

Mario Döller, Dipl.-Ing.

THE MPEG-7 MULTIMEDIA DATABASE SYSTEM  
(MPEG-7 MMDB)

**DISSERTATION**

zur Erlangung des akademischen Grades  
Doktor der Technischen Wissenschaften

Universität Klagenfurt  
Fakultät für Wirtschaftswissenschaften und Informatik

1. Begutachter: a.o.Univ.-Prof. Dipl.-Inf. Dr. Harald Kosch  
Institut: Institut für Informationstechnologie, Universität Klagenfurt
2. Begutachter: o.Univ.-Prof. Dipl.-Ing. Dr. László Böszörményi  
Institut: Institut für Informationstechnologie, Universität Klagenfurt

Oktober 2004

## Ehrenwörtliche Erklärung

Ich erkläre ehrenwörtlich, dass ich die vorliegende Schrift verfasst und die mit ihr unmittelbar verbundenen Arbeiten selbst durchgeführt habe. Die in der Schrift verwendete Literatur sowie das Ausmaß der mir im gesamten Arbeitsvorgang gewährten Unterstützung sind ausnahmslos angegeben. Die Schrift ist noch keiner anderen Prüfungsbehörde vorgelegt worden.

## Word of Honour

I honestly declare that the thesis at hand and all its directly accompanying work have been done by myself. Permission has been obtained for the use of any copyrighted material appearing in this thesis and all such use is clearly acknowledged. The thesis has not been presented to any other examination board.

Unterschrift:

---

Klagenfurt, 2. November 2004

# Contents

<b>List of Tables</b>	<b>vii</b>
<b>List of Figures</b>	<b>viii</b>
<b>I Related Work</b>	<b>1</b>
<b>1 Introduction</b>	<b>2</b>
1.1 Research Questions . . . . .	4
1.2 Outline . . . . .	5
1.3 Legend . . . . .	7
<b>2 Basic Concepts of Content-Based Retrieval</b>	<b>8</b>
2.1 CBR . . . . .	8
2.2 CBR Systems . . . . .	11
<b>3 Multimedia Metadata Standards</b>	<b>14</b>
3.1 Dublin Core . . . . .	15
3.2 SMPTE . . . . .	15
3.3 P/Meta . . . . .	15
3.4 MPEG-7 . . . . .	16
3.5 MPEG-21 . . . . .	21

<b>4</b>	<b>Topics in Multimedia Database Management</b>	<b>23</b>
4.1	Multimedia DB-Schema . . . . .	23
4.2	Multimedia Index Structures . . . . .	27
4.2.1	Definitions and Theory . . . . .	29
4.2.2	Index Structures for High-Dimensional Data . . . . .	32
4.3	Multimedia Query Optimization . . . . .	34
<b>5</b>	<b>Database Extensions to support Multimedia</b>	<b>39</b>
5.1	Query Extensions for Multimedia . . . . .	40
5.1.1	SQL/MM . . . . .	40
5.1.2	MOQL . . . . .	41
5.2	Oracle . . . . .	42
5.3	IBM DB2 . . . . .	46
5.4	IBM Informix . . . . .	50
5.5	Summarization . . . . .	52
<b>6</b>	<b>MPEG-7 Databases</b>	<b>53</b>
6.1	Classification of XML Databases . . . . .	54
6.1.1	Native XML Databases . . . . .	54
6.1.2	XML Database Extension . . . . .	56
6.2	Querying XML Databases . . . . .	59
6.3	Usability of XML Databases for MPEG-7 . . . . .	61
<b>7</b>	<b>Application Scenarios</b>	<b>64</b>
7.1	Image Retrieval . . . . .	64
7.2	Audio Recognition . . . . .	65
7.3	Video Browsing . . . . .	66
<b>II</b>	<b>MPEG-7 Multimedia DataBase (MPEG-7 MMDB)</b>	<b>69</b>
<b>8</b>	<b>Core System</b>	<b>72</b>

8.1	Oracle's Data Cartridge Technology . . . . .	73
8.1.1	Oracle's Extensibility Services . . . . .	74
8.2	DB-Schema . . . . .	77
8.2.1	Object-relational database features . . . . .	77
8.2.2	The XMLTYPE data type . . . . .	84
8.2.3	Guidelines for mapping MPEG-7 to an object-relational DB model . . . . .	86
8.2.4	Resulting DB-Schema . . . . .	94
8.3	Index Processing . . . . .	102
8.3.1	Multimedia Indexing Framework (MIF) . . . . .	102
8.3.2	GiST Framework . . . . .	106
8.3.3	Experimental Results . . . . .	108
8.4	Query Optimization . . . . .	112
8.4.1	Motivation and Definitions . . . . .	113
8.4.2	Related Work . . . . .	115
8.4.3	Approach . . . . .	117
8.4.4	Architecture . . . . .	122
8.4.5	Cluster Visualization Tool . . . . .	125
8.4.6	Evaluation . . . . .	125
8.4.7	Conclusion . . . . .	130
<b>9</b>	<b>Internal Libraries</b>	<b>132</b>
9.1	InitLib . . . . .	132
9.2	InsertLib . . . . .	133
9.3	DeleteLib . . . . .	134
9.4	QueryLib . . . . .	135
9.5	MDC data type . . . . .	139
<b>10</b>	<b>Application Libraries</b>	<b>143</b>
10.1	Blobworld Library . . . . .	143

10.2	Audio Library . . . . .	145
10.3	Video Library . . . . .	147
<b>11</b>	<b>Applications</b>	<b>149</b>
11.1	MPEG-7 Blobworld . . . . .	150
11.1.1	Introduction . . . . .	150
11.1.2	Architecture of the System . . . . .	151
11.1.3	MPEG-7 Blobworld Query . . . . .	154
11.1.4	Performance Evaluation . . . . .	155
11.2	Audio Recognition . . . . .	159
11.2.1	Introduction . . . . .	159
11.2.2	Theory of audio recognition . . . . .	160
11.2.3	Music Composition Recognition with the help of the MPEG-7 MMDB . . . . .	162
<b>12</b>	<b>Summarization and Future Research</b>	<b>165</b>
12.1	Summarization . . . . .	165
12.2	Future Research . . . . .	167
	<b>Bibliography</b>	<b>170</b>

# List of Tables

8.1	Query Plan Comparison of first query . . . . .	129
8.2	Query Plan Comparison of second query . . . . .	130
11.1	Performance results of original system . . . . .	158
11.2	Performance results of MPEG-7 Blobworld . . . . .	158

# List of Figures

1.1	Legend of various background colors of Figures . . . . .	7
2.1	Common CBR System Architecture . . . . .	12
3.1	MPEG-7 Description: StillRegionType . . . . .	19
3.2	Left side: Soccer Game Image, Right side: the corresponding Mpeg-7 document . . . . .	20
4.1	Videx video model . . . . .	26
4.2	Query Processing Steps . . . . .	34
4.3	Two 10-NN queries with query points located in + and × . . . . .	35
5.1	Oracle Spatial datatypes . . . . .	44
5.2	IBM DB2 Spatial datatypes . . . . .	47
6.1	Example DTD (left side), corresponding XML document (right side) .	59
7.1	MPEG-7 Multimedia DataBase (MPEG-7 MMDB) . . . . .	70
8.1	Oracle Data Cartridge . . . . .	73
8.2	Oracle Server Execution Environment . . . . .	76
8.3	XML constructs and their possible database counterpart . . . . .	86
8.4	MPEG-7 TextAnnotationType . . . . .	90
8.5	MPEG-7 MultimediaContentType . . . . .	91
8.6	MPEG-7 Instance document that uses the MultimediaContentType .	92



8.7	MPEG-7 ObjectType descriptor . . . . .	93
8.8	MPEG-7 Class Hierarchy . . . . .	94
8.9	Legend of DB Schema . . . . .	95
8.10	MPEG-7 DSType descriptor . . . . .	95
8.11	DB Schema - CreationInformation Part . . . . .	96
8.12	DB Schema - Image Part . . . . .	97
8.13	DB Schema - Audio Part . . . . .	98
8.14	DB Schema - Video Part . . . . .	99
8.15	DB Schema - AudioVisual Part . . . . .	100
8.16	DB Schema - Semantic Part . . . . .	101
8.17	GistService . . . . .	103
8.18	Response Time of the Insert Statements (50000 to 200000 points with 64 dimension (left) and 96 dimension (right)) . . . . .	110
8.19	Response Time of Select Statements of points with 64 dimension (left) and 96 dimension (right) . . . . .	111
8.20	Response Time of Inserts and Select Statements for color histograms (real dataset) with 64 dimension . . . . .	112
8.21	Optimal Query Plan (left) Inefficient Query Plan (right) . . . . .	115
8.22	Selectivity Results of the First Approach . . . . .	120
8.23	Internal Architecture of the MDCSelectivityOptimizer . . . . .	123
8.24	Tool for cluster visualization . . . . .	126
8.25	Approximated Selectivity Evaluation (synthetic data set) . . . . .	128
8.26	Approximated Selectivity Evaluation (city data set) . . . . .	129
8.27	Approximated Selectivity Evaluation (image data set) . . . . .	130
9.1	(a) MPEG-7 instance document describing audio data, (b) correspond- ing tree representation . . . . .	135
9.2	(a) Audio Query that returns an MPEG-7 description (b) Code frag- ment that produces same output as (a) . . . . .	138
9.3	Java Code fragment for demonstrating the MDC database type . . . . .	142

11.1	Multimedia applications based on MPEG-7 MMDB . . . . .	149
11.2	Architecture of MPEG-7 Blobworld . . . . .	152
11.3	Steps of MPEG-7 Blobworld Segmentation [1, 2] . . . . .	152
11.4	Good result of segmentation algorithm . . . . .	153
11.5	Bad result of segmentation algorithm . . . . .	154
11.6	KNN-retrieval for MPEG-7 Blobworld within MPEG-7 MMDB . . . . .	155
11.7	Original image and their blobed version . . . . .	156
11.8	Result of blob based image retrieval . . . . .	157
11.9	Performance results of (a) original system and (b) MPEG-7 Blobworld	158
11.10	Performance comparison of both systems . . . . .	159
11.11	Music composition described with MPEG-7 . . . . .	160
11.12	Music composition retrieval based on (a) interpreter and (b) audio signal	163

## Abstract

*Broadly used Database Management Systems (DBMS) propose multimedia extensions, like Oracle's interMedia. However, these extensions lack means for managing the requirements of multimedia data in terms of semantic querying, advanced indexing, content modelling and multimedia programming libraries.*

*In this context, this thesis presents a methodology for enhancing extensible Object-Relational Database Management Systems (ORDBMS) for multimedia data based on MPEG-7 and their usage in the field of multimedia applications such as audio recognition and image retrieval. The innovative parts are our metadata model for multimedia content relying on the XML-based MPEG-7 standard, a new approach for approximating the selectivity of Range queries, the enhancement of multimedia databases with multi-dimensional index structures, a multimedia querying system for MPEG-7, and their supporting internal and external application libraries.*

*The resulting system, extending Oracle 9i, is verified and demonstrated by the use of two new applications in the field of audio recognition and image retrieval, that both use the MPEG-7 standard for content retrieval.*

**Keywords:** Multimedia Databases, MPEG-7, Query Optimization, Index Structures, Image Retrieval, Audio Recognition.



**Part I**  
**Related Work**

In recent years, multimedia (a combination of text, graphics, images, video and audio) became one of the central key topics in computer science. The rapid technological progress in the area of multimedia production devices (e.g., digital cameras) and multimedia consumption devices (e.g., DVD players) leads to an increasing spreading of multimedia data. The Internet and here especially available file sharing tools advances the ubiquity of multimedia data.

One of the crucial factors for this phenomenon was and is the proceeding success in the research of new compression technologies and their standardization. Here we have to mention the MPEG-1/2/4 (AVC) standards [3, 4, 5] for video, JPEG2000 [6] for images or MP3 for audio files. These compression techniques extremely reduce the amount of data that has to be transferred and stored.

The negative aspect is the lack of sufficient retrieval and storage facilities for multimedia data. For instance, the retrieval functionality within a digital camera or a DVD player is limited to browsing. The search for similar images to a specific scenery (e.g., a sunrise and a mountain scenery) in someone's holiday image collection requires the browsing of all images. This, of course, is unpractical if the amount of stored images increases. In this scenario, a Content-Based Image Retrieval (CBIR) tool (such as Blobworld [7]) that operates on low-level features (e.g., color, texture, shape, etc.) can improve the retrieval efficiency. Nevertheless, these tools run across their limits if we extend the retrieval to specific events within images or we try to combine the retrieval of several sources (e.g., Give me all video shots where a certain music composition is played and a red car is shown). In addition, in many cases these

tools are restricted to a specific domain. For instance, the authors in [8] provide a soccer video retrieval system which ad hoc is not applicable for any other kind of sports.

Besides the retrieval of multimedia data, there is an essential need to think about efficient storage capabilities. The most obvious solution, which first comes in every mind, would be the use of Object-Relational Database Management Systems (ORDBMSs). Unfortunately, current ORDBMS are basically not suitable for managing multimedia data [9]. Therefore, most database vendors provide extenders that enable fundamental processing of multimedia data (e.g., Oracle *interMedia* [10] or IBM DB2's AIV [11]). For instance, Oracle *interMedia* provides image storage and CBIR functionality, more over, format conversion through the new data type, *ORDImage*. Basic CBIR functionality is realized by the ability to extract an image feature vector containing four different attributes (global color, local color, texture and shape) and in combination with the multidimensional index structure, *ORDImageIndex*. However, these systems are limited to basic CBIR functionality with the help of low-level features without the possibility of creating semantically rich queries. Furthermore, no means for video and audio CBR are given.

In fact, there is a need for a multimedia data model that supports all kind of data (e.g., image, audio, etc.) and takes attention on high- and low-level information. However, current multimedia data models, e.g., [12] and their implementations reveal shortcomings, mainly because of the limitation of the supported multimedia data or the expressiveness of the underlying meta-data model and the lack of supporting emerging multimedia standards.

In this context, the Moving Picture Experts Group (MPEG) introduced in 2002 a new standard, called MPEG-7 [13, 14], for describing high- and low-level content of multimedia data. There are many scenarios where a standard for describing multimedia data is useful. For instance, we are listening to a radio song and can not remember the title. Thus, one can use our audio recognition service based on MPEG-7 descriptions to identify the interpreter and the song title. Besides, describing high- and low-level multimedia content, the MPEG-7 standard also ensures interoperability among retrieval tools, databases and presentation devices. But interoperability is

only available if the interfaces between the components use a standardized format. Therefore, MPEG-7 is the choice for us in order to ensure interoperability between the database and their application. This implies the requirement for a multimedia database that on the one hand should be able to store and index MPEG-7 descriptions and on the other hand the query result should be provided as valid MPEG-7 descriptions.

Unfortunately, currently available data management systems (e.g. Tamino [15], imber [16]) for MPEG-7 descriptions do not provide adequate multimedia search functionalities (e.g., k-NN, Range searches). At the moment, the access and search within MPEG-7 descriptions is limited to simple path and element searches via, for instance XPATH expressions in a single document. A combined search in several descriptions is not supported. In addition, there is a need for a fine-grained database model that allows the direct access to each MPEG-7 element. Enhanced multimedia retrieval must support Nearest Neighbor and Range searches which is common in multimedia applications. In general, no existing solution supports necessary high-dimensional index structures for enabling multimedia retrieval.

Thus we raised the following questions:

## 1.1 Research Questions

- *How to map the MPEG-7 standard to an equivalent database schema?* This is one of the main questions and results on the emerging need for adequate storage possibilities of MPEG-7 descriptions. Until now, no approach is available that provides a sufficient solution in this area.
- *How to enable indexing of multi-dimensional data within an extensible database system?* As MPEG-7 describes among others low-level features of multimedia data represented as large feature vectors, we have to provide methods that support operations on them. Operations on low-level features (e.g., color histograms of images) include query types such as Nearest-Neighbor (NN) or Range-Searches not supported by basic DBMS. One way of enabling an efficient processing of such operations is the use of multi-dimensional index structures.



- *How to optimize multimedia queries?* Using index structures are an important method to increase the query processing efficiency. Nevertheless, these improvements are not optimal if the query optimizer does not have the necessary information for building an optimal query plan. One important aspect for the query optimizer in creating an optimal query plan is the selectivity of the operations. In this aspect, we provide an approximation of the selectivity for Range-Searches in multimedia queries.
- *How to define and implement a programmable interface for a multimedia database?* A multimedia database is only as effective as there exist programmable interfaces that provide functions for inserting, deleting and querying of the underlying data to client applications.

## 1.2 Outline

The remainder of this thesis is organized in two main parts. The first part, containing the Chapters 2 to 7, concentrates on topics in the area of multimedia data and multimedia databases such as Content-Based Retrieval, Multimedia Metadata Standards, MPEG-7 Databases, etc. The second part, containing the Chapters 8 to 12, describes all approaches that were necessary for the development of our MPEG-7 database. In the following, I will give a more detailed overview of the content of each Chapter.

Chapter 2 concentrates on Content-Based Retrieval (CBR) and the architecture of respective systems. This Chapter gives an brief overview of existing low- and high-level features, their extraction and their use in CBR systems. In addition, this Chapter reflects our claim that most CBR systems are not a MMDB.

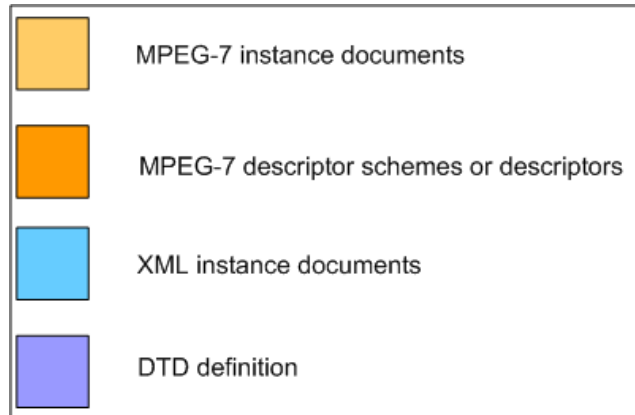
Chapter 3 explains currently available multimedia metadata standards. In detail, it addresses the Dublin Core, SMPTE, P/Meta, MPEG-7 and MPEG-21 standards. The MPEG-7 standard is covered in more detail, as it is the starting point for our research.

Chapter 4 describes main topics in Multimedia Database Management Systems (DBMS). These include multimedia database schema such as DISIMA or VIDEX, multimedia index structures and multimedia query optimization.

- Chapter 5 investigates related Database Extension technologies. In detail, it concentrates on Oracle's Data Cartridge technology, IBM DB2's Extender technology and the DataBlades from IBM Informix. In addition, this Chapter introduces query extensions for multimedia such as SQL/MM and MOQL.
- Chapter 6 takes care of MPEG-7 databases. It classifies current XML database solutions, investigates their storage behavior, the underlying database schema and their query possibilities for the use to manage MPEG-7 descriptions.
- Chapter 7 deals with multimedia applications. This Chapter especially considers applications in the area of image retrieval, audio recognition and video browsing.
- Chapter 8 gives an introduction to the practical part of this thesis.
- Chapter 9 introduces the core system of our MPEG-7 MultiMedia Database (MPEG-7 MMDB) development. This Chapter is divided in four main parts: The first one demonstrates the use of Oracle's Data Cartridge technology. The second part identifies necessary guidelines for mapping MPEG-7 to an object-relational DB model and presents the resulting database schema. The third part deals with index processing and describes our extension of Oracle's index facilities to support high-dimensional index structures which are very common in multimedia applications. The last part takes care of query optimization. In detail, our approach for approximating the selectivity of Range queries is introduced which supports the cost-based query optimizer in choosing the optimal query plan of multimedia queries containing Range-Searches.
- Chapter 10 demonstrates the implementation and usage of our internal libraries such as InsertLib, DeleteLib or QueryLib. In addition, the MDC data type is introduced.
- Chapter 11 explains with three examples all necessary steps how the MPEG-7 MMDB can be used for multimedia applications.
- Chapter 12 demonstrates the use of our MPEG-7 MMDB in combination with two CBR applications, namely an image retrieval tool and an audio recognition tool.
- Chapter 13 gives a short summarization and points out future research directions in the area of multimedia databases.

### 1.3 Legend

Through out this thesis the following background colors in the Figures are used to distinguish between the various documents (e.g., MPEG-7 instance document, ordinary XML instance document, etc.) (see Figure 1.1).







	MPEG-7 instance documents
	MPEG-7 descriptor schemes or descriptors
	XML instance documents
	DTD definition

Figure 1.1: Legend of various background colors of Figures

---

# 2 Basic Concepts of Content-Based Retrieval

## 2.1 CBR

Content Based Retrieval (CBR) can be described with the following sentence: *It is the way to retrieve content, based on what is in the content.* For instance, someone is interested in all images within his/her holiday image collection that contains an ocean. For this purpose, techniques are needed that are able to identify content (e.g., the ocean) within a multimedia source (e.g., image) which then can be compared with each other (e.g., similarity search within an MMDB). Retrieval of multimedia data started by using basically keyword matching. For every image a set of keywords have been associated that described the content (e.g., ocean, beach, sunset). The main disadvantage of this scenario is the fact that the content has to be annotated manually for every multimedia source which is impractical for large collections of multimedia data.

There was a need for an automatic classification of multimedia sources based on their corresponding content. In this context, the content itself can be categorized by low- and high-level features. In the following, we restrict ourself to image data to enable a more detailed explanation, nevertheless, the classifications can also be adopted for audiovisual and audio data.

Features, in general, are specific properties of an image. **Low-level features** (level 1 features) are primitive properties such as color, shape or texture. **High-level features** can be divided into two classes (level 2 and level 3) depending on their abstraction level. Level 2 features identify specific regions of interest that represent

a certain object. For instance, the recognition of persons or animals within an image can be defined as level 2 feature. Level 3 features specify the semantic context of participating objects or the representing scene. Let us assume the following scenario. An image shows a cheetah that hunts a zebra. The identification of the individual animals relies on level 2. Level 3 extracts the semantic of the image which is *a cheetah hunts a zebra*. Another scenario would be an image showing two persons (recognized through level 2) that are handshaking (recognized through level 3). The challenge of automatic feature extraction rises depending on the degree of the desired level. The extraction of low level features such as color or shape is a well investigated field of research and there exist a lot of extraction algorithm for that purpose. In the following, the more commonly used low-level features are described below:

**Color** One of the mostly used low level feature in the domain of color is the *color histogram* which shows the proportions of pixels of each color within the image (see [17, 18]). For retrieval, an user can either specify the desired proportion of each color, or use an example image whose color histogram is compared with the stored ones. The main disadvantage of a color histogram is its absence of positional information. Therefore, an improvement is the partition of the image into several regions whose color histograms are calculated or a combination with other color features such as the dominant color or the color layout which is the spatial distribution of the color.

**Shape** The retrieval and recognition of objects by shape is one of the most usual requirements at the primitive level (level 1). In certain applications, the shape representation has to be invariant to rotation, scaling and translation. In general, the available algorithms for shape representation can be classified by two categories: boundary-based and region-based. Boundary-based means that only the outer rectangular box boundary of the shape is used for representation, whereas a region-based algorithm uses the seal shape region. Examples of such shape representation algorithms are Fourier Descriptor and Moment Invariants (see [19] for further information).

**Texture** is used for the retrieval of similar backgrounds within images and to distinguish between areas of images with the same color (e.g., sky and sea). A

summarization of various techniques can be found in [20]. In general, the most commonly used representations for texture are *contrast*, *coarseness*, *directionality* and *regularity*, or *periodicity*, *directionality* and *randomness*. Gabor filters are also often used for texture representation.

Most of the available CBR approaches and tools (e.g., [21, 7, 22, 23, 24]) rely only on extracted low level features and therefore lack in supporting semantically meaningful queries such as: *Give me all images where a cheetah hunts another animal*. The reason for this restrictions is the so called *semantic gap*. In [25], Smeulders et al defined the semantic gap as follows: *The semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation*.

An automatic extraction of the semantic meaning of an image is still an open research issue. Several approaches exist that try to bridge the semantic gap such as [26, 27, 28], but they represent only a first step. In addition, researches are faced with diverse interpretation of an image based on different cultures and their specific characteristics. For instance, the characteristic of a church relies highly on the corresponding religion and on the age and geometrical region as well.

A short overview of several approaches trying to extract high-level features is presented in [29]. In [30], the authors presented a hybrid image retrieval system that extracts besides low level features such as color, texture and so on, semantic concepts from keywords. The drawback of this system is the fact that the authors assume that the images have textual annotations in terms of short phrases or keywords. In addition, the system provides a content learning approach that uses user feedback based on selected and weighted positive and negative samples to optimize the query result.

In several approaches, the extraction of semantic information is specialized for a certain category. For instance, Leonardi and Migliorati propose in [31] the semantic extraction of events in soccer games. The authors deployed in their approach a finite state machine which uses low-level motion indices for identifying events such as goals in an soccer video.

In addition, object recognition [32, 33, 34, 35] and detection is another challenging problem in visual information retrieval. In general, object recognition is realized as follows: An algorithm extracts a set of features (e.g., a color cooccurrence histogram [32]) that describes the desired object on the basis of a predefined test bed. The test bed contains images showing the desired object in various positions and angles in front of a black background.

## 2.2 CBR Systems

In general, the data flow of an CBR system can be described as follows (see Figure 2.1): The raw data source (e.g., an image or an video) is preprocessed by various algorithms for extracting low- and high-level features (e.g., color histogram or object recognition within images or shot and scene detection for videos and so on). An additional data manipulation process is accompanied within the feature extraction process. In this phase, the features are re-processed by, for instance, a dimensionality reduction algorithm for enabling an efficient retrieval. The resulting feature vectors are stored in an repository (e.g., a database) that provides efficient index structures (see Section 4.2.2) for similarity searches. For user interaction, CBR systems commonly provide frontend query portals, e.g., in form of web applications. This portals may include simple keyword search, QBE (query by example), QBS (query by sketch), QBH (query by humming for audio applications) etc. In case of QBE, the reference data source (e.g., an image or an video shot) is again preprocessed with the same feature extraction algorithm as the media in the repository. The resulting feature vectors serve as input data for the retrieval process of the similarity searches (see Section 4.2.1 for an introduction to similarity search). Then, the query result is presented to the user. Several systems, such as Blobworld [7], provide a relevance feedback functionality, which enables an user to improve their retrieval based on the found matches.

Samples of CBR systems in various domains (image, video and audio) are presented in Chapter 7.

In the following, we want to clarify the often misunderstood equivalence of CBR

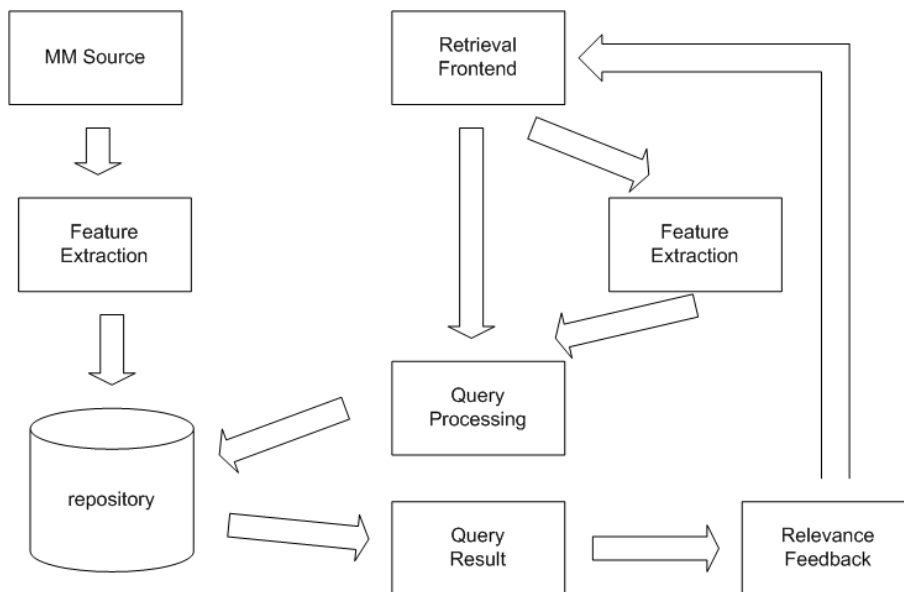


Figure 2.1: Common CBR System Architecture

systems and MMDB.

Sincerely, both systems intend to retrieve multimedia data by search facilities like QBE, QBH, etc.

At a closer inspection, the main focus of both systems differs. CBR systems mainly concentrate on the following parts: the processing of the raw data by the use of extraction algorithms, retrieval functionality by the use of index structures for realizing similarity searches and the presentation of the result to the user which includes additional features such as relevance feedback etc.

In contrast, research in the MMDB domain concentrates on the development of data models that efficiently represent the nature of multimedia data. Here, the data models are not only restricted to the content as it is the case in CBR systems. See Chapter 6 and Section 4.1 for additional information in this area. In addition, MMDBs establish query languages for retrieval (see Section 5.1) and have to take care of new approaches in the area of query optimization for multimedia data (see Section 4.3). Besides, additional topics such as the behavior of join operations between traditional data and multimedia data or the transaction handling for multimedia data have to be investigated precisely.

Therefore, one can recognize that CBR systems reside at the application layer



level that may or may not use an MMDB for realizing their retrieval functionality and for storing the necessary data. A CBR system can rely on a MMDB but on its own, it is not an MMDB yet.

---

# 3 Multimedia Metadata Standards

At the beginning, I would like to clarify the used terminology of this chapter. The common multimedia research field classifies multimedia data as a combination of image, audio, text and video data. In this way standards for multimedia data define the codec, coding and decoding approaches. The main goal of multimedia codecs is to employ several compression algorithms for identifying and removing redundancy. At the moment, there exist a lot of multimedia standards for various types of data such as the MPEG-1/2/4 [3, 4, 5] family or the H.261/H.263 [36] family for video data, MP3 (MPEG-1 Layer 3) <sup>1</sup> or MPEG 4 AAC <sup>2</sup> for audio and JPEG2000 [6] for images.

In contrast to multimedia standards, multimedia metadata standards are used for describing the content of multimedia data. This content can be described for instance with some basic information such as *author*, *title* or *creation information*. In addition, many standards also provide mechanism for describing low-level and high-level information.

In the following, an overview over some multimedia metadata standards for audiovisual data: (Dublin Core, SMPTE, P/Meta and MPEG-21) is given. The chapter is completed with a more detailed description of the MPEG-7 metadata standard.

---

<sup>1</sup><http://www.mp3-tech.org/>

<sup>2</sup><http://www.mpeg.org/MPEG/aac.html>

## 3.1 Dublin Core

Dublin Core [37] relies on the resource description framework (RDF) [38] standard and specifies an element set for multimedia metadata. The metadata elements fall into three groups which roughly indicates their main focus. The first group comprises elements for describing the content such as *title*, *subject* or *source*. The second group serves for protecting the intellectual property by offering elements such as *creator*, *publisher* or *rights*. The last group provides information about instantiation such as *date*, *format* or *language*. Dublin Core is currently used as a multimedia metadata standard in many television and bibliography archives<sup>3</sup>.

## 3.2 SMPTE

In this context, one also has to mention the Society of Motion Picture and Television Engineers (SMPTE) which is a multi-national organization for developing industry standards. This organization has created a Metadata dictionary for audiovisual data which is a collection of registered names and data types mainly developed for the television and video industry. Most metadata are media-specific attributes, such as timing information. Semantic annotation is in the current version not possible. The interested reader is referred to the SMPTE web site (<http://www.smpte.org>) which contains several documents for specifying the standard.

## 3.3 P/Meta

P/Meta is a universal standard for multimedia metadata exchange that bases on XML schema. It defines metadata for identification, description, discovery and use of multimedia content and is mainly pushed by the BBC. The standard has been established for a better understanding between EBU (European Broadcasting Union) members during media-related data interchange. The P/Meta Scheme defines the attributes and transaction sets that have been identified to support

---

<sup>3</sup><http://www.bsz-bw.de/diglib/medserv/>, <http://www.konbib.nl/>, <http://hosted.ukoln.ac.uk/biblink/>

the exchange. More information can be found at their official web site: [http://www.ebu.ch/departments/technical/pmc/pmc\\_meta.html](http://www.ebu.ch/departments/technical/pmc/pmc_meta.html)

## 3.4 MPEG-7

MPEG-7 [39, 13] is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group) and formally known as "Multimedia Content Description Interface". This organization committee also developed the successful standards known as MPEG-1 (1992), MPEG-2 (1994) and MPEG-4 (version 1 in 1998 and version 2 in 1999). Besides the previously mentioned standards, MPEG-7 is the first one that exclusively concentrates on describing multimedia content in a semantically rich manner. The standard is organized in eight parts (from system, low- and high-level multimedia description schemes, to reference software and conformance) and provides a rich set of standardized tools to describe multimedia content. It is important to notice that MPEG-7 documents are coded with the help of the XML standard following the rules of a Data Definition Language (DDL). The DDL provides the means for defining the structure, content and semantics of XML documents. In the following we shortly introduce each part.

- **MPEG-7 Systems**

The *MPEG-7 System* includes tools for preparing MPEG-7 Descriptions to allow an efficient transport and storage (BiM - binary format for MPEG-7). Further, there are mechanism for supporting synchronization between content and their descriptions. Additionally MPEG-7 System provides tools for managing and protecting intellectual property.

- **MPEG-7 Description Definition Language**

The *MPEG-7 Description Definition Language (DDL)* is one of the core parts that are used for instantiating MPEG-7 descriptions. The DDL provides a descriptive foundation for creating user defined description schemes and descriptions. A *Description (D)* is a representation of a feature (e.g., Color of a bird, or the bird itself) and defines its syntax and semantic. The *Description Scheme (DS)* specifies the structure and semantics of the relationship among

different description schemes and descriptions. Both, DS and D, can be defined and modified with the help of the DDL which directly bases on XML Schema. Additional to the basic features of XML Schema namely common data types, inheritance, abstract types, elements simple and complex types, MPEG-7 extends it mainly by the following: array and matrix data types, enumeration data types and typed references. A MPEG-7 document can be instantiated by the usage of the large set of predefined DS and D for describing visual, audio and audio-visual data which will be explained in the following points.

- **MPEG-7 Visual**

The *MPEG-7 Visual* description tools consists of basic description schemes and descriptors that mainly cover the following basic visual features: color, texture, shape, motion and localization. For describing an image, one has for instance the possibility to define the used color space, the dominant color, the shape of presented objects and so on. A detailed description of each specific DS or D is out of scope of this thesis but can be found in [40].

- **MPEG-7 Audio**

The *MPEG-7 Audio* description tools comprise basic description schemes and descriptions that cover basic audio features e.g.: sound effect description tools, melodic descriptors for query-by-humming and spoken content description which can be classified as high-level descriptors and a audio description framework for describing low level features (e.g., *AudioSignatureType* or *AudioWaveformType*). A detailed description is again out of scope of this thesis but can be found at [41].

- **MPEG-7 Multimedia Description Schemes**

The *MPEG-7 Multimedia Description Scheme* (for short MDS) standardizes on the one hand generic descriptors and descriptor schemes and on the other hand multimedia entities that represent more complex structures. Generic features are basically descriptors that are common to all media (audio, visual and text) for instance, vector, time, etc. Apart standardizing generic features, MDS provides more complex description tools such as Content Management, Content

Description, Navigation and Access, Content Organization and User Interaction. For more information see [42].

- **MPEG-7 Reference Software**

The *MPEG-7 Reference Software* is a platform for testing Descriptors (D), Descriptor Schemes (DS), the Description Definition Language (DDL) and the BiM Systems components. Besides the verification and implementation of the mentioned normative components, it also provides applications that show how one can extract the metadata from the media content or how the metadata can be used in simple applications. Interested readers are referred to [43].

- **MPEG-7 Conformance**

The *MPEG-7 Conformance* defines guidelines and approaches for testing descriptions and processing engines whether or not they are compliant with the MPEG-7 standard. See [44] for more information.

- **MPEG-7 Extraction and Use of Descriptions**

The *MPEG-7 Extraction and Use of Description* part provides information about using some of the description tools. It can be seen as an additional part to the *Reference Software* and gives insight into the software and offers some alternative approaches.

- **MPEG-7 Profiling**

The *MPEG-7 Profiling* [45] part provides the possibility to restrict descriptions of the MPEG-7 schema. Any description that is valid against the profile schema shall be valid against the MPEG-7 schema, while any description that is valid in the MPEG-7 schema may or may not be valid in the profile schema.

- **MPEG-7 Schema Definition**

The *MPEG-7 Schema Definition* [46] part contains means (e.g., description tools, namespace designator) for creating different versions of the MPEG-7 schema using the Description Definition Language (DDL).

For a better understanding, the following example should demonstrate the use of MPEG-7 by describing a soccer image (see Figure 3.2). Images are described in

MPEG-7 with the help of the *ImageType DS* and the *StillRegionType DS*. The *ImageType DS* contains subelements which allows the description of the semantics, the representation and the context of images. The *ImageType DS* and *StillRegionType DS* represent syntactically the same type with same elements, whereas for conceptual modelling the *ImageType DS* represents the root description of an image and the *StillRegionType DS* describes a subregion of the image.

```

<!-- ##### -->
<!-- Definition of StillRegion DS (11.4.2) -->
<!-- ##### -->

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

```

Figure 3.1: MPEG-7 Description: StillRegionType

Figure 3.1 shows the definition of the *StillRegionType* description scheme of the MPEG-7 standard. This DS offers a large number of different elements for describing images and subregions. For instance one has the possibility to use text annotation for describing the content of an image. Additionally, one can describe an image with

the help of some low-level descriptors (e.g. color, texture or shape).

In our example the soccer image is decomposed into two sub-images, each specified with a *StillRegion* DS. We specified for each sub-image an own id, e.g., KloseGoal, a *TextAnnotation* that contains text description of the image, and a *VisualDescription* which Figures a feature vector of size 64 that represents the color distribution of the subregion. The global information is described with an *Image DS* that contains in our example a *MediaLocator* for specifying the location and the name of the image and a *SpatialDecomposition DS* for keeping together all subregions. A possible MPEG-7 document that describes the scene demonstrated by the soccer image in Figure 3.2 can be seen on the right side of the same Figure.



```
<Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
xsi:schemaLocation="urn:mpeg:mpeg7:schema:2001 Mpeg7-2001.xsd">
  <Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="ImageType">
      <Image>
        <MediaLocator>
          <MediaUri> file://goal.jpg</MediaUri>
        </MediaLocator>
        <SpatialDecomposition gap="true" overlap="false">
          <StillRegion id="KloseGoal">
            <TextAnnotation>
              <FreeTextAnnotation>
                Klose shoots a goal with right foot.
              </FreeTextAnnotation>
            </TextAnnotation>
            <VisualDescriptor xsi:type="ScalableColorType"
              numOfCoeff="64" numOfBitplanesDiscarded="0">
              <Coeff> 2 4 6 8 10 ... </Coeff>
            </VisualDescriptor>
          </StillRegion>
          <StillRegion id="Spectators">
            <TextAnnotation>
              <FreeTextAnnotation>
                Spectators are crying
              </FreeTextAnnotation>
            </TextAnnotation>
            <VisualDescriptor xsi:type="ScalableColorType"
              numOfCoeff="64" numOfBitplanesDiscarded="0">
              <Coeff> 3 5 7 9 11 ... </Coeff>
            </VisualDescriptor>
          </StillRegion>
        </SpatialDecomposition>
      </Image>
    </MultimediaContent>
  </Description>
</Mpeg7>
```

Figure 3.2: Left side: Soccer Game Image, Right side: the corresponding Mpeg-7 document



## 3.5 MPEG-21

The MPEG-21 [47, 48] standard is the latest achievement of the Moving Picture Experts Group (MPEG). It defines an open multimedia framework. MPEG-21 is not only a *pure* multimedia metadata standard, but it also standardizes new video and audio codecs. The main idea of MPEG-21 is to cover the entire multimedia content delivery chain consisting of content creation, production, delivery, personalization, consumption, presentation and trade. The core element of MPEG-21 is the so called *Digital Item (DI)* which is based on XML Schema. A DI is a hierarchical container consisting of multimedia data, multimedia metadata and other digital items. The openness of the framework allows one to include other multimedia metadata standards within a DI. For instance one can use MPEG-7 or the Dublin Core standard for describing multimedia metadata in a DI.

In the following, the significance and possible contribution of MPEG-21 to our MMDB is considered. MPEG-7 is an exclusively content-based standard, but it does not include any new compression codec. In addition, it only contains rudimentary descriptors for property management and media adaptation. In contrast, MPEG-21 is a multi-functional standard, proposing content-based parts and codecs.

- First, in describing multimedia metadata, MPEG-21 acts as a container that can include different metadata standards (e.g., MPEG-7, Dublin Core, etc.). This means, that a management system for MPEG-21 needs to support the same openness in storing MPEG-21 descriptions. Such a system in fact is hard to realize. For instance, if we think of the underlying data model that has to support all available metadata standards, the query system, among others, needs to wrap all possible descriptors that have the same meaning. A restriction, for instance to MPEG-7, also restricts the interoperability between various MPEG-21 descriptions and their applications which do not go accordance with the original sense of MPEG-21.
- Second, the standard contains some content-based parts. Here, we have to mention the DIA (Digital Item Adaptation), REL (Rights Expression Language) and IPMP (Intellectual Property Management and Protection) all based on XML Schema. A complete list and their explanation is out of scope of this

thesis but can be found in [48]. The integration of several of the mentioned XML Schemas into our MMDB would be advantageous. Especially, the IPMP and REL are of interest which would lead to a multimedia access and control system within our MMDB. In addition, the integration of DIA could realize an internal adaptation engine for the stored MPEG-21 descriptions.

Two approaches for integrating the mentioned parts seem to be feasible:

- The methods for mapping MPEG-7 to a database schema could be analogically applied to these XML Schemas.
  - But, to build an access and control system, application logic for managing REL and IPMP expressions has to be considered. This task is ideally realized in internal libraries.
- Third, MPEG-21 defines new compression codecs for audio and video. These components are definitely out of scope for our MMDB.

---

# 4 Topics in Multimedia Database Management

This chapter summarizes research activities that concentrate on the main components of Multimedia Database Management Systems (MMDBMS). As for instance depicted in [9], multimedia data needs new approaches in storing, retrieval and optimization in contrast to alphanumerical data.

The rest of this chapter is organized as follows: Section 4.1 introduces existing database schemas for various kind of multimedia data. The theory behind multimedia query types, multimedia index structures etc. and several samples are addressed in Section 4.2. In Section 4.3, several aspects of multimedia query optimization such as cost models for index structures and approaches for efficient multimedia join operations are mentioned.

## 4.1 Multimedia DB-Schema

First approaches transferred meta-data of multimedia data into database relations in an ad-hoc way. These simple models support only certain types of queries and operations and were mostly limited to keyword retrieval. The nature of multimedia data, consisting of alphanumeric, graphical, image, video, and audio objects, differs in many ways to simple alphanumeric data that relational database management system are able to handle. One of the first models proposed for visual information that considers semantic queries, was called the *Visual Information Management System (VIMSYS)* [49] from Virage. The creators recognized within their model that image and video information is preferred to be retrieved by content rather than by keywords

or additional textual descriptions. During the past years, research concentrated on the development of data models for images (e.g., [50, 21, 51, 23]) and content-based retrieval (see Section 2.1). In most cases, these systems concentrate on the retrieval based on low-level features. Recently, data models for video and audio information that support high-level features have been introduced [8, 52, 12]. The data model of these systems are often specialized to one type of genre, e.g., retrieval on the basis of soccer games [8, 52].

In the following, various data models are introduced which main focus is on images (DISIMA [53]), videos (SMOOTH [12], ADMIRE [8]) and a generalized data model for any kind of multimedia data (MediaLand [54]).

**DISIMA** The DISIMA [53, 55] model (DIStributed Image database MANagement system) was developed at the University of Alberta and represents the content of images and spatial data. The model is composed of two main blocks: the image block and the salient object block. The image block is defined by the following quadruple:  $\langle i, R(i), C(i), D(i) \rangle$  where  $i$  is the unique image identifier,  $R(i)$  is a set of representations of the raw image data in a format such as GIF, JPEG, etc.,  $C(i)$  is the content of the image consisting of various logical salient objects (see below for an explanation), and  $D(i)$  is a textual description of the image. Further, the image block is divided into two layers: In the *image layer*, an user can define image type classifications such as *MedicalImage*, *NewsImage*, *PersonImage*, etc. The *image representation layer* provides two different models, the raster and the vector model.

The salient object block represents the content of images. DISIMA distinguishes two kinds of salient objects: physical and logical salient objects. A logical salient object contains specific information that belong to a certain physical salient object. For instance, a logical salient object may be an instance of type *Athlete* representing a famous sprinter. The sprinter is created and exists even if there is no image in the database at that time that contains the athlete. The logical salient object maintains the available information such as name, current records, previous clubs, etc. and represents the semantics for physical salient objects. The physical salient object points to a real occurrence of a logical salient object within an image and is represented by a minimum bounding rectangle (MBR).

**VIDEX** VIDEX [12], a generic object-oriented model for low- and high-level video indexing, was developed at the University of Klagenfurt. Figure 4.1 shows the VIDEX model in UML notation. In VIDEX, a video is physically structured by *PhysicalVideoUnits*. Such a physical video unit can be a *Shot*, a *Scene* or a *Sequence*. In addition, the model defines a *Frame*, which represents key frames of video shots and includes low-level spatial features. Furthermore, a frame can be divided in one or more *Regions*. A region is a spatial object and provides a set of topological relationships in combination with other spatial objects, such as overlap, covers, etc. In VIDEX, regions are represented by minimum bounding rectangles (MBRs) and can be subdivided into *BasicRegions* (region that does not consist of sub-regions) and *VisualObjects* (region that consists of various sub-regions). The temporal factor of regions and objects is modelled with the *MotionObject* which can consist of several *VisualObjects*. The *MotionObject* allows the expression of temporal, spatial and spatial-temporal relations between different low- and high-level objects.

The basic class for content annotation and high-level video indexing is the *ContentObject*. It is categorized in four elementary classes, namely *Person*, *Event*, *Location* and *Object* which define entry points for further specialization in specific application domains. A prototype implementation for soccer videos based on the VIDEX model was presented in [52].

**ADMIRE** The ADMIRE [8] model uses an object-oriented modelling technique in combination with a layered definition of information objects. The model is suitable for audiovisual information and has been applied within video retrieval for soccer videos. ADMIRE specifies information objects consisting of the following properties: *raw data*, *format* (e.g., MPEG-2), *attribute* (a characterization that can not be automatically extracted from the raw data, e.g., creation date), *feature* (e.g., color histogram), and *concept* (a semantic interpretation of various property types which is domain- and format-independent). Each information object may have several *relations* that refer to other information objects. The mentioned properties are modelled in a three-layer architecture. The lowest layer is the data layer and contains the raw data, format and attribute properties (e.g., the image stored as GIF). The feature layer is at the second level and contains information that can be extracted by the use

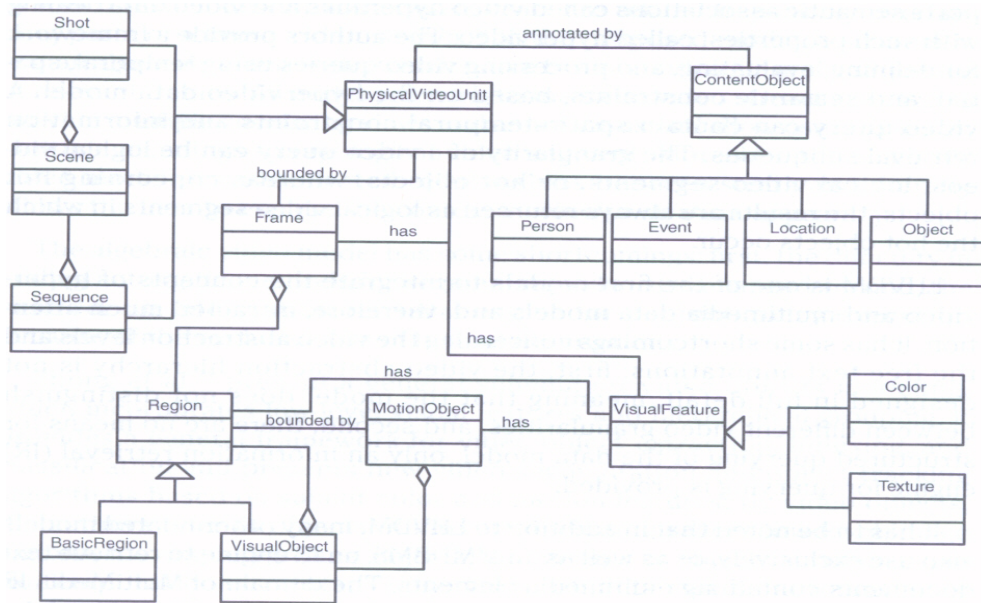


Figure 4.1: Videx video model

of the lowest layer and appropriate extraction algorithm (e.g., the color histogram). The concept layer is on the top level and uses the properties of the lower layers to determine the respective information. Concepts are context dependent and different concepts can be derived from the same lower layers. In the soccer environment, used concepts<sup>1</sup> are for instance, the ball, the goal events, etc.

The ADMIRE model distinguishes the following information object classes: *scene*, *shot*, *video shot*, *text title* and *frame*. A subpart within an information object is named *pseudo* (e.g., a pseudo video shot object). In the literature such *pseudo* objects are also mentioned as *salient* objects which refer to specific region of interest within an image, frame, etc. The identification of (pseudo) information objects is partly automated with the help of inference rules (see [8] for detailed information).

**MediaLand** The MediaLand [54] model, a development of Microsoft Asia, represents media objects by the following six-tuple:

$$MEDIA\_OBJECT = \langle OID, Type, DB\_Attribute, IR\_Feature, CBR\_Feature, Locator \rangle$$

<sup>1</sup>Note: MPEG-7 concepts have a broader meaning e.g., action concepts, acquisition concepts, etc. see [39, 13] for detailed information

The *OID* complies to the identifier of an object. The *Type* is the type of an object (e.g., audio, image, etc.). *DB\_Attribute* represents the structured data that describes the attributes of an object. Information retrieval (IR) features such as keywords that have been extracted are stored in *IR\_Feature*. Content-based features such as color or shape for content based retrieval are stored in *CBR\_Feature*. The *Locator* contains a pointer to the raw data of the media object. In addition, MediaLand provides *typed* and *weighted* links to model correlations among different media objects. The type information classifies different link semantics among objects. The weight information specifies the degree of correlation between the connected objects. The links are defined by the following six-tuple.

$$LINK = \langle LID, FromOID, ToOID, Type, Weight, Description \rangle$$

*LID* specifies the identifier of a link. The *FromOID* and *ToOID* represent the participating media objects and the *Description* is the textual data that denotes the link.

Both definitions are combined within a so called *multimedia object graph (MOG)*, which is a directed graph containing a set of media objects and a set of links. The described logical model (MEDIA\_OBJECT, LINK, MOG) is transferred to an equivalent conceptual model for the use in repositories (see [54] for a detailed description).

## 4.2 Multimedia Index Structures

In the last decades, research on multimedia index structures was very active. The main benefit of this improvements is gained by relative domains such as similarity search in CBR (see Section 2.1). Similarity search is essential in various application areas such as GIS (Geographic Information Systems) which calculates similarity among spatial objects, bioinformatics where the similarity among gene structures is computed or image retrieval where an user searches for similar images to a given one. Therefore, in the last decades, various different access methods have been established for indexing multidimensional data, for instance (to mention just a few):

LPC-File [56], X-tree [57], Pyramid Technique [58], R-tree [59] and its variants, V-tree [60], IQ approach [61], A-tree [62], SR-tree [63], SS-tree [64] and M-tree [65]. Excellent surveys and overview papers of index structures are given by Gaede et al. [66], Böhm et al. [67], Ahn et al. [68] and Guojun Lu [69]. An evaluation of different quantization techniques, such as A-tree vs. IQ-tree is given by Garcia-Arellano in his thesis [70].

In general, the mentioned access methods are tuned in order to increase performance of similarity searches in multidimensional data space. In [71, 72, 73], the authors especially improve query performance of NN-queries. The main drawback in similarity searches supported by the use of index structures are due to their high dimensionality. The feature vectors, e.g., the color histogram of an image, can vary in the amount of used features which directly correlates to a rise of dimensionality of the data space. This fact leads to the well known *curse of dimensionality* (see Subsection 4.2.1). An analysis and some performance studies for access methods supporting high-dimensional data is provided in [74, 75, 76].

A generalized framework, called GiST [76, 77], for index structures and a corresponding graphical tool set for the development of access methods [78] has been established by J. Hellerstein and his group at the University of California, Berkeley. The framework has been used for various image retrieval applications (e.g., Blobworld [7, 1]) and the development of new index structures [79]. In addition, the framework was integrated in the Informix database [80] and Oracle database [81, 82, 83]. Detailed information to our index extension for Oracle is given in Section 8.3. Whereas our implementation is specialized to spatial and multimedia feature vectors, the authors in [83] introduced a generic framework for indexing user-defined data types that combine spatial and thematic information. Based on the lack of additional technical information, it seems that their solution is restricted to Oracle databases whereas our solution can be used by any other application that connects to our shared library.

The remainder of this Section is the following. Subsection 4.2.1 introduces basic definitions in the area of distance metrics, different query types, the curse of dimensionality and how dimensionality can be reduced. In Subsection 4.2.2 a classification of existing access methods is given. Furthermore, the most common index structures for high-dimensional data are introduced in Section 4.2.2.



### 4.2.1 Definitions and Theory

The retrieval in multimedia databases depends in most cases on the comparison of various feature vectors. A feature vector represents e.g., the color histogram of an image and constitutes a point in the  $d$ -dimensional data space  $DS$  where  $d$  is addicted to the amount of different features (e.g., the amount of different colors in the color histogram). Therefore, a feature vector set  $FVS$  within a multimedia database is a set of  $n$  points (representing the feature vectors) in a  $d$ -dimensional data space  $DS$ , formally written as:

$$FVS = \{P_1, \dots, P_n\}$$

$$P_i \in DS, i = 1 \dots n, DS \subseteq \mathbb{R}^d$$

The degree of similarity between two points  $P$  and  $Q$  is measured by their distance in the data space. In this context, several metrics for defining distances between points in the data space have been established.

**Distances** The most popular and widely used metric is the Euclidean metric  $L_2$  defining the Euclidian distance function:  $\delta_{Euclid}(P, Q)$ . Other important metrics are the Manhattan metric  $L_1$  ( $\delta_{Manhattan}(P, Q)$ ) and the Maximum metric  $L_\infty$  ( $\delta_{Maximum}(P, Q)$ ).

$$\delta_{Euclid}(P, Q) = \sqrt{\sum_{i=0}^{d-1} (Q_i - P_i)^2} \quad \delta_{Manhattan}(P, Q) = \sum_{i=0}^{d-1} |Q_i - P_i|$$

$$\delta_{Max} = \max\{|Q_i - P_i|\}$$

The main difference between these metrics is the shape of space they span. For instance, the Euclidian metric represents a hypersphere, whereas the Manhattan metric defines a hypercube and the Maximum metric is represented through a rhomboid.

**Query Types** In multimedia databases, we are commonly interested in similar objects to a given one. The result of a similarity match depends on the used metric (see above) and the used query type. The simplest one is the point query which retrieves all points in the database with identical feature vectors. In all following

definitions  $Q$  represents the query point.

$$\text{PointQuery}(DB, Q) = \{P \in DB | P = Q\}$$

Another important query type is the *Range Query*. The *Range Query* returns all points  $P$  that have a smaller or equal distance  $r$  from the query point  $Q$  with respect to the used metric  $M$ . The size of the result set is hard to predict without a priori information for this query type, but the amount of visited index pages can be approximated.

$$\text{RangeQuery}(DB, Q, r, M) = \{P \in DB | \delta_M(P, Q) \leq r\}$$

The *Nearest Neighbor Query* (NNQ) is very important in content-based retrieval applications. In contrast to the *Range Query*, NNQ returns exactly one result, namely the most similar (with the lowest distance) to the query point  $Q$ . A relative to this type of query is the *k-Nearest Neighbor Query* (k-NNQ) which returns  $k$  nearest points. In contrast to the *Range Query*, the result set of the NNQ (k-NNQ) has a predictable size, whereas the amount of visited index pages can not be easily approximated.

$$\text{NNQ}(DB, Q, M) = \{P \in DB | \forall P' \in DB : \delta_M(P, Q) \leq \delta_M(P', Q)\}$$

$$\begin{aligned} k\text{-NNQ}(DB, Q, k, M) = \{P_1 \dots P_k \in DB | \neg \exists P' \in DB \setminus \{P_1 \dots P_k\} \\ \wedge \neg \exists i, 1 \leq i \leq k : \delta_M(P_i, Q) > \delta_M(P', Q)\} \end{aligned}$$

The mentioned query types represent only a subset of existing one's, but summarize the most common used types in CBR applications (see Chapter 2 and Chapter 11). A more detailed overview is given in [66, 67]. All these query types can be executed by a single scan over the database. Unfortunately, this results in a very low query performance due to the fact that every data block has to be loaded into main memory and for every tuple within a block the distance function has to be executed. Query performance can be massively increased by the use of access methods, as classified in Subsection 4.2.2. One problem in processing feature vectors is their dimensionality. The quality of results of a similarity search depends on a high degree on the underlying data set (feature vectors). For instance, the more colors a color histogram

represents the merrier the similarity search will work. This correlates with a higher dimensionality of the resulting feature vector. This is also the case, if we consider vectors as a combination of various features such as for images the color, the shape and the texture. Unfortunately, as stated first by Berchtold et. al. in [58] query performance of access methods decreases, if dimensionality of the underlying data set becomes high. This phenomenon is known as *curse of dimensionality* [58].

**Curse of Dimensionality** The term *curse of dimensionality* [66, 84, 85] is used for indicating that an access method performs worse when the dimensionality increases. The reason for this behavior is twofold: First, based on the increase of dimensionality the amount of feature vectors that fit in a single index page decreases. This leads to an overgrowth of the index tree. Second, the probability that different index regions overlap increases by growing dimensionality. Whenever the query region overlaps an index region, the search must branch into the corresponding child nodes. Then, every index region has to be visited. This fact decreases the query performance, so that in some cases it is outperformed by a simple sequential scan [74]. One possibility of avoiding this problem is the development of specific index structures that are specialized for high-dimensional data (e.g., [58]). Another possibility is the reduction of the vector dimension.

**Reduction of Dimensionality** The main goal of reducing the dimension is to find a low-dimensional representation of the vector that preserves (most of) the information of the original data. The reduction can be realized by two different approaches: *Feature selection* and *Feature extraction*. *Feature selection* stands for choosing a subset of all the features that represent most of the desired information. *Feature extraction* describes the creation of new features by combining the existing ones.

Wu et al. introduced in [86] a dimensionality reduction for image retrieval called *weighted multi-dimensional scaling*. Another commonly used technology is the singular value decomposition (SVD) [87].

### 4.2.2 Index Structures for High-Dimensional Data

The **X-tree** [57] is an extension of the  $R^*$ -tree [88] and is designed especially for the management of high-dimensional data. It extends the  $R^*$ -tree by two concepts avoiding overlapping in the directory: overlap-free split according to a split hierarchy and supernodes with enlarged page capacity. The X-tree consists of three different types of nodes: data nodes containing pointers to data objects, directory nodes containing pointers to sub-MBRs (minimum bounding rectangles) and supernodes which are large directory nodes of variable size. The main goal of supernodes is to avoid splits in the directory that would result in an inefficient directory structure. The modified split algorithm works as follows: Splits of data pages are performed with the original  $R^*$ -tree split algorithm. Splits of directory nodes are performed with a topological split algorithm. If this leads to overlapping MBRs, the X-tree applies the overlap-free split algorithm in combination with the split hierarchy. If the resulting directory structure is unbalanced, then the corresponding directory nodes are substituted by a supernode. The X-tree shows an improved performance behavior compared to the  $R^*$ -tree for all query types in medium-dimensional spaces.

The **SR-tree** [63] is a combination of the  $R^*$ -tree and the SS-tree [64]. The main difference is that the SR-tree uses the intersection solid between a rectangle (MBR represents the page region of the  $R^*$ -tree) and a sphere (MBS (minimum bounding sphere) represents the page region of the SS-tree) as page region. This approach is emerged by the fact that spheres are basically better suited for processing NN queries and Range queries based on the Euclidean metric ( $L_2$ ). The drawback of using spheres is their lower maintenance capability and their tendency in producing overlaps during split operations. The authors believe that a combination of both techniques ( $R^*$ -tree and the SS-tree) will overcome the disadvantages of spheres. In their performance evaluation, the SR-tree outperforms both access methods ( $R^*$ -tree and the SS-tree). Nevertheless, the authors evaluated that the SR-tree has a worse query performance than a sequential scan for a dimension higher than 32.

The **Pyramid-tree** [58] is an access method that maps an d-dimensional point into a 1-dimensional space which is then indexed by a  $B^+$  - tree. This mapping is called Pyramid-mapping and is optimized for Range queries on high-dimensional data. The basic idea is to divide the original data space into partitions that are

shaped like peels of an onion. In a first step, the  $d$ -dimensional space is divided into 2d pyramids having the center point of the space as top. In a second step, the single pyramids are cut into slices parallel to the basis of the pyramid. These slices form the data pages. The pyramid tree is not affected by the *curse of dimensionality*. The theoretical background for this phenomenon is given in [58]. Unfortunately, the pyramid technique is only tuned for hypercube shaped Range queries. For very skewed queries or queries specifying only one attribute, the technique is outperformed by a linear scan.

The **Independent Quantization (IQ)** [61] is an index compression technique for high-dimensional data. It combines index structure technology and compression of data for improving query performances of NN-Searches. The index comprises of three levels organized in three distinct files: The first level stores directory pages containing exact representations of MBRs. The second level contains data pages (*quantized data pages*) that store data points in compressed form. The third level stores the data point in their exact representation. The MBRs in the first level refer to a quantized data page and describe the MBR containing the points of the underlying levels. The quantized data pages are of fixed size whereas the data pages of the third level have a variable size. The data points in the third level have only be accessed during query operations if the evaluation of a query condition can not be accomplished by the corresponding quantized data page based on a too loosely compression. The authors managed to find a trade-off for the optimal degree of compression for the data points in the second level. On the one hand, too much compression will lead to many needless expensive accesses to the third level, whereas on the other hand, lower compression will increase the storage and the access cost for the first two levels. Therefore, the authors developed a cost model and an optimization algorithm for determining the optimal degree of compression for each quantized data page to minimize the query processing cost. Performance evaluation of the index structure showed, that the approach clearly outperforms the X-tree.

### 4.3 Multimedia Query Optimization

A general architecture of a query optimizer has commonly the following form [51] (see Figure 4.2): A query is forwarded to a *query parser* which checks the syntactical correctness and transforms the query into an internal representation (mostly an algebraic expression). In the following step, the *query optimizer* evaluates the most effective algebraic expressions which represents the given query and chooses the cheapest one. The *code generator* transforms the resulting query plan into calls to the *query processor* which does the actual query execution.

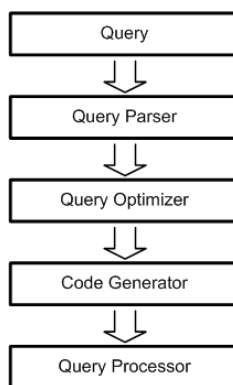


Figure 4.2: Query Processing Steps

The main task of a query optimizer is to examine all possible alternatives to a given query so that the best alternative can be chosen and the query cost is minimized [89, 90].

In multimedia applications and databases, queries often contain similarity operations such as Range or NN-operation for low-level features (e.g., color histogram), especially if the system supports content-based retrieval. This can be associated with a similarity-based selection operation in an multimedia database management system. A definition of content-based retrieval operations is given in Subsection 4.2.1. In general, a query optimizer should use 3 approaches: selectivity, cost model or operator ordering. In this thesis, we concentrate on the cost-model and the selectivity, based on the fact that modern database systems, such as Oracle, only provide means for their enhancement.

The cost of a selection operation is composed by two different cost factors. The

*selectivity* of an operation defines the amount of tuples returned by the selection operation. The *cost* of an operation is counted as the amount of data pages which has to be accessed for fulfilling the given task and as the number of necessary CPU cycles. In the multimedia environment and their applications, we are especially interested in the cost (selectivity, and amount of data pages) of Range- and k-NN-queries (see Subsection 4.2.1 for their definition) in high-dimensional spaces. The efficient processing of such kind of queries are guaranteed by multidimensional index structures (see Section 4.2.2). Therefore, the calculation of the amount of accessed pages (further simply called cost) needs an evaluation of the used index structure. This evaluation can be simple in one case, for instance the selectivity of a k-NN query is determined through k, or difficult, if we consider the selectivity for a Range query which needs an approximation (see Section 8.4). The cost for a Range query can be calculated straightforward based on the known radius which results in an intersection operation of the hypersphere and the MBRs of the index structure. The cost for a k-NN query depends largely on the density around the query location. For instance, Figure 4.3 shows two 10-NN queries with their query points located at + and  $\times$ , respectively. The query point (+) lies in a dense area, whereas the density around query point ( $\times$ ) is small. Therefore, the radius of both queries differ which leads to different costs concerning page accesses.

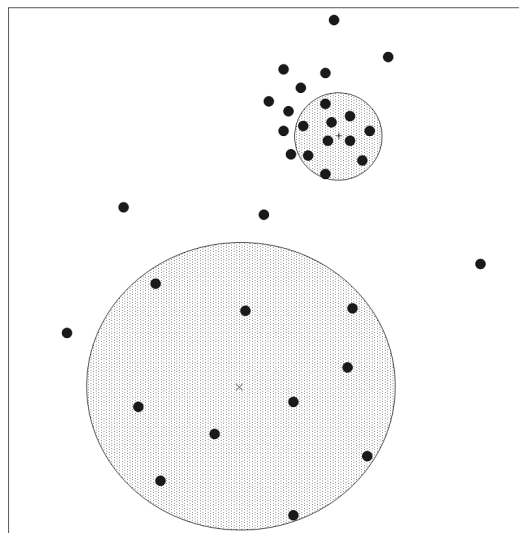


Figure 4.3: Two 10-NN queries with query points located in + and  $\times$

In the literature, several cost models exist that concentrate on calculating the amount of page access for Range- and k-NN-queries (see [91, 92, 93]). Selectivity estimation [94, 95] for Range queries is often limited to either the assumption of uniformity and independence of data sets or to the 2- or 3- dimensional data spaces in geographic information systems. The uniformity assumption does not hold for real data sets.

In [91], the authors present an efficient cost model for predicting the performance of the k-NN-query independently of the used index tree. The model is accurate for low- and mid-dimensional data with non-uniform distributions. For this purpose, the authors introduce two new concepts: the regional average volume and the density function. The regional average volume is used for calculating the average radius of the hyper-sphere which contains the k-NNs. As indicated in Figure 4.3, the average radius is not adequate for highly skewed non-uniform distributions. To overcome this limitation, the authors introduced a density function that estimates the regional radius depending on the location in the data space.

The amount of page accesses is determined by the number of index pages that intersect with the minimal hyper-sphere containing k-NNs, which leads to the following formula:

$$\sum_{i=1}^m \text{probability}(R \cap DR(s_i) \neq \emptyset)$$

where  $R$  is the minimal hypersphere and  $DR(s_i)$  is the directory region of index node  $s_i$ .  $m$  is the total number of nodes in the index structure. The probability that the query space intersects a directory region of an index node can be formulated as follows:

$$\text{Probability}(R \cap DR(s_i) \neq \emptyset) = \text{area}(DR(s_i)) + \text{perimeter}(DR(s_i)) * r + \text{area}(R)$$

In the following, the authors come up with the following generalized formula for n-dimensions:



$$DA(n) = \sum_{i=1}^m \sum_{j=0}^n \sum_{\{x_{i_1}, \dots, x_{i_j}\} \in \wp(\{x_1, \dots, x_n\})} \text{hypervolume}_{\{x_{i_1}, \dots, x_{i_j}\}}(DR(s_i)) * S_{n-j}(r)$$

where  $x_k$  is the  $k$ -th dimension of  $DR(s_i)$ ,  $\wp(A)$  is the power set of  $A$ . The hypervolume can be calculated with:

$$\text{hypervolume}_{\{x_{i_1}, \dots, x_{i_j}\}}(DR(s_i)) = \prod_{k \in \{i_1, \dots, i_j\}} x_k$$

The hypersphere with radius  $r$  can be calculated for the average region approach and the density function approach respectively with the following formulas:

average region approach	density function approach
$S_j(r) = \frac{1}{\Gamma(j/2+1)} (k\Omega\Gamma(n/2+1))^{\frac{j}{n}}$	$S_j(r_j) = \frac{1}{\Gamma(j/2+1)} \left( \frac{k\Gamma(n/2+1)}{D(s_i)} \right)^{\frac{j}{n}}$

A detailed explanation of the mathematical acronyms and the theoretical background is presented in [91], but is out of scope of this thesis.

Besides the selection operation, efficient join operations for multimedia data have to be investigated separately. A simple join ( $\bowtie$ ) operation is classified in [96] by:

$$R \bowtie_{condition} S = \{concat(r, s) | r \in R \wedge s \in S \wedge condition(r, s)\}$$

which results in a relation  $Q$  with  $n$  (attributes of  $R$ ) +  $m$  (attributes of  $S$ ) attributes that contains all combinations of tuples satisfying the join condition. A join condition can be one of the following:  $\{=, \leq, \geq, <, >, \neq\}$ .

Join operations for multimedia data have to consider similarity operations between relations containing high-dimensional data. For this purpose, similarity joins [90] have been introduced that uses index structures, such as R-tree for finding intersecting MBRs of both relations. After all leaf nodes have been identified a nested-loop join or a sort-merge join is applied for the contained data. An improvement of the classical technique is presented in [97], where the authors introduced an similarity join approach using their  $\epsilon$ -kDB tree. An approach for a multi-way (more than 2

participating relations within a join operation) spatial join using a R-tree has been introduced in [98]. The authors in [99] realized a *multimedia join* with the help of the NN-search method of an X-tree.

---

# 5 Database Extensions to support Multimedia

In the last decades, traditional (Object-) Relational Database Management Systems ((O)RDBMS) have been very effective and efficient in managing alphanumerical data. They basically offer services such as indexing, query optimization, buffer management, recovery or concurrency control, which are well investigated and optimized. Nowadays, based on the ubiquity of digital camcorders, MP3-players, digital cameras and mobile devices, the amount of multimedia data is overwhelming. In fact, these data forces one to think about repositories that offer extended functionalities in the area of e.g., retrieval, storage and security. In reality, everyone is lost by searching for a certain image in his/her holiday image collections. Unfortunately, data are useless if no efficient solution for retrieving and storing multimedia data exists. The most obvious solution, which first comes in every mind, would be the use of traditional ORDBMS for storing these kind of data.

Unfortunately, common ORDBMS and their techniques have several drawbacks in handling multimedia data [9, 100]. The most important drawback concerns the concept of matching. In traditional databases, the concept of matching relies on a filtering operation which decides for every tuple whether it fits the requirements or not. In multimedia data repositories, we basically are interested in *similar* data, like for instance, the search for beaches that are similar to another beach in our holiday image collections. Therefore, these repositories have to provide adequate query paradigms for similarity searches [17] and for ranking the results. In fact for enabling similarity searches, these systems have to provide multidimensional access methods [64, 63, 65]

and their appropriate algorithms for enabling Range queries or NN-Searches. In addition, query languages, such as SQL have to be extended for handling multimedia types and their appropriate operations to satisfy the requirements in retrieving multimedia data.

Beside, the enhancement of query languages, e.g., SLQ/MM [101] we may not neglect the performance of these operations [89]. In addition, one has to think about performance of query plans by investigating multimedia joins [99] and by enhancing cost based or rule based optimizers.

In contrast to database extensions such as Oracle's *interMedia* for managing multimedia data, MMDBs have been developed from scratch (e.g., Microsoft's MediaLand [54] or DISIMA [50, 102] which is an image database management system developed at the University of Alberta). MediaLand is a 4-tier database system providing support for all kind of multimedia and offers an integrated framework for users with different levels of experience. DISIMA is a pure image database that considers images as a set of salient objects (regions of interest). Salient objects are classified according to a user defined type hierarchy. Furthermore, DISIMA integrates a declarative query language (MOQL [103]) and a visual query language (VisualMOQL [104, 105]).

This chapter investigates the most popular commercial databases for their usage in the area of multimedia. First, Section 5.1 briefly describes two query extensions for multimedia, namely SQL/MM and MOQL. Second, Section 5.2 introduces *interMedia* which is an add-on for Oracle databases. Third, in Section 5.3, multimedia extenders, such as spatial or AIV of the IBM DB2 database are investigated and finally, Section 5.4 analyzes the Multimedia DataBlade technologies provided by IBM Informix database.

## 5.1 Query Extensions for Multimedia

### 5.1.1 SQL/MM

The SQL/MM [101] standard (MM for MultiMedia) is an extension of SQL for supporting any kind of multimedia data. SQL/MM is a multi-part standard and was developed by an ISO subcommittee, namely JTC1/SC32. The various parts of SQL/MM are independent from one another, however, Part I (framework) contains

definitions of common concepts that are used through out the other parts. The other parts are the following: *Full-Text* (Part II), *Spatial* (Part III), *Still Image* (Part V) and *Data Mining* (Part VI) respectively. The Part IV contains *General Purpose Methods* for complex numbers. A conceptual comparison between MPEG-7 and SQL/MM is given in [14]. In the following, the SQL/MM *Still Image* part is briefly described. The SQL/MM Still Image part provides structured user defined types that allow the storage of images within databases and specifies methods for modifying them in various ways and retrieving them efficiently. Images in SQL/MM are represented by the *SI\_StillImage* structured data type. The *SI\_StillImage* type can store images of several formats depending on what the underlying system supports. Furthermore, it extracts information about each image, such as format, height and width in pixels, color space, etc. In addition, the type provides several methods e.g., for scaling (e.g., change the size proportionally) , for rotating (e.g., changing the orientation from horizontal to vertical) or for creating thumbnails of the original image. Moreover, there exist various additional data types for describing several features of images. For instance, the *SI\_AverageColor* type represents the average color of a given image. The *SI\_ColorHistogram* type provides information about the distribution of colors within the image. The *SI\_PositionalColor* type represents the location of specific colors in an image. The SQL/MM Still Image part provides CBIR functionality by combining these types and methods with accurate index structures. For instance, the *SI\_PositionalColor* type supports queries like "Give me all images with a color representation of red or orange above dark blue". This query would lead to images that in most cases show sunsets at sea.

### 5.1.2 MOQL

The multimedia query language, MOQL [103] extends the ODMG's Object Query Language (OQL) [106] by adding spatial, temporal and presentation properties for content-based image and video data retrieval. OQL defines an orthogonal expression language where all defined operators can be composed with each other as long as the types of the operands are correct. OQL has the following basic statement for retrieving information:

```
select [distinct] projection_attributes
```

```
from query [[as] identifier] {, query [[as] identifier]}  
[ where query ]
```

A detailed explanation of the various concepts of OQL are out of scope of this thesis but can be found in [106]. Most of the extensions provided by MOQL concern the **where** clause by adding three new predicate expressions: *spatial-expression*, *temporal-expression* and *contains-predicate*. The *spacial-expression* is an extension for spatial data which supports various spatial objects such as points, lines etc., spatial functions such as area, intersection, etc., and spatial predicates such as cover, disjoint, etc. The *temporal-expression* enhances OQL for the use of temporal objects such as clip, frame, etc., temporal functions such as timeStamp, firstFrame, etc. and temporal predicates. The *contains-predicate* checks whether a salient object (interesting region) is covered by a particular media object (video or image) and has the following form:

```
contains_predicate ::= media_object contains salientObject
```

MOQL was successfully demonstrated within the content-based image retrieval system, DISIMA [50]. In addition, a visual image query system, namely Visual-MOQL [104, 105] which visualizes the image part of MOQL has been developed for the DISIMA [50] system (see Section 4.1).

## 5.2 Oracle

The extensibility services provided by Oracle is called *Data Cartridge* (see Section 8.1). Oracle databases are built as a modular architecture which provides several extensible services. The usual way for using Oracle's Extensibility Services is to implement a Data Cartridge that extends the extensibility interface (see [107]). Further, Oracle introduces the concept of an *indextype* for user-defined access methods. Each *indextype* has specific operators and uses an object that is responsible for the implementation. This object has to define all necessary functions which are provided by ODCI (Oracle Data Cartridge Interface) for indexing e.g., ODCIIndexCreate or ODCIIndexInsert.

Currently, several Data Cartridges have been developed by various vendors. Oracle itself provides among others, Cartridges for spatial data [108] and for multimedia

data [109, 10] such as image, audio and video. Cartridges of other vendors are for instance, the *WIISE Grid Data Cartridge (GDC)*<sup>1</sup> from Xmarc that provides an infrastructure to manage raster and regular spatial data in the form of grid data. The *Viisage Cartridge* from Virage<sup>2</sup> provides face recognition on behalf of images. The company provides several face recognition tools for controlling access to restricted areas e.g., Berlin Airport or identifying criminals in a crowd of people. In the area of chemistry, there exist, for instance the *Daylight Chemistry Cartridge*<sup>3</sup> and the *CambridgeSoft Oracle Cartridge*<sup>4</sup> that provide functionality for chemical structures and reaction data. The *Protein Family Database Cartridge (PFDB)*<sup>5</sup> manages data in the area of biology.

In the following, Oracle's Spatial and *interMedia* Data Cartridge are investigated.

**The Spatial Data Cartridge [108]** manages the storage, processing, searching and retrieval of spatial data of the following geometry types (see Figure 5.1). The Spatial Cartridge is optimized for two-dimensional data. Three- and four-dimensional geometry types are available, whereas spatial functions only work with the first two dimensions. The data model differs between three different representations. An *element* is the basic building block and includes points, line strings and polygons. The *geometry* represents a spatial feature and consists of an ordered set of *elements*. A *layer* is a collection of *geometries* that have the same attribute set. The retrieval relies on a two-tier query model. The *primary filter* enables fast selection by the usage of access methods and geometry approximations during comparison. This retrieval approach returns a superset of the exact filter method. The *secondary filter* computes, based on the output of the primary filter, exact results and therefore is computationally expensive. The retrieval engine provides for the primary filter two access methods: the R-tree (up to 4-dimensions) and the Quadtree. Furthermore, common retrieval functionalities and spatial operations such as overlap, intersect, Range-Search or NN-Search are allocated.

---

<sup>1</sup><http://www.xmarc.com/products/Core-sys.htm>

<sup>2</sup><http://www.viisage.com/>

<sup>3</sup><http://www.daylight.com/meetings/mug00/Delany/cartridge.html>

<sup>4</sup>[http://www.cambridgesoft.com/products/pdf/whitepapers/ocartridge\\_whitepaper.pdf](http://www.cambridgesoft.com/products/pdf/whitepapers/ocartridge_whitepaper.pdf)

<sup>5</sup><http://www.dcs.bbk.ac.uk/~nigel/bioinf/pfdb.html>

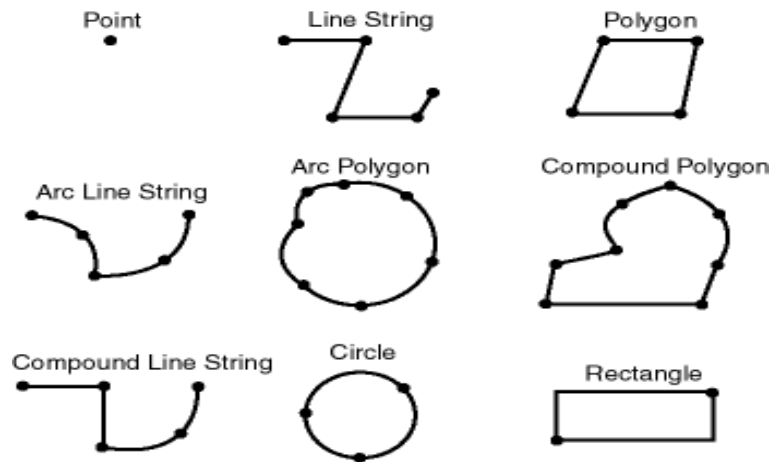


Figure 5.1: Oracle Spatial datatypes

The *interMedia Data Cartridge* enables Oracle to store, manage and retrieve audio, video, image and heterogeneous data. For this purpose, Oracle introduces the following new object types: *ORDAudio*, *ORDImage*, *ORDVideo*, *ORDDoc*, *ORDImageSignature* and *SI\_StillImage*<sup>6</sup>. These types can be used for extraction of metadata and attributes and performing manipulation operation on image data. All these types use the *ORDSource* object type for internally storing the multimedia content. Furthermore, the *ORDSource* object type can be used for extending *interMedia* by new data types and features. The bulk of the functionality of the mentioned types and their types themselves are realized as Java classes that are executed in the integrated JVM. Therefore, these types can easily be used in multimedia Java applications. In the following, each object type is briefly introduced.

- The **ORDAudio** type supports among others the following audio formats: *AIFF*, *AU*, *MPEG-1*, *MPEG-2*, *MPEG-4*, *WAV*, and *Real Audio*. Besides the extraction of basic attributes, such as encoding type, sampling rate, compression type or content length, no additional functionality is provided. Therefore, the main goal of this type is originated in storing and querying based on exact matches rather than in querying on a similarity basis (e.g., query by humming).
- The **ORDVideo** type provides support for the following movie formats: *AVI*, *QuickTime 3.0*, *MPEG-1*, *MPEG-2*, *MPEG-4* and *Real Video*. As we have

<sup>6</sup>for supporting the ISO/IEC SQL/MM StillImage standard [101]



analyzed for ORDAudio, the ORDVideo also offers basic attributes, such as frame rate, frame size, number of frames etc., which are interested for exact match queries but there exists no functionality for similarity queries.

- The **ORDImage** type supports a broad range of different formats that are not all mentioned here (e.g., *GIF*, *BMP* or *JPEG*). Instead of ORDAudio and ORDVideo, similarity search is well supported. Besides traditional information such as height, width or compression format of images, ORDImage offers several content-based retrieval concepts. These retrieval concepts are realized within Oracle's new object type called *ORDImageSignature* which is explained below.
- The **ORDImageSignature** is used to enable content-based retrieval of images. Oracle computes a feature vector (signature) of every image. The feature vector is extracted by segmenting the image into regions based on the respective color distribution. For each region, the *color*, *texture* and *shape* information is extracted. In addition, the feature *location* is computed for every extracted feature (color, shape and texture). The location specifies the layout of different features and allows a better distinction of subregions. For instance, two flags with the same color stripes but in different alignment can not be distinguished with a simple color histogram. The feature location allows us to investigate subregions of the two flags which therefore leads to a better differentiation. The MPEG-7 standard provides the *ColorLayout* descriptor for this purpose. In fact, similarity search can be more precisely in combination with the feature location and other features. Similarity search is realized by the following steps. First the user can specify the weight for each feature. A value of 0.0 means that the corresponding feature is completely irrelevant for retrieval and is ignored. During retrieval, the *score* for each feature between the reference image and the stored image is calculated. The score can range from 0.00 (no difference) up to 100.0 which specifies the maximum possible difference between two images in the respective feature. At the end, for every image a global score is calculated which indicates the overall distance between two images over all features in respect to their given weights. The result of a similarity query can be affected by a given threshold which prunes the result set based on the overall distance

of images. The performance of the similarity search is increased by the use of bitmap indexes [110].

- The **SI\_StillImage** realizes the *StillImage* object type that is specified in the ISO/IEC SQL/MM StillImage standard [101]. It provides mechanism for feature extraction, creation of thumbnails and common functionality such as identification of compression type, width or height. Content based retrieval is not supported for this kind of type, but it is realized through the not standardized *ORDImageSignature* type.

### 5.3 IBM DB2

The IBM DB2 database provides *Extenders*<sup>7</sup> for enhancing their database management system to meet new requirements, e.g., the storage and retrieval of multimedia data. DB2 Extenders generally specifies UDT's (user-defined types) for managing real world problems. For instance, the XMLCLOB type allows the handling of XML files. DB2 Extenders enable a database developer to enhance the server execution environment by user-defined functions and stored procedures implemented in PL/SQL, C(++) or Java. Furthermore, DB2 Extenders uses a specific SQL command for index extensions together with several search methods. The consistency during insertion, update and deletion has to be managed with the help of triggers. At present, there exist a large number of available extenders from different vendors. For instance, IBM itself provides the *AIV* extender for audio, image and video data, the *Spatial* extender for geo-spatial data and the *XML* extender for XML documents. Extenders of other vendors in this area are for instance, the *Spatial Data Extender* from Environmental Systems Research Institute<sup>8</sup> (ESRI) for GIS and geo-spatial data, the *SpatialWare Extender* from MapInfo<sup>9</sup> for spatial data and the *Formida Fire Extender* from Xmarc<sup>10</sup> for CAD and GIS data. A complete list can be obtained from <http://www.sqlsummit.com/extenven.htm>.

<sup>7</sup><http://www-306.ibm.com/software/data/db2/extenders/>

<sup>8</sup><http://www.esri.com>

<sup>9</sup><http://www.mapinfo.com>

<sup>10</sup><http://www.xmarc.com>

In the following, the IBM's extenders for multimedia data (AIV and Spatial) are evaluated.

**The DB2 Spatial Extender [111, 112]** provides the following key characteristics for supporting spatial processing. At first, Spatial Extender builds on the object-relational capabilities of DB2 for enhancing the type system by new user defined structured types (UDT) and user-defined functions (UDF) by the following spatial type hierarchy (see Figure 5.2). The shaded boxes represent abstract types which are not instantiable. Nevertheless, these types can be specified as column types for storing instances of their subclasses. The introduced types provide common spatial functions such as *ST\_Contains*, *ST\_Intersects* or *ST\_Overlaps* for supporting geographical applications and their database queries.

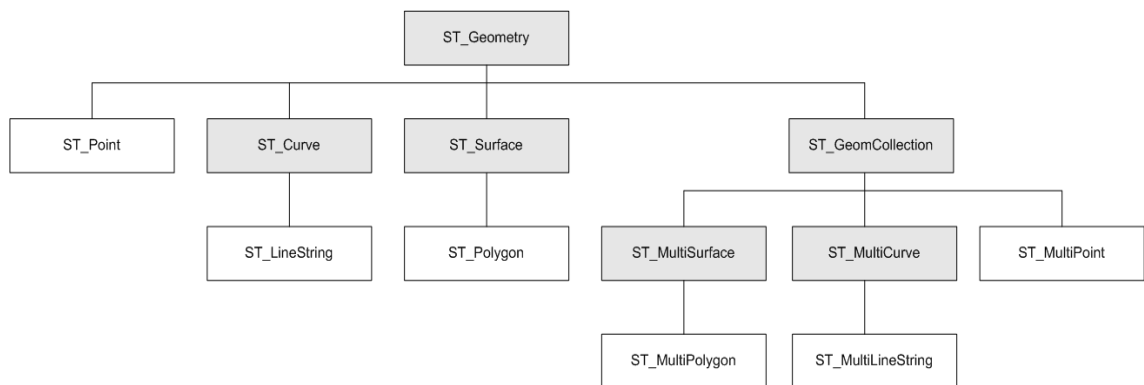


Figure 5.2: IBM DB2 Spatial datatypes

For instance, the following SQL-statement describes spatial information for states. For this purpose we could define the following table:

```

CREATE TABLE STATE (
  SNAME CHAR(30),
  ....
  TERRITORY ST_POLYGON)
  
```

In the following, we want to identify a certain state where we have received a GPS signal. For this purpose we could write an SQL query with the GPS-signal represented as 2-dimensional coordinates (43.5, 20.5):

```
SELECT SNAME FROM STATE WHERE ST_Intersects(TERRITORY, ST_POINT(43.5, 20.5))=1
```

Efficient access to spatial data requires support for 2-dimensional and 3-dimensional access structures in common ORDBMS. Therefore, the Spatial DB2 Extenders provides a *grid index* that is optimized for two-dimensional data (however, indexing of higher dimensions is possible). The grid index can be used for columns that hold instances of previously mentioned types (see Figure 5.2), such as *ST\_Point*. The grid index can be optimized by using up to three different grid levels to support different sizes of spatial data.

Furthermore, Spatial DB2 Extender supports facilities for bulk loading/storing of spatial data to/from disk. For this purpose, the Spatial Extender uses the well known *shapefile* format which is a de-facto standard for spatial data introduced by ESRI<sup>11</sup>.

**The DB2 AIV Extender** [11] provides mechanism for storing, maintaining and accessing image, audio and video data in their respective UDT's (DB2Image, DB2Audio and DB2Video) within the database. DB2 introduces the concept of *administrative, support tables*, also called *metadata tables*, for storing the multimedia data and their descriptive information such as e.g., width, height and number of colors for an image. The content of a multimedia object can be stored inside the database as a BLOB or can be kept in a file at the server's file system and is referenced by the metadata tables. Whenever an AIV column is added to a table, the respective information (e.g., an image or an audio file) is stored in the metadata tables and only a handler is stored in the table of origin.

The DB2 AIV Extender offers several services for multimedia processing. Thumbnails, which are a miniature versions of the respective data of origin, of images and videos can be generated, displayed and stored. Images can be scaled, rotated and converted into different formats. The feature of conversion is not available for audio or video data. In contrast to images, DB2Audio and DB2Video provides only rudimentary functionality.

Besides traditional queries of e.g., find all images with the given resolution, DB2Image also provides query by content. For this purpose, DB2 AIV Extender

---

<sup>11</sup><http://www.esri.com>

integrates the well known QBIC [113] system by the following means. It introduces the *QBIC catalog* which is a set of files that contains extracted visual features of images. A QBIC catalog can store the following features:

- The **Average Color** can be used to retrieve images that have a predominant color. If an image has a predominant color, the average color will be similar to the predominant one. The average color is calculated as the sum of the color values for all pixels divided by the number of pixels.
- The **Histogram Color** represented as a spectrum of 64 colors containing the distribution of colors in an image.
- The **Positional Color** can be used to search for predominant colors in a specific area of an image.
- The **Texture** can be used to search for certain images that have a particular pattern.

Based on the QBIC catalog, the DB2 AVI Extender allows one to search for similar images to a reference one. The similarity of images relies on the score that is calculated on the given query features. The result of the similarity search is ordered by the score which means that the image with the highest score corresponds to the NN of the reference image in respect to the given features.

Further, DB2Video provides the possibility to search for specific shots or frames by their distinct frame or shot number. Besides, the system enables storyboard functionality for browsing key frames of various shots. The Video Extender can detect scene changes in a video by calculating the differences between successive frames. The frames between scene changes are defined as shots. Shots are stored in so called *shot catalogs* that contain information such as starting and ending frame number, representative frame number etc. The content of the representative frame is stored in a metadata table. The Video Extender uses the *video index* to search for specific shots or frames in the shot catalog.

## 5.4 IBM Informix

The IBM Informix database management system deploys the feature of *DataBlade Modules* [114] to extend their functionality for new requirements. Beside SQL native types, Informix supports three different user defined data types. A *row type* encapsulates a grouping of multiple columns into the definition of a new data type. The *distinct type* is a customized version of an existing data type. *Opaque types* are defined by writing C, C++, or Java code to store, index and operate on that data type. DataBlades consists of user defined database objects, SQL statements and supporting code written in an external language. Further, DataBlade provides a *Virtual Index Interface (VII)* that allows a developer to create new indexes by implementing their corresponding methods (e.g., `am.create`, `am.insert`). Informix provides three different tools for developing DataBlade modules, namely *BladeSmith* for creating DataBlade modules, *BladePack* for packaging them and *BladeManager* for making them available in the database. At present, there exists a large number of DataBlade modules such as *Spatial DataBlade module* [115] for managing spatial data inside the database, *Excalibur Image DataBlade module* [116] for storing and querying images and DataBlades for MPEG-1 and VMRL developed through a spin-off company of GMD<sup>12</sup> (nowadays part of the Fraunhofer organisation). A DataBlade example, that manages spatial data, can be found at <http://geode.usgs.gov/>.

In the following, several DataBlade modules for multimedia data (images, spatial) are introduced. To the author's best knowledge, there exist no DataBlade module for managing audio or video data.

**The Spatial DataBlade Module** integrates a Geographic Information System (GIS) into the Informix database. Based on the fact, that the Informix database belongs to the same company as the DB2 database, the Spatial DataBlade Module provides very similar approaches. For example the data type hierarchy (see Figure 5.2) is the same for Informix and DB2. In addition, the same import and export file format (*shapefile*) from ESRI is used. Besides the file format, the Spatial DataBlade integrates the ArcSDE service of ESRI for converting spatial column data into ESRI shape representation for the use in their GIS applications: ArcView, AcrInfo

---

<sup>12</sup><http://www.globit.com/>

and MapObjects. Instead of grid access methods, as it is used in DB2, the Spatial DataBlade provides R-Tree indexes for increasing performance of spatial queries. In order to use the Spatial DataBlade module, one has to set up *spatial\_references tables*. As the Spatial DataBlade module is specialized for GIS applications, the *spatial\_references tables* store data about each map projection to translate the data into a flat X,Y-coordinate system.

**The Excalibur Image DataBlade Module** [116] provides storage and retrieval functionalities for images. The module consists of I/O and manipulation routines for read/write operations, scaling operations and type/format conversions. Images can be stored either inside the database by maintaining their native type (e.g., GIF) or can be kept at the server's file system and the database stores only a reference to them. Besides I/O routines, the module enables one to extract six different feature vectors for each image and stores them, combined to one single vector, in a separate table. An overview of the the different features is given below, whereas detailed information is not available.

- The **Color Content** is used for measuring the appearance of different colors and their appearance percentage of the image. The position of the colors is unaccounted.
- The **Shape Content** represents the relative orientation, curvature and contrast of lines in an image. As far the color content, the position and absolute orientation of the lines are not taken into account.
- The **Texture Content** is used for measuring the flow and roughness of an image.
- The **Brightness Structure** is a measure of the brightness at each point in the image.
- The **Color Structure** measures the hue, saturation and brightness of every pixel in the image.
- The **Aspect Ratio** measures the ratio of the width to the height in the image.

The Excalibur module allows one to perform content based searches on collection of images which features have been extracted. The comparison between images, in fact comparing their respective feature vectors, is realized with the *Resembles()* function that calculates a resemblance score over all six combined features between two images. In addition, the method assigns a relative weight for every feature. The method returns a boolean value indicating whether the resemblance score exceeds a user defined threshold. In the following, a short example is shown:

```
SELECT g.img_id, rank
FROM gallery g, gallery s
WHERE Resembles(g.fv, s.fv,
               0.0, 1, 1, 1, 0, 0, 0, rank #REAL) AND s.img_id = 3 ORDER BY rank;
```

In this example, for the sample image of gallery s with id 3 we are searching the most similar images of gallery g. The resulting set of images are ordered by the calculated rank. The weights for the various feature vectors vary between 0 and 1.

## 5.5 Summarization

This chapter introduced extensions for multimedia data on the basis of query languages (e.g., SQL/MM) and at the level of DBMS (e.g., Oracle's *interMedia* Data Cartridge). The main restriction which is common to all solutions is their specialization to one specific domain (in most cases spatial or image data, all other kind of multimedia data (e.g., audio or video) are only supported rudimentary). The retrieval and feature extraction within the image domain is limited to low level features such as color, texture and shape. High level features have not been taken into consideration in any introduced extension environment.

The mentioned multimedia database extensions rely on similar methodologies for enhancing modern database systems. In the case of Oracle, the environment is labelled Data Cartridge technology and provides means for enhancing the internal type system, indexing and query optimization facilities (see Section 8.1). In this context, our MPEG-7 MMDB is also a multimedia extension that relies on the MPEG-7 standard and uses Oracle's Data Cartridge technology. As all other database systems provide similar technologies (e.g., IBM's DataBlate) the MPEG-7 MMDB can easily be adopted.



The ISO-IEC Multimedia Content Description Interface (MPEG-7) [39, 117, 13] (see Section 3.4 for a detailed explanation) is a rich standard for describing high- and low-level features of multimedia data. Because of its importance in multimedia search, there is an emerging need for developing a management system that is able to store and retrieve MPEG-7 descriptions. Basically, MPEG-7 descriptions are XML [118] documents that rely on a language extension of XML Schema [119], called DDL.

Thus, the search for an efficient repository for MPEG-7 descriptions automatically leads us to the research field of XML databases. Two major research directions exist in this area. First, researches concentrate on finding the most suitable platform for storing XML. Here, research mainly focuses on native XML solutions [120, 121, 122] that develop XML storage solutions from scratch and database extensions such as [123, 124, 125, 126] that use the extensibility services of modern databases. Second, research concentrates on how to map DTD [127, 128] respectively XML Schema [125, 129] to an equivalent relational [127, 128, 130, 131, 132, 133], object-relational [125, 129] or object-oriented [134] database model. In this context, Schmidt et al. [135] identified 10 major points a benchmark system for XML databases has to address. Yao et al. [136] provide a benchmark family for XML database solutions that covers a large number of XML database applications. Another main question in this area concerns the retrieval of XML documents within database solutions. Some work has been done in [137] dealing with the transformation from relational tables to XML documents. Besides the transformation of database tables into XML, there exist several query languages for XML such as XML-QL [138], XQuery [139], XPATH [140],

SQL/XML [141] or XQL [142].

The remainder of this chapter is organized as follows: Section 6.1 classifies current XML repository solutions and their possible mapping strategies (see Subsection 6.1.2.1). In Section 6.2 several query languages are introduced. The usability of XML databases for MPEG-7 is investigated in Section 6.3

## 6.1 Classification of XML Databases

As mentioned earlier XML databases basically can be classified by two main groups: *native* XML databases and XML database extensions. Database extensions can be subclassified by the supported XML model (DTD or XML Schema) and the used database schema representation: relational, object-relational or object-oriented database schema. In the following, the main advantages respectively disadvantages of both groups are embossed. Furthermore, several mapping strategies for relational, object-relational and object-oriented database schemas are introduced.

### 6.1.1 Native XML Databases

The search for a clear definition about *native* XML databases leads us to a variety of different views in the literature. McGoveran [143] classifies a *native* XML database as a repository that stores XML documents in its entirety and provides means for manipulation including searching, inserting, updating, deleting, etc. Another abstract view is given in [144] that describes *native* XML databases as solutions that specifically have been developed for the management of semi-structured data. The *XML:DB Initiative*<sup>1</sup> specifies a *native* XML database as a repository containing a (logical) model for XML documents. Examples of such models are the XPATH data model or the models implied by DOM. Further, the repository stores and retrieves XML documents according the given data model. The underlying physical storage model is not explicitly described and ranges from proprietary storage solutions (compressed indexed files) up to relational or object-oriented databases. This is important as it is also stated in [145] that *native* does not imply that a native database solution

---

<sup>1</sup><http://www.xmldb.org>

has to be developed from ground up. A native solution might very well be based on conventional database technology as long as the data model of the underlying system is entirely hidden.

By summing up these definitions we come to the following least common denominator: A *native* XML database only provides a data model and manipulation facilities for the means of XML documents. The technology of the underlying system has to be hidden entirely.

Based on this definition, two native XML databases briefly will be introduced. A complete list of currently available *native* XML databases and additional information over XML and databases is given at: <http://www.rpbouret.com/>.

Tamino [15] is one of the leading commercial<sup>2</sup> native XML database producer and their acronym stands for **T**ransaction **A**rchitecture for the **M**anagement of **I**Nternet **O**bjects. Basically, Tamino relies on the following three main concepts. Besides the fact that Tamino is a native XML database, its main target group is E-Commerce and the Internet. Therefore, the corresponding data is stored next to the web, which means that Tamino can be used as module for existing webserver. The third main concept concerns their integrated gateway functionality to DBMSs where the origin of requested data is hidden to the user.

The central component of Tamino is the *X-Machine* which is responsible for storage and querying of XML documents. The XML documents are stored in the *XML-Store* repository. Further, Tamino consists of a knowledge base, *Data Map*, which contains all DTDs, XML-metadata, style-sheets, etc. that are used by the X-Machine for correctly storing and querying XML documents. Queries in Tamino are URLs and base on XQuery. The URL consists of the webserver name, the Tamino-server name, the database and collection name and the XQuery command. For instance the following query returns all students that are born between 1976 and 1978:

```
http://localhost/tamino/myDB/myCollection/?_xql=/student/birth[yearbetween'1976','1978']
```

Furthermore, Tamino provides three different indexing approaches: value-based indexing, text-indexing and structural indexing. Further information can be obtained from [15] and is out of scope of this thesis.

Timber [16, 146, 147] is designed on top of the SHORE data manager, which is

---

<sup>2</sup><http://www2.softwareag.com/>

responsible for disk memory management, buffering and concurrency control. XML documents are stored as objects within the SHORE data model which means that during parsing of the XML document each node is transformed incrementally into an internal representation and stored as an atomic unit. Furthermore, Timber provides two main types of indexes: path index and value index. With the help of path indexes, the performance of XPATH expressions is improved. Value indexes are build on the value of every element of the inserted XML document. Timber has indexes on each element tag. Query processing is realized by XQuery statements which are parsed and transformed into a TAX [148] algebraic operator tree. This tree is evaluated by the corresponding query parser.

### 6.1.2 XML Database Extension

The main difference in contrast to native XML databases is the use of traditional database technology for the storage of XML documents. Based on the extensibility functionality of modern database systems (see Section 8.1 for a sample database system) the storage can be realized by three possible approaches [145]:

- **The unstructured storage** approach is the most simplest way of storing XML documents within a database. The whole document is entirely stored in its textual representation in a character large object (CLOB). For this purpose, modern database systems provide a new data type and additional functionality for storing, searching and publishing XML documents. For instance, Oracle introduced the *XMLTYPE* for handling XML data.
- **The structured storage** approach extracts a fine-grained meta-model for representing the structure (edges of the XML graph) and content of XML documents in a relational DBMS. Florescu et al. [130] gives an overview of different approaches in this area.
- **The mapping** approach uses DTD or XML Schema information for creating an equivalent database schema. There exist different approaches for relational [127, 128], object-relational [125, 129] and object-oriented [134] database schemes. Recently, there has been research done for an *automatic* mapping [149, 150] of XML documents to a relational database schema.

The various approaches differ by their applicability for multimedia retrieval (e.g., similarity search). The unstructured and structured storage approach only provides limited retrieval functionality among various documents and shows poor scalability. Therefore, this thesis used the mapping approach for transferring the MPEG-7 standard to an equivalent object-relational database schema. In the following Subsection, mapping strategies are investigated in detail.

### 6.1.2.1 Mapping Strategies

This Subsection briefly summaries most common mapping strategies for DTD and XML Schema data model to equivalent relational, object-relational and object-oriented database models. An overview of existing XML storage techniques can be found in [151, 152, 153].

A first approach in this direction was originated by Shanmugasundaram et. al. [149] by implementing three mapping algorithms: *Basic*, *Shared* and *Hybrid*. These algorithms describe possible ways of mapping DTD to a relational database schema. The *Basic* algorithm creates a relation for each element in the DTD graph (see [149] for an definition of DTD graph). Dependencies among relations are modelled with foreign keys. The main disadvantage is the large amount of redundant relations which leads to a poor query performance. This drawback has been overcome by the *Shared* algorithm. The main idea is to identify nodes of the DTD graph that are shared and therefore, the *Basic* algorithm would create multiple relations. These nodes (which have an in-degree value greater than one) are stored in separate relations and accessed with the help of a key and foreign key system. Further, the *Shared* algorithm reduces the amount of relations by inlining nodes of the DTD graph that have an in-degree value equal to one. The main problem of this algorithm is the amount of join operations that are necessary for restoring different elements, which can be worse than in the *Basic* algorithm. The third algorithm, *Hybrid*, tries to combine the advantages of the previous algorithms, namely the join reduction feature of *Basic* and the relation reduction of *Shared*. The *Hybrid* algorithm manages inlining of elements that have an in-degree value greater than one but are neither recursive nor reachable through a "\*" node (e.g., the DTD element  $\langle !ELEMENT(e_1(e_2^*)) \rangle$  has the following graphical representation:  $e_1 \rightarrow * \rightarrow e_2$ ).

An improvement of the *Hybrid* algorithm was introduced by Lee and Chu in [127]. Their CPI (*constraints-preserving inlining*) algorithm is a method for mapping a DTD to a relational schema. The main difference to the previous algorithms which only capture the structure of a DTD, is their approach to include semantic constraints of the DTD as well. The algorithm requires several preconditions. First, the input DTD has been already simplified using the techniques introduced in [149] such as flattening (to convert a nested definition into a flat representation)  $(e_1, e_2)^* \rightarrow e_1^*, e_2^*$  or simplification (to reduce multiple unary operators into a single unary operator)  $e_1^{**} \rightarrow e_1^*$ . Second, the input XML document is valid respectively complies to the given DTD and third, the input DTD does not contain the following XML features *entities* and *notations*. Based on these preconditions, the CPI algorithm extracts semantic constraints such as *domain constraints*, *cardinality constraints*, *referential integrity*, etc. These constraints are mapped using corresponding relational constraints such as *[NOT] NULL*, *UNIQUE*, *PRIMARY/FOREIGN KEY*, etc.

Another approach is the mapping of either DTD or XML Schema to an object-relational database model. Due to the fact that this thesis realizes the mapping of a variant of XML Schema to an object-relational model (see Chapter 8.2), the DTD mapping is not investigated further. In [125, 129], the authors describe the mapping of XML Schema in Oracle XML DB. In their approach, simple types are mapped to appropriate SQL data types (e.g., the primitive types *string* or *decimal* are represented by *VARCHAR2* and *NUMBER*). Semantic constraints information such as the maximum string length is accounted in the resulting model. Complex types are represented as database object types. Nested complex elements and attributes that can occur simply once ( $\text{maxOccurs}=1$ ) are stored as columns in the parent type. Complex types that can occur several times ( $\text{maxOccurs}>1$ ) are stored with the help of collections (nested tables or *VARRAYs*). A foreign key system is introduced to associate complex types to their corresponding parent type. Fidelity of XML documents is guaranteed in respect to whitespace fidelity (by additionally storing the whole document in a single *CLOB*), ordering of elements, comments, processing instructions, namespace declarations, mixed content and prefix information. Furthermore, the shredding level of the XML Schema to the OR model can be varied by the use of the internal *XMLTYPE* which allows the storage of XML fragments as a whole. Besides,

their approach supports extension and restriction of complex types. Cycles within complex types are resolved by the use of a REF (reference) attribute that points at the XML fragments which are stored in the same or in different tables.

## 6.2 Querying XML Databases

In addition to an efficient repository for XML documents, an effective query system is essential. Effective means to easily reach each element to perform efficient join and selection operation. In the following several query languages for XML documents are introduced. The application of the query languages is evaluated with the example shown in Figure 6.1. The left side of the Figure 6.1 defines a simple DTD for a bookstore and the right side shows a possible short XML instance document.

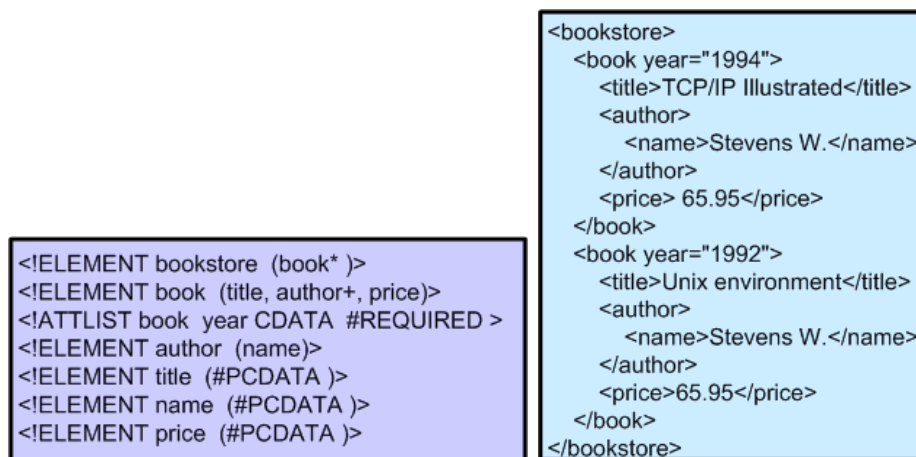


Figure 6.1: Example DTD (left side), corresponding XML document (right side)

XPath [140] (XML Path Language) is a recommendation of the W3C consortium that enables the access to individual parts of data elements in the XML document. The XPath specification considers XML documents as trees and currently distinguishes seven kinds of nodes, namely *root*, *element*, *attribute*, *text*, *comments namespace* and *processing instruction*. The primary construct in XPath is the expression which returns either a node, a boolean, a string or a number. In addition to path expressions, XPath provides a set of functions for manipulating and operating on the XML document and their tree nodes. A detailed explanation is out of scope of this

thesis but can be found at [140]. Furthermore, XPath provides a set of wildcards such as '\*' for varying expressions and enhancing or reducing the result set. Navigation is realized in XPath by introducing the concept of *axes*. An *axis* is a collection of nodes that satisfy the current navigation criteria. For example the *child axis* contains all child nodes of the present context. In general, an XPATH expression consists of a path (where the main difference to filenames or URIs is that each step selects a set of nodes and not a single node) and a possible condition that restricts the solution set. The main disadvantage of XPATH expression is their limited usability in querying XML documents. For instance, it does not provide means for grouping, joins or functions. In the running bookstore example (see Figure 6.1), the following XPath expression would select all books whose price is greater than 10.80.

```
/bookstore/book[price>10.80]
```

XQuery [139, 154, 155] is a declarative query language and consists of the following primary areas: The main areas find their counterparts in SQL. For instance, the *for/let* clauses represent the SQL SELECT and SET statements and are used for defining variables respectively iterating over a sequence of values. The *where* clause complies to the SQL WHERE statement by filtering the selection of the *for* clause. The *order-by* finds their analogous in the SQL SORT BY statement and provides an ordering over a sequence of values. The *return* clause respectively SQL RETURN uses a custom formatting language for creating output. A main part of XQuery is the integration of XPath 2.0 and their functions and axis model which enables the navigation over XML structures. Additional parts are the ability to define own functions analogous to SQL stored procedures and the handling of namespaces. The following XQuery statement uses the running example and returns a list of all books that were published since 1991, including year and title information. The resulting output again is a valid XML document based on the given schema.

```
<bookstore>
{
  for $b in doc("bookstore.xml")/bookstore/book
  where $b/@year > 1991
  return
    <book year="{ $b/@year }">
      { $b/title }
    </book>
}
```



```

}
</bookstore>

```

SQL/XML [141, 156, 157] is an extension of SQL and was developed by an informal group of companies, called SQLX<sup>3</sup>, including among others IBM, Oracle, Sybase and Microsoft. A final draft has been standardized as SQL part 14 (SQL/XML) by ANSI/ISO in 2003. Its main functionality is the creation of XML by querying relational data. For this purpose, SQL/XML proposes three different parts. The first part provides a set of functions for mapping the data of (object-) relational tables to an XML document. The second part specifies an XML data type and appropriate functions in SQL for storing XML documents or fragments of them within a (object-) relational model. The third part describes mapping strategies of SQL data types to XML Schema data types. Oracle supports a subset of the SQL/XML standard within their database by introducing the XMLTYPE data type and appropriate operators (see Section 8.2.2 for a detailed explanation). The running example can be expressed by the following SQL/XML query. Here, again we want to select all books that were published since 1991, including year and title information. For this purpose, the example DTD has to be mapped to an equivalent relational model based on one of the introduced mapping strategies (see Section 6.1.2.1). We assume that there is at least one table called *book* that contains the attributes *year* and *title*.

```

SELECT XMLELEMENT("bookstore",
  XMLAGG(
    XMLELEMENT("book",
      XMLATTRIBUTES(year AS "year"),
      XMLELEMENT("title", title)
    )
  )
)
FROM book WHERE year > 1991

```

### 6.3 Usability of XML Databases for MPEG-7

In this Section, the usability of XML databases for managing MPEG-7 documents is investigated. A very good overview of this topic is given by Westermann and

---

<sup>3</sup><http://www.sqlx.org>

Klas [145]. The authors investigated the following main requirements: *representation of media descriptions*, *access to media descriptions*, *media description schemes*, *extensibility* and *classic DBMS functionality*.

The *representation of media descriptions* considers two main areas: *typed representation* and *fine-grained representation*. With typed representation, the authors investigate the mapping strategy of existing XML solutions in contrast to the origin data type of MPEG-7 descriptions. This means that, elements of MPEG-7 descriptors should be represented by types of their designated meaning (e.g., the *Coeff* element of the *ScalableColorType* descriptor should be represented by an array of double values and not as a string).

Thus, representation is an essential pre-condition in mapping MPEG-7 descriptors to a database schema. Each of them are of particular importance for CBR. Therefore, we have to generate fine grained access to each of them. The unstructured storage approach (see Section 6.1.2) is not applicable here in the sense that it does not provide efficient access to exactly those descriptions, the applications are interested in.

The *access to media descriptions* investigates how granular the access to the desired information is arranged. This means that it is desired to access for instance the *Coeff* element of the *ScalableColor* descriptor directly without the need to parse first the *ScalableColor* descriptor or one of their super classes. In addition, the authors investigated, whether current solutions provide fast access through available index structures.

The *media description schemes* requirement shows whether current mapping strategies result in a MPEG-7 DDL compliant schema catalog for storing description schemes. Based on this catalog the XML solution should provide means for validation and optimized access to stored MPEG-7 description schemes.

The *extensibility* requirement investigates how extensible are current solution in contrast to new functionality and index structures.

The last requirement concerns the occurrence of available classic DBMS functionality.

Since the MPEG-7 standard relies on XML Schema, XML database solutions that only are able to manage DTD representation can be excluded. To summarize their findings neither native XML databases (e.g., Tamino, Xindice, Lore, etc. see

[145] for a list of tested databases) nor XML database extension (e.g., Oracle XML DB, Monet XML, etc. see [145]) provide full support for managing MPEG-7 description schemes in respect to the given requirements. The main disadvantage of all solutions is their lack of considering the correct content type of attribute values of MPEG-7 descriptions. Instead, the values of all attributes are treated as text. Native XML database solutions are strong in representing and accessing MPEG-7 descriptions but they generally lack of extensibility and the support of classic DBMS functionality. The support for indexing particularly in the domain of multimedia, which MPEG-7 addresses, is not sufficient. In fact, no solution supports multidimensional index structures such as X-tree or R-tree for indexing vectors in MPEG-7. Especially, native XML databases suffer in this area by their lack of extensibility. In contrast, XML database extensions have the advantage that they are developed on top of traditional database technology and therefore support any kind of extensibility (e.g., Oracle, DB2) in the area of indexing and enhancement of functionality is given. Nevertheless, they fail to provide an automatical mapping process for the whole MPEG-7 standard to an equivalent database schema. For instance, Oracle provides, based on the described approach in Section 6.1.2.1, an automatic mapping algorithm of XML Schema in their newest version (10g). An XML Schema can be automatically mapped with the help of their *dbms\_xmlschema.registerSchema(...)* PL/SQL function. This process works fine for small company driven XML Schemas but fails in mapping the MPEG-7 standard with the following errors: *ORA-31083: error while creating SQL type SYSTEM.Term212-T* and *ORA-04027: self-deadlock during automatic validation for object SYSTEM.InlineTermDefinitionType208-T*. In addition, Oracle provides a semi-automatic approach, where an user can tag subareas of the XML Schema that should be modelled with their internal XMLTYPE to reduce schema complexity. Nevertheless, this approach is inefficient and time consuming for the use of large and complex XML Schemas as MPEG-7 is.

As a conclusion, we can say that native XML databases do not fulfill the major requirements for multimedia based on their lack of extensibility and indexing. Besides, XML database extensions provide necessary extensibility facilities but are unable to map all MPEG-7 description schemes to an adequate database schema.

---

# 7 Application Scenarios

In the last decade, devices and techniques for producing multimedia data have become ubiquitous. Therefore, an efficient and semantically rich way of retrieving multimedia data becomes an emerging need. In this chapter, possible application scenarios and approaches are introduced. Furthermore, available applications in the main areas of multimedia data, namely image (Section 7.1), audio (Section 7.2) and video (Section 7.3) are briefly described.

## 7.1 Image Retrieval

Image retrieval is a well investigated field in computer science and the domain of multimedia retrieval. There exist a lot of various approaches and commercial and scientific applications (see Section 2.1 and [158] for a summarization). An excellent survey of existing CBIR systems is given by Veltkamp and Tanase, Utrecht University in [159]. An online version, which contains links to all systems, is available at: <http://www.aa-lab.cs.uu.nl/cbirsurvey/cbir-survey/cbir-survey.html>. In this context, this Section briefly introduces several available image retrieval applications and their approaches.

QBIC<sup>1</sup> [23] (Query by Image Content) is the most famous retrieval system in the image retrieval domain. It has been developed at the IBM's Almaden Research Center and relies on similarity searches for the features texture, color and shape. Furthermore, QBIC supports query by sketch (QBS) and query by example (QBE).

---

<sup>1</sup><http://www.qbic.almaden.ibm.com/>

The system has been integrated into the DB2 database (see Section 5.3). In addition, there are several demonstrations available such as the Carnegie Mellon University Demo<sup>2</sup> or the Hermitage Museum Demo<sup>3</sup>.

Photobook was developed from the Vision and Modelling Group at the Institute of Technology in Massachusetts (MIT). It is especially used for face recognition and uses the concept of Eigenfaces. Eigenfaces<sup>4</sup> are a set of eigenvectors of the high-dimensional vector space of faces of human beings. An online demo is available at: <http://vismod.media.mit.edu/vismod/demos/photobook/>.

MARS [21] (Multimedia Analysis and Retrieval System) is a CBIR application developed at the university of Illinois. Images are represented with the features color, texture, shape, layout and text. The system provides several distance functions to calculate similarity and uses boolean queries which internally ranks the results with the *Fuzzy Boolean Model* or the *Probabilistic Boolean Model*. MARS is used by two prototypes. The university of Irvine<sup>5</sup> developed a prototype for the Fowler Museum of Los Angeles for querying African artifacts. The same university established another online demo for terrain similarity matching<sup>6</sup>.

Blobworld is an CBIR system that bases on the blob theory [7] and has been developed at the University of California in Berkeley. Blobs are regions which are roughly homogeneous with respect to color and texture. The aim of the blob-segmentation is to automatically recognize objects (eg., the sky, a car, an animal) within an image. Based on this segmentation, the user is integrated within the retrieval process by choosing the desired region. Additional information is given in Section 11.1. An online demo version is available at: <http://elib.cs.berkeley.edu/photos/blobworld/>.

## 7.2 Audio Recognition

In the area of audio recognition, there exist several outstanding systems. The University of Bonn<sup>7</sup> provides a melody recognition service with the help of their Java

---

<sup>2</sup><http://billswin.inf.cs.cmu.edu/qbic/html/qbic.html>

<sup>3</sup><http://www.hermitagemuseum.org>

<sup>4</sup><http://en.wikipedia.org/wiki/Eigenface>

<sup>5</sup><http://www-db.ics.uci.edu:8004/marsApplet.html>

<sup>6</sup><http://www-db.ics.uci.edu:8003/>

<sup>7</sup><http://www-mmdb.iai.uni-bonn.de/projects/nwo/index.php>

client *notify!WhistleOnline (NWO)*. The service offers three application scenarios. A user can upload an MIDI file or whistles a melody through a microphone or manually insert some notes. Another internet-based melody recognition tool developed by the Fraunhofer institute of "Digitale Medientechnologie" (IDMT) can be tested at <http://www.musicline.de/de/melodiesuche>. Their query by humming (QbH) tool relies internally on the MPEG-7 standard for describing audio specific information. An user can record the chanted melody by microphone and receives a list of ten possible music compositions. The system is not able to process melodies that are recorded by a hi-fi system or are whistled by the user. Philips introduced the query by humming system *CubyHum* [160] which is optimized for the imperfection of human singers. Researches of IDMT develop an audio recognition approach called *AudioID*<sup>8</sup> for the company Thompson Multimedia. Their system also uses internally MPEG-7 for describing the fingerprints of music compositions.

Audio recognition is also used by various companies to avoid piracy of music compositions. For instance, the Dutch company *Medialogging* monitors broadcasts of commercial radio and television stations with the help of an audio recognition system to calculate royalties. The peer-to-peer file sharing tool *Napster* introduced an audio fingerprint system developed by *Relatable*<sup>9</sup> to identify music pirates. Furthermore, this technology is used by the company *Neuros* within their MP3-player.

The largest commercial company in this area is *Shazam*<sup>10</sup> from Great Britain which realizes music recognition over a mobile phone. The same service is known in Germany as *MusicFinder* offered by Vodafone.

### 7.3 Video Browsing

Video browsing and retrieval [161, 162, 163] relies on many approaches of the image and audio retrieval domain. For instance, a video roughly can be classified as a set of images, in the domain of video called frames, that are stringed together. With this knowledge, a simple video retrieval system would provide traditional content-based image retrieval functionalities e.g., similarity search on the basis of low level features

---

<sup>8</sup><http://www.idmt.fraunhofer.de/>

<sup>9</sup><http://www.relatable.com>

<sup>10</sup><http://www.shazamentertainment.com>

such as color, shape and texture that have been extracted for every frame or for a specific key frame. Another possibility is the calculation of an average color histogram over all frames within a specific shot. The main challenging difference in contrast to images is the factor of motion which demands for new approaches in the area of e.g., object recognition [35]. In [164], the authors investigate the combination of speech, image and OCR (optical character recognition) recognition and retrieval in the video domain. An overview of various challenges in the domain of video retrieval is given by Lew et al in [29].

Several approaches [161] and tools, e.g., MISE<sup>11</sup> of Mitsubishi Electric Research Laboratories which is a motion-based indexing and summarization system for video, rely on the MPEG-7 standard for describing necessary low- and high-level features.

In recent years, several video retrieval and publishing systems have been developed. For instance, the Infromedia Digital Video Library project [165] is a research project at Carnegie Mellon University for investigating how multimedia digital libraries can be established.

VideoQ [166] is a Web based video search system, where the user queries the system using animated sketches. The system extracts a number of low level visual and audio features and tracks object within video shots by motion, color and edges. The result of a query is a list of key-frames of the corresponding video shots that matches the given requirements.

The IBM video retrieval system [167] provides a number of novel methods for e.g., fully-automated content analysis, shot boundary detection and speech recognition and indexing. Shot boundary detection is performed using IBM's *CueVideo* system<sup>12</sup>. The system provides several different retrieval strategies such as content-based retrieval, model-based retrieval and speech-based retrieval. The result presents key-frames of matching video shots. Its follower is the MARVEL MPEG-7 Video Search Engine which can be found at: <http://mp7.watson.ibm.com/marvel/>

The Q-Video system [168] includes automatic shot boundary determination, semantic feature extraction and video search. Semantic feature extraction concentrates on indoor, outdoor, city- and landscape detection. The query engine supports QBE (query by example) based on video shots. Query results are displayed on basis of the

---

<sup>11</sup><http://www.merl.com/projects/video-browsing/>

<sup>12</sup><http://www.almaden.ibm.com/projects/cuevideo.shtml>

key frames of video shots.



## Part II

# MPEG-7 Multimedia DataBase (MPEG-7 MMDB)

In the previous chapters, the related work and basic notions concerning multimedia data and multimedia databases have been addressed. Based on the identified lack in supporting and managing the requirements of multimedia data in terms of semantic meaningful querying, advanced indexing, content modelling and multimedia programming libraries, we come up with the approaches described in the following chapters. Our resulting architecture is presented in Figure 7.1 and consists of four main parts.

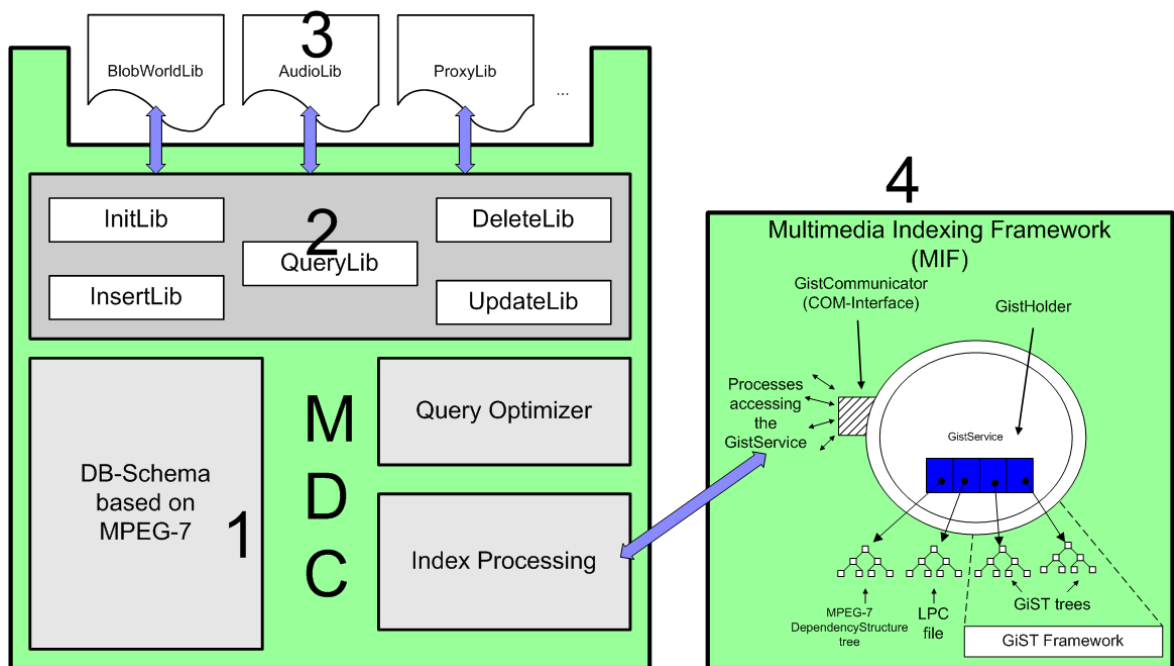


Figure 7.1: MPEG-7 Multimedia DataBase (MPEG-7 MMDB)

The core system, numbered with 1, is composed of the multimedia database schema that relies on MPEG-7, an index processing part that integrates multi-dimensional index structures and an enhancement of the cost-based query optimizer for approximating the selectivity of Range-Searches. The core system is introduced in Chapter 8. Chapter 9 presents the integration and implementation of our internal libraries (see number 2 in Figure 7.1) that provide basic functionality to the upper layers. The application libraries (see number 3 in Figure 7.1) are explained in Chapter 10. They provide three example implementations how the MPEG-7 database in combination with their internal libraries can be used to provide necessary retrieval

and browsing functionality to client applications. The sample implementations cover application libraries in the area of audio recognition, image retrieval and video browsing. The corresponding tools are introduced in Chapter 11.

The core management system of the MPEG-7 Multimedia Database (MPEG-7-MMDB, see Figure 7.1 part 1) is integrated into an Oracle database with the help of the Oracle Data Cartridge technology [107] (see Chapter 8.1 for a short introduction). Based on the Data Cartridge technology, we extended the core system by the following means:

- We introduce a novel database schema that relies on MPEG-7 and uses the *extensible type system* of the cartridge environment. A detailed explanation is given in Section 8.2.
- Further, we extend the *extensible indexing system* of the cartridge environment by a new multimedia indexing framework (MIF, see Figure 7.1 part 4) offering different access methods. MIF increases on the one hand the performance of database queries and extends on the other hand the database functionality for e.g., content-based retrieval. The different access methods, their implementation and their usage for MPEG-7 descriptions are explained in Section 8.3
- In addition, we introduce a multimedia query optimizer that relies on the *extensible optimizer system* of the cartridge technology. The multimedia query optimizer extends the internal cost-based optimizer by approximating the selectivity of Range-Search queries. The multimedia query optimizer is illustrated in Section 8.4

## 8.1 Oracle's Data Cartridge Technology

Oracle offers with its *Data Cartridge* technology a mechanism for extending the capabilities of an Oracle database. Oracle databases are built according to a modular architecture with *extensible services*. These extensible services enable database designer and programmer to extend e.g., the type system, the query processing or the data indexing (see Figure 8.1). Each extensible service offers an extensibility interface which can be used to enhance and modify the database for the users needs. The main characteristics of data cartridges are the following:

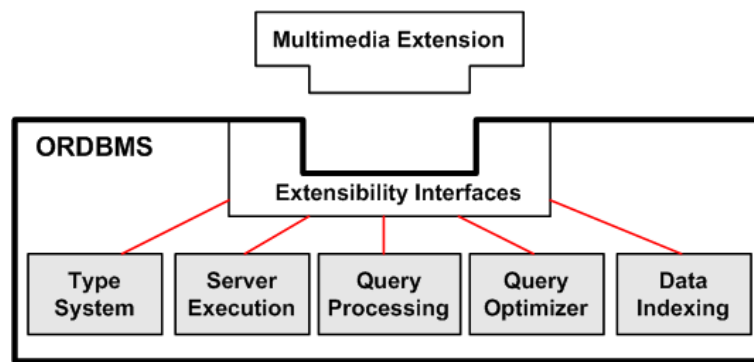


Figure 8.1: Oracle Data Cartridge

- **Data Cartridges are server-based.**

Their components reside at the server and are primary accessed from the server.

- **Data Cartridges extend the server.**

With the possibility of defining new types, one is able to create solutions for real world problems. For example, one can create a table *Company* with a column *Person*. Each *Person* object contains additional information in the form of a photo of type *Image*. All new created types (e.g., *Company*, *Person*) can be enhanced by behavior e.g., for transforming the image, for calculation of the persons salary, and so on.

- **Data Cartridges are integrated with the server.**

The extensions made to the server are integrated within the server engine, so that the optimizer, query parser, indexer and other server mechanisms recognize

and respond to these extensions. By implementing all necessary interface methods of the extensibility interface, one can be sure that e.g., a newly integrated domain-specific index is used for increasing performance in querying new types.

- **Data Cartridges are packed.**

A Data Cartridge can be seen as a unit. All extensions can be packed to a unit and easily installed on every Oracle database. In this scenario, no additional implementation has to be done and every Oracle database can be equipped with the new functionality without further maintenance.

### 8.1.1 Oracle's Extensibility Services

Since Oracle8i, the server provides services for basic storage, query processing, optimization and indexing. These services have been made *extensible* for application developers to trim the database to their special needs. The usual way for using these services is to implement a Data Cartridge that extends the extensibility interface. This chapter describes the most basic and common used extensible services. It is not necessary for a Data Cartridge developer to implement all methods in the extensible interfaces.

- **Extensible Type System**

Beside the common native SQL data types, such as `INTEGER`, `NUMBER`, `DATE` and `VARCHAR`, Oracle adds support for new types including user-defined objects, collections (`VARRAY`, nested tables), internal large object types (`BLOB`, `CLOB`) or `BFILE`.

The extensible type system is in most cases the main feature for Data Cartridge developments. Here, one may define new domain-specific object types. These types specify the underlying persistent data (which are represented through their *attributes*) and are enhanced by their relevant behavior (*methods*). Object types are used to extend the modelling capabilities in common with native data types. Every object type can have one or more *attributes*. These *attributes* consist of a name and a type. This type can again consist of domain-specific object types or native types. The *method* is a procedure or a function which

has directly access to the object's *attributes*. The reader is referred to [107] for further information on other data types.

- **Extensible Server Execution Environment**

Oracle's server execution environment (see Figure 8.2) provides two main features. First, the components of a data cartridge and other database procedures and functions can be implemented in any popular programming language such as PL/SQL, Java or external C language routines. A combination of programming languages is possible. Thus, one object method can be implemented in Java, another in PL/SQL and the Java method in turn can call, via a PL/SQL function, an external C routine. The use of external C routines is useful for computation-intensive operations. These routines are executed in a separate address space than the server. This fact ensures that the database server is insulated under all circumstances from program failures. Nevertheless external routines are able to *call back* to the Oracle Server using OCI (Oracle Call Interface).

Second, the type system decouples the implementation of an object's method from its specification. The specification just defines the head of the method with appropriate in and out parameters. It is not defined what kind of implementation is behind the object's method. Therefore, a data cartridge developer can simply exchange method implementations (either the used programming language or the algorithm).

- **Extensible Indexing**

Typical DBMS, including Oracle, supports just a few types of access methods ( $B^+$  Trees, Hash Indexes, Text Indexes) on a limited number of data types (number, strings). However, commercial DBMS have to store and handle multimedia data such as images, videos or other user-defined objects. In addition, one needs efficient query and retrieval for these kinds of data. Indexing plays a major role in this context. But database companies cannot provide access methods for all kind of user needs. Therefore, Oracle introduces an extensible server architecture for defining new index types as required. The data cartridge is responsible

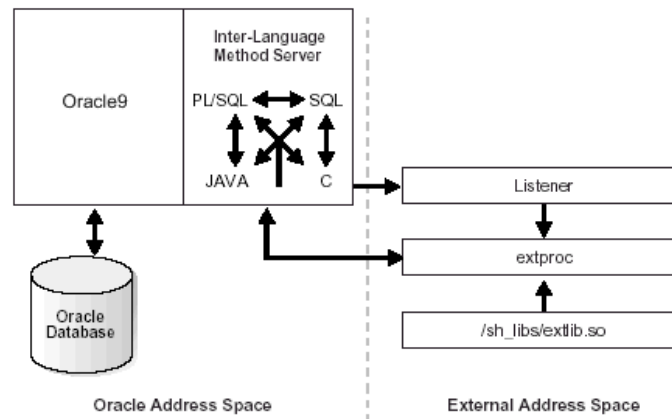


Figure 8.2: Oracle Server Execution Environment

for defining the index structure, maintaining the index content during load and update operations and searching the index during query processing. The index itself can be stored internally as a heap-organized or index-organized table or externally as an operating system file. Internal solutions lack in common in performance (e.g., no implementation in C is possible) and space restrictions.

Oracle introduces the concept of an *indextype* for user-defined access methods. Each *indextype* has specific operators and uses an object that is responsible for the implementation. The object has to implement all necessary functions which are provided by the ODCI (Oracle Data Cartridge Interface) for indexing e.g. *ODCIIndexCreate*, *ODCIIndexInsert* or *ODCIIndexDrop*.

```
CREATE OR REPLACE INDEXTYPE <indexname> FOR
  <operator1>,
  <operator2>,
  ...
USING <object>;
```

Typical databases do not support extensible indexing. Therefore, application developers have to maintain their own file-based indexes for complex data. It needs, of course, a great effort to retain consistency between external indexes and the related data in the database. This additional effort and complexity is not needed in the Oracle extensible indexing concept.



- **Extensible Optimizer**

With the extensible optimizer functionality, one is able to create statistic collections, selectivity and cost functions for the user-defined indexes, functions and operators. This information is used by the optimizer in choosing the query plan. An application developer has only the possibility to enhance the cost-based optimizer. However, there is no way in adding functionality to the rule-based optimizer.

## 8.2 DB-Schema

The MMDB schema provides a mechanism for mapping the MPEG-7 standard to an object-relational database model. Our approach has been implemented and tested with the help of Oracle's data cartridge technology (see Section 8.1), but is open to all databases that provide comparable functionality for extending the core of their system. In the following, we give a short introduction to object-relational database features, including the *XMLTYPE*. There we will give a step by step explanation on how the MPEG-7 standard with all its specific cases has been mapped to the database model. Furthermore, we compare our solution with currently available XML-databases in the way of handling MPEG-7 documents.

### 8.2.1 Object-relational database features

This Subsection concerns on object-relational features such as domain-specific types, inheritance, object tables, references and relationships among types, polymorphism, collections and the dot operator.

- **Domain-Specific Types**

A database type is a collection of attributes, relationships and methods. Attributes contain the content of the object and can be of arbitrary type. A relationship specifies the connection between different objects and are modelled in an object-relational DBMS as references. Methods are used for manipulating the object's content.

The main limitation of relational DBMS is their bondage to system specific data types such as INTEGER, VARCHAR, etc. which is a strong restriction in modelling real world problems. In contrast, object-relational DBMS allow the specification of domain-specific types. The example below shows the creation and usage of domain-specific types. The *AS OBJECT* construct creates a new separated and not derived database type. The *NOT FINAL* extension is used for allowing inheritance hierarchies.

```
CREATE OR REPLACE TYPE address_type AS OBJECT (  
    street VARCHAR2(250),  
    city   VARCHAR2(200)  
);  
/  
  
CREATE OR REPLACE TYPE person_type AS OBJECT (  
    first_name VARCHAR2(150),  
    surname    VARCHAR2(150),  
    address    address_type  
) NOT FINAL;  
/
```

Recursive types can be modelled with the help of incomplete types, which can be instantly referenced, and are completely defined later on. A recursive type is identified by the absence of the *AS OBJECT* construct.

```
CREATE OR REPLACE TYPE incomplete_Address_type;  
/
```

### • Inheritance

Oracle offers the possibility to derive subtypes of existing types by using the *UNDER* key word. In this context, inheritance is only allowed if the super type has been defined with the *NOT FINAL* key word. Multiple inheritance is not possible. The subtype can add new attributes and new methods. Besides, one can change the object's behavior by overriding methods of the super type.

```
CREATE OR REPLACE TYPE student_type UNDER person_type (  
    -- additional attributs
```

```
    student_id INTEGER
);
/
```

### • Object Tables

Instances of domain specific types are stored in object tables which are structurally equivalent to their types. Additional conditions, such as primary-/secondary keys or *NOT NULL* restrictions can only be defined at the level of tables and not in the specification of types. Data records in object tables are objects with a system wide unique object identifier (OID).

```
CREATE TABLE students OF student_type (
    surname NOT NULL,
    PRIMARY KEY (student_id)
);
/
```

The insertion process is realized, analogue to relational DBMS, with the help of the *INSERT INTO* SQL statement. One has only to take care of domain specific types that are nested into the object table, like the *address\_type* in our running example. These data needs an explicit type cast.

```
INSERT INTO students VALUES (
    'Mario',
    'Döller',
    address_type('University street', 'Klagenfurt'),
    9560334
);
/
```

### • References and Relationships among Types

Semantic connections among objects are modelled in object relational systems with the help of relations and their cardinality. As a matter of principle three classes of relationships exist:

- 1:1 relationship connects exactly 2 objects (one of type A and one of type B).

- 1:n (n:1) relationship specifies that exactly one object of type A is connected to n objects of type B.
- n:m relationship offers the possibility to connect n objects of type A with m objects of type m.

In relational DBMS, relationships has to be realized with the help of foreign keys. Besides the usage of foreign keys, nowadays object-relational DBMS provides the possibility to model 1:1 relationships with object references between tables. These object references are pointers to a single row of a database table. For example we can simply modify our *person\_type* in the following way.

```
CREATE OR REPLACE TYPE person_type2 AS OBJECT (  
    first_name  VARCHAR2(150),  
    surname     VARCHAR2(150),  
    address     REF ADDRESS_TYPE  
) NOT FINAL;  
/
```

Now the *person\_type2* has been extended with the *REF* key word for the address which allows an 1:1 relationship between an person object and an address object. The insertion process has to be slightly modified. First, we have to create a table of type *address\_type*. Second, the address information of the person has to be inserted and third the person data is inserted with the correct OID of the corresponding address entry. The OID of a row of a table can be retrieved by using the *REF* statement in the select clause.

```
CREATE TABLE address OF address_type;  
  
INSERT INTO address VALUES ('University street','Klagenfurt');  
  
CREATE TABLE person2 OF person_type2;  
  
INSERT INTO person2 VALUES (  
    'Mario',  
    'Döller',  
    (SELECT REF(a)  
     FROM address a  
     WHERE a.street LIKE 'University stree' AND  
           a.city LIKE 'Klagenfurt')  
)
```

The OID of an object can be dereferenced with the *DEREF* SQL command.

- **Polymorphism**

Inheritance in databases enables another advanced feature: polymorphism. Polymorphism in database systems means that rows of an object table of type A can contain instances of this type or instances of any subtype of A. The following example uses polymorphism by storing instances both of type *person\_type* and of type *student\_type*.

```
CREATE TYPE university_type AS OBJECT (  
    Name    VARCHAR2(200),  
    person  person_type  
);  
/  
  
CREATE TABLE university OF university_type;
```

The new type *university\_type* contains among others an attribute of *person\_type* which enables the insertion of instances of the super type (*person\_type*) as well as instances of their subtypes (e.g. *student\_type*) into the table *university*.

- **Collections**

Collections in object-relational DBMS facilitate the realization of 1:n relationships. In general, there exist two possibilities: *nested tables* and *VARRAYS*.

The data of a column of *VARRAY* is stored in-line which means that it is stored along with the remaining data of the row. The main disadvantage of *VARRAYS* is their limitation of storing only a maximum number of elements. This maximum has to be specified during instantiation. Further in contrast to nested tables, *VARRAYS* are not maintainable with the *UPDATE* and *DELETE* command.

Nested tables are tables within an attribute. They are defined with the help of a type and therefore can store a collection of attributes, references or other complex nested types. Nested tables are not limited by a maximum capacity. Let us extend our running example, so that a person can have multiple addresses. For this purpose, we have to create a nested type of type *address\_type* by using the *AS TABLE OF* key word.

```
CREATE OR REPLACE TYPE nested_address_type AS TABLE OF
address_type;
/
```

In addition, it is also possible to create a nested table of references of type *address\_type*. This can be realized with the following statement where we additionally have to use the *REF* key word.

```
CREATE OR REPLACE TYPE nested_address_type2 AS TABLE OF REF
address_type;
/
```

Further, we need to exchange the type of the address attribute at our *person\_type* by one of the two newly defined address types. We will call the new person type *person\_type\_nested*. For completing this example, we now can create a table of the new person type with the following statement, whereas we have to declare for each nested table a separate table name.

```
CREATE TABLE person_nested OF person_type_nested
    NESTED TABLE Address STORE AS Address_Tab;
/
```

The insert statement for the *person\_nested* table looks as follows.

```
INSERT INTO person_nested VALUES (
    'Mario',
    'Döller',
    nested_address_type (
        address_type (
            ...
        ),
        address_type (
            ...
        )
    )
);
```

If we have used the second nested address type which internally uses references, then the insert statement will be slightly different.

```

INSERT INTO person_nested VALUES (
  'Mario',
  'Döller',
  nested_address_type2 (
    (SELECT REF (a)
     FROM address a
     WHERE ...
    ),
    (SELECT REF (a)
     FROM address a
     WHERE ...
    )
  )
);

```

- **Dot operator**

The advantage of references and nested tables is their easy and efficient query behavior. With the help of the dot operator one can easily navigate within the schema without using foreign keys. If we want to list the address of our students then the following select statement is sufficient.

```

SELECT s.address.street, s.address.city FROM students s;

```

This statement is valid in both examples (address specified as complex data type or as reference) and hides the real complexity of the schema. We do not need any foreign keys and no JOIN between the students and the address table has to be specified.

Nested tables such as those which we have used for realizing multiple addresses can be accessed with the help of the *TABLE* keyword.

```

SELECT p.surname, a.street, a.city
FROM person_nested p, TABLE(p.address) a;

```

The last example demonstrates the access to nested tables that contain references as we had alternatively used for storing multiple addresses. For this purpose, the *DEREF* key word has to be used for dereferencing the stored OIDs.

```

SELECT p.surname, Deref(a.COLUMN_VALUE).street, Deref(a.COLUMN_VALUE).city
FROM person_nested p,
     TABLE(p.address) a;

```

## 8.2.2 The XMLTYPE data type

The *XMLTYPE* is a new DB data type which is used for storing whole XML documents or fractions of them. It internally uses a CLOB type for representing the data and has therefore a maximum capacity of 4 GB. XPath 1.0 expressions (see Oracle's XML guide [169]) can be used for querying and traversing data of the *XMLTYPE*. Furthermore, the *XMLTYPE* provides several member methods that can be used in combination with XPath expressions for database queries. For instance, one can use the *extract()* method in combination with a corresponding XPath expression and the *getStringVal()* method for querying attributes data of *XMLTYPE*. Besides the possibility of returning query results in a more or less traditional way, the *XMLTYPE* and their appropriate methods provide the necessary functionality for formatting query results as XML. In the following, we very briefly explain the mostly used methods.

- *XMLGEN* allows the transformation of exactly one attribute of a database table into an *XMLTYPE*. For instance, a query retrieving the *address* attribute of the *students* table results into the following XML output.

```

SELECT SYS_XMLGEN(address) FROM students;

SYS_XMLGEN (ADDRESS)
-----
<ADDRESS>
  <STREET>University street</STREET>
  <CITY>Klagenfurt</CITY>
</ADDRESS>

```

- The *XMLFOREST* offers more or less the same functionality as *XMLGEN* but can be used for more than one attribute at the same time.

```

SELECT SYS_XMLFOREST(s.surname, s.address) "Students" FROM
students s;

```



```

Students
-----
<SURNAME>Döllner</SURNAME>
<ADDRESS>
  <STREET>University street</STREET>
  <CITY>Klagenfurt</CITY>
</ADDRESS>

```

- The *XML*Element allows the creation of XML output with XML based attributes and a hierarchy by using the *XML*Element recursively.

```

SELECT XMLELEMENT("Student",
  XMLATTRIBUTES(s.id AS "StudentID"),
  XMLELEMENT("Name",
    XMLELEMENT("Surname",s.surname)
  )) AS "Students" FROM students s;

```

```

Students
-----
<Student StudentID="9560334">
  <Name>
    <Surname>Döllner</Surname>
  </Name>
</Student>

```

- The *XML*Agg command connects together a collection of XML fractions to one XML element.

```

SELECT XMLELEMENT("Students",
  XMLAGG(XMLELEMENT("Surname",s.surname)
  )) AS "Students" FROM students s;

```

```

Students
-----
<Students>
  <Surname>Döllner</Surname>
  <Surname>Schojer</Surname>
</Students>

```

Without using the *XML*Agg command, the output will look as follows:

```

Students
-----
<Students>

```

```

        <Surname>Döller</Surname>
    </Students>

    <Students>
        <Surname>Schojer</Surname>
    </Students>

```

### 8.2.3 Guidelines for mapping MPEG-7 to an object-relational DB model

Based on the abilities of object-relational DBMS (see Section 8.2.1 and Section 8.2.2) and Oracle's data cartridge technology (see Section 8.1) we present all steps that are necessary to map the MPEG-7 standard (see Section 3.4) to an equivalent object-relational database schema.

XML	DB
Complex data types	Domain specific types
Inheritance	Inheritance
Attributes + Elements	? (use metadata to distinguish between attributes and elements)
Collections	Nested tables/VARRAY
Sequence/Choice/Union	?
Recursion	?

Figure 8.3: XML constructs and their possible database counterpart

As the MPEG-7 standard relies on XML-Schema, we first have to find solutions for XML-Schema specific constructs. Figure 8.3 shows the main XML specific cases and their possible database counterparts. As one can see complex data types, inheritance and collections have more or less various solutions in object-relational DBMS. But the other constructs, such as recursion, sequence, choice, etc. have to be investigated more precisely.

The following main guidelines for mapping MPEG-7 were imposed. A more detailed explanation is given in the following paragraphs.

- The use of the XMLTYPE for decreasing the amount of different data types. MPEG-7 descriptors that are not in our main focus for querying and indexing have been represented with the help of the XMLTYPE. Nevertheless the information is not lost and can be retrieved by the use of XPATH expressions in SQL statements (see Section 9.4 for an example).
- Identify those descriptors which are of interest for querying. The mapping of the remaining descriptors should be done with the XMLTYPE.
- Reduce the complexity by flattening down the inheritance hierarchy to few levels. In general, all abstract data types have been skipped.
- Object references have to be used for navigation throughout the schema. Furthermore, we introduce a key system (DOC\_ID, PART\_ID) in each table. The *DOC\_ID* is used to assign each tuple to the corresponding document and the *PART\_ID* is used to identify their position within a document.
- In MPEG-7, polymorphism is realized by the use of the "xsi:type" key word (see Figure 8.5 for a possible MPEG-7 descriptor and Figure 8.6 for a possible instance document). The solution for such problems is in general the introduction of a new table column containing the name of the target descriptor and the corresponding *PART\_ID*.

The combination of object-relational database features, relational keys and object references allows the mapping of the whole MPEG-7 standard into a corresponding database schema. The reduction of the MPEG-7 inheritance hierarchy by skipping abstract types and merging types that only contain a few attributes and elements results in a compact arranged database schema that allows the storage of any kind of MPEG-7 document and offers an efficient and rich model for querying it.

In the following, the transformation rules will be considered in more detail.

**Identification of stored MPEG-7 documents** During the insertion process the MPEG-7 document is divided into several pieces and stored in different tables according their descriptors and descriptor schemes. The primary key *DOC\_ID*, which

is part of each DB type and table, is used for assigning the different parts to their correct MPEG-7 document.

**Multiple occurrence of same elements in a single MPEG-7 Document** The MPEG-7 standard allows the creation of MPEG-7 documents that contain multiple elements of the same data type in a predefined order. This order is modelled with the help of the primary key *PART\_ID* which is also part of each DB type and table. Each tuple of a table can be clearly associated to a single MPEG-7 document with a *DOC\_ID* for the document and their position within the document by *PART\_ID*.

**Reducing the inheritance hierarchy of MPEG-7 types** The MPEG-7 standard extensively uses already defined data types for extension and modification. For instance, such an inheritance hierarchy can be for example: *VideoSegmentSpatioTemporalDecompositionType* extends *SpatioTemporalSegmentDecompositionType* which extends *SegmentDecompositionType* that extends *DSType* that extends *Mpeg7BaseType* which finally extends *anyType*. The attempt to map the whole MPEG-7 inheritance hierarchy to an equivalent DB model failed by the amount of possible subtypes and technical problems of Oracle. Therefore, the root element of the current DB schema is the *DSType*. The both types *SpatioTemporalSegmentDecompositionType* and *SegmentDecompositionType* have not been modelled, in place all attributes and elements have been forwarded to the next type in the hierarchy which is in our case the *VideoSegmentSpatioTemporalDecompositionType*. The resulting schema is slimmer and contains no loss of semantic information, however the original inheritance hierarchy of MPEG-7 is hardly recognizable.

**Sequences, Choices and unbounded Collections** Complex data types of the MPEG-7 standard can contain sequences, choices and unbounded collections. The mapping rules are the following:

1.) **Unbounded Collections:**

Unbounded collections specify constructs that can have multiple (unbounded) occurrence. These constructs are realized with the help of nested tables. For this purpose, we introduce the following nomenclature and rules:

- a.) In general, unbounded collections (e.g., choices) cover a set of subelements. These subelements are mapped as attributes to new database types labelled **dummyTypeXXX**.
- b.) As these collections are unbound (e.g., an unbounded number of elements can occur), we have to introduce a nomenclature for the appropriate nested table name and its type. In the following, the nested table type is defined as **dummyNTTypeXXX** and the corresponding nested table name is called **dummyNTTabXXX**. The nested table type references to the previously created dummy type containing the subelements of the unbounded collection.
- c.) In order to represent and access the unbounded collection within its origin type, a new attribute name has to be introduced which is labelled **dummyAttrXXX** and is of the nested table dummy type created beforehand.

The **XXX**-extension represents consecutive numbering.

## 2.) Sequences:

Sequences denote the order of elements within an instance documents. In this respect, the order of the database attributes has to be the same as in the MPEG-7 descriptor.

## 3.) Choices:

Choices are constructs that provide several elements for selection whereas within an instance document only one element is allowed. Every element of the choice construct is mapped to a table attribute and the corresponding database type. Besides, if the choice is an unbounded collection then the rules of point one (see above) have to be executed additionally.

Let us illustrate the mapping rules by considering the *TextAnnotationType* MPEG-7 descriptor which is demonstrated in Figure 8.4.

The *TextAnnotationType* descriptor consists of three attributes, namely *confidence*, *relevance* and a reference to *xml:lang*. Besides, it contains a choice among

```

<!-- ##### -->
<!-- Definition of TextAnnotation datatype (7.2.3) -->
<!-- ##### -->

<!-- Definition of TextAnnotation datatype -->
<complexType name="TextAnnotationType">
  <choice minOccurs="1" maxOccurs="unbounded">
    <element name="FreeTextAnnotation" type="mpeg7:TextualType"/>
    <element name="StructuredAnnotation" type="mpeg7:StructuredAnnotationType"/>
    <element name="DependencyStructure" type="mpeg7:DependencyStructureType"/>
    <element name="KeywordAnnotation" type="mpeg7:KeywordAnnotationType"/>
  </choice>
  <attribute name="relevance" type="mpeg7:zeroToOneType" use="optional"/>
  <attribute name="confidence" type="mpeg7:zeroToOneType" use="optional"/>
  <attribute ref="xml:lang"/>
</complexType>

```

Figure 8.4: MPEG-7 TextAnnotationType

the following four elements: *FreeTextAnnotation*, *StructuredAnnotation*, *DependencyStructure* and *KeywordAnnotation*. The occurrence of the elements is defined as *unbound* which means that any amount is allowed. The choice construct is mapped according to rule 3, which determines that every element is considered in the new database type (see type *dummyType14* below). As in addition, the choice is an unbounded collection, we have to take into account rule 1. Therefore, the *TextAnnotationType* database type contains an attribute (*dummyAttr12*) that is of type *dummyNTType13* which references to a nested table of type *dummyType14*. The database table is named *TextAnnotation* and has a nested table (labelled *dummyNT-Tab15*) for the attribute *dummyAttr12*. The resulting database types are presented below:

```

CREATE TYPE dummyType14 AS OBJECT (
  DOC_ID INTEGER,
  PART_ID INTEGER,
  FreeTextAnnotation CLOB,
  lang VARCHAR2(50), -- attribute
  StructuredAnnotation SYS.XMLTYPE,
  DependencyStructure SYS.XMLTYPE,
  KeywordAnnotation SYS.XMLTYPE
);
/

CREATE TYPE dummyNTType13 AS TABLE OF REF dummyType14;
/

CREATE TYPE TextAnnotationType AS OBJECT (
  DOC_ID INTEGER,
  PART_ID INTEGER,

```

```

dummyAttr12 dummyNTType13,
relevance FLOAT(126), -- attribute
confidence FLOAT(126), -- attribute
lang VARCHAR2(50) -- attribute
);
/

CREATE TABLE TextAnnotation OF TextAnnotationType (PRIMARY KEY (DOC_ID,PART_ID))
  NESTED TABLE DUMMYATTR12 STORE AS dummyNTTab15;

```

**Dereferencing xsi:type values** MPEG-7 documents uses *xsi:type* references to link supertypes with subtypes and therefore realizes polymorphism. Figure 8.5 shows the MPEG-7 *MultimediaContentType* descriptor and one of several possible subtypes (e.g.: *VideoType*, *ImageType*, etc.)

```

<!-- Definition of MultimediaContent Content Entity -->
<complexType name="MultimediaContentType" abstract="true">
  <complexContent>
    <extension base="mpeg7:DSType">
    </extension>
  </complexContent>
</complexType>

<!-- Definition of Image Content Entity -->
<complexType name="ImageType">
  <complexContent>
    <extension base="mpeg7:MultimediaContentType">
      <sequence>
        <element name="Image" type="mpeg7:StillRegionType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

```

Figure 8.5: MPEG-7 MultimediaContentType

Figure 8.6 shows a possible instance document that uses the *MultimediaContentType*. As one can see, the *MultimediaContentType* references to an *ImageType* with the help of an *xsi:type* construct.

This XML concept can be mapped by using the *PART\_ID* of the referenced subtype (stored in *MultimediaContent.PART\_ID*) and the name of the *xsi:type* reference (inserted into *MultimediaContent*). The name identifies the following type and in which table it is inserted and the *PART\_ID* is used for identifying the position within the document. The *DOC\_ID* is not needed because such references are only allowed within a single document. Therefore, we required for mapping the MPEG-7 descriptor *MultimediaContentType* the following types.

```

<Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
xsi:schemaLocation="urn:mpeg:mpeg7:schema:2001 Mpeg7-2001.xsd">
  <Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="ImageType">
      <Image>
        ...
      </Image>
    </MultimediaContent>
  </Description>
</Mpeg7>

```

Figure 8.6: MPEG-7 Instance document that uses the MultimediaContentType

```

CREATE TYPE MultimediaContentType UNDER DSType (
  MultimediaContent VARCHAR2(60),
  MultimediaContent_PART_ID INTEGER
);
/

CREATE TYPE dummyNTType07 AS TABLE OF MultimediaContentType;
/

CREATE TYPE ContentEntityType AS OBJECT (
  DOC_ID INTEGER,
  PART_ID INTEGER,
  DescriptionMetadata SYS.XMLTYPE,
  Relationships SYS.XMLTYPE, -- unbounded
  OrderingKey SYS.XMLTYPE, -- unbounded
  Affective SYS.XMLTYPE, -- unbounded
  MultimediaContent dummyNTType07 -- unbounded
);
/

```

This mechanism has been used for each MPEG-7 descriptor scheme where *xsi:type* references are allowed (e.g., Agent, SemanticBase or VariationSet).

**Resolving recursion of MPEG-7 types** Recursions are not directly educible with conventional database methods. In general, they can be realized by breaking the recursion circle by introducing a new type. The approach for resolving recursions includes the following steps:

- 1.) Create an incomplete database type for the recursive MPEG-7 type. The incomplete type has only a name and no attributes nor methods.



- 2.) Create a type (which name should be set corresponding the previous introduced nomenclature) that maps all elements of the recursive type. The recursive element can now reference to the incomplete type.
- 3.) If the MPEG-7 type contains additional unbounded collections, then one has to proceed as described above.
- 4.) Finally, complete the original incomplete type by referencing to the newly created type (step 2).

```

<!-- ##### -->
<!-- Definition of Object DS (12.3.5) -->
<!-- ##### -->

<!-- Definition of Object DS -->
<complexType name="ObjectType">
  <complexContent>
    <extension base="mpeg7:SemanticBaseType">
      <choice minOccurs="0" maxOccurs="unbounded">
        <element name="Object" type="mpeg7:ObjectType"/>
        <element name="ObjectRef" type="mpeg7:ReferenceType"/>
      </choice>
    </extension>
  </complexContent>
</complexType>

```

Figure 8.7: MPEG-7 ObjectType descriptor

In the following example, we consider the MPEG-7 *ObjectType* descriptor (see Figure 8.7) that contains an element of its own type. The mapping is realized according the identified steps and result in the following database types:

```

CREATE TYPE ObjectType; -- incomplete definition
/

CREATE TYPE dummyType56 AS OBJECT (
  Object REF ObjectType,
  ObjectRef SYS.XMLTYPE
) NOT FINAL;
/

CREATE TYPE dummyNTTab55 AS TABLE OF dummyType56;
/

CREATE TYPE ObjectType UNDER SemanticBaseType (
  dummyAttr54 dummyNTTab55 -- unbounded
) NOT FINAL;
/

```

## 8.2.4 Resulting DB-Schema

Based on the guidelines presented in Section 8.2.3 our approach for mapping MPEG-7 is explained with the help of several MPEG-7 descriptors and descriptor schemes that represent all possible special cases. Figure 8.8 shows the parts of the class hierarchy of the MPEG-7 standard. We restricted ourself to the Content Description and Content Abstraction part of MPEG-7. The gray filled boxes represent MPEG-7 descriptions that fully have been mapped to our database schema. The dotted boxes are MPEG-7 descriptions that partly have been mapped to our schema. All other parts have not been taken into account, but can be mapped according to our guidelines.

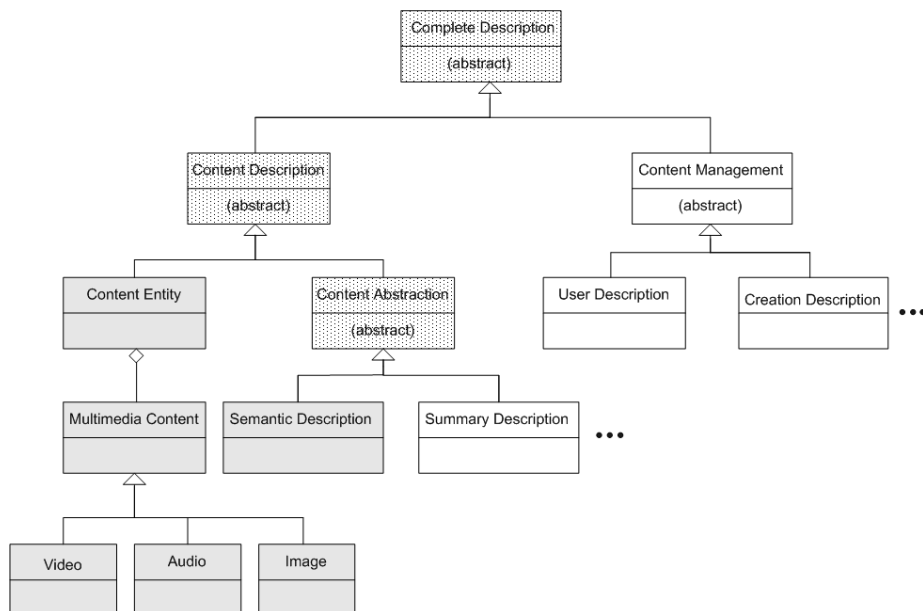


Figure 8.8: MPEG-7 Class Hierarchy

Figure 8.9 describes all graphical constructs that are used within our model. The model is split into six parts partly according to their MPEG-7 equivalents and for an better overview. These are the *Semantic* part represented in Figure 8.16, the *Image* part illustrated in Figure 8.12, the *Video* part displayed in Figure 8.14, the *Audio* part demonstrated in Figure 8.13 the *Audio-Visual* part in Figure 8.15 and the *Creation information* part shown in Figure 8.11.

Based on flattening down the inheritance hierarchy manually, the top most database type is the MPEG-7 *DSType*. It is the super type for more than 15 subtypes and has

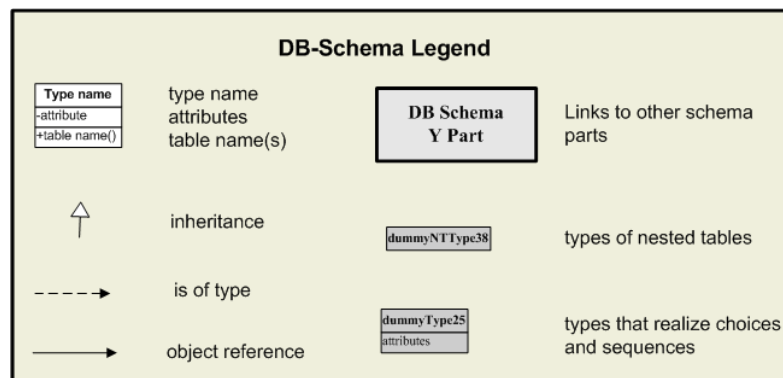


Figure 8.9: Legend of DB Schema

the following MPEG-7 definition (see Figure 8.10).

```

<!-- Definition of generic DS -->
<complexType name="DSType" abstract="true">
  <complexContent>
    <extension base="mpeg7:Mpeg7BaseType">
      <sequence>
        <element name="Header" type="mpeg7:HeaderType" minOccurs="0"
          maxOccurs="unbounded"/>
      </sequence>
      <attribute name="id" type="ID" use="optional"/>
      <attributeGroup ref="mpeg7:timePropertyGrp"/>
      <attributeGroup ref="mpeg7:mediaTimePropertyGrp"/>
    </extension>
  </complexContent>
</complexType>

<!-- Definition of timePropertyGrp attribute group -->
<attributeGroup name="timePropertyGrp">
  <attribute name="timeBase" type="mpeg7:xPathRefType"
    use="optional"/>
  <attribute name="timeUnit" type="mpeg7:durationType"
    use="optional"/>
</attributeGroup>

<!-- Definition of mediaTimePropertyGrp attribute group -->
<attributeGroup name="mediaTimePropertyGrp">
  <attribute name="mediaTimeBase" type="mpeg7:xPathRefType"
    use="optional"/>
  <attribute name="mediaTimeUnit" type="mpeg7:mediaDurationType"
    use="optional"/>
</attributeGroup>

```

Figure 8.10: MPEG-7 DSType descriptor

The *DSType* descriptor consists of one element labeled *Header* which can occur several times, one attribute named *id* and two attribute groups. The resulting database type contains all attributes and elements that are marked blue. In this example one can see that, for simplicity, all attributes of both attribute groups are integrated to their base types. They are not considered separately which reduces the amount of mapped types. The resulting database type is as follows:

```

CREATE TYPE DStype AS OBJECT ( -- extension base="mpeg7:Mpeg7BaseType"
  Header SYS.XMLTYPE, -- unbounded
  id VARCHAR2(4000), -- attribute
  timeBase SYS.XMLTYPE, -- attributeGroup timePropertyGrp
  timeUnit SYS.XMLTYPE, -- attributeGroup timePropertyGrp
  mediaTimeBase SYS.XMLTYPE, -- attributeGroup mediaTimePropertyGrp
  mediaTimeUnit SYS.XMLTYPE -- attributeGroup mediaTimePropertyGrp
) NOT FINAL;
/

```

## DB schema CreationInformation-Part derived from MPEG-7 MDS

Package 2.6  
Author: Mario Doeller, Alexander Bachlechner  
Last modified: 15.4.2003

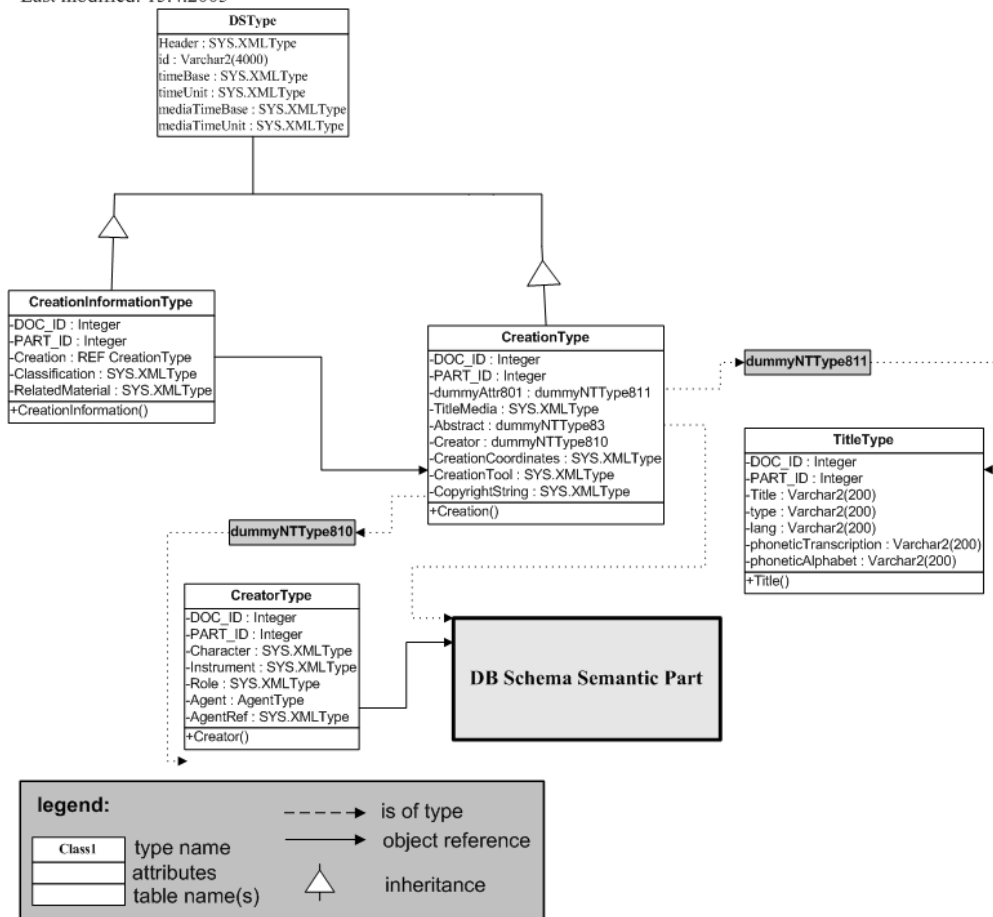


Figure 8.11: DB Schema - CreationInformation Part

# DB schema Image-Part derived from MPEG-7 MDS

Package 2.6  
Author: Alexander Bachlechner, Mario Doeller  
Last modified: 16.4.2003

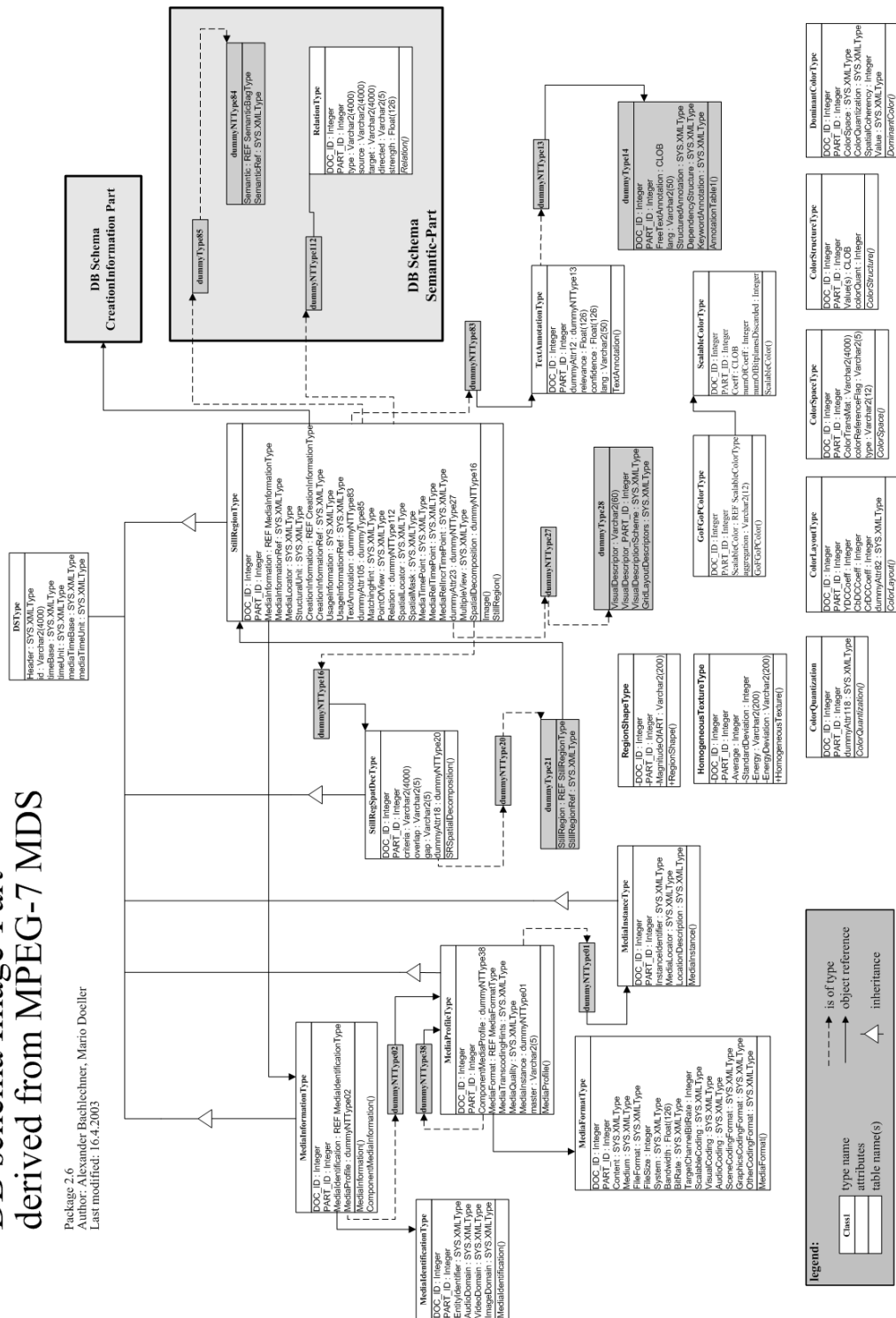


Figure 8.12: DB Schema - Image Part



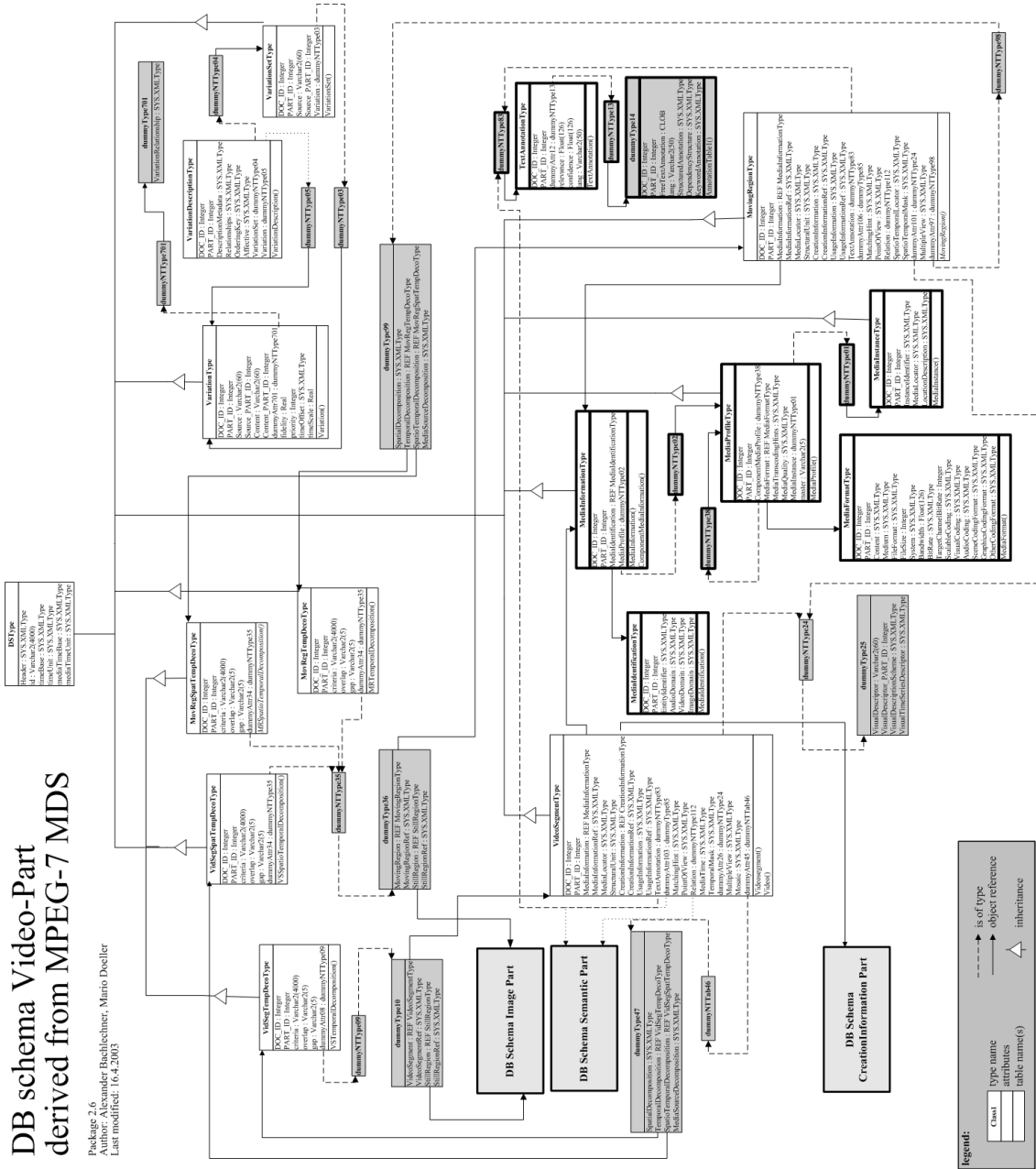


Figure 8.14: DB Schema - Video Part

# DB schema Audio Visual-Part derived from MPEG-7 MDS

Package 2.6  
 Author: Mario Doeller, Alexander Bachlechner  
 Last modified: 12.3.2003

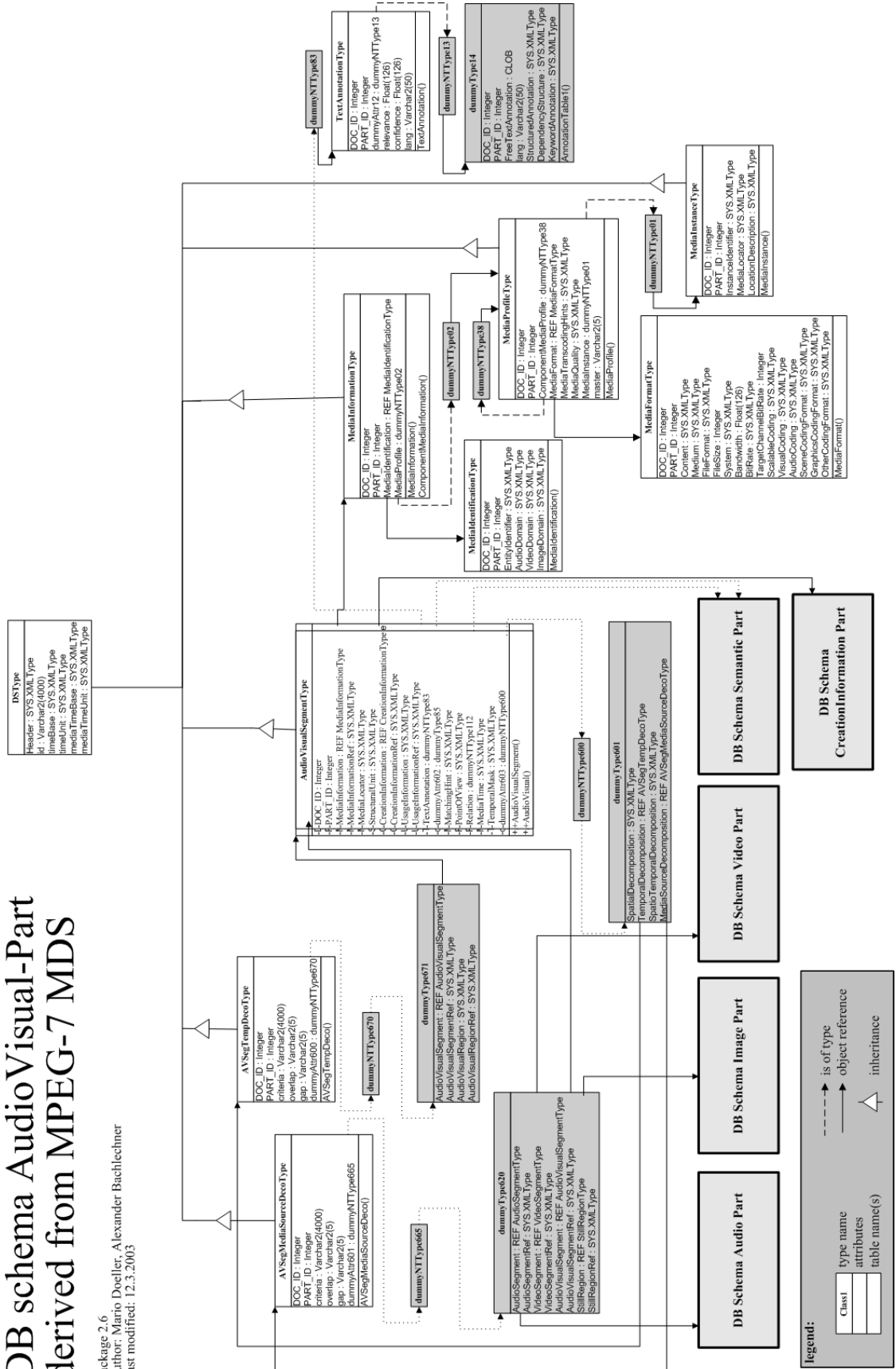


Figure 8.15: DB Schema - Audio Visual Part



### DB schema SEMANTICS-Part derived from MPEG-7 MDS

Package 2.6  
Author: Alexander Bachlechner, Mario Doeller  
Last modified: 16.4.2003

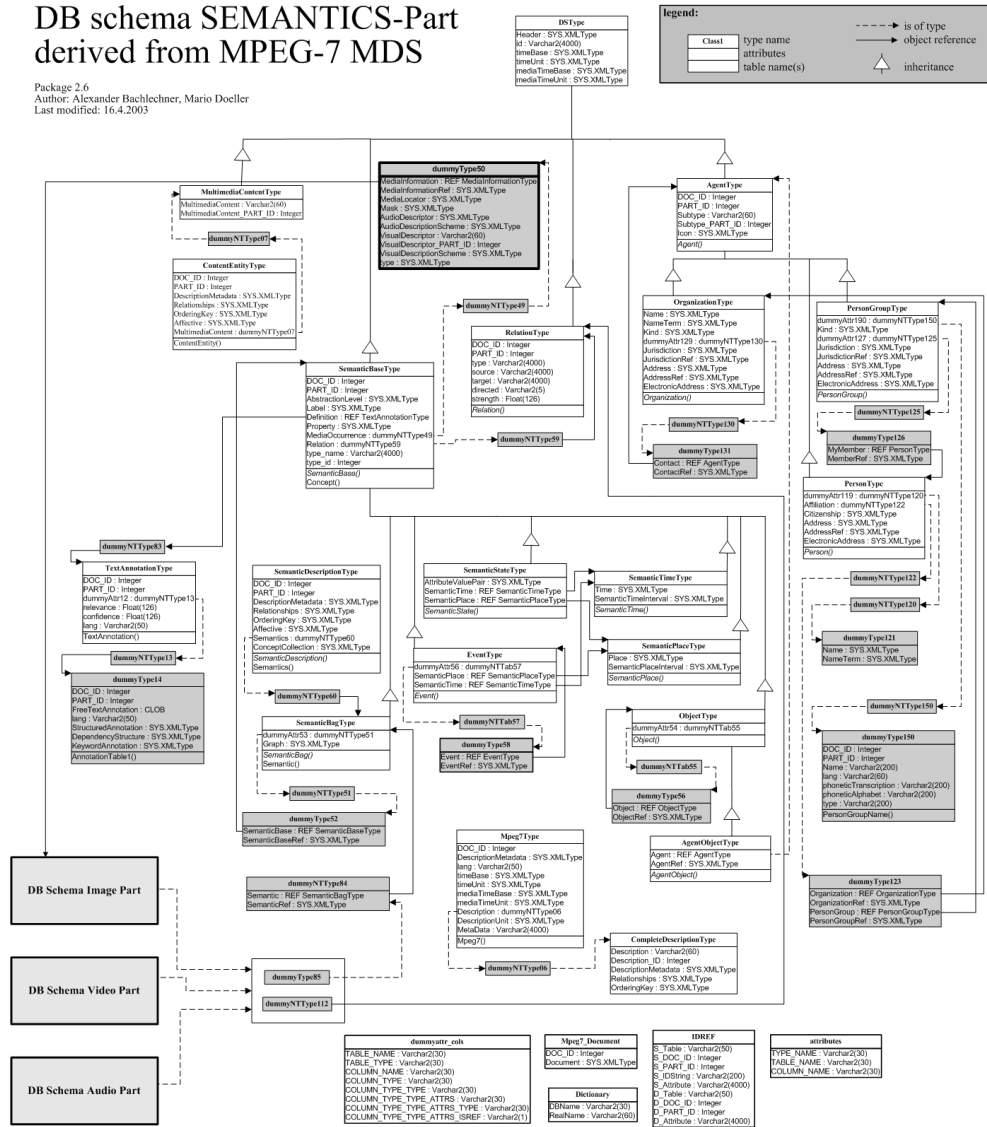


Figure 8.16: DB Schema - Semantic Part

## 8.3 Index Processing

Indexing is an important concept in modern database management systems to enhance processing efficiency. It is used to index attributes of tables to increase their access performance. Innately, most database systems provide only a limited number of integrated access methods such as a B-tree or hashing facilities. These access methods are only capable to handle uni-attributes such as integer or strings. These limited techniques restrict the usability of database systems in modern applications such as Multimedia, E-Commerce or Geographic Information Systems (GIS). Especially, they rarely handle indexing of multi-attributes such as  $d$ -dimensional data ( $d > 2$ ) which is common in multimedia applications. It is not an unusual need to manage images which are represented as color histograms with a dimensionality of 64 or higher (see Chapter 2). In this context, we provide an indexing framework for access methods which is integrated in our MPEG-7 MMDB. In detail, Subsection 8.3.1 describes the main parts of our multimedia indexing framework and their possible use. Furthermore, in Subsection 8.3.2, we briefly introduce the GIST framework which is integrated in our system. We conclude this Section by giving several experimental results (see Subsection 8.3.3) of comparing our solution to build-in access methods of Oracle.

### 8.3.1 Multimedia Indexing Framework (MIF)

The Multimedia Indexing Framework (see Figure 7.1 number 1 and 4) comprises three modules. Each module may be used on its own and may be distributed over the network.

#### 8.3.1.1 GistService

The GistService (see Figure 8.17) is realized in the external address space and is implemented in C++. It runs as an own process (called service) in the Windows Operating System environment and manages all available access methods. The current version offers support for Generalized Search Trees (GiST, see Section 8.3.2) and further access methods not relying on balanced trees (e.g., LPC-files [56] to support NN-Search in high-dimensional vector spaces).

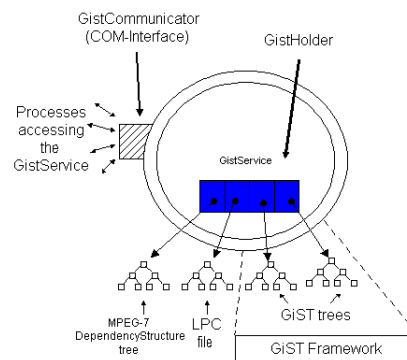


Figure 8.17: GistService

The service is split into two main components: The *GistCommunicator* and the *GistHolder*. The *GistCommunicator* is a COM-object (Component Object Model) which offers services through an IDL interface (Interface Definition Language). It is used for inter-process communication between the database (the *GistWrapper* shared library) and the implemented access methods. Thus, the *GistCommunicator* supplies the necessary functionality (e.g., creating, inserting, deleting) for accessing the index structures. The result of the operations are forwarded to the database. It is the task of the *GistHolder* to manage the currently running index trees and accesses to them. Each index tree is identified through a global and unique ID which is forwarded to the accessing process. For simplicity, the index trees are internally stored in an array, but this data structure can be replaced easily by any other more dynamic structure.

### 8.3.1.2 GistWrapper

The *GistWrapper* module is a shared library, in C++ implemented, that is used by the database to connect to the *GistService*. The library has two main tasks. First, it makes the *GistService* accessible for database procedures. An Oracle database has the possibility to call external C/C++ code via shared libraries. The *GistWrapper* acts as a wrapper for the *GistService*. The second task is the transformation of the input and output data to make it usable for both the *GistService* and the database. For instance, a simple C Char type has to be transformed into a BSTR string, or a VARIANT type into a String and so on. The *GistWrapper* module offers a similar

interface as the *GistService* module.

### 8.3.1.3 Multimedia Index Type

The multimedia index type consists of several *indextypes*, their corresponding *operators* and the appropriate implementation (*objects*). We will illustrate the integration process with the example of an R-tree.

For this purpose, we have to create a database library that links to a shared library implementing the *Gistwrapper* functionality. Further, one has to register all functions (in our example we show the *init* function) of the shared library to use them internally.

```
CREATE LIBRARY libgist_lib AS '$ORACLE_HOME/bin/gistwrapp.dll';
/

CREATE OR REPLACE FUNCTION PLS_libInit RETURN BINARY_INTEGER AS
  EXTERNAL LIBRARY libgist_lib
  NAME "gistlibInit"
  LANGUAGE C
  PARAMETERS(return int);
/
```

The next step is to create an user-defined object that implements all necessary index methods supported by ODCI (Oracle Data Cartridge Interface). In our case, the object is termed *MDCGistIndex* and offers implementations for the following methods (*ODCIGetInterface*, *ODCIIndexCreate*, *ODCIIndexDrop*, *ODCIIndexInsert*, *ODCIIndexDelete*, *ODCIIndexUpdate*, *ODCIIndexStart*, *ODCIIndexFetch*, *ODCIIndexFetch*). The following code fragment shows a cut-out of the specification of several methods pointed out above. The decoupling of method specification and implementation, described in Subsection 8.1.1 has been applied by the *ODCIIndexCreate* and the *ODCIIndexDrop* method. The first one is implemented with the help of an internal JAVA method and the other one is implemented in PL/SQL. All PL/SQL implementations are collected in the BODY (not shown in this paper) of an database object. The JAVA class in turn calls via PL/SQL functions (see above) their equivalent C routine for accessing the external *GistService*.

```

CREATE OR REPLACE TYPE MDCGistIndex AS OBJECT (
    scanctx integer,

    STATIC FUNCTION ODCIIndexCreate (ia sys.odciindexinfo, tr_tp_dim varchar2)
        RETURN NUMBER AS LANGUAGE JAVA NAME
'at.mdoeller.mdc.mdcindex.mdcgistindex.MDCGistIndex.ODCICreate(oracle.ODCI.ODCIIndexInfo,java.lang.String) return

    STATIC FUNCTION odciindexinsert(ia sys.odciindexinfo, rid VARCHAR2,
        newval VARCHAR2)
        RETURN NUMBER AS LANGUAGE JAVA NAME
'at.mdoeller.mdc.mdcindex.mdcgistindex.MDCGistIndex.ODCIInsert(oracle.ODCI.ODCIIndexInfo, java.lang.String,
    java.lang.String) return java.math.BigDecimal',

    STATIC FUNCTION ODCIIndexDrop(ia sys.odciindexinfo) RETURN NUMBER
);
/

```

The last step in creating an own index is the definition of an own *indextype*. Each *indextype* needs some *operators* that are offered by this type. Example of operators which we realized are the following:

- `rt_equal_point(VARCHAR2, VARCHAR2, NUMBER, NUMBER);`
- `rt_nearest_point(VARCHAR2, VARCHAR2, NUMBER, NUMBER);`

The first operator defines an equality search and the second one a NN-Search for point data. The parameters are defined as follows: 1) element in the database table, 2) search item, 3) amount of results, and 4) the dimension. Note, that in this example the *indextype* can only be used for attributes of type *VARCHAR2*. For all other types, one has to create separate operators, indextypes and their respective object specification and implementation. The resulting R-tree indextype for point data is shown below.

```

CREATE OR REPLACE INDEXTYPE rtree FOR

rt_equal_point(VARCHAR2, VARCHAR2, NUMBER, NUMBER),
rt_equal_rect(VARCHAR2, VARCHAR2, NUMBER, NUMBER),
rt_nearest_point(VARCHAR2, VARCHAR2, NUMBER, NUMBER),
rt_nearest_rect(VARCHAR2, VARCHAR2, NUMBER, NUMBER),
rt_overlap_rect(VARCHAR2, VARCHAR2, NUMBER, NUMBER),
rt_contains_rect(VARCHAR2, VARCHAR2, NUMBER, NUMBER),
rt_contained_rect(VARCHAR2, VARCHAR2, NUMBER, NUMBER)

USING MDCGistIndex;

```

The use of the newly created R-tree index type is illustrated in the following example. Let us assume, we want to index two-dimensional point data that is stored in the table *t1* at column *f2*. Then we create an index, named *it1*, that uses our new index type *rtree*. As parameter, one has to specify the type of the data *rt\_point* and the respective dimension (2). The necessary statements are demonstrated below.

```
CREATE TABLE t1 (f1 number, f2 varchar2(300));

CREATE INDEX it1 ON t1(f2) INDEXTYPE IS rtree PARAMETERS('rt_point 2');
```

The point data is given as a string and has to be inserted as follows.

```
INSERT INTO t1 VALUES (1, '4.234 2.45345');

INSERT INTO t1 VALUES(2, '3.45 1.3245');

INSERT INTO t1 VALUES (3, '3.565 1.3245');
```

The stored data can now be queried with the enhanced functionality such as equality (*rt\_equal\_point(...)*) or NN-Search (*rt\_nearest\_point(...)*) of the R-tree structure (see below). The parameters are specified as follows: (1) the column of the table that has to be queried (*f2*), (2) the coordinates of the spatial construct which is in our case a two-dimensional point (e.g., *'3.45 1.3245'*), (3) the maximum amount of resulting tuples (e.g., *10*) and (4) the dimension of the reference data (2). The gain in this case is twofold. First, the use of the new equality search increases the search efficiency compared to a full table scan. Second, a NN-Search extends the existing database functionality.

```
SELECT * FROM t1 WHERE rt_equal_point(f2,'3.45 1.3245',10,2) = 1;

SELECT * FROM t1 WHERE rt_nearest_point(f2,'5.5 1.5',4,2) = 1;
```

### 8.3.2 GiST Framework

As mentioned above, our indexing framework, MIF, relies partially on the *GiST framework* proposed by J. H. Hellerstein and his group at the University of California, Berkeley.

The GiST framework is widely used in the CBR research [79, 1], but yet this framework has been integrated only once into an extensible DBMS [80], which does not support non-balanced trees. In addition, it is important to note that there has been extensive testing of the framework which is reported in [76], also in combination with an CBR prototype called Blobworld [7, 1]. The efficiency of the GiST framework has been demonstrated in these previous works, yet its integration into a DMBS, together with the important multimedia extensions we made, has to be carefully reevaluated for efficiency which is detailed in the next Section 8.3.3.

The GiST framework enables a developer to plug-in new balanced tree implementations into the framework and to test their performance. This is done by implementing some specific methods for insertion, deletion and search. The current GiST version, 2.0, already includes source code<sup>1</sup> for the R-tree, R\*-tree, SS-tree, SR-tree, SP-tree and B-tree.

Generally spoken, a GiST is a balanced tree with (key, RID) pairs in the leaves and (predicate, child page pointer) pairs as internal nodes. The framework and their corresponding trees have no restrictions on the key data stored within the tree or on their organization within and across nodes. Once a new access method is set up, the balance of subtrees during insertion and deletion may be improved by an additional tool in the GiST framework environment, the *amdb* [78].

The current GiST framework (version 2.0) has some shortcomings. First of all, only one index tree may be used at time. Second, no support for additional non-balanced tree indexes is given.

These shortcomings have been disposed by our *GistService* that allows the management of more than one access method at time (realized through the *GistHolder* component which is introduced in Section 8.3.1.1) and the possibility to integrate non-balanced access methods such as LPC-file (local polar coordinate file). The latter has been realized by establishing a new top-level interface that every access method has to implement.

---

<sup>1</sup><http://gist.cs.berkeley.edu/>

### 8.3.3 Experimental Results

This Section describes a significant part of the series of experiments we performed in order to evaluate the effectiveness of our *Multimedia Indexing Framework* (MIF). The tests were carried out on two distinct datasets, one *synthetic (uniform dataset)* and one *real*. The experimental settings are as follows:

- The synthetic dataset contains 64- and 96-dimensional feature vectors that are represented as strings. The values were generated uniformly over the normalized  $[0..1]$  space.
- The real dataset was generated from an 1 hour and 46 minutes long movie, encoded with DIVX4. From the movie, we extracted 64-dimensional color histogram of 200000 frames of size 352x288 pixel, by retaining the two most significant bits in the RGB space. The generated feature vectors were inserted into the database. Further we separated color histograms of 100 frames for the query process.

In order to compare the built-in indexing mechanism and our MIF we had to employ exact match queries. The remaining supporting MIF retrieval functions, like Range-Search, NN-Search and Overlap-Search, are in addition to the build-in index functions and can not be used for comparison.

The retrieval was carried out through server-sided JDBC, i.e., the java class resides in the database and is executed through Oracle's own JVM (Java Virtual Machine). The insertion was accomplished through a java class that resides outside of the database and was connected to the database through thin JDBC. In both cases, we always used the same java classes for the measurements to have the same measurement basis.

#### 8.3.3.1 Indexing Details

The feature vectors are stored in a column of type *Character Large Object (CLOB)*. We had to use the *CLOB* data type, because of the high dimensionality of the underlying data. As a consequence, the built-in B-tree can not be used, as it does not handle CLOBs. Instead, we used the *Oracle Text Index* [170] which is able to index



CLOB string representations without severe limitations. The indexing engine of the Oracle Text Index creates an inverted index that maps tokens to the documents that contain them. The inverted index is a 'list' of the words from the document, with each word having a list of documents in which it appears. In our case the words are points in the various dimensions. Based on these index techniques we compared the response time between a normal (no index) solution, the Oracle Text Index and *MIF*. In *MIF*, we used the well known R-tree. The response time was measured for insertion and select operations. The select operation was limited to exact match queries, because of the limited functionality of current database indexing mechanisms. As mentioned above this shortcoming can be compensated with *MIF*.

### 8.3.3.2 Detailed Results

The comparison of the *MIF* using R-trees and Oracle 9i built-in Text Index [170] shows that the *MIF* based trees endue less insertion efficiency (due to the external proc calls), but offer significantly higher retrieval performance.

**Indexing – Synthetic Dataset** The following Figures show the results for the insertion process: left hand side of Figure 8.18 for 64-dimensional point entries and right hand side for 96-dimensional point entries. These Figures demonstrate that the *MIF* has a higher insertion time than the related solutions. The extra time for the *MIF* is caused by the overhead from switching and transferring the data between the Oracle address space and the external address space. This overhead does not penalize the *MIF*, because insertion are rare in our mostly read-only application context, and second the insertion time does not explode even for a large data set.

It has to be noted that the memory consumption of the *Oracle Text Index* is enormous. The table space consumed is in the worst case over 3.8 GB for an insertion operation of 200.000 point elements with 96 dimensions. Compared to this, the memory consumption of the *MIF* is significantly smaller, e.g., for the same insertion operation as above, it requires only a table space of about 220 MB.

**Retrieval Query – Synthetic Dataset** The results for the query evaluation show that our framework *MIF* outperforms clearly the related solutions.

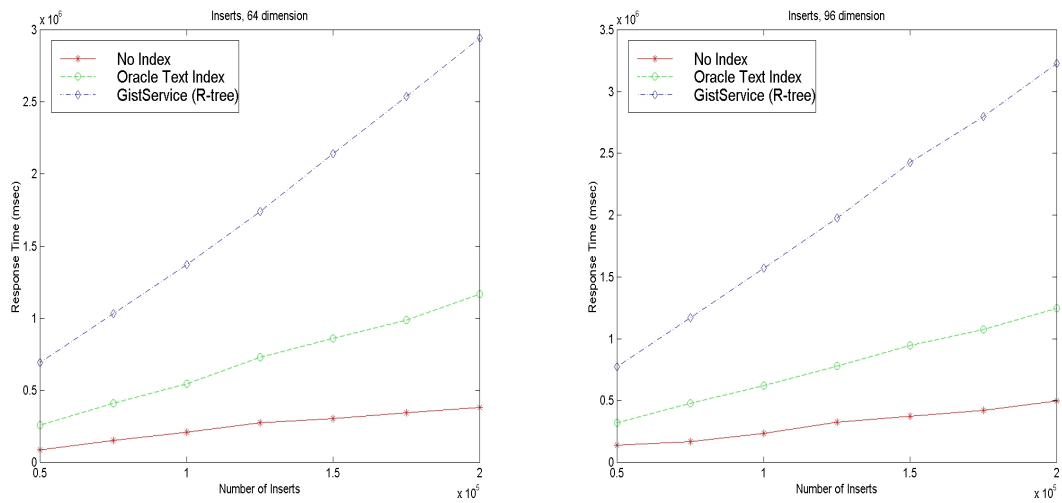


Figure 8.18: Response Time of the Insert Statements (50000 to 200000 points with 64 dimension (left) and 96 dimension (right))

The left hand side of Figure 8.19 shows that for exact match queries involving 64 dimensions, the *MIF* environment outperforms clearly both related solutions. The Figures show the mean value of 5 measurements including each 100 select statements. A sequential search is, of course, significantly slower. Positively, the *MIF* environment is from 6 to 7 times faster than the Oracle Text Index for 175.000 data entries of 65 respectively 96 dimensions. Furthermore, MIF performs from 85 to 134 times better than the no index solution for the same amount of indexed elements.

**Indexing and Retrieval – Real Dataset** The response time for the insertion process with the real dataset was similar to the results obtained from the synthetic dataset test series and is presented in Figure 8.20 (left side). The response time is again measured as the mean value of 5 measurements each including 100 select statements. The response time of the sequential scan was very similar to that measured for the synthetic data set (see Figure 8.19). Moreover, in reason of the great discrepancy between the index and non-index solutions, the results of the non-index solution are not presented in this Figure.

Figure 8.20 shows that the Oracle Text Index performs worse than in the previous test series. On average, we increased our efficiency by a factor of 6 compared to the

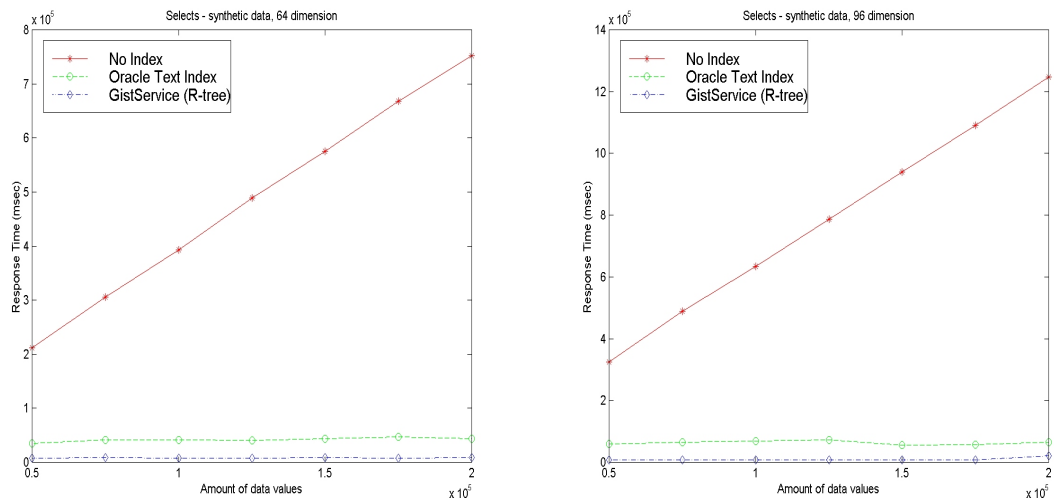


Figure 8.19: Response Time of Select Statements of points with 64 dimension (left) and 96 dimension (right)

synthetic data which leads to an overall improvement of 46 times faster than the Oracle Index (and this even with the call to an external address space) for 175000 indexed elements. Contrarily to the uniform dataset from above, the color histogram derived from this special movie contains far more zero values. This fact seems to shorten the use of the Oracle Text Index significantly.

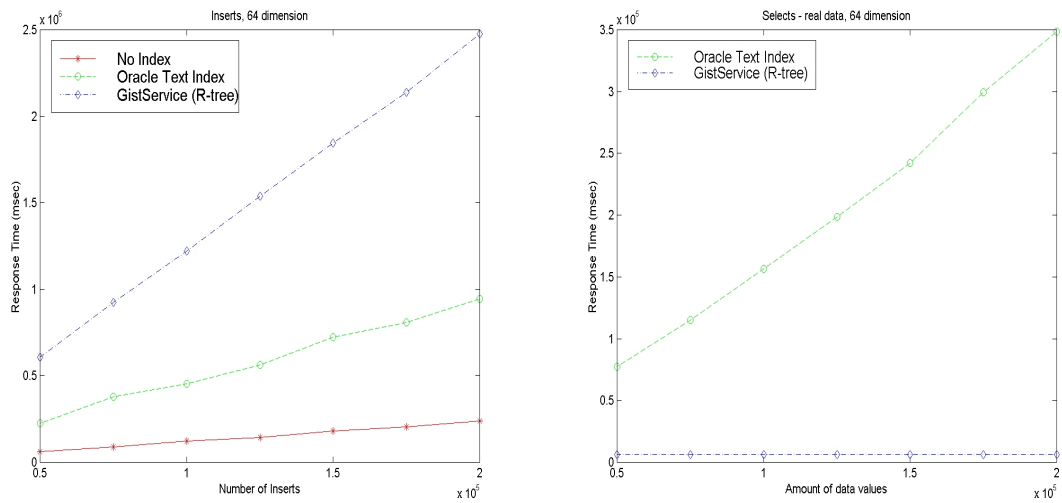


Figure 8.20: Response Time of Inserts and Select Statements for color histograms (real dataset) with 64 dimension

## 8.4 Query Optimization

Nearest-Neighbor and Range queries are very common in multimedia databases. The performance of such queries can be increased by the use of appropriate index structures and by the use of a query optimizer that takes care of the optimal query plan (see Section 4.3 for a detailed explanation of the task of a query optimizer). A cost-based query optimizer uses two factors for finding the optimal query plan: on the one hand, there is the *cost* value for an operation which is composed of: the amount of CPU cycles that is necessary to perform the operation, the used network bandwidth and the number of necessary page accesses; on the other hand, the *selectivity* of an operation is defined as the ratio of output relation versus input relation of the operation. The cost of index structures for NN- and Range Searches is well investigated [91, 92, 93, 171]. The estimation of a Range query's selectivity represents, apart from few initial approaches [94, 95] (see Section 4.3 for their limitations), an open research question.

In this context, we introduce an approach for approximating the selectivity of Range Searches for n-dimensional feature space. This Section is organized as follows: Subsection 8.4.1 introduces several definitions in the area of Range queries and

states our main motivation. Related work about query optimization and clustering is handled in Subsection 8.4.2. Our approach of approximating the selectivity is formulated in Subsection 8.4.3. Subsection 8.4.4 depicts the resulting architecture. A cluster visualization tool for evaluation and testing is presented in Subsection 8.4.5. The evaluation of our approach with 2-dimensional coordinates of cities within Austria and Germany and a 8-dimensional color histograms of images is discussed in Subsection 8.4.6.

### 8.4.1 Motivation and Definitions

The selectivity of a database operation (e.g., selection, join, etc.) is defined as the fraction of tuples of the (participating) relation(s) that is selected by the operation. This information is used by the query optimizer to create an efficient query plan [96]. The introduction of new operations for multimedia data such as k-NN or Range searches in multimedia databases leads to the investigation of their selectivity in order to support the query optimizer in finding the optimal query plan. Whereas the prediction of the selectivity for (k-)NN searches is trivial (1 or k), it is hard for a Range query. Throughout this Section, we use the following notation:  $R(A)$  and  $S(B)$  are relations with their corresponding set of attributes  $A = \{A_1, \dots, A_n\}$  and  $B = \{B_1, \dots, B_n\}$ . The number of tuples in a relation is denoted by  $|R|$ .

The SELECT operation, formally written as  $\sigma_{\langle select\_condition \rangle}(relation)$ , is used to select a subset of the tuples in the given relation that satisfy the  $\langle select\_condition \rangle$ . In case of an equality condition, it applies to:  $\sigma_{A_i=a}(R) = \{t \in R | t(A_i) = a\}$  where  $A_i=a$  denotes the select condition,  $t(A_i)$  represents the value of the attribute  $A_i$  within the tuple  $t$  and  $t$  is a tuple within the relation  $R$ . The selection selectivity is defined as the fraction of tuples of the relation  $R$  selected by the selection condition [96].

The EQUIJOIN is a JOIN operation where the join condition is restricted to the equality operator. Formally, the EQUIJOIN is written as:  $R \bowtie_{\langle A_1=B_1, \dots, A_j=B_j \rangle} S$  with  $j \leq n$  which results in a relation  $Q$  that satisfies the following condition:

$$Q(A \cup B) = \{t | \exists t_S \in S \wedge \exists t_R \in R \text{ with } t(A) = t_R(A), t(B) = t_S(B) \wedge t(A_i) = t(B_i), 1 \leq i \leq j\}$$

An efficient implementation of the EQUIJOIN uses the HASH-JOIN [96] algorithm where the tuples of R and S are both hashed to the same hash file using the same hashing function. The in-memory version works as follows: First, each tuple of the relation with the expected smaller cardinality is inserted into a hash bucket depending on the computed hash value. Second, a single pass through the other relation hashes each tuple based on the calculated hash key to the corresponding hash bucket, where it is combined with all matching tuples of the first relation. The join selectivity is determined by the following formula [96]: *join selectivity* =  $\frac{|Q|}{|R|*|S|}$  where  $|Q|$  is the expected size of the join result.

The definition of commonly used query types in multimedia databases is given in Section 4.2.1. One of them is the *Range Query* operation which definition is repeated for clearness below:

The *Range Query* returns all points  $P$  that have a smaller or equal distance  $r$  from the query point  $Q$  with respect to the used metric  $M$ .

$$RangeQuery(DB, Q, r, M) = \{P \in DB | \delta_M(P, Q) \leq r\}$$

$M$  is a similarity metric like the Euclidean metric (discussed in Section 4.2.1).  $DB$  denotes the database which is assumed to contain n-dimensional data points.

The importance of a precise selectivity estimation is shown in the following example:

Let us assume, that we have the following two relations: The first relation, named *city*, contains the attributes ID and city NAME:  $CITY(ID, NAME)$ . The second relation, named *location*, contains the attributes ID, CITYID (foreign key ID of the relation *city*) and LOCATION given as 2-dimensional coordinates:  $LOCATION(ID, CITYID, LOCATION)$ . Both relations contain 1000 elements.

In the following, we are interested in all cities that are in a specific area and whose name starts with an A. This would result in the following query:

*SELECT c.name FROM location l, city c WHERE rangeQueryOp(l.location, '50.0 10.0', 2, 0.22) = 1 AND c.name LIKE 'A%' AND l.cityid = c.id;*

If we now suppose, that the selectivity of Range Search operation (*rangeQueryOp*) is 4 and the selectivity of the selection by the LIKE operation is 400 then the optimal query plan is given at the left side of Figure 8.21. The right side of the Figure 8.21 shows the query plan which would be created if the selectivity is not known a priori. In this case, a default selectivity has to be assumed which results in an inefficient query plan (see Subsection 8.4.6 for the resulting differences in measured time). The inefficiency is reasoned by the HASH JOIN operation which requires for an optimal query performance the processing of the operation with the smallest cardinality first [172]. This cardinality is determined by the selectivity of those selection operation, executed before hand.

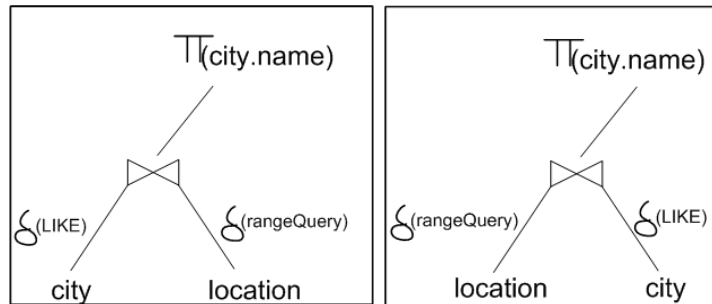


Figure 8.21: Optimal Query Plan (left) Inefficient Query Plan (right)

## 8.4.2 Related Work

In this Subsection, we concentrate on related work about clustering techniques as this is one of the main parts of our approach. Additional related work about query optimization or index structures can be found in Section 4.3 respectively Section 4.2.

In recent years, a number of clustering algorithms have been introduced [173, 174, 175, 176]. In addition, there exist several excellent surveys on clustering [177, 178, 179, 180, 181]. In general, the various approaches can be classified by the following main groups [177, 178]:

- **Hierarchical Methods** create for the given data set a tree of clusters also known as *dendrogram*. The existing approaches in this domain can be categorized into agglomerative (bottom-up) and divisive (top-down) procedure. The main difference is their respective starting point. The agglomerative clustering starts with one data element (singleton) cluster and merges two or more appropriate clusters. Whereas the divisive approach starts with one cluster that contains all data elements and recursively splits the most appropriate cluster. An example algorithm of this approach is CURE (Clustering Using REpresentatives) [173]
- **Partitioning Methods** divide the data space in several subsets. In general, there exist two approaches: the *k-medoids* and *k-means* technique. A cluster in the k-medoid technique is represented by its most appropriate point. This point is determined by a predominant fraction of points within the cluster. In the k-means technique, a cluster is represented by its centroid, which is extracted by the mean of points within the cluster. In general, the k-means technique has achieved a broader spread.
- **Density-based Methods** uses the fact that the density of data elements within a cluster is considerably higher than outside of the cluster. The main representative of this approach is the DBSCAN [174] algorithm. In the following this algorithm is described in more detail, as it is used within our selectivity approximation approach. DBSCAN identifies three different classes of points. *Core points* of a cluster are points that are encircled by at least a minimum number of points (defined through *MinPts*) within their neighborhood (given by the radius *eps*). In contrast to them, *Border points* are at the border of a cluster, which means that at least one *core point* is in their *eps*-neighborhood. *Noise points* are those points that do not belong to any cluster. The key idea of cluster finding in DBSCAN is the following: The algorithm starts with an arbitrary point  $p$  and retrieves all points that are density-reachable with respect to *eps* and *MinPts*. A point  $q$  is density-reachable from a point  $p$  if there exist a chain of points  $p_1, \dots, p_n$ ,  $p_1 = p$ ,  $p_n = q$  such that  $p_{i+1}$  is within the *eps*-neighborhood of  $p_i$ . If  $p$  is a core point then this procedure yields a cluster.



If  $p$  is a border point then DBSCAN proceeds with the next point of the data set.

- **Grid-based Methods** divide the data space in respect to the underlying space in cells (regions, grids). Cells that contain more than a certain number of data elements are identified as possible candidates for a cluster. One advantage of this approach is their independence of data ordering. One of the most important grid-based techniques is the CLIQUE [175] (CLustering In QUEst) algorithm.

Additional approaches, such as fuzzy clustering algorithm or model based algorithm can be found in [177] respectively [178]. A large sample of clustering papers can be found at: [http://prlab.ee.memphis.edu/frigui/cluster\\_paper.html](http://prlab.ee.memphis.edu/frigui/cluster_paper.html).

### 8.4.3 Approach

As stated before, a fast prediction of the selectivity for a Range query is hard to determine and highly depends on the underlying data set. The selectivity is related to the notion of density in the neighborhood of the query point, for instance, if the density around the query point is high, we have a high selectivity, otherwise a low density will lead to a small selectivity.

In this context, we introduce an approach for approximating the selectivity of Range-Searches within a  $n$ -dimensional data set with the help of a density based clustering technique (in our case DBSCAN). The DBSCAN algorithm was used because it is a density based and wide spread clustering technique. In addition, the *eps*-value of the algorithm is an important component of our approach and allows us to optimize the cluster allocation for each query circle radius. The solution finding for approximating the selectivity of Range queries contains two attempts. First, we realized our initial thoughts of approximating the selectivity by the use of a density based clustering technique. Based on the identified weaknesses (among others the approximated selectivity shows only an acceptable failure for a small range of query circle radii), we come up with an advanced approach that removes the main problems of the first one. In example, the best *eps*-value is determined for a discrete query circle radius (depending on the given query circle radius interval and the appropriate intermediate step), so that the cluster allocation can be adapted and hence, the

failure decreases.

### 8.4.3.1 First Approach

The key idea of our first approach includes the following steps:

- 1.) cluster the data set with the help of a density based cluster technique (in our case DBSCAN [174]).
- 2.) specify the (a) MBR (minimum bounding region) or (b) CH (convex hull) for each cluster  $C$ .
  - a.) In the first approach we used the MBR for calculating the volume of every cluster. The main advantage is the easy calculation and low space consumption for maintaining the list of clusters. The major disadvantage is the possible overlap with other clusters and a possible inaccurate value for the volume if the cluster's shape is not compact.
  - b.) The use of a CH for a cluster decreases the possibility of overlapping MBRs and therefore, achieve a better approximation of the selectivity. Unfortunately, we investigated during our evaluation, that the improvements are minimal with respect to the disadvantages of CH, namely time consuming task in identifying the CH for a cluster (in contrast to the MBR) and an increased storage consumption. For example, in average we had for the MBR approach a relative failure of 23,52% (15.57%-39.59%) and for the CH approach a relative failure of 23,56% (15.55%-39.70%) for a query circle radius interval of 10 - 40 in the synthetic test-bed (2000 points). Therefore, all provided results concern the MBR approach.
- 3.) calculate the density of all clusters with:

$$Density(C) = \frac{\#P \text{ of } C}{Vol(C)} \quad (8.1)$$

where  $Vol(C)$  is the hypervolume of the surrounding MBR or CH and  $\#P$  denotes the number of points that are in the cluster. Note: We assume uniform distribution within a found cluster.

- 4.) approximate the selectivity of a query point  $Q$  with radius  $r$  with the following formula:

$$\text{approx. selectivity} = \begin{cases} \text{Density}(C_i) * \text{Vol}(Q) & , Q \in \text{Area}(C_i) \\ \text{MinPts} - 1 & , \text{otherwise} \end{cases} \quad (8.2)$$

where  $\text{Vol}(Q)$  is the hypervolume of the surrounding sphere defined through the query point  $Q$  and radius  $r$ . The approximated selectivity is determined by  $\text{Density}(C) * \text{Vol}(Q)$  if and only if the query point  $Q$  falls in the area of cluster  $C$ . The area of the respective cluster is specified by the MBR. If the query point  $Q$  falls in none of the given clusters, then the approximated selectivity is defined by  $\text{MinPts} - 1$ . The  $\text{MinPts}$  value is equivalent to the DBSCAN  $\text{MinPts}$  value and specifies the absolute minimum amount of points a cluster has to have. Therefore, we can argue that, if the query point does not fall in a predefined cluster then the density around the query point is rather low.

We used several data sets for the evaluation of our approach (see Subsection 8.4.6 for a detailed explanation). Figure 8.22 shows the behavior and also the weakness of the first approach. The data set contained 2000 8-dimensional color histograms which were extracted from images of size 128x192 pixels in the RGB space. The values are within the [0..1] interval and are clustered with DBSCAN with an *eps* value of 0.1 and an *MinPts* value of 5. The *eps*-value defines the maximal distance of participating points within a cluster. We compute the selectivity for all points in the data set and compared the real selectivity with the approximated selectivity. The red line in Figure 8.22 shows the average real selectivity over all 2000 points with respect to the query circle radii (X-axis) from 0.04 to 0.16. The Y-axis represents the average query result over all 2000 points and ranges from 4 for a radius of 0.04 up to 5 for a radius of 0.16. The blue line in Figure 8.22 displays the average failure over all 2000 points which was calculated with the following formula:

$$\text{failure} = \frac{\sum_{i=1}^n |\text{RealSel}(Q_i) - \text{ApproxSel}(Q_i)|}{n} \quad (8.3)$$

where  $Q_i$  is the  $i$ -th point of the data set used as query point.  $RealSel(Q)$  is the result of the real selectivity and  $ApproxSel(Q)$  is the approximated result of our approach. The failure ranges from 1.57 for a radius of 0.04 up to 5.56 for a radius of 0.16. The Figure shows that for a query circle radius which is near to the  $eps$ -value (0.1), the approximated result has only a failure of .52 which is 12% of the average real result. Unfortunately, the failure between real and approximated selectivity for the other query circle ranges is about 50% which is unacceptable.

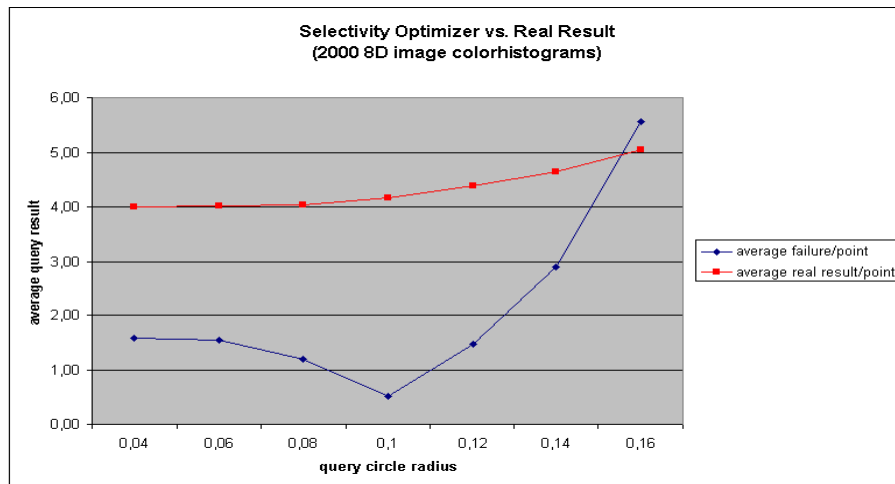


Figure 8.22: Selectivity Results of the First Approach

### 8.4.3.2 Second Approach

The main weakness of the former approach was its usage of the same  $eps$ -value and therefore the same cluster allocation for each query circle radius. This results in a good performance for a small range of query circle radii and a worse performance for the rest (see former experimental results at Figure 8.22). Hence, the new algorithm identifies and assigns the failure-minimal cluster allocation to the corresponding query circle radius which leads to an acceptable performance for the whole range of given query circle radii. The major changes in contrast to the previous approach concern the first step. In the following, all steps of the previous approach are traversed and the necessary changes are explained.

- 1.) At first, the algorithm identifies the failure-minimal cluster allocation for all

discrete query circle radii depending on the given interval and the appropriate incremental factor. The input (given by the user) for the algorithm is the minimum query circle radius ( $min\_qcr$ ), the maximum query circle radius ( $max\_qcr$ ) and the incremental factor  $incr\_factor$ . This information relies on the underlying data interval and is used to decrease the size of the search space. Figure 8.23 shows the architecture and all necessary steps for fulfilling the first phase.

Step 1: Here, all discrete query circle radii of the given query circle radius range are determined. For instance, for the following given data:  $min\_qcr=0.04$ ,  $max\_qcr=0.10$  and  $incr\_factor=0.01$ , we result in 7 intermediate radii (0.04, 0.05, ... , 0.10 ). If we increase the  $incr\_factor$  to 0.02 the amount decreases to 4 (0.04, 0.06, ... , 0.10).

Step 2: Starting from the given  $min\_qcr$ ,  $max\_qcr$  and  $incr\_factor$  values, we evaluate the necessary  $min\_eps$  and  $max\_eps$  values for computing the different cluster allocations. The  $min\_eps$  value is composed of  $min\_qcr - to\_add$ , respectively the  $max\_eps$  value is determined by  $max\_qcr + to\_add$ . The  $to\_add$  value is calculated with the following formula:

$$to\_add = \frac{max\_qcr + min\_qcr}{2} * 0.7 \quad (8.4)$$

The  $min\_eps$ -value can not be negative and the  $max\_eps$ -value has its upper boundary when all points are in one cluster. In addition, as in step 1, all necessary intermediate  $eps$ -values are determined.

Step 3: In the last step, the best cluster allocation for a certain discrete query circle radius is determined. This is done by evaluating the failure of all cluster allocations (see Equation 8.3). For this purpose, the second and third task, namely specify the MBR (task 2) and the density (task 3) for each cluster of the previous approach are integrated in this step. Then, the failure is calculated by summing up the differences between the approximated selectivity and the real selectivity for all points within the data set. The cluster allocation with the minimum failure is assigned to the corresponding query circle radius (see Figure 8.23 step 3).

- 4.) As already mentioned, the second and third task of the previous approach are already integrated without changes in the step above. For approximating the selectivity, the original fourth task has to be modified slightly. Now, the algorithm uses the given query circle radius for identifying the appropriate cluster allocation. Then, the approximated selectivity is calculated with the formula 8.2 (see fourth point of the previous approach) with the given query point and radius and the assigned cluster allocation.

Note, that the computation of clusters during query optimization is not feasible as it is a time consuming task. Therefore, this calculation is done offline with the predefined range of query circle radii given by the user. The result of this operation is stored and the query optimizer only operates on the stored results.

#### 8.4.4 Architecture

The architecture of our selectivity optimizer is twofold. The main parts of the cluster building process, which is done offline, is explained in Subsection 8.4.4.1. All necessary parts that are used by the query optimizer are described in Subsection 8.4.4.2.

##### 8.4.4.1 MDCSelectivityOptimizer

The *MDCSelectivityOptimizer* is a Java package which receives the following input data by the user: the data set (which is in general a set of n-dimensional points), their dimensionality, the desired minimum query circle radius (*min\_qcr*), the desired maximum query circle radius (*max\_qcr*), an incremental factor (*incr\_factor*) and the *MinPts* value. Figure 8.23 shows the main architecture of the *MDCSelectivityOptimizer*. In general, the data flow of the final approach is divided into two main and three intermediate steps which are described in Section 8.4.3.2.

##### 8.4.4.2 Database Extension

The main idea of our approach is to approximate the selectivity of Range-Searches so that the query optimizer is able to create an optimal query plan. Due to the fact that a query optimizer can only use a fraction of time of the real query processing for determining an optimal query plan, the approximation of the selectivity has to

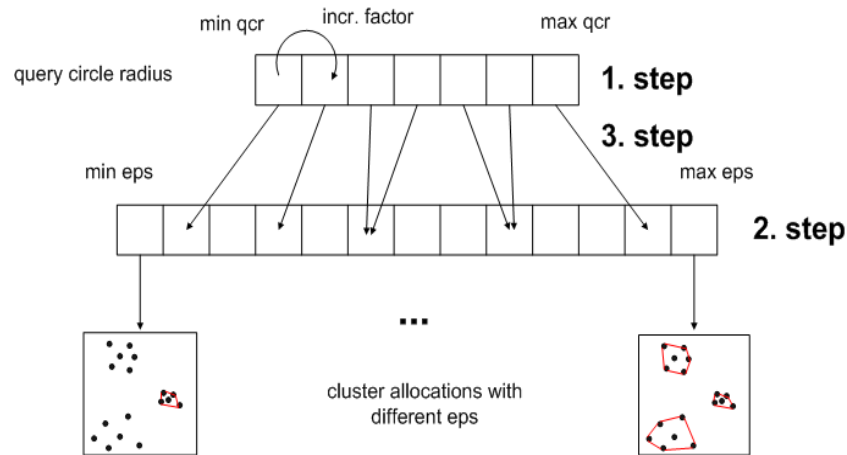


Figure 8.23: Internal Architecture of the MDCSelectivityOptimizer

be very fast. Therefore, we come up with the following architecture. As explained above, the time consuming pre-processing including minor failure determination and cluster allocation is done offline. The resulting data is then stored to disk where the amount of necessary information is minimal (e.g. in sum less than 20K for 2000 city coordinates). The query optimization process loads this data once at startup into the main memory which guarantees a fast access time. The connection between the query optimizer and our approximation selectivity implementation is realized through ODCI [107] (Oracle Data Cartridge Interface) which was also used for the integration of our indexing environment (see Section 8.3).

The following PL/SQL code specifies the necessary object and body implementation for the ODCI-Selectivity enhancement. In our case the *ODCIStatsSelectivity* method is specialized for a Range query operator that contains the following parameters: *testP* is of type VARCHAR2 and contains a n-dimensional point. *queryP* is of type VARCHAR2 and contains the given n-dimensional query point. *dim* specifies the dimension of all participating points and *radius* determines the query circle radius. All other parameters are predetermined by ODCI.

```
CREATE or REPLACE TYPE MDCSelectivityOptimizer AS OBJECT (
  curnum NUMBER,
  STATIC FUNCTION ODCIGetInterfaces(ifclist OUT sys.ODCIObjectList) RETURN NUMBER,
  -- selectivity function for range queries
  STATIC FUNCTION ODCIStatsSelectivity(pred sys.ODCIPredInfo,
```

```

        sel OUT NUMBER,
        args sys.ODCIArgDescList,
        strt NUMBER,
        stop NUMBER,
        testP VARCHAR2,
        queryP VARCHAR2,
        dim NUMBER,
        radius REAL,
        env sys.ODCIEnv)
        return NUMBER
);
/

CREATE OR REPLACE TYPE BODY MDCSelectivityOptimizer IS
    STATIC FUNCTION ODCIGetInterfaces(ifclist OUT sys.ODCIObjectList) RETURN NUMBER IS
    BEGIN
        ifclist := sys.ODCIObjectList(sys.ODCIObject('SYS','ODCISTATS2'));
        RETURN ODCIConst.Success;
    END ODCIGetInterfaces;

    STATIC FUNCTION ODCIStatsSelectivity(pred sys.ODCIPredInfo,
        sel OUT NUMBER,
        args sys.ODCIArgDescList,
        strt NUMBER,
        stop NUMBER,
        testP VARCHAR2,
        queryP VARCHAR2,
        dim NUMBER,
        radius REAL,
        env sys.ODCIEnv)
        RETURN NUMBER IS
    BEGIN
        sel := getSelectivity(queryP,dim,radius);
        RETURN ODCIConst.SUCCESS;
    END ODCIStatsSelectivity;
end;
/

```

The implementation of the *ODCIStatsSelectivity* calls the *getSelectivity(...)* function which is defined below and delegates the call to our approximation selectivity implementation. Our implementation returns for a given query point and the corresponding query circle radius an approximation of the selectivity. This approximation is forwarded to the calling query optimizer.

```

CREATE OR REPLACE FUNCTION getSelectivity(point IN VARCHAR2, dimension IN NUMBER, radius IN REAL)
    RETURN NUMBER AS LANGUAGE JAVA NAME
    'at.mdoeller.mdc.mdclibs.mpeg7optimizer.database.MDCDBSelectivity.getSelectivity(java.lang.String,

```



```

int,
float)
return int';
/

```

The following statement causes the query optimizer to use our selectivity implementation in case the *rangeQuery* function is used.

```
ASSOCIATE STATISTICS WITH FUNCTIONS rangeQuery USING MDCSelectivityOptimizer;
```

### 8.4.5 Cluster Visualization Tool

In addition, we implemented a cluster visualization tool for 2-dimensional data in order to evaluate our approach (see Figure 8.24). The Figure shows the best cluster association for the query circle radius of 0.12 over 1000 city coordinates. In this special case, our algorithm identified an *eps*-value of 0.29 best suited for the given requirements. The screenshot shows the convex hulls of all clusters (20)<sup>2</sup>. Furthermore, the tool can be used for calculating the approximated selectivity vs. the real selectivity of a user defined query point and radius.

### 8.4.6 Evaluation

This Section describes the series of experiments we performed in order to evaluate the effectiveness of our approach. The tests were carried out on three distinct data sets, one *synthetic* (uniform data set) and two real data sets: 2-dimensional coordinates of Austrian and German cities and 8-dimensional color histograms of images. Our evaluation aimed two targets: First, we compared our approximation to the real selectivity for all data sets. Second, we show the use and performance gain of our approach within an Oracle database. The experimental settings are as follows:

- The synthetic data set contains 2-dimensional feature vectors that are represented as strings. The values were generated uniformly over the normalized [0..500] interval.

---

<sup>2</sup>Note, that the color has no specific meaning.

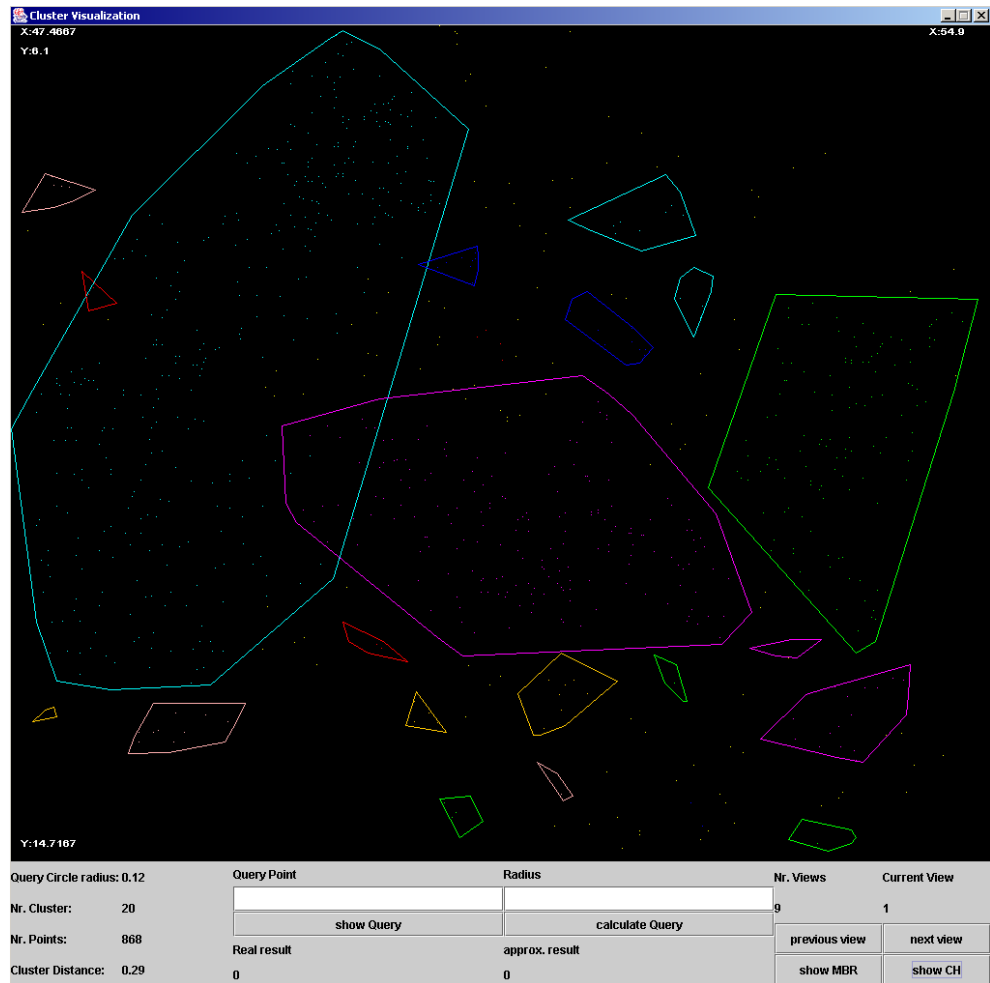


Figure 8.24: Tool for cluster visualization

- The 2-dimensional coordinates of Austrian and German cities were received from <http://opengeodb.de/>. The values of the coordinates are within an X-range of  $[47.4 .. 54.9]$  and an Y-range of  $[6.1 .. 14.7]$  and correspond to the transformed geographical latitude and longitude.
- The 8-dimensional color feature vectors are extracted from images of size  $128 \times 192$  pixels in the RGB space. The values are within the  $[0..1]$  interval.

In addition, the database setting contains the tables *city* and *location* which have been already described in Subsection 8.4.1

### 8.4.6.1 Detailed Results

In order to verify our approach, we calculated the failure (see Equation 8.3) between the real selectivity and the approximated selectivity over all points in the data set. The following Figures (Figures 8.25, 8.26, 8.27) show the results for our data sets (synthetic, city coordinates and image color histograms) each emerged over 2000 query points. The left side of the Figures shows the maximum real selectivity (black line), the average real selectivity (red line) and the failure between real selectivity and approximated selectivity (blue line). The right side of the Figures shows the average real selectivity (red line) and compares it to the average approximated selectivity (blue line). The query circle radii differ between each data set and are adapted to the corresponding space. For the synthetic data set, the query circle radius ranges between 5.0 to 40.0 with an incremental factor of 5.0. For the city coordinates, we used a query circle radius of 0.1 up to 0.24 with an incremental factor of 0.02. The query circle radius of the image test-bed ranges between 0.04 to 0.15 with an incremental factor of 0.01.

The evaluation of our test-beds results in: The average relative failure which is calculated by the sum of all percentages of average real selectivity compared to their corresponding failure yields an average percentage of 23% (15% - 39%) for the random data set, 36% (34% - 41%) for the city coordinates and 11% (0.07% - 30%) for the color histograms of images.

This means that on average, for all query circle radii, the difference of the approximated selectivity to the real result is only between 11% and 36% which is an acceptable result for an approximation. This result is confirmed by a second series of Figures (right side) which results in an average difference of 6%, 15% and 4% for the random, city and images test-bed. This means that the difference between the average real selectivity and the average approximated selectivity over all points of the data set is small.

**Database evaluation** We used the following setting for the evaluation of the approximated selectivity approach within the database. The test-bed contains two tables and simulates a geographic application: The first table, named *city*, contains an unique ID and the name of an city. The second table, named *location*, contains

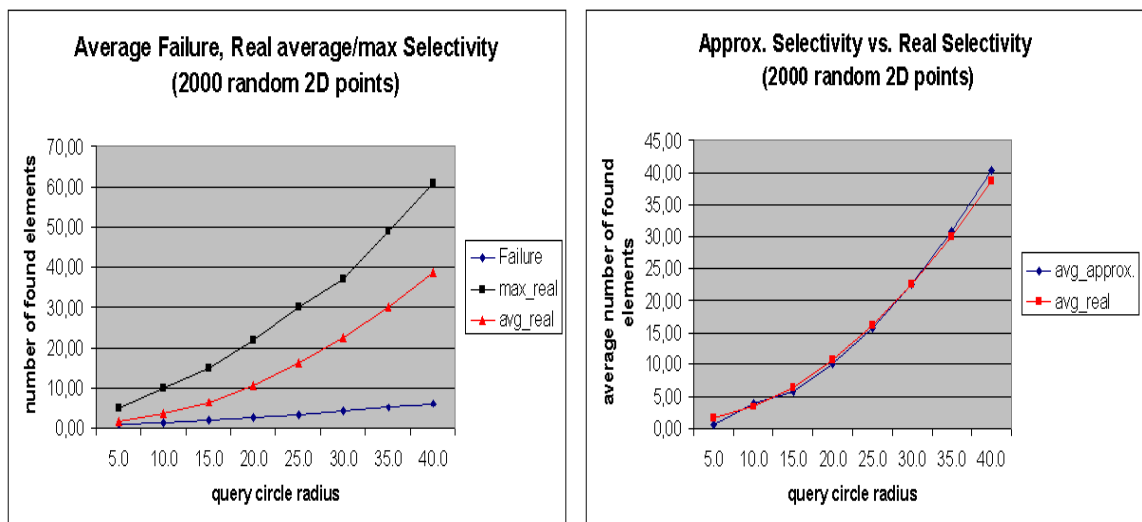


Figure 8.25: Approximated Selectivity Evaluation (synthetic data set)

the foreign key ID of the table *city* and the location of the city given as 2-dimensional coordinates. Both tables contain 1000 tuples.

The analysis was twofold. First, the changes of the query plan were evaluated with the help of the *EXPLAIN PLAN* command. Second, the execution times for the queries in combination with and without our approximation are compared.

In the following, we are interested in all cities that are in a specific area and whose name starts with an A. This would result in the following query:

```
explain plan SET STATEMENT_ID = 'selectivity'
for
select count(*) from location l, city c where rangeQueryOp(l.location, '50.0 10.0', 2, 0.22) = 1
      AND c.name LIKE 'A%'
      AND l.cityid = c.id;
```

The acquired query plan is presented in Table 8.1. The left column denotes the query plan without an a priori information of the selectivity and the right column shows an improved query plan. The main crucial factor for the performance is the ordering of the join operator. Without a priori information, the city table is left to the join. In fact, the amount of resulting tuples of the selection operation for the

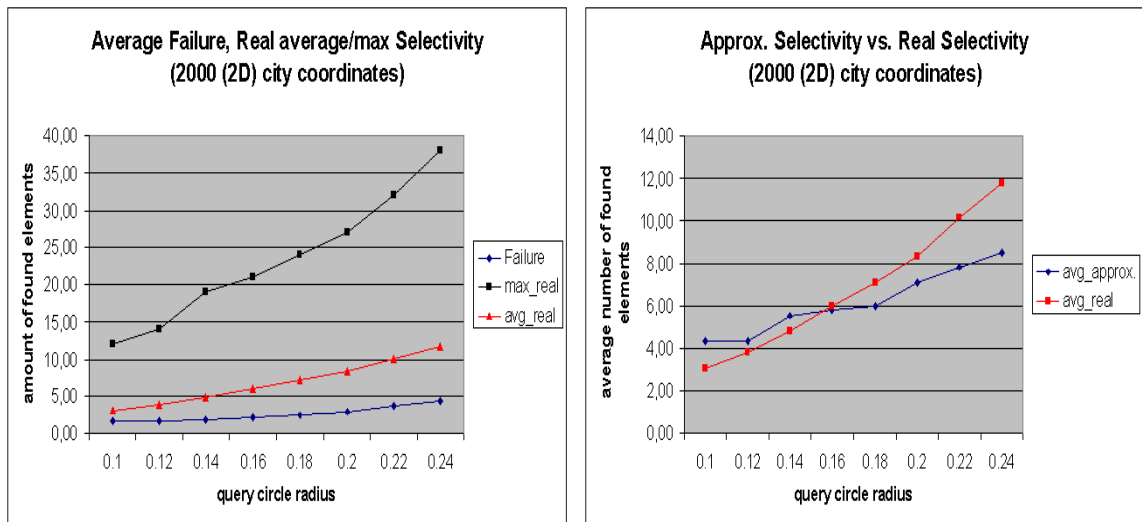


Figure 8.26: Approximated Selectivity Evaluation (city data set)

city table is much higher than for the Range query operation on the location table. Therefore, the right column of the table shows the best performance. This is also stated by the real measurement results which is 43.1 seconds for the query plan of the left column and 0.5 seconds for the plan of the right column.

without an approximation	with an approximation
QUERY PLAN	QUERY PLAN
-----	-----
SELECT STATEMENT	SELECT STATEMENT
SORT AGGREGATE	SORT AGGREGATE
NESTED LOOPS	MERGE JOIN
TABLE ACCESS FULL CITY	TABLE ACCESS FULL LOCATION
TABLE ACCESS FULL LOCATION	BUFFER SORT
	TABLE ACCESS FULL CITY

Table 8.1: Query Plan Comparison of first query

In the second query, we exchanged the first boolean operator AND to OR. The query plan again is shown in table 8.2 where the left column corresponds to a result without a priori knowledge and the query plan in the right column results with the usage of our approximated selectivity approach. Again, the main difference is the ordering of the participating tables of the join operation. This is also shown in the real measurement results of both query plans, which is 89.2 seconds for the left column

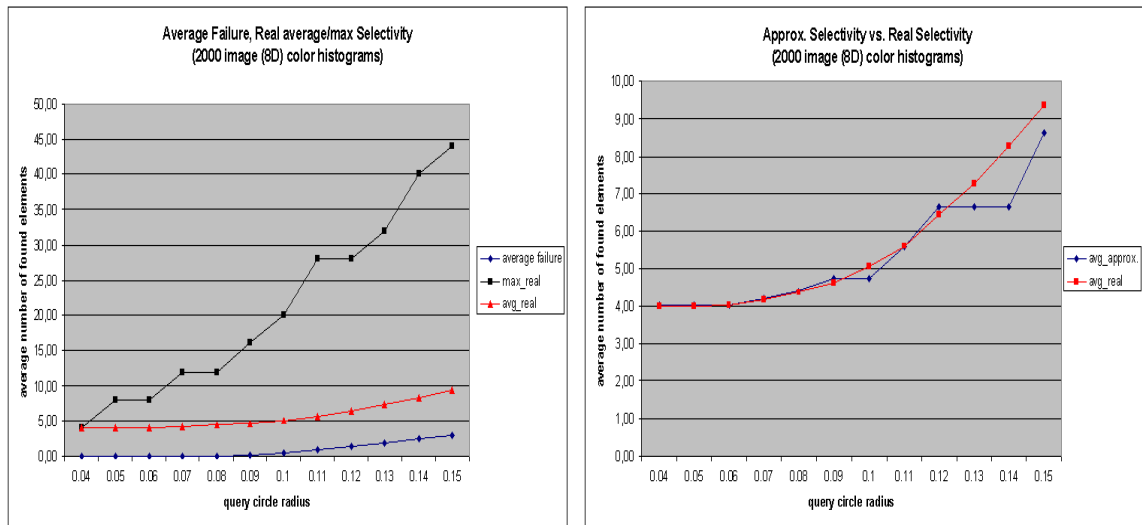


Figure 8.27: Approximated Selectivity Evaluation (image data set)

and 45.8 for the right column.

```

explain plan SET STATEMENT_ID = 'selectivity'
for
select count(*) from location l, city c where (rangeQueryOp(l.location, '50.0 10.0', 2, 0.22) = 1
                                                OR c.name LIKE 'A%')
                                                AND l.cityid = c.id;

```

without an approximation	with an approximation
QUERY PLAN	QUERY PLAN
-----	-----
SELECT STATEMENT	SELECT STATEMENT
SORT AGGREGATE	SORT AGGREGATE
NESTED LOOPS	NESTED LOOPS
TABLE ACCESS FULL CITY	TABLE ACCESS FULL LOCATION
TABLE ACCESS FULL LOCATION	TABLE ACCESS FULL CITY

Table 8.2: Query Plan Comparison of second query

### 8.4.7 Conclusion

This Section introduced our approach for approximating the selectivity of Range queries. We used the DBSCAN clustering technique for finding high density areas in

the data set. The selectivity was approximated with the help of a density function in combination with the volume of the query's hypersphere. In addition, a cluster visualization tool was presented.

The approach was evaluated on three different data sets, namely synthetic data, city coordinates and color histograms of images. The evaluation showed that on average the difference to the real selectivity was minimal and did not exceed  $1/3$ . In nearly all cases (for the random and image data set), the relative failure is around 15%. Finally, it was shown that the query plan and hence the performance of queries was improved by the use of our approach.

Future research will focus on the use of different cluster techniques, especially for high-dimensional data. In addition, the performance of the precalculation of the clusters has to be improved. At the moment, the implementation is more or less brute force. Furthermore, areas where no points are located have to be identified because, if a query point falls in such an area we return  $MinPts - 1$  instead of 0. Another performance optimization could be realized in step 3 of our approach. The *eps*-value of the best suited cluster and the query circle radius are in general near together, hence it is not necessary to test all cluster allocations for every query circle radii.

Besides the core system (see Figure 7.1 (1)), described in chapter 8, the internal libraries (see Figure 7.1 (2)) are the most important parts of the MPEG-7 MMDB. They offer interfaces for using and managing the stored data. The internal libraries offer two access scenarios. On the one hand, each library can be accessed directly and on the other hand one can take advantage of the *MDC data type* that provides the functionality of the internal libraries as an database type. In the following Sections all internal libraries and their installation guides are introduced.

## 9.1 InitLib

It is the task of the *InitLib* to initialize a new MPEG-7 MMDB instance with all necessary database objects. The *InitLib* is a package consisting of a Java-class and several PL/SQL programs. A clean installation requires that, the user specifies the database user name, the password and the tablespace. The system uses the default tablespace if no other one is given. Unfortunately, the database user as well as the tablespace must be created manually because DML database commands are not allowed in stored procedures. A MPEG-7 MMDB instance encompasses six main parts. The core system covers the database schema, the indexing environment and the query optimizer. For all these subparts, the *InitLib* creates all necessary database objects, tables, synonyms, operators and functions. During the next step the *InitLib* initializes several security features e.g., for the indexer. It finalizes the instantiation process by installing all java packages for the internal and application libraries and



their respective interfaces in terms of PL/SQL functions. After completion of the installation process the user finds a MPEG-7 MMDB that contains all necessary parts expect the indexing environment that reside in the operating system environment. The indexing environment in the external address space has to be installed beside.

## 9.2 InsertLib

This Section describes the main tasks that are performed during insertion of our example MPEG-7 document (see Figure 9.1 left side). The library expects a valid and well formed MPEG-7 description and therefore, no schema validation is performed. The basic idea is to perform a post order traversal of the document's DOM tree (see Figure 9.1 right side) which is created with the help of an corresponding XML parser. A leaf node is defined as either having no more child nodes (e.g.: FreeTextAnnotation, Note: the text element is not indicated as a node) or representing an XMLTYPE in our database schema (e.g.: MediaLocator) which is inserted with all child elements. Besides the XMLTYPE and basic types, such as Integer, the insertion process has to identify the following types: *REF Type* represents a reference pointing to a row somewhere in the database, *Dummy Types* indicates nested tables and *Empty Types* meaning that the element has no corresponding column in the current table (e.g.: VisualDescriptor element of table StillRegion). The main part is to identify a node in the DOM tree [182] that corresponds to a database table in our schema. After a successful identification the necessary insert statement has to be created. For this purpose we have to specify the table name the parameter list and their values. The attributes of the current node (identified as a database table) can be fetched with methods of the DOM API to retrieve all child nodes. The attribute name/value pairs are stored into the respective column and value list vectors. Finally, the complete insert statement can be build and executed. After a successful execution the algorithm proceeds with the next node in the post order traversal. The algorithm terminates as soon as the root element of the document is reached.

Looking at the example, the tree is traversed along the leftmost subtree until the *MediaLocator* node has been reached. As its type is XMLTYPE, the traversal is continued at its sibling, *CreationInformation*. Again, the tree is traversed until the

next leaf node is reached, namely *Title*. The two *Title* leaf nodes are stored in the corresponding nested table of their parent node. The parent node is a possible insert candidate and therefore processed as described before. The other nodes considered immediately for being inserted are (in the order of traversal): *Abstract* plus subtree, *Creator* plus subtree, *TemporalDecomposition*, *Audio*, *MultimediaContent*, *Description* and *Mpeg7*.

After a successful insertion of the document parts into their corresponding database tables, the entire document is inserted into the *Mpeg7\_Document* table. The main purpose of this is to increase query performance, if someone is interested in the entire document. Therefore, we accept the additional memory consumption.

### 9.3 DeleteLib

The *DeleteLib* allows the deletion of a set of full MPEG-7 documents from the database. The documents are represented by their document ID, which is stored in every database table. The PL/SQL procedure for deleting documents is shown below. Besides the deletion of a single document, the library offers functionality to clean the entire database or subareas thereof. Subareas are for instance the MPEG-7 descriptions of all images, videos or audio files.

```
CREATE OR REPLACE PROCEDURE delDocument(idin INTEGER) IS
  CURSOR classes IS select table_name from user_tab_columns where column_name = 'DOC_ID';
  mtable varchar2(255);
  query VARCHAR2(255);
BEGIN
  IF idin = NULL
  THEN
    dbms_output.put_line('No DOC_ID specified!');
  ELSE
    OPEN classes;
    LOOP
      FETCH classes INTO mtable;
      EXIT WHEN classes%NOTFOUND;

      query := 'DELETE FROM ' || mtable || ' WHERE DOC_ID = ' || idin;
      EXECUTE IMMEDIATE query;
      dbms_output.put_line('deleted ' || SQL%ROWCOUNT || ' row(s) from ' || mtable);
    END LOOP;
  END IF;
END;
```

```

END LOOP;
CLOSE classes;
END IF;
END;
/

```

## 9.4 QueryLib

The multimedia querying system provides three query alternatives for formulating queries. Due to the rich variety of MPEG-7 we only concentrate on the running example on an audio description. Therefore, Figure 9.1(a) represents a possible MPEG-7 description for audio data. For simplicity the audio description contains only information about *MediaLocation* and *CreationInformation* such as *Title*, *Abstract* and *Creator*. Figure 9.1(b) denotes the corresponding tree representation. In the following we discuss all three alternatives and present their advantages and disadvantages.

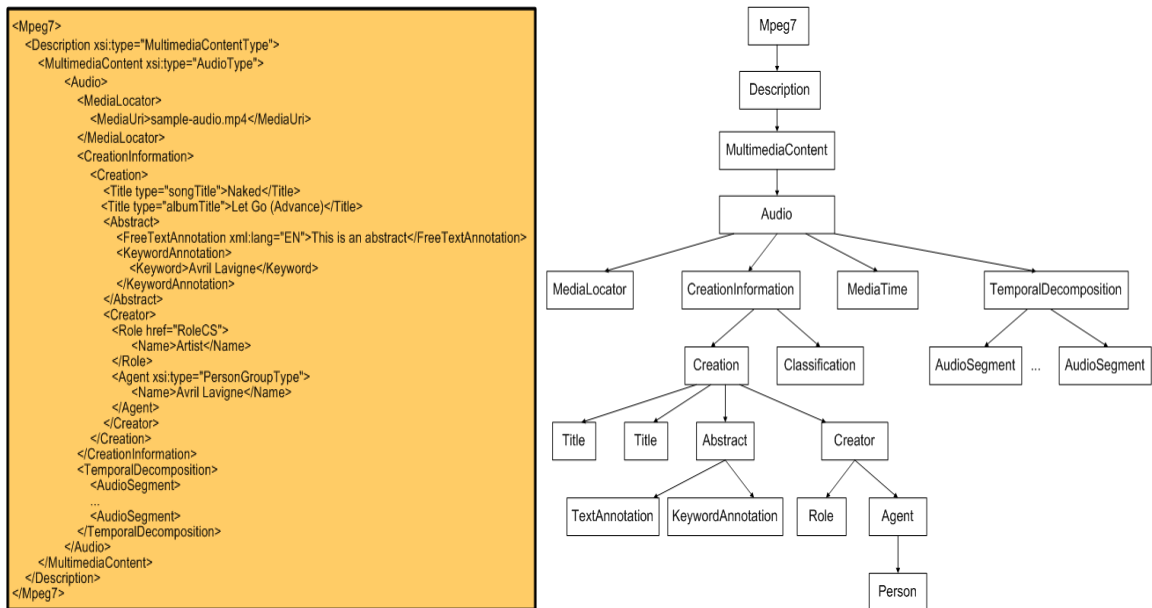


Figure 9.1: (a) MPEG-7 instance document describing audio data, (b) corresponding tree representation

The first service allows one to query the database traditionally with the help of

XPATH expressions. For instance, one is interested in all file locations of stored audio descriptions. The file location is typically described with the help of MPEG-7's *MediaLocator* descriptor (see instance document at Figure 9.1(a)). In our database schema, this descriptor is realized as an attribute of the table `audio` and is of type *XMLTYPE*. Therefore, this would result in the following query.

```
SELECT extract(mediaLocator, '/MediaLocator/MediaUri/text()')
FROM audio v, PersonGroup pg, table(pg.dummyAttr190) pg_dummy190
WHERE v.doc_id = pg.doc_id AND deref(pg_dummy190.COLUMN_VALUE).Name LIKE '%Avril%';
```

As the type of *MediaLocator* is *XMLTYPE* we have to use the XPATH `/MediaLocator/MediaUri/` expression in order to retrieve the required information. The `text()` method in combination with the `extract()` method gives the desired data. The main advantage of this technique is that every information is reachable in combination with XPATH expressions. Nevertheless, there are two disadvantages. First, the query becomes very complex in the case of combining several information both in the select clause and the where clause. Second, the output is not formatted at all.

The second service allows one to format the output of a query with the help of Oracle's XMLDB [169] functionality. By combining *XMLElement*, *XMLAgg* and *XMLAttribute* functions provided by Oracle's XMLDB, one can produce valid MPEG-7 descriptions. Let us assume, that we want to search the database for the interpreter of certain music songs. The resulting output should contain information about the location of the audio file and several *CreationInformation* such as *Title*, *Creator*, *CreationCoordinates* and *MediaTime*. Instead of using any proprietary format, we would like to describe all found matches with the help of MPEG-7 descriptions. These requirements can be fulfilled with the query presented in Figure 9.2(a). The resulting information is described with MPEG-7 as represented in Figure 9.1(a).

As one can see the main advantage of this service is the formatted output namely with MPEG-7. Therefore, the result can be forwarded to any application that can evaluate MPEG-7. Further, we keep the advantage that any information is reachable in combination with XPATH expressions.

The main disadvantage of this approach is the complexity of the query statements. Therefore, it only applies to database experts with a good knowledge of database schema obtained from MPEG-7 mapping.

Thus, one may use alternatively our QueryLib, where the user can create individual select statements with the help of a modular construction system. This modular construction enables users to specify the select part as well as the where part according to their needs. The QueryLib uses the circumstance that instance documents of the MPEG-7 standard can be represented with the help of a tree structure (see Figure 9.1(a) and (b)). In general, database queries consists of two parts, the select clause and a possible where clause. The select clause defines all information where a user is interested in, whereas the where clause constrains the amount of resulting tuples. The construction of the select clause becomes complex, if we claim that the corresponding output of a query has to follow the MPEG-7 standard, which enjoins a correct ordering of descriptors within the same level and within the tree hierarchy. For instance the *Creation* descriptor only can follow the *CreationInformation* descriptor and so on.

The main idea of the QueryLib is that the user can pick out all desired information that the resulting MPEG-7 instance document should contain. Let us illustrate this approach with the same scenario as above. The user queries for all music songs of a certain interpreter and the resulting matches should be formatted as valid MPEG-7 descriptions containing the same information as mentioned above. Figure 9.2(b) shows the code fragment using our QueryLib. First, the user has to instantiate the QueryLib by specifying the area of interest (see line 3 at the code fragment), which is in our case the audio part of MPEG-7. During instantiation, a full tree representation of the required MPEG-7 description is created internally. Afterwards, the user can modify the select clause by adding the desired information (see line 6 to 18) which internally leads to an tagging of the chosen descriptor in the tree representation. The *addSelectClause* method demands two parameters. The first one is the desired descriptor and the second one contains all father nodes up to the root which we like to include. The list of father nodes is needed because descriptors can occur in different levels (e.g.: the *MediaLocator* can be a descriptor of the top level audio descriptor or of an *AudioSegment* which is a subpart). The direct use of our QueryLib assumes a broad knowledge of MPEG-7 and the individual descriptors. This can be improved by a graphical user interface that guides the user through the process of query creation by presenting a tree representation of the desired MPEG-7 descriptions, where the

```

SELECT XMLELEMENT("Mpeg7",
  XMLAttributes('um:mpeg:mpeg7:schema:2001' as "xmlns",
    'http://www.w3.org/2001/XMLSchema-instance' as "xmlns:xsi",
    'um:mpeg:mpeg7:schema:2001' as "xmlns:mpeg7",
    'um:mpeg:mpeg7:schema:2001 definitions.xsd' as "xsi:schemaLocation"),
  XMLAgg(XMLElement("Description", XMLAttributes(d.description as "xsi:type"),
    XMLElement("Audio", extract(v.medialocator,/MediaLocator),
    XMLElement("CreationInformation",
      XMLElement("Creation",
        (select XMLAGG(XMLElement("Title",
          XMLAttributes(deref(t1.COLUMN_VALUE).type as "type"),
          deref(t1.COLUMN_VALUE).Title))
        from table (v.CreationInformation.Creation.dummyAttr801) t1
        ),
        (select XMLAGG(XMLElement("Creator",
          extract(deref(t2.COLUMN_VALUE).Role, '))
          XMLElement("Agent", XMLAttributes(deref(t2.COLUMN_VALUE).agent.subtype as "xsi:type"),
          (select XMLAGG(XMLElement("Name", deref(t3.COLUMN_VALUE).Name))
            from Person p, table (p.dummyAttr119) t3
            where p.doc_id = v.doc_id
            and p.part_id = deref(t2.COLUMN_VALUE).agent.subtype_PART_ID
          )))
        )
        from table (v.CreationInformation.Creation.Creator) t2
        ),
        extract(v.CreationInformation.Creation.CreationCoordinates, '))
        ),
        extract(v.MediaTime,/MediaTime')
        ))).getStringVal()
FROM Mpeg7 mp, audio v, table (mp.description) d,
  table(v.CreationInformation.Creation.dummyAttr801) t1,
  PersonGroup pg, table (pg.dummyAttr190) pg_dummy190
WHERE mp.doc_id = v.doc_id
and mp.doc_id = pg.doc_id
and deref(pg_dummy190.COLUMN_VALUE).Name LIKE '%Avril%';

```

```

1: public static Clob searchInterpreter(String name) {
2:
3:   QueryLibI query = new QueryLibI(QueryLibI.AUDIOPART);
4:
5:   // fill the select clause
6:   query.addSelectClause(at.mdoeller.mdc.mdclibs.mpeg7querylib.queryselect.MediaLocator.NAME, new Vector());
7:
8:   Vector vec = new Vector();
9:   vec.addElement(at.mdoeller.mdc.mdclibs.mpeg7querylib.queryselect.CreationInformation.NAME);
10:  vec.addElement(at.mdoeller.mdc.mdclibs.mpeg7querylib.queryselect.Creation.NAME);
11:  query.addSelectClause(at.mdoeller.mdc.mdclibs.mpeg7querylib.queryselect.Title.NAME, vec);
12:  query.addSelectClause(at.mdoeller.mdc.mdclibs.mpeg7querylib.queryselect.CreationCoordination.NAME, vec);
13:  query.addSelectClause(at.mdoeller.mdc.mdclibs.mpeg7querylib.queryselect.CreationPersonGroupName.NAME, vec);
14:
15:
16:  Vector vec1 = new Vector();
17:  vec1.addElement(at.mdoeller.mdc.mdclibs.mpeg7querylib.queryselect.CreationInformation.NAME);
18:  query.addSelectClause(at.mdoeller.mdc.mdclibs.mpeg7querylib.queryselect.Classification.NAME, vec1);
19:
20:
21:  // fill the where clause
22:  QueryWhereI person_name = new at.mdoeller.mdc.mdclibs.mpeg7querylib.querywhere.PersonGroupName();
23:  Vector elements = new Vector();
24:  elements.addElement(name);
25:  person_name.setWhereClauseElements(elements);
26:  query.addWhereClause(person_name);
27:
28:  String queryString = query.getQuery();
29:
30:  MDCAudioQuery mdcaq = new MDCAudioQuery();
31:  Clob clob = mdcaq.executeQuery(queryString);
32:
33:  return clob;
34: }

```

Figure 9.2: (a) Audio Query that returns an MPEG-7 description (b) Code fragment that produces same output as (a)

user only needs to click at the desired descriptors.

In our example, the resulting MPEG-7 instance document should contain the *MediaLocator* descriptor for each found music song. For this purpose, as shown at line 6, one has to call the respective method with the name of the descriptor (in our case *MediaLocator.NAME*) as first parameter and a list of all father nodes as second parameter. This list is empty, because the desired *MediaLocator* directly resides under the root. Further, we are interested in the title of the song and the name of the music group. Therefore, the user has to proceed as follows. From line 8 to 10, the list of father nodes is specified. This list contains two nodes which represent the path *CreationInformation - Creation*. Then, in line 11 to 13, the *addSelectClause* methods are given with their respective parameters (e.g., *Title.NAME* as first and the vector

as second).

The result set of our query should be restricted to a certain entertainer. Therefore, the query has to be extended by a where clause which is realized in the code fragment from line 22 to 26. There, an instance of class *PersonGroupName* is instantiated that constrains the name of the searched music group.

From line 28 to 31 the query system transfers all given information into an equivalent database query which is presented in Figure 9.2(a). This query is executed and the result is forwarded to the user.

The main advantage of this approach is that it combines the possibilities of the previous services, namely formatted MPEG-7 output and accessibility of every information in combination with XPATH expressions, by hiding the complexity of query statements by a modular construction system that allows the adaptation of the select clause as well as the where clause to the user needs.

## 9.5 MDC data type

In addition to internal libraries, the MPEG-7 MMDB provides a separate database type, called *MDC* that encapsulates the main functionality e.g., initiation, insertion, deletion and querying. This data type can be used as an entry in a database table and serves as interface to the underlying MPEG-7 schema. The *MDC* type is implemented as a JAVA class and is defined by the following object specification (see below):

```
CREATE OR REPLACE TYPE MDC AS OBJECT EXTERNAL NAME 'at.mdoeller.mdc.MDC'
LANGUAGE JAVA USING ORAData (
  user_name VARCHAR2(400) external name 'user_name',
  pw VARCHAR2(400) external name 'pw',
  tablespace VARCHAR2(400) external name 'tablespace',

  STATIC FUNCTION MDCInit(user_name CHAR, pw CHAR, tablespace CHAR) RETURN SELF
  AS RESULT EXTERNAL NAME 'create (oracle.sql.CHAR, oracle.sql.CHAR, oracle.sql.CHAR) return at.mdoeller.mdc.'

  MEMBER FUNCTION MDCInsert(xmldata CLOB) RETURN NUMBER EXTERNAL NAME 'mdcInsert (oracle.sql.CLOB) return int',

  MEMBER FUNCTION MDCDelete(doc_id NUMBER) RETURN NUMBER EXTERNAL NAME 'mdcDelete (int) return int',

  MEMBER FUNCTION MDCUpdate(doc_id NUMBER, xmldataOld CLOB, xmldataNew CLOB) RETURN NUMBER
  EXTERNAL NAME 'MDCUpdateCLOB (int, oracle.sql.CLOB, oracle.sql.CLOB) return int',
```

```
MEMBER FUNCTION MDCQuery(query CLOB) RETURN XMLType EXTERNAL NAME 'mdcQuery (oracle.sql.CLOB) return oracle.xdb.X
);
/
```

MDC stores internally the user name, the respective password and table space, and provides the following functionality:

- *MDCInit* is a static method and is used for initializing a new instance of the whole environment. Therefore, one has to pass the data for user name, password and table space. Note, that the database user with the corresponding password must already exist. The *MDCInit* method forwards all necessary data to the *InitLib* (see Section 9.1).
- The *MDCInsert* method inserts MPEG-7 documents into the MPEG-7 MMDB schema with the help of the *InsertLib* library (see Section 9.2).
- The *MDCDelete* method removes the document that is identified by the passed document ID.
- The *MDCUpdate* method updates the data of the passed MPEG-7 descriptor (*xmldataOld*) in the respective MPEG-7 document, identified by the given ID (*doc\_id*), with the new data contained in the *xmldataNew* descriptor. Internally, this leads to an deletion of the whole document and an reinsertion of the new data.
- The *MDCQuery* method executes the given query and returns the result as a valid MPEG-7 document. The query has to be created by the introduced possibilities shown in Figure 9.2.

In the following, the *MDC* type is demonstrated by the following scenario. Let us assume, an user possesses different collections of image, video and audio data. All data is described by the MPEG-7 standard and should be stored in separate archives. Furthermore, the data should be accessible by Java clients that reside outside of the database. The user can realize this scenario with the following commands. First, a table is created that contains a unique number and an attribute of type *MDC*. Then, the archives are created as three different instances of *MDC*, one for image, one for video and the last one for audio data.



```

CREATE TABLE archive (id NUMBER, m MDC);

INSERT INTO archive VALUES (1, MDC.MDCInit('mdc_image', 'mdc_image', 'mdc_space1'));

INSERT INTO archive VALUES (2, MDC.MDCInit('mdc_video', 'mdc_video', 'mdc_space1'));

INSERT INTO archive VALUES (3, MDC.MDCInit('mdc_audio', 'mdc_audio', 'mdc_space1'));

```

The archives can easily be manipulated with every Java client (e.g., see Figure 9.3 that demonstrates the insertion and querying of MPEG-7 documents) or PL/SQL function (see below).

```

create or replace procedure MDC_InsertTest IS
  obj MDC;
  res NUMBER;
  path CONSTANT VARCHAR2(400) := 'READ_MP7_FILE';
  filename VARCHAR2(200);
  ldoc BFILE;
BEGIN
  -- read from file
  filename := 'AL_Tomorrow.mp7';
  ldoc := BFILENAME(path,filename);

  select m into obj from archive where id = 1;
  res := obj.MDCInsert(ldoc);
  dbms_output.put_line('result of mdc_insert: ' || res);
END;
/

```

In the following, the insert and query process for the Java client is presented. First, shown in line 4, the client needs to access the database via JDBC. As we want to demonstrate the insertion of a MPEG-7 document describing audio data, the query in line 8 retrieves the required database tuple. The *MDC* object of the tuple is instantiated with the command in line 11. From line 15 to 24, the MPEG-7 document is read from disk and transferred into *CLOB* data type. In line 27 the MPEG-7 document is inserted by calling the *mdcInsert* method of the *MDC* object.

The query process starts at line 31. First, we created with the help of our *QueryLib* a query similar to that in Figure 9.2. The result of the *QueryLib* is transformed into Oracle's *CLOB* datatype (line 32) and serves as input parameter for the *mdcQuery()*

method of the *MDC* datatype (see line 33). The result is presented by the lines 34 to 38.

```
1: public static void main (String [] argv) {
2:
3:   try {
4:     Connection con = createConnection("WEB_R_BLOB", "WEB_R_BLOB");
5:
6:     Statement stmt = con.createStatement();
7:
8:     OracleResultSet rset = (OracleResultSet)
9:       stmt.executeQuery("Select * from archive where id=3");
10:
11:     while (rset.next()) {
12:       Object o = rset.getORAData (2, MDC.getORADataFactory());
13:       if (o != null) {
14:         if (o instanceof MDC) {
15:           MDC p = (MDC) o;
16:           File xmlfile = new File("./mp7/AL T.mp7");
17:           FileReader fr = new FileReader(xmlfile);
18:           BufferedReader br = new BufferedReader(fr);
19:           String xmldata = "";
20:           String line = null;
21:           while ((line = br.readLine()) != null)
22:             xmldata += line;
23:
24:           br.close();
25:           CLOB xmlClob = getCLOB(xmldata, con);
26:
27:           p.setConnection(con);
28:           p.mdcInsert(xmlClob);
29:           System.out.println("executed!!!");
30:
31:           p.setConnection(con);
32:           CLOB queryClob = getCLOB(createTestQuery(), con);
33:           CLOB result = p.mdcQuery(queryClob);
34:           if (result != null) {
35:             System.out.println("Result :\n"+result.getSubString(1, (int)result.length()))
36:           }
37:           else
38:             System.out.println("no result available!");
39:
40:         }
41:       }
42:     }
43:     con.close();
44:   } catch (Exception e) {
45:     System.out.println("Exception raised!!!: "+e);
46:   }
47: }
```

Figure 9.3: Java Code fragment for demonstrating the MDC database type

# CHAPTER --- 10 Application Libraries

The application libraries act as interfaces between the MPEG-7 MMDB and their applications. Note, that usually every application needs its own library which is caused by different requirements of methods as well as their in- and output parameters. The following Sections describe libraries for applications in the field of blob based image retrieval (see Section 10.1), audio recognition (see Section 10.2) and video browsing (see Section 10.3).

## 10.1 Blobworld Library

The *Blobworld library* is used as interface for our blob based image retrieval system *MPEG-7 Blobworld* (see Section 11.1). Blobs are regions of images which are roughly homogeneous in respect to color and texture. Blobs and their used features (color, texture and shape) are extracted automatically and described with their corresponding MPEG-7 descriptions. A blob of an image is described with the *StillImage* descriptor. Their features are characterized with the *ScalableColor*, *Homogeneous-Texture* and *RegionShape* descriptors. The MPEG-7 Blobworld application offers two methods. The first method, *PLS\_getBlob*, is used for retrieving a certain Blob of a certain image. The image is identified through the *file name* which is described by the *MediaLocator* descriptor. A Blob within an image is identified through its unique Blob ID. The resulting MPEG-7 description contains basic information of the image (e.g., *MediaLocator*) and the entire information of the found Blob. The second method provides functionality for finding a certain number of nearest neighbors

to a given Blob. It requires the following parameter: At first, the referencing Blob which is described with MPEG-7 descriptors and may be the output of the previous method (*PLS\_getBlob*). The second, third and fourth parameter specifies the weight for the represented feature (texture, shape and color). The weight is defined as a three-tuple with the following parameters (NOT, SOMEWHAT, VERY). The fifth parameter specifies the amount of resulting Blobs. The resulting output, which represents a valid MPEG-7 document, contains information of all found Blobs and their corresponding image descriptions. Below, the method specification is given. The implementation is realized with a collection of Java classes that reside inside of the database.

```
CREATE OR REPLACE FUNCTION PLS_getBlob(filename IN VARCHAR2,
                                     blob_id IN VARCHAR2)
RETURN CLOB AS LANGUAGE JAVA NAME
'at.mdoeller.mdc.mdclibs.mpeg7blobworldlib.MDCBlobWorld.getBlob(java.lang.String,
                                                                java.lang.String)
                                                                return java.sql.Clob';
/

CREATE OR REPLACE FUNCTION PLS_getNearestBlobs(blob IN VARCHAR2,
                                               texture IN NUMBER,
                                               shape IN NUMBER,
                                               color IN NUMBER,
                                               amount IN NUMBER)
RETURN VARCHAR2 AS LANGUAGE JAVA NAME
'at.mdoeller.mdc.mdclibs.mpeg7blobworldlib.MDCBlobWorld.getNearestBlobs(java.lang.String,
                                                                           int,
                                                                           int,
                                                                           int,
                                                                           int)
                                                                           return java.lang.String';
/
```

In order to enable NN-Search, the library creates three instances of access methods provided by our Multimedia Indexing Framework (see Section 8.3). Features, described through their respective MPEG-7 descriptors, of each Blob are stored in the corresponding database table (see Figure 8.12 at Section 8.2). The table *RegionShape*, which stores the shape of a Blob, is indexed with the help of an R-tree and handles 40-dimensional point data. The *HomogeneousTexture* table is used for

storing feature vectors representing the texture of a Blob. It is also indexed with the help of an R-tree that stores two-dimensional point data. The 218-dimensional feature vector representing the color of an Blob is stored in the *ScalableColor* table and is indexed by an SS-tree. Detailed information of the retrieval process and the MPEG-7 Blobworld environment is given in Section 11.1.

```
CREATE INDEX rs ON RegionShape(MagnitudeOfART) INDEXTYPE IS rtree PARAMETERS('rt_point 40');
```

```
CREATE INDEX ht ON HomogeneousTexture(Energy) INDEXTYPE IS rtree PARAMETERS('rt_point 2');
```

```
CREATE INDEX sc ON ScalableColor(coeff) INDEXTYPE IS sstree_clob PARAMETERS('ss_point 218');
```

## 10.2 Audio Library

The *Audio library* is used as interface for our audio recognition client, described in Section 11.2. The Java based audio recognition tool enables one to search for music compositions in our multimedia database. The developed tool offers, with the help of the *Audio library*, two kind of search techniques. The first technique relies on a search of music title, interpreter and genre. This information is described with the *CreationInformation* descriptor of MPEG-7 and internally stored in respective database tables and types (see Section 8.2). The second search technique uses audio signals that are specified by the *AudioSignature* descriptor. The audio signal is recorded with a microphone, converted to MPEG-7 descriptions and forwarded to the *Audio library*. The library consists of a Java package, several PL/SQL functions and indexes and provides the following services to ensure both search techniques.

The first service offers functionality for searching music compositions based on their title, interpreter or genre. The PL/SQL function specifications are demonstrated below. Their implementation is realized in a Java class that uses the *QueryLib* for construction of the required query. The result correlates to the MPEG-7 standard and contains a collection of all matching music compositions and information on them. Query efficiency is increased by the use of Oracle's text [170] indexes.

```
CREATE OR REPLACE FUNCTION PLS_searchInterpreter(name IN VARCHAR2) RETURN CLOB AS LANGUAGE JAVA NAME
```

```

    'at.mdoeller.mdc.mdclibs.mpeg7audiolib.MDCAudio.searchInterpreter(java.lang.String) return java.sql.Clob';
/

CREATE OR REPLACE FUNCTION PLS_searchSong(song IN VARCHAR2) RETURN CLOB AS LANGUAGE JAVA NAME
    'at.mdoeller.mdc.mdclibs.mpeg7audiolib.MDCAudio.searchSong(java.lang.String) return java.sql.Clob';
/

CREATE OR REPLACE FUNCTION PLS_searchGenre(genre IN VARCHAR2) RETURN CLOB AS LANGUAGE JAVA NAME
    'at.mdoeller.mdc.mdclibs.mpeg7audiolib.MDCAudio.searchGenre(java.lang.String) return java.sql.Clob';
/

```

The second service identifies unknown music compositions based on a 10 second fingerprint. The fingerprint is described by the MPEG-7 *AudioSignature* descriptor and is stored in the respective database tables (e.g., parts of the fingerprint are stored in the *SeriesOfVector* table). The search relies on a NN-retrieval and is realized by an R-tree structure of the *Mean* attribute of the *SeriesOfVector* table (see below). The library provides for the NN-Search the *PLS\_getNearestAudioInformation()* method with the following parameter: The first parameter contains the fingerprint of the unknown audio signal. The second parameter specifies the amount of resulting matches and the third one defines the used dimension of the index tree. The resulting MPEG-7 document is a collection of all matching music compositions and information on them. The best match is described first.

```

CREATE OR REPLACE FUNCTION PLS_getNearestAudioInformation(blob IN VARCHAR2,
                                                         amount IN NUMBER,
                                                         dimension IN NUMBER)
RETURN VARCHAR2 AS LANGUAGE JAVA NAME
    'at.mdoeller.mdc.mdclibs.mpeg7audiolib.MDCAudioSignature.getNearestAudioInformation(java.lang.String,
                                                                                          int,
                                                                                          int)
                                                                                          return java.lang.String';
/

CREATE INDEX audio_mean ON seriesofvector(Mean) INDEXTYPE IS audiortree PARAMETERS('rt_point 11');

```

## 10.3 Video Library

The *Video library* is used as interface for the *DAHL* project<sup>1</sup>. The *DAHL* project aims to provide a content-based retrieval and an adaptive video streaming environment for the *Virtual Computer Science Exhibition*<sup>2</sup>. The virtual exhibition is emerged in connection with a real exhibition in memory of Ole-Johan Dahl, Edsger W. Dijkstra and Kristen Nygaard at the University Klagenfurt, Austria. The video and audio data is annotated with the help of MPEG-7 and stored in our MPEG-7 MMDB.

The library provides the following retrieval functionality:

- **Vector searchByMediaURI(String mediaURI)**: Returns all original MPEG-7 descriptions whose *MediaURI* contained in the *MediaLocator* of the top-level video segment matches the given *mediaURI* parameter.
- **String searchSegments(String [] phrases, String semPlace, String from, String to, boolean and, String mediaURI)**: Returns, depending on the given parameters (see below for an explanation), all video segments and the corresponding video description as a valid MPEG-7 document.
- **String searchVideos(String [] phrases, String semPlace, String from, String to, boolean and)**: Returns only the video description that corresponds to the matching video segments.

### Parameters:

- *String [] phrases*: This parameter contains a list of phrases to search for in the *FreeTextAnnotation* and *KeywordAnnotation* descriptor of video segments. The phrases have to be combined logically using the given *and* parameter
- *String semPlace*: This parameter has to be matched against the *SemanticPlace* descriptor of video segments.

---

<sup>1</sup><http://ftp2-itec.uni-klu.ac.at/~mt/dahl/>

<sup>2</sup><http://cs-exhibitions.uni-klu.ac.at/>

- *String from*: This parameter represents the beginning of a time period and can be given in the following formats (YYYY, YYYY-MM-DD). The given data is matched against the *SemanticTime* descriptor of video segments.
- *String to*: This parameter represents the end of a time period and can be given in the same format as specified above. It also is matched against the *SemanticTime* descriptor of video segments.
- *String mediaURI*: This parameter restricts the search to videos whose *MediaURI* element is equal to the given data.

Furthermore, the *Video library* introduces two new operators for comparing the given time period (*from*, *to* parameters) with the stored *SemanticTime* descriptor. The *timeAfter* operator returns 1, iff the second parameter is temporally after the first parameter. The *timeBefore* operator provides the inverted functionality. Both operators are used in the where clause of a query and delegate their execution to an internal Java class.

```
CREATE OR REPLACE OPERATOR timeAfter BINDING (varchar2, varchar2) RETURN NUMBER USING PLS_timeAfter;
```

```
CREATE OR REPLACE OPERATOR timeBefore BINDING (varchar2, varchar2) RETURN NUMBER USING PLS_timeBefore;
```



# CHAPTER 11 Applications

This chapter presents our MPEG-7 MMDB by the use of two applications in the field of blob based image retrieval and audio recognition (see Figure 11.1).

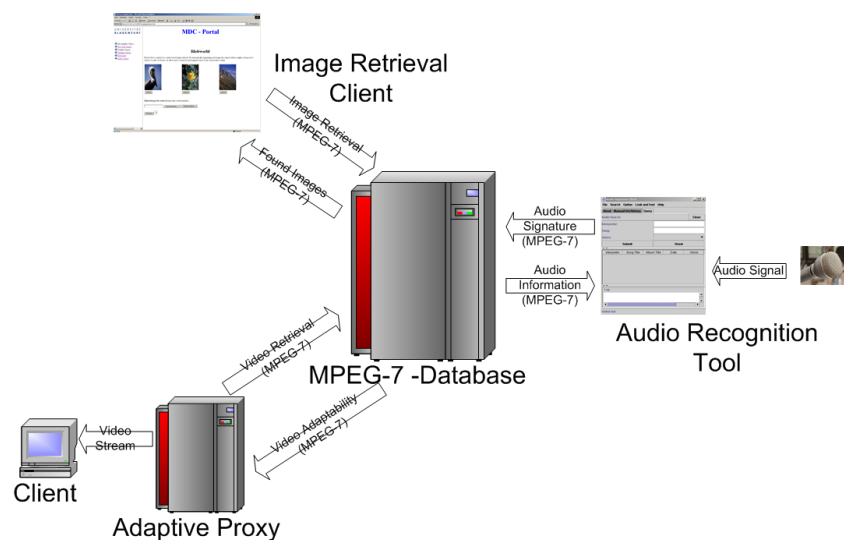


Figure 11.1: Multimedia applications based on MPEG-7 MMDB

In the following, Section 11.1 describes an image retrieval system that relies on the blob theory [7, 1] which extends the original version, implemented at University of California in Berkeley<sup>1</sup>. Section 11.2 introduces a new audio recognition tool, that on the one hand enables an user to retrieve the database for music compositions selected by interpreter, song title or genre and on the other hand allows one to identify unknown music compositions by the use of a 10 seconds fingerprint.

<sup>1</sup><http://datasplash.cs.berkeley.edu:8000/gist/>

## 11.1 MPEG-7 Blobworld

In this Section, our content based image retrieval system, MPEG-7 Blobworld, is introduced. Subsection 11.1.1 gives an introduction of the original system MPEG-7 Blobworld relies on. In Subsection 11.1.2 the architecture of our system is presented. Besides, this Subsection also illustrates the segmentation process and the similarity search. The following Subsection 11.1.3 shows a possible MPEG-7 Blobworld query and the resulting matches. This Section is concluded by an performance evaluation between the original system and MPEG-7 Blobworld.

### 11.1.1 Introduction

MPEG-7 Blobworld adapts the original system [7, 1, 183], which was developed at the University of California in Berkeley, to perform with our MPEG-7 MMDB. Therefore, almost every part of the original system has been extended or completely re-implemented. MPEG-7 Blobworld is a content based image retrieval system that relies on the blob theory. Blobs are regions which are roughly homogeneous with respect to color and texture. The aim of the blob-segmentation is to recognize objects (eg., the sky, a car, an animal) within an image automatically. Based on this segmentation, the user is integrated within the retrieval process by two facts. First, the internal representation is not hidden by the system and the user can choose between different blobs. Second, by demonstrating the original image with their blobbed version and all found matches, the user can understand and optimize the retrieval functionality lot easier.

In contrast to the original system, MPEG-7 Blobworld provides three main improvements. First, the images and their corresponding blobs are described with MPEG-7 descriptors. This results in the advantage, that in contrast to the use of a proprietary format, the resulting descriptions can be forwarded to any application, database, etc. that understands MPEG-7. Second, the MPEG-7 descriptions are stored and indexed by our MPEG-7 MMDB which increases the system's performance, availability, useability and scalability. Third, in contrast to the original system, MPEG-7 Blobworld provides means for enhancing the image pool at runtime. For this purpose, the segmentation algorithm has been extended, so that it can

be used online by our MPEG-7 Blobworld client. The resulting MPEG-7 description of the image can either be used as reference image for retrieval or can be inserted into the database. In the following, the architecture of the MPEG-7 Blobworld system is introduced.

### 11.1.2 Architecture of the System

The architecture of MPEG-7 Blobworld is illustrated in Figure 11.2 and consists of the following parts. First, each image of the collection has to be segmented in regions based on the algorithm explained in Subsection 11.1.2.1. The algorithm is implemented in the MatLab environment and returns for each image a separate file containing all information (color, shape, texture features) for all blobs. This file is transferred into a MPEG-7 description where each blob is described with the *Still-Region* descriptor and the extracted features with their respective MPEG-7 counterparts (*ScalableColor*, *RegionShape* and *HomogeneousTexture*). The MPEG-7 descriptions are inserted into the MPEG-7 MMDB with the help of the *InsertLib* (see Section 9.2). The similarity search relies on an index based k-NN-retrieval provided by our database. For this purpose the application library *BlobworldLib* (see Section 10.1) provides two methods (*getBlob()* and *getNearestBlob()*). The result of the similarity search is given as a MPEG-7 description and has to be parsed (*parseXML()*) in order to be presented to the user.

#### 11.1.2.1 Segmentation of Images into regions

The segmentation process of MPEG-7 Blobworld basically relies on the same process as given in [2, 7]. The main difference is that our extension allows the segmentation of query images online so that it can be used as an reference image for further queries. The segmentation process shown in Figure 11.3 consists of the following steps: First, the color, texture and position features are extracted based on a predefined scale. Second, the pixels are grouped into regions by using Gaussian distribution and Expectation-Maximization. Third, the regions are described, based on their color distribution and texture, for enabling blob-based image retrieval. The main problem of the given approach is that borders of regions not necessarily belong to borders of objects. This is the case if color and/or texture differences are small. A successful



the windowed second moment matrix [184]. A complete description of the feature extraction process is out of scope of this thesis, but can be found at [2, 22, 185]. The resulting feature vector contains six values: the three color coordinates ( $L^*a^*b$ ) and the three texture values (polarity, anisotropy and contrast). The feature vector of a pixel is concluded by two values ( $x,y$ ) for its position.

**Group features:** The feature extraction process results in a large amount of 8-dimensional feature vectors. In order to group these vectors, the segmentation process makes use of the Expectation-Maximization (EM) algorithm for finding maximum likelihood parameters. Spatial grouping of those pixels that belong to the same color/texture cluster is realized by a connected-components algorithm that interprets the highest likelihood calculated before. See [2, 22, 185] for detailed information.

**Describe regions** In the original system, the characteristics of a blob are described with a maximum of 218 color values and the texture values for contrast and anisotropy. The polarity description is not included. In addition, the blob description contains two values for the blob position and 40 shape values. These values are stored in a simple text file. The MPEG-7 Blobworld uses for describing the extracted features MPEG-7 descriptions. A blob within an image is described by the *StillRegion* descriptor and a unique ID (within the image description) as attribute. The ( $x,y$ ) position of a blob is denoted with the *SpatialLocator* descriptor. The color, shape and texture feature vectors are characterized with the respective MPEG-7 descriptors (*ScalabelColor*, *RegionShape* and *HomogeneousTexture*). Note: the segmentation algorithm does not provide MPEG-7 conform feature extraction. Nevertheless, this was accepted to be comparable in the performance analysis where we want to use the same data basis.



Figure 11.4: Good result of segmentation algorithm



Figure 11.5: Bad result of segmentation algorithm

### 11.1.2.2 Retrieval of similar Images

The resulting MPEG-7 descriptions of the segmentation process are inserted into our MPEG-7 MMDB. There, all descriptors representing a blob and their features (e.g., *StillRegion* or *ScalableColor*) are stored in their corresponding database tables and types (see Section 9.2). In MPEG-7 Blobworld, similarity search relies on a NN-retrieval for the features color, shape and texture. For this purpose, our integrated multimedia indexing framework (MIF) (see Section 8.3) provides several access methods for realizing CBIR functionality. As illustrated in Figure 11.6, we use an R-tree for indexing texture and shape feature vectors and an SS-tree for indexing color. The retrieval process proceeds as follows: Based on the given reference blob containing values for the three features (color, texture and shape) a k-NN-Search is initiated. The resulting k-NNs are weighted depending on the given user preferences and merged together sorted by their occurrences in the respective result sets. For instance, a stored blob has very similar color and texture features in comparison to the reference blob then the same blob will probably occur in both result sets (color and texture). During the merge process this blob will be ranked higher than blobs that occur only once in the three resulting sets. After merging all three result sets, we have a maximum of  $3K$  and a minimum of  $K$  different blobs. The top  $K$  elements are returned to the user.

### 11.1.3 MPEG-7 Blobworld Query

MPEG-7 Blobworld provides two different query scenarios. On the one hand an user can choose one of the predefined images that are already stored in the database or on the other hand the user can upload the reference image to the server where it is segmented based on the described process in Subsection 11.1.2.1. In both cases the

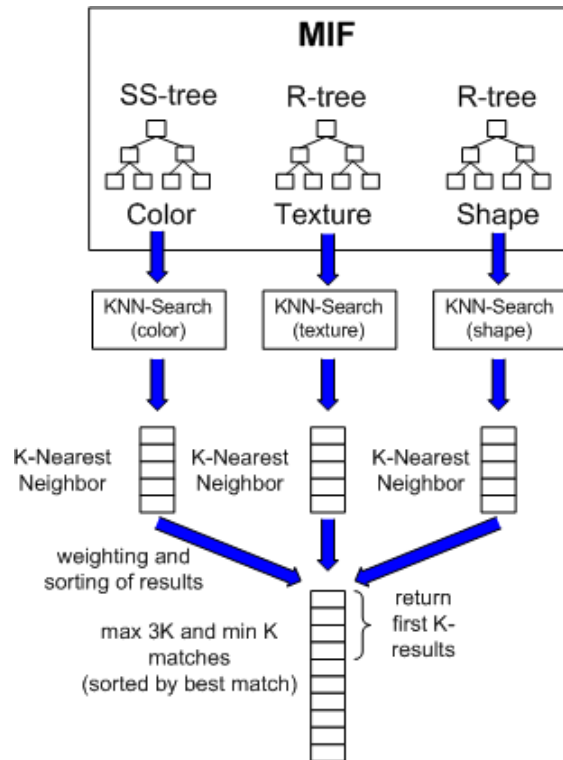


Figure 11.6: KNN-retrieval for MPEG-7 Blobworld within MPEG-7 MMDB

user retrieves the original image and the respective blobbed version for choosing the desired region (see Figure 11.7). Based on this region, the user can specify the desired quality rating for the three features (color, texture and shape). The rating provides three settings (*not*, *somewhat* and *very*). Finally, the user can define the amount of returned matches. The collected data (reference blob as MPEG-7 description, quality rating and size of result set) is send via the *BlobworldLib* (see Section 10.1) to the MPEG-7 MMDB where the mentioned k-NN-retrieval (see Subsection 11.1.2.2) is executed. The found matches are presented to the user as illustrated in Figure 11.8. They can be used for further database search.

#### 11.1.4 Performance Evaluation

In this Subsection the performance of both systems, MPEG-7 Blobworld and the original one, is compared. The measurements rely on a k-NN-Search ( $k=10$ ) of the features (color, texture and shape) for an image collection that contains between



Figure 11.7: Original image and their blobbed version

500, 1000, 1500 and 2000 different elements. The features, position and background provided by the original system, have not been considered.

The original system provides two different adjustments.

- *Similarity search without filtering*: Similarity search in the original system is realized by a sequential scan of binary files that contain all blobs and their respective feature values. These are a maximum of 218 color values, two texture values and 40 shape values. The distance of nearest neighbors is calculated with the help of the Mahalanobis function.
- *Similarity search with filtering*: To improve performance, the original system provides a specific filtering functionality. A filtering file is created during the segmentation process which contains for every blob, two texture values, four shape values and five color values. K-NNs are determined based on the same strategy as explained above, whereas the amount of values, that have been compared, is now based on the filtering file reduced to 11. The found matches are precisely rearranged by comparing the original values.

The response time for the similarity search of the original system is presented in table 11.1. The response time is calculated as the mean of the time used for 10 queries. Figure 11.9 (a) demonstrates the results graphically.

The measurements of the similarity search of the MPEG-7 Blobworld bases on the following two points.



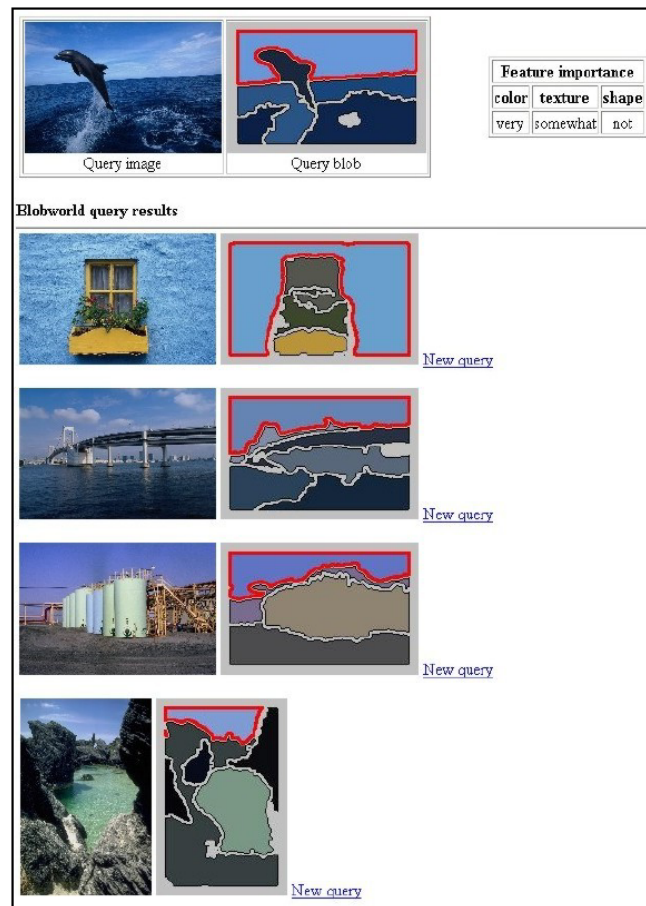


Figure 11.8: Result of blob based image retrieval

- *Similarity search without parsing*: Similarity search in MPEG-7 Blobworld relies on the same features depicted in the original system without filtering and is explained in detail in Subsection 11.1.2.2. MPEG-7 Blobworld forwards the reference blob to the database as a MPEG-7 description. Therefore, this description has to be parsed in order to extract the necessary information for the three features. The time used for parsing is not included in this measurement.
- *Similarity search with parsing*: The communication between clients and the MPEG-7 MMDB relies on MPEG-7 descriptions. The time used for parsing at the database is explicitly integrated into the similarity search measurements.

The response time for the similarity search of MPEG-7 Blobworld is presented in

Amount of images	NN-retrieval with filtering	NN-retrieval without filtering
500	379,7	420,3
1000	426,6	545,4
1500	492,3	670,3
2000	528,1	798,2

Table 11.1: Performance results of original system

table 11.2. The equivalent graphically representation is shown in Figure 11.9 (b).

Amount of images	NN-retrieval without parsing	NN-retrieval with parsing
500	1601,3	1851,5
1000	1682,7	1882,9
1500	1813,8	2014,1
2000	1824,9	2053,1

Table 11.2: Performance results of MPEG-7 Blobworld

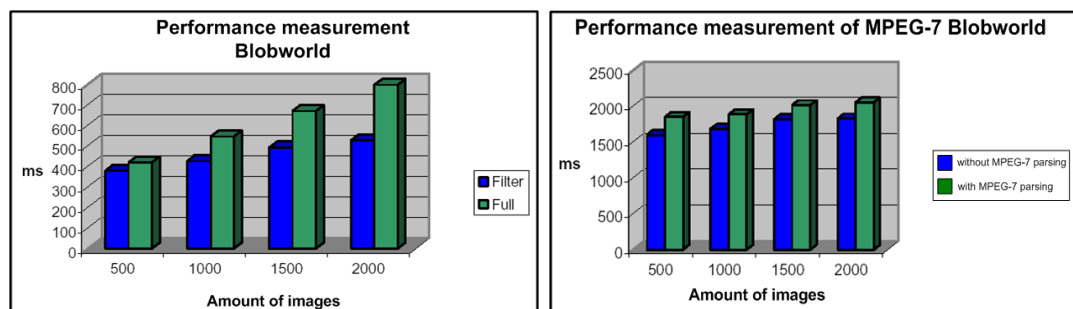


Figure 11.9: Performance results of (a) original system and (b) MPEG-7 Blobworld

By comparing the results, one can recognize that the original system beats the MPEG-7 Blobworld by a factor of 2-3. The reasons are twofold. On the one hand a sequential scan is very fast for few images and on the other hand the overhead produced by the MPEG-7 MMDB for parsing MPEG-7, communicating with the indexes etc. is too large for this small amount of images. But Figure 11.10 shows that for large image collections the sequential scan will become a bottleneck. The increase in percent for the similarity search of 500 images to 2000 images is roughly at 90 percent for the original system, whereas the MPEG-7 Blobworld registers only an increase of 13 percent. This directly results in a better scalability in contrast to the original system for large image collections.

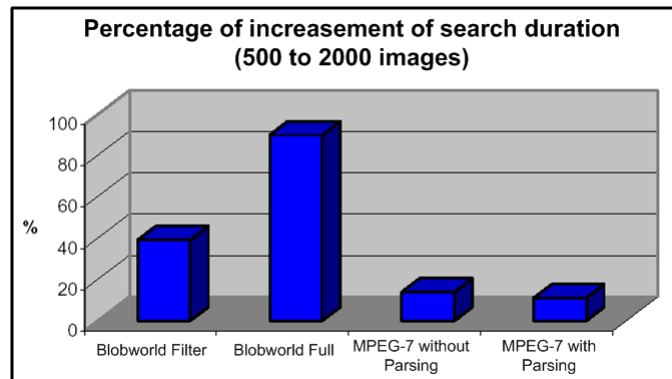


Figure 11.10: Performance comparison of both systems

## 11.2 Audio Recognition

In this Section, our audio recognition system is introduced. Subsection 11.2.1 starts with an introduction of the system. Subsection 11.2.2 presents the theory behind audio recognition. In the following, Subsection 11.2.3 demonstrates how audio recognition is realized with the help of our MPEG-7 MMDB.

### 11.2.1 Introduction

The Java based audio recognition tool (see Figure 11.12) enables one to search for music compositions in our multimedia database. The developed tool offers two kind of search techniques. The first search technique is based on a search of music title, interpreter and genre. This information is described with the *CreationInformation* descriptor of MPEG-7 and internally stored in respective database tables and types. The used MPEG-7 description of music compositions is presented in Figure 11.11. The second search technique (depicted in detail in Section 11.2.3) uses audio signals that are specified with the *AudioSignature* descriptor. The audio signal is recorded with a microphone, converted to MPEG-7 descriptions and forwarded to the *AudioLib* (see Section 10.2). The result of both techniques is given in the MPEG-7 format and presented by the audio tool.

```

<Mpeg7>
  <Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="AudioType">
      <Audio xsi:type="AudioSegmentType">
        <MediaLocator>
          <MediaUri>
            Avril Lavigne - Complicated.mp3
          </MediaUri>
        </MediaLocator>
        <CreationInformation>
          <Creation>
            <Title type="songTitle">Complicated</Title>
            <Title type="albumTitle">Let Go (Advance)</Title>
          </Creation>
          <Creator>
            <Role href="RoleCS">
              <Name>Artist</Name>
            </Role>
            <Agent xsi:type="PersonGroupType">
              <Name>Avril Lavigne</Name>
            </Agent>
          </Creator>
          <CreationCoordinates>
            <Date>
              <TimePoint>2002</TimePoint>
            </Date>
          </CreationCoordinates>
        </CreationInformation>
        <Classification>
          <Genre href="urn:mpeg:mpeg7:cs:ContentCS:2001:1.3.1">
            <Name>Pop</Name>
          </Genre>
        </Classification>
        <AudioDescriptionScheme xsi:type="AudioSignatureType">
          <Flatness hiEdge="4000.0" loEdge="250.0">
            <SeriesOfVector hopSize="PT30N1000F" vectorSize="16" totalNumOfSamples="8192">
              <Scaling ratio="32" numOfElements="256"/>
              <Mean mpeg7:dim="256 16">
                0.77140942 0.71303663 0.73747904 0.71532651 ...
              </Mean>
              <Variance mpeg7:dim="256 16">
                0.02713493 0.04134227 0.01990115 0.04246818 ...
              </Variance>
            </SeriesOfVector>
          </Flatness>
        </AudioDescriptionScheme>
      </Audio>
    </MultimediaContent>
  </Description>
</Mpeg7>

```

Figure 11.11: Music composition described with MPEG-7

### 11.2.2 Theory of audio recognition

Beside information about e.g., format, storage, author or producer, the MPEG-7 description of a audio content can also include descriptions of the signal on a more or less abstract level.

The *AudioSignature Description Schema* is specialized to describe the content of a piece of music for identification. In spite of its compact size it was shown that it is very robust against most common modifications [186, 187] of a piece of music like:

- filtering (lowpass, highpass, bandpass, equalizer, ...)
- perceptual audio coding (mp3, ogg, ...)
- compressor, limiter (radio stations)
- additive noise

- amplitude change
- random start

**Applied Audio Signature Description** In order to create the *AudioSignatureDS* the average flatness of the spectrum is determined approximately once per second (default) for 16 sub-bands (default). Small values for flatness indicate peaks in the spectrum which is typical for tonal components; values close to one refer to flat spectrum corresponding to a noise-like or an impulse-like signal. The temporal resolution and the bandwidth of the *AudioSignatureDS* can be varied to find a good tradeoff between the size of the description schema and its robustness.

The *AudioSignature DS* consists of two matrices. Based on both matrices, a feature vector for each segment can be defined. For each segment of 10 seconds of music a feature vector is created by arranging the values of both matrices and reading the values of the corresponding rows of the first matrix and appending the rows with the same index of the second matrix.

Note: The order of the values of the feature vector has no influence on the performance of the identification algorithm. It can be adapted to the memory model of the application. If the matrices are stored column-major it might be more performant to read the values of the corresponding segment columnwise.

**Similarity between two segments of music** With the feature vectors described above the similarity between two segments can be defined by the *Mahalanobis distance* (11.1).

$$(q - s_{n,\Delta t})^T R_{xx}^{-1} (q - s_{n,\Delta t}) \quad (11.1)$$

with:

$q$ : feature vector of query

$s_{n,\Delta t}$ : feature vector of song number  $n$  after  $\Delta t$  seconds

$R_{xx}$ : Covariance matrix (11.2)

$$R_{xx} = \begin{pmatrix} Var(v_1) & Cov(v_1, v_2) & \dots & Cov(v_1, v_n) \\ Cov(v_2, v_1) & Var(v_2) & \dots & Cov(v_2, v_n) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(v_N, v_1) & Cov(v_N, v_2) & \dots & Var(v_N) \end{pmatrix} \quad (11.2)$$

The covariance matrix  $R_{xx}$  (11.2) is a  $N \times N$ -matrix ( $N$ : length of feature vector). It describes the covariance between each pair of dimensions of the feature vector. It depends on the channel with which the music signal is modified and has to be measured or estimated.

Sub-bands with a low distortion are weighted stronger than sub-bands with a high distortion. The *Mahalanobis distance* takes also into account the correlation between different dimensions especially the covariance between successive values of the same sub-band and the correlation between values of adjoining sub-bands. How the *Mahalanobis distance* can be transformed to a less complex *Euclidian distance* is described in [188].

### 11.2.3 Music Composition Recognition with the help of the MPEG-7 MMDB

The music composition recognition relies on the following architecture: During insertion of a new music composition, the audio signal described with the *AudioSignature* descriptor, is split into pieces corresponding to fragments of 10 seconds of an audio signal. From each fragment an 160-dimensional vector is computed. To take into account the effect of audio signal inaccuracy, we have to recalculate the given vector based on the approach described in Section 11.2.2. Further, we use the concept of a singular value decomposition (SVD) to reduce the dimensionality of each fragment to an 11-dimensional feature vector. This vector is inserted into an R-tree provided by our multimedia indexing framework (MIF) [81]. In addition we insert the reduced feature vector (11 dimension), the original vector (160 dimension), the rowid of the music composition and an unique id into a temporary table which is necessary for the retrieval process.

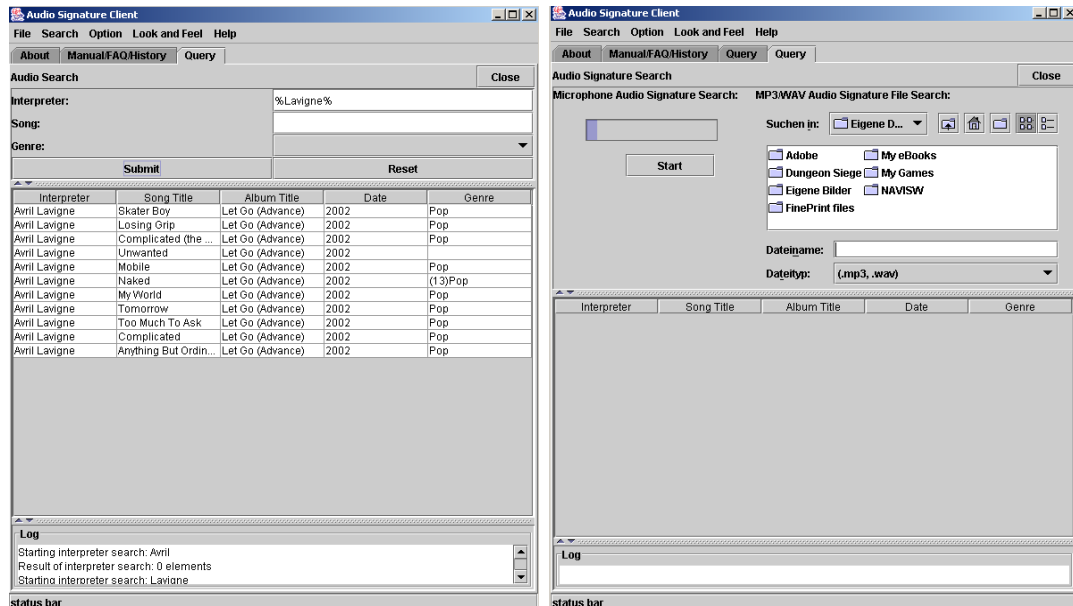


Figure 11.12: Music composition retrieval based on (a) interpreter and (b) audio signal

As mentioned above our audio recognition tool provides two retrieval functionalities.

- **Search by interpreter, song title or genre:** Besides searching for an unknown music composition, our Java tool provides the functionality for browsing the database for a certain interpreter, song title or genre. The necessary information such as interpreter or song title are described with the respective MPEG-7 descriptors *PersonGroup* and *Title* which are subelements of the *CreationInformation* descriptor (see Figure 11.11). The query performance is increased by using Oracles Text Index for the respective database tables. Figure 11.12 (a) shows a screen shot presenting the result of an interpreter retrieval.
- **Search by audio signal:** The query process within our MPEG-7 MMDB for audio recognition (see Figure 11.12 (b)) is realized by the following means. The audio recognition tool retrieves audio signals with a duration of 10 seconds of

an unknown music composition which is recorded with the help of a microphone or an equivalent input device. Further, the signal is transformed and described with the *AudioSignature* descriptor. The resulting MPEG-7 description (see Figure 11.11) is forwarded to the *AudioLib* (see Section 10.2), where the document is parsed and a music composition retrieval is generated. For retrieval, our internal audio index implementation is used with the given audio signature. The audio signature is recalculated and reduced in dimension (see above). The resulting 11-dimensional vector is used as input for the NN-search provided by the MIF. The found music composition is returned to the *AudioLib* library. Further, the library selects all additional available information (*CreationInformation* descriptor) of the music composition and creates an MPEG-7 description. This document is forwarded to the audio recognition tool, where it is parsed and presented.



# CHAPTER --- 12 Summarization and Future Research

## 12.1 Summarization

This thesis successfully provided solutions for the following research questions which we raised at the beginning of this thesis:

- *How to map the MPEG-7 standard to an equivalent database schema?*

We developed guidelines for mapping the MPEG-7 standard to an equivalent object-relational database schema. The guidelines made use of existing object-relational database features such as inheritance, polymorphism, domain-specific types, etc. and the database internal XMLTYPE for the development of a semi-automatic mapping approach. In addition, these guidelines contained solutions for XML-Schema specific constructs such as recursions, sequences, choices, etc. Our mapping strategy was driven by the fact, that selected MPEG-7 descriptors (e.g., TextAnnotationType) have been modelled more precisely than others. The decision was oriented by the fact that several descriptors contain few information about multimedia data (e.g., UsageInformation). In the following, these descriptors have been generalized with the XMLTYPE without losing their information for the retrieval process. Besides, we installed a key system which enables the assignment of each tuple to their corresponding MPEG-7 descriptions.

- *How to enable indexing of multi-dimensional data within an extensible database system?*

We proposed an extension of the indexing mechanism of extensible database systems based on our new *MIF (Multimedia Indexing Framework)* which includes the GiST-environment. Besides the execution of exact-match queries, this framework also supplied more multimedia specific operations, like Range-Search, NN-Search, Overlap etc. on multi-dimensional data. Thus, it meets more precisely the requirements of multimedia search and filter applications than the broadly used DBMS Extenders (e.g., DB2 Extenders). We furthermore demonstrated that a new index type could be instantiated with only few steps using the Data Cartridge technology in combination with our MIF-framework. Finally, the experimental analysis showed that the build-in indexes of Oracle were outperformed for exact match queries. Moreover, the performance for an NN-search with MIF were comparable to those of the exact match. The performance gain in combination with our extension mechanism (support for similarity search) stated the effectiveness of our approach.

- *How to optimize multimedia queries?*

The task of optimizing multimedia queries cover research of cost models, operator execution and selectivity. The first two are well investigated (e.g., [91, 92]) hence, we concentrated our research efforts on the selectivity of multimedia queries. In detail, we introduced an approach for approximating the selectivity of Range queries. Our approach used the DBSCAN clustering technique to find high density areas in the data set. The selectivity was approximated with the help of a density function in combination with the volume of the query's hypersphere. The approach was evaluated on three different data sets, namely synthetic data, city coordinates and color histograms of images. The evaluation showed that on average the difference to the real selectivity was minimal and did not exceed  $1/3$ . In most cases (for the random and image data set), the difference is around 15%. In addition, it was shown that the query plan and hence the performance of queries was improved by the use of our approach for approximating the selectivity of Range queries.

- *How to define and implement a programmable interface for a multimedia database?*

In order to enable an effective use of our MPEG-7 MMDB, we introduced a set of internal libraries that act as interface for multimedia applications. In general, we applied libraries for initialization, insertion, deletion and querying of MPEG-7 descriptions within our database. The initialization library provided means for creating a new instance of our MPEG-7 MMDB. The insertion library implemented algorithms for splitting MPEG-7 descriptions according to the introduced MPEG-7 database schema. The deletion library allowed the deletion of a set of MPEG-7 descriptions according their ID. The query library provided means for creating individual select statements with the help of our modular construction system. This modular construction enabled users to specify the select part as well as the where part according to their needs. The output was formatted by the use of the MPEG-7 standard which made it usable for any application that is able to process MPEG-7 descriptors. The functionality of the internal libraries had been combined within our MDC database type in order to allow a bunched access to the underlying data.

In addition to the main research topics, the use of our MPEG-7 database was demonstrated by two multimedia applications (audio recognition and image retrieval) that operate on the corresponding application libraries. These application libraries act as interfaces between the internal libraries and the applications.

## 12.2 Future Research

This thesis describes a complete approach for storing, indexing and querying of MPEG-7 descriptions. Nevertheless, there are some open issues in this area which are rudimentary listed below.

- MPEG-21: as MPEG-7 is partly integrated in the MPEG-21 standard, it has to be investigated how on the one side the existing solution can be used for MPEG-21 and on the other side what kind of extensions have to be made in order to allow the processing of MPEG-21 descriptions. As stated in Section 3.5, the core element of MPEG-21 is the *Digital Item (DI)*. A DI is a hierarchical container consisting of multimedia data, multimedia metadata and other digital

items. Especially, the multimedia metadata (which does not necessarily has to be MPEG-7) is the interesting part our database should be able to process. In the case of MPEG-7, the integration and process of an DI should be warranted.

In addition, we have to mention other core features of MPEG-21 such as the Digital Item Adaptation (DIA), the Rights Expression Language (REL) and the Intellectual Property Management and Protection Language (IPMP) all based on XML Schema. The integration of several of the mentioned XML Schemas into our MMDB would be advantageous. Especially, the IPMP and REL are of interest which could be used to realize a multimedia access and control system within our MMDB. In addition, the integration of DIA could realize an internal adaptation engine for the stored MPEG-21 descriptions.

Two issues for integrating the mentioned parts have to be considered:

- The methods for mapping MPEG-7 to a database schema could be analogically applied to these XML Schemas.
  - To build an access and control system, application logic for managing REL and IPMP expressions has to be considered. This task is ideally realized in the internal libraries.
- 
- XML-Schema: MPEG-7 relies on XML-Schema. Until now, there exist no sufficient solution how to map automatically a complex XML-Schema (e.g. MPEG-7) to a ORDBMS model. In our approach we used a semi-automatic mapping strategy. It has to be investigated how this strategy can be transferred into a complete automatic mapping process. One approach would be the introduction of a namespace that declares for all MPEG-7 descriptions if they should be considered as XMLTYPES or database tables. The combination with our mapping rules would lead to an automatic mapping process.
  - Query Performance: Our MPEG-7 MMDB was evaluated by the use of two multimedia applications in the area of blob based image retrieval and audio recognition. The evaluation was restricted by the following sense. On the one hand the amount of participating MPEG-7 descriptions (i.e., several hundreds for the image retrieval tool) was limited and on the other hand the diversity

of used descriptors was restricted (i.e., the audio recognition application used around 10 different MPEG-7 descriptors and descriptor schemes for describing the necessary information). In a second step, this evaluation has to be extended to a higher number of MPEG-7 descriptions. This could be done in the context of the *DAHL* project (see <http://www-itec.uni-klu.ac.at/~mt/dahl/>) which was funded to develop a web application that provides content-based search mechanism and an adaptive video streaming environment for the virtual computer science exhibition. This exhibition is in memory of Ole-Johan Dahl, Edsger Wybe Dijkstra and Kristen Nygaard and was established at the University of Klagenfurt (see <http://cs-exhibitions.uni-klu.ac.at/>). In this context, an overall evaluation of the performance of our database in respect to all participating parts has to be performed (e.g., indexing facility, extended optimizer, internal libraries, etc.).

- Security and legal effects: In general, our research concentrates on realizing the main goals of storing, indexing and querying of MPEG-7 descriptions. No research has been done so far on the topics of security issues within the database or legal effects by the use of MPEG-7. The security issues cover for instance the realization of a limited access to the stored descriptions. A possible scenario is that all users of an imaginary group A are only allowed to access the information that is stored in the image sub-part but should not have access to the appropriate video or audio information. This restriction is due to missing licence credits in the MPEG-7 standard. The integration of the MPEG-21 IPMP-tool gives us the necessary meta-data to manage legal effects. Another important part is the streaming of MPEG-7 descriptions. In the current version, the outgoing query results are streamed as plain text. Here, we have to think about encryption of the streamed MPEG-7 descriptions. This is related to the streaming of the binary format of MPEG-7 (BiM). Furthermore, the insert engine should also be enhanced for supporting this binary format.

# Bibliography

- [1] M. Thomas, C. Carson, and J. M. Hellerstein. Creating a Customized Access Method for Blobworld. In *16th Int. IEEE Conf. on Data Engineering*, page 82, San Diego, CA, March 2000. IEEE Computer Society 2000.
- [2] Chad Carson, Serge Belongie, Hayit Greenspan, and Jitendra Malik. Blobworld: Image segmentation using Expectation-Maximization and its application to image querying. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 24(8):1026–1038, August 2002.
- [3] Leonardo Chiariglione. ISO/IEC JTC1/SC29/WG1 - Short MPEG-1 description. <http://www.chiariglione.org/mpeg/standards/mpeg-1/mpeg-1.htm>, June 1996.
- [4] Leonardo Chiariglione. ISO/IEC JTC1/SC29/WG11 - Short MPEG-2 description. <http://www.chiariglione.org/mpeg/standards/mpeg-2/mpeg-2.htm>, October 2000.
- [5] Rob Koenen. N4668 - Overview of the MPEG-4 Standard (v.21). <http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm>, March 2002.
- [6] A. N. Skodras, C. A. Christopoulos, and T. Ebrahimi. JPEG2000: The Upcoming Still Image Compression Standard. In *Proceedings of the 11th Portuguese Conference on Pattern Recognition (RECAP00D 20; invited paper)*, pages 359–366, Porto, Portugal, May 2000.
- [7] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik. Blobworld: A System for Region-Based Image Indexing and Retrieval. In *Third International Conference on Visual Information Systems*, pages 509–517, Amsterdam, The Netherlands, June 1999. Springer Verlag.

- 
- [8] A. Woudstra, D. D. Velthausz, H. J. G. de Poot, F. Moelaert El-Hadidy, W. Jonker, M. A. W. Houtsma, R. G. Heller, and J. N. H. Heemskerk. Modeling and Retrieving Audiovisual Information - A Soccer Video Retrieval System. In *Proceedings of the 4th International Workshop for Advances in Multimedia Information Systems*, pages 161–173, Istanbul, Turkey, 1998.
- [9] Simone Santini and Ramesh Jain. Image Databases are not Databases with Images. In *Proceedings of the 9th International Conference on Image Analysis and Processing 2*, pages 38–45, Florence, Italy, 1997.
- [10] Oracle. Oracle interMedia Reference, 10g Release 1. <http://download-east.oracle.com/docs/>, December 2003.
- [11] IBM. IBM DB2 Universal Database - Image, Audio and Video Extenders Administration and Programming, Version 7. <http://www-306.ibm.com/software/data/db2/>, 2000.
- [12] R. Tusch, H. Kosch, and L. Böszörményi. VIDEX: An Integrated Generic Video Indexing Approach. In *ACM Multimedia Conference 2000*, pages 448–451, Los Angeles, USA, November 2000.
- [13] J. M. Martinez, R. Koenen, and F. Pereira. MPEG-7. *IEEE Multimedia*, 9(2):78–87, April-June 2002.
- [14] Harald Kosch. *Distributed Multimedia Database Technologies supported by MPEG-7 and MPEG-21*. CRC Press, November 2003. 248 pages, ISBN: 0-849-31854-8.
- [15] Harald Schöning. Tamino - a DBMS designed for XML. In *Proceedings of the 17th International Conference on Data Engineering (ICDE)*, pages 149–154, April 2001.
- [16] H. V. Jagadish, Shurug Al-Khalifa, Adriane Chapman, Laks V.S. Lakshmanan, Andrew Nierman, Stelios Pappas, Jignesh M. Patel, Divesh Srivastava, Nuwee Wiwatwattana, Yuqing Wu, and Cong Yu. TIMBER: A Native XML Database. *The VLDB Journal*, 11(4):274–291, 2002.
- [17] M. A. Stricker and M. Orengo. Similarity of color images. In *Storage and Retrieval for Image and Video Databases, SPIE*, pages 381–392, San Jose, CA, 1995.

- 
- [18] Marius Tico, Taneli Haverinen, and Pauli Kuosmanen. A Method of Color Histogram Creation for Image Retrieval. In *NORSIG 2000, IEEE Nordic Signal Processing Symposium*, pages 157–160, Kolmarden, Sweden, 2000.
- [19] Yong Rui and Thomas S. Huang and Shih-Fu Chang. Image Retrieval: Past, Present, and Future. *Journal of Visual Communication and Image Representation*, 10:1–23, 1999.
- [20] John P. Eakins and Margaret E. Graham. Content-based Image Retrieval. Technical report, Institute for Image Data Research, University of Northumbria, Newcastle, Great Britain, 1999.
- [21] Sharad Mehrotra, Yong Rui, Michael Ortega-Binderberger, and Thomas S. Huang. Supporting Content-based Queries over Images in MARS. In *Proceedings of the 1997 International Conference on Multimedia Computing and Systems (ICMCS '97)*, page 632, 1997.
- [22] Serge Belongie, Chad Carson, Hayit Greenspan, and Jitendra Malik. Color- and Texture-Based Image Segmentation Using EM and Its Application to Content-Based Image Retrieval. In *Proceedings of the International Conference on Computer Vision (ICCV'98)*, pages 675–682, Bombay, India, 1998.
- [23] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Sashley, Qian Huang, Byron Dom, Monika Gorkani, Him Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker. Query by Image and Video Content: The QBIC System. *IEEE Computer*, 28(9):46–52, 1996.
- [24] John R. Smith and Shih-Fu Chang. Single Color Extraction and Image Query. In *Proceedings of the 1995 International Conference on Image Processing (Vol. 3)-Volume 3 - Volume 3*, Washington, DC, USA, 1995. IEEE Computer Society.
- [25] Arnold W.M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-Based Image Retrieval at the End of the Early Years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349 – 1380, 2000.
- [26] Chitra Dorai and Svetha Venkatesh. Bridging the Semantic Gap in Content Management Systems: Computational Media Aesthetics. In *Proceedings Conf. on Computational Semiotics for Games and New Media*, pages 94–99. Amsterdam, Netherlands, 2001.



- 
- [27] Chih-Fong Tsai. Stacked Generalization: A Novel Solution to Bridge the Semantic Gap for Content-Based Image Retrieval. *Online Information Review*, 27(6):442–445, 2003.
- [28] Horst Eidenberger and Christian Breiteneder. Semantic Feature Layers in Content-based Image Retrieval: Implementation of Human World Features. In *Proceedings of International Conference on Control, Automation, Robotic and Vision*, Singapore, Singapore, 2002.
- [29] Michael S. Lew, Nicu Sebe, and John P. Eakins. Challenges of Image and Video Retrieval. In *Proceedings of the International Conference on Image and Video Retrieval*, pages 1–6, 2002.
- [30] Xiang Sean Zhou and Thomas S. Huang. CBIR: From Low-Level Features to High-Level Semantics. In *Proceedings SPIE Vol. 3974, Image and Video Communications and Processing*, pages 426–431, 2000.
- [31] Riccardo Leonardi and Pierangelo Migliorati. Semantic Indexing of Multimedia Documents. *IEEE Multimedia*, 9(2):44–51, 2002.
- [32] Peng Chang and John Krumm. Object Recognition with Color Cooccurrence Histograms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 2498–2504, Fort Collins, CO, USA, 1999.
- [33] Richard Campbell and John Krumm. Object Recognition for an Intelligent Room. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 1691–1698, Hilton Head, SC, USA, 2000.
- [34] Christopher M. Cyr and Benjamin B. Kimia. 3D Object Recognition Using Shape Similiarity-Based Aspect Graph. In *Proceedings of the International Conference on Computer Vision*, pages 254–261, 2001.
- [35] Rene Visser, Nicu Sebe, and Erwin Bakker. Object Recognition for Video Retrieval. In *Proceedings of the International Conference on Image and Video Retrieval*, pages 262–270, 2002.
- [36] Guy Cote, Berna Erol, Michael Gallant, and Faouzi Kossentini. H.263+: Video Coding at Low Bit Rates. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(7):849–866, November 1998.

- [37] Dublin Core Metadata Initiative. Dublin core metadata element set, version 1.1: Reference description. <http://dublincore.org/documents/1999/07/02/dces/>, 1999.
- [38] Frank Manola and Eric Miller. Document Object Model (DOM). <http://www.w3.org/DOM/>, 2004.
- [39] J. M. Martinez. *MPEG-7 Overview*. ISO/IEC JTC1/SC29/W11 N5525, Pattaya, March 2003.
- [40] ISO/IEC. Text of ISO/IEC Final Draft International Standard 15938-3 Information Technology - Multimedia Content Description Interface - Part 3 Visual, N4358. <http://www.chiariglione.org/mpeg/>, July 2001.
- [41] ISO/IEC. Multimedia content description interface-part 4: Audio. <http://www.chiariglione.org/mpeg/>, (International Standard 15938-4), 2001.
- [42] ISO/IEC. Text of ISO/IEC Final Draft International Standard 15938-5 Information Technology - Multimedia Content Description Interface - Part 5 Multimedia Description Schemes, N4242. <http://www.chiariglione.org/mpeg/>, July 2001.
- [43] ISO/IEC. Text of ISO/IEC Final Draft International Standard 15938-6 Information Technology - Multimedia Content Description Interface - Part 6 Reference Software, N4206. <http://www.chiariglione.org/mpeg/>, July 2001.
- [44] ISO/IEC. FCD 15938-7 Information Technology - Multimedia Content Description Interface - Part 7 Conformance, N4633. <http://www.chiariglione.org/mpeg/>, March 2002.
- [45] ISO/IEC. Information technology - Multimedia content description interface - Part 9: MPEG-7 profiles, CD 15938-9, N6263. <http://www.chiariglione.org/mpeg/>, December 2003.
- [46] ISO/IEC. Information technology - Multimedia content description interface - Part 10: Schema definition, CD 15938-10. <http://www.chiariglione.org/mpeg/>, November 2003.
- [47] Jan Bormans and Keith Hill. Overview of the MPEG-21 standard. *ISO/IEC JTC1/SC29/WG11/N5231*, October 2002.

- [48] Fernando Pereira. The MPEG-21 standard: why an open multimedia framework? In *Proceedings of the 8th International Workshop on Interactive Distributed Multimedia Systems (IDMS 2001)*, pages 219–220, Lancaster, 2001. LNCS 2158, Springer Verlag Heidelberg.
- [49] Amarnath Gupta, Terry Weymouth, and Ramesh Jain. Semantic Queries with Picture: The VIMSYS Model. In *Proceedings of the 17th Conference on Very Large Databases (VLDB)*, pages 69–79, Palo Alto, California, 1991.
- [50] Vincent Oria, M. Tamer Özsu, Paul J. Iglinski, Bing Xu, and L. Irene Cheng. DISIMA: An Object-Oriented Approach to Developing an Image Database System. In *Proceedings of the 16th International Conference on Data Engineering*, pages 672–674, San Diego, California, USA, 2000.
- [51] Solomon Atnafu Besufekad. *Modélisation et traitement de requêtes images complexes*. Dissertation, L’Institut National des Sciences Appliquées de Lyon, 2003.
- [52] H. Kosch, L. Böszörményi, A. Bachlechner, C. Hanin, C. Hofbauer, M. Lang, C. Riedler, and R. Tusch. SMOOTH - A Distributed Multimedia Database System. In *Proceedings of the International VLDB Conference*, pages 713–714, Rome, Italy, 2001.
- [53] Vincent Oria, M. Tamer Özsu, Ling Liu, Xiaobo Li, John Z. Li, Youping Niu, and Paul J. Iglinski. Modeling Images for Content-Based Queries: The DISIMA Approach. In *Proceedings of the 2nd International Conference of Visual Information Systems*, pages 339–346, San Diego, California, 1997.
- [54] Ji-Rong Wen, Qing Li, Wei-Ying Ma, and Hong-Jiang Zhang. A Multi-paradigm Querying Approach for a Generic Multimedia Database Management System. *ACM SIGMOD Record*, 32(1):26–34, 2003.
- [55] V. Oria, M. T. Özsu, and P. J. Iglinski. Foundation for the DISIMA Image Query Languages. *Multimedia Tools and Applications Journal*, 23:185–201, 2004.
- [56] G. Cha, X. Zhu, D. Petkovic, and C. Chung. An Efficient Indexing Method for Nearest Neighbor Searches in High-Dimensional Image Databases. *IEEE Transaction on Multimedia*, 4(1):76–87, March 2002.

- [57] S. Berchtold, D. A. Keim, and H. P. Kriegel. The X-Tree: An Index Structure for High-Dimensional Data. In *Proceedings of the 22nd Int. Conf. on Very Large Data Bases (VLDB)*, pages 28–39, Mumbai (Bombay), India, August 1996. Morgan Kaufmann, ISBN 1-55860-382-4.
- [58] Stefan Berchtold, Christian Böhm, and Hans-Peter Kriegel. The Pyramid-Technique: Towards Breaking the Curse of Dimensionality. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 142–153. ACM Press, 1998.
- [59] Antonin Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *SIGMOD 1984, Proceedings ACM SIGMOD International Conference on Management of Data, June 18-21, 1984*, pages 47–57. ACM Press, 1984.
- [60] Mauricio R. Mediano, Macro A. Casanova, and Marcelo Dreux. V-Trees - A Storage Method for Long Vector Data. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 321–330, 1994.
- [61] Stefan Berchtold, Christian Böhm, H. V. Jagadish, Hans-Peter Kriegel, and Jörg Sander. Independent Quantization: An Index Compression Technique for High-Dimensional Data Spaces. In *Proceedings of the 16th International Conference on Data Engineering*, pages 577–589, San Diego, California, 2000.
- [62] Yasushi Sakurai, Masatoshi Yoshikawa, Shunsuke Uemura, and Haruhiko Kojima. The A-tree: An Index Structure for High-Dimensional Spaces Using Relative Approximation. In *Proceedings of the 26th VLDB Conference*, pages 516–526, Cairo, Egypt, 2000.
- [63] N. Katayama and S. Satoh. The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries. In *ACM SIGMOD Int. Conf. on Management of Data*, pages 369–380, 1997.
- [64] D. A. White and R. Jain. Similarity Indexing with the SS-tree. In *Proceedings of the 12th Int. IEEE Conf. on Data Engineering*, pages 516–523, New Orleans, Louisiana, 1996. IEEE Computer Society 1996.
- [65] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient Access Method for Similarity Search in Metric Spaces. In *Proceedings of the 23rd Int. Conf. on Very Large*

- Data Bases (VLDB)*, pages 426–435, Athens, Greece, 1997. Morgan Kaufmann, ISBN 1-55860-470-7.
- [66] Volker Gaede and Oliver Günther. Multidimensional Access Methods. *ACM Computing Surveys (CSUR)*, 30(2):170–231, 1998.
- [67] Christian Böhm, Stefan Berchtold, and Daniel A. Keim. Searching in High-dimensional Spaces: Index Structures for Improving the Performance of Multimedia Databases. *ACM Computing Surveys (CSUR)*, 33(3):322–373, 2001.
- [68] Hee-Kap Ahn, Nikos Mamoulis, and Ho Min Wong. A Survey on Multidimensional Access Methods. Technical report, UU-CS-2001-14, Utrecht University, Netherlands, 2001.
- [69] Guojun Lu. Techniques and Data Structures for Efficient Multimedia Retrieval Based on Similarity. *IEEE Transactions on Multimedia*, 4(3):372–384, 2002.
- [70] Christian Garcia-Arellano. *Quantization Techniques for Similarity Search in High-Dimensional Data Spaces*. Master thesis, University of Toronto, 2002.
- [71] Stefan Berchtold, Bernhard Ertl, Daniel A. Keim, Hans-Peter Kriegel, and Thomas Seidl. Fast Nearest Neighbor Search in High-dimensional Space. In *Proceedings of the 14th International Conference on Data Engineering (ICDE'98)*, pages 209–218, Orlando, Florida, 1998.
- [72] Thomas Seidl and Hans-Peter Kriegel. Optimal Multi-Step k-Nearest Neighbor Search. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of data*, pages 154–165, Seattle, Washington, United States, 1998.
- [73] Thomas Seidl and Hans-Peter Kriegel. Efficient User-Adaptable Similarity Search in Large Multimedia Databases. In *Proceedings of the 23rd Int. Conf. on Very Large Data Bases (VLDB)*, pages 506–515, Athens, Greece, 1997.
- [74] Roger Weber, Hans J. Schek, and Stephen Blott. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In *Proceedings of the 24th International Conference on Very Large Data Bases*, pages 194–205, 1998.

- [75] Joseph M. Hellerstein, Elias Koutsoupias, and Christos H. Papadimitriou. On the Analysis of Indexing Schemes. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 249–256, Tucson, Arizona, United States, 1997.
- [76] Nathan G. Colossi and Mario A. Nascimento. Benchmarking Access Structures for High-Dimensional Multimedia Data. Technical report, TR/99-05, Department of Computing Science, University of Alberta, Canada, 1999.
- [77] Marcel Kornacker. *Access Methods for Next-Generation Database Systems*. Dissertation, University of California at Berkley, 2000.
- [78] M. Shah, M. Kornacker, and J. M. Hellerstein. Amdb: A Visual Access Method Development Tool. In *User Interfaces To Data Intensive Systems*, pages 130–140, Edinburgh, Scotland, 1999.
- [79] R. Bliujute, C. S. Jensen, S. Saltenis, and G. Slivinskas. R-tree Based Indexing of Now-Relative Bitemporal Data. In *Proceedings of the 24th Int. Conf. on Very Large Data Bases (VLDB)*, pages 345–356, New York, USA, 1998. Morgan Kaufmann.
- [80] M. Kornacker. High-Performance Extensible Indexing. In *Proceedings of the 25th Int. Conf. on Very Large Data Bases (VLDB)*, pages 699–708, Edinburgh, Scotland, 1999. Morgan Kaufmann.
- [81] Mario Döller and Harald Kosch. An MPEG-7 Multimedia Data Cartridge. In *SPIE Conference on Multimedia Computing and Networking 2003 (MMCN03)*, pages 126–137, Santa Clara, CA, January 29-31 2003.
- [82] Mario Döller and Harald Kosch. Demonstration of an MPEG-7 Multimedia Data Cartridge. In *Proceedings of the 10th ACM International Conference on Multimedia*, pages 85–86, Antibes, France, December 2002. ACM Press.
- [83] Carsten Kleiner and Udo W. Lipeck. OraGiST - How to Make User-Defined Indexing Become Usable and Useful. In *BTW 2003 - Datenbanksysteme für Business, Technologie und Web - Tagungsband der 10. BTW-Konferenz*, pages 324–334, Leipzig, Germany, 2003.
- [84] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is Nearest Neighbor Meaningful? In *Proceedings of the 7th International Conference on Database Theory (ICDT'99)*, pages 217–235, 1999.

- 
- [85] Ratko Orlandic, Jack Lukaszuk, and Craig Swietlik. The Design of a Retrieval Technique for High-Dimensional Data on Tertiary Storage. *ACM SIGMOD Record*, 31(2):15–21, 2002.
- [86] P. Wu, B.S. Manjunath, and H.D. Shin. Dimensionality Reduction for Image Retrieval. In *Proceedings of the IEEE International Conference on Image Processing (ICIP 2000)*, pages Vol III: 726–729, 2000.
- [87] P. Husbands, H. Simon, and C. Ding. On the use of singular value decomposition for text retrieval. In *Proceedings of SIAM Comp. Info. Retrieval Workshop*, October 2000.
- [88] Hector Garcia-Molina and H. V. Jagadish. The R\*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 322–331, Atlantic City, NJ, USA, 1990.
- [89] Surajit Chaudhuri and Luis Gravano. Optimizing Queries over Multimedia Repositories. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of data*, pages 91–102, Montreal, Canada, 1996.
- [90] Thomas Brinkhoff, Hans-Peter Kriegel, and Bernhard Seeger. Efficient Processing of Spatial Joins Using R-trees. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of data*, pages 237–246, Washington, D.C., United States, 1993.
- [91] Ju-Hong Lee, Guang-Ho Cha, and Chin-Wan Chung. A Model for k-Nearest Neighbor Query Processing Cost in Multidimensional Data Spaces. *Information Processing Letters*, 69(2):69–76, 1999.
- [92] Stefan Berchtold, Christian Böhm, Danial A. Keim, and Hans-Peter Kriegel. A Cost Model for Nearest Neighbor Search in High-Dimensional Data Space. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 78–86, Tucson, Arizona, United States, 1997.
- [93] Christian Böhm. A Cost Model for Query Processing in High Dimensional Data Spaces. *ACM Transactions on Database Systems (TODS)*, 25(2):129–178, 2000.

- 
- [94] Swarup Acharya, Viswanath Poosala, and Sridhar Ramaswamy. Selectivity Estimation in Spatial Databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, pages 13–24. ACM Press, 1999.
- [95] Alberto Belussi and Christos Faloutsos. Estimating the Selectivity of Spatial Queries Using the Correlation Fractal Dimension. In *Proceedings of the 21th International Conference on Very Large Data Bases*, pages 299–310, 1995.
- [96] Ramez Elmasri and Navathe Shamkant B. *Fundamentals of Database Systems*. The Benjamin/Cummings Publishing Company, 1994. 873 pages, ISBN: 0-8053-1753-8.
- [97] Kyuseok Shim, Ramakrishnan Srikant, and Rakesh Agrawal. High-dimensional Similarity Joins. In *Proceedings of the Thirteenth International Conference on Data Engineering*, pages 301–311. IEEE Computer Society, Washington, DC, USA, 1997.
- [98] Dimitris Papadias, Nikos Mamoulis, and Yannis Theodoridis. Processing and Optimization of Multiway Spatial Joins Using R-trees. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 44–55. ACM Press, New York, NY, USA, 1999.
- [99] Harald Kosch and Solomon Atnafu. A Multimedia Join by the Method of Nearest Neighbor Search. *Information Processing Letters (IPL)*, Elsevier Press, 82(5):269–276, June 2002.
- [100] William I. Grosky. Managing Multimedia Information in Database Systems. *Communications of the ACM*, 40(12):73–80, December 1997.
- [101] Jim Melton and Andrew Eisenberg. SQL Multimedia Application packages (SQL/MM). *ACM SIGMOD Record*, 30(4):97–102, December 2001.
- [102] Final Report NSERC Strategic Grant STR181014. DISIMA: A Distributed Image Database Management System. Technical report, University of Alberta, Canada, 2000.
- [103] John Z. Li, M. Tamer Özsu, Duane Szafron, and Vincent Oria. MOQL: A Multimedia Object Query Language. In *Proceedings of the third International Workshop on Multimedia Information Systems*, pages 19–28, Como Italy, 1997.



- [104] Vincent Oria, M. Tamer Ozsu, Bing Xu, L. Irene Cheng, and Paul Iglinski. Visualmoql: The DISIMA visual query language. In *International Conference on Multimedia Computing and Systems (ICMCS), Vol. 1*, pages 536–542, 1999.
- [105] Vincent Oria, Bing Xu, and M. Tamer Ozsu. Visualmoql: A visual query language for image databases. In *VDB*, pages 186–191, 1998.
- [106] David Jordan. *C++ Object Databases, Programming with the ODMG Standard*. Addison-Wesley, 1998. 456 pages, ISBN: 0-201-63488-0.
- [107] Oracle. Data Cartridge Developer's Guide. [http://otn.oracle.com/docs/products/oracle9i/doc\\_library/release2/appdev.920/a96595.pdf](http://otn.oracle.com/docs/products/oracle9i/doc_library/release2/appdev.920/a96595.pdf), March 2002.
- [108] Oracle. Oracle Spatial User's Guide and Reference, Release 9.2. [http://download-east.oracle.com/docs/html/A96630\\_01/toc.htm](http://download-east.oracle.com/docs/html/A96630_01/toc.htm), 2002.
- [109] Oracle. Oracle interMedia User's Guide and Reference, Release 9.0.1. <http://download-east.oracle.com/docs/>, June 2001.
- [110] Rajiv Chopra Meliyal Annamalai and Samuel DeFazio. Indexing Images in Oracle8i. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of data*, pages 539–547, Dallas, United States, 2000. ACM Press.
- [111] David W. Adler. IBM DB2 Spatial Extender - Spatial data within the RDBMS. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01)*, pages 687–692, Rom, Italy, Sep 2001. Morgan Kaufmann.
- [112] IBM. IBM DB2 Spatial Extender - User's Guide and Reference, Version 8. <http://www-306.ibm.com/software/data/db2/>, December 2002.
- [113] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *Computer*, 28(9):23–32, Sep 1995.
- [114] IBM. DataBlade Module Development Overview, Version 4.0. <http://www-306.ibm.com/software/data/informix/blades/>, August 2001.
- [115] IBM. Informix Spatial DataBlade Module - User's Guide, Version 8.11. <http://www-306.ibm.com/software/data/informix/blades/>, August 2001.

- 
- [116] IBM. Excalibur Image DataBlade Module - User's Guide, Version 1.2. <http://www-306.ibm.com/software/data/informix/blades/>, March 1999.
- [117] J. M. Martinez. Overview of the MPEG-7 standard, v5.0. ISO/MPEG N4509, MPEG Requirements Group, December 2001.
- [118] W3C. Extensible Markup Language (XML) 1.1, W3C Recommendation. <http://www.w3.org/XML/>, February 2004.
- [119] W3C. XML Schema 1.1, W3C Recommendation. <http://www.w3.org/TR/xmlschema-0/>, May 2001.
- [120] K. Staken. Xindice Developers Guide 0.7. *The Apache Foundation*, <http://www.apache.org>, December 2002.
- [121] H. Jagadish, S. Al-Khalifa, and et Al. A. Chapman. TIMBER: A Native XML database. *The VLDB Journal*, 11(4):274–291, 2002.
- [122] Carl-Christian Kanne and Guido Moerkotte. Efficient Storage of XML Data. In *Proceedings of the IEEE Conference on Data Engineering (ICDE)*, page 198, 2000.
- [123] Takeyuki Shimura, Masatoshi Yoshikawa, and Shunsuke Uemura. Storage and retrieval of XML documents using object-relational databases. In *Database and Expert Systems Applications*, pages 206–217, 1999.
- [124] Kanda Runapongsa and jignesh M. Patel. Storing and Querying XML Data in Object-Relational DBMSs. In *EDBT 2002 Workshop on XML-Based Data Management (XMLDM'02)*, pages 266–285, 2002.
- [125] Ravi Murthy and Sandeepan Banerjee. XML Schemas in Oracle XML DB. In *Proceedings of the 29th VLDB Conference*, pages 1009–1018, Berlin, Germany, 2003. Morgan Kaufmann.
- [126] Dan Suci. On Database Theory and XML. *ACM SIGMOD Record, Special section on advanced XML data processing*, 30(3):39–45, September 2001.
- [127] Dongwon Lee and Wesley W. Chu. Constraints-preserving Transformation from XML Document Type Definition to Relational Schema. *Data and Knowledge Engineering*, 39(1):3 – 25, October 2001.

- [128] Jayavel Shanmugasundaram, Rajasekar Krishnamurthy, Igor Tatarinov, Eugene Shekita, Efstratios Viglas, Jerry Kierman, and Jeffrey Naughton. A General Technique for Querying XML Documents using a Relational Database System. *ACM SIGMOD Record*, 30(3):20–26, September 2001.
- [129] Sandeepan Banerjee. Implementing XML Schema inside a 'Relational' Database. In *Proceedings of ACM SIGMOD Intl. Conf. on Management of Data*, pages 313–324, Mineapolis, USA, May 1994.
- [130] Daniela Florescu and Donald Kossmann. Storing and Querying XML Data using RDBMS. *IEEE Data Engineering Bulletin*, 22(3):27–34, September 1999.
- [131] Jayavel Shanmugasundaram, Eugene J. Shekita, Jerry Kiernan, Rajasekar Krishnamurthy, Stratis Viglas, Jeffrey F. Naughton, and Igor Tatarinov. A general techniques for querying XML documents using a relational database system. *SIGMOD Record*, 30(3):20–26, 2001.
- [132] Albrecht Schmidt, Martin Kersten, Menzo Windhouwer, and Florian Waas. Efficient relational storage and retrieval of XML documents. *Lecture Notes in Computer Science*, 1997:137+, 2001.
- [133] Dongwon Lee, Murali Mani, and Wesley W. Chu. Effective Schema Conversions between XML and Relational Models. In *European Conf. on Artificial Intelligence (ECAI), Knowledge Transformation Workshop (ECAI-OT)*, pages 3–11, Lyon, France, July 2002.
- [134] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl. From Structured Documents to Novel Query Facilities. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, pages 313–324, Minneapolis, Minnesota, May 1994.
- [135] Albrecht Schmidt, Florian Waas, Martin Kersten, Daniela Florescu, Michael J. Carey, Ioana Manolescu, and Ralph Busse. Why and How to Benchmark XML Databases. *ACM SIGMOD Record*, 3(30):27–32, September 2001.
- [136] Benjamin Bin Yao, M. Tamer Özsu, and Nitin Khandelwal. XBench Benchmark and Performance Testing of XML DBMSs. In *Proceedings of 20th International Conference on Data Engineering*, pages 621–632, Boston, USA, March 2004.

- [137] Volker Turau. Making legacy data accessible for XML applications. *Internet Document*, <http://www.ti5.tu-harburg.de/Staff/Turau/pubs/legacy.pdf>, October 1999.
- [138] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. XML-QL: A Query Language for XML. *W3C*, <http://www.w3.org/TR/1998/NOTE-xml-ql-19980819/>, August 1998.
- [139] W3C. XML Query (XQuery). *W3C*, <http://www.w3.org/XML/Query>, 2003.
- [140] James Clark and Steve DeRose. XML Path Language (XPath). *W3C Recommendation*, <http://www.w3.org/TR/xpath>, 1999.
- [141] J.E. Funderburk, S. Malaika, and B. Reinwald. XML programming with SQL/XML and XQuery. *IBM Systems Journal*, 41(4):642–665, 2002.
- [142] Jonathan Robie. XQL (XML Query Language). <http://www.ibiblio.org/xql/xql-proposal.html>, 1999.
- [143] David McGoveran. The Age of XML Database. *EAI Journal*, pages 18–21, October 2001.
- [144] Bhavani Thuraisingham. *XML Databases and the Semantic Web*. CRC Press, 2002. 306 pages, ISBN: 0-8493-1031-8.
- [145] Utz Westermann and Wolfgang Klas. An Analysis of XML Database Solutions for the Management of MPEG-7 Media Descriptions. *ACM Computing Surveys*, 35(4):331–373, December 2003.
- [146] Stelios Paparizos, Shurug Al-Khalifa, Adriane Chapman, H.V. Jagadish, Laks V.S. Lakshmanan, Andrew Nierman, Jignesh M. Patel, Divesh Srivastava, Nuwee Wiwatwattana, Yuqing Wu, and Cong Yu. TIMBER: A native system for querying XML. In *Proceedings of the SIGMOD Conference*, San Diego, USA, June 2003.
- [147] Cong Yu, H. V. Jagadish, and Dragomir Radev. Querying XML Using Structures and Keywords in Timber. In *Proceedings of the SIGIR Conference*, page 463, Toronto, Canada, July 2003.
- [148] H. V. Jagadish, Laks V.S.Lakshmanan, Divesh Srivastava, and Keith Thompson. TAX: A Tree Algebra for XML. In *Proceedings of the DBPL Conference*, pages 149–164, Rome, Italy, September 2001.

- 
- [149] Jayavel Shanmugasundaram, Kristin Tufte, Gang He, Chun Zhang, David De Witt, and Jeffrey Naughton. Relational Databases for Querying XML Documents: Limitations and Opportunities. In *Proceedings of the 25th VLDB Conference*, pages 302–314, Edinburgh, Scotland, September 1999.
- [150] Alin Deutsch, Mary Fernandez, and Dan Suciu. Storing Semistructured Data with STORED. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 431–442, Philadelphia, USA, June 1999.
- [151] Sihem Amer-Yahia and Mary Fernandez. Overview of Existing XML Storage Techniques. *AT&T Labs Research*, 2001.
- [152] Irena Mlynkova and Jaroslav Pokorny. XML in the World of (Object-) Relational Database Systems. *Technical Report 2003/8, Charles University, Prag*, December 2003.
- [153] Dongwon Lee, Murali Mani, and Wesley W. Chu. Schema Conversion Methods between XML and Relational Models. In *Knowledge Transformation for the Semantic Web*, pages 1–17, Amsterdam, Netherlands, 2003. IOS Press.
- [154] James McGovern, Per Bothner, Kurt Cagle, James Linn, and Vaidyanathan Nagarajan. *XQuery*. Sams Publishing, 2003. 355 pages, ISBN: 0-672-32479-2.
- [155] Don Chamberlin, Denise Draper, Mary Fernandez, Michael Kay, Jonathan Robie, Michael Rys, Jerome Simeon, Jim Tivy, and Philip Wadler. *XQuery from the Experts, A Guide to the W3C XML Query Language*. Addison-Wesley, 2003. 484 pages, ISBN: 0-321-18060-7.
- [156] Andrew Eisenberg and Jim Melton. SQL/XML and the SQLX Informal Group of Companies. *ACM SIGMOD Record, Web Edition*, 30(3):105–108, September 2001.
- [157] Andrew Eisenberg and Jim Melton. SQL/XML is Making Good Progress. *ACM SIGMOD Record*, 31(2):101–108, June 2002.
- [158] Atsuo Yoshitaka and Tadao Ichikawa. A Survey on Content-Based Retrieval for Multimedia Databases. *IEEE Transaction on Knowledge and Data Engineering*, 11(1):81–93, January 1999.

- [159] Remco C. Veltkamp and Mirela Tanase. Content-Based Image Retrieval Systems: A Survey. Technical report, Department of Computing Science, Utrecht University, The Netherlands, 2000.
- [160] Steffen Pauws. CubyHum: A Fully Operational Query by Humming System. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR)*, pages 187–196, Paris, France, October 2002.
- [161] Werner Bailer, Harald Mayer, Helmut Neuschmied, Werner Haas, Mathias Lux, and Werner Klieber. Content-based Video Retrieval and Summarization using MPEG-7. In *Proceeding Internet Imaging V*, pages 1–12, San Jose, CA, USA, January 2004.
- [162] Alexander G. Hauptmann, Michael G. Christel, and Norman D. Papernick. Video Retrieval with Multiple Image Search Strategies. In *Proceeding of the second ACM/IEEE-CS joint conference on Digital libraries (JCDL)*, page 376, Portland, Oregon, July 14-19 2002.
- [163] Gary Geisler, Gary Marchionini, Barbara M. Wildemuth, Anthony Hughes, Meng Yang, Todd Wilkens, and Richard Spinks. Video Browsing Interfaces for the Open Video Project. In *Proceeding of Conference on Human Factors in Computing Systems*, pages 514–515, Minneapolis, Minnesota, USA, 2002.
- [164] Alexander G. Hauptmann, Rong Jin, and Tobun D. Ng. Video Retrieval using Speech and Image Information. In *Proceedings of SPIE Volume: 5021 Storage and Retrieval for Media Databases 2003*, pages 148–159, Santa Clara, CA, January 29-31 2003.
- [165] Alexander G. Hauptmann and Michael J. Witbrock. Informedia: News-on-Demand Multimedia Information Acquisition and Retrieval. In *Intelligent Multimedia Information Retrieval*, pages 213–239. AAAI Press, 1997.
- [166] Shih-Fu Chang, William Chen, Horace J. Meng, Hari Sundaram, and Di Zhong. VideoQ: An Automated Content Based Video Search System Using Visual Cues. In *Proceedings of the ACM Multimedia Conference*, pages 313–324, Seattle, WA, USA, 1997.
- [167] Bill Adams, Arnon Amir, Chitra Dorai, Sugata Chosal, Giridharan Iyengar, Alejandro Jaimes, Christian Lang, Ching yung Lin, Apostol Natsev, Milind Naphade, Chalapathy Neti, Harriet J. Noch, Haim H. Permuter, Rahavendra Singh, John R. Smith, Savitha Srinivasan, Belle L. Tseng, Ashwin T. V., and Dongqing Zhang. IBM Research

- TREC-2002 Video Retrieval System. In *Proceedings of the Text Retrieval Conference TREC 2002 Video Track*, pages 182–198, Gaithersburg, MD, 2002.
- [168] Xian-Sheng Hua, Pei Yin, Huaqian Wang, Junfen Chen, Lie Lu, Mingjing Li, and Hong-Jiang Zhang. MSR-Asia at TREC-11 Video Track. In *Proceedings of the Text Retrieval Conference TREC 2002 Video Track*, Gaithersburg, MD, 2002.
- [169] Oracle. Database Support for XML. *Oracle9i Application Developer's Guide - XML*, <http://otn.oracle.com/tech/xml/index.html>, 2001.
- [170] Oracle. Oracle Text. [http://otn.oracle.com/products/text/pdf/10gR1text\\_twp\\_f.pdf](http://otn.oracle.com/products/text/pdf/10gR1text_twp_f.pdf), March 2002.
- [171] Bernd-Uwe Pagel, Hans-Werner Six, Heinrich Toben, and Peter Widmayer. Towards an Analysis of Range Query Performance in Spatial Data Structures. In *Proceedings of the Twelfth ACM SIGACT-SIGMODSIGART Symposium on Principles of Database Systems*, pages 214–221, Washington, DC, USA, 1993.
- [172] G. Graefe. Query Evaluation Techniques for Large Databases. *ACM Computing Surveys (CSUR)*, 25(2):73–170, 1993.
- [173] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: An Efficient Clustering Algorithm for Large Databases. In *Proceedings of the ACM SIGMOD Conference*, pages 73–84, 1998.
- [174] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, OR, USA, 1996.
- [175] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of data*, pages 94–105, Seattle, Washington, United States, 1998.
- [176] Levent Ertöz, Michael Steinbach, and Vipin Kumar. Finding Clusters of Different Sizes, Shapes and Densities in Noisy, High Dimensional Data. In *Proceedings of the Third SIAM International Conference on Data Mining*, San Francisco, CA, USA, 2003.

- [177] A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A Review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999.
- [178] S. B. Kotsiantis and P. E. Pintelas. Recent Advances in Clustering: A Brief Survey. *WSEAS Transactions on Information Science and Applications*, 1(1):73–81, 2004.
- [179] Erica Kolatch. Clustering Algorithms for Spatial Databases: A Survey. Technical report, Department of Computing Science, University of Maryland, College Park, 2001.
- [180] Agustino Kurniawan, Nicolas Benech, Tao Yufei, Tian Feng, Wang Jiying, and Theocharis Malamatos. *Towards High-Dimensional Clustering, COMP 530: Database Architecture and Implementation*. Hong Kong University of Science and Technology, November 1999.
- [181] Pavel Berkhin. Survey Of Clustering Data Mining Techniques. Technical report, Accrue Software, San Jose, CA, 2002. [http://www.accrue.com/products/rp\\_cluster\\_review.pdf](http://www.accrue.com/products/rp_cluster_review.pdf).
- [182] W3C. RDF Primer. <http://www.w3.org/TR/REC-rdf-syntax/>, 1999.
- [183] Megan Thomas, Chad Carson, and Joseph M. Hellerstein. Creating a Customized Access Method for Blobworld. In *Proceedings of the IEEE Conference on Data Engineering (ICDE)*, page 82, 2000.
- [184] D. Martin, C. Fowlkes, and J. Malik. Learning to Detect Natural Image Boundaries Using Brightness and Texture. *Neural Information Processing Systems*, December 2002.
- [185] Serge Belongie, Chad Carson, Hayit Greenspan, and Jitendra Malik. Region-Based Image Querying. In *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 42–49, San Juan, Puerto Rico, 1997.
- [186] Oliver Hellmuth, Eric Allamanche, Jürgen Herre, Thorsten Kastner, Markus Cremera, and Wolfgang Hirsch. Advanced Audio Identification using MPEG-7 Content Description. In *Proceedings of the 111th AES Convention, New York*. Audio Engineering Society, 2001.



- 
- [187] Thorsten Kastner, Eric Allamanche, Jürgen Herre, Oliver Hellmuth, Markus Cremer, and Holger Grossmann. MPEG-7 Scalable Robust Audio Fingerprinting. In *Proceedings of the 112th AES Convention, Munich, Germany*. Audio Engineering Society, May 2002.
- [188] Holger Crysandt. Music identification with MPEG-7 audio. In *Proceedings of the 115th AES Convention, New York*. Audio Engineering Society, October 2003.