AN MPEG-7 VIDEO DATABASE SYSTEM FOR CONTENT-BASED MANAGEMENT AND RETRIEVAL

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÇİĞDEM ÇELİK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2005

Approval of the Graduate School of Natural and Applied Sciences,

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Ayşe Kiper
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Nihan Kesim Çiçekli
Supervisor

Examining Committee Members

Assoc. Prof. Dr. Ferda Nur Alpaslan	(METU)	
Assoc. Prof. Dr. Nihan Kesim Çiçekli	(METU)	
Assist. Prof. Dr. İlyas Çiçekli	(Bilkent Univ.)	
Assoc. Prof. Dr. Ahmet Coşar	(METU)	
Dr. Pınar Şenkul	(METU)	

ABSTRACT

AN MPEG-7 VIDEO DATABASE SYSTEM FOR CONTENT-BASED MANAGEMENT AND RETRIEVAL

Çiğdem Çelik
M.Sc., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Nihan Kesim Çiçekli

September 2005, 142 pages

A video data model that allows efficient and effective representation and querying of spatio-temporal properties of objects has been previously developed. The data model is focused on the semantic content of video streams. Objects, events, activities performed by objects are the main interests of the model. The model supports fuzzy spatial queries including querying spatial relationships between objects and querying the trajectories of objects. In this thesis, this work is used as a basis for the development of an XML-based video database system. This system is aimed to be compliant with the MPEG-7 Multimedia Description Schemes in order to obey a universal standard. The system is implemented using a native XML database management system. Query entrance facilities are enhanced via integrating an NLP interface.

Keywords: Video Database, MPEG-7, Native XML Database

ÖZ

KAPSAM TABANLI YÖNETİM VE ERİŞİM İÇİN MPEG-7 VIDEO VERİ TABANI SİSTEMİ

Çiğdem Çelik
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doçent. Dr. Nihan Kesim Çiçekli

Eylül 2005, 142 sayfa

Nesnelerin uzay-zamansal özelliklerini verimli ve etkin bir biçimde temsil eden ve sorgulayan bir video veri modeli daha önce geliştirilmişti. Bu model videonun anlamsal içeriği üzerine odaklıdır. Nesneler, olaylar ve nesneler tarafından gerçekleştirilen aktiviteler bu modelin ana konusunu oluşturmaktadır. Ayrıca nesnelerin uzaysal ilişkileri ve yörüngelerine yönelik sorgulamaları da desteklemektedir. Bu tezde bu modeli temel alarak XML tabanlı bir video veri tabanı sistemi geliştirilmiştir. Sistemin uluslararası bir standard olan MPEG-7 multimedya tanımlama şemalarına uygun geliştirilmiştir. Bu sistemin uygulaması yerli XML veri tabanı yönetim sistemi kullanılarak yapılmıştır. Geliştirilen bu sisteme sorgu girişini kolaylaştırmak amacıyla bir NLP arayüzü entegre edilmiştir.

Anahtar Kelimeler: Video Veri Tabanı, MPEG-7, Yerli XML Veri Tabanı

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

The increased usage of multimedia systems in our daily life has paved the way for querying multimedia and consequently video related data. The content of the video is the most considerable metadata subject to querying. The two main branches of video content are low-level features dealing with properties such as shape, color, motion and luminance values and the high level features addressing entities like persons and events. Entities like persons, trees, cars are called objects, whereas the flying of a bird or the crash of a car is of type event. The entities together with their spatial, temporal and spatio-temporal properties form the semantic content of a video.

A video database system should use a model that provides querying the semantic content of the videos. There are basically three approaches used in model development studies: the *annotation based* technique that uses user specified annotations to describe semantics, the *physical level video segmentation* approach that describes video as a stream of small segments with application specific temporal and spatial properties, and the *object-*

based modeling that adopts object-oriented programming concepts to define video semantics.

In this thesis, the aim is to develop a video database management system built on an object-oriented video data model. The main focus of the model is on semantic entities, such as objects (cars, buildings, persons, etc.), activities (eating, flying, etc.) and events (eating spaghetti by Ali). The spatio-temporal properties of objects, spatial relationships between objects, and temporal properties of activities and events are also considered. Queries dealing with object, activity, and event occurrences; spatial locations of objects, spatial relationships between two objects; and object trajectories are supported by the system. Also, the spatial, spatial relationship and spatio-temporal queries consider fuzzy behavior to cope up with the uncertainties originating from the spatial properties of objects.

Other than a video data model, the way the semantic content is stored arises as a critical decision. Having a reliable, efficient, extendible, and adaptable storage strategy is the key point to construct a rigid video database system that provides accurate search functionality. Moving Pictures Expert Group (MPEG) offers MPEG-7 standard [1] for the description of audiovisual content. The MPEG-7 framework consists of Descriptors (Ds), Description Schemes (DSs), a Description Definition Language (DDL), and coding schemes [2]. With the set of description tools it presents, MPEG-7 promises to allow applications to enable effective search, filtering, navigation and reuse of content [3].

The metadata descriptions of videos in this thesis are embedded into the determined XML schemes that are useful for search and retrieval processes. Moreover, the schemes are modeled to conform to the MPEG-7 standard. Rather than using partial descriptions, complete descriptions are adopted. Spatio-temporal decomposition of video with VideoSegment DSs and semantic entity descriptions with SemanticDesription DSs are studied separately and combined via defining relations with SegmentSemanticBaseRelation DSs.

The question that concerns any multimedia database system is how to store the data physically. Since XML files are determined to be the storage units for our study, there were actually three different choices. First one suggests storage directly on the file system. This type of usage is advantageous if a few files are subject to management. This is not a valid condition for our study. Also this choice does not provide any database management facilities for our application. Another approach addresses storing files as columns of a table in a relational database management system. This way of managing documents requires loading files to memory to manipulate the documents for querying purposes. We preferred to use the last choice that proposes native XML databases to meet storage and search requirements. Because, native XML databases are specialized for storing XML documents and provide useful facilities for managing these documents. In our study, all queries are achieved using an XML Query Language (XQuery) provided by the native XML database we use. The semantic content is divided into three different groups to perform each query type over only the related group of files. The collections in native XML databases are used as the means for this file classification.

Query interfaces are also important for any search system. People tend to use systems which can parse spoken language. In order to extract the query parameters from the sentences entered by the user, an existing NLP Query entrance interface [4] is integrated to the developed system.

1.2 Related Studies

AVIS (Advanced Video Information System) is an object-based video data model [5]. It models the semantic entities (object, activity, event) together with the temporal properties of these entities. The model utilizes a number of data structures to figure out the entities and their properties. The video is decomposed into segments that are stored in a frame segment tree structure. The nodes of the tree addresses frame sequences together with the objects and events that take place in the frame sequence. The system can answer only temporal queries including object, activity and event occurrence queries.

The study in [6] extends the model proposed by AVIS with spatio-temporal properties of object. The data structures in AVIS are modified to adapt for the newly added properties. The content of the nodes of the frame segment tree are remodeled to include object-region pairs instead of just objects. In this way the number of supported query types are increased with spatial, spatial relationship (e.g. left, right, top, bottom, etc.), and spatio-temporal (e.g. the trajectory of an object) queries. This extended system is called ST-AVIS, ST standing for spatio-temporal.

Our video database management system is based on ST-AVIS model. That is, the data structures proposed by the model are completely mapped to MPEG-7 XML schemas and each supported query type is implemented with XQuery. Support for multiple video querying is also added to this system.

OVID [7] is a content based object oriented model. The common descriptional data among objects are shared via *interval-inclusion based inheritance*. The study defines operations to composite video objects. VideoSQL that is an SQL based query language is introduced for query entrance.

BilVideo provides an integrated support for queries on spatio-temporal, semantic and low-level features (color, shape, and texture) on video data [8]. The spatio-temporal queries are handled using a knowledge-base, and the semantic queries are performed via an object-based relational database. All queries are handled by the native XML database in our study. The study in [8] does not use MPEG-7 standard. In addition to the graphical query interface a text-based SQL-like query language is available for users while we provide an NLP query interface.

In [9] an object-oriented video database system is introduced. The system is capable of accessing the video content via high level query-based retrieval. To cope up with the dynamic spatial and temporal properties of semantic entities, a versatile modeling technique is devised. The system uses an Object-oriented database engine (NeoAccess). A query language supporting spatial and temporal primitives is introduced to be used in the query interface.

There is a growing effort to adopt MPEG-7 descriptors for video metadata management. The systems that use MPEG-7 descriptors are introduced in the following.

PANORAMA [10] is a system for digitization, storage and retrieval of audiovisual information and its associated data. A branch of this study deals with developing a set of algorithms to extract visual features such as color and texture information, moving objects, faces and regions with text. Actually, the web-based system uses XML documents to exchange data among subsystems. Therefore, it uses MPEG-7 compliant XML schemas for communication purposes rather than using it as a storage format.

[11] introduces COSMOS-7 as an MPEG-7 compliant metadata modeling and filtering scheme for digital video. The scheme considers all of the semantic entities modeled in our study. It also introduces a content-based filtering strategy to extract user's preferred content. The output of the filtering operation might be the metadata or the frame sequences satisfying the filter expression. COSMOSIS is developed as a Java-based front-end to COSMOS-7. No special implementation detail emphasizing strategy for storing the XML documents is given.

AMOS [2] is a video object segmentation and retrieval system. It models the objects and their trajectories. The descriptions of visual features and spatio-temporal relationships are also included. AMOS also provides similarity retrieval. The study concerns only objects as opposed to our study that additionally models activities and events.

Yavuz [12] uses MPEG-7 Description Tools to build a video database management system based on the ST-AVIS model. This study adopts partial descriptions to construct a schema for the semantic content while our study concerns complete descriptions. Only one scheme is used and all information about a video is embedded into one file. We preferred to classify the content considering the execution of the query types we support. In [12] a relational database (SQL Server 2000) management system is used and the file is stored as *text large object type*. The queries are performed with XPath on the DOM (Document Object Model) tree loaded into the memory [12]. Our system executes XQuery expressions via the Native XML database. Lastly, [12] allows classification of videos according to their subject, such as news, sports, etc.

1.3 Contributions of the Thesis

The contributions of this study can be listed as follows:

1. Originating from the Spatio-Temporal Video Information System [6], that is capable of querying only one video at a time a video database management system that can query all videos has been developed.

2. A video semantic content description is modeled adopting Multimedia Description Schemes to ST-AVIS data model. Complete descriptions are used to have a more flexible design that can easily be adapted to contain new features.

3. Among the alternatives for storing XML tagged video metadata a native XML Database is determined. In order to provide each

query type to be performed over only the related files, the semantic content is distributed into three different collections, one for object related data, the other for event related data and the last is for frame segment tree.

4. XQuery is used to extract data that is required to process the queries.

5. A prototype system is implemented. Three applications are combined into one application group that work interactively with one another. A user interface for data and query entry is integrated with an NLP interface for parsing query sentences. This combination is adapted to communicate with our application that carries out actual database operations using Berkeley DB XML [13]. The communication is achieved by sending and receiving messages via network.

6. None of the systems introduced in 1.2 allow entering queries in natural language. The NLP interface embedded into the system has put forward a considerably important facility that extracts query parameters automatically from the sentence entered by the user. The NLP interface uses ontology to find words similar to the ones extracted from the query sentence. These words are also searched for to allow the user to access a larger set of results similar to the actual ones.

Considering the systems designed for video data management, this thesis study contributes the features listed above. To the best of our knowledge this study is unique in that it collects object-oriented video data model, spatio-temporal querying, MPEG-7 adoption with complete semantic descriptions, native XML database usage with XQuery, and NLP query interface into one complete system.

1.4 Scope

Chapter 2 explains XML and XML databases in detail. Chapter 3 presents MPEG-7 description tools addressing structural and semantic content definition. Chapter 4 explains how ST-AVIS entities and their spatio/temporal properties are expressed with MPEG-7 description tools. Chapter 5 explains the overall structure of the implemented video database management system. Chapter 6 concludes the thesis study pointing out future extensions.

CHAPTER 2

XML DATABASES

2.1 XML

XML [14] stands for eXtensible Markup Language. XML is a W3C recommendation for marking up data using tags [15]. XML was developed to overcome the shortcomings of HTML. In HTML the semantics and syntax of tags is fixed. As more and more functionality was added to HTML to account for the diverse needs of users of the Web, the language began to grow increasingly complex and unwieldy [14]. So a language like XML, which allows creation of unique data formats for specific applications, was inevitable to be built.

XML was originally developed as a way to mark up content but it also became an important as a data storage method and interchange format. As a data format the advantages of XML over others are [14]:

1. *Built in support for internationalization due to the fact that it utilizes unicode,*

2. *Platform independence (for instance, no need to worry about endianess),*

3. *Human readable format makes it easier for developers to locate and fix errors than with previous data storage formats,*

4. *Extensibility in a manner that allows developers to add extra information to a format without breaking applications that where based on older versions of the format,*

5. *Large number of off-the-shelf tools for processing XML documents already exists.*

2.1.1 Structuring XML

Since XML provides defining document structures specific to the application, there should be a means for defining the structure and constraints on the content model of XML documents. Document Type Definitions (DTD) and XML Schemas are two different ways to describe the format of a valid XML document.

2.1.1.1 Document Type Definitions(DTD)

DTDs are the original means of specifying the structure of an XML document. They are used to specify the order and occurrence of elements in an XML document. Below is an example DTD:

```
<!ELEMENT student (name, surname)>
<!ATTLIST student student_no CDATA>
<!ELEMENT name (#PCDATA)>
<!ELEMENT surname (#PCDATA)>
```

Figure 2.1 - A Sample DTD

In this specification, the student element has two child elements, name and surname; and an attribute, student_no. The two child elements and the attribute contain character data.

2.1.1.2 XML Data Reduced(XDR)

Some data types used in XML were not defined in DTD. Due to weakness of DTD, XDR was recommended as a potential XML schema standard. XDR was XML based and overcame some inadequacies related with DTD but it was eventually rejected. Below is the specification of the same scheme given in Figure 2.1:

```
<Schema        name="student_schema"        xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="name" dt:type=" string" />
  <ElementType name="surname" dt:type="string" />
  <AttributeType name="student_no" dt:type="ui1" />
  <ElementType name="student" order="seq">
    <element type="name" minOccurs="1" maxOccurs="1"/>
    <element type="surname" minOccurs="1" maxOccurs="1"/>
    <attribute type="student_no" />
  </ElementType>
</Schema>
```

Figure 2.2 - An Example XDR Schema

This schema specifies that the *name* and *surname* elements contain string type of content and the *student_no* attribute contains an unsigned integer as its content. It also specifies that the student element is composed of *name* and *surname* elements and a *student_no* attribute. The *name* and *surname* elements occurs exactly once in a *student* element.

2.1.1.3 XML Schema Definitions(XSD)

The W3C XML Scheme recommendation provides XSD as a means to define XML Schema. XSD supports more data types than XDR. XSD also supports the creation of custom data types, and object oriented programming concepts like inheritance and polymorphism. Example 2.3 is the specification of the same schema given in Figures 2.1 and 2.2:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema" >
        <element name="student">
        <complexType>
        <sequence>
                <element name="name" type="string"/>
                <element name="surname" type="string"/>
                </sequence>
                <attribute name="student_no">
                <simpleType>
                        <restriction base="string">
                        <pattern value="e\d{6}"/>
                        </restriction>
                </simpleType>
                </attribute>
        </complexType>
        </element>
</schema>
```

Figure 2.3 - An Example XSD Schema

This schema specifies *student* element as a complex type, namely it may have element type children. The *student* element is composed of a sequence of *name* and *surname* elements together with a *student_no* attribute. The *name* and *surname* elements and the *student_no* attribute have string type of content. Also the *student_no* element should match the regular expression that matches the letter "e" followed by 6 digits.

2.2 DATA STORAGE WITH XML

XML is used with databases. The most common reason to use XML with a database is to publish the data stored in the database as XML. For example a student registry system might return the academic record sheet of a student as an XML document. Similarly, data transfer to a database may be carried out by means of an XML document. For example, a university may submit the transcript of a student in XML format to a company. An application within an institute may extract the data (i.e. university, GPA) from the XML document and store it in the database. The most popular usage of XML is to model semi-structured data due to the extensibility it provides. If it is hard to predicate all kinds of data that may be required in the near future, it is impractical to determine and use a relational schema. Also, for large volumes of data, XML provides better storage, management and querying facilities.

In the previous examples XML documents are used in a data-centric way. Data-centric XML documents contain discrete pieces of regularly structured, fine-grained data. The example in Figure 2.4 shows the characteristic of a data-centric XML document:

```
<Student>
  <Name>Cigdem Celik</Name>
  <No>1164623</No>
</Student>
```

Figure 2.4 - An Example Data Centric XML Document

Another way of using XML documents is document-centric usage. Document-centric documents contain mixed content, larger-grained data and have irregular structure. Mixed content is a mixture of child and text elements like in the example in Figure 2.5:

```
<element>This element has <type>mixed</type> content</element>
```

Figure 2.5 - An Example Document Centric XML Document

Document-centric XML documents are mainly used with databases for managing and querying purposes. For example, the end user documentation for a thermal camera system may contain vast amount of data. Managing such kind of a document set with a classical database is critical. In such a case, document-centric usage may be utilized to freely extract new XML documents from fragments of existing documents and perform queries over these newly created documents in an XML query language.

Actually, using XML as a database is not always that advantageous. The very first argument against using XML as a database is that XML is an inefficient storage format. It is verbose, so data access will be slow due to the time spent for parsing. Also, the XML family lacks many of the technologies commonly found in modern databases, such as indexes, transactions, multi-user access, security, logging, referential integrity, triggers, and so on [16]. As a result, using XML for data storage and management issues requires a lot of coding. So it may be advantageous for small, single-user databases but impractical for large, multi-user applications that require real database facilities.

2.2.1 Using XML with Databases

Actually, there is no strict distinction between a data-centric and a document-centric XML document. For example, a data-centric, i.e. academic record sheet of a student, document may contain a description part as a large-grained, irregularly piece of data. Likewise, a document-centric, i.e. user documentation for a thermal camera system, document may contain release data in the form of regularly structured, fine-grained data. However, classifying the documents as data-centric and document centric is useful in the perspective of determining which type of a database to use. Databases are classified into two categories in the respect that they use XML. XML is used to:

1. To exchange data between a database and an application or between two databases:

   For example, an e-commerce Web service may return the properties of a product, which is stored in a database, as an XML document and this document may be used by an end-user application or database. The first part of this process, namely extracting data from database and constructing XML document, is called *publishing*. The reverse process, namely extracting data from an XML document and recording it to a database, is called *shredding*. If shredding and publishing are carried out by the built in capabilities of the database, the database is said to be XML-enabled.

2. To be directly stored in a database:

   For example, the end-user documentation of a product may be stored in the database. Native XML databases are developed to use XML in this fashion.

2.2.1.1 XML Enabled Databases

An XML-enabled database uses a data model, i.e. relational, other than XML. So the actual data model should be mapped onto XML data model and vice versa. The XML-enabled databases utilize a special application, which carries out mapping operations, either internal or external to the database engine.

> *For example, DB2 XML Extender, the XML Wrapper, and SQL/XML can all transfer data between XML documents and the DB2 database. XML Extender and XML Wrapper are external to the database engine, while SQL/XML support is integrated into the DB2 database engine itself [16].*

The main advantage of using an XML-enabled database is that it does not require data model redesign or application update. The existing data and applications are kept intact. A built in facility behaves as a glue application providing some sort of communication between XML document and the database.

Mapping Database Schema to an XML Schema

In order to transfer data between an XML document and an XML-enabled database, it is necessary to map the database schema to the XML schema (or vice versa).

Mapping is a many-to-many operation. You can map the data in the database to many different types of results. For example the academic record sheet of a student may be mapped to a report of the student's current status or to a report showing the status of all of the students.

Mappings are generally a design time issue. Mapping is an operation applied on element types, attribute and text. The physical structure, such as entities and encoding information, and logical structure, such as processing instructions and the order, in which the elements occur, are almost always omitted. This may cause the creation of a different file when the file is stored in a database and later reconstructed from the data in the database. This situation may be acceptable depending on the application.

Table-based and object-relational mappings are two types of mappings commonly used by the applications. Query languages are also accepted as a means for mapping since they are able to return results in XML format. In other words, they may define a one-directional mapping from the database to XML.

1. **Table Based Mapping:** The table-based mapping models XML documents as a single table or set of tables. The database is modeled as the XML Schema in Figure 2.1 viewing with the

perspective of table-based mapping. As seen, columns are modeled as elements. But depending on the application they can also be mapped to attributes. The application can also use a naming strategy for tables, elements and attributes. Also, products that use table-based mappings can optionally contain metadata for tables and columns either as attributes of each related table or column element or at the start of the document.

Table-based mapping is useful for serializing relational data, if the format of the XML document matches the schema in Figure 2.6. Figure 2.7 is an example for table-based mapping:

```
<database>
    <table1>
      <row1>
        <column1>...</column1>
        <column2>...</column2>
        ...
                    <column(n)>...</column(n)>
      </row1>
      <row2>
        ...
      </row2>
      ...
      <row(n)>
        ...
      </row(n)>
    </table1>
    <table2>
      ...
    </table2>
    ...
    <table(n)>
      ...
    </table(n)>
  </database>
```

Figure 2.6 - XML Schema Suitable for Table-Based Mapping

```
<database>
<students>
  <student description = “This is where student personal data is stored”>
    <name>Jonathan</name>
    <surname>Rice</surname>
    <student_no>111111</student_no>
  </student>
</students>
<courses>
  <course>
    <name>Statistics - I</name>
    <code>5550444</code>
  </course>
</courses>
</database>
```

Figure 2.7 - XML Document Created with Table-Based Mapping

The database in Figure 2.7 is composed of two tables, namely *students* and *courses*. The rows of the *students* table is tagged in between <*student*> and </*student*>. The same case holds for the *courses* table. Also each column of each table is tagged as an element.

2. **Object Relational Mapping:** The object-relational mapping is used by all XML-enabled relational databases and some middleware products [17]. Object-relational mapping views XML document as a tree of objects that are specific to the data in the document. Objects are mapped to tables, properties are mapped to columns. The relationships between objects are viewed as integrity constraints. The example in Figures 2.8 and 2.9 illustrates object-relational mapping:

```
<database>
  <students>
           <student>
                      <name>Jonathan</name>
                      <surname>Rice</surname>
                      <student_no>111111</student_no>
                      <course description = "course taken by the student">
                         <code>5550111</code>
                         <grade>AA</grade>
                      </course>
                      <course>
                         <code>5550222</code>
                         <grade>BA</grade>
                      </course>
           </student>
  </students>
</database>
```

Figure 2.8 - XML Document Created with Object-Relational Mapping

The student object contains two course objects representing the courses taken by the student. The student and the courses taken by the student may be stored in two different tables and the tabular representation of this mapping is as in Figure 2.9.

Student

Name	Surname	Student No
Jonathan	Rice	111111
...	...	...

TakesCourse

Student No	Code	Grade
111111	5550111	AA
111111	5550222	BA
...	...	...

Figure 2.9 - Tabular Representation of XML Dcoument

2.2.1.2 Managing XML Documents

There are two alternative ways of managing XML documents as data storage. The first alternative proposes keeping files in the file system or in a relational database. In the second alternative the files are stored in native XML databases. The former one causes limited XML functionality.

XML Documents in the File System

For small application that uses a small set of files it is fine to keep the files in the file system. The system commands can then be used to query, i.e. *grep* in UNIX, or modify the content of these documents. A simple transaction control may also be organized via using a version control system.

XML Document as a Column of a Table

Another alternative is keeping documents as BLOBs in a relational database. This is more advantageous than the former due to additional facilities provided by the database engine such as transactional control, security, multi-user access. In addition, many relational databases provide means for full-text, proximity, synonym and fuzzy searches. Some of these databases have XML-aware searching tools that provide XML Querying. Also elements and attributes may be indexed. If an index is applied on an item (element or attribute), the document is searched for all instances of the indexed element or attribute. The found instances are recorded into the index table together with the identifier of the document the instance is found in. Then the values of the items are indexed in order to provide fast access to a searched item and then the corresponding XML document. Figure 2.10 demonstrates this indexing strategy.

Suppose we have a set of documents with the following schema:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        <element name="student">
        <complexType>
                <sequence>
                  <element name="name" type="string"/>
                  <element name="surname" type="string"/>
                </sequence>
                <attribute name="student_no">
                <simpleType>
                  <restriction base="string">
                    <pattern value="e\d{6}"/>
                  </restriction>
                </simpleType>
                </attribute>
        </complexType>
        </element>
</schema>
```

If we want to index the element *student_no* we can store the indexes in the following tables:

1. StudentNumbers:

StudentNo	VARCHAR(50)
DocumentId	INTEGER

2. Documents:

DocumentId	INTEGER
Content	LONGVARCHAR

When a document is inserted to the database the application adds the document to the *Documents* table. It searches the content of the document for the attribute <*student_no*> and records the found values of the attribute and the *DocumentId* of the document to the *StudentNumbers* table. To find the contents of all the documents those contain the *student* with the *student_no* 111111 the application executes the following query:

```
SELECT Content
FROM Documents
WHERE DocumentId IN (SELECT DocumentId
                     FROM StudentNumbers
                          WHERE StudentNo ='111111')
```

Figure 2.10 - Example XML Element Indexing

Native XML Databases

A native XML database is one that [18]:

- *Defines a (logical) model for an XML document -- as opposed to the data in that document -- and stores and retrieves documents according to that model. At a minimum, the model must include elements, attributes, PCDATA, and document order. Examples of such models are the XPath data model, the XML Infoset, and the models implied by the DOM and the events in SAX 1.0.*

- *Has an XML document as its fundamental unit of (logical) storage, just as a relational database has a row in a table as its fundamental unit of (logical) storage.*

- *Is not required to have any particular underlying physical storage model. For example, it can be built on a relational, hierarchical, or object-oriented database, or use a proprietary storage format such as indexed, compressed files.*

Application Areas:

Native XML databases preserve issues such as document order, processing instructions, comments, CDATA sections, entity usage. So they are suitable for storing document-centric documents. Native XML databases support XML query languages. So they provide answers to queries like "Find all documents in which the first paragraph contains a bold word.” which are difficult with other query languages.

If the natural format of the documents that are used in an application is XML, it is a good choice to use a native XML. Think of an application that uses XML documents for communication. If these documents required to be queued, it is best to use a message queuing system built on a native XML database. This is because XML-specific capabilities (i.e. XML query languages) offered by a native XML database, will usually provide faster retrieval of whole messages.

Since native XML databases can accept, store, and understand any XML document without prior configuration, they present the most convenient solution for storing documents that do not conform to an XML Schema. Otherwise, transferring the data in an XML document to a relational or object-oriented database requires you to first create a mapping and a database schema [17].

Native XML databases are also used to store semi-structured data, and to increase retrieval speed in certain situations.

Features of Native XML Databases:

1. **Document Collections:** Most native XML databases allow users to logically group the documents into collections. This is analogous to a table in a relational database. For example, if the students of a class are stored in different XML documents then a collection might be defined to collect these documents under a group, so that queries related to these files can execute apart from other unrelated files. Collections may also be nested.

2. **Query Languages:** Native XML databases support one or more query languages. XPath and XQuery are the most popular ones. The capabilities of the query language play an important role in determining a database suitable for an application.

3. **Updates and Deletes:** There are different strategies for updating and deleting documents in a native XML database. The problem may be solved in a manner that just replaces or deletes the original document or that modifies the document through a live DOM tree.

   There are also languages that specify ways to modify the fragments of a document. Today two languages are available working in this fashion, namely XUpdate and XQuery. With XUpdate a set of nodes are defined with XPath, then the type of the operation is specified like deleting these nodes or inserting before or after these nodes. The W3C XQuery working group and Patrick Lehti proposed a set of extensions to XQuery for update facilities.

4. **Transactions, Locking, and Concurrency:** Native XML databases support transactions. But the level of locking for an XML document is a critical issue. When locking is at document level, multi-user concurrency decreases. Document-level locking might not be a matter if the users updating the same document at the same time. However, document-level locking may cause unacceptable delays in high-volume applications.

Node-level locking seems to be a solution for applications for which high multi-user concurrency is required. However, this is not the case yet, since implementing node-level locking is problematic. Usually locking a node requires locking its parent, and so on up to the root of the document. This condition is the same with locking the whole document. A partial solution to this problem is proposed by Stijn Dekeyser [19]. This solution offers annotating the locks with the path from the locked node to the target node. This allows other transactions to determine whether they may cause a confliction with any of the queries currently in process.

Node-level locking seems to be supported by most native XML databases in the future.

5. **Application Programming Interfaces (APIs):** Nearly all native XML databases offer programmatic APIs. These APIS generally support ODBC-like functionalities such as connecting to the database, executing the query and returning the results. Results are usually of type XML string, a DOM tree or an XMLReader over the returned XML document.

   The APIs are specific to the database but there is a tendency towards having a unique programming interface for native XML databases. XML:DB API and JSR 225: XQuery API for Java (XQJ) are being developed for this purpose.

6. **Round-Tripping:** Native XML databases are able to round-trip XML documents. That is, an XML document can be stored in a native XML database and retrieved back. This facility is vital for document-centric applications for which it is important to get the exact copy of the stored documents.

7. **Remote Data:** XML documents stored in a native XML database can contain remote data. This remote data is generally data retrieved from relational database using table-based or object-based mapping. Remote data may be live or not depending on the native XML database. The changes made on the live remote data are reflected in the remote database. Most native XML databases will probably support live remote data.

8. **Indexes:** Native XML databases support indexes. There are three types of indexing. Value indexes index text and attributes. Structural indexes index the location of elements and attributes. Full-text indexes index the individual tokens in the text. While only a few native XML databases support full-text indexes, value and structural indexes are supported by most of the native XML databases.

9. **External Entity Storage:** An external entity is any reference that points information stored outside the database. It may be the URL of a Web page or the chapters of a manual.

   External entity storage is currently an unsolved problem by the native XML databases. Specific to the application it may be desirable to keep the reference as it is or to expand the external entity and then store it. So the solution to this problem should be such a flexible one that allows choosing whether or not to expand external entity references.

2.2.2 Querying with XML

XPath [20] and XQuery [21] are the two most common XML Query languages.

2.2.2.1 XPath

XPath is an expression language used to select portions of an XML document. XPath expressions are just like regular expressions operating on XML nodes. Examples in Figure 2.11 are useful means for understanding XPath:

- **/a/b** Find all *b* elements that are children of element *a* which is the child of the root of the document.
- **//b** Find all *b* elements that appear at any depth (//) in the document.
- **count(//b)** Find the count of all *b* elements that appear anywhere in the document.
- **//video[title = “Brave Heart”]** Find all video elements, which have a child *title* element of value “Brave Heart”. A predicate surrounded with a bracket behaves as a filter over the results. These brackets can be nested
- **//video[@duration > 20]** Find all video elements, which have a duration attribute of value greater than 20. The @ sign marks the attributes.
- **//video[@duration]** Find all video elements, which have a duration attribute of any value.
- **//video/@duration** Find all duration attributes, which are attached to a video element.
- **(a | b)** Find all *a* or *b* child elements of the current context node.
- **//a[. = “avalue”]** Find all *a* elements that have a value “avalue”. “.” represents the current context node.
- **(//a)[1]/text()** Find the text nodes of the element a, which appears first in the document. [] is now used to specify the rank of the element.

Figure 2.11 - XPath Examples

2.2.2.2 XQuery

XQuery is a technology under development by the World Wide Web Consortium that's designed to query collections of XML data -- not just XML files, but anything that can appear as XML, including relational databases [22].

XPath expressions form the simple parts in an XQuery. In other words, XQuery is built on XPath extending it with F(or) L(et) W(here) O(order by) R(eturn) expressions.

XQuery Explained with Examples

The features of XQuery are explained with examples in the following.

Element Constructors: It is possible to create or generate elements in an XQuery expression by embedding element constructors into the query.

```
<video videoid = {$id}>
            {$title}
            {$duration}
</video>
```

Figure 2.12 - Example Element Constructor

The expression in Figure 2.12 generates a *video* element with a *videoid* attribute. *$id*, *$title* and *$duration* are variables that are assigned a value in the previous parts of the query and they compose the value of the *videoid* attribute and the content of the *video* element.

FLWOR Expressions: FLWOR (pronounced "flower") stands for For, Let, Where, Order by, and Return keywords. These keywords are the names of the clauses that form the basic building blocks of XQuery.

1. A FOR clause provides an iteration construct over the set of results returned by a query.

2. A LET clause works like assigning a single value to a single variable.

3. A WHERE clause contains a logical expression that filters the results returned by the preceding LET and FOR clauses.

4. The ORDER BY clause sorts the results according to the variable given with the clause.

5. The RETURN clause produces the output of the FLWOR expression.

As an example, consider the query in Figure 2.13.

```
FOR $student IN document("students.xml")//student
WHERE $student/name = "Ozge" OR $student/surname = "Celik"
RETURN $student/student_no
```

Figure 2.13 - Example FLOWR Expression

This query returns a list of numbers of students whose names are *Ozge* or surnames are *Celik*.

Conditional Expressions: A conditional expression works just like if statements in a programming language. It evaluates a logical expression and returns the corresponding result expression according to the evaluated value.

```
FOR $vid IN
          RETURN
                    <video>
                    {
                     $vid/title,
                     IF ($vid/@type = "Fiction")
                     THEN $vid/editor
                     ELSE $vid/director
                    }
           </video>
```

Figure 2.14 - Example Conditional Expression

The query in Figure 2.14 returns a list of videos. If the type of the video is fiction it includes the editor information. Otherwise it includes the director information.

Quantified Expressions: The SOME clause in XQuery is an existential quantifier that is satisfied if there is at least one node satisfying the related predicate. The EVERY clause is a universal quantifier used to test if all nodes in a series of values satisfy the related predicate. The following query in Figure 2.15 finds the titles of clips in which every object has a name that contains the word *man*.

```
FOR $c IN //clip
WHERE EVERY $obj IN $c//object SATISFIES contains($obj/name, "man")
RETURN $c/title
```

Figure 2.15 - Example Quantified Expression

Functions and Operators: XQuery supports a large set of functions and operators that provide math, string, regular expression, logical expression, XML node, sequence and input manipulation, type conversions, date and time comparisons. The most common ones are listed in Appendix A.

User Defined Functions: XQuery also allows users to define and use their own functions. Figure 2.16 illustrates a user defined function.

```
This example is directly taken from W3C Working Draft [23]
NAMESPACE xsd = http://www.w3.org/2001/XMLSchema
DEFINE FUNCTION depth($e) RETURNS xsd:integer
{
  # An empty element has depth 1
  IF (empty($e/*)) THEN 1
                    # Otherwise, add 1 to max depth of children
                       ELSE max(depth($e/*)) + 1
}
depth(document("partlist.xml"))
This function finds the maximum depth of the document named "partlist.xml."
```

Figure 2.16 - Example User-Defined Function

2.2.3 BERKELEY DB XML

Berkeley DB XML (BDB XML) is an open source, embedded XML database engine developed by Sleepycat Software. It is built on top of Berkeley DB, a "key-value" database which provides record storage and transaction management [24].

BDB XML is a C ++ library that links into the application. A programming API is available that provides managing, querying and modifying the documents. BDB XML supports multiple processes per application and multiple threads per process.

BDB XML provides command line tools to load, backup, dump and interact with the XML databases.

2.2.3.1 Languages and Platforms

Although written in C++, a BDB XML API is available for the languages C++, Java, Perl, Python, PHP, and Tcl. BDB XML also has third party bindings for Ruby and other popular languages.

BDB XML supports the platforms Windows, Linux, UNIX and other O/S.

2.2.3.2 XML Features

BDB XML conforms to the W3C standards for XML and XML Namespaces, XPath 2.0 and the XQuery 1.0 July 2004 draft. It additionally offers the following features related to XML data management and queries:

Containers (Collections): BDB XML store documents in containers. A container is a single file that contains one or more XML documents, together with their metadata and indices. Adding, deleting, and modifying documents and managing indices are carried out via containers. BDB XML allows operating on multiple containers at the same time.

Indices: BDB XML provides flexible indexing of XML nodes, elements, attributes and document meta-data. BDB XML offers a single mechanism for indexing, which indexes XML data according to four criteria, namely Uniqueness, Path Type, Node Type, and Key Type.

Indices are declared after a container is initialized and before any documents are added. In addition, multiple indices can be declared per container.

1. **Uniqueness:** If an item is indexed with uniqueness, it can occur only once in a container. By default, indices are not unique; uniqueness for an index should be explicitly declared.

2. **Path Type**: There are two types of path elements in the tree, a node type and an edge type. Node type is just the node itself. Edge type is any location where two nodes meet. If an XML document is viewed as a tree then there are also two types of path elements in an XML document. Path type *node* indicates that a node in the path is indexed. Path type *edge* indicates that the portion of the path where two nodes meet is indexed. BDB XML query processor uses edge-type indices because they are more specific than a node-type index.

3. **Node Type**: If an element is subject to indexing node type element should be used. Similarly attributes should be indexed via node type attribute.

4. **Key Type**: When testing the value of the element against a provided value (//a [b='value b']), equality index is recommended.

If testing is for the existence of an element, presence index is useful, and if testing using the XPath contains () function (//a [contains (b, 'value')]), substring index should be used.

Query Optimization: BDB XML implementation of XQuery uses a cost-based query optimizer. Pre-compiling a query can triple the performance.

Query results: BDB XML retrieves results that match a given XQuery query either as a set of values or as an XML document.

Storage: BDB XML can store documents as whole documents or as document nodes. In the former one the documents are kept in their entirety. In the second one the document is broken down into nodes and each resulting node is kept as a different record in the container. Considering the query performance, storing the whole document is best for small documents, while the other is shown to be ideal for XML documents with size greater than 1 MB.

BDB XML preserves the encoding and keeps the documents in their native format.

Metadata attribute support: Recording metadata associated with a document is useful since it provides keeping data that do not conform the schema of the document. Documents in BDB XML can have metadata attributes.

Document modification: BDB XML supports efficient updates to document fragments. BDB XML's robust document modification facility allows easily adding and deleting documents, and modifying selected portions of documents [25].

2.2.3.3 Database Features

BDB XML inherits several database features from Berkeley DB. The most important ones are listed below:

In-process Data Access: BDB XML is a library running in the same process space as the application using it. So BDB XML is better than other traditional client/server-based database implementations, which cause IPC-overhead.

Database Environment Support: BDB XML environments support:

1. multiple databases,
2. transactions,
3. deadlock detection,
4. lock and page control, and
5. encryption.

Atomic Operations: A series of read and write operations to BDB XML can be compiled into a single atomic operation as a transaction.

Isolated Operations: Operations performed inside a transaction see all XML documents as if no other transactions are currently operating on them [25].

Recoverability: Data saved inside BDB XML's transactions are guaranteed to be fully available even if the application system fails.

Concurrent Access: More than one thread or process can read or write XML data set in parallel via employed isolation and deadlock handling mechanisms supplied by BDB XML.

CHAPTER 3

MPEG-7 MULTIMEDIA DESCRIPTION SCHEMES

3.1 MPEG-7

In recent years, as the amount of multimedia increased, the attempt to develop technologies to manipulate audiovisual data has also increased. One of the crucial points subject to standardization is to extract useful information from multimedia and to apply it in a search system. To have a common protocol for defining and using multimedia data is important due to scalability, intelligence and interoperability it will provide for multimedia applications. MPEG-7 [26] has improved a set of description tools considering different sights of multimedia at different levels of abstraction.

MPEG-7 standard deals with a wide range of multimedia properties and has tools to describe:

1. visual features (e.g., color),
2. audio features (e.g., timbre),
3. structure (e.g., moving regions and video segments),
4. semantics (e.g., objects and events),

5. management (e.g., creator and format),
6. collection organization (e.g., collections and models),
7. summaries (e.g., hierarchies of key frames) and,
8. user preferences (e.g., for search) of multimedia.

The ones describing the structure and semantics of multimedia form the main concern of this thesis and they will be examined in section 3.3.2.

The MPEG-7 standard also known as "Multimedia Content Description Interface" aims at providing standardized core technologies allowing description of audiovisual data content in multimedia environments [27]. This aim is really a hard task to achieve, considering the variety of requirements and multimedia applications, and the broad number of crucial audiovisual features. In order to achieve this broad goal, MPEG-7 aims to standardize [28]:

1. *Descriptors (D): representations of features, that define the syntax and the semantics of each feature representation,*
2. *Description Schemes (DS), that specify the structure and semantics of the relationships between their components, which may be both D's and DS's,*
3. *A Description Definition Language (DDL), to allow the creation of new DS's and, possibly, D's and to allows the extension and modification of existing DS's,*
4. *System tools, to support multiplexing of description, synchronization issues, transmission mechanisms, file format, etc.*

The standard is subdivided into seven parts [28]:

1. ***Systems**: Architecture of the standard, tools that are needed to prepare MPEG-7 Descriptions for efficient transport and storage, and to allow synchronization between content and descriptions. Also tools related to managing and protecting intellectual property.*

2. ***Description Definition Language**: Language for defining new DSs and perhaps eventually also for new Ds, binary representation of DDL expressions.*

3. ***Visual:** Visual elements (Ds and DSs).*

4. ***Audio**: Audio elements (Ds and DSs).*

5. ***Multimedia Description Schemes:** Elements (Ds and DSs) that are generic, i.e. neither purely visual nor purely audio.*

6. ***Reference Software**: Software implementation of relevant parts of the MPEG-7 Standard.*

7. ***Conformance:** Guidelines and procedures for testing conformance of MPEG-7 implementations.*

Parts 2 and 5 will be examined within the scope of thesis in sections 3.2 and 3.3.

3.2 DATA DESCRIPTION LANGUAGE

The MPEG-7 standard has a central component, namely the Description Definition Language (DDL). DDL enables users to compose their own

Description Schemes (DSs) and Descriptors (Ds) by providing them with a sound descriptive basis. The expression, combination, extension and refinement of DSs and Ds are carried out by syntactic rules, which are defined by DDL.

XML is used as the syntax for the MPEG-7 DDL. Due to the fact that XML Schema language was not designed particularly for audiovisual content, certain extensions are indispensable if all the MPEG-7 DDL requirements must be met. Therefore, DDL consists of the below components whose descriptions:

1. XML Schema structural components;
2. XML Schema datatypes;
3. MPEG-7-specific extensions: XML Schema is extended to satisfy the MPEG-7 DDL requirements such as:
    a. array and matrix datatypes,
    b. typed references,
    c. built-in derived datatypes such as enumerated datatypes for MimeType, CountryCode, RegionCode, Character-SetCode, and CurrencyCode and a set of additional timedatatypes.

For complete specifications, MPEG-7 DDL Committee Draft [29] and the W3C XML Schema Candidate Recommendations [30] can be consulted.

3.3 MULTIMEDIA DESCRIPTION TOOLS

3.3.1 Overview

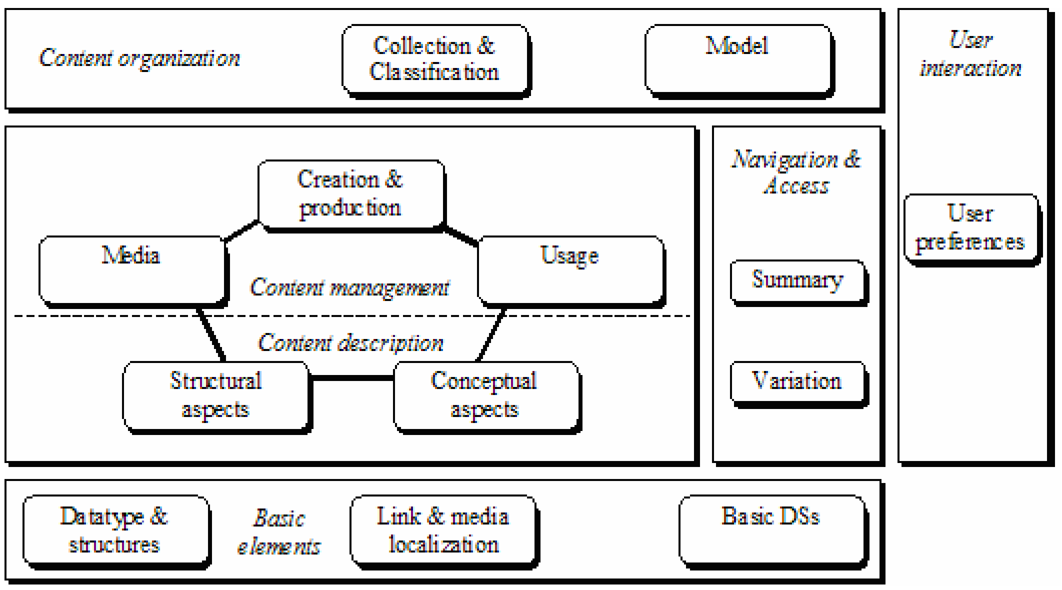


Figure 3.1 - Overview of MDSs

Figure 3.1 taken from [27] overviews the Multimedia Description Schemes (MDSs). *Basic elements* are placed at the lower level in the Multimedia Description Scheme (MDS) organization. Their work area consists of basic datatypes, mathematical structures, and tools for linking and media localization along with basic DSs, from which more complex DSs are formed. *Content management and description elements,* which give descriptions of the content from several viewpoints, stem from this lower level. There are three elements (Creation and Production, Media, Usage) for identifying information about the content management. On the other hand, the Structural and Conceptual aspects describe the perceivable *content*. A more precise definition for each set of these elements is listed in the Table 3.1 which is adopted from [28]:

The five sets of elements are shown as separate entities. However, there are interrelations between them and they may partially include the one another. To illustrate, the structural content description may involve individual segments where Media, Usage or Creation & Production elements are attached. Depending on the application, different combinations of the areas of the content description may be chosen.

In addition to the direct content description obtained from the five sets of elements mentioned in Table 3.1, there are tools defined for navigation and access. The summary elements provide browsing. Information on probable content variations is supplied as well. In order to adapt different multimedia presentations to the facilities of the client terminals, network conditions or user preferences, the original Audio Visual (AV) content can be replaced by variations.

Content Organization is the set of tools for describing content organization via classification, definition of collections and modeling.

The final set of tools collected in *User Interaction*, are specified to denote user choices for the utilization of multimedia material.

Table 3.1 - Types of Multimedia Meta Data

Elements	**Functionality**
Creation & Production	Meta information describing the creation and production of the content: typical features include title, creator, classification, purpose of the creation, etc. This information is most of the time author generated since it cannot be extracted from the content.
Usage	Meta information related to the usage of the content: typical features involve rights holders, access right, publication, and financial information. This information may very likely be subject to change during the lifetime of the Audio Visual (AV) content.
Media	Description of the storage media: typical features include the storage format, the encoding of the AV content, elements for the identification of the media. Note that several instances of storage media for the same AV content can be described.
Structural aspects	Description of the AV content from the viewpoint of its structure: the description is structured around segments that represent physical spatial, temporal or spatio-temporal components of the AV content. Each segment may be described by signal-based features (color, texture, shape, motion, audio features) and some elementary semantic information.
Conceptual aspects	Description of the AV content from the viewpoint of its conceptual notions. (Note that currently this part of the MDS is still under Core Experiment and no elements are included in the XM [27] or WD [28]).

3.3.2 Content Description Tools

Content description tools describe the structure and semantics of multimedia data [31]. The hierarchical view of content description tools are given in Figure 3.2 [32].

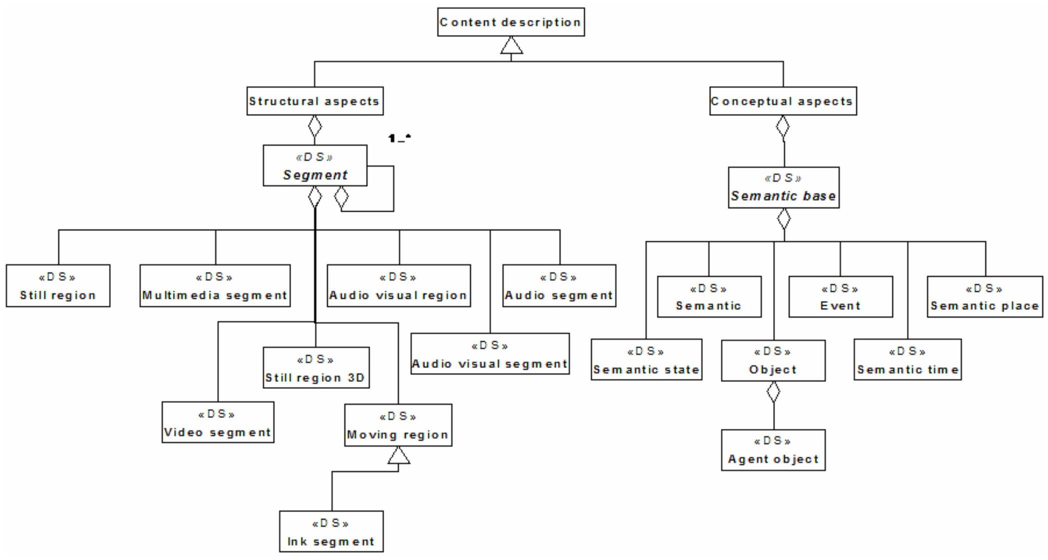


Figure 3.2 - MPEG7 Multimedia DS - content description

The structural and semantic tools are examined separately in sections 3.3.2.1 and 3.3.2.2.

3.3.2.1 Structural MPEG-7 Description Tools

MPEG-7 has description tools for describing the structure of the multimedia data in space, time, and media source. Structural description of multimedia data consists of general or application specific segments and the corresponding attributes, hierarchical decomposition, and relations.

Still Region is the most fundamental visual segment. It represents a group of pixels in an image of a frame in a video. In Figure 3.3 three still regions are illustrated. The first one (SR1) corresponds to a full image, while the other two, SR2 and SR3, show the regions where two object reside in the video frame. The description of StillRegion DS is given in Table 3.2 adopted from [33].

One other visual segment is the *Video Segment* that describes a sequence of frames. The definition of elements of VideoSegment DS is given in Table 3.3 adopted from [33].

The *Moving Region* is yet another visual segment that is a set of pixels in a group of frames in a video.

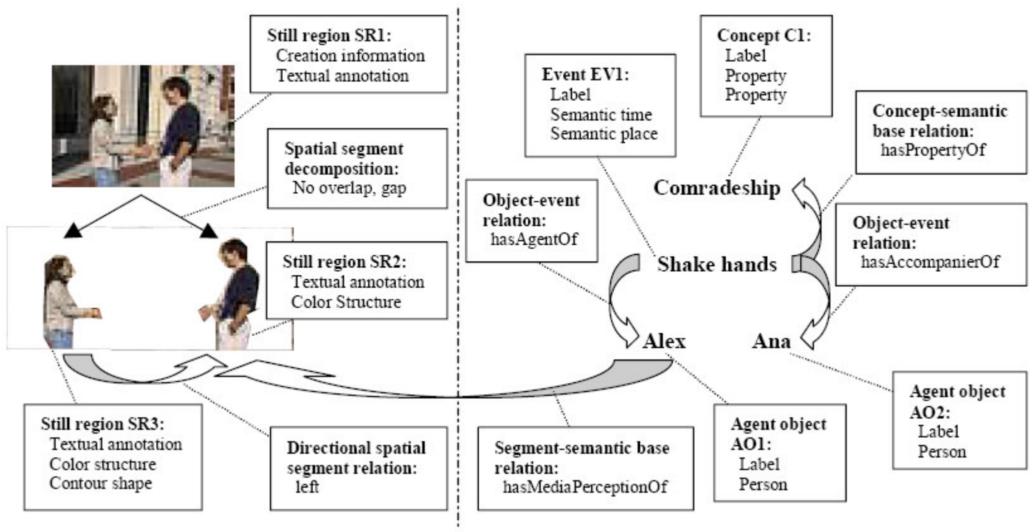


Figure 3.3 - Example Multimedia Description

The visual and audio features of the multimedia data can be represented by the visual and audio Ds and DSs. Also, media, creation and usage information represented by segments can be described using media, creation and usage tools, respectively. *Segment attribute* description tools are designed to define segment related attributes such as spatio-temporal, media, and graph masks; importance for matching and point of view; creation and media of ink segments; and hand writing recognition.

The *segment decomposition* tools specify the decomposition of multimedia data into a hierarchy of segments. There are four types of decompositions defined by MPEG-7, namely spatial, temporal, spatio-temporal and media resource decompositions. Using spatial decomposition, a frame in a video can be divided into still regions depicting the regions of objects in concern.

The still regions can then be decomposed into other still regions. The example in Figure 3.3 adopted from [31] illustrates the spatial decomposition of spatial region SR1 into other still regions, SR2 and SR3. Similarly temporal, and spatiotemporal decompositions can be applied to multimedia data.

The role of media source decomposition is to provide a means to break segments into their media components such as audio and video tracks. The resulting segments may overlap in time, space, and/or media. Also the union of them may have gaps compared to the original segment. Figure 3.3 illustrates a spatial decomposition that has gaps but no overlaps.

In order to designate the common relations among segments, the *segment relation* description tools are developed. The spatial relation *left* between the still regions SR2 and SR3 is shown in Figure 3.3.

Table 3.2 - StillRegion DS

Name	**Definition**
StillRegionType	Describes 2d spatial region of an image or video frame, which can correspond to an arbitrary set of pixels, a single pixel, or even the full image or video frame. The still region does not need to be connected in space.
SpatialLocator	Describes the spatial localization of the still region (optional).
SpatialMask	Describes the composition of spatially non-overlapping, connected components that form a non-connected still region (optional). If absent, the segment is composed of the single connected region in space defined by the SpatialLocator element. If present, the segment refers to the set of non-overlapping sub-regions in space defined by the SpatialMask element.
MediaTimePoint	Indicates the time point of the still region when it belongs to a video using an element of type mediaTimePointType (optional).
MediaRelTimePoint	Indicates the time point of the still region when it belongs to a video frame in a video using an element of type MediaRelTimePointType (optional).
MediaRelIncrTimePoint	Indicates the time point of the still region when it belongs to a video frame in a video frame using an element of type MediaRelIncrTimePointType (optional).
Visual Descriptor	Describes a visual feature of the still region using a visual descriptor (optional).
VisualDescriptionScheme	Describes complex visual features of the still region using a visual description scheme (optional).
GridLayoutDescriptors	Describes visual features of the sub-regions resulting from a grid decomposition of the still region (optional). GridLayoutDescriptors descriptions only apply to rectangular still regions.
SpatialDecomposition	Describes a spatial decomposition of the still region into one or more sub-segments (optional).

MPEG-7 defines a set of directional (e.g. right), topological (e.g. touches), temporal (e.g. before, after, during), spatio-temporal (e.g. union, intersection) and additional (e.g. annotates) relations. A classified list of relation types are given in the Table 3.4:

Table 3.3 - VideoSegment DS

Name	**Definition**
VideoSegmentType	Describes a temporal interval or segment of video data, which can correspond to an arbitrary sequence of frames, a single frame, or even the full video sequence.
MediaTime	Describes the temporal localization of the video segment by specifying the start time and the duration of the video segment. If the video segment is non-connected, the duration should be equal to the duration of the smallest connected temporal interval that includes the video segment. For example, if a video segment is composed of two connected components of duration 2 and 3 seconds and there is a gap of 1 second between them, then, the duration of the smallest connected temporal interval that includes the video segment is 6 seconds. MediaTime is optionally described in cases in which the video segment refers to an entire video.
TemporalMask	Describes the composition of temporally non-overlapping, connected components that form a non-connected video segment (optional). If absent, the segment is composed of the single connected interval in time defined by the MediaTime element. If present, the segment refers to the set of non-overlapping sub-intervals in time defined by the TemporalMask element.
VisualDescriptor	Describes a visual feature of the video segment using a visual descriptor (optional).
VisualDescriptionScheme	Describes complete visual features of the video segment using a visual descriptor (optional).
TimeSeriesDescriptors	Describe a temporal sequence of visual features in the video segment (optional). TimeSeriesDescriptors descriptions only apply to connected video segments.
Mosaic	Describes a mosaic of the video segment (optional).
SpatialDecomposition	Describes a spatial decomposition of the video segment into one or more sub-segments (optional).
TemporalDecomposition	Describes a temporal decomposition of the video segment into one or more sub-segments (optional).
SpatioTemporalDecoposition	Describes a spatio-temporal decomposition of the video segment into one or more sub-segments (optional).
MediaSourceDecoposition	Describes a media source decomposition of the video segment into one or more sub-segments (optional).

Table 3.4 - Classification framework for segment relationships

Types of relationships			**Examples**
Structural	Spatial	Topological	▪ Adjacent to, Overlap, Contained in, Composed of, Consist of
			▪ The union, The intersection, The negation
		Metric	▪ Near from, Far from
			▪ R in Theta-R Graph, 0.5 inches from
		Directional	▪ Left of, Top of, Upper left of, Lower right of. Behind ▪ 2D-String's spatial relationships
			▪ 20 degrees north from, 40 degrees east from, the union of two segments ▪ Theta and R in Theta-R Graph
	Temporal	Topological	▪ Co-begin, Co-End, Parallel, Sequential, Overlap, Adjacent, Within, Composed, Consist of ▪ SMIL's <seq> and <par>
			▪ The union, The intersection, The negation ▪ SMIL's <seq> and <par> with attributes (start time, end time, MediaDuration)
		Metric	▪ Near from, Far from
			▪ 20 min. apart from, 20 sec. overlap
		Directional	▪ Before, After
			▪ 20 min. after
	Visual	Global	▪ Smoother than, Darker than, More yellow than, Similar texture, Similar Color, Similar speed
			▪ Distance in texture feature, Distance in color histogram ▪ Indexing hierarchy based on color histogram
		Local	▪ Faster than, To grow slower than, Similar speed, Similar shape
			▪ 20 miles/hour faster than, Grow 4 inches/sec. faster than ▪ Indexing hierarchy based on local motion, deformation features
		Composition	More symmetric than,
			Distance in symmetry feature Indexing hierarchy based on symmetry feature

3.3.2.2 Semantic MPEG-7 Description Tools

The semantic descriptions take part in a narrative world, which refers to the world depicted in the multimedia data such as the context of a movie. Objects, agent objects, events, concepts, semantic states, semantic places, and semantic times, together with their attributes and relations constitute the semantic entities in multimedia data. The perceivable items that occupy time and space in the narrative world are mapped to objects. Events are another group of entities understood as occasions upon which something happens [33]. Agent objects refer to objects that are persons, a group of persons, or organizations. Figure 3.3 shows an event, EV1 (Shake hands), and two agent objects AO1 (Alex), and AO2 (Ana).

The semantic entities of the narrative world can possess properties and they can pass through states. Semantic states are parametric attributes of semantic entities and semantic relations at a specific point in time and space (i.e., weight and height of person) [31]. Also, they may have relations with semantic and other entities. A narrative world is illustrated in Figure 3.4 adopted from [34]. This world is built upon an image that contains two persons, an event, a place, a time and a concept. The relationships between these entities are also illustrated.

To sum up, entities in narrative worlds, their attributes and their relations constitute the semantic content of multimedia. The MPEG-7 tools describing these constituents are reviewed in the following sections.

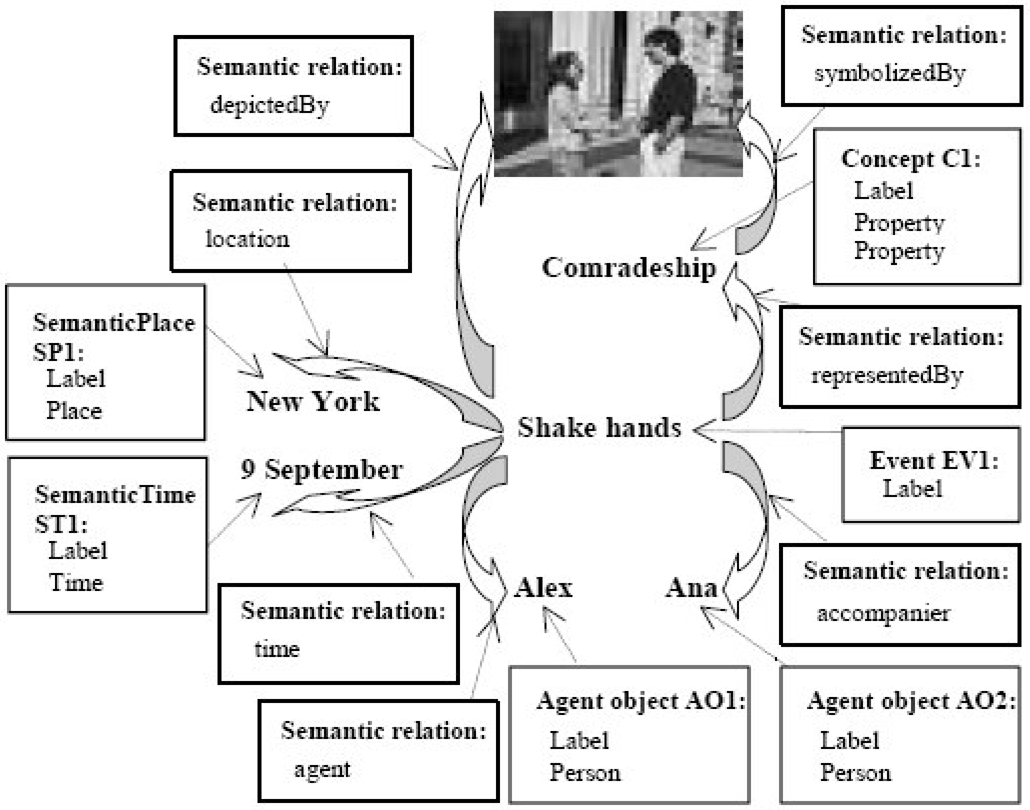

Semantic relation: depictedBy
Semantic relation: symbolizedBy
Concept C1: Label Property Property
Semantic relation: location
Comradeship
SemanticPlace SP1: Label Place
Semantic relation: representedBy
New York
Shake hands
Event EV1: Label
9 September
SemanticTime ST1: Label Time
Alex
Ana
Semantic relation: accompanier
Semantic relation: time
Semantic relation: agent
Agent object AO1: Label Person
Agent object AO2: Label Person

Figure 3.4 - Example Narrative World

The Abstraction Model

Abstraction in logic replaces one or more of the constant expressions in a statement by a variable [34]. That is, abstraction replaces an instance with a more general class the instance belongs to. Since abstraction forms a means to perceive the world, it is appropriate to use it to develop descriptive techniques for describing the narrative world in multimedia. In MPEG-7, abstract properties can be represented without abstraction, but media and formal abstractions are also considered. A media abstraction is a description that has been drawn from a specific entity and is capable of describing similar instances of multimedia. The description of the image in Figure 3.4 is "Alex is shaking hands with Ana in New

York on the 9th of September". This description can be a media abstraction and can be used to describe the same event in another media. To clarify, the variable in media abstraction is the media itself.

A formal abstraction is a description that represents a pattern common to a set of examples. The pattern contains placeholders or variables that may be substituted by those that are common to the set. The description is called formal since it contains variables. When these variables are filled in, the result should be media abstractions or concrete descriptions. "Alex is shaking hands with any woman in New York on the 9th of September", is a formal abstraction for the example in Figure 3.4. The variable in this abstraction is "any woman" and when filled with "Ana" it forms the aforementioned media abstraction "Alex is shaking hands with Ana in New York on the 9th of September".

MPEG-7 also allows description of properties and concepts. The property "Hardness" can be an attribute of the object "Rock". Concepts are abstract entities that are not directly perceived but used to name a group of properties. In a sense, they can be defined as the generalization of perceivable semantic entities. "Comradeship" (C1) in Figure 3.4 is a concept. Concept DS is specialized to describe relationships of a property, or allow multiple semantic entities to be related to a single property, or specify the strength of a property. The property or group of properties must be described as a concept [34].

Semantic Entities

The Semantic DS and its components are shown in Figure 3.5 taken from [34]. The MPEG-7 semantic entity tools describe narrative worlds and semantic entities such as objects, events, concepts, states, places and times [34].

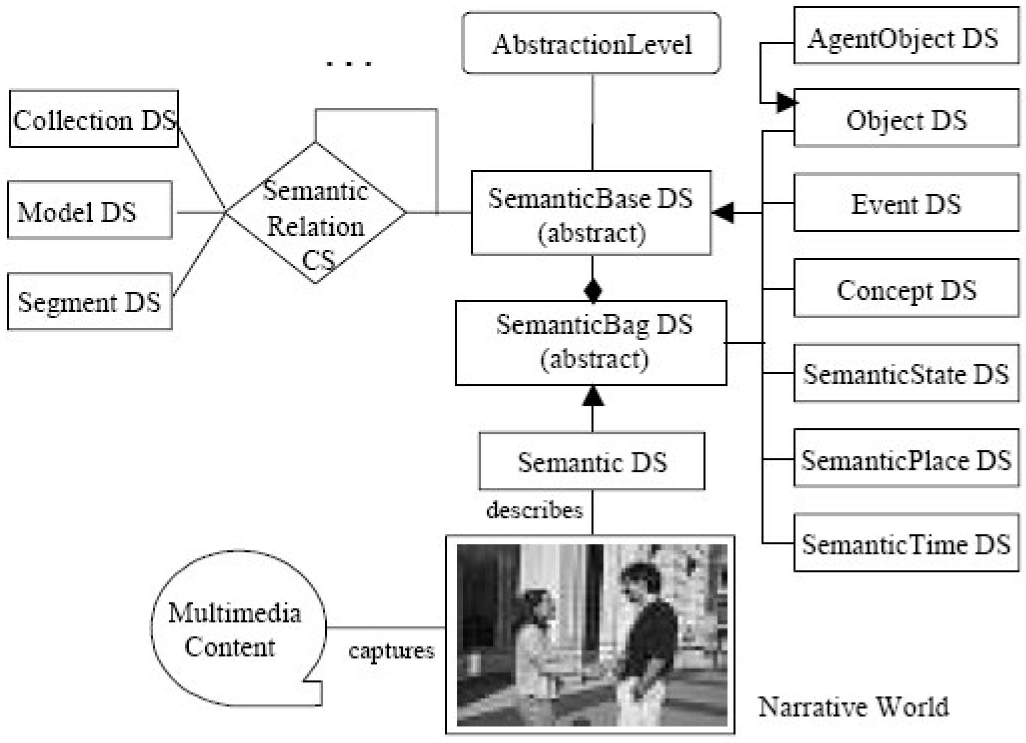


Figure 3.5 - Semantic DS and Components

The SemanticBase DS forms the basis for semantic entity tools. SemanticBase DS is an abstract type that represents any semantic entity including a narrative world. Different semantic tools provide a way to describe different status or functionality related to the same semantic entity in the narrative world.

The Semantic DS is derived from another abstract type the SemanticBag DS, which is used for the representation of any kind of collection of semantic entities and their relationships.

The Semantic DS is specialized to form other DSs, namely Object, AgentObject, Event, SemanticPlace, SemanticTime, SemanticState and Concept DSs. Object DS and Event DS describe objects and events present in the narrative world. Trivially they describe the semantic entities described in part 4.3.2.2.1, namely objects, agent objects, events and semantic place, time, state, and concepts.

Object and Event DSs allow recursive definitions of objects and events.

Semantic Attributes

MPEG-7 can describe semantics entity labels (Label), by a textual definition (Definition), or in terms of properties (Property) or of features of the media segments where they occur (MediaOccurence). These and other semantic attributes are given in Table 3.5 which is adopted from [33].

The content of derived semantic tools, Object DS, AgentObject DS, Event DS are given in Tables 3.6, 3.7 and 3.8 (all adopted from [33]) respectively.

Semantic Relations

MPEG-7 has standardized common semantic relations such as agent and time but it allows the description of non-normative relations too. The normative semantic relations in MPEG-7 are listed in Table 3.9.

The Semantic Relation description tools present means for defining the common relations among semantic entities and other entities. The semantic relation tools include the SemanticRelation CS, which specifies semantic relations that apply to entities that have semantic information. Normative semantic relations may describe how several semantic entities relate in a narrative or story. The element definitions of SemanticRelation CS are given in Table 3.10 which is adopted from [33].

Currently defined semantic relations include [31]:

1. relations among semantic entities,
2. relations among semantic entities and semantic relations,
3. relations among semantic entities and segment, and relations among semantic entities and models.

The relations classified according to their types are listed in Table 3.9. The inverse of these relations also exist.

Table 3.5 - SematicBase DS

Name	**Definition**
SemanticBaseType	Describes a semantic entity
AbstractionLevel	Indicates the kind of abstraction performed in the description of the semantic entity (optional).
Label	Identifies the type of the semantic entity.
Definition	Defines the semantic entity (optional).
Property	Describes a quality or adjectival property associated with the semantic entity (optional).
MediaOccurence	Describes an appearance of the semantic entity in the media (optional).
MediaLocator	Locates the media in which the semantic entity appears.
TemporalMask	Describes the temporal intervals of the media in which the semantic entity appears (optional).
SpatialMask	Describes the spatial intervals of the media in which the semantic entity appears (optional).
SpatioTemporalMask	Describes the spatiotemporal intervals of the media in which the semantic entity appears (optional).
Relation	Describes a relation between the semantic entity and other content description entities such as still regions, objects events, and models, among others (optional).
Type	Indicates the type of media occurrence. The types of media occurrences are defined as follows: Perceivable – the semantic entity is perceivable in the media. For example, Bill Clinton is perceivable in a picture of him. Reference – the semantic entity is a reference in the media. For example, Bill Clinton is a reference in a news reports about him but where he cannot be seen or heard. Symbol – The semantic entity is symbolized in the media. For example, freedom is a symbol in a picture of the Statue of Liberty. The attribute value is “perceivable” by default.

Table 3.6 - Object DS

Name	Definition
ObjectType	Describes a semantic object that exists in the narrative world with temporal and spatial extent (perceivable object, e.g., Tom's piano) or an abstraction of a perceivable object (abstract object, e.g., any piano)
Object	Describes one object resulting from the decomposition of the parent object (optional).
ObjectRef	References an existing description of an object resulting from the decomposition of the parent object (optional).

Table 3.7 - AgentObject DS

Name	Definition
AgentObjectType	Describes an object that is an agent – a person, an organization, or a group of people in a narrative world.
Agent	Describes the agent represented by the object (optional).
AgentRef	References the existing description of the agent represented by the object (optional).

Table 3.8 - Event DS

Name	Definition
EventType	Describes a dynamic relation involving one or more objects occurring in a region in time and space of a narrative world (perceivable event, Tom playing the piano) or an abstraction of a perceivable event (abstract event, e.g., anyone playing the piano). The place where an event takes place can be described by the SemanticPlace DS and Event DS or a semantic relation locationOf to the SemanticPlace DS representing that place. The time when an event happens happens can be described by the SemanticTime DS in Event DS or a semantic relation timeOf to the SemanticTime DS representing that time.
Event	Describes one event resulting from the decomposition of the parent event.
EventRef	References an existing description of an event resulting from the decomposition of the parent event (optional).
SemanticPlace	Describes semantically a place where the event occurs, and/or its extent (optional).
SemanticTime	Describes semantically a time when the event occurs, and/or its duration.

Table 3.9 - Semantic RelationTypes

Type	Relations
Semantic	agent, agentOf, patient, patientOf, experiencer, experiencerOf, stimulus, stimulusOf, causer, causerOf, goal, goalOf, beneficiary, beneficiaryOf, them, themOf, result, resultOf, instrument, instrumentOf, accompanier, accompanierOf, summarizes, summarizedBy, state, stateOf.
Combination	specializes, generalizes, similar, opposite, exemplifies, exemplifiedBy, interchangeable, identifier, part, partOf, contrasts, property, propertyOf, user, userOf, component, componentOf, substance, substanceOf, entailment, entailmentOf, manner, mannerOf, influences, dependsOn, membershipFunction
Key	keyFor, annotes, annotatedBy, shows, appearsIn, reference, referenceOf, quality, qualityOf, symbolizes, symbolizedBy, location, locationOf, source, sourceOf, destination, destinationOf, path, pathOf, time, timeOf, depicts, depictedBy, represents, representedBy, context, contextFor, interprets, interpretatedBy

Table 3.10 - SemanticRelation CS

Name	Definition
RelationBaseType	Base DS for the relations amongst a set of two or more description schemes (abstract).
Property	Describes the properties of the relation. The properties of a relation may be defined either for each instance of the Relation Property data type using the type attribute (see below). Only one of these may be specified.
Argument	Describes one argument of a relation. The arguments appear in a Relation DS instance in order: first argument appears, second argument, followed by each subsequent argument, followed by each subsequent argument. This element shall not be used if values for source and taget (or both) are present.
Source	References the description scheme that is the first argument of the relation. This attribute must not be present if the arguments of the relation are specified using the Argument.
Target	References the description scheme that is the second argument of the relation. This attribute must not be present if the arguments of the relation are specified using the Argument.
Properties	Describes the properties of the relation.
Strength	Indicates the strength of the relationship on a fuzzy scale from [0,1], where one is the strongest value and zero the weakest. This can be used represent fuzzy graphs. The default value is one.
RelationType	DS describing a relation amongst a set of two or more description schemes.
Name	Identifies the relations using an XML qualified name; for example "before".
Arity	Indicates the number of arguments (arity) in the relation. A relation must have an arity of a relation is determined from the number of actual arguments present, either as values of Arguments, or as the values of source and target.

CHAPTER 4

DESCRIBING VIDEO SEMANTIC CONTENT WITH MPEG-7 DESCRIPTION TOOLS FOR A SPATIO-TEMPORAL VIDEO INFORMATION SYSTEM

4.1 VIDEO CONTENT MODELING

Video is a complex data type that needs a rich data model for representing the important aspects of video information [5]. Video content can be modelled at several levels and of granularity and abstraction. Determining this level is highly dependent on the searching facilities that are planned to be provided. Actually, today there are two different approaches for video content queriying. While the first one deals with the low-level features, such as color, texture and shapes, the other concentrates on high-level semantics, such as objects, spatial relationships between objects, events and actions involving objects, temporal relationships between events and actions. This thesis study concerns the second one. Therefore, it fundementally neccesiates a semantic data model.

There are also different aspects for modelling the semantic content, namely annotation-based modelling, physical level video segmentation approach, and object based modelling approaches.

1. Annotation-based video models semantics and spatial features are annotated with free text or attributes or keywords.

2. The physical level video segmentation approach represents video data as a stream of small segments together with temporal and spatial properties specific to the application. This approach does no directly point the semantic concepts such as, objects, events, roles, players, etc.

3. Object based models focus on the modelling of semantic content of the video data using object-oriented modelling techniques.

For this thesis study a pre-extisting object-based model is chosen that will be intoduced in section 4.1.1.

4.1.1 ST-AVIS - Spatio-Temporal Video Information System

ST-AVIS, the Spatio-Temporal Video Information System, is an extension to the object-based model, of an advanced video information system, the AVIS [5]. The AVIS model itself and how ST-AVIS is extended with the integration of spatio-temporal properties and relations are explained in the succeeding subsections.

4.1.1.1 AVIS - Advanced Video Information System

AVIS (Advanced Video Information System) [5], is an object-based video data model that can be used for any kind of video data. It models the semantic entities objects, events and activities. The individual entities, such as the characters of a movie like Homer, Marge or things like cars, trees, etc., are called objects. An activity describes the subject of a given video frame-sequence [30]. Eating, walking and flying are examples of activities. Instantiation of activities results in events such as "Ali is walking". An event has additional features other than an activity, roles and actors. For instance, Ali is the actor of the "Ali is walking" event and his role is eater.

A video stream is composed of a set of video frames. Contiguous frames that contain semantically meaningful data form a frame sequence. Each semantic entity is related with one or more frame sequences in which they are seen. The association map introduced in [5] illustrates these relations in a graphical form.

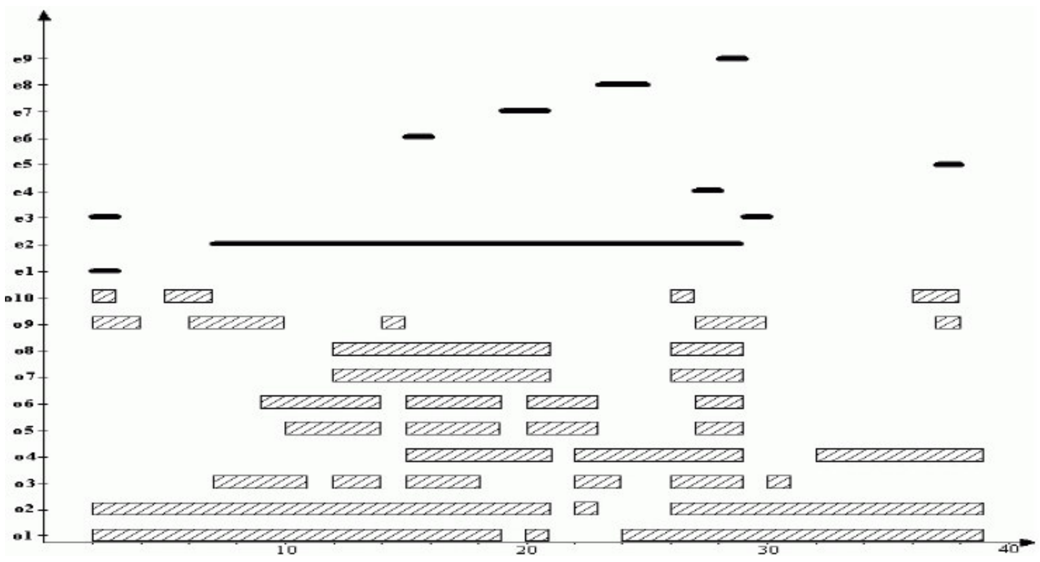


Figure 4.1 - A sample association map

Figure 4.1 taken from [5] is an example association map. The x-axis represents the frames; y-axis represents the entities. The line segments establish the relation of the entities on the y-axis with the frames in which they appear. The thick lines are used for event while the thin ones are used for objects. For example the entity e1 occurs in the frame segment [2, 4).

Based upon the association map, a frame segment tree (FST) is built for indexing purposes. The nodes of the tree represent the overlapping line segments. Figure 4.2 shows the frame segment tree equivalence of the association map in Figure 4.1. The identities of the list of entities appearing in an interval are listed in each node, corresponding to that interval.

An object cannot be listed in a node if it does not appear in all the frames of the node. For example, o9 cannot be included in the list of node 13 because node 13 represents the frame segment [1, 6) and the object o9 appears in the frame segment [2, 4). In addition, if the appearance of an object exceeds the time interval related to a node, it is included in more than one node. Object o9 exceeds the time interval represented by node 2 and, therefore it is also listed in node 3.

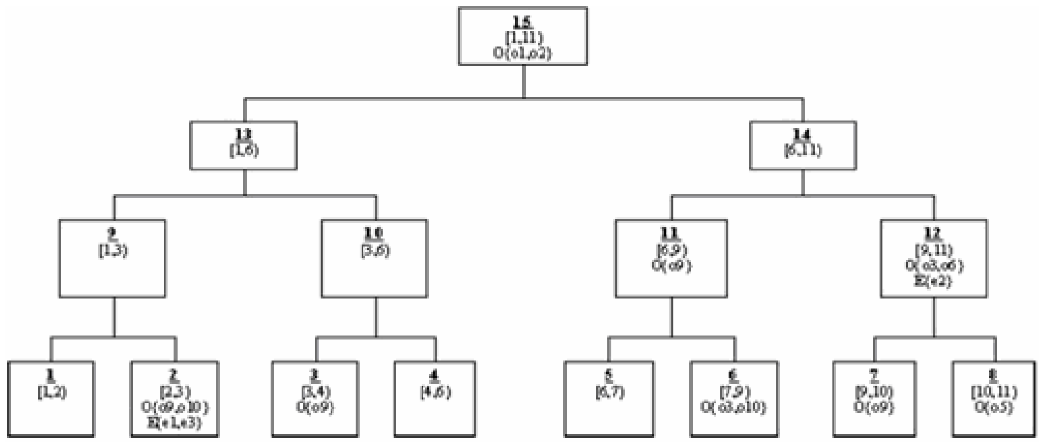


Figure 4.2 - Frame Segment Tree

FST is used for evaluating queries to search for the interval a semantic entity occurs. Traversing the tree to answer such kind of queries can be done fast and easily.

4.1.1.2 Spatio-Temporal Properties

To specify the spatial location of an entity, ST-AVIS model utilizes a 2-dimensional coordinate system. Since the location of an object in a video is not that exact all the time, an approximation strategy has been used. Minimum Bounding Rectangle (MBR) approach, which is a common strategy for 2-dimensional coordinate system defines MBR as the minimum rectangles that covers all parts of an object. ST-AVIS model makes use of MBR approach for approximating the location of objects. ST-AVIS defines the spatial property of an object as follows [6]: "The spatial property of an object A is a *tuple* (*R*, *I*), where, R is a rectangular area (a region) that covers all area in which object A appears during the time interval $I=[t_i, t_f]$. R is not exactly the minimum-bounding rectangle of A. At any time t in $[t_i, t_f]$, object A may be located anywhere in R."

If an object is static in a location during a time interval, R is actually the minimum-bounding rectangle of A. Otherwise, R is determined so that the object has a uniform movement in R.

4.1.1.3 Spatio-Temporal Relations

The spatial relationship between two objects is obtained from the spatial relationship between the regions of each object. ST-AVIS model uses a rule base that covers the relations *top, bottom, right, left, top-right, top-left, bottomright, bottom-left, overlaps, equal, inside, contain, touch*, and *disjoint*. The movement of objects causes spatial relations to change over time. Therefore the relationships between objects may not be exactly the same among a time interval. To cope up with this fact the ST-AVIS introduced fuzzy spatio-temporal relationships. ST-AVIS defines the spatio-temporal relationship as follows [6]: "Let A_i and A_j be two objects. Their *fuzzy spatio-temporal relationship* during time interval I_k is $A_i(\alpha,\mu,I_k)A_j$ where α is one of the spatial relations supported by our model (top, left, etc.), and μ is the value of membership. $A_i\alpha\mu A_j$ is true during the interval I_k." The membership value, μ, is the degree two objects satisfy the spatial relationship. μ can take values in [0,1]. In Figure 4.3 [6], the relation "A being at the LEFT of B" possessing different membership values is illustrated. The angle between the x-axis and the line passing through the centers of the rectangles are used to calculate the membership value. Table 4.1 taken from [6] illustrates the relation between the angle and the membership value for each relation.

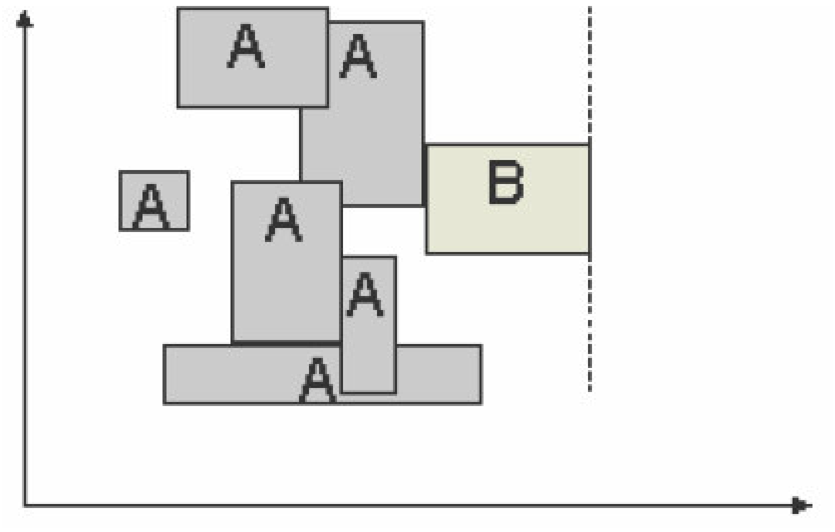


Figure 4.3 - Examples for LEFT relationship for two objects

Table 4.1 - Relation between the membership value and angle between the centers of rectangles

Relation	Angle (*)	Membership Value
TOP	arctan(x/y)	1- (angle/90)
LEFT	arctan(y/x)	angle/90
TOP-LEFT	arctan(x/y)	1 – (abs(angle-45) / 45)
TOP-RIGHT	arctan(y/x)	1 – ((angle – 45) / 45)
* x is the horizontal distance between centers of two rectangles. * y is the vertical distance between centers of two rectangles.		

4.1.1.4 ST-AVIS Extensions

ST-AVIS model is an extension of AVIS model with spatio-temporal properties/relationships. Association map and consequently the frame segment tree are redesigned to represent the newly added features.

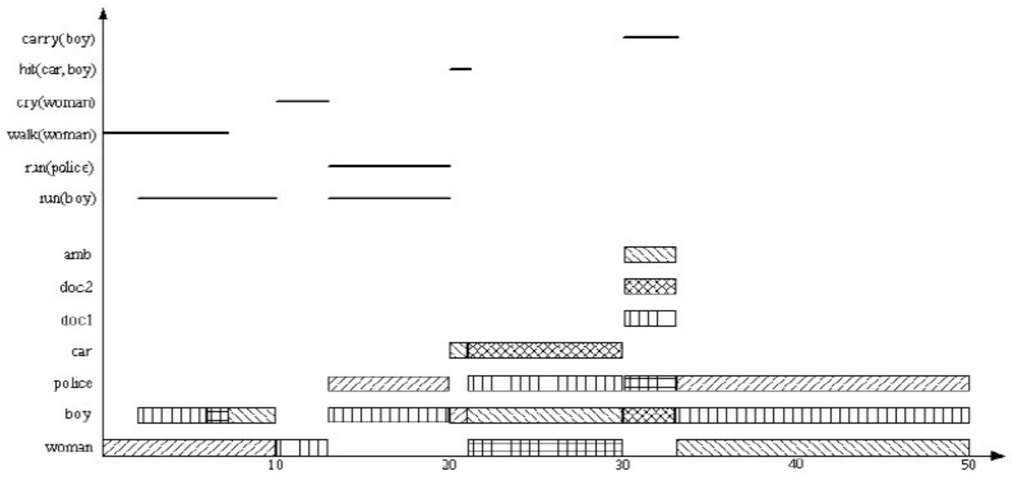


Figure 4.4 - The extended association map

In AVIS the association map segments the video considering only the existence of an object. ST-AVIS further divides the frame sequences according to the locations of the objects. As a result, the extended association map represents (*interval, region*) pairs for objects. Figure 4.4 shows an example ST-AVIS association map. The different patterns on a line segment are used to discriminate the changing locations of an object. To illustrate, the boy is seen in three different locations in the first frame sequence.

The node structure in the frame segment tree is also extended to contain a list of (*object, region*) pairs instead of just the object list. Each object region pair entails that the object is seen in the region within the interval associated with the node.

4.1.1.5 Supported Query Types

ST-AVIS video model supports the following types of queries:

1. Elementary Object Queries: Asks for videos together with the frame sequences where an object occurs.

2. Elementary Activity Type Queries: Asks for videos together with the frame sequences where an activity occurs.

3. Elementary Event Queries: Asks for videos together with the frame sequences where an event occurs.

4. Object Occurrence Queries: Asks for the objects in a given interval and video.

5. Activity Type Occurrence Queries: Asks for the activities in a given interval and video.

6. Event Occurrence Queries: Asks for the events in a given an interval and video.

7. Fuzzy Spatial Relationship Queries: Asks for the videos together with the frame sequences where two objects satisfy a given relation with a given membership value.

8. Regional(Frame) Queries: Asks for the videos together with the frame sequences where an object is seen in a given region with a given membership value.

9. Regional(Interval) Queries: Asks for the videos together with the regions where an object is seen in a given interval.

10. Trajectory Queries: Asks for the videos together with a number of lists of regions which form a trajectory that begin and end in two given regions with a given membership value.

4.1.2 Application of Multimedia Description Schemes Over ST-AVIS

In this thesis, the ST-AVIS model is used to build a database system. The database is intended to provide querying the content of all videos at the same time.

Three different XML files are designed to store the whole data related to a single video:

- Object File: Stores objects seen in a video together with their spatio-temporal properties,
- Event File: Stores events together with their temporal properties,
- Frame Segment File: Stores the data of the calculated frame segment tree. This file has the same functionality with the frame segment tree in query execution.

The example files are illustrated in Appendices E, F, and G.

The succeeding sections illustrate how ST-AVIS modeling constructs are associated with MPEG-7 Multimedia Description Tools in order to design these three file types. The complete description adopted for ST-AVIS will be given in two steps, the video segment decomposition is introduced followed by the description of events, objects and their relations with segments.

4.1.2.1 MPEG-7 Root and Top Level Elements

The MPEG-7 root and top level elements are necessary to create valid descriptions. The root element is the main element enclosing the entire description.

The organization of MPEG-7 root and top-level elements are illustrated in Figure 4.5 taken from [36]. The Description Units carry partial descriptions and can include any MPEG-7 element. Semantically complete descriptions that include top-level elements immediately after the root do also exist. The three different types of description tasks, namely Content Entity Description, Content Abstraction Description, and Content Management are wrapped into top-level elements to compose complete descriptions.

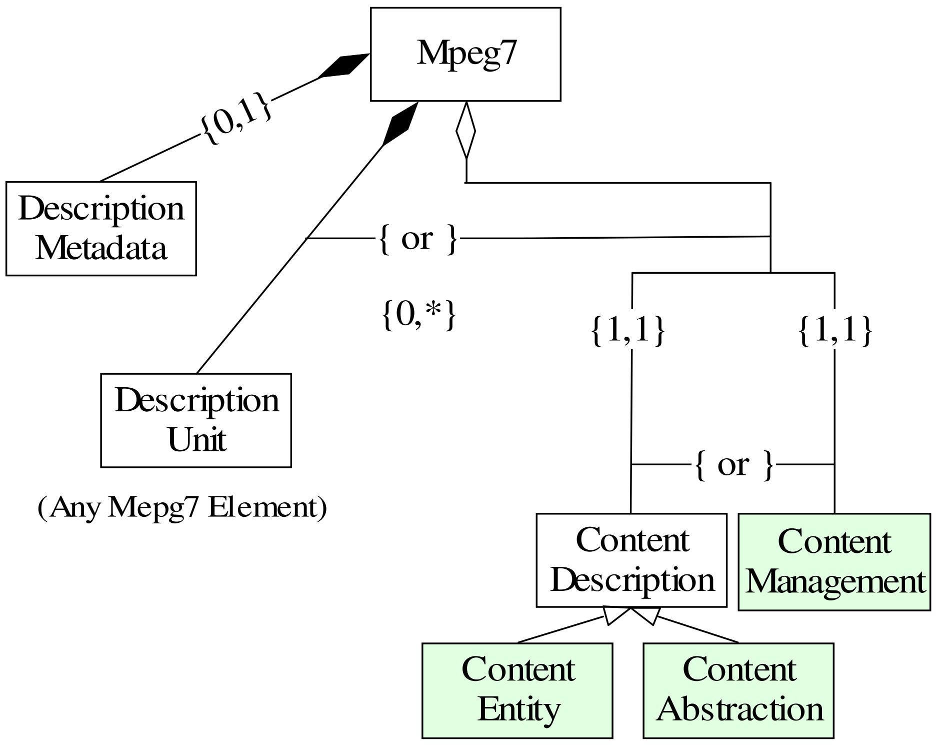


Figure 4.5 - Organization of Mpeg7 root and top-level elements

Different types of multimedia, such as image, video and audio, may be described using the models provided by the *Content Entity Description* elements. Header, MediaInformation, Media Locators, creation information, Usage information, Spatial locators, Spatial Masks, visual features, textual annotation, Structural unit, Matching unit, Point of view and Relation are the elements described in this class.

The summary of the multimedia content, the different explanations of the image, video and audio signals, variations of the content and real-world semantics of multimedia content can be described by the elements defined with *Content Abstraction Description*.

The *Content Management* element contains the description of the creation, the classification and the usage of multimedia content.

4.1.2.2 Modeling Video Structure

The structural data related to a video is wrapped into the ContentEntity description. The ContentEntityType defines the structure of ContentEntity. Figure 4.6 displays the ContentEntityType expressed with DDL.

```
<complexType name="ContentEntityType">
  <complexContent>
    <extension base="mpeg7:ContentDescriptionType">
      <sequence>
        <element name="MultimediaContent"
          type="mpeg7:MultimediaContentType"
          minOccurs="1" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Figure 4.6 - ContentEntityType

ContentEntityType extends ContentDescriptionType. Additionally it contains a sequence of child elements of type MultimediaContentType. VideoType that is a specialized MultimediaContentType is chosen to further cover the VideoSegment DS. VideoType is shown in Figure 4.7.

```
<complexType name="VideoType">
  <complexContent>
   <extension base="mpeg7:MultimediaContentType">
    <sequence>
     <element name="Video"
      type="mpeg7:VideoSegmentType"/>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
```

Figure 4.7 - VideoType

In Figure 4.7, VideoSegmentType defines the structure of a VideoSegment DS. The detailed content of VideoSegmentType is given in Figure 4.8. Out of the content of this description TemporalDecomposition DS is chosen to model the temporal properties of events and the spatio-temporal properties of objects. With the VideoSegmentTemporalDecompositionType, distinct frame sequences in the video are mapped to different segments. MediaTime element in VideoSegmentType delineates these segments. Among the alternative elements in MediaTime the MediaRelIncrTimePoint and MediaIncrDuration in Figure 4.11 are chosen since their combination is appropriate for representing a frame sequence. MediaRelIncrTimePoint is a descriptor specifying a media time point relative to a time base by counting time units [28]. The time units correspond to the time increment of the timestamps of successive frames in the video stream in our model. The timeUnit attribute MediaRelIncrTimePoint and MediaIncrDuration types is set to *"PT1N25F"* indicating that 25 frames are seen in 1 second. The media time point is then specified by the number of these time units.

MediaIncrDuration is a descriptor specifying the duration of a media time period by counting time units [28].

```
<complexType name="VideoSegmentType">
   <complexContent>
    <extension base="mpeg7:SegmentType">
      <sequence><choice minOccurs="0">
        <element name="MediaTime" type="mpeg7:MediaTimeType"/>
        <element name="TemporalMask" type="mpeg7:TemporalMaskType"/>
       </choice>
       <choice minOccurs="0" maxOccurs="unbounded">
        <element name="VisualDescriptor" type="mpeg7:VisualDType"/>
        <element name="VisualDescriptionScheme" type="mpeg7:VisualDSType"/>
        <element name="VisualTimeSeriesDescriptor"
          type="mpeg7:VisualTimeSeriesType"/>
       </choice>
       <element name="MultipleView"
type="mpeg7:MultipleViewType" minOccurs="0"/>
       <element name="Mosaic" type="mpeg7:MosaicType" minOccurs="0"
         maxOccurs="unbounded"/>
       <choice minOccurs="0" maxOccurs="unbounded">
        <element name="SpatialDecomposition"
          type="mpeg7:VideoSegmentSpatialDecompositionType"/>
        <element name="TemporalDecomposition"
          type="mpeg7:VideoSegmentTemporalDecompositionType"/>
        <element name="SpatioTemporalDecomposition"
          type="mpeg7:VideoSegmentSpatioTemporalDecompositionType"/>
        <element name="MediaSourceDecomposition"
          type="mpeg7:VideoSegmentMediaSourceDecompositionType"/>
       </choice>
      </sequence>
    </extension>
   </complexContent>
  </complexType>
```

Figure 4.8 - VideoSegmentType

```
<complexType name="VideoSegmentTemporalDecompositionType">
  <complexContent>
   <extension base="mpeg7:TemporalSegmentDecompositionType">
    <choice minOccurs="1" maxOccurs="unbounded">
     <element name="VideoSegment" type="mpeg7:VideoSegmentType"/>
     <element name="VideoSegmentRef" type="mpeg7:ReferenceType"/>
     <element name="StillRegion" type="mpeg7:StillRegionType"/>
     <element name="StillRegionRef" type="mpeg7:ReferenceType"/>
    </choice>
   </extension>
  </complexContent>
 </complexType>
```

Figure 4.9 - VideoSegmentTemporalDecompositionType

```
<complexType name="VideoSegmentSpatioTemporalDecompositionType">
  <complexContent>
   <extension base="mpeg7:SpatioTemporalSegmentDecompositionType">
    <choice minOccurs="1" maxOccurs="unbounded">
     <element name="MovingRegion" type="mpeg7:MovingRegionType"/>
     <element name="MovingRegionRef" type="mpeg7:ReferenceType"/>
     <element name="StillRegion" type="mpeg7:StillRegionType"/>
     <element name="StillRegionRef" type="mpeg7:ReferenceType"/>
    </choice>
   </extension>
  </complexContent>
 </complexType>
```

Figure 4.10 - VideoSegmentSpatioTemporalDescriptionType

```
<complexType name="MediaTimeType"><sequence>
   <choice>
    <element name="MediaTimePoint" type="mpeg7:mediaTimePointType"/>
    <element name="MediaRelTimePoint" type="mpeg7:MediaRelTimePointType"/>
    <element name="MediaRelIncrTimePoint"
     type="mpeg7:MediaRelIncrTimePointType"/>
   </choice>
   <choice minOccurs="0">
    <element name="MediaDuration" type="mpeg7:mediaDurationType"/>
    <element name="MediaIncrDuration"
     type="mpeg7:MediaIncrDurationType"/>
   </choice>
  </sequence></complexType>
```

Figure 4.11 - MediaTimeType

```
<complexType name="StillRegionType">
  <complexContent>
   <extension base="mpeg7:SegmentType">
    <sequence>
     <choice minOccurs="0">
      <element name="SpatialLocator"
       type="mpeg7:RegionLocatorType"/>
      <element name="SpatialMask"
        type="mpeg7:SpatialMaskType"/>
     </choice>
     <choice minOccurs="0">
      <element name="MediaTimePoint"
        type="mpeg7:mediaTimePointType"/>
      <element name="MediaRelTimePoint"
        type="mpeg7:MediaRelTimePointType"/>
      <element name="MediaRelIncrTimePoint"
        type="mpeg7:MediaRelIncrTimePointType"/>
     </choice>
     <choice minOccurs="0" maxOccurs="unbounded">
      <element name="VisualDescriptor"
        type="mpeg7:VisualDType"/>
      <element name="VisualDescriptionScheme"
        type="mpeg7:VisualDSType"/>
      <element name="GridLayoutDescriptors"
        type="mpeg7:GridLayoutType"/>
     </choice>
     <element name="MultipleView"
       type="mpeg7:MultipleViewType" minOccurs="0"/>
     <element name="SpatialDecomposition"
       type="mpeg7:StillRegionSpatialDecompositionType"
       minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
```

Figure 4.12 - StillRegionType

```
<complexType name="RegionLocatorType" final="#all">
  <sequence>
   <element name="CoordRef" minOccurs="0">
    <complexType>
     <attribute name="ref" type="IDREF" use="required"/>
     <attribute name="spatialRef" type="boolean" use="optional" default="false"/>
    </complexType>
   </element>
   <element name="Box" minOccurs="0" maxOccurs="unbounded">
    <complexType>
     <simpleContent>
      <extension base="mpeg7:BoxListType">
       <attribute name="unlocatedRegion" type="boolean"
use="optional" default="false"/>
      </extension>
     </simpleContent>
    </complexType>
   </element>
   <element name="Polygon" minOccurs="0" maxOccurs="unbounded">
    <complexType>
     <sequence>
      <element name="Coords" type="mpeg7:IntegerMatrixType"/>
     </sequence>
     <attribute name="unlocatedRegion" type="boolean"
use="optional" default="false"/>
    </complexType>
   </element>
  </sequence>
 </complexType>
```

Figure 4.13 - RegionLocatorType

VideoSegmentType also contains SpatioTemporalDecomposition DS which further includes StillRegion DS to describe the spatial property of an object. StillType that describes StillRegion DS comprises the RegionLocatorType that allows defining the rectangular region mentioned in ST-AVIS. The *Box* element within the definition of *RegionLocatorType* given in Figure 4.13 is where the coordinates of the rectangle reside. The *Box* element contains a *Coords* element where the four vertices of a rectangle are specified as a 2*2 matrix. The X-coordinates are written first followed by the y coordinates.

The example structural decompositions prepared for the ST_AVIS video data in Appendices B and C illustrate what is meant by all these data types. In Appendix B, there are two different segments whose ids are *Seg0* and *Seg1*. *Seg0* describes a frame sequence [775, 790] together with the rectangular region [[x1, y1] : [127, 173] [x2, y2] : [162, 216]]. That is to say, the object related to *Seg0* appears at the region expressed as [[x1, y1] : [127, 173], [x2, y2]] : [162, 216]], in the time interval [775, 790].

Similarly the segment decomposition in Appendix C illustrates the segments related to events. In this description only temporal decompositions can take place since events in ST-AVIS are not related with location information.

A more sophisticated structural decomposition forms the content of the frame segment tree file. This decomposition is based on TemporalDecomposition DS. In addition to the interval and region data that exist in the previous decomposition the segments also contain the relationship data. A sequence of Relation elements in SegmentType that is the base class of VideoSegmentType provides representation of relationship of a video segment with the objects and events. Moreover the regions that cover different objects should also be discriminated. The StillRegion is identified with the element name to handle this distinction. The example in Appendix D shows how the regions covering "*rachel*" and "*joey*" in Seg67 are represented.

The content description presented in the frame segment tree file also includes video's full path name and duration meta-data. SegmentType which is the base class of VideoSegmentType contains MediaInformation element of MediaElementType which contains the MediaProfile of type MediaProfileType. The MediaProfileType includes MediaInstance element and consequently the MediaUri via MediaLocator element. MediaUri is where the full path name of the video takes place. The MediaTime element of VideoSegmentType is used to describe the video duration in this case.

4.1.2.3 Modeling Semantic Entities

This section explains the way semantic entities, i.e. objects and events, are described for ST-AVIS and how they are related with the segments extracted with the structural decomposition.

The SemanticDescriptionType, which is used as the wrapper for semantic content of an ST-AVIS video, groups metadata regarding objects and object properties (including object inter-relationships), events and temporal relationships between events [11]. The Semantics is an element in the description of SemanticDescriptionType is where description of each event and object will occupy. The Semantic DS is derived from SemanticBag DS (Figure 4.14) that contains a sequence of elements of SemanticBaseType. The Object DS and Event DS are specialized description tools derived from the SemanticBase DS and they are used to describe objects and events. That is to say, the Semantics element includes a sequence of object and event descriptions.

```
<complexType name="SemanticBagType" abstract="true">
  <complexContent>
   <extension base="mpeg7:SemanticBaseType">
    <sequence>
     <choice minOccurs="0" maxOccurs="unbounded">
      <element name="SemanticBase" type="mpeg7:SemanticBaseType"/>
      <element name="SemanticBaseRef" type="mpeg7:ReferenceType"/>
     </choice>
     <element name="Graph" type="mpeg7:GraphType"
      minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
```

Figure 4.14 - SemanticBagType

The Label element of SemanticBaseType represents the names of the entities, while a sequence of Relation elements of type RelationType keeps the relations of these entities with the video segments. The RelationType given in Figure 4.16 has three attributes to define a relationship, namely type, source and target. The source attribute is omitted and type and target attributes are used to demonstrate the relationships of the described entity with the segments. A semantic description extracted from the object file is given in Figure 4.17. This figure describes a video with a full path name *"D:\Cigdem\TezAvi\Friends - 108 - Tow nana dies twice.avi"*. Two objects named *cup* and *joey* are given together with their relations with identities of segments in which they are seen. The *cup* object has a relationship with *Seg0* and *Seg1* of type *hasMediaPerceptionOf*.

```
<complexType name="SemanticBaseType" abstract="true"><complexContent>
      <extension base="mpeg7:DSType">
        <sequence>
          <element name="AbstractionLevel"
             type="mpeg7:AbstractionLevelType" minOccurs="0"/>
          <element name="Label" type="mpeg7:TermUseType"
            minOccurs="1" maxOccurs="unbounded"/>
          <element name="Definition" type="mpeg7:TextAnnotationType" minOccurs="0"/>
          <element name="Property" type="mpeg7:TermUseType"
             minOccurs="0" maxOccurs="unbounded"/>
          <element name="MediaOccurrence" minOccurs="0" maxOccurs="unbounded">
          <complexType>
            <sequence>
              <choice minOccurs="0">
                 <element name="MediaInformation" type="mpeg7:MediaInformationType"/>
                 <element name="MediaInformationRef" type="mpeg7:ReferenceType"/>
                 <element name="MediaLocator" type="mpeg7:MediaLocatorType"/>
               </choice>
               <element name="Mask" type="mpeg7:MaskType" minOccurs="0"/>
               <element name="AudioDescriptor" type="mpeg7:AudioDType"
                 minOccurs="0" maxOccurs="unbounded"/>
                <element name="AudioDescriptionScheme" type="mpeg7:AudioDSType"
                   minOccurs="0" maxOccurs="unbounded"/>
                <element name="VisualDescriptor" type="mpeg7:VisualDType"
                  minOccurs="0" maxOccurs="unbounded"/>
                 <element name="VisualDescriptionScheme" type="mpeg7:VisualDSType"
                   minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
            <attribute name="type" use="optional" default="perceivable">
              <simpleType>
                <union>
                   <simpleType>
                    <restriction base="NMTOKEN">
                       <enumeration value="perceivable"/>
                       <enumeration value="reference"/>
                       <enumeration value="symbol"/>
                    </restriction>
                   </simpleType>
                   <simpleType>
                    <restriction base="mpeg7:termReferenceType"/>
                   </simpleType>
                </union>
              </simpleType>
            </attribute>
          </complexType>
          </element>
          <element name="Relation" type="mpeg7:RelationType"
            minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
                 </extension>
</complexContent> </complexType>
```

Figure 4.15 - SemanticBaseType

```
<complexType name="RelationType">
  <complexContent>
   <extension base="mpeg7:DSType">
    <attribute name="type" type="mpeg7:termReferenceType" use="optional"/>
    <attribute name="source" use="optional">
     <simpleType>
      <list itemType="anyURI"/>
     </simpleType>
    </attribute>
    <attribute name="target">
     <simpleType>
      <list itemType="anyURI"/>
     </simpleType>
    </attribute>
    <attribute name="directed" type="boolean"
     use="optional" default="true"/>
    <attribute name="strength" type="mpeg7:zeroToOneType"
     use="optional" default="1.0"/>
   </extension>
  </complexContent>
 </complexType>
```

Figure 4.16 - RelationType

```
<mpeg7:Description xsi:type="SemanticDescriptionType"><mpeg7:Semantics>
                    <mpeg7:Label><mpeg7:Name>D:\Cigdem\TezAvi\Friends - 108 - Tow
          nana dies twice.avi</mpeg7:Name></mpeg7:Label>
                    <mpeg7:SemanticBase xsi:type="ObjectType">
                    <mpeg7:Label>
                        <mpeg7:Name>cup</mpeg7:Name>
                    </mpeg7:Label>
                    <mpeg7:Relation
          type="urn:...:hasMediaPerceptionOf" target="Seg0"/>
                    <mpeg7:Relation
          type="urn:...:hasMediaPerceptionOf" target="Seg1"/>
                    </mpeg7:SemanticBase>
                    <mpeg7:SemanticBase xsi:type="ObjectType">
                    <mpeg7:Label><mpeg7:Name>joey</mpeg7:Name></mpeg7:Label>
                    <mpeg7:Relation
          type="urn:...:hasMediaPerceptionOf" target="Seg2"/>
                    <mpeg7:Relation
          type="urn:...:hasMediaPerceptionOf" target="Seg3"/>
                    <mpeg7:Relation
          type="urn:...:hasMediaPerceptionOf" target="Seg4"/>
                    <mpeg7:Relation
          type="urn:...:hasMediaPerceptionOf" target="Seg5"/>
                    </mpeg7:SemanticBase>
</mpeg7:Semantics></mpeg7:Description>
```

Figure 4.17 - Semantic Description of Object

The description of an event is more complex since the role player data should also be modeled. The Definition element in SemanticBaseType allows inserting free form text. A notation is determined to show the list of role/players. The role name followed by a colon followed by the player name forms a role/player pair. The blank separated role/player pairs form the list of role/players of an event. That is to say, the role/players list of the event "Ali calls Ahmet" is represented as *caller:Ali callee:Ahmet*. The role/players list expressed in the given notation is wrapped into the Definition element.

CHAPTER 5

IMPLEMENTATION OF THE VIDEO DATABASE MANAGEMENT SYSTEM

5.1 ARCHITECTURE

The general overview of the developed video database system is as follows:

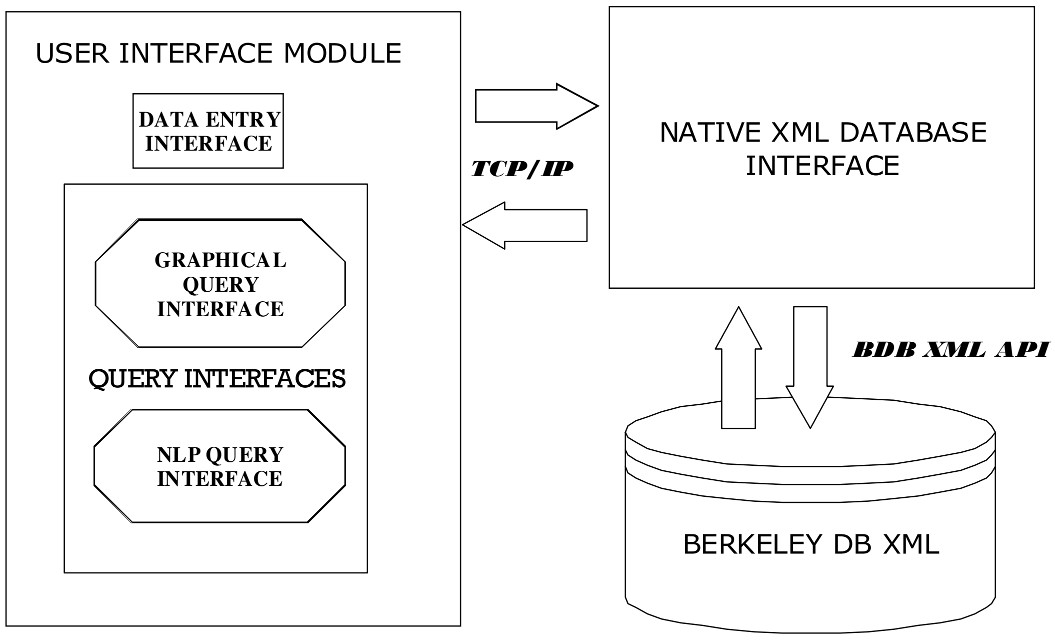


Figure 5.1 - The System Architecture

The user interface module provides users a means for data entrance and requests. Native XML Database Interface that communicates with the user interface module via TCP/IP handles actual database operations. These operations are carried out on Berkeley native XML database through the API it presents.

5.1.1 User Interface Module

The user interface module that consists of the data entry module and the query interface was developed with Borland C++ Builder previously [6]. Actually, the application that was initially using Borland's built in database facilities could not be used because table designs were missing. Since we were to develop our own database operations, we did not need the ones in the existing code. As the first step of the implementation phase the existing code for user interface module is compiled. A network communication infrastructure is added to this system that provides communication with the new database interface implemented in Java.

When the application starts, it presents the interface shown in Figure 5.2. We named our application M-AVIS standing for METU Advanced Video Information System.

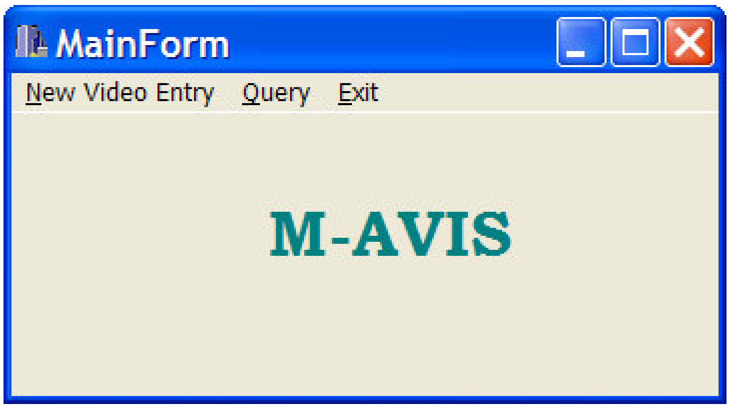


Figure 5.2 - M-AVIS Opening Page

5.1.1.1 Data Entry Module

Data entry module is accessed under the menu item New Video Entry. After the menu item is triggered the user is asked to select the path of the video that is subject to annotation. The New Video Entry form shown in Figure 5.3 allows the user to track the video via a media player API and enter video semantic data.

Navigating through the video by the available buttons on the form, the object region pairs or event and role/players information related to frames can be entered. The current frame number is shown below the media player pane. When a rectangle is drawn on the media player pane that covers the object the coordinates of the rectangle is automatically written in the fields under the title *Area*. The name of the object should be written into the *Name* field under the *Object* title and the frame interval into the fields of *Frames*. Then the *ADDOBJECT* button should be pressed. An illustration is given in Figure 5.4.

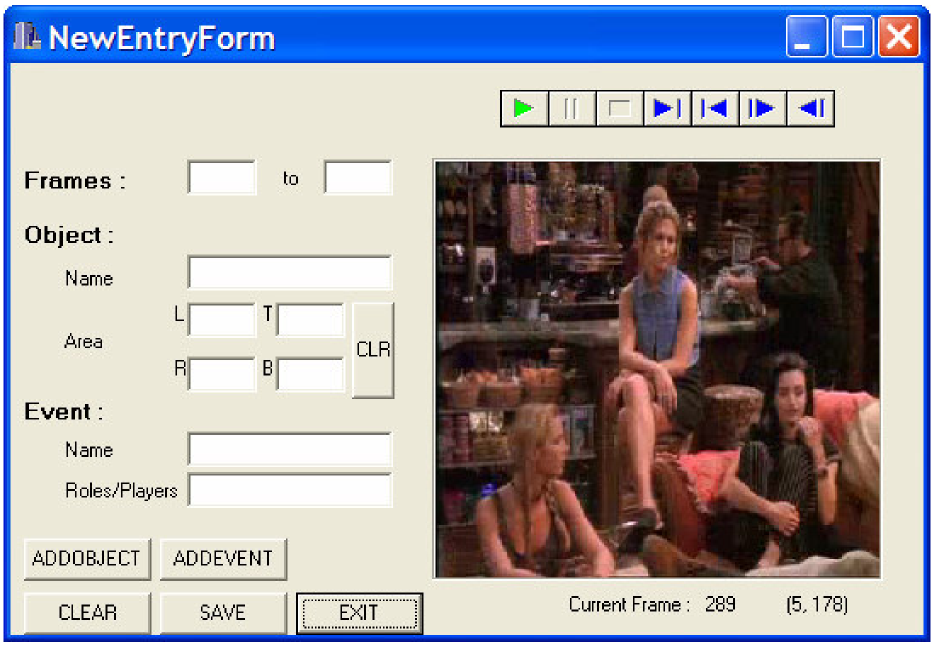


Figure 5.3 - New Video Entry Form

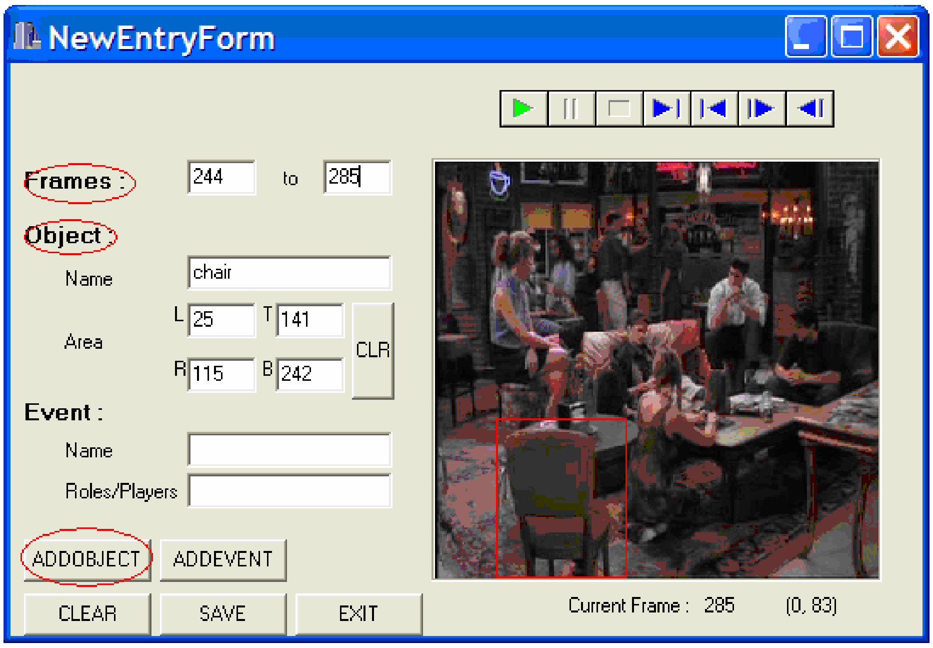
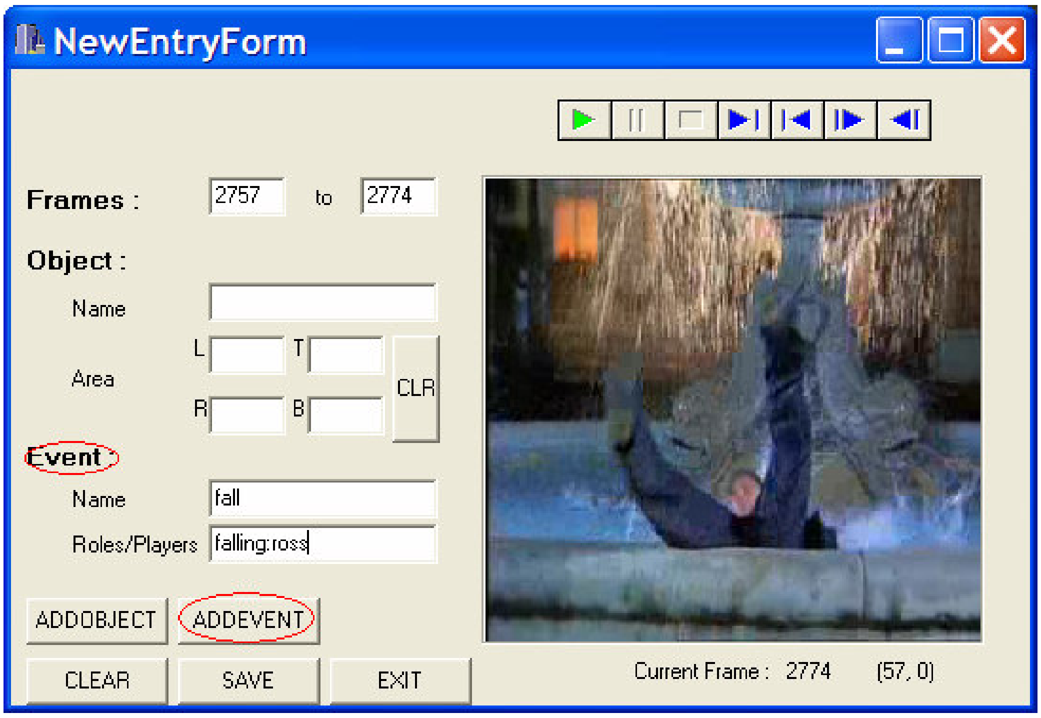


Figure 5.4 - Adding Object Region Pair

The event addition operation requires the event name together with role/players. The Role/Players is filled with each colon-separated role/player pair separated with a blank character. The *ADDEVENT* button is pressed to add the event data. The illustration is in Figure 5.5.

After the annotation for a video is completed the Save button is used to save the data entered.



Frames : 2757 to 2774
Object :
Name
Area L T R B
Event
Name fall
Roles/Players falling:ross
Current Frame : 2774 (57, 0)

Figure 5.5 - Add Event

5.1.1.2 Graphical Query Form

This form provides an interface for the user to view the possible query parameters and select among these parameters and start the query. The possible query parameters are all the items recorded into the database. The graphical query from is shown in Figure 5.6. The objects are listed in *Object1* and *Object2*. The relations supported by the system are listed in

Relation. The threshold value for fuzzy relationship queries can be selected from *Threshold*. The events can be accessed from *Event Name*. When an *Event Name* is chosen the corresponding role/players are added to *Role Player List*. The buttons allow users to query about over the video content in different ways. The query types listed below will later be referenced by the numbers in this list.

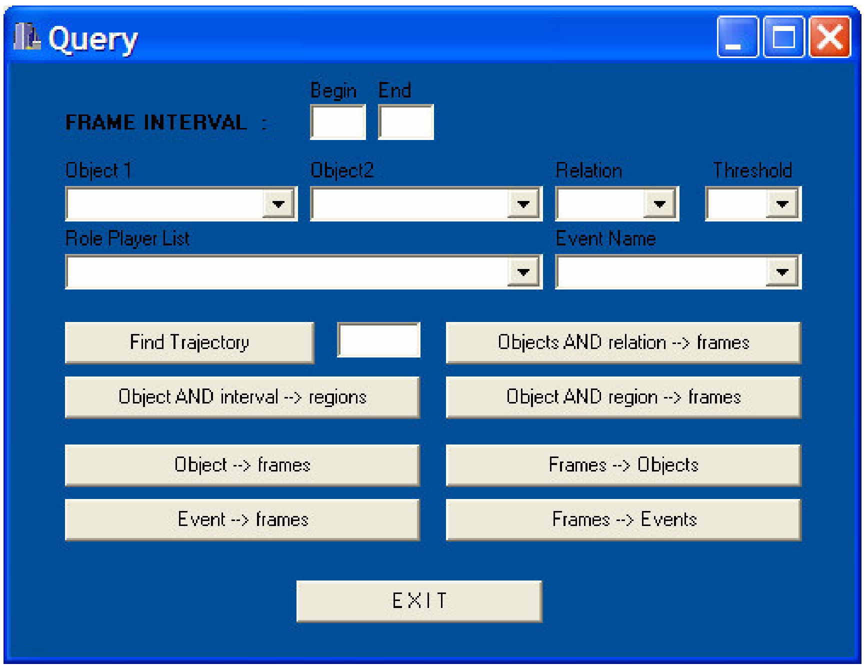
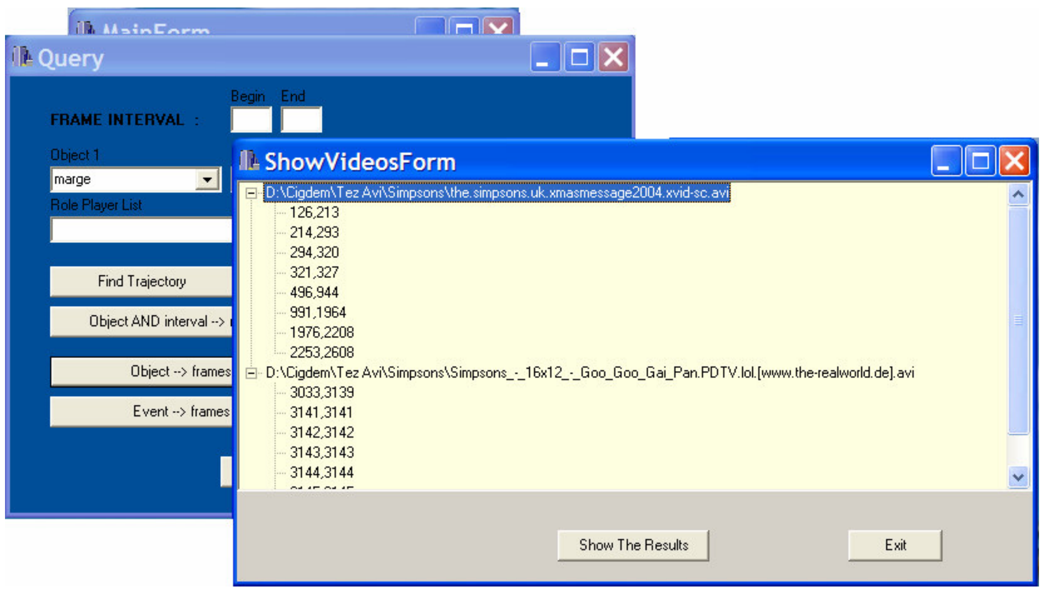


Figure 5.6 - Graphical Query Interface

1. **Elementary Object Queries:** When *Object → frames* button is pressed the frames in which the selected object from *Object1* are queried.

   An example query result is demonstrated in Figures 5.7 and 5.8. The *ShowVideosForm* appears when the query result is not null.

This form lists the videos and the frames in these videos in a tree view. When the video name is selected and the *Show The Results* button is pressed, the solid frame segment set calculated is listed in *Show Frames* form from where the user is enabled to play the resulting segments. If an interval under a video name is selected in *ShowVideosForm* form, only that interval is presented again in the *ShowFrames* form.



```
D:\Cigdem\Tez Avi\Simpsons\the.simpsons.uk.xmasmessage2004.xvid-sc.avi
    126,213
    214,293
    294,320
    321,327
    496,944
    991,1964
    1976,2208
    2253,2608
D:\Cigdem\Tez Avi\Simpsons\Simpsons_-_16x12_-_Goo_Goo_Gai_Pan.PDTV.lol.[www.the-realworld.de].avi
    3033,3139
    3141,3141
    3142,3142
    3143,3143
    3144,3144
```

Figure 5.7 - List of Video & Intervals

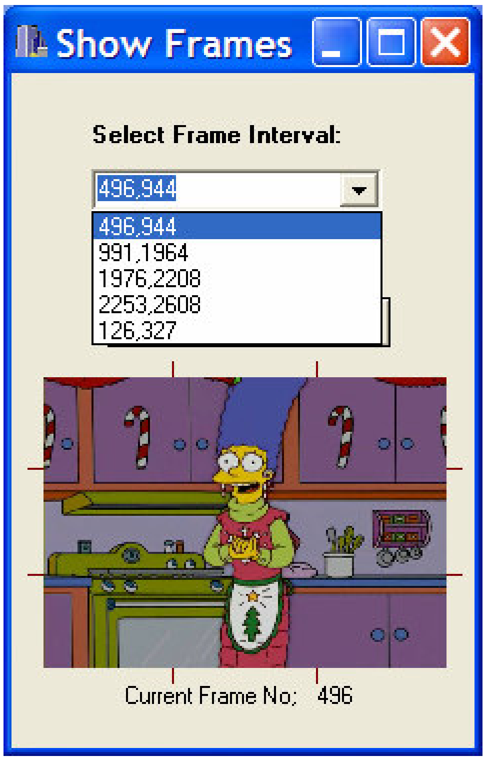


Figure 5.8 - Playing Result Video Clips

2. **Elementary Activity Type Queries:** This query type is not supported via this interface.

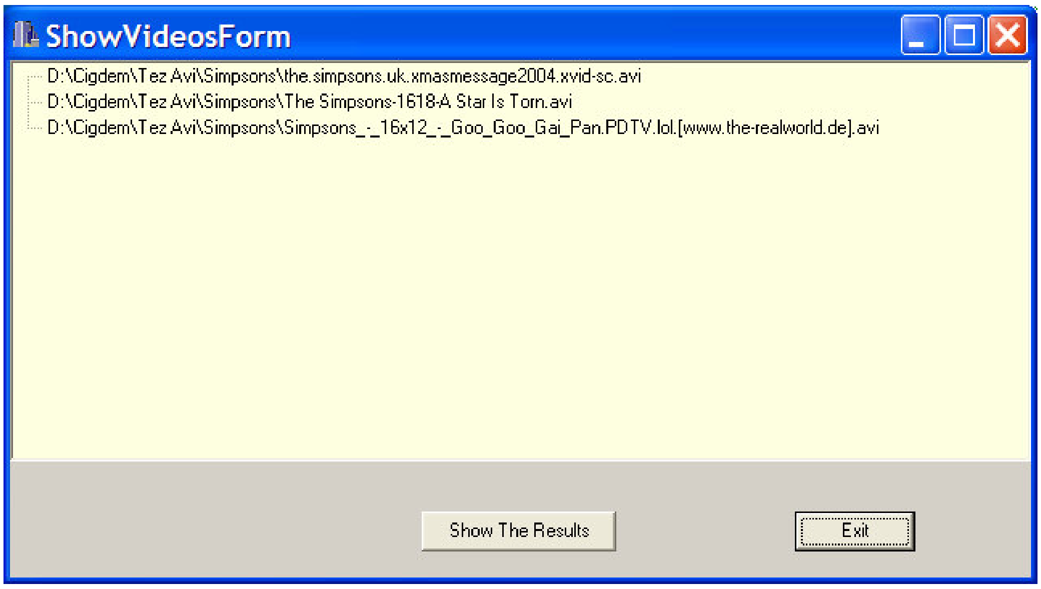
3. **Elementary Event Queries:** This query can be performed over the event specified by the selected *Event Name* and *Role Player List*. The *Event* → *frames* button should be pressed to perform the query. The result is presented in a similar way to the elementary object queries.

4. **Object Occurrence Queries:** The frame interval on which this query performed is specified by the *Begin* and *End*. The *Frames* → *Objects* button is used for executing this type of queries.

When the *Frames* →*Objects* button is pressed the ShowVideosForm is opened listing all the videos annotated, as in Figure 5.9. And if a video is selected and *Show The Results* button is pressed the Show List form opens with the list of objects given in the video together with the interval they are seen. An example result is seen in Figure 5.10. The user can then watch the resulting scenes by selecting either the object name (provides a solid set of intervals) or just an interval on the Show Frames form.

5. **Activity Type Occurrence Queries:** This query type is not supported via this interface.

6. **Event Occurrence Queries:** This is analogous to query type 4. The *Frames* → *Events* button is used in this case.


ShowVideosForm

```
D:\Cigdem\Tez Avi\Simpsons\the.simpsons.uk.xmasmessage2004.xvid-sc.avi
D:\Cigdem\Tez Avi\Simpsons\The Simpsons-1618-A Star Is Torn.avi
D:\Cigdem\Tez Avi\Simpsons\Simpsons_-_16x12_-_Goo_Goo_Gai_Pan.PDTV.lol.[www.the-realworld.de].avi
```

Figure 5.9 - Available Video Name List

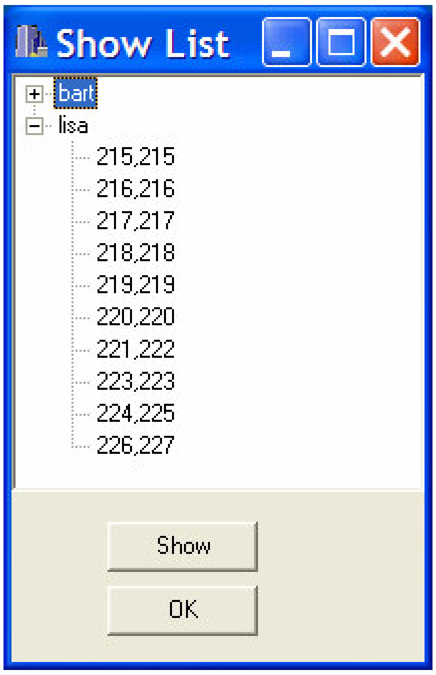


Figure 5.10 - List of Objects & Intervals

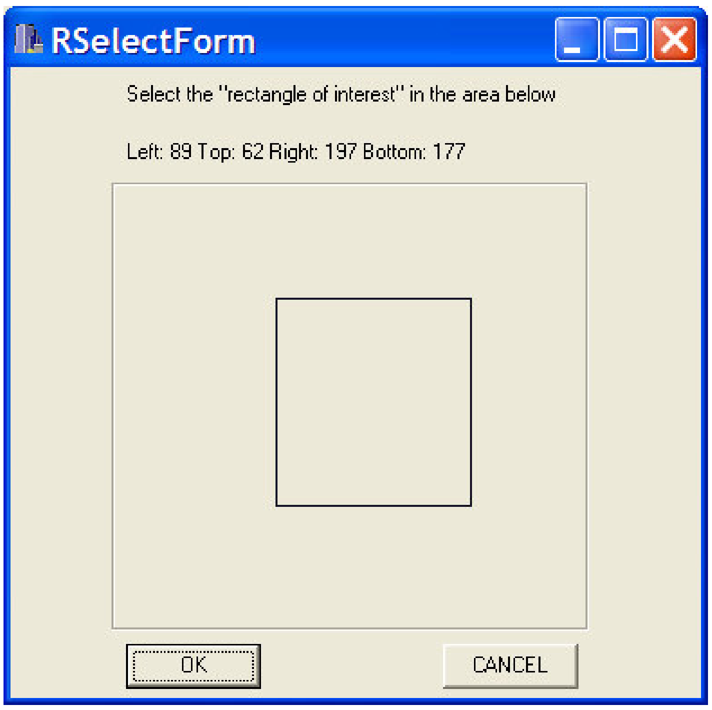
7. **Fuzzy Spatial Relationship Queries:** This query requires four different parameters. The relation that is subject to query is determined by *Relation*. The source object is selected from *Object1* and the target object is selected from *Object2*. Also the membership value can be determined from *Threshold*. The *Objects and relation* → *frames* provides access to this type of query.

   The result of this query is presented in a similar way to elementary object queries.

8. **Regional(Frame) Queries**: The object for this query is selected from *Object1*. When the *Object AND region* → *frames* button is pressed the region is also requested from the user by the form given in Figure 5.11. After selecting the region, the query is carried out by the user determined parameters.

The result of this query is presented analogous to elementary object queries.

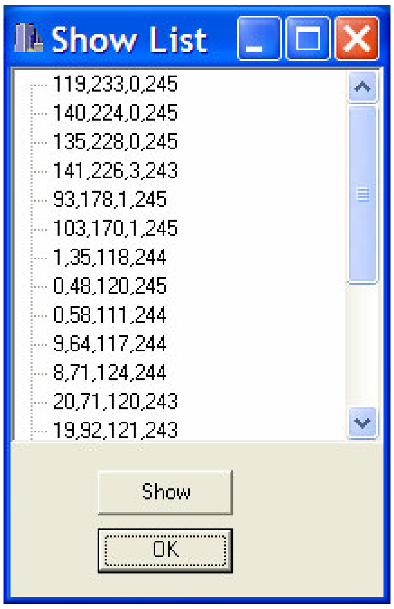
9. **Regional(Interval) Queries:** The object parameter is selected from *Object1* and the interval is specified by *Begin* and *End*. The query is performed when the *Object AND interval* → *regions* button is pressed. The users selects a video from the ShowVideosForm. The coordinates of the returned regions are shown on the Show List form as in Figure 5.12.



Select the "rectangle of interest" in the area below

Left: 89 Top: 62 Right: 197 Bottom: 177

Figure 5.11 - Region Selection Form



```
119,233,0,245
140,224,0,245
135,228,0,245
141,226,3,243
93,178,1,245
103,170,1,245
1,35,118,244
0,48,120,245
0,58,111,244
9,64,117,244
8,71,124,244
20,71,120,243
19,92,121,243
```

Figure 5.12 - Region List

10. **Trajectory Queries:** The object parameter is selected from *Object1*. The edit box near *Find Trajectory* button is for threshold value specification. When the *Find Trajectory* button is pressed two regions are requested from the user via the region selection form in Figure 5.11.

    The resulting videos and the related intervals are placed in a tree view on *ShowVideosForm* form. If the video name is selected all trajectories are displayed on the *ShowTraj* form as in Figure 5.13. If an interval is selected the trajectory is played on the *Show Frames* form. Three frames from the trajectory in Figure 5.13 are shown in Figure 5.14.

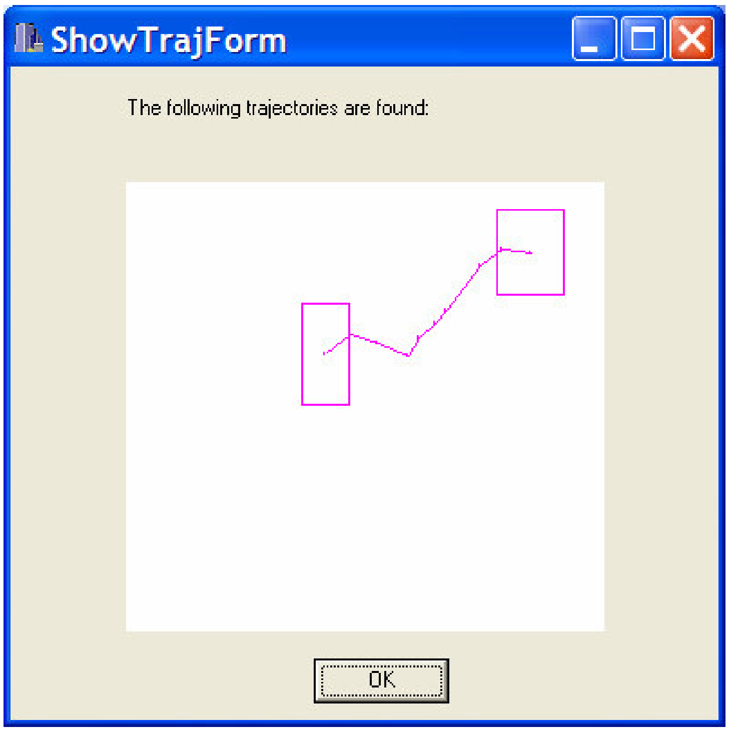


Figure 5.13 - Draw Trajectory

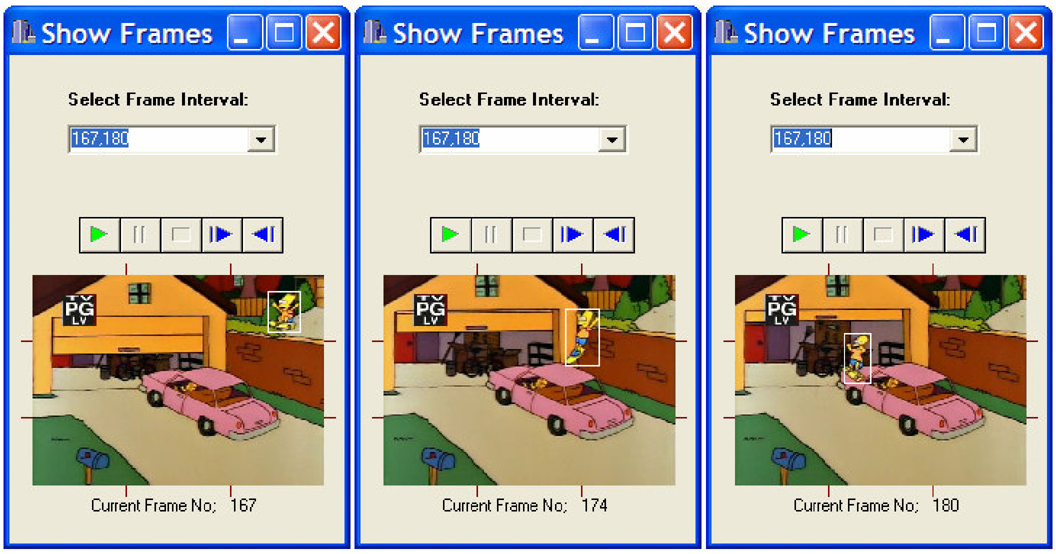


Figure 5.14 - Play Trajectory

5.1.1.3 NLP Query Form

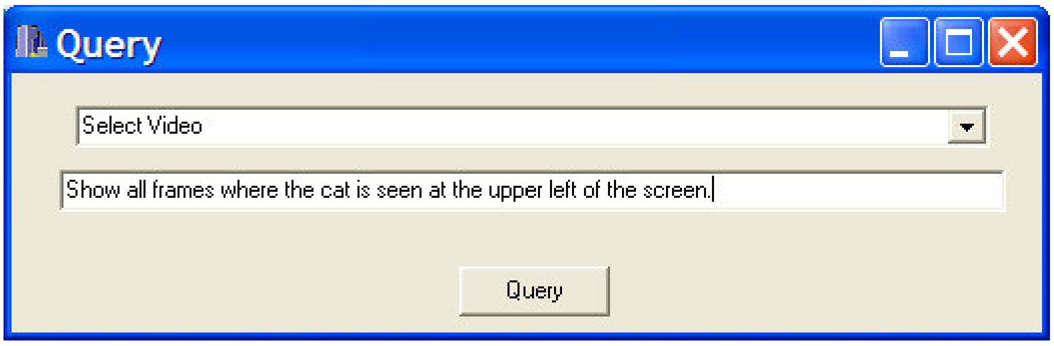
Although the graphical query interface meets majority of the requirements for query processing it may also be problematic. All the objects and events recorded to the database are listed for selection purposes. The number of these items may be so high that listing them into a combo box may not be an appropriate usage. In addition, a query system that understands human language is highly desirable. To provide a better query interface an NLP form is plugged into the developed system.

The code developed for the study in [4] is used as the query parser in our study. The code that was written in C and the user interface module written in C++ are compiled into just one executable code.

The parser code actually maps queries to semantic representations. The parameters that are directly determined via the graphical query interface is determined out of the query sentence in this case. As an example, consider the query *"Retrieve all frames in which Bush is seen."*. The first parameter that should be extracted from this query sentence is the type of the query. The example query is an elementary object query represented with the number *1*. For an elementary object query the only parameter is the object name that is also extracted by the parser.

The query type and the parameters are written to intermediate files to allow the data storage and querying module to access them.

The code utilized for natural language processing can also exhibit ontological behavior. This facility is beneficial in the respect that synonymous or similar words are also searched. The interface is shown is Figure 5.15.



Show all frames where the cat is seen at the upper left of the screen.

Figure 5.15 - NLP Query Interface

The presentation of the results is identical to the graphical query interface. Query types 2 and 5 are also supported via this interface.

5.1.2 Data Storage and Querying Module

The data entered for each video is first embedded into the MPEG-7 compliant schemes and then saved into three different files. After the object related data, event related data are entered and the frame segment tree is derived from the former ones are stored in video object file, video event file and frame segment tree file. To store each type of file a different container is created namely *VideoObj.bdbxml*, *VideoEvt.bdbxml*, *VideoFst.bdbxml*. In this way, a query is executed just against the relevant container eliminating searching irrelevant files.

5.1.2.1 Query Expressions

The following sections explain the XQueries used to evaluate the supported queries.

Elementary Object Queries

"Retrieve all frames in which Beckham is seen." is an example elementary object query. The XQuery expression used for evaluating this class of queries is given in Figure 5.16.

```
for $videofilename in collection('VideoObj.bdbxml')
            /mpeg7:Mpeg7/mpeg7:Description/mpeg7:Semantics
            /mpeg7:Label/mpeg7:Name/text()

    for $segmentid in(collection('VideoObj.bdbxml')
            /mpeg7:Mpeg7/mpeg7:Description
            /mpeg7:Semantics[mpeg7:Label/mpeg7:Name=$videofilename]
            /mpeg7:SemanticBase[mpeg7:Label/mpeg7:Name="ObjectName"]
            /mpeg7:Relation/@target/text())

    let $segmentdata :=
            (collection('VideoObj.bdbxml')
            /mpeg7:Mpeg7/mpeg7:Description/mpeg7:MultimediaContent
            /mpeg7:Video
            /mpeg7:TemporalDecomposition
            /mpeg7:VideoSegment[@id=$segmentid])

    let $startingframe :=
            $segmentdata/mpeg7:MediaTime/mpeg7:MediaRelIncrTimePoint/text()

    let $duration :=
            $segmentdata/mpeg7:MediaTime/mpeg7:MediaIncrDuration/text()

    let $endingframe := $startingframe + $duration
            return
            <result>
                        <videofilename> { $videofilename } </videofilename>
                        <startingframe> { $startingframe } </startingframe>
                        <endingframe> { $endingframe } </endingframe>
            </result>
```

Figure 5.16 - Elementary Object Query

The outer loop of the expression extracts the video file names in the video object container and executes the rest of the query for each of the name found. The second for loop calculates the ids of the video segments (intervals) that the object is related to and iterates again on these results. The rest of the query finds the starting and duration times of the segment and returns the result in the XML structure that is composed of the video file name, starting and ending frame. That is to say, the videos and the intervals in these videos form the outcome of this query.

The NLP interface module lists a set of object names that are ontologically related to the original name drawn from the query. To search for a set of names the expression [*mpeg7:Label/mpeg7:Name="ObjectName"*] is extended to include a test for each name. For instance, the word set containing the words *"joey"* and *"boy"* is queried with the expression is [*mpeg7:Label/mpeg7:Name="joey" or mpeg7:Label/mpeg7:Name="boy"*].

Elementary Activity Type Queries

```
for $videofilename in collection('VideoEvt.bdbxml')
   /mpeg7:Mpeg7/mpeg7:Description/mpeg7:Semantics
   /mpeg7:Label/mpeg7:Name/text()

   for $eventdata in collection('VideoEvt.bdbxml')
            /mpeg7:Mpeg7/mpeg7:Description
            /mpeg7:Semantics[mpeg7:Label/mpeg7:Name=$videofilename]
            /mpeg7:SemanticBase[mpeg7:Label/mpeg7:Name="EventName"]

   for $segmentid in $eventdata/mpeg7:Relation/@target/text()

   let $segmentdata := (collection('VideoEvt.bdbxml')
            /mpeg7:Mpeg7/mpeg7:Description
            /mpeg7:MultimediaContent/mpeg7:Video
            /mpeg7:TemporalDecomposition
            /mpeg7:VideoSegment[@id=$segmentid])

   let $startingframe := $segmentdata/mpeg7:MediaTime
                        /mpeg7:MediaRelIncrTimePoint/text()

   let $duration := $segmentdata/mpeg7:MediaTime
                        /mpeg7:MediaIncrDuration/text()

   let $endingframe := $startingframe + $duration

   return
            <result>
                        <videofilename> { $videofilename } </videofilename>
                        <startingframe> { $startingframe } </startingframe>
                        <endingframe> { $endingframe } </endingframe>
            </result>
```

Figure 5.17 - Elementary Activity Type Query

A query that asks the question "Find all frames in which somebody plays football." is an elementary activity type query. The expression is just like the previous elementary object query, but this time the event name is seeked for.

Elementary Event Queries

```
for $videofilename in collection('VideoEvt.bdbxml')
  /mpeg7:Mpeg7/mpeg7:Description/mpeg7:Semantics
          /mpeg7:Label/mpeg7:Name/text()

  for $eventdata in collection('VideoEvt.bdbxml')
          /mpeg7:Mpeg7/mpeg7:Description
          /mpeg7:Semantics[mpeg7:Label/mpeg7:Name=$videofilename]
          /mpeg7:SemanticBase[mpeg7:Label/mpeg7:Name="EventName"]

          for $segmentid in $eventdata/mpeg7:Relation/@target/text()

  let $segmentdata := (collection('VideoEvt.bdbxml')
          /mpeg7:Mpeg7/mpeg7:Description/mpeg7:MultimediaContent
          /mpeg7:Video
                  /mpeg7:TemporalDecomposition
                  /mpeg7:VideoSegment[@id=$segmentid])

  let $startingframe :=
  $segmentdata/mpeg7:MediaTime/mpeg7:MediaRelIncrTimePoint/text()

          let $roleplayer :=
  $eventdata/mpeg7:Definition/mpeg7:FreeTextAnnotation/text()

  let $duration :=
  $segmentdata/mpeg7:MediaTime/mpeg7:MediaIncrDuration/text()

  let $endingframe := $startingframe + $duration

          return
                  <result>
                          <videofilename> { $videofilename } </videofilename>
                          <startingframe> { $startingframe } </startingframe>
                          <endingframe> { $endingframe } </endingframe>
                          <roleplayer> { $roleplayer } </roleplayer>
                  </result>
```

Figure 5.18 - Elementary Event Query

The specialized form of the elementary activity type query is the elementary event query. For example, "Find all frames in which *Ali* plays football." is an instantiation of the query "Find all frames in which *somebody* plays football.". Somebody is substituted with the object *Ali*, meaning role/player data is added. To extract the actual results, the role/player information returned by the query is compared with the queried role/player data. If they are equivalent, the video- interval is evaluated as a valid result.

Object Occurrence Queries

```
for $segmentdata in collection('VideoFst.bdbxml')
  /mpeg7:Mpeg7/mpeg7:Description/mpeg7:MultimediaContent
  /mpeg7:Video[mpeg7:MediaInformation/mpeg7:MediaProfile
  /mpeg7:MediaInstance/mpeg7:MediaLocator
  /mpeg7:MediaUri="VideoFileName"]
  /mpeg7:TemporalDecomposition/mpeg7:VideoSegment

  let $startingframe :=  $segmentdata/mpeg7:MediaTime
          /mpeg7:MediaRelIncrTimePoint/text()

  let $duration :=
          $segmentdata/mpeg7:MediaTime/mpeg7:MediaIncrDuration/text()

  let $endingframe := $startingframe + $duration

  for $objectname in $segmentdata/mpeg7:Relation
          [@type="urn:mpeg:mpeg7:cs:SemanticRelationCS:
                2001:hasMediaPerceptionOfObject"]
                    /@target/text()

  where
  ( StartingFrame <= $startingframe and
          $startingframe <= EndingFrame ) or
          ( StartingFrame <= $endingframe and
                    $endingframe <= EndingFrame ) or
          ( $startingframe <= StartingFrame and
          EndingFrame <= $endingframe )
          return
  <result>
          <itemname> { $objectname } </itemname>
                    <startingframe> { $startingframe } </startingframe>
                    <endingframe> { $endingframe } </endingframe>
  </result>
```

Figure 5.19 - Object Occurrence Query

"Show all objects present in the last 5 minutes in the clip." is an object occurence query. The frame segment tree container is searched to answer the query. Since the query concerns a specific video, the segment data related to it is fetched by the outermost for loop. For each segment information fetched, the starting and ending frames are calculated and tested against the condition that asks if the segment interval intersects with the interval in the query. If the condition holds the name of the object is returned as a result together with the segment interval.

Activity Type Occurrence Queries

```
for $segmentdata in collection('VideoFst.bdbxml')
  /mpeg7:Mpeg7/mpeg7:Description/mpeg7:MultimediaContent
  /mpeg7:Video[mpeg7:MediaInformation/mpeg7:MediaProfile
  /mpeg7:MediaInstance/mpeg7:MediaLocator
  /mpeg7:MediaUri="VideoFileName"]
  /mpeg7:TemporalDecomposition/mpeg7:VideoSegment

  let $startingframe := $segmentdata/mpeg7:MediaTime
                                    /mpeg7:MediaRelIncrTimePoint/text()

  let $duration := $segmentdata/mpeg7:MediaTime
                                    /mpeg7:MediaIncrDuration/text()

  let $endingframe := $startingframe + $duration

  for $eventname in $segmentdata/mpeg7:Relation
  [@type="urn:mpeg:mpeg7:cs:SemanticRelationCS:
          2001:hasMediaPerceptionOfEvent"]
          /@target/text()

  where
  ( StartingFrame <= $startingframe and
                $startingframe <= EndingFrame ) or
          ( StartingFrame <= $endingframe and
                $endingframe <= EndingFrame ) or
          ( $startingframe <= StartingFrame and
                EndingFrame <=  $endingframe)

          return
          <result>
                <itemname> { $eventname } </itemname>
                <startingframe> { $startingframe } </startingframe>
                <endingframe> { $endingframe } </endingframe>
          </result>
```

Figure 5.20 - Activity Type Occurrence Query

Activity type occurrence queries seek for the list of activities performed in a specified time of interval, such as "Retrieve activities performed in the first 20 minutes.". In the query expression in Figure 5.20, each evaluated segment interval is tested if it intersects with the queried interval. The event names occurring in the segments that pass the test are returned together with the segment interval. The intervals are later used to play the corresponding video segments by the user interface module.

Event Occurrence Queries

```
for $eventdata in collection("VideoEvt.bdbxml")
    /mpeg7:Mpeg7/mpeg7:Description
    /mpeg7:Semantics[mpeg7:Label/mpeg7:Name = "VideoFileName"]
    /mpeg7:SemanticBase[@xsi:type="EventType"]

    for $segmentid in $eventdata/mpeg7:Relation/@target/text()

    let $segmentdata :=
            (collection("VideoEvt.bdbxml")/mpeg7:Mpeg7
            /mpeg7:Description/mpeg7:MultimediaContent
            /mpeg7:Video/mpeg7:TemporalDecomposition
            /mpeg7:VideoSegment[@id=$segmentid])

    let $eventname := $eventdata/mpeg7:Label/mpeg7:Name/text()

    let $roleplayer := $eventdata/mpeg7:Definition
            /mpeg7:FreeTextAnnotation/text()

    let $startingframe := $segmentdata/mpeg7:MediaTime
            /mpeg7:MediaRelIncrTimePoint/text()

    let $duration := $segmentdata/mpeg7:MediaTime
            /mpeg7:MediaIncrDuration/text()

    let $endingframe := $startingframe + $duration

    where
    (StartingFrame <= $startingframe and
                    $startingframe <= EndingFrame) or
            (StartingFrame<= $endingframe and
                    $endingframe <= EndingFrame) or
            ($startingframe <= StartingFrame and
                    EndingFrame <= $endingframe)

            return

    <result>
            <itemname> { $eventname },{ $roleplayer } </itemname>
            <startingframe> { $startingframe } </startingframe>
            <endingframe> { $endingframe } </endingframe>
   </result>
```

Figure 5.21 - Event Occurrence Query

The event occurrence queries are executed against the container *VideoEvt.bdbxml*. The query asks for the events occurring in a video in a specified time interval. An example is "Find all events performed in the last 10 minutes." To achieve this query, the expression in Figure 5.21 is executed. In this expression, the data related to all of the events in the specified video is extracted. For each value returned, the intervals in which the event occurs are extracted. The events occurring in the intervals intersecting with the interval in the query are listed together with the intervals.

Fuzzy Spatial Relationship Queries

```
for $videofilename in collection('VideoFst.bdbxml')
  /mpeg7:Mpeg7/mpeg7:Description/mpeg7:MultimediaContent/mpeg7:Video
  /mpeg7:MediaInformation/mpeg7:MediaProfile
  /mpeg7:MediaInstance/mpeg7:MediaLocator/mpeg7:MediaUri/text()
  for $segmentdata in collection('VideoFst.bdbxml')
        /mpeg7:Mpeg7/mpeg7:Description/mpeg7:MultimediaContent
        /mpeg7:Video[mpeg7:MediaInformation/mpeg7:MediaProfile
        /mpeg7:MediaInstance/mpeg7:MediaLocator/mpeg7:MediaUri=$videofilename]
        /mpeg7:TemporalDecomposition/mpeg7:VideoSegment
        [mpeg7:Relation
        [@type=
        "urn:mpeg:mpeg7:cs:SemanticRelationCS:2001:hasMediaPerceptionOfObject"
        and
        @target="Object1Name"] and
        mpeg7:Relation
        [@type=
        "urn:mpeg:mpeg7:cs:SemanticRelationCS:2001:hasMediaPerceptionOfObject"
                              and @target="Object2Name"]]

  let $startingframe :=
        $segmentdata/mpeg7:MediaTime/mpeg7:MediaRelIncrTimePoint/text()

  let $duration := $segmentdata/mpeg7:MediaTime/mpeg7:MediaIncrDuration/text()

  let $endingframe := $startingframe + $duration

  let $box1 := $segmentdata/mpeg7:SpatioTemporalDecomposition
        /mpeg7:StillRegion[@id="Object1Name"]
        /mpeg7:SpatialLocator/mpeg7:Box/text()

  let $box2 := $segmentdata/mpeg7:SpatioTemporalDecomposition
        /mpeg7:StillRegion[@id="Object2Name"]
        /mpeg7:SpatialLocator/mpeg7:Box/text()

  return
  <result>
          <videofilename> { $videofilename } </videofilename>
          <startingframe> { $startingframe } </startingframe>
          <endingframe> { $endingframe } </endingframe>
          <box1> { $box1 } </box1>
          <box2> { $box2 } </box2>
  </result>
```

Figure 5.22 - Fuzzy Spatial Relationship Query

A fuzzy spatial relationship query finds the video and intervals in which a specified relationship between two objects is satisfied within the given threshold.

The video file names are extracted as a first step to execute the query in Figure 5.22. For each video at hand, the container *VideoFst.bdbxml* is searched for the segment data that contains the views of both of the objects. From the segment data the interval and the regions in which the objects occur are extracted. The video name, the starting and ending frames of the segment and the regions related to the first and second objects are listed for further programmatic eliminations. The elements of the list are examined one by one to test if the regions satisfy the relation with a specified membership value. The satisfying elements are returned as the actual result of the query.

Regional(Frame) Queries

```
Distinct-values(
for $segmentdata in (collection('VideoObj.bdbxml')
  /mpeg7:Mpeg7[mpeg7:Description/mpeg7:Semantics
  /mpeg7:SemanticBase/mpeg7:Label/mpeg7:Name="ObjectName" and
  mpeg7:Description/mpeg7:Semantics/mpeg7:Label/mpeg7:Name="VideoFileName"]
  /mpeg7:Description/mpeg7:MultimediaContent/mpeg7:Video
  /mpeg7:TemporalDecomposition/mpeg7:VideoSegment)\n"

  let $segmentid := $segmentdata/@id

  let $box :=  $segmentdata/mpeg7:SpatioTemporalDecomposition
          /mpeg7:StillRegion/mpeg7:SpatialLocator/mpeg7:Box/text()

  let $startingframe :=
        $segmentdata/mpeg7:MediaTime/mpeg7:MediaRelIncrTimePoint/text()

  let $duration := $segmentdata/mpeg7:MediaTime/mpeg7:MediaIncrDuration/text()

  let $endingframe := $duration + $startingframe

  where
          ( StartingFrame <= $startingframe and
                    $startingframe <= EndingFrame ) or
          ( StartingFrame <= $endingframe and
                    $endingframe <= EndingFrame ) or
          ( $startingframe <= StartingFrame and
                    EndingFrame <= $endingframe)
          return
          <result>
                    <box> { $box } </box>
          </result>
  )
```

Figure 5.23 - Regional(Frame) Query

Regional(Frame) queries evaluate the regions in which an object is seen in a specified time interval in a specified video. For instance "Show all frames where Bill is seen at the upper left of the screen." is a regional (frame) query.

In Figure 5.23, the segment data is extracted considering both the video and the object name. The related interval is tested to see if it somehow intersects with the specified interval. If this is the case the region information extracted from the segment data is returned as a result.

Regional(Interval) Queries

```
for $videofilename in collection('VideoObj.bdbxml')
            /mpeg7:Mpeg7/mpeg7:Description/mpeg7:Semantics
            /mpeg7:Label/mpeg7:Name/text()

            for $segment in(collection('VideoObj.bdbxml')
                        /mpeg7:Mpeg7/mpeg7:Description
                        /mpeg7:Semantics[mpeg7:Label/mpeg7:Name=$videofilename]

            /mpeg7:SemanticBase[mpeg7:Label/mpeg7:Name="ObjectName"]
                        /mpeg7:Relation/@target/text())

            let $segmentdata := (collection('VideoObj.bdbxml')
                  /mpeg7:Mpeg7/mpeg7:Description/mpeg7:MultimediaContent
                  /mpeg7:Video/mpeg7:TemporalDecomposition
                  /mpeg7:VideoSegment[@id=$segment])

            let $box :=  $segmentdata/mpeg7:SpatioTemporalDecomposition
                        /mpeg7:StillRegion/mpeg7:SpatialLocator/mpeg7:Box/text()

            let $startingframe :=
                        $segmentdata/mpeg7:MediaTime
                        /mpeg7:MediaRelIncrTimePoint/text()

            let $duration :=

            $segmentdata/mpeg7:MediaTime/mpeg7:MediaIncrDuration/text()

            let $endingframe := $duration + $startingframe

            return
            <result>
                        <videofilename> { $videofilename } </videofilename>
                        <startingframe> { $startingframe } </startingframe>
                        <endingframe> { $endingframe } </endingframe>"
                        <box> { $box } </box>
            </result>
```

Figure 5.24 - Regional(Interval) Query

The Regional(Interval) queries find the video and intervals that an object occurs in a given region. In Figure 5.24, for each video found in the container *VideoObj.bdbxml*, the segment data, in which the object appears is extracted. From the segment data the interval and the region information are obtained and appended to the video name to form the return value. From the results returned by the XQuery, the region information is extracted. If the extracted region is within the queried region the video-interval pairs form the final results to the query.

Trajectory Queries

A query like "Show the trajectory of a ball that moves from the left to the center." is a trajectory query.

In order not to mix the object's interval-region data related to different videos, an extra query that lists the video file names is executed first.

```
distinct-values( collection( 'VideoObj.bdbxml')
            /mpeg7:Mpeg7/mpeg7:Description/mpeg7:Semantics
            /mpeg7:Label/mpeg7:Name/text())
```

Figure 5.25 - Trajectory Query Part.1

Each value obtained from the query in Figure 5.25 is passed as a parameter to the query in Figure 5.26. The expression in Figure 5.26 lists the interval-region data related to the queried object in ascending order with respect to the starting frame. The results returned from the query are loaded into interval-region list that is further manipulated to look for a trajectory.

The algorithm searching the trajectory examines each interval-region one by one. Whenever a region within the boundaries of the starting region in query is encountered, a new trajectory is started. Actually, the comparison of two regions is done in a fuzzy way. Then the consecutive intervals are tracked to look for a region within the boundaries of the ending region. If the second region is also found, a trajectory is obtained and recorded. The rest of the list is further searched for new trajectories in a similar way.

```
for $segment in(collection('VideoObj.bdbxml')
  /mpeg7:Mpeg7/mpeg7:Description
  /mpeg7:Semantics[mpeg7:Label/mpeg7:Name="VideoFileName"]
  /mpeg7:SemanticBase[mpeg7:Label/mpeg7:Name="ObjectName"]
  /mpeg7:Relation/@target/text())

  let $segmentdata := (collection('VideoObj.bdbxml')
        /mpeg7:Mpeg7/mpeg7:Description/mpeg7:MultimediaContent
        /mpeg7:Video/mpeg7:TemporalDecomposition
        /mpeg7:VideoSegment[@id=$segment])

  let $box :=  $segmentdata/mpeg7:SpatioTemporalDecomposition
          /mpeg7:StillRegion/mpeg7:SpatialLocator/mpeg7:Box/text()

  let $startingframe :=
          $segmentdata/mpeg7:MediaTime/mpeg7:MediaRelIncrTimePoint/text()

  let $duration :=
          $segmentdata/mpeg7:MediaTime/mpeg7:MediaIncrDuration/text()

  order by $startingframe ascending

  return
          <result>
                    <box> { $box } </box>
                    <startingframe> { $startingframe } </startingframe>
                    <duration> { $duration } </duration>
          </result>
```

Figure 5.26 - Trajectory Query Part.1

CHAPTER 6

CONCLUSION

6.1 Comments

A video database management system has come out as the result of this thesis that has gathered a number of considerably important topics within the literature of video metadata storage and retrieval.

1. An object-based video data model is adopted to develop a system that provides full video database search.

2. To meet the description requirements, MPEG-7 description tools defined in the Multimedia Description Scheme and the Visual Description Scheme part of the standard are examined. The study revealed that the standard is well-founded and that it provides various description views of video content. This universally committed standard was adequate to describe the video content that satisfies the requirements of the adopted video data model. For large amounts of audiovisual content, the employment of a standard description will appear to be quite beneficial.

3. Native XML databases are utilized to store the metadata employed into MPEG-7 compliant XML files.

Using XML files was advantageous since the data in our database can be used in any platform for further studies. Also the addition of extra elements to enrich the content is allowed due to the nature of XML.

The survey on native XML databases illustrates that they can provide many of the facilities as that a relational database does. The Berkeley DBXML served well for our needs. With the support for XQuery, complex queries are easily performed. Also the results returned in XML format were easily manipulated. Grouping the files into logical collections with the containers in BDB XML both eliminated to search unrelated files and provided a well-structured design.

4. To overcome the deficiencies related to query entrance in graphical query interface, an NLP interface is adopted to the system.

6.2 Future Work

The biggest handicap in video data management and retrieval systems and consequently our work is automatic data extraction. Utilizing image processing techniques different approaches might be devised. At least the objects introduced to the system might be tracked to extract the corresponding occurrence data.

Since this thesis study dealt mainly with semantic content modeling the focus was on data entry rather than data modification. However Berkeley DB XML has XUpdate support that may be easily used to enhance the system with metadata update facilities. In addition indexing facilities provided by BDB XML may be used for faster access in case of large compiles of video data.

New query types can be added. The simple type queries in use might be broadened via conjunction, or union operations applied over them. Moreover support for new topological and temporal relations might be added.

The content of the video metadata can be enriched considering different necessities for different applications. The newly added data can be appended or inserted into the existing XML files with small amount of effort.

Multimodality is one of today's hot topics whose employment to current systems has been studied in several works. In this respect the video data model might be extended to include textual and audio data.

REFERENCES

[1] Martínez, J.M., ISO/MPEG N4674, *Overview of the MPEG-7 Standard, v 6.0*, MPEG Requirements Group, Jeju, March 2002

[2] Ana B. Benitez, Shih-Fu Chang, *Extraction, Description and Application of Multimedia Using MPEG-7*, Asimolar Conference of Signals, Systems, and Computers, Invited Paper on Special Session on Document Image Processing, Monterey, CA, November 2003

[3] Divitini, M., *Multimedia Description with MPEG-7*, Trondheim, 16 June 2003

[4] Erözel, G., *Natural Language Interface on a Video Data Mode*l, Ms Thesis, METU, July 2005

[5] Adali, S., Candan, K.S., Chen, S., Erol, K., Subrahmanian, V.S., *The Advanced Video Information System: data structures and query processing*, Multimedia Systems, vol. 4, pp. 172-186, 1996

[6] Koprulu, M., Cicekli, N.K. and Yazici, A., *Spatio-Temporal Querying in Video Databases*, Proc. of the Sixth International Conf. on Flexible Query Answering Systems (FQAS'2002), Denmark, Oct 2002

[7] Oomoto, E., Tanaka, K., *OVID: Design and implementation of a video-object database system*, IEEE Transactions on Knowledge and Data Engineering, Vol.5, No.4, August 1993

[8] Dönderler, M.E., Şaykol E., Ulusoy, Ö., Güdükbay , U., BilVideo, *A Video Database Management System*, IEEE Multimedia, Vol. 10, No. 1, pp. 66-70, January/March 2003

[9] Chan, S. S. M., Li, Q., *Developing an Object-Oriented Video Database System with SpatioTemporal Reasoning Capabilities*, Proceedings of the International Conference on Conceptual Modeling (ER'99), pp. 47-61, 1999.

[10] Akrivas, G., Ioannou, S., Karakoulakis, E., Karpouzis, K., Avrithis, Y., Delopoulos, A., Kollias, S., Varlamis I., and Vaziriannis, M., *An Intelligent System for Retrieval and Mining of Audiovisual Material Based on the MPEG-7 Description Schemes.*, Proc. of the European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE), Tenerife, Spain, 12-14 December 2001

[11] Aguis, H., Angelides, M. C., *Modeling and Filtering of MPEG-7-Compliant Meta-Data for Digital Video*, 2004 ACM Symposium on Applied Computing, 2004

[12] Yavuz, Ö., *A Video Database Management System Based on MPEG-7 Standard*, METU, 2002

[13] Sleepycat, *Sleepycat Software:Products:Berkeley DB XML*, http://www.sleepycat.com/products/xml.shtml, Last Accessed Date: 31 August 2005

[14] Obasanjo D., *An Exploration of XML in Database Management Systems*, http://www.25hoursaday.com/StoringAndQueryingXML.html, Last Update Date: 2001, Last Accessed Date: 1 September 2005

[15] Bray, T., Paoli, J., Sperberg-McQueen, C. M., *Extensible Markup Language (XML) 1.0 W3C Recommendation*, http://www.w3.org/TR/1998/REC-xml-19980210, Last Update Date: 10 February 1998, Last Accessed Date: 1 September 2005

[16] Steegmans, B., *XML for DB2 Information Integration*, ibm-redbooks, July 2004

[17] Bourret R., *XML and Databases*, http://www.rpbourret.com/xml/XMLAndDatabases.htm, Last Update Date: December 2004, Last Accessed Date: 2 September 2005

[18] Bourret R., *XML Database Products: Native XML Databases*, http://www.rpbourret.com/xml/XMLDatabaseProds.htm, Last Update Date: December 2004, Last Accessed Date: 10 September 2005

[19] Dekeyser S., Hidders, J., Paredaens, J., *A Transaction Model for XML Databases*, World Wide Web, v.7 n.1, p.29-57, March 2004

[20] Clark, J., DeRose, S., *XML Path Language (XPath) Version 1.0 W3C Recommendation*, http://www.w3.org/TR/xpath, Last Update Date: 16 November 1999, Last Accessed Date: 1 September 2005

[21] Boag, S., Chamberlin, D., Fernández, M. F., Florescu, D., Robie, J., Siméon, J., *XQuery 1.0: An XML Query Language W3C Working Draft*, http://www.w3.org/TR/xquery, Last Update Date: 04 April 2005, Last Accessed Date: 1 September 2005

[22] Jason Hunter, *Xquery.com: Specifications, Articles, Mailing List, and Vendors*, http://www.xquery.com/, Last Update Date: 2003, Last Accessed Date: 18 August 2005

[23] Hunter, J., *X is for XQuery*, Oracle Magazine, May/June 2003

[24] Paul Ford, *Berkeley DB XML: An Embedded XML Database*, http://www.xml.com/pub/a/2003/05/07/bdb.html, Last Update Date: 07 May 2003, Last Accessed Date: 1 September 2005

[25] Sleepycat Software, *Getting Started with Berkeley DB XML for Java*, http://www.sleepycat.com/xmldocs/gsg_xml/java/BerkeleyDBXML-JAVA-GSG.pdf, Last Update Date: 25 April 2005, Last Accessed Date: 1 September 2005

[26] *MPEG-7 Home Page*, http://www.chiariglione.org/mpeg, Last Accessed Date: 31 August 2005

[27] Peter Van Beek, Ana B. Benitez, Joerg Heuer, Jose Martinez, Philippe Salembier, John Smith, Toby Walker, *MPEG-7 Multimedia Description Schemes XM (Version 3.1)*, July 2000, Beijing

[28] Peter Van Beek, Ana B. Benitez, Joerg Heuer, Jose Martinez, Philippe Salembier, John R. Smith, Toby Walker, *MPEG-7 Multimedia Description Schemes WD (Version 3.1)*, July 2000, Beijing

[29] Ernest Wan (CISRA), Claude Seyrat (UPMC), Cédric Thiénot (UPMC), Frank Nack (CWI), *DDL Working Draft 4.0*, July 2000 (Beijing)

[30] *World Wide Web Consortium Issues XML Schema as a Candidate Recommendation*, http://www.w3.org/2000/10/xml-schema-pressrelease.html, Last Update Date: November 2000, Last Accessed Date: 2 September 2005

[31] Benitez, A. B., Rising, H., Jörgensen, C., Leonardi, R., Bugatti, A., Hasida, K., Mehrotra, R., Tekalp, A. M., Ekin, A., Walker, T., *Semantics of Multimedia in MPEG-7*, Proceedings of 2002 IEEE Conference on Image Processing (ICIP-2002), Rochester, New York, USA, 22-25 September 2002.

[32] Dr Matt Roach, Lecture Notes, *Lecture 3 MPEG 7 standard, Multimedia Communications EG 371,* http://galilee.swan.ac.uk/homepages/Home/pub/clabs/MPEG7Standard_3.ppt, Last Accessed Date: 21 August 2005

[33] Peter Van Beek, Ana B. Benitez, Joerg Heuer, Jose Martinez, Philippe Salembier, Shibata Yoshiaki, John R. Smith, Toby, Walker , *Text of 15938-5 FCD Information Technology - Multimedia Content Description Interface – Part 5 Multimedia Description*, ISO/IEC JC 1/SV 29/WG 11 MPEG01/M7009, Singapore, March 2001

[34] Tsinaraki C., Papadomanolakis S., Christodoulakis S., *A Video Metadata Model supporting Personalization & Recommendation in Video-based Services*, MDDE Workshop 2001 (in conjunction with ReTIS 2001), July 2001

[35] Hjelsvold, R., Midtstraum, R., and Sandst, O., *Multimedia Database Systems: Design and Implementation Strategies, chapter Searching and Browsing a Shared Video Database*, pages 89 - 122. Kluwer Academic Publisher, 1996

[36] Salembier, P., Smith, J., *Overview of MPEG-7 multimedia description schemes and schema tools*, In B. S. Manjunath, P. Salembier, T. Sikora, Introduction to MPEG-7: Multimedia Content Description Interface, Chapter 6, Wiley, 2002

[37] Hunter, J., *X Is for Xquery: Part 2*, Oracle Magazine, May/June 2003

APPENDIX A

MOST COMMON XQUERY OPERATONS AND FUNCTIONS [37]

1. Math: +, -, *, div, idiv, mod, =, !=, <, >, <=, >= floor(), ceiling(), round(), count(), min(), max(), avg(), sum()

Division is done using div rather than a slash because a slash indicates an XPath step expression. idiv is a special operator for integer-only division that returns an integer and ignores any remainder.

2. Strings and Regular Expressions: compare(), concat(), starts-with(), ends-with(), contains(), substring(), string-length(), substring-before(), substring-after(), normalize-space(), upper-case(), lower-case(), translate(), matches(), replace(), tokenize()

compare() dictates string ordering. translate() performs a special mapping of characters. matches(), replace(), and tokenize() use regular expressions to find, manipulate, and split string values.

3. Date and Time: current-date(), current-time(), current-dateTime() +, -, div eq, ne, lt, gt, le, gt

XQuery has many special types for date and time values such as duration, dateTime, date, and time. On most you can do arithmetic and comparison operators as if they were numeric. The two-letter abbreviations stand for equal, not equal, less than, greater than, less than or equal, and greater than or equal.

4. XML node and QNames: node-kind(), node-name(), base-uri() eq, ne, is, isnot, get-local-name-from-QName(), get-namespace-from-QName() deep-equal() >>, <<

node-kind() returns the type of a node (i.e. "element"). node-name() returns the QName of the node, if it exists. base-uri() returns the URI this node is from.

Nodes and QName values can also be compared using eq and ne (for value comparison), or is and isnot (for identity comparison). deep-equal() compares two nodes based on their full recursive content.

The << operator returns true if the left operand precedes the right operand in document order. The >> operator is a following comparison.

5. Sequences: item-at(), index-of(), empty(), exists(), distinct-nodes(), distinct-values(), insert(), remove(), subsequence(), unordered(), position(), last()

item-at() returns an item at a given position while index-of() attempts to find a position for a given item. empty() returns true if the sequence is empty and exists() returns true if it's not. dictinct-nodes() returns a sequence with exactly identical nodes removed and distinct-values() returns a sequence with any duplicate atomic values removed. unordered() allows the query engine to optimize without preserving order. position() returns the position of the context item currently being processed. last() returns the index of the last item.

6. Type Conversion: string(), data(), decimal(), boolean()

These functions return the node as the given type, where possible. data() returns the "typed value" of the node.

7. Booleans: true(), false(), not()

There's no "true" or "false" keywords in XQuery but rather true() and false() functions. not() returns the boolean negation of its argument.

8. Input functions: document(), input(), collection()

document() returns a document of nodes based on a URI parameter. collection() returns a collection based on a string parameter (perhaps multiple documents). input() returns s general engine-provided set of input nodes.

APPENDIX B

Example Structural Content Description for ST-AVIS Object

```
<mpeg7:Description xsi:type="ContentEntityType">
<mpeg7:MultimediaContent xsi:type="VideoType">
  <mpeg7:Video xsi:type="VideoSegmentType">
      <mpeg7:TemporalDecomposition
        xsi:type="mpeg7:VideoSegmentTemporalDecompositionType">
      <mpeg7:VideoSegment id="Seg0">
        <mpeg7:MediaTime>
          <mpeg7:MediaRelIncrTimePoint
             timeUnit="PT1N25F">775</mpeg7:MediaRelIncrTimePoint>
          <mpeg7:MediaIncrDuration
           timeUnit="PT1N25F">15</mpeg7:MediaIncrDuration>
        </mpeg7:MediaTime>
                <mpeg7:SpatioTemporalDecomposition
          xsi:type="mpeg7:VideoSegmentSpatioTemporalDecompositionType">
          <mpeg7:StillRegion>
              <mpeg7:SpatialLocator>
               <mpeg7:Box mpeg7:dim="2 2">127 162 173 216</mpeg7:Box>
               </mpeg7:SpatialLocator>
                 </mpeg7:StillRegion>
        </mpeg7:SpatioTemporalDecomposition>
      </mpeg7:VideoSegment>
      <mpeg7:VideoSegment id="Seg1">
        <mpeg7:MediaTime>
          <mpeg7:MediaRelIncrTimePoint
           timeUnit="PT1N25F">31544</mpeg7:MediaRelIncrTimePoint>
          <mpeg7:MediaIncrDuration
           timeUnit="PT1N25F">41</mpeg7:MediaIncrDuration>
        </mpeg7:MediaTime>
       <mpeg7:SpatioTemporalDecomposition
           xsi:type="mpeg7:VideoSegmentSpatioTemporalDecompositionType">
           <mpeg7:StillRegion>
                <mpeg7:SpatialLocator>
                  <mpeg7:Box mpeg7:dim="2 2">163 210 175 216</mpeg7:Box>
                </mpeg7:SpatialLocator>
              </mpeg7:StillRegion>
              </mpeg7:SpatioTemporalDecomposition>
           </mpeg7:VideoSegment>
          </mpeg7:TemporalDecomposition>
        </mpeg7:Video>
      </mpeg7:MultimediaContent>
</mpeg7:Description>
```

APPENDIX C

Example Structural Content Description for ST-AVIS Event

```
<mpeg7:Description xsi:type="ContentEntityType">
<mpeg7:MultimediaContent xsi:type="VideoType">
    <mpeg7:Video xsi:type="VideoSegmentType">
          <mpeg7:TemporalDecomposition xsi:
       type="mpeg7:VideoSegmentTemporalDecompositionType">
          <mpeg7:VideoSegment id="Seg52">
           <mpeg7:MediaTime>
                <mpeg7:MediaRelIncrTimePoint
                  timeUnit="PT1N25F">2425</mpeg7:MediaRelIncrTimePoint>
                    <mpeg7:MediaIncrDuration
                  timeUnit="PT1N10F">50</mpeg7:MediaIncrDuration>
           </mpeg7:MediaTime>
          </mpeg7:VideoSegment>
          <mpeg7:VideoSegment id="Seg53">
           <mpeg7:MediaTime>
                    <mpeg7:MediaRelIncrTimePoint  timeUnit="PT1N25F">
                        4520
                    </mpeg7:MediaRelIncrTimePoint>
                    <mpeg7:MediaIncrDuration timeUnit="PT1N10F">
                        109
                </mpeg7:MediaIncrDuration>
           </mpeg7:MediaTime>
          </mpeg7:VideoSegment>
        </mpeg7:TemporalDecomposition>
      </mpeg7:Video>
    </mpeg7:MultimediaContent>
</mpeg7:Description>
```

APPENDIX D

FST Description Example

```
<mpeg7:Description xsi:type="ContentEntityType">
<mpeg7:MultimediaContent xsi:type="VideoType">
  <mpeg7:Video xsi:type="VideoSegmentType">
    <mpeg7:MediaInformation>
          <mpeg7:MediaProfile>
            <mpeg7:MediaInstance>
              <mpeg7:MediaLocator><mpeg7:MediaUri>D:\Cigdem\TezAvi\Friends - 108 -
             Tow nana dies twice.avi</mpeg7:MediaUri>
              </mpeg7:MediaLocator>
             </mpeg7:MediaInstance>
           </mpeg7:MediaProfile>
          </mpeg7:MediaInformation>
          <mpeg7:MediaTime>
           <mpeg7:MediaTimePoint timeUnit="PT1N25F">0</mpeg7:MediaTimePoint>
           <mpeg7:MediaDuration timeUnit="PT1N25F">1309560</mpeg7:MediaDuration>
          </mpeg7:MediaTime>
          <mpeg7:VideoSegment id="Seg67">
             <mpeg7:Relation source="Seg67"
             type="urn:mpeg:mpeg7:cs:SemanticRelationCS:2001:hasMediaPerceptionOfObj
             ect"  target="rachel"/>
             <mpeg7:Relation source="Seg67"
             type="urn:mpeg:mpeg7:cs:SemanticRelationCS:2001:hasMediaPerceptionOfObj
             ect" target="joey"/>
             <mpeg7:SpatioTemporalDecomposition
               xsi:type="mpeg7:VideoSegmentSpatioTemporalDecompositionType">
             <mpeg7:StillRegion id="rachel">
                 <mpeg7:SpatialLocator>
                    <mpeg7:Box mpeg7:dim="2 2">10 59 53 176 </mpeg7:Box>
               </mpeg7:SpatialLocator>
              </mpeg7:StillRegion>
                    <mpeg7:StillRegion id="joey">
                  <mpeg7:SpatialLocator>
                    <mpeg7:Box mpeg7:dim="2 2">147 215 67 166</mpeg7:Box>
                  </mpeg7:SpatialLocator>
                         </mpeg7:StillRegion>
             </mpeg7:SpatioTemporalDecomposition>
             <mpeg7:MediaTime>
                 <mpeg7:MediaRelIncrTimePoint>4092</mpeg7:MediaRelIncrTimePoint>
                 <mpeg7:MediaIncrDuration>28</mpeg7:MediaIncrDuration>
             </mpeg7:MediaTime>
             </mpeg7:VideoSegment>
  </mpeg7:TemporalDecomposition>
 </mpeg7:Video>
</mpeg7:MultimediaContent>
</mpeg7:Description>
```

APPENDIX E

EXAMPLE OBJECT FILE

```
<mpeg7:Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
  xsi:schemaLocation="urn:mpeg:mpeg7:schema:2001 Mpeg7-2001.xsd">
<mpeg7:Description xsi:type="SemanticDescriptionType">
      <mpeg7:Semantics>
            <mpeg7:Label>
            <mpeg7:Name>D:\Cigdem\TezAvi\Friends - 108 - Tow nana
dies twice.avi</mpeg7:Name>
            </mpeg7:Label>
            <mpeg7:SemanticBase xsi:type="ObjectType">
                  <mpeg7:Label>
                        <mpeg7:Name>cup</mpeg7:Name>
                  </mpeg7:Label>
                  <mpeg7:Relation
type="urn:...:hasMediaPerceptionOf" target="Seg0"/>
                  <mpeg7:Relation
type="urn:...:hasMediaPerceptionOf" target="Seg1"/>
            </mpeg7:SemanticBase>

            <mpeg7:SemanticBase xsi:type="ObjectType">
                  <mpeg7:Label>
                        <mpeg7:Name>joey</mpeg7:Name>
                  </mpeg7:Label>
                  <mpeg7:Relation
type="urn:...:hasMediaPerceptionOf" target="Seg2"/>
            </mpeg7:SemanticBase>
      </mpeg7:Semantics>
</mpeg7:Description>
<mpeg7:Description xsi:type="ContentEntityType">
<mpeg7:MultimediaContent xsi:type="VideoType">
      <mpeg7:Video xsi:type="VideoSegmentType">
            <mpeg7:TemporalDecomposition
xsi:type="mpeg7:VideoSegmentTemporalDecompositionType">

                  <mpeg7:VideoSegment id="Seg0">
                        <mpeg7:MediaTime>
                              <mpeg7:MediaRelIncrTimePoint
timeUnit="PT1N25F">775</mpeg7:MediaRelIncrTimePoint>
                              <mpeg7:MediaIncrDuration
timeUnit="PT1N25F">15</mpeg7:MediaIncrDuration>
                        </mpeg7:MediaTime>
                        <mpeg7:SpatioTemporalDecomposition
xsi:type="mpeg7:VideoSegmentSpatioTemporalDecompositionType">
                                    <mpeg7:StillRegion>
                                          <mpeg7:SpatialLocator>
                                                <mpeg7:Box
mpeg7:dim="2 2">127 162 173 216</mpeg7:Box>
                                          </mpeg7:SpatialLocator>
                                    </mpeg7:StillRegion>
                        </mpeg7:SpatioTemporalDecomposition>
                  </mpeg7:VideoSegment>

                  <mpeg7:VideoSegment id="Seg1">
                        <mpeg7:MediaTime>
                              <mpeg7:MediaRelIncrTimePoint
timeUnit="PT1N25F">31544</mpeg7:MediaRelIncrTimePoint>
                              <mpeg7:MediaIncrDuration
timeUnit="PT1N25F">41</mpeg7:MediaIncrDuration>
                        </mpeg7:MediaTime>
                        <mpeg7:SpatioTemporalDecomposition
xsi:type="mpeg7:VideoSegmentSpatioTemporalDecompositionType">
                              <mpeg7:StillRegion>
                                    <mpeg7:SpatialLocator>
                                          <mpeg7:Box mpeg7:dim="2
2">163 210 175 216</mpeg7:Box>
                                    </mpeg7:SpatialLocator>
                              </mpeg7:StillRegion>
                        </mpeg7:SpatioTemporalDecomposition>
                  </mpeg7:VideoSegment>

                  <mpeg7:VideoSegment id="Seg2">
                        <mpeg7:MediaTime>
                              <mpeg7:MediaRelIncrTimePoint
timeUnit="PT1N25F">3300</mpeg7:MediaRelIncrTimePoint>
                              <mpeg7:MediaIncrDuration
timeUnit="PT1N25F">30</mpeg7:MediaIncrDuration>
                        </mpeg7:MediaTime>
                        <mpeg7:SpatioTemporalDecomposition
xsi:type="mpeg7:VideoSegmentSpatioTemporalDecompositionType">
                              <mpeg7:StillRegion>
                                    <mpeg7:SpatialLocator>
                                          <mpeg7:Box mpeg7:dim="2
2">15 117 89 222</mpeg7:Box>
                                    </mpeg7:SpatialLocator>
                              </mpeg7:StillRegion>
                        </mpeg7:SpatioTemporalDecomposition>
                  </mpeg7:VideoSegment>
            </mpeg7:TemporalDecomposition>
            </mpeg7:Video>
      </mpeg7:MultimediaContent>
</mpeg7:Description>
</mpeg7:Mpeg7>
```

APPENDIX F

EXAMPLE EVENT FILE

```
<mpeg7:Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
xsi:schemaLocation="urn:mpeg:mpeg7:schema:2001 Mpeg7-2001.xsd">
<mpeg7:Description xsi:type="SemanticDescriptionType">
      <mpeg7:Semantics>
      <mpeg7:Label><mpeg7:Name>D:\Cigdem\TezAvi\Friends - 108 -
Tow nana dies twice.avi</mpeg7:Name></mpeg7:Label>
      <mpeg7:SemanticBase xsi:type="EventType">
            <mpeg7:Label>
                  <mpeg7:Name>eat</mpeg7:Name>
            </mpeg7:Label>
            <mpeg7:Definition
xsi:type="mpeg7:TextAnnotationType">
                  <mpeg7:FreeTextAnnotation
xsi:type="mpeg7:TextualType">eater:monica
                  </mpeg7:FreeTextAnnotation>
            </mpeg7:Definition>
            <mpeg7:Relation
      type="urn:...:hasMediaPerceptionOf" target="Seg52"/>
      </mpeg7:SemanticBase>
      <mpeg7:SemanticBase xsi:type="EventType">
            <mpeg7:Label>
                  <mpeg7:Name>talk on the phone</mpeg7:Name>
            </mpeg7:Label>
            <mpeg7:Definition
xsi:type="mpeg7:TextAnnotationType">
                  <mpeg7:FreeTextAnnotation
xsi:type="mpeg7:TextualType">talker:monica
                  </mpeg7:FreeTextAnnotation>
            </mpeg7:Definition>
            <mpeg7:Relation
      type="urn:...:hasMediaPerceptionOf" target="Seg53"/>
      </mpeg7:SemanticBase>
      <mpeg7:SemanticBase xsi:type="EventType">
            <mpeg7:Label>
                  <mpeg7:Name>wear coat</mpeg7:Name>
            </mpeg7:Label>
            <mpeg7:Definition
xsi:type="mpeg7:TextAnnotationType">
                  <mpeg7:FreeTextAnnotation
xsi:type="mpeg7:TextualType">wearer:monica
                  </mpeg7:FreeTextAnnotation>
            </mpeg7:Definition>
            <mpeg7:Relation
      type="urn:...:hasMediaPerceptionOf" target="Seg54"/>
      </mpeg7:SemanticBase>
      <mpeg7:SemanticBase xsi:type="EventType">
            <mpeg7:Label>
                  <mpeg7:Name>fall</mpeg7:Name>
            </mpeg7:Label>
            <mpeg7:Definition
xsi:type="mpeg7:TextAnnotationType">
                  <mpeg7:FreeTextAnnotation
xsi:type="mpeg7:TextualType">faller:ross
                  </mpeg7:FreeTextAnnotation>
            </mpeg7:Definition>
            <mpeg7:Relation
      type="urn:...:hasMediaPerceptionOf" target="Seg55"/>
      </mpeg7:SemanticBase>
      <mpeg7:SemanticBase xsi:type="EventType">
            <mpeg7:Label>
                  <mpeg7:Name>hug</mpeg7:Name>
            </mpeg7:Label>
            <mpeg7:Definition
xsi:type="mpeg7:TextAnnotationType">
                  <mpeg7:FreeTextAnnotation
xsi:type="mpeg7:TextualType">hugger1:ross hugger2:monica
                  </mpeg7:FreeTextAnnotation>
            </mpeg7:Definition>
            <mpeg7:Relation
      type="urn:...:hasMediaPerceptionOf" target="Seg56"/>
      </mpeg7:SemanticBase>
</mpeg7:Semantics>
</mpeg7:Description>

<mpeg7:Description xsi:type="ContentEntityType">
      <mpeg7:MultimediaContent xsi:type="VideoType">
            <mpeg7:Video xsi:type="VideoSegmentType">
                  <mpeg7:TemporalDecomposition
xsi:type="mpeg7:VideoSegmentTemporalDecompositionType">

                        <mpeg7:VideoSegment id="Seg52">
                              <mpeg7:MediaTime>
                                    <mpeg7:MediaRelIncrTimePoint
timeUnit="PT1N25F">2425</mpeg7:MediaRelIncrTimePoint>
                                    <mpeg7:MediaIncrDuration
timeUnit="PT1N25F">50</mpeg7:MediaIncrDuration>
                              </mpeg7:MediaTime>
                        </mpeg7:VideoSegment>

                        <mpeg7:VideoSegment id="Seg53">
                              <mpeg7:MediaTime>
                                    <mpeg7:MediaRelIncrTimePoint
timeUnit="PT1N25F">4520</mpeg7:MediaRelIncrTimePoint>
                                    <mpeg7:MediaIncrDuration
timeUnit="PT1N25F">109</mpeg7:MediaIncrDuration>
                              </mpeg7:MediaTime>
                        </mpeg7:VideoSegment>
                        <mpeg7:VideoSegment id="Seg54">
                              <mpeg7:MediaTime>
                                    <mpeg7:MediaRelIncrTimePoint
timeUnit="PT1N25F">19315</mpeg7:MediaRelIncrTimePoint>
                                    <mpeg7:MediaIncrDuration
timeUnit="PT1N25F">22</mpeg7:MediaIncrDuration>
                              </mpeg7:MediaTime>
                        </mpeg7:VideoSegment>

                        <mpeg7:VideoSegment id="Seg55">
                              <mpeg7:MediaTime>
                                    <mpeg7:MediaRelIncrTimePoint
timeUnit="PT1N25F">21833</mpeg7:MediaRelIncrTimePoint>
                                    <mpeg7:MediaIncrDuration
timeUnit="PT1N25F">47</mpeg7:MediaIncrDuration>
                              </mpeg7:MediaTime>
                        </mpeg7:VideoSegment>

                        <mpeg7:VideoSegment id="Seg56">
                              <mpeg7:MediaTime>
                                    <mpeg7:MediaRelIncrTimePoint
timeUnit="PT1N25F">23902</mpeg7:MediaRelIncrTimePoint>
                                    <mpeg7:MediaIncrDuration
timeUnit="PT1N25F">48</mpeg7:MediaIncrDuration>
                              </mpeg7:MediaTime>
                        </mpeg7:VideoSegment>
                  </mpeg7:TemporalDecomposition>
            </mpeg7:Video>
      </mpeg7:MultimediaContent>
</mpeg7:Description>
</mpeg7:Mpeg7>
```

APPENDIX G

EXAMPLE FRAME SEGMENT TREE FILE

```
mpeg7:Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
xsi:schemaLocation="urn:mpeg:mpeg7:schema:2001 Mpeg7-2001.xsd">
<mpeg7:Description xsi:type="ContentEntityType">
      <mpeg7:MultimediaContent xsi:type="VideoType">
            <mpeg7:Video xsi:type="VideoSegmentType">
            <mpeg7:MediaInformation>
                  <mpeg7:MediaProfile>
                        <mpeg7:MediaInstance>
                              <mpeg7:MediaLocator>
                        <mpeg7:MediaUri>D:\Cigdem\TezAvi\Friends
- 108 - Tow nana dies twice.avi</mpeg7:MediaUri>
                              </mpeg7:MediaLocator>
                        </mpeg7:MediaInstance>
                  </mpeg7:MediaProfile>
            </mpeg7:MediaInformation>

            <mpeg7:MediaTime>
                  <mpeg7:MediaTimePoint
timeUnit="PT1N25F">0</mpeg7:MediaTimePoint>
                  <mpeg7:MediaDuration
timeUnit="PT1N25F">1309560</mpeg7:MediaDuration>
            </mpeg7:MediaTime>

            <mpeg7:TemporalDecomposition
xsi:type="mpeg7:VideoSegmentTemporalDecompositionType">

                  <mpeg7:VideoSegment id="Seg61">
                        <mpeg7:Relation source="Seg61"
type="urn:mpeg:mpeg7:cs:SemanticRelationCS:2001:hasMediaPerceptio
nOfObject" target="cup"/>
                        <mpeg7:SpatioTemporalDecomposition
xsi:type="mpeg7:VideoSegmentSpatioTemporalDecompositionType">
                              <mpeg7:StillRegion id="cup">
                                    <mpeg7:SpatialLocator>
<mpeg7:Box mpeg7:dim="2 2">127 162 173 216 </mpeg7:Box>
                                    </mpeg7:SpatialLocator>
                              </mpeg7:StillRegion>
                        </mpeg7:SpatioTemporalDecomposition>
                        <mpeg7:MediaTime>
<mpeg7:MediaRelIncrTimePoint>775</mpeg7:MediaRelIncrTimePoint>
<mpeg7:MediaIncrDuration>15</mpeg7:MediaIncrDuration>
                        </mpeg7:MediaTime>
                  </mpeg7:VideoSegment>

                  <mpeg7:VideoSegment id="Seg62">
                        <mpeg7:Relation source="Seg62"
type="urn:mpeg:mpeg7:cs:SemanticRelationCS:2001:hasMediaPerceptio
nOfEvent" target="eat"/>
                        <mpeg7:SpatioTemporalDecomposition
xsi:type="mpeg7:VideoSegmentSpatioTemporalDecompositionType">
                        </mpeg7:SpatioTemporalDecomposition>
                        <mpeg7:MediaTime>
<mpeg7:MediaRelIncrTimePoint>2425</mpeg7:MediaRelIncrTimePoint>
<mpeg7:MediaIncrDuration>50</mpeg7:MediaIncrDuration>
                        </mpeg7:MediaTime>
                  </mpeg7:VideoSegment>

                  <mpeg7:VideoSegment id="Seg63">
                        <mpeg7:Relation source="Seg63"
type="urn:mpeg:mpeg7:cs:SemanticRelationCS:2001:hasMediaPerceptio
nOfObject" target="joey"/>
                        <mpeg7:SpatioTemporalDecomposition
xsi:type="mpeg7:VideoSegmentSpatioTemporalDecompositionType">
                              <mpeg7:StillRegion id="joey">
                                    <mpeg7:SpatialLocator>
<mpeg7:Box mpeg7:dim="2 2">15 117 89 222 </mpeg7:Box>
                                    </mpeg7:SpatialLocator>
                              </mpeg7:StillRegion>
                        </mpeg7:SpatioTemporalDecomposition>
                        <mpeg7:MediaTime>
<mpeg7:MediaRelIncrTimePoint>3300</mpeg7:MediaRelIncrTimePoint>
<mpeg7:MediaIncrDuration>30</mpeg7:MediaIncrDuration>
                        </mpeg7:MediaTime>
                  </mpeg7:VideoSegment>

                  <mpeg7:VideoSegment id="Seg64">
                        <mpeg7:Relation source="Seg64"
type="urn:mpeg:mpeg7:cs:SemanticRelationCS:2001:hasMediaPerceptio
nOfObject" target="plate"/>
                        <mpeg7:SpatioTemporalDecomposition
xsi:type="mpeg7:VideoSegmentSpatioTemporalDecompositionType">
                              <mpeg7:StillRegion id="plate">
                                    <mpeg7:SpatialLocator>
<mpeg7:Box mpeg7:dim="2 2">142 211 143 171 </mpeg7:Box>
                                    </mpeg7:SpatialLocator>
                              </mpeg7:StillRegion>
                        </mpeg7:SpatioTemporalDecomposition>
                        <mpeg7:MediaTime>
<mpeg7:MediaRelIncrTimePoint>3480</mpeg7:MediaRelIncrTimePoint>
<mpeg7:MediaIncrDuration>100</mpeg7:MediaIncrDuration>
                        </mpeg7:MediaTime>
                  </mpeg7:VideoSegment>

                  <mpeg7:VideoSegment id="Seg65">
                        <mpeg7:Relation source="Seg65"
type="urn:mpeg:mpeg7:cs:SemanticRelationCS:2001:hasMediaPerceptio
nOfObject" target="monica"/>
```

```
                        <mpeg7:Relation source="Seg65"
type="urn:mpeg:mpeg7:cs:SemanticRelationCS:2001:hasMediaPerceptio
nOfObject" target="chandler"/>
                        <mpeg7:SpatioTemporalDecomposition
xsi:type="mpeg7:VideoSegmentSpatioTemporalDecompositionType">
                              <mpeg7:StillRegion id="monica">
                                    <mpeg7:SpatialLocator>
<mpeg7:Box mpeg7:dim="2 2">18 122 67 236 </mpeg7:Box>
                                    </mpeg7:SpatialLocator>
                              </mpeg7:StillRegion>
                              <mpeg7:StillRegion id="chandler">
                                    <mpeg7:SpatialLocator>
<mpeg7:Box mpeg7:dim="2 2">150 243 39 235 </mpeg7:Box>
                                    </mpeg7:SpatialLocator>
                                    </mpeg7:StillRegion>
                        </mpeg7:SpatioTemporalDecomposition>
                        <mpeg7:MediaTime>
<mpeg7:MediaRelIncrTimePoint>3930</mpeg7:MediaRelIncrTimePoint>
<mpeg7:MediaIncrDuration>4</mpeg7:MediaIncrDuration>
                        </mpeg7:MediaTime>
                  </mpeg7:VideoSegment>
            </mpeg7:TemporalDecomposition>
            </mpeg7:Video>
      </mpeg7:MultimediaContent>
</mpeg7:Description>
</mpeg7:Mpeg7>
```