

# Utilizing XML Schema for Describing and Querying Still Image Databases

Kazem Taghva, Min Xu, Emma Regentova, and Tom Nartker

Technical Report 2002-02  
Information Science Research Institute  
University of Nevada, Las Vegas

April 2002

## Abstract

We report on modeling of still images by hierarchical tree structures and object relational graphs. We show how naturally these modeling concepts can be described using XML schema. Of primary interest is the notion of complex types and referential integrity to fully describe the physical and semantic properties of images. We further show how complex types of XML can be used to overcome the shortcomings of reported image database descriptions in the literature.

## 1 Introduction

The phrase “Database Management System” implies the existence of a formal model and an efficient implementation. The formal model represents a mathematical notation and a set of operations, where as the implementation usually refers to indexing and query optimization. Readers familiar with relational databases can associate sets, relational algebra, B-trees, and various optimization techniques to modeling and implementation. Unfortunately, with the exception of a few works, the phrase “Image Database” points to a collection of images and a few extraction algorithms[10]. This model leaves a gap between the definition of *Database Management* and an *Image System*.

If we isolate feature extraction, *Image Databases* resemble *Information Retrieval Systems*. Before the database gets populated, various extraction algorithms need to be applied to identify *features* for indexing[7][17]. One example technique is word extraction using Optical Character Recognition (OCR) for subsequent indexing[14][15]. The project MARS at the University of Illinois[10] is an example of an image database that relies on information retrieval methods such as feedback[5]. Because of the resemblance of image databases to both structured databases and information retrieval systems, one can consider it to be a system that deals with *semi-structured* data[1]. We believe this maybe a good starting place to formalize a model in order to semantically explain the nature of image databases.

This paper is organized into five sections: Section 1 is the Introduction. Section 2 covers some background on still image modeling, XML DTDs, and XML Schemas. Section 3 describes our approach and the detailed description of our XML Schema. Section 4 shows how this XML schema can be queried using XQuery. Section 5 is the conclusion and future work.

## 2 Modeling Background

Data modeling is an integral part of any database project. Many approaches such as Entity-Relationship Diagrams[6], IDEF1X[3], and Object Oriented[6][4] methods are applied to model a database implementation. Due to the nature of image properties, it is a more difficult task to describe and model an image database. Some of the widely publicized image modeling tools are MPEG-7[8][2], Summary Tree[12][11], and UC DL[9].

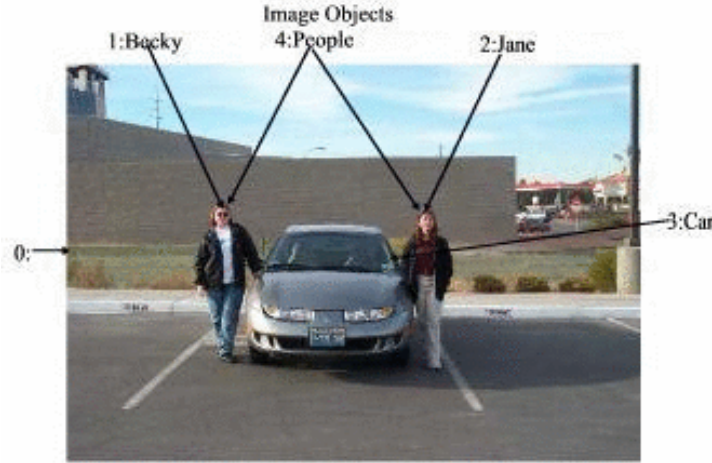


Figure 1: Image and Image Objects



Figure 2: Image and Image Objects

Among these, the most prominent is MPEG-7 which describes an image (or more generally a medium) as a set of image objects related by object hierarchies and entity-relationship graphs. The structure of the image objects are similar to summary trees as defined by Shaft and Ramakrishnan[11]. The object hierarchies and entity-relationship graphs are essentially capturing the information which is described by the object relation graphical notation of UCDL[9].

Consider the image in Figure 1. Following the approach given in[8], the image has objects 0 (the photograph), 1 (Becky), 2 (Jane), 3 (Car), and 4 (People). This image object set is described using XML as follows:

```
<image-objects>
  <image-object id="0" type="global"> <!-- Photograph -->
  <image-object id="1" type="local"> <!-- Becky -->
  <image-object id="2" type="local"> <!-- Jane -->
  <image-object id="3" type="local"> <!-- Car -->
  <image-object id="4" type="local"> <!-- People -->
</image-objects>
```

Most recent work in image database modeling uses XML to describe the image databases without capturing obvious constraints. There is a relationship *belongs-to* between an image and its image-objects as shown in figure 2. Each image has several image objects but each image object belongs to a single image. This is known as a *key constraint* in database terminology. However, the key constraint does not guarantee that each image object occurs in a *relationship* of this kind. In other words, every image object in the database must belong to an image. This is known as *participation constraint*[6]. These constraints generally can be enforced by *key* and *foreign key* constraints. Unfortunately, these constraints are impossible to describe in XML, thus their omission in image database literature. A restricted version of key and foreign key constraints can be enforced using types ID, IDREF, and IDREFS in XML DTDs. Consequently, most authors describe their models using DTDs.

One shortcoming of DTDs is the lack of the ability to define complex types. One can not specify that a *position descriptor* must have a float representation within a certain range. Another example of type

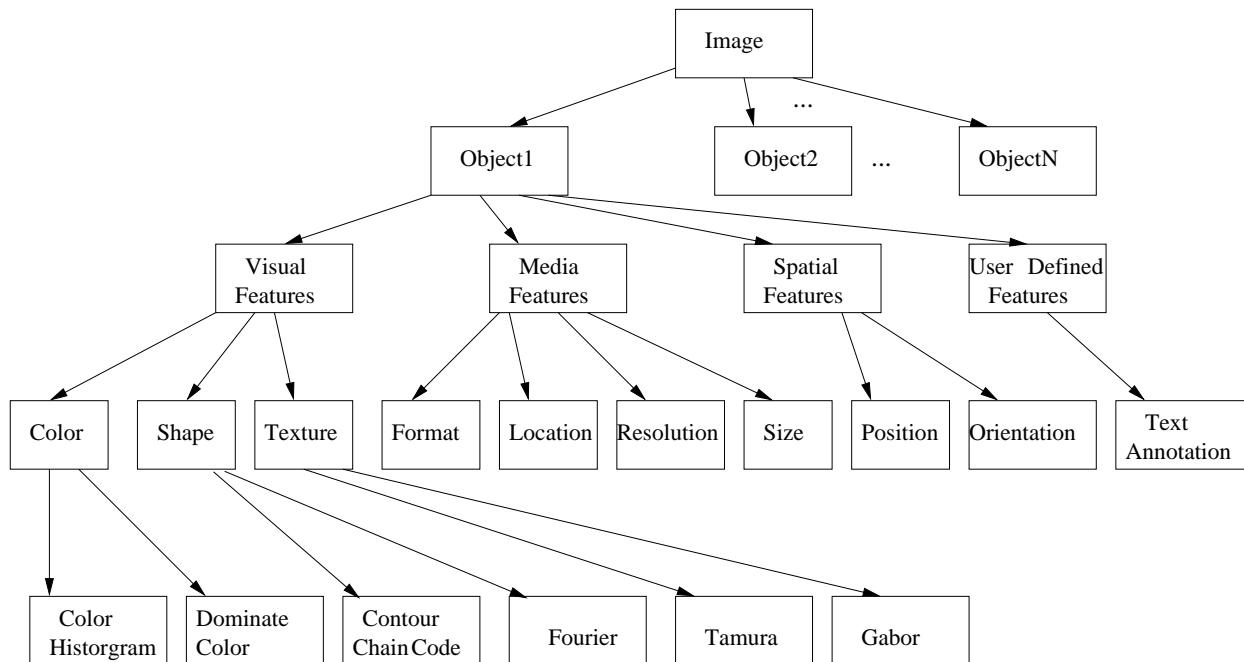


Figure 3: Summary Tree

problems is the inability to enforce certain *order* among objects (e.g. Becky is to the *left of* Jane). The ability to enforce certain constraints together with the freedom to define complex types are the main reasons for the development of the XML *Schema*[6].

In the next section, we will describe our model using XML schema complex types equipped with examples of key and foreign key constraints. We will show how we use the complex type constructs to overcome the shortcomings previously encountered when modeling an image database.

### 3 Summary Tree, Object Relational Graph, and XML Schema

We will use two graphical representations to fully describe image contents. Although both representations are previously used in the literature, our description of these representations in XML schema is new. In the next section, we will show how this schema can be used for query purposes using standard XPath and Xquery.

#### 3.1 Summary Tree Schema

The purpose of a summary tree is to decompose an image into image objects[13]. An image object refers to one or more arbitrary regions of an image, so it can be continuous or discontinuous in space. In particular, one of the image objects is the image itself. Each object is described by four high level *visual*, *media*, *spatial*, and *user-defined* features. Each of these features are further described in terms of other properties as depicted in the summary tree of Figure 3. For example, the visual feature is described by *color*, *shape*, and *texture*. The feature *color* is further described by color histograms, dominate color, etc. Table 1 provides a possible list of descriptors for color, texture, and shape features.

In order to fully describe this summary tree in XML schema, we will first define a complex type for image as follows:

```

<complexType name="imageType">
  <sequence>

```

Features	Descriptors
Color	Color Histogram, Dominant Color, Color Coherence Vector, Visual Sprite Color
Texture	Tamura, MSAR, Edge Direction Histogram, DCT Coefficient Energies, Visual Sprite Texture
Shape	Bounding box, Binary Mask, Chroma Key, Polygon Shape, Fourier Shape, Boundary Size, Symmetry, Orientation Contour Chain Code

Table 1: Examples of Visual Features

```
<element name="object" Type=objectType" minOccurs="1" maxOccurs="unbounded"/>
</sequence>
  <attribute name="imageID" type="integer"/>
</complexType>
```

This says that an image has one or more objects and an attribute *imageID* of type integer which will serve as the key of the image. Each object has four features and one key as described by the following XML:

```
<complexType name="objectType">
  <sequence>
    <element name="visualFeatures" type="visualType" maxOccurs="1"/>
    <element name="mediaFeatures" type="mediaType" maxOccurs="1"/>
    <element name="spatialFeatures" type="spatialType" maxOccurs="1"/>
    <element name="UDTFeatures" type="UDTType" maxOccurs="1"/>
  </sequence>
  <attribute name="objectID" type="integer"/>
</complexType>
```

Next, we need to enforce the relationship between the image and image objects by key and foreign key constraints as follows:

```
<element name="image" type="imageType">
  <key name="imagePK"><!--image Primary Key-->
    <selector xpath="//image"/>
    <field xpath="@imageID"/>
  </key>
</element>
<element name="object" type="objectType">
  <key name="objectPK"><!--object Primary Key-->
    <selector xpath="//object"/>
    <field xpath="@objectID"/>
  </key>
  <keyref name="objectFK" refer="imagePK"><!--object Foreign Key-->
    <selector xpath="//object"/>
    <field xpath="@imageID"/>
  </keyref>
```

```
</element>
```

The above key constraint addresses the problem we pointed out in Section 2. This XML code does not capture the *total participation* as was discussed in Section 2. The necessary modification for total participation is in the complete schema of Appendix A. In Section 2 we also pointed out two other problems that we can overcome using XML Schema. One of the advantages of using XML Schema is the ability to choose one and only one of many choices provided by the system. For example, we may want to choose one specific routine for feature extraction. This can be done using the choice constructor as follows:

```
<complexType name="colorType"><!--color descriptors-->
  <choice minOccurs=maxOccurs=1>
    <element name="colorHistogram" type="colorHistogramType"/>
    <element name="dominateColor" type="dominateColorType"/>
    <element name="colorCoherence" type="colorCoherenceType"/>
    <element name="vector" type="vectorType"/>
  </choice>
</complexType>
```

As we pointed out in Section 2, there is also no way to specify that the *position descriptors* are valid representations of a float within a given range in DTDs.

The XML Schema remedied this situation by making strong typing available in its specification. In our model, we scale an image to a 0.0 to a 1.0 scale. We can define a new simple data type "coordinateType" which is analogous to the built-in type defined in traditional database management systems that will overcome the float representation problem:

```
<simpleType name="coordinateType" base="float">
  <minInclusive value="0.0">
  <maxInclusive value="1.0">
</simpleType>
```

Finally, we complete our schema by describing the summary tree in Appendix A. In the next subsection, we will develop our schema for the object-relation graph.

### 3.2 Object-relation Graph Schema

Some spatial relationships can be deduced from spatial features. For example, in 1, the normalized centers of Jane and Becky are (0.2, 0.5) and (0.7, 0.5), respectively. From these coordinates we can conclude that Becky is to the *left of* Jane. In general, it may be very difficult to automatically capture this kind of relationship. This is the reason, we need another structure to describe more semantic information. A good way of modeling relationships among image objects is by using an *object-relation* graph. In this structure, objects and relationships are represented using circles and arrows, respectively. Arrows with rectangles represent labeled relationships while arrows with no rectangles represent inheritance. Figure 4 describes some of the obvious relationships of the photograph in Figure 1.

In order to describe the fact that Becky is to the *left of* Jane, we can define a new complex type using the sequence constructor as follows:

```
<complexType name="LeftofType">
```

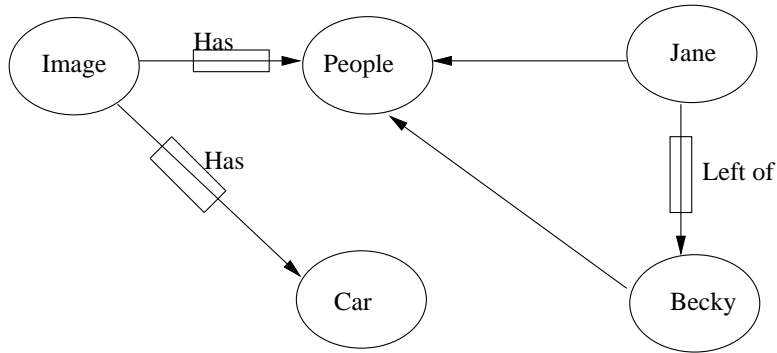


Figure 4: Example of Object-Relation Graph

Relation Types	Relations
Spatial	Top of, Bottom of, Right of, Left of, Upper Right of, Upper Left of, Lower Right of, Lower Left of
Semantics	Same as, Related to, Belongs to, Has

Table 2: Examples of Relation Types

```

<sequence >
  <element name="object1" type="objectTupleType"/>
  <element name="object2" type="objectTupleType"/>
</sequence>
</complexType>

```

The `sequence` tag is used to specify that the elements `object1`, and `object2` must occur in the given order explicitly. This is a crucial point when compared with the following implied order as described in MPEG-7:

```

<entity_relation_graph>
  <entity_relation><!--spatial,direction entity relation-->
    <relation type="SPATIALDIRECTIONAL">Left of</relation>
    <entity_node id="ETN1" object_ref="01"/>
    <entity_node id="ETN2" object_ref="02"/>
  </entity_relation>
</entity_relation_graph>

```

Examples of other relation types are given in Table 2. The complete schema for the object-relation graph is shown in Appendix B.

## 4 Query Processing

In content-based image retrieval systems, similarity functions are used to support query processing. For example, we can issue a query to find images with similar color histograms to `<0.3,0.4,0.3,...>` or find all images similar to a particular image. Typically, queries of similarity type have their own user defined functions. One immediate advantage of our XML schema is that we can consider the image database as a collection of XML documents. Each document represents the description of an image. This representation

naturally fits into the query processing capabilities of XML query languages such as Xpath, XSLT, and XQuery[6].

Consider the query to retrieve all images similar to the image with imageID equal to 53795. We can formulate this query in XQuery using the pre-defined “distanceFromExample” function as follows:

```
FOR      $c IN document("http://www.isri.unlv.edu/images.xml")//image
WHERE    distanceFromExample($c,'53795',[coarsness,contrast,orientation],
                             <0.2,0.7,0.1>, 'tamura') < . 0.5
RETURN   $c/imageID
```

The variable \$c ranges over the images which are obtained using the function document(). The example image is specified by the imageID 53795 and compared with the current image as specified by the first argument of the function distanceFromExample. The third argument of the distance function is a list of values that should be taken into account when determining the similarity. This example query asks for images that are similar in texture with respect to coarsness, contrast and orientation.

The fourth argument is the weight vector indicating the relative importance of the third vector argument of the distance function. The sum of the weight vector should be one. In this example, the contrast is more important than the coarsness or orientation. There are various methods to calculate similarity matching. The fifth argument specifies the method to use is “tamura.” Finally the result of the function is matched against the threshold of 0.5.

Another interesting way of reformulating the query is to use the SORTBY function of XQuery to produce a ranked list as follows:

```
<Result>
FOR      $c IN document("http://www.isri.unlv.edu/images.xml")//image
RETURN   <Result dist=distanceFromExample($c,'53795',
                                           [coarsness,contrast,orientation],<0.2,0.7,0.1>,'tamura')> $c
        </Result>
SORTBY(@dist)
</Result>
```

## 5 Conclusion

We have given a detailed XML Schema to describe an image database system. The schema uses key and foreign key constraints to enforce database integrity. The schema also uses the strong type capabilities of XML Schema to overcome certain shortcomings of typical DTDs that have been used in the literature. This schema is used to represent a collection of images as a collection of XML documents in order to allow query processing using standard XML query languages such as XQuery.

Our schema can be easily extended to other modeling techniques such as MPEG-7. Our schema is currently being used to model a collection of still images for medical patients records[16].

## A XML Image Schema

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:sid="http://www.isri.unlv.edu/name/sid"
        TargetNameSpace="http://www.isri.unlv.edu/name/sid">

<element name="image" type="imageType">
  <key name="imagePK"><!--image Primary Key-->
```

```

        <selector xpath="//image"/>
        <field xpath="@imageID"/>
    </key>
</element>

<complexType name="imageType">
    <sequence>
        <element name="object" Type="objectType" minOccurs="1"
            maxOccurs="unbounded"/>
    </sequence>
    <attribute name="imageID" type="interger"/>
</complexType>

<element name="object" type="objectType">
    <key name="objectPK"><!--object Primary Key-->
        <selector xpath="//object"/>
        <field xpath="@objectID"/>
    </key>
    <keyref name="objectFK" refer="belongsPK"><!--object Foreign Key-->
        <selector xpath="//belongs"/>
        <field xpath="@objectID"/>
    </keyref>
</element>

<!-- definition of belongsto relation to enforce "total participant"-->

<element name="belongsTo" type="belongsToType">
    <key name="belongsPK"><!--Belongsto Primary key-->
        <selector xpath="//belongs"/>
        <field xpath="@objectID"/>
    </key>
    <keyref name="belongsFK1" refer="imagePK"><!--belongsto Foriegn Key1-->
        <selector xpath="//belongs"/>
        <field xpath="@imageID"/>
    </keyref>
    <keyref name="belongsFK2" refer="objectPK"><!--belongsto Foriegn Key2-->
        <selector xpath="//belongs"/>
        <field xpath="@objectID"/>
    </keyref>
</element>

<complexType name="objectType">
    <sequence>
        <element name="visualFeatures" type="visualType" maxOccurs="1"/>
        <element name="mediaFeatures" type="mediaType" maxOccurs="1"/>
        <element name="spatialFeatures" type="spatialType" maxOccurs="1"/>
        <element name="UDTFeatures" type="UDTType" maxOccurs="1"/>
    </sequence>
    <attribute name="objectID" type="interger"/>
</complexType>

<complexType name="visualType"><!--Visual Features-->
    <sequence>

```

```

        <element name="color" type="colorType" maxOccurs="1"/>
        <element name="shape" type="shapeType" maxOccurs="1"/>
        <element name="texture" type="textureType" maxOccurs="1"/>
    </sequence>
</complexType>

<complexType name="mediaType"><!-- media Features-->
    <sequence>
        <element name="format" type="string"/>
        <element name="location" type="locationType"/>
        <element name="size" type="byte"/>
        <element name="resolution" type="resolution"/>
    </sequence>
</complexType>

<complexType name="spatialType"> <!--spatial Features-->
    <sequence>
        <element name="position" type="positionType"/>
        <element name="orientation" type="float"/>
    </sequence>
</complexType>

<complexType name="UDTType"><!-- userdefined Features-->
    <element name="textAnotation" type="string"/>
    <element name="keyword" type="string"/>
</complexType>

<complexType name="colorType"><!--color descriptors-->
    <choice minOccurs=maxOccurs=1>
        <element name="colorHistogram" type="colorHistogramType"/>
        <element name="dominateColor" type="dominateColorType"/>
        <element name="colorCoherence" type="colorCoherenceType"/>
        <element name="vector" type="vectorType"/>
    </choice>
</complexType>

<complexType name="shapeType"><!--shape descriptors-->
    <choice minOccurs=maxOccurs="1">
        <element name="boundingBox" type="boundingBoxType"/>
        <element name="binaryMask" type="binaryMaskType"/>
        <element name="chromaKey" type="chromaKeyType"/>
        <element name="polygonShape" type="polygonShapeType"/>
        <element name="contourChainCode" type="contourChainCodeType"/>
    </choice>
</complexType>

<complexType name="textureType"><!-- texture descriptors-->
    <choice minOccurs=maxOccurs="1">
        <element name="tamura" type="tamuraType"/>
        <element name="Histogram" type="HistogramType"/>
        <element name="MSAR" type="MSARType"/>
    </choice>
</complexType>

```

```

<!--in every feature descriptors, we show one example-->

<simpleType name="ArrayOfFloat" base="float" derivedBy="list"/>
  <length value="166"/>
</simpleType>
<simpleType name="ArrayOfInteger" base="integer" derivedBy="list"/>
  <length value="16"/>
</simpleType>
<simpleType name="MatrixofFloat" base="ArrayOfFloat"/>
  <dim value="4 16"/>
</simpleType>
<simpleType name="colorHistogramType" type="ArrayOfFloat"></simpleType>
<simpleType name="dominateColorType" type="MatrixofFloat"></simpleType>
<simpleType name="contourChainCodeType" type="ArrayOfInteger"></simpleType>

<complexType name="tamuraType">
  <sequence>
    <element name="coarseness" type="float"/>
    <element name="contrast" type="float"/>
    <element name="orientation" type="float"/>
  </sequence>
</complexType>

<simpleType name="locationType">
  <restriction base="string">
    <pattern vlaue="/[[a-z,0-9]*/]*[a-z,0-9]*.jpg"/>
  </restriction>
</simpleType>

<complexType name="resolutionType">
  <sequence>
    <element name="width" type="integer"/>
    <element name="height" type="integer"/>
  </sequence>
</complexType>
<simpleType name="coordinateType" base="float">
  <minInclusive value="0.0">
  <maxInclusive value="1.0">
</simpleType>
<complexType name="positionType">
  <sequence>
    <element name="x-coordinate" type="coordinateType"/>
    <element name="y-coordinate" type="coordinateType"/>
  </sequence>
</complexType>

</schema>

```

## B Object-relation Graph Schema

```
<?xml version="1.0"?>
```

```

<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:sid="http://www.isri.unlv.edu/name/sid"
        TargetNameSpace="http://www.isri.unlv.edu/name/sid">

<complexType>
  <sequence>
    <element name="objectRelation" type="objectRelationType"/>
  </sequence>
</complexType>

<complexType name="objectRelationType">
  <sequence>
    <element name="semanticRelation" type="semanticRelationType"/>
    <element name="spatialRelation" type="spatialRelationType"/>
  </sequence>
</complexType>

<complexType name="semanticRelationType"/>
  <sequence>
    <element name="SameAs" type="SameAsType"/>
    <element name="RelatedTo" type="RelatedToType"/>
    <element name="BelongsTo" type="BelongsToType"/>
    <element name="Consistof" type="ConsistOfType"/>
    <element name="IsA" type="IsAType"/>
  </sequence>
</complexType>

<complexType name="spatialRelationType"/>
  <choice minOccurs="0" maxOccurs="1">
    <element name="TopOf" type="TopOfType"/>
    <element name="BottomOf" type="BottomOfType"/>
    <element name="RightOf" type="RightOfType"/>
    <element name="LeftOf" type="LeftOfType"/>
    <element name="UpperRightOf" type="UpperRightOfType"/>
    <element name="LowerLeftOf" type="LowerLeftOfType"/>
    <element name="Within" type="WithinType"/>
    <element name="Contain" type="ContainType"/>
  </choice>
</complexType>

<complexType name="IsAType">
  <sequence>
    <element name="object1" type="objectType"/>
    <element name="object2" type="objectType"/>
  </sequence>
</complexType>

<complexType name="SameAsType">
  <sequence>
    <element name="object1" type="objectType"/>
    <element name="object2" type="objectType"/>
  </sequence>
</complexType>

```

```

<complexType name="RelatedToType">
  <sequence>
    <element name="object1" type="objectType"/>
    <element name="object2" type="objectType"/>
  </sequence>
</complexType>

<complexType name="BelongsToType">
  <sequence>
    <element name="object1" type="objectType"/>
    <element name="object2" type="objectType"/>
  </sequence>
</complexType>

<complexType name="ConsistofType">
  <sequence>
    <element name="object1" type="objectType"/>
    <element name="object2" type="objectType"/>
  </sequence>
</complexType>

<complexType name="TopOfType">
  <sequence>
    <element name="object1" type="objectType"/>
    <element name="object2" type="objectType"/>
  </sequence>
</complexType>

<complexType name="BottomOfType">
  <sequence>
    <element name="object1" type="objectType"/>
    <element name="object2" type="objectType"/>
  </sequence>
</complexType>

<complexType name="RightOfType">
  <sequence>
    <element name="object1" type="objectType"/>
    <element name="object2" type="objectType"/>
  </sequence>
</complexType>

<complexType name="LeftOfType">
  <sequence>
    <element name="object1" type="objectType"/>
    <element name="object2" type="objectType"/>
  </sequence>
</complexType>

<complexType name="UpperRightOfType">
  <sequence>
    <element name="object1" type="objectType"/>

```

```

        <element name="object2" type="objectType"/>
    </sequence>
</complexType>

<complexType name="LowerLeftOfType">
    <sequence>
        <element name="object1" type="objectType"/>
        <element name="object2" type="objectType"/>
    </sequence>
</complexType>

<complexType name="WithinType">
    <sequence>
        <element name="object1" type="objectType"/>
        <element name="object2" type="objectType"/>
    </sequence>
</complexType>

<complexType name="ContainType">
    <sequence>
        <element name="object1" type="objectType"/>
        <element name="object2" type="objectType"/>
    </sequence>
</complexType>

</schema>

```

## References

- [1] Serge Abiteboul, Peter Buneman, and Dan Suciu. *Data on the Web*. Morgan Kaufmann, San Francisco, CA, 2000.
- [2] A. B. Benitez, S. Paek, et al. Object-based multimedia content description schemes and applications for mpeg-7. *Image Communication Journal*, 16(1-2):235–269, 2000.
- [3] Thomas A. Bruce. *Designing Quality Databases with IDEF1X Information Models*. Dorset House Publishing, New York, 1992.
- [4] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems*. Addison-Wesley, Reading, MA, 2000.
- [5] William B. Frakes and Ricardo Baeza-Yates, editors. *Information Retrieval, Data Structures & Algorithms*. Prentice Hall, Englewood Cliffs, NJ, 1992.
- [6] Philip M. Lewis, Arthur Bernstein, and Michael Kifer. *Databases and Transaction Processing*. Addison-Wesley, Reading, MA, 2002.
- [7] B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, August 1996.
- [8] S. Paek, A. B. Benitez, et al. Proposal for mpeg-7 image description scheme. Proposal to ISO/IEC JTC1/SC29/WG11 MPEG99/P480, February 1999.
- [9] S. Panchanathan, F. Golshani, and Y. C. Park. Videoroadmap: A system for interactive classification and indexing of still and motion pictures. In *Proc. IEEE Instrumentation and Measurement Technology Conference*, pages 18–21, St. Paul, MN, May 1998.

- [10] Y. Rui, T. S. Huang, and S. F. Chang. Image retrieval: Past, present, and future. *Journal of Visual Communication and Image Representation*, 10:1–23, 1999.
- [11] Uri Shaft and Raghu Ramakrishnan. Data modeling and feature extraction for image databases. In C. C. Kuo, editor, *Proc. SPIE Conf. on Multimedia Storage and Archiving Systems*, volume 2916, Boston, MA, November 1996. SPIE.
- [12] Uri Shaft and Raghu Ramakrishnan. Data modeling and querying in the piq image dbms. *Data Engineering Bulletin*, 19(4):28–36, 1996.
- [13] Uri Shaft and Raghu Ramakrishnan. Content-based queries in image databases. Technical Report TR1309, Computer Science Department, Univ. of Wisconsin Madison, Madison, WI, March 1996.
- [14] Kazem Taghva, Julie Borsack, and Allen Condit. Effects of OCR errors on ranking and feedback using the vector space model. *Inf. Proc. and Management*, 32(3):317–327, 1996.
- [15] Kazem Taghva, Julie Borsack, and Allen Condit. Evaluation of model-based retrieval effectiveness with OCR text. *ACM Transactions on Information Systems*, 14(1):64–93, January 1996.
- [16] Kazem Taghva, Julie Borsack, Tom Nartker, Steve Lumos, and Allen Condit. Managing occupational medicine historical data. Technical Report 2002-04, Information Science Research Institute, University of Nevada, Las Vegas, April 2002.
- [17] James Z. Wang. Simplicity: A region-based image retrieval system for picture libraries and biomedical image databases. In *Proc. ACM Multimedia*, pages 483–484, Los Angeles, CA, October 2000.