Comparing data summaries for processing live queries over Linked Data

Jürgen Umbrich · Katja Hose · Marcel Karnstedt · Andreas Harth · Axel Polleres

Received: 15 May 2010 / Revised: 30 October 2010 / Accepted: 21 December 2010 / Published online: 7 January 2011 © Springer Science+Business Media, LLC 2011

Abstract A growing amount of Linked Data—graph-structured data accessible at sources distributed across the Web—enables advanced data integration and decision-making applications. Typical systems operating on Linked Data collect (crawl) and pre-process (index) large amounts of data, and evaluate queries against a centralised repository. Given that crawling and indexing are time-consuming operations, the data in the centralised index may be out of date at query execution time. An ideal query answering system for querying Linked Data live should return current answers in a reasonable amount of time, even on corpora as large as the Web. In such a live query system source selection—determining which sources contribute answers to a query—is a crucial step. In this article we propose to use lightweight data summaries for determining relevant sources during query evaluation. We compare several data structures and hash functions with respect to their suitability for building such summaries, stressing benefits for queries that contain joins and require ranking of results and sources. We elaborate on join variants, join ordering and ranking. We analyse the different approaches theoretically and provide results of an extensive experimental evaluation.

Keywords index structures · Linked Data · RDF querying

J. Umbrich · M. Karnstedt · A. Polleres
Digital Enterprise Research Institute, National University of Ireland, Galway, Ireland

K. Hose
Max-Planck-Institut für Informatik, Saarbrücken, Germany

A. Harth
Institute AIFB, Karlsruhe Institute of Technology, Karlsruhe, Germany
e-mail: harth@kit.edu
1 Introduction

The recent developments around Linked Data have led to the exposure of large amounts of data on the Semantic Web amenable to automated processing in software programs [6]. Linked Data sources use RDF (the Resource Description Framework) in various serialisation syntaxes for encoding graph-structured data, and the Hypertext Transfer Protocol (HTTP) for data transfer. The Linked Data effort is part of a trend towards highly distributed systems, with thousands or potentially millions of independent sources providing small amounts of structured data that form a global data graph. Advanced data integration and decision-making applications require query processing over the global graph. We can distinguish two broad directions:

- **data warehousing or materialisation-based approaches (MAT)**, which collect the data from all known sources in advance and pre-process, index and store the combined data in a central database; queries are evaluated using the local database.

- **distributed query processing approaches (DQP)**, which parse, normalise and split the query into sub-queries, determine the sources containing results for sub-queries, and evaluate the sub-queries against the sources directly.

Most current approaches enabling query processing over RDF data, such as Semantic Web search engines [11, 16, 32, 51], are based on MAT. Centralised approaches provide excellent query response times due to extensive preprocessing carried out during the load and indexing steps, but suffer from a number of drawbacks. First, the aggregated data is never current as the process of collecting and indexing vast amounts of data is time-consuming. Second, from the viewpoint of a single query, MAT involves unnecessary data gathering, processing, and storage since large portions of the data might not be used for answering the particular query. Lastly, due to replicated data storage, data providers have to give up sole sovereignty on their data (e.g., they cannot restrict or log access anymore since queries are answered against a copy of the data).

On the other end of the spectrum, DQP approaches typically assume processing power attainable at the sources themselves, which could be leveraged in parallel for query processing. Such distributed or federated approaches [28] offer several advantages: the system is more dynamic with up-to-date data and new sources can be added easily without time lag for indexing and integrating the data, and the systems require less storage and processing resources at the site that issues the query. The potential drawback, however, is that DQP systems cannot give strict guarantees on query performance since the integration system relies on a large number of potentially unreliable sources.

DQP is a well-studied problem in the database community [42], however, existing DQP approaches do not scale above a relatively small number of sources and heavily exploit schema information typically not available on linked RDF data. Previous results for query processing over distributed RDF stores [57] assume, similar to approaches from the traditional database works, few endpoints with local query processing capabilities and sizable amounts of data, rather than many small Web resources only accessible via HTTP GET.

Currently only a few data sources on the Web provide full query processing capabilities (e.g., by implementing SPARQL [12, 56], a query language and protocol