Database Techniques for Linked Data Management

Andreas Harth
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
harth@kit.edu

Katja Hose
Max-Planck Institute for Informatics
Saarbrücken, Germany
khose@mpi-inf.mpg.de

Ralf Schenkel
Saarland University
Saarbrücken, Germany
schenkel@mmci.uni-saarland.de

ABSTRACT

Linked Data refers to data published in accordance with a number of principles rooted in web standards. In the past few years we have witnessed a tremendous growth in Linked Data publishing on the web, leading to tens of billions of data items published online. Querying the data is a key functionality required to make use of the wealth of rich interlinked data. The goal of the tutorial is to introduce, motivate, and detail techniques for querying heterogeneous structured data from across the web. Our tutorial aims to introduce database researchers and practitioners to the new publishing paradigm on the web, and show how the abundance of data published as Linked Data can serve as fertile ground for database research and experimentation. As such, the tutorial focuses on applying database techniques to processing Linked Data, such as optimized indexing and query processing methods in the centralized setting as well as distributed approaches for querying. At the same time, we make the connection from Linked Data best practices to established technologies in distributed databases and the concept of Dataspaces and show differences as well as commonalities between the fields.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services; H.2.4 [**Database Management**]: Systems

General Terms

Algorithms, Design, Experimentation, Theory

Keywords

Linked Data, Semantic Web, Dataspaces, query processing, distributed databases, RDF, SPARQL

SIGMOD '12, May 20–24, 2012, Scottsdale, Arizona, USA.

1. INTRODUCTION

There is a general trend towards distributed data management systems, as data generation, publication, maintenance and integration are increasingly distributed and carried out by different actors. Consider a typical data integration system optimized for a representative query load, which is implemented and maintained inside an organization or department, where there is full control over the sources. For Semantic Web applications that we consider, the data may originate from a multitude of sources scattered across organizations, and the target queries are less specified a priori; the system should allow for exploring the integrated datasets.

The defining feature of these systems are ad-hoc integration capabilities and means for incremental integration (so-called pay-as-you-go integration). Linked Data [3] and Dataspaces [8] are two prime examples of new architectures for such data management systems. The web community has always stressed the distributed aspects and designed systems that operate with little central control and without the need for coordination between providers of content. The database community, on the other hand, stressed the data aspects of systems, often in a controlled setting (in contrast to the uncoordinated open web environment). In this tutorial, we bring together these two philosophies and show how to apply database techniques to the large corpus of data that has been published on the web. Following the general sentiment in the Linked Data community, we emphasize the aspects related to interlinking and decentralized publication and consumption.

In recent years, we have witnessed a rapid increase in the amount of data published in accordance with Linked Data principles. The Linking Open Data cloud[1] – a graphical representation of data published as Linked Data on the web – now covers 295 sources with over 31 billion data items. The growing availability and use of Linked Data has led to an increasing need for systems that can store Linked Data and, more importantly, efficiently evaluate complex queries over large bodies of interlinked datasets. For instance, a user looking for information about an actor, e.g., Matt Damon, can use search over centrally indexed Linked Data and find links to LinkedMDB, DBpedia, Freebase and other sources, all providing information about the actor. More complex information needs are also possible. Consider a user who is interested in finding answers to complex queries such as "movies starring Matt Damon for which he also wrote the screenplay"[2]. Such an information need cannot be easily

[1] `http://lod-cloud.net/`
[2] One answer is "Good Will Hunting".

expressed with keywords in a search engine, but requires a structured query language.

In general, there are two basic approaches to enable query processing over Linked Data: first, downloading and storing the data in a centralized repository and second, applying distributed query processing techniques. Both approaches have been very active research topics in the recent past. We structure our tutorial accordingly; we start with presenting foundational aspects of Linked Data and then present research on these two architectural choices.

The outline of the tutorial is as follows:

- The first session (see Section 2) introduces basic concepts behind Linked Data together with the conventions that data publishers have to adhere to in order to allow clients to uniformly access the data. In the session we contrast the architectural principles of Linked Data with those of Dataspaces.
- The second session (see Section 3) gives an overview of techniques for storing and querying Linked Data in a centralized system. The focus of the session will be on storage structures, query optimization, and query processing in triple stores. The session also introduces the main aspects of other centralized approaches.
- The third session (see Section 4) gives an overview of state-of-the-art techniques for querying Linked Data in distributed environments. In this session, we point out how these techniques relate to database architectures and focus on how to answer structured queries in a distributed manner, distinguishing between architectures with and without query endpoints.

2. LINKED DATA FOUNDATIONS

In the first session we introduce Linked Data principles [3] and technologies, survey best practices for data publishing while pointing out the differences to Dataspaces, list selected datasets currently available and discuss commonly encountered issues with currently available data.

Linked Data serves as common abstraction to yield a simple unified interface over data sources. Linked Data combines ideas from web architecture with Semantic Web knowledge representation. From web architecture, Linked Data takes Uniform Resource Identifiers (URIs) for naming things and the Hypertext Transport Protocol (HTTP) for transmitting content. A tenet of Linked Data is distributed interlinking of data, based on the hypermedia principle to achieve an architectural style called Representational State Transfer (REST). From the Semantic Web, Linked Data takes the Resource Description Framework (RDF) for representing graph-structured data in subject/predicate/object triples. Technologies on the web are layered, so that elaborate functionality builds on top of more basic functionality, leading to architectures where basic functionality can be made available easily, while more elaborate functionality is still possible to achieve.

Widely deployed Linked Data technologies as such allows for only basic retrieval functionality: looking up information about a single entity, and following links to other entities (potentially at other sources). Thus, there is a need for a query language to express more complex information needs. SPARQL (a recursive acronym for SPARQL Protocol and RDF Query Language) provides means for expressing queries and accessing elaborate query functionality over the network.

Data publishing and integration on the Linked Data web differs from traditional integration systems as there are no central components or coordinators (except for the Domain Name Service to hierarchically assign and resolve identifiers). The integration tasks are split up and spread across multiple participants: separate systems and people are responsible for data generation, publishing, mapping/linking and querying. Data providers often publish data without an immediate application in mind. Interlinking can be disconnected in time and space – other people than the data publisher can do the interlinking, even long after the original data has been published. Links are equal to data itself, and either published alongside the data or at a separate location. The Linked Data vision is towards a decentralized setting facilitating distributed publishing and collaborative maintenance of the combination of data.

In contrast to the web where the mantra is decentralized linking (the hypermedia principle), the architecture of Dataspace systems include a catalog of data sources [8]. Such a registry is not immediately required on the Linked Data web, where URIs serve two purposes: as names for things, but also as names for data sources. There is an intrinsic relation between identifier and source (via syntactic or protocol means). That is, a software program can get a description of a thing via dereferencing the thing (looking up the thing via HTTP, which leads via redirects or via syntactic conventions to the data source with data about the thing).

The architecture of Linked Data shares several characteristics with Dataspaces, for example, the means for incremental integration (so-called pay-as-you-go integration). To facilitate this integration style, the data models are schemaless; in Linked Data, even the distinction between schema and instance data is blurry. Also, Linked Data inherits the so-called open-world assumption, which favors a lenient type of semantics drawing conclusions based on data to reconcile potential inconsistencies rather than integrity checking which would require flagging errors in the data. On the Linked Data web, mappings between sources can include statements of equivalence of identifiers and subclass and subproperty relations. However, systems often do not make use of these mappings due to computational complexity involved when processing very large amounts of data alongside the mappings, and the quality of the mappings. Following the prevalent opinion in the Linked Data community, we focus on data access and querying rather than aspects related to logics and reasoning, which, however, could be layered on top of the existing Linked Data technology stack.

3. CENTRALIZED STORAGE AND QUERY PROCESSING

In the second session we introduce solutions for storing and querying RDF data on centralized systems. Both the Semantic Web and the database communities have developed a large number of such systems, partly reusing and adapting well-established techniques from relational databases [21]. With a few exceptions, all centralized storage and querying systems can be grouped into one of the following three classes:

1. Triple stores that store RDF triples in a single relational table, usually with additional indexes and statistics,

2. Vertically partitioned tables that maintain one table for each property, and

3. Schema-specific solutions that store RDF in a number of property tables representing several properties.

Triple stores keep RDF triples in a single relational table with three or four attributes. Prominent examples of triple stores include 3store [10] and Virtuoso [7] from the Semantic Web community, and RDF-3X [19] and HexaStore [25] from the database community. RDF facts are mapped to a generic three-attribute table of the form `(subject,property,object)`; most systems convert resource identifiers, properties and constants to numeric identifiers to save space and make access structures more efficient. To represent triples from multiple sources, the relation may be extended by a fourth attribute that uniquely identifies the source of a triple; the relation is then also called a quadruple table. For efficient query processing, indexes on (a subset of) all combinations of subject, property, and object are maintained (introduced by [11] as complete indexes), allowing to efficiently retrieve all matches for a triple pattern of a SPARQL query. A complex SPARQL query is processed by (at least conceptually) transforming the query into an equivalent relational algebra expression on the triple store. As in relational databases, an efficient execution plan must be determined from possible variants of physical plans (such as different join implementations or different join orders), which requires accurate estimations of the execution costs of the different alternatives. Estimation techniques implemented in current relational databases, for example attribute-level histograms, often generate wrong estimations since they were not primarily build for self joins of a single table, so many triple stores come with dedicated estimation techniques.

A second line of systems exploits the fact that many triple patterns in real queries have fixed properties. Consequently, the systems create a separate table with two attributes (one for the subject and one for the object) for each property (a third attribute for the graph id may be added). A natural extension of this approach is to store these narrow tables in a column store [1,23], with the additional advantage that all entries within a column have the same type and can thus be efficiently compressed.

A third line of systems exploits the fact that a large number of subjects often have the same or at least are largely overlapping set of properties that will often be accessed together in queries. Property tables represent groups of subjects with similar properties in a single table, making queries on these properties very efficient, at the price of more effort for storing subjects where one of these properties is missing or has multiple objects. A prominent example for this storage structure are early versions of Jena [6, 26]. Levandoski et al. [17] demonstrate that property tables can outperform triple stores and vertically partitioned tables for RDF data collections with regular structure.

Beyond the three classes of systems we presented so far, there are a number of systems that do not fit into these categories, such as representing RDF in matrix form [2], as XML [27], or as graph [4, 18].

4. DISTRIBUTED STORAGE AND QUERY PROCESSING

In the third session we cover distributed query processing techniques [9, 14, 15]. Downloading Linked Data from web sources into a centralized repository is expensive in terms of time and resources consumption and has to be repeated periodically to compensate for updates in the sources. Therefore, an alternative is the application of distributed query processing techniques to evaluate SPARQL queries. In this respect, we distinguish two basic approaches:

1. Lookup-based query processing via traversing links over Linked Data sources, and

2. Distributed query processing over autonomous RDF data sources.

The main principle of lookup-based query processing is that the query is evaluated at an instance which downloads the data from the sources during query execution. Thus, given a query, the key is to identify relevant sources whose data contributes to the final result. Explorative query processing [13], for instance, solves this problem by iteratively evaluating a part of the query, dereferencing URIs occurring in the intermediate results, downloading the data, evaluating another part of the query, and so on. Index-based approaches [12, 24] on the other hand, use precomputed indexes for all available sources to efficiently identify which sources may contribute to the result for a given SPARQL query. Based on the source descriptions, the data of relevant sources is downloaded and the query is evaluated on the data.

As downloading chunks of data from the sources and filtering the data locally can become expensive, an alternative is to use SPARQL endpoints in a distributed setup and apply distributed query processing and optimization techniques [16, 20, 22]. Query optimization in this setup is in many regards similar to distributed database systems, i.e., aspects such as query simplification, filter pushing, source selection, statistics, indexing, join order, cost model, cardinality estimation and operator implementations are considered with the goal of finding the most efficient query execution plan.

As Linked Data sources are mostly autonomous, there is only little cooperation from the sources. Thus, distributed query processing can only benefit from a limited amount of statistics about the sources' data. Moreover, without cooperation that goes beyond providing access to the data via a standard SPARQL 1.0 interface (the current standard), efficient query evaluation that involves direct communication between sources is not possible. The upcoming release of the SPARQL 1.1 standard, however, will finally bring new opportunities for optimization that have not been available before [5]. The SERVICE operator, for instance, allows direct communication between sources, e.g., a query sent for evaluation to source 1 might contain a subquery that needs to be evaluated at source 2. With SPARQL 1.1. source 1 will independently query source 2 without requiring any interference from the central coordinating unit.

5. CONCLUSION

In the tutorial, we present foundational aspects related to Linked Data, and survey existing work on query evaluation

on both centralized and distributed architectures. We expect two main learning outcomes: first, the attendees should get an overview of the state-of-the-art in Linked Data and query processing over Linked Data, and second, the attendees should get a feeling of how the research in the separate communities relates to each other as we point out similarities as well as differences.

6. REFERENCES

[1] D. J. Abadi, A. Marcus, S. Madden, and K. Hollenbach. SW-Store: a vertically partitioned DBMS for Semantic Web data management. *The VLDB Journal*, 18(2):385–406, 2009.

[2] M. Atre, V. Chaoji, M. J. Zaki, and J. A. Hendler. Matrix "bit" loaded: a scalable lightweight join query processor for RDF data. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pages 41–50, 2010.

[3] T. Berners-Lee. Linked Data - Design Issues, 2006. `http://www.w3.org/DesignIssues/LinkedData.html`.

[4] M. Bröcheler, A. Pugliese, and V. S. Subrahmanian. DOGMA: A disk-oriented graph matching algorithm for RDF databases. In *Procedings of the 8th International Semantic Web Conference (ISWC)*, pages 97–113, 2009.

[5] C. Buil-Aranda, M. Arenas, and Ó. Corcho. Semantics and Optimization of the SPARQL 1.1 Federation Extension. In *Proceedings of the 8th Extended Semantic Web Conference (ESWC)*, pages 1–15, 2011.

[6] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: implementing the Semantic Web recommendations. In *Proceedings of the 13th International Conference on World Wide Web (WWW) - Alternate Track Papers & Posters*, pages 74–83, 2004.

[7] O. Erling and I. Mikhailov. RDF Support in the Virtuoso DBMS. In *Networked Knowledge - Networked Media*, volume 221 of *Studies in Computational Intelligence*, pages 7–24, 2009.

[8] M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: a new abstraction for information management. *SIGMOD Record*, 34:27–33, 2005.

[9] O. Görlitz and S. Staab. Federated Data Management and Query Optimization for Linked Open Data. In *New Directions in Web Data Management*, chapter 5, pages 109–137. Springer, 2011.

[10] S. Harris and N. Gibbins. 3store: Efficient bulk RDF storage. In *Proceedings of the First International Workshop on Practical and Scalable Semantic Systems (PSSS)*, 2003.

[11] A. Harth and S. Decker. Optimized index structures for querying rdf from the web. In *Proceedings of the Latin American Web Congress*, pages 71–80, 2005.

[12] A. Harth, K. Hose, M. Karnstedt, A. Polleres, K. Sattler, and J. Umbrich. Data Summaries for On-Demand Queries over Linked Data. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pages 411–420, 2010.

[13] O. Hartig. Zero-Knowledge Query Planning for an Iterator Implementation of Link Traversal Based Query Execution. In *Proceedings of the 8th Extended Semantic Web Conference (ESWC)*, pages 154–169, 2011.

[14] O. Hartig and A. Langegger. A Database Perspective on Consuming Linked Data on the Web. *Datenbank-Spektrum*, 10(2):57–66, 2010.

[15] K. Hose, R. Schenkel, M. Theobald, and G. Weikum. Database Foundations for Scalable RDF Processing. In *Reasoning Web. Semantic Technologies for the Web of Data. Tutorial Lectures of the 7th International Summer School*, pages 202–249, 2011.

[16] A. Langegger, W. Wöß, and M. Blöchl. A semantic web middleware for virtual data integration on the web. In *Proceedings of the 5th European Semantic Web Conference (ESWC)*, pages 493–507, 2008.

[17] J. J. Levandoski and M. F. Mokbel. RDF data-centric storage. In *Proceedings of the 7th IEEE International Conference on Web Services (ICWS)*, pages 911–918, 2009.

[18] B. Liu and B. Hu. HPRD: A high performance RDF database. In *Proceedings of IFIP International Conference on Network and Parallel Computing*, pages 364–374, 2007.

[19] T. Neumann and G. Weikum. The RDF-3X engine for scalable management of rdf data. *The VLDB Journal*, 19(1):91–113, 2010.

[20] B. Quilitz and U. Leser. Querying distributed RDF data sources with SPARQL. In *Proceedings of the 5th European Semantic Web Conference (ESWC)*, pages 524–538, 2008.

[21] S. Sakr and G. Al-Naymat. Relational processing of rdf queries: a survey. *SIGMOD Record*, 38(4):23–28, 2009.

[22] A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt. FedX: Optimization techniques for federated query processing on linked data. In *Procedings of the 10th International Semantic Web Conference (ISWC)*, pages 601–616, 2011.

[23] L. Sidirourgos, R. Goncalves, M. L. Kersten, N. Nes, and S. Manegold. Column-store support for RDF data management: not all swans are white. *Proceedings of the VLDB Endowment*, 1(2):1553–1563, 2008.

[24] J. Umbrich, K. Hose, M. Karnstedt, A. Harth, and A. Polleres. Comparing data summaries for processing live queries over linked data. *World Wide Web*, 14(5-6):495–544, 2011.

[25] C. Weiss, P. Karras, and A. Bernstein. Hexastore: sextuple indexing for Semantic Web data management. *Proceedings of the VLDB Endowment*, 1(1):1008–1019, 2008.

[26] K. Wilkinson, C. Sayers, H. A. Kuno, and D. Reynolds. Efficient RDF storage and retrieval in Jena2. In *Proceedings of the First International Workshop on Semantic Web and Databases (SWDB)*, pages 131–150, 2003.

[27] M. Zhou and Y. Wu. XML-based RDF data management for efficient query processing. In *Proceedings of the 13th International Workshop on the Web and Databases (WebDB)*, 2010.