USING MPQF FOR QUERYING MPEG-7 RDF DESCRIPTIONS

Mohammed A. Al-Zoube

Department of Computer Graphics and Animation
Princess Sumaya University for Technology, Jordan
mzoube@psut.edu.jo

ABSTRACT

MPEG-7 and MPEG Query Format (MPQF) are two related standards developed by the Moving Picture Experts Group (MPEG) to address the multimedia retrieval problem. MPEG-7 aims to provide standardized comprehensive set of tools for description of audiovisual content, while MPQF aims to provide a standardized interface which unifies the retrieval over different multimedia repositories. However, both standards have some limitations. On one hand, MPEG-7 is based on XML Schema, therefore, does not have formal semantics, which makes difficult to be managed by computers. Machine understandable metadata forms the main prerequisite for the intelligent services envisaged in a Web, which going beyond mere data exchange and provides for effective content access, sharing and reuse. Consequently, there have been several attempts to move MPEG-7 to the Semantic Web to provide it with formal semantics. On the other hand, MPQF is limited to querying XML-based multimedia metadata. Therefore if MPQF is intended to support queries on non-XML multimedia metadata, mapping of MPQF to the specific database query language is necessary. In this paper MPQF is used to query semantic MPEG-7 RDF descriptions. A set of rules are proposed for converting MPQF query into SPARQL query without introducing new construct to MPQF. These rules are mainly based on the new features added in SPARQL1.1.

KEYWORDS

Multimedia Retrieval, Semantic MPEG-7, MPQF, SPARQL, RDF

1. INTRODUCTION

Due to the huge amount of digitally stored multimedia contents multimedia retrieval is gaining increasing amount of research. Efficient multimedia retrieval requires an extensive annotating the media content with multimedia metadata, and formalizing the user information request in the form of a query expressed in terms of the available metadata model. Metadata annotations provide information about the media content at different levels, from low-level features and management information, to semantic-level descriptions. MPEG-7 is the main multimedia metadata framework created to date which provides a comprehensive set of standardized tools for the description of audiovisual content at multiple granularities. It addresses a variety of dimensions that range from structural and low level features that can often be automatically extracted from media types, to aspects related to navigation, creators, content organization, as well as user preferences and usage [1]. MPEG-7 metadata descriptions support some degree of interpretation of the information meaning and can describe semantic concepts associated with a multimedia content. Thus, the extensive description features specialized in multimedia and the structured semantic descriptions in MPEG-7 promise efficient retrieval of metadata information and a good match for semantic user queries.

However, since MPEG-7 is an XML-based metadata standard that is defined in terms of an XML schema, there are no formal semantics assigned to the description elements, which has led to

DOI : 10.5121/ijma.2012.4303

design decisions that leave the annotations conceptually ambiguous [2]. For example, different semantic concepts like frame, shot or video cannot be distinguished based on the provided XML schema. Thus, uncertainty can appear because of the flexibility in structuring the descriptions. Although MPEG-7 metadata documents can contain information about semantic concepts perceivable in a multimedia resource, it is not compatible with semantic web technologies and cannot be combined with these concepts defined in domain-specific ontology because of it is not open to standards that represent knowledge and make use of existing controlled vocabularies for describing the subject matter. Moreover, the meaning of elements described in the standard specification is intended for human readability and prevent direct machine processing of semantic content descriptions.

To take advantage of MPEG-7 as comprehensive multimedia metadata framework and to enhance the semantic expressiveness and to support it with formal semantics, it is desirable to have the MPEG-7 XML descriptions expressed in Resource Description Framework (RDF) [3] to express MPEG-7 metadata terms with ontology [4, 5, 6, 7]. The basic idea of RDF is to decompose knowledge into triples, where each triple comprises a subject, a predicate, and an object. An RDF instance represents a labeled directed graph, consisting of vertices, which represent *subjects* or *objects*, and labeled edges, which represent *predicates* (semantic relations between *subjects* and *objects*). A subset of RDF triples from an RDF graph which can be used separately, keeping a consistent RDF model. Moreover, the formal semantics for RDF make RDF documents machine-processable and allow applying reasoning techniques, which is an advantage over plain XML-based documents [8, 9, 10, 11]. Named graphs are an extension of the RDF specification which allows for the expression of meta information about graphs and the relationships between them [12]. RDF by itself only provides means to represent the structure and semantics of one single graph and does not include means that allow for identifying or referring to a set of triples defined in another graph. Naming RDF graphs with URIs makes them to be uniquely identified, and therefore, a single RDF document can host multiple graphs where each graph is identified by its URI.

RDF is not specialized in multimedia; however, it can be used for modeling multimedia metadata thanks to its advantages with respect to other models. There are formal semantics for RDF, and therefore, reasoning techniques can be applied so that semantic knowledge about an application domain can be utilized for the computation of a query result [13]. Another possible benefit is to enable deriving high-level concepts from low-level content-based metadata descriptions which is known as the semantic gap problem. Moreover, expressing data in RDF is one of the principles to be considered when making data available as Linked Data on the Web [14]. Furthermore, when the meaning metadata terms is available using ontologic vocabulary, the semantic knowledge about multimedia metadata terms and their relationships can be utilized for the computation of query results which promises to improve the retrieval of multimedia resources [2]. In addition, expressing multimedia knowledge by means of ontologies has the potential to improve the interoperability of different applications producing and consuming multimedia annotations and increases the precision of multimedia retrieval information systems [6].

The second issue in multimedia retrieval is concerning *querying* the multimedia metadata. Specialized query languages are required for the retrieval of information from the metadata repositories. In general, the metadata are not compatible with each other, and hence, multimedia repositories with different metadata interfaces cannot be queried in a unified way. The MPEG Query Format (MPQF) was developed to solve this problem [15, 16, 17]. The main goals of the MPQF are to facilitate and unify access to standalone and distributed multimedia repositories in an efficient, precise and expressive ways. To achieve these goals, the MPQF standard specifies precise input and output parameters to express multimedia requests and to allow clients easy interpretation and processing of result sets. Moreover, the management component of the MPQF

covers searching and the choice of the desired multimedia services for retrieval. For this purpose, the standard provides a means to describe service capabilities and to undertake service discovery. MPQF is based on XML, and therefore, is platform independent. So, developers can write their applications involving multimedia queries independently of the system used, which fosters software reusability and maintainability. Querying semantic RDF MPEG-7 descriptions with MPQF requires conversion of MPQF to SPARQL because MPQF is not supported. To the best of the author knowledge, two methods have been proposed to query RDF with MPQF (see Section 2). Both methods suggested adding new constructs to MPQF in order to convert the user query to SPARQL. Furthermore, the methods have some limitations on the query types. So, in this paper presents a set of rules to translate MPQF queries to SPARQL queries. The proposed rules do not add any construct to MPQF, and allow mapping a wider range of MPQF query types. The rest of the paper is organized as follows: Section 2 presents related work. Section 3 presents the details of the conversion rules from MPQF to SPARQL. Section 4 presents the implementation of a test application and finally, Section 5 concludes the paper.

2. RELATED WORK

There had been two proposals to use MPQF for querying RDF multimedia metadata. The first work suggested incorporating a new query type, the QueryBySPARQL, in addition to the existing types [18]. QueryBySPARQL allows RDF-based queries to be partly expressed in native SPARQL syntax. However, QueryBySPARQL defines some restrictions on the string that may be the content of element SPARQL. For example, SELECT clause is not allowed. The embedded query only makes use of the *ASK* clause to test whether or not a query pattern has a solution. The ASK query would result in a boolean value that indicates whether a solution to the query *exists* or not. No information is returned about the possible query solutions.

The second proposal is the Semantic Enhancement of MPQF [19], which adds Semantic Web query features to MPQF in order to query RDF metadata in a way that is appropriate for the RDF data model. The proposed extensions include a generalization of the MPQF metadata processing model which is derived from SPARQL query language and support for *triple patterns*. The Semantic Enhancement of MPQF introduces language constructs that can contain semantic query variables, such as ReqSemanticField and SemanticFieldType. The type SemanticRelation defines that an element with this type has three child elements with the names Subject, Property, and Object. These child elements hold the corresponding triple pattern parts in MPQF. The type SemanticRelation can appear in the type declaration of elements with name Condition. As an example, the following query condition:

```
<QueryCondition >
<Condition xsi:type ="SemanticRelation">
        <Subject >? movie </ Subject >
        <Property > ex:rank </ Property >
        <Object >? rank </ Object >
</ Condition >
</ QueryCondition >
```

is mapped to the following SPRQL construct:

```
WHERE {
        ?movie    ex: rank    ?rank .
}
```

As can be noted from the example above, the Semantic Enhancement of MPQF can be viewed as XML notation of the SPARQL query. This means, to write a query, the user should know RDF

metadata as well as the SPARQL language. Moreover, some of MPQF construct such as *NOT* and *XOR* conditions and query types cannot be expressed by the Semantic Enhancement.

3. MPQF TO SPARQL CONVERSION RULES

In the following subsections, we present a set of rules for mapping the MPQF Input Query Format, which represents the query request, to a SPARQL query [20, 21]. A detailed description of MPQF is not presented here to avoid repetition. The main idea of the mapping rules is to specify the triples in the basic graph pattern *(BGP)* of the SPARQL *SELECT* query that reflects the required data specified in the MPQF query. It is assumed that each MPEG-7 XML description is converted to RDF named graph, where elements and attributes are mapped to triples predicate. There are several methods to perform such conversion, see for example [22].

3.1 Mapping OutputDescription Element

OutputDescription element enables the user to specify the structure and information of the result set. It also allows limiting the maximum number of items per output page and the overall items number, and enables the user to use aggregation and sorting processes. *OutputDescription* element consists of the four major elements: *ReqField, ReqAggregateID, GroupBy,* and *SortBy*. *ReqField* element describes a data path within the item's metadata, which a user asks to be returned. Paths are specified using absolute XPath expressions, which refer to the root of the item's metadata. *ReqField* element has an optional attribute named *typeName* which is used to specify the name of the complex data type defined in the schema. *ReqAggregateID* element describes the ID of the aggregate operation, the requester asks to be returned. When one or more *ReqAggregateID*s are used, the aggregate ID should be in the *GroupBy* element. The *SortBy* element describes the sort operation the user wants to apply on the query results. In MPQF, the *SortBy* is performed by either using *SortByFieldType* or by using *SortByAggregateType*.

The following MPQF query comprises an *OutputDescription* element which contains the child elements mentioned above:

```
<MpegQuery>
  <Query>
    <Input>
      <OutputDescription >
        <ReqField typeName="CreationInformation">/Creation/Title</ReqField>
        <ReqField typeName="Creator">/Character/FamilyName</ReqField>
        <ReqAggregateID>avgSize</ReqAggregateID>
        <GroupBy>
          <GroupByField typeName="Creator">/Character/FamilyName</GroupByField>
          <Aggregate xsi:type="AVG" aggregateID="avgSize">
            <Field typeName="MediaFormat">/FileSize</Field>
          </Aggregate>
        </GroupBy>
        <SortBy xsi:type="SortByAggregateType" order="ascending">
          <AggregateID>avgSize</AggregateID>
        </SortBy>
      </OutputDescription>
    </Input>
  </Query>
</MpegQuery>
```

This MPQF query is mapped to SPARQL query based on the following rules:

1. Add one triple to the *BGP* for each *ReqField*, *GroupByField*, and *Field* elements, where the subject is a variable, the predicate is the value of these elements, and object is a variable (with the same name of requested data), i.e. the *BGP* must retrieve the data specified in the *ReqField* elements and the data required to perform the Grouping and Sorting operations.
2. Map the *GroupBy* element to Group By clause based on variables specified by the value of the GroupByField element.
3. The aggregate operation is mapped to an Expression in the *SELECT* clause.
4. Map the *SortBy* element to an Order By clause, with the specified order and dependent variable.

Based on these rules SPARQL translation of the above MPQF query is:

```
PREFIX mp: <urn:mpeg:mpeg7:schema:2001#>
SELECT ?title ?FamilyName AVG(?FileSize) AS ?avgSize
 WHERE {
        ?a      mp:FamilyName       ?FamilyName.
        ?b      mp:title            ?title.
        ?c      mp:FileSize         ?FileSize.
}
GROUP BY ?FamilyName
ORDER BY ASC (?avgSize)
```

As can be noted from the MPQF query, the data requested by the *ReqField*, *GroupByField*, and *Field* elements, are (FamilyName, title, and FileSize). So, three triples added to the *BGP* to retrieve the variables bounded to these data.

3.2 Mapping TargetMediaType and EvaluationPath Elements

TargetMediaType element is part of the *QueryCondition* element that contains MIME type descriptions of media formats, which are the targets for retrieval. The *QueryCondition* element is the part of the Input Query Format where the user specifies the properties of the media or the metadata to be retrieved. A MIME type is composed of (at least) two parts: a *type* and a *subtype* separated by "/". For instance, the MIME type audio/mp3 would filter all results for audio files depending on the MP3 format. The following MPQF query asks for MediaUri(s) of JPEG images:

```
<MpegQuery>
  <Query>
    <Input>
      <OutputDescription >
            <ReqField>MediaUri</ReqField>
       </OutputDescription>
       <QueryCondition>
            <TargetMediaType>image/JPEG</TargetMediaType>
       </QueryCondition>
    </Input>
  </Query>
</MpegQuery>
```

TargetMediaType element is mapped based on the following rules:

5. A Subquery is added to the *WHERE* clause to get the graphs from the dataset which has the specified MIME type. The *BGP* of this Subquery consists of two triples to retrieve the

two components of the MIME type and a *FILTER* clause to restrict the graphs to the specified MIME value.

6. The *BGP* of the main query is preceded by *GRAPH* keyword and a variable projected from the sub query which represents the graphs that satisfy the TargetMediaType.

Based on these rules, the conversion of the above query is:

```
PREFIX mp: <urn:mpeg:mpeg7:schema:2001#>
SELECT  ?MediaUri
WHERE {
GRAPH ?g {
            ?x  mp:MediaUri          ?MediaUri
          }
          {
            SELECT  ?g
            WHERE{
            GRAPH ?g {?a    mp:MediaFormat/mp:Content/@href         ?href.
                        ?b    mp:MediaFormat/mp:FileFormat/Name     ?Name.
                  FILTER (regex (?Name, JPEG) && (regex (?href, image))
            }}
 }}
```

EvaluationPath element is an optional XPath expression, which specifies the evaluation node of the metadata the query should consider. It also determines the structure of the output; one result item will be returned for each evaluation item if it matches the condition. The following example illustrates the use of the EvaluationPath element. It shows a query to search and retrieve for all image segments containing the word “Lausanne” somewhere in a textual description. Note, that for each matching segment, one different result item will be returned.

```
<MpegQuery>
  <Query>
   <Input>
     <QueryCondition>
       <EvaluationPath>//Image</EvaluationPath>
         <Condition xsi:type="Equal">
           <StringField typeName="CreationType">Title</StringField>
           <StringValue> Lausanne </StringValue>
         </Condition>
    </QueryCondition>
   </Input>
  </Query>
</MpegQuery>
```

If the EvaluationPath is specified, then when writing the triples in the *BGP,* the names of the predicates are set starting with the EvaluationPath value. Mapping the query above will be:

```
PREFIX mp: <urn:mpeg:mpeg7:schema:2001#>
SELECT  ?MediaUri
WHERE {
GRAPH ?g {
            ?a      mp:Image/../mp:MediaUri      ?MediaUri.
            ?b      mp:Image/../mp: Title        ?Title.
            FILTER regex (?Title, Lausanne)
```

```
        }
    }
```

Considering the predicates that start with Image element, guarantees that only data descendent from that element will be evaluated.

3.3 Mapping Comparison Conditions

The *Condition* element is the place where user expresses the filter criteria for information retrieval. It is a placeholder for a Boolean expression and may result in a filter tree. The filter tree can be constructed by three main constructs, namely comparison expressions, Boolean operators, and query types. A comparison expression is defined by an operation and two operands. The operations defined include: *GreaterThan*, *GreaterThanEqual*, *LessThan*, *LessThanEqual*, *Equal, NotEqual* and *Contains.* Both operands should belong to the same Operand Class within a comparison expression. The operands can be described by a value of the data type, an XPath expression pointing to a value of the specific data type, or a corresponding expression resulting to a value of the specific data type. Comparison conditions are mapped according to the following rule:

7. For each condition, a triple is added to the *BGP*, with *FILTER* clause to restrict the solution to those for which the comparison condition is true. The object and subject of the triple are variables, while the predicate is the name of the MPEG-7 element or attribute to be evaluated. The filter expression could be a value, or expression.

Consider the following examples:

```
<Condition xsi:type="Equal">
        <ArithmeticField typeName="MediaFormatType">FileSize</ArithmeticField>
        <LongValue>1000</LongValue>
</Condition>
```

Is mapped to:

```
 WHERE {
        ?x      mp:FileSize     ?FileSize
         FILTER (?FileSize=1000)
 }
```

The condition:

```
 <Condition xsi:type="Contains">
        <StringField typeName="CreationType">Title</StringField>
        <StringValue>MPEG Query</StringValue>
 </Condition>
```

Is mapped to:

```
 WHERE {
        ?x      mp:title        ?title
         FILTER regex (?title, MPEG Query*)
  }
```

The condition:

```
<Condition xsi:type="GreaterThan">
```

```
    <ArithmeticExpression xsi:type="Abs">
        <ArithmeticField>Mpeg7/Description/AudioVisual/Semantic/SemanticBaseType/
                AttributeValuePair/IntegerValue
        </ArithmeticField>
    </ArithmeticExpression>
    <LongValue>40</LongValue>
</Condition>
```

Is mapped to:

```
 WHERE {
       ?x
       Mpeg7/Description/AudioVisual/Semantic/SemanticBaseType/AttributeValuePair/IntegerV
       alue          ?IntegerValue
       FILTER ( Abs (?IntegerValue)> 40)
 }
```

3.4 Mapping Boolean Conditions

Boolean conditions are built using Boolean operators (*AND, OR, NOT, XOR*), which evaluate to a Boolean value. The following MPQF query contains *AND* condition which asks to retrieve all MediaUri which have a title equals Barcelona and date after 1-5-2005:

```
<MpegQuery>
  <Query>
    <Input>
      <OutputDescription>
        <ReqField>MediaUri</ReqField>
      </OutputDescription>
      <Condition xsi:type="AND" >
        <Condition xsi:type="Equal">
          <StringField typeName="CreationType">Title</StringField>
          <StringValue> Barcelona </StringValue>
        </Condition>
        <Condition xsi:type="GreaterThan">
            <DateTimeField typeName="CreationType">
                /CreationCoordinates/Date/TimePoint
            </DateTimeField>
            <DateTimeValue>2005-05-01</DateTimeValue>
        </Condition>
      </Condition>
    </Input>
  </Query>
</MpegQuery>
```

And conditions are mapped to SPARQL based on the following rules:

8. A Subquery is set to retrieve the graphs that satisfy the first condition
9. The retrieved graphs are then used to retrieve the requested data that satisfy the second condition.

Based on these rules, SPARQL translation of the above MPQF query is:

```
PREFIX mp: <urn:mpeg:mpeg7:schema:2001#>
```

```
SELECT  ?MediaUri
WHERE {
GRAPH ?g {
        ?x      mp:MediaUri   ?MediaUri
        ?y      mp: title          ?title
                FILTER regex (?title, Barcelona)
        }
        {
        SELECT  ?g
        WHERE {
                GRAPH ?g { ?a          mp:TimePoint   ?TP
                FILTER (CheckDate (?TP, "2005-05-01", "GreaterThan"))
        }}}
}
```

Where ChechDate is a user defined *FILTER* function which takes three parameters: a variable, date, and operator, and returns true if the date is correct with respect to the condition.
If the *AND* condition is replaced with *OR* condition in the MPQF query above, then, *OR* conditions are mapped based on the following rules:

10. Construct two *BGPs,* one for each of the *OR* operands' conditions, where in each *BGP* a distinct graph variable is used.
11. Use *UNION* clause to combine the results from both *BGP*s.

Based on these rules SPQRQL translation of the above MPQF query is:

```
PREFIX mp: <urn:mpeg:mpeg7:schema:2001#>
SELECT  ?MediaUri
WHERE {
{GRAPH ?h {
                ?x      mp:MediaUri   ?MediaUri
                ?y      mp:Title          ?title.
                FILTER  regex(?title , Barcelona)
      }
}
UNION
{
        GRAPH ?g {
                ?x      mp:MediaUri   ?MediaUri
                ?y      mp:TimePoint   ?TP
                FILTER  (checkDate (?TP, "2005-05-01", "GreaterThan"))

}}}
```

Note that both *BGP*s contain triples that retrieve the requested data (MediaUri).

Finally, the following query contains *NOT* condition which asks to retrieve all MediaUri which have a file size not equal to 1000:

```
<MpegQuery>
  <Query>
    <Input>
      <OutputDescription>
        <ReqField>MediaUri</ReqField>
      </OutputDescription>
```

```
        <Condition xsi:type="NOT" >
          <Condition xsi:type="Equal" >
            <ArithmeticField typeName="MediaFormatType">FileSize</ArithmeticField>
            <ArithmeticExpression xsi:type="NumericConstantValue">
                <Long>1000</Long>
            </ArithmeticExpression>
          </Condition>
        </Condition>
    </Input>
  </Query>
</MpegQuery>
```

To map this query to SPARQL:

12. A Subquery is set to retrieve all named graphs that satisfy the condition before negation.
13. Retrieve requested data in the main *BGP* using distinct variable for the *GRAPH* clause.
14. Filter the results by retrieving data only where the two named graphs (the one used for the Subquery and the one used for the main query) are not equal.

Based on these rules, SPARQL translation of the above MPQF query is:

```
PREFIX mp: <urn:mpeg:mpeg7:schema:2001#>
SELECT  ?MediaUri
WHERE {
        FILTER ( ?g != ?h)
        GRAPH ?h {
                ?x     mp:MediaUri        ?MediaUri
        }
        {
        SELECT  ?g
        WHERE{
                GRAPH ?g {
                    ?x    mp:FileSize       ?FileSize
                     FILTER  (?FileSize==1000)
                }
         }}.
```

3.5 Mapping Query Types

MPQF provides the following query types: QueryByMedia, QueryByDescription, QueryByFeatureRange, SpatialQuery, TemporalQuery, QueryByXQuery, QueryByFreeText, QueryByROI, and QueryByRelevanceFeedback. The following subsections will present how to map some of these query type into SPARQL queries.

3.5.1 QueryByMedia

QueryByMedia type enables the user to perform a search based on a given example of media resource. It provides an attribute (*matchType*) to set the search criteria whether similar-match or exact-match. To illustrate how this query type is translated to SPARQL, consider the following query sample which asks to retrieve images that are similar to the query one:

```
<MpegQuery>
  <Query>
    <Input>
      <QueryCondition>
```

```
        <Condition xsi:type="QueryByMedia" matchType="similar">
        <MediaResource resourceID="Image001">
          <MediaResource>
            <MediaUri> //Images/testimage.jpg</MediaUri>
          </MediaResource>
        </MediaResource>
        </Condition>
       </QueryCondition>
    </Input>
  </Query>
</MpegQuery>
```

Conversion QueryByMedia to SPARQL is performed as follows:

15. Implement a *FILTR* LIKE function which finds whether two images are similar or not.
16. Add a triple, that retrieves the MediaUri, to *BGP*, and use the LIKE function to filter the results.

Based on these rules SPARQL translation of the above MPQF query is:

```
PREFIX mp: <urn:mpeg:mpeg7:schema:2001#>
SELECT  ?MediaUri
WHERE {
GRAPH ?g {
                ?a        mp:MediaUri    ?MediaUri
                 FILTER  LIKE(?MediaUri,'testimage.jpg')
        }
}
```

Where LIKE is a user defined *FILTER* function that returns true if the query resource is similar to those in the database.

3.5.2 QueryByFreeText

QueryByFreeText type enables a requester to perform a search based on free-text. It contains a *FreeText* element containing text description as a condition, and an optional choice of fields (*SearchField, IgnoreField*), which allows the user to if the search should be performed in specific elements only or if specific elements should be ignored. The following example illustrates the use of the QueryByFreeText type, without any SearchField and IgnoreField elements.

```
<MpegQuery>
  <Query>
    <Input>
      <QueryCondition>
        <Condition xsi:type="QueryByFreeText">
          <FreeText>Barcelona</FreeText>
        </Condition>
      </QueryCondition>
    </Input>
  </Query>
</MpegQuery>
```

To map this query type:

17. The text fields that are possible for text search are predefined, for example: FreeTextAnnotation, Title, etc.

18. A triple is added for each text field to get its value with a *FILTER* to limit the results to those which match the query text.

Based on these rules, SPARQL translation of the above MPQF query is:

```
PREFIX mp: <urn:mpeg:mpeg7:schema:2001#>
SELECT  ?MediaUri
WHERE {
{GRAPH ?h {
                    ?a     mp:MediaUri         ?MediaUri
                    ?b    mp:TextAnnotation/mp:FreeTextAnnotation            ?text.
                    ?c    mp:Title                          ?title.
                    FILTER  regex(?text , Barcelona*)
                    FILTER  regex(?title , Barcelona*)

       }
}}
```

3.5.3 QueryByDescription

QueryByDescription is a type of query by example which enables a requester to perform a search based on a given example description. Any description can be embedded if it is based on XML schema and the description conforms to the schema. QueryByDescription also provides an attribute to indicate the search criteria regarding similar-match or exact-match. Consider the following example which shows an MPEG-7 description included in a query which asks for descriptions exactly matching the attached one:

```
<MpegQuery>
  <Query>
    <Input>
      <QueryCondition>
        <Condition xsi:type= ‘QueryByDescription’ matchType= ‘exact’>
          <DescriptionResource resourceID= ‘desc001’>
            <AnyDescription xmlns:mp7= ‘urn:mpeg:mpeg7:schema:2001’>
              <mp7:Mpeg7>
                <mp7:DescriptionUnit xsi:type=”mpeg7:CreationInformationType”>
                    <mp7:Creation>
                      <mp7:Title>
                              Miracle Query Format
                       </mp7:Title>
                       <mp7:CreationCoordinates>
                          < mp7:Date>
                               < mp7:TimePoint>2011-05-10</ mp7:TimePoint>
                          </ mp7:Date>
                       </mp7:CreationCoordinates>
                    </mp7:Creation>
                </mp7:DescriptionUnit>
              </mp7:Mpeg7>
            </AnyDescription>
          </DescriptionResource>
        </Condition>
      </QueryCondition>
    </Input>
  </Query>
```

</MpegQuery>

For exact match, all leaf nodes in the XML description must be matched. To achieve this:

19. A Subquery is set to match every leaf node, by including a triple to retrieve the value on the node.
20. All Subqueries use the same *GRAPH* variable so that *AND* operation is performed on them.

Based on these rules SPARQL translation of the above MPQF query is:

```
PREFIX mp: <urn:mpeg:mpeg7:schema:2001#>
SELECT  ?MediaUri
WHERE {
        GRAPH ?g {
                ?x  mp:MediaUri          ?MediaUri
        }
        {
        SELECT  ?g
                WHERE{
                        GRAPH ?g {
                                ?y  mp:Creation/mp:Title          ?title
                                FILTER regex (?title, Miracle Query Format)
                        }
        {
        SELECT  ?g
        WHERE{
                GRAPH ?g {
                ?y      mp:Creation/mp:CreationCoordinates/mp:Date/mp:TimePoint      ?TP
                FILTER  (checkDate (?TP, “2011-05-10”, “GreaterThan”))

        }}
}}
```

4. IMPLEMENTATION OF A TEST-BED APPLICATION

To test the correctness of the proposed mapping rules detailed above, a Java application is developed using Jena framework for managing and storing RDF metadata, and ARQ for querying this metadata with SPARQL. Figure.1 depicts the different components of the application.

- MPEG-7 documents, which can be produced by any MPEG-7 based annotation tools such as Caliph Image annotation tool [23] or VAnalyzer video annotation tool [24].
- Gloze [22]: this is an open-source Java library which can perform automatic lossless round-tripping between XML and RDF as well as generate OWL ontologies from XSD schemas. Gloze is attractive since it does not require an XSLT mapping, nor does the conversion depend on serializing RDF as RDF/XML.
- The Jena/ARQ framework [25, 26]: Jena is a Java Semantic Web framework which provides classes and interfaces for the creation and manipulation of RDF repositories and OWL ontologies. Moreover, Jena provides reasoning over ontology models. Jena provides querying capability by using ARQ framework which provides a SPARQL query engine for the Jena. The classes of ARQ framework complement the classes of Jena framework, so that instances of the query class from Jena can be executed with class instances from the ARQ framework. With Jena/ARQ framework, SPARQL queries are

created and executed, and using ontologies provides more reasoning capabilities than the simple entailment which is required by the SPARQL specification. Jena SDB [27] is another component of the Jena Semantic Web Framework and provides storage and query for RDF datasets using conventional relational databases. It can be accessed through the Java-based Jena API library as well as through a set of command line utilities.

- MPQF to SPQRQL: this component implements the conversion rules presented in Section 3. An input MPQF query is translated to SPARQL query which can be then executed be Jena/ARQ engine.
- SPARQL FILTER library: in this component a set of SPARQL FILTER, such as LIKE FILTER function, that are required for perform MPQF query types are implemented.

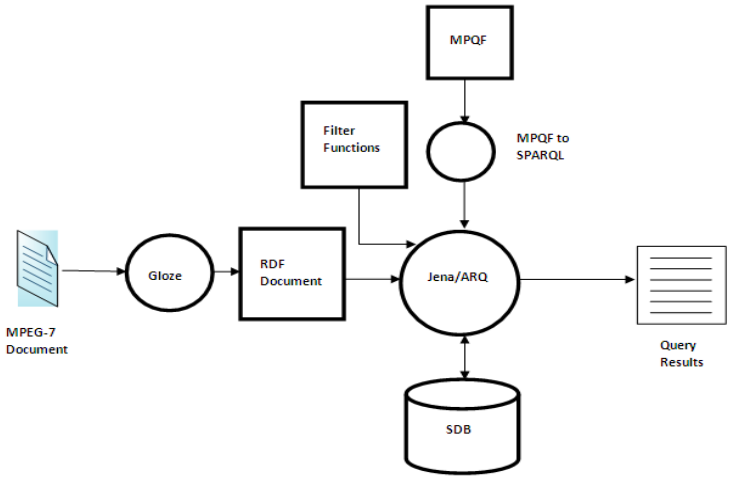

Figure. 1 Components of the test application

A number of images were annotated using Caliph Image annotation tool, and the generated MPEG-7 documents were converted into RDF documents using Gloze. A post-processing step is performed on each RDF document to make sure that the datatypes are correct. The RDF descriptions are then used to build a dataset which can be stored in an RDF data store such are SDB. Once the RDF dataset is constructed, user queries, which are formulated via a form based user interface, are converted to SPARQL queries based on the rules presented in Section 3. Finally, Jena ARQ SPARQL query engine executes the queries and returns the query results. The aim of this implementation is to test the mapping rules not the efficiency of retrieval process. All the examples presented in the Section 3 have been tested and the results showed the mapping rules are correct. Of course, the performance of the system depends on the size of the database and efficiency of the implementation especially the FILTER functions which require data type conversion from string to array of integers.

5. CONCLUSIONS

Having MPEG-7 descriptions transformed to semantic RDF metadata has several advantages. This transformation strongly impacts the way multimedia contents are searched and retrieved in a broad range of application scenarios. Querying MPEG-7 RDF descriptions with a modern standard multimedia query language like MPQF allows for more flexible interoperability approaches, like those involving multiple metadata formats. In this paper we have purposed a set of rules for conversion MPQF query to SPARQL query which will allow MPQF to manage MPEG-7 RDF metadata. Contrary to the other proposed methods presented in Section 2, the described rules do not include a conceptual generalization of the MPQF metadata processing

model or definition of a new MPQF query type as in [23], nor do they add new elements to the standard as in [17]. This goes in line with the main aim of MPQF to unify access to multimedia metadata. The proposed mapping rules heavily depend on the new constructs introduced in SPARQL1.1 such as sub query, property paths, aggregate functions, and FILTER functions.

As for future work, we plan to continue to evaluate this proposed framework with MPEG-7 based ontologies and other metadata like DC. We also plan to experiment with more query types such as special and temporal queries. Finally, we intend to work on using MPQF for querying non XML based RDF metadata.

REFERENCES

[1] ISO/IEC 15938 version 2 (2004) Information technology—multimedia content description interface (MPEG-7)

[2] Dasiopoulou S, Tzouvaras V, Kompatsiaris I, Strintzis M (2010) Enquiring MPEG-7 based multimedia ontologies. Special Issue on Data Semantics for Multimedia Systems; Guest Editors: Mei-Ling Shyu, YuCao, Jun Kong, Ming Li, Mathias Lux and Jie Bao. In Journal Multimedia Tools and Applications. Volume 46, Numbers 2–3, 331–370. January 2010

[3] Frank Manola and Eric Miller, eds.(2004) *RDF Primer. W3C Recommendation*. URL: http://www.w3.org/TR/rdf-primer/

[4] Arndt R, Troncy R, Staab S, Hardman L (2009) COMM: A Core Ontology for Multimedia Annotation. In Staab S, Studer R (Eds.) Handbook on Ontologies, 2nd ed., Series: International Handbooks on Information Systems. Springer Verlag, pp. 403–421, 2009.

[5] Hunter J. (2001) Adding Multimedia to the Semantic Web - Building MPEG-7 Ontology. International Semantic Web Working Symposium (SWWS), Stanford, July 30 - August 1, 2001

[6] Mari Carmen Suarez-Figueroa, Ghislain Auguste Atemezing, and Oscar Corcho (2011) The landscape of multimedia ontologies in the last decade. Multimedia Tools and Applications, DOi 10.1007/s11042-011-0905-z

[7] Troncy R, Celma O, Little S, Garcia R, Tsinaraki C (2007) MPEG-7 based Multimedia Ontologies: Interoperability Support or Interoperability Issue? In International Workshop on Multimedia Annotation and Retrieval enabled by Shared Ontologies (MAReSO), p. 2–15.

[8] Boris Motik, Peter F. Patel-Schneider, and Bernardo Cuenca Grau, eds. (2009) *OWL 2 Web Ontology Language Direct Semantics. W3C Recommendation.* URL: http://www.w3.org/TR/owl2-direct-semantics/

[9] Dan Brickley and R. V. Guha, eds. (2004) RDF *Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation*. URL: http://www.w3.org/TR/rdf-schema/

[10] Deborah L. McGuinness and Frank van Harmelen, eds. (2004) *OWL Web Ontology Language Overview. W3C Recommendation*. URL: http://www.w3.org/TR/owl-features/

[11] Jie Bao, et.al eds. (2009) *OWL 2 Web Ontology Language Document Overview. W3C Recommendation*. URL: http://www.w3.org/TR/owl2-overview/

[12] Jeremy J. Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler (2005) Named graphs provenance and trust. In WWW '05: Proceedings of the 14th international conference on World Wide Web, pages 613-622, New York, NY, USA

[13] García R, Celma O. (2005) Semantic Integration and Retrieval of Multimedia Metadata. 5th Knowledge Markup and Semantic Annotation Workshop, Sem. Annot 2005. CEUR Workshop Proceedings, Vol. 185,pp. 69–80, 2006 ISSN 1613–0073

[14] Michael Hausenblas, Raphael Troncy, Yves Raimond, Tobias Bürger (2009) Interlinking Multimedia: How to Apply Linked Data Principles to Multimedia Fragments, Linked Data on the Web Workshop (LDOW 09), in conjunction with 18th International World Wide Web Conference (WWW 09).

[15] Döller M, Tous R, Gruhne M, Yoon K, Sano M, Burnett IS (2008) The MPEG Query Format: on the way to unify the access to multimedia retrieval systems. IEEE Multimedia, ISSN: 1070-986X,15(4)

[16] Mario Döller, Ruben Tous, Matthias Gruhne, Miran Choi, Tae-Beom Lim, Jaime Delgado, and Armelle Yakou. (2009) Semantic MPEG Query Format Validation and Processing. In: *IEEE MultiMedia* 99.1 ISSN: 1070-986X.

[17] Mario Döller, Ruben Tous, Matthias Gruhne, Kyoungro Yoon, Masanori Sano, and Ian S. Burnett. (2008) The MPEG Query Format: Unifying Access to Multimedia Retrieval Systems. In: *IEEE MultiMedia.* Ed. by John R. Smith, pp. 82–95. ISSN:1070-986X.

[18] Ruben Tous and Jaime Delgado, (2008) Semantic-Driven Multimedia Retrieval with the MPEG Query Format. In: *SAMT '08: Proceedings of the 3rd International Conference on Semantic and Digital Media Technologies*. Koblenz, Germany: Springer-Verlag. ,pp. 149–163. ISBN: 978-3-540-92234-6.

[19] Katrin Fehlner, (2011) Semantic Retrieval by means of the MPEG Query Format, Master Thesis, University of Passau, http://www.dimis.fim.unipassau.de/iris/mpqf/Master_Thesis_Katrin_Fehlner.pdf

[20] Eric Prud'hommeaux and Andy Seaborne, eds. (2008) *SPARQL Query Language for RDF. W3C Recommendation,* URL: http://www.w3.org/TR/rdf-sparql-query

[21] Steve Harris and Andy Seaborne, eds. (2011) *SPARQL 1.1 Query Language. W3C Working Draft 12 May 2011*. W3C. 2011. URL: http://www.w3.org/TR/sparql11-query/

[22] Battle, S. (2006) Gloze: XML to RDF and back again. In: Proceedings of the First Jena User Conference, Bristol, UK

[23] Mathias Lux. (2009) Caliph & Emir: MPEG-7 Photo Annotation and Retrieval. In: *MM '09:Proceedings of the Seventeenth ACM International Conference on Multimedia*. Beijing, China: ACM, pp. 925–926. ISBN: 978-1-60558-608-3.

[24] Florian Stegmaier , Mario Döller, David Coquil, Vanessa El khoury, Harald Kosch (2010) VAnalyzer: a MPEG-7 based Semantic Video Annotation Tool. Dans *Workshop on Interoperable Social Multimedia Applications (WISMA 2010)*, Barcelona, Spain

[25] http://incubator.apache.org/jena/index.html

[26] http://incubator.apache.org/jena/documentation/query/

[27] http://incubator.apache.org/jena/documentation/sdb/

Author

Dr. Mohammed Al-Zoube received his B.Sc. and M.Sc. in Electrical Engineering from Jordan University of Science and Technology in 1990 and 1994, respectively. In 2002, he received his PhD in Computer Science from University Science Malaysia, specializing in multimedia systems. Currently he is working for Princess Sumaya University for Technology at the department of Computer Graphics and Animation. His research interest includes multimedia retrieval, e-learning, cloud computing, and semantic web.