Mapping, indexing and querying of MPEG-7 descriptors and description schemes

Chu, Yang

2006

Chu, Y. (2006). Mapping, indexing and querying of MPEG-7 descriptors and description schemes. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/41736>

<https://doi.org/10.32657/10356/41736>

NANYANG TECHNOLOGICAL UNIVERSITY

Mapping, Indexing & Querying of MPEG-7 Descriptors & Description Schemes

A Thesis Submitted to the
School of Computer Engineering
of the Nanyang Technological University

by

Chu Yang

in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy (Ph.D)

2006

Abstract

MPEG-7 is a promising standard for the description of multimedia content. Today, more and more multimedia applications based on MPEG-7 descriptions have been set up for research, commercial and industrial applications. Therefore, an efficient storage solution for large amounts of MPEG-7 descriptions is certainly desirable. However, few existing works focus on a comprehensive MPEG-7 descriptions management system. Although the MPEG-7 descriptions are also the XML documents, the current database solutions for XML documents do not fulfill all the requirements for the management of MPEG-7 descriptions. The goal of this thesis is to investigate the MPEG-7 descriptions storage solution and present the implementation of an MPEG-7 descriptions management system: IXMMS, which is based on RDBMS and ORDBMS. IXMMS is a novel XML storage solution to overcome the problems in the current XML storage solutions for the management of MPEG-7 descriptions. The design of IXMMS pays attention to both multimedia information exchange and multimedia data manipulation. IXMMS introduces two separate databases, one to store MPEG-7 descriptions from the XML perspective, and the other for the low-level multimedia content within the MPEG-7 descriptors. Thus, IXMMS supports the management of MPEG-7 descriptions and MPEG-7 information exchange not only from the XML perspective, but also from the multimedia perspective, and its features can reach the most critical requirements for the MPEG-7 documents storage and management.

In IXMMS, we propose a novel XML storage approach, called SM3, for the MPEG-7 documents storage solution. As a kind of data-centric XML documents, MPEG-7 descriptions can be stored in the relational DBMS for efficient and effective management. The approaches of storing XML data in relational DBMS can be classified into two classes of storage model: schema-conscious and schema-oblivious. The schema-conscious

model, however, cannot support complex XPath-based queries efficiently and the schema-oblivious approach lacks the flexibility in typed representation and access. SM3 integrates the advantages of both the schema-conscious method and the schema-oblivious method, and avoids the main drawbacks from each method. A translation mechanism for converting XQuery to SQL is also provided with SM3. Furthermore, the impact of SM3 is not just limited to the domain of MPEG-7. It can be used as the storage solution of arbitrary XML documents.

In order to retrieve MPEG-7 descriptions, an adequate query language for XML data is required. Several XML query languages have been proposed for querying XML data. Among these languages, XQuery is a forthcoming standard for XML document retrievals. However, XQuery cannot adequately support MPEG-7 document queries due to the particularities of MPEG-7 descriptions, such as datatype extensions and queries from multimedia perspective. To provide an adequate MPEG-7 query language in IXMMS, an extension of XQuery, called MXQuery, is proposed. In addition to supporting the general XML queries, MXQuery focuses on the queries of complex datatypes within the MPEG-7 descriptions, queries on spatial-temporal relationships and query-by-example.

To evaluate the efficiency and effectivity of IXMMS, performance studies are conducted by performing a set of queries from the XML perspective and from the multimedia perspective. The experimental results are presented in this thesis and initial results are encouraging.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Chia Liang Tien, Clement, for all the support and directions he gave me during my research efforts towards this dissertation. His encouragement, scientific competence and constructive remarks are inspiring and helpful for my current and future work. I also thank him for his precious help in writing my PhD dissertation.

Secondly, I would like to express my appreciation to Dr. Sourav S. Bhowmick, from whom I learned a lot and I received valuable help all during this research work. I thank him for all his exceptional guidance, constructive remarks and scientific skills.

Furthermore, I am thankful to all the members of our research group with whom I worked closely. I thank them for their patience for listening and the enlightening discussions I had with them. They are Mrs. Wang Surong, Mr. Liu Song, Mr. Zhou Chen, Mrs. Chen Ling, Mr. Yi Haoran, Mr. Koh Su Ming, as well as others.

Finally, my deepest appreciations go to my friends and my kind family. I would not have been able to get this far without their constant support and encouragement.

Contents

Abstract	i
Acknowledgments	iii
List of Figures	vii
List of Tables	x
1 Introduction	1
1.1 An Overview of MPEG-7	2
1.2 Research Issues and Challenges	5
1.2.1 Research Issues in MPEG-7 Descriptions Management	5
1.2.2 XML Storage Approaches	7
1.2.3 Challenges in MPEG-7 Descriptions Storage Solutions	8
1.3 Contribution	11
1.4 Dissertation Organization	13
2 Backgrounds	14
2.1 Existing XML Storage Solutions	14
2.1.1 Native XML Database	14
2.1.2 XML-Enabled Database	16
2.1.3 RDBMS-based XML storage solutions	18
2.2 MPEG-7-related MMDBMS Products	29
2.3 MPEG-7 descriptions storage solutions	33
2.4 Summary	36

3	Design of IXMMS	38
3.1	Motivation	38
3.2	System structure	39
3.3	Back-end data repository	41
4	MPEG-7 Description Database (MDDB)	44
4.1	Overview of SM3	44
4.2	Relation Schema of SM3	46
4.2.1	Internal node storage	46
4.2.2	MPEG-7-DDL-Compliant Mapping Schema	49
4.2.3	Leaf node storage	50
4.2.4	Complex datatype representation	53
4.2.5	Summary	62
4.3	Insertion and Marshalling	62
4.3.1	Insertion Algorithm	62
4.3.2	Marshalling Algorithm	64
4.4	XPath-based Query Support	66
4.4.1	Query Translation Algorithm	67
4.4.2	Query Rewriting	74
5	Multimedia Content Database (MCDB)	78
5.1	Overview of MCDB	78
5.2	Synchronization mechanism	80
5.3	Extensible high-dimensional index mechanism	82
5.3.1	Overview of GiST framework	83
5.3.2	High-dimensional index system in MCDB	84
6	MXQuery	85
6.1	Introduction	85
6.2	Special Query Issues for MPEG-7 Descriptions	87
6.3	MXQuery Specification	88
6.3.1	Specification for complex datatypes	88

6.3.2	Specification for spatial-temporal information	89
6.3.3	Specification for Query-by-example	93
7	Performance Study	95
7.1	Introduction	95
7.1.1	Dataset	96
7.1.2	Overview of performance evaluation	97
7.2	Insertion Performance and Storage Size	98
7.3	Marshalling Performance	101
7.4	Deletion and Update Performance	104
7.5	Query Performance	106
7.5.1	Queries from the XML perspective	106
7.5.2	Performance on array and matrix data	113
7.5.3	Queries from the multimedia perspective	116
8	Application	119
8.1	Introduction to PET-DEVICE++	119
8.2	A Content Based Image Retrieval Tool	121
9	Conclusions and Future Work	127
9.1	Conclusions	127
9.2	Future Work	131
A	Test Operations	133
B	Test Operation Timings	137
C	SQL Queries in IXMMS	141
	References	147
	List of Publications	156

List of Figures

1.1	MPEG-7 main elements [1]	3
1.2	Organization of MPEG-7 Root and Top-level elements	4
1.3	An example of MPEG-7 description	6
1.4	Relational schemas of schema-conscious and schema-oblivious approaches	10
2.1	Tree representation of the example in Figure 1.3	18
2.2	XParent: database for the MPEG-7 document shown in Figure 1.3	26
2.3	SUCXENT++ database schema for the MPEG-7 document shown in Figure 1.3	27
2.4	Set of tools proposed in MARS [2]	30
2.5	The general system architecture of SMOOTH [2]	31
2.6	The MPEG-7 standard support in the MARVEL system [3]	32
2.7	The architecture of the MPEG-7 MDC [4]	34
2.8	PTDOM architecture [5]	35
3.1	The Architecture of IXMMS	40
4.1	DTD for MPD file	51
4.2	MPD for the DominantColor descriptor	52
4.3	SM3 database schema for the example of MPEG-7 document in Figure 1.3	54
4.4	Three relational schemas for storing a matrix	56
4.5	Performance of three matrix storage schemas	56
4.6	Storage scheme for basicTimePoint and basicDuration	61
4.7	Insertion Algorithm	63
4.8	Marshalling algorithm	65

4.9 An example of XQuery and corresponding SQL translated by SM3 67

4.10 Parse tree for the XQuery in Figure 4.9 68

4.11 PathExprs and ComparisonExprs of the XQuery in Figure 4.9 69

4.12 Translation algorithm 69

4.13 Procedure *processPathExpr* 71

4.14 Procedure *processComparisonExpr* 72

4.15 Access plan for SQL statement in Figure 4.9 75

5.1 Examples of Object Type Definitions 79

5.2 Database Model Diagram for DominantColor Descriptor 80

6.1 Function definitions of *arrayItem()* and *matrixItem()* 89

6.2 An MXQuery example for *arrayItem()* function call 90

6.3 An MXQuery example for *matrixItem()* function call 90

6.4 An MXQuery example of spatial relationship 91

6.5 An MXQuery example of temporal relationship 92

6.6 Definition of QBE function 93

6.7 Definition of *featureExtract* function 94

7.1 Experimental Result: Insertion Performance 98

7.2 Experimental Result: Marshalling Performance 103

7.3 Deletion and Update Performance 105

7.4 Experimental Result: Query Performance from the XML perspective . . 107

7.5 Translated SQL with SUCXENT++ 109

7.6 Access plan for MQ3 110

7.7 Access plan for Q15 112

7.8 Experimental Result: Query Performance on Array/Matrix 115

7.9 Experimental Result: Query Performance from the multimedia perspective 116

7.10 Performance on different high dimensional index structures 118

8.1 PET-DEVICE++ System Architecture 120

8.2 UML sequence diagram of Content Based Image Retrieval Tool 122

8.3 A snapshot of Content Based Image Retrieval Tool 123

8.4 Retrieval result when using ColorLayout and M-Tree 124

8.5 Retrieval result when using EdgeHistogram and M-Tree 125

8.6 Upload a query image 125

8.7 Retrieval result for the uploaded image 126

List of Tables

7.1	Characteristics of the experimental data sets	97
7.2	Performance summary and statistics	99
7.3	Storage Size	102
7.4	Running time (s) in terms of two steps of marshalling process	103
7.5	Features of array/matrix storage	114
B.1	Insertion Timings (s)	137
B.2	Marshalling Timings (s)	137
B.3	Deletion and Update Timings (ms)	137
B.4	Query Timings – on BENCH001 (ms)	138
B.5	Query Timings – on BENCH01 (ms)	138
B.6	Query Timings – on BENCH (ms)	139
B.7	Query Timings – on MPEG-7 (ms)	139
B.8	Query Timings – on 1M array/matrix dataset (ms)	139
B.9	Query Timings – on 10M array/matrix dataset (ms)	140
B.10	Query Timings – on 100M array/matrix dataset (ms)	140
B.11	Similarity Search Timings (s)	140

Chapter 1

Introduction

Nowadays, because of the rapid increase of multimedia data in various areas, the demand for multimedia database management system (MMDBMS) and multimedia information systems presents numerous opportunities and challenges. As opposed to earlier development in MMDBMS products, the current MMDBMS projects primarily address the needs of applications for richer semantic content. Most of these applications rely on the new MPEG standard: MPEG-7.

MPEG-7[6] is a standard for describing the content of different types of the multimedia data. As the first standard of the Moving Picture Experts Group to focus not on compression, but rather on metadata or descriptions for the multimedia content, it offers richer semantics as compared with other existing audiovisual metadata like Dublin Core[7] and TV-Anytime[8]. With MPEG-7, multimedia content can be exchanged between heterogeneous systems; plain text files can be used to store and share multimedia information; and multimedia data will be readily available to most users. Thanks to these advantages, more and more applications are based on MPEG-7 descriptions, and the number of MPEG-7 descriptions is inevitably increasing dramatically. Therefore, a critical requirement has arisen: how to develop an adequate database solution for the management of larger numbers of MPEG-7 descriptions.

The topic of this Ph.D dissertation is the management of the MPEG-7 descriptions.

CHAPTER 1. INTRODUCTION

After the literature review, we present our MPEG-7 Description Management System, which is designed for mapping, indexing and querying MPEG-7 Descriptors and Description Schemes. In this system, a novel XML storage schema is proposed for the MPEG-7 description storage.

This chapter provides a general introduction to this Ph.D dissertation. In Section 1.1, MPEG-7 standard is briefly reviewed. Section 1.2 explains the motivation of our research work, and the contributions of this research are presented in Section 1.3. Finally, Section 1.4 describes the organization of this dissertation.

1.1 An Overview of MPEG-7

MPEG-7[6] is an ISO/IEC standard developed by Moving Picture Experts Group, formally known as “Multimedia Content Description Interface”. MPEG-7 specifies the description of features related to the multimedia content as well as information related its management. The MPEG-7 standard defines a normative indexing of multimedia content at many levels ranging from low-level audiovisual (AV) features (such as color, texture, shapes, timbre, and tempo), mid-level AV features (such as spatio-temporal segmentation), to high-level descriptions (structural and semantic content of multimedia data) [6].

MPEG-7 standard has three main elements:

- **Description Tool:** It includes Descriptors (D) and Description Schemes (DS). Descriptors define the syntax and the semantics of each feature, and Description Schemes specify the structure and semantics of the relationships between their components, that may be both Descriptors and Description Schemes;
- **Description Definition Language (DDL):** It defines the syntax of the MPEG-7 Description Tools. It allows the creation of new Description Schemes and Descriptors and the extension and modification of existing Description Schemes. It is based

Figure 1.1: MPEG-7 main elements [1]

on the XML Schema extended by new data types, such as array and matrix data types, basicTimePoint and basicDuration data types;

- **System tool:** It supports binary coded representation for efficient storage and transmission. It is related to the binarization, synchronization, transport and storage of descriptions, as well as to the management and protection of intellectual property.

Figure 1.1 shows the relationship among the different MPEG-7 elements introduced above.

In order to allow the various clusters of technology to be used as stand-alone and allow the editing of the standard to be manageable, the MPEG-7 standard is organized in ten parts along with the different major functionalities. Among these parts, the parts of MPEG-7 Visual, MPEG-7 Audio and MPEG-7 Multimedia Description Schemes (MDS) standardize all the description tools, which comprise all of MPEG-7 predefined descriptors and description schemes. MPEG-7 Visual and MPEG-7 Audio specify the description

CHAPTER 1. INTRODUCTION

Figure 1.2: Organization of MPEG-7 Root and Top-level elements

tools dealing with visual-only and audio-only information respectively. MDS specifies the Descriptors and the Description Schemes dealing with generic and multimedia features, not specific to the audio or the video information.

The MPEG-7 descriptions strongly comply with the XML standard as XML Schema has been the base for the DDL that is used for the syntactic definition of MPEG-7 Description Tools. Each MPEG-7 description should start with the MPEG-7 root element (`<Mpeg7>`) including the description metadata header (`<DescriptionMetadata>`), which provides metadata about the description, and either a description unit (`<DescriptionUnit>`) or a complete description (`<Description>`). The description unit lets us create valid MPEG-7 descriptions containing any MPEG-7 element: Descriptor or Description Scheme. The complete description tag implies that the enclosed description's structure follows one of the MPEG-7 top-level elements, which are organized in three groups: `ContentEntity`, `ContentAbstraction` and `ContentManagement`. The organization of the MPEG-7 root ele-

CHAPTER 1. INTRODUCTION

ments and the top-level elements is presented in Figure 1.2. This figure is updated from a figure in [6].

Figure 1.3 illustrates an MPEG-7 media description complying to the MPEG-7 DDL. It shows an example of **DominantColor** descriptor, which provides a compact description of the dominant colours in an image. The brief description of each element and attribute is given as follows [9]:

- Size: specifies the number of dominant colors in the region;
- ColorSpace: specifies the color space in which the dominant color descriptor is expressed. This descriptor uses ‘HMMD’ color space, which is defined by a nonlinear, reversible transformation from the RGB color space;
- ColorQuantization: defines the uniform quantization of a color space;
- SpatialCoherency: specifies the spatial coherency of the dominant colors;
- Value: specifies an array of elements that hold percentages and values of colors in a visual item;
- Percentage: specifies the percentage of pixels that have the associated color value;
- Index: specifies the index of the dominant color in the selected color space as defined in ColorQuantization;
- ColorVariance: specifies an integer array containing the value of the variance of color values of pixels corresponding to the dominant color in the selected color space.

1.2 Research Issues and Challenges

1.2.1 Research Issues in MPEG-7 Descriptions Management

Since MPEG-7 descriptions can be used as the base of all application domains making use of multimedia, such as classic multimedia archives, broadcast media selection, digital libraries, home entertainment, e-commerce, AI, etc., the number of MPEG-7 descriptions

CHAPTER 1. INTRODUCTION

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<Mpeg7 xmlns="http://www.mpeg7.org/2001/MPEG-7_Schema"
  xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance">
  <DescriptionUnit xsi:type="DescriptorCollectionType">
    <Descriptor size="7" xsi:type="DominantColorType">
      <ColorSpace type="HMMD" colorReferenceFlag="false"/>
      <ColorQuantization>
        <Component>H</Component>
        <NumOfBins>360</NumOfBins>
        <Component>Sum</Component>
        <NumOfBins>100</NumOfBins>
        <Component>Diff</Component>
        <NumOfBins>100</NumOfBins>
      </ColorQuantization>
      <SpatialCoherency>21</SpatialCoherency>
      <Values>
        <Percentage>4</Percentage>
        <ColorValueIndex>216 23 43 </ColorValueIndex>
        <ColorVariance>0 0 0 </ColorVariance>
      </Values>
      ... ..
      <Values>
        <Percentage>0</Percentage>
        <ColorValueIndex>55 35 9 </ColorValueIndex>
        <ColorVariance>1 0 1 </ColorVariance>
      </Values>
    </Descriptor>
  </DescriptionUnit>
</Mpeg7>

```

Figure 1.3: An example of MPEG-7 description

has increased considerably. Thus, there is a demand for adequate database support for the management of these larger numbers of MPEG-7 descriptions.

Before managing MPEG-7 documents, it is important to highlight several main features of the MPEG-7 documents. First, since an MPEG-7 description is also an XML document, managing it means managing an XML document. Second, the MPEG-7 standard allows applications to create new or recombine existing description schemes with MPEG-7 DDL. Therefore, the MPEG-7 description schemes are not fixed and the MPEG-7 management system should be suitable for an arbitrary MPEG-7 description. Finally, to represent the rich datatypes contained in the MPEG-7 descriptions and facilitate manipulating them, much of the data within the MPEG-7 descriptions cannot be simply treated as the textual type.

As discussed in [10], a suitable MPEG-7 storage solution should satisfy several critical requirements: fine-grained storage, representation and access, typed representation and access, providing both classic one-dimensional index structures and multidimensional in-

CHAPTER 1. INTRODUCTION

index structures for efficient access and query, and providing path indexing to navigate through the hierarchical structure of the MPEG-7 documents and efficiently extract the desired information. In addition to these requirements, an appropriate MPEG-7 management solution should emphasize other special issues. As pointed out in [10], one challenge of the management of the MPEG-7 descriptions is how to make use of the MPEG-7 schemas and fulfill the requirement of accessing and processing the arrays and the matrices within the MPEG-7 documents which make up the low-level multimedia content. Another challenge is to provide an extensible high-dimension index structure to support efficient multimedia retrieval applications based on the MPEG-7 media descriptions.

MPEG-7 descriptions have two main functions: to facilitate the exchange of multimedia information between multiple systems and users, and to provide multimedia data for all kinds of multimedia applications to operate on. Therefore, a suitable MPEG-7 storage solution should pay attention to the following two aspects: multimedia information exchange and multimedia data manipulation.

1.2.2 XML Storage Approaches

Since MPEG-7 descriptions are also XML documents, the first consideration of the management of MPEG-7 descriptions is how to employ an XML document storage schema to fulfill the MPEG-7 descriptions storage requirements. There exist many XML storage solutions: Native XML database solutions [11, 12, 13], XML extensions of leading DBMS[14, 15, 16], and research works on the RDBMS-based XML management [17, 18, 19, 20, 21, 22, 23, 24, 25, 26].

To store XML documents efficiently and effectively in a relational database, there is a need to map the XML DTD/Schema to the database schema. The RDBMS-based XML storage solutions can be classified into two major categories according to their

mapping schemas: the **schema-conscious** approach and the **schema-oblivious** approach. In the **schema-conscious** approach, the design of the database schema is based on the understanding of DTD or XML Schema. It defines a relation for each DTD subgraph and uses a primary-key and a foreign-key to describe the parent-child relationship between two elements. While in the **schema-oblivious** approach, a fixed database schema is used to store the structure and the data of any XML document without the assistance of document schema. The **schema-conscious** approach supports typed representation and access for the XML data. It has better query performance than the **schema-oblivious** approach since it partitions XML data based on DTD/XML Schema. While the **schema-oblivious** approach keeps the whole hierarchical structure information of an XML document. It will thus perform complex XPath-based query more efficiently and make it easier to re-construct the data back into the XML format than the **schema-conscious** approach.

1.2.3 Challenges in MPEG-7 Descriptions Storage Solutions

Since there exist various XML storage solutions with different efficiency and functions, the most puzzling problem is which one is the most suitable choice for the MPEG-7 documents storage.

Native XML database is designed especially for XML documents storage. Its fundamental logical storage unit is XML document, which is represented as text format. Traditional Native XML databases cannot support typed representations of the data within MPEG-7 documents since they represent the contents of an XML document as text. Recent research works on the native XML database have proposed the powerful XML storage schemas to support appropriate access to the non-textual data, e.g., Berkeley DB XML[27] and TDOM[28]. However, Berkeley DB XML is not extensible with index structures to support the index on individual items in the array/matrix datatype and the multidimensional index on the multimedia data of which the MPEG-7 media

CHAPTER 1. INTRODUCTION

description tools make heavy use. TDOM is designed to represent the basic contents of an XML document in a typed fashion and constitute a solid foundation for an XML database solution enabling the adequate management of MPEG-7 media descriptions [28]. However, as the other DOM-based native XML databases, it is costly to build in-memory trees of very large documents and then query those trees.

Furthermore, it is difficult for Native XML database systems to create a flexible and extensible index structure on the data with various datatypes and to query multimedia information across multiple MPEG-7 documents efficiently.

To a certain extent, the XML storage and management technology in the leading database systems, such as IBM DB2, Microsoft SQL Server and Oracle, can be viewed as a technology integrating native XML database and relational database. These XML-Enabled database systems introduce a special datatype for the XML storage and provide a set of functions on this datatype to support the XML management. The intact XML content can be stored in a column with VARCHAR or CLOB datatype. Although these XML-Enabled database systems provide powerful functions to satisfy most XML applications, they cannot reach fine-grained and typed representation requirements for the MPEG-7 description storage. Furthermore, with this XML storage technology, XPath operations are evaluated by constructing DOM from CLOB and using functional evaluations. This can be very expensive when performing the operations on large collections of documents. IBM DB2 and Oracle provide an alternative option for the XML storage, which is called XML collection in IBM DB2 and structured storage in Oracle. Such an alternative can support fine-grained and typed representation of XML documents, more powerful index structures and SQL constraints. However, they raise some other critical problems. They have limited flexibility. Only the documents that conform to the XML Schema can be stored with this technology. XPath operations are evaluated with the aid of the XML Schema. However, it is impossible to find a fixed schema for an arbitrary MPEG-7 document.

Figure 1.4: Relational schemas of schema-conscious and schema-oblivious approaches

While for RDBMS-based XML storage solutions, no matter what approach we use, the **schema-conscious** approach or the **schema-oblivious** approach, we cannot overcome their intrinsic drawbacks. Figure 1.4 shows the relational schema for storing an MPEG-7 example shown in Figure 1.3 with the basic idea of the **schema-conscious** approach and the **schema-oblivious** approach respectively. For the **schema-conscious** approach, it provides weak support for the hierarchical structure of the original XML documents. As shown in Figure 1.4, only parent-child relationship between two elements can be reserved by creating the primary-foreign keys between corresponding tables, whereas the path expression from root element to an arbitrary element and the whole hierarchical structure information, including the ancestor-descendant relationships, will be lost. This drawback makes it difficult to efficiently perform complex XPath-based queries. Furthermore, the process of re-constructing the data from the RDBMS into the XML format may be expensive because of the access to multiple tables and the inflexible representations of the structure information.

As shown in Figure 1.4, because of using a fixed table to store each element or attribute

and the mapping process without the assistant of the DTD or the XML schema, the schema-oblivious approach has to establish only a single value column for storing the value of each element and attribute within the XML documents as strings, the most generic type. Such a storage scheme would make it difficult to reflect all kinds of datatypes and then create an efficient index mechanism to speed up the queries with the conditions based on the datatypes other than the string type.

In addition to the drawbacks mentioned above, few existing XML storage solutions address the problem of storage of special datatypes introduced in the MPEG-7 DDL, such as `array`, `matrix`, `basicTimePoint` and `basicDuration`. Furthermore, one of the critical challenges for the MPEG-7 descriptions management solution is to provide an extensible multidimensional index mechanism to support the multimedia content retrieval. Unfortunately, the multidimensional access methods are rarely available in the most of current RDBMS-based XML storage solutions.

1.3 Contribution

In order to provide an adequate MPEG-7 descriptions management system, this dissertation introduces a research prototype known as IXMMS, abbreviated from “*Integrated XML-Enabled MPEG-7 Management System*”. In summary, the dissertation makes several substantial contributions:

- The dissertation introduces the state-of-the-art XML storage solutions, and analyzes the drawbacks of them for the management of MPEG-7 descriptions;
- To solve the problems of existing XML database solutions, the dissertation proposes the IXMMS, a research prototype for the management of MPEG-7 descriptions. IXMMS can support fine-grained, typed representation and access of MPEG-7 descriptions. The value index structures supported by IXMMS include not only classic

CHAPTER 1. INTRODUCTION

one dimensional value index structures provided with RDBMS, but also multidimensional index structures to allow adequate indexing of multimedia data. The design structure of IXMMS facilitates the multimedia information exchange and multimedia data manipulation. The objective of IXMMS is not only to create an integrated and substantive MMDBMS, but also to provide an MPEG-7 description repository engine with a set of management functions for other MPEG-7-based multimedia applications and MMDBMSs;

- In IXMMS, a novel XML storage solution, known as SM3, is proposed. The motivation of SM3 is to integrate the advantages of two main RDBMS-based XML storage approaches: the **schema-conscious** approach and the **schema-oblivious** approach. It offers adequate means to fulfill fine-grained and typed representation and access requirements for the MPEG-7 description storage. In addition to benefiting from the sophisticated index structures provided by RDBMS, this technique provides the path index structure for the MPEG-7 documents navigation. The flexible storage schema defined by this technique makes it efficient to store and manipulate the special datatypes within the MPEG-7 descriptions, such as **array**, **matrix**, **basicTimePoint** and **basicDuration**. Since SM3 combines the advantages of the **schema-conscious** approach and the **schema-oblivious** approach, it achieves a balanced performance on the operations of insertion and marshalling, and it is significantly more efficient than most existing RDBMS-based XML storage solutions in terms of querying;
- In IXMMS, an extension of XQuery, called MXQuery, is proposed to provide an adequate MPEG-7 query language. In addition to supporting the general XML queries, MXQuery focuses on the queries on complex datatypes within the MPEG-7 descriptions, queries on spatial-temporal relationships and query-by-example.

1.4 Dissertation Organization

The rest of the dissertation is organized as follows. The representative XML database solutions, native XML database solutions, XML-Enabled database and RDBMS-based XML storage schemas, are introduced in Chapter 2.

Given the overview of the existing XML storage solutions and the analysis of the deficiencies of these solutions for the management of MPEG-7 descriptions, the thesis continues with proposing a new MPEG-7 descriptions management system: IXMMS. Chapter 3 describes the system structure of IXMMS in summary, and introduces the design of back-end data repository that is composed of two separate databases: MDDB and MCDB. The detailed design of MDDB is presented in Chapter 4. To provide an adequate MPEG-7 storage solution, a novel RDBMS-based XML storage schema, SM3, is proposed. In Chapter 4, the relation schema of SM3 is introduced. This storage schema integrates the basic idea of **schema-conscious** approach and **schema-oblivious** approach, and pays attention to the storage of the complex datatypes defined in the MPEG-7 DDL. Following the introduction to the storage schema, the insertion, marshalling and query translation algorithms are represented. Chapter 5 presents the design of MCDB, and highlights the synchronization mechanism and extensible high-dimensional index mechanism in MCDB. The query language of MPEG-7 descriptions is an important issue of MPEG-7 descriptions management system. Chapter 6 identifies the special issues of querying MPEG-7 descriptions, and depicts our proposed MPEG-7 query language, MXQuery, with several examples of MPEG-7 descriptions query.

Chapter 7 talks about the implementation and the performance study. The experimental results show that IXMMS has encouraging performance compared with existing XML storage solutions. Chapter 8 describes an application that is based on IXMMS. Finally, we conclude in Chapter 9 by summarizing our research works and the contributions of this dissertation, and give an outlook to some future directions of our research.

Chapter 2

Backgrounds

This chapter gives an introduction to the prior works in topics related to this dissertation. It begins with an introduction to existing works on XML storage solution (Section 2.1). This is followed by an overview of the MPEG-7-related MMDBMS products (Section 2.2). Finally, existing MPEG-7 description storage solutions are described (Section 2.3).

2.1 Existing XML Storage Solutions

The research field of XML storage solutions is still very active. There exist a lot of XML database solutions with different levels of maturity and capabilities, including native XML database, XML-Enabled database and research works on RDBMS-based XML storage schema.

2.1.1 Native XML Database

Native XML database is database designed especially to store XML documents. It defines a logical model for an XML document and treats an XML document as its fundamental logical storage unit, just as a relational database views a row in a table as its fundamental unit of logical storage. Any particular underlying physical storage model is not required in the native XML database. Examples include Lore[11], NATIX[12], TIMBER[13] and Berkeley DB XML[27].

CHAPTER 2. BACKGROUNDS

Lore[11] The **Lore** (Lightweight Object Repository) project was launched at Stanford University in 1995. It is a Database Management System designed specifically for XML data. **Lore**'s data model is OEM (for Object Exchange Model)[29], which is a flexible and self-describing data model for encoding complex information. **Lore** has a fully-functional prototype DBMS, as well as several novel technologies, such as the external data manager and DataGuides.

NATIX[12] **NATIX** is a native repository for supporting tree-structured objects like XML documents at low architecture levels. The core module of **NATIX** is the tree storage manager that maps XML labelled tree model into records. The additional modules include the index management, the query engine, the schema manager and the document manager. **NATIX** takes advantage of the semantic structure of large objects to provide an advanced splitting algorithm, which can speed up updates and queries on XML data.

TIMBER[13] The **TIMBER** is designed to natively store XML, and is based upon a bulk algebra for manipulating trees. The architecture of **TIMBER** XML database is constructed as close as possible to that of a relational database. In order to support to manipulate collections of trees of XML data, the core ideas of database technology, such as declarative querying, a bulk algebra, and cost-based query optimization, are introduced into **TIMBER**.

Berkeley DB XML[27] **Berkeley DB XML** is an open source, embedded XML database built on top of Berkeley DB. **Berkeley DB XML** is a library, not a server, and is designed to store arbitrary trees of XML data. **Berkeley DB XML** enables the efficient native storage and retrieval of XML documents by offering flexible typed indexing of XML nodes, elements, attributes and meta-data, and rapid access using XQuery or XPath. The other specific features of **Berkeley DB XML** include flexible storage control, logical document grouping, XML validation, XML namespace support, white space preservation, and so on.

2.1.2 XML-Enabled Database

XML-Enabled database is the database with the extensions for transferring the data between XML documents and themselves, and it has an added XML mapping layer provided by the database vendor. Database vendors such as IBM, Microsoft, Oracle, and Sybase have developed tools to map XML documents into relational tables. Their database products are also the XML-Enabled databases.

IBM DB2 XML Extender[14] IBM DB2 XML Extender not only serves as a repository for XML documents as well as their DTDs, but also supports XML data management. It provides two options, `XML column` and `XML collection`, for storing XML documents. With these two options, a user has the option to store an entire XML document as a user-defined column or to decompose the document into multiple tables and columns. The `XML column` option allows users to store and retrieve entire XML documents as XML user-defined type column, which includes three types: `XMLCLOB`, `XMLVARCHAR` and `XMLFile`. With the `XML collection` option, users can decompose XML documents into a collection of relational tables, or compose XML documents from a collection of relational tables. DTDs can be stored in the DTD repository. The mapping between the database tables and the structure of the XML document is defined by means of a Data Access Definition (DAD) file, which is an XML document. DAD provides a bridge between an XML document, its DTD, and mapping rules onto database tables.

XML Support in Microsoft SQL Server 2005[15] Microsoft SQL Server 2005 provides a native data type called XML. A table can be created with one or more columns of type XML besides relational columns. XML values are stored in the XML column as large binary objects (BLOB). This preserves XML model characteristics more faithfully such as document order and recursive structures. Five built-in methods on the XML data

CHAPTER 2. BACKGROUNDS

type, including `query()`, `value()`, `exist()`, `nodes()` and `modify()`, are provided to query and modify XML instances and it also accepts XQuery. Users can retrieve XML instances by using a `SELECT` statement. A mechanism for indexing XML columns is provided to speed up queries. A B+ tree index on all tags, values, and paths of the XML instances in the column, which is referred to as the primary XML index, is created on an XML column. A novel technique for indexing XML data called ORDPATH[30] has been implemented in SQL Server 2005 to capture XML document order and document hierarchy.

Oracle XML DB[16] Oracle XML DB is a feature of the Oracle Database. It provides a storage-independent, content-independent and programming language-independent infrastructure to store and manage XML data with high-performance. It fully absorbs the W3C XML data model into the Oracle Database and adds all of the functions associated with native XML databases to extend the Oracle relational database. It introduces a native XML data-type, `XMLType`, to store and manage XML documents. An `XMLType` table or column can contain any well-formed XML document. XML can be stored one of two ways: an `XMLType` column in a relational table or an XML object in an `XMLType` table. Non-Schema based XML is always stored as CLOB (unstructured storage). Schema based XML can be stored as a CLOB or as a set of objects (structured storage). Oracle XML DB provides a number of XML specific methods to operate on the `XMLType` objects. These methods support for common operations like extracting a subset of the nodes contained in the `XMLType`, checking whether or not a particular node exists in the `XMLType`, validating the contents of the `XMLType` against an XML Schema, and performing XSL transformation. Oracle XML DB supports XML/SQL duality, which allows XML operations on SQL data and SQL operations on XML content.

Figure 2.1: Tree representation of the example in Figure 1.3

2.1.3 RDBMS-based XML storage solutions

As introduced in the previous section, the existing RDBMS-based XML storage approaches can be classified into two major categories: the schema-conscious approach and the schema-oblivious approach. Some examples of the schema-conscious approach are found in Basic, Shared and Hybrid Inlining Technique[17], STORED[18], X-Ray[19] and LegoDB[20]. Examples of the schema-oblivious approach include The Edge Approach[22], Monet[23], XRel[24], XParent[25] and SUCXENT++[26]. These methods will be described in greater detail. The MPEG-7 document shown in Figure 1.3 will be used as the running example in the following discussions, and its tree representation is shown in Figure 2.1.

CHAPTER 2. BACKGROUNDS

2.1.3.1 Schema-conscious approach

The schema-conscious approach is also called the **structure-mapping** approach [24]. The schema-conscious approach uses the relational schema to express the logical structure of XML documents. It maps XML Schemas or DTDs into a set of database relations, generally one relation for each element type in the XML documents. The parent-child relationship between two elements of the XML document is represented by a pair of primary-foreign keys in the database schema. Examples of this kind of approach can be found in [31] and some examples are discussed in detail as follows.

Basic, Shared and Hybrid Inlining Technique[17] In this approach, the authors first simplified the details of a DTD by proposing a set of transformations that include flattening transformations, simplification transformations and grouping transformations. Flattening transformations are used to convert a nested definition into a flat representation. Simplification transformations are used to reduce many unary operators to a single operator. Grouping transformations are used to group sub-elements that having the same name. Then, a transformed DTD graph was obtained and three techniques were proposed for converting the simplified DTD to a relational schema:

- (a) *Basic Inlining* — this method constructs relations for every element as an XML document can be rooted at any element in a DTD. It inlines all the element's descendants into the relation generated for this element with two exceptions: (1) children directly below a '*' node, which represents the set with one or more elements, are made into separate relations – this is for set-valued child; and (2) each node having a backpointer edge pointing to it is made into a separate relation – this is for handling recursion;
- (b) *Shared Inlining* — this method attempts to avoid the drawbacks of *Basic Inlining* by ensuring that an element node is represented in exactly one relation. It identifies

CHAPTER 2. BACKGROUNDS

commonly used element nodes that are represented in multiple relations in Basic Inlining and share them by creating separate relations for them. In *Shared Inlining*, relations are created according to the DTD graph. The elements with in-degree greater than one or with in-degree equal to 0 have relations made and the elements with in-degree equal to 1 are inlined. Finally, one of the mutually recursive elements all having in-degree one is also made a separate relation;

- (c) *Hybrid Inlining* — this method is the same as *Shared Inlining* except that it inlines some elements that are not inlined in *Shared Inlining* method. It inlines elements that have in-degree greater than one and are not recursive and are not reached through a repeatable node.

With the *Shared Inlining* method, following relations would be created for storing the MPEG-7 example showed in Figure 1.3 into the relational database.

```
mpeg7 (ID, parentID, parentCODE, mpeg7.xmlns, mpeg7.xmlnsxsi,
      mpeg7.descriptionunit.isroot, mpeg7.descriptionunit.xsitype)
dominantcolor (ID, parentID, parentCODE, dominantcolor.size, dominantcolor.
      xsitype, dominantcolor.colorsapce.isroot, dominantcolor.colorsapce.type,
      dominantcolor.colorsapce.colorreferenceflag, dominantcolor.colorquantization.
      isroot, dominantcolor.spatialcoherency)
component (ID, parentID, parentCODE, component)
numofbins (ID, parentID, parentCODE, numofbins)
values (ID, parentID, parentCODE, values.colorvalueindex, values.colorvariance,
      values.percentage)
```

STORED[18] It seems unconvincing to classify STORED as the schema-conscious approach, since the generation of a STORED mapping does not require a DTD or an XML Schema as input. However, such a mapping schema is still based on the structure of the given XML documents. The STORED mapping schema is automatically generated from patterns discovered in the structure of the data instance by using data-mining techniques.

The STORED mapping includes two parts: the relational mapping and the overflow mapping. The relational mapping generates a ‘good’ relational schema and STORED mapping to this schema for the given semi-structured data. A modified apriori association

CHAPTER 2. BACKGROUNDS

rule algorithm is used to identify the most commonly used structures in the data instance to produce such a reasonable relational schema. The overflow mapping is necessary to ensure the storage is lossless for any data instance. With the overflow mapping, the data not conforming to the relational schema are stored in an overflow graph. In **STORED**, a query- and an update-rewriting algorithm is developed to translate the queries and updates over the semi-structured view into SQL.

X-Ray[19] The idea of **X-Ray** is that mappings may be defined between XML schema specifications and relational schemas while preserving their autonomy. This is made possible by introducing a meta schema and meta knowledge for resolving data model heterogeneity and schema heterogeneity. The meta schema provides the basis for **X-Ray** to automatically compose XML documents out of the relational database when requested and decompose them when they have to be stored. Basically the meta schema consists of three components describing the relevant meta knowledge: **DBSchema**, **XML DTD/XMLSchema** and **XMLDBSchemaMapping**. The **DBSchema** component is responsible for storing information about relational schemas, i.e., about relations, database attributes, relationships, and joins. The **XML DTD/XMLSchema** component stores schema information about XML documents as specified by means of DTDs, i.e., element types, XML attributes, and the composition structure. Finally, the **XMLDBSchemaMapping** component stores the knowledge about mappings between DTDs, XML Schemas and relational schemas, whereby a single DTD or XML Schema may be mapped to several relational schemas and vice versa.

LegoDB: a cost-based approach[20] **LegoDB** is a cost-based XML-to-relational mapping engine that provides the best distribution of relations for each XML storage application. A fixed mapping is unlikely to work well as different applications may present different access patterns. It is hard to determine a good mapping schema for a complex

CHAPTER 2. BACKGROUNDS

application. LegoDB is designed to automatically find an efficient relational configuration for a target XML application. It has three important features:

- Cost-based search – LegoDB can takes application characteristics (including XML schema, data statistics and query workload) into account, and derives a mapping with the lowest cost.
- Logical/physical independence – The LegoDB interface is purely XML-based.
- Reuse of existing XML and relational technologies whenever possible.

Since XML Schema has no information about the data to be stored and can not be straightforwardly mapped to relations, LegoDB introduces the notion of **physical XML schema** or **p-schema**, which contains useful statistics about data to be stored, and can be easily mapped into relational tables. LegoDB also defines a fixed mapping from a **p-schema** to a relational schema. In order to obtain an optimal mapping schema, LegoDB provides a set of algebraic transformations applied to a **p-schema** to generate a space of distinct relational configurations. Once these configurations are known, LegoDB uses a relational optimizer as a black box to obtain cost estimates for each configuration. Finally, a simple greedy evaluation strategy is used to search for the optimal solution.

2.1.3.2 Schema-oblivious approach

The **schema-oblivious** approach is also known as the **model-mapping** approach in [24]. The basic idea of this approach is to capture information about the structure of an XML document and use a fixed database schema to store all XML documents. Some **schema-oblivious** approaches are described in [31, 32]. The major ones are discussed as follows.

CHAPTER 2. BACKGROUNDS

Edge Approach[22] The Edge approach is a pioneering schema-oblivious approach. The main idea of this technique is to model an XML document as an ordered and labeled directed graph, and all the edges of an XML document tree are stored as relational tuples. The authors proposed several schemes for mapping XML data into relational tables to solve the following two problems: how to map edges, and how to map values. There are three alternative approaches to store the edges of an XML graph:

- (i) Edge approach – all the edges of an XML document are stored in a single table.

This table can be represented as,

Edge(source, ordinal, name, flag, target).

In this table, the attributes **source** and **target** record the identifications of the source and target nodes of each edge; **ordinal** is used to represent the order of the edge among its siblings; **name** records the label of the edge; and **flag** is used to indicate whether the edge represents an inter-object reference or points to a value.

- (ii) Binary approach – all edges with the same label are grouped into a single table.

This approach horizontally partitions the Edge table by using the element name as the partitioning attribute. The binary table has the following structure:

$B_{name}(\text{source, ordinal, flag, target}).$

- (iii) Universal table – a single table is used to store all edges. This table contains attributes for all elements and attribute names. The universal table can be represented as follows,

Universal(source, ordinal_{*n1*}, flag_{*n1*}, target_{*n1*}, ordinal_{*n2*}, flag_{*n2*}, target_{*n2*}, ... ,
ordinal_{*nk*}, flag_{*nk*}, target_{*nk*}).

Here, *n1*, *n2*, ..., *nk* are the label names.

To map the values, two approaches are proposed:

CHAPTER 2. BACKGROUNDS

- (i) Separate value tables – it stores values by using separate **Value** tables for each conceivable data type.
- (ii) Inlining – it stores all values and attributes in the same table. For example, in the **Edge** approach, this corresponds to an outer join of the **Edge** table and the **Value** tables [22].

The two approaches for mapping values can be combined with the three approaches for mapping edges. The performance analysis carried out by the authors shows that the combination of ‘Binary approach’ and ‘Inlining’ outperforms the others. However, this approach is still extremely costly to reconstruct a very large XML document.

Monet[23] **Monet** is a variation of the **Edge** approach. It partitions the XML data according to all possible paths. For each unique path, **Monet** creates a table named from the path string. For example, the leftmost path in Figure 2.1 can be stored in the following five tables:

```

Mpeg7.DescriptionUnit
Mpeg7.DescriptionUnit.Descriptor
Mpeg7.DescriptionUnit.Descriptor.Values
Mpeg7.DescriptionUnit.Descriptor.Values.ColorVariance
Mpeg7.DescriptionUnit.Descriptor.Values.ColorVariance.CDATA.

```

The first four are for the element nodes and the last one is for the text node. For element nodes, each table has three attributes: **source**, **target** and **ordinal**. The **source** and **target** attributes together specify a unique edge in an XML data graph, and the **ordinal** attribute is used to represent the order of an edge. For text nodes, **Monet** creates a table with two attributes, **id** and **value**. Thus, **Monet** needs to store the XML data in a large number of small tables.

CHAPTER 2. BACKGROUNDS

XRel[24] Unlike **Edge** and **Monet**, which are designed to maintain edges, **XRel** maintains nodes rather than edges. **XRel** records containment relationships, called as **region**, specified by the start and end positions of a node in an XML document. If the **region** of a node n_j covers the **region** of another node n_i , then n_i is reachable from n_j . **XRel** also stores every possible path expressions as strings in the database. The queries containing path expressions can be efficiently processed by using string-matching operators.

XRel schema consists of four tables: **Element**, **Attribute**, **Text** and **Path**.

Element(docID, pathID, start, end, index, reindex)

Attribute(docID, pathID, start, end, value)

Text(docID, pathID, start, end, value)

Path(pathID, pathexp).

All path expressions are stored in the table **Path**. The attributes, **docID**, **pathID**, **start**, **end** and **value**, represent document identifier, path expression identifier, start position of a **region**, end position of a **region**, and string value, respectively. The database attributes **index** and **reindex** record the order information of an element node among its siblings in document order and reverse document order respectively.[24] Because the schema for both **Attribute** and **Text** are the same, **XRel** stores both text nodes and attribute nodes in the **Text** table.

XParent[25] **XParent** maps XML data into the following five relational tables to generate a schema-oblivious XML storage schema.

LabelPath(ID, Len, Path)

DataPath(Pid, Cid)

Element(PathID, Did, Ordinal)

Data(PathID, Did, Ordinal, Value)

Ancestor(Did, Ancestor, Level)

CHAPTER 2. BACKGROUNDS

ID	Len	Path	Did	PathID	Ordinal	Did	Ancestor	Level
1	1	./Mpeg7	1	1	1	2	1	1
2	2	./Mpeg7./@xmlns	2	2	1	3	1	1
3	2	./Mpeg7./@xmlns:xsi	3	3	1	4	1	1
4	2	./Mpeg7./DescriptionUnit	4	4	1	5	4	1
... ..						5	1	2
18	5	./Mpeg7./DescriptionUnit./Descriptor./Values./ColorValueIndex	46	18	1			
19	5	./Mpeg7./DescriptionUnit./Descriptor./Values./ColorVariance	47	19	1	45	44	1
LabelPath			Element			45	6	2
						45	4	3
						45	1	4
Did	PathID	Ordinal	Value	Cid	Pid	46	44	1
2	2	1	http://www.mpeg7.org/2001/MPEG-7_Schema	1	0	46	6	2
3	3	1	http://www.w3.org/2000/10/XMLSchema-instance	2	1	46	4	3
5	5	1	DescriptorCollectionType	3	1	46	1	4
7	7	1	7	4	1	47	44	1
... ..						47	6	2
46	18	1	55 35 9	46	44	47	4	3
47	19	1	1 0 1	47	44	47	1	4
Data			DataPath			Ancestor		

Figure 2.2: XParent: database for the MPEG-7 document shown in Figure 1.3

The **LabelPath** table stores path expressions. The **DataPath** table holds pairs of node identifiers, which specify the parent-child relationship between two nodes of each edge in an XML tree. In **Element** and **Data** tables, **PathID** is a foreign key to the **ID** in the **LabelPath** table. The **Did** is a node identifier. The value of each leaf node is stored in the **Data** table. The **Ancestor** table maintains ancestor-descendant relationships.

XParent explicitly stores label-paths and data-paths. The label-paths give a global view on the XML documents stored in the RDBMS. Two relations are proposed to keep parent-child and ancestor-descendant relationships respectively, and then speed up the process of checking edge connections. The database for the MPEG-7 document (Figure 1.3) is shown in Figure 2.2.

SUCXENT++[26] SUCXENT++ only stores leaf nodes and their associated paths. The relationship between leaf nodes can be captured with two attributes: **BranchOrder** and **BranchOrderSum**. SUCXENT++ is a four table database schema and its relational schema is shown as follows:

CHAPTER 2. BACKGROUNDS

DOCID	PAT HID	LEAFO RDER	BRANCH ORDER	BRANCHO RDSUM	LEAFVALUE
1	1	1	0	0	http://www.mpeg7.org/2001/MPEG-7_Schema
1	2	2	2	1	http://www.w3.org/2000/10/XMLSchema-instance
1	3	3	1	0	DescriptorCollectionType
1	4	4	2	1	7
1	5	5	4	3	DominantColorType
1	6	6	3	14	HMMO
1	7	7	5	15	FALSE
1	8	8	3	26	H
1	9	9	4	28	360
1	8	10	4	30	Sum
1	9	11	4	32	100
1	8	12	4	34	Diff
1	9	13	4	36	100
1	10	14	3	47	21
1	11	15	3	58	4
1	12	16	4	60	216 23 43
1	13	17	4	62	0 0 0
1	11	18	3	73	10
1	12	19	4	75	44 67 30
1	13	20	4	77	0 0 0
1	11	21	3	88	2
1	12	22	4	90	210 33 39
1	13	23	4	92	0 0 0
1	11	24	3	103	6
1	12	25	4	105	150 80 2
1	13	26	4	107	1 0 0
1	11	27	3	118	4
1	12	28	4	120	209 60 15
1	13	29	4	122	0 0 0
1	11	30	3	133	3
1	12	31	4	135	206 42 22
1	13	32	4	137	0 0 0
1	11	33	3	148	0
1	12	34	4	150	55 35 9
1	13	35	4	152	1 0 1

ID	NAME
1	DominantColor.xml

Document

PATHID	PATHEXP	LENGTH
1	Mpeg7.@xmlns	1
2	Mpeg7.@xmlns:xsi	1
3	Mpeg7.DescriptionUnit.@xsi:type	2
4	Mpeg7.DescriptionUnit.Descriptor.@size	3
5	Mpeg7.DescriptionUnit.Descriptor.@xsi:type	3
6	Mpeg7.DescriptionUnit.Descriptor.ColorSpace.@type	4
7	Mpeg7.DescriptionUnit.Descriptor.ColorSpace.@colorReferenceFlag	4
8	Mpeg7.DescriptionUnit.Descriptor.ColorQuantization.Component	4
9	Mpeg7.DescriptionUnit.Descriptor.ColorQuantization.NumOfBins	4
10	Mpeg7.DescriptionUnit.Descriptor.SpatialCoherency	3
11	Mpeg7.DescriptionUnit.Descriptor.Values.Percentage	4
12	Mpeg7.DescriptionUnit.Descriptor.Values.ColorValueIndex	4
13	Mpeg7.DescriptionUnit.Descriptor.Values.ColorVariance	4

Path

DOCID	LEVEL	RVALUE
1	1	2
1	2	1
1	3	11
1	4	2
1	5	1

DocumentRValue

PathValue

Figure 2.3: SUCXENT++ database schema for the MPEG-7 document shown in Figure 1.3

- Document (DocId, Name)
- Path (PathId, PathExp, Length)
- PathValue (DocId, PathId, LeafOrder, BranchOrder, BranchOrderSum, LeafValue)
- DocumentRValue (DocId, Level, RValue)

The table Document is used for storing the names of the documents in the database. The Path table records every unique root-to-leaf path encountered in the XML documents. The PathValue table stores the leaf nodes of the XML documents stored in the database. Each tuple stores one leaf node. The LeafOrder attribute records the order of the leaf node in which leaf nodes are encountered when the XML document is parsed in depth-first traversal. The BranchOrder attribute records the intersection level of the leaf node with the leaf node that immediately precedes it. The attribute LeafValue is used to store the value of each leaf node. The attribute BranchOrderSum and RValue in table DocumentRValue are required for efficient query processing. Given an XML document

CHAPTER 2. BACKGROUNDS

with maximum depth D , the value of $RValue$ and $BranchOrderSum$ is determined as follows[26]:

- (i) $RValue$ is assigned recursively based on the equation: $r_i = r_{i+1} \times c_{i+1} + 1$, where r_k is the $RValue$ with depth k , c_k is the maximum number of consecutive leaf nodes with $BranchOrder \geq k$ and $r_D = 1$.
- (ii) Let the $BranchOrder$ of a node with $LeafOrder$ n be b_n . Then, the $BranchOrderSum$ of this node is $s_n = \sum_{i=1}^{b_n} r_{b_i}$.

The relationship between two leaf nodes can be determined by using the following lemma: if $|s_n - s_m| < r_l$, then nodes with $LeafOrders$ n and m intersect at a level greater than l , where s_n , s_m are the $BranchOrderSum$ of the leaf nodes with $LeafOrders$ n and m respectively, and r_l is the $RValue$ with depth l .

Figure 2.3 shows the SUCXENT++ database for the MPEG-7 example shown in Figure 1.3.

2.1.3.3 Other RDBMS-related XML storage solutions

There exist some research works that focus on the hybrid relational and XML database system. The authors of [21] provide several reasons for why there should be a union of XML and the relational database systems. First, XML and relational data may co-exist and complement each other. Second, a successful XML database system should take advantage of many advanced mechanisms existing in the relational database system. Finally, there exists a conceptual and functional overlap between XML query language and SQL. System RX[21] is the first truly relational-XML hybrid data management system, which is based on DB2 UDB technology. In System RX, XML data are stored natively in a tree data structure. The system supports an XML column type in a relational model. The XML column may contain data that is validated according to many schemas or that

have not been validated at all. The system defines a set of functions to provide both SQL/XML and XQuery to query the XML column. **System RX** can make good use of the already existing relational infrastructure. The traditional rewrite optimizations are extended to accommodate XML query language, and many advanced mechanisms, such as the existing table spaces, buffer pools, lock manager, etc., are used without modification. However, in **System RX**, the XML type has implementation-dependent internal format, which can be inflexible. Furthermore, **System RX** uses a set of special functions. This can cause a loss of some common functions, e.g., usual string operations.

There are some other research works that focus on how to implement XQuery in a relational database. These works can be found in [33, 34].

2.2 MPEG-7-related MMDBMS Products

In [2], the author introduces three waves of the evolution of Multimedia Database Systems. The first wave of the full-fledged MMDBMS emerged in the mid 90s. Some of them were **ORION**[35], **MediaWay**[36] and **JASMINE**[37]. They were all mainly dependent on the operating system for querying, retrieving, inserting, and updating multimedia data. In the second wave, MMDBMSs were extended to the ORDBMS management system by providing complex object types for various kinds of media. The commercial systems could be found in Oracle and IBM DB2, and a representative research project is **MIRROR**[38]. The third wave, which includes the currently running projects, addresses the needs of applications for richer semantic content. Most of them are dependent on the new MPEG standards MPEG-7 and MPEG-21. The representative running projects include **MARS**[39, 40, 41], **SMOOTH**[42] and **MARVEL**[3].

MARS[39, 40, 41] The goal of **MARS**, an acronym for Multimedia Analysis and Retrieval System, is to realize an integrated multimedia information retrieval and database

Figure 2.4: Set of tools proposed in MARS [2]

management infrastructure. It is categorized into four sub-areas: Multimedia Content Representation, Multimedia Information Retrieval, Multimedia Feature Indexing and Multimedia Database Management. The MARS project introduces the concept of the multimedia object model for organizing all the multimedia low-level features in such a way that the appropriate features are invoked at the right place and the right time in answer to the user’s information need, and also for supporting information abstraction at various semantic levels. The proposed multimedia data model influenced the development of the descriptors in the MDS of MPEG-7[43]. MARS proposes a set of mechanisms and tools for an MMDBMS back end, such as relevance feedback approach, query refinement processing, table of contents (ToC) extraction mechanism for videos, hybrid tree data structure, granular locking approach, etc., as shown in Figure 2.4.

SMOOTH[42] The SMOOTH system is a prototype of a distributed multimedia database system. It implements an integrated framework for semantic indexing and retrieval of video objects. The SMOOTH system is dependent on an integrated model for low- and high-level video indexing. This model is similar to the MDS of MPEG-7 standard, concerning content descriptions. The difference is that it focuses on the modeling of video data

CHAPTER 2. BACKGROUNDS

Figure 2.5: The general system architecture of SMOOTH [2]

supported by a meta-database.

Figure 2.5 displays the general system architecture of SMOOTH. The SMOOTH system consists of a video server, a metadatabase (or an index database) and a set of interfaces. The video server, which is constructed on the Oracle Video Server with the supported protocol types UDP and RTP, provides selective access to the physical video streams. The metadatabase contains the base classes: events, objects, persons, and locations. These high-level content classes are the subclasses of a general `ContentObject`, which may refer to low-level motion objects. The interfaces provide the functionalities of annotate, query, and navigate through video material.

MARVEL[3] MARVEL is an MPEG-7 multimedia search engine developed by the IBM research team. It allows users to construct queries on video databases using techniques based on content-based retrieval (CBR), model-based retrieval (MBR), text- and speech-based retrieval and cluster navigation. MARVEL uses a semantics machine learning approach to build statistical models from multi-modal features and applies these models for automatically annotating larger numbers of multimedia objects. The MARVEL system consists of two components: the MARVEL multimedia analysis engine and the MARVEL multimedia search engine. The former component is to model semantic concepts

CHAPTER 2. BACKGROUNDS

Figure 2.6: The MPEG-7 standard support in the MARVEL system [3]

in video and organize these semantic concepts using ontology. The latter one provides a multimedia semantics-based searching engine that combines other search techniques (speech, text, metadata, audio-visual features, etc.).

The MARVEL system was designed to support the full emerging MPEG-7 multimedia content description standard. MPEG-7 standard provides a number of content description tools for effective indexing and retrieval of video content. These tools are used in the MARVEL system to provide rich description of video content in terms of shot boundaries & key-frames, textual annotations and transcriptions, features, semantics and models [3]. The MPEG-7 standard support in the MARVEL system involves two processes: producing or extracting MPEG-7 descriptions from video content and searching for video content based on the MPEG-7 descriptions. Figure 2.6 shows the MPEG-7 support in the MARVEL system.

2.3 MPEG-7 descriptions storage solutions

There are a few research works on the storage schema and the management of MPEG-7 descriptions. The examples of these works are the MPEG-7 Multimedia Data Cartridge (MDC)[4] and PTDOM[5].

MPEG-7 MDC[4] MPEG-7 MDC is a system extension of the Oracle 9i DBMS to provide a new indexing and query framework for various types of retrieval operations and a semantically rich metadata model for multimedia content relying on the MPEG-7 standard. It builds on two main parts: the **Multimedia Data Model** and the **MIF** (Multimedia Indexing Framework). The **Multimedia Data Model** is a database schema that is derived from MPEG-7 descriptions and contains the metadata describing the multimedia content. With this data model, the MPEG-7 descriptors are mapped into object types and tables with the help of the extensible type system of the cartridge environment. The **MIF**, which is based on the Generalized Search Trees (GiST)[44], provides an extensible indexing environment for multimedia retrieval. It is integrated into the **MDC** via the index extension mechanisms of Oracle 9i. Each new index type can be added by defining a new Oracle indextype, but the interface remains unchanged. The **MIF** is divided into three modules: **GistService**, **GistWrapper** and **Multimedia Index Type**. Each module may be used on its own and may be distributed over the network. Figure 2.7 shows the architecture of the MPEG-7 MDC.

Although the MPEG-7 MDC provides a robust storage solution for the MPEG-7 descriptions, it falls short when evaluated in terms of the requirements listed in [10]. **MDC** is a system extension of the Oracle 9i DBMS. It defines a set of object types to map the MPEG-7 standard into a database model. The data within such object types, e.g., **XMLType** object type, may not be represented and accessed in fine-grained and typed

Figure 2.7: The architecture of the MPEG-7 MDC [4]

manner. Furthermore, the predefined object types may not be suitable for the non-fixed MPEG-7 descriptions with volatile structures.

PTDOM[5] PTDOM, the abbreviation for the Persistent Typed Document Object Model, is a schema-aware native XML database system originally developed for the management of the MPEG-7 descriptions. PTDOM makes use of an MPEG-7 DDL-compliant schema catalog that is based on typing automata. This schema catalog can be employed to validate MPEG-7 documents, infer and construct appropriate typed representations of the contents of MPEG-7 documents, and provide a highly effective path index. PTDOM provides the support for datatypes, database internal UDFs, B-tree index for data elements, multi-dimension indexing structures, and profound extensibility.

CHAPTER 2. BACKGROUNDS

Figure 2.8: PTDOM architecture [5]

The architecture of PTDOM is shown in Figure 2.8. The functionalities of each component are shown as follows:

- **Document Manager** – It constitutes a central registry for managing all XML documents stored with PTDOM;
- **Query Evaluator** – It serves as a first step towards a query processor by providing a query algebra for evaluating XPath expressions and an optimized query translator;
- **Schema Catalog** – As the heart of PTDOM, it accommodates the schema definitions. It provides the means of importing and exporting arbitrary schema definitions, ensuring the integrity of schema definitions and representing the data in a fine-grained way;
- **Simple Type Framework** – It provides the data structures for keeping the values of simple elements and attributes within the XML document in a typed manner;

CHAPTER 2. BACKGROUNDS

- **Routine Framework** – It allows PTDOM to seamlessly integrate user-defined routines to extend the system with arbitrary functionality; and
- **Index Framework** – It not only provides the value index structures including hash tables, B-Trees, and R-Trees, but also allows the integration of new value index structures for unordered, ordered, as well as spatial data.

To represent the contents of the MPEG-7 descriptions, the document manager of PTDOM applies the TDOM[28], an object-oriented model based on the traditional DOM[45] approach. TDOM differs from the traditional DOM in that it pays particular attention to the type information contained in the schema definitions to which the XML documents comply, and then the basic contents of an XML document can be represented in a typed fashion. However, similar to the other traditional DOM-based approaches, the fact that it is costly to build in-memory trees of very large documents and then query those trees can degrade the query efficiency with TDOM.

2.4 Summary

The above sections have given an overview of how to use XML with relational databases and native XML databases. RDBMS-based XML storage approaches can benefit from many advantages provided with relational database, such as widespread usage, proven underlying mathematical theory, query optimization techniques, advanced processing mechanism, etc. The **schema-oblivious** approach maintains a fix storage schema and captures the tree structure of an XML document. But it has a limitation of representation of all kinds of datatypes. The **schema-conscious** approach eliminates this limitation and takes full advantage of index mechanism of relational database. However, it is dependant on the DTD or XML schema of the XML document. Furthermore, this technique has a disadvantage of preserving XML document structure and element order. All the leading

CHAPTER 2. BACKGROUNDS

database management systems provide the support of XML document storage. However, these XML-enabled databases only implement limited XML query functionality. Native XML databases are designed especially to store XML documents. Since its internal model is based on XML, native XML database can preserve XML document order. Another advantage of native XML database is their support of XML query languages, which are difficult to express in SQL. Many native XML databases, however, need to improve their abilities of datatype representation and index mechanism.

There exist a few research works on the MPEG-7 storage solution. Each of them has more or less limitations on the datatype representation and performance. For example, the MPEG-7 MDC does not provide fully data representation and access in fine-grained and typed manner. The PTDOM has the performance drawback due to its DOM-based model.

Chapter 3

Design of IXMMS

This chapter describes the design of the IXMMS system. IXMMS, an MPEG-7 management system, is based on RDBMS and ORDBMS. In this system, the information within the MPEG-7 descriptions is mapped into RDBMS for storage and management. The multimedia content information within the low-level MPEG-7 descriptors is reorganized and stored in ORDBMS. An extensible high-dimension index structure is created for content-based retrieval. We also propose an MPEG-7 query language: MXQuery, which is an extension of XQuery, to support special queries on MPEG-7 descriptions, such as queries on the arrays/matrices and queries from the multimedia perspective. To support the communication between our system and other MPEG-7-based multimedia applications in the network environment, the Web services solution has been considered.

3.1 Motivation

XML documents tend to be either data-centric, which are primarily for data transport, or document-centric, which are used for capturing natural (human) languages. Obviously, MPEG-7 documents are data-centric. Among existing XML storage solutions, such as Native XML database, XML-enabled database and RDBMS-based XML storage solution, the relational database can provide robust solutions for the storage of data-centric XML documents, since many advantages can be found in the relational database: widespread

CHAPTER 3. DESIGN OF IXMMS

usage, proven underlying mathematical theory, query optimization techniques, advanced processing mechanism, etc. Unfortunately, based on the observations discussed in the previous chapters, the existing RDBMS-based XML storage solutions do not provide sufficient functions to support the management of MPEG-7 descriptions. There is a need of new solution designed specially for the management of MPEG-7 documents. We proposed a novel storage schema to satisfy this need and the design of IXMMS is based on this innovation.

Another important issue of MPEG-7 management system is to support multimedia data manipulation. There are many MMDBMSs that have implemented this objective. However, in a MPEG-7 management system, this function should be implemented based on the MPEG-7 description storage schema. In IXMMS, since the MPEG-7 descriptions are stored in the leading RDBMS, the reliable solution of multimedia data manipulation should be provided with the same DBMS. We extended the ORDBMS provided by the leading DBMS to support the queries on MPEG-7 descriptions from multimedia perspective.

MPEG-7 query language is a necessary part of MPEG-7 description management system. There is few existing research work addressing this topic. To enable such a query language compatible with the most popular XML query language, we extended the XQuery by introducing many new features to support the queries of MPEG-7 descriptions.

3.2 System structure

We propose a general design of an architecture based on the requirements for the management of MPEG-7 descriptions. From a system architecture viewpoint, an ideal picture is given in Figure 3.1.

CHAPTER 3. DESIGN OF IXMMS

Figure 3.1: The Architecture of IXMMS

To enable the MPEG-7 management system to be as flexible as the highly fluid multimedia applications, our system is built with the Web services architecture [46], which focuses on simple, Internet-based standards to address heterogeneous distributed computing. Web services represent the convergence of the service-oriented architecture (SOA) and the web. The Web services architecture takes all the best features of the service oriented architecture and combines it with the web. It supports universal communication using loosely coupled connections, and the corresponding web protocols are completely vendor-, platform-, and language-independent.

In our system, two main Web services are designed to deliver a fundamental view of MPEG-7 descriptions: they are the **management web service** and the **retrieval web service**. Using these web services, all kinds of multimedia applications from any web or system environment can pull the multimedia information from the MPEG-7 descriptions across our MPEG-7 descriptions repository. The **management web service** serves as the

service for common manipulation of the data between the MPEG-7 descriptions and the RDBMS. It encapsulates a set of activities to allow external applications to map MPEG-7 descriptions to RDBMS, and to delete or update existing descriptions. The retrieval web service provides a fundamental retrieval engine for all kinds of applications. Such a search engine could support heterogeneous queries, including text-based, model-based or content-based queries, with various formats, such as SQL, XQuery and other types of queries. Although XQuery is a forthcoming standard for XML document retrieval, it cannot adequately support MPEG-7 document queries because of the particularities of the MPEG-7 standard. To overcome this, we introduce MXQuery, an extension of XQuery, as an MPEG-7 query language. A translation mechanism has been developed to enable the back-end databases in our system to support MXQuery. For content-based retrieval, an extensible high-dimension index structure should be considered.

The back-end data repository consists of two types of databases, one based on RDBMS and designed for storing all the information within the MPEG-7 descriptions, and the other based on ORDBMS, which is designed for multimedia content data stored in the MPEG-7 low-level descriptors. Between these two databases, a synchronization mechanism is provided to allow the data in the multimedia content database to be synchronized with the corresponding fields in the MPEG-7 descriptions database. The back-end data repository is described in detail in the following subsection.

3.3 Back-end data repository

There are several reasons for applications to access the MPEG-7 descriptions repository, such as extracting special parts of MPEG-7 descriptions, reorganizing MPEG-7 descriptions, content based retrieval, and semantic multimedia information query. To support all kinds of applications, our system uses RDBMS to store MPEG-7 descriptions with fine-grained and typed manner, as well as complete structure information. However, such

CHAPTER 3. DESIGN OF IXMMS

a storage schema makes the data access operations much more complex when performing a content-based search, since the related data is not in the best format for the content-based query requirement. For example, the similarity search function for `DominantColor` descriptor (an example shown in Figure 1.3) [47, 48] would require data from several attributes and elements: `size`, `SpatialCoherency`, `Percentage` and `ColorValueIndex`. After mapping this descriptor into RDBMS, the data from these elements and attributes can be stored in different tables or records. Capturing these data from the database for similarity search is likely to result in access of several tables or records. Such an operation is expensive if the dataset is considerably large. A better way is to collect the necessary data for similarity search into one data sequence. However, if the values within these elements and attributes are combined and stored in only one table, it would lead to the following shortcoming: the queries on the individual elements or attributes and the inverse operation that re-organizes these values into the original MPEG-7 descriptor format would become much more complex. Furthermore, in this case, the original content within some MPEG-7 descriptors cannot be directly used for similarity search and would require a data transformation, e.g., the descriptors `RegionShape` and `DominantColor`.

To overcome these limitations, we designed two separate database structures to support the MPEG-7 descriptions storage. One database is for the MPEG-7 descriptions storage; the other is for the multimedia content data storage. All the data within MPEG-7 descriptions and the corresponding structure information will be extracted and stored in the MPEG-7 Descriptions Database (MDDB) with the aid of a mapping schema compliant with the MPEG-7 DDL schema. For the content-based retrieval, the necessary data within the low-level MPEG-7 descriptors would be extracted from the MPEG-7 description database, re-organized according to the search function, and stored in the Multimedia Content Database (MCDB). In addition, an appropriate high-dimension index structure would be created on these content data for search performance improvement. Since the

CHAPTER 3. DESIGN OF IXMMS

data in the Multimedia Content Database is extracted from the MPEG-7 Description Database, a synchronization mechanism is necessary for data consistency between these two databases. Following chapters will introduce them in detail.

It may be argued that the idea of creating MCDB makes some content stored repeatedly, and increases the complexity of data management. However, such an idea enables the query mechanism in IXMMS to be more flexible, efficient and robust. It is somewhat similar to the data warehousing technique in the leading database systems. In data warehousing, you can extract desired historical data from the operational system (which runs the daily transactions of your business), clean the data, transform the data for decision making, and write the summarized data to a target in a separate database from the operational system. After that, the complex queries and analysis can be performed on the information efficiently.

Chapter 4

MPEG-7 Description Database (MDDDB)

This chapter represents one of the main components of IXMMS: MDDDB. MDDDB is a RDBMS-based XML database in which storage schema design focuses particular attention to the critical requirements of the management of MPEG-7 descriptions. To overcome the limitations of the current XML database solutions for the management of MPEG-7 descriptions, as analyzed in Chapter 1, we proposed an XML storage approach in MDDDB, known as SM3[49], as an adequate solution for MPEG-7 documents storage. The detailed description of SM3 is found in the following sections.

4.1 Overview of SM3

SM3, abbreviated from “*integrated **Structure-mapping** approach and **Model-mapping** approach*”, is a novel XML storage schema. The motivation of SM3 is to integrate the advantages of the schema-conscious (structure-mapping) approach and the schema-oblivious (model-mapping) approach, and avoid the main drawbacks from each of those methods.

An MPEG-7 document can be viewed as an XML tree. In this tree structure, the internal node, the element type with element contents, represents the structure of a document and can be viewed as the node that is only meaningful for document traversal.

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDb)

The leaf node, which is a single-valued attribute or element type with text content, has little usage for XML tree navigation, as it is always a ‘leaf’ in the XML tree. So it can be viewed as the node that is only useful for holding value. SM3 was designed to use the schema-oblivious approach to store all the internal nodes and the schema-conscious method to store all the leaf nodes. Since all the leaf nodes are stored with the schema-conscious approach, the contents of MPEG-7 documents can be mapped into RDBMS with fine-grained manner and appropriate data types. As all the internal nodes are mapped with the schema-oblivious approach, the complete hierarchical structure information of the original MPEG-7 documents can be kept in the database. Based on the fundamental scheme of SM3, we propose a method for storing complex datatypes within MPEG-7 descriptions, such as `basicTimePoint`, `basicDuration`, `array` and `matrix`, with relational tables.

According to our performance study, the amount of storage space SM3 consumed is between that of the existing schema-conscious methods (e.g., `Shared-Inlining`) and schema-oblivious methods (e.g., `SUCXENT++`). The performance of mapping XML documents to RDBMS with SM3 is slightly slower than `Shared-Inlining`, and similar to `SUCXENT++`. SM3 can reconstruct original XML documents from RDBMS up to two times faster than `Shared-Inlining`, and the same as `SUCXENT++`. For the XPath-based queries, SM3 outperforms `SUCXENT++` by up to 12 times and `Shared-Inlining` by up to 20 times for most testing queries, including recursive queries and ordered XPath queries. Since SM3 provides a special storage schema for the complex datatypes defined in MPEG-7 DDL, e.g. `basicTimePoint`, `basicDuration`, `array` and `matrix`, SM3 outperforms all the other approaches for the queries on these datatypes.

4.2 Relation Schema of SM3

An XML document is often represented as an XML tree. In an XML tree, the internal nodes correspond to the element types with element content in the XML document, while the leaf nodes correspond to the single-valued attributes and element types with PCData-only content in the XML document. An example of XML tree structure is shown in Figure 2.1.

The idea of SM3 is to use the **schema-oblivious** approach to map all the internal nodes and use the **schema-conscious** approach to map all the leaf nodes. In an XML tree, the internal nodes depict the structure of the XML document and are only useful for document navigation. Storing them by using **schema-oblivious** method, which can provide complete structure information of an XML document, can support efficient and easy document traversal. The leaf nodes hold all the data of an XML document. They are the ‘leaves’ in the XML tree, so they have little usage for the XML tree navigation. They can be considered as the nodes only for storing the data of an XML document. Using the **schema-conscious** approach to store them can better represent appropriate datatype of each leaf node and provide a flexible storage scheme to satisfy different storage requests.

4.2.1 Internal node storage

To speed up the processing of the XML tree navigation, it is important to adopt an efficient numbering scheme to encode the nodes of a tree and quickly determine the ancestor-descendant relationship between two arbitrary nodes in the XML tree based on such a numbering scheme. Thus, for RDBMS-based XML storage solutions, it is important to capture the encoding information of each node into the relational data model. Note that, unlike most pure **schema-oblivious** methods that label all the nodes in an XML tree, our approach only needs to encode internal nodes.

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDb)

Motivated by searching XML documents efficiently, several research efforts have addressed the problem of numbering scheme specification. In [50], the authors proposed three order encoding methods that can be used to represent XML order in the relational data model. These three methods are Global Order, Local Order and Dewey Order. Among them, as claimed by the authors, Dewey Order performs reasonably well on both queries and updates. With Dewey Order, each node is assigned an *id* value, a sequence of numeric values separated by a dot that represents the path from the document's root to the node. The root node is assigned a single numeric value. Child node *id* starts with the *id* of the parent node appended by a dot and the local order of the node, as illustrated in Figure 2.1.

With Dewey Order, the ancestor-descendant relationship can be determined using only the *id* value. However, the *id* length depends on the tree depth and a string comparison of the *ids* may degrade the query performance and deliver wrong results with respect to the total node order, e.g., comparing 1.9 and 1.10. In [30], the authors provided a solution for avoiding these shortcomings and proposed a novel hierarchical labeling scheme called ORDPATH.

Although ORDPATH labeling schema still depends on the tree depth, it provides a compressed binary representation of Dewey Order. It uses successive variable-length L_i/O_i bitstrings to represent the *id* value of each node. Each L_i bitstring, which are represented using a form of prefix-free encoding, specifies the length in bits of the succeeding O_i bitstring. For example, if the L_i bitstring 01 is assigned length 3, this L_i will indicate a 3-bit O_i bitstring. The bitstrings (000, 001, 010, ..., 111) can represent O_i values of the first eight integers, (0, 1, 2, . . . , 7). Thus '01101' is the bitstring for ordinal '5' [30].

With ORDPATH, the *id* value of each node is constructed as a binary string and document order can be preserved and yielded by simple bitstring comparison. The ancestor-descendent relationships between any two nodes X and Y can be determined equally

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDB)

simply: X as a strict substring of Y or vice versa implies there is an ancestry relationship.

The update performance is also improved with ORDPATH by introducing ‘caretting’ concept. ORDPATH assigns odd integers to the nodes during an initial load, and even numbers are reserved for later insertions. Therefore, new nodes can be inserted at arbitrary positions in the XML tree without relabeling any old nodes.

Whereas ORDPATH provides these advantages, SM3 uses it to encode the position of each internal node and construct the document structure information in the relational database model. The following are the relational schemas to store internal nodes:

```
xpath (xpathid, length, xpathexp)
internalnode (uid, xpathid, nodename, ordpath, parent,
              grdesc, lid, oid, tablename)
```

The semantics of the attributes in the above relations are as follows:

- The `xpath` table records the XPath information of the XML tree. `xpathid` and `xpathexp` represent the XPath identifier and the path expression. The number of edges for an XPath is recorded in the attribute `length`;
- The `internalnode` table represents the information of each internal node. `uid` is used to identify each internal node. It is also referenced by the leaf node tables as a foreign key to create the association between the internal nodes and the leaf nodes. `ordpath` records the ORDPATH of this internal node. `lid` is the internal node local identifier, which depicts the position of a node among sibling nodes. `oid` is an index of the node that occurs more than once in the XML document. It is useful for answering the queries that include index predicates, such as the following query: “What is the 6th book’s title (`//book[6]/title`)”. `tablename` is used to indicate which table stores the value of this internal node’s leaf nodes children; and

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDDB)

- The attributes `parent` and `grdesc` in the `internalnode` table are used for ancestor-descendant relationship determination between two internal nodes. In [30], the authors introduced two functions to determine the parent and an upper bound on all descendants of a given node. One is `PARENT(ORDPATH X)`, which presents the parent of `X`, the other is `GRDESC(ORDPATH X)`, which is the smallest ORDPATH-like value greater than any descendant of a node with ORDPATH `X`. We can use the user-defined functions (UDF) in RDBMS to implement these two functions. However, not all RDBMS support the creation of an index on function, e.g., IBM DB2. In order to benefit from the index mechanism in RDBMS and improve query process, we introduce two columns, `parent` and `grdesc`, in the `internalnode` table to store the parent of the corresponding internal node and the smallest value greater than any descendant of this node, rather than using UDF.

4.2.2 MPEG-7-DDL-Compliant Mapping Schema

All MPEG-7 descriptions should comply with the schema definition language MPEG-7 DDL which defines the structure of multimedia descriptions and the types of their basic content. It is indispensable for our MPEG-7 storage solution to provide a schema, which is fully compliant with MPEG-7 DDL, for mapping MPEG-7 documents to the database, for inference of typed representation, and for access optimization. We define a mapping processing definition (MPD) file, which is also an XML file, to support the above requirements. The storage schema of SM3 views an MPEG-7 document as a tree of objects and then uses the MPD file to map these objects to a relational database. In this view, an internal node corresponds to a class and the leaf node is usually viewed as property. The MPD file indicates which table corresponds to each internal node and how to store its leaf nodes to columns with appropriate datatypes.

The MPD file is a bridge between the MPEG-7 description schema and the database schema. The MPD configuration file provides the relational database schema for MPEG-7

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDB)

descriptions management, and the corresponding relational database can then be created automatically. As plain text files, MPEG-7 documents themselves do not contain any type information about their basic contents for appropriate typed representations. MPD provides type information of basic contents of MPEG-7 descriptions based on the MPEG-7 DDL. When mapping an arbitrary MPEG-7 document, our system can infer the types from the information provided by the MPD and store the content into the relational database with appropriate datatypes. Another important advantage with MPD is access optimization. An application could access MPEG-7 descriptions stored in our system via a high-level declarative query language such as XQuery. In this case, it is important for the query processor to optimize the access automatically. With MPD and XPath information stored in the system, the declarative query language can be rewritten into an equivalent form, which then could be evaluated more efficiently by the relational database system, and thus benefit from the powerful existing index mechanism provided by RDBMS.

A function has been implemented in our system to automatically generate an MPD file and a set of CREATE TABLE statements for creating the database when given a DTD or an XML schema. The end users can modify the MPD files according to their special storage requirements. Figure 4.1 shows the DTD defined for MPD file, and Figure 4.2 gives an example of MPD file for mapping the `DominantColor` descriptor.

4.2.3 Leaf node storage

SM3 views an XML document as a tree of objects and then uses MPD file to map these objects to a relational database. In this view, an internal node is usually viewed as a class and mapped to a table. For example, as shown in Figure 4.2, the following declares the internal node `Descriptor` to be a class and maps it to the `dominantcolor` table:

```
<InternalNodeClass Name="Descriptor" ToTable="dominantcolor">
... ..
</InternalNodeClass>
```

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDb)

Figure 4.1: DTD for MPD file

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDDB)

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MPD SYSTEM "mpd.dtd">
<MPD>
  <InternalNodeClass Name="Mpeg7" ToTable="mpeg7"/>
  <InternalNodeClass Name="DescriptionUnit" ToTable="descriptorcollection">
    <AttributeClass>
      <AttributeType Name="xsi:type"/>
      <ToColumn Name="xsitype" Datatype="varchar"/>
    </AttributeClass>
  </InternalNodeClass>
  <InternalNodeClass Name="Descriptor" ToTable="dominantcolor">
    <AttributeClass>
      <AttributeType Name="xsi:type"/>
      <ToColumn Name="xsitype" Datatype="varchar"/>
    </AttributeClass>
    <AttributeClass>
      <AttributeType Name="size"/>
      <ToColumn Name="size" Datatype="smallint"/>
    </AttributeClass>
    <LeafNodeClass>
      <ElementType Name="SpatialCoherency"/>
      <ToColumn Name="spatialcoherency" Datatype="smallint"/>
    </LeafNodeClass>
  </InternalNodeClass>
  ...
  <InternalNodeClass Name="Values" ToTable="values">
    <LeafNodeClass>
      <ElementType Name="Percentage"/>
      <ToColumn Name="percentage" Datatype="smallint"/>
    </LeafNodeClass>
    <LeafNodeClass>
      <ElementType Name="ColorValueIndex"/>
      <ToColumn Name="colorvalueindex" Datatype="array"/>
      <ToColumn Name="arrayid" Datatype="integer"/>
    </LeafNodeClass>
    <LeafNodeClass>
      <ElementType Name="ColorVariance"/>
      <ToColumn Name="colorvariance" Datatype="char"/>
    </LeafNodeClass>
  </InternalNodeClass>
</MPD>

```

MPD for DominantColor Descriptor

Figure 4.2: MPD for the DominantColor descriptor

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDb)

The leaf nodes are usually viewed as properties and mapped to columns. For example, the following schema, which is nested inside the above mapping schema, declares the `xsi:type` and `size` attributes and the `SpatialCoherency` element to be properties and maps them to the `xsitype`, `size` and `spatialcoherency` columns respectively.

```
<AttributeClass>
  <AttributeType Name="xsi:type"/>
  <ToColumn Name="xsitype" Datatype="varchar"/>
</AttributeClass>
<AttributeClass>
  <AttributeType Name="size"/>
  <ToColumn Name="size" Datatype="smallint"/>
</AttributeClass>
<LeafNodeClass>
  <ElementType Name="SpatialCoherency"/>
  <ToColumn Name="spatialcoherency" Datatype="smallint"/>
</LeafNodeClass>
```

Our mapping schema for the leaf nodes is much simpler than the existing schema-conscious methods. The information needed to map a single internal node class only includes the table to which the internal node is mapped and the information of each property in this internal node class. It is not necessary to list its related internal node classes, such as its parent and child internal nodes. Note that the ancestor-descendant relationships have been mapped to the relational model via the schema-oblivious technique.

The database schema for the MPEG-7 document example shown in Figure 1.3 is shown in Figure 4.3.

4.2.4 Complex datatype representation

As introduced previously, MPEG-7 descriptions not only use the standard datatypes, but also add the extension datatypes, including `array`, `matrix`, `basicTimePoint` and `basicDuration`. The appropriate MPEG-7 storage solution should fit two application requirements: multimedia information exchange, and multimedia data manipulation. There is no problem for the storage of the data with standard datatypes with respect to these

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDb)

For internal nodes:

XPATHID	LENGTH	XPATHEXP
1	1	#/Mpeg7
2	2	#/Mpeg7#/DescriptionUnit
3	3	#/Mpeg7#/DescriptionUnit#/Descriptor
4	4	#/Mpeg7#/DescriptionUnit#/Descriptor#/ColorSpace
5	4	#/Mpeg7#/DescriptionUnit#/Descriptor#/ColorQuantization
6	4	#/Mpeg7#/DescriptionUnit#/Descriptor#/Values

(a) xpath table

UID	XPATHID	NODENAME	ORDPATH	PARENT	GRDESC	LID	OID	TABLENAME
1	1	Mpeg7	x'48		x'49	1	1	
2	2	DescriptionUnit	x'4A40	x'48	x'4A41	1	1	descriptorcollection
3	3	Descriptor	x'4A52	x'4A40	x'4A53	1	1	dominantcolor
4	4	ColorSpace	x'4A5290	x'4A52	x'4A5291	1	1	colorspace
5	5	ColorQuantization	x'4A52B0	x'4A52	x'4A52B1	2	1	colorquantization
6	6	Values	x'4A52D0	x'4A52	x'4A52D1	4	1	values
7	6	Values	x'4A52F0	x'4A52	x'4A52F1	5	2	values
8	6	Values	x'4A5304	x'4A52	x'4A5305	6	3	values
9	6	Values	x'4A530C	x'4A52	x'4A530D	7	4	values
10	6	Values	x'4A5314	x'4A52	x'4A5315	8	5	values
11	6	Values	x'4A531C	x'4A52	x'4A531D	9	6	values
12	6	Values	x'4A5324	x'4A52	x'4A5325	10	7	values

(b) internalnode table

For leaf nodes:

UID	XSITYPE	UID	SIZE	XSITYPE	SPATIALCOHERENCY
2	DescriptorCollectionType	3	5	DominantColorType 0	

(c) descriptorcollection table

(d) dominantcolor table

UID	LID	OID	NODENAME	VALUE
5	1	1	Component	H
5	2	1	NumOfBins	360
5	3	2	Component	Sum
5	4	2	NumOfBins	100
5	5	3	Component	Diff
5	6	3	NumOfBins	100

(e) colorquantization table

UID	TYPE	COLORREFERENCEFLAG
4	HMMD	FALSE

(f) colorspace table

UID	PERCENTAGE	COLORVALUEINDEX	ARRAYID
6	4	216 23 43	1
7	10	44 67 30	2
8	2	210 33 39	3
9	6	150 80 2	4
10	4	209 60 15	5
11	3	206 42 22	6
12	0	55 35 9	7

(g) values table

ARRAYID	COL0	COL1	COL2
1	216	23	43
2	44	67	30
3	210	33	39
4	150	80	2
5	209	60	15
6	206	42	22
7	55	35	9

(h) array table

Figure 4.3: SM3 database schema for the example of MPEG-7 document in Figure 1.3

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDB)

two requirements. However, it raises an efficiency problem that the storage schema for the above extension datatypes has to give attention to both multimedia information exchange and multimedia data manipulation. Undoubtedly, text-based format is the most efficient storage model for data exchange since it is not limited to any computer platforms and languages. Thus, storing the above special datatypes with text format in RDBMS enables the extraction of them from the database, without any additional operations, and exchange of them between normally incompatible systems efficiently. However, such a storage schema makes it inefficient to manipulate these special datatypes due to the expensive process of character string parsing and datatype conversion. In this subsection, we will introduce how to store these complex datatypes in relational database and avoid the above efficiency problem.

Array and Matrix In order to manipulate the array or matrix data efficiently, we can store individual cells in the array or matrix in a pure relational table with appropriate datatype. There are three relational schemas for array and matrix storage (Figure 4.4 illustrates these three schemas):

- Normalized schema - in this schema, each row will identify a cell in the array or matrix. The schema would be (ARRAYID, COLUMNID, VALUE) for array, or (MATRIXID, COLUMNID, ROWID, VALUE) for matrix;
- Semi-normalized schema - in this schema, each row will store a row in the matrix. The corresponding schema would be (MATRIXID, ROWID, COL0, COL1, ..., COLn) for $m \times n$ matrix; and
- Denormalized schema - in this schema, each row will record one array or matrix. It would be (ARRAYID, COL0, COL1, ..., COLn) for array, or (MATRIXID, R0_C0, R0_C1, ..., Rm_Cn) for $m \times n$ matrix.

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDB)

Figure 4.4: Three relational schemas for storing a matrix

These three storage schemas have different flexibility and efficiency for the operations of insertion, marshalling and query. To evaluate the performance of these three array/matrix storage schemas, we performed a set of operations, including insertion, marshalling and query (selecting an individual item), with 100 21×21 matrices extracted from Sound-ClassificationModel DS. The experimental results are shown in Figure 4.5.

Figure 4.5: Performance of three matrix storage schemas

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDDB)

For the semi-normalized and denormalized schemas, the length of the array or matrix must be fixed, while the normalized schema is flexible enough to store the arrays or matrices of arbitrary cardinality. However, compared to the semi-normalized and denormalized schemas, the normalized schema needs to load many more tuples when performing insertion process, and need more joins to locate the desired individual items when performing queries. For array, the semi-normalized and denormalized schemas have the same storage schema; therefore, they have the same performance on insertion, marshalling and query. For matrix, the denormalized schema has better performance on insertion and query than the semi-normalized schema because it inserts fewer tuples and needs fewer joins for the query process. With the semi-normalized schema, each row within a matrix will be still stored as a row in the database. Among these three schemas, it is the easiest for the semi-normalized schema to compose the data in the database to original matrix format. Thus, for marshalling operation, the semi-normalized schema has the best performance, while the normalized schema performs the worst as it stores the array or matrix data fragmentally.

As a performance trade-off on the operations of insertion, marshalling and query, the semi-normalized schema can be used to store the arrays or matrices with fixed cardinality. However, only normalized schema can be adopted to store the arrays or matrices with arbitrary length.

Even though the individual cell in the array or matrix could be stored in the special table with appropriate datatype, we still store the array or matrix as a character string type in the table, since the character string type is still the most efficient format for data exchange. Although this storage schema requires more space, it speeds up the operations of reconstructing the data into XML format and exchanging required information between incompatible systems. In addition, we define a column for storing `ARRAYID` or `MATRIXID`, which points to the corresponding record in the `ARRAY` or `MATRIX` table,

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDB)

which are defined to store array or matrix data. Referring to Figure 4.2, we define two columns for the element `ColorValueIndex`, which is defined as an array type. One is for storing it as a character string type, and another is for storing the `ARRAYID` that points to the records in the `ARRAY` table. The corresponding example of database schema can be found in Figure 4.3.

Note that it is not necessary that users use the above schema to store all the arrays and matrices in MPEG-7 descriptions. This schema is designed to speed up the operations on the individual items of an array/matrix. For many arrays or matrices, however, no one is interested in a certain individual item, e.g., the arrays in `ColorStructure` descriptor. The common operations on these arrays or matrices require all the items in them. The above schema cannot improve such operations since it needs more joins to get corresponding data and the index on individual items is meaningless for such operations. They can only be stored with a character string datatype in the database.

The above storage schema, which stores the array/matrix data twice, may raise the problem of data consistency and integrity. This problem can occur during the updating of the array/matrix data. To solve this problem, we can define the triggers in the relational database to support the data consistency and integrity. If the array/matrix values stored as a character string type are updated, the triggers can perform the actions to update the corresponding values stored in `ARRAY/MATRIX` table. In addition, one limitation of this storage schema is the failure to support the nested arrays/matrices. Fortunately, it is practically not very relevant to the storage of MPEG-7 descriptions.

basicTimePoint and basicDuration The `basicTimePoint` datatype is used to describe a time point according to the Gregorian dates, day time and the time zone. It is represented in the following lexical format:

`YYYY-MM-DDThh:mm:ss.nnn.ff FNNN±hh:mm` [51]

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDb)

The following list gives the meaning of the lexical expression:[51]

- Y: year,
- M: month,
- D: day,
- T: the delimiter for the time specification,
- h: hour,
- m: minute,
- s: second,
- nnn.ff: decimal numbers of fractions,
- F: stands for the number of fractions of one second,
- N: number of fractions of one second which are counted by nnn,
- $\pm hh:mm$: represents the time zone.

The *basicDuration* datatype is used to specify the interval of time according to days and time of day. The lexical format of this type is given by:

$$PnDTnHnMnSnNfnF\pm hh:mmZ \text{ [51]}$$

In this format, the separators specify the semantic of the number n : D (days), H (hours), M (minutes), S (seconds), N (number of fractions), f (for a decimal expression of fractions), F (number of fractions of one second) [51]. Two examples of *basicTimePoint* and *basicDuration* would be expressed as: '2000-01-01T13:20:01:235F1000' and 'PT1M45S'.

All the leading database systems support time point type, e.g., 'timestamp' datatype in DB2 and Oracle, and 'datetime' datatype in SQL Server. Although the *basicTimePoint* datatype is based on ISO 8601, it is slightly different from the format accepted by the database system. We cannot directly insert this type into a relational table with built-in time point type in RDBMS without conversion. We also use two columns in a relational table to store the data with *basicTimePoint* datatype to give attention to data exchange and manipulation. One is for storing *basicTimePoint* as a character string for efficient

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDB)

data exchange, and the other is defined as a built-in time point datatype, e.g., timestamp in DB2, for time data operation. One of the most important operations on the time point data is time comparison. The original time point values in MPEG-7 descriptions are often represented with time zone. It is not efficient to directly compare the time point values with different time zones. During the mapping process, we first convert the time point data in MPEG-7 descriptions into a format acceptable to the database system, translate the time point data into local time, and store them into relational table with timestamp datatype; thereby utilizing the relational DB functionalities to directly and efficiently compare time point values.

The duration type may be involved in date and time arithmetic operations. All leading database systems support such operations as addition and subtraction. However, they differ in date arithmetic operations and the representation of duration operand. This article focuses on how to store **basicDuration** type in DB2. DB2 introduces four types of durations: labeled-duration, date duration, time duration and timestamp duration. Since the **basicTimePoint** type is stored as timestamp datatype, the timestamp duration type, which is expressed as a decimal number with precision 20 and scale 6, is the best mode to represent **basicDuration** type and then utilize the date and time arithmetic operations in DB2. For **basicDuration** type, we also use two columns, one for storage as character string, and the other for manipulating and recording as timestamp duration type.

Figure 4.6 uses an example extracted from a **UserDescription** description to illustrate the storage scheme for the **basicTimePoint** and **basicDuration** types. The date and time operations between **MediaTimePoint** and **MediaDuration** can be easily implemented with date and time functionalities provided by RDBMS. For example, if a user would like to get the result of adding **MediaDuration** to **MediaTimePoint**, the following simple SQL could be issued:

```
SELECT timestamp + decimal(timestampduration,20,6) FROM mediatime
```

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDB)

Example of timepoint and duration extracted from 'UserDescription' description:

```
<MediaTime>
  <MediaTimePoint>2000-10-09T19:10:12</MediaTimePoint>
  <MediaDuration>PT1M45S</MediaDuration>
</MediaTime>
```

Mapping scheme in MPD file:

```
<NodeClass Name="MediaTime" ToTable="mediatime">
  <LeafNodeClass>
    <ElementType Name="MediaTimePoint" />
    <ToColumn Name="mediatimepoint" Datatype="timepoint" />
    <ToColumn Name="timestamp" Datatype="timestamp" />
  </LeafNodeClass>
  <LeafNodeClass>
    <ElementType Name="MediaDuration" />
    <ToColumn Name="mediaduration" Datatype="duration" />
    <ToColumn Name="timestampduration" Datatype="varchar" />
  </LeafNodeClass>
</NodeClass>
```

Database schema:

....	MEDIATIME POINT	TIMESTAMP	MEDIA DURATION	TIMESTAMP DURATION
....	2000-10-09T19:10:12	2000-10-09-19.10.12.000000	PT1M45S	145

Figure 4.6: Storage scheme for basicTimePoint and basicDuration

4.2.5 Summary

According to the above storage schema, all the leaf nodes within MPEG-7 descriptions are mapped in fine-grained manner to corresponding columns with appropriate datatype. For the complex datatypes defined by MPEG-7 DDL, such as `array`, `matrix`, `basicTimePoint` and `basicDuration`, a special storage schema is designed to store them in relational database. Although such a storage schema needs more storage space and an additional mechanism to keep data consistency, it can speed up the special queries on these complex datatypes. SM3 can provide the most support for the fine-grained and typed representations and access of the contents of the MPEG-7 descriptions. With such a storage schema, the MPEG-7 description content can be indexed easily by built-in database indexes. Since the path expressions of all the internal nodes are kept and each internal node is labelled by ORDPATH, the sufficient structure information of MPEG-7 documents can be stored in RDBMS.

4.3 Insertion and Marshalling

4.3.1 Insertion Algorithm

The algorithm for inserting MPEG-7 data into RDBMS is shown in the Figure 4.7. The input of the algorithm is the original MPEG-7 document. At the end of the algorithm, all the data and structure information of the MPEG-7 document would be stored in the relational database. The insertion proceeds as follows:

- (i) The original MPEG-7 document is first parsed and the corresponding document tree is generated (line 3);
- (ii) Obtain the root element and corresponding XPath and ORDPATH of the root element (lines 4 to 6);

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDb)

Input: $\mathcal{D}$ - the MPEG-7 document to be mapped**Output:** $\mathcal{S}$ - collection of 'insert' SQL statements

```

1: interNodeRows is the collection of row data of each internal node
2: leafNodeRows is the collection of row data of leaf nodes
3: document = parse( $\mathcal{D}$ )
4: root = document.getDocumentElement( )
5: rootXPath = getXPath( root )
6: rootORDPath = getORDPath( root )
7: mappingNode( 1, root, rootXPath, rootORDPath, null )
8:  $\mathcal{S}$ .addInsertStatement( interNodeRows )
9:  $\mathcal{S}$ .addInsertStatement( leafNodeRows )
10: return  $\mathcal{S}$ 

```

(a) Insertion Algorithm

Input: level, node, xpath, ordpath, parent_ordpath**Output:** *interNodeRows* - collection of row data of internal nodes*leafNodeRows* - collection of row data of leaf nodes

```

1: interNodeRow  $\leftarrow$  new InternalNodeRowClass(level, node, xpath, ordpath )
2: interNodeRows.add( interNodeRow )
3: leafNodeRow  $\leftarrow$  new RowClass( )
4: attributes = node.getAttributes( )
5: processAttributes( leafNodeRow, attributes )
6: for ( childNode = node.getFirstChild();
7:   childNode != null;
8:   childNode = childNode.getNextSibling( ) ) do
9:   if ( childNode is internal node ) then
10:    childNode_xpath = getXPath( childNode )
11:    childNode_ordpath = getORDPath( childNode, parent_ordpath )
12:    mappingNode( level+1, childNode, childNode_xpath, childNode_ordpath, ordpath )
13:   else if ( childNode is leaf node ) then
14:    processLeafNode( leafNodeRow, childNode )
15:   end if
16: end for
17: leafNodeRows.add( leafNodeRow )
18: return interNodeRows, leafNodeRows

```

(b) Procedure mappingNode

Figure 4.7: Insertion Algorithm

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDB)

- (iii) Call the function `mappingNode( )` to process the root element (line 7);
- (iv) Generate 'insert' SQL statements (lines 8 to 9); and
- (v) The function `mappingNode` is used to map each internal node which proceeds as follows (refer to Figure 4.7(b)):
 - generate one row of data for this internal node and add it to the internal node rows collection (lines 1 to 2);
 - define a variable `leafNodeRow` as an instance of `RowClass` class to record the row data of each leaf node that belongs to this internal node (line 3);
 - call function `processAttributes` to process attributes of this internal node with the aid of an MPD file. After this process, the corresponding column information and the value with appropriate datatype of each attribute will be recorded into `leafNodeRow` (lines 4 to 5);
 - process all children of this internal node (lines 6 to 16). If the child node is also an internal node, call the function `mappingNode` to map this node (lines 9 to 12). If the child node is a leaf node, call function `processLeafNode` to process it, including getting column name, converting datatype and adding corresponding row information to the `leafNodeRow` (lines 13 to 14); and
 - add `leafNodeRow` to leaf node rows collection (line 17).

4.3.2 Marshalling Algorithm

The marshalling process is the counter-procedure of the insertion process. The first step is to extract data from the database, and then reconstruct them to revert to the original XML format. The algorithm for reconstruction is presented in Figure 4.8. The marshalling process proceeds as follows:

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDb)

<p>Input: $\mathcal{L}_1 \{l_1, l_2, \dots, l_k\}$ - list of internal nodes data ordered by UID, which is depth-first order of internal node $\mathcal{L}_2$ - list of all leaf nodes data</p> <p>Output: $\mathcal{D}$ - an XML document</p> <p>1: n, p are instance of Class 'NodeClass' which including XML node and level information.</p> <p>2: s is a stack.</p> <p>3: for all elements in $\mathcal{L}_1$ do</p> <p>4: if $n = \text{null}$ then</p> <p>5: $n.\text{node} = \mathcal{D}.\text{createElement}(l_i.\text{nodeName})$</p> <p>6: $n.\text{level} = l_i.\text{level}$</p> <p>7: $\mathcal{D}.\text{appendChild}(n.\text{node})$</p> <p>8: $\text{setLeafNodeChildren}(n.\text{node}, \mathcal{L}_2)$</p> <p>9: $p = n$</p> <p>10: else if</p> <p>11: $n.\text{node} = \mathcal{D}.\text{createElement}(l_i.\text{nodeName})$</p> <p>12: $n.\text{level} = l_i.\text{level}$</p>	<p>13: if $n.\text{level} > p.\text{level}$ then</p> <p>14: $s.\text{push}(p)$</p> <p>15: else if $n.\text{level} = p.\text{level}$ then</p> <p>16: $p = s.\text{peek}()$</p> <p>17: else if</p> <p>18: while $n.\text{level} \leq p.\text{level}$ do</p> <p>19: $s.\text{pop}()$</p> <p>20: $p = s.\text{peek}()$</p> <p>21: end while</p> <p>22: end if</p> <p>23: $\text{setLeafNodeChildren}(n.\text{node}, \mathcal{L}_2)$</p> <p>24: $p.\text{node}.\text{appendChild}(n.\text{node})$</p> <p>25: $p = n$</p> <p>26: end if</p> <p>27: end for</p> <p>28: return $\mathcal{D}$</p>
---	--

Figure 4.8: Marshalling algorithm

- (i) Extract all the internal nodes data and leaf nodes data from the database as the input parameters of reconstruction algorithm;
- (ii) The variables n and p are the instance of NodeClass class that records XML node and its level information. n is for current node and p is for the parent node of n and the node processed as previously;
- (iii) For all the internal nodes, the root element is first processed (lines 4 to 9). Then, the rest of the internal nodes are handled one by one;
- (iv) For each internal node data, after creating a corresponding XML node (line 11) and getting its level information (line 12), we can find its parent node information (lines 13 to 22). Then, add leaf nodes to the current internal node (line 23) and append the current internal node to the parent node as a child (line 24); and
- (v) The function `setLeafNodeChildren` (line 8 and line 23) is used to insert the leaf nodes to the current internal node as its children. With the aid of MPD file, we can identify the element type of these leaf nodes, i.e. attribute or element type

with PCData-only content, and then insert them into current internal node with correct positions.

4.4 XPath-based Query Support

With SM3, applications can directly use SQL to access the contents of media descriptions in a fine-grained manner. However, the database schema of MPEG-7 storage solution is often not transparent to the users or applications. They prefer to access MPEG-7 media descriptions through some form of declarative XML query language, e.g., XPath and XQuery. In order to support the requirement of XML query language, it is necessary for the MPEG-7 description management solution to be powerful enough to provide appropriate translators from XPath and XQuery to SQL.

XQuery[52] is a query language built on XPath expressions to find and extract elements and attributes from XML documents. It has been widely applied across many types of XML data sources. XQuery offers iterative and transformative capabilities through FLWOR expressions, which stand for the five major clauses: *for*, *let*, *where*, *order by* and *return*. To support XQuery in SM3, we developed a translation mechanism for converting XQuery to SQL. This mechanism supports many features of XQuery, which include simple or recursive path expressions, predicate expressions (including order predicate and value comparison predicate), arithmetic expressions, comparison expressions and logical expressions[52]. However, due to the complexity of XQuery and the gaps between XQuery and SQL, it is difficult to translate all the features of XQuery into SQL, such as conditional expressions, quantified expressions and expressions on sequence types. SQL does not completely support these expressions. The conditional expressions may be implemented with PL/SQL in ORACLE, but it is not applicable to SQL. In addition, XQuery includes over 100 built-in functions. Many of them are not supported with SQL,

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDB)

Figure 4.9: An example of XQuery and corresponding SQL translated by SM3

for example, codepoints-to-string, translate, escape-uri, etc. Figure 4.9 demonstrates an XQuery expression.

4.4.1 Query Translation Algorithm

There are several steps in our query translation process. The first step is to use a parser to parse the XQuery and then generate a corresponding parse tree. One example of a parser generator tool is JavaCC, the parser generator used with Java applications. Figure 4.10 shows the parse tree for the XQuery shown in Figure 4.9.

The second step is to walk this parse tree, and generate all the PathExpr and ComparisonExpr within the XQuery in question. In XQuery, PathExpr represents a path expression that can be used to locate nodes within the XML tree, and ComparisonExpr represents a comparison expression that allows two values to be compared. For example, in the XQuery shown in Figure 4.9, the PathExprs and ComparisonExprs are listed

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDB)

Figure 4.10: Parse tree for the XQuery in Figure 4.9

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDB)

PathExpr :

```
//Descriptor
//Descriptor/Values/ColorValueIndex ($b/Values/ColorValueIndex)
//Descriptor/Values/Percentage ($b/Values/Percentage)
//Descriptor/SpatialCoherency ($b/SpatialCoherency)
```

ComparisonExpr :

```
$b/Values/ColorValueIndex = '44 67 30'
$b/Values/Percentage > 5
```

Figure 4.11: PathExprs and ComparisonExprs of the XQuery in Figure 4.9

```
Input: XQuery query  $X$ 
Output: Translated SQL query  $S$ 

1: parse tree  $T = \text{parse}(X)$ 
2:  $E = \text{walkParseTree}(T)$ 
3: SelectClause  $s = S.\text{getSelectClause}()$ 
4: FromClause  $f = S.\text{getFromClause}()$ 
5: WhereClause  $w = S.\text{getWhereClause}()$ 
6: for all PathExpr  $p_i$  in  $E.\text{allPathExpr}$  do
7:   processPathExpr( $p_i, f, w$ )
8:   if  $p_i$  included in return clause then
9:      $s.\text{add}("Ti." + p_i.\text{getColumn}())$ 
10:   end if
11: end for
12: for all ComparisonExpr  $c_i$  in  $E.\text{allComparisonExpr}$  do
13:   processComparisonExpr( $c_i, f, w$ )
14: end for
15: return  $S$ 
```

Figure 4.12: Translation algorithm

in Figure 4.11. Finally, these PathExprs and ComparisonExprs can be translated into a corresponding SQL component. Figure 4.12 shows the translation algorithm for SM3.

The translation proceeds as follows:

- (i) Parse the XQuery and generate a parse tree (line 1). Then, walk the parse tree and generate all the PathExprs and ComparisonExprs in XQuery (line 2);
- (ii) Process all the PathExprs (lines 6 to 11). The procedure processPathExpr will be

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDB)

called to generate corresponding ‘from’ and ‘where’ clause; and

- (iii) Procedure `processComparisonExpr` is called to process all the `ComparisonExprs` (lines 12 to 14).

Procedure of `processPathExpr` is shown in Figure 4.13 and this procedure proceeds as follow:

- (i) Since the path expressions are stored in the ‘xpath’ table and the internal node information is stored in the ‘internalnode’ table, we need to join the two tables to get the internal node information (lines 1 to 2);
- (ii) If the end node of this path expression is leaf node, add the corresponding table name to `FROM` clause (lines 3 to 6);
- (iii) Lines 7 to 37 are for processing filter expressions in the path expression. First, parse each filter expression to extract prefix path expression and filter path expression, and then add corresponding `FROM` and `WHERE` clauses (lines 8 to 15). For example, in the following path expression:

```
//Descriptor[Values/Percentage='3']/SpatialCoherency
```

the prefix path expression is `//Descriptor`, the filter path expression is `//Descriptor/Values/Percentage`, and path expression `//Descriptor/SpatialCoherency` could be called a destination path expression. In later processes, we may check the ancestor-descendant relationships among these three types of path expression to gain the final result; and

- (iv) Handle the conditions in filter expression (lines 16 to 36). There are two types of conditions to be handled: index predicate and comparison predicate.

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDb)

```

Input:
    PathExpression  $p_i$ , FromClause  $f$ , WhereClause  $w$ 
Output:
    FromClause  $f$ , WhereClause  $w$ 

1:  $f.add("xpath\ Xi,\ internalnode\ Ii")$ 
2:  $w.add("Xi.xpathexp = " + p_i.getPathExpr() + " and Xi.xpathid = Ii.xpathid")$ 
3: if the end node in  $p_i$  is leaf node then
4:    $table_i = p_i.getTable()$  /* get the table which stores this leaf node */
5:    $f.add(table_i + " Ti")$ 
6: end if
7: for all FilterExpr  $f_j$  in  $p_i$  do
8:   PathExpr  $p_{ij0} =$  prefix pathexpr of  $f_j$ 
9:   if  $p_{ij0}.length = p_i.length$  then
10:     $p_{ij0} = p_i$ 
11:   else
12:     $f.add("xpath\ Xij0,\ internalnode\ Iij0")$ 
13:     $w.add("Xij0.xpathexp = " + p_{ij0}.getPathExpr() + " and Xij0.xpathid = Iij0.xpathid")$ 
14:     $addRelationship(p_{ij0}, p_i, w)$ 
15:   end if
16:   if  $f_j$  is index predicate then /* handle PathExpr like //Descriptor[6]... */
17:     if the end node in  $p_{ij0}$  is leaf node then
18:        $w.add("Ti.oid=" + f_j.getIndexPredicate())$ 
19:     else /* the end node in  $p_{ij0}$  is internal node */
20:        $w.add("Iij0.oid=" + f_j.getIndexPredicate())$ 
21:     end if
22:   else if  $f_j$  is comparison express then
23:     if the first node in filter is leaf node then
24:        $table_{ij0} = p_{ij0}.getTable()$ 
25:        $column =$  column corresponding to this leaf node
26:        $f.add(table_{ij0} + " Tij0")$ 
27:        $w.add("Tij0." + column + f_j.getOperator() + f_j.getRightOperand())$ 
28:     else /* the first node in the filter is internal node */
29:        $p_{ij1} =$  full pathexpr of left operand of the comparison express in  $f_j$ 
30:        $table_{ij1} = p_{ij1}.getTable()$ 
31:        $column = p_{ij1}.getColumn()$ 
32:        $f.add(table_{ij1} + " Tij1")$ 
33:        $w.add("Tij1." + column + f_j.getOperator() + f_j.getRightOperand())$ 
34:        $addRelationship(p_{ij0}, p_{ij1}, w)$ 
35:     end if
36:   end if
37: end for
38: return  $f, w$ 

```

Figure 4.13: Procedure *processPathExpr*

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDb)

```

Input:
    ComparisonExpr  $c_i$ , WhereClause  $w$ 
Output:
    WhereClause  $w$ 

1: PathExpr  $l = c_i.getLeftOperand()$ 
2:  $operator = c_i.getOperator()$ 
3:  $r = c_i.getRightOperand()$ 
4: if  $r$  is Literal then
5:    $w.add(l.getTableAlias() + "." + l.getColumn() + operator$ 
      $+ r.getLiteralExpr())$ 
6: else if  $r$  is PathExpr then
7:    $w.add(l.getTableAlias() + "." + l.getColumn() + operator$ 
      $+ r.getTableAlias() + "." + r.getColumn())$ 
8: end if
9: return  $w$ 

```

Figure 4.14: Procedure *processComparisonExpr*- **index predicate**

Consider the following example: `//Descriptor[6]/Values/...`

In our storage schema, the column *oid* is introduced to store the ordinal information of each element. In the above example, if the element *Descriptor* is a leaf node (repeatable leaf node), the following statement would be added in WHERE clause: $T.oid = 6$, where T is the alias of the table that stores this leaf node value (lines 17 to 18). In the tables that store the repeatable leaf nodes, we also introduce the column *oid* to record the corresponding ordinal information of each repeatable leaf node (see Figure 4.3 (e) colorquantization table). If element *Descriptor* is an internal node, the following statement would be added in WHERE clause: $I.oid = 6$, where I is the alias of corresponding internalnode table (lines 19 to 20); and

- **comparison predicate**

If the first element of a filter path expression is a leaf node, for example, the element *Percentage* in the path expression

`//Descriptor/Values[Percentage='3']/...`,

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDDB)

the following statement would be added in WHERE clause: $T.\text{percentage} = 3$, where T is the alias of the table that corresponds to the internal node **Values** (lines 23 to 27). If the first element of a filter path expression is an internal node, for example, the element **Values** in the path expression

`//Descriptor[Values/Percentage='3']/...`,

after adding the statement $T.\text{percentage} = 3$, we need to add the statement to check the parent-child relationship between `//Descriptor` and `//Descriptor/Values` (lines 28 to 34). In subsection 4.2.1, we have discussed how to get the parent-child or the ancestor-descendant relationship between two nodes with ORD-PATH. The following statement would implement the above relationship evaluation: $I_1.\text{ordpath} = I_2.\text{parent}$, where I_1 is the alias of the `internalnode` table that corresponds to `//Descriptor` and I_2 corresponds to `//Descriptor/Values`.

Figure 4.14 shows the procedure of `processComparisonExpr`. This procedure is as follows:

- (i) Obtain left and right operands and operator in the comparison expression (lines 1 to 3); and
- (ii) Two types of comparison expression need to be handled.

- In the first type, the right operand is literal, e.g.

`//Descriptor/Values/Percentage='5'.`

The following SQL statement would be added in WHERE clause:

$T.\text{percentage} = 5$,

where T is the alias of the table that corresponds to the internal node **Values** (lines 4 to 5); and

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDB)

- In the second type, the right operand is also a path expression, e.g.

```
//FilteringAndSearchPreferences/PreferenceCondition/Place/Name
= //BrowsingPreferences/PreferenceCondition/Place/Name.
```

The following SQL statement would be added in WHERE clause:

$$T_l.name = T_r.name,$$

where T_l is the alias of the table that corresponds to the left path expression and T_r corresponds to the right path expression (lines 6 to 7).

The SQL result for translating the XQuery example in Figure 4.9 according to the above translation algorithm is shown in the right panel in Figure 4.9.

With the aid of the XPath expressions in XQuery and the position information of related nodes stored in the database, the return of results could be organized with XML format. The output algorithm is similar to the corresponding parts of marshalling algorithm.

4.4.2 Query Rewriting

The leading RDBMS products have provided multiple ways to improve the query performance, such as indexing and query optimizer. However, there is still space for performing the query more efficiently. To evaluate the performance of the above translation procedure, we captured information about the access plan of the above SQL statement. The access plan in DB2 is generated by DB2 optimizer. An access plan specifies an order of operations for accessing the data that is necessary to resolve a SQL statement. The captured information helps us understand how individual SQL statements are executed so that we can tune the statements. The access plan of the SQL statement in Figure 4.9 that is based on 1GB MPEG-7 dataset is shown in the Figure 4.15 (a).

In this access plan graph, rectangles represent tables and diamonds represent operators. Operator is either an action that must be performed on data, or the output from a

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDDB)

Figure 4.15: Access plan for SQL statement in Figure 4.9

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDb)

table or an index, when the access plan for an SQL statement is executed. The operators occurring in this graph are explained as follows:

RETURN – Represents the return of data from the query to the user;

MSJOIN – Represents a merge join, where both outer and inner tables must be in join-predicate order;

NLJOIN – Represents a nested loop join that accesses an inner table once for each row of the outer table;

FILTER – Filters data by applying one or more predicates to it;

TBSCAN – Retrieves rows by reading all required data directly from the data pages; and

FETCH – Fetches columns from a table using a specific record identifier.

The number under each operator indicates the total cost that is the estimated total resource usage necessary to execute a corresponding operation. Cost is derived from a combination of CPU cost (in number of instructions) and I/O cost (in numbers of seeks and page transfers).

In our optimization study with access plan, we noticed that the joins between the `xpath` and `internalnode` tables consumed a considerable portion of the query processing time, and the cost of such joins cannot be decreased with the query optimizer of database system. The main effects of the joins between `xpath` and `internalnode` on the total query performance for the large dataset, is that large amounts of records in an `internalnode` table increase the size of joins between `xpath` and `internalnode` tables dramatically.

In order to avoid these time-consuming joins, we re-wrote the queries to optimize the query process. This process is similar to the optimization technique discussed in SUCXENT++[26]. In the query rewriting process, the join expression:

$$\text{xpath.xpathexp} = \text{xpath} \text{ and } \text{internalnode.xpathid} = \text{xpath.xpathid}$$

would be placed with:

$$\text{internalnode.xpathid} = n$$

where n is the `xpathid` value corresponding to the path expression in the `xpath` table.

Similarly, the following expression:

CHAPTER 4. MPEG-7 DESCRIPTION DATABASE (MDDB)

`xpath.xpathexp` like `xpath%` and `internalnode.xpathid = xpath.xpathid`

is replaced with:

`internalnode.xpathid in (...)`

where the value in the parentheses is the set of `xpathid` values corresponding to the path expression in the `xpath` table. The rewriting query process includes two steps: first, accessing `xpath` table and getting the `xpathid` value corresponding to the path expression; second, using these values to write the final SQL statement and then perform it to get the final result. The optimized access plan for SQL in Figure 4.9 is shown in Figure 4.15 (b).

Although such optimized query processes need to access the database multiple times, it avoids the joins between `xpath` and `internalnode` tables. When there exists a large amount of data in `internalnode` table, the query performance is improved dramatically (up to 10 times in our experiment).

Chapter 5

Multimedia Content Database (MCDB)

This chapter represents the other main component of IXMMS: MCDB. MCDB is designed to organize the low-level features of multimedia objects stored in the MPEG-7 descriptions, provide multi-dimensional index mechanism, and support the content-based retrieval.

5.1 Overview of MCDB

As mentioned in Chapter 3, Multimedia Content Database (MCDB) is used for multimedia content storage. Once the MPEG-7 low-level descriptors are mapped into MDDB by SM3, the multimedia content recorded in these low-level descriptors can be extracted from MDDB and re-organized in terms of the requirements for content-based retrieval and the corresponding similarity function. At this stage, we can also clean up the multimedia content data, e.g., dimension reduction for creating a more efficient index system. Whereafter the prepared data will be written into MCDB.

A possible argument may be “why not extract the multimedia content directly from the original MPEG-7 descriptors?” In fact, the intrinsic drawbacks of XML documents prevent such an extraction operation from being performed efficiently. The multimedia

CHAPTER 5. MULTIMEDIA CONTENT DATABASE (MCDB)

content of one multimedia object could be kept in several individual MPEG-7 documents. An extraction process would require access to these documents one by one. The document parsing and the text format conversion make the extraction operation very expensive when there are large collections of MPEG-7 descriptors. Undoubtedly, RDBMS could overcome these drawbacks by drawing upon its many advantages, such as efficient index mechanism, query optimization techniques, advanced processing mechanism, etc.

MCDB is based on ORDBMS. A set of object types is defined to store multimedia content. The multimedia content within each low-level MPEG-7 descriptor is mapped into its respective object type, which is defined by the corresponding similarity search function. Figure 5.1 shows examples of object type definitions for the descriptors RegionShape and DominantColor.

RegionShapeType	DominantColorType
-ObjectID : int	-ObjectID : int
-FID : int	-FID : int
-OID : int	-OID : int
-MagnitudeOfART : String	-Size : int
+RegionShapeType()	-SpatialCoherency : int
	-Value : String
	+DominantColorType()

Figure 5.1: Examples of Object Type Definitions

The attributes FID and OID in the above definitions are used to identify the corresponding MPEG-7 file and the position of the descriptor in the MPEG-7 file. These two attributes facilitate the connection between the object type in the MCDB and the corresponding content in the MDDB. In the RegionShapeType, the value of MagnitudeOfART is different from its original value because of the data transformation required for the similarity matching, e.g., the original value of ‘1’ should be changed to ‘0.005468893’ [47]. In the DominantColorType, the value of attribute Value is the collection of the values of

CHAPTER 5. MULTIMEDIA CONTENT DATABASE (MCDB)

Percentage and ColorValueIndex in the DominantColor descriptor. Data transformation is also required for the DominantColorType, since the DominantColor descriptors can be created with different color spaces. To make the color values comparable for the similarity matching in the MCDB, all the color values are converted to the LUV color space.

When constructing the Multimedia Content Database, two critical mechanisms need to be considered in depth: the synchronization mechanism and the high dimension index mechanism. We will discuss them in detail in the following sections.

5.2 Synchronization mechanism

Since the multimedia content is extracted from the MDDB, the data in the MCDB must be synchronized with the corresponding data in the MDDB. The synchronization mechanism needs to deal with two key problems: which data needs to be extracted and how to synchronize the data?

Figure 5.2: Database Model Diagram for DominantColor Descriptor

CHAPTER 5. MULTIMEDIA CONTENT DATABASE (MCDB)

The selection of the fields in the MDDB to be synchronized with the fields in the MCDB depends on the MPEG-7 description mapping schema and the similarity function for individual descriptors. The **DominantColor** descriptor is used as an example to illustrate this point. Figure 5.2 shows a database model diagram for storing the **DominantColor** descriptor. The similarity function for **DominantColor** descriptor described in [47] is shown as follows:

Given two **DominantColor** Descriptors:

$$F_1 = \{(c_{1i}, p_{1i}), s_1\}, i = 1, \dots, N_1 \text{ and} \\ F_2 = \{(c_{2j}, p_{2j}), s_2\}, j = 1, \dots, N_2$$

where N is the number of the colors, c is color index, p is percentage, and s is spatial coherency. The distance function using **DominantColor** descriptor is

$$Dist = W1 * SC_Diff * DC_Diff + W2 * DC_Diff$$

where $W1$ and $W2$ are weight, $SC_Diff = abs(s_1 - s_2)$, and DC_Diff is computed by the function $D(F_1, F_2)$:

$$D^2(F_1, F_2) = \sum_{i=1}^{N_1} p_{1i}^2 + \sum_{j=1}^{N_2} p_{2j}^2 - \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} 2a_{1i,2j} p_{1i} p_{2j}$$

$$\text{where } a_{k,l} = \begin{cases} 1 - d_{k,l}/d_{max} & d_{k,l} \leq T_d \\ 0 & d_{k,l} > T_d \end{cases}, \quad d_{k,l} = \|c_k - c_l\|$$

T_d is the maximum distance for two colors to be considered similar, and $d_{max} = \alpha T_d$. A normal value for T_d is between 10-20 in the CIE LUV color space and for α is between 1.0–1.5.[47]

According to Figure 5.2, the **INTERNALNODE** table is for storing information on internal nodes, and the other three tables are for all the leaf nodes within the **DominantColor** descriptor. According to the above similarity function for the **DominantColor** descriptor, the fields, size and spatialcoherency in the **DOMINANTCOLOR** table, percentage and colorvalueindex in the **VALUES** table, are to be synchronized with the data in the MCDB for similarity search on the dominant color feature. This synchronization information needs to be written in a configuration file and then transferred to the synchronization engine, or directly embedded in the synchronization engine, in order to specify which fields are to be synchronized.

CHAPTER 5. MULTIMEDIA CONTENT DATABASE (MCDB)

There are two ways for data synchronization: polling and callback. A callback mechanism allows a synchronized database to push required data to the synchronizing database when specific data changes in the synchronized database. The polling mechanism allows the synchronizing database to get required data from the synchronized database periodically. The polling process is more complex than callback. The synchronizing database needs to send synchronization calls periodically, and pass a timestamp of the last call to the synchronization engine. This engine then retrieves all the new or updated data from the synchronized database since the last timestamp and maps this data to the synchronizing database.

For simplicity, a callback mechanism has been considered in our system. A trigger can be installed in the MDDB to call the synchronization operation whenever necessary. This operation requires parameters to be defined which describe the data change to the MDDB, the type of action (insert or update), and the name of the affected descriptors. These parameters will be passed to the MCDB and trigger the update operations in the MCDB. If the MDDB and the MCDB reside in a distributed environment and are located in different sites, a synchronization web service is then required to be defined and the trigger in the MDDB would call this synchronization service when specific data changes in the MDDB.

5.3 Extensible high-dimensional index mechanism

Due to the nature of the multimedia data, a multidimensional data index structure is important for IXMMS to support multimedia content retrieval. As indicated in [10], it is impossible to choose the best index structure for all kinds of MPEG-7 applications and a certain index structure which best fits the needs of a specific application might not be useful for other applications. Therefore, the index mechanism in the MPEG-7 management system should be extensible. However, most existing database systems

offer only one- or low-dimensional index structures, and rarely handle high-dimensional index structures and similarity searching functionality for multimedia content retrieval. These observations led us to design an extensible multidimensional index mechanism based on the GiST framework.

5.3.1 Overview of GiST framework

The GiST framework[44] is a widely used template indexing structure for enhancing the DBMS to support extensible indexing mechanism. An earlier attempt in this direction was described in [53], which proposed a new approach to extensible indexing that integrated the GiST framework into Informix Dynamic Server with Universal Data Option (IDS/UDO). GiST supports any kind of balanced index tree on multidimensional data. The DBMS can be enhanced with the GiST framework to support all newly developed index structures, which can be used to index the corresponding data types in the database.

The GiST framework is easily extensible in both the data types and the index structures. The GiST is a balanced tree structure with $\langle \text{key}, \text{RID} \rangle$ pairs in the leaf nodes and $\langle \text{predicate}, \text{pointer} \rangle$ pairs as internal nodes. For the leaf nodes, the key is a member of a user-defined class, and the RID points to the corresponding records on the data pages. For the internal nodes, the predicate represents some property that holds true for all data items reachable from the pointer. The GiST provides a set of algorithms for the standard index operations: SEARCH, which is used to search any dataset with any predicate by traversing the tree to satisfy that predicate; INSERT, which adds a $\langle \text{key}, \text{RID} \rangle$ pair to the tree and guarantees that the GiST remains balanced; and DELETE, which removes a node and maintains the balance of the tree. Since the GiST has no restrictions on the key data stored within the tree and the key space is not imposed to be ordered, the GiST framework can support multidimensional data and to build any kind of balanced index tree on such data.

5.3.2 High-dimensional index system in MCDB

In IXMMS, the GiST framework runs as an independent process and manages all available access methods, which can be used for the data types in the MCDB. To enable the MCDB to access the GiST framework, we developed a set of database routines that are used by the database to connect to the GiST framework. These routines provide necessary functionalities, such as, creating, inserting, deleting and searching, for accessing the index structures created in the GiST framework. They also transfer input and output data between the GiST framework and the database. With these routines, the database server is insulated against failures during high-dimensional index operations. The end users can benefit from this index mechanism by calling these database routines while the index system itself is transparent to them.

The current GiST version comes prepackaged with extensions for some multidimensional index structures, such as R-tree[54], R*-tree[55], SS-tree[56] and SR-tree[57]. These index structures can be viewed as the spatial access methods. To support the similarity searching function based on metric space, the M-tree[58], a representative of the metric-based indices, is added to IXMMS.

Chapter 6

MXQuery

An adequate query language is important for the MPEG-7 management system. In IXMMS, an extension of XQuery, known as MXQuery, is proposed as an adequate MPEG-7 query language, which supports the queries on complex datatypes defined in MPEG-7 DDL, queries on spatial-temporal relationships and query-by-example. This chapter gives a detailed discussion on MXQuery. Since there exist a large number of different multimedia query demands and it is difficult to elaborate all kinds of high level multimedia query functions, this research work only focuses on the fundamental query issues. Based on these basic definitions, the MPEG-7-based multimedia applications can extend them to implement their special query demands.

6.1 Introduction

Although MPEG-7 descriptions are stored into traditional database system, which can be accessed via SQL, the end users prefer to access MPEG-7 media descriptions through some form of a declarative XML query language. Several XML query languages have been proposed for querying XML data, such as XQL[59], XML-QL[60], XQuery[52], etc. Among these languages, XQuery [52] is a forthcoming standard for XML document retrieval. XQuery is an expression language built on XPath expressions, together with some optional functions and other definitions for finding and extracting the elements and

CHAPTER 6. MXQUERY

attributes from the XML documents. As a powerful and convenient language for querying XML data, XQuery has become a W3C Candidate Recommendation (previously it was only a Working Draft) and will become a W3C standard. It has been supported by all the major database engines (IBM, Oracle, Microsoft, etc.).

However, due to the limitations of specifying spatial, temporal or visual data and relationships in the MPEG-7 descriptions, XQuery cannot adequately support MPEG-7 descriptions queries. An adequate MPEG-7 query language should fulfill two main requirements. First, as MPEG-7 is a standard for multimedia information, same as the other multimedia query languages, the MPEG-7 query language should support various multimedia datatypes and relationships. For example, the queries may look like “What is the transition probability from ‘Pass’ to ‘Shot on goal’ in a football match video”, or “What’s the object which locates at the right of point (10,10) in an image”. Second, since the MPEG-7 descriptions are also XML documents, the MPEG-7 query language should support the retrieval of MPEG-7 descriptions in XML format.

There are a few research works on designing a special XML query language to support MPEG-7 document queries. MMDOC-QL[61] is an early example of MPEG-7 query language. MMDOC-QL uses a logic formalism called **path predicate calculus** for specifying spatial and temporal relationships to support MPEG-7 descriptions retrieval and modification. In path predicate calculus, the atomic logic formulas are element predicates rather than relation predicates in relational calculus.[61] However, MMDOC-QL is not based on XQuery. An example of MPEG-7 query language based on XQuery is SVQL[62]. SVQL is an adaptation of W3C XQuery based on the semantic view model (PhysicalView, ProductionView, ThematicView, VisualView and AudioView) for retrieving MPEG-7 descriptions in a TV news retrieval application. The semantic view model for MPEG-7 descriptions is like the ‘View’ concept for traditional RDBMS. Such a view model is built on the explicit content within the MPEG-7 descriptions. Therefore, SVQL

lacks the ability to support the implicit multimedia relationships such as point-inside, region-overlap, etc. Furthermore, SVQL is designed for TV news retrieval application, thus it lacks extensibility and comprehensive applicability.

To support the queries on MPEG-7 descriptions, we propose an extension of XQuery, called MXQuery, as an adequate MPEG-7 query language. In addition to supporting general XML queries, MXQuery focuses on queries on complex datatypes within the MPEG-7 descriptions, queries on spatial-temporal relationships and query-by-example.

6.2 Special Query Issues for MPEG-7 Descriptions

Since the MPEG-7 descriptions heavily depend on the XML schemas, the fundamental queries on MPEG-7 descriptions are similar to the queries on the general XML documents. That is to find and extract elements and attributes from XML documents. However, the multimedia speciality of MPEG-7 descriptions poses some special issues for the MPEG-7 query language design. Three crucial issues are addressed as follows.

Complex datatypes The first issue is the complex datatypes defined in MPEG-7 DDL. MPEG-7 descriptions not only use standard datatypes, but also add extension datatypes, including `array`, `matrix`, and some temporal datatypes, e.g., `basicTimePoint` and `basicDuration`. An adequate MPEG-7 query language should support the operations on these complex datatypes. For the temporal datatypes, the MPEG-7 query language should provide a way to represent all kinds of temporal relationships, such as `before`, `after`, `during`, etc. For the array/matrix datatype, an operation of extracting a certain individual item within an array or matrix should be supported. For example, for the `DominantColor` descriptor, if the colour space is HMMD, the value of `ColorValueIndex`, which is defined as array type with 3 length, would be a set of three components: `Hue`, `Diff` and `Sum`. If the users want to find the pictures with blue dominant colour, the

following condition will be issued: “the value of Hue should be from 160 to 210” (this value range specifies the blue colour). In the above query, the operation of extracting a certain item in the array data of `ColorValueIndex` will be issued.

Spatial and temporal information The complex spatial and temporal information embedded in the MPEG-7 descriptions is the main factor that makes an adequate MPEG-7 query language different from general XML query language. The temporal data represents the relationships in time between the media objects and the spatial data specifies the positions of the media objects within the media. The spatial-temporal operators should be provided with the MPEG-7 query language to support the spatial and temporal relationships.

Query by example (QBE) Query-by-example allows users to find media objects information in large multimedia databases based on low-level audiovisual features, such as color, shape, texture, etc. As a multimedia content description standard, MPEG-7 descriptions store a large amount of low-level multimedia information. Therefore, the QBE facility should be provided within the MPEG-7 query language.

6.3 MXQuery Specification

6.3.1 Specification for complex datatypes

The MPEG-7 standard introduces several extension datatypes, including `array`, `matrix`, and some temporal datatypes, e.g., `basicTimePoint` and `basicDuration`. The format of `basicTimePoint` and `basicDuration` is similar to the `dateTime` and `duration` datatypes in the specification of the XML Schema language[63]. They are all defined following the ISO 8601 standard. W3C has provided adequate functions for `dateTime` and `duration` datatypes [64], thus MXQuery does not provide any special function for `basicTimePoint` and `basicDuration` datatypes.


```

DECLARE FUNCTION mpeg7:arrayItem ($array as string, $i as integer)
as double
{
  for $a at $k in fn:tokenize($array, "\s+")
  where $k = $i
  return fn:number($a)
}

DECLARE FUNCTION mpeg7:matrixItem ($array as string, $i as integer,
  $j as integer, $n as integer)
as double
{
  for $a at $k in fn:tokenize($array, "\s+")
  let $id := ($i-1) * $n + $j
  where $k = $id
  return fn:number($a)
}

```

Figure 6.1: Function definitions of arrayItem() and matrixItem()

We focus on the operation of extracting a certain item from an array or matrix. To support this operation, we introduce two functions: `arrayItem()` and `matrixItem()` to extract the desired item from an array or matrix. Figure 6.1 shows the function definitions of these two functions. In the function `arrayItem()`, the parameter ‘\$array’ stands for the array data and ‘\$i’ for the i -th item in the array. In the function `matrixItem()`, ‘\$i’ and ‘\$j’ stand for the i -th row and the j -th column, and ‘\$n’ for the dimension of the column.

Figure 6.2 and Figure 6.3 show the MXQuery examples for the `arrayItem()` and `matrixItem()` functions.

6.3.2 Specification for spatial-temporal information

XQuery is sufficient for finding and extracting explicit content from MPEG-7 descriptions. However, it lacks the ability to handle the spatial-temporal relationships between media objects represented in an implicit manner inside MPEG-7 descriptions. Hence, a set of operators are added to MXQuery to express the spatial-temporal relationships.

CHAPTER 6. MXQUERY

Figure 6.2: An MXQuery example for arrayItem() function call

Figure 6.3: An MXQuery example for matrixItem() function call

CHAPTER 6. MXQUERY

MPEG-7 description:

```

... ..
<Object id="tabletop-object">
  <Label><Name>Table top</Name></Label>
  <MediaOccurrence>
    <MediaLocator>
      <MediaUri>image.jpg</MediaUri>
    </MediaLocator>
    <Mask xsi:type="SpatialMaskType">
      <SubRegion>
        <Polygon>
          <Coords mpeg7:dim="2 5">5 25 10 20 15 15 10 10 5 15</Coords>
        </Polygon>
      </SubRegion>
    </Mask>
  </MediaOccurrence>
</Object>
<Object id="Leg1-object">
  ... ..
</Object>
<Object id="Leg4-object">
  ... ..
</Object>
... ..

```

Query:

In the image 'image.jpg', which objects are located on the left of object 'Leg4-object'.

MXQuery:

```

for $i in //Object[MediaOccurrence/MediaLocator/MediaUr = 'image.jpg']
let $a := //Object[@id = 'Leg4-object'][MediaOccurrence/MediaLocator/MediaUr = 'image.jpg']
where $i left $a
return $i/@id

```

Figure 6.4: An MXQuery example of spatial relationship

Spatial relationships The spatial relationship operator compares the spatial features of spatial objects and returns a boolean value. All spatial operators can be classified in two groups: directional relationships and topological relationships.

- *Directional relationships*: Directional relationships include *left*, *right*, *above*, *below*, *front*, *back*, *south*, *north*, *west*, *east*, *northwest*, *northeast*, *southwest* and *southeast*. The operators, *front* and *back* can be combined with other directional operators to express more precise directional relationships, such as *front_left*, *back_northwest*, etc. Precise definitions of directional relationships can be found in [65].

CHAPTER 6. MXQUERY

Figure 6.5: An MXQuery example of temporal relationship

- *Topological relationships*: Topological relationships include *inside*, *contains*, *covers*, *coveredBy*, *touch*, *overlap*, *disjoint* and *equals*. A detailed explanation of topological spatial relations is given in [66].

The example of MPEG-7 description shown in Figure 6.4 represents several objects within an image. A query about spatial relationship issued on this example is also shown in Figure 6.4.

Temporal relationships Temporal information tells a query system of changes to the media action along a time line (either absolute time or relative time). In [67], the authors described a set of thirteen temporal relationships between two events: *equal*, *before*, *after*, *meet*, *metBy*, *overlap*, *overlapedBy*, *during*, *include*, *start*, *startedBy*, *finish*

```

DECLARE FUNCTION mpeg7:qbe ($e as string, $d as string)
{
    let $example := doc($e),
        $imgDB := doc($d)
        $descriptor := $example//VisualDescriptor/@xsi:type
        $f1 := mpeg7:featureExtract($example//VisualDescriptor)
    for $imgs in $imgDB/Image[//VisualDescriptor/@xsi:type = $descriptor]
    for $a at $i in ( for $a1 in $imgs//VisualDescriptor,
                    $f2 in mpeg7:featureExtract(a1)
                    for $d in mpeg7:distance(f1, f2, descriptor)
                    order by $d
                    return $a1 )
    where $i = 1
    return $imgs/MediaLocator/MediaUri
}

```

Figure 6.6: Definition of QBE function

and *finishedBy*. These thirteen temporal relationships have been widely accepted, hence, they are also incorporated into MXQuery. An query on temporal relationship is shown in Figure 6.5 (The MPEG-7 description shown in this figure is extracted from [68]).

6.3.3 Specification for Query-by-example

In the multimedia applications, the query-by-example mechanism is often used in visual information retrieval. It allows the users to retrieve the most similar multimedia objects by giving an example. The query-by-example process is based on the low-level features extracted from the example and the corresponding similarity function. The definition of the query-by-example function in MXQuery for image retrieval is given in Figure 6.6. The input parameters of function QBE are the MPEG-7 description of the example image and the MPEG-7 description of the images in the image database.

In the function definition of QBE, the procedure `featureExtract()` is used to extract the feature vector from MPEG-7 descriptions. The procedure `distance()` is introduced to compute the distance between two feature vectors. The distance functions comply with

CHAPTER 6. MXQUERY

```

DECLARE FUNCTION mpeg7: featureExtract ($n as node())
{
  let $descriptor := $n/@xsi:type
  return if ($descriptor = 'ColorStructureType')
    then $n/Values
    else if ($descriptor = 'EdgeHistogramType')
    then $n/BinCounts
    else if ($descriptor = 'RegionShapeType')
    then $n/MagnitudeOfART
    else if ($descriptor = 'DominantColorType')
    then {let $s := $n/@size,
          $sc := $n/SpatialCoherency,
          $b := (for $p in $n/Values/Percentage,
                  $c in $n/Values/ColorValueIndex
                  return ($p,$c))
          return ($s, $sc, $b) }
    else
    ... ..
}

```

Figure 6.7: Definition of featureExtract function

the similarity functions described in [47]. Figure 6.7 shows the definition of procedure `featureExtract`. The definition of distance is not given here since some of distance functions are complex and additional program code is needed to implement them.

Chapter 7

Performance Study

With the novel XML storage schema and the design of MPEG-7 Content Database, IXMMS can fulfill the most requirements of the management of MPEG-7 descriptions. In this chapter, we describe the performance study on the Java-based prototype of IXMMS. The experimental results are presented in this chapter and initial results are encouraging.

7.1 Introduction

Our system was implemented on a Dell PowerEdge 2650 with a Xeon 2.8GHz CPU and 1.00GB of RAM, running Windows Server 2003 Enterprise Edition. The database system was IBM DB2 Universal Database Enterprise Server Edition V8.1. Each function module was developed by using JDK1.5. To evaluate the effectiveness of our MPEG-7 management system, we performed a series of experiments and compared our system to several existing XML storage solutions, which included DB2 XML Extender – an XML-Enabled database, XParent and SUCXENT++ – the pure schema-oblivious approaches, and Shared-Inlining – a pure schema-conscious approach. DB2 XML Extender provides two storage and access methods for integrating XML documents with DB2 data structures: XML column and XML collection. Since the storage schema of XML collection is much similar to the schema-conscious approach and a reputed schema-conscious method, Shared-Inlining, is used for comparison, we adopted the XML column method in our

CHAPTER 7. PERFORMANCE STUDY

performance studies. In the following sections, we firstly present the performance study on insertion, marshalling, deletion and update operations, and discuss the storage space requirements. Next, we compare the query performances of IXMMS to the above approaches. To test how IXMMS supports the queries from multimedia perspective efficiently, we performed a set of experiments to evaluate the efficiency of the multidimensional index system.

7.1.1 Dataset

We used two experimental data sets. One is a synthetic dataset, which is from the XMark project[69], a benchmark for XML data management. The other is a real dataset, which consists of MPEG-7 descriptions. Since MPEG-7 descriptions are also XML documents, in order to test the effectiveness of IXMMS from XML perspective, we used XMark benchmark data as one of experimental data sets. We generated the XMark benchmark data with different scale factors. Three different sizes of data were used: BENCH001 (which means 1% of the original BENCH) with 1.1MB size, BENCH01 with 11.3MB size and BENCH with 113MB size. The MPEG-7 descriptions data set with the size of 1GB includes four Description Schemes: UserDescription DS, Object DS (including SpatialMask D), StateTransitionModel DS and SoundClassificationModel DS, and eight low-level descriptors extracted from about 240,000 pictures: ColorLayout, ColorStructure, ContourShape, DominantColor, EdgeHistogram, HomogeneousTexture, RegionShape and ScalableColor. Table 7.1 summarizes the characteristics of the experimental data sets.

Test queries need to be carefully selected for the performance study. XMark issues 20 benchmark queries that cover different aspects of XML queries for accessing XML data. We also issued twelve common queries to test query performance on MPEG-7 documents. These queries and other test operations can be found in Appendix A. In our performance study, we labelled the XMark queries as Q1-Q20, and the MPEG-7 queries

CHAPTER 7. PERFORMANCE STUDY

Dataset	No of Internal Nodes	No of Leaf Nodes	No of Attributes	No of Nodes	Size (MB)	Max Depth
BENCH001	7802	9330	3919	21051	1.1	12
BENCH01	76749	91116	38265	206130	11.3	12
BENCH	762838	903477	381878	2048193	113.0	12
MPEG-7	5013953	10158454	3819292	18991699	1030.0	11

Table 7.1: Characteristics of the experimental data sets

as MQ1-MQ12. The corresponding SQLs with IXMMS for these queries are shown in Appendix C. In addition, the update performance was also tested. The corresponding test scripts were labelled as D1-D3 (for deletion performance) and U1-U3 (for updating performance). To evaluate the effectiveness of the multidimensional index system, we performed similarity search on four low-level descriptors with different dimensions.

7.1.2 Overview of performance evaluation

Table 7.2 summarizes how each method performed in the test operations. It also shows the performance statistic, including the minimum, maximum, average, and standard deviation of the performance for all the test operations. In Table 7.2(b,c), the ‘Ave’ is the arithmetic mean of the running time of all the approaches for each query. The ‘S.D.’ is the standard deviation, which is a measure of how widely values are dispersed from the average value. S.D. can be used to indicate how much query performance difference these six approaches would make. For example, for Q7, the S.D. is much larger than the mean. It indicates that these six approaches make enormous performance differences when performing Q7. However, for Q18, the S.D. is much smaller than the mean value. It means that there are only slight differences among the performance of these methods for Q18. According to Table 7.2(a), IXMMS not only converges the advantages of the schema-conscious approach and the schema-oblivious approach (having the largest number

Figure 7.1: Experimental Result: Insertion Performance

of operations with the best performance), but also avoids their intrinsic disadvantages (no operations with the worst performance). For the queries from XML perspective, IXMMS outperforms Shared-Inlining by up to 24 times, SUCXENT++ by up to 23 times, XParent by up to 40 times and DB2 XML Extender by up to 430 times for most testing queries. Thanks to the design of MCDB, IXMMS has encouraging performances for all the queries from the multimedia perspective (similarity search). It outperforms the other systems by up to 15 times on these queries. Exact performing times for each test operation can be found in Appendix B. We discuss the observations in detail as follows.

7.2 Insertion Performance and Storage Size

Figure 7.1 presents the insertion performance (including index creation during insertion process) for different sample data sets. This figure shows that XParent performs the worst and DB2 XML Extender is the best, while IXMMS, Shared-Inlining and SUCXENT++ have slight differences on insertion performance. This is because these methods store different amounts of data during the insertion process. Note that, to make the comparison

CHAPTER 7. PERFORMANCE STUDY

	Update		Query	
	Best	Worst	Best	Worst
IXMMS	4 of 6	0 of 6	24 of 36	0 of 36
Inlining	0 of 6	0 of 6	5 of 36	3 of 36
SUCXENT++	1 of 6	0 of 6	2 of 36	1 of 36
XParent	0 of 6	2 of 6	0 of 36	22 of 36
DB2 XML Ext.	1 of 6	4 of 6	5 of 36	10 of 36

(a) Performance Summary

Operations	Min	Max	Ave.	S.D.
Q1	102ms	11156ms	2253ms	4374ms
Q2	515ms	9953ms	2842ms	3687ms
Q3	10254ms	16523ms	12441ms	3537ms
Q4	160ms	625ms	418ms	236ms
Q5	78ms	8860ms	2085ms	3480ms
Q6	91ms	9031ms	2051ms	3453ms
Q7	63ms	27359ms	5298ms	10822ms
Q8	656ms	1147ms	886ms	246ms
Q9	2037ms	4150ms	3406ms	1187ms
Q10	2775ms	50437ms	19111ms	27137ms
Q11	32703ms	61721ms	51418ms	16235ms
Q12	14563ms	31253ms	24280ms	8676ms
Q13	2181ms	6234ms	3659ms	2238ms
Q14	6105ms	9136ms	7164ms	1709ms
Q15	110ms	9344ms	1968ms	3644ms
Q16	1116ms	12946ms	5540ms	6453ms
Q17	343ms	1932ms	945ms	861ms
Q18	188ms	271ms	221ms	43ms
Q19	750ms	18669ms	8663ms	8313ms
Q20	172ms	403ms	315ms	125ms

(b) Query performance statistics
for 5 testing methods on XMark

Operations	Min	Max	Ave.	S.D.
D1	136ms	4421ms	2293ms	1577ms
D2	460ms	2154ms	1250ms	703ms
D3	3172ms	4850ms	3847ms	721ms
U1	362ms	1965ms	827ms	651ms
U2	1105ms	13609ms	4582ms	5124ms
U3	3644ms	11719ms	5902ms	3364ms
MQ1	65ms	1915ms	674ms	717ms
MQ2	42ms	4466ms	1982ms	2148ms
MQ3	62ms	18313ms	6797ms	8666ms
MQ4	158ms	19891ms	7366ms	9234ms
MQ5	3012ms	5578ms	3865ms	1096ms
MQ6	906ms	9872ms	3964ms	4050ms
MQ7	719ms	1521ms	1182ms	294ms
MQ8	245ms	909ms	551ms	280ms
MQ9	547ms	2034ms	1230ms	623ms
MQ10	760ms	5736ms	2658ms	2194ms
MQ11	210ms	690ms	459ms	202ms
MQ12	378ms	3021ms	1535ms	1167ms
MQ13	1.64s	26.13s	15.00s	10.14s
MQ14	3.45s	30.38s	18.30s	9.60s
MQ15	16.58s	45.73s	33.81s	10.58s
MQ16	33.25s	72.08s	47.28s	14.57s

(c) Performance Statistics
for 5 testing methods on MPEG-7

Table 7.2: Performance summary and statistics

CHAPTER 7. PERFORMANCE STUDY

fair, the running time for IXMMS does not include the MDDB- $\rightarrow$ MCDB synchronization time, since such a synchronization operation is not involved in the other approaches.

With **DB2 XML Extender**, the intact MPEG-7 documents are inserted into columns directly. This is unlike the methods employed by the other four, which first parse MPEG-7 documents, and then insert individual parsed elements one by one. This is why **DB2 XML Extender** performs the best. The difference in performance among the other four methods is explained by the different number of tuples inserted into the database with their respective methods.

When mapping an XML file, **XParent** needs to insert each node as one tuple into the database. That means the number of inserted tuples with **XParent** is equal to the total number of nodes. Furthermore, **XParent** stores all the ancestors of each node into the **Ancestor** table. There are more than 9 million records inserted into this table for **BENCH** dataset to represent such ancestor-descendant relationship. **SUCXENT++** only stores leaf nodes, thus the number of its inserted tuples is equal to the number of leaf nodes. **Shared-Inlining** method creates tables for the internal nodes, and their leaf node children, and some of leaf node descendants may be inlined into these tables. Therefore, the number of inserted tuples with **Shared-Inlining** is close to the number of internal nodes. For **IXMMS**, all internal nodes need to be inserted; and for each internal node that has attributes or leaf nodes, one tuple would be inserted to store its leaf node children. Therefore, the number of inserted tuples with **IXMMS** is close to double of the number of internal nodes. Compared to **XParent** and **SUCXENT++**, **Shared-Inlining** and **IXMMS** have a disadvantage for insertion process. That is, they need to insert data into more tables than **XParent** and **SUCXENT++** do. However, such disadvantages can be compensated by the smaller size of inserted tuples and less elapsed time for index creation with **Shared-Inlining** and **IXMMS**. Since **XParent** and **SUCXENT++** store the data in several fixed tables, inserting the value of each leaf node would require an additional storage process, i.e., storing additional information into the columns

other than the column that records the value of leaf node in the table. While for **Shared-Inlining** and **IXMMS**, the value of each leaf node is stored in a corresponding column and does not raise an additional storage requirement. Furthermore, **XParent** and **SUCXENT++** need to create the index on all the values with character string type, while **IXMMS** and **Shared-Inlining** generate the indexes on the required columns with different datatypes. Thus, **IXMMS** and **Shared-Inlining** take less time to create indexes. In summary, **IXMMS**, **Shared-Inlining** and **SUCXENT++** have slight differences in insertion performance due to their intrinsic advantage and disadvantage in insertion process, while **XParent** performs the worst since it has to insert much more tuples.

Table 7.3 shows their different storage requirements. One of the common drawbacks of XML is verbose due to repeated information. Thus, although **DB2 XML Extender**, which stores the intact XML documents into database, has the best insertion performance, it does not have the advantage in the requirement of storage space. The above discussions can also explain the different storage requirements for the other four methods.

7.3 Marshalling Performance

Figure 7.2 shows the marshalling performance of these approaches. Marshalling is the reverse operation of insertion process. That means marshalling operation is the process to extract the data from database and reconstruct them with original XML format.

Since **DB2 XML Extender** stores the intact XML documents in the database, it undoubtedly performs the marshalling process the most efficiently. For the other four approaches, the marshalling time is made up of the time taken to extract the relevant data from database and main memory processing time to reconstruct the data into XML document. Table 7.4 also shows the running time of these methods in terms of the two steps of marshalling process. Based on these figure and table, we observe the following:

CHAPTER 7. PERFORMANCE STUDY

Approach	Dataset	Inserted Tuples	Table Size (MB)	Index Size (MB)	Total Size (MB)
IXMMS	BENCH001	18498	1.6	0.6	2.2
	BENCH01	183326	15.4	6.6	22
	BENCH	1814296	149.2	60.8	210
	MPEG-7	9411958	910.3	420.5	1330.8
SUCXENT++	BENCH001	13946	1.3	0.8	2.1
	BENCH01	136262	12.4	9.1	21.5
	BENCH	1339355	120.5	113.3	233.8
	MPEG-7	14409746	1126.9	814.8	1941.7
XParent	BENCH001	116129	3.2	4.2	7.4
	BENCH01	1143854	31.2	48.9	80.1
	BENCH	11201769	305.7	370.2	675.9
	MPEG-7	78502888	1909.7	2265.3	4175
Shared-Inlining	BENCH001	7971	0.6	0.3	0.9
	BENCH01	78126	5.8	3.6	9.4
	BENCH	772838	56.8	32.5	89.3
	MPEG-7	5089703	660.7	129.5	790.2
DB2 XML Extender	BENCH001	1	2.8	0	2.8
	BENCH01	1	16.8	0	16.8
	BENCH	1	127.9	0	127.9
	MPEG-7	120	1679.5	0	1679.5

Table 7.3: Storage Size

Figure 7.2: Experimental Result: Marshalling Performance

	BENCH001		BENCH01		BENCH	
	Extraction	Construction	Extraction	Construction	Extraction	Construction
IXMMS	0.6	0.4	5.8	2.8	141	55
Inlining	1.5	0.6	10.5	3.1	240	66
SUCXENT++	0.4	1.1	2.6	6.8	70	146
XParent	0.3	1.5	2.8	7.7	82	149

Table 7.4: Running time (s) in terms of two steps of marshalling process

there are slight differences among IXMMS, XParent and SUCXENT++, and they are about 50% faster than Shared-Inlining in terms of marshalling performance.

In the process of extracting relevant data from database, the performances of SUCXENT++ and XParent are better than IXMMS, since they only retrieve the data from a few tables. IXMMS needs to retrieve all internal nodes and corresponding leaf node children from more tables. Note that it is not necessary for IXMMS to join the internalnode table and each leaf node table to get the leaf nodes data, since with the aid of MPD file and the given document identifier, it is easy to extract all the leaf nodes value from database without any internal nodes information. Thus performance of SUCXENT++ is slightly bet-

CHAPTER 7. PERFORMANCE STUDY

ter than XParent, since SUCXENT++ only extracts the data of leaf nodes, while XParent needs to extract the data of all the nodes.

However, for the process of reconstructing the extracted data into XML format, SUCXENT++ and XParent perform worse than IXMMS due to their storage schemas. SUCXENT++ only stores the path expressions of leaf nodes and there is a lack of the information about each internal node. When SUCXENT++ creates the XML document tree, it needs to first parse the path expression of each leaf node, gain the internal nodes within this path expression, and then decide the appropriate positions of these internal nodes and leaf nodes in the document tree. Therefore, with SUCXENT++, although the performance of extracting data from database is better, the time taken for reconstruction is more. Although XParent stores the position information of all the internal nodes, it needs to process all the nodes one by one. IXMMS consumes the least processing time to reconstruct XML document because it creates the document tree by only organizing the internal nodes. With the aid of an MPD file, the corresponding leaf nodes of each internal node can be efficiently placed to the appropriate position in the document tree.

Although Shared-Inlining returns the smallest number of tuples among these approaches when extracting data from database, the performance is worse than the others because Shared-Inlining uses primary key and foreign key to represent the parent-child relationship between two nodes. Such relationship information needs to be extracted to generate XML document tree. This means a large amount of join queries need to be executed to extract data. The marshalling process would become more expensive when there exist many relations for storing XML documents.

7.4 Deletion and Update Performance

Figure 7.3 shows the deletion and update performance on a 1GB MPEG-7 dataset. According to Figure 7.3, none of these five methods has a dominant performance on all

Figure 7.3: Deletion and Update Performance

the deletion operations. For D1, which deletes an entire MPEG-7 document, DB2 XML Extender outperforms the others. This is because DB2 XML Extender needs to delete only a single row containing the MPEG-7 document from an XML column. However, DB2 XML Extender cannot perform D2 and D3, which need to delete a fragment of the content in the MPEG-7 documents, since it does not provide the corresponding UDFs. The main drawback of our method and Shared-Inlining is the requirement to access several tables to delete the required information. This drawback is the main reason for the poor performance on D2 with our method and Shared-Inlining. In the case of SUCXENT++ and XParent, they need to access larger tables when large collections of MPEG-7 documents are stored in the database. This drawback gets magnified when testing is done with a large number of ancestor-descendant relationships during the deletion process.

To perform the update process, DB2 XML Extender first parses the MPEG-7 document within the XML column, changes the required parts of the original document based on the output from the XML parser, and finally replaces the changed MPEG-7 document in the XML column. Therefore, its performance degrades dramatically when the size of the MPEG-7 document within the XML column is large, e.g., for U2 and U3. As

for the other four methods, when updating a specified element value (U1 and U2), the performance mainly depends on the process of locating the destination element, which is similar to the query process. The resultant performance discussion is shown in the following section. To replace a fragment of data (U3), two steps are performed. The first step is to delete old data, and the second step is to insert new data. Therefore, the replacement performance depends on the deletion performance and the insertion performance.

7.5 Query Performance

MPEG-7 descriptions are XML documents storing multimedia information. The queries on MPEG-7 descriptions include retrieval from the XML perspective and multimedia content retrieval. To evaluate the query performance for our system, we used benchmark queries from XMark project and issued sixteen queries on the real MPEG-7 descriptions dataset. These queries are listed in Appendix A. The queries from Q1 to Q20 and MQ1 to MQ12 are from the XML perspective and the queries from MQ13 to MQ16 are similarity search. The corresponding SQLs with IXMMS for some of these queries are shown in Appendix C.

7.5.1 Queries from the XML perspective

Figure 7.4 represents the query performance of each approach for the queries from XML perspective. This figure shows that IXMMS has encouraging query performance for most of the queries, and justifies the main contribution of our novel XML storage schema, which is to integrate the advantages of the pure schema-conscious approach and the pure schema-oblivious approach, and avoid their intrinsic disadvantages. We discuss the observations in detail as follows.

CHAPTER 7. PERFORMANCE STUDY

Figure 7.4: Experimental Result: Query Performance from the XML perspective

7.5.1.1 IXMMS vs. pure schema-oblivious approach

The common feature of the test queries is the application of predicates related to several sub-elements. In general, for such queries, the **schema-conscious** method outperforms the **schema-oblivious** method because of the reasons described in [70]. The **schema-conscious** method clusters elements corresponding to the same real world object whereas the **schema-oblivious** method does not reap this benefit and has to issue more SQL joins to capture the parent-children or ancestor-descendant relationships between XML elements. This results in much worse performance for the **schema-oblivious** method. Consider the following XQuery:

```
//Descriptor/Values[Percentage>5]/ColorValueIndex
```

With the **schema-conscious** approach, elements **Percentage** and **ColorValueIndex**, which are children of element **Values**, would be clustered as two attributes of one relation. The above XQuery can be translated into a simple SQL:

```
SELECT colorvalueindex FROM values WHERE percentage > 5
```

where table **values** corresponds to the element **Values** and columns **percentage** and **colorvalueindex** correspond to the elements **Percentage** and **ColorValueIndex** respectively. While with the **schema-oblivious** approach, these two elements cannot be clustered and would be stored in separate rows. The **schema-oblivious** approach needs more joins to check sibling relationship between elements **Percentage** and **ColorValueIndex**.

Since our storage schema integrates the advantages of the **schema-conscious** approach and the **schema-oblivious** approach, it can benefit from the characteristic mentioned above. Figure 7.5 shows the translated SQL with SUCXENT++ for the XQuery shown in Figure 4.9. According to its storage schema, IXMMS can benefit from the above advantage of **schema-conscious** approach. Thus, SUCXENT++ needs more joins to check sibling relationships in comparison with the translated SQL with IXMMS shown in Figure 4.9.

CHAPTER 7. PERFORMANCE STUDY

```
SELECT      v3.LeafValue
FROM        Path p1, Path p2, Path p3,
            PathValue v1, PathValue v2, PathValue v3,
            DocumentRValue r1, DocumentRValue r2
WHERE       p1.PathExp = '/Mpeg7/DescriptionUnit/Descriptor/Values/ColorValueIndex'
            and p2.PathExp = '/Mpeg7/DescriptionUnit/Descriptor/Values/Percentage'
            and p3.PathExp = '/Mpeg7/DescriptionUnit/Descriptor/SpatialCoherency'
            and v1.PathId = p1.PathId
            and v2.PathId = p2.PathId
            and v3.PathId = p3.PathId
            and v1.LeafValue = '44 67 30'
            and cast(v2.LeafValue as integer) > 5
            and r1.Level = 4
            and abs(v1.BranchOrderSum - v2.BranchOrderSum) < r1.RValue
            and r2.Level = 3
            and abs(v1.BranchOrderSum - v3.BranchOrderSum) < r2.RValue
```

Figure 7.5: Translated SQL with SUCXENT++

The schema-conscious approach and IXMMS store the leaf nodes value in many different tables, while schema-oblivious approach, e.g. SUCXENT++, stores all leaf nodes value within a single table. Therefore, schema-conscious approach and IXMMS require the join of many smaller tables when performing queries, whereas schema-oblivious approach needs to self-join a single large table. The performance of schema-oblivious approach is substantially degraded when large collections of XML documents exist. Figure 7.6 shows the access plan for MQ3 with IXMMS and SUCXENT++ respectively. According to this figure, we can observe that SUCXENT++ requires more joins and performs this query at a much higher cost.

The above analysis explains why IXMMS outperforms SUCXENT++ and XParent for queries Q1, Q8-10, Q14, Q17, MQ1-3, etc. For example, to perform MQ3, SUCXENT++ and XParent need to test the ancestor-descendance relationship among four elements. With SUCXENT++ and XParent, all the values of the test dataset are stored in only one table. That means much larger joins would occur when issuing this query with SUCXENT++ or XParent. IXMMS, however, needs to only join a few tables of a much smaller size.

Furthermore, the schema-oblivious approach cannot provide the typed representation and access of the content within the XML documents. With the schema-oblivious ap-

CHAPTER 7. PERFORMANCE STUDY

Figure 7.6: Access plan for MQ3

proach, the data within the XML documents will be only stored as character string datatype and the corresponding index system is only created on string datatype. IXMMS and the schema-conscious approach, however, can benefit from the efficient index mechanism created on all kinds of datatypes. For some test queries, SUCXENT++ and XParent has to spend time to convert datatype, for example, Q5, Q11, Q12 and Q18, which include numeric comparison.

Because the required data is stored in several tables and the extraction process involves several joins with IXMMS; for reconstructing a fragment of original XML document, e.g., Q13, MQ5 and MQ7, IXMMS performs worse than SUCXENT++. According to Figure 7.2, the extraction performance of IXMMS is slightly better than SUCXENT++. However, for IXMMS, the process of reconstructing a fragment of document is different

CHAPTER 7. PERFORMANCE STUDY

from the process of reconstructing the whole document. During the former process, several joins between the leaf node tables and the internal node tables are involved when extracting the required data, while for reconstructing the whole document, these joins are not necessary, since with the document identifier and the table information kept in MPD file, we can extract the required data from leaf node tables directly.

Compared to the other schema-oblivious approaches, **SUCXENT++** only stores the leaf nodes information and the internal nodes information has been eliminated. This gives rise to a drawback that it is inefficient to implement the queries with conditions on internal nodes, for example, Q2-4, MQ5, MQ7 and MQ8, which are ordered access queries, and Q6 and Q7, which are to count the occurrence of given internal nodes. To perform these queries, **SUCXENT++** needs the assistance of an additional programming code.

According to Figure 7.4, **XParent** performs the worst in respect to the most of test queries. This is because **XParent** stores all parents and ancestors of each node in **DataPath** and **Ancestor** tables and it results in an explosion in the database size. **XParent** requires joins between the **DataPath**, **Element** and **Ancestor** tables to determine the relationships between nodes. Such joins may be quite expensive due to the large size of **Ancestor** table.

7.5.1.2 IXMMS vs. pure schema-conscious approach

The main drawback of the schema-conscious approach, like **Shared-Inlining**, is the lack of path expression information and path index. For the recursive queries (e.g. Q6, Q7, Q14 and Q19) and the queries including longer path expressions (e.g. Q15 and Q16), the **Shared-Inlining** method performs worse than **IXMMS** and the schema-oblivious approach. **Shared-Inlining** only keeps the parent-children relationships by defining a set of primary keys and foreign keys. **IXMMS** does not suffer from such limitations since it preserves all the parent-children and the ancestor-descendant relationships and the path expressions of the original XML documents. **IXMMS** only requests two θ -joins

CHAPTER 7. PERFORMANCE STUDY

Figure 7.7: Access plan for Q15

to check the ancestor-descendent relationships, while Shared-Inlining may perform large numbers of equijoins to check ancestor-descendent relationships or travel from root node to destination node along the path expression to access desired data. The number of equijoins depends on the depth of the path expression. Therefore, the recursive queries in which the exact depth is unknown (e.g. /Mpeg7//UserActionHistory) and the path expression with large depth affect the query performance of Shared-Inlining. Figure 7.7 shows the access plan for Q15 with IXMMS and Shared-Inlining respectively, and illustrates the great impact of the long path expression on the performance of Shared-Inlining.

7.5.1.3 IXMMS vs. DB2 XML Extender

With DB2 XML Extender, XPath-based queries are evaluated by constructing DOM from CLOB and using functional evaluations. Thus, the query performance depends on the size of the retrieved documents, and this can be very expensive when performing queries

CHAPTER 7. PERFORMANCE STUDY

on large collections of MPEG-7 documents. For example, `DB2 XML Extender` performs the best on MQ1-4 because of the small size of the retrieved MPEG-7 documents, while for the dataset `BENCH` or query MQ5, its performance degrades substantially because of the large size of the queried documents. Furthermore, since `DB2 XML Extender` provides limited support for XPath-based query and does not provide typed representation and access for the MPEG-7 documents storage, it does not support many complex queries, e.g. Q3-4, Q8-14, Q16-18, MQ6 and MQ8-12.

7.5.1.4 Queries on complex datatypes in MPEG-7 descriptions

MQ4 and MQ8-12 are the queries for testing the performance on complex datatypes defined in MPEG-7 DDL. None of the existing XML storage solutions provide the schema to handle the complex datatypes in MPEG-7 descriptions. These datatypes are only stored as character string type in the database. It makes manipulation on these datatypes very inefficient.

MQ4 is to test the performance of date arithmetic operation on `basicTimePoint` and `basicDuration` types. As these types are stored as character string, the existing XML solutions need additional program code to implement date arithmetic operations. While IXMMS only requests a single SQL to implement this query.

Queries MQ8-12 include the operations on array and matrix data. The detailed discussion can be found in the following subsection.

7.5.2 Performance on array and matrix data

To evaluate further the efficiency of our storage schema for array and matrix data, we generated additional MPEG-7 description data sets with the size of 1MB, 10MB and 100MB, which include `DominantColor D`, `Object DS`, `StateTransitionModel DS` and `Sound-ClassificationModel DS`. `DominantColor D` has the array data with 3 dimensions and we used semi-normalized schema to store them. `StateTransitionModel DS` has 3×3 matrices,

CHAPTER 7. PERFORMANCE STUDY

Dataset	Num of arrays	Num of matrices	Num of tuples in array table	Size of array table (MB)	Num of tuples in matrix table	Size of matrix table (MB)
1M	4048	58	4048	0.17	19816	0.65
10M	44431	592	44431	1.85	118773	3.92
100M	440965	6069	440965	18.32	1189151	39.04

Table 7.5: Features of array/matrix storage

while the matrices in Object DS and SoundClassificationModel DS have the dimensions of 2×4 , 2×5 , 20×20 , 31×20 , etc. We used normalized schema to store these matrices. Table 7.5 shows the characteristic of these three data sets and the table sizes of IXMMS for storing these arrays and matrices. We used MQ8-12 for performance testing and Figure 7.8 shows the experimental results.

Queries MQ8 and MQ9 include the operations on the individual item within the array data. For example, for the DominantColor descriptor, if the colour space is HMMD, the value of ColorValueIndex, which is defined as array type with 3 length, would be a set of three components: Hue, Diff and Sum. As shown in MQ9, if the users want to find the pictures with blue dominant colour, the following condition will be issued: “the value of Hue should be from 160 to 210” (this value range specifies the blue colour). According to the storage schema for the array or matrix data designed with IXMMS, IXMMS can benefit from the index on the individual items of the array data and speed up these queries. However, the other methods, which store the array data as character string, need to parse each candidate array string, get the individual item within the array string, convert the datatype, and then test whether it accords with the query conditions. Thus, IXMMS outperforms them undoubtedly.

Queries MQ10-12 are the queries for testing the performance on the matrix datatype. They are issued against Object DS, StateTransitionModel DS and SoundClassificationModel DS respectively. Same as the performance on array data, XParent, SUCXENT++ and

CHAPTER 7. PERFORMANCE STUDY

Figure 7.8: Experimental Result: Query Performance on Array/Matrix

Shared-Inlining underperform IXMMS due to the additional operations on character string parsing, datatype conversion and without index on the individual items of the matrices. Furthermore, to parse the character string, XParent, SUCXENT++ and Shared-Inlining need the value of the attribute dim, which records the dimension information of the matrix. This results in one more join for XParent and SUCXENT++ to get the value of dim that corresponds to the matrix in question. As discussed in the previous subsection, XParent costs more time to extract desired array/matrix data from

Figure 7.9: Experimental Result: Query Performance from the multimedia perspective

database than SUCXENT++. That is why SUCXENT++ performs better than XParent on the array/matrix operations.

According to the Figure 7.8, we observed that for the small data set, the advantage of IXMMS is slight. In fact, our array/matrix storage schema has a disadvantage. It needs more joins (joins between leaf node table and array/matrix table) to get required array/matrix data. Compared to the other methods without the indices on the individual items of array/matrix data, however, with large collections of array/matrix data, the impact of index mechanism in IXMMS results in the significant efficient performance.

7.5.3 Queries from the multimedia perspective

As mentioned in the previous chapter, the GiST framework is connected with our MPEG-7 storage system via a set of UDFs. This enables our system to support efficient multimedia-related queries. To evaluate the performance from the multimedia perspective, we performed similarity search on four low-level MPEG-7 descriptors extracted from 300,000 pictures. The tested descriptors included ColorLayout(CLD) with 12 dimensions,

CHAPTER 7. PERFORMANCE STUDY

RegionShape(RSD) with 35 dimensions, EdgeHistogram(EHD) with 80 dimensions, and ColorStructure(CSD) with 128 dimensions. The similarity search in our experiment consisted of finding the top 10 objects that are most similar to the given object based on each descriptor. Figure 7.9 presents the corresponding experimental results. In this experiment, we adopted M-Tree in IXMMS, since M-Tree is based on the distance function.

As shown in Figure 7.9, IXMMS outperforms the existing XML storage systems on similarity search. There are two main facts that affect the similarity searching performance for these systems. The first is the extraction process. These systems do not provide a special storage mechanism for storing multimedia content. They need to extract the desired multimedia content from their MPEG-7 descriptions repository before performing the similarity search. The performance of this extraction process depends on the complexity of the related multimedia content and their storage schema. For the ColorLayout descriptor (MQ13), the related multimedia content is stored in several elements, e.g., YDCCoeff, CbDCCoeff, YACCCoeff5, etc. SUCXENT++ and XParent need several joins to extract related information and this operation is expensive for them. On the other hand Shared-Inlining can carry out the extraction process efficiently since it clusters the related content into one relation and performs this extraction operation without any joins. However, SUCXENT++ and XParent have efficient extraction performances on MQ14, MQ15 and MQ16, since the related multimedia content for these queries is stored in only one element. The extraction process for DB2 XML Extender is always expensive because of the costly operations required to build in-memory trees of very large MPEG-7 documents and then to access those trees. The second fact is the similarity search process. Without an efficient high-dimensional index mechanism, these systems can only perform a sequence scan to achieve similarity matching. However, IXMMS introduces MCDB, which re-organizes the multimedia content according to the similarity matching function and provides a high-dimensional index mechanism. Therefore, it has a better performance on similarity search than the existing XML storage systems.

Figure 7.10: Performance on different high dimensional index structures

Since it is difficult to assess what index structures best fit the needs of different MPEG-7 applications [10], IXMMS provides several multidimensional index structures, including R-tree, R*-tree, SS-tree, SR-tree and M-tree. It is certain that they have different performances for queries on different descriptors. We also evaluated the performance of these index structures on the real MPEG-7 descriptors dataset. Figure 7.10 shows the experimental results. The end users can choose an appropriate index to fit the needs of their MPEG-7 applications.

Chapter 8

Application

Our research work is part of the PET-DEVICE++ project. This project focuses on multimedia indexing storage and retrieval based on MPEG-7 standard. To implement this project, one of the issues is to investigate efficient and effective methods for storage and indexing of MPEG-7 Descriptors and Description Schema data in the Descriptors Database (DDB). IXMMS can provide such a solution, and serve as a multimedia data repository for some other applications developed for the PET-DEVICE++ project. In this chapter, an application scenario that could benefit from IXMMS will be summarized.

8.1 Introduction to PET-DEVICE++

The motivation of PET-DEVICE++[71] project is to implement a distributed multimedia retrieval system with an acceptable level of content interpretation, which is based on the MPEG-7 standard and can offer users a meaningful result set consisting of multiple media types. PET means “Push-pull Extraction Tool”. Retrieval from digital archives is known as a type of Pull application, whereas filtering from audiovisual broadcasts or multimedia databases are known as Push applications. DEVICE is the abbreviation of “DistributEd audioVIsual Content tErMinal”. It can be viewed as an MPEG-7 compliance database that is provided as a Web Service. The enhanced DEVICE (DEVICE++) will consist of a MediaCreateStore plus the DEVICE. MediaCreateStore focuses on the implementation

CHAPTER 8. APPLICATION

Figure 8.1: PET-DEVICE++ System Architecture

of a frame and object-based multimedia creation and storage environment to provide an encoding, recording, storage and retrieval environment for multimedia data streams.

One of the main issues to implement PET-DEVICE++ is to investigate efficient and effective methods for storage and indexing of MPEG-7 Descriptors and Description Schema data. Our research work focuses on developing such an MPEG-7 description database. Figure 8.1 shows the PET-DEVICE++ system architecture.

8.2 A Content Based Image Retrieval Tool

Content-based image retrieval based on the MPEG-7 standard is an important research issue of the PET-DEVICE++ project. Content Based Image Retrieval Tool is a demo for image retrieval developed by the members of PET-DEVICE++ project. With this tool, the users can search for similar images from our image database that stores more than 300,000 images.

Given a query image, if this image is stored in the image database, the retrieval process will be invoked directly. If the query image does not exist in the image database, it is firstly analyzed and the corresponding descriptor is extracted from this image. Secondly, the descriptor is parsed and the data for similarity match is extracted from the descriptor. Then, the retrieval process is invoked and the query data and conditions are passed to IXMMS. Finally, the similarity searching is performed in IXMMS and the retrieval results will be returned. Figure 8.2 shows the UML sequence diagram of this image retrieval tool.

Figure 8.3 shows a snapshot of this demo. In the browsing page, the function view is given on the left. From the left panel, the users can upload the query image, decide which descriptors can be used for similarity search, and select desired multidimensional index methods. The right panel displays the images randomly selected from the image database. At the bottom of the right panel, there is a button ‘Continue Browsing’, which starts a new random image selection process.

If a user clicks one of the images shown in the right panel, the similarity search will be triggered. The process of similarity search is based on IXMMS and can benefit from the multidimensional index mechanism provided with IXMMS. For example, if the user selects the ‘ColorLayout’ descriptor and ‘M-Tree’ index method, and then clicks the first image shown in Figure 8.3, the similar images stored in the database will be returned within 1 minute, as shown in Figure 8.4. According to this figure, we can find the

CHAPTER 8. APPLICATION

Figure 8.2: UML sequence diagram of Content Based Image Retrieval Tool

CHAPTER 8. APPLICATION

Figure 8.3: A snapshot of Content Based Image Retrieval Tool

last image in the result set is an error image. Since the different descriptors emphasize image attributes in different domains, the similarity search with different descriptors will achieve different accuracy. For example, if the ‘EdgeHistogram’ descriptor is used for the similarity search of the same query image, the result set will include much more error images, as shown in Figure 8.5. To get more accurate retrieval results, the similarity search could be performed not only on one descriptor, but also on the combination of multiple descriptors. This topic is beyond the scope of this dissertation.

You can also upload an image as query image, as shown in Figure 8.6, and then launch the similarity search to find its similar images in the database. Figure 8.7 shows the corresponding retrieval results for the image in Figure 8.6.

CHAPTER 8. APPLICATION

Figure 8.4: Retrieval result when using ColorLayout and M-Tree

CHAPTER 8. APPLICATION

Figure 8.5: Retrieval result when using EdgeHistogram and M-Tree

Figure 8.6: Upload a query image

CHAPTER 8. APPLICATION

Figure 8.7: Retrieval result for the uploaded image

Chapter 9

Conclusions and Future Work

More and more multimedia applications are based on MPEG-7 descriptions and the number of MPEG-7 descriptions is increasing by the day. As a result, there is an emerging need for an adequate data management system that provides an efficient mechanism to store, index and query arbitrary MPEG-7 media descriptions. The goal of this thesis is to explore the application of relational database solutions for the management of MPEG-7 descriptions.

In this chapter, we first summarize our research work on the management of MPEG-7 descriptions. We then present the conclusions of our performance study, and highlight the summarized contributions made in the thesis. Finally an overview of the possible future directions of this research work is presented based on our findings.

9.1 Conclusions

In the beginning of this dissertation, we present the critical requirements for the management of MPEG-7 media descriptions according to the discussion in [10]. These requirements include the representation of and the access to MPEG-7 descriptions with fine-grained and typed manner, the support to the complex datatypes defined in the MPEG-7 DDL, an extensible index mechanism for one-dimensional and multidimensional data, path index for the efficient navigation of MPEG-7 documents, and an extensible

CHAPTER 9. CONCLUSIONS AND FUTURE WORK

storage schema for an arbitrary MPEG-7 document that may comply with a new MPEG-7 description schema.

Against these requirements, we reviewed a number of representative XML database solutions, including native XML database solutions, XML-Enabled database and RDBMS-based XML storage solutions that are classified into two groups: **schema-conscious** approach and **schema-oblivious** approach. Although many of these solutions have great capabilities in the management of XML documents, all of them have different degrees of deficiencies for the management of MPEG-7 media descriptions. Most of the native XML database solutions largely neglect type information and lack the extensibility for organizing and indexing multimedia content available with MPEG-7 description schemas. To a certain extent, XML-Enabled database could be viewed as the combination of native XML database and traditional DBMS. The XML extensions of traditional DBMSs retain the disadvantages of the native XML database in terms of the management of MPEG-7 descriptions, even though they can benefit from the powerful index mechanism provided with the leading DBMSs. The relational database can provide robust solutions for the storage of data-centric XML documents. The existing RDBMS-based XML storage solutions, however, are still not adequate for the management of MPEG-7 descriptions. The **schema-conscious** approach has deficiencies in the document navigation because it neglects the complete hierarchical structure information of the original XML documents. The weakness of the **schema-oblivious** approach is the lack of capability of typed representation of and access to the MPEG-7 descriptions.

Facing the deficiencies of existing XML storage solutions in terms of the management of MPEG-7 descriptions, this dissertation proposes an MPEG-7 description management system, IXMMS, which is based on RDBMS and ORDBMS. IXMMS comprises several function modules: *Communication*, *Management Activities*, *Retrieval Activities* and *Back-end Database*. To enable IXMMS to be as flexible as the highly fluid multimedia

CHAPTER 9. CONCLUSIONS AND FUTURE WORK

applications, the *Communication* module is built with the Web services architecture. The *Management Activities* and *Retrieval Activities* provide the functionalities to deliver the fundamental views of MPEG-7 descriptions between the end users and the back-end MPEG-7 database. The *back-end Database* module is constructed based on two separate database systems: MPEG-7 Descriptions DB (MDDB) and Multimedia Content DB (MCDB). MDDB is based on RDBMS and designed for storing all the contents within the MPEG-7 descriptions, while MCDB is based on ORDBMS and designed for managing and indexing multimedia content stored in the MPEG-7 low-level descriptors.

In MDDB, a novel RDBMS-based XML storage solution, SM3, is proposed to store MPEG-7 descriptions. SM3 integrates the advantages of **schema-conscious** approach and **schema-oblivious** approach. Unlike the **schema-conscious** method, SM3 supports XPath-based query efficiently without involving many joins in SQL. Compared with the **schema-oblivious** method, SM3 solves the datatype problem in **schema-oblivious** approach without sacrificing the performance, and SM3 even performs better than most **schema-oblivious** approaches in the case of many XPath based queries. Furthermore, SM3 provides a flexible storage schema to satisfy all kinds of storage requirements, especially for the special datatypes within MPEG-7 descriptions, such as **array**, **matrix**, **basicTimePoint** and **basicDuration**. Although SM3 cannot avoid the assistance of MPEG-7 scheme, it can support arbitrary MPEG-7 description storage. SM3 supports the most critical requirements for the MPEG-7 descriptions management, such as fine-grained and typed representation and access, index system and XPath-based query.

MCDB is created specially for the low-level multimedia content extracted from MDDB. To support the queries from the multimedia perspective, an extensible multidimensional index system is introduced in the MCDB. A synchronization mechanism is designed to allow the data in MCDB to be synchronized with the corresponding fields in MDDB.

CHAPTER 9. CONCLUSIONS AND FUTURE WORK

Query language is a necessary issue of data management system. Although XQuery is a powerful and convenient language for querying XML data, it still has the disadvantage of specifying many special queries of MPEG-7 descriptions. In this dissertation, we discuss the special issues on MPEG-7 description query. According to these issues, we propose MXQuery, an extension of XQuery, to support the queries of MPEG-7 descriptions. MXQuery can support the queries on complex datatypes within MPEG-7 documents, such as array and matrix, the queries on spatial-temporal relationships implicitly represented in MPEG-7 descriptions, and the formulation of QBE queries.

To evaluate the performance of IXMMS, we carried out a set of experiments made up of various operations, such as insertion, marshalling, updating and querying. The experimental results can testify to the main advantage of our novel XML storage schema. That is to integrate the advantages of schema-conscious approach and schema-oblivious approach and avoid their disadvantages in terms of XPath-based queries. Compared to the existing RDBMS-based XML storage solutions, IXMMS has no dominant advantage in the operations of insertion and marshalling, since our novel XML storage schema is the combination of schema-conscious approach and schema-oblivious approach. However, IXMMS has prominent performance advantage in the query testing. It outperforms pure schema-conscious approach by up to 24 times, and pure schema-oblivious approach by up to 40 times. Compared to the XML-enabled database, IXMMS underperforms in terms of insertion and marshalling. However, it provides much more powerful query capability than the XML-enabled database, and outperforms the XML-enabled database by up to 430 times in terms of query testing. Since IXMMS introduces the multidimensional index mechanism, it also shows encouraging performance on the queries from multimedia perspective.

To summarize, the main contributions of this dissertation are listed as follows.

CHAPTER 9. CONCLUSIONS AND FUTURE WORK

- (i) Against the requirements for the management of MPEG-7 descriptions, a number of representative XML database solutions are investigated, and their deficiencies in the management of MPEG-7 descriptions are analyzed.
- (ii) To provide an adequate MPEG-7 descriptions management system, a research prototype, known as IXMMS, is proposed. IXMMS is based on RDBMS and OR-DBMS. The design of IXMMS pays attention to both XML perspective and multimedia perspective. It can fulfill most critical requirements for the management of MPEG-7 descriptions.
- (iii) A novel XML storage schema, known as SM3, is proposed. The main contribution of SM3 is to integrate the advantages of existing RDBMS-based XML storage schemas, avoid their intrinsic disadvantages, and provide much more efficient query performance.
- (iv) XQuery is extended to satisfy the special requirements of querying MPEG-7 descriptions. The proposed extension of XQuery, known as MXQuery, can support the queries on complex datatypes defined in MPEG-7 DDL, queries on spatial-temporal relationships and query-by-example.

9.2 Future Work

The results of this thesis call for various future research directions. Here we list a few possibilities.

- Capability of managing MPEG-21 documents. MPEG-21[72] is also an XML-based standard. The aim for MPEG-21 standard is to define a multimedia framework to enable the sharing of multimedia resources from content creator to content consumer across a wide range of networks and devices. If our system is extended to be

CHAPTER 9. CONCLUSIONS AND FUTURE WORK

capable of managing MPEG-21 standard, such a management system can be more powerful for the support of various multimedia applications.

- Improving the query translation algorithm. The back-end database is based on RDBMS, while the front-end query is based on XML schema. Although the query translation algorithm has been provided, it cannot support all the features of MXQuery due to the complexity of MXQuery and the gaps between MXQuery and SQL. Therefore, the query translation algorithm needs to be improved to narrow such gaps as much as possible.
- Extending the query language. MXQuery is fairly limited in representing complex multimedia queries. It needs to be improved to support the combination of high-level multimedia information and low-level multimedia content to further refine the query results.
- Introducing the state-of-the-art high dimensional index algorithms. The multidimensional index mechanism in IXMMS is extensible with new index structures. Thus, we should pay attention to any state-of-the-art high dimensional index algorithms, and introduce the appropriate ones to IXMMS to improve the performance of content based retrieval.

Appendix A

Test Operations

XMark

Exact Match:

- Q1. Return the name of the person with ID 'person0'.

Ordered Access:

- Q2. Return the initial increases of all open auctions.
- Q3. Return the first and current increases of all open auctions whose current increase is at least twice as high as the initial increase.
- Q4. List the reserves of those open auctions where a certain person issued a bid before another person.

Casting:

- Q5. How many sold items cost more than 40?

Regular Path Expressions:

- Q6. How many items are listed on all continents?
- Q7. How many pieces of prose are in our database?

Chasing References:

- Q8. List the names of persons and the number of items they bought. (joins person, closed auction)
- Q9. List the names of persons and the names of the items they bought in Europe. (joins person, closed auction, item)

Construction of Complex Results:

CHAPTER A. TEST OPERATIONS

- Q10. List all persons according to their interest; use French markup in the result.

Joins on Values:

- Q11. For each person, list the number of items currently on sale whose price does not exceed 0.02% of the person's income.
- Q12. For each person with an income of more than 50000, list the number of items currently on sale whose price does not exceed 0.02% of the person's income.

Reconstruction:

- Q13. List the names of items registered in Australia along with their descriptions.

Full Text:

- Q14. Return the names of all items whose description contains the word 'gold'.

Path Traversals:

- Q15. Print the keywords in emphasis in annotations of closed auctions.
- Q16. (Confer Q 15.) Return the IDs of the sellers of those auctions that have one or more keywords in emphasis.

Missing Elements:

- Q17. Which persons don't have a homepage?

Function Application:

- Q18. Convert the currency of the reserves of all open auctions to another currency.

Sorting:

- Q19. Give an alphabetically ordered list of all items along with their location.

Aggregation:

- Q20. Group customers by their income and output the cardinality of each group.

CHAPTER A. TEST OPERATIONS

MPEG-7

Deletion

- D1. Delete an MPEG-7 document which records the RegionShape descriptors of 20,000 images.
- D2. Delete UsageHistory with id "usage-history-001" in UserDescriptionType description.
- D3. Delete ColorStructure descriptor of the 5th image.

Update

- U1. Update the value of "Protected" attribute of the user "John Doe" as "False".
- U2. Update the ColorStructure value of the 10th image from 256 dimensions to 128 dimensions.
- U3. Replace the ScalableColor descriptor of the 100th image

Query

- MQ1. Find whether user 'John Doe' requests that his identity is revealed to third parties or not.
- MQ2. List the collected actions' type for user 'John Doe' in his usage history.
- MQ3. Return the user name who conducted 'Record' action over a period of six hours on the evening of October 10,2000.
- MQ4. Return the end time point of the user John Doe's observation that started at 18:00, October 10,2000.
- MQ5. Return the DominantColor descriptor information of the sixth image.
- MQ6. Return the value of ColorValueIndex with the maximum Percentage value for the 88th image.
- MQ7. Return the ScalableColor descriptor information of the sixth image.
- MQ8. Return the images that have red dominant color with more than 30 Percentage value.
- MQ9. Return the local-edge distribution information (EdgeHistogram D) of the images that have blue dominant color with more than 20 Percentage value.
- MQ10. Return the object name which locates at the right of point (10,10) in the image which media URL is 'image0.jpg'.

(ObjectType DS)

- MQ11. Return the transition probability from 'Pass' to 'Shot on goal'. (StateTransitionModelType DS)
- MQ12. Return the largest transition probability among the states and the labels of corresponding two states in the sound model which ID is 'ID3'. (SoundClassificationModelType DS)
- MQ13. Similarity searching on ColorLayout (12 dimensions).

CHAPTER A. TEST OPERATIONS

- MQ14. Similarity searching on RegionShape (35 dimensions).
- MQ15. Similarity searching on EdgeHistogram (80 dimensions).
- MQ16. Similarity searching on ColorStructure (128 dimensions).

Appendix B

Test Operation Timings

	BENCH001	BENCH01	BENCH	MPEG-7
IXMMS	5	70	1085	1933
Shared-Inlining	4	63	956	1799
SUCXENT++	5	72	1140	2920
XParent	21	251	2920	13060
DB2 XML Extender	1.2	9.8	327	1370

Table B.1: Insertion Timings (s)

	BENCH001	BENCH01	BENCH
IXMMS	1	8.6	196
Shared-Inlining	2.1	13.6	306
SUCXENT++	1.5	9.4	216
XParent	1.8	10.5	231
DB2 XML Extender	0.1	1.2	7.3

Table B.2: Marshalling Timings (s)

	D1	D2	D3	U1	U2	U3
IXMMS	1913	1331	3172	362	1105	3644
Shared-Inlining	1976	2154	3530	612	2056	4021
SUCXENT++	3019	460	3837	462	2565	4230
XParent	4421	1056	4850	1965	3579	5898
DB2 XML Extender	136			734	13609	11719

Table B.3: Deletion and Update Timings (ms)

CHAPTER B. TEST OPERATION TIMINGS

	IXMMS (with optimization)	IXMMS (without optimization)	Shared- Inlining	SUCXENT++	XParent	DB2 XML Extender
Q1	16	24	26	32	42	110
Q2	30	34	32	46	65	109
Q3	468	596	392	506	603	
Q4	26	31	30	29	306	
Q5	26	29	34	32	172	110
Q6	34	45	135	112	42	94
Q7	23	89	104	96	56	297
Q8	36	63	46	42	196	
Q9	165	201	96	135	789	
Q10	236	314	136	1436	1903	
Q11	1109	1284	947	1209	2560	
Q12	568	658	301	723	1256	
Q13	86	118	213	62	105	
Q14	165	235	326	155	416	
Q15	18	31	92	26	34	94
Q16	102	169	796	203	805	
Q17	36	43	63	89	136	
Q18	29	46	43	34	56	
Q19	23	46	256	97	297	109
Q20	45	96	89	86	125	

Table B.4: Query Timings – on BENCH001 (ms)

	IXMMS (with optimization)	IXMMS (without optimization)	Shared- Inlining	SUCXENT++	XParent	DB2 XML Extender
Q1	86	103	114	232	272	1281
Q2	283	320	332	603	787	938
Q3	4231	5803	4026	5236	6989	
Q4	89	295	302	265	1963	
Q5	65	105	163	156	701	906
Q6	78	134	689	563	172	891
Q7	43	365	412	412	205	2671
Q8	183	314	206	196	1135	
Q9	1123	1426	503	1056	4063	
Q10	1369	1708	769	10234	12690	
Q11	8532	9256	6658	9023	39015	
Q12	4063	4756	2196	5169	19806	
Q13	603	904	1630	506	872	
Q14	1359	2056	2656	1369	3276	
Q15	93	158	672	112	203	875
Q16	686	1063	5769	1295	5863	
Q17	205	302	396	598	1015	
Q18	102	215	203	186	234	
Q19	168	319	1938	609	2242	1182
Q20	48	123	105	96	187	

Table B.5: Query Timings – on BENCH01 (ms)

CHAPTER B. TEST OPERATION TIMINGS

	IXMMS (with optimization)	IXMMS (without optimization)	Shared- Inlining	SUCXENT++	XParent	DB2 XML Extender
Q1	102	196	388	703	972	11156
Q2	515	737	756	1312	3781	9953
Q3	10547	13797	10254	16523	36204	
Q4	160	592	625	470	6409	
Q5	78	132	325	301	2813	8860
Q6	91	225	1394	984	582	9031
Q7	63	641	1550	1460	712	27359
Q8	656	1262	1147	856	5844	
Q9	4032	6047	2037	4150	16509	
Q10	4122	7203	2775	50437	55140	
Q11	59832	63758	32703	61721	298437	
Q12	27025	29625	14563	31253	147406	
Q13	2562	2913	6234	2181	2863	
Q14	6105	6953	9136	6251	18961	
Q15	110	225	1360	141	628	9344
Q16	1116	1969	12946	2560	12312	
Q17	343	544	560	1932	5697	
Q18	188	381	271	206	1310	
Q19	750	1147	16856	2015	18669	12538
Q20	172	435	403	372	610	

Table B.6: Query Timings – on BENCH (ms)

	IXMMS (with optimization)	IXMMS (without optimization)	Shared- Inlining	SUCXENT++	XParent	DB2 XML Extender
MQ1	417	685	562	412	1915	65
MQ2	453	963	786	4466	4165	42
MQ3	797	1676	861	13953	18313	62
MQ4	866	1936	1276	14641	19891	158
MQ5	3265	3696	4352	3012	3122	5578
MQ6	906	1219	1865	3215	9872	
MQ7	1144	1220	1521	1230	1298	719
MQ8	245	392	446	605	909	
MQ9	547	838	1016	1325	2034	
MQ10	760	1401	1486	2652	5736	
MQ11	210	356	406	530	690	
MQ12	378	523	876	1865	3021	

Table B.7: Query Timings – on MPEG-7 (ms)

	MQ8	MQ9	MQ10	MQ11	MQ12
IXMMS	16	24	39	16	22
Inlining	20	32	49	23	30
SUCXENT++	23	35	66	20	41
XParent	28	45	125	22	56

Table B.8: Query Timings – on 1M array/matrix dataset (ms)

CHAPTER B. TEST OPERATION TIMINGS

	MQ8	MQ9	MQ10	MQ11	MQ12
IXMMS	75	94	135	60	96
Inlining	126	170	205	106	138
SUCXENT++	155	205	362	116	209
XParent	215	287	720	140	311

Table B.9: Query Timings – on 10M array/matrix dataset (ms)

	MQ8	MQ9	MQ10	MQ11	MQ12
IXMMS	145	254	420	103	196
Inlining	346	620	867	216	453
SUCXENT++	415	815	1652	276	965
XParent	622	1255	3463	348	1542

Table B.10: Query Timings – on 100M array/matrix dataset (ms)

	MQ13	MQ14	MQ15	MQ16
IXMMS	1.641	3.454	16.581	33.25
Shared-Inlining	7.813	19.187	35.507	43.61
SUCXENT++	17.153	19.207	35.587	43.71
XParent	22.263	19.287	35.657	43.79
DB2 XML Extender	26.133	30.387	45.737	72.08

Table B.11: Similarity Search Timings (s)

Appendix C

SQL Queries in IXMMS

Q1

```
SELECT p.name
FROM person p, internalnode i, xpath x
WHERE x.xpathexp = '/site#/people#/person'
      and i.xpathid = x.xpathid
      and p.id = 'person0'
      and p.uid = i.uid
```

Q2

```
SELECT b.increase
FROM bidder b, internalnode i, xpath x
WHERE x.xpathexp = '/site#/open_auctions#/open_auction#/bidder'
      and i.xpathid = x.xpathid
      and i.oid = 1
      and b.uid = i.uid
```

Q5

```
SELECT count(c.price)
FROM closed_auction c, internalnode i, xpath x
WHERE x.xpathexp = '/site#/closed_auctions#/closed_auction'
      and i.xpathid = x.xpathid
      and c.uid = i.uid
      and c.price >= 40
```

Q6

```
SELECT count(i.uid)
```

CHAPTER C. SQL QUERIES IN IXMMS

```

FROM internalnode i, xpath x
WHERE x.xpathexp like '#/site#/regions#%/item'
      and i.xpathid = x.xpathid

```

Q8

```

SELECT p.name, count(b.person)
FROM xpath x1,xpath x2,internalnode i1,internalnode i2,
      buyer b,person p
WHERE x1.xpathexp = '#/site#/closed_auctions#/closed_auction#/buyer'
      and x2.xpathexp = '#/site#/people#/person'
      and i1.xpathid = x1.xpathid
      and i2.xpathid = x2.xpathid
      and b.uid = i1.uid
      and p.uid = i2.uid
      and b.person = p.id
GROUP BY p.name

```

Q11

```

SELECT pe.name, count(o.uid)
FROM xpath x1,xpath x2,xpath x3,internalnode i1,internalnode i2,
      internalnode i3,open_auction o,profile pr,person pe
WHERE x1.xpathexp = '#/site#/people#/person#/profile'
      and x2.xpathexp = '#/site#/open_auctions#/open_auction'
      and x3.xpathexp = '#/site#/people#/person'
      and i1.xpathid = x1.xpathid
      and i2.xpathid = x2.xpathid
      and i3.xpathid = x3.xpathid
      and pr.uid = i1.uid
      and o.uid = i2.uid
      and pe.uid = i3.uid
      and pr.income > 5000*o.initial
      and i1.parent = i3.ordpath
GROUP BY pe.name

```

Q15

```

FROM xpath x,internalnode i,keyword k
WHERE x.xpathexp = '#/site#/closed_auctions#/closed_auction#/annotation
                  #/description#/parlist#/listitem#/parlist#/listitem
                  #/text#/emph#/keyword'

```

CHAPTER C. SQL QUERIES IN IXMMS

```
and i.xpathid = x.xpathid
and k.uid = i.uid
```

Q17

```
SELECT p.name
FROM xpath x,internalnode i,person p
WHERE x.xpathexp = '/site#/people#/person'
      and i.xpathid = x.xpathid
      and p.uid = i.uid
      and p.homepage is null
```

Q18

```
SELECT MULTIPLY_ALT(2.20371,o.reserve)
FROM xpath x,internalnode i, open_auction o
WHERE x.xpathexp = '/site#/open_auctions#/open_auction'
      and i.xpathid = x.xpathid
      and o.uid = i.uid
      and o.reserve is not null
```

Q19

```
SELECT it.name,it.location
FROM xpath x,internalnode i,item it
WHERE x.xpathexp like '/site#/regions#%/item'
      and i.xpathid = x.xpathid
      and it.uid = i.uid
ORDER BY it.name
```

Q20

```
WITH temp(id,level) AS
( SELECT p.uid, CASE
      WHEN p.income >= 100000 THEN 'preferred'
      WHEN p.income >= 30000 and p.income < 100000 THEN 'standard'
      WHEN p.income < 30000 THEN 'challenge'
      ELSE 'na'
    END
  FROM xpath x,internalnode i,profile p
  WHERE x.xpathexp = '/site#/people#/person#/profile'
        and i.xpathid = x.xpathid
```

CHAPTER C. SQL QUERIES IN IXMMS

```

        and p.uid = i.uid )
SELECT t.level, count(t.id)
FROM temp t
GROUP BY t.level

```

MQ1

```

SELECT u.protected
FROM xpath x1,xpath x2,internalnode i1,internalnode i2,
     useridentifier u,name n
WHERE x1.xpathexp like '#%/UserIdentifier#/Name'
      and x2.xpathexp like '#%/UserIdentifier'
      and i1.xpathid = x1.xpathid
      and i2.xpathid = x2.xpathid
      and n.uid = i1.uid
      and u.uid = i2.uid
      and i1.parent = i2.ordpath
      and n.value = 'John Doe'

```

MQ4

```

SELECT o.timestamp + decimal(o.timestampduration,20,6)
FROM xpath x1,xpath x2,xpath x3,internalnode i1,internalnode i2,
     internalnode i3,name n,observationperiod o
WHERE x1.xpathexp like '#%/UsageHistory'
      and x2.xpathexp like '#%/UsageHistory#/UserIdentifier#/Name'
      and x3.xpathexp like '#%/UsageHistory#/UserActionHistory
                          #/ObservationPeriod'
      and i1.xpathid = x1.xpathid
      and i2.xpathid = x2.xpathid
      and i3.xpathid = x3.xpathid
      and n.uid = i2.uid
      and o.uid = i3.uid
      and i2.ordpath > i1.ordpath and i2.ordpath < i1.grdesc
      and i3.ordpath > i1.ordpath and i3.ordpath < i1.grdesc
      and n.value = 'John Doe'
      and o.timepoint like '%2000-10-10T18:00%'

```

MQ9

```

SELECT e.bincounts
FROM xpath x1,xpath x2,xpath x3,internalnode i1,internalnode i2,

```


CHAPTER C. SQL QUERIES IN IXMMS

```

        internalnode i3, values v, edgehistogram e, array a
WHERE x1.xpathexp like '%/Descriptor#/Values'
      and x2.xpathexp like '%/Descriptor'
      and x3.xpathexp like '%/Descriptor'
      and i1.xpathid = x1.xpathid
      and i2.xpathid = x2.xpathid
      and i3.xpathid = x3.xpathid
      and i1.parent = i2.ordpath
      and v.uid = i1.uid
      and e.uid = i3.uid
      and i3.oid = i2.oid
      and v.percentage > 20
      and v.arrayid = a.id
      and a.col0 > 160 and a.col0 < 210

```

MQ10

```

WITH temp(id,value) AS
( SELECT matrixid,min(value)
  FROM matrix
 WHERE columnid = 0
 GROUP BY matrixid )
SELECT l.name
FROM xpath x1,xpath x2,xpath x3,xpath x4,internalnode i1,
      internalnode i2,internalnode i3,internalnode i4,
      label l, medialocator m, coords c,temp t
WHERE x1.xpathexp like '%/Object'
      and x2.xpathexp like '%/Object#/Label'
      and x3.xpathexp like '%/Object#/MediaOccurrence#/MediaLocator'
      and x4.xpathexp like '%/Object#/MediaOccurrence#/SubRegion
                          #/Polygon#/Coords'
      and i1.xpathid = x1.xpathid
      and i2.xpathid = x2.xpathid
      and i3.xpathid = x3.xpathid
      and i4.xpathid = x4.xpathid
      and i2.parent = i1.ordpath
      and i3.ordpath > i1.ordpath and i3.ordpath < i1.grdesc
      and i4.ordpath > i1.ordpath and i4.ordpath < i1.grdesc
      and l.uid = i2.uid
      and m.uid = i3.uid
      and c.uid = i4.uid
      and m.mediauri = 'image0.jpg'

```

CHAPTER C. SQL QUERIES IN IXMMS

```
and c.matrixid = t.id
and t.value > 10
```

MQ11

```
WITH temp(id,value) AS
( SELECT i1.oid,l.name
  FROM xpath x1,xpath x2,internalnode i1,internalnode i2, label l
 WHERE x1.xpathexp like '#%/Model#/State'
       and x2.xpathexp like '#%/Model#/State#/Label'
       and i1.xpathid = x1.xpathid
       and i2.xpathid = x2.xpathid
       and i2.parent = i1.ordpath )
SELECT m.value
FROM xpath x1,internalnode i1,transitions t,matrix m,temp te
WHERE x1.xpathexp like '#%/Model#/Transitions'
      and i1.xpathid = x1.xpathid
      and t.uid = i1.uid
      and t.matrixid = m.matrixid
      and m.columnid = te.id
      and te.value = 'Pass'
      and m.rowid = te.id
      and te.value = 'Shot on goal'
```

References

- [1] José M. Martínez, “MPEG-7 Overview (version 10),” Tech. Rep., ISO/IEC JTC1/SC29/WG11, Palma de Mallorca, 2004.
- [2] Harald Kosch, *Distributed Multimedia Database Technologies Supported by MPEG-7 and MPEG-21*, CRC Press, Boca Raton, 2003.
- [3] Intelligent Information Management Dept., “MARVEL: Multimedia Analysis and Retrieval System,” Tech. Rep., IBM T. J. Watson Research Center, New York, USA, 2005.
- [4] Mario Döller and Harald Kosch, “An MPEG-7 Multimedia Data Cartridge,” in *Proceedings of the SPIE Conference on Multimedia Computing and Networking 2003 (MMCN03)*, Santa Clara, CA, USA, 2003, pp. 29–31.
- [5] Utz Westermann and Wolfgang Klas, “PTDOM: a schema-aware XML database system for MPEG-7 media descriptions,” *Software-Practice & Experience*, vol. 36, no. 8, pp. 785–834, Mar. 2006.
- [6] B. S. Manjunath, Philippe Salembier, and Thomas Sikora, *Introduction to MPEG-7: multimedia content description interface*, John Wiley & Sons, New York, 2002.
- [7] Diane Hillmann, “Using Dublin Core,” Tech. Rep., Dublin Core Metadata Initiative (DCMI), 2005, Available at <http://dublincore.org/documents/usageguide/>.
- [8] STC Working Group, “Specification Series: S-2 (Phase 2) On: System Description,” Tech. Rep., The TV-Anytime Forum, 2005, Available at <http://www.tv-anytime.org/>.

REFERENCES

- [9] Leszek Cieplinski, Whoi-Yul Kim, Jens-Rainer Ohm, Mark Pickering, and Akio Yamada, "Information technology - Multimedia content description interface - Part 3 Visual," Tech. Rep., ISO and IEC, 2001.
- [10] Utz Westermann and Wolfgang Klas, "An Analysis of XML Database Solutions for the Management of MPEG-7 Media Descriptions," *ACM Computing Surveys*, vol. 35, no. 4, pp. 331–373, Dec. 2003.
- [11] Jason McHugh, Serge Abiteboul, Roy Goldman, Dallan Quass, and Jennifer Widom, "Lore: A database management system for semistructured data," *ACM SIGMOD Record*, vol. 26, no. 3, pp. 54–66, Sept. 1997.
- [12] Carl-Christian Kanne and Guido Moerkotte, "Efficient storage of XML data," in *Proceedings of the 16th International Conference on Data Engineering (ICDE'00)*, San Diego, California, USA, 2000, p. 198.
- [13] H. V. Jagadish, Shurug Al-Khalifa, Adriane Chapman, Laks V. S. Lakshmanan, Andrew Nierman, Stelios Paparizos, Jignesh M. Patel, Divesh Srivastava, Nuwee Wiwatwattana, Yuqing Wu, and Cong Yu, "TIMBER: A Native XML Database," *The VLDB Journal*, vol. 11, no. 4, pp. 274–291, Dec. 2002.
- [14] Josephine Cheng and Jane Xu, "IBM DB2 XML Extender: An end-to-end solution for storing and retrieving XML documents," in *Reprinted, with permission from the 16th International Conference on Data Engineering (ICDE'00)*, San Diego, California, USA, 2000.
- [15] Shankar Pal, Mark Fussell, and Irwin Dolobowsky, "XML Support in Microsoft SQL Server 2005," Tech. Rep., Microsoft Corporation, 2004, Available at Microsoft Developer Network (MSDN).
- [16] Mark Drake, "Oracle Database 10G Release 2 XML DB," Tech. Rep., Oracle Corporation, 2005, Available at <http://www.oracle.com/technology/tech/xml/xmlldb/index.html>.

REFERENCES

- [17] Jayavel Shanmugasundaram, Kristin Tufte, Gang He, Chun Zhang, David DeWitt, and Jeffrey Naughton, "Relational Databases for Querying XML Documents: Limitations and Opportunities," in *Proceedings of the 25th VLDB Conference*, Edinburgh, 1999, pp. 302–314.
- [18] Alin Deutsch, Mary Fernandez, and Dan Suciu, "Storing Semistructured Data with STORED," in *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, Philadelphia, Pennsylvania, USA, 1999, pp. 431–442.
- [19] Gerti Kappel, Elisabeth Kapsammer, Stefan Rausch-Schott, and Werner Retschitzegger, "X-Ray - Towards Integrating XML and Relational Database," in *Proceedings of the 19th International Conference on Conceptual Modeling (ER 2000)*, Salt Lake City, Utah, USA, 2000, pp. 339–353.
- [20] Philip Bohannon, Juliana Freire, Prasan Roy, and Jérôme Siméon, "From XML Schema to Relations: A Cost-Based Approach to XML Storage," in *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, San Jose, CA, USA, 2002, p. 64.
- [21] evin Beyer, Roberta J. Cochrane, and Vanja Josifovski, "System RX: one part relational, one part XML," in *Proceedings of the 2005 ACM SIGMOD*, Baltimore, Maryland, 2005, pp. 347–358.
- [22] Daniela Florescu and Donald Kossmann, "Storing and querying XML data using an RDBMS," *IEEE Data Engineering Bulletin*, vol. 22, no. 3, pp. 27–34, Sept. 1999.
- [23] Albrecht Schmidt, Martin Kersten, Menzo Windhouwer, and Florian Waas, "Efficient Relational Storage and Retrieval of XML Documents," in *Proceedings of the Third International Workshop on the Web and Databases (WebDB 2000)*, Dallas, Texas, USA, 2000, pp. 137–150.
- [24] Masatoshi Yoshikawa and Toshiyuki Amagasa, "XRel: A path-based approach to storage and retrieval of XML documents using relational databases," *ACM Transactions on Internet Technology*, vol. 1, no. 1, pp. 110–141, Jan. 2001.

REFERENCES

- [25] Haifeng Jiang, Hongjun Lu, Wei Wang, and Jeffrey Xu Yu, “XParent: An Efficient RDBMS-Based XML Database System,” in *Proceedings of the 18th International Conference on Data Engineering (ICDE’02)*, San Jose, CA, USA, 2002, pp. 335–336.
- [26] Sandeep Prakash, Sourav S. Bhowmick, and Sanjay Madria, “Efficient recursive XML query processing using relational database systems,” *Data & Knowledge Engineering*, vol. 58, no. 3, pp. 207–242, 2006.
- [27] Sleepycat Software, “Introduction to Berkeley DB XML,” Tech. Rep., Sleepycat Software, Massachusetts, USA, 2005.
- [28] Utz Westermann and Wolfgang Klas, “A Typed DOM for the Management of MPEG-7 Media Descriptions,” *Multimedia Tools and Applications*, vol. 27, no. 3, pp. 291–322, Dec. 2005.
- [29] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom, “Object Exchange Across Heterogeneous Information Sources,” in *Proceedings of the Eleventh International Conference on Data Engineering*, Taipei, Taiwan, 1995, pp. 251–260.
- [30] Patrick O’Neil, Elizabeth O’Neil, Shankar Pal, Istvan Cseri, Gideon Schaller, and Nigel Westbury, “ORDPATHs: Insert-Friendly XML Node Labels,” in *Proceedings of ACM SIGMOD*, Paris, 2004, pp. 903–908.
- [31] Feng Tian, David J. DeWitt, Jianjun Chen, and Chun Zhang, “The design and performance evaluation of alternative XML storage strategies,” *ACM SIGMOD Record*, vol. 31, no. 1, pp. 5–10, 2002.
- [32] Haifeng Jiang, Hongjun Lu, Wei Wang, and Jeffrey Xu Yu, “Path Materialization Revisited: An Efficient Storage Model for XML Data,” in *Proceedings of the Thirteenth Australasian Database Conference (ADC2002)*, Melbourne, Australia, 2002, pp. 85–94.
- [33] David DeHaan, David Toman, Mariano P. Consens, and M. Tamer özsü, “A comprehensive XQuery to SQL translation using dynamic interval encoding,” in *Proceedings of the 2003 ACM SIGMOD*, San Diego, California, 2003, pp. 623–634.

REFERENCES

- [34] Shankar Pal, Istvan Cseri, Oliver Seeliger, and et al, "XQuery implementation in a relational database system," in *Proceedings of the 31st international conference on Very large data bases (VLDB)*, Trondheim, Norway, 2005, pp. 1175–1186.
- [35] Darrell Woelk, Won Kim, and Willis Luther, "An object-oriented approach to multimedia databases," in *Proceedings of the 1986 ACM SIGMOD*, Washington, D.C., United States, 1986, pp. 311–325.
- [36] B. Phillips, "Mediaway presses access to multimedia database," *PC Week*, vol. 13, no. 7, pp. 39–40, 1996.
- [37] Setrag Khoshafian and Norayr Minassian, *The Jasmine Object Database: Multimedia Applications for the Web*, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1998.
- [38] Arjen P. de Vries, Mark G. L. M. van Doorn, Henk M. Blanken, and Peter M. G. Apers, "The Mirror MMDBMS Architecture," in *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB)*, Edinburgh, Scotland, UK, 1999, pp. 758–761.
- [39] Yong Rui, Thomas S. Huang, Michael Ortega, and Sharad Mehrotra, "Relevance feedback: a powerful tool for content-based image retrieval," *IEEE Transaction on Circuits and System for video technology*, vol. 8, no. 5, pp. 644–655, 1998.
- [40] Kriengkrai Porkaew, Sharad Mehrotra, and Michael Ortega, "Query reformulation for content based multimedia retrieval in MARS," in *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, Florence, 1999, p. 747.
- [41] Kaushik Chakrabarti, Kriengkrai Porkaew, and Sharad Mehrotra, "Efficient query refinement in multimedia databases," in *Proceedings of the 16th International Conference on Data Engineering (ICDE'00)*, San Diego, California, USA, 2000, p. 196.
- [42] Harald Kosch, László Böszörményi, Alexander Bachlechner, Christian Hanin, Christian Hofbauer, Margit Lang, Carmen Riedler, and Roland Tusch, "SMOOTH - A Distributed Multimedia Database System," in *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB)*, Roma, Italy, 2001, pp. 713–714.

REFERENCES

- [43] Yong Rui, Thomas S. Huang, and Sharad Mehrotra, "MARS and Its Applications to MPEG-7," Tech. Rep., ISO/IEC JTC1/SC29/WG11, 1997.
- [44] Joseph M. Hellerstein, Jeffrey F. Naughton, and Avi Pfeffer, "Generalized Search Trees for Database Systems," in *Proceedings of the 21th International Conference on Very Large Data Bases (VLDB'95)*, Zurich, Switzerland, 1995, pp. 562–573.
- [45] Arnaud Le Hors and Philippe Le Hégarret and Lauren Wood and Gavin Nicol and Jonathan Robie and Mike Champion and Steve Byrne, "Document Object Model (DOM) Level 2 Core Specification," Tech. Rep., World Wide Web Consortium (W3C), 2000, Available at <http://www.w3.org/TR/DOM-Level-2-Core/>.
- [46] Karl Gottschalk and et al, "Web Services architecture overview," Tech. Rep., IBM developerWorks, 2000, Available at <http://www-128.ibm.com/developerworks/library/w-ovr/>.
- [47] P. van Beek, A. B. Benitez, L. Cieplinski, J. Heuer, W. Kim, J. Martinez, J.-R. Ohm, M. Pickering, P. Salembier, Y. Shibata, J. Smith, and T. Walker A. Yamada, "Information Technology - Multimedia Content Description Interface - Part 8: Extraction and use of MPEG-7 descriptions," Tech. Rep., ISO/IEC JTC 1/SC 29/WG 11 N4579, 2002.
- [48] Surong Wang, Liang-Tien Chia, and Deepu Rajan, "Efficient image retrieval using MPEG-7 descriptors," in *Proceedings of International Conference on Image Processing (ICIP)*, Barcelona, 2003, pp. 1005–1008.
- [49] Yang Chu, Liang-Tien Chia, and Sourav S. Bhowmick, "SM3+: An XML Database Solution for the Management of MPEG-7 Descriptions," in *Proceedings of the 16th International Conference on Database and Expert Systems Applications (DEXA 2005)*, Copenhagen, Denmark, 2005, pp. 134–144.
- [50] Igor Tatarinov and Stratis D. Viglas, "Storing and Querying Ordered XML Using a Relational Database System," in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, Madison, Wisconsin, USA, 2002, pp. 204–215.

REFERENCES

- [51] ISO/IEC JTC 1/SC 29/WG 11, "Information Technology - Multimedia Content Description Interface - Part 2: Description Definition Language," Tech. Rep., ISO and IEC, 2001.
- [52] Scott Boag and Don Chamberlin and Mary F. Fernández and Daniela Florescu and Jonathan Robie and Jérôme Siméon, "XQuery 1.0: An XML Query Language," Tech. Rep., W3C Working Draft 29, 2004, Available at <http://www.w3.org/TR/xquery/>.
- [53] Marcel Kornacker, "High-Performance Extensible Indexing," in *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB'99)*, Edinburgh, Scotland, 1999, pp. 699–708.
- [54] Antonin Guttman, "R-trees: a dynamic index structure for spatial searching," in *Proceedings of ACM SIGMOD*, Boston, MA, USA, 1984, pp. 47–57.
- [55] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger, "The R*-tree: an efficient and robust access method for points and rectangles," in *Proceedings of ACM SIGMOD*, Atlantic City, NJ, USA, 1990, pp. 322–331.
- [56] David A. White and Ramesh Jain, "Similarity Indexing with the SS-tree," in *Proceedings of the 12th International Conference on Data Engineering*, Lillehammer, Norway, 1996, pp. 516–523.
- [57] Norio Katayama and Shin'ichi Satoh, "The SR-tree: an index structure for high-dimensional nearest neighbor queries," in *Proceedings of ACM SIGMOD*, Tucson, Arizona, United States, 1997, pp. 369–380.
- [58] Paolo Ciaccia, Marco Patella, and Pavel Zezula, "M-tree: an efficient access method for similarity search in metric spaces," in *Proceedings of the 23rd International Conference on Very Large Databases (VLDB'97)*, Athens, Greece, 1997, pp. 426–435.
- [59] Jonathan Robie, Joe Lapp, and David Schach, "XML Query Language (XQL)," in *Proceedings of the Query Languages Workshop (QL'98)*, Boston, Massachusetts, USA, 1998, pp. 303–314.

REFERENCES

- [60] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu, “A query language for XML,” *Computer Networks (Amsterdam, Netherlands: 1999)*, vol. 31, no. 11–16, pp. 1155–1169, 1999.
- [61] Peiya Liu, Liang H. Hsu, and Amit Chakraborty, “A predicate logic approach for MPEG-7 XML document queries,” *Markup Languages*, vol. 3, no. 3, pp. 365–381, 2001.
- [62] Nastaran Fatemi, Omar Abou Khaled, and Giovanni Coray, “An XQuery Adaptation for MPEG-7 Documents Retrieval,” in *Proceedings of XML conference and Exposition*, Philadelphia, USA, 2003.
- [63] Paul V. Biron, Kaiser Permanente, and Ashok Malhotra, “XML Schema Part 2: Datatypes Second Edition,” Tech. Rep., W3C Recommendation, 2004, Available at <http://www.w3.org/TR/xmlschema-2>.
- [64] Ashok Malhotra, Jim Melton, and Norman Walsh, “XQuery 1.0 and XPath 2.0 Functions and Operators,” Tech. Rep., W3C Candidate Recommendation, 2006, Available at <http://www.w3.org/TR/xpath-functions>.
- [65] John Z. Li, M. Tamer Özsu, and Duane Szafron, “Modeling of video spatial relationships in an object database management system,” in *Proceeding of the 1996 International Workshop on Multi-Media Database Management Systems*, Washington, DC, USA, 1996, pp. 124–133.
- [66] Max Egenhofer and Robert Franzosa, “Point-Set Topological Spatial Relations,” *International Journal of Geographical Information Systems*, vol. 5, no. 2, pp. 161–174, 1991.
- [67] James F. Allen, “Maintaining knowledge about temporal intervals,” *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983.
- [68] Nastaran Fatemi, *A Semantic Views Model for Audiovisual Indexing and Retrieval*, Ph.D. thesis, Swiss Federal Institute of Technology of Lausanne, 2003.

REFERENCES

- [69] Albrecht Schmidt, Florian Waas, Martin Kersten, Michael J. Carey, Ioana Manolescu, and Ralph Busse, “XMark: A Benchmark for XML Data Management,” in *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02)*, Hong Kong, China, 2002, pp. 974–985.
- [70] Feng Tian, David J. DeWitt, Jianjun Chen, and Chun Zhang, “The design and performance evaluation of alternative XML storage strategies,” *ACM Sigmod Record*, vol. 31, no. 1, pp. 5–10, 2002.
- [71] Centre for Multimedia And Network Technology, “PET - DEVICE ++: Push-Pull Extraction Tool For Distributed Audiovisual Content Terminal,” Tech. Rep., Nanyang Technological University, 2002.
- [72] Jan Bormans and Keith Hill, “MPEG-21 Overview v.5,” Tech. Rep., ISO/IEC JTC1/SC29/WG11/N5231, 2002.

List of Publications

Journal

1. Yang Chu, Liang-Tien Chia, and Sourav S. Bhowmick, Mapping, Indexing & Querying of MPEG-7 Descriptors in RDBMS with IXMDB, *Data & Knowledge Engineering (DKE)*, 63(2) (2007) pp. 224-257, 2007
2. Yang Chu and Liang-Tien Chia, IXMMS: an MPEG-7 descriptions management system based on RDBMS and ORDBMS, *Multimedia System Journal*, Submitted, 2007

Conference

1. Yang Chu, Liang-Tien Chia, and Sourav S. Bhowmick, Looking at Mapping, Indexing & Querying of MPEG-7 Descriptors in RDBMS with SM3, *in the proceeding of the Second ACM International Workshop on Multimedia Databases (ACM-MMDB 2004)*, Nov, 2004
2. Yang Chu, Liang-Tien Chia, and Sourav S. Bhowmick, SM3+: An XML Database Solution for the Management of MPEG-7 Descriptions, *in the proceeding of the 16th International Conference on Database and Expert Systems Applications (DEXA '05)*, Aug, 2005