Semantic-Driven Multimedia Retrieval with the MPEG Query Format

Ruben Tous and Jaime Delgado

Distributed Multimedia Applications Group (DMAG)
Universitat Politecnica de Catalunya (UPC), Dpt. d'Arquitectura de Computadors
rtous@ac.upc.edu, jaime.delgado@ac.upc.edu

Abstract. The MPEG Query Format (MPQF) is a new standard from the MPEG standardization committee which provides a standardized interface to multimedia document repositories. The purpose of this paper is describing the necessary modifications which will allow MPQF to manage metadata modelled with Semantic Web languages like RDF and OWL, and query constructs based on SPARQL. The suggested modifications include the definition of a new MPQF query type, and a generalization of the MPQF metadata processing model. As far as we know, this is the first work to apply the MPEG Query Format to semantic-driven search and retrieval of multimedia contents.

1 Introduction

The research around multimedia information retrieval (MIR) is gaining relevance due to the increasing amount of digitally stored multimedia contents and the evolution of the ways of managing and consuming them. The main goals are to enable efficient, precise and expressive ways to query standalone and distributed multimedia data collections. This responds to the necessity to fullfill the new requirements imposed by the scaling of traditional collections, but specially to satisfy new requirements related to novel usages.

1.1 Multimedia Search and Retrieval: Information Retrieval and Data Retrieval

The MIR process usually starts with an end-user expressing his information needs through a human-friendly query interface. User's information needs come from the conceptual level, and combine criteria about the information represented by the content's data, i.e. the *meaning* of the content (e.g. *"images of a bird"*) with other criteria about the features of the content's data itself (e.g. *"image files of less than one megabyte"*). At the end, the only way of fulfilling the user information needs is translating these criteria into machine-readable conditions, as precise as possible, over the content's data, being these data the media binary representation or some metadata annotations. Metadata annotations provide information about the content at different levels, from low-level features and management information, to semantic-level descriptions.

D. Duke et al. (Eds.): SAMT 2008, LNCS 5392, pp. 149–163, 2008.
© Springer-Verlag Berlin Heidelberg 2008

So, the problem is twofold, in one hand we have the challenge of enriching the media data with metadata useful for solving the queries. On the other hand we need to face the problem of formalizing the user information needs, as far as possible, in the form of a query expressed in terms of the available metadata model. Even in we succeed in the first challenge, the second remains non trivial, because end-users expresses their criteria performing actions over the provided human-friendly user interface. While some criteria can be easy to formalize, *semantic-level* criteria use to be more difficult to express and also to be solved in a deterministic way. For these situations, formal conditions are usually combined with non-formal or "fuzzy" query terms (e.g. Query By Example, Query By Keywords) and Information Retrieval techniques are applied.

So, MIR systems have the particularity that they must combine Information Retrieval (IR) techniques, with techniques for querying metadata, which belong to the Data Retrieval (DR) area within the Databases discipline. Both approaches aim to facilitate users access to information, but from different points-of-view. On one hand, an Information Retrieval system aims to retrieve information that might be relevant to the user even though the query is not formalized, or the criteria are fuzzy. In contrast, a Data Retrieval system (e.g. an XQuery-based database) deals with a well defined data model and aims to determine which objects of the collection satisfy clearly defined conditions (e.g. the title of a movie, the size of a video file or the fundamental frequency of an audio signal).

1.2 MPEG Query Format (MPQF)

Recently, the MPEG standardization committee (ISO/IEC JTC1/SC29/WG11) is developing a new standard, the MPEG Query Format (MPQF) [1,4], which aims to provide a standardized interface to multimedia document repositories. MPQF defines the format of queries and responses between parties in a multimedia search and retrieval process. In one hand, standardizing such kind of language fosters *interoperability* between parties in the multimedia value chain (e.g. content providers, aggregators and user agents). On the other hand, MPQF favours also *platform independence*; developers can write their applications involving multimedia queries independently of the system used, which fosters software reusability and maintainability.

One of the key features of MPQF is that it is designed for expressing queries combining the expressive style of Information Retrieval (IR) systems (e.g. query-by-example and query-by-keywords) with the expressive style of XML Data Retrieval (DR) systems (e.g. XQuery [17]), embracing a broad range of ways of expressing user information needs. Regarding IR-like criteria, MPQF offers a broad range of possibilities that include but are not limited to *QueryByDescription* (query by example description), *QueryByFreeText*, *QueryByMedia* (query by example media), *QueryByROI* (query by example region of interest), *QueryByFeatureRange*, *QueryBySpatialRelationships*, *QueryByTemporalRelationships* and *QueryByRelevanceFeedback*. Regarding DR-like criteria, MPQF offers its own XML query algebra for expressing conditions over the multimedia related

XML metadata (e.g. Dublin Core , MPEG-7 or any other XML-based metadata format) but also offers the possibility to embed XQuery expressions.

1.3 Semantic-Driven Multimedia Information Retrieval

So, a MIR system will deal with two related but different challenges, IR and DR. Though there is a solid research basis regarding the Information Retrieval challenge, the necessity to face such problem appears, in fact, because of the difficulty of annotating the content with the necessary metadata and the difficulty of formalizing the end-user's *semantic-level* criteria. The gap between low-level content description and querying, and the related high-level or semantic description and querying is known as the *semantic gap*. As a result, from the multimedia retrieval point-of-view, measures are needed to deal with uncertainty and the potential lack of search precision. However, in a vast number of scenarios, simple IR-like mechanisms like keywords-based search use to offer pretty satisfactory results even when the size of the target collections is big.

There are, nevertheless, situations in which the end-user requirements, and/or the circumstances, motivate the efforts of producing higher-level metadata descriptors (semantic or not) and formalizing parts of the user's *semantic-level* criteria moving them to the Data Retrieval realm. An example could be the video surveillance scenario, in which a huge quantity of information is stored, and the query expressiveness and results precision are critical. This *formalization* task requires enhancing the metadata production layer but also implies offering to the user a richer interface or, in subsequent layers, post-processing the initial non-formalized query. This enrichment of the querying process is related to the improvement of the metadata-level query capabilities. The result is the starting point of what is known as *semantic-driven* Multimedia Information Retrieval, whose evolution leads to the usage of *semantic-specific* technologies as those from the Semantic Web initiative.

1.4 MPQF Limitations Managing Semantic Web Related Metadata

MPQF is an XML-based language in the sense that all MPQF instances (queries and responses) must be XML documents. Formally, MPQF is Part 12 of ISO/IEC 15938, "Information Technology - Multimedia Content Description Interface" better known as MPEG-7 [8]. However, the query format was technically decoupled from MPEG-7 and is now metadata-neutral. So, MPQF is not coupled to any particular metadata standard, but it assumes that any metadata related to multimedia content being modelled as an XML Infoset. So, this metadata *neutrality* is constrained by the fact that MPQF expresses conditions and projections over the metadata using XPath expressions, i.e. privileging XML-enabled metadata repositories but restraining those based in other models, specially those based in RDF [10] metadata.

In this paper we describe the necessary modifications which will allow MPQF to manage metadata modelled with Semantic Web languages like RDF and OWL [9], and query constructs based on SPARQL [14]. The suggested modifications

include the definition of a new MPQF query type, and a generalization of the MPQF metadata processing model.

2 MPQF in Depth

2.1 MPEG Query Format Syntax and Terminology

Before entering the discussion of how the MPQF syntax and data model could be adapted for semantic-driven retrieval, let's briefly describe the structure of MPQF instances. MPQF queries (requests and responses) are XML documents that can be validated against the MPQF XML schema (see Figure 1). MPQF instances include always the *MpegQuery* element as the root element. Below the root element, an MPQF instance includes the *Query* element or the *Management* element. MPQF instances with the *Query* element are the usual requests and responses of a digital content search process. The *Query* element can include the *Input* element or the *Output* element, depending if the document is a request or a response. The part of the language describing the contents of the *Input* element (requests) is named the Input Query Format (IQF) in the standard. The part of the language describing the *Output* element (responses) is named the Output Query Format (OQF) in the standard. IQF and OQF are just used to facilitate understanding, but do not have representation in the schema. Alternatively, below the root element, an MPQF document can include the *Management* element. Management messages (which in turn can be requests and responses) provide means for requesting service-level functionalities like interrogating the capabilities of a MPQF processor.

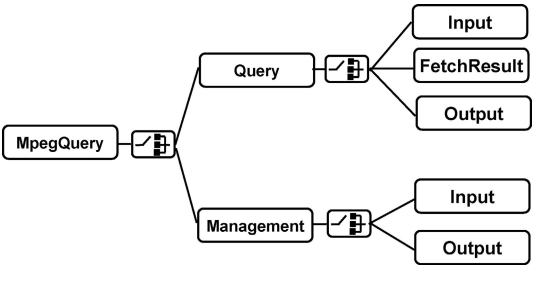

Fig. 1. MPQF language parts

Example in Code 1 shows an input MPQF query asking for JPEG images related to the keyword *"Barcelona"*.

2.2 MPEG Query Format Database Model

As happens with any other query language, an MPQF query is expressed in terms of a specific database model. The MPQF database model formally defines

Code 1. Example MPQF input query

```
<MpegQuery>
  <Query>
    <Input>
      <OutputDescription outputNameSpace="//purl.org/dc/elements/1.1/">
        <ReqField>title</ReqField>
        <ReqField>date</ReqField>
      </OutputDescription>
      <QueryCondition>
        <TargetMediaType>image/jpg</TargetMediaType>
        <Condition xsi:type="AND" preferenceValue="10">
          <Condition xsi:type="QueryByFreeText">
            <FreeText>Barcelona</FreeText>
          </Condition>
          <Condition xsi:type="GreaterThanEqual">
            <DateTimeField>date</DateTimeField>
            <DateValue>2008-01-15</DateValue>
          </Condition>
        </Condition>
      </QueryCondition>
    </Input>
  </Query>
</MpegQuery>
```

the information representation space which constitutes the evaluation basis of an MPQF query processor. MPQF queries are evaluated against one or more multimedia databases which, from the point-of-view of MPQF, are unordered sets of *Multimedia Contents*. The concept of *Multimedia Content* (MC) refers to the combination of multimedia data (resource) and its associated metadata. MPQF allows retrieving complete or partial MC's data and metadata by specification of a condition tree. So, MPQF deals with a dual database model (see Figure 2) constituted by content and metadata.

Example in Figure 3 shows a graphical representation of an MPQF condition tree carrying two different conditions which reflect the duality of the database model used by the language. In one hand, there's an IR-like condition using the

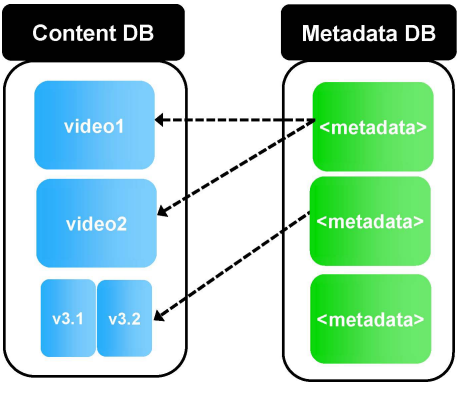

Fig. 2. Dual database model

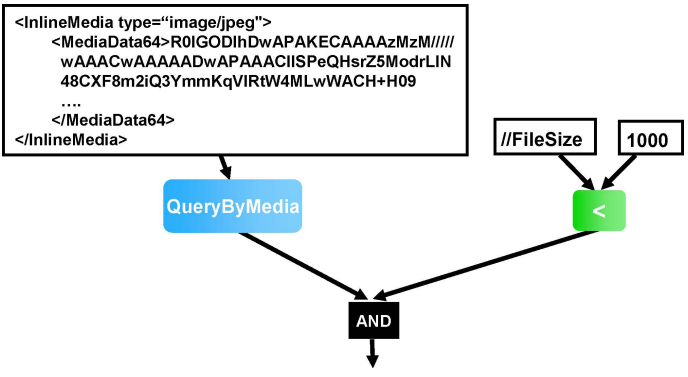

Fig. 3. Example condition tree

QueryByExample query type and including the Base64 encoding of the binary contents of an example JPEG image. On the other hand, there is a simple DR-like condition specifying that the metadata field *FileSize* must be inferior to 1000 bytes.

In order to deal this model duality, MPQF operates over sequences of what the standard calls *evaluation-items*. By default, an *evaluation-item* (EI) is a multimedia content in the multimedia database, but other types of EIs are also possible. For instance, an EI can be a segment of a multimedia resource, or an XPath-item related to a metadata XML tree. The scope of query evaluation and the granularity of the result set (the granularity of EIs) can be determined by a *EvaluationPath* element specified within the query. If this *EvaluationPath* element is not specified, the output result is provided as a collection of multimedia contents, as stored in the repository, all satisfying the query condition.

The *EvaluationPath* element determines the query scope on basis to a hypothetical XML metadata tree covering the entire database. So, we can redrawn

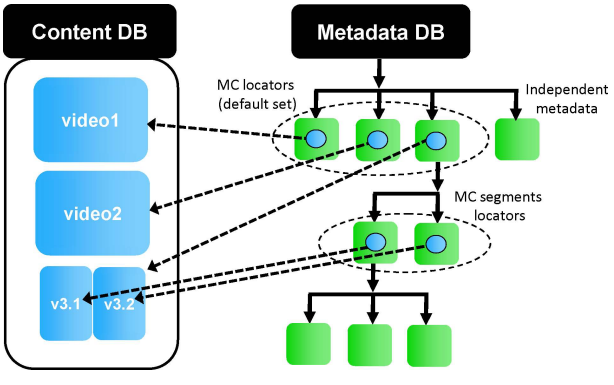

Fig. 4. Hierarchical MPQF model

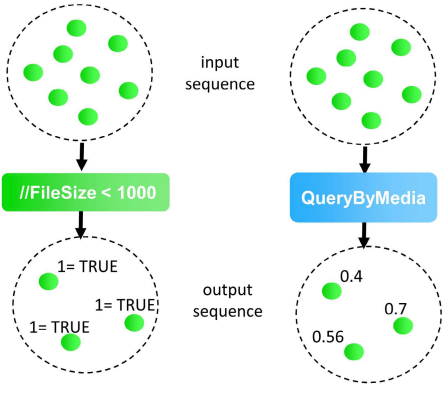

Fig. 5. Two evaluation styles in MPQF: Boolean and fuzzy-logic

our visual representation of the model in order to show this hierarchical nature, as shown in Figure 4.

2.3 MPQF Evaluation Model

The condition tree of an MPQF query is constructed combining filtering elements (conditions) from the *BooleanExpressionType* and interconnecting them with Boolean operators (*AND*, *OR*, *NOT* and *XOR)*. Each condition acts over a sequence of evaluation-items and, for each one, return a value in the range of [0..1]. In the case of DR-like conditions (e.g. it "the size of the file must be smaller than 1000 bytes") the condition can return just 1 or 0, which mean *true* and *false* respectively. In the case of IR-like conditions, they can evaluate to any value in the range of [0..1]. A *threshold* value within a condition is used to indicate the minimum value the score of an evaluation-item is required to have. Otherwise the evaluation-item is not considered further during evaluation.

So, with respect to DR-like conditions, MPQF acts as a conventional Boolean-based filtering language, while with respect to IR-like conditions MPQF acts preserving scores as a fuzzy-logic system. The standard specifies the behaviour of the provided Boolean operators in presence of non-Boolean values.

3 An MPQF Extension to Manage Semantic-Modelled Metadata

3.1 Metadata and RDF

Current practices in the metadata community show an increasing usage of Semantic Web technologies like RDF and OWL. Some relevant initiatives are choosing the RDF language (e.g. [2]) for modelling metadata (semantic or not) because of its advantages with respect to other formalisms. RDF is modular; a subset of RDF triples from an RDF graph can be used separately, keeping a consistent

RDF model. So it can be used in presence of partial information, an essential feature in a distributed environment. The union of knowledge is mapped into the union of the corresponding RDF graphs (information can be gathered incrementally from multiple sources). Furthermore, RDF is the main building block of the Semantic Web initiative, together with a set of technologies for defining RDF vocabularies and ontologies like RDF Schema [12] and the Ontology Web Language (OWL) [9].

RDF comprises several related elements, including a formal model, and an XML serialization syntax. The basic building block of the RDF model is the triple *subject-predicate-object*. In a graph theory sense, an RDF instance is a labelled directed graph, consisting of vertices, which represent *subjects* or *objects*, and labelled edges, which represent *predicates* (semantic relations between *subjects* and *objects*). Our proposal is to generalize the metadata part of the MPQF database model to allow the presence (in requests and responses) of RDF predicates. Example in Code 2 shows metadata including semantic information about a JPEG image in the triples notation [10].

Code 2. Example RDF graph using the triples notation

```
example:bigbird.jpg    dc:creator                "Ruben Tous .
example:bigbird.jpg    example:terms/represents  example:terms/Eagle .
example:terms/Eagle    rdfs:subClassOf           example:terms/Bird .
```

3.2 OWL, Ontologies, and Knowledge Bases

OWL is a vocabulary for describing properties and classes of RDF resources, complementing the RDF Schema (RDFS) capabilities in providing semantics for generalization-hierarchies of such properties and classes. OWL enriches the RDFS vocabulary by adding, among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes. OWL has the influence of more than 10 years of Description Logic research, and is based on the SH family of Description Logics [5]. The language has three increasingly expressive sublanguages designed for different uses: OWL Lite, OWL DL and OWL Full.

The combination of OWL ontologies with RDF statements compliant with the vocabularies and constraints defined in these ontologies make up what is known as a knowledge base (KB). The terms *Tbox* (terminological component) and *Abox* (assertion component) are used to describe these two different types of statements in a KB (see Figure 6).

3.3 SPARQL

SPARQL [14] is a popular RDF query language which consists of triple patterns. SPARQL is based in RDQL [13] from HP Labs Bristol, which in turn was based

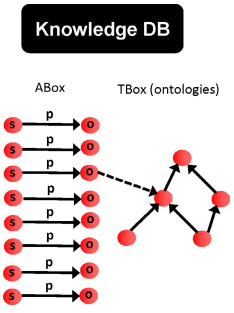

Fig. 6. Knowledge Base (KB)

in SquishQL [15], derived from rdfDB [11]. SPAQL is standardized by the RDF Data Access Working Group (DAWG) of the World Wide Web Consortium, and has become an official W3C Recommendation on 15th January 2008.

An SPARQL query consists of a *basic graph pattern*, expressed as a list of triple patterns. Triple patterns are like RDF triples except that each of the subject, predicate and object may be a variable. Each triple pattern is comprised of named variables and RDF values (URIs and literals). An SAPQRL query can additionally have a set of constraints on the values of those variables, and a list of the variables required in the answer set. An example SPARQL query is shown in Code 3.

Code 3. Example SPARQL query

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  ns:  <http://example.org/ns#>
SELECT  ?title ?price
WHERE   { ?x ns:price ?price .
          FILTER (?price < 30.5)
          ?x dc:title ?title . }
```

Each solution to the example query will expose one way in which the selected variables can be bound to RDF terms so that the query pattern matches the data. The result set gives all the possible solutions. We envisage that SPARQL will become the standard language for querying semantic metadata. In the following sections we describe the necessary actions to allow SPARQL expressions to be embedded within MPQF queries, in a similar way that XQuery expressions can.

3.4 MPQF Database Model and Semantic-Modelled Metadata

We defend that the MPQF model should be generalized in such a way that semantic metadata, modelled with knowledge representation techniques such as RDF and OWL, could be also used in query conditions and resultsets. MPQF strongly relies in the existence (real or just logical) of an XML metadata hierarchy acting as a catalogue for all the multimedia contents and their related

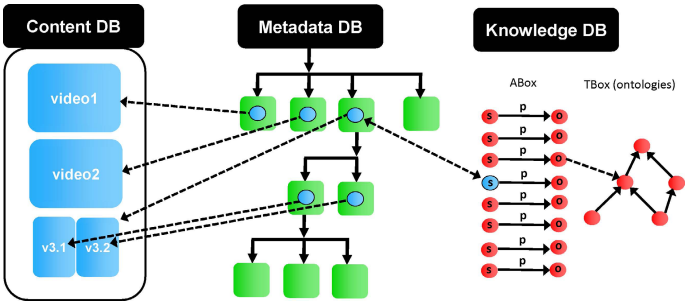

Fig. 7. MPQF model and semantic metadata

segments. Of course our approach does not try modifying this feature, as it is an essential part of MPQF, and our aim is just extending the language. A certain multimedia database could combine an XML metadata catalogue with a knowledge base composed by a *Tbox* and an *Abox*. Both kinds of metadata (XML catalogue and the KB) could be managed separately, and the binding between metadata descriptions could be the URIs of the media locators (see Figure 7).

3.5 A New Query Type: QueryBySPARQL

Complex conditions in MPQF are encapsulated within elements from the *QueryType*. This type is an abstract type which extends the *BooleanExpressionType* and is the parent class of the *QueryByMedia*, *QueryByDescription*, *QueryByFeatureRange*, *SpatialQuery*, *TemporalQuery*, *QueryByXQuery*, *QueryByFreeText*, and *QueryByRelevanceFeedback* types.

We suggest incorporating a new query type, the *QueryBySPARQL*, which will inherit from the *QueryType* and will act in a similar way as *QueryByXQuery* type. We suggest adding the declaration appearing in Code 4 within the MPQF XML schema.

Code 4. New *QueryBySPARQL* type declaration

```
<complexType name="QueryBySPARQL">
  <complexContent>
    <extension base="mpqf:QueryType">
      <sequence>
        <element name="SPARQLQuery" type="string"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Table 1 includes the proposed normative definition of the semantics of all the components of the new query type specification.

Code 5 shows an example MPQF input query with an SPARQL query embedded in it. The embedded query makes use of the *ASK* clause described in [14].

Table 1. Semantics of the new *QueryBySPARQL* type

Name	*Definition*
QueryBySPARQL	Extends the abstract *QueryType* type and denotes a query operation for the use of SPARQL expressions
SPARQLQuery	Specifies the SPARQL expression that is used to filter information. Because this query type only can return *true* or *false*, not all possible SPARQL espressions are allowed (no output generation is allowed in query types). The only valid SPARQL expressions te be used using the SPARQL *ASK* clause and the *?resource* variable appearing unbounded within the graph pattern. The variable *?resource* will be the reference to the evaluation item being processed. If the query pattern matches the RDF data stored in the database (or can be inferred from it) the result will be *true*. Otherwise it will be *false*.

This clause serves to test whether or not a query pattern has a solution. No information is returned about the possible query solutions, just whether or not a solution exists. Figure 8 shows graphically the behaviour of new *QueryBySPARQL* query type. The evaluation-item being processed, the one highlighted within the metadata XML tree, evaluates to *true* because there is a graph of triples matching the given pattern.

Code 5. Example query with a *QueryBySPARQL* condition

```
<MpegQuery mpqfID="someID">
  <Query>
    <Input>
      <QueryCondition>
        <Condition xsi:type="QueryBySPARQL">
          <XQuery>
                  <![CDATA[
                  PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
                  ASK     { ?resource dc:title "Barcelona . }
                  ]]>
          </XQuery>
        </Condition>
      </QueryCondition>
    </Input>
  </Query>
</MpegQuery>
```

3.6 RDF Triples in the Resultset

The ability to express SPARQL expressions within MPQF queries would not be very useful if one could not know which RDF triples are related to an item. Because the allowed SPARQL expressions are constrained to return *true* or *false*, another mechanism is required in order to allow users to query the RDF data in the database. Of course this could be done through an independent RDF query interface, but we believe that it could be interesting if MPQF queries could return also RDF metadata.

The suggested extension implies two different aspects, the MPQF Input Query Format and the MPQF Output Query Format. In one hand, the current MPQF

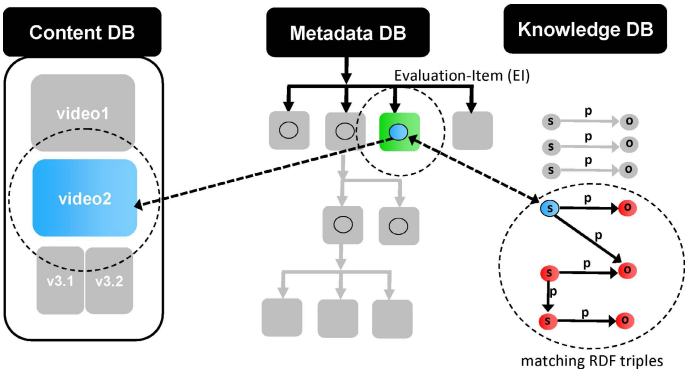

Fig. 8. Visual representation of the execution of the new *QueryBySPARQL* query type

Input Query Format allows specifying which is the desired data to be returned within the result records. This is the part of the Input Query Format called *output description*. Code 6 shows an example MPQF input query with its corresponding output description.

Code 6. Output description example

```
<MpegQuery mpqfID="someID">
 <Query>
  <Input>
    <OutputDescription freeTextUse="true" outputNameSpace="urn:mpeg:mpeg7:schema:2004" >
      <ReqField typeName="CreationInformationType">/Creation/Title</ReqField>
      <ReqField typeName="CreationInformationType">/Creation/Creator</ReqField>
      <ReqField typeName="MediaFormatType">/FileFormat</ReqField>
     </OutputDescription>
    <QueryCondition>
      <Condition xsi:type="QueryByFreeText">
        <FreeText>San Jose</FreeText>
      </Condition>
    </QueryCondition>
  </Input>
  </Query>
</MpegQuery>
```

Currently there are two different mechanisms for describing the output, some fixed attributes (*freeTextUse*, *mediaResourceUse*, *thumbnailUse* ,etc.) and user defined *ReqField* elements which carry an XPath expression. So, there are two possible solutions in order to allow describing RDF output. The simple one could consist on just adding a *RDFUse* optional attribute which would signal the database to return all RDF triples related to the resource. However, there could be also a more flexible solution which would consist on extending the semantics of the *ReqField* element in order to allow selective harvesting of RDF metadata.

On the other hand, it is also necessary to consider where the resulting RDF triples would be placed within an output query. Fortunately, the MPQF Output Query Format is very flexible in that sense, and it allows the inclusion of XML metadata from any format. Making use of the available XML serialization for RDF, we can embed the RDF triples within an MPQF resultset, as shown in Code 7.

Code 7. Example showing RDF data within an output MPQF query

```
<MpegQuery mpqfID="someID">
  <Query>
    <Output currPage="1" totalPages="1" expirationDate="2008-05-30T09:00:00">
      <ResultItem xsi:type="ResultItemType" recordNumber="1">
        <TextResult>Item 01</TextResult>
        <MediaResource>http://dmag.upf.es/mpqf/repository/item01.avi</MediaResource>
        <Description xmlns:dc="http://purl.org/dc/elements/1.1/"/>
          <dc:title>Barcelona</dc:title>
         <dc:creator>John Smith</dc:creator>
         <dc:format>image/jpeg</dc:format>
        </Description>
      </ResultItem>
    </Output>
  </Query>
</MpegQuery>
```

4 Related Work

As far as we know, this is the first work to apply the MPEG Query Format to semantic-driven search and retrieval of multimedia contents. The initial work related to MPQF can be found in 2004, as a cooperation between MPEG (ISO/IEC JTC 1/SC 29 WG11), its MPEG-7 standard and JPEG (ISO/IEC JTC 1/SC 29 WG1) and its JPSearch initiative. In July 2006, during the 77th MPEG meeting, the MPEG committee released a final Call for Proposal and specified a final set of requirements. MPQF reached the status of Final Draft International Standard (FDIS) after the 84th MPEG meeting, April 2008. It is expected that MPQF will become an ISO standard after the 85th MPEG meeting, July 2008.

Regarding other multimedia query formats, there exist several old languages explicitly for multimedia data such as SQL/MM [7], which have limitations in handling XML data and video contents (generally because of the date in which their design started). In the last years, these kind of languages have been usually based on MPEG-7 descriptors and the MPEG-7 data model. Some simply defend the use of XQuery or some extensions of it. Others define a more high-level and user-oriented approach. MPQF outperforms XQuery-based approaches like [6,16,3] because, while offering the same level of expressiveness, it offers multiple content-based search functionalities (QBE, query-by-freetext) and other IR-like features (e.g. paging or relevance feedback). Besides, XQuery does not provide means for querying multiple databases in one request and does not support multimodal or spatial/temporal queries.

Another novel feature of MPQF is its *metadata-neutrality*, which allows using it over metadata formats other than MPEG-7. However, as pointed in previous

sections, this *metadata-neutrality* is constrained by the fact that MPQF is currently limited to be used with XML-based metadata formats. Our work describe the necessary modifications which will allow MPQF to manage metadata modelled with Semantic Web languages like RDF and OWL, and query constructs based on SPARQL.

5 Conclusions and Future Work

The MPEG Query Format (MPQF) is one of the new standardization initiatives from the MPEG standardization committee, which aims providing a standardized interface to multimedia document repositories. In this paper we have described the necessary extensions which will allow MPQF to manage metadata modelled with Semantic Web languages like RDF and OWL, and query constructs based on SPARQL, an RDF query language. The described extensions include a conceptual generalization of the MPQF metadata processing model, the definition of a new MPQF query type (*QueryBySPARQL*), and a minor extension of the *output description* part of the MPQF's Input Query Format.

Currently there is an increasing interest in the advantages of having multimedia contents annotated with Semantic Web languages and bounded to multimedia ontologies. This evolution strongly impacts the way multimedia contents are searched and retrieved in a broad range of application scenarios. Our approach opens the possibility to use a modern standard multimedia query language like MPQF in combination with these new semantic metadata modelling technologies.

Acknowledgements

This work has been partly supported by the Spanish government (DRM-MM project, TSI 2005-05277) and the European Network of Excellence VISNET-II (IST-2005-2.41.6), funded under the European Commission IST 6th Framework Program.

References

1. Adistambha, K., Doeller, M., Tous, R., Gruhne, M., Sano, M., Tsinaraki, C., Christodoulakis, S., Yoon, K., Ritz, C., Burnett, I.: The MPEG-7 Query Format: A New Standard in Progress for Multimedia Query by Content. In: Proceedings of the 7th International IEEE Symposium on Communications and Information Technologies (ISCIT 2007), pp. 479–484, Sydney, Australia (2007)
2. The dublin core metadata initiative (dcmi), `http://hublincore.org/`
3. Fatemi, N., Khaled, O.A., Coray, G.: An xquery adaptation for mpeg-7 documents retrieval, `http://www.idealliance.org/papers/dx_xml03/papers/05-03-0105-03-01.html`

4. Gruhne, M., Tous, R., Döller, M., Delgado, J., Kosch, H.: MP7QF: An MPEG-7 Query Format. In: Proceedings of the 3rd International Conference on Automated Production of Cross Media Content for Multi-channel Distribution (AXMEDIS 2007), Barcelona, Spain, pp. 15–18 (2007)
5. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for very expressive description logics. J. of the Interest Group in Pure and Applied Logic (2000)
6. Kang, J., et al.: An xquery engine for digital library systems. In: 3rd ACM/IEEE-CS Joint Conference on Digital Libraries, Houston, Texas (May 2003)
7. Melton, J., Eisenberg, A.: SQL Multimedia Application packages (SQL/MM). ACM SIGMOD Record 30(4), 97–102 (2001)
8. Iso/iec 15938 version 2. information technology - multimedia content description interface (mpeg-7) (2004)
9. Owl web ontology language overview. w3c recommendation (February 10, 2004), http://www.w3.org/TR/owl-features/
10. Resource description framework, http://www.w3.org/RDF/
11. rdfdb query language, http://www.guha.com/rdfdb/query.html
12. Rdf vocabulary description language 1.0: Rdf schema, http://www.w3.org/TR/2003/WD-rdf-schema-20030123/
13. Rdql - a query language for rdf. w3c member submission (January 9, 2004), http://www.w3.org/Submission/RDQL/
14. Sparql query language for rdf. w3c working draft (October 12, 2004), http://www.w3.org/TR/rdf-sparql-query/
15. Inkling: Rdf query using squishql, http://swordfish.rdfweb.org/rdfquery/
16. Tjondronegoro, D., Chen, Y.P.P.: Content-based indexing and retrieval using mpeg-7 and xquery in video data management systems. World Wide Web: Internet and Web Information Systems, pp. 207–227 (2002)
17. Xquery 1.0: An xml query language. w3c proposed recommendation (November 21, 2006), http://www.w3.org/TR/xquery/