Chapter 13
Summary, Conclusions, and Future Work

This book covers a wide range of topics in the area of the Semantic Web related to query processing.

First of all in Chap. 2 after the introduction in Chap. 1, we have learnt about the basic specifications of the Semantic Web, its data language RDF, its ontology languages RDF Schema and OWL, its query language SPARQL, and its rule language RIF. We got to know that the various different Semantic Web languages are powerful enough for nearly any kind of application.

Then the B^+-tree has been introduced as an efficient index for large-scale datasets (Chap. 3). Also, external sorting for efficiently building the B^+-tree from large-scale datasets from scratch has been discussed. These are the basics for efficient indexing and querying large-scale datasets, which are the topics in later chapters.

An overview of query optimization phases has been given afterward (Chap. 4). We have then provided the transformation from SPARQL to a core of SPARQL without redundancies in the language constructs as basic operation used in many tools to simplify, for example, building an operatorgraph or a visual query.

An algebra for SPARQL and its logical optimizations are discussed in Chap. 5. Applying equivalency rules to the query expressed in the notion of the algebra is the basis for any kind of optimization.

The algorithms for processing of the operators of a SPARQL query and indexing techniques for large-scale as well as small datasets are the content of Chap. 6. Experimental results show the superior performance for large-scale as well as small datasets.

The processing of infinite data streams as generated, for example, from sensors is the topic of Chap. 7. Infinite data streams require another kind of operators that periodically calculate query results. Because differences to previous query results are calculated internally, the traditional operators for joins, sorting, and so on must also deal with requests to delete a solution from their temporary indices and buffers. The demonstrated solution shows the practical relevance of this young field of research.

Chapter 8 introduces into the world of multicore processors and their optimizations for Semantic Web parallel databases. Not all queries can benefit from parallel

S. Groppe, *Data Management and Query Processing in Semantic Web Databases*,
DOI 10.1007/978-3-642-19357-6_13, © Springer-Verlag Berlin Heidelberg 2011

processing. However, the query optimizer can already estimate if an operator is worth to be parallelized and can generate the operator graph accordingly.

Inference is a highly costly operation, which needs optimization. Inference materialization strategies as well as optimization approaches have been discussed in Chap. 9. The optimizations work for database queries as well as for stream queries and significantly improve the performance.

Visual query languages to help users formulating queries are introduced in Chap. 10. Users do not need to learn the syntax of SPARQL and the visual representation helps them to easily see connected terms in their query. Furthermore, the system can provide suggestions to extend and refine their queries.

Chapter 11 discusses the features, possibilities, and the technology behind embedded Semantic Web languages in programming languages. The advantage is that a static analysis can detect already many errors during compilation, which may be otherwise only detected after extensive tests during runtime. By determining the types of query results already at compile time, the java type system can also be orthogonally used to detect this kind of type errors. Therefore, embedded query languages ensure more stable programs and applications.

Chapter 12 compares the XML world as alternative to the Semantic Web world and shows common features as well as differences. Furthermore, this chapter shows that it is possible to embed SPARQL into XQuery and into XSLT as well as to embed XPath into SPARQL. We can conclude that you can theoretically use XML or the Semantic Web (or both in parallel) for a solution involving data and queries. However, some solutions are more elegant and easier to formulate when using XML with its tree model or the Semantic Web with its graph model.

This book already covers many aspects of data management and query processing. It demonstrates that the technologies for a high performance Semantic Web have been developed. These (or even more advanced) technologies must now find their ways into industry and their commercial products. Only then the Semantic Web will be successful and accepted by the user and at the market. Many companies such as Oracle have already started to support Semantic Web technologies, but it is still a long way until it is a *must* for companies to support Semantic Web technologies.

13.1 Possibilities for Future Work

A book can cover only a snapshot of current research. In the following paragraphs, we will give hints where future work can be done on the presented topics.

Of course, future work can deal with more equivalency rules for logical optimizations in the operator graph as well as for inference and specialized equivalency rules for stream queries.

Stream processing can be further extended by developing new types of windows and update streams with the possibility to delete or update an older triple.

Parallel algorithms can be further investigated for remaining operators. However, the most rewarding operators to be parallelized are the join operators, which have already been discussed in this book.

The obvious next step for research on inference is to support OWL and OWL 2, where probably further optimization rules must be developed.

Research on visual query languages can include research on further simplifying query creation by having a more browser-like graphical user interface without losing the provided flexibility by, for example, using the current approach as fallback to further manipulate the visual query.

Research on embedded languages can focus on using the static analysis to optimize queries and the program containing the queries together instead separately, which is state of the art. This promises much better optimization possibilities. Furthermore, also the costly inference could be precomputed as much as possible based on a static analysis and optimized together with the program code.

The relationship to XML can be further investigated and, for example, inference can be included to be simulated by XQuery and/or XSLT. Other data models such as the object-oriented one and query languages such as object-oriented query languages can be investigated if they are as powerful as the Semantic Web ones and how to transform data and translate queries into each other.

Another big topic would be to efficiently support rules (e.g., RIF) and focus on optimizations for rules in large-scale databases. It is also an open research question how to combine inference based on ontologies as well as based on rules and query processing and apply all these three different kinds of processing in one system. Research must be further driven to rule stream processing, parallel rule processing, visual rules, and embedded rules in programming languages.

Microformats are used to embed RDF data into webpages. Research is missing which can deal with these highly distributed and split pieces of data, their retrieval, integration, and processing.

A new research trend in databases is to use hardware such as FPGAs or the power of graphical processors for optimizing processing. New optimization techniques have to be developed to optimize Semantic Web queries, inference, and rules on this hardware, but the benefits seem to be enormous.