TURKKA NÄPPILÄ

Serving Sophisticated Ad Hoc Information Needs Based on Beforehand Unknown, Autonomous, and Heterogeneous XML Data Sources

■

ACADEMIC DISSERTATION
To be presented, with the permission of
the Board of the School of Information Sciences of the University of Tampere,
for public discussion in the Auditorium Pinni B 1096,
Kanslerinrinne 1, Tampere, on December 4th, 2013, at 12 o'clock.

UNIVERSITY OF TAMPERE

ACADEMIC DISSERTATION
University of Tampere, School of Information Sciences
Finland

Cover design by
Mikko Reinikka

Acta Universitatis Tamperensis 1878
ISBN 978-951-44-9284-6 (print)
ISSN-L 1455-1616
ISSN 1455-1616

Acta Electronica Universitatis Tamperensis 1359
ISBN 978-951-44-9285-3 (pdf)
ISSN 1456-954X
http://tampub.uta.fi

Suomen Yliopistopaino Oy – Juvenes Print
Tampere 2013

Abstract

In this dissertation, we study serving of the user's sophisticated ad hoc information needs based on beforehand unknown, autonomous and heterogeneous XML data sources. Serving sophisticated information needs that concern comparing and analyzing products, places, individuals, and so on, often requires that the underlying data are combined from disparate and independently organized data sources. In other words, data integration is needed. However, when the information needs at hand are ad hoc or short-term in nature, the use of traditional data integration techniques is often not an option because of the time and resources their effective application requires. Moreover, they also typically presume considerable technical expertise from their user as well as intimate familiarity with the underlying data sources.

The dataspace approach has been introduced to overcome the limitations of the traditional data integration approaches when operating in modern information environments. A dataspace is a collection of data sources intended to provide all of the information relevant to a particular user or task, regardless of the format or the systems and interfaces through which they are accessed. Data integration in dataspaces follows the so-called pay-as-you-go principle. The idea is that the user can be provided with useful services without the need for full integration between the underlying data sources from the start, but the integration is tightened gradually as the user interacts with the underlying dataspace over the time. The work presented in this dissertation can be characterized as a dataspace-oriented approach.

While dataspace systems are supposed to support multiple data formats in parallel, our starting point, however, is that the corresponding applicability can be achieved by adopting XML (Extensible Markup Language) as the general data format and model. There are several reasons that support this decision. Most importantly, XML, being based on a semistructured data model, provides a platform-independent means to represent information, and it is today the de facto standard for exchanging both unstructured documents and structured data.

This dissertation presents a four-phased framework for describing how the user's sophisticated ad hoc information needs are served based on beforehand

unknown, autonomous and heterogeneous XML data sources. Within this framework, we contribute several novel methods, techniques, and tools (i) for searching potentially useful data sources; (ii) for assessing their actual suitability with respect to the task at hand as well as their mutual consistency; (iii) for overcoming and removing the possible inconsistencies in them; and, finally, (iv) for satisfying the user's information needs through a powerful query language, in which data integration is seamlessly combined with typical data-centric manipulation.

Keywords: data integration, data management, dataspaces, formal specification, information retrieval, query languages, XML.

Contents

Preface

I would like to express my deepest gratitude to Dr. Timo Niemi, my supervisor, who gave the impetus to my research work, patiently kept it going over the years, and finally oversaw its completion. I can only concur with the words of my colleague: "This has been a harder and more extended school than I ever expected." I am also most indebted to Professor Kalervo Järvelin, my co-author on two papers, for his insightfulness and eloquence and, to a no lesser degree, for financing my research. I also feel hugely privileged to have had the opportunity to work with my other co-authors, Paavo Arvola, Janne Jämsen, and Katja Moilanen.

I wish to thank Professor Emeritus Hannu Kangassalo, who initially raised my interest in the field of data management. Special thanks should be given to Kati Iltanen for her continuous support and presence in the midst of the sometimes rough seas of academic life. I am also grateful to my colleagues (in alphabetical order) Henry Joutsijoki, Marko Junkkari, Jaana Kekälainen, Jorma Laurikkala, Erkki Mäkinen, Pekka Niemenlehto, Tapio Niemi, and Marko Niinimäki. My thanks are extended to the staff and other colleagues in the School of Information Sciences and its predecessors. Special thanks go to Kirsi Varpa without whom I might never become a computer scientist in the first place.

This research has been financially supported in part by the Tampere Graduate School in Information Science and Engineering (TISE; with special thanks to its former Coordinator Pertti Koivisto) and the Academy of Finland (under project nos. 209960, 140315, and 133021).

Finally, I wish to thank my parents for their support and encouragement throughout my studies. I also wish to thank Matleena for her love and caring during these last years.

I dedicate this dissertation to our son Veikko.

Tampere, October 2013

Turkka Näppilä

List of Abbreviations

Abbreviations

B2B	Business-to-Business
DTD	Document Type Declaration
ETL	Extract–Transform–Load
GAV	Global-as-View
HTML	Hypertext Markup Language
IR	Information Retrieval
LAV	Local-as-View
LCA	Lowest Common Ancestor
MLCA	Meaningful Lowest Common Ancestor
OLAP	Online Analytical Processing
OLTP	Online Transactional Processing
RDBMS	Relational Database Management System
RXQL	Relational XML Query Language
SGML	Standard Generalized Markup Language
SLCA	Smallest Lowest Common Ancestor
SPC	Smallest Possible Context
SQL	Structured Query Language
W3C	World Wide Web Consortium
XML	Extensible Markup Language

List of Original Publications

This dissertation is based on the following five publications. In the text, they are referred to by Paper with their respective Roman numerals.

Paper I — Janne Jämsen, Turkka Näppilä, and Paavo Arvola. Entity ranking based on category expansion. In Norbert Fuhr et al. (eds.): *Focused Access to XML Documents: The 6th International Workshop of the Initiative for the Evaluation of XML Retrieval* (Selected Papers). LNCS **4862**, 2008: 264–278.

Paper II — Turkka Näppilä and Timo Niemi. An approach for developing a schemaless XML dataspace profiling system. *Journal of Information Science* **38** (3), 2012: 234–257.

Paper III — Turkka Näppilä, Kalervo Järvelin, and Timo Niemi. A tool for data cube construction from structurally heterogeneous XML documents. *Journal of the American Society for Information Science and Technology* **59** (3), 2008, 435–449.

Paper IV — Timo Niemi, Turkka Näppilä, and Kalervo Järvelin. A relational data harmonization approach to XML. *Journal of Information Science* **35** (5), 2009: 571–601.

Paper V — Turkka Näppilä, Katja Moilanen, and Timo Niemi. A query language for selecting, harmonizing, and aggregating heterogeneous XML data. *International Journal of Web Information Systems* **7** (1), 2011: 62–99.

Part I

Chapter 1

Introduction

1.1 Motivation

In the contemporary world, people often have unexpected needs to compare and analyze products, places, individuals, and so on. To serve such sophisticated information needs, which typically require merging and/or aggregating data from multiple sources, data integration is often needed. Data integration[1] can be defined as the problem of combining data from a variety of sources in order to provide the user with a unified access to them [59]. More specifically, data integration aims to enable the interoperation between autonomous (i.e., designed from independent starting points) and heterogeneous data sources. The heterogeneity here refers to the fact that the underlying data may be structured, labeled, and/or represented differently or they are based on different data models, formats, and/or technologies.[2] Data integration has been an active topic in the database research community since the problems faced in the context of distributed databases in the early 1980s. Outside academia, data integration is traditionally needed in various enterprise scenarios, such as mergers and acquisitions, internal restructuring, and operating with third parties [27].

The emergence of modern information environments with the ever increasing presence of computing devices and, not least, the World Wide Web and its related technologies, has created new ways to publish, share, and consume data. The utilization of such resources also often requires data integration. However, the starting points of data integration in these modern information environments are in many respects quite different from those underlying the traditional data integration approaches that were developed for closed (enterprise) information environments. For example, traditional data integration systems are typically built for a defined set of applications and/or business

[1] Synonyms: information integration, enterprise information integration.
[2] The latter three types of heterogeneity are not considered in this dissertation.

needs, which are specified for a defined set of data sources [30]. In contrast, in modern information environments the users' information needs are more often ad hoc and/or short-term in nature. Therefore, the satisfaction of these kinds of information needs cannot usually be a justification for building a full-scale data integration system, which has been acknowledged to be a time- and resource-consuming process [8]. Moreover, due to the volume and diversity of the data sources available in modern information environments, it cannot be assumed that a single person or group could have intimate (a priori) knowledge of them, as is typically the case in closed information environments. Indeed, when operating in modern information environments, the first task is often to find out the data sources that are the most relevant from the perspective of the information need at hand. Lastly, the traditional data integration techniques were intended for expert users, such as database administrators, whereas in modern information environments users are increasingly less technically-centered people, such as journalists, scientists, and business analysts, with their professional expertise being in other fields. For the reasons above, it becomes apparent that new kinds of approaches and techniques must be developed to assist the user in satisfying his or her sophisticated ad hoc information needs in modern information environments.

1.2 Approach

In this dissertation, we address the challenge above. More specifically, we present a framework, and contribute several novel methods, techniques, and tools within it, for serving the user's sophisticated ad hoc information needs based on beforehand unknown, autonomous and heterogeneous data sources. It should be noted that we do not assume that the data sources at hand are solely open data, although our work does have much in common with the starting points of open data approaches (e.g., [30, 31]). A typical scenario in which the approach presented in this dissertation can be applied, is the one in which internal (database) data are combined with data derived from external sources, such as the Web. The key feature is that at least a part of the data sources are unknown to the user.

Our work bears its closest resemblance to the dataspace approach and pay-as-you-go data integration. A dataspace [39] is a collection of data sources intended to provide all of the information relevant to a particular user or task, regardless of the format or the systems and interfaces through which they are accessed. Typically, initially no one is intimately familiar with a dataspace composed of autonomous and heterogeneous data sources [49]. Hence, much of the user's interaction with the underlying dataspace is of an exploratory nature [28]. Dataspace systems follow the so-called pay-as-you-go principle in data integration (see, [61, 52, 74]). This means that the integration between

the data sources of the underlying dataspace is tightened over the time as the user's understanding on the underlying dataspace increases through his or her interactions with the available data sources. Hence, dataspace systems do not require full integration between the data sources from the start but provide a set of tools, ranging from simple keyword search to more sophisticated mechanisms, through which the degree of integration can be gradually increased. The dataspace approach has been applied, for example, in personal information management (e.g., [72]) and in scientific/medical domain (e.g., [33, 49]).

The approach presented in this dissertation can be characterized as a dataspace-oriented one for two reasons. First, the information environments, at which the methods, techniques, and tools proposed in this dissertation are targeted, can be considered as dataspaces. Second, in assisting the user in increasing his or her understanding on the available, initially unknown data sources stepwise we also follow a pay-as-you-go-like principle. However, we do not follow the original dataspace vision, as presented in [39, 45], strictly. For example, we depart from it in that we assume that, instead of supporting multiple data formats in parallel, all the data sources that comprise the dataspace at hand are represented as XML (Extensible Markup Language; [35]). In fact, to the best of our knowledge none of the dataspace systems developed thus far (e.g., [33, 49, 72]) also implement the multi-format support. This is mostly due to the fact that the effective and efficient querying and manipulation of available data sources requires in practice that the dataspace must subscribe to a single, general data model and format. However, by adopting XML as the common data format, we believe that the general applicability of the approach presented in this dissertation is, in this respect, comparable to that of the original dataspace vision.

There are several reasons that support our decision of adopting XML as the general data format. First, XML, as a markup language for representing information in a textual format that is both human-readable and machine-processable, is today the de facto standard both for representing (unstructured) documents and for exchanging and sharing (structured) data. For example, HTML5,[3] the forthcoming major revision of HTML (Hypertext Markup Language; [50]), the core language of the Web, will be specified in the XML syntax, called XHTML, in addition to the conventional HTML syntax. Furthermore, most business-to-business (B2B; see, [58]) and e-Government (see, [68]) applications are nowadays based on XML with domain-specific vocabularies. Second, many structured and unstructured data formats can be converted into XML in a straightforward manner (but not necessarily vice versa). For example, the relational database systems by major vendors (IBM,[4] Microsoft,[5]

[3] *HTML5: A vocabulary and associated APIs for HTML and XHTML.* W3C Candidate Recommendation. Available at http://www.w3.org/TR/html5/ (2013, accessed October 2013).

[4] *IBM DB2.* See, http://www.ibm.com/db2/ (accessed October 2013).

[5] *Microsoft SQL Server.* See, http://www.microsoft.com/sqlserver/ (accessed October 2013).

and Oracle[6]) support XML publishing. Finally, the single greatest advantage of XML is that – due to its semi-structured nature that allows the coexistence of data and metadata within the instances of XML data – it provides a platform-independent means of representing information.

1.3 Goals

In summary, in this dissertation we study serving of the user's sophisticated ad hoc information needs based on beforehand unknown, autonomous and heterogeneous XML data sources. We specifically argue that data integration, which is typically a prerequisite for utilizing these kinds of resources, cannot be based on the same principles in emerging modern information environments as it has been in traditional closed information environments, but new ones need to be developed. In this respect, the approach proposed in this dissertation shares the same starting point with the dataspace approach that has been proposed to address the above challenge. Our approach also shares many other characteristics with the dataspace approach but departs from it on two important points. First, instead of supporting multiple data formats, we assume that the available data sources are represented as, or can be converted into, XML. Second, since we consider supporting ad hoc information needs, the resulting integration need not be stored for later use.

This dissertation contributes several novel methods, techniques, and tools for dealing with heterogeneous XML data sources. These contributions are placed in a four-phased framework describing the process of serving the user's sophisticated ad hoc information needs based on beforehand unknown, autonomous and heterogeneous XML data sources. The specific contributions of this dissertation are as follows:

1. We develop methods to be utilized in the searching of potentially useful data sources in the underlying XML dataspace.
2. We develop an XML dataspace profiling approach to assist the user in assessing the actual suitability of the retrieved XML data sources as well as their mutual consistency.
3. We develop principles, techniques, and tools that can be used to
   (a) overcome the possible structural heterogeneity among the available XML data sources; and
   (b) harmonize the available XML data sources by removing the types of heterogeneity that occur in them.
4. We develop a high-level query language, through which the user is able to satisfy his or her sophisticated ad hoc information needs based on autonomous and heterogeneous XML data sources and which combines data integration with typical data-centric manipulation.

[6] *Oracle Database 12*c. See, http://www.oracle.com/database/ (accessed October 2013).

1.4 Organization

The remainder of Part I of this dissertation is organized as follows. Chapters 2 and 3 introduce some central concepts and approaches related to data integration and XML, respectively. In Chapter 4, the contributions of this dissertation are outlined and the roles and interrelationships of the individual works presented in the five publications comprising Part II of this dissertation are considered. These contributions are discussed and the conclusions are given finally in Chapter 5.

Chapter 2

Data Integration

2.1 Types of Data

The data[7] to be integrated can be classified into three categories: structured, unstructured, and semi-structured. Structured data are regular and organized based on systematic principles. Typically, structured data are represented as a set of specific entities, their properties, and the relationships between the entities. These are also the basic constituents of a database. How the data in a database are described and what operations are used to query and manipulate them is determined by a mathematical formalism called data model [81]. A well-known data model for organizing a database is the relational model, introduced in [20]. In the relational model, the underlying data are described by a set of labeled relations, which, in turn, consist of a set of labeled attributes.[8] This description is called a (relational) schema. The operations that can be used to query and manipulate relational data (i.e., tuples, or rows of tables) are defined by the relational calculus and algebra [21] on which the well-known structured query language, SQL [80], is based. Other well-known data models are the so-called legacy data models – the network model (e.g., [25]) and the hierarchical model (e.g., [62]) –, the functional data model (e.g., [76]), the object data model (e.g., [14]), the deductive (logic-based) data model (e.g., [81]), as well as their variants. In a database system, the stored data

[7] In the English-language literature, a distinction is often made between data and the related concepts of information, knowledge, and wisdom, known as the DIKW (data–information–knowledge–wisdom) hierarchy or pyramid (see, [4]), with the concept of understanding sometimes inserted between knowledge and wisdom. The DIKW hierarchy is intended to describe the refinement process through which "raw" data are turned into the valued wisdom. In this dissertation, the raw data we are dealing with are mainly textual as opposed to, for example, numeric data matrices that are used in many applications. In addition, we use the terms 'data' and 'information' as near synonyms for the contents of these textual sources, whereas the terms 'understanding' and 'knowledge' are used to refer to the cognitive states of the persons interpreting them. The concept of wisdom is altogether left out of consideration.

[8] The widely used visual metaphor for the set-theoretic notion of relation is a table. Consequently, the attributes of a relation are also typically visualized as the columns (column names) of these tables.

are separated from their description and manipulation. From this follows that in the context of structured data, two levels of abstraction, called the schema level and the instance level, are distinguished, of which the former provides the metadata for the latter.

The word 'unstructured' is a somewhat misleading epithet for describing these kinds of data sources, since in most cases the instances of unstructured data do have a structure. For example, let us consider a typical example of unstructured data: a book. A book obviously has a structure in that it is divided into chapters, which consist of sections, which, in turn, contain subsections composed of text paragraphs, and so on. Besides books, other representatives of unstructured data are all kinds of textual documents, such as articles, memoranda, e-mails, slide shows, Web pages, and so on.[9] Therefore, in the context of unstructured data the word 'unstructured' refers to the fact that the structure among data is implicit and needs to be reasoned out, rather than to the complete absence of structure. One reason for this is that unstructured data are often irregular in nature. For example, it is much more difficult to restrict the number of paragraphs in a book than the number of the valid addresses for an employee in a database.

Although the term "semi-structured data" was used for the first time in [77] in the early 1990s, a more widespread interest in them followed only later in that decade in the wake of the works of Abiteboul [1] and Buneman [12]. Concurrently with these works, the newly formed World Wide Web Consortium (W3C) introduced XML. Quickly overruling the other proposed semi-structured data models (such as OEM [3, 67] and UnQL [13]), XML has by now become the synonym for semi-structured data. Semi-structured and XML data can be either structured or unstructured in nature – in fact, an instance of semi-structured data may have both structured and unstructured parts. As a consequence of this, for example, XML data sources are often classified into data- and document-centric ones, depending on how regularly or irregularly their contents are structured [41].

Conceptually, instances of semi-structured data are often modeled as ordered labeled trees,[10] consisting of nodes and the edges between them. The use of trees as modeling constructs provides the semi-structured data models with richer and more flexible modeling capabilities compared to the relational model, in which the original application structures are forced to be organized as a set of flat tables.[11] However, the price of this is that the instances of semi-

[9] Multimedia data, such as still image, video, and audio files, are also generally considered to fall into this category. The study of these kinds of data is, however, outside the scope of this dissertation.

[10] Synonym: directed acyclic graph.

[11] In addition, the relational calculus and algebra and, consequently, the original form of SQL, which was developed based on them, have one important limitation: they cannot express queries involving paths through an instance, such as the transitive closure over a binary relation [53]. However, recursive queries including transitive closure have been implemented in later versions

structured data cannot be controlled by means similar to relational schemata, which are expected to be accurate reflections of the underlying data. In the context of semi-structured data, the metadata (node labels) are intermingled with the data (node values). For this reason, semi-structured data are sometimes called self-describing. In its part, this property also enables the management of the variability that is typical among the instances of semi-structured data.

2.2 Traditional Data Integration Approaches

The central concepts and techniques for data integration originate from research related to distributed databases, more specifically heterogeneous distributed databases. In contrast to homogeneous distributed databases, which are based on identical modeling and processing principles, heterogeneous distributed databases are a conglomeration of autonomous databases designed from independent starting points and thus displaying diversity in their data models, database management systems (DBMSs), data representation, and so on [7]. Two architectural solutions for heterogeneous distributed databases have been proposed, both of which have to do with a global schema[12] that provides a reconciled view over the available databases [59]. Federated databases (see, [47, 75]) follow the so-called global-as-view (GAV) approach, in which the global schema is defined as a view over the local databases. Multi-databases (see, [78, 24]), on the other hand, follow the local-as-view (LAV) approach, in which each of the local databases is defined as a view over the global schema. The GAV approach offers conceptual simplicity by describing directly how the instances conforming to the global schema are produced from the source data, whereas in the LAV approach the focus is on describing each source as precisely as possible and in isolation from the others, which makes it easier to add or remove sources compared to the GAV approach but also requires that the user masters several database languages [27]. In the GLAV (global-and-local-as-view) approach, the respective advantages of the GAV and LAV approaches are combined.

A popular means to specify the relationships between schemata are schema mappings. Schema mappings are declarative formalisms that capture the interaction between the schemata at the logical level but do not pay attention to the implementation details at the physical level [55]. On the other hand, the procedures that are used to transform the data at the physical level are known as data translations[13] (see, [2, 37]). Data translations are used to produce the instance of a target schema from source data when a source schema, a target

of SQL (from 1999 onwards).

[12] Synonyms: integration schema, mediated schema, (almost) target schema.

[13] Synonym: data exchange.

schema, and the mappings between them are given. Besides these conventional data translations (or data-to-data translations, as they are called in [48]), oftentimes also data-to-metadata translations are needed to transpose a piece of information expressed at the instance level into a schema-level component. Several operations for data-to-metadata translations have been developed in the context of relational databases (see, [85]).[14] For example, SQL has been proposed to be extended with capabilities of including metadata seamlessly with data in queries [57].[15]

Distributed databases, as presented above, represent the so-called OLTP (online transactional processing) paradigm of information processing, which means that they are optimized for transactions (insertions, updates, and deletions). As a consequence, the primary goal of the data integration approaches for distributed databases has been to ensure the efficient and effective execution of these operations. In contrast, analytical databases are optimized for query processing. Analytical databases are developed to support decision making by providing capabilities for advanced analyses based on historical or (near) real-time data. OLAP (online analytical processing; see, [16]) is an exemplar of this kind of analyzing power. In OLAP, the data to be analyzed are organized as multidimensional data cubes containing aggregated numeric data, called measures or facts, that are related to a specific subject (e.g., sales). These measures are then analyzed with respect to certain factors (e.g., time, location, and category), called dimensions. The dimensions are typically hierarchical (e.g., time can be broke down into days, weeks, months, etc.), which makes it possible to analyze the measures at multiple levels of granularity by rolling up or drilling down through the dimension hierarchy.

The data for data cubes are typically drawn from a data warehouse. Data warehouses are famously defined as being "a subject-oriented, integrated, time-variant, and non-volatile collections of data" [51]. Two approaches for data warehouse architecture have been proposed. The so-called top-down approach

[14] Probably the most well-known of these operations is *pivot* [23, 86], which has also been implemented in many spreadsheet applications.

[15] (Semi-)automatic techniques for finding out the potential correspondences between different schemata are called schema matching (see, [69]). The related problem of finding correspondences at the instance level is, in turn, known as entity matching [56]. (In literature, entity matching is also variously referred to as duplicate identification, record linkage, entity resolution, reference reconciliation, instance identification, name matching, duplicate record detection, and data instance matching. In addition, the name disambiguation problem is its special case.) Extensive surveys on general comparison techniques proposed for entity matching are provided in [56, 32, 29]. These techniques typically incorporate some metrics to determine the similarity between the found values, but the performance of individual methods depends heavily on the nature of the underlying data sets. In many cases, the best results are achieved by combining different techniques and metrics. Entity matching focuses on finding out the different disguises of semantically equivalent data. A related task occurs when semantically non-equivalent data are represented identically. This is more commonly known as the problem of homonyms. For example, [66] reviews and devises techniques for detecting homonyms. Since the study of (semi-)automatic detection of semantically equivalent and non-equivalent data and schema components is a wide and independent research problem, its further examination is outside the scope of this dissertation.

consists of one centralized data warehouse, from which the distinct parts of the organization can derive data needed in their specific analytical purposes. In the bottom-up approach, the corporate data warehouse is, instead, considered as a view over a set of autonomous departmental analytical databases, or data marts.

In the background of data warehouse architecture, the extraction, transformation, and loading (ETL) processes seek to ensure the availability of reliable and up-to-date data [54]. In ETL, extraction refers to the operations, which retrieve the appropriate data from operational databases and other available data sources. In the transformation phase, the extracted data are then restructured to conform to the data warehouse schema, standardized in their representation, and cleansed of possible errors. Taking data quality issues into account is especially important in data warehouses, since their data are typically used for decision making.[16] Finally, the transformed data are loaded into the central data warehouse or local data marts, depending on the design methodology used, to be used in analytical applications [82]. It has been acknowledged that, although being costly, labor-intensive, and mission-critical, ETL processes are increasingly important for the success of data warehousing projects [82]. Today, ETL processes, data warehouses, and OLAP are essential components in business intelligence (BI) architecture (see, [17]).

The operating environment and information needs determine how the actual integration takes place. The two common approaches for this purpose are materialization and mediation [84].[17] In materialization, data from each local database are extracted in advance, filtered and transformed into some appropriate form, merged with relevant data from other sources, and loaded into a centralized repository. When a query is made, it is evaluated directly against the data at this repository, without the need to access the original data sources. Data warehouses typically implement this approach. In mediation, on the other hand, data are extracted from their original sources only when a query is made. Mediation consists of two steps. First, the user formulates his or her query based on the global schema. This query is then reformulated into a set of sub-queries, each of which conforms to the schemata of individual local databases and is executed against them. Second, the results retrieved from the local databases are combined (and the appropriate filtering, transfor-

[16] Data quality issues refer to the fact that the underlying data may be erroneous and/or incomplete. We consider 'erroneous' being distinct from 'heterogeneous' in that errors in data are mostly unintentional (e.g., typing errors), whereas heterogeneity is intentional (it results from the autonomy in the design of the original data sources). In this dissertation, our starting point is that the available data sources are correct and complete in the sense that they do not contain errors and they are able to provide all the information relevant to the task at hand. The in-depth study of the issues affecting data quality and how they can be overcome is outside the scope of this dissertation.

[17] It should be noted that in some terminology, such as in [37], materialization is referred to as data exchange (cf. Footnote 13) and mediation as data integration. Federation is another synonym for mediation.

mation, and merging of the retrieved data is performed) and the final result, which conforms to the global schema, is returned to the user. The module which decomposes the user's original query and combines the results is called a mediator. Distributed databases traditionally follow this approach.

2.3 Dataspace Approach and Pay-As-You-Go Data Integration

As stated, building a full-scale data integration system is a costly and time-consuming project [8] and thus hardly appropriate if the information needs at hand are ad hoc or short-term in nature. Dataspace systems [39, 45] have been proposed to address this challenge. Unlike traditional data integration systems, dataspace systems do not require full integration of the available data sources in order to be able to provide the user with useful services [45]. In fact, their central idea is that the integration between the underlying data sources is tightened gradually over the time as the user's understanding on them increases through his or her interaction with the underlying dataspace.

A dataspace, as defined in [39], is a collection of data sources that intend to provide all of the information relevant to a particular user or task, regardless of the format of the underlying data sources or the systems and interfaces through which they are accessed. The authors, who originally proposed the dataspace approach in [39, 45], argue that dataspaces are not a data integration but a data coexistence approach. This can be understood such that, due to the diversity of different formats among the underlying data sources, it is not possible to construct a conventional global schema to provide a reconciled view over the available data sources. Therefore, in the dataspace approach the mappings between disparate data sources are specified only when the task at hand requires them, and not in advance as in traditional data integration approaches [6]. In doing so, the dataspace approach applies the so-called pay-as-you-go principle in data integration (see, [61, 52, 74]).

A dataspace system (or a dataspace support platform, the term suggested by the original authors) is, in essence, a collection of varied tools that enable the user's interaction with the underlying data sources in order to satisfy his or her information needs. The dataspace system architecture consists of tools, among others, for locating and understanding the available data sources, assessing their usefulness for the task at hand as well as their mutual consistency, and, when necessary, modifying them to be more suitable for the task at hand [40]. For example, the facilities for browsing and searching belong to the basic functionalities of a dataspace system. More advanced functionalities include indexing the dataspaces for efficient search and querying, support for discovering the semantic relationships between the underlying data sources, and mechanisms for keeping track of the provenance of the data.

The work presented in this dissertation follows loosely the dataspace approach. Most importantly, our work and the dataspace approach share some fundamental background assumptions in that both of us start from the premise that (i) the user is not familiar with the autonomous and heterogeneous data sources at hand; (ii) no a priori schema is available to describe the underlying data sources and their interrelationships; and (iii) traditional data integration techniques cannot be applied to consolidate the data from the underlying data sources. However, we also make some considerable departures from the original dataspace vision as presented in [39, 45]. For example, we assume that the user's ad hoc information needs are served through a one-off act of integration that can be revoked once the information need is satisfied. In other words, we do not pay attention to storing the results of the integration for some later use. This is contrary to the dataspace approach, in which the goal is to ultimately establish, in a pay-as-you-go manner, full and permanent integration between the underlying data sources. Furthermore, the original dataspace approach envisioned that dataspace systems should support multiple data formats in parallel. However, in order to enable the effective and efficient querying and manipulation of the available data sources, a dataspace system must in practice subscribe to a single, general data model and format. For example, to the best of our knowledge, none of the dataspace systems developed thus far (see, [33, 49, 72]) implement the multi-format support. In this dissertation, we assume that the data sources constituting the dataspace at hand are represented based on XML.

Chapter 3

XML

3.1 Basics

XML and HTML both derive from SGML (Standard Generalized Markup Language; [79]), which was developed in the 1980s as a language to describe electronic documents (i.e., marked-up text) in a way that is both human-readable and machine-processable, and they thus follow many of the syntactical conventions introduced in SGML. However, the major difference between XML and HTML/SGML is that, while HTML/SGML focus on how to display the available information to the user, XML in turn facilitates the annotation of the meaningful information units (i.e., semantics) among the underlying data in such a way that they can be interpreted by human and computer alike. For example, with HTML one is able to point out that a certain passage in a text is a typographical paragraph, whereas with XML one can denote that certain words in that text passage stand for a person. Hence, XML is considered as a major contributing factor in the transition from the current "Web of documents" to the forthcoming "Web of data", also known as the Semantic Web [9]. In fact, the principal technology of the Semantic Web, RDF (Resource Description Framework; [70]), is most often expressed in XML.

Similarly to HTML/SGML, instances of XML data are serialized as documents, consisting of elements, attributes, and textual/numeric values (character data).[18] In the conventional tree model, the element and attribute occurrences correspond to the inner nodes and the occurrences of textual/numeric values to the leaf nodes of the tree. In addition, an element occurrence may enclose occurrences of other elements, attributes, and/or textual/numeric val-

[18] Besides the aforementioned data items, other language constructs, such as comments, namespaces, and processing instructions, are also available. However, due to their secondary role in the work presented in this dissertation (e.g., the authors in [49] liken shared namespaces to a global schema, which we assume are not, or even cannot be, available), the more detailed study of them is omitted here.

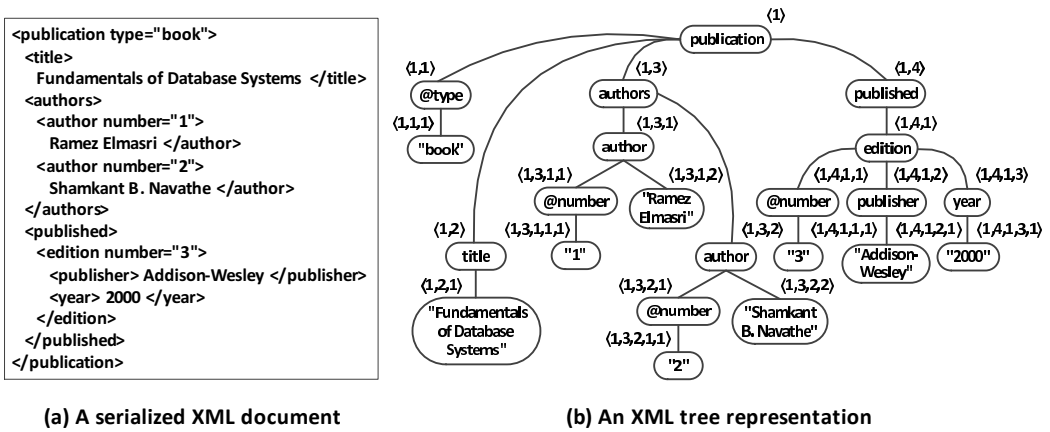

Figure 3.1: A serialized XML document and its corresponding tree representation.

ues.[19] In a well-formed XML document, one element occurrence, called a root element, contains the occurrences of all the other data items (i.e., elements and attributes) as well as their values in that document. Hence, the elements of an XML document form a nested (possibly recursive) structure: the document hierarchy. Syntactically, an occurrence of an element is a document fragment delimited by a start tag and an end tag, both of which carry the element's name; the former also carries attributes that are possibly attached to that element occurrence. The order in which the elements are placed in an XML document constitutes the document order. The document order corresponds to the left-to-right, depth-first traversal of a tree.

In order to enable efficient processing, the logical structures of XML documents must be indexed. The Dewey indexing scheme is a popular way to implement such structural indexing. In Dewey indexing,[20] the root of the document tree is equipped with the index $\langle 1 \rangle$, its first child with $\langle 1, 1 \rangle$, the first child of this with $\langle 1, 1, 1 \rangle$, and so on. Dewey indexing allows easy access to the ancestors and descendants of a specific data item. However, its drawbacks are the space required to store the indices and the potentially costly update operations (e.g., when a new first child is inserted, the entire sub-tree rooted by its parent needs to be re-indexed). An overview of alternative indexing schemes can be found in [42]. Figure 3.1 depicts a serialized XML document and its corresponding tree representation equipped with structural indices (attributes are marked with the prefix '@' and values within quotation marks).

[19] We see that attributes are, in essence, elements with textual/numeric contents. Compared to elements, attributes provide a more compact way to represent information; they are unordered by definition and unique within the element occurrence to which they are attached.

[20] We follow here, as well as throughout the publications comprising Part II of this dissertation, the notational conventions introduced in [64].

3.2 DTDs and XML Schemas

A DTD (Document Type Declaration; [35]) describing the structure of a class of valid documents is an integral part of an SGML document. Therefore, it is not surprising that this feature has subsequently also passed on to XML. DTDs are used to specify the names of elements whose occurrences may be included in documents conforming to it, what attributes may be attached to these elements, and the order and nesting among the elements. Later, the XML Schema Definition Language [89, 90] was introduced as an extension to DTDs with some considerable additional capabilities, most notably a typing system. While DTDs follows a BNF (Backus–Naur Form)-like syntax, the XML Schemas are themselves well-formed XML documents (and thus may have their own DTDs).

Unlike relational schemata, DTDs and XML Schemas are not intended to describe the structure actually present in a single XML document but to specify the valid structural alternatives for a set of documents. This has several consequences [22]. For example, the documents conforming to the same DTD/XML Schema may have significant variation with respect to their size and structure, depending on the degree of repetition and nesting allowed in them. Furthermore, through a DTD/XML Schema one can also specify that some elements and/or attributes are optional. Extensive use of such elements/attributes may, however, result in DTDs/XML Schemas becoming excessively all-inclusive, which threatens to reduce their usability in accurately describing the characteristics and state of a given document collection. Another potential usability-reducing factor is their large size. This is often the case with industry standards of which only some parts are used. Lastly and most importantly, as a recent survey in [43] suggests, a major part of the real-world XML documents are not associated with any DTD or XML Schema. In other words, most XML documents are, in this sense, schemaless. This is also our starting point in this dissertation.[21]

3.3 Path-Oriented Querying

Over the years, W3C, the organization responsible for developing XML, has introduced a number of technologies for querying and manipulating XML data. The most popular of them, XSLT and XQuery, are based on the use of path expressions specified by XPath. Next, we take a brief look at them.

[21] It should be noted that DTDs and XML Schemas are not solely a negative phenomenon. For example, the current XML Schema standard [89, 90] was developed in conjunction with the XPath 2.0 [88] and XQuery 1.0 [91] standards. Together they form a powerful toolbox for describing, manipulating, and querying XML data. However, as previously discussed, in this dissertation our starting point is that no external schemata are provided, so we do not deal with DTDs/XML Schemas even if they would be available.

3.3.1 XPath

XPath is an addressing mechanism for selecting data from XML documents. The current version of XPath, XPath 2.0 [88], differs considerably from its predecessor XPath 1.0 [87], with some of the basic concepts, such as the data model, typing system, and function libraries, being completely changed. In addition, XPath expressions return no longer sets, as in the previous version, but ordered sequences. This section gives a short overview of the main functionalities of XPath.

XPath models an XML document as a tree composed of nodes. Each node is of one of the seven node kinds: a document node, an element node, an attribute node, a text node, a comment node, a namespace node, or a processing instruction node. The aforementioned document node does not fully correspond to the root element in a serialized XML document because it also encompasses the possible comments and processing instructions that may precede the root element. The XPath expressions are always evaluated with respect to the node at hand, called the context node. The context node changes as the evaluation of an XPath expression moves on.

In XPath, path expressions are used to address the parts of XML documents. A path expression indicates the path that one must traverse from the context node so that the desired target node is reached. There are frequently multiple subtrees, which correspond to a single path expression. Hence, as the result of the evaluation of a single path expression, several nodes may be produced. Syntactically, a path expression consists of steps, separated by the symbol '/'. Path expressions can be combined with each other by set-theoretic operations (union, intersection, and difference). Each step, in turn, may contain three components: an axis, a node test, and/or a predicate.

An axis specifies the direction of navigation in the XML structure. A total of 13 different axes are defined: ancestor, ancestor-or-self, attribute, child, descendant, descendant-or-self, following, following-sibling, namespace, parent, preceding, preceding-sibling, and self. Except for the document node, every element and attribute node has exactly one parent. An element node may have zero or more children. The nodes having a common parent are siblings. The ancestors of a specific node are the nodes at the higher levels of the document hierarchy, defined as the transitive closure of the parent axis. Analogously, the nodes at the lower levels of the document hierarchy, defined as the transitive closure of the child axis, are its descendants.

Through node tests one specifies which nodes are selected by a given step. A node test is a condition that must be true for each node selected by a step. The condition may be based on the kind or the name of the node. For example, the name test book selects all element nodes whose name is *book*, whereas the kind test attribute(book) matches any attribute node whose name is *book*.

Nodes are filtered by predicates that select from the available nodes those that fulfill the given conditions. If there are multiple adjacent predicates, they are applied from left to right, and the result of one predicate may be used as the input for the following ones. Conditions that can be specified through a predicate include value comparisons, existential predicates that check whether a specific path expression returns a non-empty result, and positional predicates that select nodes based on their position in the document hierarchy. The latter applies only to elements, since attributes are, by definition, mutually unordered.

3.3.2 XSLT

XSLT (XSL Transformations; [93]) is part of the XSL (Extensible Stylesheet Language; [36]) family of W3C technologies for specifying transformations and presentation of XML documents. Due to its widespread use, XSLT is defined through its own independent specification. The term 'stylesheet' in its name reflects the fact that one of the major roles of XSLT is to add styling information to XML documents by transforming them into some presentation-oriented format, such as HTML or PDF (Portable Document Format). However, XSLT is not exclusively used for formatting and presentation applications, but also for generic transformation tasks.

A transformation in XSLT is expressed in the form of a stylesheet that follows the XML syntax. A stylesheet may include both XSLT-defined and other elements. The XSLT-defined elements are distinguished by use of their own specific namespace.[22] Being based on XPath, XSLT treats XML documents as trees. A transformation in XSLT describes a set of rules, through which zero or more source trees are transformed into one or more result trees. The transformation is produced by applying the so-called templates which contain patterns expressed in XPath that are matched against the nodes of the source tree. The structure of the result trees may be completely different from the structure of the original source trees in that the nodes from the source trees may be filtered and/or reordered and new structures may be added.

The mechanism described above allows a specific XSLT stylesheet to be applicable to a wide class of documents with similar source tree structures. On the other hand, for each class of documents with dissimilar source tree structures, a new stylesheet must be generated.

3.3.3 XQuery

XQuery [91] is not only a query language but also a full-scale programming language. The design of XQuery is influenced by several other languages,

[22] *XSLT Namespace*. Available at http://www.w3.org/1999/XSL/Transform (1999; accessed October 2013).

such as XQL [71], XML-QL [26], and Quilt [15]) intended for XML, Lorel [3] and YATL [19] developed for semi-structured data, as well as SQL and OQL (Object Query Language; [14]). Based on nearly a decade's work, XQuery finally achieved a W3C recommendation status in 2007.[23]

The core expressions of XQuery are collectively known by the acronym FLWOR, which comes from the initials of its component clauses: for, let, where, order by, and return. The for and let clauses are used to bind variables. The for clause binds the variables iteratively by traversing the element structures, whereas the let clause binds the variables directly to all occurrences that match the given expression. The optional where and order by clauses specify, respectively, the selection conditions and sorting criteria for the result. Lastly, the return clause is used to specify how the result is constructed based on the information produced by the other clauses.

In an XQuery query, the user can also specify conditional (*if–then–else*) and quantified expressions (with *some* and *any* quantifiers). A large set of built-in functions for XQuery is defined in [92], but users are also allowed to define functions of their own. Since these user-defined functions may be recursive or mutually recursive, XQuery is a Turing complete language [38]. XQuery is also a typed language and shares the type system of the XML Schema standard [90].

3.4 XML Relations

Owing to the fact that the effective use of the path-oriented XML languages described above presupposes from the user a considerable a priori knowledge of the structures, contents, and semantics of the underlying data sources, it has been acknowledged (e.g., [60, 94]) that they are not the most suitable tools for dealing with autonomous and beforehand unknown XML data sources. Moreover, these path-oriented languages also typically perform poorly in handling heterogeneity, because the manipulation of different classes of documents typically requires that new expressions are specified for each of them. What is more, the above languages presume that their user masters advanced programming techniques, such as pattern matching, iterative structures, and instantiation of procedural variables, which we, in accordance with [11], do not see as a reasonable requirement for an end-user satisfying his or her ad hoc information needs.

In this dissertation, we propose several solutions to overcome these limitations. For the most part, our work is based on XML relations. XML relations come from a relational representation of XML data, called XML relation representation, which is defined through a set-theoretic formalism that was origi-

[23] The latest (second) edition is from 2010.

nally introduced in [65] and further elaborated on in Papers IV and II. At the core of this formalism is a constructor algebra that facilitates, among others, the unambiguous and lossless conversion between the serialized and relational representations of XML.[24] In general, it is only possible to convert (structured) relational data to (semi-structured) XML data without losing some information. This is due to the mismatch between the principles of structural data organization in their underlying data models (i.e., flat versus hierarchical). Paper IV provides a detailed discussion of the similarities and differences between the conventional relations and the XML relations.

Since XML relations are structurally – but not conceptually – compatible with the relations of the relational model, we are able to utilize the storage and processing methods of the relational database systems in implementing the XML relations; we have done this in the PostgreSQL relational database management system environment in Paper V. Next, we briefly introduce some notations related to the XML relation representation. It should be noted that, although these notations and their related formalisms form the basis of the processing of XML data in the works presented in this dissertation, they remain invisible to the user in the front-end tools developed based on them.

Formally, an XML relation is a named ternary relation with the schema $D(C, T, I)$, in which D stands for the name of the XML document at hand; C is a data item occurrence (i.e., an element name, an attribute name, or a textual/numeric value) in D; T expresses the type of C (the valid alternatives are 'e', 'a', or 'v'); and I is a Dewey index expressing the exact and unambiguous location of C in the document hierarchy of D. The XML relation D consists of triples constructed in this way. Hence, for example, the expression $(c, t, ind) \in D$ refers to an arbitrary triple in D. By analyzing the values of T in the context of any triple, one is able decide whether C represents schema-level information (in this case $T \in \{$'a', 'e'$\}$) or instance-level information ($T =$ 'v'). If the contents of an attribute or element occurrence consist of a string in which the substrings (i.e., individual words and/or number sequences) have been delimited by space characters, then each of these substrings is individually indexed and represented as its own triple with the type 'v'.[25] Since an XML relation also carries the schema-level information, we need not construct or infer external schemata, but are able to utilize this information directly when analyzing previously unknown schemaless XML documents. The fact that an attribute or element name may have several occurrences in one XML document means that the corresponding XML relation contains triples with

[24] The constructor algebra has currently been defined for elements, attributes, and values (character data) only. See also Footnote 18.

[25] However, as the textual/numeric contents of the available attributes and elements are typically compact in data-centric applications, it is sometimes useful to be able to manipulate them as one, undivided string. For this purpose, in Paper II we introduce (and formally specify) the notion of *compound value*.

Table 3.1: The XML relation corresponding to the XML document in Figure 3.1

VALUE	TYPE	INDEX	VALUE	TYPE	INDEX
publication	'e'	⟨1⟩	author	'e'	⟨1,3,2⟩
type	'a'	⟨1,1⟩	number	'a'	⟨1,3,2,1⟩
book	'v'	⟨1,1,1⟩	2	'v'	⟨1,3,2,1,1⟩
title	'e'	⟨1,2⟩	Shamkant	'v'	⟨1,3,2,2⟩
Fundamentals	'v'	⟨1,2,1⟩	B.	'v'	⟨1,3,2,3⟩
of	'v'	⟨1,2,2⟩	Navathe	'v'	⟨1,3,2,4⟩
Database	'v'	⟨1,2,3⟩	published	'e'	⟨1,4⟩
Systems	'v'	⟨1,2,4⟩	edition	'e'	⟨1,4,1⟩
authors	'e'	⟨1,3⟩	number	'a'	⟨1,4,1,1⟩
author	'e'	⟨1,3,1⟩	3	'v'	⟨1,4,1,1,1⟩
number	'a'	⟨1,3,1,1⟩	publisher	'e'	⟨1,4,1,2⟩
1	'v'	⟨1,3,1,1,1⟩	Addison-Wesley	'v'	⟨1,4,1,2,1⟩
Ramez	'v'	⟨1,3,1,2⟩	year	'e'	⟨1,4,1,3⟩
Elmasri	'v'	⟨1,3,1,3⟩	2000	'v'	⟨1,4,1,3,1⟩

identical values in their C and T components but different values in the related I components. In other words, I is the key attribute of an XML relation. The XML relation representation corresponding to the XML document in Figure 3.1 is shown in Table 3.1.

Chapter 4

Summary of Contributions

4.1 Overview

The contributions of this dissertation can be considered against the framework provided in [44] that models data integration as a process consisting of four phases: (i) understanding, (ii) standardization, (iii) specification, and (iv) execution. As its name suggests, the purpose of the first phase is to increase the integrator's awareness of the structures, contents, and semantics of the underlying data sources. Traditionally, this means familiarization with database schemata and other available documentation (such as, entity–relationship [ER] diagrams [18]) or, in the case these are not available, generating them for review. In standardization, the next phase in the integration process, the key objective is to determine the common form, in which the data to be integrated are structured and represented. This phase is closely connected to the subsequent phase of specification, in which the artifacts (i.e., schema mappings and/or data translations), through which the data are brought into the chosen uniform form, are specified. In the last phase of the process, these artifacts are then executed (based on mediation or materialization) in order to provide unified access to the data of interest.

The above four-phased framework, designed to describe the traditional expert-oriented data integration process in closed information environments, is, however, not able to accommodate our case of serving the user's sophisticated ad hoc information needs based on beforehand unknown, autonomous and heterogeneous data sources. For example, when operating in this kind of information environment, the first task is usually to locate the data sources that might be useful with respect to one's information needs. Additionally, as discussed in Chapter 2, since we consider ad hoc information needs, it is often not an option to either generate schemata to describe the available data sources or to apply conventional data integration techniques. We believe that ad hoc information needs are best served through a powerful query language

Table 4.1: Summary of the contributions.

PHASE	CONTRIBUTION(S)	PUBLICATION(S)
1. Searching	Entity ranking	Paper I
2. Profiling	XML dataspace profiling	Paper II (Paper IV)
3. Harmonizing	(a) Structure-unaware data extraction (b) XML data harmonization	Paper III, Paper IV (Paper V, Paper II)
4. Querying	RXQL	Paper V (Paper III, Paper IV)

that is designed specifically for dealing with heterogeneous data sources and that combines in a seamless and straightforward manner data integration with typical data-centric manipulation, such as restructuring, grouping, ordering, and aggregation. In Paper V, we introduce such a language.

However, before the user is able to use this kind of a query language, some preparatory steps need to be taken. As stated, in our scenario the user's first task is usually to find, among the data sources in the underlying XML dataspace, those data sources based on which his or her ad hoc information needs can be best served. For this, techniques and tools from (XML) information retrieval can be applied. Through them, a collection of potentially useful data sources can be produced for the user. The logical next step is to provide the user with the means to assess the actual suitability of the retrieved data sources from the view point of his or her information needs. Since the available data sources are assumed to be autonomous and heterogeneous, it is unlikely that they could be utilized as such for serving the user's information needs. Therefore, the user must also be provided with the means to detect the potential inconsistencies among (and within) them as well as to overcome or remove the inconsistencies that are met. Only after these steps have been completed is the user able to specify and pose meaningful queries through which his or her sophisticated ad hoc information needs are satisfied.

The above is a rough description of how we in this dissertation conceive the process of serving the user's sophisticated ad hoc information needs based on beforehand unknown, autonomous and heterogeneous XML data sources. Table 4.1 lists the four phases of our framework – which are *searching*, *profiling*, *harmonizing*, and *querying* – as well as our contributions to each of the phases with the publications in which they are covered.[26] Since many of these con-

[26] The publications marked with parentheses bring some additional contribution to the other publications.

tributions are covered by multiple publications, in the following we examine them per phase rather than per publication.

4.2 Roles of the Original Publications

4.2.1 Searching: Entity Ranking

In Paper I, we develop methods that can be used to assist the user in finding among the data sources of the underlying dataspace those data sources that are potentially useful with respect to his or her information needs. Specifically, Paper I considers entity ranking, which aims to retrieve and rank entities matching a given keyword query. The entities in question may be concrete objects (e.g., people, places, and organizations) or abstract concepts (e.g., mathematical formulae). In retrieving entities, entity ranking differs from the traditional information retrieval (IR) whose focus is on finding topically relevant documents or their parts. In this sense, entity ranking resembles database querying in which facts (records) rather than documents are retrieved. Entity ranking also has similarities with information extraction (see, [73]), which develops techniques for automatically extracting structured information from unstructured sources in order to provide richer forms of querying unstructured data than simple keyword queries.

Ranking is a central concept in information retrieval. The idea in ranking is to display all the retrieved information items (i.e., documents, document fragments, or individual data items in them), but to order them according to some criterion [46]. In information retrieval, the criterion typically is the relevance of the retrieved information item with respect to the information need at hand. Hence, ranking differs from the exact-match approach of database querying, in which query results should contain only those data items that match (all) the formally expressed query conditions. In this respect, IR-style querying can be characterized as a best-effort approach.

In Paper I, two methods for entity ranking are developed. These methods were implemented and evaluated within the framework of the XML Entity Ranking (XER) track (see, [83]) of the INEX 2007 workshop that provided the corpus of 659,000 Wikipedia articles in XML format. The available articles were classified using approximately 113,000 categories, and some 13,900,000 links existed between the articles. The entities to be retrieved were the titles of these articles. We contributed to the both tasks of the XER track: Entity Ranking and List Completion. In the former task, the goal was to produce a ranked list of entities that match a topic expressed as a natural language sentence (e.g., "countries where I can pay with Euros"). In the latter, the task was to complete a list when two to three sample entities and a free-text description about the desired result were given. The methods we developed were category

expansion and link expansion. Category expansion is a coefficient propagation method for the Wikipedia category hierarchy based on given categories or categories derived from the sample entities, whereas link expansion utilizes the link structure of the underlying Wikipedia collection. The developed methods performed fair in comparison with the proposals of the other track participants (reaching the position 5 of 8 in the Entity Ranking task and shared 2 of 6 in the List Completion task; [83]).

It should be noted that, in producing entities rather than documents, entity ranking cannot be used as a stand-alone method for retrieving the data sources based on which the user's information needs are served. However, the retrieved entities can be considered to form a list of ranked keywords, which, in turn, can be used as inputs to conventional search tools through which the potentially useful data sources can be retrieved. In other words, we believe that entity ranking provides means for (semi-)automatic query generation, and thus simplifies the user's search task.

4.2.2 Profiling: XML Dataspace Profiling

As the output of the previous phase of our framework, the user now has a collection of XML data sources that are potentially useful with respect to his or her information need. Hence, the next step is to develop means that enable the user to assess the actual usefulness of the retrieved data sources. Paper II studies this issue.

In the spirit of [49], we refer to the user's activity of assessing the suitability of available XML data sources as XML dataspace profiling. Through XML dataspace profiling, the user is able (i) to analyze the information contents of the available data sources at the schema and instance levels; (ii) to discover the structural relationships among their data items; and (iii) to attach the correct intended meaning to the names of these data items. In Paper II, we developed (formal) tools for the above purposes. In the course of XML dataspace profiling, the user is able to discard those data sources that he or she considers irrelevant, and the process results in a collection of data sources based on which the user's information needs can be served.

Since in our scenario the data are drawn from autonomous and heterogeneous XML data sources, it is unlikely that they could be utilized as such for serving the user's information needs. Therefore, the user also needs to be able to assess the mutual (and internal) consistency of the data sources at hand. This means that he or she must be able to detect the types of heterogeneity that may possibly occur in them. Generally speaking, three types of heterogeneity may occur in XML data: structural, semantic, and syntactic. Structural heterogeneity refers to different structural organizations of semantically equivalent information (i.e., different document hierarchies). Differences

in naming conventions (e.g., *customer* vs. *client*) result, in turn, in semantic heterogeneity. Sources of syntactic heterogeneity are, for example, the differences in the representations (e.g., *m/f* vs. *male/female*) and interpretations (e.g., *EUR* vs. *USD*) of the same values. From the perspective of an individual data source, these types of heterogeneity manifest themselves as data conflicts.

To the best of our knowledge, Paper IV is the first work to identify and classify the data conflict types that may occur in XML data. The XML data conflicts can be characterized as follows:[27]

1. *Document-to-document conflicts* occur when the same information content is organized based on different XML structures.
2. *Attribute-to-element conflicts* occur when the same piece of information is represented both as an attribute and as an element.
3. *Data item-to-data item conflicts* occur when semantically equivalent attributes/elements are labeled differently (or semantically non-equivalent attributes/elements are labeled identically).
4. In *data item-to-value conflicts*, the semantically equivalent piece of information is represented both as a data item name and as a data item value.
5. In *value-to-value conflicts*, the semantically equivalent data item values are represented non-uniformly (or semantically non-equivalent data item values are represented identically).

In addition to developing tools for assessing the suitability of the retrieved XML data sources as discussed above, in Paper II we also developed tools for detecting potential data conflicts among them. However, it should be noted that these tools do not automatically guarantee that actual data conflicts are met but merely indicate the existence of potential ones. The verification of whether or not a data conflict actually occurs is left to the consideration of the user.

4.2.3 Harmonizing: Structure-Unaware Data Extraction and XML Data Harmonization

In the next phase of our framework, the user needs to be provided with the means to overcome and/or remove the heterogeneity that was detected among data sources at hand. For this purpose, we introduce two kinds of solutions: structure-unaware data extraction and XML data harmonization.

Structure-Unaware Data Extraction. As discussed in Chapter 3, in querying XML data sources XSLT and XQuery rely on the extensive use of XPath path expressions. On one hand, path expressions enable targeting queries directly at the desired data items but, on the other hand, their effective use presupposes from the user a considerable a priori knowledge of the structures,

[27] We follow here the terminology revised in Paper II.

contents, and semantics of the underlying data sources. This requirement can be considered as problematic when dealing with beforehand unknown, autonomous and heterogeneous XML data sources.

A well-known means to specify meaningful queries with limited knowledge of the underlying data sources is to make keyword searches. Hence, an obvious solution to overcome the knowledge-based requirements of path-oriented XML query languages is to develop mechanisms that combine the expressiveness of the structured query approach with the ease-of-use of the keyword search approach. This has been the goal in the approaches that incorporate LCA (Lowest Common Ancestor)-based query evaluation into structured XML query languages. LCA (see, [5]) refers to a mechanism of restricting the query evaluation in tree-structured data into those subtrees whose roots are the lowest common ancestors of the nodes associated with the search terms. The adoption of this kind of query evaluation strategy will not only raise the degree of declarativeness of the underlying query language by freeing the user from specifying the explicit navigation in the document structure, but also make it possible to use the same query over multiple, differently structured data sources.[28] In other words, XML query languages with LCA-based query evaluation mechanisms can be used to overcome the structural heterogeneity.

In Paper III, we examined XML query evaluation based on two restrictions to the original LCA semantics. MLCA (Meaningful LCA; [60]) and SLCA (Smallest LCA; [94]) semantics modify the original LCA semantics by setting the condition that the subtrees to be retrieved must be minimal, meaning that they must not contain any other subtrees that would match all the given search terms. (This is allowed by the original LCA semantics.) However, what we discovered was that even this restriction was not sufficient to prevent undesired results in some, not so uncommon, cases. Particularly, we found that MLCA and SLCA semantics produce undesired results when

1. the XML document at hand contains parts that match different subsets of search terms; and/or
2. nodes at higher levels of the document hierarchy need to be replicated with lower-level nodes for the query result.

In order to avoid these shortcomings, we developed the Smallest Possible Context (SPC) query evaluation strategy. It produces meaningful results in all the cases in which the MLCA and SLCA semantics do so, but also in the two special cases above. In Paper III, the SPC query evaluation strategy was further implemented as a part of a high-level XML data extraction primitive based on the notion of the variable of logic programming and deductive databases. Through this primitive, we are able to overcome the document-to-document and attribute-to-element conflicts among structurally heterogeneous

[28] This latter applies only on the condition that the naming conventions in the available data sources are the same.

XML documents. Later, in Papers IV and V, we respectively formalized the SPC query evaluation strategy based on XML relations and re-implemented it as a part of the RXQL query language.

XML Data Harmonization. As discussed in Chapter 2, in the context of structured and semi-structured data, two levels of abstraction are distinguished, the schema level and the instance level. Depending on modeling practices, the same piece of information can be presented at either of these levels. For example, in XML the information about books may be presented within an element named *book* (<book> ... </book>; i.e., at the schema level) in some document, whereas the same information may be presented in some other document as a value of a *type* attribute within a *publication* element (<publication type="book">; i.e., at the instance level).[29] This type of heterogeneity cannot be removed by conventional data translation techniques which employ schema mappings to specify the relationships between the instance data conforming to different schemata. These techniques are, in other words, capable of producing only data-to-data translations, whereas in the above case data-to-metadata translations are needed.

Although operations for data-to-metadata translations have been developed in the context of relational databases (see, Section 2.2), producing data-to-metadata translations in XML is, however, a non-trivial task because of the hierarchical structuring of data [48]. Being a computationally complete language, XQuery is naturally capable of producing data-to-metadata translations. However, as discussed above, XQuery is not the optimal tool for dealing with autonomous and heterogeneous data sources due to the degree of a priori knowledge of the underlying data sources its effective use presupposes from the user. For this reason, a different approach to produce data-to-metadata translations among these kinds of XML data sources should to be developed.

In Paper IV, we defined, based on XML relations, a formal methodology for producing data-to-metadata translations in XML. Through data-to-metadata translations one is able to resolve the data item-to-value conflicts among XML data. In Paper IV, we further showed how the document-to-document and attribute-to-element conflicts can be resolved based on XML relations through data restructuring. Moreover, the query specification mechanism of the RXQL query language introduced in Paper V enables the user to resolve the data item-to-data item conflicts, whereas in Paper II we outlined how the value-to-value conflicts are resolved based on XML relations. To distinguish our comprehensive XML data translation methodology – with

[29] This kind of heterogeneity can, depending on the view point, be considered as either structural or semantic. It is structural in the sense that the same piece of information is represented with different structural constructs (element vs. value). On the other hand, it is semantic since it represents the problem of hypernyms/hyponyms. Namely, in the latter case, a more general term (*publication*) is used to denote a more specific term (*book*), whereas in the former case, a more specific term is used to denote a more general term.

the core capability of producing data-to-metadata translations – from the conventional data translation methodologies, we call our approach XML data harmonization.

4.2.4 Querying: RXQL

As stated, we believe that ad hoc information needs are best served through a powerful query language. Hence, in Paper V we developed a relational XML query language, called RXQL, which is specifically designed to extract, select, rename, restructure, order, group, and aggregate heterogeneous XML data. In other words, it facilitates both data integration and conventional data-centric manipulation. To the best of our knowledge, RXQL is the first XML query language tailored for dealing with heterogeneous XML data sources. Being based on XML relations and implemented as a Java front-end tool on top of a PostgreSQL relational database management system, RXQL incorporates the techniques introduced in Papers III (SPC) and IV (XML data harmonization).

RXQL is a declarative XML query language with two principal design goals. First, in the query specification of RXQL the literal XML syntax is not used. In this, we have been influenced by the findings, for example, in [34, 10] that suggest that the XML syntax is often unfriendly and verbose from the perspective of the user and should thus be avoided. Second, the query specification in RXQL is not based on the use of the conventions from procedural programming languages, such as iterative control structures or explicit variable instantiations. In designing RXQL, our ideal has been the SQL-style query specification, in which the user needs only to specify *what* he or she wants to get as a query result without needing to pay great attention to *how* the result is actually constructed from the available data.

An RXQL query consists of CONSTRUCT, FROM, WHERE, ORDER BY, and GROUP BY clauses, of which only the first two are mandatory. In the CONSTRUCT clause, the user specifies, as a linearized hierarchy, the structure of the query result and the relationships between the elements and attributes in it. The data from the source documents are extracted in the FROM clause through expressions that utilize the SPC query evaluation strategy. Apart from specifying which of the data items are to be extracted, the user can also rename data items and, thus, resolve the potential data item-to-data item conflicts. Through the WHERE clause, the user is able to filter out the undesired data item values from the query result and specify simple aggregations. The ORDER BY and GROUP BY clauses are used to order and group the information in the result, respectively. Through a set of sample queries, we further showed the advances of RXQL query specification by comparing it to the corresponding XQuery queries.

Chapter 5

Discussion and Conclusions

In this dissertation, we studied serving of the user's sophisticated ad hoc information needs in modern information environments. Data integration is often needed in serving sophisticated information needs which typically involve comparing and analyzing data combined from disparate data sources. However, we showed that the starting points of data integration in modern information environments composed of beforehand unknown, autonomous and heterogeneous data sources are considerably different from those in closed information environments, which have thus far provided the standard for developing data integration solutions. Therefore, it becomes necessary to develop new approaches that take these altered starting points into account. For this purpose, in this dissertation we present several novel methods, techniques, and tools which devise a four-phased framework for serving the user's sophisticated ad hoc information needs based on beforehand unknown, autonomous and heterogeneous XML data sources.

Our work shares many of its characteristics with the dataspace approach, although we explicitly refer to it only in Paper II. For example, analogously to our general background assumption, the dataspace approach starts from the premise that the starting points of the traditional data integration approaches are no longer valid in emerging information environments. It is further acknowledged in the dataspace approach that, due to the nature of the available data sources as well as the users' information needs, more lightweight solutions are needed compared to those provided by the traditional data integration approaches. Developing such tools is also one of the starting points in this dissertation. However, our work does not strictly follow the dataspace approach as originally envisioned in [39, 45]. For example, the dataspace approach calls for multi-format support, whereas we believe that the corresponding general applicability can be achieved by adopting XML as the common data format.

Excluding Papers I and III, in the works presented in this disseration the available XML data sources are represented and processed as XML relations.

XML relations are based on a set-theoretic formalism, called the XML relation representation, which was originally introduced in [65] and is further elaborated on in Papers IV and II. The XML relation representation affords a lossless conversion between a serialized XML document and its corresponding XML relation representation (and vice versa). Furthermore, as XML relations are structurally compatible with the relations of the relational model, the relational database technology can be used for processing XML relations. This dissertation shows that the XML relation representation provides a good starting point for developing applications dealing with beforehand unknown, autonomous and heterogeneous XML data sources. It should also be noted that the indexing and index manipulation mechanisms related to the XML relation representation are utilized in all the five publications.

The work presented in this dissertation belongs (for the most part) to the theoretic-constructive tradition of research in computer science. This means that we identify problems which are then solved by constructing and implementing mechanisms and tools. The mechanisms and tools developed in this dissertation are often both specified formally and implemented programmatically. The XML relation representation affords a good starting point for both purposes. For example, the mechanisms presented in Paper II are formal, whereas the query language in Paper V is implemented utilizing the relational database technology.[30] Since we regard our work as basic research in which the emphasis is on effectiveness rather than efficiency, the proposed tools have been developed only to the prototype level and only little attention has been paid, for example, to performance issues or optimization.

This dissertation does not intend to provide a comprehensive solution to the problems related to the integration of XML-based data sources. For this reason, a number of important issues affecting real-world data integration scenarios have been left outside the scope of this dissertation. These include, among others, the following:

- Data quality. When data are used for decision-making, one must be able to ensure the data is of high quality, meaning that they are (ideally) complete and do not contain errors or redundancy.
- Entity matching. As briefly discussed in Section 2.2, (semi-)automatically deciding on whether two different data item names or values are used to denote the same real-world entities (or two identical names/values denote different entities) is a challenging problem. However, mostly due to the diversity and complexity of the natural language based on which these names and values are represented, this is a common issue in every-day data integration scenarios.
- Data provenance.[31] When data from autonomous and heterogeneous data sources are used for decision making, it becomes increasingly important to be able to trace their origins and life span stages in order to ensure their reliability.

[30] However, the mechanisms in Paper II were later implemented programmatically in [63], and Paper V also includes the formal specification for the developed query language.

[31] Synonym: data lineage.

- Information visualization and visual front-end tools. The utilization of large datasets is practically unfeasible unless the user is provided with visualizations over the underlying data. Similarly, the user requirements of today are usually such that tasks should be able to be executed through visual front-end tools.

It should be emphasized that some of the tools presented in this dissertation, especially those in Papers, I, II, and III, can also be used for purposes other than data integration. For example, the methods presented in Paper I can be used for independent tasks of retrieving and ranking entities, or, as suggested, for (semi-)automatically generating queries in order to retrieve data sources of interest. Similarly, the XML dataspace profiling approach presented in Paper II can used as a general tool for exploring the structures, contents, and semantics of the underlying XML data sources as well as assessing their consistency. Finally, the SPC query evaluation strategy introduced in Paper III can be used as a general tool for retrieving meaningful data items from structurally heterogeneous XML data sources. Additionally, we also showed in that paper how the adoption of the notion of variable of logic programming and deductive databases allows us to construct expressions in which different query components can be combined in a way that frees the user from taking care of variable instantiations, as it is the case in the context of languages based on procedural programming.

To conclude, this dissertation presents some novel methods, techniques, and tools for integrating and manipulating data in heterogeneous XML-based environments. First, we introduced methods that can be used to assist the user in finding the potentially useful data sources from those available. Second, we developed a methodology through which the user is able to assess the actual usefulness of the available, beforehand unknown data sources with respect to his or her ad hoc information needs. The methodology also includes tools for assessing the consistency of the data sources of interest. Third, we developed a query evaluation strategy that allows the user to make meaningful queries over the available XML data sources without needing to pay attention to the possible structural heterogeneity among them. Furthermore, we specified a comprehensive methodology, called XML data harmonization, for producing XML data translations, with the core capability of producing data-to-metadata translations in XML data. Finally, we designed and implemented a powerful, declarative relational XML query language, called RXQL, which combines data integration with typical data-centric manipulation in a seamless and straightforward manner. Collectively, the methods, techniques, and tools proposed in this dissertation devise a four-phased framework for serving the user's sophisticated ad hoc information needs based on beforehand unknown, autonomous and heterogeneous XML data sources.

References

[1] Serge Abiteboul. Querying semi-structured data. In: *Proceedings of the 6th International Conference on Database Theory*. LNCS **1186**, 1997: 1–18.

[2] Serge Abiteboul, Sophie Cluet, and Tova Milo. Correspondence and translation for heterogeneous data. *Theoretical Computer Science* **275** (1-2), 2002: 179–213.

[3] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet L. Wiener. The Lorel Query Language for Semistructured Data. *International Journal on Digital Libraries* **1** (1), 1997: 68–88.

[4] Russell L. Ackoff. From data to wisdom. *Journal of Applied Systems Analysis* **16**, 1989: 3–9.

[5] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. On finding lowest common ancestors in trees. In: *Proceedings of the 5th Annual ACM Symposium on Theory of Computing*, 1973: 253–265.

[6] Maurizio Atzori and Nicoletta Dessí. Dataspaces: Where structure and schema meet. In M. Biba, F. Xhafa (eds.): *Learning Structure and Schemas from Documents*. SCI **371**, 2011: 97–119.

[7] Carlo Batini, Maurizio Lenzerini, and Shamkant B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys* **18** (4), 1986: 323–364.

[8] Khalid Belhajjame, Norman W. Paton, Suzanne M. Embury, Alvaro A. A. Fernandes, and Cornelia Hedeler. Feedback-based annotation, selection and refinement of schema mappings for dataspaces. In: *Proceedings of the 13th International Conference on Extending Database Technology*. AICPS **426**, 2010: 573–584.

[9] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American* **284** (5), May, 2001: 34–43.

[10] Claus Brabrand, Anders Møller, and Michael I. Schwarzbach. Dual syntax for XML languages. *Information Systems* **33** (4-5), 2008: 385–406.

[11] Katrin Braunschweig, Julian Eberius, Maik Thiele, and Wolfgang Lehner. OPEN: Enabling non-expert users to extract, integrate, and analyze open data. *Datenbank-Spektrum* **12** (2), 2012: 121–130.

[12] Peter Buneman. Semistructured data. In: *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1997: 117–121.

[13] Peter Buneman, Susan B. Davidson, and Dan Suciu. Programming constructs for unstructured data. In: *Proceedings of the 5th International Workshop on Database Programming Languages*, 1996: 12.

[14] Roderick G. G. Cattell, and Douglas K. Barry (eds.): *The Object Data Standard: ODMG 3.0*. Morgan Kaufmann, 2000.

[15] Don Chamberlin, Jonathan Robie, and Daniela Florescu. Quilt: an XML query language for heterogeneous data sources. In: Dan Suciu, Gottfried Vossen (eds.): *The 3rd International Workshop on The World Wide Web and Databases* (Selected Papers). LNCS **1997**, 2000: 1–25.

[16] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and OLAP technology. *SIGMOD Record* **26** (1), 1997: 65–74.

[17] Surajit Chaudhuri, Umeshwar Dayal, and Vivek R. Narasayya. An overview of business intelligence technology. *Communications of the ACM* **54** (8), 2011: 88–98.

[18] Peter P. Chen. The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems* **1** (1), 1976: 9–36.

[19] Sophie Cluet, Claude Delobel, Jérôme Siméon, and Katarzyna Smaga. Your mediators need data conversion! *ACM SIGMOD Record* **27** (2), 1998: 177–188.

[20] Edgar F. Codd. A relational model of data for large shared data banks. *Communications of the ACM* **13** (6), 1970: 377–387.

[21] Edgar F. Codd. Relational completeness of database sublanguages. In: *Data Base Systems (Courant Computer Science Symposium 6)* l, 1972: 65–98.

[22] Mariano P. Consens, Flavio Rizzolo, and Alejandro A. Vaisman. AxPRE summaries: Exploring the (semi-) structure of XML Web collections. In: *Proceedings of the 24th International Conference on Data Engineering*, 2008: 1519–1521.

[23] Conor Cunningham, Goetz Graefe, and César A. Galindo-Legaria. PIVOT and UNPIVOT: Optimization and execution strategies in an RDBMS. In: *Proceedings of the 30th International Conference on Very Large Data Bases*, 2004: 998–1009.

[24] Umeshwar Dayal. Processing queries over generalization hierarchies in a multidatabase system. In: *Proceedings of the 9th International Conference on Very Large Data Bases*, 1983: 342–353.

[25] *Data Base Task Group Report to the CODASYL Programming Language Committee (April 1971)*. ACM, 1971.

[26] Alin Deutsch, Mary Fernández, Daniela Florescu, Alon Levy, and Dan Suciu. A query language for XML. In: *Proceedings of the 8th International World Wide Web Conference*, 1999: 77–91.

[27] AnHai Doan, Alon Y. Halevy, and Zachary G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[28] Xin Luna Dong and Alon Y. Halevy. Indexing dataspaces. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2007: 43–54.

[29] Carina F. Dorneles, Rodrigo Gonçalves, and Ronaldo dos Santos Mello. Approximate data instance matching: A survey. *Knowledge and Information Systems* **27** (1), 2011: 1–21.

[30] Julian Eberius, Katrin Braunschweig, Maik Thiele, and Wolfgang Lehner. Identifying and weighting integration hypotheses on Open Data Platforms. In: *Proceedings of the 1st International Workshop on Open Data*, 2012: 22–29.

[31] Julian Eberius, Maik Thiele, Katrin Braunschweig, and Wolfgang Lehner. DrillBeyond: Enabling business analysts to explore the Web of Open Data. *Proceedings of the VLDB Endowment* **5** (12), 2012: 1978–1981.

[32] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering* **19** (1), 2007: 1–16.

[33] Ibrahim Elsayed and Peter Brezany. Towards large-scale scientific dataspaces for e-Science applications. In: *Proceedings of the 15th International Conference on Database Systems for Advanced Applications Workshops*. LNCS **6193**, 2010: 69–80.

[34] Martin Erwig. Xing: A visual XML query language. *Journal of Visual Languages and Computing* **14** (1), 2003: 5–45.

[35] *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. W3C Recommendation. Available at http://www.w3.org/TR/xml (2008, accessed October 2013).

[36] *Extensible Stylesheet Language (XSL): Version 1.1*. W3C Recommendation. Available at http://www.w3.org/TR/xsl11/ (2006, accessed October 2013).

[37] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. *Theoretical Computer Science* **336** (1), 2005: 89–124.

[38] Mary Fernández and Jérôme Siméon. Growing XQuery. In: *Proceedings of the 17th European Conference on Object-Oriented Programming*. LNCS **2743**, 2003: 405–430.

[39] Michael J. Franklin, Alon Y. Halevy, and David Maier. From databases to dataspaces: A new abstraction for information management. *SIGMOD Record* **34** (4), 2005: 27–33.

[40] Michael J. Franklin, Alon Y. Halevy, and David Maier. A first tutorial to dataspaces. *Proceedings of the VLDB Endowment* **1** (2), 2008: 1516–1517.

[41] Norbert Fuhr and Kai Großjohann. XIRQL: An XML query language based on information retrieval concepts. *ACM Transactions on Information Systems* **22** (2), 2004: 313–356.

[42] Gang Gou and Rada Chirkova. Efficiently querying large XML data repositories: A survey. *IEEE Transactions on Knowledge and Data Engineering* **19** (10), 2007: 1381–1403.

[43] Steven Grijzenhout and Maarten Marx. The quality of the XML web. In: *Proceedings of the 20th ACM Conference on Information and Knowledge Management*, 2011: 1719–1724.

[44] Laura M. Haas. Beauty and the Beast: The theory and practice of information integration. In: *Proceedings of the 11th International Conference on Database Theory*. LNCS **4353**, 2007: 28–43.

[45] Alon Y. Halevy, Michael J. Franklin, and David Maier. Principles of dataspace systems. In: *Proceedings of the 25th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 2006: 1–9.

[46] Tom Heath and Christian Bizer. Linked data: Evolving the Web into a global data space. *Synthesis Lectures on the Semantic Web: Theory and Technology* **1** (1), 2011: 1–136.

[47] Dennis Heimbigner and Dennis McLeod. A federated architecture for information management. *ACM Transactions on Office Information Systems* **3** (3), 1985: 253–278.

[48] Mauricio A. Hernández, Paolo Papotti, and Wang Chiew Tan. Data exchange with data-metadata translations. *Proceedings of the VLDB Endowment* **1** (1), 2008: 260–273.

[49] Bill Howe, David Maier, Nicolas Rayner, and James Rucker. Quarrying dataspaces: Schemaless profiling of unfamiliar information sources. In: *Proceedings of the 24th International Conference on Data Engineering Workshops*, 2008: 270–277.

[50] *HyperText Markup Language (HTML): ISO/IEC 15445:2000*. International Organization for Standardization / International Electrotechnical Commission, 2000.

[51] William H. Inmon. *Building the Data Warehouse (1st Edition)*. Wiley, 1992.

[52] Shawn R. Jeffery, Michael J. Franklin, and Alon Y. Halevy: Pay-as-you-go user feedback for dataspace systems. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2008: 847–860.

[53] Grigoris Karvounarakis. Datalog. In L. Liu, M. T. Özsu (eds.): *Encyclopedia of Database Systems*, 2009: 751–754.

[54] Ralph Kimball and Joe Caserta. *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. Wiley, 2004.

[55] Phokion G. Kolaitis. Schema mappings, data exchange, and metadata management. In: *Proceedings of the 24th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 2005: 61–75.

[56] Hanna Köpcke and Erhard Rahm. Frameworks for entity matching: A comparison. *Data & Knowledge Engineering* **69** (2), 2010: 197–210.

[57] Laks V. S. Lakshmanan, Fereidoon Sadri, and Subbu N. Subramanian. SchemaSQL: An extension to SQL for multidatabase interoperability. *ACM Transactions on Database Systems* **26** (4), 2001: 476–519.

[58] Fenareti Lampathaki, Spiros Mouzakitis, George Gionis, Yannis Charalabidis, and Dimitris Askounis. Business to business interoperability: A current review of XML data integration standards. *Computer Standards & Interfaces* **31** (6), 2009: 1045–1055.

[59] Maurizio Lenzerini. Data integration: A theoretical perspective. In: *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2002: 233–246.

[60] Yunyao Li, Cong Yu, and H. V. Jagadish. Schema-free XQuery. In: *Proceedings of the 13th International Conference on Very Large Data Bases*, 2004: 72–83.

[61] Jayant Madhavan, Shirley Cohen, Xin Luna Dong, Alon Y. Halevy, Shawn R. Jeffery, David Ko, and Cong Yu: Web-scale data integration: You can afford to pay as you go. In: *Proceeding of the 3rd Biennial Conference on Innovative Data Systems Research*, 2007: 342–350.

[62] William C. McGee. The information management system IMS/VS, part I: General structure and operation. *IBM Systems Journal* **16** (2), 1977: 84–95.

[63] Katja Moilanen, Timo Niemi, Mikko Kuru, and Turkka Näppilä. *A Visual XML Dataspace Approach for Satisfying Ad Hoc Information Needs*. Submitted for review.

[64] Timo Niemi. A seven-tuple representation for hierarchical data structures. *Information Systems* **8** (3), 1983: 151–157.

[65] Timo Niemi and Kalervo Järvelin. *Another Look at XML.* Technical Report **A-2006-1**, University of Tampere, Department of Computer Sciences, 2006.

[66] Byung-Won On, Ingyu Lee, and Dongwon Lee. Scalable clustering methods for the name disambiguation problem. *Knowledge and Information Systems* **31** (1), 2012: 129–151.

[67] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object exchange across heterogeneous information sources. In: *Proceedings of the Eleventh International Conference on Data Engineering*, 1995: 251–260.

[68] Vassilios Peristeras, Konstantinos Tarabanis, and Sotirios K. Goudos. Model-driven eGovernment interoperability: A review of the state of the art. *Computer Standards & Interfaces* **31** (4), 2009: 613–628.

[69] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal* **10** (4), 2001: 334–350.

[70] *Resource Description Framework (RDF).* W3C Recommendation. Available at http://www.w3.org/RDF/ (2004, accessed October 2013).

[71] Jonathan Robie, Joe Lapp, and David Schach. *XML Query Language (XQL).* Available as http://www.w3.org/TandS/QL/QL98/pp/xql.html (1998, accessed October 2013).

[72] Marcos Antonio Vaz Salles, Jens-Peter Dittrich, Shant Karakashian, Olivier René Girard, and Lukas Blunschi. iTrails: Pay-as-you-go information integration in dataspaces. In: *Proceedings of the 33rd International Conference on Very Large Data Bases*, 2007: 663–674.

[73] Sunita Sarawagi. Information Extraction. *Foundations and Trends in Databases* **1** (3), 2008: 261–377.

[74] Anish Das Sarma, Xin Dong, and Alon Y. Halevy. Bootstrapping pay-as-you-go data integration systems. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2008: 861–874.

[75] Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys* **22** (3), 1990: 183–236.

[76] David W. Shipman. The functional data model and the data language DAPLEX. *ACM Transactions on Database Systems* **6** (1), 1981: 140–173.

[77] Kurt A. Shoens, Allen Luniewski, Peter M. Schwarz, James W. Stamos, and Joachim Thomas II. The Rufus system: Information organization for semi-structured data. In: *Proceedings of the 19th International Conference on Very Large Data Bases*, 1993: 97–107.

[78] John Miles Smith, Philip A. Bernstein, Umeshwar Dayal, Nathan Goodman, Terry A. Landers, Ken W. T. Lin, and Eugene Wong. Multibase: Integrating heterogeneous distributed database systems. In: *Proceedings of the AFIPS National Computer Conference*, 1981: 487–499.

[79] *Standard Generalized Markup Language (SGML): ISO 8879:1986.* International Organization for Standardization, 1986.

[80] *Structured Query Language (SQL): ISO/IEC 9075(1-4,9-11,13,14):2011.* International Organization for Standardization / International Electrotechnical Commission, 2011.

[81] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems. Volume 1.* Computer Science Press, 1988.

[82] Panos Vassiliadis and Alkis Simitis. Extraction, transformation, and loading. In L. Liu, M. T. Özsu (eds.): *Encyclopedia of Database Systems*, 2009: 1095–1101.

[83] Arjen P. de Vries, Anne-Marie Vercoustre, James A. Thom, Nick Craswell, and Mounia Lalmas. Overview of the INEX 2007 Entity Ranking Track. In Norbert Fuhr et al. (eds.): *Focused Access to XML Documents: The 6th International Workshop of the Initiative for the Evaluation of XML Retrieval* (Selected Papers). LNCS **4862**, 2008: 245–251.

[84] Jennifer Widom. Integrating heterogeneous databases: Lazy or eager? *ACM Computing Surveys* **28** (4es), 1996: Article 91.

[85] Catharine M. Wyss and Edward L. Robertson. Relational languages for metadata integration. *ACM Transactions on Database Systems* **30** (2), 2005: 624–660.

[86] Catharine M. Wyss and Edward L. Robertson. A formal characterization of PIVOT/UNPIVOT. In: *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management*, 2005: 602–608.

[87] *XML Path Language (XPath): Version 1.0)*. W3C Recommendation. Available at http://www.w3.org/TR/xpath (1999, accessed October 2013).

[88] *XML Path Language (XPath) 2.0: Second Edition*. W3C Recommendation. Available at http://www.w3.org/TR/xpath20/ (2010, accessed October 2013).

[89] *XML Schema Definition Language (XSD) 1.1 Part 1: Structures*. W3C Recommendation. Available at http://www.w3.org/TR/xmlschema11-1/ (2012, accessed October 2013).

[90] *XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes*. W3C Recommendation. Available at http://www.w3.org/TR/xmlschema11-2/ (2012, accessed October 2013).

[91] *XQuery 1.0: An XML Query Language (Second Edition)*. W3C Recommendation. Available at http://www.w3.org/TR/xquery/ (2010, accessed October 2013).

[92] *XQuery 1.0 and XPath 2.0 Functions and Operators: Second Edition*. W3C Recommendation. Available at http://www.w3.org/TR/xpath-functions/ (2010, accessed October 2013).

[93] *XSL Transformations (XSLT): Version 2.0*. W3C Recommendation. Available at http://www.w3.org/TR/xslt20/ (2007, accessed October 2013).

[94] Yu Xu and Yannis Papakonstantinou. Efficient keyword search for smallest LCAs in XML databases. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2005: 537–538.

Appendix A

Personal Contributions

The personal contributions of the present author (below TUN) to the individual publications included in this dissertation are as follows:

Paper I Janne Jämsen proposed the original idea of the methods developed in this paper, as well as formalized and implemented them. In the overall implementation, TUN was responsible for the data preparation and processing. The TRIX information retrieval system used for query evaluation was based on the previous work by Paavo Arvola. The proposed methods were developed, the results were analyzed, and the paper was written collectively by the three authors.

Paper II The tools presented in this paper were designed collectively by the two authors. TUN specified the formalisms, designed the sample environment, and largely wrote the paper. Timo Niemi proposed the original idea and supervised the research.

Paper III TUN proposed and developed the SPC query evaluation strategy and the related query primitive. He also designed and implemented the sample environment. The paper was written collectively by the three authors.

Paper IV Timo Niemi and Kalervo Järvelin developed the XML relation representation and its related constructor algebra. Timo Niemi also proposed the original idea of XML data harmonization. TUN contributed to the XML relation-based formalization of the SPC data extraction strategy, as well as designed the sample environment and specified the sample queries. The paper was written collectively by the three authors.

Paper V The proposed RXQL query language was designed and implemented by TUN and Katja Moilanen, who both also designed the sample environment. Timo Niemi gave impetus to developing the query language and contributed to the formalization of its query evaluation mechanism. TUN had the main responsibility for writing the paper.

Part II

Paper I

Entity Ranking Based on Category Expansion

Available at **DOI:10.1007/978-3-540-85902-4_24**.

Paper II

An Approach for Developing a Schemaless XML Dataspace Profiling System

Available at **DOI:10.1177/0165551512437532**.

Paper III

A Tool for Data Cube Construction from Structurally Heterogeneous XML Documents

Available at **DOI:10.1002/asi.20756**.

Paper IV

A Relational Data Harmonization Approach to XML

Available at **DOI:10.1177/0165551509104231**.

Corrigendum

- Formula (8) should be: $I(name) = \{ind | (name, t \in \{\text{'a', 'e'}\}, ind) \in XMLR\}$.

Paper V

A Query Language for Selecting, Harmonizing, and Aggregating Heterogeneous XML Data

Available at **DOI:10.1108/17440081111125662**.