at the risk of becoming unresponsive. For example, the C-SPARQL engine delegates the evaluation of the SPARQL operators to the ARQ engine[1]: `SERVICE` clauses are managed through sequences of invocations to the remote endpoints. In this way, they generate high loads on remote services and suffer from slow response time.

Instead of pulling data from remote SPARQL endpoints at each evaluation, a possible solution is to store the intermediate results of SERVICE clauses in *local views* inside the query processor. This idea is widely adopted in databases to improve the performance, availability, and scalability of the query processor [2]. However, the freshness of the local view degrades over time because background data in the remote service can change and the updates are not reflected in the local view. To overcome this issue, a *maintenance process* is introduced: it identifies the out-dated (namely *stale*) data items in the local view and replaces them with the up-to-date (namely *fresh*) values retrieved from the remote services.

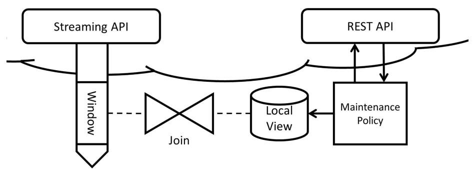

Fig. 1: Query Processor Overview

In this work, we present the requirements that leads the design of maintenance processes (MP), to help researchers and developers of the RSP community in defining new solutions and optimizing existent RSP engines.

Analysis of the Problem. Database community widely studied the local view maintenance problem; however, the Web setting introduces new challenges, which we present in the following. The main differences are 1) data is owned by

[1] C-SPARQL Version 0.48; ARQ Version 2.11.1

different entities in the distributed setting; 2) there are no update streams – streams bringing content changes.

R1: Data Access Constraints. When data are retrieved from Web APIs, MP should take into account the remote service constraints. In particular, MP should consider the **data access patterns (R1.1)**, i.e., it is only possible to access data through sets of pattern that conform to the API definitions; the **access rate (R1.2)**, i.e., there are time constraints on the pulling requests that can be submit.

R2: Quality of Service (QoS) Constraints. The introduction of local views that become stale causes approximate results. Users can be interested in supplying several QoS constraints, such as query **response time** and the **accuracy** of the answer. The scope and frequency of the MP should take into account those **QoS constraints (R2.1)**. However, there are cases where it is not possible to achieve the goal: when it happens, the maintenance process should **raise alerts to the query processor (R2.2)**

R3: (Dynamic) Change Rate Distribution. The MP should consider the fact that different elements in background data change over time at **different rates (R3.1)**, e.g., the follwer number of a famous music artist changes more often than that of a PhD student. Moreover, the change rate of elements can vary over time, e.g., the follwer number of a user changes more often during Twitter activity peeks and less often otherwise. Therefore, the MP may consider the **dynamic change rates (R3.2)**.

R4: Query Features. Joins may pair streaming and background data. The MP may exploit this unique feature to optimize the maintenance. Specifically, the MP may exploit the **sliding window definition (R4.1)**: at each evaluation, part of the window content does not change (as the window slides), which can be used to select the local view elements to be maintained. The MP may also consider the **join selectivity (R4.2)** to find the most influential data elements on the response accuracy.

Initial Solution and Results. In [1] we proposed the *Window-Based Maintenance policy* (WBM), an initial solution to the problem presented above. WBM is a query-driven maintenance policy, which selects the mappings involved in the current query, from which it picks the ones to be refreshed. The intuition behind WBM is to prioritize the refresh of the mappings that are going to be used in the upcoming evaluations and that allows saving future refresh operations. WBM considers the locality effect created by the window and identifies expired data more precisely by estimating their change rates. Our initial evaluation shows that our solution outperforms a set of baselines by up to 13.5%.

References

1. S. Dehghanzadeh, D. Dell'Aglio, S. Gao, E. Della Valle, A. Mileo, and A. Bernstein. Online view maintenance for continuous query evaluation. In *WWW 2015 Companion Volume II (to appear)*, pages 1–2, 2015.
2. H. Guo, P.-Å. Larson, and R. Ramakrishnan. Caching with good enough currency, consistency, and completeness. In *VLDB*, pages 457–468. VLDB Endowment, 2005.

Authors

Soheila Dehghanzadeh is studying in National University of Ireland, Galway. Her research is focused on query optimziation and view management.

Daniele Dell'Aglio is a PhD student at DEIB, Politecnico di Milano and he works on optimization of the continuous query answering process in presence of volatile and heterogeneous data.

Shen Gao is a PhD student at the Department of Informatics of the University of Zurich. His research focuses on scalable RDF stream processing.

Emanuele Della Valle is assistant professor at Politecnico di Milano. He coined the term Stream Reasoning, and he showed that it is possible with the Continuous SPARQL query language (C-SPARQL), the C-SPARQL Engine and the Incremental Materialization for RDF Stream algorithm (IMaRS). He applied Stream Reasoning to social media analytics.

Dr. Alessandra Mileo is a senior Research Fellow heavily involved in research and development activities that are relevant to the RSP group, including stream reasoning, query optimization and event processing, and she is PI in EU funded smart city projects and industry funded projects in the area of the Internet of Everything.

Abraham Bernstein is a full Professor at the Department of Informatics (Institut für Informatik) of the University of Zurich. He mainly conducts research on the Semantic Web and Knowledge Discovery. His work draws from both social science (organizational psychology/sociology) and technical (computer science, artificial intelligence) foundations.