Viewing and Querying Topic Maps in terms of RDF

Silvia Stefanova[1] and Tore Risch[1]

[1] Uppsala University, Department of Information Technology
Uppsala, Sweden
Silvia.Stefanova, Tore.Risch@it.uu.se

Abstract. Both Topic Maps and RDF are popular semantic web standards designed for machine processing of web documents. Since these representations were originally created for different purposes, they have conceptual differences in their data models, and therefore have different tools to parse, store, and query them. However, there are more tools to handle RDF data than those existing for Topic Maps. Our approach is to map Topic Maps to a view expressed in RDF and then query this view by the RDF query language SPARQL. To achieve this, a generic conceptual schema of Topic Maps is defined using a functional data model. Based on the conceptual schema of Topic Maps, an automatic mapping from Topic Maps to the RDF data model is developed. The mapping provides a general view of any Topic Map data in terms of RDF that can be queried using SPARQL. Query rewriting techniques based on partial evaluation enable realistic performance.

Keywords: Topic Map, RDF Schema view, SPARQL, query processing

1 Introduction

Topic Maps [19], [28] and RDF [23], [24] have obvious similarities but, since they have been originally created for different purposes, they conceptually differ in their data models. Topic Maps started in the 90's from the idea of managing indices to documents and was published as ISO standard in 2000 [5], [18], [30], while RDF came out from work on the Meta Content Framework [5] and became a W3C Recommendation year 1999 [29] as a general knowledge representation language. Despite the fact that both standards have the same main concepts, i.e. they represent facts about entities, they treat concepts in different ways.

In Topic Maps each entity, called a *topic*, is identified by a URI that represents either the subject of the topic itself or referring to resources that indicate what the subject of the topic is [8]. In RDF the entity is always identified by the subject's URI. Classification of entities is managed in Topic Maps by making a topic an instance of another topic, so any topic can classify any other topic. Furthermore, occurrences and associations in Topic Maps can also be instances of other topics. By contrast, classifications in the RDF-Schema (RDFS) model [25] are always uniformly specified as instances of RDF-Schema *classes*. Assigning predefined attributes in Topic Maps is made by built-in semantics, for instances giving names of the topics and defining

occurrences or participations in associations. RDFS provides classes and properties as basic meta-attributes and the user can define domain specific ontologies in terms of these. Thus RDF-Schema is a more basic knowledge representation language than Topic Maps, having fewer built-in concepts with extensibility to define any kind of general ontology, while Topic Maps is more specialized with more built-in concepts.

Despite the mentioned differences, Topic Maps and RDF technologies are often mentioned as two alternatives for the semantic web [5], [8], [14], [17]. Both of them have their own tools for creating, editing, wrapping, querying, etc. However, the existence and variety of RDF tools is bigger than those available for Topic Maps. Moreover, the applications of the RDF query languages to extract information from semantic web documents have gained certain popularity recently [3], [10], [14]. This has been our motivation to implement a system, *Semantic Web Abridged Topic Maps (SWATM)*, for viewing Topic Map data in terms of RDF Schema ontology. It is based on mapping a conceptual schema of Topic Maps to a view in terms of the RDF Schema data model over which SPARQL queries can be specified.

In SWATM, the *XTMImporter* [22] parses Topic Map data stored in XTM files [28]. The parsed data is translated into a data representation in terms of the Topic Map conceptual schema. A functional data model [26] that is straight-forward to map to RDF Schema is used for representing the conceptual schema in terms of functions and types. Based on the functional schema, corresponding RDFS classes are defined for each type and RDF properties for each function. Since the conceptual schema is generic, a generic RDF Schema view that represents any imported Topic Map is automatically generated. The Topic Map view can be queried with SPARQL [27]. Queries can search both Topic Map meta-objects and the content of any imported XTM file.

2 Architecture of the SWATM System

The architecture of the SWATM system is depicted in Fig. 1. The core of the system is the *SWATM database* that internally represents imported *Topic Map data*. General Topic Map data is described by the functional *Topic Map conceptual schema.* The *TM view* is a system generated generic RDF view of Topic Maps in terms of the RDF-Schema data model. It is generated by the *RDF view generator* based on *Mapping rules* between basic Topic Map concepts to the corresponding RDF-Schema concepts in terms of the Topic Map conceptual schema. The *XTM importer* parses queried XTM files and populates the SWATM database.

The TM view is defined in terms of a *TM schema* and a *TM data* RDF view. The TM schema view represents the elements of the Topic Map data model in terms of RDFS, while the TM data view represents imported Topic Map data.

A SPARQL query is specified in terms of the TM view. It is first parsed into a Datalog dialect [11] by the *SPARQL parser* [2]. The *query optimizer* rewrites and optimizes the generated Datalog query to produce an execution plan. The *query executer* interprets the execution plan. However, queries over the TM view definitions are often very complex to process, because the TM view definitions are complicated with many disjunctions and hence the intermediate expressions become large.

Therefore, as in the SWARD system [20], the query optimizer performs *partial evaluation* [9], [16] i.e. compile time evaluation of query fragments to reduce the size of the query. This substantially improves the query processing time and is often required for being able to optimize large queries in reasonable time.

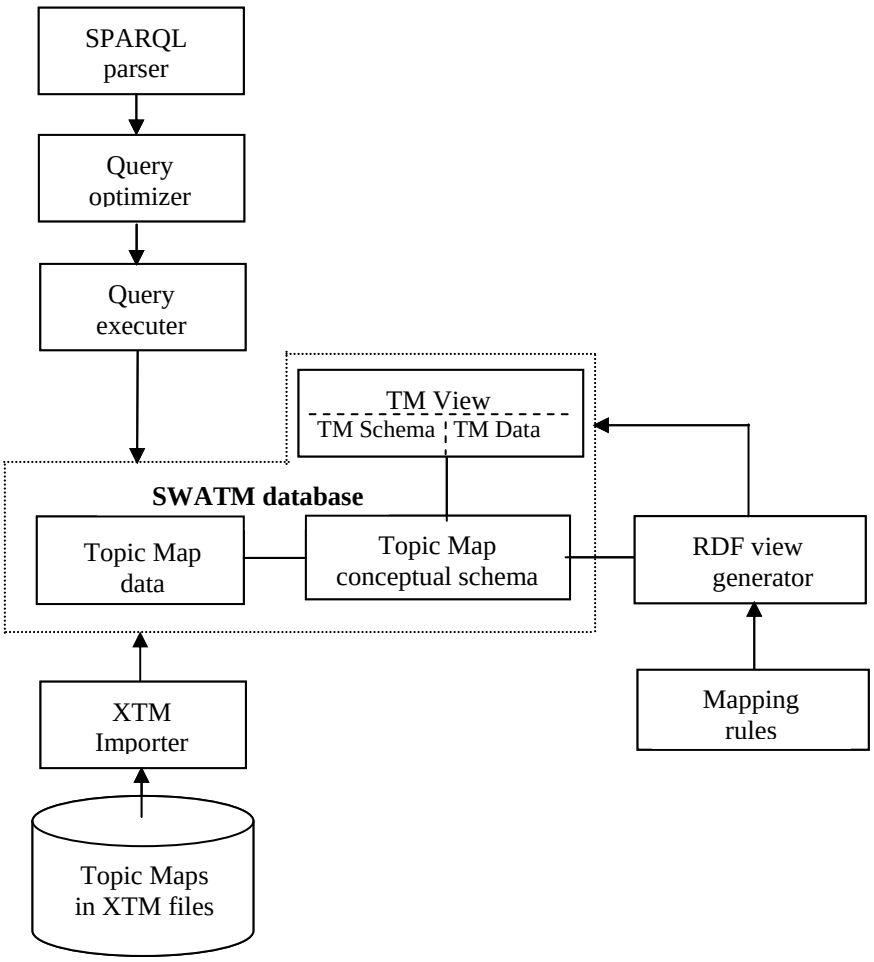

Fig.1. SWATM Architecture

3 Topic Map Conceptual Schema

The mapping between the Topic Map data model and the RDFS data model is based on defining a conceptual schema of the Topic Map data model in terms of a functional data model [26], which is illustrated in Fig. 2. The conceptual schema extends the Topic Map data model definition in [31], [32] to enable 1:1 mappings with both Topic Map and RDF Schema representations. The rectangles present the types and the ovals attributes. The arrows between the types depict relationships between types representing Topic Map concepts. Both attributes and relationships are represented as *functions* in the functional data model. The arrows' directions reflect the functional dependencies between the types.

To enable 1:1 mapping to RDF-Schema all function and type names in the conceptual schema are unique. We have modified the names of Topic Map elements in the XTM 1.0 standard [28] to remove ambiguous names. This allows the RDF view generator to generate automatically unique URIs of RDFS classes and RDF properties in the process of generating the TM schema view from the conceptual schema.

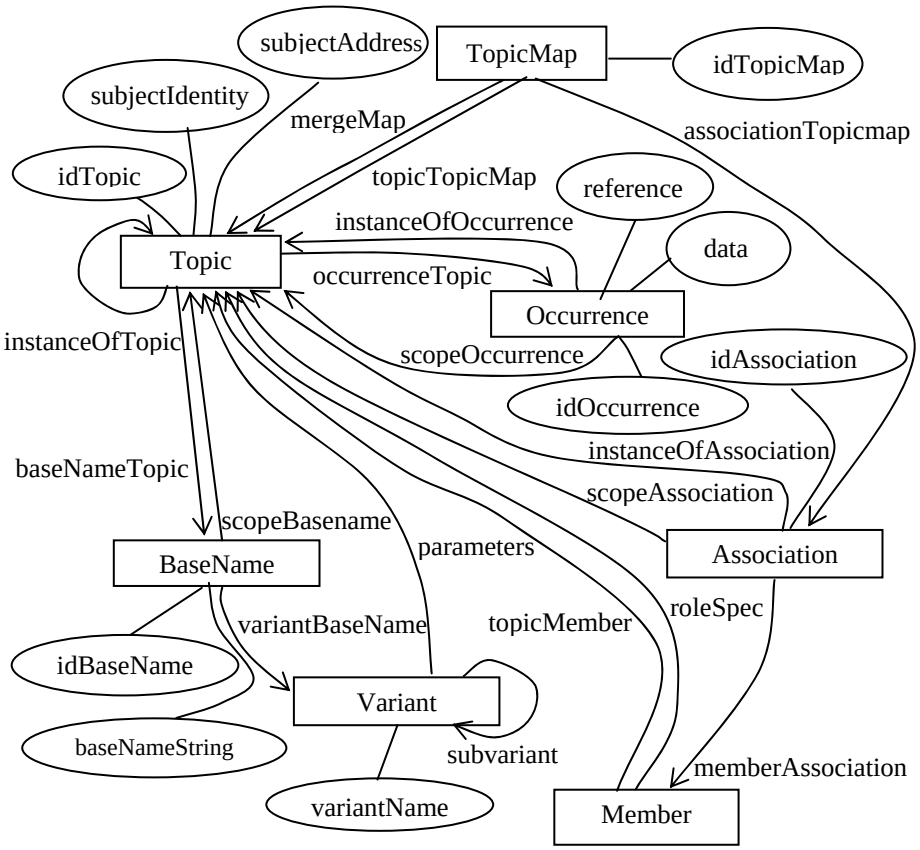

Fig.2. Functional conceptual schema of Topic Map

4 Definition of the Topic Map View in terms of RDF

The notation <***s**ubject*, ***p**roperty*, ***v**alue*> is used to represent RDF triples [23]. Using ObjectLog [11], which is a Datalog dialect having disjunctions and object representation, SWATM defines a view of Topic Map data as RDF triples. The meta-model of RDFS is used for defining mapping rules between the Topic Map data in some source and the corresponding RDF triples presented by the view. Both triples inferred from the Topic Map conceptual schema definition itself (TM Schema view) and triples inferred from imported XTM data files (TM Data view) are part of the TM view.

The generated TM view definition, *TMTriples*, over a Topic Map data source *src* is defined as a disjunction of the TM schema and TM data views, *TMSchemaTriples* and *TMDataTriples*, respectively:

```
TMTriples(src, s, p, v):-
TMSchemaTriples(s, p, v)   OR
TMDataTriples(src, s, p, v)
```

The TM schema triple view *TMSchemaTriples* maps the meta-information in Fig. 2 into an RDFS view for Topic Maps. The RDFS view is expressed in terms of an automatically generated RDFS ontology corresponding to the conceptual schema. The data triple view, *TMDataTriples*, represent the data in each imported XTM file. The variable *src* holds the URL of the imported Topic Map data, e.g. *http://www.isotopicmaps.org/tmql/uc-literature.xtm*. *TMSchemaTriples* are the same for all data sources, because they are inferred from the generic definition of the conceptual schema, while *TMDataTriples* represent a particular Topic Map database in a source.

The following namespaces are used in the TM view definitions:

- *rdf:* is the namespace for RDF, *http://www.w3.org/1999/02/22-rdf-syntax-ns/*
- *rdfs:* is the namespace for RDFS, *http://www.w3.org/2000/01/rdf-schema*
- *swatm:* is the namespace used to represent the TM schema, e.g. *http://udbl2.it.uu.se/swatm*

4.1 TM Schema View Definition

The mapping rules used in the definitions below were inspired by Garshol's RDFS for Topic Maps [7]. Each Topic Map entity type (rectangles in Fig. 2) corresponds to one RDFS *class*, while its attributes (the ovals in Fig. 2) and functional relationships (the arrows) correspond to a set of RDF *properties.* The argument of a function specifies the RDFS *domain* of a property. The result of a function specifies the RDFS *range* of the property. In the conceptual schema functions represent both relationships and attributes. In case a function represents a relationship the range is the RDFS class of the destination of the arrow in Fig. 2; in case it represents an attribute the range is a literal. The corresponding TM schema triples are defined based on these mapping rules.

The TM schema triples are defined as a union between i) triples representing RDFS classes corresponding to the types in the conceptual schema, *TMClassTriples*, and ii) triples representing RDF properties, *TMPropertyTriples*:

```
TMSchemaTriples(s, p, v):-
TMClassTriples(s, p, v)    OR
TMPropertyTriples(s, p, v)
```

The TM schema class triples are defined as:

```
TMClassTriples(s, p, v):-
SurrogateTypeMap(t, s)          AND
p = 'rdf:type'                  AND
v = 'rdfs:Class'
```

Each conceptual schema entity type is represented as an instance of a surrogate type *t*. The predicate *SurrogateTypeMap* maps between a type *t* and the corresponding RDF subject *s* in case t is a surrogate type:

```
SurrogateTypeMap(t, s) :-
isSurrogate(t)                  AND
s = concat('swatm:',name(t))
```

The name of the RDFS class is computed by concatenating the namespace '*swatm:*' with the name of *t*. An example of a *TMClassTriple* is thus:

```
<swatm:TOPIC, rdf:type, rdf:Class>.
```

Here *swatm:TOPIC* is the URI corresponding to the *Topic* entity type.

The Topic Map property triples are defined as:

```
TMPropertyTriples(s, p, v):-
TMFunctions(f)                              AND
(FunctionPropertyTriple(f, s, p, v)  OR
 FunctionDomainTriple(f, s, p, v)    OR
 FunctionRangeTriple(f, s, p, v))
```

The predicate *TMFunction*s defines the conceptual schema functions *f* representing Topic Map attributes and relationships.

FunctionPropertyTriple(f, s, p, v) declares that the attribute or relationship represented in the conceptual schema by the function *f* is mapped to an RDF property, e.g. the triple:

```
FunctionPropertyTriple(f, s, p, v):-
functionMap(f, s)                       AND
p = 'rdf:type'                          AND
v = 'rdf:Property'
```

functionMap(f, s) gets the URI for *f* by concatenation the SWATM name space and the name of the function, i.e

```
functionMap(f, s) :-
s = concat('swatm:', name(f)
```

An example of a *FunctionPropertyTriple is:*

```
<swatm:SUBJECTIDENTITY,rdf:type,rdf:Property>
```

Here *swatm:SUBJECTIDENTITY* is the URI corresponding to the *subjectIdentity* attribute of the type *Topic* in the conceptual schema.

Analogously, the relationship represented by the function *baseNameTopic* is mapped to the triple:

```
<swatm:BASENAMETOPIC,rdf:type,rdf:Property>
```

FunctionDomainTriple defines the domain of the RDF property and *FunctionRangeTriple* its range.

FunctionDomainTriple has the definition:

```
FunctionDomainTriple(f, s, p, v):-
functionMap(f, s)                AND
p = 'rdfs:domain'                AND
argtype(f, t)                    AND
SurrogateTypeMap(t, v)
```

The argument of a function corresponds to the RDFS *domain* of the property. All functions in the conceptual schema are binary where the argument is always a surrogate type. The built-in predicate *argtype(f, t)* returns the argument type *t* of the function *f*.

For example, the domain triples for the above two RDF properties are:

```
<swatm:SUBJECTIDENTITY,rdf:domain,swatm:TOPIC>
<swatm:BASENAMETOPIC,rdf:domain,swatm:TOPIC>
```

The range of a property is defined as:

```
FunctionRangeTriple(f, s, p, v):-
functionMap(f, s)                AND
p = 'rdfs:range'                 AND
restype(f, t)                    AND
(SurrogateTypeMap(t, v) OR
 LiteralTypeMap(t, v))
```

The result type *t* of a function *f* is defined by the built-in predicate *restype(f, t).* In case the function *f* represents a relationship its result is a surrogate type and the range is the RDFS class of the result, defined by *SurrogateTypeMap* above. In case it represents an attribute the range is a literal defined by *LiteralTypeMap*, with definition:

```
LiteralTypeMap(t, v) :- isLiteral(t)      AND
v = 'rdfs:Literal'
```

The range triples of the two earlier defined RDF properties are:

```
<swatm:SUBJECTIDENTITY,rdf:range,rdfs:Literal>
<swatm:BASENAMETOPIC,rdf:range,swatm:BASENAME>
```

This concludes the TM schema view definition in terms of RDFS. Since it is independent of Topic Map data it is materialized by the system once and for all. The materialized view contains 91 triples.

4.2 TM Data view definition

The TM data view, *TMDataTriples* is defined as a union of three sub-views: i) the *class membership view*, *TMInstanceOf*, defining classes of created objects, ii) the *attribute view*, *TMAttrView*, defining object attribute values, and iii) the *relationship view*, *TMRelationshipView*, defining relationships between objects:

```
TMDataTriples(src, s, p, v) :-
TMInstanceOf(src, s, p, v)          OR
TMAttrView(src, s, p, v)            OR
TMRelationshipView(src, s, p, v)
```

The XTM importer creates internal surrogate objects for each Topic Map element imported from an XTM file. Depending on what kinds of Topic Map elements are read, objects of different types according to the conceptual schema are created. The type of the created object furthermore will determine the corresponding RDFS class in the TM data view.

Unique URIs are associated with each created object by concatenating the URL of the XTM file with an identifier number, e.g. *http://www.isotopicmaps.org/tmql/uc-literature.xtm#4*. To generate unique URIs, the enumeration is separate for each XTM file. This makes possible in SWATM to load and query several Topic Maps, originating in different XTM files. The same URIs are generated if the same file is imported at different times or locations, which provides repeatable query results.

The stored system table *idTMO(o, src, num)* maps between an internal object *o* and the corresponding Topic Map element read from an XTM file with URL *src*. The internal identifier number *num* is assigned by the XTM importer. Thus *o* is primary key and *scr* + *num* composite secondary key. The Topic Map importer populates *idTMO* when objects are created while parsing XTM files.

The importer furthermore populates the functions representing attributes and relationships in the conceptual schema. The TM data view is defined in terms of these functions, as will be shown.

The *class membership view*, representing the RDFS classes to which imported objects belong, is defined by *TMInstanceof:*

```
TMinstanceof (src, o, s, p, v):-
TMURI(o, src, s)                AND
p = 'rdf:type'                  AND
typeOf(o, tp)                   AND
SurrogateTypeMap(tp, v)
```

TMURI(o, src, u) maps between the internal object o and its corresponding URI by concatenation:

```
TMURI(o, src, u) :-
idTMO(o, src, num)       AND
u = concat(src,num)
```

typeOf(o, tp) associates a type *tp* with an object *o*. Then the URI *v* of the RDFS class corresponding to the object *o* is determined by means of *SurrogateTypeMap(tp, v)*, applied on the type *tp*.

An example of a triple in the class membership view is:

```
<http://www.isotopicmaps.org/tmql/uc-literature.xtm#65,
rdf:type, swatm:TOPIC>
```

It states that the object identified by *http://www.isotopicmaps.org/tmql/uc-literature.xtm#65* has an RDFS class 'swatm:TOPIC', i.e. it is an instance of that class.

The *attribute view*, *TMAttrView*, defines RDF properties for each attribute in the conceptual schema. It is defined as a union of all *specific* attribute views, one for each attributes. SWATM generates a specific attribute view for *attr* as follows:

```
TMattr (src, s, p, v):-
TMURI(o, src, s)          AND
p = 'swatm:attr'          AND
attr(o, v)
```

A specific attribute view TM*attr* defines the RDF triples of the attribute *attr* for an XTM source *src*. For example, the following defines the specific attribute view for *basenameString* in the conceptual schema:

```
TMBASENAMESTRING(src, s, p, v):-
TMURI(o, src, s)                AND
p = 'swatm:BASENAMESTRING'      AND
BASENAMESTRING(o, v)
```

After creating *o* and populating *idTMO*, the importer sets the attribute *attr* (e.g *BASENAMESTRING*) mapping between *o* and *attr*. As before the URI corresponding to *o* is determined by *TMURI(o, src, s)*.

An example of an attribute view triples is:

```
<http://www.isotopicmaps.org/tmql/uc-literature.xtm#66,
swatm:BASENAMESTRING, 'John Smith'>,
```

where *swatm:BASENAMESTRING* is the URI corresponding to the *'baseNameString'* attribute of the entity type *'baseName'* in the conceptual schema, while *http://www.isotopicmaps.org/tmql/uc-literature.xtm#66* is an instance of the RDFS class *swatm:BASENAME* corresponding to the schema entity type *'Basename'*.

The *relationship view*, *TMRelationshipView*, defines RDF properties for all relationships in the conceptual schema. It is defined as the union of all *specific relationship views*, one for each relationship. For each relationship *rel* in the conceptual schema SWATM generates a corresponding specific relationship view:

```
TMrel (src, s, p, v):-
TMURI(o1, src, s)              AND
p = 'swatm: rel'               AND
rel(o1, o2)                    AND
TMURI(o2, src, v)
```

The triples of a specific relationship view are inferred from the internal stored table named *rel* that determines the relationship between two internal objects *o1* and *o2*. The table is populated by the XTM importer.

An example of a triple in a relationship view is:

```
<http://www.isotopicmaps.org/tmql/uc-literature.xtm#12,
swatm:SCOPEOCCURRENCE,
http://www.isotopicmaps.org/tmql/uc-literature.xtm#14>
```

where:

- *swatm:SCOPEOCCURRENCE* is the URI of the RDF property corresponding to the relationship '*scopeOccurrence*' between the entity types '*Occurrence'* and '*Topic'* in Fig. 2.
- *http://www.isotopicmaps.org/tmql/uc-literature.xtm#12* is the URI representing an instance of the RDFS class corresponding to the entity type *'Occurrence'*.
- *http://www.isotopicmaps.org/tmql/uc-literature.xtm#14* is the URI representing an instance of the RDFS class corresponding to the entity type *'Topic'*.

5 Queries to RDF Views of Topic Map

The TM view can be queried using SPARQL. The FROM clause of a SPARQL query is the URL of the XTM file being queried. Queries retrieving data triples return imported data while queries retrieving only schema triples are independent of imported data. The XTM file must be accessed by the XTM importer and RDF view generator (Fig. 1) before it can be queried, using the command:

```
importTopicMap(Charstring URL);
```

e.g.

```
importTopicMap(
```

```
"http://www.isotopicmaps.org/tmql/uc-literature.xtm");
```

The URL is used in FROM clauses of SPARQL queries.

The examples below illustrate SPARQL queries to a Topic Map view over XTM data. The first example query Q1 is a very simple query containing two disjunctions. It searches a large XTM file *http://user.it.uu.se/~udbl/software/swatm/mondial.xtm* of size 10.5 MB, which contains large amounts of data, i.e. 69800 Topic Map objects. The intent of Q1 is to measure the query processing time of SPARQL queries to a large XTM file. The second example query Q2 is analogous to a query in Topic Map Query Language Use Cases [21], here expressed as a SPARQL query in terms of the TM view. It is an example of a complex query having nine disjunctions. The amount of data Q2 is searching is small (only 113 Topic Map objects). This data is used as a test data in [21] and resides in the file *http://www.isotopicmaps.org/tmql/uc-literature.xtm*. Since the example query Q2 is very complex it is a challenge to perform the query processing itself. The third example query Q3 is a SPARQL query that references only the TM schema and therefore needs no FROM clause.

5.1 Example Query Q1

This query retrieves the occurrences of the topics having the name 'Nottinghamshire'.

```
SELECT ?tm2
FROM
<http://user.it.uu.se/~udbl/software/swatm/mondial.xtm>
WHERE
{ ?tm0 <swatm:BASENAMETOPIC>       ?tm1.
  ?tm1 <swatm:BASENAMESTRING>       'Nottinghamshire'.
  ?tm0 <swatm:OCCURRENCETOPIC>     ?tm2 }.
```

The result of the query is:

```
{"http://user.it.uu.se/~udbl/software/swatm/mondial.xtm
#2671"}
{"http://user.it.uu.se/~udbl/software/swatm/mondial.xtm
#2670"}.
```

5.2 Example Query Q2

The following is an example of a large SPARQL query [21] that retrieves the 'sort' names of all authors. Several Topic Map concepts have to be traversed in order to answer the query.

```
SELECT DISTINCT ?bnt1
FROM <http://www.isotopicmaps.org/tmql/uc-
literature.xtm>
WHERE
{ ?ass <swatm:INSTANCEOFASSOCIATION>    ?t2.
  ?t2  <swatm:IDTOPIC>                  'is-author-of'.
  ?ass <swatm:MEMBERASSOCIATION>        ?ma1.
  ?ma1 <swatm:ROLESPEC>                 ?t3.
  ?t3  <swatm:IDTOPIC>                  'author'.
  ?ma1 <swatm:TOPICMEMBER>              ?t1.
```

```
?t1  <swatm:BASENAMETOPIC>          ?bn.
?bn  <swatm:BASENAMESTRING>         ?bnt1.
?bn  <swatm:SCOPEBASENAME>          ?t4.
?t4  <swatm:IDTOPIC>                'sort' }.
```

The result of the query is the following tuples:

```
{"Pepper, Steve"}
{"Newcomb, Steve"}
{"Rath, Holger"}
{"Biezunski, Michel"},
```

which is the expected result given in [21].

5.3 Example Query Q3

This query retrieves all the RDF properties with a range rdfs:Literal of the RDFS class swatm:TOPIC in the TM schema except the property swatm:IDTOPIC:

```
SELECT ?tp
WHERE
{?tp      <rdf:type>              <rdf:Property>.
 ?tp      <rdfs:domain>           <swatm:TOPIC>.
 ?tp      <rdfs:range>            <rdfs:Literal>.
FILTER  (?tp != <swatm:IDTOPIC>).}
```

The result of the query is the following tuple:

```
{"swatm:SUBJECTIDENTITY"}
{"swatm:SUBJECTADDRESS"}
```

All query examples can be run on http://udblserv2.it.uu.se/semma.php.

5.4 Measurements

We measured the processing and execution times for the three queries. As shown in Section 4, the TM view is defined in terms of many disjunctive Datalog rules (views). Therefore the number of predicates in internal query expressions becomes very large during query processing after normalization.

As explained in Section 4.1, the TM Schema view never changes and contains only 91 triples. It is therefore materialized in SWATM once and for all and poses no problem for query processing. However, the TM Data view contains many triples and varies since it is defined over the contents of the particular imported XTM files. It can therefore not be materialized. Furthermore, the TM Data view is defined in terms of many disjunctions as a union of many attribute and relationship views (Section 4.2). For this reason, queries over the TM Data view generate very large internal expressions, which makes query processing slow.

The technique of partial evaluation [9], [16], [20] is used to significantly reduce the sizes of internal query expressions, by evaluating at query processing time those predicates used to define the TM Data view that return at most one triple. Predicates

returning more than one triple are not partially evaluated since that would not reduce the query.

To see the impact of query reduction by partial evaluation we measured the size of the generated execution plans in terms of number of operators. The measurements were made on a Dell OPTIPLEX GX270 with 2.40 GHz CPU, 512 MB main memory and Windows XP Professional OS. The results from the measurements are presented in Table 1.

Table1. Measurement results

Measures		With partial evaluation	Without partial evaluation
Q1	Optimization time	0.28	0.44
	Execution time	0.000062	0.000118
	Number of operators	23	104
Q 2	Optimization time	3.2	318
	Execution time	0.00078	0.074
	Number of operators	75	43009
Q 3	Optimization time	0.015	0.015
	Execution time	0.00008	0.00008
	Number of operators	4	4

As it can be seen from the measurement results, using partial evaluation in the large query Q2 contributes to a substantial (around 100 times) decrease of both query optimization and query execution times. The reason is that partial evaluation leads to enormous decrease of the number of operators in the query execution plan, from 43009 to 75. The query optimization time is improved because of smaller size query expressions to process. The query execution time improves because the query optimizer can produce better execution plans for smaller queries.

By contrast, query Q1 is rather simple, and hence partial evaluation has less impact on both query processing and optimization time (factor 2).

Since the third query accesses only the materialized schema data it executes in 0,00008 sec, independent of partial evaluation.

The full effect of partial evaluation remains to be further investigated.

6 Related Work

The existing approaches for transformation between Topic Map and RDF can be divided into two main groups, called *object mappings* and *semantic mappings* in [17]. Semantic mappings are based on finding equivalences between a Topic Map schema and the corresponding RDF Schema, while object mapping is based on representing the Topic Map data model in terms of the RDF Schema model [13].

The most extensive proposal for semantic mapping is described in [5], [15], [30]. The contribution [30] is in fact guidelines for interoperability between the two standards Topic Maps and RDF based on the semantic approach. It is basically a complete proposal for semantic mapping with some limitations of its use in terms of

non-deterministic results and unsupported constructs. The semantic mapping can give good results from a "naturalness" and flexibility point of view [5], [13] but it is not always possible due to lack of semantic equivalences [3]. It is not either of general usefulness since it requires an application-dependent approach.

On the other hand the object mapping approach provides a generic mapping from Topic Map to RDF, which is always possible. This has been our motivation to implement an object mapping in SWATM. Work on object mapping the Topic Maps data model to RDF was done by [7], [10], [14]. The proposal [7] is based on the ISO/IEC model of Topic Maps, i.e. *TMDM* [6]. The so called items in TMDM become RDFS classes and the properties of the items become RDF properties. Unlike SWATM, [7] is incomplete because there are no definitions of the RDFS ranges and domains of the properties and no description of how Topic Map data is mapped to RDF. The authors in [10] use the *Processing Model for Topic Maps, PMTM4* [1], which is very simple model and is not considered as a complete model for Topic Maps [17]. This disadvantage has been overcome in [14] where a Topic Maps model is defined partly in terms of PMTM4 and completed with extra XTM terms. The proposal [14] is fairly complete but very complicated and the translation from the Topic Map data model to an RDF-Schema is non-reversible [17]. For example, it requires seven statements to represent the information content that would be modeled using one statement in RDF [17]. By contrast, SWATM is based on a canonical and yet simple conceptual schema representation that maps 1:1 to both Topic Maps and the corresponding RDF Schema ontology representation of Topic Maps. The mapping rules from the conceptual schema to RDF Schema 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 provide the general RDFS based TM view over any Topic Map data imported to SWATM that can be queried with SPARQL.

In previous proposals [10], [14] it has been illustrated that a Topic Map transformed to RDF can be queried using F-Logic syntax [4], [10] or the RDF query language SquishQL [12]. We support querying of the Topic Map 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. We measured that partial evaluation [9], [16] applied on query processing [20] significantly improves performance.

7 Summary

The SWATM system provides interoperability between the two Semantic web standards Topic Map and RDF. The following results were presented:

- A functional conceptual schema was defined for the Topic Map data model.
- Generic 1:1 mappings from the conceptual schema into the RDF Schema model were defined. The mappings are defined as an automatically generated RDF view, the *TM view*, in terms of disjunctive Datalog rules. The TM View consists of two parts, the *TM Schema view* and the *TM Data view*. The TM Schema view describes the Topic Map conceptual schema as RDF triples,

while the TM Data view describes data represented by the Topic Map conceptual schema as RDF.

- The Datalog based TM view definition enables processing of SPARQL queries to any Topic Map XTM file.
- Since the TM view contains many disjunctions, query processing becomes slow. Preliminary results show substantial performance improvements by using partial evaluation techniques. Future work includes more detailed investigations of the impact of partial evaluation.

SWATM is the first system to enable SPARQL queries over a general RDF Schema view of Topic Map data.

Acknowledgments. This work was partially funded by the EU project Advanced eGovernment Information Service Bus, FP6-IST-2004-26727.

References

1. Biezunski, M., Newcomb, S.: Topicmaps.net's Processing Model for XTM 1.0, version 1.0.1, A Processing Model for XML Topic Maps, http://www.topicmaps.net/pmtm4.htm
2. Cao, Yu.: Processing SparQL Queries in an Object Oriented Mediator, Uppsala Master's Theses in Computing Science, ISSN 1100-1836 (2007)
3. Ciancarini, P., Gentilucci, R., Pirruccio, M., Presutti, V., Vitali, F.: Metadata on the Web: On the integration of RDF and Topic Maps, http://www.idealliance.org/papers/extreme03/html/2003/Presutti01/EML2003Presutti01.html
4. Decker, S., Brickley, D., Saarela, J., Angele, J.: A query and Inference Service for RDF. In: QL '98 - Query Languages Workshop, (1998)
5. Garshol, L.: Living with Topic Maps and RDF, Topic maps, RDF, DAML, OIL, OWL, TMCL, http://www.ontopia.net/topicmaps/materials/tmrdf.html.
6. Garshol, L., More, G.: ISO/IEC 13250: Topic Maps — Data Model, http://www.isotopicmaps.org/sam/sam-model
7. Garshol, L.: RDF Schema for topic maps, http://psi.ontopia.net/rdf/
8. Garshol, L.: Topic Maps, RDF, DAML, OIL, A Comparison, http://www.ontopia.net/topicmaps/materials/tmrdfoildaml.html
9. Jones, N. D.: An Introduction to Partial Evaluation, ACM Computing Surveys, 28(3), (1996)
10. Lacher, M. S., Decker, S.: On the Integration of Topic Maps and RDF Data, http://www.idealliance.org/papers/extreme03/html/2001/Lacher01/EML2001Lacher01-toc.html .
11. Litwin, W., Risch, T.: Main Memory Oriented Optimization of OO Queries using Typed Datalog with Foreign Predicates. In: IEEE Transactions on Knowledge and Data Engineering, vol. 4, No 6 (1992)
12. Miller, L.: Inkling: RDF query using SquishQL, http://swordfish.rdfweb.org/rdfquery/
13. Moore, G.: RDF and Topic Maps: An exercise in convergence, http://xml.coverpages.org/moore-topicmapsrdf200105.pdf
14. Ogievetsky, N.: XML Topic Maps through RDF glasses, http://www.cogx.com/?si=urn:cogx:resource:rdfglasses
15. Ontopia, A TM-to-RDF mapping, Available: http://psi.ontopia.net/tm2rdf/
16. Partial Evaluation, http://partial-eval.org/

17. Pepper, S., Vitali, F., Garshol, L. M., Gessa, N., Presuti, V.: A survey of Topic Map's interoperability proposals. W3C Working Group Note 10 February 2006, http://www.w3.org/TR/rdftm-survey/
18. Pepper, S.: Euler, Topic Maps and Evolution, http://www.ontopia.net/topicmaps/materials/euler.pdf
19. Pepper, S.: The TAO of Topic Maps, http://www.ontopia.net/topicmaps/materials/tao.html
20. Petrini, J., Risch, T.: SWARD: Semantic Web Abridged Relational Databases, 6th International Workshop on Web Semantics, WEBS 2007, Regensburg, Germany (2007).
21. Garshol, L. M., Barta, R.: Topic Map Query Language Use Cases (editor's draft), http://www.isotopicmaps.org/tmql/use-cases.html
22. Qin, Z.: Wrapping Topic Maps in an Object-Relational Database System, Uppsala Master's Theses in Computing Science 309, ISSN 1100-1836 (2007)
23. RDF Primer. W3C Recommendation, http://www.w3.org/TR/rdf-primer/
24. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, http://www.w3.org/TR/rdf-concepts/
25. RDF Vocabulary Description Language 1.0: RDF Schema, http://www.w3.org/TR/rdf-schema/
26. Risch, T., Josifovski, V., Katchaounov, T.: 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, (2003)
27. SPARQL Query Language for RDF, W3C Recommendation, 15 January 2008, http://www.w3.org/TR/rdf-sparql-query/
28. XML Topic Maps (XTM) 1.0. TopicMaps.Org.Specification, http://www.topicmaps.org/xtm/
29. W3C Issues Recommendation for Resource Description Framework, http://www.w3.org/Press/1999/RDF-REC
30. Guidelines for RDF/Topic Maps Interoperability. W3C Editor's Draft 30 June 2006, http://www.ontopia.net/work/guidelines.html
31. XML Topic Map Data Type Definition, http://www.topicmaps.org/xtm/1.0/xtm1.dtd
32. Mugnaini, L.: Mapping Topic Maps on Relational Databses, http://www.geocities.com/xtopicmaps/mapping_xtm_on_databases.html