Technical Report, No. TR-2002302, Dept. of Computer Science and Business Informatics, University of Vienna, September, 2002

An Analysis of XML Database Solutions Concerning the Management of MPEG-7 Media Descriptions

Utz Westermann, Wolfgang Klas

Multimedia Information Systems

Department of Computer Science and Business Informatics

University of Vienna, Austria

{gerd-utz.westermann,wolfgang.klas}@univie.ac.at

Abstract

MPEG-7 constitutes a promising standard for the description of multimedia content. It can be expected that a lot of applications based on MPEG-7 media descriptions will be set up in the near future. Therefore, means for the adequate management of large amounts of MPEG-7-compliant media descriptions are certainly desirable. Essentially, MPEG-7 media descriptions are XML documents following media description schemes defined with a variant of XML Schema. Thus, it is reasonable to investigate current database solutions for XML documents regarding their suitability for the management of these descriptions. In this paper, we motivate and present critical requirements for the management of MPEG-7 media descriptions and the resulting consequences for XML database solutions. Along these requirements, we discuss current state-of-the-art database solutions for XML documents. The analysis and comparison unveil the limitations of current database solutions with respect to the management of MPEG-7 media descriptions and point the way to the need for new solutions adequate to MPEG-7.

1 Introduction

The Multimedia Content Description Interface (MPEG-7) [33, 52, 53] is an ISO standardization effort that is attracting considerable attention of the multimedia community. Driven by broadcasting companies, consumer electronics manufacturers, and telecommuncation service providers, the standardization process started back in 1996 and reached a mature state in November 2001. MPEG-7 aims at the standardization of the description of multimedia content not only on a technical, feature-oriented level but also on a higher semantic level. Using MPEG-7, for example, one can describe the frequency spectrum of a song recording as well as the song's lyrics and musical score – all within one and the same media description.

MPEG-7 essentially standardizes two things: a so-called Description Definition Language (MPEG-7 DDL) [34] that allows the specification of schemes for the description of media, and, defined with MPEG-7 DDL, a comprehensive set of media description schemes that are considered to be of use in many application contexts. The standardized media description schemes comprise schemes for the description of visual and audible content [35, 36] and schemes that are of general use [37]. Applications do not need to stick to the standardized description schemes; they can create new description schemes with MPEG-7 DDL, either from scratch or by extending or combining existing schemes.

Because of its inherent flexibility via MPEG-7 DDL and because of the wealth of media description schemes already shipping with the standard, MPEG-7 is expected to face widespread use in a broad spectrum of applications, such as media archives, journalism, education, entertainment, etc. Thus, there is definitely the need for adequate means for the management of large numbers of MPEG-7 media descriptions. Since MPEG-7 media descriptions are XML documents which conform to a media description scheme expressed in MPEG-7 DDL, it is a self-suggesting idea to employ XML database solutions for their management, as it is proposed by [42] for instance.

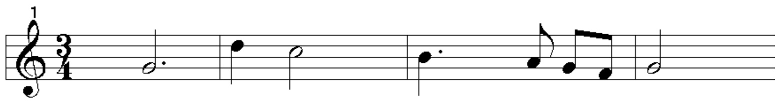
However, there exists a confusing variety of XML database solutions with different maturity and capabilities: "native" XML database solutions as well as XML extensions of traditional database management systems, commercial products as well as open source projects and research prototypes [7]. In order to be able to reasonably decide for an XML database solution for the management of MPEG-7 media descriptions, it is clearly necessary to have an overview of current XML database solutions and their capabilities and limitations.

In this regard, we make two substantial contributions in the paper: firstly, we motivate and present an extensive set of important requirements for the management of MPEG-7 media descriptions in a database. The requirements concern the representation of MPEG-7 media descriptions, the access to media descriptions, the ability to process media description schemes, extensibility, as well as classic database management functionality such as transactions and concurrency control. Secondly, we conduct a detailed analysis of 20 prominent and representative state-of-the-art XML database solutions with regard to these requirements. As we will see, the analysis exposes significant deficiencies of the examined solutions seriously affecting their eligibility for the management of MPEG-7 media descriptions. It indicates the need for a new generation of XML database solutions and gives a hint on the topics that need to be addressed by research.

The remainder of the paper is organized as follows: in Section 2, we give a short introduction to MPEG-7 media descriptions and observe some of their fundamental characteristics that are of importance for the management of MPEG-7 media descriptions with a database system. Based on these observations, we develop a comprehensive set of requirements for the management of MPEG-7 media descriptions in Section 3. We then briefly introduce a set of prominent XML database solutions in Section 4 which is followed by an in-depth analysis of these solutions with regard to the requirements in Section 5. Finally, we present our conclusions and an outlook in Section 6.

2 Observations about MPEG-7

Several basic observations about the MPEG-7 standard can be made that influence the management of MPEG-7 media descriptions. The first very basic observation is that MPEG-7 media descriptions are XML documents. Figure 1 gives a representative example of an MPEG-7 media description (taken from [36], page 101). The description captures the melody of a small part of a song by its meter and its melody contour. While the meter of a melody is a fraction number together with numerator and denominator, the melody contour is a compound measure consisting of contour and beat, both being lists of integer values. The contour measures the distance between every two consecutive notes of a melody; the beat associates every note with its position in the beat.



```
<!-- Melody description of 8 notes taken from “Moon River” by Henry Mancini -->

<AudioDescriptionScheme xmlns=“http://www.mpeg7.org/...”
                        xmlns:xsi=“http://www.w3.org/...”
                        xsi:type=“MelodyType”>
        <Meter>
                <Numerator>3</Numerator>
                <Denominator>4</Denominator>
        </Meter>
        <MelodyContour>
                <!-- Distance between two notes -->

                <Contour>2  -1  -1  -1  -1  -1  1</Contour>

                <!-- Beat position of notes -->

                <Beat>1  4  5  7  8  9  9  10</Beat>
        </MelodyContour>
</AudioDescriptionScheme>
```

Figure 1: Example of an MPEG-7 media description

Secondly, every MPEG-7 media description is valid to a media description scheme which is expressed in MPEG-7 DDL [34], a schema definition language for XML documents. MPEG-7 DDL is a superset of XML Schema [65, 3], the prevalent schema definition language for XML documents. MPEG-7 DDL extends XML Schema with support for array and matrix data types. As an example, Figure 2 gives a slightly simplified excerpt from the media description scheme to which our sample description of Figure 1 complies, namely the `Melody` media description scheme [36]. The given excerpt does not use any constructs specific to MPEG-7 DDL. Thus, it is also conforms to XML Schema.

As a third observation, we find that much of the information typically encoded in XML documents constituting MPEG-7 media descriptions is not of a textual nature. Large portions consist of numbers and mathematical structures like vectors and matrices – usually describing rather technical aspects of media content. The particular types of the data carried in a media description are specified in the associated media description scheme. The description of Figure

```
...

<complexType name="MelodyType">
  <complexContent>
    <extension base="mpeg7:AudioDSType">
      <sequence>
        <element name="Meter"
          type="mpeg7:MeterType"
          minOccurs="0"/>
        <element name="MelodyContour"
          type="mpeg7:MelodyContourType"
          minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<complexType name="MelodyContourType">
  <complexContent>
    <extension base="mpeg7:AudioDSType">
      <sequence>
        <element name="Contour">
          <simpleType>
            <list itemType="integer"/>
          </simpleType>
        </element>
        <element name="Beat">
          <simpleType>
            <list itemType="integer"/>
          </simpleType>
        </element>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

```
<complexType name="MeterType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <sequence>
        <element name="Numerator">
          <simpleType>
            <restriction base="integer">
              <minInclusive value="1"/>
              <maxInclusive value="128"/>
            </restriction>
          </simpleType>
        </element>
        <element name="Denominator">
          <simpleType>
            <restriction base="integer">
              <enumeration value="1"/>
              <enumeration value="2"/>
              <enumeration value="4"/>
              <enumeration value="8"/>
              <enumeration value="16"/>
              <enumeration value="32"/>
              <enumeration value="64"/>
              <enumeration value="128"/>
            </restriction>
          </simpleType>
        </element>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<element name="AudioDescriptionScheme"
  type="mpeg7:AudioDSType"/>

...
```

Figure 2: Simplified `Melody` media description scheme

1 is a prime example for this observation: the information contained consists solely of non-textual data such as numbers for measuring the meter of the song and lists of integer values for capturing the song's contour and beat.

Fourthly, the standard predefines a large variety of media description schemes using MPEG-7 DDL such as the `Melody` media description scheme we employed in our example. The predefined description schemes include schemes for visual content [35], schemes for audio content [36], and schemes for general use [37]. However, it has to be considered that applications are not limited to these predefined schemes – new media description schemes can be created or existing schemes may be extended with MPEG-7 DDL whenever necessary.

Finally, MPEG-7 makes extensive use of complex type derivation for the definition of the standardized media description schemes. By complex type derivation, the predefined description schemes are organized hierarchically. This hierarchical organization facilitates a form of polymorphism which is frequently encountered in MPEG-7 media descriptions: with the `xsi:type` attribute provided by MPEG-7 DDL, it can be expressed that the content of an element is not valid with respect to the complex type used in the element type declaration in the media description scheme. Instead, it is expressed that the content is valid to the complex type given by `xsi:type` which has to be derived from the original complex type.

This can be illustrated with our example media description scheme of Figure 2: the complex type `MelodyType` defining the structure of a melody description extends the predefined complex type `AudioDSType`[1]. By employing an `xsi:type` attribute, it is possible to use an `<AudioDescriptionScheme>` element as the root element of our sample melody description like in Figure 1. This is permitted even though, according to the media description scheme, the content of the element type `AudioDescriptionScheme` has to comply to the complex type `AudioDSType` originally, and not to the complex type `MelodyType`.

3 Requirements

In this section, we present essential requirements for the management of MPEG-7 media descriptions. As we have observed previously, MPEG-7 media descriptions are XML documents. The problem of adequately managing MPEG-7 media descriptions thus constitutes a problem of managing XML documents. General requirements for XML database solutions have already been addressed in literature [56]. In the following, however, we specifically look into the management of XML documents from the viewpoint of MPEG-7. Considering our basic observations about MPEG-7 of Section 2, we derive several critical requirements a database solution should fulfil to suit the needs of MPEG-7 media descriptions. We have grouped these requirements into requirements concerning the representation of media descriptions (3.1), the access to media descriptions (3.2), media description schemes (3.3), extensibility (3.4), and classic database management system (DBMS) functionality (3.5).

[1]Actually, `AudioDSType` itself extends the predefined complex type `DSType`, with `DSType` again extending the predefined complex type `Mpeg7RootType`. `Mpeg7RootType` constitutes the root of the derivation hierarchy from which every complex type predefined with MPEG-7 is ultimately derived.

3.1 Representation of media descriptions

The basic characteristics of MPEG-7 media descriptions considerably affect the way in which they should be represented for storage in an MPEG-7 database solution. As we will set out in the following, the nature of MPEG-7 especially calls for the *fine-grained representation* of a media description's structure as well as for the *typed representation* of the basic contents of a media description.

Fine-grained representation. MPEG-7 allows the definition of arbitrarily complex media description schemes with MPEG-7 DDL, capturing media content from possibly very different points of view and levels of abstraction. Regarding this potential complexity of MPEG-7 media descriptions, typical applications cannot be expected to process the full scope of a description. Instead, applications will likely process only those parts of a description necessary to achieve their particular tasks.

As a result, it is a fundamental requirement for a suitable MPEG-7 database solution that the structure of a media description is represented and stored with a fine granularity. A fine-grained storage representation that models the hierarchy of nodes in detail, i.e., the markups, of which an MPEG-7 media description consists, allows a database solution to efficiently provide applications with exactly those parts of a description that they are interested in.

If a media description was stored as a single unstructured object, in contrast, a database solution would have to read and parse the whole media description every time an application needs to access even just a small fraction of the description. Besides, the fine-grained storage representation of media descriptions is a prerequisite not only for the realization of a fine-grained concurrency control but also for the realization of a fine-grained access control, very desirable properties for the management of MPEG-7 media descriptions that we will motivate below.

A wide variety of fine-grained data models for XML documents that could be used for the storage of MPEG-7 media descriptions have been proposed in literature, e.g., [40, 24, 21, 60]. Moreover, several fine-grained data models for XML documents have appeared in the context of standardization efforts, such as the DOM Structure Model which is specified along with the DOM API by the DOM standard [44], the XPath Data Model which provides the basis for the evaluation of XPath expressions [9], the XML Information Set [10], and the XQuery 1.0 and XPath 2.0 Data Model [16], an early working draft currently being defined in connection with the XQuery standardization effort for a common XML query language [4].

Typed representation. As we have already exemplified by the means of the `Melody` media description scheme in Section 2, a large portion of the information contained in MPEG-7 media descriptions typically consists of non-textual data. Since MPEG-7 media descriptions are XML documents and, as such, a form of text documents, all these data are, self-evidently, encoded as text.

While this might be appropriate for the platform-independent exchange of media descriptions, we argue that the textual representation of non-textual data is inadequate for the storage of media descriptions within a database. Usually, textual representations of non-textual data not only consume more storage space than corresponding binary representations; typically, they are also less efficient and more complicate to handle. It is plausible, for example, that handling the list of integer values constituting the content of the `<Contour>` element in Figure 1 on the basis of the depicted textual representation – i.e., using string operations – is rather cumbersome compared to a data structure more appropriate for lists, e.g., an array. Finally, the textual representation of non-textual information is not always adequate to the semantics of the data, for instance with regard to ordering. As an example, the alphanumeric order of the textual representation of integer values differs from their numeric order. This complicates, for instance, meaningful indexing of integer values.

As a result, we demand that an MPEG-7 database solution stores the basic contents of a media description – more precisely, simple element content and the content of attribute values occurring in a media description – in a typed representation and not just textually. Typed representation means that these contents are encoded in binary data structures that suit the particular content type. In this regard, an eligible database solution should support the rich set of simple data types predefined by MPEG-7 DDL [34, 3] as well as the numerous derivation methods for simple types with which MPEG-7 DDL allows to flexibly derive new simple types from existing ones in a schema definition[2].

[2]The `Melody` media description scheme depicted in Figure 2 makes use of simple type derivation: among

Some of the data models for XML documents that have been proposed in literature, e.g., [24, 21], support – at least to some extent – the typed representation of a document's basic contents and could therefore be suitable foundations for the representation of MPEG-7 media descriptions in a database. Among the standard data models for XML documents, the current working draft of the XQuery 1.0 and XPath 2.0 Data Model permits the typed representation of simple element content and the content of attribute values as well.

3.2 Access to media descriptions

A fundamental service of an MPEG-7 database solution is to provide applications with adequate means for accessing media descriptions stored in a database. In this subsection, we argue that a suitable solution should allow *fine-grained access* to the structure of a media description, *typed access* to the basic contents carried by the description, and *fine-grained updates* of media descriptions. Moreover, we consider the availability of sophisticated index structures enabling efficient access even to large collections of media descriptions an imperative requirement for MPEG-7 database solutions. In the following, we particularly demand that a database solution for MPEG-7 provides powerful *value index structures*, *text index structures*, and *path index structures.*

Fine-grained access. As we have set out before, MPEG-7-based applications will typically not process the full scope of a media description but rather selectively access only those parts necessary to fulfil their particular tasks. A database solution for MPEG-7 media descriptions should therefore not only represent the structure of media descriptions with a fine granularity. Naturally, it should also provide applications with adequate means for the fine-grained, selective access to these descriptions to exploit that fine-grained representation.

The spectrum of possible means for the fine-grained access to the constituents of an MPEG-7 media description ranges from APIs such as the DOM API [44] that facilitate the programming of navigational access to XML documents to declarative querying facilities for XML documents like XPath expressions [9], XQL [55], and XQuery [4]. Since an expressive standard query

others, the content of the `Contour` and `Numerator` element types are specified with a list type and a range-restriced integer type respectively, both being derived from the predefined type integer.

language, namely XQuery, has not left the state of a working draft up today, programming navigational access to XML documents via the DOM API is still very common for XML document processing in practice. Given this situation, we do not have a preference for either paradigm in this paper but simply demand that fine-grained access to the structure of media descriptions is possible – be it with a navigational API, a query language, or perhaps even a combination of both.

Typed access. Non-textual information making up a significant portion of typical MPEG-7 media descriptions must be accessible for applications in a way that is appropriate for the particular data type. Again taking the list of integer values making up the content of the `<Contour>` element of Figure 1 as an example, accessing that list as a string in the depicted textual format has considerable drawbacks: apart from the fact that there is no indication for an application that this string actually represents a list of integer values, the application would either have to access the list by the means of string operations – which is not adequate – or to explicitly cast the string to a list of integer values before processing – which is cumbersome and costs additional processing power.

Closely related to our demand for the typed representation of a description's content, we therefore demand that an appropriate MPEG-7 database solution gives applications typed access to the basic contents of a media description: simple element content and the content of attribute values should be accessed via appropriate typed representations and not as text. A solution should additionally provide a rich set of type-specific operations such that applications can process the content in a way that is reasonable for the particular content type. Picking up once more our example list of integer values, among such type-specific operations could be functions for accessing and retrieving the single elements of the lists or for querying the size of the list.

Fine-grained updates. Similar to software development, multimedia content production can be seen as a repetitive process cycling through the phases of content production, post-production, delivery, and consumption with the latter phase potentially triggering content production again [38]. It is no surprise that media descriptions, just as the content they describe, are continuously evolving during the different phases of this cycle.

Regarding the fact that media descriptions are subject to change, an MPEG-7 database solution must provide adequate means for updating media descriptions. Hence, our call for the fine-grained access to media descriptions must be extended from mere read access to cover fine-grained update operations: unloading a complete media description from the database, modify it outside the database, and reinsert it back into the database just to change a potentially small fraction of the description is definitely not efficient and hampers concurrent access.

As for read access, the span of available means to support fine-grained updates ranges from DOM-like APIs allowing to imperatively program updates of XML documents to declarative update languages for XML documents such as XUpdate [43]. Again, we do not have a preference; we simply demand that a means for fine-grained update operations is provided.

Value index structures. To allow the efficient retrieval of media descriptions according to the content of elements or attribute values out of a large number of descriptions, the availability of value index structures is an indispensable requirement for an MPEG-7 database solution. These should at least include classic one-dimensional value index structures such as B-Trees that have long proved their effectiveness for traditional database applications.

However, MPEG-7 media descriptions often carry complex multimedia data that cannot be effectively indexed by one-dimensional value index structures. What are reasonable "less than" and "equal to" relations according to which, e.g., the `<Contour>` elements defined by the `Melody` media description scheme (see again Figure 1 for an example element) could be organized in an ordered, one-dimensional index structure like a B-Tree? In this case, indexing according to topological relations is more appropriate: for instance, a query-by-humming application needs to efficiently find out whether the contour of a melody fragment that has been hummed by a user is "contained" in the melody contour of a song that is stored in the database. Multidimensional value index structures [20] like R-Trees or extended k-d-Trees can support such queries. Therefore, we demand that an MPEG-7 database solution also provides multidimensional structures for the adequate indexing of complex multimedia data.

Text index structures. So far, we have been stressing the fact that non-textual content contributes significantly to the content of typical MPEG-7 media descriptions. Nevertheless, a considerable portion of the content still consists of textual information. To support the realization of versatile search engines for songs, for example, it would make perfect sense to capture not only the melody of a song using the `Melody` media description scheme but also the song's lyrics.

To facilitate powerful access to textual content, e.g., to retrieve all songs with lyrics containing a certain phrase, an MPEG-7 database solution should therefore have sophisticated text index structures at the disposal. Structures for text indexing [19] are well-known from the domain of information retrieval, e.g., inverted files and PAT-Trees.

Path index structures. Since applications seldom process complete MPEG-7 media descriptions but rather selected parts of media descriptions only, applications accessing an MPEG-7 database will often need to efficiently extract these parts from the descriptions stored in the database. The naive approach, i.e., accessing every media description in the database and traversing the description to reach the parts of interest employing the means for fine-grained access offered by the underlying MPEG-7 database solution, might turn out to be inefficient for large databases. Not only could the traversal times for all accessed descriptions pile up prohibitively for complex traversals. Also, media descriptions will be accessed unnecessarily if they do not contain the part of interest because, for instance, it has been declared optional in the media description scheme.

To accelerate the traversal times of frequently followed access paths and to prevent unnecessary access to media descriptions, the availability of structures for path indexing is an essential requirement for MPEG-7 database solutions. Applicable path index structures are well known from the domains of object-oriented databases, e.g., Multiindexes [2], semi-structured databases, e.g., DataGuides [25] and T-Indexes [50], and XML databases, e.g., SphinX [54].

3.3 Media description schemes

Media description schemes specifying the form of media descriptions are a central concept of the MPEG-7 standard. An MPEG-7 database solution must therefore provide an *MPEG-7-DDL-compliant schema catalog* for the management of media description schemes. Based on this catalog, a suitable solution should exploit media description schemes for the *validation of media descriptions*, for the *inference of typed representations*, and for *query optimization.*

MPEG-7-DDL-compliant schema catalog. We have discussed in Section 2 that MPEG-7 provides the schema definition language MPEG-7 DDL for the specification of media description schemes which define the allowable structure of media descriptions and the types of their basic contents. It should be no surprise that this information is valuable for the effective management of MPEG-7 media descriptions. Also, this information is of value for applications that require knowledge of the nature of media descriptions, such as editors for media descriptions.

To facilitate the utilization of schema and type information carried in media description schemes, an adequate MPEG-7 database solution must provide an MPEG-7-DDL-compliant schema catalog. The schema catalog is responsible for the management of the media description schemes to which the media descriptions contained in a database comply. For this purpose, the catalog must be able to parse media description schemes written in MPEG-7 DDL, to check their correctness and integrity, and to bring them into an appropriate representation for later utilization by the database solution and applications.

Since MPEG-7 DDL constitutes an extension of XML Schema, compliance to XML Schema might be sufficient for the schema catalog to cope with many description schemes such as our example `Melody` media description scheme. Nevertheless, an understanding of the MPEG-7-DDL-specific extensions is indispensable to deal with every media description scheme.

There exist data models for the detailed representation of schema definitions written in XML Schema. These might serve as a basis for the representation of media description schemes written in MPEG-7 DDL in a schema catalog. The XML Schema standard itself provides the XML Schema Component Data Model [65]; Abstract Schemas [8] have been proposed in the context of the current DOM Level 3 standardization [45] as a data model for representing schema definitions for XML documents in a schema dialect neutral way.

Validation of media descriptions. It is a fundamental requirement that an MPEG-7 database solution assures the consistency of the media descriptions stored in a database. For that purpose, a database solution must utilize the media description schemes kept in the schema catalog to validate the correctness of media descriptions with respect to their associated description schemes.

Such a validation must be performed during the import of a media description into a database so that the insertion of inconsistent media descriptions is prohibited. Moreover, the validation of media descriptions is necessary during updates: whenever an update of a media description violates its description scheme, it has to be rejected.

Inference of typed representations. We have already motivated that the provision of typed representations of the basic contents of a media description is an important requirement for an MPEG-7 database solution. However, MPEG-7 media descriptions themselves do not contain any type information for their basic contents that could be used to create appropriate typed representations; this information is contained in the media description scheme associated with the description. Without the `Melody` media description scheme of Figure 2, for instance, there is no indication that the content of the `<Contour>` element in the media description shown in Figure 1 constitutes a list of integer values and not just an arbitrary text.

To be able to automatically provide typed representations of the basic contents of a media description, it is therefore essential that an MPEG-7 database solution makes use of the type information available with the media description schemes of the schema catalog. Employing this information, the solution must infer the types and, by these types, create appropriate typed representations of the basic contents of the description.

Query optimization. If an MPEG-7 database solution possesses a declarative query language for the fine-grained access to media descriptions, it should also provide means for query optimization to reduce query evaluation times. An essential prerequisite to the sophisticated optimization of queries against media descriptions is the availability of schema information which is contained in media description schemes.

An MPEG-7 database solution can utilize this schema information to optimize queries against media descriptions in a number of ways [5, 17]: expressions contained in a query that, according to a media description scheme, can never yield a result or that are redundant to other expressions contained in the same query can be cut off thereby reducing the complexity of queries. Also, queries can be equivalently rewritten such that existing index structures can be exploited during query evaluation.

3.4 Extensibility

We consider extensibility a desirable property of an MPEG-7 database solution. As we motivate in the following, an ideal MPEG-7 database solution should offer *extensibility with functionality* and *extensibility with index structures.*

Extensibility with functionality. MPEG-7 basically constitutes a means for the definition of schemes for the description of multimedia content that is supplemented by a profound library of standardized, ready-to-use description schemes. However, the standard does not prescribe how MPEG-7-compliant media descriptions are to be processed by applications. For example, MPEG-7 defines a way how the melody of a song can be described with the `Melody` media description scheme; but it is left to an application how descriptions following the `Melody` media description scheme are employed, e.g., to compare two songs for similarity.

Given this situation, it is difficult for an MPEG-7 database solution to anticipate and provide meaningful operations for the treatment of media descriptions or parts of media descriptions that are of use for all applications. Coming back to our example, different applications might even need different measures of similarity for the comparison of songs on the basis of the `Melody` media description scheme.

Therefore, we demand that an MPEG-7 database solution is extensible with functionality. Applications should be able to plug in procedures and functions for the processing of media descriptions and parts of media descriptions that suit their particular purposes. This corresponds to the concept of stored procedures that is well-known from relational DBMSs. The functional extensions should be callable with the particular means for accessing media descriptions offered by a database solution, such as an API or a declarative query language. An example of a query language for XML documents that explicitly considers functional extensions is XQuery: the XQuery language allows the definition of custom functions which can then be used in queries just like any other function that is built-in with the language.

Extensibility with index structures. Since the MPEG-7 standard does not prescribe how media descriptions are to be processed, it is difficult to assess what index structures should be available with an MPEG-7 database solution to best fit the needs of different applications that are going to work with the solution. For instance, the availability of an R-Tree index structure for the indexing of media descriptions could perfectly suit the needs of a particular application while an extended k-d-Tree index structure could prove more effective for other applications.

Moreover, specialized index structures might exist focused on the needs of a specific application that are not useful for other applications. For example, one could imagine a highly specialized value index structure specifically designed for a classic music archive application enabling very efficient retrieval of classic music based on the `<Contour>` element of media descriptions complying to the `Melody` media description scheme. Such an index structure does not need to be effective for the retrieval of pop songs as well.

To flexibly accomodate the different indexation needs of applications, we believe that an MPEG-7 database solution should be extensible with new index structures. It should provide an open, sophisticated programming interface facilitating the integration of new value, text, or path index structures without the need to modify the system's kernel. Interfaces for the integration of new index structures exist in the context of object-relational DBMSs. For example, Oracle features the Extensible Indexing API allowing the provision of new index structures [22].

3.5 Classic DBMS functionality

So far, we have been focusing mainly on requirements specific to the management of MPEG-7 media descriptions. Nevertheless, there are general requirements for DBMSs that we also consider important for an MPEG-7 database solution. In the following, we demand that an MPEG-7 database solution provides *transactions*, *fine-grained concurrency control*, *fine-grained access control*, *backup and recovery* mechanisms, and *versioning* functionality.

Transactions. If an MPEG-7 database solution is not just intended to be used in pure content retrieval application scenarios where read accesses to media descriptions dominate but rather aimed at supporting the whole process of multimedia content production where concurrent and potentially conflicting read and write accesses to media descriptions occur during the different phases of the process, support for transactions is a highly desirable requirement.

Transactions serve to group logically related access operations to a database. For the execution of these operations, transactions provide an isolated, consistent view on the database content giving the impression of a single user system. Thereby, the availability of transactions greatly simplifies application development in a multiuser scenario. Also, transactions guarantee atomicity and durability of all the write accesses performed during a transaction significantly enhancing fault tolerance and consistency of database content. These characteristics of transactions are often referred to as the ACID (atomicity, consistency, isolation, durability) properties.

The ACID properties of traditional transactions might turn out to be too rigorous for accessing MPEG-7 media descriptions in a database. For example, if media descriptions are created and manipulated cooperatively by several people at a time or if the process of creating media descriptions is so time consuming that it results in very long running transactions, the use of non-standard transaction models [14] softening up the ACID properties can be of advantage. It should be evident, for example, that the isolation property directly contravenes the desire for cooperative access and that rolling back an entire transaction that took a week because of a slight inconsistency just for the sake of atomicity is not really satisfactory.

However, the decision of applying standard or non-standard transaction models for the management of MPEG-7 media descriptions highly depends on the particular process applied for multimedia content production. Nevertheless, some kind of transaction support should be available with a MPEG-7 database solution.

Fine-grained concurrency control. For the synchronization of conflicting read and write accesses to the media descriptions of a database potentially taking place in concurrent transactions, an MPEG-7 database solution must provide mechanisms for concurrency control usually based on locking techniques. A concurrency control mechanism for an MPEG-7 database solution can be either coarse-grained, i.e., synchronization is performed by locking complete databases or media descriptions, or fine-grained, i.e., synchronization is performed by locking single elements and attribute values of a media description or single disk pages that are used for the storage of a media description.

For the reasonable management of MPEG-7 media descriptions, we demand that fine-grained concurrency control is possible. This is motivated by the fact that different parts of a complex MPEG-7 media description may describe potentially very different, independent aspects of media content. For example, one part of a media description might describe the melody of a song according to the `Melody` media description scheme while another part of the same description might contain statistical usage information, e.g., profile data regarding the users that have downloaded the song from a media server. In contrast to a fine-grained concurrency control mechanism, a coarse-grained mechanism locking complete media descriptions or even complete databases would interdict concurrent accesses to different aspects of a media description. This is often unnecessary: in our example, there is no reason to hinder a song retrieval engine from accessing the melody description of a song just because another application is simultaneously updating the usage information.

Fine-grained access control. Controlling access to media descriptions by enforcing access rights is an important task of an MPEG-7 database solution. This is of particular relevance if it is intended to support the whole process of multimedia content production: during the phases of this process different applications and different users with different responsibilities and different rights are going to access the same database content. Similar to concurrency control, access control can be either coarse-grained, i.e., access can be restricted for complete databases or media descriptions, or fine-grained, i.e., access can be restricted for single elements and attribute values contained in a media description.

Regarding the fact that different parts of a MPEG-7 media description may treat very different aspects of the media content described, an ideal MPEG-7 database solution should facilitate fine-grained access control: different applications and users might be authorized to access and modify only certain parts of a media description. Again considering the example of a media description capturing the melody of a song as well as statistical usage information, fine-grained access control, unlike coarse-grained access control, allows to ensure that the statistics application can change the usage information for the song but not the song's melody contour.

For the specification of fine-grained access rights for MPEG-7 media descriptions, recent research results concerning fine-grained access control to XML documents [11, 26] are available. This research has also triggered standardization efforts such as XACML [23] which are, however, still in an early state.

Backup and recovery. MPEG-7 media descriptions, just like the content they describe, constitute valuable assets: creating and mainting large amounts of MPEG-7 media descriptions is expensive because this typically requires domain knowledge that generally cannot be automatically extracted from the content but rather has to be brought in by humans manually. Furthermore, media descriptions are critical for the effective organization of multimedia content. Without media descriptions, efficiently retrieving a piece from a multimedia content collection closely resembles the proverbial search for the needle in the haystack.

Given their value, it is highly desireable that MPEG-7 media descriptions are adequately protected against imponderabilities such as system crashes and media failures. Therefore, we demand that an MPEG-7 database solution provides reliable means for the backup and recovery of media descriptions.

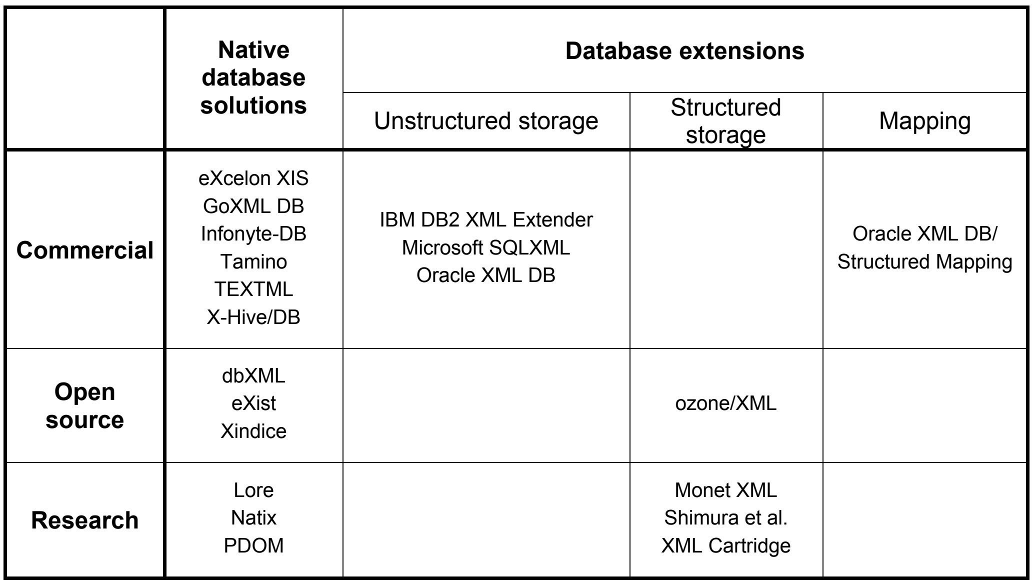
Versioning. As we have set out before, multimedia content production constitutes a repetitive process. During the iterations of this process, multimedia content is constantly edited and rearranged. As a consequence, different versions of the same content can exist at a particular time. For instance, a life recording of a pop song could be edited during the postproduction phase to conceal musical deficiencies thereby producing a new version of the original recording. As another example, a longer version of a song intended for distribution on a Maxi CD could be produced as an alternative to the shorter standard version of the song intended for distribution via radio broadcast.

If multimedia content is accompanied by media descriptions throughout the whole production process, different versions of media descriptions will have to be produced along with the different versions of the content. Accordingly, it would be advantageous if an MPEG-7 database solution provided means for the versioning of media descriptions. For example, a sophisticated versioning of media descriptions would allow applications to access all current and past versions of a media description, to trace the predecessor and successor relationships between the different versions of a description, and to derive new versions of a media description from existing ones (see, e.g., [58]).

Roughly, two approaches to versioning can be distinguished: linear versioning, where there is only one current version of a media description and a new version of a description always replaces the older one, and branched versioning, where there can be multiple concurrent versions of a single media description. It depends on the concrete production workflow according to which media descriptions are created whether linear or branched version is more appropriate for a particular application.

4 XML database solutions

Since MPEG-7 media descriptions are XML documents, it is self-suggesting to employ XML database solutions for their management. Unfortunately for our investigations, development in the area of XML database solutions is still very active: constantly, new XML database solutions emerge while other solutions vanish. Thus, there exists a confusing variety of systems at different development stages and with different degrees of maturity that are concerned with the management of XML documents in a database [7]. To permit an analysis of XML database solutions with regard to the management of MPEG-7 media descriptions within the scope of this paper, we therefore have restricted our investigations on a set of prominent and representative XML database solutions sufficiently mature for reasonable examination.



	Native database solutions	**Database extensions**		
		Unstructured storage	Structured storage	Mapping
Commercial	eXcelon XIS GoXML DB Infonyte-DB Tamino TEXTML X-Hive/DB	IBM DB2 XML Extender Microsoft SQLXML Oracle XML DB		Oracle XML DB/ Structured Mapping
Open source	dbXML eXist Xindice		ozone/XML	
Research	Lore Natix PDOM		Monet XML Shimura et al. XML Cartridge	

Figure 3: Selection of XML database solutions

As shown in Figure 3, the selected set of XML database solutions consists of commercial products, open source projects, and research prototypes that we have roughly categorized into native database solutions specifically developed for the management of XML documents and database extensions augmenting conventional, typically relational or object-oriented database technology with support for XML. We regard as the essential difference between both categories that a native database solution allows the modeling of data only by the means of XML documents while a database extension still offers applications the modeling primitives of the extended system. Hence, the term “native” does not necessarily imply that a native database solution has always been implemented specifically for the management of XML documents from the ground up; a native solution might very well base on conventional database technology as long as the data model of the underlying database system is entirely hidden.

In the rest of this section, we briefly introduce the native XML database solutions (4.1) and XML database extensions (4.2) that are given in Figure 3.

4.1 Native database solutions

A variety of commercial native XML database solutions have appeared on the market to serve the increasing need for the efficient management of large amounts of XML documents. Arguing that XML documents cannot be efficiently stored in conventional DBMSs because of the documents' hierarchical and semistructured nature, several vendors have developed entire DBMSs specialized on the management of XML documents. Perhaps the most prominent representative of this group of native XML database solutions is Software AG's Tamino [61]; commercial solutions falling in the same category are X-Hive/DB [67] and GoXML DB [68]. Infonyte-DB [32] which evolved from the research prototype PDOM [28] has also been specifically developed for the management of XML documents. However, Infonyte-DB does not provide a full-fledged database server with all the standard DBMS functionality such as transaction management and concurrency control but rather constitutes a lightweight in-process storage solution for XML documents.

Furthermore, vendors of object-oriented DBMSs have seized the opportunity to get their slice of cake from the market of XML document management and reshaped their existing DBMSs to native XML database solutions. A prominent representative of this approach is eXcelon XIS [15] which internally makes use of the object-oriented DBMS ObjectStore for the storage of XML documents. In a similar manner, vendors of document management systems have reused their existing technology to produce native XML database solutions such as TEXTML [39].

Apart from these commercial solutions, several open source projects aiming at the development of native XML database solutions have emerged recently. The Apache XML Project has started an initiative aiming at the implementation of a specialized DBMS for XML documents called Xindice [63] from the gound up. Xindice is the successor of dbXML [62] previously driven by the dbXML Project. eXist [47] is another example of an open source project implementing a native database solution. In contrast to Xindice, however, eXist is built on top of a relational database system, optionally MySQL or PostgreSQL, which internally serves as the persistent storage backend.

Finally, there has also been considerable research concerning native XML database solutions. A prominent example is the semistructured DBMS research prototype Lore that has been developed towards a native XML database solution [24]. The idea is to exploit Lore's ability to represent irregular graph structures (including hierarchies) and the system's powerful query language for management of XML documents. Kanne and Moerkotte [41] have proposed another prototype of a native database solution called Natix. A main research focus of Natix is the provision of efficient techniques for mapping the hierarchical structure of XML documents to disc pages.

4.2 Database extensions

Basically, three approaches for representing XML documents in conventional DBMSs can be distinguished. In the first approach, which we call unstructured storage in the following, an XML document is stored directly in its textual format in a character large object (CLOB). This is the approach supported by most of the major relational DBMSs today: these systems have been extended with CLOB-based data types for XML documents along with more or less sophisticated functions for accessing a document's content from SQL. Prominent representatives of relational database extensions supporting the unstructured storage of XML documents are Oracle XML DB [27], IBM DB2 XML Extender [29], and Microsoft SQLXML [49].

In the second approach, which we call structured storage, a fine-grained metamodel of XML documents capable of representing the node trees of arbitrary XML documents is built by employing the modeling primitives of the underlying conventional DBMS. This opens up the structure and the contents of XML documents to the querying facilities provided with the

DBMS. There has been considerable research in this area (see [18] for an overview) and a lot of research prototypes of database extensions for the structured storage of XML documents have been developed, mostly for relational DBMSs. For our analysis, we focus on the representative prototypes XML Cartridge [21], Shimura et al. [60], and Monet XML [57, 6] extending the relational DBMSs Oracle, PostgreSQL, and Monet respectively with support for the management of XML documents.

There are also open source projects that provide structured storage extensions for conventional DBMSs. We have included the open source project ozone/XML [13] into our analysis. ozone/XML is a library of persistent-capable classes for the object-oriented DBMS ozone that implements the DOM standard for the representation of XML documents in a database.

Finally, in the third approach for representing XML documents in conventional DBMSs, which we call mapping, the content of XML documents is mapped to database schemas specifically designed for that content. This, in principle, places all the modeling capabilities available with a conventional DBMS at the disposal to efficiently and adequately represent document content. There exists a large number of tools and formalisms for the specification of the mapping between an XML format and a database schema. For example, IBM DB2 XML Extender and Microsoft SQLXML provide the concepts of document access definitions (DAD) and annotated schemas respectively for this purpose. Nevertheless, we have decided to neglect such tools and formalisms for our analysis: with these, the design of a database schema appropriate for the content of an XML format and the specification of the mapping between the XML format and the database schema are elaborate manual tasks. Bearing in mind that MPEG-7 allows the definition of arbitrary media description schemes in excess to those predefined with the standard, the effort necessary to cope with a media description following a previously unknown description scheme would be prohibitive.

Recently, there has been considerable research concerning the automatic derivation of relational database schemas from schema definitions for XML documents and the automatic mapping between them [59, 66, 12]. As these approaches are based on Document Type Definitions (DTDs) and not on schema definitions written in the far more complex MPEG-7 DDL, they do not qualify for the management of MPEG-7 media descriptions. However, in its latest release of the Oracle XML DB database extension, Oracle has picked up the basic approach and extended it for XML Schema to provide an additional storage option for XML documents in excess to unstructured storage. Oracle calls this storage option Structured Mapping [27]. Since XML Schema constitutes a large subset of MPEG-7 DDL, we have taken Oracle XML DB/Structured Mapping into account for our analysis.

5 Analysis

In this section, we analyze in detail to what extent the XML database solutions introduced in Section 4 are suitable for the management of MPEG-7 media descriptions. For that purpose, we evaluate these database solutions along our specific requirements for the management of MPEG-7 media descriptions outlined earlier in Section 3.

Accordingly, we in the following examine the fulfillment of the requirements concerning the representation of media descriptions (5.1), the access to media descriptions (5.2), media description schemes (5.3), extensibility (5.4), and classic DBMS functionality (5.5). We conclude this section with a summary of our findings (5.6). For a better orientation in the ensuing discussion, we have visualized the results of our analysis in Figure 4.

5.1 Representation of media descriptions

Fine-grained representation. Native XML database solutions typically represent XML documents, and hence also MPEG-7 media descriptions, in a fine-grained manner. For storage, they internally decompose an XML document into the individual nodes of which it consists. eXcelon XIS, Infonyte-DB, PDOM, X-Hive/DB, Xindice, dbXML, and eXist represent these nodes according to the DOM Structure Model. GoXML DB employs the XQuery 1.0 and XPath 2.0 Data Model for the same purpose whereas the research prototypes Lore and Natix define their own, non-standard data models based on edge-labeled trees for the internal representation of XML documents.

In contrast, the native solutions Tamino and TEXTML basically store XML documents in their entirety in their original textual format, only applying compression techniques to reduce the consumption of storage space. This is comparable to the XML database extensions IBM DB2 XML Extender, Microsoft SQLXML, and Oracle XML DB which store XML documents

		Native database solutions										Database extensions							
		eXcelon XIS	GoXML DB	Infonyte-DB/ PDOM	Tamino	TEXTML	X-Hive/DB	Xindice/ dbXML	eXist	Lore	Natix	IBM DB2 XML Extender	Microsoft SQLXML	Oracle XML DB	ozone/XML	Monet XML	Shimura et al.	XML Cartridge	Oracle XML DB/ Structured Mapping
Representation of media descriptions	Fine-grained representation	■	■	■	□	–	■	■	■	■	■	–	–	–	■	■	■	■	□
	Typed representation	–	–	–	–	–	–	–	–	□	□	–	–	–	–	–	–	□	□
Access to media descriptions	Fine-grained access	■	■	■	■	–	■	■	■	■	■	■	■	■	■	■	■	■	■
	Typed access	□	–	–	□	□	–	–	–	□	□	–	–	–	–	–	–	□	□
	Fine-grained updates	■	■	■	–	–	■	■	–	■	■	–	–	–	■	■	■	■	■
	Value index structures	□	□	–	□	□	□	□	–	□	–	□	–	□	–	□	□	□	□
	Text index structures	■	■	–	■	■	■	–	■	■	–	■	■	■	–	–	–	■	■
	Path index structures	■	–	■	■	–	■	–	–	■	–	–	–	■	–	■	■	■	–
Media description schemes	MPEG-7-DDL-compliant schema catalog	□	□	–	□	–	–	–	–	–	–	–	–	□	–	–	–	–	□
	Validation of media descriptions	□	□	–	□	–	–	–	–	–	–	–	–	□	–	–	–	–	□
	Inference of typed representations	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	□
	Query optimization	–	–	–	■	–	–	–	–	–	–	–	–	–	–	–	–	–	■
Extensibility	Extensibility with functionality	■	–	–	■	–	–	■	–	–	–	■	■	■	■	■	■	■	■
	Extensibility with index structures	–	–	–	–	–	□	–	–	–	–	–	–	■	–	■	–	■	■
Classic DBMS functionality	Transactions	■	■	–	■	–	■	–	–	■	–	■	■	■	■	■	■	■	■
	Fine-grained concurrency control	■	■	–	–	–	–	–	–	■	–	–	–	–	■	□	■	■	□
	Fine-grained access control	–	–	–	■	–	–	–	–	–	–	–	–	–	■	–	–	–	□
	Backup and recovery	■	□	□	■	□	■	–	■	□	–	■	■	■	■	■	■	■	■
	Versioning	–	–	–	–	–	■	–	–	–	–	–	–	■	–	–	–	–	■

Figure 4: Analysis results (■ support, □ partial support, – no support)

in CLOBs in an unstructured manner. Tamino somewhat alleviates the disadvantages of such a coarse-grained storage representation by managing additional fine-grained structural information: a structure index maintains path information for the individual elements and attribute values occurring in a document.

Since XML database extensions that follow the structured storage approach apply the modeling primitives of the underlying DBMS for the definition of detailed metamodels for XML documents, they offer a fine-grained storage representation of MPEG-7 media descriptions by construction: the object-oriented database extension ozone/XML is a library of persistent classes for the object-oriented DBMS ozone that constitutes a 1:1 implementation of the DOM Structure Model and API. The relational extensions Shimura et al. and XML Cartridge represent XML documents by the means of edge-labeled trees which they store in a central edge table, whereas Monet XML dynamically creates a table for every distinct path by which an element or attribute value occurring in an XML document contained in the database can be reached from the root of the respective document. Every element and attribute value of a document is then stored in the table created for the corresponding path.

Finally, the database extension Oracle XML DB/Structured Mapping in principle stores XML documents with a fine granularity as well. The system's mapping scheme basically creates an SQL object type for every complex type declared in a schema definition written in XML Schema. Such an SQL object type has one field each for every element type and attribute of which the corresponding complex type consists. In case that a field corresponds to an element type with simple content or an attribute, the field is assigned the atomic SQL data type coming closest to the simple type defining the content of the element type or the attribute. In case that a field corresponds to an element type with complex content, the field is assigned the SQL object type created for the complex type defining the content of the element type. Multiple occurrences of element types are handled by collection-typed fields. For every root element type of the schema definition, a table is created with a column that is assigned the SQL object type corresponding to the complex type defining the content of the respective element type. To store an XML document that complies to the given schema definition, Oracle XML DB/Structured Mapping then maps the content of that document in a fine-grained manner to the corresponding columns and fields of the tables and SQL object types created.

Nevertheless, the mapping scheme of Oracle XML DB/Structured Mapping has got limitations: elements with mixed content and elements with arbitrary content cannot be represented, as well as elements which make use of the `xsi:type` attribute for polymorphism. Oracle XML DB/Structured Mapping circumvents these restrictions by augmenting the SQL object types with additional CLOB fields serving as overflow stores that keep those fractions of XML documents that cannot be represented otherwise textually in an unstructured manner. As MPEG-7 makes considerable use of the problematic constructs mentioned above (especially of polymorphism via the `xsi:type` attribute, see Section 2), significant portions of typical MPEG-7 media descriptions can be expected to be kept in these overflow stores. This considerably reduces the granularity of the storage representation for MPEG-7 media descriptions. Our example media description of Figure 1, for instance, would be stored completely in an overflow store, because the root `<AudioDescriptionScheme>` element makes use of the `xsi:type` attribute already.

Typed representation. The investigated XML database solutions have strong deficiencies with regard to the adequate storage representation of non-textual information significantly contributing to the content of MPEG-7 media descriptions. Most of them store the basic contents of an XML document, i.e. simple element content and the content of attribute values, as text regardless of the particular content type.

In fact, only a few solutions address the issue of typed representation of basic document contents. The native database solution GoXML DB, as it employs the XQuery 1.0 and XPath 2.0 Data Model for the storage of XML documents, could in principle exploit the model's capabilities with regard to the typed representation of simple element content and the content of attribute values. Experiments with the current Version 2.0.2 revealed, however, that this model feature has apparently not been implemented: such content is always stored and treated as text.

The native database solutions Lore and Natix as well as the database extension XML Cartridge to some extent implement typed representations: for the representation of the content of attribute values, Lore provides a set of atomic data types including integer, real, and string. Simple element content, however, is always represented as text. The storage representation of Natix can represent the basic contents of a document as strings, float values, URIs, and integer values of different precision. XML Cartridge keeps the basic contents of a document in specific leaf tables which use the atomic SQL data types provided by the underlying Oracle DBMS to store values, with one leaf table for every supported data type.

Whether this support for typed representations leads to an adequate storage of the basic contents of MPEG-7 media descriptions, however, is at least questionable. Apart from the fact that the data types supported by Lore, Natix, and XML Cartridge only constitute a limited subset of the simple types that are predefined with MPEG-7 DDL, all the systems do not support the simple type derivation methods coming with MPEG-7 DDL. Among others, they therefore cannot adequately represent lists and matrices commonly occurring in MPEG-7 media descriptions. Moreover, it remains unclear how Lore, Natix, and XML Cartridge are actually supposed to create typed representations, since they do not take schema definitions containing type information indispensible for that purpose into account for the storage of XML documents.

Out of the investigated database solutions, Oracle XML DB/Structured Mapping is the one providing yet the most comprehensive support for the typed representation of the basic contents of an MPEG-7 media description. According to the mapping scheme explained above, simple element content and the content of attribute values of an XML document are kept in fields of SQL object types which are assigned the atomic SQL data type best suiting the particular content type as given by the schema definition associated with the document.

Nevertheless, there are again considerable limitations. On the one hand, the mapping scheme does not support many of the simple type derivation methods provided with MPEG-7 DDL: among others, lists are simply mapped to text fields and the mapping of matrices is not supported at all since Oracle XML DB/Structured Mapping cannot cope with the extensions of MPEG-7 DDL to XML Schema. On the other hand, the typed representation of simple element content or the content of an attribute value depends on whether the respective element or attribute value is kept in an overflow store or not, because it occurs in an element with mixed or arbitrary content or in an element which employs polymorphism based on the `xsi:type` attribute. Both factors are likely to result in the textual representation of large parts of non-textual data carried by MPEG-7 media descriptions.

5.2 Access to media descriptions

Fine-grained access. With the exception of TEXTML which allows the retrieval of complete XML documents only, the XML database solutions covered by our analysis provide applications with a variety of means that give applications fine-grained access to the constituents of MPEG-7 media descriptions. Many solutions offer navigational programming interfaces which can be used to program fine-grained traversals of the structure of media descriptions. The native solutions eXcelon XIS, Infonyte DB, PDOM, and X-Hive/DB as well as the database extensions Oracle XML DB, ozone/XML, and Oracle XML DB/Structured Mapping all offer implementations of the DOM API. Natix comes with a file system driver that gives a file system view on the hierarchical structure of XML documents: document content can thus simply be traversed via file operations.

With XML database extensions that follow the structured storage and mapping approaches, applications can directly use query mechanisms available with the DBMS underlying the extension to access the content of media descriptions in a fine-grained manner. These extensions explicitly represent the content of XML documents with the modeling primitives of the DBMS. For example, the tables fine-grainedly representing XML document content with the relational database extensions Monet XML, Shimura et al., XML Cartridge, and Oracle XML DB/Structured Mapping can be directly queried with SQL.

Finally, most of the examined database solutions support some form of declarative XML query language facilitating fine-grained access to MPEG-7 media descriptions. The native solutions eXcelon XIS, Infonyte-DB, X-Hive/DB, Xindice, dbXML, eXist, and Natix are capable of evaluating XPath expressions. Tamino implements a proprietary, slightly extended variant of XPath expressions called X-Query, which should not be confused with the XQuery standardization effort, whereas Infonyte-DB and PDOM support XQL. Lore supports the powerful Lorel query language for semistructured data [1]. GoXML DB implements a subset of the current working draft of the XQuery standard.

The relational database extensions IBM DB2 XML Extender, Microsoft SQLXML, and Oracle XML DB following the unstructured storage approach are capable of evaluating XPath expressions on XML documents contained in a CLOB. For this purpose, they offer dedicated functions for use within SQL queries that take a CLOB containing an XML document and an XPath expression as arguments. When invoked, these functions internally load the whole XML document from the CLOB into main memory, parse it, evaluate the expression, and return the result of the evaluation as a string.

As the querying of XML documents stored with relational database extensions that follow the structured storage and mapping approaches by the means of SQL can be complex and cumbersome, the database extensions XML Cartridge, Shimura et al., and Oracle XML DB/Structured Mapping offer additional functions that take path expressions, XQL queries, and XPath expressions respectively as arguments and rewrite them to equivalent SQL statements.

Typed access. We have already pointed out that the investigated XML database solutions are weak in representing the basic contents of MPEG-7 media descriptions in an appropriate typed manner. Thus, it is no surprise that applications have difficulties in accessing these contents in a way adequate to the particular content type with these systems. As a matter of fact, those XML database solutions that store the basic contents of an XML document textually per se give applications textual access to basic document contents only. To access and process non-textual content in a way that is reasonable for the content type, applications must manually transform the content to appropriate typed representations everytime the content is accessed, e.g., by explicitly performing type conversions or by exploiting implicit type coercion rules associated with the operators of a query language such as XPath or XQuery.

Some of the XML database solutions that store the basic contents of a document textually facilitate a form of typed access to all those contents for which value indexes have been defined. For the definition of a value index on the content of elements or attribute values, the database solutions eXcelon XIS, Tamino, and TEXTML allow to specify whether the content is to be interpreted as strings, numbers, or dates for the purpose of indexing. Whenever an element or attribute value indexed in such a way is accessed in a query, it is treated according the data type specified.

This form of typed access, however, quickly meets its limitations with MPEG-7: the set of data types available for value indexing does not come even close to the broad variety of predefined simple types and simple type derivations methods available with MPEG-7 DDL. For instance, reasonable typed access to lists and matrices frequently occurring in media descriptions is not possible. Moreover, indexed content is still retrieved as text typically forcing applications to transform retrieved non-textual content to typed representations more appropriate for further internal processing. Finally, wide-spread typed access to the basic contents of a media description requires the definition of many value indexes which is likely to slow down update performance prohibitively.

Those XML database solutions that already store the basic contents of an XML document in typed representations generally can also offer applications typed access. But just as it is questionable how Lore, Natix, and XML Cartridge are supposed to obtain typed representations of the basic contents of XML documents without employing type information contained in schema definitions, it similarly remains unclear how these systems are supposed to realize typed access to these contents in practice.

The relational database extension Oracle XML DB/Structured Mapping, in contrast, facilitates typed access to the basic contents of an MPEG-7 media description within SQL queries: as we have explained before, it stores simple element content and the content of attribute values of XML documents in typed fields of SQL object types which are assigned the SQL data type coming closest to the simple type of the content specified in the schema definition. Since Oracle XML DB/Structured Mapping is capable of evaluating XPath expressions by rewriting them to equivalent SQL queries, the system supports typed access within XPath expressions as well.

However, the limitations of Oracle XML DB/Structured Mapping regarding the typed representation of the content of MPEG-7 media descriptions also apply to typed access. In particular, lists and matrices cannot be accessed in a typed manner because they are stored in textual fields. Similarly, any simple element content and the content of attribute values that is kept in textual overflow stores cannot be accessed in a typed way.

Fine-grained updates. Many of the XML database solutions investigated in our analysis not only provide mechanisms that give applications fine-grained read access to MPEG-7 media descriptions stored with them; in several cases, they also permit fine-grained updates of media descriptions. A common means offered by the examined systems for this purpose are navigational APIs allowing to program fine-grained updates of XML documents. The native database solutions eXcelon XIS, Infonyte-DB, PDOM, and X-Hive/DB as well as the database extensions ozone/XML and Oracle XML DB/Structured Mapping all implement the update-related methods of the DOM API; the file system interface offered by Natix allows fine-grained updates as well by the means of file operations.

Since database extensions following the structured storage and mapping approaches in detail represent the content of MPEG-7 media descriptions with the data model of the extended DBMS, they can consequently employ update mechanisms available with the DBMS to perform fine-grained modifications of media descriptions as well: the relational database extensions Monet XML, Shimura et al., XML Cartridge, and Oracle XML DB/Structured Mapping allow fine-grained updates of XML documents using suitable SQL UPDATE statements.

Apart from these update mechanisms, many native database solutions enable fine-grained updates of media descriptions via dedicated XML update languages. eXcelon XIS supports Updategrams; GoXML DB extends XQuery with update operations. The database solutions Xindice and dbXML offer implementations of the XUpdate language. Lore's declarative query language Lorel includes constructs for performing fine-grained updates as well. As fine-grained updates of XML documents may result in complex SQL UPDATE statements, the relational database extension Oracle XML DB/Structured Mapping offers a function that rewrites update requests specified in a simple XML update language that is based on XPath expressions to semantically equivalent UPDATE statements.

In contrast, the native database solutions Tamino, TEXTML, and eXist do not allow fine-grained updates of MPEG-7 media descriptions. With these systems, an update of even a small fraction of an XML document can only be performed by replacing the complete document with an updated version. This is also the case with the IBM DB2 XML Extender, Microsoft SQLXML, and Oracle XML DB database extensions that unstructuredly store XML documents in CLOBs.

Value index structures. The support for value indexing offered by the XML database solutions considered in our analysis is not sufficient for the reasonable indexing of the basic contents of MPEG-7 media descriptions. The native XML database solutions we looked at typically come with ordered, B-Tree-based one-dimensional value index structures that can be used for the indexing of simple element content and the content of attribute values of XML documents. The only exceptions to this are Infonyte-DB[3], PDOM, eXist, and Natix which do

[3]A B-Tree index structure is currently implemented for Infonyte-DB, but efforts are still beta

not provide value index structures at all. For effective indexing of MPEG-7 media descriptions, however, the capabilities of the native database solutions are limited: none of the examined native solutions offers multidimensional index structures such as R-Trees that are certainly desirable for the indexing of complex multimedia data often contained in media descriptions.

XML database extensions can rely on value index structures already offered by the extended DBMS for the indexing of the basic contents of MPEG-7 media descriptions. Database extensions that implement the structured storage and mapping approaches for the representation of XML documents can directly use these index structures, because they explicitly represent XML documents and their basic contents with the modeling primitives of the underlying DBMS. Among the investigated database extensions, this solely does not apply to ozone/XML as the DBMS ozone does not offer any value index structures.

Regarding the database extensions allowing the unstructured storage of XML documents, the question comes up how the value index structures of the extended DBMS can be applied to the basic contents of a media description buried inside a CLOB. In fact, this is not possible with Microsoft SQLXML at all. IBM DB2 XML Extender allows to map the content of an element or an attribute value of an XML document contained in a CLOB to a column of a so-called side table with the help of XPath expressions. This column can then be indexed with DB2's value index structures just as any other column. IBM DB2 XML Extender automatically keeps documents and side tables synchronized. Finally, Oracle XML DB allows to index the basic contents of an XML document stored in a CLOB with functional indexes which employ the functions for XPath evaluation provided by the system.

In contrast to the examined native database solutions, some of the extended DBMSs not only provide one-dimensional value index structures such as B-Trees but also multidimensional value index structures that can, in principle, be exploited by the XML database extensions for the effective indexing of complex multimedia data carried in MPEG-7 media descriptions. There are grid index and R-Tree index structures available for IBM DB2 and Oracle as well as for PostgreSQL [30, 51, 64].

But it is questionable whether the multidimensional index structures and the database extensions fit together. The grid index and R-Tree index structures of IBM DB2 and Oracle can only index table columns which are assigned specific geometric data types for the representation of two-dimensional shapes. It is unclear, how instances of these geometric data types can be obtained from complex multimedia data such as melody contours contained in MPEG-7 media descriptions stored with one of the XML database extensions based on IBM DB2 and Oracle so that the data can be indexed effectively.

PostgreSQL comes with an R-Tree index structure that can index columns of any data type as long as the data type implements a specific set of operations. However, the XML database extension Shimura et al. that is based on PostgreSQL cannot benefit from this index structure: Shimura et al. stores all simple element content and the content of attribute values as text in one single string column. It is doubtable that the set of operations required by the R-Tree index structure can be implemented for strings in such a way that the content of all elements and attribute values is reasonably indexable.

Text index structures. Quite a few XML database solutions offer dedicated text index structures that can be exploited by applications for the realization of efficient text retrieval on MPEG-7 media descriptions. Among the native database solutions, X-Hive/DB provides an index structure allowing the full-text indexing of entire XML documents while eXcelon XIS, GoXML DB, Tamino, TEXTML, eXist, and Lore possess index structures allowing to textually index the content of single elements and attribute values at any level in an XML document. The native solutions Infonyte-DB, PDOM, Xindice, dbXML, and Natix, in contrast, do not offer any special structures for text indexing.

As with value index structures, XML database extensions can benefit from existing text index structures of the underlying DBMS for the realization of efficient text retrieval on MPEG-7 media descriptions. As there are text index structures available for the relational DBMSs IBM DB2, Microsoft SQL Server, and Oracle that can be applied to texts stored in CLOBs and character columns [31, 48, 46], the XML database extensions IBM DB2 XML Extender, Microsoft SQLXML, and Oracle XML DB that store XML documents in their original textual format in CLOBs can directly employ these structures for the full-text indexing of entire XML documents. The database extensions XML Cartridge and Oracle XML DB/Structured Mapping founding on the DBMS Oracle can use the available text index structure as well to index the basic contents of XML documents: the text index structure can be applied to the corresponding leaf table columns and SQL object type fields in which the extensions store textual simple element content and attribute values in a fine-grained fashion.

The database extensions ozone/XML, Monet XML, and Shimura et al., however, do not support text retrieval with dedicated index structures: they do not implement text index structures themselves and the underlying DBMSs ozone, Monet, and PostgreSQL do not provide any special text index structures either.

Path index structures. Several native XML database solutions and XML database extensions offer index structures that can accelerate the traversal of frequently followed access paths in MPEG-7 media descriptions in miscellaneous ways. With the native solution eXcelon XIS, a so-called structure index can be defined for a given XPath expression. The structure index precomputes and maintains the results of the evaluation of that expression on all documents in a database. Whenever a query includes the indexed XPath expression, the query processor of eXcelon XIS makes use of the precomputed evaluation results instead of accessing all documents and evaluating the expression itself.

The native database solution Tamino also features a structure index that is, however, different from the one provided by eXcelon XIS. Tamino's structure index associates every XML document stored in a database with all access paths which lead from the document root to the individual elements and attribute values of the document. With the structure index, Tamino's query processor can reduce the number of documents accessed during query evaluation: those documents can be filtered out which cannot contribute to the result of a query because none of their elements or attribute values can be reached via an access path that suits the query.

Infonyte-DB and PDOM maintain so-called signature indexes along with XML documents. Quite similar in idea to Tamino's structure index, a signature index keeps signatures for all access paths from the root of a document to the document's elements and attribute values. When traversing an XML document for the evaluation of an XPath expression, both systems exploit the signature index to efficiently identify and prune subtrees of the document which do not contain nodes that could contribute to the result of the expression.

X-Hive/DB offers element name indexes to support path traversal. An element name index collects all elements of a given element type that occur in a document contained in the database.

Whenever an XPath expression refers to an element type for which an element name index exists, the query processor exploits the name index to directly obtain the elements of that type instead of accessing and traversing all documents to seek out these elements.

The research prototype Lore provides DataGuides for path indexing [25]. A DataGuide is a graph that constitutes a concise structural summary of all access paths that are possible in the XML documents contained in a database. Based on a DataGuide, Lore can not only decide whether a path traversal can yield a result for any of the documents contained in a database. Also, as a DataGuide references all elements or attribute values which can be reached by traversing a given path, Lore can perform path traversals to a large extent on the DataGuide itself avoiding the need to access the documents in the database.

Among the XML database extensions that follow the unstructured storage approach that keeps XML documents in a CLOB, only Oracle XML DB offers means for path indexing. The full-text index structure offered by Oracle has been extended to help with the evaluation of XPath expressions as well. The index structure can filter out XML documents for which an XPath expression cannot yield a result. This avoids unnecessary loading, parsing, and traversal of such documents for the evaluation of an XPath expression.

The examined database extensions for the structured storage of XML documents all offer index support for path traversal. Shimura et al. and XML Cartridge keep path tables along with the edge tables. An entry in the path table augments an element or attribute value stored in the edge table with the access path by which it can be reached from the root of the document. The access path is encoded as a string consisting of the concatenated names of the elements which are traversed on the path. Hence, elements and attribute values potentially qualifying for a given path expression can be found by performing appropriate string matching operations on the path table instead of traversing all documents for the evaluation of the expression. Monet XML supports path indexing by construction. As we have explained before, Monet XML keeps the elements and attribute values of an XML document in tables specifically created for the access paths by which the respective element or attribute value can be reached. The traversal of an access path thus essentially consists of selecting the rows of the table representing the access path.

5.3 Media description schemes

MPEG-7-DDL-compliant schema catalog. None of the XML database solutions considered in our analysis provides a schema catalog that is fully compliant to MPEG-7 DDL. In fact, the majority of the investigated solutions – namely Infonyte-DB, PDOM, TEXTML, Xindice, dbXML, eXist, Lore[4], Natix, Microsoft SQLXML, ozone/XML, Monet XML, Shimura et al., and XML Cartridge – do not maintain schema catalogs at all; they do not make use of available schema information for the storage of XML documents.

Other database solutions, such as X-Hive/DB and IBM DB2 XML Extender, maintain a DTD-based schema catalog for the documents stored in a database. However, DTD-based schema catalogs in general cannot be considered appropriate for managing MPEG-7 media description schemes: MPEG-7 DDL is far more expressive than the DTD language and MPEG-7 media description schemes typically make extensive use of the advanced constructs of MPEG-7 DDL not expressible with DTDs, such as complex type derivation and typing of simple content. Therefore, it can be doubted that MPEG-7 media description schemes can be reasonably translated to DTDs.

As XML Schema constitutes a large subset of MPEG-7 DDL, database solutions that implement XML-Schema-compliant schema catalogs promise at least partial support for the management of MPEG-7 media description schemes. eXcelon XIS, Oracle XML DB, and Oracle XML DB/Structured Mapping provide schema catalogs that more or less support the XML Schema standard and therefore can be expected to cope with many MPEG-7 media description schemes. In contrast, the schema catalogs of Tamino and GoXML DB implement very restricted subsets of XML Schema only, significantly limiting their ability to manage typical MPEG-7 media description schemes. Just to name a few restrictions on constructs commonly occurring in media description schemes, Tamino and GoXML DB both do not support globally visible named complex types and complex type derivation.

[4]Lore maintains a DataGuide for the schematic description of the XML documents contained in a database. However, a DataGuide is merely dynamically generated summary of the content of these documents. It does not serve to define and restrict content type and structure as it is the aim of media description schemes. Thus, Lore does not maintain a schema catalog in terms of our discussion.

Validation of media descriptions. All examined database solutions that maintain a schema catalog for the XML documents contained in a database employ the schema information available with the catalog for document validation. Up to the respective degree of MPEG-7 DDL / XML Schema supported by the schema catalog, the database solutions eXcelon XIS, GoXML DB, Tamino, Oracle XML DB, and Oracle XML DB/Structured Mapping are thus able to validate MPEG-7 media descriptions as well.

With these solutions, the validation of XML documents against their schema definition automatically takes place during document import. Hence, only MPEG-7 media descriptions that are consistent with their media description schemes can be inserted into a database. As Tamino and Oracle XML DB support coarse-grained updates of XML documents only and, therefore, the update of a document is equivalent to the import of a new document into the database with these systems, both solutions also validate documents automatically after an update. This ensures that no update of an MPEG-7 media description stored with these solutions can violate its media description scheme.

eXcelon XIS, GoXML DB, and Oracle XML DB/Structured Mapping, in contrast, do not validate a document after an update automatically. Thus, update operations can result in media descriptions inconsistent with their description scheme. However, they allow applications to explicitly initiate document validation anytime.

Inference of typed representations. Out of all investigated database solutions, only Oracle XML DB/Structured Mapping can to some extent utilize type information available in the schema catalog to infer typed representations of the basic contents of MPEG-7 media descriptions. As we have explained before, the mapping scheme of Oracle XML DB/Structured Mapping translates a schema definition written in XML Schema to a relational database schema consisting of a series of SQL object types which have appropriately typed fields to store simple element content and the content of attribute values. When mapping an XML document to a relational schema created in such a manner, the document's basic contents are automatically brought from textual to typed representation when assigned the typed field specifically prepared for the respective content. Nevertheless, the ability of Oracle XML DB/Structured Mapping to produce typed representations for MPEG-7 media descriptions is limited: recall, for instance, that list and matrices are generally assigned to textual fields and that elements with mixed and arbitrary content as well as elements using the `xsi:type` attribute for polymorphism are kept in textual overflow stores.

All the other XML database solutions that to some extent support the typed representation of simple element content and the content of attribute values, namely, Lore, Natix, and XML Cartridge, do not maintain schema catalogs. As they therefore do not have necessary type information at their disposal, it remains unclear how they are supposed to infer appropriate typed representations of the basic contents of a document.

Query optimization. The analyzed XML database solutions, native database solutions as well as database extensions, are particularly weak when it comes to the use of available schema information contained in media description schemes for query optimization. Even though many database solutions come with index structures that can considerably accelerate query evaluation, applications typically have to formulate queries in such a way that it is clear to the solution's query processor that an index is to be used for query evaluation. Rarely, query processors can equivalently rewrite queries based on the schema definition of an XML document such that relevant indexes are found and applied automatically. Moreover, most database solutions do not employ schema information to reduce the complexity of queries by identifying and cutting off subexpressions that are redundant or that can never yield a result. At best, they feature some simple, schema-independent heuristics for query simplification.

Out of the examined solutions that maintain a schema catalog, only Tamino and Oracle XML DB/Structured Mapping offer more sophisticated techniques for query optimization based on the schema definition of an XML document. Tamino employs schema information to bring an X-Query query into a canonical form. Among others, path expressions (which may include wildcards and traversals to indirect child nodes) occurring in a query are replaced in the canonical form by the disjunction of all those paths that qualify for these expressions according to the schema definition. Based on this canonical form, it is straightforward for the query processor to decide which value or text indexes are available for query evaluation and to apply the structure index to filter out documents which cannot contribute to the query result.

Oracle XML DB/Structured Mapping employs the schema definition to rewrite an XPath expression that is to be evaluated on an XML document to an equivalent SQL query that operates on the tables and SQL object types of the relational schema specifically created for the schema definition. The SQL query is then passed on to the query processor of the underlying Oracle DBMS which can then apply all the sophisticated relational query optimization techniques that are available with the system.

5.4 Extensibility

Extensibility with functionality. The examined categories of XML database solutions, native solutions and database extensions, differ considerably with regard to their extensibility with application-specific functionality. Most native database solutions, i.e., Infonyte-DB, PDOM, X-Hive/DB, eXist, Lore, and Natix, do not provide means with which new, custom functions and procedures can be integrated into these systems. And even though the native solution GoXML DB employs XQuery for accessing XML documents, the system is not extensible with functionality as well: in the current Version 2.0.2, XQuery's support for the definition of custom functions has apparently not been implemented.

The only native solutions that can be extended with application-specific functions are eXcelon XIS, Xindice, dbXML, and Tamino. Such a functional extension is called DXE Servlet with eXcelon XIS, XMLObject with Xindice and dbXML, and query function with Tamino. DXE Servlets, XMLObjects, and query functions are realized as Java classes; optionally, a Tamino query function can also be implemented as a COM class. Though very similar at first glance, the concepts of DXE Servlets, XML Objects, and query functions differ considerably in the way how seamless they can be applied for the processing of XML documents contained in a database: query functions can be used in Tamino's X-Query language just like any of the built-in functions whereas both DXE Servlets as well as XMLObjects cannot be invoked from XPath expressions which are supported by eXcelon XIS, Xindice, and dbXML. Instead, DXE Servlets and XML Objects have to be called via dedicated APIs.

The examined XML database extensions, in contrast, can all be augmented with custom functions and procedures using the mechanisms for functional extension available with the underlying DBMS. With the object-oriented DBMS ozone which is the basis of the database extension ozone/XML, for example, arbitrary persistence-capable classes can be defined providing methods that realize application-specific functionality in addition to the classes of the ozone/XML library. The relational DBMS research prototype Monet, which serves as the basis of the relational extension Monet XML, has a modular microkernel architecture that allows the integration of arbitrary application-specific operators and functions into the system. The relational DBMSs IBM DB2, Microsoft SQL Server, Oracle, and PostgreSQL, which are the foundation of the other relational XML database extensions investigated in our analysis, all support the definition of custom stored procedures and functions that are usually implemented in vendor-specific stored procedure languages such as PL/SQL or in external programming languages like C or Java. These custom procedures and functions perfectly fit together with the XML database extensions on the level of SQL where they can be used in queries for XML documents just like built-in functions. However, they typically cannot be used within XML query languages which are often additionally supported by relational database extensions. For example, an application-specific stored function cannot be called within an XPath expression that is to be executed with the XPath evaluation function on an XML document stored in a CLOB with Oracle XML DB.

Extensibility with index structures. Most of the XML database solutions considered in our analysis do not offer dedicated interfaces with which new index structures can be integrated if desired by an application. Instead, database solutions usually present themselves to application developers as monolithic systems. To integrate a new index structure into such a system, there are basically two rather unsatisfactory options: on the one hand, the new index structure can be directly incorporated into the system's kernel. Apart from the fact that this requires the source code of the database solution (which is often not available, at least for commercial solutions), it is a demanding and costly endeavor to become acquainted with and change the source code of a database system. On the other hand, one could extend the system with functions – if it is extensible with functionality – for constructing, maintaining, and querying the new index structure which then have to be explicitly invoked by applications. The index structure itself could either be stored just like other application data in the database or outside the database in the filesystem. This approach, however, not only completely hides the existence of the index structure from the database solution so that, for example, it cannot be considered for query optimization; also, indexing is not transparent to applications.

Out of the investigated native XML database solutions, only X-Hive/DB offers an open interface for the integration of new index structures into the system. However, this interface is limited to the integration of full-text index structures only.

For XML database extensions, the question of integrating a new index structure comes down to the question whether the underlying DBMS is extensible with new index structures. Among the DBMSs IBM DB2, Microsoft SQL Server, Oracle, ozone, Monet, and PostgreSQL which form the basis of the examined XML database solutions, only Oracle and Monet can be extended with index structures without the need to change and rebuild the system's kernel. Oracle can dynamically integrate new index structures into the systems via the Extensible Indexing API; Monet's microkernel has been specifically designed to be dynamically extensible with so-called extension modules that can, among others, contain application-specific index structures. As a result, the investigated XML database solutions that base on Oracle and Monet, i.e., Oracle XML DB, Monet XML, XML Cartridge, and Oracle XML DB/Structured Mapping, can be extended with new index structures as well.

5.5 Classic DBMS functionality

Transactions. Even though transactions are a central concept of traditional DBMSs, surprisingly many of the examined native XML database solutions do not offer transaction support. Only eXcelon XIS, GoXML DB, Tamino, X-Hive/DB, and Lore support standard transactions guaranteeing the ACID properties for access to MPEG-7 media descriptions stored in a database.

The native solutions Infonyte-DB, PDOM, TEXTML, Xindice, dbXML, eXist, and Natix, in contrast, do not offer applications transaction support. Infonyte-DB and PDOM, being designed as lightweight storage options for XML documents, allow a database to be accessed by a single process only. Thus, an isolated view is ensured on the process level without the need for transactions.

The native database solution TEXTML, originating from the domain of document management systems, implements a check-out mechanism for XML documents instead of transactions. Whenever an application needs to modify an MPEG-7 media description stored in a TEXTML database, it must explicitly check out the description, download it to the local filesystem, change it there, and finally check it in back to the database. A description may only be checked out once. Unlike transactions, this mechanism does not make concurrent access transparent to applications: it is the responsibility of the applications to get all MPEG-7 media descriptions which they intend to modify checked out and to resolve any conflicts in this regard among each other.

The remaining native database solutions Xindice, dbXML, eXist, and Natix, do not provide any means to shield applications from each other. Concurrent access to MPEG-7 media descriptions stored with these solutions is performed in indeterministic order and every update of a description is immediately visible to all applications.

The XML database extensions that we have taken into account for our analysis base on conventional relational and object-oriented DBMSs that all offer support for classic ACID transactions. Naturally, this transaction support is available for access to MPEG-7 media descriptions stored with one of these XML database extensions as well.

Fine-grained concurrency control. The granularity of the concurrency control mechanisms provided by the investigated XML database solutions varies considerably. Among the native database solutions, only eXcelon XIS, GoXML DB, and Lore exert fine-grained concurrency control: these solutions are capable of locking the single elements and attribute values of an XML document thereby allowing different transactions to concurrently access and modify different parts of one and the same MPEG-7 media description that is stored by them.

Compared to these systems, the concurrency control mechanisms of Tamino, X-Hive/DB, Infonyte-DB, and PDOM are rather coarse-grained. Tamino performs concurrency control by placing locks on the document level. This has the effect that two transactions may not concurrently read and modify one and the same MPEG-7 media description that is managed with Tamino. In order to change a media description that is managed with X-Hive/DB, the system's concurrency control mechanism requires transactions to obtain a write lock on the entire database. Thus, a transaction may not access any media description in a database whenever a description is currently updated by another transaction. The concurrency controls of Infonyte-DB and PDOM lock the entire database as soon as it is accessed by a process with the effect that only one application can access a database at a time.

The already mentioned check-out mechanism for XML documents offered by TEXTML also constitutes a form of coarse-grained concurrency control: when an MPEG-7 media description stored with TEXTML is checked out in order to be updated by an application, a lock is acquired on the entire description. While the media description can still be accessed for reading, this lock prevents other applications from checking out the same description.

The native database solutions Xindice, dbXML, eXist, and Natix, as we have already pointed out before, do not perform any concurrency control at all.

The conventional DBMSs which are the foundation of the XML database extensions examined in our analysis are typically able to conduct fine-grained concurrency control. The object-oriented DBMS ozone performs concurrency control by locking the single objects of which a database consists whereas the relational DBMSs IBM DB2, Microsoft SQL Server, Oracle, and PostgreSQL can be configured to exert concurrency control by locking the single rows of the tables in a database.

This in principle implies that fine-grained concurrency control for access to MPEG-7 media descriptions managed with the XML database extensions ozone/XML, Shimura et al., XML Cartridge, and Oracle XML DB/Structured Mapping can be realized as well: as these extensions follow the structured storage and mapping approaches and therefore represent the content of XML documents in a fine-grained manner using the modeling primitives available with the underlying DBMS, the locking of a single object or a row typically corresponds to the locking of a single element or attribute value of a media description. The locking granularity of Oracle XML DB/Structured Mapping, however, is compromised by the fact that significant portions of an MPEG-7 media description can be expected to be kept unstructuredly in textual overflow stores. Being merely a CLOB, Oracle can lock such an overflow store in its entirety only, and with it the whole fraction of the media description contained.

Even though the database extension Monet XML follows the structured storage approach as well, concurrency control for access to MPEG-7 media descriptions stored with Monet XML does not show the fine granularity of the database extensions mentioned above. This is due to the fact that the relational research DBMS prototype Monet performs concurrency control by locking the single tables of a database. Given the storage scheme for XML documents applied by Monet XML, a lock on a table does not correspond to a lock on a single element or attribute value of a media description; rather, it corresponds to a lock on all elements or attribute values contained in the descriptions stored in the database which can be reached by the access path that is represented by the table.

Finally, fine-grained concurrency control is not possible with the examined XML database extensions that follow the unstructured storage approach. All the DBMSs IBM DB2, Microsoft SQL Server, and Oracle forming the basis of these extensions are capable of locking complete CLOBs only, and not parts of CLOBs. As a consequence, an MPEG-7 media description stored in a CLOB with IBM DB2 XML Extender, Oracle XML DB, or Microsoft SQLXML is always locked as a whole resulting in a coarse-grained concurrency control on the description level.

Fine-grained access control. The investigated XML database solutions differ remarkably in the way they are able to control the access of different users to MPEG-7 media descriptions that they manage. The native solutions Infonyte-DB, PDOM, Xindice, dbXML, eXist, Lore, and Natix do not implement access control at all; any user can access any MPEG-7 media description stored with one of these systems. eXcelon XIS, TEXTML, and X-Hive/DB perform user authentication when a database connection is being established. Once user authentication has succeeded, however, every media description stored in the database is fully accessible. GoXML DB provides an access control mechanism that goes a step further and allows to restrict the access rights of users to single media descriptions.

While so far access control of the examined native database solutions (if at all existent) can only be regared as coarse-grained, the native solution Tamino offers a fine-grained access control mechanism allowing to regulate access rights down to the single elements of an MPEG-7 media description. For every element type declared in a schema definition kept by Tamino's schema catalog, the specific rights can be defined that the different users of the system have for accessing the elements of this type in the XML documents conforming to the schema definition.

With the exception of Monet which, to our knowledge, does not provide an access control mechanism and thus allows any user to access any media description stored with Monet XML, the conventional DBMSs which serve as the foundation of the XML database extensions of our analysis all offer means for fine-grained access control. The object-oriented DBMS ozone allows to specify access rights on the level of single objects; the relational DBMSs IBM DB2, Microsoft SQL Server, Oracle, and PostgreSQL allow to regulate access rights for the individual tables of a database.

Whether the fine-grained access control mechanisms of these systems also provide fine-grained access control to MPEG-7 media descriptions stored with an XML database extension highly depends on the particular technique applied by the extension for the representation of XML document content. As the XML database extensions IBM DB2 XML Extender, Microsoft SQLXML, and Oracle XML DB follow the unstructured storage approach and thus keep MPEG-7 media descriptions in their entirety in CLOBs in table columns, the specification of access rights on the table level effectively regulates the access to complete descriptions only. Hence, only coarse-grained access control is available with these extensions.

The database extension ozone/XML which follows the structured storage approach for XML documents permits fine-grained access control to the content of an MPEG-7 media description, because it stores the single elements and attribute values of an XML document as individual objects of the corresponding classes that come with the persistent class library implementing the DOM standard. Specifying access rights on an object therefore corresponds to specification of access rights for a single element or attribute value.

Even though the database extensions Shimura et al. and XML Cartridge follow the structured storage approach as well, they do not support fine-grained access control to the content of a media description. Both extensions store the elements and attribute values of all XML documents in a central edge table. Modifying the access rights for the edge table therefore affects the access to all elements and attribute values of all documents.

Finally, Oracle XML DB/Structured Mapping basically permits fine-grained access control to XML documents. Because the underlying DBMS Oracle allows the definition of access rights for SQL object types, this feature can also be used for the specification of access rights for those SQL object types that are generated by the mapping scheme of Oracle XML DB/Structured Mapping to represent the content of XML documents in a fine-grained fashion. The granularity of access control is reduced, however, if significant portions of an XML document are kept in textual overflow stores in an unstructured manner – as it is the case with MPEG-7 media descriptions. Since an overflow store is basically a CLOB, fine-grained access control to the elements and attribute values contained in such overflow stores is not possible.

Backup and recovery. The analyzed XML database solutions make different provisions to protect MPEG-7 media descriptions and other XML documents that they manage from imponderabilities such as system crashes and media failures. The native database solutions Xindice, dbXML, and Natix do not take any special precautions at all. They rely on the database administrator to regularly shut down the system and to backup the hopefully consistent database files to a safe place via file copy.

GoXML DB, Infonyte-DB, PDOM, and Lore rely on the database administrator to manually shut down the system and to copy the database files for backup as well but additionally maintain a write-ahead log of the changes to the database content. In case of a system crash which leaves the database in an inconsistent state, they can use the log to recover the database back to a recent, consistent state and do not need to fall back on the state of the last backup.

The native solutions eXcelon XIS, Tamino, and X-Hive/DB not only maintain write-ahead logs; they also come with dedicated backup tools that allow to backup a consistent snapshot of the database content without the need to shut down the system. eXcelon XIS and Tamino support only full backups of the database content whereas X-Hive/DB additionally supports incremental backups of the changes since the last backup. This significantly reduces the storage space and time required for the backup of large databases.

The native solution TEXTML comes with a backup tool allowing full backups of the database content without the need to shut down the system as well. However, TEXTML does not log the changes to the database content. In case of a failure leaving the database in an inconsistent state, one can only fall back on the state of the last backup.

The investigated XML database extensions, as well as the native solution eXist which internally employs the relational DBMSs PostgreSQL or MySQL as the storage backend, can make use of the backup and recovery mechanisms available with the underlying DBMSs. The open source systems PostgreSQL, ozone, and the research prototype Monet which form the basis of the database solutions eXist, ozone/XML, Monet XML, and Shimura et al., maintain write-ahead logs and also come with tools allowing full backups of the database content.

Being industrial strength DBMSs targeted at mission-critical enterprise applications, IBM

DB2, Oracle, and Microsoft SQL Server offer high-capacity backup and recovery mechanisms from which the XML database extensions IBM DB2 XML Extender, Microsoft SQLXML, Oracle XML DB, XML Cartridge, and Oracle XML DB/Structured Mapping can benefit. With these DBMSs, logging database changes and tools allowing the implementation of complex incremental backup strategies for large databases are a matter of course.

Versioning. Among the XML database solutions covered by our analysis, only X-Hive/DB, Oracle XML DB, and Oracle XML DB/Structured Mapping feature ready-to-use versioning mechanisms that can directly be employed by applications to establish a version management for MPEG-7 media descriptions. When storing MPEG-7 media descriptions with the other database solutions, an update of a media description automatically destroys the previous version of the description if an application does not implement versioning functionality itself on top of these solutions.

With the native database solution X-Hive/DB, an XML document stored in the database can be marked as versionable. Once this has been done, the document cannot be modified via the DOM API anymore; instead, a check-out mechanism similar to the one of TEXTML applies. Whenever a previously checked out XML document is checked in back to the database, a new version of the document is created. As with the check-out mechanism new versions of an XML document can be derived from older versions of the document and not just from the newest one, it is possible to establish a version tree for a document. Hence, X-Hive/DB supports branched versioning. X-Hive/DB provides an API with which applications can navigate between the different versions of a document.

The relational database extensions Oracle XML DB and Oracle XML DB/Structured Mapping come with an add-on called Oracle XML DB Repository which allows to organize XML documents hierarchically in a folder structure. Apart from SQL, applications can access these folders via a variety of protocols such as FTP, HTTP, and WebDAV. Similar to X-Hive/DB, Oracle XML DB Repository allows an XML document to be marked as versionable and provides a check-out mechanism with which new versions of a document can be created. Just like X-Hive/DB, Oracle XML DB Repository supports branched versioning. Oracle XML DB Repository comes with several functions for traversing the predecessor and successor relationships between the different versions of a document from within SQL queries.

5.6 Summary

Summarizing we can say that none of the investigated XML database solutions suffices all requirements for the management of MPEG-7 media descriptions. Even though, if we take a look back at Figure 4, there are database extensions such as Oracle XML DB/Structured Mapping and native database solutions like eXcelon XIS, Tamino, and X-Hive/DB which cover quite a lot of these requirements, this should not belie the substantial limitations that are seriously compromising their eligibility for the management of MPEG-7 media descriptions.

The main weakness of the examined solutions is that they store and treat simple element content and the content of attribute values of MPEG-7 media descriptions largely as text, regardless of the particular content type. This is unsatisfactory because of the large fractions of media descriptions consisting of non-textual data like numbers, vectors, and matrices making up technical characteristics of media such as melody contours; applications will definitely want to access and process these data according to their type and not as text.

The source of this weakness is the fact that the inspected solutions do not sufficiently make use of schema and type information available with media description schemes for the management of MPEG-7 media descriptions. As a result, the systems often lack valuable data crucial not only for ensuring the consistency of database content by validating media descriptions and for the optimization of queries, but also for inferring the types of the basic contents of a media description. The majority of inspected database solutions – like Infonyte-DB, TEXTML, Xindice, Microsoft SQLXML, Monet XML, and Shimura et al. to cite some representative examples – totally ignore schema definitions for the storage of XML documents. And even those database solutions that maintain schema catalogs – such as eXcelon XIS, Tamino, GoXML DB, X-Hive/DB, IBM DB2 XML Extender, and Oracle XML DB – primarily use the information contained therein for the validation of XML documents only. Moreover, the schema catalogs just support DTDs or more or less XML Schema; none of the solutions fully supports MPEG-7 DDL.

Among all XML database solutions covered by our analysis, solely the relational database extension Oracle XML DB/Structured Mapping in principle constitutes a step into the right direction and makes use of schema definitions written in XML Schema to manage simple element content and the content of attribute values in a typed fashion. However, as we have set out in detail before, the system nevertheless represents a very high percentage of the constituents of typical MPEG-7 media descriptions as text due to inherent limitations of its mapping scheme, e.g., lists, matrices, and elements making use of polymorphism via the `xsi:type` attribute.

In addition to the issue of typed representations, there are further aspects that constrain the applicability of the examined XML database solutions for the management of MPEG-7 media descriptions. The value indexing support offered by the database solutions is generally not sufficient. At best, the analyzed solutions offer one-dimensional, B-Tree-based index structures for the indexing of the basic contents of XML documents. However, no solution supports multidimensional index structures such as R-Trees for the indexing of document content. This notably obfuscates the prospects of successfully implementing efficient multimedia retrieval applications on large collections of MPEG-7 media descriptions with these systems.

When applying existing XML database solutions for the management of MPEG-7 media descriptions – native database solutions as well as database extensions – one should also be aware of the fact that rather basic DBMS functionality such as fine-grained currency control and fine-grained access control cannot be taken for granted, not to speak of more advanced issues such as the versioning of media descriptions. While several solutions, e.g., Tamino, X-Hive/DB, ozone/XML, and Oracle XML DB/Structured Mapping, offer support for some of these aspects, none of the investigated solutions satisfyingly covers all.

Disregard of these general weaknesses of the investigated XML database solutions: what are individual strengths and weaknesses of native database solutions and database extensions concerning the management of MPEG-7 media descriptions?

Native XML database solutions have been designed for the sole purpose of efficiently storing XML documents. Hence, it is no surprise that they are typically strong with regard to the representation of and access to MPEG-7 media descriptions. Taking a look at Figure 4, we can see that native database solutions normally store MPEG-7 media descriptions with a fine granularity. Usually backing these fine-grained storage representations with an array of path, value, and text index structures, they also come with navigational APIs and dedicated XML query languages, in many cases not only allowing fine-grained access to but also fine-grained updates of media descriptions.

However, native database solutions are rather weak when it comes to extensibility and classic DBMS functionality. Only a few of the examined native solutions – namely eXcelon XIS, Tamino, Xindice, and dbXML – can be extended with custom functions and procedures. With respect to the extensibility with index structures, the situation is even worse: just X-Hive/DB allows the integration of new text index structures into the system. Concerning classic DBMS functionality, our analysis has also unveiled significant deficiencies of many native solutions. Several native solutions, i.e., Infonyte-DB, PDOM, TEXTML, Xindice, dbXML, eXist, and Natix, do not implement transaction support; some of them are not even laid out for concurrent access and do not conduct any concurrency control at all. Furthermore considering that X-Hive/DB is the only native solution that supports incremental backups, we must also come to the conclusion that the means for backup and recovery of the native database solutions are not very mature yet.

Looking at Figure 4, we find that XML database extensions offer a more heterogeneous picture compared to native database solutions with regard to the representation of and access to MPEG-7 media descriptions. It highly depends on the particular storage approach followed by an extension with which granularity MPEG-7 media descriptions are represented, whether and by what means fine-grained access and updates of media descriptions are possible, and whether and which index structures of the underlying DBMS can be exploited for the indexing of media descriptions as well. In these respects, database extensions that follow the unstructured storage approach like IBM DB2 XML Extender and Microsoft SQLXML perform rather weak whereas several of the database extensions that follow the structured storage and mapping approaches such as XML Cartridge and Oracle XML DB/Structured Mapping are comparable to native database solutions.

The main strength of XML database extensions is that they found on long established traditional database technology. Consequently, they can benefit from the mature implementations of classic DBMS functionality of the underlying DBMSs. XML database extensions are therefore especially strong with regard to transaction support and backup and recovery. Moreover, since it has been a trend in the recent years to make traditional DBMS more extensible, most DBMSs offer mechanisms that allow the integration of new functionality and, as it is the case with Monet and Oracle, new index structures into the system. As XML database extensions can directly employ these extension mechanisms for their own purposes, they are generally strong with respect to extensibility.

6 Conclusion

In this paper, we have investigated the suitability of existing XML database solutions for the management of MPEG-7 media descriptions. Outgoing from a brief observation of the key characteristics of typical MPEG-7 media descriptions, we have derived an extensive set of requirements that should be met by an MPEG-7 database solution. Along these requirements, we have thoroughly analyzed a variety of current, state-of-the-art native XML database solutions and XML extensions for traditional DBMSs – commercial systems, open source projects, as well as research prototypes.

With our analysis, we have reveiled fundamental deficiencies of the investigated database solutions. The focus of our criticism is that the solutions largely neglect valuable schema and type information available with media description schemes written in MPEG-7 DDL for the management of MPEG-7 media descriptions, with considerable negative consequences regarding the ability of the systems to validate media descriptions against their description schemes, to represent non-textual content contributing significantly to typical MPEG-7 media descriptions in a manner appropriate to the content type and not just as text, to give applications reasonable typed access to such non-textual contents, and to perform more sophisticated query optimizations.

For the management of MPEG-7 media descriptions (and certainly for the management of other data-centric XML documents as well, for instance, in the domain of electronic interchange of business data), we therefore see the need for a new generation of XML database solutions which recognize the central importance of using the schema and type information contained schema definitions for the adequate management of XML documents. At the same time, these solutions should not neglect other important issues such as sophisticated (multidimensional) value, text, and path index structures, profound extensibility with custom functionality and index structures, and, not to forget, classic DBMS functionality such as transactions, fine-grained concurrency and access control, reliable means for backup and recovery, etc.

The necessity of using schema definitions to achieve an adequate management of XML documents seems to become more and more recognized. As a newer XML database solution, Oracle XML DB/Structured Mapping already to some extent makes use of schema definitions written in XML Schema for document validation, for the typing of basic document contents, as well as for query optimization. At least for the management of MPEG-7 media descriptions, however, the system has to be developed further to overcome its limitations with regard to the more complicated constructs of MPEG-7 DDL/XML Schema in order to be considered more than just a harbinger of a new generation of XML database solutions.

References

[1] S. Abiteboul, D. Quass, J. McHugh, et al. The Lorel Query Language for Semistructured Data. *Journal of Digital Libraries*, 1(1), 1997.

[2] E. Bertino and W. Kim. Indexing Techniques for Queries on Nested Objects. *IEEE Transactions on Knowledge and Data Engineering*, 1(2), 1989.

[3] P.V. Biron and A. Mahotra. XML Schema Part 2: Datatypes. W3C Recommendation, World Wide Web Consortium (W3C), May 2001.

[4] S. Boag, D. Chamberlin, M.F. Fernandez, et al. XQuery 1.0: An XML Query Language. W3C Working Draft, World Wide Web Consortium (W3C), August 2002.

[5] K. Böhm, K. Aberer, M.T. Öszu, and K. Gayer. Query Optimization for Structured Documents Based on Knowledge on the Document Type Definition. In *Proc. of the IEEE Forum on Research and Technology Advances in Digital Libraries (ADL '98)*, Santa Barbara, California, April 1998.

[6] P.A. Boncz and M.L. Kersten. MIL Primitives for Querying a Fragmented World. *The VLDB Journal*, 8(2), 1999.

[7] R. Bourret. XML Database Products. Online Article, available under http://www.rpbourret.com/xml/XMLDatabaseProds.htm, May 2002.

[8] B. Chang, E. Litani, J. Kesselman, and R. Rahman. Document Object Model (DOM) Level 3 Abstract Schemas Specification. W3C Note Version 1.0, World Wide Web Consortium (W3C), July 2002.

[9] J. Clark and S. DeRose. XML Path Language (XPath). W3C Recommendation Version 1.0, World Wide Web Consortium (W3C), November 1999.

[10] R. Cowan and R. Tobin. XML Information Set. W3C Recommendation, World Wide Web Consortium (W3C), October 2001.

[11] E. Damiani, S. de Capitani di Vimercati, S. Paraboschi, and P. Samarati. A Fine-Grained Access Control System for XML Documents. *ACM Transactions on Information and System Security*, 5(2), 2002.

[12] A. Deutsch, M. Fernandez, and D. Suciu. Storing Semistructured Data with STORED. In *Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD 1999)*, Philadelphia, Pennsylvania, June 1999.

[13] Y. Duchesne and P. Nyfeld. Ozone Developer's Guide. System Documentation Version 1.0, SMB GmbH, 2001.

[14] A.K. Elmagarmid, editor. *Database Transaction Models for Advanced Applications.* Morgan Kaufmann Publishers, San Mateo, California, 1992.

[15] eXcelon Corp. Managing DXE. System Documentation Release 3.5, eXcelon Corp., December 2001.

[16] M. Fernandez, J. Marsh, and M. Nagy. XQuery 1.0 and XPath 2.0 Data Model. W3C Working Draft, World Wide Web Consortium (W3C), August 2002.

[17] M. Fernandez and D. Suciu. Optimizing Regular Path Expressions Using Graph Schemas. In *Proc. of the Fourteenth International Conference on Data Engineering (ICDE '98)*, Orlando, Florida, February 1998.

[18] D. Florescu and D. Kossmann. Storing and Querying XML Data Using an RDBMS. *IEEE Data Engineering Bulletin*, 22(3), 1999.

[19] W.B. Frakes and R. Baeza-Yates, editors. *Information Retrieval – Data Structures & Algorithms.* Prentice Hall, Upper Saddle River, New Jersey, 1992.

[20] V. Gaede and O. Günther. Multidimensional Access Methods. *ACM Computing Surveys*, 30(2), 1998.

[21] G. Gardarin, F. Sha, and T.D. Ngoc. XML-Based Components for Federating Multiple Heterogeneous Data Sources. In *Proc. of the 18th International Conference on Conceptual Modeling (Conceptual Modeling - ER '99)*, Paris, France, November 1999.

[22] W. Gietz and C. Dupree. Oracle 9i Data Cartridge Developer's Guide . System Documentation Release 2 (9.2), Oracle Corp., March 2002.

[23] S. Godik and T. Moses. OASIS eXtensible Access Control Markup Language (XACML). OASIS Committee Draft Version 0.13, Organization for the Advancement of Structured Information Standards (OASIS), May 2002.

[24] R. Goldman, J. McHugh, and J. Widom. From Semistructured Data to XML: Migrating the Lore Data Model and Query Language. In *Proc. of the ACM SIGMOD Workshop on The Web and Databases (WebDB '99)*, Philadelphia, Pennsylvania, June 1999.

[25] R. Goldman and J. Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In *Proc. of the 23rd International Conference on Very Large Data Bases (VLDB '97)*, Athens, Greece, August 1997.

[26] S. Hada and M. Kudo. XML Access Control Language: Provisional Authorisation for XML Documents. Draft Specification, Tokyo Research Laboratory, IBM Corp., October 2000.

[27] S. Higgins, N. Agarwal, A. Agrawal, et al. Oracle 9i XML Database Developer's Guide – Oracle XML DB. System Documentation Release 2 (9.2), Oracle Corp., March 2002.

[28] G. Huck, I. Macherius, and P. Fankhauser. PDOM: Lightweight Persistency Support for the Document Object Model. In *Proc. of the Workshop "Java and Databases: Persistence Options" of the 14th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '99)*, Denver, Colorado, November 1999.

[29] IBM Corp. IBM DB2 Universal Database – XML Extender Administration and Programming. System Documentation Version 7, IBM Corp., 2000.

[30] IBM Corp. IBM DB2 Spatial Extender – User's Guide and Reference. System Documentation Version 7, IBM Corp., June 2001.

[31] IBM Corp. IBM DB2 Universal Database – Text Extender Administration and Programming. System Documentation Version 7, IBM Corp., March 2001.

[32] Infonyte GmbH. Infonyte-DB – User Manual and Programmers Guide. System Documentation Version 2.0.2, Infonyte GmbH, May 2002.

[33] ISO/IEC JTC 1/SC 29/WG 11. MPEG-7: Context, Objectives and Technical Roadmap, V.12. ISO/IEC Document N2861, International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), July 1999.

[34] ISO/IEC JTC 1/SC 29/WG 11. Information Technology – Multimedia Content Description Interface – Part 2: Description Definition Language. ISO/IEC Final Draft International Standard 15938-2:2001, International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), September 2001.

[35] ISO/IEC JTC 1/SC 29/WG 11. Information Technology – Multimedia Content Description Interface – Part 3: Visual. ISO/IEC Final Draft International Standard 15938-3:2001, International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), July 2001.

[36] ISO/IEC JTC 1/SC 29/WG 11. Information Technology – Multimedia Content Description Interface – Part 4: Audio. ISO/IEC Final Draft International Standard 15938-4:2001, International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), June 2001.

[37] ISO/IEC JTC 1/SC 29/WG 11. Information Technology – Multimedia Content Description Interface – Part 5: Multimedia Description Schemes. ISO/IEC Final Draft International Standard 15938-5:2001, International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), October 2001.

[38] ISO/IEC JTC 1/SC 29/WG 11. Information Technology – Multimedia Content Description Interface – Part 1: Systems. ISO/IEC Final Draft International Standard 15938-1:2001, International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), November 2001.

[39] IXIASOFT Inc. Creating Client Applications for TEXTML Server – Programmer's Guide. System Documentation Version 2.1, IXIASOFT Inc., December 2001.

[40] H.V. Jagadish, L.V.S. Lakshmanan, and D. Srivastava. Hierarchical or Relational? A Case for a Modern Hierarchical Data Model. In *Proc. of the IEEE Workshop on Knowledge and Data Engineering Exchange (KDEX'99)*, Chicago, Illinois, November 1999.

[41] C.C. Kanne and G. Moerkotte. Efficient Storage of XML Data. Technical Report 8/99, University of Mannheim, Germany, August 1999.

[42] H. Kosch. MPEG-7 and Multimedia Database Systems. *ACM SIGMOD Record*, 31(2), 2002.

[43] A. Laux and L. Martin. XUpdate – XML Update Language. XML:DB Working Draft 2000-09-14, XML:DB Initiative, September 2000.

[44] A. Le Hors, P. Le Hégaret, L. Wood, et al. Document Object Model (DOM) Level 2 Core Specification. W3C Recommendation Version 1.0, World Wide Web Consortium (W3C), November 2000.

[45] A. Le Hors, P. Le Hégaret, L. Wood, et al. Document Object Model (DOM) Level 3 Core Specification. W3C Working Draft Version 1.0, World Wide Web Consortium (W3C), April 2002.

[46] C. McGregor, O. Alonso, S. Alpha, et al. Oracle Text – Reference. System Documentation Release 2 (9.2), Oracle Corp., March 2002.

[47] W.M. Meyer. eXist User's Guide. System Documentation Version 0.71, 2002.

[48] Microsoft Corp. Microsoft SQL Server 2000 – Creating and Maintaining Databases. System Documentation, Microsoft Corp., 2000.

[49] Microsoft Corp. Microsoft SQL Server 2000 – SQLXML 2.0. System Documentation, Microsoft Corp., 2000.

[50] T. Milo and D. Suciu. Index Structures for Path Expressions. In *Proc. of the 7th International Conference on Database Theory (ICDT '99)*, Jerusalem, Israel, January 1999.

[51] C. Murray, D. Abugov, N. Alexander, et al. Oracle Spatial – User's Guide and Reference. System Documentation Release 2 (9.2), Oracle Corp., March 2002.

[52] F. Nack and A.T. Lindsay. Everything You Wanted to Know About MPEG-7: Part 1. *IEEE MultiMedia*, 6(3), 1999.

[53] F. Nack and A.T. Lindsay. Everything You Wanted to Know About MPEG-7: Part 2. *IEEE MultiMedia*, 6(4), 1999.

[54] L. Poola and J. Haritsa. SphinX: Schema-conscious XML Indexing. Technical Report TR-2001-04, Database Systems Lab, Indian Institute of Science, Bangalore, India, 2001.

[55] J. Robie, J. Lapp, and D. Schach. XML Query Language (XQL). In *Proc. of the W3C Query Languages Workshop (QL'98)*, Boston, Massachussets, December 1998.

[56] A. Salminen and F.W. Tompa. Requirements for XML Document Database Systems. In *Proc. of the ACM Symposium on Document Engineering 2001 (DocEng '01)*, Atlanta, Georgia, November 2001.

[57] A. Schmidt, M. Kersten, M. Windhouwer, et al. Efficient Relational Storage and Retrieval of XML Documents. In *Proc. of the Third International Workshop on the Web and Databases (WebDB 2000)*, Dallas, Texas, May 2000.

[58] E. Sciore. Versioning and Configuration Management in an Object-Oriented Data Model. *The VLDB Journal*, 3(1), 1994.

[59] J. Shanmugasundaram, K. Tufte, G. He, et al. Relational Databases for Querying XML Documents: Limitations and Opportunities. In *Proc. of the 25th International Conference on Very Large Data Bases (VLDB '99)*, Edinburgh, Scotland, September 1999.

[60] T. Shimura, M. Yoshikawa, and S. Uemura. Storage and Retrieval of XML Documents Using Object-Relational Databases. In *Proc. of the Database and Expert Systems Applications, 10th International Conference (DEXA '99)*, Florence, Italy, September 1999.

[61] Software AG. User Guide. System Documentation Version 3.1.1, Software AG, November 2001.

[62] K. Staken. dbXML Developers Guide 0.5. System Documentation Version 1.0, The dbXML Project, September 2001.

[63] K. Staken. Xindice Developers Guide 0.7. System Documentation Version 1.0, The Apache Software Foundation, March 2002.

[64] The PostgreSQL Global Development Group. The PostgreSQL 7.3devel Documentation. System Documentation Version 7.3, The PostgreSQL Global Development Group, 2002.

[65] H.S. Thompson, D. Beech, M. Maloney, et al. XML Schema Part 1: Structures. W3C Recommendation, World Wide Web Consortium (W3C), May 2001.

[66] F. Tian, D.J. DeWitt, J. Chen, and C. Zhang. The Design and Performance Evaluation of Alternative XML Storage Strategies. *ACM SIGMOD Record*, 31(1), 2002.

[67] X-Hive Corp. X-Hive/DB 2.0 – Manual. System Documentation Release 2.0.2, X-Hive Corp., May 2002.

[68] XML Global Technologies, Inc. GoXML DB Administrator Help. System Documentation Version 2.0.1, XML Global Technologies, Inc., December 2001.