UPPSALA
UNIVERSITET

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 1052

Scalable Preservation, Reconstruction, and Querying of Databases in terms of Semantic Web Representations

SILVIA STEFANOVA

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2013

ISSN 1651-6214
ISBN 978-91-554-8690-7
urn:nbn:se:uu:diva-199573

Dissertation presented at Uppsala University to be publicly examined in Pol/2446, Department of Information Technology Polacksbacken, Lägerhyddsvägen 2, Uppsala, Friday, June 14, 2013 at 13:00 for the degree of Doctor of Philosophy. The examination will be conducted in English.

Abstract
Stefanova, S. 2013. Scalable Preservation, Reconstruction, and Querying of Databases in terms of Semantic Web Representations. Acta Universitatis Upsaliensis. *Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology* 1052. 59 pp. Uppsala. ISBN 978-91-554-8690-7.

This Thesis addresses how Semantic Web representations, in particular RDF, can enable flexible and scalable preservation, recreation, and querying of databases.

An approach has been developed for selective scalable long-term archival of relational databases (RDBs) as RDF, implemented in the *SAQ (Semantic Archive and Query)* system. The archival of user-specified parts of a RDB is specified using an extension of SPARQL, *A-SPARQL*. SAQ automatically generates an RDF view of an RDB, the *RD-view*. The result of an archival query is RDF triples stored in: i)*a data archive file* containing the preserved RDB content, and ii)*a schema archive file* containing sufficient meta-data to reconstruct the archived database. To achieve scalable data preservation SAQ uses special query rewriting optimizations for the archival queries. It was experimentally shown that they improve query execution time compared with naïve processing. The performance of SAQ was compared with that of other systems supporting SPARQL queries to views of existing RDBs.

To reconstruct an RDF-archived RDB an approach was developed and implemented in the reloader module of SAQ. When an archived RDB is to be reconstructed, the reloader first reads the schema archive file and executes a schema reconstruction algorithm to automatically construct the RDB schema. The thus created RDB is populated by reading the data archive and converting the read data into relational attribute values. For scalable reconstruction of RDF archived data we have developed the *Triple Bulk Load (TBL)* approach where the relational data is reconstructed by using the bulk load facility of the RDBMS. Our experiments show that the TBL approach is substantially faster than the naïve *Insert Attribute Value (IAV) approach*, despite the added sorting and post-processing.

To view and query the semi-structured data Topic Maps as RDF the prototype system, TM-Viewer was implemented. A declarative RDF view of Topic Maps, the *TM-view* is automatically generated by the TM-viewer using a developed conceptual schema for the Topic Maps data model. To achieve efficient query processing of SPARQL queries to the TM-view query rewrite transformations were developed and evaluated. It was shown that they significantly improve the query processing time.

Keywords: RDF, RDFS, RDF view, SPARQL, SPARQL query processing, rewrite optimization, Topic Maps, querying of RDF views, archive relational databases, reconstruct archived databases

Silvia Stefanova, Uppsala University, Department of Information Technology, Division of Computing Science, Box 337, SE-751 05 Uppsala, Sweden.

ISSN 1651-6214
ISBN 978-91-554-8690-7
urn:nbn:se:uu:diva-199573 (http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-199573)

To my friends in Uppsala

List of Papers

This thesis is based on the following papers, which are referred to in the text by their Roman numerals.

I Stefanova, S., Risch, T. (2010) SPARQL queries to RDFS views of Topic Maps. *Int. J. Metadata, Semantics and Ontologies,* Vol. 5, No. 1, pp. pp.1 – 16.

II Stefanova, S., Risch, T. (2011) Optimizing Unbound-property Queries to RDF Views of Relational Databases, 7th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2011), at the 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, *In Proc. SSWS2011*, pp. 43-58.

III Stefanova, S., Risch, T. (2013) Scalable Long-term Preservation of Relational Data through SPARQL queries, *Under review in Semantic Web Journal*.

IV Stefanova, S., Risch, T. (2013) Scalable Reconstruction of RDF-archived Relational Databases, a short version accepted at 5th International Workshop on Semantic Web Information Management (SWIM 2013) in conjunction with the 2013 ACM International Conference on Management of Data (SIGMOD 2013) New York, New York, USA - June 23, 2013.

Other Related Publications

V Stefanova S., Risch, T. (2008) Viewing and Querying Topic Maps in terms of RDF, SeMMA2008, 1st International Workshop on Semantic Metadata Management and Applications, Proceedings, at the 5th European Semantic Web Conference ESWC2008, June 2008.

VI Stefanova S., Risch, T. (2009) Searchable Long-term Preservation of Scientific Data through Semantic Web representations, Doctoral Symposium, Calabria, Italy, September, 2009.

Contents

Abbreviations

Amos	Active Mediator Object System
A-SPARQL	Archive SPARQL
BPTP	Bound-Property Triple Pattern
CSV	Comma Separated Value
DBMS	Database Management System
DTD	Document Type Definition
IAV	Insert Attribute Value
IRI	Internationalized Resource Identifier
OWL	Web Ontology Language
OAIS	Open Archival Information System
RDBMS	Relational Database Management System
RDB	Relational Database System
RDF	Resource Description Framework
RDFS	RDF-Schema
SAQ	Semantic Web Archival and Query
SPARQL	Simple Protocol and RDF Query Language
SPP	Scientific Publication Packages
SQL	Structured Query Language
TBL	Triple Bulk Load
TP	Triple Pattern
UPTP	Unbound-Property Triple Pattern
URI	Uniform Resource Identifier
XML	Extensible Markup Language
XSD	XML Schema Definition

1 Introduction

The importance of digital preservation research has been growing for the past ten-fifteen years. Many papers and books [1][28][77] describing problems, tools, and techniques for digital preservation, have been written, and standards providing preservation models have been published [57][67]. However, most of the work focuses on preservation of file-based digital objects like documents, images, and web pages [1]. Much less work has focused on the preservation of structured data, i.e. databases and scientific data, where there is a recognized need to preserve scientific data [1][27][48][51][68]. Furthermore, preserving scientific data together with scientific publications would contribute to documenting the origin and lineage of scientific achievements [27]. For this a concept of 'Scientific Publication Packages (SPPs)' was introduced in [27]. The SPPs were described as composite digital objects linking experimental raw data, associated with metadata, 'derived information', and knowledge, including associated publications.

Scientific data, i.e. experimental and observational data, as well as data generated by instruments and sensors, reside in large datasets that are often stored in databases. During the research process the scientists need to select subsets of these databases to be analyzed and processed in order to create a scientific model. Once the model is validated the research results are documented and published. By preserving the selected subsets of both data and publications within one digital object, future reuse, verification, and heritage [77] of the published scientific results can be guaranteed.

Selective data preservation is needed also for example in cases of medical data preservation where, in order to protect the privacy of patients, sensitive data like zip code, dates of birth, salaries, etc. should be excluded from archiving [40]. Other examples are when selecting representative geospatial data for preservation [43] or preserving web resources based on some criteria [28].

For long-term preservation of data, it is desirable for the contents of a database to be saved in a neutral format, so that it can be reconstructed and used after a very long time using current technologies for data representation, which are continuously evolving. Furthermore, preserved representations must include sufficient meta-data to retrieve, explain, reproduce, and disseminate the experiments.

The Semantic Web is a universal medium for data, information, and knowledge exchange. It uses standard formats like RDF (Resource Description Framework) [16][17], RDFS (RDF-Schema) [11] and OWL (Web Ontology Language) [13] to enable integration and combination of data from different sources. RDF is a simple metadata model where all data is represented as triples, while RDFS is a set of classes with certain properties to represent RDF data. SPARQL [66] is the standard query language for querying RDF triples data.

XML (Extensible Markup Language) [76] is another data representation standard where data is encoded in a text format that is both human-readable and machine-readable. It is simple and general, and therefore it is also used for data exchange.

Both XML and RDF along with RDFS are neutral data formats that do not rely on current DBMS technology and provide hardware and software independence. This makes both of them suitable for long-term preservation of databases. However, RDF has the following advantages compared to XML:

- RDF provides URIs, which are universal global unique identifiers that allow identifiers from one database or table to be linked with identifiers from other data.
- Data can be represented as XML in many different ways described by a DTD (Document Type Definition) grammar or XML schema data types [32] while RDFS provides standard meta-data representation for describing semantics of data, including relational databases [21].
- RDF facilitates interoperation because its data model can be extended to address sophisticated ontologies [60].
- Representing relational data as RDF allows migration from RDBs to RDF repositories which are gaining increasing popularity compared to XML native repositories [2][4]49[64][79].

RDF based neutral formats seem promising as a database technology-independent format for long-term preservation of data, which provides standard meta-data representation for all kinds of data, including relational databases.

This Thesis addresses how Semantic Web representations, in particular RDF, can enable flexible and scalable preservation, reconstruction, and querying of databases. The following overall research questions are studied:

1. How can different kinds of databases be preserved in terms of RDF views of the databases? In particular:
    a. How should structured data, in particular relational databases be represented as RDF views?
    b. How should semi-structured data, in particular Topic Maps be represented as RDF views?

2. How can queries to RDF views enable flexible preservation of database contents?
3. How should such *archival queries* to RDF views be optimized?
    a. What kinds of novel query processing strategies are useful for scalable relational database preservation by archival queries?
    b. What kinds of query processing strategies are useful for Topic Map queries?
4. How can relational databases be scalably reconstructed from archived RDF representations?

To answer the above research question two prototype systems have been developed and evaluated:

I *SAQ (Semantic Archive and Query)* – a system for long-term preservation, querying and reconstruction of structured data, i.e. relational databases in terms of RDF.

II *Topic Map Viewer (TM-Viewer)* – a system for view and querying semi-structured data, i.e. Topic Maps data in terms of RDFS.

The following sections present an overview of the prototype systems and how they answer the research questions.

2 Motivating Example

This chapter presents a motivating example for selective preservation and reconstruction of a relational database as RDF.

Figure 2.1 shows an example of a small relational database *Ebird* for storing bird watching data from Avian Knowledge Network [3]. It contains four tables *occurrence*, *taxon*, *event*, and *location*, storing data about an occurrence of a bird at some event and location. The primary keys of the tables are the following: *oid* is the primary key in *occurrence*, *tid* is the primary key in *taxonomy*, *eid* is the primary key in *event*, and *lid* is the primary in *location*. The attributes *taxon* and *event* in *occurrence* are foreign keys referencing tables *taxonomy* and *event* respectively. Likewise, the attribute *location* in table *event* is a foreign key referencing table *location*.

Table *occurrence*

oid	recordedBy	event	taxon
1	Obs6453	1	63
2	Obs6453	1	65
3	Obs1000	2	63

Table location

lid	country	stateProvince
2	USA	Florida
3	USA	New York

Table *taxonomy*

tid	name	genus
63	Parula_americana	Parula
65	Parula_superciliosa	Parula

Table *event*

eid	eventDate	location
1	2003-06-07	2
2	2003-10-08	3

Figure 2.1. Ebird database

A SAQ user preserves *taxonomy* data about the bird *Parula_americana* and data about all *locations* where this bird has been seen executing the following A-SPARQL query to the RDF view named <*Ebird*> representing the RDB *Ebird*:

```
ARCHIVE AS 'data.nt', 'schema.nt'
FROM <Ebird>
TRIPLES
{?subject1  ?property1             ?value1        }
WHERE
{?subject1  rdf:type               ebird:taxonomy .
 ?subject1  ebird:taxonomy_name    "Parula_americana"}
   UNION
TRIPLES
{?subject2  ?property2             ?value2      }
WHERE
{?subject2  rdf:type               ebird:location .
 ?subject3  rdf:type               ebird:event .
 ?subject3  ebird:event_location   ?subject2 .
 ?subject4  rdf:type               ebird:occurrence .
 ?subject4  ebird:occurrence_event ?subject3 .
 ?subject4  ebird:occurrence_taxon ?subject5 .
 ?subject5  rdf:type               ebird:taxonomy .
 ?subject5  ebird:taxonomy_name    "Parula_americana"}
```

The execution of the above archival query produces two N-Triples [46] files:

1) The data archive file, *'data.nt'*, containing the archived bird taxonomy and locations data.
2) The schema archive file, *'schema.nt'*, containing the schema information required for reconstructing the parts of the RDB schema describing the archived data.

```
<ebird:taxonomy>          <rdf:type>          <rdfs:Class> .
<ebird:taxonomy>          <saq:TName>         "taxonomy" .
<ebird:location>          <rdf:type>          <rdfs:Class> .
<ebird:location>          <saq:TName>         "location"
<ebird:taxonomy_tid>      <rdf:type>          <rdf:Property>  .
<ebird:taxonomy_tid>      <saq:AName>         "tid" .
<ebird:taxonomy_tid>      <rdfs:domain>       <ebird:taxonomy> .
<ebird:taxonomy_tid>      <rdfs:range>        <xsd:integer> .
<ebird:taxonomy>          <saq:Primary_key>   :_taxonomy_PrK .
:_taxonomy_PrK .          <rdf:type>          <rdf:Seq> .
:_taxonomy_PrK .          <rdf:_1>            <ebird:taxonomy_tid> .
<ebird:taxonomy_genus>    <rdf:type>          <rdf:Property> .
<ebird:taxonomy_genus>    <saq:AName>         "genus" .
<ebird:taxonomy_genus>    <rdfs:domain>       <ebird:taxonomy>  .
<ebird:taxonomy_genus>    <rdfs:range>        <xsd:String> .
```

Figure 2.2. Data archive *'schema.nt'*

<ebird:taxonomy/63>	<rdf:type>	<ebird:taxonomy> .
<ebird:taxonomy/63>	<ebird:taxonomy_name>	“Parula_americana”
<ebird:taxonomy/63>	<ebird:taxonomy_tid>	“63”^^<xsd:int> .
<ebird:taxonomy/63>	<ebird:taxonomy_genus>	“Parula” .
<ebird:location/2>	<rdf:type>	<ebird:location> .
<ebird:location/2>	<ebird:location_country>	“USA” .
<ebird:location/2>	<ebird:location_stateProvince>	“Florida” .
<ebird:location/2>	<ebird:location_lid>	“2”^^<xsd:int> .
<ebird:location/3>	<rdf:type>	<ebird:location> .
<ebird:location/3>	<ebird:location_lid>	“3”^^<xsd:int> .
<ebird:location/3>	<ebird:location_country>	“USA” .
<ebird:location/3>	<ebird:location_stateProvince>	“New York” .

Figure 2.3. Data archive *‘data.nt’*

Part of the ‘schema.nt’ is shown in Figure 2.2, while ‘data.nt’ is shown in Figure 2.3. The namespace *ebird* in both ‘schema.nt’ and ‘data.nt’ is used as a prefix for an ontology describing the archived RDB *Ebird*. When the archived database is re-created as a relational database the schema archive ‘schema.nt’ is first read and the RDB schema of the reconstructed RDB, named *r_Ebird* in Figure 2.4 is created. It contains only the parts of the tables of the RDB in Figure 2.1 archived by the archival query. Then the RDB is populated by reading the data archive ‘data.nt’.

Table taxonomy

tid	name	genus
63	Parula_americana	Parula

Table *location*

lid	country	stateProvince
2	USA	Florida
3	USA	New York

Figure 2.4. The reconstructed RDB *r_Ebird*

3 Technical Background

This chapter presents the technical background of the major technologies for preservation, reconstruction and querying of databases in terms of Semantic Web representations.

3.1 Relational Database Management Systems

A *database* is a large collection of data, while a *database management system (DBMS)* is a collection of programs that enables users to create and maintain a database [56]. The DBMS is a general-purpose software system for:

- Defining a database by specifying the data types, structures and constraints of the data to be stored in the database;
- Constructing the database, which is the process to store the data on a storage medium controlled by the DBMS;
- Manipulating a database involves querying the database to retrieve data and also update the database;
- Sharing a database allows multiple users and programs to access the database simultaneously.

Data model and Query Language in RDBMS

A *relational database management system (RDBMS)* is a DBMS that is based on the *relational data model* introduced by Codd [14]. In the relational data model all data is represented in terms of tuples or rows, grouped into relations described as 2-dimensional tables with one or several columns.

An example of a small relational database is shown in Figure 2.1. The database called *Ebird* stores data about appearance of birds at some event and location. It contains four tables *occurrence*, *taxon*, *event*, and *location*.

In the relational model a user defines queries in a high-level query language where the *Structured Query Language (SQL)* is the most widely used. SQL queries allow the user to describe desired data to retrieve, leaving the RDBMS to decide how the data should be accessed. The RDBMS, not the user is responsible for planning, optimizing, and performing the physical operations necessary to produce the result.

For example, the SQL query in Figure 3.1 retrieves the countries and provinces where the bird species 'Parula americana' has been seen during the period from June until October 2003

```
SELECT DISTINCT l.country, l.stateProvince
FROM location l, taxonomy t, event e, occurrence o
WHERE t.name='Parula_americana' AND o.taxon=t.tid
AND o.event=e.eid AND e.location=l.lid AND
e.eventDate>'2003-06-01' AND e.eventDate<'2003-10-01'
```

Figure 3.1. Example SQL query to Ebird

The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control. Although SQL is partially a declarative language, it also includes procedural elements.

Query processing in RDBMS

Figure 3.2 shows the typical query processing steps in a RDBMS [26]. The SQL query is first checked for syntactic and semantic correctness by the *query parser and validator*, respectively. For example, it is checked that the query refers to only existing table and column names.

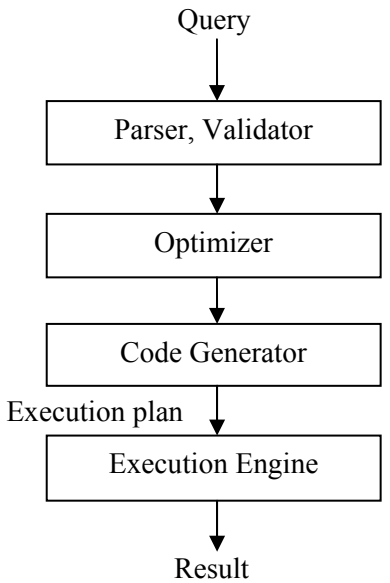

Figure 3.2. Query processing steps in RDBMS

The result of parsing and validating the query is an algebraic representation of the query in the form of a logical query plan in a relational algebra that is, a sequence of operators to be executed. No algorithms have yet been assigned to implement the operators in the query plan. The task of the *optimizer* is to produce an efficient *execution plan* (strategy) since a query often can be executed in numerous ways. Each query plan has a predefined cost according to some cost model and the cheapest plan is picked by the optimizer. The 'cost' can, for example, be approximated by the number of disk accesses performed by the RDBMS when executing a specific query plan. For the query optimizer to correctly estimate the cost of alternative query plans it is therefore important that there exist valid statistics about data characteristics such as number of rows in a table and the number of different values in table column.

Finally, the resulting physical query plan is interpreted by the *execution engine* producing the result.

In this Thesis it is presented the SAQ prototype system that automatically generates RDF views over relational databases. The RDF view can be queried by SPARQL. SPARQL queries are processed in SAQ so that the system generates and sends SQL queries to the back-end RDBMS. The SQL queries are cost-based optimized and executed in the RDBMS, and the result is post-processed in SAQ.

3.2 Long-term Preservation of Databases

Most of the work done on digital preservation of data has focused on preservation of digital object in conventional file-based formats like documents, images, and web pages [12]. Very little work has been devoted to preservation of databases despite the fact that databases are central to scientific research [7].

The databases are different from conventional file-based digital objects since they have internal schema and constraints. This is the reason that the "existing preservation techniques for fixed digital objects are not suited for databases" [7]. Furthermore, "long-term preservation of relational databases is much more than just making backups of export or dump files from database management applications, though IT professionals usually use archiving and backup as synonyms" [23].

This section introduces concepts, requirements and existing strategies for digital long-term preservation of relational databases.

Requirements for Long-term Preservation of Databases

For long-term preservation of databases it is necessary to preserve the "intelligibility and comprehensibility" of the database as well as its data [23]. It is required to collect and store enough metadata together with the archived data in order to keep the data understandable and meaningful.

There are some main requirements during the process of long-term preservation of databases [23]:

- Integrity – Data integrity ensures that the data stored in the database is complete, correct and consistent.
- Intelligibility – Both the database schema and data have to be understandable and related to what they represent. The database intelligibility is the ability to perceive and interpret the data formats and the relationships between the tables and what they represent.
- Authenticity – It relates the preserved information with its source. It can be guaranteed by keeping record of the actors, tools and operations involved in the preservation process.
- Originality – The data structure has to be "as close to the original as possible".
- Accessibility – It refers to technical readability and usability

Open Archival Information System Model Reference

Open Archival Information System (OAIS) [57] reference model is an ISO standard with directives for long-term storage of digital information.

The data flow diagram for the OAIS model is shown in Figure 3.3. The

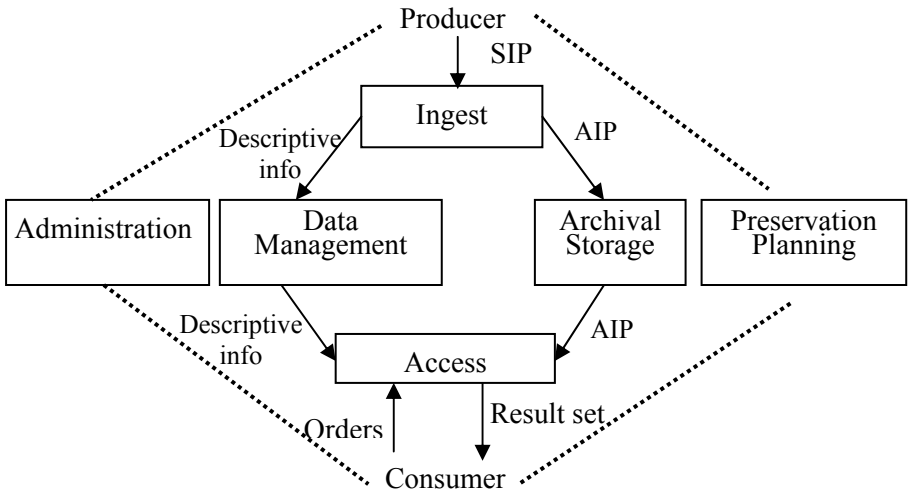

Figure 3.3. OAIS data flow diagram

OAIS model is composed by four functional units*: Ingest, Archival Storage, Data Management*, and *Access*. The *Ingest* unit accepts *Submission Information Packages (SIPs)* and generates *Archival Information Packages (AIPs)* for storage and management. The *Archival Storage* unit receives AIPs from Ingest and adds them to permanent storage. *The Data Management unit* provides functions for populating, maintaining, and accessing variety of metadata stored in the repository. *The Access* unit provides an interface between the archive and the consumer.

An OAIS' AIP contains *Content Information*, i.e. the archived data and the representational metadata, together with a *Preservation Description Information (PDI)*. The PDI contains reference information, context information, provenance information, etc.

The Preservation Planning unit defines and manages strategies to AIPs to move through time without losses and changes the content or functionality. It starts long before the digital objects to be moved to a long-term preservation system and continues throughout their existence.

The Administration unit contains the services and functions for controlling the other OAIS functional units.

The prototype system SAQ provides functionality for the Ingest component, in particularly on generating the content information in the AIPs when preserving relational database contents as RDF

Preservation Strategies

Some well-known strategies for digital preservation are the following:

- Emulation – preservation strategy where the digital objects are kept in their original digital formats since it has "the ability to execute the software needed to process data stored in its 'original' encodings" [34];
- Encapsulation – preservation strategy that is based on re-creating the original technological environment of the application in the future [8][62];
- Migration – preservation strategy that transfers a digital object from one technological environment to another [8]. It is used to transfer information between systems or applications.

The migration preservation strategy doesn't preserve digital objects in their original formats; instead they are transformed to other more 'up-to-date' formats that users will be able to interpret using common software [54]. This can be regarded as a format conversion of the archived digital objects.

The migration preservation strategy is the one used in the SAQ system where relational data and schema is migrated to RDF.

Existing Solutions for Long-term Preservation of Relational Databases

This section summarizes exiting projects for digital preservation of relational databases.

Digital Preservation Testbed

The Digital Preservation Testbed [9] is an initiative of the Dutch National Archives and the Dutch Ministry of the Interior and Kingdom Relations. It studies different strategies for long-term preserving of governmental and other type of digital information.

Digital Preservation Testbed has recommended preservation of relational databases as XML. A conversion tool was developed for converting relational databases in Microsoft Access and Oracle into XML.

SIARD

SIARD (Software Independent Archival of Relational Databases) [72] was developed as part of the SFA's (Swiss Federal Archives) ARELDA project for digital archiving.

SIARD is an open standard supported by the SIARD Suite, which can be used to convert relational databases into the SIARD format [73]. The SIARD format is based on open Standards, e.g. ISO norms Unicode, XML, SQL1999 and the industry standard ZIP.

A relational database archived in the SIARD format [73] consists of two components: *metadata*, describing the structure of the archived database and *primary data*, representing the table contents. The metadata provide also information about where to find primary data in the archive.

Database metadata and primary data are stored together in an uncompressed ZIP archive with the filename extension ".siard". The primary data is stored in the folder *content* and the metadata in the folder *header* (Figure 3.4).

The metadata is stored in a file *metadata.xml*. The metadata folder contains also an XSD (XML Schema Definition) schema *metadata.xsd* for XML file validation.

The data from each table is stored in an XML file, *table.xml* representing its rows and columns and a corresponding XSD file, *table.xsd*. If there are Large Objects (LOBs) they are stored in binary files or text in a separate folder for each such object. The primary data for each database schema is stored in a separated folder sequentially numbered, e.g. *schema1*, *schema2*, etc.

content
schema1
table1
table.xsd
table.xml
lob1
record1.txt / record1.bin
lob2
record1.txt / record1.bin
...
table2
table.xsd
table.xml
...
schema2
...
header
metadata.xsd
metadata.xml

Figure 3.4. Structure of the SIARD archive file

LOB: Large Object (BLOB, CLOB)
XSD: XML-Schema-Definition

RODA

Work on relational database preservation as XML was done also in The Portuguese Repository of Authentic Digital Objects project (RODA) [54] run by the National Archives of Portugal in partnership with the University of Minho, Portugal. A prototype system converting relational databases to XML that conforms to the OAIS reference model was created.

RODA uses Database Markup Language (DBML) and XML to migrate a relational database into a single XML file containing both the database structure and data.

3.3 Semantic Web, RDF, SPARQL

The term "Semantic Web" refers to the vision of W3C about the Web of linked data [82]. The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is based on the Resource Description Framework (RDF).

Resource Description Framework (RDF)

RDF [16] is a W3C standard for representation of web resources in the World Wide Web. It can be seen as a language for representing meta-data about resources.

RDF is intended to be processed by applications, rather than being only displayed to people [15]. It provides a common framework for information representation so that information can be exchanged between applications without loss of meaning. The ability to exchange information between different applications means that the information may be made available to applications other than those for which it was originally created.

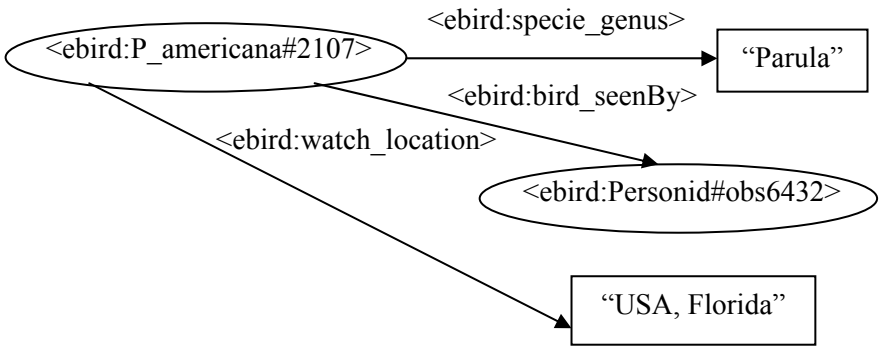

Figure 3.5. RDF example graph

RDF is based on identifying things using Web identifiers [15], called *Uniform Resource Identifiers* (*URI*s) [78]. Furthermore, RDF describes resources in terms of simple properties and property values, which enables simple representation as statements about resources. The statements are expressed by a graph of nodes and arcs representing the resources, and their properties and values. For example, the group of statements with data from Avian Knowledge Network [3] "A bird identified by the URI *<ebird:P_americana#2107>* from the specie 'Parula americana' and genus 'Parula' has been seen in USA, Florida by the bird watcher identified by the URI *<ebird:Personid#obs6432>* could be represented as the RDF graph in Figure 3.5. The namespace *ebird* is a prefix for ontology describing birds watching data.

The statements in RDF are always triples having the shape *(subject, predicate, object)* where a particular terminology is used for describing their various parts. The part that identifies the thing the statement is about is called the *subject*, e.g. *<ebird:P_americana#2107>*. The part that identifies the property or characteristic of the subject that the statement specifies is called the *predicate* or the *property*, e.g. *<ebird:watch_location>*, and the part that

identifies the value of that property is called the *object* or the *value,* e.g. *"Usa, Florida"* or <*ebird:Personid#obs6432*>.

For example, the RDF triples representing the RDF graph in Figure 3.6 are shown in Figure 3.6.

<ebird:P_americana#2107>	<ebird:specie_genus>	"Parula"
<ebird:P_americana#2107>	<ebird:bird_seenBy>	<ebird:Personid#obs6432>
<ebird:P_americana#2107>	<ebird:watch_location>	"USA, Florida"

Figure 3.6. RDF triples

RDF-Schema (RDFS)

RDFS [11] is a high level language to represent RDF data. It contains a set of classes with certain properties. In other words, RDFS provides a type system for RDF. For example, RDFS allows resources to be defined as instances of one or more classes. In addition, it allows classes to be organized in a hierarchical fashion; for example a class with URI <*ebird:P_americana*> might be defined as a subclass of <*ebird:Parula*> which is a subclass of <*ebird:Observed_birds*>, meaning that any resource which is in class <*ebird:P_americana*> is also implicitly in class <*ebird:Observed_birds*> as well.

Everything described by RDF is called an *RDF resource*; it is identified by an URI, and it is an instance of the class <*rdfs:Resource*>, which is the root class of everything. All other classes are subclasses of this class. <*rdfs:Resource*> is an instance of the main RDF class of resources <*rdfs:Class*>. An RDF property <*rdf:Property*> is a relation between subject resources and object resources. The RDF property <*rdf:type*> is used to define the data type of an RDF resource by associating it to one or several classes. The domain property <*rdfs:domain*> identifies the class(es) for which a property is defined. The range property <*rdfs:range*> defines that the values of a property are instances of one or more classes.

For example, Figure 3.7 illustrates RDF triples from an RDFS ontology

1	<ebird:Observed_birds>	<rdf:type>	<rdfs:Class>
2	<ebird:P_americana>	<rdfs:subClassOf>	<ebird:Observed_birds>
3	<ebird:specie_genus>	<rdf:type>	<rdf:Property>
4	<ebird:specie_genus>	<rdfs:domain>	<ebird:Observed_birds>
5	<ebird:specie_genus>	<rdfs:range>	<xsd:String>

Figure 3.7. RDFS for the example RDF data

defined to represent the meta-data for the example in Figure 3.5.

The RDF triple 1 defines an RDFS class <*ebird:Observed_birds*> to represent all types observed birds, while the triple 2 defines a subclass of that class <*ebird:P_americana*> to represent only the birds 'Parula americana'. Triple 3 defines an RDF property <*ebird:specie_genus*> for the main class (represented by triple 4) that has values of type string (represented by triple 5).

In this Thesis RDF and RDFS are used for representing RDF views over relational databases and Topic Maps.

SPARQL and SPARQL Query Classes

SPARQL (Simple Protocol and RDF Query Language) [66] is the language proposed by W3C for querying RDF triples data. SPARQL is a syntactically SQL-like language for querying RDF graphs via pattern matching [25].

A SPARQL query is composed of the following five parts [25]:

- Prefix declarations – zero or more; used to introduce shortcuts for long identifiers;
- Query result clause - specifies the form of the result; the result clause can be a *SELECT, ASK, CONSTRUCT* or *DESCRIBE* clause explained below;
- FROM or FROM NAMED clauses – zero or more; defines the data set against which the query is executed;
- WHERE clause – a set of *triple patterns* (*TP*s) to specify the search conditions to match against the RDF triples.
- Query modifiers – zero or more; operate over the triples selected by the WHERE clause. As in SQL, the clause ORDER BY orders the result set, the LIMIT and OFFSET allow getting the result in chunks

A SELECT SPARQL query provides an answer that is a tuple similar to the SQL result in a relational database. An ASK query checks whether there is at least one query result; if it does the query answer is YES; otherwise the answer is NO. The CONSTRUCT query provides the answer as an RDF graph, i.e. the query result is a set of triples. The DESCRIBE query returns an RDF graph that describes RDF resources.

The *archival queries* in SAQ prototype system select parts of an RDB to archive. Archival queries are translated to CONSTRUCT SPARQL queries, which are processed by a SPARQL query processor. The motivation to translate to CONSTRUCT queries is that the result of an archival query should be a user selected subset of the RDF graph representing the RDB.

For example, the query in Figure 3.8 is a SPARQL CONSTRUCT query against the RDF triple data in Fig. 3.7 represented by <*ebird*> in the

WHERE clause. The query returns as a RDF graph (triples) all the birds with their properties and values that are from the genus “Parula”.

```
PREFIX ebird: <http://birdwatching.org/ebird/>
CONSTRUCT {?subject ?property ?value }
FROM <ebird>
WHERE {?subject  ebird:specie_genus  “Parula” .
       ?subject  ?property           ?value . }
```

Figure 3.8. CONSTRUCT SPARQL Query

Query Classes

A *bound-property triple pattern (BPTP)* is a SPARQL triple pattern (?s P ?o) where the property P is a constant URI. An *unbound-property triple pattern (UPTP)* is a triple pattern (?s ?u ?o) where *u* is a variable, e.g. (?subject ?property ?value). A *bound-property query* is a SPARQL query having only BPTPes. An *unbound-property query* is a SPARQL query having one or several UPTPs.

In a SPARQL query with a TP *(?s ?p ?o)* we call the variable *s* a *subject variable*, *p* a *predicate variable*, and *o* an *object variable*. In a query, if the same variable is an object variable in one TP, e.g. *s1* in *(?s ?p ?s1)*, and a subject variable in another TP, e.g. *s1* in *(?s1 ?p1 ?o1)*, we call the variable *s1* a *subject-object join variable*. A subject-object join variable cannot be a literal, since subjects are always URIs.

3.4 Mapping Relational Databases as RDF

There are two standardized approaches for mapping relational databases to RDF, *RDB2RDF* [58]:

- The *Direct mapping approach* maps an RDB into an RDF graph where “the graph should be explored/transformed further by the application”.
- *The R2RML approach* provides a mapping language that allows the creation of a “final RDF graph for an application”.

The Direct Mapping Approach

In the direct mapping an RDB is mapped into an RDF graph called *direct graph* where the structure of the result RDF graph directly reflects the RDB schema elements [41]. There is a direct correspondence between the target RDF vocabulary and the names of the RDB schema elements.

Each table row produces a set of RDF triples with a common subject. The subject is an IRI [24] formed from the concatenation of the base IRI, table name, primary key column name(s) and primary key value. The predicate for

each column is an IRI formed from the concatenation of the base IRI, the table name and the column name. The values are RDF literals formed from the lexical form of the column value.

Each foreign key produces a triple with a predicate composed from the foreign key column names, the referenced table, and the referenced column names. The object of these triples is the row identifier for the referenced table. The reference row identifiers must coincide with the subject used for the triples generated from the referenced row. The direct mapping does not generate triples for NULL values.

For example, applying the direct mapping on table *event* from the RDB *Ebird* (Figure 2.1) produces the RDF triples shown in Figure 3.9.

```
1  <Event/eid=1>  <rdf:type>            <Event>
2  <Event/eid=1>  <Event#eid>           “1”^^xsd:int
3  <Event/eid=1>  <Event#eventDate>     “2003-06-07” ^^xsd:dateTime
4  <Event/eid=1>  <Event#location>      <Location/lid=2>
5  <Event/eid=2>  <rdf:type>            <Event>
6  <Event/eid=2>  <Event#eid>           “2”^^xsd:int
7  <Event/eid=2>  <Event#eventDate>     “2003-10-08”^^xsd:dateTime
8  <Event/eid=2>  <Event#location>      <Location/lid=3>
```

Figure 3.9. RDF triples for table event by the direct mapping

The triples 1 and 4 define the types of the row identifiers *<Event/eid=1>* and *<Event/eid=2>* representing the *event* table rows with primary key values 1 and 2. The triples 2,3 and 6,7 respectively represent as literals the values of the non-foreign key columns *eid* and *eventDate* for the table rows. Finally, the triples 4 and 8 represent the values for the foreign key column *location* in table *event* as the IRIs *<Location/lid#2>* and *<Location/lid#3>* which are row identifiers in the owning table *location*.

The R2RML Approach

R2RML is a language for expressing customized mappings from RDB to RDF [63]. Each mapping is tailored for a specific RDB schema and target vocabulary according to the “mapping author's choice”. An R2RML mapping is described as RDF graphs in Turtle syntax [10]. R2RML enables different types of mapping implementations.

An R2RML mapping refers to *logical tables* to retrieve relational data from the input database. A logical table can be a base RDB table, a view, or a valid SQL query. Each logical table is mapped to RDF using a *triples map*. The triples map is a rule that maps each row in the logical table to a number of RDF triples. The rule has two main parts:

- *The subject map* generates the subject for all RDF triples to be generated from a logical table row. The subjects often are IRIs generated from the primary key column(s) of the table.
- *The multiple predicate-object map* consists of *predicate map*s and *object maps*.

RDF triples are produced by applying the subject map with a predicate map and an object map to each logical table row.

For example, the triples map rule shown as Turtle in Figure 3.10 will produce the desired triples from the column *eventDate* in table *event* (Figure 2.1).

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns#>.

<#TriplesMap1>
   rr:logicalTable [ rr:tableName "event" ];
   rr:subjectMap [
      rr:template "http://example.com/ebird_event/{eid}";
      rr:class ex:Event;
   ];
   rr:predicateObjectMap [
      rr:predicate ex:date;
      rr:objectMap [ rr:column "eventDate" ];
   ].
```

Figure 3.10. R2RML mapping for column *eventDate* in table *event*

The RDF triples generated from the mapping rule in Figure 3.10 are shown in Figure 3.11.

```
< http://example.com/ebird_event /1>   <rdf:type>   <ex:Event>
< http://example.com/ebird_event /1>   <ex:date>    “2003-06-07” ^^xsd:dateTime
< http://example.com/ebird_event /2>   <rdf:type>   <ex:Event>
< http://example.com/ebird_event /2>   <ex:date>    “2003-10-08”^^xsd:dateTime
```

Figure 3.11. RDF triples for the R2RML mapping in Figure 3.10

The RDB to RDF automatically generated mapping in the prototype system SAQ conforms to the direct mapping, and more particularly to the augmented direct mapping proposed in [30], which is proven to guarantee information preservation. Furthermore, a SAQ user is allowed to assign own IRIs for representing tables and columns in the mapped RDB instead of using the automatically generated ones.

3.5 Topic Maps

Topic Maps [74] is a standard for a semi-structured data representation started in the 1990s from the idea of managing indices to documents so that multiple indexes from different sources could be merged. It was published as a data model standard in 1999 as ISO/IEC 13250-Topic Navigation Maps.

The first edition of ISO 13250 included an interchange syntax based on SGML and the hypermedia linking language known as HyTime[20]. The second edition, published in 2001 [69], added an interchange syntax based on XML. This is the syntax with the widest support in Topic Maps processing products.

The Topic Maps data model can be seen as ontology for describing how to navigate into document collections since it provides built-in concepts useful to describe indices of documents or websites. The Topic Maps model can be summarized in the following way. *A topic map* consists of a collection of *topics*, each of which represents some concept. Topics are related to each other by *associations*, which are many-to-many relationships to other topics. A topic may also be related to any number of resources by its *occurrences*.

Topic Maps can be merged. Merging can take place at the discretion of the user or application (at runtime). It can be also indicated by the Topic Map's author at the time of its creation [69].

The main concepts of the Topic Maps data model are described below.

Topics

A topic is a representation of a concept. The Topic Maps standard does not restrict the set of concepts that can be represented. Typically topics are used to represent electronic resources, i.e. Web pages, Web services, and non-electronic resources. Topics can be also used to represent abstract concepts like "Pensions" or "Insurance" [33].

A topic may have zero or more names, each of which is considered to be valid within a certain scope. Each name may exist in multiple forms. A name always has exactly one base form, known as the *base name*, and it may, in addition, have one or more variants, called *variant name* for use in specific processing contexts [69].

Associations

Associations are the general form for the representation of relationships between topics in a topic map. An association can be seen as an n-ary aggregate of topics [33], which is a grouping of topics with no implied direction or order.

Occurrences

Occurrences are used to represent or refer to information about a concept represented by a topic. Occurrences can be used either to store string data

within the topic map, or to reference any kind of Web-addressable resource external to the topic map. There is no restriction on what type of resource is addressed by an occurrence.

Scope

Scope is the term used in the Topic Maps standard to refer to a constraint or a context in which something is said about a topic. The way in which such statements about topics are made is by:

- Adding a name to the topic specifying an occurrence for a topic
- Creating an association between topics.

Context scope is often used e.g. to facilitate multi-lingual interfaces, so the concept "Dog" may have the label "dog" in the context of the English language, "perro" in Spanish, and "hund" in Swedish.

In a topic map, scope is defined by a collection of topics that can be assigned to a name, an occurrence, or an association. The default scope is known as the unconstrained scope and means that the name, occurrence, or association is always valid.

A topic representing the city of Helsingborg in Sweden may look like this:

```
<topic id="cty-cid-cia-Sweden-11">
    <instanceOf>
      <topicRef xlink:href="#city"></topicRef>
    </instanceOf>
    <baseName>
      <baseNameString>Helsingborg</baseNameString>
    </baseName>
    <occurrence>
      <instanceOf>
        <topicRef xlink:href="#population"></topicRef>
      </instanceOf>
      <resourceData>114339</resourceData>
    </occurrence>
  </topic>
```

The topic is identified by a topic id, *cty-cid-cia-Sweden-11*and the name of the city “Helsingborg” appears as a value for the topic’s *baseName*. An occurrence to the topic represents information about the population of the city.

An association representing the relationship between the city of Helsingborg, identified by a topic’s id *cty-cid-cia-Sweden-11* and the province Malmöhus, identified by a topic’s id *prov-cid-cia-Sweden-14* might look like this:

```
<association>
    <instanceOf>
      <topicRef xlink:href="#contained-in"></topicRef>
    </instanceOf>
    <member>
      <roleSpec>
        <topicRef xlink:href="#container"></topicRef>
      </roleSpec>
      <topicRef xlink:href="#prov-cid-cia-Sweden-14”>
                                  </topicRef>
    </member>
    <member>
      <roleSpec>
        <topicRef xlink:href="#containee"></topicRef>
      </roleSpec>
      <topicRef xlink:href="#cty-cid-cia-Sweden-11">
                                            </topicRef>
    </member>
  </association>
```

Topic Maps provide a meta-model that can be used to build a flexible application model. Since its ontology is expressed as topics and associations between topics, extension of the ontology becomes an issue of adding data, not an issue of redesigning the underlying schema used by an application [33].

3.6 Mapping Topic Maps as RDF

The existing approaches for mapping Topic Maps as RDF can be divided into two main groups [71]:

- *The Semantic mapping approach* – based on finding semantic equivalences between the Topic Maps data model and the RDF data model.
- *The Object mapping approach* - based on representing the Topic Maps data model in terms of the RDF model.

The Semantic Mapping

The semantic mapping starts from higher level concepts that carry the semantics of The Topic Maps model and looks for equivalences with the RDF model [70][71]. For example, a binary association in Topic Maps can be seen to represent a relationship between two entities and therefore it can be represented using a single RDF statement. If there is no direct semantic

equivalent of a Topic Map concept in RDF, new RDF classes and properties are defined to represent the missing concept.

The semantic mapping can give good results from a "naturalness" and flexibility point of view [35][19] but it is not always possible due to lack of semantic equivalences [52]. It is not of general usefulness either since it requires an application-dependent approach.

The Object Mapping

The object mapping approach provides a general mapping from Topic Maps to RDF. Such mappings have been proposed in [36][44][47].

The proposal [36] by Garshol is based on an earlier version of the ISO/IEC model of Topic Maps, i.e. the *TMDM* model [37]. The so called items in TMDM become RDFS classes and the properties of the items become RDF properties. The object mapping [36] is incomplete because there are no definitions of the RDFS ranges and domains of the properties and no description of how Topic Maps data is mapped to RDF.

The authors in [44] use the *Processing Model for Topic Maps*, *PMTM4* [42], which is very simple model and is not considered as a complete model for Topic Maps [71]. This disadvantage has been overcome in [47] where a Topic Maps model is defined partly in terms of PMTM4 and completed with extra XTM terms.

The proposal [47] is fairly complete but very complicated and the translation from the Topic Maps data model to RDF is non-reversible [71]. For example, it requires seven statements to represent the information content that would be modeled using one statement in RDF [71].

An RDB to archive as RDF in terms of SPARQL queries can be indexed by a large Topic Map serving as ontology for describing how to navigate into the database. Exposing Topic Maps as RDF and making them searchable by SPARQL would allow search and archive as RDF both the archived RDB and its navigation ontology in Topic Map. This is our motivation to investigate how to represent and query Topic Maps as RDF. For that the prototype system TM-viewer has been developed. In the TM-Viewer an object mapping of Topic Maps to RDF is applied. The mapping is based on a proposed conceptual schema of Topic Maps representing its concepts.

3.7 Functional and Object-relational DBMS (Amos II)

Both prototype systems SAQ and TM-Viewer are implemented using the functional and object-relational database system Amos II [75]. Amos II is an extensible main memory DBMS and it can store data in its main-memory object store. It contains functionality for processing end executing queries

over data stored locally as well as data stored in external data sources, such as relational databases [18]. In order to access data from external data sources Amos II contains several *wrappers*. A wrapper is a program module in Amos II having specialized facilities for query processing and translation of data from a particular class of external data sources. For example, Amos II has wrappers to access relational databases [18], Topic Maps files [81], XML files [22], CAD systems [45], etc.

Query language

The query language in Amos II is the object-relational and functional query language AmosQL [75]. AmosQL is a functional language based on the functional query languages OSQL [53] and DAPLEX [59]. Queries are specified using the Select - From - Where clauses in SQL. AmosQL furthermore has aggregation operators, nested subqueries, disjunctive queries, quantifiers, and is relationally complete.

Data model

The data model of Amos II is a functional data model. The basic concepts of the data model are objects, types, and functions. All entities in the database are represented as objects. There are both system and user-defined objects. Objects are classified into types making each object an instance of one or several types. The set of all instances of a type is called the extent of the type. Functions model the semantics of objects. They model properties of objects, computations over objects, and relationships between objects. They furthermore are basic primitives in functional queries and views.

Query processing

The query processing in Amos II has the following steps. The query compiler translates AmosQL statements first into object calculus and then into object algebra expressions. The object calculus is expressed in an internal simple logic based language called ObjectLog [80], which is an object-oriented dialect of Datalog. As part of the translation into object algebra, many optimizations are applied on AmosQL expressions relying on its functional and multi-database properties. During the optimization steps, the object calculus expressions are rewritten into equivalent but more efficient expressions.

The prototype systems SAQ and TM-Viewer utilize the data model, query language and query processing of the Amos II system. They extend the Amos II system in the following way:

- Both SAQ and TM-Viewer implement functionality in Amos II for automatic generation of RDF views over relational databases and Topic Maps documents correspondingly.

- SAQ and TM-Viewer extend Amos II query processing by rewrite transformations applied on Datalog that are shown to substantially improve the query processing time of SPARQL queries to RDF views of relational databases and Topic Maps respectively.

4 The Prototype Systems

In this chapter an overview of the architecture and functionality of the prototype systems SAQ and TM-Viewer is presented. Furthermore, the steps of the processing of SPARQL queries to RDF views over relational databases and Topic Maps respectively are also described.

4.1 SAQ System

The developed SAQ system for long-term preservation of relational databases follows conceptually the OAIS reference model [57]. SAQ provides

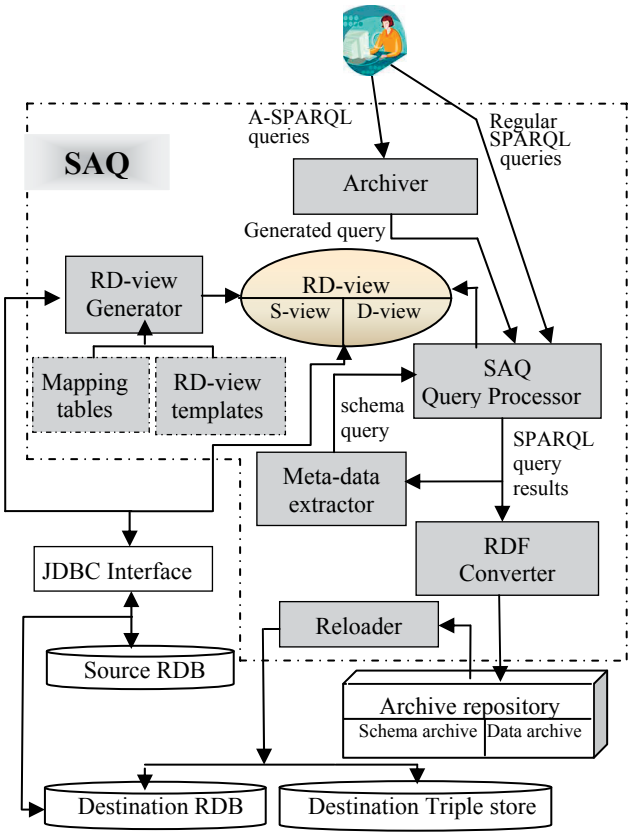

Figure 4.1. SAQ Architecture

functionality for the OAIS' Ingest component, in particularly on generating the content information in the AIPs (Archival Information Packages) when preserving relational database contents as RDF.

In order to preserve both schema and data from an RDB, it is important to represent not only the contents of the RDB as RDF, but also the schema. Therefore the RD-view is defined as a union of a *schema view (the S-view)*, representing the RDB schema, and a *data view (the D-view),* representing the RDB contents. To allow for interoperability with other systems mapping RDBs to RDF, e.g. [6][31][50], the data view mappings conform to the direct mapping recommendations by W3C Recommendation [41] and more particularly to the augmented direct mapping proposed in [30].

The architecture of the SAQ system is presented in Figure 4.1. *The source RDB* is the underlying RDB, which can be queried by SPARQL and preserved by A-SPARQL queries.

The RD-view generator automatically generates one RDF-view, *RD-view* over each *source RDB* by reading the database schema though a *JDBC interface*. The *RD-view templates* thereby provide general prototypes for the structure of the RD-view for any relational database, and the contents of the *mapping tables* provide RDB-to-RDF mappings for specific relational meta-data into RDF. An archival query defined in an extended SPARQL dialect, *A-SPARQL* is processed by the *archiver* and translated into a corresponding *generated query* in SPARQL, which is sent to the *SAQ query processor*. The generated query retrieves the data to archive from the RDB. Regular non-archiving SPARQL queries are sent directly to the SPARQL query processor.

The SAQ query processor executes the SPARQL queries to the RD-view by accessing the source RDB through the JDBC interface.

During the execution of the generated SPARQL query, the property URIs of the triples to be archived are collected by the *meta-data extractor* while iterating over the result stream. When all data to archive have been processed, the meta-data extractor joins the collected properties with the S-view by issuing *a schema query*.

The archived content retrieved by the generated query and the corrsponding meta-data retrieved by the schema query are written by *the RDF converter* into two N-Triples [46] formatted files, i.e. *the schema archive* a*nd the data archive* in the archive repository.

Later on, when a preserved database is to be restored, the *reloader* reads from the archive repository the two archive files and makes the database live again by populating it into a *destination RDB* or alternatively a *destination triple store.* When an RDB is restored, the reloader first reads the schema archive in order to generate the RDB schema and then populates the destination RDB by reading the data archive. After the destination RDB is restored, it can be queried or re-archived with A-SPARQL using SAQ. When the destination DBMS is an RDF triple store system, both the schema and data ar-

chive files are loaded directly into the triple store and can there be queried with SPARQL.

4.2 TM-Viewer System

The architecture of the TM-Viewer system is depicted in Figure 4.2. The core of the system is the *TM-Viewer database* that represents imported *Topic Maps data*. The Topic Maps data model is defined by the *Topic Maps conceptual schema*. The *TM-view* is a system generic RDF view of Topic Maps. It is generated by the *RDF view generator* based on mapping rules between basic Topic Maps concepts and the corresponding RDF and RDFS concepts. The *TM Importer* parses queried *XTM files* and populates the TM-Viewer database.

The TM-view itself is defined in terms of a *TM-Schema* and a *TM-Data view*. The TM-Schema view maps the elements of the Topic Maps data

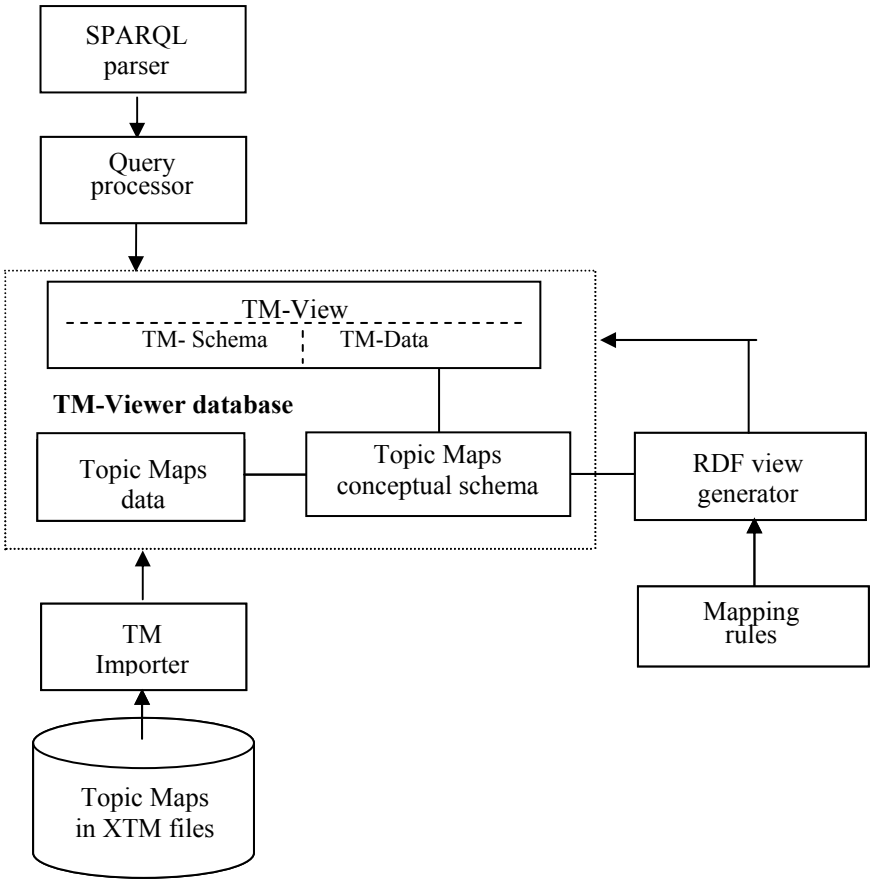

Figure 4.2. TM-Viewer Architecture

model to RDF concepts, while the TM-Data view represents imported Topic Maps data objects as RDF.

A SPARQL query is specified in terms of the TM-view. It is first parsed into a Datalog dialect by *the SPARQL parser*. *The query processor* rewrites and optimizes the generated Datalog query to produce an execution plan, which is interpreted.

4.3 Datalog based SPARQL Query Processing

To process SPARQL queries both prototype systems SAQ and TM-Viewer first generate Datalog queries to a declarative RDF view of a relational database or a Topic Map, respectively, and then transforms the SPARQL queries to SQL or an execution plan, based on logical transformations. The steps of the query processing are illustrated in Figure 4.3 where:

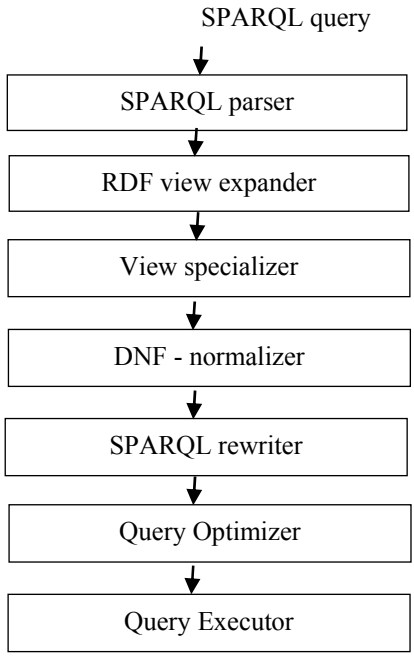

Figure 4.3. SPARQL Query Processing

1) *The SPARQL parser* transforms the SPARQL query into a Datalog expression where each triple pattern (TP) in the query becomes a reference to the RDF view.
2) *The RDF view expander* recursively expands each RDF view reference in the query into the RDF view definition.

3) *The view specializer* then enables a transformation called *view specialization* [29] that substantially reduces bound-property queries.
4) *The DNF-normalizer* transforms further the query into a disjunctive normal form (DNF) predicate. We call the DNF-normalized query a *DNF-normalized query*.
5) *The SPARQL rewriter* applies on the DNF-normalized and simplified query a number of query transformations that simplify the query and improve the execution time.
6) *The Query Optimizer* translates the DNF query into an optimized execution plan. In SAQ it includes a SQL generator that generates SQL from operators calling SQL.
7) *The Query Executor* interprets the optimized algebra expression to produce the result of the query. In SAQ this includes execution of the generated SQL queries sent to the RDBMS followed by *post-processing* of expressions in the RDF view that are not processed by the SQL engine.

5 Technical Contributions

In this chapter the technical contributions are described to some detail in order to explain how they contribute to answering the research questions and give the reader the intuition behind the chosen approaches. The technical details are presented in Paper I – IV.

5.1. Scalable Long-term Preservation and Querying of Structured Data in terms of RDF

To represent relational databases (RDBs) in RDF, the SAQ system automatically generates a specialized RDF view, *RD-view* (Paper II, Paper III) for each given RDB by accessing the RDB catalogue. The RDB to RDF mapping in SAQ conforms to the *direct mapping* recommended by W3C [41], and more particularly to the *augmented direct mapping* proposed in [30], which is proven to guarantee information preservation. In the RD-view a unique RDFS class is defined for each relational table, except for *link tables* representing set-valued properties as many-to-many relationships. In addition, a unique RDF property is defined for each attribute in a table. The RD-view is defined as a union of an *S-view*, representing the schema of the RDB, and a *D-view*, representing the data stored in the relational database.

The S-view definition itself is the same for any RDB. It is defined as a large union of unions of sub-views representing relational schema concepts about tables, columns, types, primary keys, foreign keys, other constraints, and indexes. Since the S-view is complex but contains little data and its extent changes only when the database schema is altered, the S-view is materialized in main memory in SAQ.

Based on the S-view, i.e. on the imported RDB schema information, SAQ generates a D-view for each specific RDB. A D-view definition for each concrete database is generated instead of defining a generic D-view since this enables substantial query reduction at run time via specialization of the view definitions [29].

The D-view is defined as a large union of:

- A union of sub-views for each non-foreign attribute to represent the attribute values as literals;

- A union of sub-views for each foreign key relationship to represent foreign key values by URIs;
- A union of sub-views for each many-to-many link table to represent the values in link tables as URIs;
- A union of sub-views for each non-link table to represent the classes of its row identifiers.

The automatic generation of the RD-view having the above described definition provides the answer to **research question 1a**.

Flexible long-term preservation of a selected part of an RDB is enabled in SAQ by specifying an *archival query* to the RD-view in an extended SPARQL dialect, A-SPARQL (Paper III). This is the provided answer to the **research question 2** since A-SPARQL allows a flexible selection of data to be archived in terms of a SPARQL-like query to the RD-view. The result of the archival query is the RDF triples representing the archived data stored in the *data archive file*. While executing the archival query, the SAQ system simultaneously produces sufficient meta-data to enable reconstruction of the selected part of the archived RDB. These meta-data are stored in the *schema archive file.*

For processing an archival query in A-SPARQL SAQ internally generates a corresponding SPARQL query to select the triples of the database to archive. The archival queries are straight-forward to translate into SPARQL CONSTRUCT queries. Archiving unions of sets of triples, e.g. for different classes and properties, makes the generated SPARQL queries become UNION CONSTRUCT queries.

Archival queries typically select sets of attributes of tables to archive. This corresponds to selecting sets of RDF properties in the RD-view of the database to be archived. In the motivating example in section 2 all properties of the subjects from classes representing tables *taxonomy* and *location* that have some special properties are selected for archival. Therefore, in the generated SPARQL CONSTRUCT query property *p* in one or several TPs is a variable, i.e. the queries are unbound-property queries.

Unlike processing of bound-property queries to the RD-view, processing and optimization of unbound-property queries to the RD-view have been little studied [31]. To address this problem in connection with scalable data preservation of relational databases some special query rewriting optimizations for optimizing unbound-property queries to the RD-view have been developed (Paper II, Paper III). This answers **research question 3a:**

- The *Group Common Terms (GCT) query transformation* optimizes SPARQL queries in such a way that, instead of naively accessing the RDB column-by-column following the RD-view definition, the RDB is accessed row-by-row instead.

- The *is-literal query transformation* optimizes SPARQL queries in such a way that SQL LIKE conditions are not issued on table attributes whose values are represented by URIs in the RD-view.
- The *type-match query transformation* optimizes SPARQL unbound-property queries so that SQL comparison conditions are issued only on attributes represented with correct literal types in the RD-view.
- The *foreign-key relationship (FKR) transformation* optimizes SPARQL unbound-property queries where a subject-object join variable is shared between two UPTPs. This enables SQL queries to be generated only for tables where there is a foreign key relationship between the tables referenced by the joined UPTPs.
- *The eliminate S-view transformation* optimizes unbound-property queries so that a subject in the S-view is never joined with a subject in the D-view constructed to be an URI, which reduces the number of generated SQL queries.

Archival queries can also be bound-property queries containing only BPTPs where the properties are URIs representing RDF properties in the RD-view. Such queries are processed in SAQ using the methods in [29][31].

To evaluate the performance of typical archival queries a new benchmark called *ABench* was developed (Paper III). ABench is defined as a set of typical archival queries, specified in A-SPARQL, that archive selected parts of databases generated by the Berlin benchmark data generator [5]. A new benchmark was developed since the archival queries generate SPARQL CONSTRUCT unbound-property queries with UNION clauses, which is not covered by any existing benchmark.

SAQ query optimization strategies have been evaluated using ABench (Paper III). The experiments showed that the proposed query rewriting optimizations substantially improve the query execution time for unbound-property queries selecting RDB contents to archive. We also compared the performance of our approach with other systems processing SPARQL queries over views of RDBs and found that the proposed optimizations improve query scalability compared with the approaches used in those systems.

5.2. Scalable Reconstruction of RDF archived Structured Data

To reconstruct an RDF-archived RDB and make it live again the *reloader* module of SAQ was developed (Paper IV). When the contents of an archived RDB is to be restored, the reloader first reads the schema archive file and executes a schema reconstruction algorithm to automatically construct the minimal RDB schema required for reconstructing the archived data. Since

only selected parts of the RDB are archived, a corresponding partial RDB, called the *reconstructed RDB,* is created containing only the parts of the schema describing the archived data. The thus created RDB is then populated by reading the data archive and converting the read data into relational attribute values according to the schema.

For migrating data from RDBs to RDF repositories, the contents of the schema and data archive files can be directly loaded into an RDF repository system, e.g. [2][64][79].

For scalable reconstruction of RDF-archived data in the reconstructed RDB we have developed the *Triple Bulk Load (TBL)* approach (Paper IV). With the TBL approach the relational data is reconstructed by using the bulk load facility of the RDBMS. The relational bulk loader requires one CSV file of rows for each reconstructed table. Since RDF does not prescribe any specific triple order, the reloader first joins the data archive triples with metadata describing the attributes for each table and then orders the data archive per CSV file per table to populate, per triple subject, and per attribute order. To do this grouping of RDF triples, the reloader uses an *Order-by* query in the RDBMS. Once the triples are ordered the CSV files are generated through a *CSV row generation algorithm* that post-processes the result scan of the Order-by query.

We compare the performance for the TBL approach with the naïve *Insert Attribute Value (IAV)* approach where relational data is straight-forwardly populated by executing SQL INSERT or UPDATE statements to incrementally insert attribute values for each read RDF data archive triple. The performance results show that the TBL approach is substantially faster than IAV, which provides the answer to the **research question 4**.

5.3. Scalable Querying of Semi-structured Data in terms of RDF

Topic Maps are represented in the TM-Viewer (Paper I) in terms of a defined general conceptual schema of the Topic Maps data model. The designed conceptual schema is a modification of the definitions in [39][69] to enable 1:1 mappings between Topic Maps concepts and RDF representations. A functional data model [75] is used for representing the conceptual schema.

A declarative RDF view in Datalog of Topic Maps, the *TM –view,* represents the conceptual schema is defined (Paper I). It is automatically generated in the TM-Viewer system. The TM-view over a Topic Maps data source, i.e. an XTM file is defined as a union of a schema view, *TM-Schema* representing meta-information in the conceptual schema and a data view, *TM-Data* representing Topic Maps data imported from XTM file.

The schema view (TM-Schema) is independent of Topic Maps data, and it is always the same. Therefore it is materialized by the system once and for all.

The *TM Importer* module in the TM-Viewer system translates each Topic Maps entity read from an XTM file into a Topic Maps object representing an entity in terms of the Topic Maps conceptual schema. Depending on what kinds of Topic Maps elements are read, imported Topic Maps objects of different types following the conceptual schema are created. The type of the imported object furthermore determines the corresponding RDFS class in the data view.

The data view (TM-Data) is defined as a large union of:

- A union of the sub-views that specify imported Topic Map objects as instances of RDFS classes defined in the schema view;
- A union of the sub-views defining attribute values of imported Topic Maps objects as literals;
- A union of the sub-views defining relationships between imported Topic Maps objects.

The automatically generated TM-view with the described definition provides the answer to **research question 1b** since it represents both Topic Maps schema and data as RDF. Furthermore, it enables RDF-based tools to process Topic Maps, and SPARQL can be used to search indices of documents and websites defined by Topic Maps.

The view definition of the TM-view, particularly the data view definition of the TM-Data becomes complex for large XTM files containing many Topic Maps objects and therefore efficient query processing is an issue and therefore the query processing of SPARQL queries to the TM-view was studied in details (Paper I). The query-processing time for queries over large Topic Maps documents, both the optimization and execution time, have shown to substantially reduce by the following Datalog-based rewrite transformations:

- The *Property reduction transformation* optimizes SPARQL bound-property queries to the TM-view. BPTPs in a view expanded query are reduced from large disjunctions to single conjunctions.
- The *Disjunct reduction transformation* optimizes unbound-property queries to the TM-view where a property variable in a UPTP is restricted to be a specific URI in another TP. This reduces substantially the number of the disjuncts in the large disjunction of a DNF normalized unbound-property query.

Furthermore, *Bi-directional encoding of URIs* through multi-directional foreign functions [80] optimizes SPARQL queries to the TM-view in such a way that it allows straightforward Topic Maps-RDF-Topic Maps transformations, which prevents from scanning the whole data.

The developed rewrite techniques provide the answer to **research question 3b** since they optimize SPARQL queries to RDF views over Topic Maps data.

The above rewrite transformations were evaluated (Paper I) using Internet available Topic Maps data and SPARQL queries, some of which are taken from the Topic Maps query language use cases [55]. The performance measurements clearly show that the rewrite techniques significantly improve the query processing time for both bound-property and unbound-property queries to TM-view.

6 Related Work

Long-term Preservation of Relational Databases

Testbed [9], SIARD [72] and RODA [54] are projects that have developed strategies for long-term preservation of relational databases based on XML. In both Testbed and RODA the data and metadata of relational databases are preserved as XML. SIARD has an own format for preservation which is based on XML and SQL1999, and the industry standard ZIP. In contrast, in SAQ we use RDF to represent the relational database to archive. Both XML and RDF are neutral data formats that don't rely on current DBMS technology and provide hardware and software independence. These make both of them suitable for long-term preservation of databases. However, RDF has the following advantages comparing to XML. In RDF the identifiers are URIs which are universal global unique identifiers that allow identifiers from one database or table to be linked with identifiers from other data. Data can be represented as XML in many different ways depending on a defined DTD or XML schema [32] while the RDF-Schema in RDF provides standard meta-data representation for describing all kinds of data, including relational databases [21]. Furthermore, representing relational data as RDF allows migration from RDBs to RDF repositories which are gaining increasing popularity compared to XML native repositories.

In the above mentioned related approaches the entire relational database, both the data and schema are migrated into XML or XML-based format and stored in files. By contrast, in SAQ we provide selective archival of user-specified parts of a relational database as RDF using an extended SPARQL query language, A-SPARQL.

Mapping and Querying Relational Databases as RDF

Virtuoso RDF Views [50], D2RQ [6], and SquirrelRDF [65] are systems that allow mapping of relational tables and views into RDF to make them queriable by SPARQL. These systems implement compilers that translate SPARQL directly to SQL. In contrast, SAQ first generates Datalog queries to a declarative RD-view of the relational database, and then transforms the SPARQL queries to SQL, based on logical transformations. We have shown that query transformations on this representation significantly improve per-

formance for SPARQL unbound-property queries selecting RDB contents to archive.

The system closest to SAQ is Ultrawrap [31] where, like in SAQ, an RDF view over a relational database is generated as a union of sub-views. While the RDF view in Ultrawrap is defined in SQL in a specific SQL dialect, in SAQ the view is defined in a Datalog dialect and thus it is independent of the RDBMS. Furthermore, since the view in Ultrawrap is defined in a concrete RDBMS the query optimizations are also dependent on the RDBMS, and thus the performance measurements in [31] show different results in different systems. By contrast, in SAQ the proposed optimizations are made in the SAQ query processor and are not dependent on the back-end RDBMS.

Unlike SAQ, neither D2RQ, nor Virtuoso, nor Ultrawrap includes the schema view in the RDF view of RDBs. The inclusion of the S-view is very important when archiving relational databases, since the database schema is needed to reconstruct an archived database. The logical rewrites of SAQ enable scalable processing over full RDF views, including the schema part.

Mapping and Querying Topic Maps as RDF

The proposal [47] for mapping Topic Maps to RDF is fairly complete but very complicated and the translation from the Topic Maps data model to RDF is non-reversible [71]. For example, it requires seven statements to represent the information content that would be modeled using one statement in RDF [71]. By contrast, the prototype system TM-Viewer is based on a canonical and yet simple conceptual schema representation that maps 1:1 to both Topic Maps and the corresponding RDF ontology representation of Topic Maps. The mapping rules from the conceptual schema to RDF are very straightforward: An RDFS class is defined for each entity type as well as an RDF property for each function along with its range and domain definitions. These rules define the declarative general RDF based TM-view over any Topic Maps data imported to the TM-Viewer system. The TM-view can be queried with SPARQL.

In [44][47] it was shown that a Topic Map transformed to RDF can be queried using F-Logic syntax [61] or the RDF query language SquishQL [38]. We support querying of the Topic Maps view by the standard RDF query language SPARQL. We are not aware of any other implementation of general queries over RDF views of Topic Maps. Moreover, in this Thesis it is studied in details the problem concerning the query processing of SPARQL queries to the RDF views of Topic Maps. We defined a small number of rules important for improving performance of such queries. We made theoretical proofs how queries were reduced by the rules and showed by measurements their practical impact on different query types.

7 Summary

This chapter summarizes the technical contributions of the Thesis.

7.1 Scalable Preservation, Reconstruction and Querying of Relational Databases in terms of Semantic Web Representations

An approach has been developed for selective scalable long-term preservation of RDBs as RDF in terms of SPARQL queries, implemented in the SAQ system. The proposed approach is suitable for archiving research data used in scientific publications where it is desirable to preserve only selected parts of an RDB. The archival of user-specified parts of a RDB is specified using an extension of SPARQL, A-SPARQL, having an archival statement for selective archival.

The SAQ system for long-term preservation of relational databases follows conceptually the OAIS reference model [57]. In particular, this work has concentrated on the functionality of the Ingest component in the OAIS model on generating the content information when preserving relational database content as RDF.

To evaluate the performance of typical archival queries, the archival benchmark ABench has been defined that archives selected parts of databases generated by the Berlin benchmark data generator [5]. In experiments, the SAQ optimization strategies were evaluated by measuring the performance of A-SPARQL queries selecting triples for archival queries in ABench.

SAQ automatically generates an RDF view of an RDB called the RD-view. The RD-view can be queried and archived with A-SPARQL queries that are translated into SQL queries sent to the RDB. An archival query internally generates a corresponding CONSTRUCT SPARQL query. Since an archival query usually selects sets of attributes of tables to archive, the generated CONSTRUCT SPARQL query is typically an unbound-property or UNION query. To achieve scalable data preservation and reconstruction for such queries, SAQ uses some special query rewriting optimizations.

Using ABench queries and data generated by the Berlin benchmark generator, SAQ's rewriting optimizations were experimentally shown to improve query execution time compared with naïve processing. Compared with

not using the optimizations, they reduce the number of SQL queries to execute. The performance of SAQ was compared with that of other systems that support SPARQL queries to views of existing relational databases. It was shown experimentally that SAQ with the rewrite optimizations performs better than those systems for all queries returning large results. In general, the SAQ optimizations are useful not only for archival queries, but also for unbound-property and UNION queries.

7.2 Scalable Reconstruction of RDF-archived Relational Databases

Approaches for scalable reconstruction of relational databases archived as RDF have been investigated. An RDF-archived relational database is reconstructed from a schema archive file and a data archive file, both in N-Triples format. The archives contain RDF triples representing the relational schema for the archived content, and relational data content, respectively. When an archived RDB is to be reconstructed, the schema archive file is read to automatically reconstruct the RDB schema in another RDBMS. The schema reconstruction algorithm is based on identifying relational database schema elements by queries to the schema archive. The reconstructed RDB is then populated by reading the data archive file and converting the stored RDF triples into relational attribute values according to the schema. We have investigated two approaches to populate data into the reconstructed RDB: the IAV approach and the TBL approach.

With the IAV approach the reconstructed RDB is populated by first generating stored procedures, assigners that execute an SQL INSERT or UPDATE statement for each archived attribute value.

With the TBL approach the reconstructed RDB is populated by bulk loading CSV files generated from the data archive for the tables to be reconstructed. The bulk loader requires one CSV file of rows for each reconstructed table. Since RDF does not prescribe any specific triple order, the data archive file needs to be first regrouped per table and row in order to create the CSV files. The TBL approach therefore requires pre-steps to regroup the data archive file in order to generate a CSV file per reconstructed table. For the regrouping the RDF triples in the data archive file are first bulk-loaded into an RDB together with meta-data. Then the order-by facility of the RDBMS is used to group the bulk-loaded triples per CSV file, row identifier, and attribute position in order to produce the CSV files in one pass with limited memory.

Our experiments show that the TBL approach is substantially faster than the IAV approach, despite the added grouping and post-processing.

7.3 Scalable querying of Topic Maps data in terms of RDF

A general system for exposing the semi-structured data Topic Maps as RDF was implemented, the TM-Viewer. With the TM-Viewer RDF views of Topic Maps can be queried using SPARQL.

A functional conceptual schema was defined for the Topic Maps data model. Generic 1:1 mappings from the conceptual schema into RDF were defined as an automatically generated declarative RDF view, the TM view. The TM-View consists of two parts, the TM-Schema view and the TM-Data view. The TM-Schema view describes the Topic Maps conceptual schema as RDF triples, while the TM-Data view describes data represented by the Topic Maps conceptual schema as RDF.

The TM-Viewer enables SPARQL queries to the TM-view which enables SPARQL queries to any Topic Maps XTM file.

The query processing of SPARQL queries to the TM-view is based on the two following query rewrite transformations, i.e. the property reduction transformation and the disjunct reduction transformation. It was proved that the property reduction transformation reduce a class of common disjunctive SPARQL queries, i.e. bound-property queries to the TM-view into conjunctions. Thus no normalization is needed and these queries are executed more efficiently. This class includes all queries in the standard Topic Map query language use case [55].

Furthermore, the approach also allows expressing SPARQL queries that combine Topic Maps meta-data with Topic Maps contents, i.e. unbound-property property queries. It was shown that processing of unbound-property queries was substantially improved by the disjunct reduction transformation.

It was in addition shown that bi-directional encoding of Topic Maps object URIs enables substantial performance improvement for SPARQL queries to the TM-view over large XTM files.

8 Summary in Swedish

Betydelsen av forskning om digital långtidsarkivering har ökat under de senaste tio-femton åren. Många tidskriftsartiklar och böcker som beskriver problem, verktyg och tekniker för digital dataarkivering har skrivits, och många standarder för olika arkiveringsmodeller har publicerats. Det mesta av detta arbete har fokuserat på arkivering av filbaserade digitala objekt som dokument, bilder och webbsidor. Betydligt mindre arbete har fokuserat på långtidsarkivering av strukturerade data, dvs databaser och vetenskapliga mätdata, trots att det finns ett erkänt stort behov för sådan forskning. Arkivering av vetenskapliga data tillsammans med vetenskapliga publikationer skulle bland annat bidra till att bättre dokumentera ursprunget för vetenskapliga resultat och underlätta återskapande av forskningsresultat.

För att uppnå en långsiktig dataarkivering är det önskvärt att innehållet i en databas sparas i ett databasoberoende format, så att databasen kan återskapas efter lång tid med hjälp av senaste databasteknik. Ett lovande format för databasoberoende långtidsarkivering av data är RDF (Resource Description Framework). RDF är ett dataformat för lagring och utbyte av olika sorters kunskap och data i form av satser uttryckta som trippler, (subjekt, predikat, objekt), t.ex. (<eBird:Havsörn>, <eBird:boplats>, <eBird:Havsband>) för att representera att havsörnar bor i havsbandet. RDF är ett centralt begrepp för kunskapsrepresentation inom det forskningsområdet "semantisk web". Med hjälp av RDF-baserade format kan man skapa standardiserade representationer för alla typer av data, inklusive konventionella relationsdatabaser. RDF tillhandahåller vidare ett standardiserat frågespråk SPARQL med vilket man kan uttrycka godtyckliga sökningar bland RDF-data. Denna avhandling utreder hur semantiska webb-representationer, särskilt RDF och SPARQL, kan möjliggöra flexibel och skalbar arkivering, rekonstruktion och sökning i databaser.

En metod för skalbar långtidsarkivering av utvalda delar av relationsdatabaser i RDF-format med hjälp av SPARQL-frågor har utvecklats, implementerats och utvärderats i det egenutvecklade prototypsystemet *SAQ (Semantic Archieve and Query)*. Den föreslagna metoden är lämplig för arkivering av forskningsdata som används i vetenskapliga publikationer, där det är önskvärt att arkivera endast de delar av en databas som är relevanta för en publikation. Arkivering av valda delar av en databas specificeras genom en föreslagen utvidgning av SPARQL, benämnd *A-SPARQL*. SAQ genererar automatiskt en RDF-vy av en relationsdatabas så att man kan söka i databasen

med SPARQL-frågor och arkivera valda delar med arkiveringsfrågor uttryckta i A-SPARQL. SAQ översätter därvid frågorna till det frågespråk, SQL, som används för att söka i relationsdatabaser för att skickas till relationsdatabasen för utförande. Resultatet av en arkiveringsfråga är en *dataarkivfil*, som innehåller en RDF- representation av den utvalda data i relationsdatabas för arkiveringen. Systemet genererar därvid också en *schemafil*, som innehåller all nödvändig information om strukturen för arkiverade data för att senare kunna rekonstruera den arkiverade databasen. För att uppnå skalbar arkivering använder SAQ speciella regler för att optimera frågor, vilka experimentellt har visats förkorta tiden väsentligt, även jämfört med andra system som tillhandahåller SPARQL-frågor över befintliga relationsdatabaser.

För återskapning av en arkiverad relationsdatabas har vi utvecklat en strategi *Triple Bulk Load (TBL)*, som snabbt rekonstruerar och återupplivar en RDF-arkiverad databas. När en arkiverad relationsdatabas skall återskapas läser SAQ först schemaarkivfilen och utför en rekonstruktionssalgoritm för att automatiskt återskapa endast den del av databasstrukturen som representerar arkiverade data. Därefter läses datarkivfilen för att lägga in arkiverade data i databasen. TBL använder därvid s.k. “bulk loading”- vilket är en facilitet i en relationsdatabas för att snabbt lägga in nya data. Våra experiment visat att TBL är betydligt snabbare än en naiv ansats, trots att TBL kräver mer komplicerad sortering och efterbehandling.

Som alternativ till relationsdatabaser kan man använda s.k. semistrukturerade databaser där databasstrukturen är mer informell. Ett sådant databasformat är s.k. Topic Maps, vilka används för beskriva digitala innehållsförteckningar. För att kunna söka och arkivera Topic Map databaser i termer av RDF och SPARQL har prototypsystemet *TM-Viewer* implementerats. Systemet baseras på ett en generell beskrivning av Topic Map modellen i form av ett s.k. konceptuellt schema. Generella ett-till-ett-avbildningar från det konceptuella schemat till RDF har definierats som en automatiskt genererad RDF-vy, *TM-vyn*, som representerar vilken Topic Map databas som helst. Metoder för att optimera SPARQL frågor mot TM-vyn har utvecklats och utvärderats. Utvärderingarna visade att optimeringsmetoderna avsevärt förbättrar söktiden.

9 Acknowledgments

Foremost I want to thank my supervisor Prof. Tore Risch. Thank you first for giving me the opportunity for PhD studies in Database Technology in Uppsala. This was a turning point for me. Thank you for the time and efforts you invested teaching and sharing your knowledge in Computer Science and Databases. I learned a lot under you. I felt that I grew and developed both my knowledge and self-confidence. Thank you for fruitful research discussions, and interesting conversations. They made me always think and look for the "proper solution". Thank you for being indefatigable. This made me stronger.

My special thanks to my best friend Ventzislav Yantchev. You were with me during all these difficult years. You were always listening to me, always encouraging me and not least always making me laugh. I appreciate a lot our conversions, the funny silly things that we say to each other, and even your Vratza crazy entertaining stories. Thank you for being "my brother".

I want to express my considerable gratefulness to my female friends Caroline Olofsson and Nora Masszi. Dear Caroline, thank you for letting me become a small part of your busy life, you were welcoming me at your home meeting your family and friends, and of course we were having from time to time babble. Nora, my dear friend, you have always been so supportive and understandable, and always ready to listen and to calm me down despite your heavy time committed with your big family.

Thanks to Mikael Sternad, a good friend of mine, who was patiently listening during our lunches to my sometimes too long stories about troubles and difficulties that I met during the years.

I am very grateful to the former UDBL members Sabesan, Johan, Russlan and Erik. My friend Sabesan, you were always nice and I enjoyed a lot our chats. Johan, thank you for giving me initial directions about how things work in the department and in the UDBL group, and not least in the Semantic Web research. Russlan, thank you for practical advice. Erik, thank you for helping me understand "the Swedish way". Here, I would like also to mention Milena, a former UDBL member and a friend of mine. Thank you for convincing and encouraging me to make a PhD in Database Technology.

To the current UDBL members Sobhan, Minpeng, Thanh, Andrej, Cheng, Kjell and Lars I am also grateful. I enjoyed the time spent with you on discussing research and teaching problems, and certainly the time when we were having small chats about insignificant things. Sobhan, it was a pleasure

to work with you on the Database course. Minpeng, I appreciated our dialogs in the coffee room.

I do not want to miss the chance to thank to the CSD department and the IT institution leaders Lars-Henrik Eriksson, Håkan Lanshammar, and Michael Thuné for their help with administrative and financial issues. My special thanks to Ulrika Andersson, Ulrika Jaresund and Anne-Marie Jalstrand. You were very nice and helpful in personnel and teaching process issues.

I am also grateful to my Bulgarian friends Maya Neytcheva, Ilia Katardjiev, Ivan Christoff, Pavlin Mitev and Gergana Angelova Hamberg who I spent a pleasant time with having a lot of home discussions and nice parties.

Finally but not least, my dear Lars, thank you for being beside me during the latest years of my PhD. I appreciate your support, logical reasoning and patience, and most of all “the quiet tranquility” and love that you give me.

Bibliography

1. Ad hoc Strategic Committee on Information and Data, Final Report to the ICSU Committee on Scientific Planning and Review, http://www.icsu.org/publications/reports-and-reviews/scid-report/scid-report.pdf, 2008.
2. Allegro Graph, http://www.franz.com/agraph/allegrograph/ .
3. Avian Knowledge Network, http://www.avianknowledge.net/content/
4. Bigdata, http://www.bigdata.com/ .
5. C. Bizer and A. Schultz: Berlin SPARQL Benchmark (BSBM) Specification, Data Generator and Test Driver, http://www4.wiwiss.fuberlin.de/bizer/BerlinSPARQLBenchmark/spec/20080912/index.html#datagenerator.
6. C. Bizer, and A. Seaborne, D2RQ-Treating Non-RDF Databases as Virtual RDF Graphs, Poster at 3rd International Semantic Web Conference, 2004.
7. V. Christophides, P., Buneman: Report on the First International Workshop on Database Preservation, PresDB'07. SIGMOD Record, Vol. 36, No. 3:55–58, September 2007.
8. C. F. Pereira Aldeias: Open Archival Information Systems for Database Preservation, PhD Thesis, University of Porto, 2011.
9. Digital Preservation Testbed. From Digital Volatility to Digital Permanence: Preserving Databases. Technical report, Dutch National Archives and the Dutch Ministry of the Interior and Kingdom Relations, 2003.
10. D. Beckett, T. Berners-Lee: Turtle - Terse RDF Triple Language, W3C Team Submission 28 March 2011, http://www.w3.org/TeamSubmission/turtle/ .
11. D. Brickley, R.V. Guha (ed.): RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-schema-20040210/ .
12. D. Giaretta: Advanced Digital Preservation, Springer, 2011.
13. D. L. McGuinness, F. van Harmelen (ed.): OWL Web Ontology Language Overview, W3C Recommendation 10 February 2004, http://www.w3.org/TR/owl-features/ .
14. E. F. Codd: A relational model for large shared data banks, Communications of the ACM, 13(6): 377-387, 1970.
15. F. Manola, E. Miller (ed): RDF Primer, W3C Recommendation 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-primer-20040210/ .
16. G. Klyne, J. J. Carroll (ed.): Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ .
17. G. Klyne, J. J. Carroll, B. McBride (ed.): RDF 1.1 Concepts and Abstract Syntax, W3C Working Draft 15 January 2013, http://www.w3.org/TR/2013/WD-rdf11-concepts-20130115/.
18. G. Fahl,T. Risch: Query Processing over Object Views of Relational Data, The VLDB Journal , Vol. 6 No. 4, November 1997, pp 261-281.

19. G. Moore: RDF and Topic Maps: An exercise in convergence, XML Europe 2001 Berlin, http://xml.coverpages.org/moore-topicmapsrdf200105.pdf.
20. The Hypermedia/Time-based Structuring Language, http://eprints.soton.ac.uk/250827/6/html/node16.html.
21. H. Stuckenschmidt, F. Harmelen: Information Sharing on the Semantic Web, Springer, 2005, ISBN 3-540-20594-2.
22. H.Lin, T.Risch, T.Katchaounov: Adaptive data mediation over XML data. In special issue on 'Web Information Systems Applications' of *Journal of Applied System Studies (JASS)* ,Cambridge International Science Publishing, 3(2), 2002.
23. S. Heuscher, S, Jarmann, P. Keller-Marxer, F., Mohle.: Providing authentic long-term archival access to complex relational data. In: Ensuring Long-Term Preservation and Adding Value to Scientific and Technical Data. European Space Agency, 2004.
24. Internationalized Resource Identifiers (IRIs), http://www.ietf.org/rfc/rfc3987.txt.
25. J. Domingue, D. Fensel, J. A. Hendler (ed): Handbook of Semantic Web Technologies, Springer, 2011.
26. J. D. Ullman, H. Garcia-Molina, J. Widom: Database Systems: The Complete Book, Prentice Hall, 2002.
27. J. Hunter: Scientific Publication Packages – A Selective Approach to the Communication and Archival of Scientific Output, *The International Journal of Digital Curation*, 2006, iss. 1 vol. 1, pp. 33-52.
28. J. Masanès (ed.): Web Archiving, Springer, 2006.
29. J. Petrini: Querying RDF Schema Views of Relational Databases, PhD Thesis, Uppsala University, Department of IT, ISSN1104-2516, http://www.it.uu.se/research/group/udbl/Theses/JohanPetriniPhD.pdf, 2008.
30. J. Sequeda, M. Arenas and D. P. Miranker, On Directly Mapping Relational Databases to RDF and OWL, *WWW 2012*, April 16–20, 2012, Lyon, France.
31. J. F. Sequeda and D. Miranker, Ultrawrap: SPARQL Execution on Relational Data, Technical Report http://apps.cs.utexas.edu/tech_reports/reports/tr/TR-2078.pdf .
32. J. Tauberer: What is RDF and what is it good for?, http://www.rdfabout.com/intro/#Comparing%20RDF%20with%20XML, 2008.
33. K. Ahmed, G. Moore: An Introduction to Topic Maps, http://msdn.microsoft.com/en-us/library/aa480048.aspx.
34. K. Thibodeau: Overview of Technological Approaches to Digital Preservation and Challenges in Coming Years. In The State of Digital Preservation: An International Perspective. Documentation Abstracts, Inc. - Institutes for Information Science, 2002.
35. L. Garshol: Living with Topic Maps and RDF, Topic maps, RDF, DAML, OIL, OWL, TMCL, http://www.ontopia.net/topicmaps/materials/tmrdf.html.
36. L. Garshol: An RDF Schema for topic maps, http://psi.ontopia.net/rdf/.
37. L. Garshol, G. Moore: Topic Maps — Data Model, ISO/IEC JTC1/SC34 Information Technology, Document Description and Processing Languages, http://www.isotopicmaps.org/sam/sam-model/.
38. L. Miller: Inkling: RDF query using SquishQL, http://swordfish.rdfweb.org/rdfquery/.
39. L. Mugnaini: Mapping Topic Maps on Relational Databases, *XML Europe 2000*.
40. L. Sweeney: k -Anonymity: A Model for Protecting Privacy, *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10 (5), 2002; 557-570.

41. M. Arenas, A. Bertails, E. Prud'hommeaux, J. Sequeda: A Direct Mapping of Relational Data to RDF, W3C Recommendation 27 September 2012, http://www.w3.org/TR/rdb-direct-mapping/.
42. M. Biezunski, S. Newcomb: Topicmaps.net's Processing Model for XTM 1.0, version 1.0.2, A Processing Model for XML Topic Maps, http://www.topicmaps.net/pmtm4.htm.
43. M. Jobst: Preservation in Digital Cartography, Springer, 2011.
44. M. Lacher, S. Decker: On the Integration of Topic Maps and RDF Data, Extreme Markup Languages 2001, Proceedings.
45. M. Koparanova, T.Risch: Completing CAD Data Queries for Visualization, International Database Engineering and Applications Symposium (IDEAS 2002) , Edmonton, Alberta, Canada, July 17-19, 2002.
46. N-Triples, W3C RDF Core WG Internal Working Draft, http://www.w3.org/2001/sw/RDFCore/ntriples/.
47. N. Ogievetsky: XML Topic Maps through RDF glasses, Proceedings of Extreme Markup Languages, 2001.
48. National Science Board: Long-Lived Digital Data Collections: Enabling Research and Education in the 21st Century, NSB 05-40, http://www.nsf.gov/pubs/2005/nsb0540/nsb0540.pdf,2005.
49. OWLIM, www.ontotext.com/owlim.
50. O. Erling: Declaring RDF views of SQL Data, W3C Workshop on RDF Access to Relational Databases, 25-26 October, Cambridge, MA, USA, 2007.
51. P. Buneman, S. Khanna, K. Tajima, W. Tan: Archiving Scientific Data, *ACM Transactions on Database Systems*, 2004, Vol. 29, No. 1, pp. 2-42.
52. P. Ciancarini, R. Gentilucci, M. Pirruccio, V. Presutti, F. Vitali: Metadata on the Web: On the integration of RDF and Topic Maps, Proceedings of Extreme Markup Languages 2003.
53. P. Lyngbaek: OSQL: A Language for Object Databases ,Tech.Report, HP Labs, HPL-DTD-91-4, 1991.
54. J. C. Ramalho, M. Ferreira, L. Faria, R. Gastro: Relational Database Preservation through XML modeling. Extreme Markup Languages 2007, Montreal, Quebec, August 7-10, 2007.
55. R. Barta, L. M. Garshol: Topic Map Query Language Use Cases, 2003, Technical document 2003-12-04, http://www.isotopicmaps.org/tmql/use-cases.html.
56. R. Elmasri, S. B. Navathe: Fundamentals of Database Systems, 5th Edition, 2007.
57. Reference Model for an Open Archival Information System (OAIS), Recommended Practice CCSDS 650.0-M-2, Magenta Book, Consultative Committee for Space Data Systems, 2012.
58. Relational Databases to RDF (RDB2RDF), http://www.w3.org/2001/sw/wiki/RDB2RDF .
59. D. Shipman: The Functional Data Model and the Data Language DAPLEX. ACM Transactions on Database Systems , 6(1), 140-173, 1981.
60. S. Decker, S. Melnik, F. Van Harmelen, D. Fensel , M. Klein, J. Broekstra, M. Erdmann , I. Horrocks: The Semantic Web: The Roles of XML and RDF, IEEE Inrenet Computing, 2000, Vol. 4, Iss. 6.
61. S. Decker, D. Brickley, J. Saarela, J. Angele: A query and Inference Service for RDF, Query Languages Workshop, 1998.
62. T. Shepard, D. MacCarn: Universal Preservation Format - Background and Fundamentals. In Sixth DELOSWorkshop Preservation of Digital Information, Tomar, Portugal, 1998.

63. S. Das, S. Sundara, R. Cyganiak (ed.): R2RML: RDB to RDF Mapping Language, http://www.w3.org/2001/sw/rdb2rdf/r2rml/ .
64. Sesame, http://www.openrdf.org/.
65. SqirrelRDF, http://jena.sourceforge.net/SquirrelRDF/ .
66. S. Harris, A. Seaborne (ed.): SPARQL 1.1 Query Language, W3C Recommendation 21 March 2013, http://www.w3.org/TR/2013/REC-sparql11-query-20130321/.
67. S. Higgins: The DCC Curation Lifecycle Model, *International Journal of Digital Curation*, 2008, Vol. 3, No. 1, pp. 134-140.
68. Selection of Research Data; Guidelines for appraising and selecting research data; A report by DANS and 3TU Datacentrum, Netherlands, http://www.surf.nl/nl/themas/openonderzoek/cris/Documents/SURFshare_Collectioneren_Selection%20of%20Research%20Data_DANS_3TU_DEFtt.pdf .
69. S. Pepper, G. Moore (ed.): XML Topic Maps (XTM) 1.0, TopicMaps.Org Specification, http://www.topicmaps.org/xtm/.
70. S. Pepper, V. Presutti, L. Garshol, F. Vitali: Guidelines for RDF/Topic Maps Interoperability, http://www.ontopia.net/work/guidelines.html.
71. S. Pepper, F. Vitali, L. Garshol, M., N. Gessa, V. Presuti: A survey of RDF/Topic Maps Interoperability Proposals, W3C Working Group Note 10 February 2006, http://www.w3.org/TR/rdftm-survey/.
72. Swiss Federal Archives SFA Unit Innovation and Preservation. SIARD Format Description. Technical report, Federal Department of Home Affairs, FDHA, Berne, 2008.
73. Format description SIARD for preservation of relational databases, Berne, 15. October 2011, http://www.bar.admin.ch/dienstleistungen/00823/00825/index.html?lang=en&download=NHzLpZeg7t,lnp6I0NTU042l2Z6ln1ad1IZn4Z2qZpnO2Yuq2Z6gpJCDdIR8fmym162epYbg2c_JjKbNoKSn6A-- .
74. Topic Maps, http://www.topicmaps.org/.
75. T. Risch, V. Josifovski, T. Katchaounov: Functional Data Integration in a Distributed Mediator System in P.Gray, L.Kerschberg, P.King, and A.Poulovassilis (eds.): Functional Approach to Data Management - Modeling, Analyzing and Integrating Heterogeneous Data, Springer, ISBN 3-540-00375-4, 2004.
76. T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau: Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation 26 November 2008, http://www.w3.org/TR/REC-xml/.
77. U. M. Borghoff, P. Rödig, J. Scheffczyk, L. Schmitz: Long-Term Preservation of Digital Documents, Springer, 2010.
78. Uniform Resource Identifiers (URI): Generic Syntax, http://www.ietf.org/rfc/rfc2396.txt.
79. Virtuoso Universal Server, http://virtuoso.openlinksw.com/.
80. W. Litwin, T. Risch: Main Memory Oriented Optimization of OO Queries using Typed Datalog with Foreign Predicates'. *IEEE Transactions on Knowledge and Data Engineering*, vol. 4, No 6, 1992.
81. Qin Zhang: Wrapping Topic Maps in an Object-Relational Database System Uppsala Master's Theses in Computing Science 309, ISSN 1100-1836, 2007.
82. W3C, Semantic Web, http://www.w3.org/standards/semanticweb/.

Acta Universitatis Upsaliensis

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 1052

Editor: The Dean of the Faculty of Science and Technology

A doctoral dissertation from the Faculty of Science and Technology, Uppsala University, is usually a summary of a number of papers. A few copies of the complete dissertation are kept at major Swedish research libraries, while the summary alone is distributed internationally through the series Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology.

Distribution: publications.uu.se
urn:nbn:se:uu:diva-199573

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2013