FMDBMS - A FUZZY MPEG-7 DATABASE MANAGEMENT SYSTEM

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

BY

NAZİF İLKER ERÇİN

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

JUNE 2012

Approval of the thesis:

FMDBMS - A FUZZY MPEG-7 DATABASE MANAGEMENT SYSTEM

submitted by **NAZİF İLKER ERÇİN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering** _____

Prof. Dr. Adnan Yazıcı
Supervisor, **Computer Engineering Department, METU** _____

Examining Committee Members:

Prof. Dr. Özgür Ulusoy
Computer Engineering Department, Bilkent University _____

Prof. Dr. Adnan Yazıcı
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Ahmet Coşar
Computer Engineering Department, METU _____

Assist. Prof. Dr. Murat Koyuncu
Information Systems Engineering Department, Atılım University _____

Assist. Prof. Dr. Mustafa Sert
Computer Engineering Department, Başkent University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: NAZİF İLKER ERÇİN

Signature :

ABSTRACT

FMDBMS - A FUZZY MPEG-7 DATABASE MANAGEMENT SYSTEM

Erçin, Nazif İlker
M.Sc., Department of Computer Engineering
Supervisor : Prof. Dr. Adnan Yazıcı

June 2012, 103 pages

Continuous progress in multimedia research in recent years have led to proliferation of their applications in everyday life. The ever-growing demand in high performance multimedia applications creates the need for new and efficient storage and retrieval techniques.

There exist numerous studies in the literature attempting to describe the content of these multimedia documents. Moving Picture Experts Group's XML based MPEG-7 is one of these studies that makes it possible to describe multimedia content in terms of both low and high level properties. MPEG-7 DDL allows defining new types using already defined types. Within the past ten years, it became a widely accepted standard in multimedia applications.

In this thesis, an XML database application is developed to manage MPEG-7 descriptions, utilizing eXist XML DB as the database management system and a JAVA application as the frontend. MPEG-7 Description Schemes are extended by introducing fuzzy semantic types, such as FuzzyObject and FuzzyEvent, using the MPEG-7 DDL. From this point of view, the application of fuzzy XML methods in MPEG-7 standard is a novel approach.

Keywords: Fuzzy, Multimedia, MPEG-7, Database, XML

ÖZ

FMDBMS - BULANIK MPEG-7 VERİTABANI YÖNETİM SİSTEMİ

Erçin, Nazif İlker
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Adnan Yazıcı

Haziran 2012, 103 sayfa

Son yıllarda çokluortam uygulamalarında birçok ilerleme kaydedilmiştir. Bu ilerleme, yeni ve verimli veri saklama ve sorgulama tekniklerine olan ihtiyacı arttırmıştır.

Çokluortam dokümanlarının içeriğini ifade etmeye yönelik bir çok çalışma yapılmıştır. Moving Pictures Experts Group tarafından yapılan bir çalışma, MPEG-7, çokluortam doküman içeriklerinin düşük ve yüksek seviye özellikleri de içerecek şekilde saklamayı mümkün kılmıştır. MPEG-7 standardı XML tabanlıdır. MPEG-7 DDL, tanımlı tipleri kullanarak yeni tipler yaratabilmeyi sağlar. Geçtiğimiz on yıl süresince, MPEG-7 çok sayıda çokluortam çalışmasında kullanılır hale gelmiştir.

Bu tez çalışmasında, MPEG-7 tanımlarını yönetebilmeyi sağlayacak bir XML veritabanı uygulaması oluşturulmuştur. Sistemde veritabanı yönetimi için eXist XML DB, kullanıcı etkileşimi için ise bir JAVA uygulaması kullanılmıştır. MPEG-7 Tanım Şemaları, MPEG-7 DDL kullanılarak, bulanık anlamsal tiplerle (BulanıkNesne, BulanıkOlay vb.) genişletilmiştir.

Anahtar Kelimeler: Bulanık, Çokluortam, MPEG-7, Veritabanı, XML

To my grandparents

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude to my advisor, Prof. Dr. Adnan Yazıcı, for his great support throughout this study. I would also like to say thanks to my thesis committee members, Prof. Dr. Özgür Ulusoy, Assoc. Prof. Dr. Ahmet Coşar, Assist. Prof. Dr. Murat Koyuncu and Assist. Prof. Dr. Mustafa Sert for their valueable feedbacks and support.

Mom, dad and my little brother, I know I ignored you a lot during the past one and a half year, but I promise, I will make it up to you from now on. Thank you for always being there for me. Although you are far away, I always knew that you are by my side and you will always be. A part of this study is yours.

I also want to mention my dearest friends and home mates, Ender Erel and Semih Yağcıoğlu, for always helping me through my stressful times and encouraging me whenever I need a push. Thanks guys, we would not be reading these words if you were not in my life. A part of this study is yours.

Çağrı Çetintepe, you are the smartest thing I know on the whole planet. I will never forget your support. I will never forget the horror movie nights and the tasty beer with jager. Those were some of the few things that kept me as a human being during these times. A part of this thesis is yours.

My dearest friends, brothers and sisters, in METU Computer Engineering Department, Çiğdem Okuyucu, Merve Aydınlılar, Elvan Gülen, Kerem Hadımlı, Can Hoşgör, Serra Sinem Tekiroğlu, Meryem Ayas, Çağla Çığ and Utku Erdoğdu, thanks for being my friends, thanks for everything. I know even when we were away from each other, our minds keep us together. I feel your presence in every single page of this thesis.

Another person I would like to mention, who I would say that we share the same destiny with, Gökhan Uğurel, we fought the hard times together and you were the only one I believe who understands me the best. Thanks for the earl gray supply in all weekends we shared studying. I dedicate a part of this thesis to you.

I would also like to state that I am grateful to my employer, Aselsan A.Ş., and all my co-workers and friends, Emre Turgay, Buse Gül Atlı, Gökhan Tanışık, Alparslan Mustafa Çil and my project manager Hikmet Balcı for their precious support almost everyday. It is an honour to work with you. A part of thesis is yours.

Eren Arslan and İlke Mırık, I cannot finish this without mentioning your names. I would not be who I am without you and this moment would only be a dream in my life. I know you will not accept this but I am aware that I owe you a lot. If you let me, I would like to share a part of this thesis with you.

I will say my last words to the most precious one, who survived all this depressing times with me, ignoring her many chances to get away whenever she wants. Güliz Kazanç, now you know how to create a thesis study, you completed it with me, by my side, living every horrible detail, facing the most disgusting side of me every single day. Thanks for everything, thanks for being my dearest thing. The rest of this thesis is completely yours.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

The growth in the type and number of multimedia files in the past one and a half decade, reveals the problem of managing this vast amount of data in a faster and more efficient manner. This problem can be considered as the combination of a number of sub-problems, mainly storage, annotation and querying of multimedia documents.

Multimedia documents differ from text documents in that text content is self descriptive while multimedia files are not. What makes a multimedia document meaningful is its semantic content, because the low-level audiovisual data are meaningless to human perception [1]. For instance, if a person looks at a photograph of a butterfly, the inferred meaning is not the color or pixel information; but it is *the state of being a butterfly photograph* which is the semantic content. This content-based nature of multimedia documents highlights the requirement of describing their semantic content.

MPEG-7 is a standard proposed in order to unify the multimedia content descriptions [2][3]. MPEG-7 enables one to describe the semantic content of multimedia documents, supplying tools to describe events, objects, relations etc. MPEG-7 descriptions are XML files and its description tools are implemented using XSD schemas. The predefined schemes of MPEG-7 MDS are able to describe visual, audio and general purpose multimedia content. The MPEG-7 standard, however, does not limit applications to these predefined schemes: Using MPEG-7 DDL, one can extend these predefined schemes or create new types whenever needed [4].

Fuzzy set theory [5] is introduced by Zadeh in 1965. The difference of fuzzy sets from classical sets is that the membership to a fuzzy set is a matter of degree whereas the membership

to a classical set is a binary value. Fuzzy logic is a branch of fuzzy set theory [6]. From the perspective of fuzzy logic, linguistic variables are of primary importance. Using linguistic variables, one can declare an attribute, say *Length*, that may have a linguistic value such as *short*, *medium* or *long*.

XML is a text-based, simple and a very flexible format, recommended by W3C and became the most commonly used tool to transform a wide variety of data in between a large number of applications and WEB [7][8]. XSD, which is a special form of XML, is introduced by W3C and can be used to define the content of an XML instance [9][10]. XQuery is a query language that became a W3C recommendation in 2007 [11]. XQuery is designed to query and modify anything, e.g., documents and databases, as long as it is in XML format.

As mentioned before, MPEG-7 descriptions are XML documents and for this reason, one must successfully manage XML documents to be able to handle MPEG-7 descriptions properly [4]. From the database point of view, there are two major approaches toward that end, namely native XML databases and XML extensions of relational DBMSs [4]. BaseX, eXist and Tamino are examples of the former and Oracle XML DB and IBM DB2 pureXML are examples of the latter. In [4], the state of art XML database solutions are tested for their ability to manage MPEG-7 descriptions in terms of representation of media descriptions, access to media descriptions, media description schemes, extensibility and classic DBMS functionality.

In order to include fuzziness in the database area, two approaches have been considered in the literature: The first approach is to query crisp data in a fuzzy manner, and second one is to store the fuzzy data itself in the database [12]. These techniques can be equally applied to XML documents and doing so translates to performing fuzzy querying on XML data and representing fuzziness as the XML data. Possible methodologies for the querying of XML data include point, range and KNN queries [13], schema utilization for query optimization [14], usage of fuzzy conditions and similarity relations [15][16], modeling of XML content and queries as weighted graphs and employing graph matching between them [17]. Using fuzzy linguistic terms as element and attribute values [12][15][16][18] and introducing possibility distributions in XML to define fuzzy content of an element [18][19] are some of the methods to store fuzzy data in XML documents.

The concepts summarized in this overview section are examined in more detail in Chapters 2 and 3.

1.2 Contribution of the Thesis

The work presented in this thesis contributes to the previously conducted studies in the following aspects:

- Introducing possibility distributions to MPEG-7 media descriptions,
- Introducing fuzzy linguistic terms as attribute and element values of MPEG-7 semantic descriptions,
- Introducing possibility distributions as element values of MPEG-7 semantic descriptions,
- Introducing fuzzy semantic types as an extension to MPEG-7 MDS,
- Querying of the semantic content of MPEG-7 description documents, either with fuzzy or crisp query attributes.

1.3 Organization

The organization of the rest of this thesis is follows. Following this chapter, Chapter 2 lists and explains the standards, tools and former studies employed in this thesis. Then, Chapter 3 provides a detailed investigation of the previous studies related to this work. Next, Chapter 4 elaborates on the first step of this thesis study, namely, fuzzy extension of MPEG-7 semantic tools. Chapter 5 then gives details of the XML database design and the implemented FMDBMS client application. The implementation details of Document Content Manager Tool is explained in detail in Chapter 6. Subsequently, Chapter 7 provides sample querying scenarios using the FMDBMS client, clarifying the extraction of queried data and calculation of possibility values. Finally, Chapter 8 briefly summarizes the overall thesis study and suggests potential future work.

CHAPTER 2

BACKGROUND

2.1 Fuzzy Set Theory and Fuzzy Logic

Theory of fuzzy sets [5] is introduced by L.A. Zadeh in 1965. A fuzzy set extends the notion of classical set by introducing the term partial membership to a set: As opposed to the bivalent nature of classical sets, i.e. an object is a member of a set or not, a fuzzy set supplies a membership degree (μ) of any value in the range between zero and one. The value of membership of all possible members in the universe of discourse is defined using a membership function. A membership function is a function which maps all members in the universe to a real number in the range between zero and one, which is the membership degree itself. A fuzzy set is empty, when its membership function is level zero for any member in the space. As in the regular sets, *complement*, *containment*, *union*, *intersection* operations are also defined for fuzzy sets.

Fuzzy logic [6] is a multi-valued logic dealing with approximate rather than exact reasoning. It is an extension of traditional logic theory, to handle partial truth, which is a value of truth lying between true and false.

Membership functions, as stated before, are implemented in order to define the membership values of an object or an instance to a class. These functions appear in different forms where triangular, trapezoidal and Gaussian forms are some examples. As an illustration, Figure 2.1 displays the membership functions for linguistic terms *Slow*, *Average-Speed* and *Fast*. These are the possible values of the linguistic variable *Speed*. The values can be also quantified in a fuzzy manner, using the fuzzy quantifiers. For example, consider the fuzzy quantifier *very*. The fuzzy value of the linguistic variable *Speed* can be quantified with the quantifier *very* and

fuzzy values like *Very Fast* or *Very Slow* can be also declared.

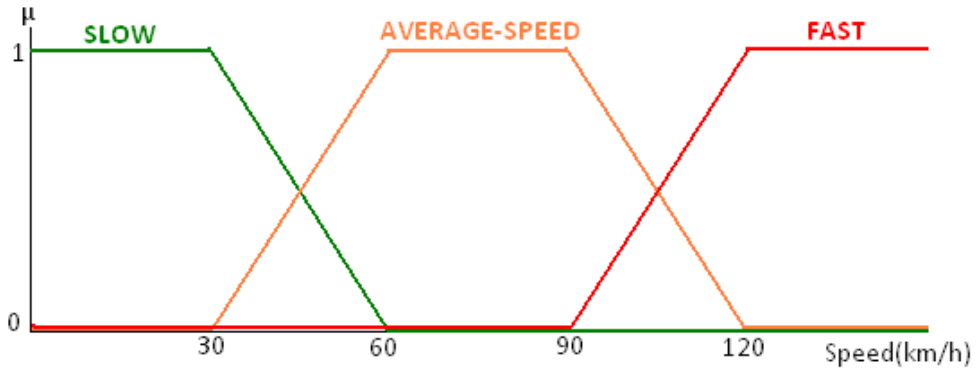


Figure 2.1: Membership functions of Slow, Average-Speed and Fast classes.

As another example, let the membership functions in Figure 2.1 is for implementing a fuzzy brake-support system for vehicles. The strength of brake-support will be determined using the *Speed* and *Weight* variables. Toward that end, a set of fuzzy IF-THEN rules should be declared in the system, e.g. *IF Speed is FAST and Weight is HEAVY THEN Brake-Strength is HIGH*. Fuzzy rule sets, combined with the implication methods, helps create an approximate result for *Brake-Strength* variable, which would be applied on the vehicle's brakes in order to help slowing down the vehicle.

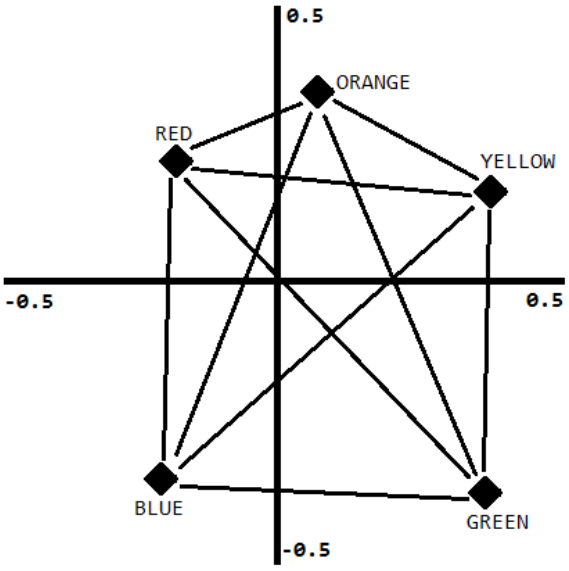
In addition to membership functions, the relation between fuzzy linguistic terms can be defined using a similarity relation [20]. Similarity relations are defined over attribute domains, in order to support tolerance and interchangeability between different linguistic terms [12]. Similarity relations between linguistic terms are generally represented by a table whose columns and rows include the linguistic terms and their similarity in the range [0, 1]. Following our previous example, Table 2.1 shows a sample similarity relation matrix between *Slow*, *Average-Speed* and *Fast* linguistic terms. Using only the given matrix in Table 2.1 and nothing else, if a user asks for "Select all vehicles with Speed=SLOW", the query would return results including *AVERAGE-SPEED* vehicles with 0.4 similarity value and *FAST* vehicles with 0.0001 similarity value. As can be seen, the results can be easily filtered applying a threshold on similarity values.

Table 2.1: Similarity matrix of Slow, Average-Speed and Fast linguistic terms.

	SLOW	**AVERAGE-SPEED**	**FAST**
SLOW	1	0.4	0.0001
AVERAGE-SPEED	0.4	1	0.43
FAST	0.0001	0.43	1

2.2 Similarity and Possibility

Similarity relations constitute an important part of fuzzy relations, in which the similarity values are used to express the degree of similarity of objects in a universe [21]. In [21], it is stated that any object can be represented as a point in a pseudo-metric coordinate space and smaller distances between objects indicate higher similarity values. As a result, it can be derived that the similarity between two objects can be represented as *(1 - d(u,v))*, where *d(u, v)* is the unit distance between objects *u* and *v*. At this point it should be noted that all of the objects in the universe are located in a unit hyper-sphere. An illustration for the relation between unit distance functions and similarity relations can be seen in Figure 2.2.



DISTANCES
d(RED,ORANGE) = 0.12
d(RED,YELLOW) = 0.54
d(RED,BLUE) = 0.77
d(RED,GREEN) = 0.83
d(BLUE,ORANGE) = 0.85
d(BLUE,YELLOW) = 0.86
d(BLUE,GREEN) = 0.67
d(GREEN,YELLOW) = 0.51
d(GREEN,ORANGE) = 0.78
d(YELLOW,ORANGE) = 0.2

Similarity Matrix

	RED	ORNG	YLLW	GRN	BLUE
RED	1	0.88	0.46	0.23	0.17
ORNG	0.88	1	0.8	0.15	0.22
YLLW	0.46	0.8	1	0.14	0.49
GRN	0.23	0.15	0.14	1	0.33
BLUE	0.17	0.22	0.49	0.33	1

Figure 2.2: Unit distances and similarity relations.

Possibility distributions are discrete forms of membership functions, which define the degree of compatibility of an object in the universe with a specific concept [21]. An object *p* (which

does not necessarily exist in the universe) that fully satisfies a concept, i.e. $Poss(p) = 1$, is called a *paradigmatic example* of that concept. Combining the idea of paradigmatic examples and the distance functions defining a similarity relation, it can be stated that the possibility value of an object for a concept is equal to the similarity of the object to the paradigmatic example of the concept. Here it should be noted that the objects in the universe and the paradigmatic example object are all located in a unit hyper-sphere. A sample demonstration of the relation between unit distance functions and possibility distributions can be seen in Figure 2.3.

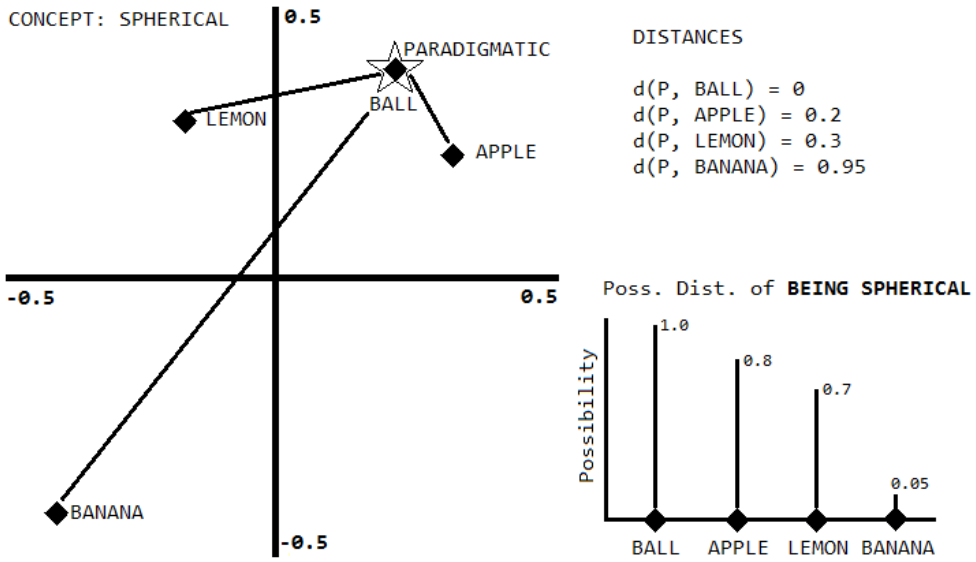


Figure 2.3: Unit distances and possibility distributions.

2.3 Weighted Averaging

Weighted arithmetic averaging (WA) is a form of arithmetic averaging, where the effect of each term is modified with a corresponding weight value. In this study, WA is used as a conjunction operator, providing the ability to weight the fuzzy query predicates and let them affect the resulting value more or less depending on their weight. The adopted formula for the conjunction using WA is

$$\text{Weighted\_Conjunction}(P, W) = \frac{\sum_{i=1}^{n} w_i \times p_i}{\sum_{i=1}^{n} w_i} \tag{2.1}$$

$$P = \{p_1, p_2, ..., p_n\}$$

$$W = \{w_1, w_2, ..., w_n\}$$

represent the sets of predicates and weights respectively.

2.4 XML and XML Schema

XML is a text-based, flexible, simple and self-descriptive format, derived from SGML, which became a W3C recommendation in 1998 [7][8]. The motivation of XML is to structure, store and transport the data, rather than displaying it. XML, however, is not able to do anything by itself: XML only contains the data in tags, and a software is needed to send, receive and display XML data. In real world, different applications have different and probably incompatible formats of data representation. XML standard enables simplification and unification of data storage, thus forming a methodology for data representation which is independent of software and hardware.

A well-formed XML document has a tree structure so it must include a root element. Figure 2.4 illustrates a sample well-formed XML document and its corresponding tree structure. Here the tree is made up of three levels of elements.

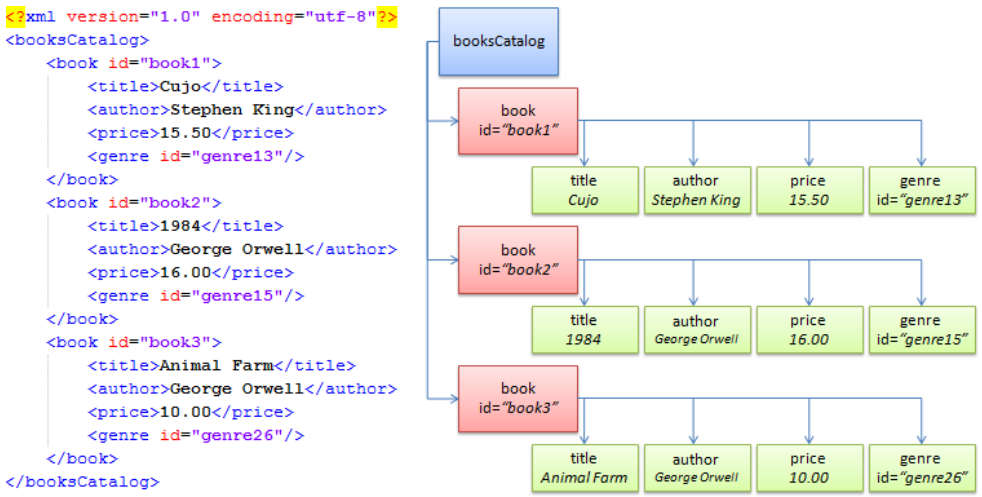


Figure 2.4: Sample XML document and corresponding tree structure.

In XML notation, an element is composed of starting and ending tags, and everything in between those tags. An element may contain other elements, text and attributes [8]. In Figure 2.4, *booksCatalog* is an element carrying *book* element as child elements. *book* elements carry *id* as their attributes. *title*, *author* and *price* are sibling elements carrying textual data and no attributes. Elements can also be empty [10], like the *genre* elements in Figure 2.4. Empty elements are not allowed to carry any data inside but attributes.

XML schema (XSD) is a special kind of XML document that can be used to define the structure of XML documents [9][10]. XSD enables one to define simple and complex elements. Simple types may carry only textual data and they cannot contain other elements or attributes. XSD has a number of built-in types where *xs:string*, *xs:integer*, *xs:float*, *xs:boolean*, *xs:date* and *xs:time* are the most commonly used ones. Complex types can carry other elements or attributes. Empty elements are complex elements. Using XSD, one is able to declare recursive types, i.e. a complex element may include an element of its own type.

2.5 XQuery

XQuery is a querying language that is designed to query the XML content, exploiting the structured format of XML [11][22]. XQuery became a W3C recommendation in 2007 and it is compatible with many W3C standards such as XML, Namespaces, XSLT, XPath and XML Schema.

In XQuery, queries are formed to be FLWOR expressions, which is an acronym for *For*, *Let*, *Where*, *Order By* and *Return*. *For* is used to iterate on XML nodes. *Let* is for assigning values to variables. *Where* clauses are the places to specify criteria of a selection. *Order By* is used to order the output in terms of the given element or attribute, or a combination of both. *Return*, as the name implies, is to specify the output of the query. On the XML document given in Figure 2.4, one can execute queries like *"select books written by George Orwell where genre id is genre26"* or *"select books where price is greater than 12.00"*. XQuery also allows one to modify the XML content by executing queries like *"update titles of books where price >15 as 'title'+'-(expensive)' "*[1].

XQuery supplies a set of functions with more than a hundred members to use in queries. It

[1] For the sake of clarity, sample queries are not given in XQuery format.

also allows one to declare and employ user-defined functions.

2.6 MPEG-7

MPEG-7 is an ISO/IEC standard developed by Moving Pictures Experts Group [2]. The same committee previously published the standards MPEG-1 [23], MPEG-2 [24] and MPEG-4 [25]. In particular, MPEG-1 is for coding the video and audio content for digital storage, MPEG-2 deals with the generic coding of visual and audio information, and MPEG-4 provides standardized elements in order to enable the integration of the production, distribution and content access paradigms of the three fields, namely digital television, interactive graphics applications and interactive multimedia. While these three standards deal with the technical background of the multimedia content, the objective of MPEG-7 is to describe the metadata of it.

MPEG-7 has become a widespread standard since it was first published. Some of its application areas include digital libraries, multimedia editing, security services, educational applications and biomedical applications. MPEG-7 motivates the researchers to standardize the multimedia descriptions. Even some already defined multimedia description tools tried to map their work on MPEG-7 standard [3].

MPEG-7 makes use of the following tools to describe the multimedia content:

- Descriptors (Ds): Descriptors are used for defining the syntactic and semantic structure of features. *Silence D* is an example of audio descriptors.
- Description Schemes (DSs): Description schemes are implemented to define the structure and semantics of Descriptors and Description Schemes, that both can be found as a content of a description scheme. *Object DS* is an example of semantic content description schemes.
- Description Definition Language (DDL): Description definition language is implemented to enable one to create new description schemes and descriptors, and it can be used to extend and modify the already defined description schemes.
- System Tools: System tools are for binary coding, efficient storage and transmission, multiplexing of descriptions, management and protection purposes.

MPEG-7 is composed of the following parts:

- MPEG-7 Systems: The set of tools to prepare MPEG-7 descriptions for efficient storage and transportation issues.
- MPEG-7 Description Definition Language: Defines the syntax of the MPEG-7 Description Tools and can be used in order to define new description schemes.
- MPEG-7 Visual: Contains the description tools for visual descriptions.
- MPEG-7 Audio: Contains the description tools for audio descriptions.
- MPEG-7 Multimedia Description Schemes: Contains the description tools for generic descriptions.
- MPEG-7 Reference Software: An implementation of relevant parts of MPEG-7 with normative status.
- MPEG-7 Conformance Testing: Includes the instructions and procedures for testing conformance of MPEG-7 implementation.
- MPEG-7 Extraction and Use of Descriptions: A technical report including information about extracting and using description tools.
- MPEG-7 Profiles and Levels: Contains standard MPEG-7 profiles and levels.
- MPEG-7 Schema Definition: Schema definition of standard description tools.

2.6.1 Multimedia Description Schemes

An overview of MPEG-7 multimedia description schemes (MDSs) is given in Figure 2.5 [26]. As it can be observed from Figure2.5, MDSs can be divided into five main parts, namely, basic elements, content organization, content descriptions & management, navigation & access and user interaction [27].

"Basic elements" part contains five sets of description tools, which are schema tools, basic datatypes, mathematical structures, linking and media localization tools, and basic description schemes. "Content description & management" part is built on the "basic elements" part,

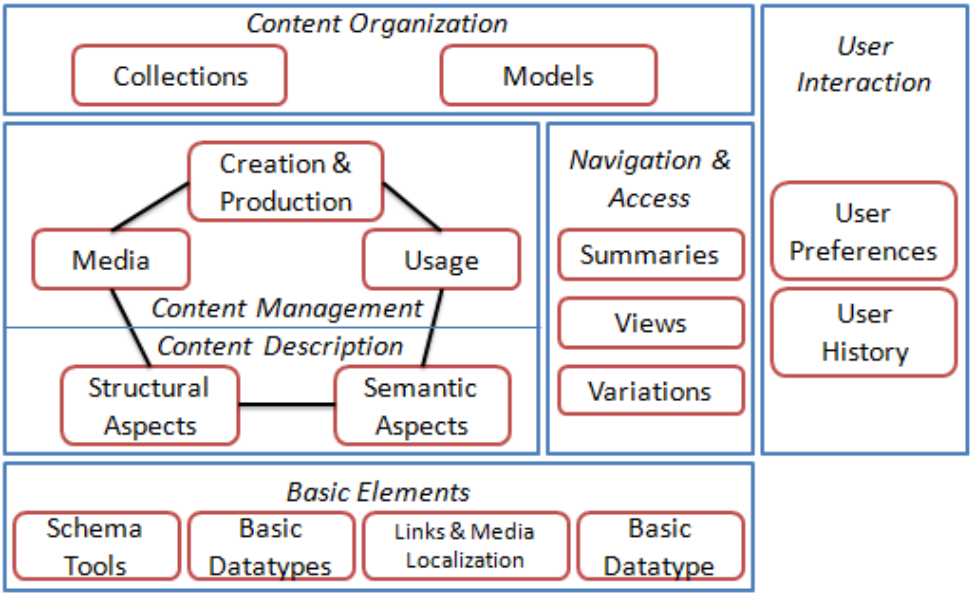


Figure 2.5: Overview of multimedia description schemes.

which is in turn divided into five sets. These sets and the corresponding descriptions are as follows [26]:

- Media: Media information describes the storage media, in terms of format, compression and coding of audio-visual content. *Media Information DS* and *Media Profiles* are employed in order to identify the media.
- Creation & Production: Creation and production, as the name implies, describes the creation and production information of content. This information is composed of features such as title, creator, classification, purpose of extraction, and generally generated manually by the user.
- Usage: Usage information describes usage information, including features like rights holders, access rights, publication and financial information.
- Structural Aspects: This part deals with describing the content in terms of its structure. The segments representing physical, spatial, temporal and spatio-temporal components form the structure of descriptions. Segments are represented by some signal-based features, such as color, texture, shape and motion, and some semantic information.
- Semantic Aspects: This part is used in order to describe the content from semantic

point of view, using semantic features such as events, objects, actors and relationship between these features.

2.6.2 Visual Descriptors

MPEG-7 standard supplies a set of visual low-level descriptors to describe the content of a visual multimedia document. The ones that are used in this thesis are listed and explained as follows [28]:

- **Color Layout:** Color layout descriptor is implemented to represent spatial distribution of colors of an image in YCbCr color space.
- **Dominant Color:** Dominant color descriptor is implemented to summarize the color information of an image, using the color spaces RGB, YCbCr, HVS and HMMD.
- **Scalable Color:** Scalable color descriptor is implemented to represent the color histogram of an image in HSV color space.
- **Edge Histogram:** Edge histogram descriptor is implemented to represent the spatial distributions of edges in an image.
- **Homogeneous Texture:** Homogeneous texture descriptor is implemented to represent statistical distribution of the texture of an image by making use of mean energy and energy deviation of frequency layout channels.
- **Contour Shape:** Contour shape descriptor is implemented to represent a shape based on Curvature Scale-Space.
- **Region Shape:** Region shape descriptor is implemented to represent the shape of an object by its edge and filling pixels.

2.6.3 Semantic Description Schemes

MPEG-7 MDS provides tools for describing the content in terms of semantics [26][29]. These tools can be used to describe the narrative world, in terms of objects, events, concepts, places, times and the relations in between. The whole set of semantic description tools can be divided into three, namely, semantic entity tools, semantic attribute tools and semantic relation tools.

Figure 2.6 illustrates the model of MPEG-7 semantic tools to describe the semantics of the audio-visual content “Cristiano Ronaldo shoots the ball in Estádio da Luz” [26].

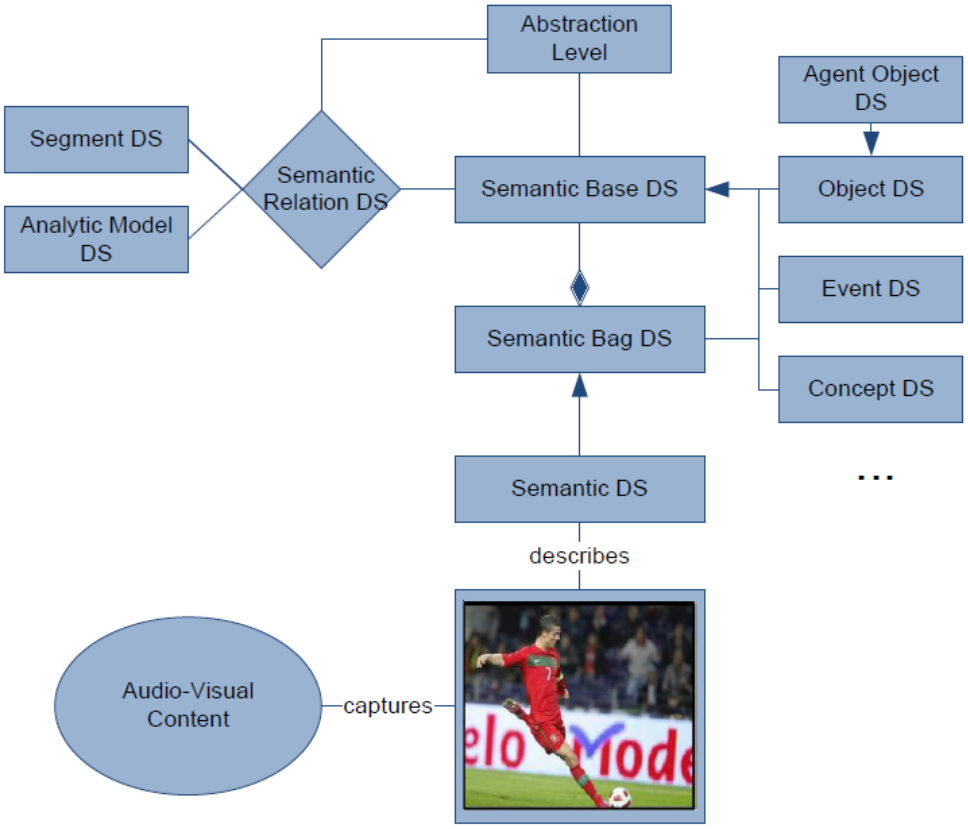


Figure 2.6: MPEG-7 semantic description tools.

The semantic entity tools given in Figure 2.6 can be described as follows:

- **SemanticBase DS:** This tool is defined in order to describe a semantic entity in narrative world. This is an abstract type provided to derive specialized tools like Semantic DS, Object DS, AgentObject DS, Event DS, Concept DS, SemanticState DS, SemanticPlace DS and SemanticTime DS. These specialized tools are also described below.

- **SemanticBag DS:** SemanticBag is an abstract type. Any semantic description tool to define a set of semantic entities and their relations is derived from SemanticBag DS.

- **Semantic DS:** Semantic DS tool is used to describe a narrative world captured by the multimedia content.

- **Object DS:** Object DS is used to describe an existing entity in narrative world in terms of time and space (e.g. the ball hit by Cristiano Ronaldo) or an abstraction of it (e.g. a ball).

- **AgentObject DS:** This type is derived from Object DS in order to describe an agent, which might be a person, a group of people, an organization or a personalized object (e.g. a speaking pencil in a cartoon).

- **Event DS:** Event DS is specialized in order describe an action in narrative world in temporal and spatial manner, including one or more objects (e.g. Cristiano Ronaldo shoots the ball) or an abstraction of it (e.g. a player shoots the ball).

- **Concept DS:** Concept DS is developed to enable description of a semantic entity that cannot be described as a generalization or an abstraction of a specific entity. Concepts are formally a collection of properties.

- **SemanticState DS:** In order to represent a collection of numerical and verbal attributes (e.g. weight of the ball, quality of the pitch) that is attached a semantic entity, SemanticState DS is implemented.

- **SemanticPlace DS:** SemanticPlace DS describes a semantic place in a narrative world (e.g. Estádio da Luz).

- **SemanticTime DS:** SemanticTime DS describes a time in narrative world (e.g. fifteenth minute of the match, night hours).

2.6.4 MPEG-7 Profiles

MPEG-7 profiling can be easily described as selecting the descriptors and description schemes related to a specific application domain in order to reduce the complexity and prevent the ambiguity caused by semantically same descriptions having different description contents which leads to interoperability problems between different applications.

Profiling is conducted in three steps [30]:

- Selecting description tools: The number of descriptors and description tools are reduced, selecting the related ones to be used in a specific domain.

- Determining description tool constraints: The number of elements for description tools and attribute usage is determined.
- Determining semantic constraints: Usage of selected and constrained tools to describe multimedia content from selected application domain is determined, expressed in natural language.

Currently, three standardized MPEG-7 profiles are available [31]:

- Simple Metadata Profile: Used to describe simple metadata tagging of audio, image and video clips.
- User Description Profile: Used to describe multimedia content, user preferences and usage patterns to support personalized multimedia storage.
- Core Description Profile: Used to describe general multimedia content for creation, consumption, distribution and archiving.

In addition to these standards, there are also two commonly used non-standard profiles, namely TRECVID Profile and Detailed Audio Visual Profile (DAVP) [32][33]. The former is used in studies that participate in TRECVID Video Retrieval Evaluation[2]. The latter, on the other hand, is proposed for describing single audiovisual content entities so its main description type is ContentEntityType. However, DAVP also allows utilization of Semantic Description tools (Section 2.6.3) beside some other tools such as Classification Description tools and Summary Description Tools. The availability of these tools makes DAVP a useful profile for this study and accordingly, it is adopted in this thesis work rather than creating another profile from scratch for the same domain. The available semantic tools will be extended with fuzzy semantic types, such as FuzzyObjectType and FuzzyEventType, as it will be explained in the following chapters.

2.6.5 MPEG-7 Reference Software

MPEG-7 Reference Software, commonly known as Experimentation Model (XM) Software [34], is a software that is capable of extracting visual and audio low-level features and presenting the results in standard MPEG-7 description documents. Another capability of XM

[2] http://trecvid.nist.gov

Software is comparing these low-level features and calculating a distance based on their dissimilarities.

FMDBMS makes use of XM Software while extracting the low-level features of images (color layout, dominant color, scalable color, edge histogram, homogeneous texture, region shape and contour shape features) to store the results in the data storage. In addition, it is also employed to compare the features in the storage with the features of a query image while performing query-by-example task, which is described in more detail in Chapter 5.

2.7 eXist XML DB

In this thesis study, eXist-db v.1.4.1 is used for native XML storage environment. eXist-db is an open source, native XML database system [35]. The main features of eXist can be listed as:

- Data Storage: XML data storage is built on B+ trees and document nodes are stored in a persistent DOM.
- Collections: Document management is handled in hierarchical collections, similar to the folder structures of file management systems.
- Indexing: Indexing is based on numeric indexing scheme supporting quick identification of structural relationships, such as parent-child, ancestor-descendant or previous-next sibling. Structural index is used for element and attribute nodes. A fulltext index is used for text and attribute values, and range indexing is applied for typed values.
- Query Engine: An optimized, index-based XQuery processor is included in the system. Document and node-level updates are supported with XUpdate and XQuery update extensions.
- Authorization: Authorization of users is arranged using a unix-like access permission scheme, for users/groups at collection and document levels.
- Deployment: eXist can be deployed as a stand-alone database server or as a Java module embedded to an application.

- Validation: In eXist, XML document validation is supported in both implicit and explicit manners. Implicit validation is executed automatically while inserting new documents to the database and explicit validation can be performed through use of related XQuery functions provided by the system.

In addition to the features listed above, eXist also supports XML standards such as XInclude, XPointer, XSL/XSLT. DOM, Fluent and SAX APIs are provided. The maximum number of files that can be stored in database is 2^{31}.

2.8 VLC Player and vlcj

VLC (Video LAN Client) Player is a free, open-source and cross-platform multimedia player which is able to play almost all kinds of multimedia files [36]. vlcj (Video LAN Client for Java) is an open source project, which supplies wrapper interfaces to embed VLC Player in Java applications, playing videos in Java AWT Window or Swing JFrame [37]. In this thesis study, VLC Player v.1.1.1 and vlcj v.1.1.5.1 are used in client application to allow the user to play the related multimedia content.

2.9 Java Architecture for XML Binding

JAXB (Java Architecture for XML Binding) is an open-source XML data binding tool, that can compile XML Schema documents and create appropriate Java classes for the types declared in the schema [38]. After creating the set of Java classes, a valid XML document, which is consistent with the specific XML schema, can be parsed into an object and managed in the code. The edited content can be exported to a new XML document. In this thesis, JAXB v.2.2.5 is used in the XML Document Manager software, which will be explained in more detail in Chapter 6.

CHAPTER 3

RELATED WORK

3.1 MPEG-7 Description Databases

In order to efficiently manage the MPEG-7 descriptions, a number of studies have been proposed in recent years. Some of those studies are summarized in the following subsections.

3.1.1 MPEG-7 MMDB

Döller and Kosch proposed an MPEG-7 multimedia database system, MPEG-7 MMDB, in 2008 [13]. The novelty of this study is that MPEG-7 is considered in a DBMS data model for the first time.

As stated earlier, MPEG-7 standard relies on XML schema. For this reason, their approach needs to deal with constructs specific to XML schema. It is stated that although some of these constructs such as complex types, inheritance and collections have proper counterparts in object-relational database management systems (ORDBMS), other constructs like recursion, sequence, choice, etc. have to be investigated further.

MPEG-7 MMDB uses a supplemental table column in order to represent polymorphism. This column contains the name of the target descriptor and the corresponding PART_ID. Collections are represented using nested tables.

MPEG-7 MMDB supports the following types of queries:

- Point Queries: This query type is for finding exact matches. For example, *"Select all Objects where Height = 5 units"*.

- Range Queries: This query type is for finding all the instances with the related metric satisfying a prescribed range. For example, *"Select all Objects where Height >5 units and Height <10 units."*

- NN (Nearest Neighbor) Queries: This query type is for finding the most similar instance to the queried attributes. Say, an object with 5 units of height does not exist in the database, however there is an object with 4.99 units of height. So, the query *"Select all Objects where Height = 5"* would return that particular object with height 4.99, provided that no other object exists in the database with the height in the range (4.99, 5.01). Euclidean distance method is used to calculate the distances.

- kNN (k-Nearest Neighbor) Queries: kNN queries are quite similar to the NN queries. The difference is that NN queries return exactly one result, which is the closest object with respect to the queried attributes, whereas kNN queries return a set of closest instances with cardinality k. Similar to the NN queries, the distances are Euclidean distances.

GiST (Generalized Search Trees) methodology [39] is employed as the indexing backend. Oracle 10g XML DB is used as the database backend, which can be extended by implementations using PL/SQL, Java and external C language routines.

3.1.2 SM3+

Chu *et.al.* proposed SM3+ as an XML database solution for MPEG-7 descriptions [40]. SM3+ uses IBM DB2 as the underlying database management system.

Storing an XML document, which has a tree structure, in a relational database system requires the XML file to be captured in the relational data model. SM3+ makes use of a combination of model-mapping and structure-mapping strategies to map the XML nodes.

Model-mapping is employed in order to map all internal nodes of XML document. Internal nodes shows the structure of the XML document and they are useful for document navigation. SM3+ stores internal nodes using a model-mapping approach and makes use of them for efficient and easy document traverse. Encoding of model-mapped internal nodes is performed using Dewey Order method, which determines the ancestor-descendant relationship using

only an *id* value.

Evaluated nodes, which are the leaf nodes of the XML tree, hold the data. In SM3+, evaluated nodes are stored using structure-mapping, thus representing appropriate datatypes for each of them and providing a flexible storage schema in order to supply solutions for different storage requests. SM3+ makes use of a mapping processing file, which is an XML file, in order to identify the datatype of each evaluated node and map them into database.

SM3+ provides the opportunity of extracting the original XML document from the relational database structure. Although the XML data is stored in relational data tables, XML tree structure is visible to users and other applications. So the high-level XPath-based queries are converted to proper SQL queries in SM3+.

As a conclusion, SM3+ overcomes some of the most critical issues for MPEG-7 descriptions management such as index system, fine-grained and typed representation, access of data, and querying the content.

3.1.3 PTDOM

PTDOM, which stands for Persistent Typed Document Object Model, is a schema-aware XML database solution for MPEG-7 descriptions [14]. Its motivation is to make extensive use of XML schema definitions.

PTDOM utilizes XML schema definitions for the following purposes:

- Document validation: Most common usage of XML schema definitions is for document validation. Document validation is important since it ensures the consistency of the storage content. Consistency is provided by preventing insertion of inconsistent XML documents to the database and preventing one to update the XML documents that are already stored in the database to an inconsistent state.
- Document typing: As stated earlier, XML is a text-based document format. Thus all the data stored is encoded as text even though it is not text naturally, e.g. XML stores number 1234 as a character array (‘1’,‘2’,‘3’,‘4’). This is good for platform-independent data transportation, however from the point of view of storage, this could be quite time and space consuming. So typed representation of XML content means that the data

will be stored in appropriate data structures and the supplied set of type-specific operators will be available to use. Since the type information is contained in XML schema definitions, using these definitions would be helpful to deal with this issue.

- Query optimization: Query optimization is an important part of database systems since it provides an abstraction level between the developer and the database content. Since XML schema definitions depict the structure and content of XML documents, they could be of great use for optimizing the queries.

- Path indexing: Applications generally process selected parts of whole XML document. Because of this reason, path indexing is an important issue for speeding up the query executions. PTDOM makes use of XML schema definitions as path indices.

To sum up, PTDOM is an XML storage system with the capability of extensive schema utilization, fine-grained document management, typed document management, rich indexing capabilities (e.g. Hash-tables, B-Trees, R-Trees), efficient querying mechanisms, efficient updates, extensibility and classic database functionalities.

3.1.4 BilVideo-7

BilVideo-7 is not only an MPEG-7 description storage solution, but a comprehensive MPEG-7-compatible multimedia (video) indexing and retrieval framework [41].

BilVideo-7 extracts features, annotates the video and saves the output MPEG-7 description in a native XML database, namely Tamino. A native XML database is preferred because mappings and conversions to other data models are not required and it is easy to set up the database using the MPEG-7 schema.

BilVideo-7 includes a visual query interface, which supports several query types and supplies a large set of query options. The query types are textual queries, motion-queries, composite-queries and XQuery. Query processing is handled in a query processing server and queries are executed in a multithreaded fashion.

3.2 Fuzziness in XML Storage and Querying

There are several studies in the literature on using fuzziness with XML format. These studies can be divided into two main groups: Storing fuzzy information in XML and querying crisp fuzzy data using a fuzzy approach.

In [15], a fuzzy meta-knowledge and data storage base is implemented. Fuzzy meta-knowledge base is responsible for the storage of the information related to imprecise representation of data. Three sorts of fuzziness can be stored in the system:

- Linguistic Label with Possibility Distribution: These are represented in fuzzy meta-knowledge base as trapezoidal membership functions.
- Linguistic Label with Possibility Interval: These are represented by a function that calculates to 1 in the range [m, n] and 0 elsewhere. When m = n, this function can be used to represent crisp values as well.
- Approximate Values: These are represented in fuzzy-meta knowledge base as triangular membership functions.

The system has a query language that extends XQuery by usage of fuzzy conditions and uses Oracle Berkeley XML DB, a native XML database, for XML storage.

In [17], XML documents are represented as graphs and queries are represented as subgraphs. As stated earlier, XML content represents a tree structure. The system works in three steps:

- Weighting: In order to successfully extend the XML tree to form a graph, the XML tree is weighted according to the importance of the elements. This weighting is done in two ways. First one is the tag-related weighting, which is done to label the nodes of the XML document with fuzzy weights. Tag-related weighting can be estimated manually. At the end of this step, each tag is weighted in range [0, 1]. Second one is structure-related weighting, which weights the arcs of the tree. These weights can be evaluated as the normalized distances from the root. Then, these two weighting schemes can be combined using an arithmetic average strategy. Finally we have an XML tree with weighted arcs.

- Extending: Using the weighted tree constructed in the previous step, the fuzzy closure of the tree is calculated to form a graph. The non-existent arcs in the XML tree is assumed to be existent, and weighted with the arithmetic average of the arcs that form the shortest path between two elements. For example, say, element <root>, has two child elements as siblings <sibling1> and <sibling2>. The non-existent arc between <sibling1> and <sibling2> is calculated with the arithmetic average of the weights of the arcs (sibling1, root) and (root, sibling2). By the end of this step, a fuzzy closure of the original XML tree is constructed with assigned weights on each arc.

- Querying: Querying is done via a similarity matching between the graph constructed in the previous step and the query subgraph. First, the XML graph is filtered with an α-cut operation, removing the arcs with weights smaller than a given α value. Then the similarity matching is performed. For example, say the query is "Select all cars whose maker is in Stuttgart". Thus, the query has the path <car><maker><address>. So the result of the query will be assigned a truth value that is the arithmetic average of the arcs (car, maker) and (maker, address) available in the α-cut XML graph.

In [16], a fuzzy XML representation is implemented based on Extended Non-First Normal Form [42]. Researchers developed a system that maps Extended Non-First Normal Form to XML and an XML database system that stores and queries the XML content. In Extended Non-First Normal Form, uncertainty is represented in seven different types:

- Atomic-Valued Attributes: These attributes represent crisp values. The domain D_j includes only crisp values. For example, the title of a book may be represented using atomic valued attributes.

- Null-Valued Attributes: The domain of this attributes simply extends the domain of atomic valued attributes, D_j by adding *Unknown(unk)*, *Does Not Exist(dne)* and *No Information(ni)* values. For example, the image description of the book cover may be represented by these values, since there may be no image on the book cover.

- Incomplete (range)-Valued Attributes: These attributes may have a range of values. The value of the attribute is represented by a range, namely $[a_i, a_j]$ where $a_i, a_j \in D_j$. For example, the price range of a book in the market may be represented using these attributes.

- Fuzzy-Valued Attributes: These attributes have a domain D_f which contains fuzzy linguistic terms. For example, the age group of people may be represented by these attributes, such as *Young*, *Old*, etc.

- Set-Valued Attributes: These values are used to represent a set of crisp values altogether, $\{a_1, a_2, a_3, ..., a_n\}$where a_1 ... $a_n \in D_j$. For example, these values may be used to represent the authors of a book.

- Relation-Valued Attributes: These attributes represent tuples of values from several domains, $a_j = <a_{j1}, a_{j2}, a_{j3}, ..., a_{jn}>$where $a_j \in (D_{j1} \times D_{j2} \times D_{j3} \times ... \times D_{jn})$. For example, these values may be used to represent the cover of the book in terms of colour, image, description, etc.

- Combinations: The uncertainty types above may also be used as combined, such as *a set of relation-valued attributes* or *relation of fuzzy-valued attributes*, etc.

The system extends the notion of XQuery with the usage of fuzzy query conditions. Querying back-end makes use of similarity matrices between fuzzy linguistic terms. Conjunction($\wedge$) and disjunction($\vee$) operators are also defined. The query results may be filtered with a given threshold value.

In [19], researchers treat imperfect information as either inconsistent (e.g. age of a person is stored as 34 and 37 at the same time), imprecise (e.g. age of a person is available as a range of values {18, 19, 20, 21}), vague (e.g. age of a person is stored as OLD) or uncertain (e.g. age of a person is stored to be 37 with a possibility of 95%) data. In order to represent fuzzy data in XML, a possibility distribution scheme is introduced and the classical XML Document Type Definition (DTD) is extended to handle this scheme. The membership degree of an element is calculated considering the membership of that specific element and its parent element.

Researchers also focus on mapping UML and Relational Models to Fuzzy XML [18]. In order to solve this issue, three types of fuzziness in UML classes are listed:

- First Level: In this level, classes and attribute sets of a class may have a possibility degree in the model. These are represented in UML model with the phrase "WITH μ DEGREE" typed near the class or element name.

- Second Level: In this level, fuzzy occurrences of objects are handled. In order to deal with this issue, a special member, namely μ is added to each class, denoting the possibility that an object belongs to its specific type.
- Third Level: In this level, attributes and elements may have fuzzy linguistic values. These are represented by the "FUZZY" keyword added to before the element name.

The relations of these classes are also defined, namely *Generalization*, *Aggregation*, *Association*, and *Dependency*. These fuzzy classes and their relations are mapped to XML domain, extending the classical DTD to represent all of them in XML format.

In [12], a methodology for incorporating fuzzy and imprecise data in XML is investigated. It is stated that, two levels of work have been conducted in the relational database area to include fuzziness. First one is to query the precise information in a fuzzy and flexible sense, and the second one is to represent fuzzy data in the storage. In this study, the latter is taken into consideration. In particular, three interpretations of fuzzy and imprecise data are considered in database level:

- Relations: Classical relations are extended with a fuzzy relation, which is a subset of the Cartesian product of several domains. For example, consider the relation HATES, and the tuple (Ali, Ayşe) with a degree of membership 0.7. This means that the tuple belongs to the concept of HATES with a degree of 0.7, or there is a 0.7 certainty that Ali HATES Ayşe.
- Values: In this level, fuzziness is introduced as the attribute values. For example, the tuple (name='Ali', age='About 30', salary='LOW') includes fuzziness in age and salary attributes.
- Similarities: In this interpretation, the interchangeability of fuzzy values is considered. This issue is handled with the help of similarity relations. For example, say Age = {u $\in$ U where U = {YOUNG, MID-AGED, OLD}}. A similarity relation defined over class Age may present (YOUNG, MID-AGED) with 0.3, (YOUNG, OLD) with 0.0, and (MID-AGED, OLD) with 0.4 similarity values.

In XML, fuzziness may include fuzzy data in various parts. Fuzzy notions can be incorporated into the value of the element for simple elements, or into the attribute values, occurrence of

a sub-element within an element or the combinations of these for complex elements. In the research, the fuzziness in relational data is mapped to XML.

In [43], an XML data warehouse is developed using the semantically tagged forms of XML documents extracted from the WEB. The XML content is semantically tagged, that is all the possible values of an attribute in the ontology are expressed in terms of possibility distributions, each term weighted with a possibility degree. The querying is also conducted in a fuzzy manner, so that the user can express fuzzy query constraints. MIEL++ query processor is used in the system. It relies on the pre-defined queries called views, to help user define their own queries. The results of queries are expressed as fuzzy sets, i.e. each result is extracted with a degree of membership, which allows the user to see all possible results in addition to the exactly-matched ones.

CHAPTER 4

FUZZY EXTENSION OF MPEG-7 SEMANTIC TOOLS

4.1 Fuzzy Semantic Extension

In order to create a fuzzy MPEG-7 document storage system, the first step to be taken is extending the MPEG-7 profile by adding new fuzzy semantic types. Since DAVP-2004 MPEG-7 profile will be used in (FM)DBMS, the fuzzy semantic types are built on this profile, without corrupting the already existing semantic types. These new fuzzy semantic types can be applied to other profiles, provided that the profile is created on the same revision of MPEG-7 schema definitions, i.e. 2004, and it includes the semantic description tools.

The fuzzy extension comprises two main tasks:

- Declaring New Types: New types should be declared to represent fuzziness in semantic objects.
- Extending Semantic Description Types: The already defined semantic description tools should be extended to include fuzziness using the new fuzzy types defined in the previous item.

4.1.1 Declaration of New Types

Table 4.1 summarizes the new types declared for handling fuzziness in MPEG-7 schema. Detailed descriptions of these new types can be found in this section.

- FuzzyLinguisticTermType: As the name implies, this type is used to define a fuzzy lin-

Table 4.1: New types declared to handle fuzziness in fuzzy MPEG-7 documents.

TYPE NAME	EXTENDS
FuzzyLinguisticTermType	string
FuzzySimilarityType	zeroToOneType
FuzzySimilarityRelationType	none
FuzzyTriangularMembershipFunctionType	none
FuzzyTrapezoidalMembershipFunctionType	none
FuzzyGaussianMembershipFunctionType	none
FuzzyMembershipFunctionCollectionType	none
FuzzyVariableType	none
FuzzyValueRefType	none
FuzzyTextualDistributionType	none
FuzzyAgentDistributionType	none

guistic term such as SLOW, FAST, etc. It has a required fuzzyValueID attribute, which identifies an instance of the linguistic term and should be unique for each instance. This type can only be used in an instance of FuzzyVariableType type.

- FuzzySimilarityType: The similarities between two linguistic terms can be defined using this type. It has two required attributes, namely fuzzyVal1REF and fuzzyVal2REF, which are of type IDREF and should take the fuzzyValueID of a FuzzyLinguisticTermType instance. The similarity value should be a floating point value in the range $[0, 1]$. Similarity values are application specific, thus they can be defined arbitrarily within the allowed range. This type can only be used in an instance of FuzzySimilarityRelationType type.

- FuzzySimilarityRelationType: This type includes one or more instances of FuzzySimilarityType so that it can define the similarities between all linguistic values belonging to that specific FuzzyVariableType. This type can only be used in an instance of FuzzyVariableType type.

- FuzzyTriangularMembershipFunctionType: This type can be used to define the membership function of a crisp value to the fuzzy set defined by a linguistic term in triangular form. The instances of this type have four required attributes, namely a, b, c and fuzzyValREF. The first three attributes, a, b and c, define the X coordinates of the corners of a triangle from left to right. An example can be seen in Figure 4.1. The fuzzyValREF attribute is of type IDREF and should take the fuzzyValueID of

a FuzzyLinguisticTermType instance. This type can only be used in an instance of FuzzyMembershipFunctionCollectionType type.

- FuzzyTrapezoidalMembershipFunctionType: This type can be used to define the membership function of a crisp value to the fuzzy set defined by a linguistic term in trapezoidal form. The instances of this type have five required attributes, namely a, b, c, d and fuzzyValREF. The first four attributes, a, b, c and d, define the X coordinates of the corners of a trapezoid from left to right. An example can be seen in Figure 4.1. The fuzzyValREF attribute is of type IDREF and should take the fuzzyValueID of a FuzzyLinguisticTermType instance. This type can only be used in an instance of FuzzyMembershipFunctionCollectionType type.

- FuzzyGaussianMembershipFunctionType: This type can be used to define the membership function of a crisp value to the fuzzy set defined by a linguistic term in Gaussian function form. The instances of this type have three required attributes, namely mu (μ), stdDeviation (σ) and fuzzyValREF. The first two attributes, mu and stdDeviation, define the mean and standard deviation of the Gaussian function respectively. An example can be seen in Figure 4.1. The fuzzyValREF attribute is of type IDREF and should take the fuzzyValueID of a FuzzyLinguisticTermType instance. This type can only be used in an instance of FuzzyMembershipFunctionCollectionType type.

- FuzzyMembershipFunctionCollectionType: This type can be used to contain the membership functions of all instances of FuzzyLinguisticTermType type, belonging to an instance of FuzzyVariableType type.

- FuzzyVariableType: This type includes instances of FuzzyLinguisticType, FuzzySimilarityRelationType and FuzzyMembershipFunctionCollectionType types. It has a required fuzzyVarID attribute, which identifies an instance of the fuzzy variable, and should be unique for each instance of this type. Figure 4.2 shows a sample FuzzyVariableType instance.

- FuzzyValueRefType: This type is used to define a fuzzy property in an instance of FuzzyEventType, FuzzyObjectType and etc. There are two required attributes to relate the instances of this type to the instances of FuzzyVariableType and FuzzyLinguisticTermType types: fuzzyVarREF and fuzzyValREF, both of which are of type IDREF.

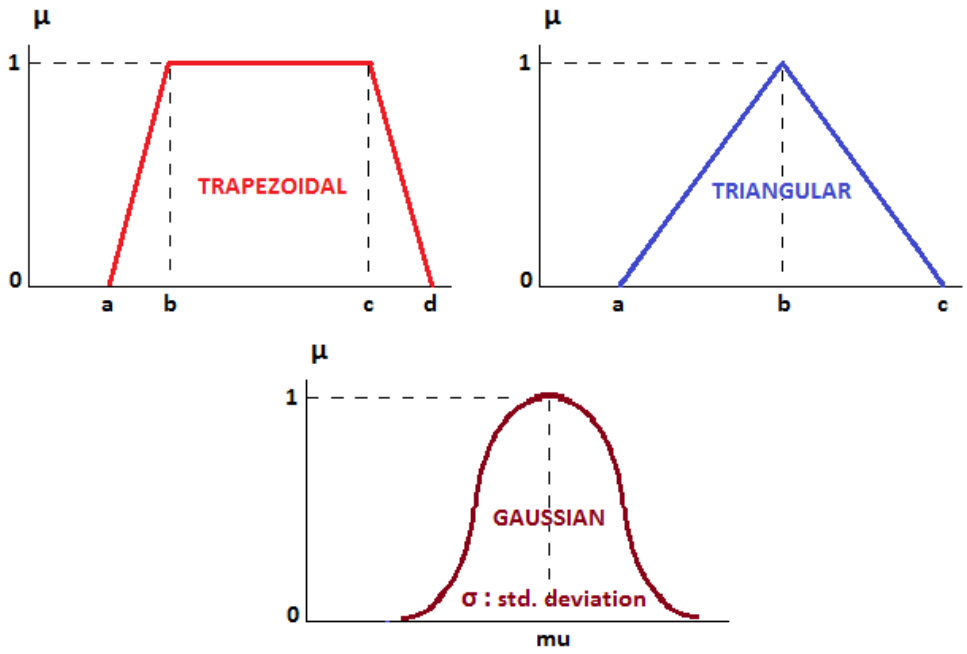


Figure 4.1: Examples of membership function types.

The former should take the fuzzyVarID of an instance of FuzzyVariableType and the latter should take the fuzzyValID of an instance of FuzzyLinguisticTermType. There is also a name attribute, which is optional, for clarity purposes. For example, using an instance of this type, the user can define a SHOT event to be SLOW SHOT, relating the value SLOW to the proper instances of FuzzyLinguisticTermType and FuzzyVariableType through their IDs.

- FuzzyTextualDistributionType: This type includes an array of XML elements named *Val*, which can include arbitrary textual data. These elements have a required attribute called *poss*, which is a floating point value in the range [0, 1] identifying the possibility of the element. This type is the implementation of *possibility distributions* [19] in XML documents for textual data content.

- FuzzyAgentDistributionType: This type includes an array of XML elements named *Val*, which can include an element of type Agent or AgentRef, defined in MPEG-7 standard. Similar to the FuzzyTextualDistributionType, these *Val* elements have a required attribute called *poss*, which is a floating point value in the range [0, 1] identifying the possibility of the element. This type is the implementation of *possibility distributions*

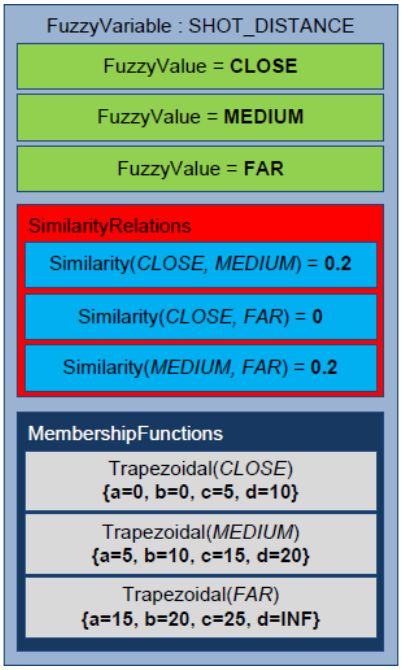


Figure 4.2: An example of FuzzyVariableType instance.

[19] in XML documents for Agent or AgentRef data content.

4.1.2 Extension of Semantic Description Types

Extension of already existing MPEG-7 semantic types is done in two manners. In the first group, the types are slightly modified versions of already existing types in order to handle fuzziness in their content, and they do not extend an existing type. In the second group, the modifications are done in terms of *extension base* property of XSD format. In Table 4.2, the column named EXTENDS shows the name of the existing type from which the new type is actually extended (using "extension base" of XSD). The column named DERIVED FROM shows the name of the existing type that is slightly modified to create the new type for handling fuzziness.

Detailed descriptions of the newly introduced types in Table 4.2 follow next.

- FuzzyTextualType: This type is used for replacing a textual value in MPEG-7 semantic types with fuzzy values. In this extension work, fuzziness is introduced in two

Table 4.2: Extended types declared to handle fuzziness in fuzzy MPEG-7 documents.

TYPE NAME	EXTENDS	DERIVED FROM
FuzzyTextualType	none	TextualType
FuzzyStructuredAnnotationType	none	StructuredAnnotationType
FuzzyKeywordAnnotationType	none	KeywordAnnotationType
FuzzyTextAnnotationType	none	TextAnnotationType
FuzzyInlineTermDefinitionType	none	InlineTermDefinitionType
FuzzyTermUseType	FuzzyInlineTermDefinitionType	TermUseType
FuzzySemanticBaseType	DSType	SemanticBaseType
FuzzyObjectType	FuzzySemanticBaseType	ObjectType
FuzzyAgentObjectType	FuzzyObjectType	AgentObjectType
FuzzyEventType	FuzzySemanticBaseType	EventType
FuzzySemanticBagType	FuzzySemanticBaseType	SemanticBagType
FuzzySemanticType	FuzzySemanticBagType	SemanticType
FuzzySemanticDescriptionType	ContentAbstractionType	SemanticDescriptionType

manners, i.e. possibility distributions or fuzzy linguistic terms. So this type may exclusively include an instance of FuzzyTextualDistributionType type or a set of instances of FuzzyValueRefType type.

- FuzzyStructuredAnnotationType: This type replaces the StructuredAnnotationType in MPEG-7 standard, introducing fuzziness for each sub-element. The type of each sub-element is changed from TermUseType to FuzzyTermUseType, which will be explained in detail later.

- FuzzyKeywordAnnotationType: This type can be used for introducing fuzziness in Keyword annotations of MPEG-7 standard semantic types. It includes an element called FuzzyKeyword of FuzzyTextual type. Fuzzy keyword annotation is currently used for identifying the events, for example, in football domain if an event is a SHOT, PASS, OFFSIDE, etc.

- FuzzyTextAnnotationType: This type replaces the instances of TextAnnotationType in MPEG-7 standard semantic types in order to deal with fuzziness. Any instance of FuzzyTextAnnotationType may include elements named FuzzyFreeTextAnnotation, FuzzyStructuredAnnotation and FuzzyKeywordAnnotation, which are of types FuzzyTextualType, FuzzyStructuredAnnotationType and FuzzyKeywordAnnotationType respectively.

- FuzzyInlineTermDefinitionType: This type replaces the instances of InlineTermDefinitionType in MPEG-7 standard semantic types in order to deal with fuzziness. Any

instance of FuzzyInlineTermDefinitionType may include elements named FuzzyName, FuzzyDefinition and FuzzyTerm, which are of type FuzzyTextualType, FuzzyTextualType and FuzzyInlineTermDefinitionType respectively. FuzzyInlineTermDefinitionType is a recursive type due to the inclusion of FuzzyTerm element (which is of type FuzzyInlineTermDefinitionType).

- FuzzyTermUseType: This type extends FuzzyInlineTermDefinitionType by the addition of an optional attribute, href, which is of type termReferenceType defined in MPEG-7 standard.

- FuzzySemanticBaseType: This type is the base type of all other fuzzy semantic types, replacing the type SemanticBaseType in regular MPEG-7 semantic description tools. It includes generic features of fuzzy semantic types, such as AbstractionLevel, FuzzyLabel (FuzzyTermUseType), FuzzyDefinition (FuzzyTextAnnotationType), FuzzyProperty (FuzzyTermUseType), MediaOccurence and Relations (RelationType).

- FuzzyObjectType: This type enhances the ObjectType of MPEG-7 standard with the ability of handling fuzziness. Instances of FuzzyObjectType type are used to identify objects in a multimedia file (for example, a COMPUTER object in an image). It extends FuzzySemanticBaseType, thus includes all of its properties and attributes. In addition to those, FuzzyObjectType may include elements named FuzzyObject, Object and ObjectRef which are of types FuzzyObjectType, ObjectType and ReferenceType respectively. These three sub-elements are used to define the sub-objects of an object (for example, CASE, MONITOR, KEYBOARD and MOUSE objects that form a COMPUTER object in an image). A sample FuzzyObjectType instance is given in Figure 4.3.

- FuzzyAgentObjectType: This type enhances the AgentObjectType of MPEG-7 standard with the ability of handling fuzziness. Instances of FuzzyAgentObjectType type are used to identify a person or a group of people in a multimedia file (for example, MESUT OZIL as a person or REAL MADRID as a group of people in a football match video). It extends FuzzyObjectType, thus includes all of its properties and attributes. In addition to those, FuzzyAgentObjectType may include elements named FuzzyAgent of type FuzzyAgentDistributionType, that is used to identify an agent as a possibility distribution. That means, an instance of FuzzyAgentObjectType may include more than

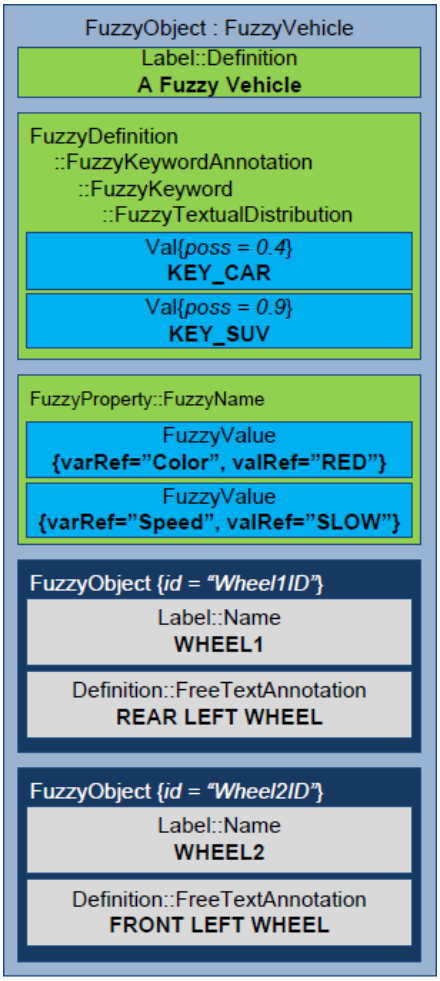


Figure 4.3: An example of FuzzyObjectType instance.

one agent definition, with varying possibility values. A sample FuzzyAgentObjectType instance is given in Figure 4.4.

- FuzzyEventType: This type enhances the EventType of MPEG-7 standard with the ability of handling fuzziness. Instances of FuzzyEventType type are used to annotate events in a multimedia file (for example, a GOAL event in a video). It extends FuzzyAgentObjectType, thus includes all of its properties and attributes. In addition to those, FuzzyEventType may include elements named FuzzyEvent, Event, EventRef, SemanticPlace and SemanticTime of types FuzzyEventType, Event, ReferenceType, SemanticPlaceType and SemanticTimeType types respectively. FuzzyEvent, Event, EventRef elements are used define the sub-events of an event (for example, PASS

Figure 4.4: An example of FuzzyAgentObjectType instance.

and SHOT events that occurs during the GOAL event). SemanticPlace and SemanticTime attributes, as their names imply, define the time and place of an event. A sample FuzzyEventType instance is given in Figure 4.5.

- FuzzySemanticBagType: This type enhances the SemanticBagType of MPEG-7 standard with the ability of handling fuzziness. It extends FuzzySemanticBaseType, thus includes all of its properties and attributes. In addition to those, FuzzySemanticBaseType may include elements named FuzzyVariable, FuzzySemanticBase, FuzzySemanticBaseRef, SemanticBase, SemanticBaseRef and Graph. FuzzyVariable element is an instance of FuzzyVariableType which is described previously. FuzzySemanticBase and SemanticBase elements are of types FuzzySemanticBaseType and SemanticBaseType respectively, which are used to define any semantic instance, i.e. Events, FuzzyEvents,

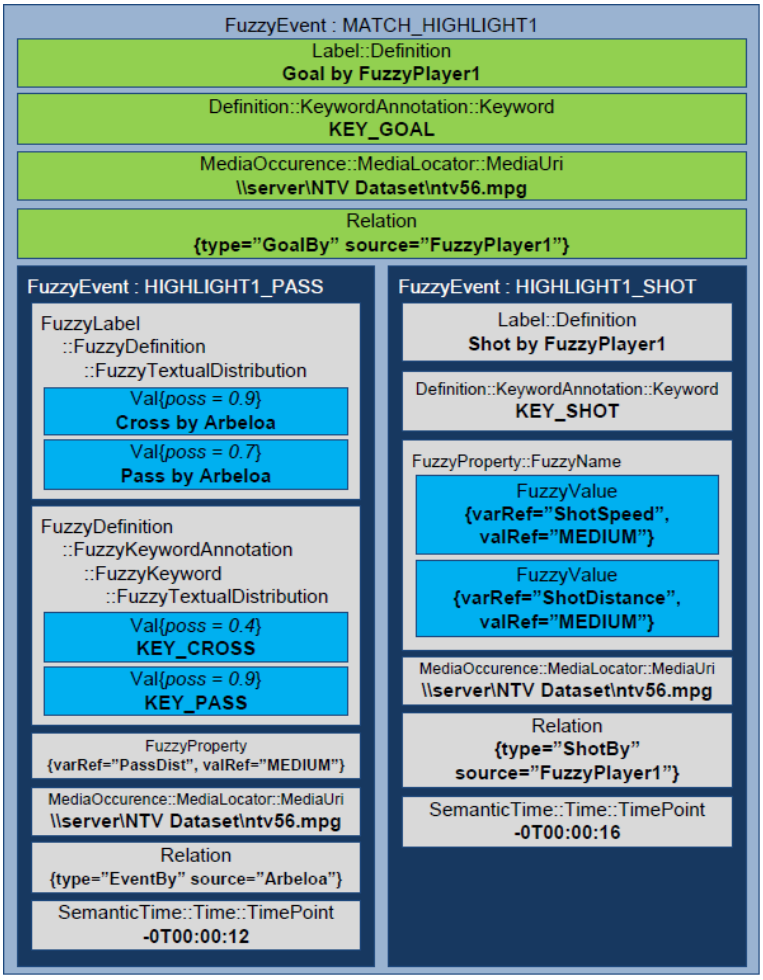


Figure 4.5: An example of FuzzyEventType instance.

Objects, FuzzyObjects, AgentObjects, FuzzyAgentObjects etc. FuzzySemanticBaseRef and SemanticBaseRef elements are of type ReferenceType and are used to refer other FuzzySemanticBaseType and SemanticBaseType instances through their IDs, instead of redefining them.

- FuzzySemanticType: This type enhances the SemanticType of MPEG-7 standard with the ability of handling fuzziness. It extends FuzzySemanticBagType and adds no other attributes or elements.

- FuzzySemanticDescriptionType: This type enhances the SemanticDescriptionType of MPEG-7 standard with the ability of handling fuzziness. It extends ContentAbstractionType and is used to cover all semantic descriptions. The instances of this type take

place just under the root of MPEG-7 description files.

After extending the MPEG-7 standard semantic description tools, the block diagram given in Figure 2.6 evolves and forms the diagram shown in Figure 4.6.

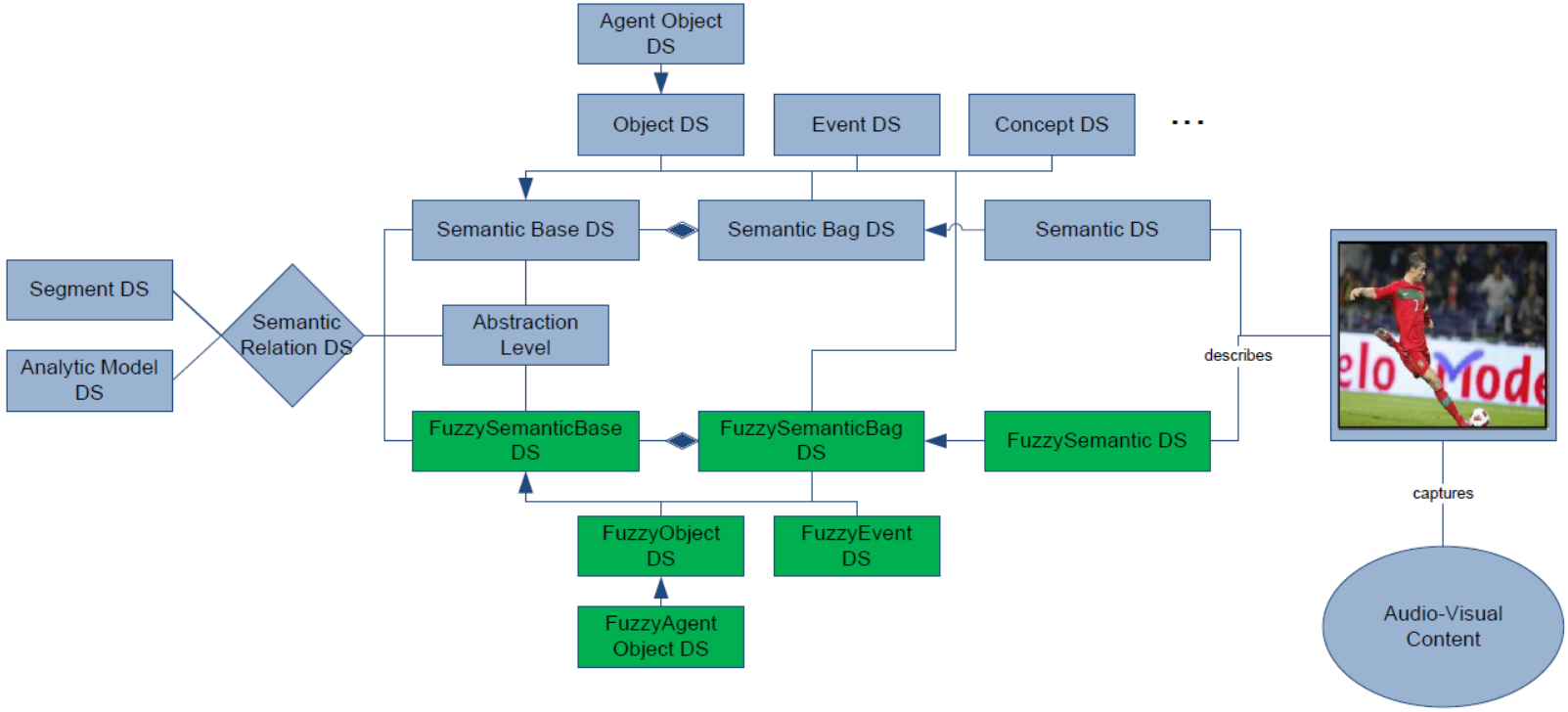


Figure 4.6: Extended MPEG-7 semantic description tools.

CHAPTER 5

FMDBMS IMPLEMENTATION

5.1 eXist Database Design

eXist-db, as stated in earlier chapters, is a native XML database system. Unlike the tables in relational database systems, content is formed as *collections*. Collections in eXist are similar to folders in a file system. A collection can contain files (in this thesis, files are XML files) or other collections. Using eXist-db client application, a collection tree of two levels is created in the database. The tree of collections can be seen in Figure 5.1.

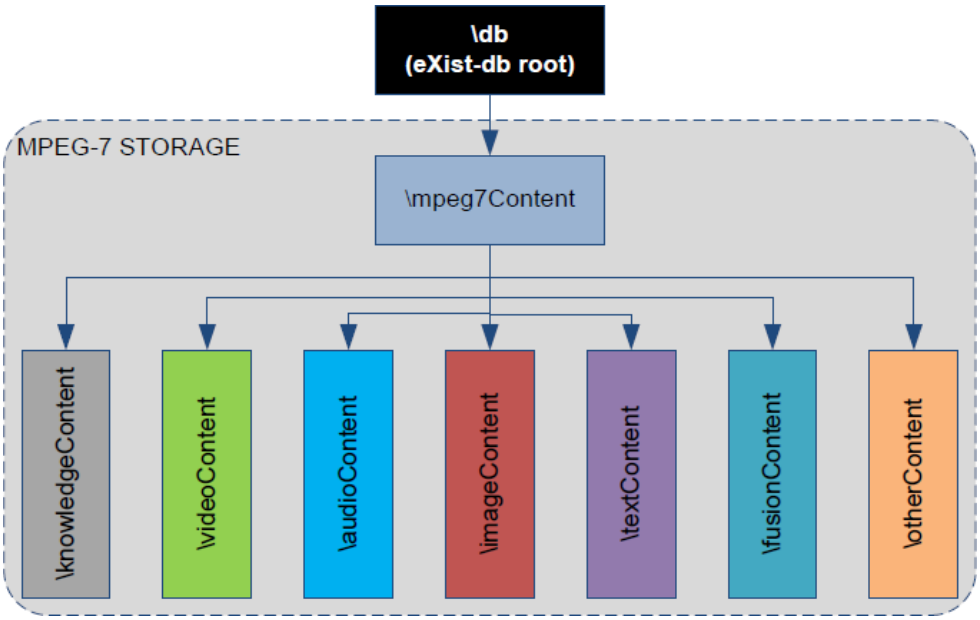


Figure 5.1: eXist-db design to store MPEG-7 description documents.

The collections shown in Figure 5.1 can be described as follows:

- \db: The default root collection of eXist-db. Each eXist-db installation contains this collection. All other collections or XML documents are stored under this collection.

- \mpeg7Content: The root collection of all MPEG-7 description documents and other collections pertaining to MPEG-7 documents. Although it is not forbidden to store an MPEG-7 description document in this collection, it is aimed to store only other collections. All MPEG-7 description documents will be stored in a proper sub-collection of this collection, which will be manually controlled by the database administrator. When a user wants to gather information related to any media type, i.e. video, audio, image, text and fusion, s/he should execute queries on this collection since it contains all other collections.

- \knowledgeContent: This is a sub-collection of \mpeg7Content described above. MPEG-7 description documents containing the definitions of fuzzy variables and values, along with their similarity relations and membership functions will be stored here. When a user wants to execute a query including a fuzzy predicate, the similarities or membership functions of those fuzzy terms will be searched in this collection.

- \videoContent: This is a sub-collection of \mpeg7Content described above. MPEG-7 description documents of video media will be stored here. When a user wants to gather video related information, s/he should execute queries on this collection.

- \audioContent: This is a sub-collection of \mpeg7Content described above. MPEG-7 description documents of audio media will be stored here. When a user wants to gather audio related information, s/he should execute queries on this collection.

- \imageContent: This is a sub-collection of \mpeg7Content described above. MPEG-7 description documents of image media will be stored here. When a user wants to gather image related information, s/he should execute queries on this collection.

- \textContent: This is a sub-collection of \mpeg7Content described above. MPEG-7 description documents of textual data will be stored here. When a user wants to gather text related information, s/he should execute queries on this collection.

- \fusionContent: This is a sub-collection of \mpeg7Content described above. MPEG-7 description documents extracted as a result of data fusion will be stored here. When a

user wants to gather information on fusion results, s/he should execute queries on this collection.

- \otherContent: This is a sub-collection of \mpeg7Content described above. This collection is created for general-purpose content, i.e. MPEG-7 documents that does not conform to any other collection above.

5.2 FMDBMS Client Application

In order to communicate with the eXist-db server, where the MPEG-7 semantic description documents are stored in, a client application is developed. The development is done with Eclipse IDE 3.7.1[1], using Java programming language with Java Development Kit (JDK) 1.7. The software is running on Java Runtime Environment 7, both in Windows 7 Home Edition and Windows XP SP3 operating systems. Since eXist-db installation and the third party libraries are available in both Windows & Unix environments, the client software is expected to be cross-platform, provided that all of its dependencies are installed correctly.

Developed FMDBMS client software has the following capabilities:

- Connecting to the database server: Using the *Connection* tab on the client GUI, user is able to connect to the eXist-db server, if s/he inputs the correct credentials.
- Insert/Delete MPEG-7 description files: Using the *Insert/Delete Files* tab on the client GUI, user is able to insert new files into any of the sub-collections described in Section 5.1. Using the same tab, user can browse and delete the existent XML files in the database.
- Write an XQuery and execute: Using the *Free XQuery* tab on the client GUI, user is able to execute XQuery strings that s/he types in the related text area. The results will be displayed textually, right below the query text area.
- Query objects: Using the *Query Objects* tab on the client GUI, user is able to query ObjectType and FuzzyObjectType instances, just by selecting the object properties s/he wants to query. When the query execution is started, query is automatically generated and the results are displayed as a list, including the ID, possibility degree and media

[1] http://eclipse.org/

path of each result. The XML forms of the results can be displayed as a tree. The multimedia document that the object is annotated in, can also be displayed.

- Query agent objects: Using the *Query Agent Objects* tab on the client GUI, user is able to query AgentObjectType and FuzzyAgentObjectType instances, just by typing in the agent properties s/he wants to query. When the query execution is started, query is automatically generated and the results are displayed as a list, including the ID and possibility degree of each result. The XML forms of the results can be displayed as a tree. The multimedia document that the agent object is annotated in, can also be displayed.

- Query events: Using the *Query Events* tab on the client GUI, user is able to query EventType and FuzzyEventType instances, just by typing in the event properties supplied on the same tab. When the query execution is started, query is automatically generated and the results are displayed as a list, including the ID, possibility, media time and media URI of each result. The XML forms of the results can displayed as a tree. If available, the media file in given in media URI can be played starting from the given media time.

- Query spatial relations: Using the *Spatial Query* tab on the client GUI, user is able to query the ObjectType and FuzzyObjectType instances which are spatially related to each other. User can choose the properties of source and target objects as well as their spatial relation type. When the query execution is started, the query is automatically generated and the results are displayed as a list, including the source and target object IDs, possibility degree and the multimedia file path. The multimedia files of each result can be displayed.

- Query temporal relations: Using the *Temporal Query* tab on the client GUI, user is able to query EventType and FuzzyEventType instances which are temporally related to each other. User can choose the properties of source and target events as well as their temporal relation type. When the query execution is started, the query is automatically generated and the results are displayed as a list, including the source and target event IDs, possibility degree and the multimedia file path. The multimedia files of each result can be displayed.

- Query by example: Using the *Query By Example* tab on the client GUI, user is able to perform querying by example on images. User can select an image file on the file

system and the low-level properties of the image that s/he wants to compare. When the query execution starts, the system compares the selected low-level properties of the query image with the low-level properties of the images that are previously extracted and the comparison results are displayed as a list. In the results list, each item is displayed by its multimedia file path and its similarity to the query image.

5.2.1 Connection Screen

A screenshot of the *Connection* screen is given in Figure 5.2.

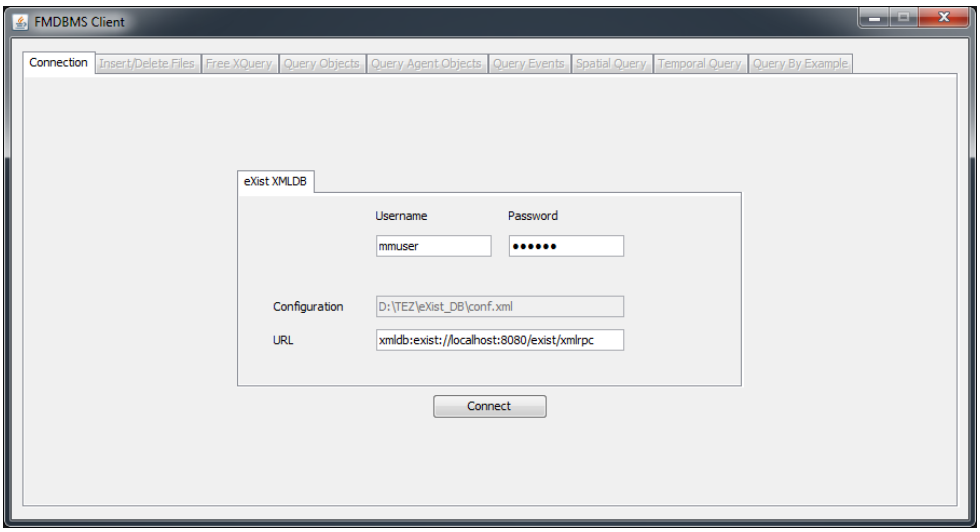


Figure 5.2: Screenshot of the connection tab.

The use of *Username* and *Password* text areas is straightforward, user types in his/her username and password. The text area labeled *Configuration* indicates the file path of the configuration file (conf.xml) for eXist-db. The *URL* text area is where the URL of the eXist-db server is typed in. It identifies the database instance to connect to.

After completing all the necessary text areas just explained above, the user clicks on the *Connect* button. At this point, it should be noted that if the client is not connected to the database server, all other tabs except *Connection* tab are inactive and cannot be selected or used. If the database is set properly, as shown in Figure 5.1, the connection will be established and all tabs will be activated. After the connection is set, the *Connect* button is replaced with

a *Disconnect* button, which will disconnect the client from the eXist-db server. When the client is disconnected from eXist-db server, the GUI will return to its initial state, at which all tabs except *Connection* tab are inactive and the *Connect* button is shown.

5.2.2 File Insertion and Deletion Screen

A screenshot of the *Insert/Delete Files* screen is given in Figure 5.3.

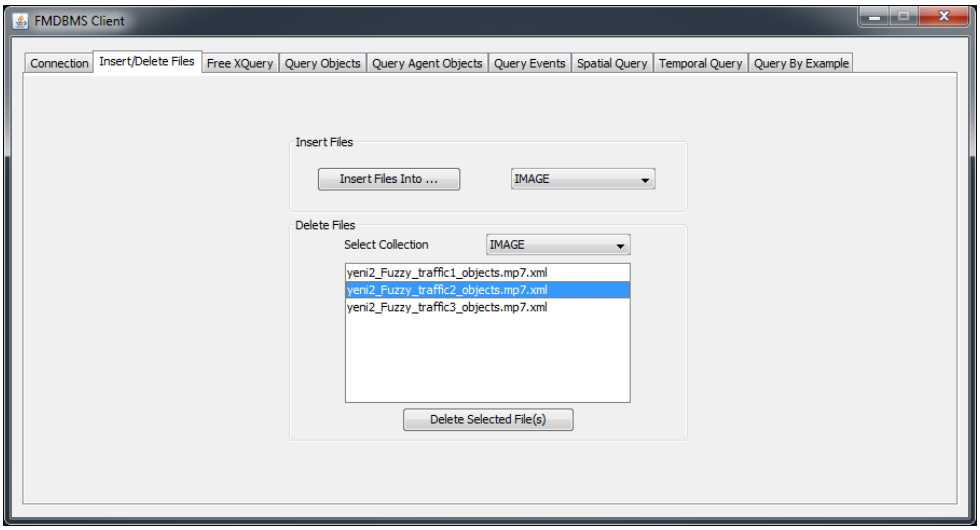


Figure 5.3: Screenshot of the insert/delete files tab.

In order to insert files into the database, first the user has to choose a collection to insert files, from the combobox in *Insert New Files* groupbox. This combobox contains the names of the collections which are the leaves of the tree in Figure 5.1. After selecting the collection, say videoContent, user presses the *Insert File Into* button and a file selection dialog is displayed which enables the user to select the files s/he wants to insert. After pressing the *OK* button in the file dialog, the selected files are inserted into the videoContent collection and a message is displayed reporting the completion of insertion.

Deleting files is also trivial. The user selects the collection, say videoContent, from the combobox in the *Delete Files* groupbox and the names of the files in the videoContent are listed. If user wants to delete file(s), s/he selects the names of the files in the list and presses the *Delete Selected File(s)* button located at the bottom of the *Delete Files* groupbox.

5.2.3 Free XQuery Screen

A screenshot of the *Free XQuery* screen is given in Figure 5.4.

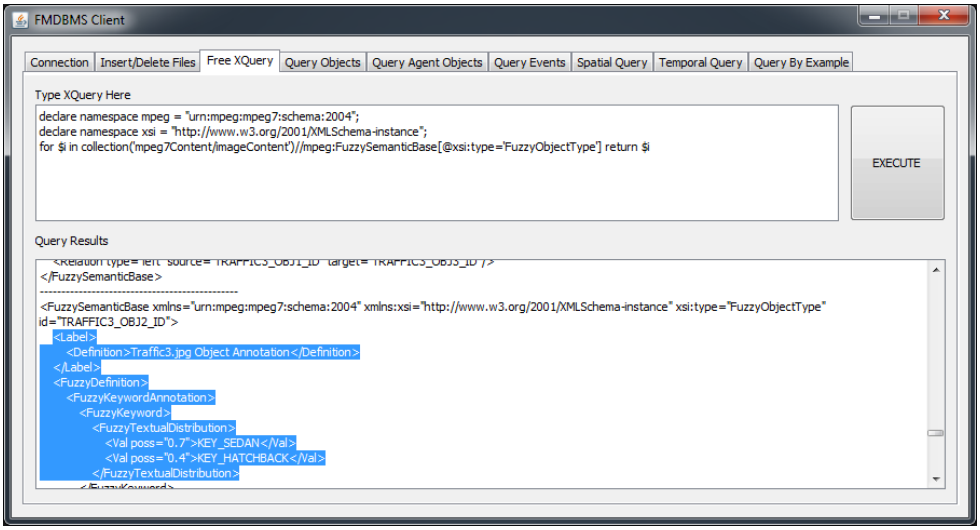


Figure 5.4: Screenshot of the free querying tab.

The usage of this tab is straightforward. The user types an XQuery in the text area labeled *Type XQuery Here* then presses the *Execute* button. Provided that the user types in a syntactically correct XQuery, the query is executed and results will be listed in the text area labeled *Query Result*, each result separated from the other with a string of '-' characters. If the result of the query does not contain any elements, the results area will be empty. If the query is syntactically incorrect or cannot be executed for some reason, an informative error message will be displayed.

5.2.4 Object Query Screen

A screenshot of the Object Query screen is given in Figure 5.5.

For querying objects and fuzzy objects, the client software supplies necessary GUI elements defining the properties of the objects to be queried in the database, and the XQuery string is automatically generated by the software. In order to query an object, the user selects object properties that s/he is interested in, such as *object type*, *color*, *size*, *width* and *length* of the

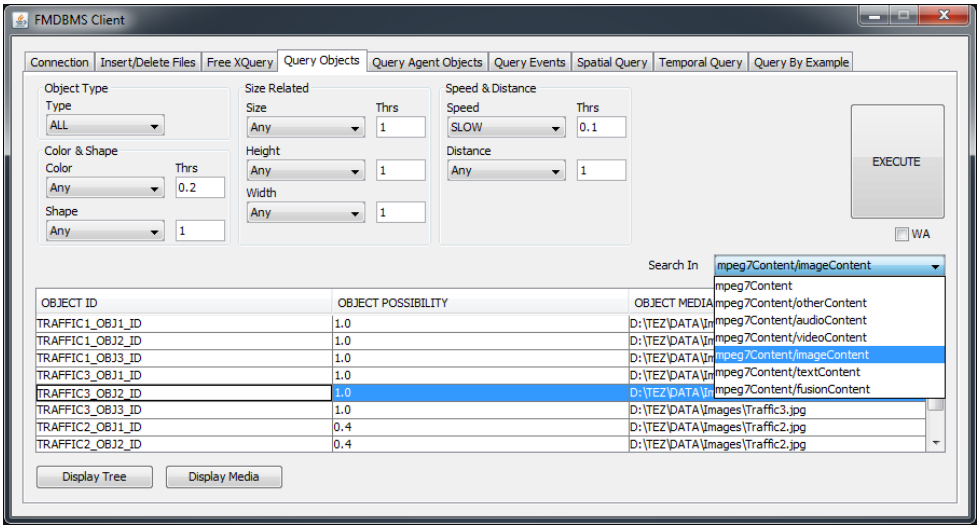


Figure 5.5: Screenshot of the object query tab.

object. If the initial option 'Any' is selected for a certain property, that property will not be included in the query as a predicate. For instance, if the user leaves all properties at their initial states, all of the objects in the database will be returned as the query result.

Using the default mode, the possibility value of each resulting object is calculated by multiplying the possibility or similarity value of each object property. Using the WA option, by checking the WA checkbox on the tab, the weight of each condition is multiplied with its possibility or similarity value and the sum of those values are displayed as the *WA Result* instead of object possibility.

The user can select a result from the results list and display it in tree view by clicking the *Display Tree* button located at the bottom of the tab. Object tree display can be seen in Figure 5.6.

The user may also view the multimedia content that the object is annotated in, by clicking the *Display Media* button at the bottom of the window. Object media display can be seen in Figure 5.7.

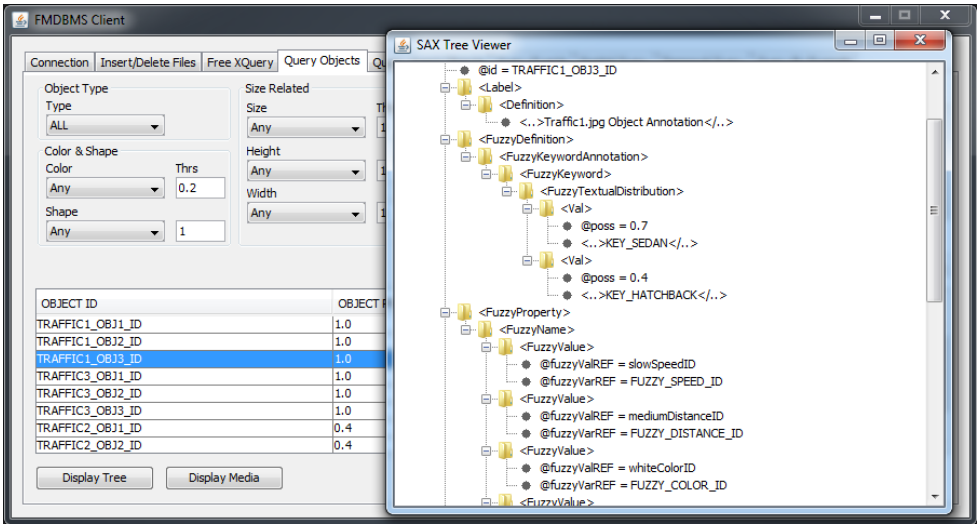


Figure 5.6: Screenshot of the object tree display.

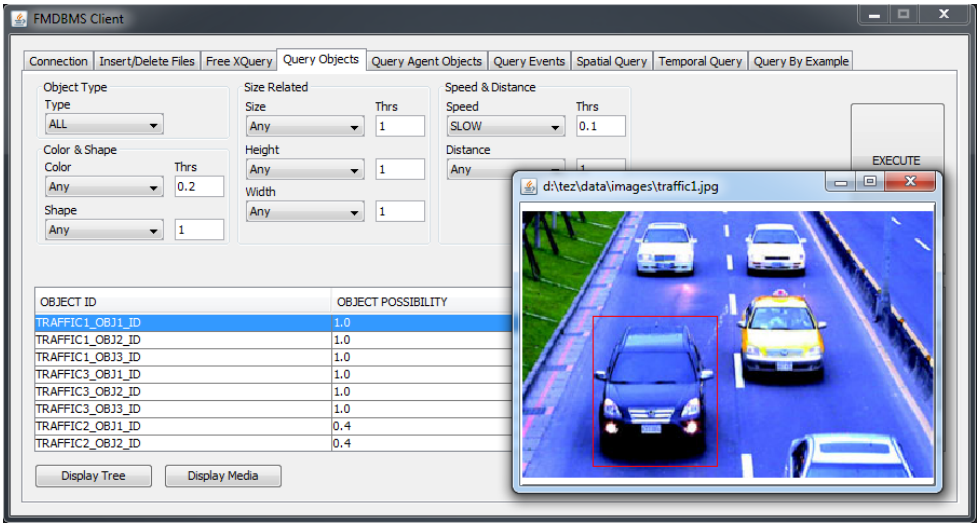


Figure 5.7: Screenshot of the object media display.

5.2.5 Agent Query Screen

A screenshot of the *Agent Query* screen is given in Figure 5.8.

For querying agents and fuzzy agents, the client software supplies necessary GUI elements defining the properties of the agents to be queried in the database, and the XQuery string is automatically generated by the software. In order to query an agent, the user types in *Given*

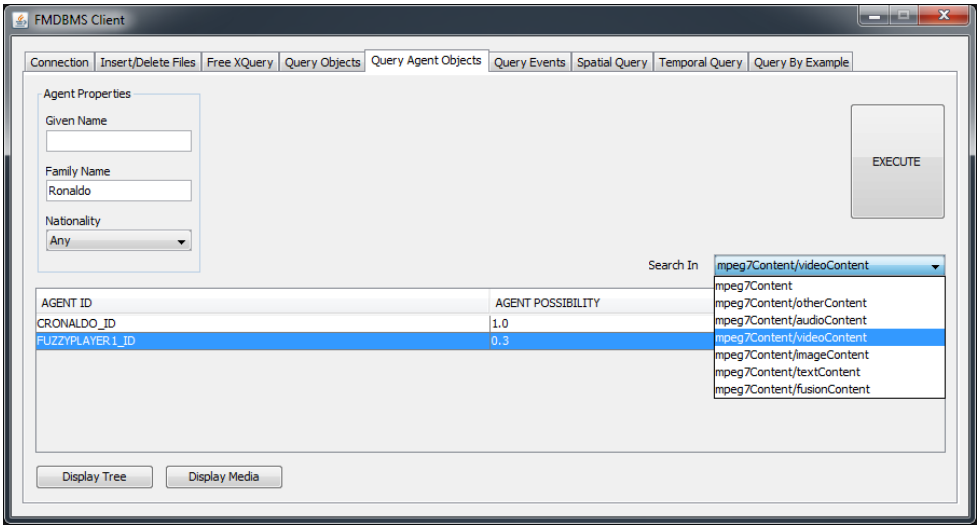


Figure 5.8: Screenshot of the agent query tab.

Name and *Family Name* text areas and selects the *Nationality*. All of the elements considered can be used altogether or separately. If a text area is left empty it is not included as a condition in the XQuery string. Similarly, if *ALL* is selected as the nationality, it is not included as a condition as well. For example, if user leaves all text areas and the combobox at their initial states and clicks the *Execute* button, all of the agents and fuzzy agents are displayed in the results list.

The user can select a result from the results list and display it in tree view by clicking the *Display Tree* button located at the bottom of the tab. A screenshot of agent tree display is show in Figure 5.9.

The user may also view the multimedia content that the agent object is annotated in, by clicking the *Display Media* button at the bottom of the window. Agent object media display can be seen in Figure 5.10.

5.2.6 Event Query Screen

A screenshot of the *Event Query* screen is given in Figure 5.11.

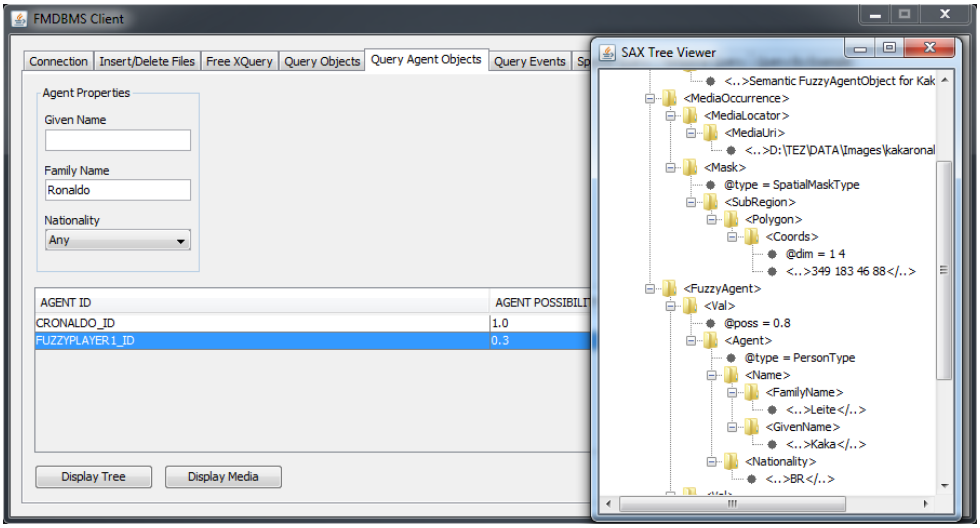


Figure 5.9: Screenshot of the agent tree display.

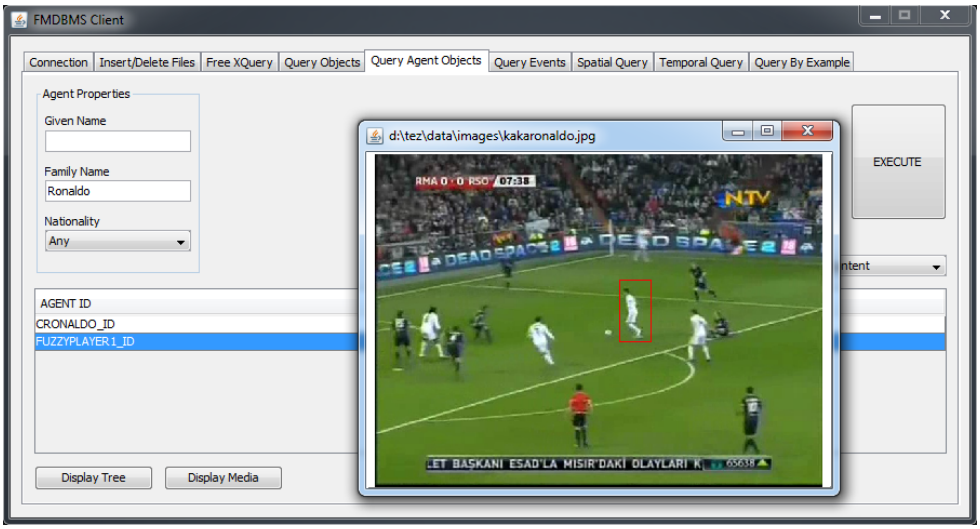


Figure 5.10: Screenshot of the agent object media display.

Usage of event querying tab is similar to the agent querying tab; but it is a little more complicated since an event contains more information than an agent object. The event querying tab is for displaying events and fuzzy events stored in the database. The user selects or defines the condition information supplied in the event querying tab. The XQuery string is automatically generated by the software. The querying part of the event querying tab consists of three main parts, namely querying by agent, querying by event type and querying by fuzzy event properties as described below:

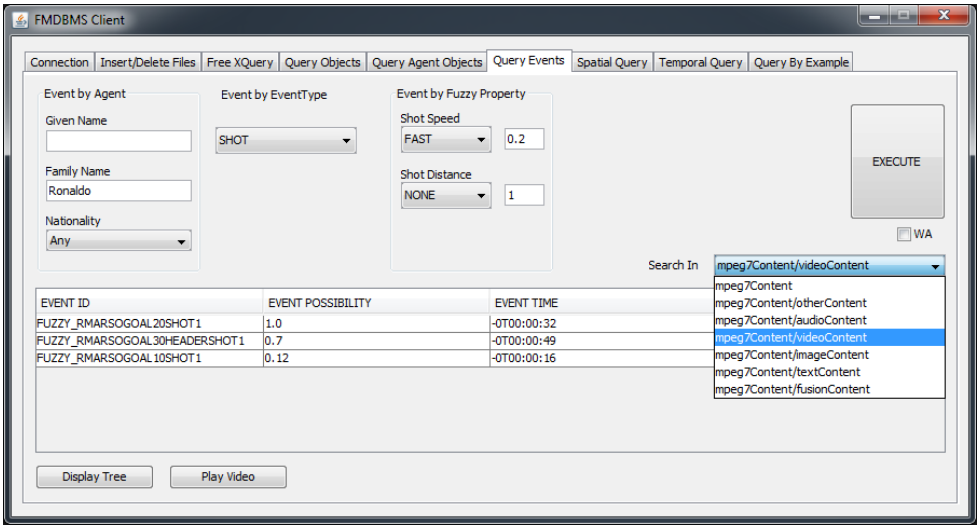


Figure 5.11: Screenshot of the event query tab.

- Querying by Agent: The *Event by Agent* groupbox is used to query the events by the agent object information that causes or does the action in the event. Similar to the agent querying tab, there are *Given Name* and *Family Name* text areas, and a *Nationality* combobox. If all of the elements are left in their original states, the agent object information will not be included in the event query. For example, if the user types *Ronaldo* as the family name, selects *Portuguese* as the nationality, and leaves given name empty, provided all other GUI elements are left in their initial states, the query is constructed as "select all events and fuzzy events, caused or done by the agent with family name Ronaldo and nationality Portuguese".

- Querying by Event Type: The *Event by Event Type* combobox is used to query the events by its type. Using the combobox, one selects a predefined event type, say GOAL. Provided that all other GUI elements are left in their initial states, the query is constructed as "select all events tagged as GOAL".

- Querying by Fuzzy Properties: The *Event by Fuzzy Property* groupbox is used to query the event by its fuzzy properties, in other words via linguistic terms. *Shot Speed*, *Shot Distance* and *Pass Distance* comboboxes are supplied in the groupbox. To the right of each combobox, a text area is supplied to define the similarity thresholds, to be used while computing the similar linguistic values in the same FuzzyVariableType instance. For example, let the user select the SHOT type in *Event by Event Type* combobox.

Assume that *CLOSE* is selected as the shot distance property and its similarity threshold is set as 0.5. Let *FAST* be selected as the shot speed property and its similarity threshold be set as 0.33. Provided all other GUI elements are left in their initial states, the query is constructed as "select all events tagged as SHOT, with CLOSE shot distance with 0.5 threshold and FAST shot speed with 0.33 threshold". The results are filtered with all shot distance linguistic values having similarity to CLOSE more or equal to 0.5, which is likewise for the shot speed property.

All of the main parts discussed above can be used separately from each other, as well as they can be used altogether. When used altogether, the conditions of all cases are ANDed. The default AND operation is performed by multiplying the possibility and similarity values of each condition. Using the WA option, activated via the WA checkbox on the tab, the weight of each condition is multiplied with its possibility or similarity value and the sum of the results are displayed as the *WA Result* instead of event possibility.

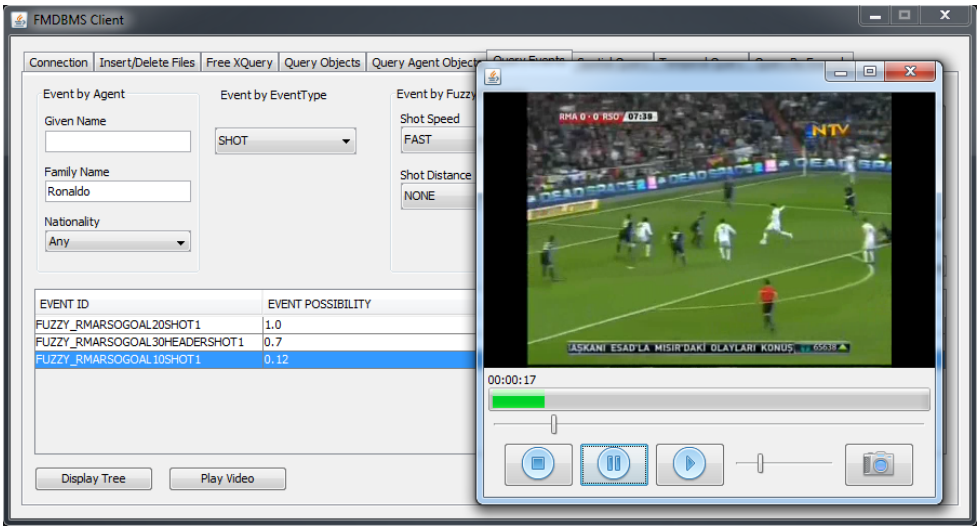


Figure 5.12: Screenshot of the event video player.

After executing a query, the results will be listed in the *Query Results* list. The user can display the video of the event by clicking on the *Play Video* button. Once this button is clicked, a video player window is displayed playing the video starting from the event occurrence time. A screenshot of event video player is given in Figure 5.12.

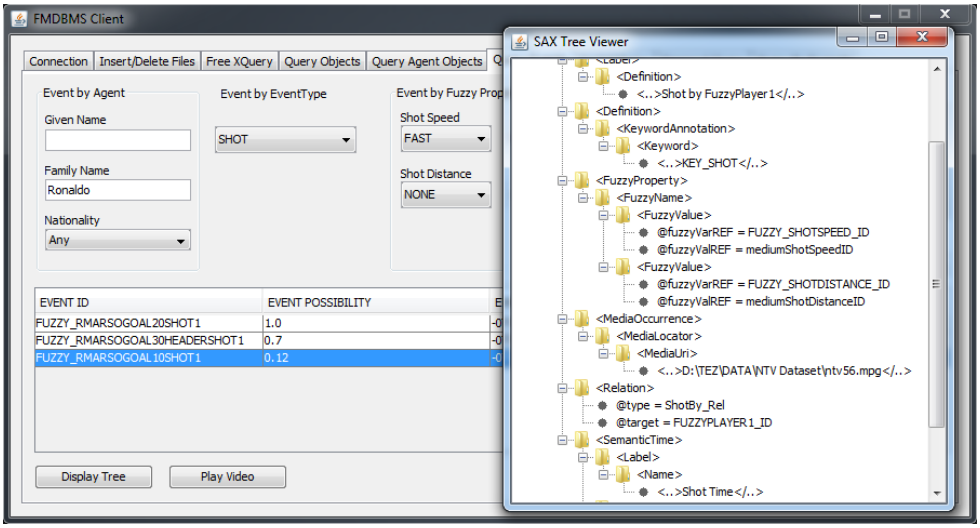


Figure 5.13: Screenshot of the event tree display.

Similar to the agent and object tree displays, the user can select a result from the results list and display it in tree view clicking on the *Display Tree* button located at the bottom of the window. A screenshot of event tree display is show in Figure 5.13.

5.2.7 Spatial Query Screen

A screenshot of the *Spatial Query* screen is given in Figure 5.14.

Using the spatial querying tab, the objects and the fuzzy objects in the database can be queried according to their spatial relations with each other. Similar to the object querying screen, the user can select the fuzzy properties of the source and target objects, and their spatial relation type. For instance, if the user selects *YELLOW* for the source object color, *RED* for the target object color and *LEFT* as the spatial relation, this means that s/he wants to query *all the YELLOW objects to the LEFT of the RED objects*. The spatial relation between objects are defined using the *RelationType* instances of MPEG-7 standard.

Spatial querying tab also supplies a textbox element for the user to define a *Time Point* information in his/her query. This way, the user can also apply a time filtering to the spatial relations s/he is searching for. If the objects in the relation are both annotated with a kind of time point statement, that specific relations are returned as results of the query. If time point

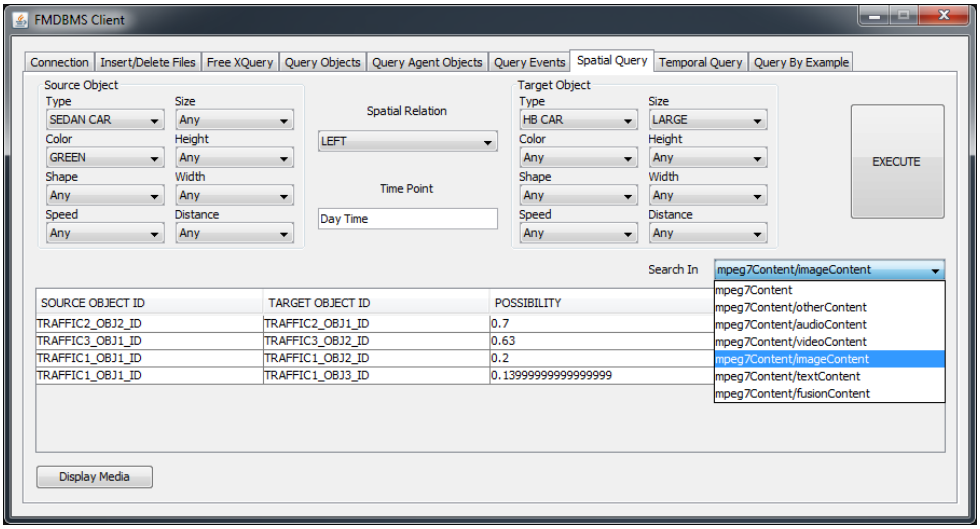


Figure 5.14: Screenshot of the spatial query tab.

text box is left empty, it is not included in the query as a filtering step. With the *Time Point* text box, a very simple kind of spatio-temporal querying is added to the FMDBMS Client.

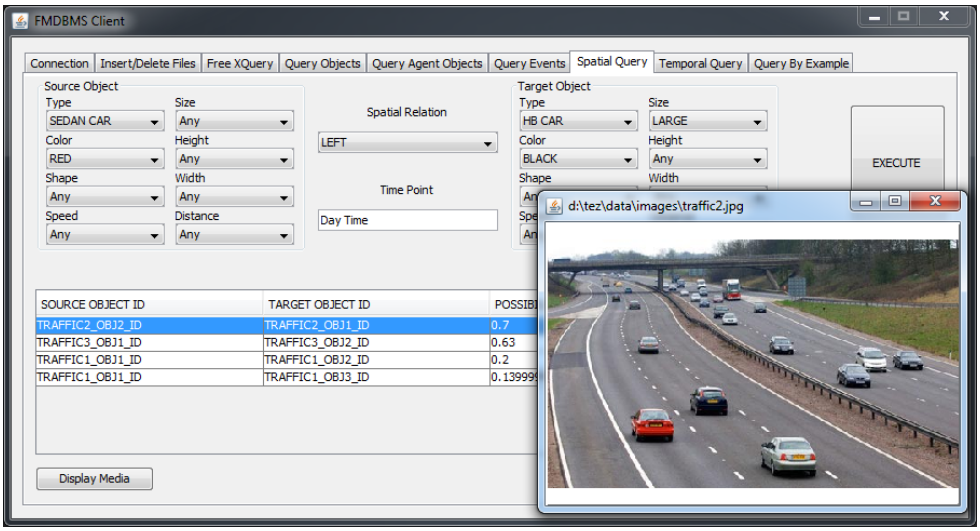


Figure 5.15: Screenshot of the spatial media display.

After executing a query, the results will be listed in the *Query Results* list. User can display the media file of the source object by clicking on the *Display Media* button. A screenshot of spatial query media display is given in Figure 5.15.

5.2.8 Temporal Query Screen

A screenshot of the *Temporal Query* screen is given in Figure 5.16.

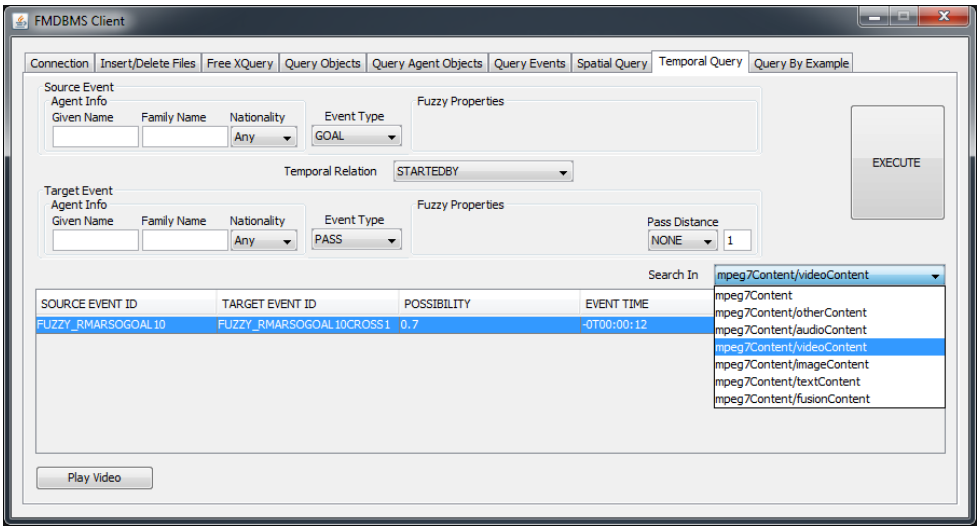


Figure 5.16: Screenshot of the temporal query tab.

Using the temporal querying tab, the events and the fuzzy events in the database can be queried according to their temporal relations with each other. Similar to the event querying screen, the user can select the fuzzy properties of the source and target events, and their temporal relation type. For instance, if the user selects *GOAL* for the source event type, *CROSS* for the target event type and *STARTEDBY* as the temporal relation, this means that s/he wants to query *all the GOAL events STARTING WITH a CROSS event*. The temporal relation between events are defined using the *RelationType* instances of MPEG-7 standard.

After executing a query, the results will be listed in the *Query Results* list. User can display the media file of the source event by clicking on the *Play Video* button. A screenshot of temporal query video player is given in Figure 5.17.

5.2.9 Query By Example Screen

A screenshot of the *Query By Example* tab is given in Figure 5.18.

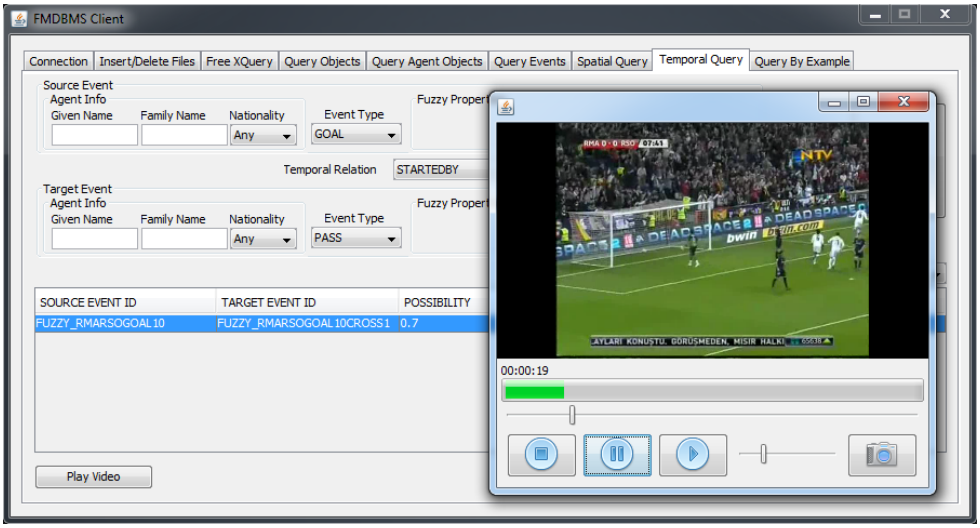


Figure 5.17: Screenshot of the temporal video player.

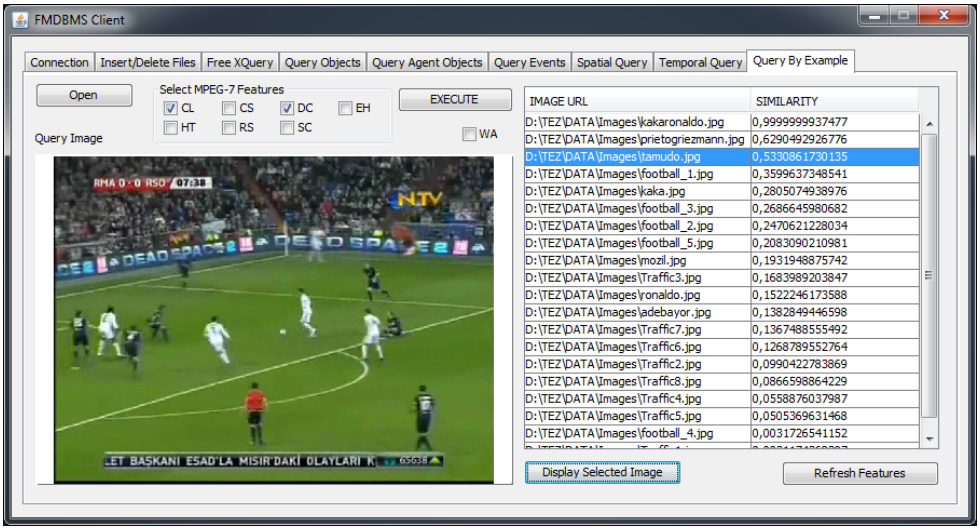


Figure 5.18: Screenshot of the query by example tab.

The query by example tab is implemented to find the similar images to a query image. The comparisons are performed on the low-level image features, namely color layout, scalable color, dominant color, edge histogram, homogeneous texture, contour shape and region shape features. The process consists of the following steps:

1. Create low-level feature database: In order to search for similar images in the database according to their low-level features, as the first step the low-level features of the im-

ages should be extracted. The low-level features of the images in a certain directory is extracted using XM Software feature extraction utilities, and the resulting features are inserted into the database. This step can be performed using the *Refresh Features* button located at the bottom of the tab.

2. Select a query image: As a second step, the user should select a query image in the file system. When s/he selects an image file, the content will be displayed on the *Query Image* part of the tab.

3. Select low-level features: The user has to select at least one low-level feature to perform her/his comparison. The low-level feature selection can be any combination of the low-level features available on the tab.

4. Find similar images: When the user clicks on the *Execute* button on the tab, the selected low-level features of the query image are extracted. Then each query feature will be compared with the corresponding feature of each image created in the first step. The comparison is performed using XM Software feature comparison utilities.

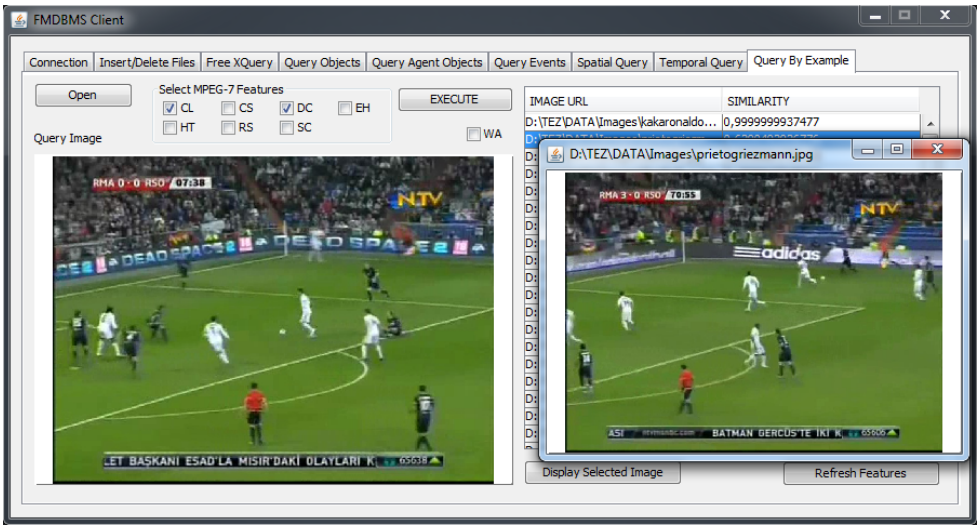


Figure 5.19: Screenshot of the query by example media display.

5. Display similar images: Comparison result of each low-level feature is normalized. If more than one low-level feature is selected in the third step, by default, individual normalized feature comparison results of an image are multiplied and the result will be displayed as the similarity of that image to the query image. On the other hand, using

the WA option, activated via the WA checkbox on the tab, the weight of each low-level feature will be multiplied by its normalized similarity value and all those results are added together to display the *WA Result* instead of the similarity value.

After executing a query, the results will be displayed in the list to the right of the tab. If the user selects a result in the list and clicks on the *Display Selected Image* button at the bottom of the list, the selected image file will be shown on a separate image display window. A screenshot of the *Query By Example* image display window can be seen in Figure 5.19.

The overview diagram of the whole FMDBMS system is given in Figure 5.20.

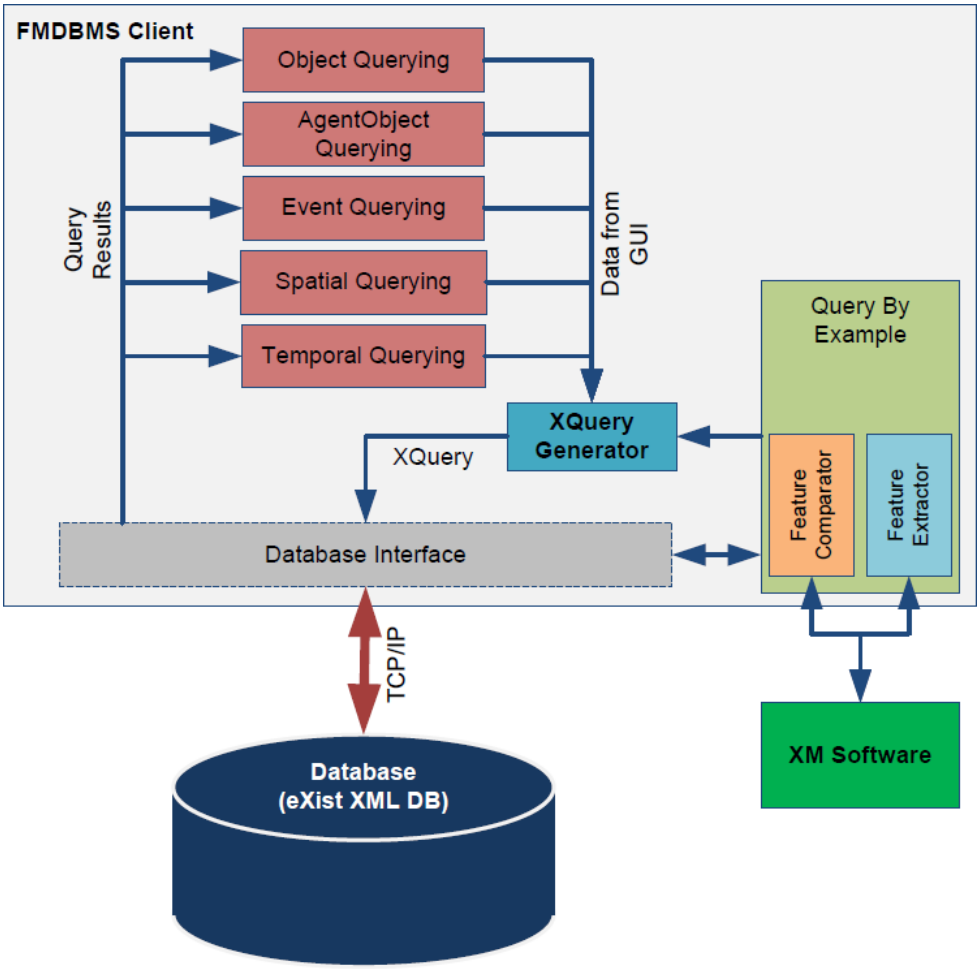


Figure 5.20: Overview of FMDBMS system.

CHAPTER 6

DOCUMENT CONTENT MANAGER IMPLEMENTATION

Document Content Manager Software is developed in order to manage the MPEG-7 documents in the file system. It is designed to have two main parts. These parts are Document Converter and Document Manager parts, which are described in detail in following sections. Both parts makes excessive use of JAXB tool, which is described in Chapter 2. Moreover, the document to be converted or managed is assumed to conform to the DAVP-2004 MPEG-7 schema.

6.1 Document Converter

It is possible to semantically annotate multimedia documents using the *ContentEntityType* description tool of MPEG-7 standard. However, *SemanticDescriptionType* description tool is more elegant for semantic annotation, supplying more description tools for the semantic entities like events, objects and agent objects.

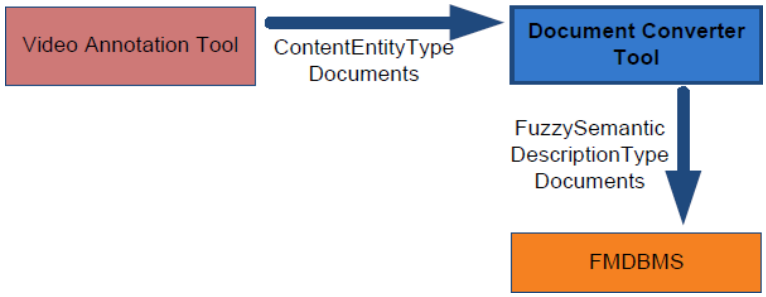


Figure 6.1: Block diagram describing the Document Converter tool.

Document Converter is responsible for converting the *ContentEntityType* description instances into *FuzzySemanticDescriptionType* description instances. Such a conversion is required since it is desired to have all of the database content to be queried successfully using the FMDBMS client software, which is described in Chapter 5. Figure 6.1 shows a block diagram describing the role of the Document Converter.

6.1.1 Conversion Process

Conversion of *ContentEntityType* descriptions into *FuzzySemanticDescriptionType* instances involves several steps, which can be described as follows:

- Media Locator Information: Media locator information is used in order to define the location of the multimedia document, on which the semantic annotation is performed, in the file system or on the web. This information is available under the *AudioVisual* element of type *AudioVisualSegmentType* in *ContentEntityType* descriptions. This information will be placed under the instances of events, objects and agent objects.

- Conversion of Events: In *ContentEntityType* descriptions, semantic event annotations are placed under *TemporalDecomposition/AudioVisualSegment* elements. The *ID* of the *AudioVisualSegment* element becomes the ID of the converted *FuzzyEventType* instance. The text annotation of *AudioVisualSegment* element is kept as the *Label* information of *FuzzyEventType* instance after conversion. All *Relation* elements defined in *AudioVisualSegment* are kept the same in the converted *FuzzyEventType*. The sub-events of an event are annotated under a *TemporalDecomposition* element under an *AudioVisualSegment* element in *ContentEntityType* descriptions. These sub-events are converted to sub-events of the *FuzzyEventType* instance recursively. The media time information of *AudioVisualSegment* element is converted to a *SemanticTimeType* instance and added to the *FuzzyEventType* instance. The media locator information, described in the first step, is copied the same to the *FuzzyEventType* object.

- Conversion of Objects: In *ContentEntityType* descriptions, semantic objects are annotated as *StillRegion* elements of type *StillRegionType*, which are placed under *SpatioTemporalDecomposition* elements of type *MovingRegionSpatioTemporalDecompositionType* elements with criteria attribute equal to the value “objects”. All these *StillRe-*

gion elements are converted to *FuzzyObjectType* instances. The *id* attributes of *StillRegion* elements are kept the same. The *TextAnnotation* element of each *StillRegion* instance is converted to *Label* information to be put under the corresponding *FuzzyObjectType* instance.

- Conversion of Agent Objects: In *ContentEntityType* descriptions, semantic agent objects are annotated as *StillRegion* elements of type *StillRegionType*, which are placed under *SpatioTemporalDecomposition* elements of type *MovingRegionSpatioTemporalDecompositionType* elements with criteria attribute equal to the value "agents". All these *StillRegion* elements are converted to *FuzzyAgentObjectType* instances. The *id* attributes of *StillRegion* elements are kept the same. The *TextAnnotation* element of each *StillRegion* instance is converted to the *GivenName* element of the corresponding *FuzzyAgentObjectType* instance.

An example of the *ContentEntityType* description and its converted form of *FuzzySemanticDescriptionType* can be found in Appendix B.

6.2 Document Manager

Document Manager part is responsible for changing the content of the MPEG-7 description documents which are of type *SemanticDescriptionType* or *FuzzySemanticDescriptionType*. The software can also create a new MPEG-7 description document based on these types. Actually, Document Manager tool can be thought as a simple manual video annotation tool, with a video player embedded in it.

The user of the system is able to edit almost the whole content of an MPEG-7 description document, provided that it is syntactically valid against the fuzzy extended DAVP-2004 schema. The content of the document can be edited in several levels of the MPEG-7 description document. In the following subsections, the detailed description of Document Manager can be found.

6.2.1 Mpeg7 Element Editor

Mpeg7 element of *Mpeg7Type* is the root element of all MPEG-7 description documents. *Mpeg7* elements contain *Description* elements of type *ContentEntityType*, *SemanticDescriptionType*, *FuzzySemanticDescriptionType* etc. Using the Mpeg7 Element Editor, whose snapshot is given in Figure 6.2, the user is able to do the following:

- Add new *Description* elements of *SemanticDescriptionType* or remove already existing instances.
- Add new *Description* elements of *FuzzySemanticDescriptionType* or remove already existing instances.

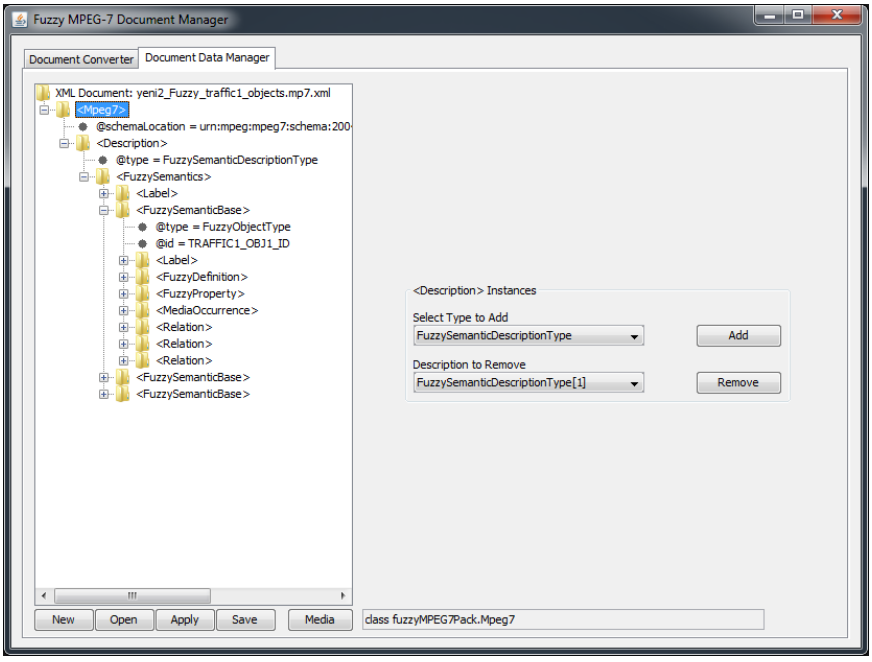


Figure 6.2: Mpeg7 element content editing screen.

6.2.2 Description Element Editor

Description elements of types *FuzzySemanticDescriptionType* and *SemanticDescriptionType* are placed as child elements of the *Mpeg7* element. These elements contain *SemanticType* or

FuzzySemanticType instances depending on their types. Using Description Element Editor, whose snapshot is given in Figure 6.3, the user is able to do the following:

- Add new instances of *SemanticType* or remove already existing instances.
- Add new instances of *FuzzySemanticType* or remove already existing instances.

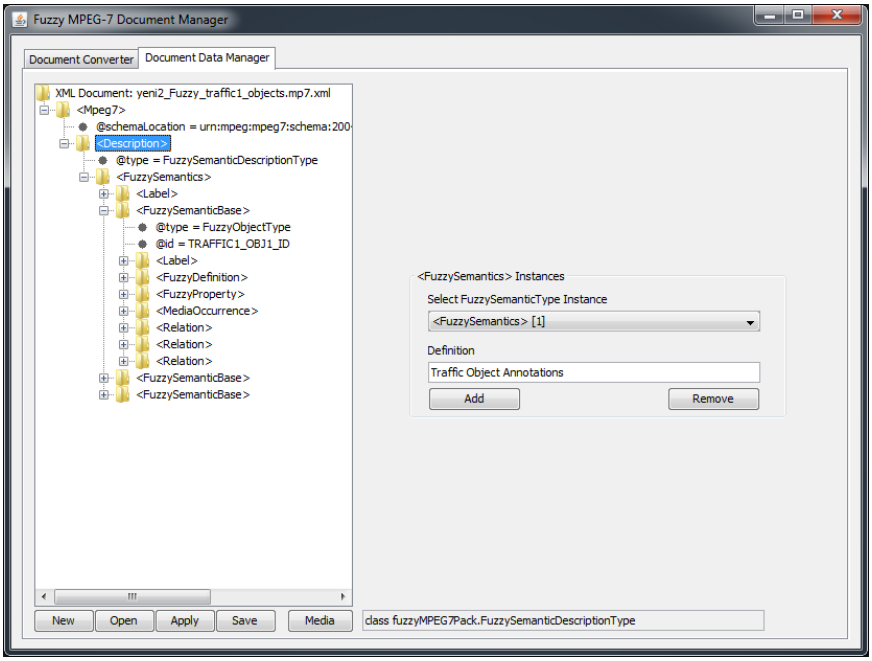


Figure 6.3: Description element content editing screen.

6.2.3 FuzzySemantics Element Editor

Semantics elements of type *SemanticType* and *FuzzySemantics* elements of type *FuzzySemanticsType* are placed as child elements of *Description* elements. *Semantics* elements may contain several elements of *ObjectType*, *EventType* and *AgentObjectType* elements and their fuzzy extended forms. Using FuzzySemantics Element Editor, whose snapshot is given in Figure 6.4, the user is able to do the following:

- Add instances of fuzzy and regular events with an ID value or remove already existing instances.

- Add instances of fuzzy and regular objects with an ID value or remove already existing instances.

- Add instances of fuzzy and regular agent objects with an ID value or remove already exiting instances.

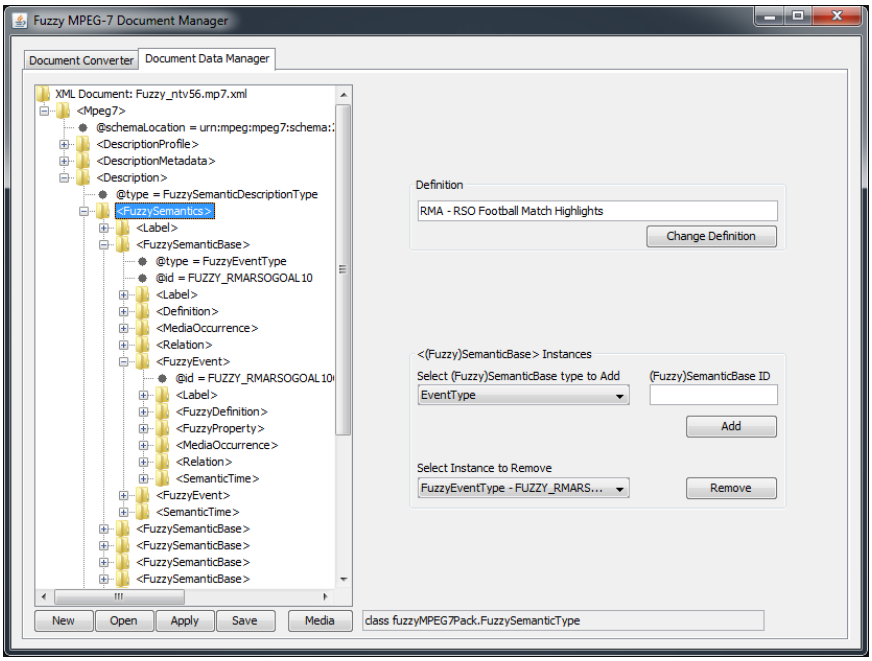


Figure 6.4: FuzzySemantics element content editing screen.

6.2.4 FuzzyEvent Element Editor

Instances of *FuzzyEventType* and *EventType* are placed under *FuzzySemantics* or *Semantics* elements. As described previously, these semantic description tools are implemented to annotate an event in multimedia environment. Using the FuzzyEvent Element Editor, whose snapshot is given in Figure 6.5, the user is able to do the following:

- Edit ID and Label (Definition) information of an event.

- Edit the Keyword information of an event. In this study, keywords are used to define the event types of a fuzzy event instance.

- Edit the Fuzzy Properties of an event. In this study, Fuzzy Properties are used to define the fuzzy properties of an event such as pass speed, shot speed etc.
- Add new relations or remove the existing relations of an event.
- Edit the media information of an event, which are media file path, starting time point and duration of an event.

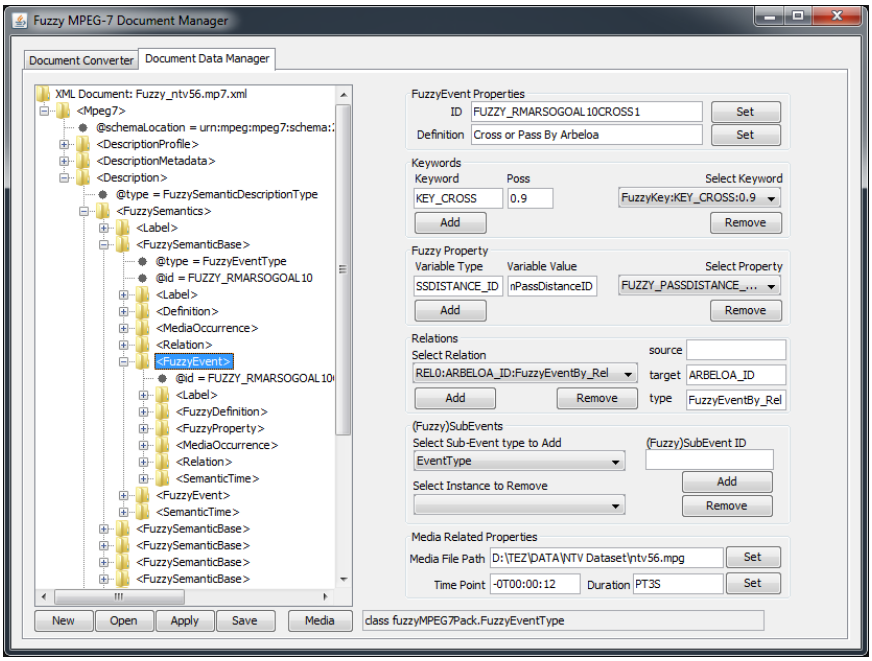


Figure 6.5: FuzzyEvent element content editing screen.

6.2.5 FuzzyObject Element Editor

Instances of *FuzzyObjectType* and *ObjectType* are placed under *FuzzySemantics* or *Semantics* elements. As described previously, these semantic description tools are implemented to annotate an object in multimedia environment. Using the FuzzyObject Element Editor, whose snapshot is given in Figure 6.6, user is able to do the following:

- Edit ID and Label (Definition) information of an object.
- Edit the Keyword information of an object. In this study, keywords are used to define the object types of a fuzzy object instance.

- Edit the Fuzzy Properties of an object. In this study, Fuzzy Properties are used to define the fuzzy properties of an object such as its color, speed, distance and etc.
- Add new relations or remove the existing relations of an object.
- Edit the media file path information of an object.

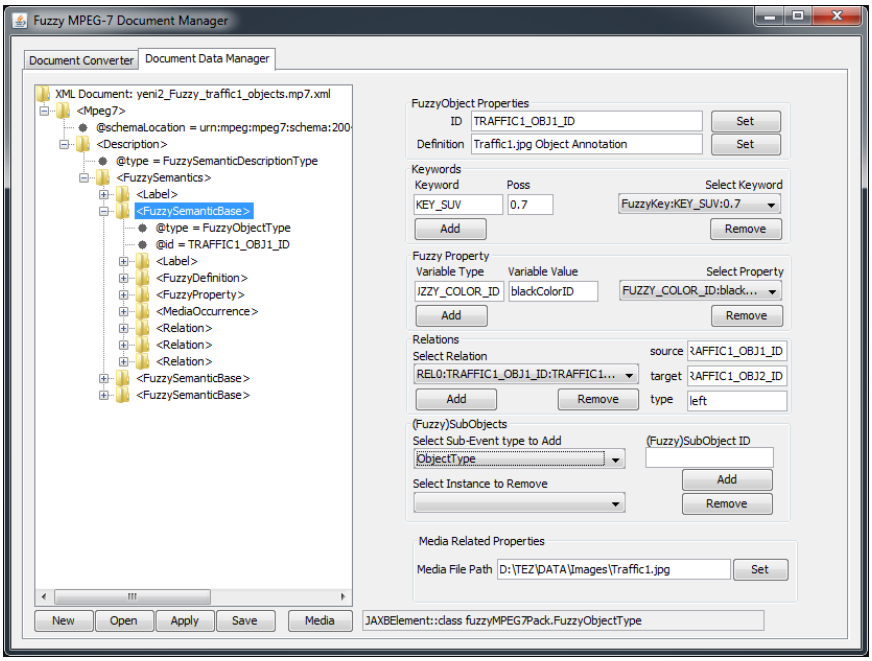


Figure 6.6: FuzzyObject element content editing screen.

6.2.6 FuzzyAgentObject Element Editor

Instances of *FuzzyAgentObjectType* and *AgentObjectType* are placed under *FuzzySemantics* and *Semantics* elements. As described previously, these semantic description tools are implemented to annotate an agent object in multimedia environment. Using FuzzyAgentObject Element Editor, whose snapshot is given in Figure 6.7, user is able to do the following:

- Edit ID and Label (Definition) information of an agent object.
- Edit the Keyword information of an agent object. In this study, keywords are used to define the agent object types: In football domain, for instance, it is used to define the position of a player such as forward, mid-field or defense.

- Edit the Fuzzy Properties of an agent object. In this study, Fuzzy Properties are used to define the fuzzy properties of an agent object such as its height, age and etc.
- Add new relations or remove the existing relations of an agent object.
- Edit the media file path information of an agent object.
- Edit the agent data, i.e. given name, family name, nationality, of an agent object.

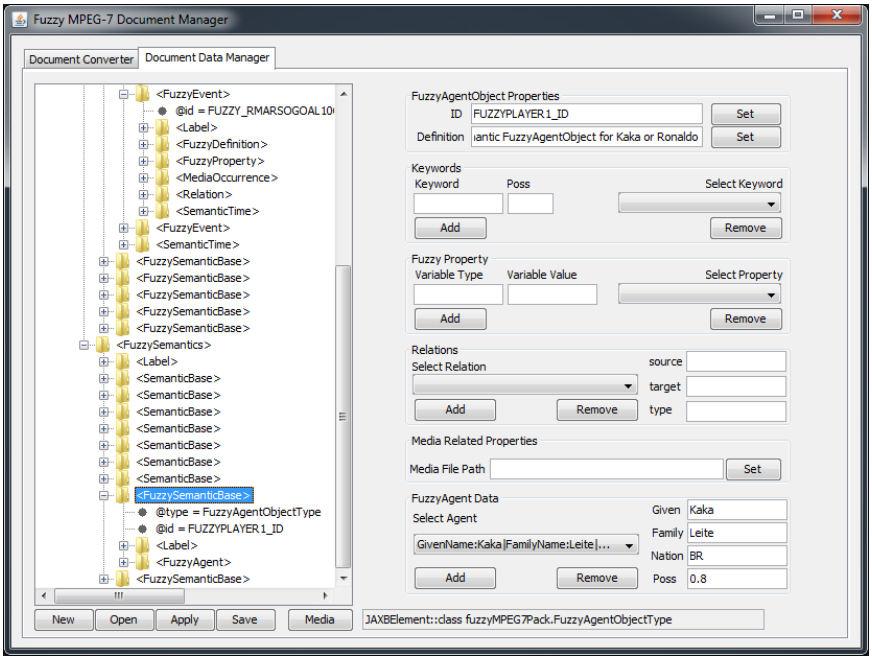


Figure 6.7: FuzzyAgentObject element content editing screen.

6.2.7 Relation Element Editor

Instances of *RelationType* can be found in several locations in an MPEG-7 description document, such as events, agent objects, objects and their fuzzy extended forms. *Relation* elements are used to relate two elements in a description document, depending on their ID values, along with the type of the relation. Using the Relation Element Editor, whose snapshot is given in Figure 6.8, source and target IDs or the type information of a *Relation* can be edited.

As mentioned before, using the element content editors that are described above along with

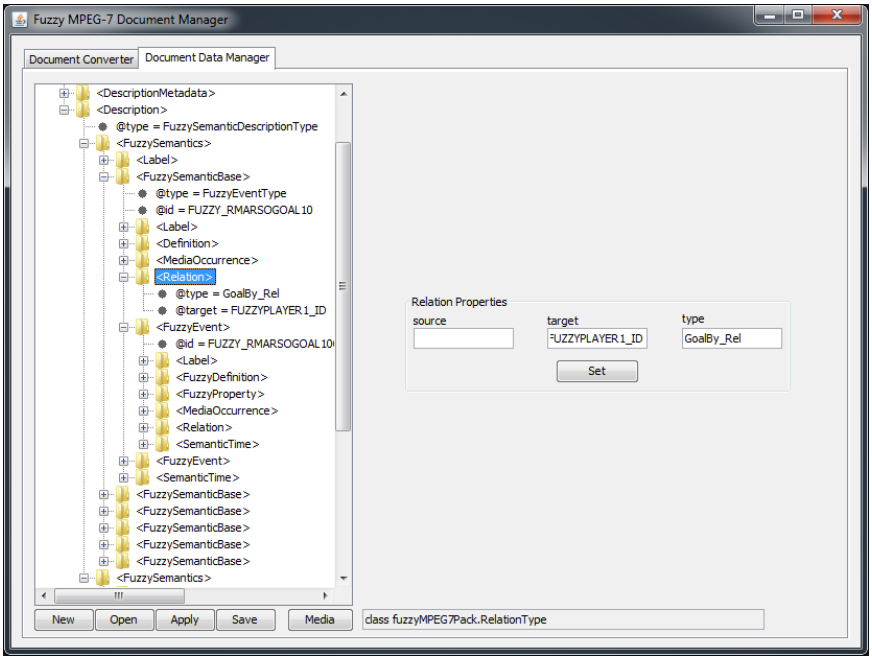


Figure 6.8: Relation element content editing screen.

the embedded video player of the Document Content Manager tool, the content manager can be used as a very basic multimedia semantic document annotator.

CHAPTER 7

CASE STUDY

In this chapter, the usage of the FMDBMS client, which is described in detail in Chapter 5, is shown with some examples. In order to analyze the output of the queries properly, the data itself is given wherever needed.

7.1 Querying Objects

In the FMDBMS client, *Query Objects* tab can be used to query object annotations in the database. The properties of objects given in the tab are color, shape, size, height, width, speed and distance of the object, as shown in Figure 5.5.

Example: In this example, the *SEDAN* vehicles which have *RED* as their *COLOR* (with similarity threshold 0.5) and *TALL* as their *HEIGHT* (with similarity threshold 0.3) are queried. The corresponding query is "Return all RED and TALL SEDAN vehicles". In order construct this query, the user selects the relevant values in *Query Objects* tab, then clicks on the EXECUTE button. The results are displayed in *Query Results* list.

Table 7.1: Fuzzy terms of objects and similarity values.

FUZZY TYPE	TERM 1	TERM 2	SIMILARITY
COLOR	RED	DARK RED	0.7
	RED	LIGHT RED	0.7
	RED	ORANGE	0.3
HEIGHT	SHORT	MEDIUM	0.4
	SHORT	TALL	0
	MEDIUM	TALL	0.4

Table 7.2: ObjectType and FuzzyObjectType instances.

OBJECT ID	OBJECT DATA
OBJECT1_ID	Type : SEDAN(1.0)
	Color : RED
	Height : MEDIUM
OBJECT2_ID	Type : HATCHBACK(0.3) Type : SEDAN(0.9)
	Color : LIGHT RED
	Height : TALL
OBJECT3_ID	Type : SEDAN(1.0)
	Color : DARK RED
	Height : TALL

The fuzzy object variables and values pertaining to this example are given in Table 7.1. The associated *FuzzyObjectType* instances can be found in Table 7.2. Given the data in the latter tables, step by step analysis of the example query can be performed as follows:

1. As the first step, the similarity relations of fuzzy linguistic terms are used to calculate the possible values of the fuzzy object properties. The calculation requires the threshold values given in the query (0.5 for *COLOR = RED* and 0.3 for *HEIGHT = TALL*) and the similarity data given in Table 7.1. It can be seen that Color property may have *RED*, *DARK RED* and *LIGHT RED* values with 1.0, 0.7, 0.7 possibility values respectively. On the other hand, Height property may have *TALL* and *MEDIUM* values with 1.0 and 0.4 possibility values respectively.

2. In this step, the objects are filtered by their object types. The query in this step is simply "Return all objects with Object Type = *SEDAN*". As can be seen in Table 7.2, the object *OBJECT2_ID* is a *SEDAN* vehicle with a possibility value of 0.9. The rest are SEDAN vehicles with unity possibility. So the initial possibility value of *OBJECT2_ID*, i.e. 1, is multiplied with 0.9 and the results in Table 7.3 are accumulated to the next step.

3. In this step, the objects are filtered by their *COLOR* properties. The query in this step is simply "Return all objects from previous step with Color = *RED* (1.0) or *DARK RED* (0.7) or *LIGHT RED* (0.7)". The *COLOR* property of object *OBJECT2_ID* is *LIGHT RED* so the similarity value 0.7 is multiplied with its possibility from the first step. The *COLOR* property of object *OBJECT3_ID* is *DARK RED* so the similarity value 0.7

Table 7.3: Object query results after the second step.

OBJECT ID	OBJECT POSSIBILITY
OBJECT1_ID	1.0
OBJECT2_ID	0.9
OBJECT3_ID	1.0

is multiplied with its possibility from the first step. For the object *OBJECT1_ID*, the *Color* attribute is set as *RED* as can be seen in Table 7.2. So after this step, the results given in Table 7.4 are calculated and accumulated to the next step.

Table 7.4: Object query results after the third step.

OBJECT ID	OBJECT POSSIBILITY
OBJECT1_ID	1.0
OBJECT2_ID	0.63
OBJECT3_ID	0.7

4. Similar to the previous step, in this step the objects are filtered by their *HEIGHT* properties. The query in this step is simply "Return all objects from the previous step with Height = *TALL* (1.0) or *MEDIUM* (0.4)". The *HEIGHT* property of object *OBJECT1_ID* is *MEDIUM*, so the similarity value 0.4 is multiplied with its possibility value from the previous step and the results are displayed in the GUI as shown in Table 7.5.

Table 7.5: Sample object query results.

OBJECT ID	OBJECT POSSIBILITY
OBJECT1_ID	0.4
OBJECT2_ID	0.63
OBJECT3_ID	0.7

5. At this step, WA is employed while calculating the possibility of each result. Suppose the user started querying at the beginning by selecting the WA option. When the user clicks on the Execute button, a separate window is shown asking for the weight of each query parameter. Let the user input the weights of object type, object color and object height as 0.5, 0.3 and 0.2 respectively. This time, AND operation for each step

is performed using Equation (2.1). The results calculated according to the weights and the data in Tables 7.1 and 7.2 are given in Table 7.6.

Table 7.6: Sample object query WA results.

OBJECT ID	**TYPE** (Weight=0.5)	**COLOR** (Weight=0.3)	**HEIGHT** (Weight=0.2)	**WA RESULT**
OBJECT1_ID	SEDAN(1.0)	RED(1.0)	MEDIUM(0.4)	1.0*0.5 + 1.0*0.3 + 0.4*0.2 = **0.88**
OBJECT2_ID	SEDAN(0.9)	LIGHT RED(0.7)	TALL(1.0)	0.9*0.5 + 0.7*0.3 + 1.0*0.2 = **0.86**
OBJECT3_ID	SEDAN(1.0)	DARK RED(0.7)	TALL(1.0)	1.0*0.5 + 0.7*0.3 + 1.0*0.2 = **0.91**

7.2 Querying Agent Objects

In the FMDBMS client, *Query Agent Objects* tab can be used to query agent object annotations in the database. The properties of agent objects given in the tab are Given Name, Family Name and Nationality of the agent describing the Agent Object, which can be seen in Figure 5.8.

Example: In this example, the agent objects which have *Ronaldo* as their *Family Name* property, are queried. In order to do this, the user types in *Ronaldo* in the Family Name text area given in the *Query Agent Objects* tab, then clicks on the EXECUTE button. The results will be displayed in *Query Results* list.

The *AgentObject* and *FuzzyAgentObject* instances in the database pertaining to this query example are given in Table 7.7. Inspecting the data given in Table 7.7, the *Agent Possibility* values listed in Table 7.8 can be verified easily. It should be noted here that the possibility of *AgentObjectType* instances are implicitly equal to unity since they describe the agent(s) without a possibility distribution. On the other hand, *FuzzyAgentObjectType* instances include possibility distributions of agents, so the possibility values in the results are equal to the

Table 7.7: AgentObject and FuzzyAgentObject instances with FamilyName Ronaldo.

AGENT OBJECT ID	DATA TYPE	POSSIBILITY	DATA DESCRIPTION
CRONALDO_ID	AgentObjectType	1	Given Name : Cristiano Family Name : Ronaldo Nationality : Portuguese
FUZZYPLAYER1_ID	FuzzyAgentObjectType	0.7	Given Name : Kaka Family Name : Leite Nationality : Brazilian
		0.3	Given Name : Cristiano Family Name : Ronaldo Nationality : Portuguese

possibility values given in those distributions. The details of possibility distributions and their usage in *FuzzyAgentObjectType* instances are described in Chapters 3 and 4.

Table 7.8: Sample agent query results.

AGENT ID	AGENT POSSIBILITY
CRONALDO_ID	1
FUZZYPLAYER1_ID	0.3

In the *Query Results* table, if one selects a result and clicks on the *Display Tree* button, the data given in Table 7.7 are displayed in detail.

7.3 Querying Events

In the FMDBMS client, *Query Events* tab can be used to query events stored in the database. The events can be queried according to agent objects related with the event, event types, and the fuzzy event properties related to a specific event type. A screenshot of the *Query Events* tab is given in Figure 5.11.

Example: In this example, the events are filtered by all three subjects mentioned in the previous paragraph, namely agent objects (with Family Name = *Ronaldo*), event type (SHOT events) and fuzzy event properties (with Shot Speed = *FAST* with similarity threshold 0.2 and Shot Distance = *CLOSE* with similarity threshold 0.3). So the query is "Return all *FAST* and *CLOSE SHOT* events performed by the agent(s) with Family Name = *Ronaldo*." When the user constructs this query using the available GUI components as explained in Chapter 5 and

clicks on the EXECUTE button, the results are displayed in the *Query Results* table on the GUI.

Table 7.9: Fuzzy terms of events and similarity values.

FUZZY TYPE	TERM 1	TERM 2	SIMILARITY
SHOT SPEED	SLOW	MEDIUM	0.4
	SLOW	FAST	0
	MEDIUM	FAST	0.4
SHOT DISTANCE	CLOSE	MEDIUM	0.4
	CLOSE	FAR	0
	MEDIUM	FAR	0.4

Table 7.10: EventType and FuzzyEventType instances.

EVENT ID	EVENT DATA
FUZZY_RMARSOGOAL20SHOT1	EventType : SHOT (1.0)
	Performer : CRONALDO_ID
	Shot Speed : FAST
	Shot Dist : MEDIUM
FUZZY_RMARSOGOAL30HEADERSHOT1	EventType : HEADER (1.0) EventType : SHOT (0.7)
	Performer : CRONALDO_ID
	Shot Speed : FAST
	Shot Dist : CLOSE
FUZZY_RMARSOGOAL10SHOT1	EventType : SHOT (1.0)
	Performer : FUZZYPLAYER1_ID
	Shot Speed : MEDIUM
	Shot Dist : MEDIUM

The related *AgentObject* and *FuzzyAgentObject* instances are given in Table 7.7. The fuzzy linguistic terms and the similarity relations related with the query are available in Table 7.9. The *EventType* and *FuzzyEventType* instances related with this example can be found in Table 7.10. Given the data in the tables, step by step analysis of the example query can be performed as follows:

1. At this first step, the results of the agent query example explained in the previous section are used as they are, in order to find the agent objects with family name *Ronaldo*. The resulting agent object IDs and corresponding possibility values can be seen in Table 7.8. These results are used in the next step in order to filter all events according to their

performers.

2. At this step, the similarity relations of fuzzy linguistic terms are used to calculate the possible values of the fuzzy event properties. The calculation requires the threshold values given in the query (0.2 for *Shot Speed = FAST* and 0.3 for *Shot Distance = CLOSE*) and the similarity relations data given in Table 7.9. It can be seen that Shot Speed property may have *FAST* and *MEDIUM* values with 1.0 and 0.4 possibility values respectively, whereas Shot Distance property can be *CLOSE* with 1.0 and *MEDIUM* with 0.4 possibility values.

3. At this step, the events are filtered by their performers. The query in this step is simply "Return all events performed by *CRONALDO_ID* or *FUZZYPLAYER1_ID*". Note that the agent object IDs given in the query are the results of the first step. In event *FUZZY_RMARSOGOAL10SHOT1*, the performer agent is *FUZZYPLAYER1_ID*. The possibility of *FUZZYPLAYER1_ID* agent to have the Family Name *Ronaldo* is 0.3, according to the data given in Tables 7.7 and 7.10. So, after this step, the possibility value 0.3 for *FUZZY_RMARSOGOAL10SHOT1* is accumulated to the next step. The rest of the events in Table 7.10 are accumulated with possibility values of 1.0. The results and accumulated possibilities after this step are given in Table 7.11.

Table 7.11: Event query results after the third step.

EVENT ID	**EVENT POSSIBILITY**
FUZZY_RMARSOGOAL20SHOT1	1.0
FUZZY_RMARSOGOAL30HEADERSHOT1	1.0
FUZZY_RMARSOGOAL10SHOT1	0.3

4. At this step, the events from the third step are filtered by their event types. The query in this step is simply "Return all events from the previous step with Event Type = *SHOT*". As can be seen in Table 7.10, the event *FUZZY_RMARSOGOAL30HEADERSHOT1* is a SHOT event with possibility value 0.7. The rest are SHOT events with possibility values of 1.0. So the possibility value of *FUZZY_RMARSOGOAL30HEADERSHOT1* from the previous step is multiplied with the possibility calculated in this step and the results in Table 7.12 are accumulated to the next step.

5. At this step, the events from the fourth step are filtered by their *SHOT SPEED* prop-

Table 7.12: Event query results after the fourth step.

EVENT ID	EVENT POSSIBILITY
FUZZY_RMARSOGOAL20SHOT1	1.0
FUZZY_RMARSOGOAL30HEADERSHOT1	0.7
FUZZY_RMARSOGOAL10SHOT1	0.3

erties. The query in this step is simply "Return all events from the previous step with Shot Speed = *FAST* (1.0) or *MEDIUM* (0.4)". The similarity of *MEDIUM* and *FAST* linguistic terms are 0.4 as given in Table 7.9. The *SHOT SPEED* property of event *FUZZY_RMARSOGOAL10SHOT1* is *MEDIUM* so the similarity value of 0.4 is multiplied with its possibility from the fourth step. For the rest of the events, the *SHOT SPEED* property is set as *FAST* as can be seen in Table 7.10. So after this step, the results given in Table 7.13 are calculated and accumulated to the next step.

Table 7.13: Event query results after the fifth step.

EVENT ID	EVENT POSSIBILITY
FUZZY_RMARSOGOAL20SHOT1	1.0
FUZZY_RMARSOGOAL30HEADERSHOT1	0.7
FUZZY_RMARSOGOAL10SHOT1	0.12

6. At this step, the events from the fifth step are filtered by their *SHOT DISTANCE* properties. The query in this step is simply "Return all events from the previous step with Shot Distance = *CLOSE* (1.0) or *MEDIUM* (0.4)". The similarity of *MEDIUM* and *CLOSE* linguistic terms are 0.4 as given in Table 7.9. The *SHOT DISTANCE* properties of events *FUZZY_RMARSOGOAL20SHOT1* and *FUZZY_RMARSOGOAL10SHOT1* are *MEDIUM* so the similarity value of 0.4 is multiplied with their possibilities from the fifth step. For the other event, the SHOT DISTANCE property is set as *CLOSE* as can be seen in Table 7.10. So after this step, the results given in Table 7.14 are calculated and the results are printed on the GUI.

7. At this step, WA is employed while calculating the possibility of each result. Suppose the user started querying at the beginning by selecting the WA option. When the user clicks on the Execute button, a separate window is shown asking for the weight of each query parameter. Let the user input the weights of performer, event type, shot speed

Table 7.14: Sample event query results.

EVENT ID	EVENT POSSIBILITY
FUZZY_RMARSOGOAL20SHOT1	0.4
FUZZY_RMARSOGOAL30HEADERSHOT1	0.7
FUZZY_RMARSOGOAL10SHOT1	0.048

and shot distance as 0.3, 0.2, 0.25 and 0.25 respectively. This time, AND operation for each step is performed using Equation (2.1). Table 7.15 lists the results calculated according to the weights and the data in Tables 7.7, 7.9 and 7.10.

Table 7.15: Sample event query WA results.

EVENT ID	PERFORMER (Weight=0.3)	TYPE (Weight=0.2)	SPEED (Weight=0.25)	DISTANCE (Weight=0.25)	WA RESULT
FUZZY_RMARSOGOAL20SHOT1	CRONALDO_ID(1.0)	SHOT(1.0)	FAST(1.0)	MEDIUM(0.4)	1.0*0.3 + 1.0*0.2 + 1.0*0.25 + 0.4*0.25 = **0.85**
FUZZY_RMARSOGOAL30HEADERSHOT1	CRONALDO_ID(1.0)	SHOT(0.7)	FAST(1.0)	CLOSE(1.0)	1.0*0.3 + 0.7*0.2 + 1.0*0.25 + 1.0*0.25 = **0.94**
FUZZY_RMARSOGOAL10SHOT1	FUZZYPLAYER1_ID(0.3)	SHOT(1.0)	MEDIUM(0.4)	MEDIUM(0.4)	0.3*0.5 + 1.0*0.2 + 0.4*0.25 + 0.4*0.25 = **0.55**

7.4 Querying Spatial Relations

FMDBMS is capable of querying objects as explained before in this chapter. Spatial querying can be considered as an extension of object querying, giving the user the chance to query the spatial relations between two objects.

Spatial relations of two objects are represented by instances of *RelationType* of MPEG-7 standard. To visualize a relation element in MPEG-7, one can think of a directed arc drawn between two nodes. The properties of this imaginary arc are the *source ID*, *target ID* and

the *type* of the relation represented by this arc. In spatial terms, if we want to represent "OBJECT1 is to the LEFT of OBJECT2" using a relation element, we should create it as *<Relation source="OBJECT1" target="OBJECT2" type="LEFT"/>* and put it as a member of the *ObjectType* (or *FuzzyObjectType*) instance that represents *OBJECT1*. An example of spatial querying can be found below.

Example: In this example, the steps while executing the spatial query "Return all *BLACK* and *HATCHBACK* vehicles to the *LEFT* of the *YELLOW* and *SEDAN* vehicles" are described. The usage of *Spatial Querying* tab is described in detail in Chapter 5.

Table 7.16: FuzzyObjectType instances for the spatial query example.

OBJECT ID	**OBJECT DATA**
OBJECT1_ID	Type : SEDAN(1.0)
	Color : ORANGE
	Height : MEDIUM
OBJECT2_ID	Type : HATCHBACK(0.3) Type : SEDAN(0.9)
	Color : BLACK
	Height : TALL
	Relation : LEFT of OBJECT1_ID
OBJECT3_ID	Type : HATCHBACK(1.0)
	Color : GRAY
	Height : TALL
	Relation : LEFT of OBJECT1_ID

Table 7.17: Color similarity values for the spatial query example.

FUZZY TYPE	**TERM 1**	**TERM 2**	**SIMILARITY**
COLOR	ORANGE	YELLOW	0.4
	BLACK	GRAY	0.5

1. As the first step, the instances of the source object (HATCHBACK vehicles in BLACK color) are extracted from the database. In other words, the system performs object querying to extract BLACK HATCHBACK vehicles. Using the data supplied in Tables 7.16 and 7.17, the results of the first step are calculated as shown in Table 7.18.

2. At this step, the instances of the target object (SEDAN vehicles in YELLOW color) are extracted from the database. In other words, the system performs object querying to

Table 7.18: Spatial query results after the first step.

OBJECT ID	OBJECT POSSIBILITY
OBJECT2_ID	0.3
OBJECT3_ID	0.5

extract YELLOW SEDAN vehicles. Using the data supplied in Tables 7.16 and 7.17, the results of this step are calculated as shown in Table 7.19.

Table 7.19: Spatial query results after the second step.

OBJECT ID	OBJECT POSSIBILITY
OBJECT1_ID	0.4

3. At this step, the spatial relations defined in the results of first step are analyzed. Using the data given in Tables 7.18 and 7.16, it can be seen that the objects *OBJECT2_ID* and *OBJECT3_ID* have the spatial relations of type *LEFT* targeted to the object *OBJECT1_ID*. Since the object *OBJECT1_ID* is located as the target in both spatial relations, these results are returned as those of the query, where the possibility of each result is calculated to be the possibility of its source and target objects. The results of the spatial relations are given in Table 7.20.

Table 7.20: Sample spatial query results.

SOURCE ID	TARGET ID	POSSIBILITY
OBJECT2_ID	OBJECT1_ID	0.12
OBJECT3_ID	OBJECT1_ID	0.2

7.5 Querying Temporal Relations

FMDBMS is capable of querying events as described before in this chapter. Temporal querying can be considered as an extension of event querying, giving the user the chance to query the temporal relations between two events.

Temporal relations of two events are represented by instances of *RelationType* of MPEG-

7 standard. To visualize a relation element in MPEG-7, one can think of a directed arc drawn between two nodes. The properties of this imaginary arc are the *source ID*, *target ID* and the *type* of the relation represented by this arc. In temporal terms, if we want to represent "EVENT1 is FOLLOWS of EVENT2" using a relation element, we should create it as *<Relation source="EVENT1" target="EVENT2" type="FOLLOWS"/>* and put it as a member of the *EventType* (or *FuzzyEventType*) instance that represents *EVENT1*. An example of temporal querying can be found below.

Example: In this example, the steps while executing the temporal query "Return all *FAST SHOT* events that *FOLLOW LONG DISTANCE PASS* events" are described. The usage of *Temporal Querying* tab is described in detail in Chapter 5.

Table 7.21: FuzzyEventType instances for the temporal query example.

OBJECT ID	**OBJECT DATA**
EVENT1_ID	Type : PASS(0.6) Type : CROSS(0.7)
	Distance : LONG
EVENT2_ID	Type : PASS(1.0)
	Distance : MEDIUM
EVENT3_ID	Type : SHOT(1.0)
	Speed : MEDIUM
	Relation : FOLLOWS - EVENT1_ID
EVENT4_ID	Type : SHOT(0.5) Type : HEADERSHOT(0.7)
	Speed : FAST
	Relation : FOLLOWS - EVENT2_ID

Table 7.22: Similarity values for the temporal query example.

FUZZY TYPE	**TERM 1**	**TERM 2**	**SIMILARITY**
SHOT SPEED	SLOW	MEDIUM	0.4
	SLOW	FAST	0.0
	MEDIUM	FAST	0.4
PASS DISTANCE	CLOSE	MEDIUM	0.4
	CLOSE	FAR	0.0
	MEDIUM	FAR	0.4

1. As the first step, the instances of the source event (*FAST SHOT* events) are extracted from the database. In other words, the system performs event querying to extract *FAST*

SHOT events. Using the data supplied in Tables 7.21 and 7.22, the results of the first step are calculated as shown in Table 7.23.

Table 7.23: Temporal query results after the first step.

EVENT ID	EVENT POSSIBILITY
EVENT3_ID	0.4
EVENT4_ID	0.5

2. At this step, the instances of the target event (LONG PASS events) are extracted from the database. In other words, the system performs event querying to extract LONG PASS events. Using the data supplied in Tables 7.21 and 7.22, the result of this step are calculated as shown in Table 7.24.

Table 7.24: Temporal query results after the second step.

EVENT ID	EVENT POSSIBILITY
EVENT1_ID	0.6
EVENT2_ID	0.4

3. At this step, the temporal relations defined in the results of first step are analyzed. Using the data given in Tables 7.21 and 7.18, it can be seen that the events *EVENT3_ID* and *EVENT4_ID* have the temporal relations of type *FOLLOWS* targeted to the events *EVENT1_ID* and *EVENT2_ID* respectively. Since both these relations are consistent with the conditions of our initial query, these two relations are returned as results. The possibility of each relation is calculated as the multiplication of the possibilities of its source and target events. The result of the query are given in Table 7.25.

Table 7.25: Sample temporal query results.

SOURCE ID	TARGET ID	POSSIBILITY
EVENT3_ID	EVENT1_ID	0.24
EVENT4_ID	EVENT2_ID	0.2

7.6 Automatic XQuery Generation

All of the information that FMDBMS uses, including fuzzy variable declarations, events, objects, agents and their relations, are stored in eXist XML DB, as XML documents. In order to query the content, an automatic XQuery generator is implemented in FMDBMS client application. *XQuery Generator* is responsible for generating proper XQuery statements to query data in the XML database as a response to various needs of the user.

The query generation is performed according to the user input from the FMDBMS client GUI. Sample query generation for an object query is given in the following steps. The *Query Objects* window can be seen in Figure 5.5.

1. Choosing query parameters: The user has to choose some object parameters and define a threshold value for each parameter wherever possible. Suppose the user chooses the query parameters and corresponding threshold values as in Table 7.26, then s/he clicks on the *Execute* button.

Table 7.26: Sample object query parameters.

PARAMETER	VALUE	THRESHOLD
Object Type	SEDAN CAR	NA
Color	RED	0.1
Height	TALL	0.2

2. Finding fuzzy object color values and their similarities: According to Table 7.26, the user chooses RED as the object color and 0.1 as its similarity threshold. In order to use this information, the system has to create a query to list all the colors with similarity values to RED greater than 0.1. The sample object color query is given in Appendix C. After the query execution, the results given in Table 7.27 are returned from the database.

3. Finding fuzzy object height values and their similarities: According to Table 7.26, the user chooses TALL as the object height and 0.2 as its similarity threshold. At this point, the system has to list all the fuzzy height values with similarity values to TALL greater than 0.2. The sample object height query is given in Appendix C.

Table 7.27: Sample color query results.

COLOR ID	SIMILARITY
darkRedColorID	0.7
lightRedColorID	0.7
orangeColorID	0.3
redColorID	1.0

Table 7.28: Sample height query results.

HEIGHT ID	SIMILARITY
mediumHeightID	0.4
tallHeightID	1.0

4. Finding objects: At this step, the system has to extract all of the objects of type *SEDAN CAR* from the XML database with the fuzzy attributes found in previous steps. The object extraction query generated by the *XQuery Generator*, using the results of the previous steps, is given in Appendix C.

7.7 eXist XML DB Indexing Performance Analysis

As an additional part of this thesis work, the indexing performance of eXist XML DB is analyzed. eXist-db has several indexing methods, including range and fulltext indexing schemes. Range indexing and fulltext indexing schemes are used in FMDBMS data and queries, wherever applicable. The details of range and fulltext indexing schemes are explained in the following subsections.

7.7.1 Range Index

Range indexing is used to index typed element and attribute values which are commonly used in queries. Since the frequency at which an element or attribute is used is implementation-specific, the administrator of the eXist-db server can easily modify the values to apply indexing on. For example, the possibility and similarity values in fuzzy extended MPEG-7 schema implemented in this thesis are floating point numbers. When range index is applied on these

values, the queries including predicates over them are executed faster, since it helps the query optimizer to evaluate possibility and similarity values as floating point numbers, eliminating the step of parsing their string content in the document and convert that into floating point numbers. The performance improvement becomes significant as the number of documents including possibility and similarity values increases.

7.7.2 Fulltext Index

Fulltext (FT) indexing is used to index element and attribute values including text (string) values which are commonly used in queries. Since the frequency at which an element or attribute is used is implementation-specific, the administrator of the eXist-db server can easily modify the values to apply indexing on. For example, the Keyword values defining object and event types in fuzzy extended MPEG-7 schema implemented in this thesis are text values. When FT index is applied on these values, the queries including predicates over them are executed faster. The performance improvement becomes significant as the number of documents including Keyword values increases.

The querying performance of FMDBMS with and without using the mentioned indexing schemes are given in Figures 7.1-7.3. The eXist-db Client application is used during the tests. The execution times involve the time to extract data from the database and show the results in the client GUI. The tests are run on a machine with the given properties: Intel Core2Duo P8400 2.26 GHz processor, 3 GB RAM, Windows 7 Professional 32-bit operating system.

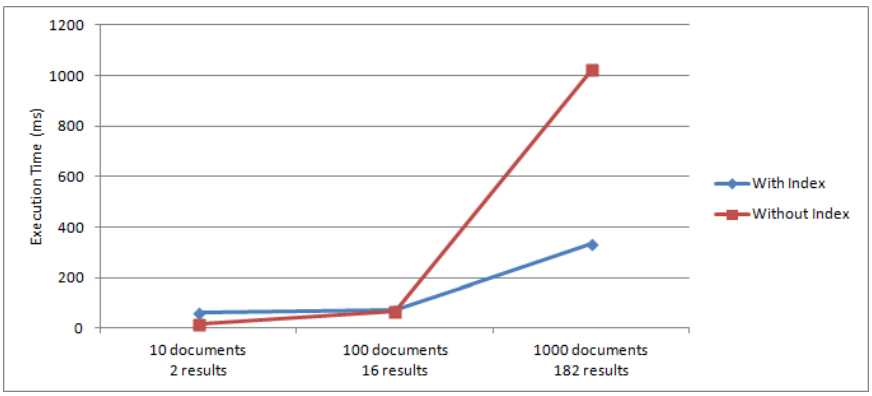


Figure 7.1: Execution times to extract all agent objects with Ronaldo family name.

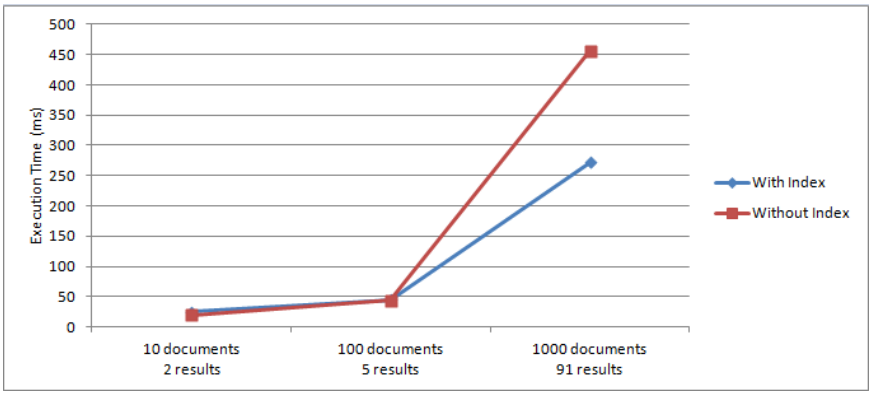


Figure 7.2: Execution times to extract all red hatchback vehicle objects.

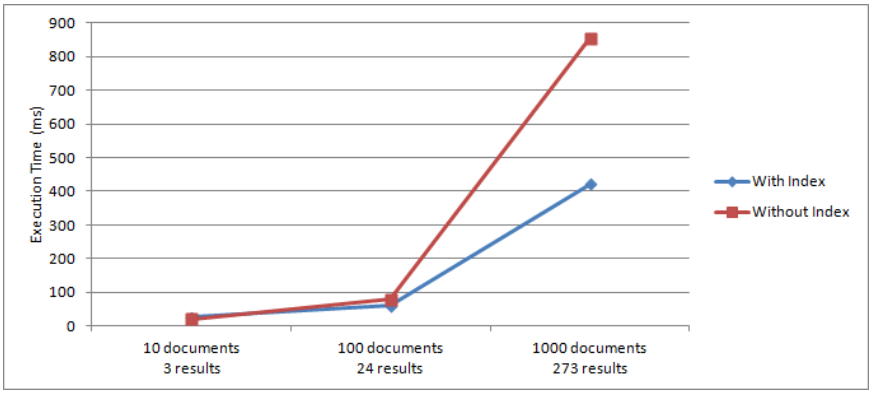


Figure 7.3: Execution times to extract all fast shot events performed by Ronaldo.

CHAPTER 8

CONCLUSION

MPEG-7 is a standard developed by Moving Picture Experts Group, in order to describe the multimedia content in textual form, as XML files. It contains several description tools, developed using XML Schema. Semantic description tools form a subset of whole set, and as the name implies, they are used to semantically describe the multimedia content. Event DS, Object DS and AgentObject DS are some examples of semantic description tools of MPEG-7 standard.

In this thesis, the set of semantic description tools are extended, actually on DAVP-2004 (Detailed Audio-Visual Profile) profile, in order to store fuzzy data in instances of semantic types. Since MPEG-7 description documents are XML documents, introducing fuzziness into MPEG-7 description documents can be regarded as introducing fuzziness into XML documents.

In order to store fuzzy data in XML documents, several studies are conducted. Using possibility distributions [19][18] is one of them, which suggests to cover a set of XML elements and their values with a special *Val* element, that has an attribute named *poss*. The value of *poss* attribute represents the possibility of occurrence of its child elements and values. Another one is to use fuzzy linguistic values [15] and their similarities. Usage of this technique enables one to define a fuzzy property in a semantic description type, for example the shot speed can be defined as *FAST* or *SLOW*. The similarity relations or membership functions of these linguistic terms can also be defined using XML schema.

MPEG-7 standard supplies a description definition language (DDL) which is based on XML Schema, for introducing new types. The fuzzy extension of MPEG-7 semantic description

types is conducted using DDL. The new fuzzy semantic types are capable of storing fuzzy data in terms of possibility distributions and fuzzy linguistic terms. Similarity relations and membership functions of linguistic terms are also declared in the extension.

The aim of the fuzzy extension is to prevent the data loss that may occur during the manual or automatic annotation of multimedia content. For example, say in a football match video, Mesut Ozil actually scores a goal. An automatic annotation system may annotate the agent object to be Cristiano Ronaldo with a normalized score 0.9, and the same agent object to be Mesut Ozil with the score 0.88. So the system decides that the agent in this event is Cristiano Ronaldo, which creates the loss of data that the agent object was actually Mesut Ozil. If we store this incorrect result in the database as it is, and query for the goals scores by Mesut Ozil, the system will not be able to return this event, missing the correct result. Using the fuzzy extension proposed in this thesis, the agent object can be defined including the information of both agents, Cristiano Ronaldo and Mesut Ozil, at the same time with their scores as their possibility values. This time the same query will also return this specific goal event with a degree of possibility, proving that the data is not lost during the annotation step. The same also applies to any kind of semantic entities, i.e. events or objects.

In order to manage the MPEG-7 documents, eXist-db is used as the storage base. eXist-db has the ability to query its content using XQuery. The client application developed in this study queries the content using automatically generated or user-typed XQuery statements, and displays the results in a proper format. The FMDBMS implementation developed in this thesis can be considered as proof of concept for the mentioned fuzzy extension scheme.

The study presented in this thesis can be improved further with the following future work:

- The fuzzy extension can also be extended if other fuzzy XML techniques can be discovered in literature, or a new fuzzy XML technique is introduced. It can be applied on MPEG-7 semantic description tools.
- Application of possibility distributions and linguistic terms on new fuzzy semantic types can be widened.
- Spatio-temporal querying can be adapted to the system in a more sophisticated way as presented in numerous studies in the literature.
- The query schemes can be extended so that the system would make use of membership

function declarations while querying fuzzy data. This way, fuzzy quantifiers can be adapted to the system for use in queries so FMBDMS would become a more powerful system.

- The FMDBMS client GUI can be created using configuration files so that its content can be modified without changing the code. This way, it would become a complete domain-independent system.

REFERENCES

[1] G. Akrivas, G. Stamou, and S. Kollias, "Semantic association of multimedia document descriptions through fuzzy relational algebra and fuzzy reasoning," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 34, no. 2, pp. 190–196, 2004.

[2] "MPEG-7 overview." from http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm, last accessed date: 05.06.2012.

[3] A. Gkoritsas and M. Angelides, "COSMOS-7: A video content modeling framework for MPEG-7," in *Multimedia Modelling Conference, 2005. MMM 2005. Proceedings of the 11th International*, pp. 123–130, IEEE, 2005.

[4] U. Westermann and W. Klas, "An analysis of XML database solutions for the management of MPEG-7 media descriptions," *ACM Computing Surveys (CSUR)*, vol. 35, no. 4, pp. 331–373, 2003.

[5] L. Zadeh, "Fuzzy sets*," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[6] L. Zadeh, "Fuzzy logic, neural networks, and soft computing," *Communications of the ACM*, vol. 37, no. 3, pp. 77–84, 1994.

[7] "Extensible Markup Language (XML)." from http://www.w3.org/XML/, last accessed date: 05.06.2012.

[8] "XML tutorial." from http://www.w3schools.com/xml/, last accessed date: 05.06.2012.

[9] "XML schema." from http://www.w3.org/XML/Schema, last accessed date: 05.06.2012.

[10] "XML schema tutorial." from http://www.w3schools.com/schema/, last accessed date: 05.06.2012.

[11] "XQuery 1.0: An XML Query Language (second edition)." from http://www.w3.org/TR/xquery/, last accessed date: 05.06.2012.

[12] A. Gaurav and R. Alhajj, "Incorporating fuzziness in XML and mapping fuzzy relational data into fuzzy XML," in *Proceedings of the 2006 ACM symposium on Applied computing*, pp. 456–460, ACM, 2006.

[13] M. Döller and H. Kosch, "The MPEG-7 multimedia database system (MPEG-7 MMDB)," *Journal of Systems and Software*, vol. 81, no. 9, pp. 1559–1580, 2008.

[14] U. Westermann and W. Klas, "PTDOM: a schema-aware XML database system for MPEG-7 media descriptions," *Software: Practice and Experience*, vol. 36, no. 8, pp. 785–834, 2006.

[15] Y. Jin and S. Veerappan, "A fuzzy XML database system: Data storage and query processing," in *Information Reuse and Integration (IRI), 2010 IEEE International Conference on*, pp. 318–321, IEEE, 2010.

[16] E. Ustunkaya, A. Yazici, and R. George, "Fuzzy data representation and querying in XML database," *International Journal of Uncertainty Fuzziness and Knowledge Based Systems*, vol. 15, p. 43, 2007.

[17] E. Damiani, L. Tanca, and F. Fontana, "Fuzzy XML queries via context-based choice of aggregations," *Kybernetika-Praha*, vol. 36, no. 6, pp. 635–656, 2000.

[18] Z. Ma and L. Yan, "Fuzzy XML data modeling with the UML and relational data models," *Data & Knowledge Engineering*, vol. 63, no. 3, pp. 972–996, 2007.

[19] L. Yan, Z. Ma, and J. Liu, "Fuzzy data modeling based on XML schema," in *Proceedings of the 2009 ACM symposium on Applied Computing*, pp. 1563–1567, ACM, 2009.

[20] L. Zadeh, "Similarity relations and fuzzy orderings+," *Information Sciences*, vol. 3, no. 2, pp. 177–200, 1971.

[21] T. Sudkamp, "Similarity as a foundation for possibility," in *Fuzzy Systems, 2000. FUZZ IEEE 2000. The Ninth IEEE International Conference on*, vol. 2, pp. 735–740, IEEE, 2000.

[22] "XQuery Tutorial." from http://www.w3schools.com/xquery/, last accessed date: 05.06.2012.

[23] "MPEG-1." from http://mpeg.chiariglione.org/standards/mpeg-1/mpeg-1.htm, last accessed date: 05.06.2012.

[24] "MPEG-2." from http://mpeg.chiariglione.org/standards/mpeg-2/mpeg-2.htm, last accessed date: 05.06.2012.

[25] "Overview of the MPEG-4 Standard." from http://mpeg.chiariglione.org/standards/mpeg-4/mpeg-4.htm, last accessed date: 05.06.2012.

[26] "Text of 15938-5 FCD Information Technology - Multimedia Content Description Interface - Part 5 Multimedia Description Schemes," 2000.

[27] P. Salembier and J. Smith, "MPEG-7 multimedia description schemes," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 11, no. 6, pp. 748–759, 2001.

[28] M. Aydınlılar, "Semi-Automatic Semantic Video Annotation Tool," Master's thesis, Middle East Technical University, Ankara, Turkey, December 2011.

[29] A. Benitez, H. Rising, C. Jorgensen, R. Leonardi, A. Bugatti, K. Hasida, R. Mehrotra, A. Murat Tekalp, A. Ekin, and T. Walker, "Semantics of Multimedia in MPEG-7," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 1, pp. 134–137, IEEE, 2002.

[30] M. R. Group, "MPEG-7 Interoperability, Conformance Testing and Profiling, v.2. ISO/IEC JTC1/SC29/WG11 N4039. Singapore," March 2001.

[31] M. R. Group, "MPEG-7 profiles and levels, ISO/IEC JTC1/SC29 15938-9," 2005.

[32] W. Bailer and P. Schallauer, "The Detailed Audiovisual Profile: Enabling Interoperability between MPEG-7 based Systems," in *In 12 th International MultiMedia Modelling Conference*, pp. 217–224, 2006.

[33] "Detailed Audiovisual Profile." from http://www.clinicip.org/index.php?id=119, last accessed date: 05.06.2012.

[34] M. R. Group, "Reference software, ISO/IEC JTC1/SC29 15938-6," 2003.

[35] "eXist-db Open Source Native XML Database." from http://exist.sourceforge.net/, last accessed date: 05.06.2012.

[36] "VideoLAN Organization." from http://www.videolan.org/vlc/, last accessed date: 05.06.2012.

[37] "vlcj - Java Framework for the VLC Media Player." from http://code.google.com/p/vlcj/, last accessed date: 05.06.2012.

[38] "JAXB Reference Implementation - Java.net." from http://jaxb.java.net/, last accessed date: 05.06.2012.

[39] J. Hellerstein, J. Naughton, and A. Pfeffer, *Generalized search trees for database systems*. September, 1995.

[40] Y. Chu, L. Chia, and S. Bhowmick, "SM3+: An XML database solution for the management of MPEG-7 descriptions," in *Database and Expert Systems Applications*, pp. 134–144, Springer, 2005.

[41] M. Bastan, H. Cam, U. Gudukbay, and O. Ulusoy, "BilVideo-7: an MPEG-7-compatible video indexing and retrieval system," *Multimedia, IEEE*, vol. 17, no. 3, pp. 62–73, 2010.

[42] A. Yazici, A. Soysal, B. Buckles, and F. Petry, "Uncertainty in a nested relational database model," *Data & Knowledge Engineering*, vol. 30, no. 3, pp. 275–301, 1999.

[43] P. Buche, J. Dibie-Barthélemy, O. Haemmerlé, and G. Hignette, "Fuzzy semantic tagging and flexible querying of XML documents extracted from the Web," *Journal of Intelligent Information Systems*, vol. 26, no. 1, pp. 25–40, 2006.

APPENDIX A

FUZZY MPEG-7 EXTENSIONS

A.1 FuzzySemanticBaseType

```
<complexType name="FuzzySemanticBaseType" 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"/>
    <choice minOccurs="0" maxOccurs="1">
     <element name="FuzzyDefinition" type="mpeg7:FuzzyTextAnnotationType"/>
     <element name="Definition" type="mpeg7:TextAnnotationType"/>
    </choice>
    <choice minOccurs="0">
     <element name="FuzzyProperty" type="mpeg7:FuzzyTermUseType"/>
     <element name="Property" type="mpeg7:TermUseType"/>
    </choice>
    <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" default="perceivable" use="optional">
       <simpleType>
        <union>
         <simpleType>
          <restriction base="NMTOKEN">
           <enumeration value="perceivable"/>
           <enumeration value="reference"/>
           <enumeration value="symbol"/>
          </restriction>
         </simpleType>
         <simpleType>
          <restriction base="mpeg7:termAliasReferenceType"/>
         </simpleType>
         <simpleType>
```

```
          <restriction base="mpeg7:termURIReferenceType"/>
         </simpleType>
        </union>
       </simpleType>
      </attribute>
     </complexType>
    </element>
    <element name="Relation" type="mpeg7:RelationType" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

A.2 FuzzyObjectType

```
<complexType name="FuzzyObjectType">
 <complexContent>
  <extension base="mpeg7:FuzzySemanticBaseType">
   <choice minOccurs="0" maxOccurs="unbounded">
    <element name="FuzzyObject" type="mpeg7:FuzzyObjectType"/>
    <element name="Object" type="mpeg7:ObjectType"/>
    <element name="ObjectRef" type="mpeg7:ReferenceType"/>
   </choice>
  </extension>
 </complexContent>
</complexType>
```

A.3 FuzzyAgentObjectType

```
<complexType name="FuzzyAgentObjectType">
 <complexContent>
  <extension base="mpeg7:FuzzyObjectType">
   <sequence minOccurs="0">
    <element name="FuzzyAgent" type="mpeg7:FuzzyAgentDistributionType"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

A.4 FuzzyEventType

```
<complexType name="FuzzyEventType">
 <complexContent>
  <extension base="mpeg7:FuzzySemanticBaseType">
   <sequence>
    <choice minOccurs="0" maxOccurs="unbounded">
     <element name="FuzzyEvent" type="mpeg7:FuzzyEventType"/>
     <element name="Event" type="mpeg7:EventType"/>
     <element name="EventRef" type="mpeg7:ReferenceType"/>
    </choice>
    <element name="SemanticPlace" type="mpeg7:SemanticPlaceType" minOccurs="0" maxOccurs="unbounded"/>
    <element name="SemanticTime" type="mpeg7:SemanticTimeType" minOccurs="0" maxOccurs="unbounded"/>
```

```
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

A.5 FuzzyVariableType

```
<complexType name="FuzzyVariableType">
 <sequence>
  <element name="FuzzyValue" type="mpeg7:FuzzyLinguisticTermType"
           minOccurs="1" maxOccurs="unbounded"/>
  <element name="SimilarityRelations" type="mpeg7:FuzzySimilarityRelationType"
           minOccurs="0" maxOccurs="1"/>
  <element name="MembershipFunctions" type="mpeg7:FuzzyMembershipFunctionCollectionType"
           minOccurs="0" maxOccurs="1"/>
 </sequence>
 <attribute name="fuzzyVarID" type="ID" use="required"/>
</complexType>
```

A.6 FuzzyLinguisticTermType

```
<complexType name="FuzzyLinguisticTermType">
 <simpleContent>
  <extension base="string">
   <attribute name="fuzzyValueID" type="ID" use="required"/>
  </extension>
 </simpleContent>
</complexType>
```

A.7 FuzzySimilarityRelationType

```
<complexType name="FuzzySimilarityRelationType">
 <sequence>
  <element name="Similarity" minOccurs="1" maxOccurs="unbounded">
   <complexType>
    <simpleContent>
     <extension base="mpeg7:FuzzySimilarityType"/>
    </simpleContent>
   </complexType>
  </element>
 </sequence>
</complexType>
```

A.8 FuzzySimilarityType

```
<complexType name="FuzzySimilarityType">
 <simpleContent>
  <extension base="mpeg7:zeroToOneType">
   <attribute name="fuzzyVal1REF" type="IDREF" use="required"/>
```

```
   <attribute name="fuzzyVal2REF" type="IDREF" use="required"/>
  </extension>
 </simpleContent>
</complexType>
```

A.9 FuzzyMembershipFunctionCollectionType

```
<complexType name="FuzzyMembershipFunctionCollectionType">
 <choice>
  <sequence>
   <element name="Triangular" minOccurs="1" maxOccurs="unbounded">
    <complexType>
     <complexContent>
      <extension base="mpeg7:FuzzyTriangularMembershipFunctionType"/>
     </complexContent>
    </complexType>
   </element>
  </sequence>
  <sequence>
   <element name="Triangular" minOccurs="1" maxOccurs="unbounded">
    <complexType>
     <complexContent>
      <extension base="mpeg7:FuzzyTrapezoidalMembershipFunctionType"/>
     </complexContent>
    </complexType>
   </element>
  </sequence>
  <sequence>
   <element name="Gaussian" minOccurs="1" maxOccurs="unbounded">
    <complexType>
     <complexContent>
      <extension base="mpeg7:FuzzyGaussianMembershipFunctionType"/>
     </complexContent>
    </complexType>
   </element>
  </sequence>
 </choice>
</complexType>
```

A.10 FuzzyTriangularMembershipFunctionType

```
<complexType name="FuzzyTriangularMembershipFunctionType">
 <attribute name="a" type="float" use="required"/>
 <attribute name="b" type="float" use="required"/>
 <attribute name="c" type="float" use="required"/>
 <attribute name="fuzzyValueREF" type="IDREF" use="required"/>
</complexType>
```

A.11 FuzzyTrapezoidalMembershipFunctionType

```
<complexType name="FuzzyTrapezoidalMembershipFunctionType">
 <attribute name="a" type="float" use="required"/>
```

```
 <attribute name="b" type="float" use="required"/>
 <attribute name="c" type="float" use="required"/>
 <attribute name="d" type="float" use="required"/>
 <attribute name="fuzzyValueREF" type="IDREF" use="required"/>
</complexType>
```

A.12 FuzzyGaussianMembershipFunctionType

```
<complexType name="FuzzyGaussianMembershipFunctionType">
 <attribute name="mu" type="float" use="required"/>
 <attribute name="stdDeviation" type="float" use="required"/>
 <attribute name="fuzzyValueREF" type="IDREF" use="required"/>
</complexType>
```

A.13 FuzzyTextualDistributionType

```
<complexType name="FuzzyTextualDistributionType">
 <sequence>
  <element name="Val" minOccurs="1" maxOccurs="unbounded">
   <complexType>
    <simpleContent>
     <extension base="string">
      <attribute name="poss" type="mpeg7:zeroToOneType" use="required"/>
     </extension>
    </simpleContent>
   </complexType>
  </element>
 </sequence>
</complexType>
```

A.14 FuzzyAgentDistributionType

```
<complexType name="FuzzyAgentDistributionType">
 <sequence>
  <element name="Val" minOccurs="1" maxOccurs="unbounded">
   <complexType>
    <choice>
     <element name="Agent" type="mpeg7:AgentType"/>
     <element name="AgentRef" type="mpeg7:ReferenceType"/>
    </choice>
    <attribute name="poss" type="mpeg7:zeroToOneType" use="required"/>
   </complexType>
  </element>
 </sequence>
</complexType>
```

APPENDIX B

DOCUMENT CONVERSION

B.1 ContentEntityType Description

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2004">
  <Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="AudioVisualType">
      <AudioVisual>
        <MediaLocator>
          <MediaUri>/home/merve/Desktop/ilker/ntv56/ntv56.mpg</MediaUri>
        </MediaLocator>
        <MediaTime>
          <MediaTimePoint>-0T00:00:00</MediaTimePoint>
          <MediaDuration>P0DT0H1M3S</MediaDuration>
        </MediaTime>
        <TemporalDecomposition criteria="events">
          <AudioVisualSegment id="EVT0">
            <TextAnnotation>
              <FreeTextAnnotation>Goal1</FreeTextAnnotation>
            </TextAnnotation>
            <Relation source="EVT0" target="EVT0SUB0" type="overlaps" />
            <Relation source="EVT0" target="EVT0SUB1" type="contains" />
            <Relation source="EVT0" target="EVT0SUB0" type="startedBy" />
            <MediaTime>
              <MediaTimePoint>-0T00:00:13</MediaTimePoint>
              <MediaDuration>PT6S</MediaDuration>
            </MediaTime>
            <TemporalDecomposition criteria="events">
              <AudioVisualSegment id="EVT0SUB0">
                <TextAnnotation>
                  <FreeTextAnnotation>Cross1</FreeTextAnnotation>
                </TextAnnotation>
                <Relation source="EVT0SUB0" target="EVT0SUB1" type="sequential" />
                <Relation source="EVT0SUB0" target="EVT0SUB1" type="precedes" />
                <MediaTime>
                  <MediaTimePoint>-0T00:00:12</MediaTimePoint>
                  <MediaDuration>PT3S</MediaDuration>
                </MediaTime>
              </AudioVisualSegment>
              <AudioVisualSegment id="EVT0SUB1">
                <TextAnnotation>
                  <FreeTextAnnotation>Shot1</FreeTextAnnotation>
                </TextAnnotation>
                <Relation source="EVT0SUB1" target="EVT0" type="finishes" />
                <MediaTime>
```

```
          <MediaTimePoint>-0T00:00:16</MediaTimePoint>
          <MediaDuration>PT4S</MediaDuration>
        </MediaTime>
      </AudioVisualSegment>
    </TemporalDecomposition>
  </AudioVisualSegment>
  <AudioVisualSegment id="EVT1">
    <TextAnnotation>
      <FreeTextAnnotation>Goal2</FreeTextAnnotation>
    </TextAnnotation>
    <MediaTime>
      <MediaTimePoint>-0T00:00:26</MediaTimePoint>
      <MediaDuration>PT8S</MediaDuration>
    </MediaTime>
    <TemporalDecomposition criteria="events">
      <AudioVisualSegment id="EVT1SUB0">
        <TextAnnotation>
          <FreeTextAnnotation>Pass1</FreeTextAnnotation>
        </TextAnnotation>
        <MediaTime>
          <MediaTimePoint>-0T00:00:25</MediaTimePoint>
          <MediaDuration>PT4S</MediaDuration>
        </MediaTime>
      </AudioVisualSegment>
      <AudioVisualSegment id="EVT1SUB1">
        <TextAnnotation>
          <FreeTextAnnotation>Shot2</FreeTextAnnotation>
        </TextAnnotation>
        <MediaTime>
          <MediaTimePoint>-0T00:00:30</MediaTimePoint>
          <MediaDuration>PT2S</MediaDuration>
        </MediaTime>
      </AudioVisualSegment>
    </TemporalDecomposition>
  </AudioVisualSegment>
</TemporalDecomposition>
<MediaSourceDecomposition criteria="modalities">
  <VideoSegment>
    <TemporalDecomposition criteria="visual shots">
      <VideoSegment id="SEG0">
        <StructuralUnit>
          <Name>Shot</Name>
        </StructuralUnit>
        <MediaTime>
          <MediaTimePoint>-0T00:00:00</MediaTimePoint>
          <MediaDuration>P0DT0H2M1S</MediaDuration>
        </MediaTime>
        <SpatioTemporalDecomposition criteria="moving objects">
          <MovingRegion>
            <SpatioTemporalDecomposition criteria="agents">
              <StillRegion id="KF396SR0">
                <TextAnnotation>
                  <FreeTextAnnotation>Mesut Ozil</FreeTextAnnotation>
                </TextAnnotation>
                <SpatialLocator>
                  <Polygon>
                    <Coords dim="1 4">557 140 23 51</Coords>
                  </Polygon>
                </SpatialLocator>
                <MediaTimePoint>-00T00:00:13</MediaTimePoint>
              </StillRegion>
              <StillRegion id="KF504SR0">
                <TextAnnotation>
```

```
                    <FreeTextAnnotation>Kaka</FreeTextAnnotation>
                  </TextAnnotation>
                  <SpatialLocator>
                    <Polygon>
                      <Coords dim="1 4">417 190 52 88</Coords>
                    </Polygon>
                  </SpatialLocator>
                  <MediaTimePoint>-00T00:00:16</MediaTimePoint>
                </StillRegion>
              </SpatioTemporalDecomposition>
            </MovingRegion>
          </SpatioTemporalDecomposition>
        </VideoSegment>
      </TemporalDecomposition>
    </VideoSegment>
  </MediaSourceDecomposition>
 </AudioVisual>
</MultimediaContent>
</Description>
</Mpeg7>
```

B.2 FuzzySemanticDescriptionType Description

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2004">
  <Description xsi:type="FuzzySemanticDescriptionType">
    <FuzzySemantics>
      <Label>
        <Definition>Semantic Event Annotations</Definition>
      </Label>
      <FuzzySemanticBase xsi:type="FuzzyEventType" id="EVT0">
        <Label>
          <Definition>Goal1</Definition>
        </Label>
        <MediaOccurrence>
          <MediaLocator>
            <MediaUri>/home/merve/Desktop/ilker/ntv56/ntv56.mpg</MediaUri>
          </MediaLocator>
        </MediaOccurrence>
        <Relation type="overlaps" source="EVT0" target="EVT0SUB0"/>
        <Relation type="contains" source="EVT0" target="EVT0SUB1"/>
        <Relation type="startedBy" source="EVT0" target="EVT0SUB0"/>
        <FuzzyEvent id="EVT0SUB0">
          <Label>
            <Definition>Cross1</Definition>
          </Label>
          <MediaOccurrence>
            <MediaLocator>
              <MediaUri>/home/merve/Desktop/ilker/ntv56/ntv56.mpg</MediaUri>
            </MediaLocator>
          </MediaOccurrence>
          <Relation type="sequential" source="EVT0SUB0" target="EVT0SUB1"/>
          <Relation type="precedes" source="EVT0SUB0" target="EVT0SUB1"/>
          <SemanticTime>
            <Label>
              <Definition>Event : EVT0SUB0 time info</Definition>
            </Label>
            <Time>
              <TimePoint>-0T00:00:12</TimePoint>
              <Duration>PT3S</Duration>
```

```
      </Time>
    </SemanticTime>
  </FuzzyEvent>
  <FuzzyEvent id="EVT0SUB1">
    <Label>
      <Definition>Shot1</Definition>
    </Label>
    <MediaOccurrence>
      <MediaLocator>
        <MediaUri>/home/merve/Desktop/ilker/ntv56/ntv56.mpg</MediaUri>
      </MediaLocator>
    </MediaOccurrence>
    <Relation type="finishes" source="EVT0SUB1" target="EVT0"/>
    <SemanticTime>
      <Label>
        <Definition>Event : EVT0SUB1 time info</Definition>
      </Label>
      <Time>
        <TimePoint>-0T00:00:16</TimePoint>
        <Duration>PT4S</Duration>
      </Time>
    </SemanticTime>
  </FuzzyEvent>
  <SemanticTime>
    <Label>
      <Definition>Event : EVT0 time info</Definition>
    </Label>
    <Time>
      <TimePoint>-0T00:00:13</TimePoint>
      <Duration>PT6S</Duration>
    </Time>
  </SemanticTime>
</FuzzySemanticBase>
<FuzzySemanticBase xsi:type="FuzzyEventType" id="EVT1">
  <Label>
    <Definition>Goal2</Definition>
  </Label>
  <MediaOccurrence>
    <MediaLocator>
      <MediaUri>/home/merve/Desktop/ilker/ntv56/ntv56.mpg</MediaUri>
    </MediaLocator>
  </MediaOccurrence>
  <FuzzyEvent id="EVT1SUB0">
    <Label>
      <Definition>Pass1</Definition>
    </Label>
    <MediaOccurrence>
      <MediaLocator>
        <MediaUri>/home/merve/Desktop/ilker/ntv56/ntv56.mpg</MediaUri>
      </MediaLocator>
    </MediaOccurrence>
    <SemanticTime>
      <Label>
        <Definition>Event : EVT1SUB0 time info</Definition>
      </Label>
      <Time>
        <TimePoint>-0T00:00:25</TimePoint>
        <Duration>PT4S</Duration>
      </Time>
    </SemanticTime>
  </FuzzyEvent>
  <FuzzyEvent id="EVT1SUB1">
    <Label>
```

```
            <Definition>Shot2</Definition>
          </Label>
          <MediaOccurrence>
            <MediaLocator>
              <MediaUri>/home/merve/Desktop/ilker/ntv56/ntv56.mpg</MediaUri>
            </MediaLocator>
          </MediaOccurrence>
          <SemanticTime>
            <Label>
              <Definition>Event : EVT1SUB1 time info</Definition>
            </Label>
            <Time>
              <TimePoint>-0T00:00:30</TimePoint>
              <Duration>PT2S</Duration>
            </Time>
          </SemanticTime>
        </FuzzyEvent>
        <SemanticTime>
          <Label>
            <Definition>Event : EVT1 time info</Definition>
          </Label>
          <Time>
            <TimePoint>-0T00:00:26</TimePoint>
            <Duration>PT8S</Duration>
          </Time>
        </SemanticTime>
      </FuzzySemanticBase>
    </FuzzySemantics>
    <FuzzySemantics>
      <Label>
        <Definition>Semantic Object and AgentObject Annotations</Definition>
      </Label>
      <FuzzySemanticBase xsi:type="FuzzyAgentObjectType" id="KF396SR0">
        <Label>
          <Definition>AgentObject : KF396SR0 : Mesut Ozil annotation</Definition>
        </Label>
        <FuzzyAgent>
          <Val poss="1.0">
            <Agent xsi:type="PersonType">
              <Name>
                <GivenName>Mesut Ozil</GivenName>
              </Name>
            </Agent>
          </Val>
        </FuzzyAgent>
      </FuzzySemanticBase>
      <FuzzySemanticBase xsi:type="FuzzyAgentObjectType" id="KF504SR0">
        <Label>
          <Definition>AgentObject : KF504SR0 : Kaka annotation</Definition>
        </Label>
        <FuzzyAgent>
          <Val poss="1.0">
            <Agent xsi:type="PersonType">
              <Name>
                <GivenName>Kaka</GivenName>
              </Name>
            </Agent>
          </Val>
        </FuzzyAgent>
      </FuzzySemanticBase>
    </FuzzySemantics>
  </Description>
</Mpeg7>
```

APPENDIX C

AUTOMATIC XQUERY GENERATION

C.1 Sample Object Color Query

```
declare namespace mpeg = "urn:mpeg:mpeg7:schema:2004";
declare namespace xsi = "http://www.w3.org/2001/XMLSchema-instance";

let $fVar := for $i in collection('mpeg7Content/knowledgeContent')//mpeg:FuzzyVariable
    where $i[@fuzzyVarID[ft:query(., "FUZZY_COLOR_ID")]] return $i

let $rSims := for $i in $fVar/mpeg:SimilarityRelations/mpeg:Similarity
    where $i[@fuzzyVal1REF[ft:query(., "redColorID")]]
    or $i[@fuzzyVal2REF[ft:query(., "redColorID")]] return $i

let $tSims := for $i in $rSims where $i >= 0.1 return $i

let $similars := for $i in $tSims return
    if($i[@fuzzyVal1REF[ft:query(., "redColorID")]])
    then concat(string($i/@fuzzyVal2REF), "||", $i/text())
    else if($i[@fuzzyVal2REF[ft:query(., "redColorID")]])
    then concat(string($i/@fuzzyVal1REF), "||", $i/text())
    else ()

return ($similars, "redColorID||1")
```

C.2 Sample Object Height Query

```
declare namespace mpeg = "urn:mpeg:mpeg7:schema:2004";
declare namespace xsi = "http://www.w3.org/2001/XMLSchema-instance";

let $fVar := for $i in collection('mpeg7Content/knowledgeContent')//mpeg:FuzzyVariable
    where $i[@fuzzyVarID[ft:query(., "FUZZY_HEIGHT_ID")]] return $i

let $rSims := for $i in $fVar/mpeg:SimilarityRelations/mpeg:Similarity
    where $i[@fuzzyVal1REF[ft:query(., "tallHeightID")]]
    or $i[@fuzzyVal2REF[ft:query(., "tallHeightID")]] return $i

let $tSims := for $i in $rSims where $i >= 0.2 return $i

let $similars := for $i in $tSims return
    if($i[@fuzzyVal1REF[ft:query(., "tallHeightID")]])
    then concat(string($i/@fuzzyVal2REF), "||", $i/text())
    else if($i[@fuzzyVal2REF[ft:query(., "tallHeightID")]])
```

```
    then concat(string($i/@fuzzyVal1REF), "||", $i/text())
    else ()

return ($similars, "tallHeightID||1")
```

C.3 Sample Object Query

```
declare namespace mpeg = "urn:mpeg:mpeg7:schema:2004";
declare namespace xsi = "http://www.w3.org/2001/XMLSchema-instance";

let $fuzzySemanticBase := for $i in collection('mpeg7Content/imageContent')//mpeg:FuzzySemanticBase
    where $i/@xsi:type="FuzzyObjectType" return $i
let $semanticBase := for $i in collection('mpeg7Content/imageContent')//mpeg:SemanticBase
    where $i/@xsi:type="ObjectType" return $i
let $fuzzyObjects := for $i in collection('mpeg7Content/imageContent')//mpeg:FuzzyObject
    return $i
let $regObjects := for $i in collection('mpeg7Content/imageContent')//mpeg:Object
    return $i
let $allObjects := ($fuzzySemanticBase, $semanticBase, $fuzzyObjects, $regObjects)

let $objectTypeFiltered1 := for $i in $allObjects
    where $i/mpeg:Definition//mpeg:Keyword[ft:query(., "KEY_SEDAN")]
    return $i
let $objectTypeFiltered2 := for $i in $allObjects/mpeg:FuzzyDefinition
    //mpeg:FuzzyTextualDistribution/mpeg:Val[ft:query(., "KEY_SEDAN")]
    return $i/../../../../..
let $objectTypeFiltered := ($objectTypeFiltered1, $objectTypeFiltered2)

let $colorFiltered0 := for $i in $objectTypeFiltered
    where $i/mpeg:FuzzyProperty//mpeg:FuzzyValue[@fuzzyValREF[ft:query(., "darkRedColorID")]]
    return $i
let $colorFiltered1 := for $i in $objectTypeFiltered
    where $i/mpeg:FuzzyProperty//mpeg:FuzzyValue[@fuzzyValREF[ft:query(., "lightRedColorID")]]
    return $i
let $colorFiltered2 := for $i in $objectTypeFiltered
    where $i/mpeg:FuzzyProperty//mpeg:FuzzyValue[@fuzzyValREF[ft:query(., "orangeColorID")]]
    return $i
let $colorFiltered3 := for $i in $objectTypeFiltered
    where $i/mpeg:FuzzyProperty//mpeg:FuzzyValue[@fuzzyValREF[ft:query(., "redColorID")]]
    return $i
let $colorFiltered := ($colorFiltered0,$colorFiltered1,$colorFiltered2,$colorFiltered3)

let $heightFiltered0 := for $i in $colorFiltered
    where $i/mpeg:FuzzyProperty//mpeg:FuzzyValue[@fuzzyValREF[ft:query(., "mediumHeightID")]]
    return $i
let $heightFiltered1 := for $i in $colorFiltered
    where $i/mpeg:FuzzyProperty//mpeg:FuzzyValue[@fuzzyValREF[ft:query(., "tallHeightID")]]
    return $i
let $heightFiltered := ($heightFiltered0,$heightFiltered1)

return $heightFiltered
```