

Multimedia Databases

Seminar Report

by

Ajit Burad

Roll No : 03005009

Computer Science and Engineering
Indian Institute of Technology

under the guidance of
Prof N. L. Sarda

April 6, 2006

Acknowledgments

I would like to thank **Prof. N. L. Sarda** for his invaluable guidance and encouragement and support for making this seminar as informative experience for me.

Ajit Burad

Abstract

As the use of multimedia data has increased tremendously in various software applications. The applications include digital libraries (text documents, images, sound, video), manufacturing and retailing, art and entertainment, journalism and so on.. Normal databases are incapable of handling such wide range and huge amount of data. So we need database to support storage, indexing, retrieval of huge and wide variety of data. This report presents different ways of storing multimedia data in order facilitate easy indexing and retrieval. This includes advanced data structures and use of metadata to store multimedia data . This report also gives an insight of multimedia data support provided by SQL. The goal of this report is to provide details to understand various techniques to support, improve efficiency of multimedia database.

Contents

1	Intoduction	3
1.1	What is Multimedia Data	3
1.2	How is Multimedia Data Different ?	4
1.3	Basic Approaches for Data Retrieval	4
2	Advanced Data Structure to represent Multimedia data :	6
2.1	Why are access methods important	6
2.2	k-d Trees	7
2.2.1	Adding elements to k-d Tree	7
2.2.2	Deleting element from k-d tree	8
2.3	Division of Space by Quad-trees	8
2.3.1	Simple definition of node structure of a point quad-tree: . . .	8
2.3.2	Common uses of Quad-trees are :	8
2.3.3	Representing Image Using Quad-tree:	9
2.3.4	Indexing Using Quadtrees:	10
2.3.5	Advantages of quad trees include:	12
2.4	R-Trees	13
2.4.1	Structure of an R-tree node is defined as follows:	13
2.4.2	To insert a node	14
2.4.3	To delete node form R-tree	14
2.4.4	Searching R-tree	14
2.4.5	Variants of R-tree	15
2.5	Types of Queries:	15
2.6	Comparison of Different Data Structures	16
3	Metadata	17
3.1	Need of Metadata	17
3.2	Metadata Classification	18
3.2.1	Based on dependence on content	18
3.2.2	Hierarchical Classification	19
3.3	Source of Metadata	19
3.4	Generation of Metadata	20
3.5	Metadata standards	21
3.6	Some points to remember while generating metadata	22

4	Multimedia database support by SQL and other packages	23
4.1	Large Object Types in Oracle and SQL3	24
4.2	SQL/MM	24
4.3	Oracle's interMedia	25
5	Epilogue	27
5.1	Conclusion	27
5.2	Future Scope	27

Chapter 1

Introduction

1.1 What is Multimedia Data

There are number of data types that can be characterized as multimedia data types. These are typically the elements for the building blocks of ore generalized multimedia environments, platforms, or integrating tools. The basic types can be described as follows :

- **Text** : The form in which the text can be stored can vary greatly. In addition to ASCII based files, text is typically stored in processor files, spreadsheets, databases and annotations on more general multimedia objects. With availability and proliferation of GUIs, text fonts the job of storing text is becoming complex allowing special effects(color, shades..).
- **Images** : There is great variance in the quality and size of storage for still images. Digitalized images are sequence of pixels that represents a region in the user's graphical display. The space overhead for still images varies on the basis of resolution, size, complexity, and compression scheme used to store image. The popular image formats are jpg, png, bmp, tiff.
- **Audio** : An increasingly popular datatype being integrated in most of applications is Audio. Its quite space intensive. One minute of sound can take up to 2-3 Mbs of space. Several techniques are used to compress it in suitable format.
- **Video** : One on the most space consuming multimedia data type is digitalized video. The digitalized videos are stored as sequence of frames. Depending upon its resolution and size a single frame can consume upto 1 MB. Also to have realistic video playback, the transmission, compression, and decompression of digitalized require continuous transfer rate.
- **Graphic Objects**: These consists of special data structures used to define 2D and 3D shapes through which we can define multimedia objects. These includes various formats used by image, video editing applications. Examples are CAD / CAM objects.

1.2 How is Multimedia Data Different ?

Conceptually it should be possible to treat multimedia data in the same way as data based on the data types (e.g. numbers, dates and characters). However, there are few challenges that arise from multimedia data as described in [7].

- The content of multimedia data is often captured with different "capture" techniques (e.g., image processing) that may be rather unreliable. Multimedia processing techniques need to be able to handle different ways of content capture including automated ways and/or manual methods.
- Queries posed by the user in multimedia databases often cannot come back with a textual answer. Rather, the answer to a query may be a complex multimedia presentation that the user can browse at his/her leisure. Our framework shows how queries to multimedia databases may be used to generate multimedia presentations that satisfy users queries—a factor that is unique to our framework.
- Multimedia data is large and affects the storage, retrieval and transmission of multimedia data.
- In case of video and audio databases time to retrieve information may be critical (Video on demand).
- Automatic feature extraction and Indexing: In conventional databases user explicitly submits the attribute values of objects inserted into the database. In contrast, advanced tools such as image processing and pattern recognition tools for images, to extract the various features and content of multimedia objects. As size of data is very large we need special data structures for storing and indexing.

1.3 Basic Approaches for Data Retrieval

Data management has a long history and many approaches have been invented to manage and query diverse data types in the computer systems. As given in [15], the basic approaches being used for data management can be classified into the following categories:

- **Conventional database system** : This is the widely-used approach to manage and search for structured data. All data in a database system must conform to some predefined structures and constraints (i.e., schema's). To formulate a database query the user must specify which data objects are to be retrieved, the database tables from which they are to be extracted and predicate on which the retrieval is based. A query language for the database will generally be of the artificial kind, one with restricted syntax and vocabulary, such as SQL.

- **Information retrieval (IR) system** : IR system is mainly used to search large text collections, in which the content of the (text) data is described by an indexer using keywords or a textual abstract, and keywords or natural language is used to express query demands. For example for an image or video we have to describe it in words or in a way need to store lot of metadata (textual form).
- **Content based retrieval (CBR) system** : This approach is used to retrieve desired multimedia objects from a large collection on the basis of features (such as color, texture and shape, etc.) that can be automatically extracted from the objects themselves. Although keyword can be treated as a "feature" for text data, traditional information retrieval has much more higher performance than content-based retrieval because keyword has the proven ability to represent semantics, while no features have shown convincing semantic describing ability. But major drawback of this method is that it lacks precision.
- **Graph or tree pattern matching** : This approach aims to retrieve object sub-graphs from an object graph according to some denoted patterns.

Chapter 2

Advanced Data Structure to represent Multimedia data :

Many modern database applications deal with large amounts of multidimensional data. Examples include multimedia content-based retrieval (high dimensional multimedia feature data), time-series similarity retrieval, data mining and spatial/spatio-temporal applications. To be able to handle multidimensional data efficiently, we need access methods (AMs) to selectively access some data items in a large collection associatively.

2.1 Why are access methods important

The main purpose is to support efficient spatial selection, for example range queries or nearest neighbour queries of spatial objects. *Peter Van Oosterom* [17] describes importance of these access methods and how they also take into account both spatial indexing and clustering techniques. Without a spatial index, every object in database need to be checked whether it meets selection criterion, i.e. complete linear scan of relational database.

Clustering is needed to group those objects which are often requested together. Otherwise, many different disk pages will have to be fetched, resulting in slow response. For spatial selecting the clustering implies storing objects which are close together in reality also close together in the computer memory (instead of scattered over the whole memory).

In traditional database systems sorting (or ordering) the data is the basis for efficient searching. Higher dimensional data can not be sorted in an obvious manner, as it is possible for text strings, numbers, or dates (one-dimensional data). Basically, computer memory is one-dimensional. However, spatial data is 2D, 3D (or even higher) and must be organized somehow in the memory. An intuitive solution to organize the data is using a regular grid just as on a paper map. Each grid cell has a unique name; e.g. 'A3', 'C6', or 'D5'. The cells are stored in some order in the memory and can each contain a (fixed) number of object references. In a grid cell, a reference is stored to an object whenever the object (partially) overlaps the cell. However, this will not be very efficient due to the irregular data distribution

of spatial data: many cells will be empty, e.g. in the ocean, while many other cells will be overfull, e.g. in the city center. Therefore, more advanced techniques have been developed.

2.2 k-d Trees

Early structure used for indexing in multiple dimensions. The k-d tree is used to store k-dimensional points data. For example Image may have many attributes such as spatial position, texture, color. So k-d trees can be used to represent such data. Data in this setion taken from [1].

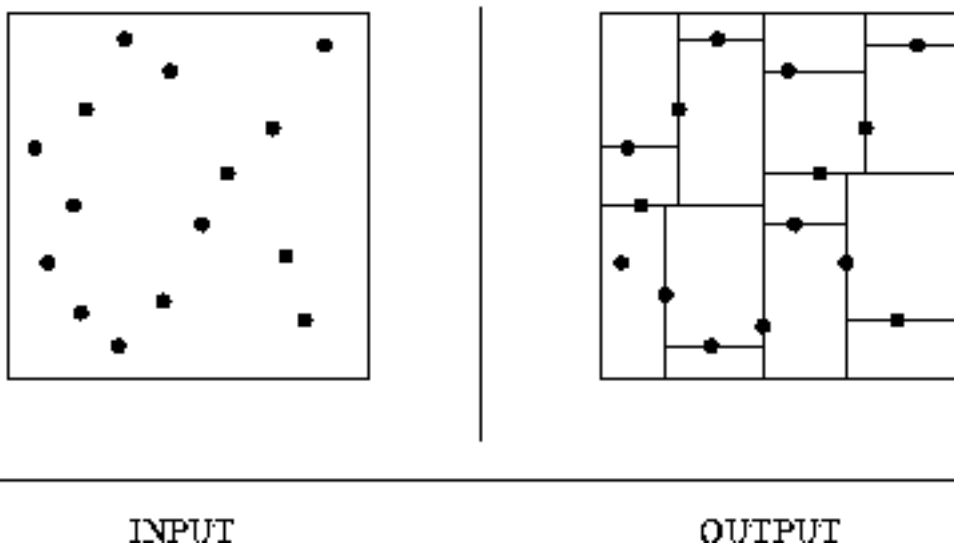


Figure 2.1: Division of points by k-d Tree

Purpose is always to hierarchically decompose space into a relatively small number of cells such that no cell contains too many input objects. This provides a fast way to access any input object by position. We traverse down the hierarchy until we find the cell containing the object and then scan through the few objects in the cell to identify the right one. Typical algorithms construct kd-trees by partitioning point sets. Each node in the tree is defined by a plane through one of the dimensions that partitions the set of points into left/right (or up/down) sets, each with half the points of the parent node. These children are again partitioned into equal halves, using planes through a different dimension. Partitioning stops after $\lg(n)$ levels, with each point in its own leaf cell. Alternate kd-tree construction algorithms insert points incrementally and divide the appropriate cell, although such trees can become seriously unbalanced.

2.2.1 Adding elements to k-d Tree

One adds a new point to a kd-tree in the same way as one adds an element to any other tree. First, traverse the tree, starting from the root and moving to either the

left or the right child depending on whether the point to be inserted is on the "left" or "right" side of the splitting plane. Once you get to a leaf node, add the new point as either the left or right child of the leaf node, again depending on which side of the node's splitting plane contains the new point.

2.2.2 Deleting element from k-d tree

Deletion similar to BST but slightly harder.

Step1 find node to be deleted.

Step2 two cases must be handled :

- (a) No children - replace ptr to node by NULL
- (b) Has children - replace node by minimum node in right subtree. If no right subtree exists, then first move left subtree to become right subtree.

2.3 Division of Space by Quad-trees

Each node of a quad-tree is associated with a rectangular region of space; the top node is associated with the entire target space. Each non-leaf nodes divides its region into four equal sized quadrants correspondingly each such node has four child nodes corresponding to the four quadrants and so on. Leaf nodes have between zero and some fixed maximum number of points (Fig 2.2: set to 1 in example).

2.3.1 Simple definition of node structure of a point quad-tree:

```
qtnodetype = record
    INFO: infotype;
    XVAL: real;
    YVAL: real;
    NW, SW, NE, SE: *qtnodetype
end
```

here INFO is some additional info regarding that point
XVAL, YVAL are coordinates of that point.
NW, SW, NE, SE are pointers to regions obtained by dividing given region

2.3.2 Common uses of Quad-trees are :

1. Image Representation
2. Spatial Indexing
3. Efficient collision detection in two dimensions

4. Storing sparse data, such as a formatting information for a spreadsheet or for some matrix calculations .

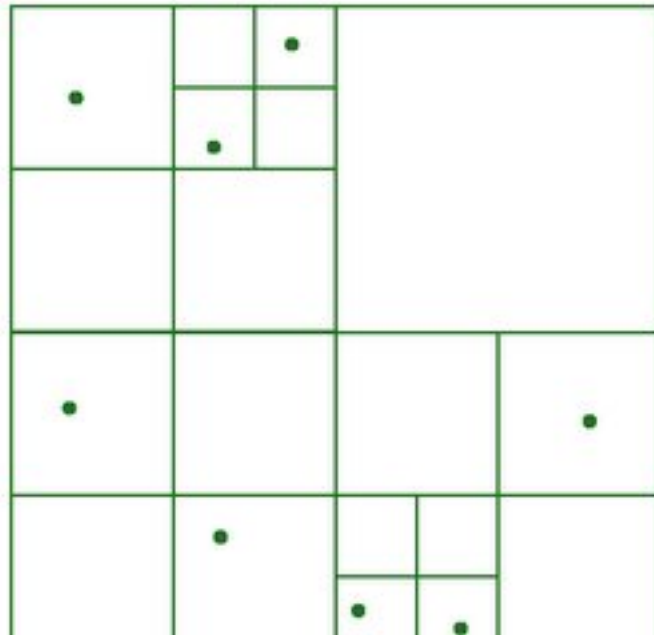


Figure 2.2: Division of points by Quad-tree

2.3.3 Representing Image Using Quad-tree:

Let's say we divide the picture area into 4 sections. Those 4 sections are then further divided into 4 subsections. We continue this process, repeatedly dividing a square region by 4. We must impose a limit to the levels of division otherwise we could go on dividing the picture forever. Generally, this limit is imposed due to storage considerations or to limit processing time or due to the resolution of the output device. A pixel is the smallest subsection of the quad tree.

To summarize, a square or quadrant in the picture is either :

1. entirely one color
2. composed of 4 smaller sub-squares

To represent a picture using a quad tree, each leaf must represent a uniform area of the picture. If the picture is black and white, we only need one bit to represent the colour in each leaf; for example, 0 could mean black and 1 could mean white. Now consider the following image : The definition of a picture is a two-dimensional array, where the elements of the array are colored points

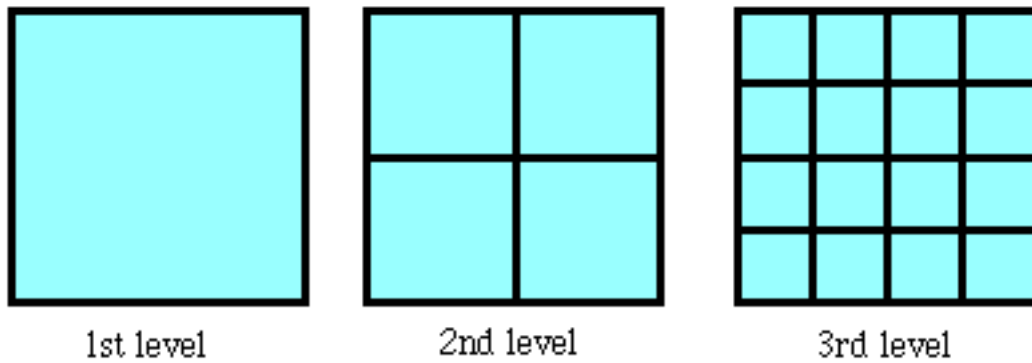


Figure 2.3: First three levels of quad tree

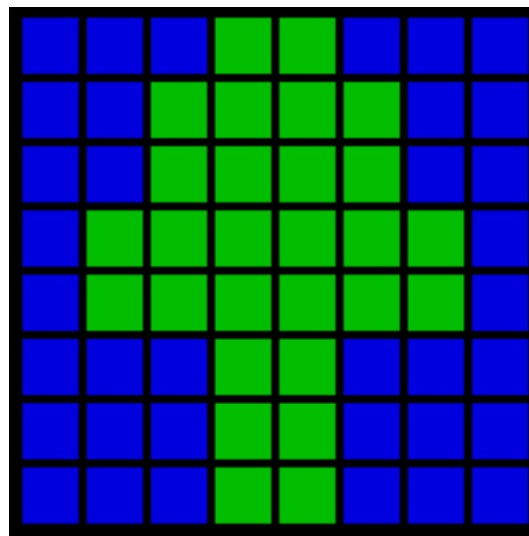


Figure 2.4: Given Image

This is how the above image could be stored in a quad tree :

2.3.4 Indexing Using Quadtrees:

Indexing procedure is triggered whenever a new image is submitted to the DB: the image is processed and a quadtree-based color structure descriptor is stored in the DB in the form of a XML document.

Retrieval is performed on the basis of a sketch drawn by the user, which is transformed into a quadtree structure consistent with the description of the images in the DB and sent to the retrieval engine. The retrieval module compares the description given by the user with those already available in the DB (in XML format) and returns to the user interface a sorted list of the images. Here is an schema representing *Indexing and retrieval* system:

Each image, which is submitted for indexing to the database, is partitioned using a quadtree structure, for achieving a compact representation of the color distribution

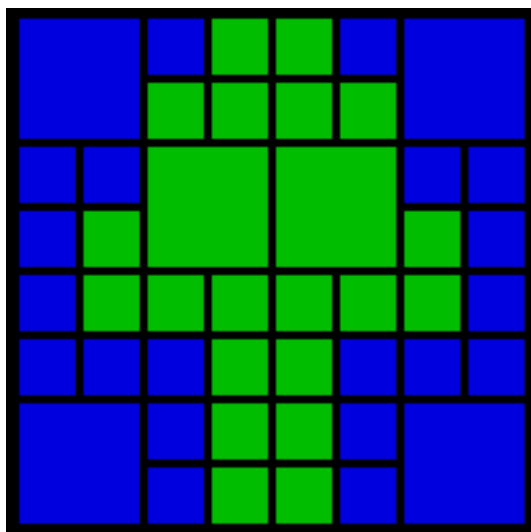


Figure 2.5: 8 x 8 pixel picture represented in a quad tree

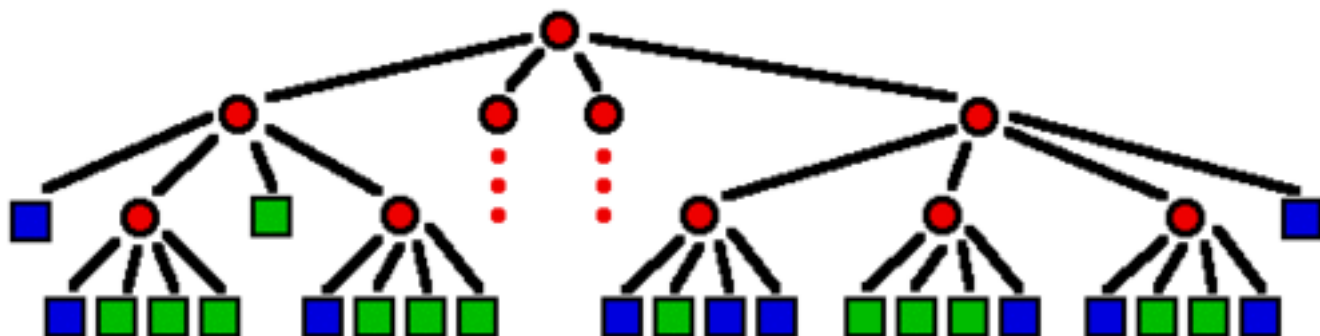


Figure 2.6: The quad tree of the above example picture. The quadrants are shown in counterclockwise order from the top-right quadrant. The root is the top node. (The 2nd and 3rd quadrants are not shown.)

in the image. In this work, the quadtree segmentation is used to extract a compact description of the distribution of colors in an image: a hierarchical structure is associated to the image, in which a dominant color is associated to leaves as well as to intermediate nodes of the quadtree.

The descriptor is extracted in 3 steps:

1. the image color space is quantized to 64 representative colors, following the recommendations by MPEG-7 committee.
2. the quadtree is recursively built from the color-quantized image, up to a given size of blocks represented by each leaf. Each leaf or node is assigned the *dominant color* in the corresponding image region. The dominant color is defined as the color with the higher percentage of occurrence inside the region represented by the node.

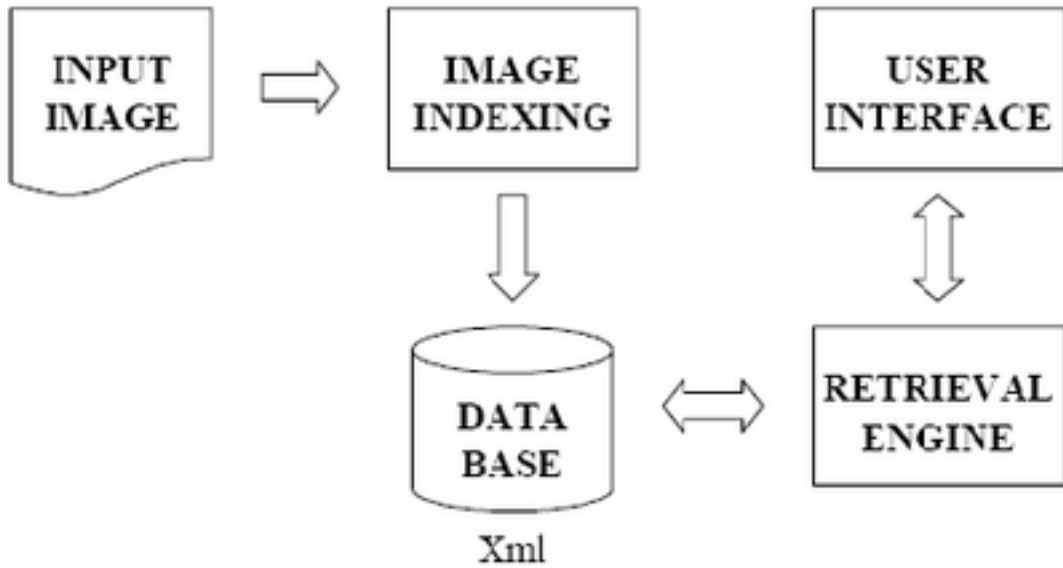


Fig. 1. The scheme of the proposed IR system.

For matching procedure, color structure descriptor is first extracted from sample image and then matched with the descriptors associated to the images contained in the DB. Now here we can have a result image in certain range of tolerance according to two criterion : **Quadtree Structure Similarity (QSS)** and **Quadtree Color Similarity(QCS)**. The main concept is that the difference in the structure of two quadtrees can be evaluated through the number of changes in the structure that need to be performed to make one of the quadtrees equivalent to the other. This process is called *quadtree warping*. Once the two quadtrees have the same structure, they are recursively navigated and the difference is computed between the colors of the corresponding leaves. The final formula as stated in [2], used for the **similarity matching(SM)** is the following:

$$SM = \alpha_1 QSS + \alpha_2 QCS$$

where the constants α_1 and α_2 are determined empirically for normalization and weighting of the two metrics

2.3.5 Advantages of quad trees include:

1. Quad trees can be manipulated and accessed much quicker than other models.
2. Erasing a picture takes only one step. All that is required is to set the root node to neutral.
3. Zooming to a particular quadrant in the tree is a one step operation.
4. To reduce the complexity of the image, it suffices to remove the final level of nodes.

5. Accessing particular regions of the image is a very fast operation. This is useful for updating certain regions of an image, perhaps for an environment with multiple windows.

The Main dis-advantage is that it take up lots of space.

2.4 R-Trees

R-trees are a N-dimensional extension of B+-trees, but are used for spatial access methods i.e., for indexing multi-dimensional information; for example, the (X, Y) coordinates of geographical data. Represent a spatial object by its minimum bounding rectangle (MBR). Supported in many modern database systems, along with variants like R+ -trees and R*-trees. The data structure splits space with hierarchically nested, and possibly overlapping boxes. The data for this section is taken from paper by Antonm Guttman [6]

A rectangular bounding box is associated with each tree node.

- Bounding box of a leaf node is a minimum sized rectangle that contains all the rectangles/polygons associated with the leaf node.
- The bounding box associated with a non-leaf node contains the bounding box associated with all its children.
- Bounding box of a node serves as its key in its parent node (if any)
- Bounding boxes of children of a node are allowed to overlap.

2.4.1 Structure of an R-tree node is defined as follows:

```
rtnodetype = record
    Rec1, ..., Reck : rectangle
    P1, ..., Pk : *rtnodetype
end
```

A polygon is stored only in one node, and the bounding box of the node must contain the polygon The storage efficiency or R-trees is better than that of k-d trees or quad-trees since a polygon is stored only once.

The **insertion** and **deletion** algorithms use the bounding boxes from the nodes to ensure that "nearby" elements are placed in the same leaf node (in particular, a new element will go into the leaf node that requires the least enlargement in its bounding box). Each entry within a leaf node stores two pieces of information; a way of identifying the actual data element (which, alternatively, may be placed directly in the node), and the bounding box of the data element.

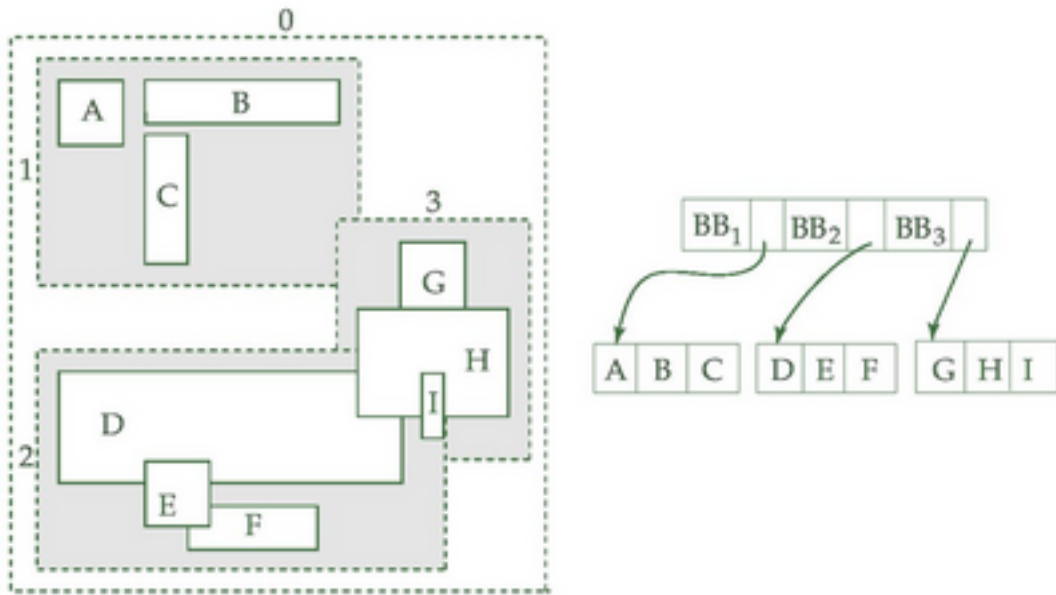


Figure 2.7: Sample R-tree

2.4.2 To insert a node

1. Find a leaf to store it, and add it to the leaf
 - To find leaf, follow a child (if any) whose bounding box contains bounding box of data item, else child whose overlap with data item bounding box is maximum
2. Handle overflows (here by overflow we mean if no of objects/rectangles inside given region increases too much) by splits. We may need to divide entries of an overflow node into two sets such that the bounding boxes have minimum total area .

2.4.3 To delete node from R-tree

1. Find the leaf (node) and delete object; determine new (possibly smaller) MBR
2. If the node is too empty (μ m entries):
 - delete the node recursively at its parent
 - insert all entries of the deleted node into the R-tree

2.4.4 Searching R-tree

Similarly, the searching algorithms (for example; intersection, containment, nearest) use the bounding boxes to decide whether or not to search inside a child node. Here we need to find **minimal bounding rectangle (MBR)**. In this way, most of the nodes in the tree are never "touched" during a search.

1. If the node is a leaf node, output the data items whose keys intersect the given query point/region
2. Else, for each child of the current node whose bounding box overlaps the query point/region, recursively search the child

Can be very inefficient in worst case since multiple paths may need to be searched. but works acceptably in practice.

2.4.5 Variants of R-tree

R* tree is a variant of R-tree for indexing spatial information. R* tree supports point and spatial data at the same time with a slightly higher cost than other R-trees.

R+ tree is a tree data structure, a variant of R-tree. It serves to index spatial data. They avoid overlapping of internal nodes by inserting an object into multiple leaves if necessary.

2.5 Types of Queries:

Now after defining these data structure our database is ready to answer fundamental queries [4] like

- **Whole Match Queries:** Given a collection of N objects O_1, \dots, O_n and a query object Q find data objects that are within distance ϵ from Q
- **Sub-pattern Match:** Given a collection of N objects O_1, \dots, O_n and a query (sub-) object Q and a tolerance ϵ identify the parts of the data objects that match the query Q
- **K- Nearest Neighbor queries:** Given a collection of N objects O_1, \dots, O_n and a query object Q find the K most similar data objects to Q .
- **All pairs queries (or "spatial joins"):** Given a collection of N objects O_1, \dots, O_n find all objects that are within distance ϵ from each other.

for solving such queries we first need to find a distance function between two objects and find one or more numerical feature-extraction functions (to provide a quick test). Then Use a SAM (e.g., R-tree) to store and retrieve k-d feature vectors.

here is an example of queries in context to computer games which uses these data structures :

- Visibility - What can I see?
- Ray intersections - What did the player just shoot?
- Collision detection - Did the player just hit a wall?
- Proximity queries - Where is the nearest power-up?

2.6 Comparison of Different Data Structures

- k-d trees are very easy to implement. However, in general a k-d tree containing k nodes may have height k, which causes the complexity of both insertion and search in k-d trees to be high. In practice, path lengths(root to leaf) in k-d trees tend to be longer than those in point quadtree because these trees are binary trees(as opposite to potentially having four children, as in case of point quadtree)
- R-trees have large number of rectangles potentially stored in each node, they are appropriate for disk access by reducing the height of the tree, thus leading to fewer disk access.
- One disadvantage of R-trees is that the bounding rectangle associated with different nodes may overlap. Thus when searching an R-tree, instead of following one path(as in case of quadtree), we might follow multiple path down the tree. This distinction grows even more acute when range search and neighbor searches are considered.
- In case of point quadtrees, while performing search/insertion each comparison requires comparisons on two coordinates, not just one. Deletion in point quadtree is difficult because finding a candidate replacement node for the node being deleted is often difficult.

Chapter 3

Metadata

Metadata are data about data. The term "meta" comes from a Greek word, denoting something of a higher or more fundamental nature. Broadly speaking, metadata can refer to any data that are used to describe the content, quality, condition and other aspects of data for humans or machines to locate, access and understand the data. Metadata information can help users to get an overview of the data.

3.1 Need of Metadata

Figure 3.1 shows an example of metadata contents for an image file. The image itself tells nothing more than the plain fact that it is a picture of a mountain view with snow. Without reading the associated metadata, it is impossible for a user to know the properties of the image such as who took the picture, when and where was the picture taken, what is the resolution of the picture etc, all of which are important information that helps to determine the suitability of the image for a particular application before the user takes a look at the actual data.

Metadata plays far more important role in managing multimedia data than does the management of traditional (well)-structured data. Some of the reasons as described by [12] are :

- **Different Query Paradigm** The exact-match paradigm for querying is no longer suitable or adequate for querying or retrieving various types of digital data.
- **Inadequate Processing Technique** Content-based processing techniques are too hard to analyze and very large data-sets are often limited or inadequate.
- **Lacking efficiency** When a content-based search is possible, it cannot be used very frequently (e.g. for every query) due to performance reasons and because of varying application.
- **Semantics of multimedia data** Derive and Interpreted data(which may be considered a part of metadata) as well as context and semantics (which may

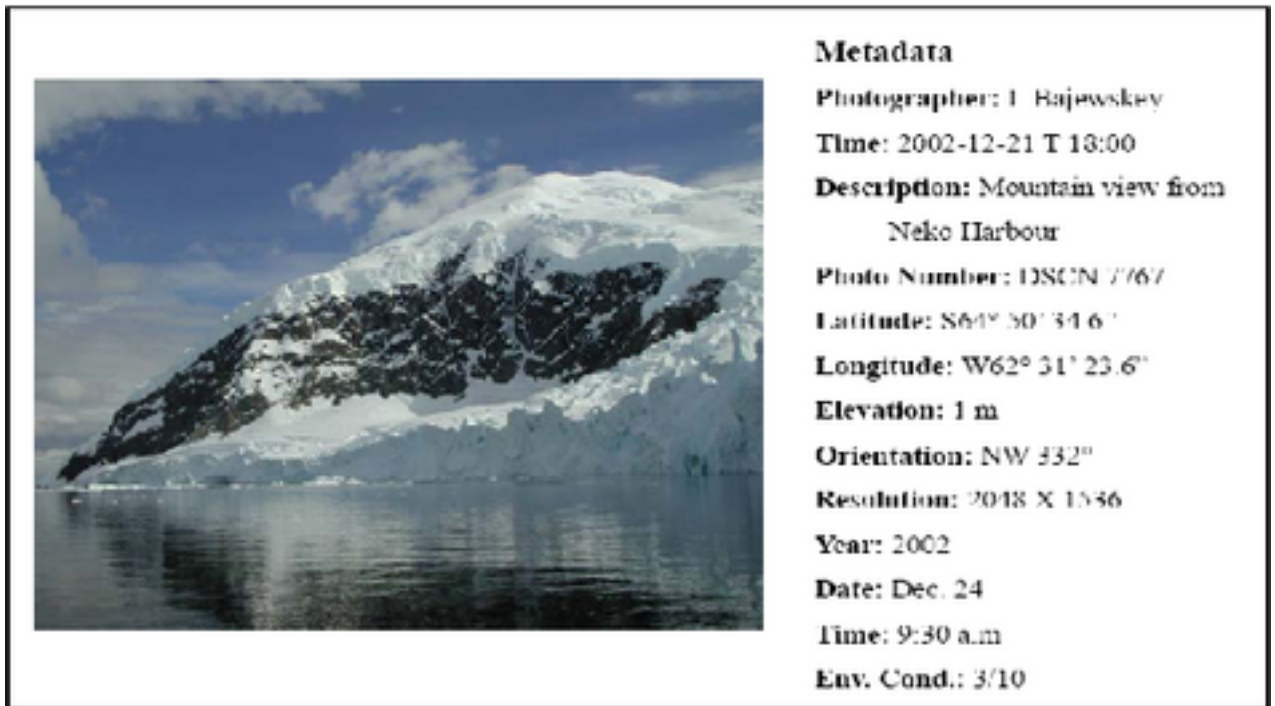


Figure 3.1: Image With Metadata[9]

be easier to base on metadata rather than raw data) are of greater value when dealing with multimedia data(especially audio-visual data).

3.2 Metadata Classification

Classifying Metadata to get suitable abstractions aids in exploring metadata.

3.2.1 Based on dependence on content

- **Content-independent metadata** This type of metadata captures information that does not depend on the content of the document with which it is associated. Example of this type are *location*, *modification-date* of document and *type-of-sensor* used to record it. There is no information content captured by these metadata but they still be useful for retrieval of documents from their actual physical location.
- **Content-dependent metadata** This type of metadata depends on the content of the document it is associated with. Example of content dependent metadata are *size* of a document, *max-colors*, *number of rows and columns* in an image. Content-dependent metadata can be further sub-divided as follows:
 - **Direct content-based metadata:** This type metadata is based directly on the contents of a document. A popular example of this is full-text indices based on the text of the documents. Inverted tree and document [4] vectors are examples of this type of metadata.
 - **Content descriptive metadata** This type of metadata describes the content of document without direct utilization of those contents. This

type of metadata often involves use of knowledge or human perception/cognition. An example of this type of metadata is textual annotations describing the contents of an image. This type of metadata come in two flavors

1. **Domain-independent metadata:** These metadata capture information present in the document independent of the application or subject domain of information. Example of these are the *C/C++ parse trees* and *HTML document type definition*.
2. **Domain-specific metadata:** Metadata of this type is described in a manner specific to the application or subject domain of information. Example of such metadata are *land-cover from GIS* and *population from Census domain*. In case of structural data, the database schema is an example of such metadata. Another example is *domain specific ontologies*, terms from which may be used to construct metadata specific to domain.

3.2.2 Hierarchical Classification

An another type of classification of metadata is possible as proposed by *Gilliland-Swetland (1998)* :[8]

- **Administrative** Metadata used in managing and administering information resources.
- **Descriptive** Metadata used to describe or identify information resources.
- **Preservation** Metadata related to the preservation management of information resource
- **Technical** Metadata related to how a system functions or metadata behave

3.3 Source of Metadata

Metadata can be extracted from various sources that are available from system. We distinguish four main categories [16] of metadata sources

- **Document content analysis:** One obvious source for metadata about an object is the object itself. An objectbased indexer generates metadata using the object independent from any specific usage. Typical content analyzers are keyword extractors, language analyzers for text documents or pattern recognizers for images.
- **Document context analysis:** When an object is used in a specific context and data about that context are available, we can rely on the context to obtain information about the object itself. One single learning object typically can be deployed in several contexts which provide us with metadata about it.

- **Document usage:** Real use of objects can provide us with more flexible and lively metadata than the sometimes more "theoretical" values provided by other metadata sources, or even by human indexers. Systems that track and log the real use of documents by learners are therefore a valuable source. These logs for example store the time spent reading a document or solving exercises. This metadata source category could be considered as a "usage context", and as such as a special case of document context analysis.
- **Composite documents structure:** In some cases, learning objects are parts of a whole but stored separately. In such a case, the metadata available for the whole is an interesting source for metadata about a component. Not only is the enclosing object a source, also the sibling components can provide relevant metadata. For example, one slide in a slide show often gives relevant context about the content of the next slide. This could be considered as a special case of document context, namely "aggregation related context".

3.4 Generation of Metadata

In the case of structured databases, the norm is to use schema descriptions and associated information (such as database statistics) as metadata. In the case of unstructured textual data and information retrieval, metadata is generally limited to indexes and textual descriptions of data. Metadata in such cases provides a suitable basis for building the higher forms of information. Metadata is commonly generated via three methods [12] :

- **Analyzing raw data** In many cases, media objects are analyzed and metadata is generated according to the focus of analysis. This is also known as explicit metadata generation.
- **Semi-automatic augmentation** Semi automatic augmentation of media results in addition meta-information, which cannot be derived from the raw material as such. Examples are the diagnostic findings of a doctor related to computer tomography image, which are based on doctor's experience and state of art in medicine.
- **Processing with implicit metadata generation** Metadata can be generated implicitly when creating raw media data. For example, a digital camera can implicitly deliver time and date for picture and video taken. Similarly, an SGML editor generate metadata according to document type definition when document is edited.

Generating the metadata can easily be a tedious task although using automatic tools may help. The task is more daunting when attempting to generate a huge volume of metadata without knowing the data, its usage, its background knowledge, and its accuracy, etc. Before generating the metadata, it is necessary to review all the relevant documentation about the data.

3.5 Metadata standards

Standards are an important mean to achieve common representation schemes and interoperability of system, and hence can play a pivotal role in exploiting metadata[12]. There are very many activities going on area including

- the development of a metadata taxonomy to help structure the discourse on metadata.
- the development of ontologies related to metadata attribute, and description of data elements and domains in terms of naming, typing, classification, and semantics.
- the definition of a meta-model registry structure to achieve mapping among different meta-model, and
- the definition of generic functionality for tools for the development and operation of metadata base.

Here are some organization providing metadata standards :

- The Dublin Core Metadata Element Set is a consensus which represents a simple resource description record that has the potential to provide foundation for electronic bibliography description. These standards are deliberately kept simple and flexible, so that authors can provide metadata by themselves (Simplicity of creation and maintenance), that any Internet document can be described with it (Commonly understood semantics, Extensibility, Interoperability among collections and indexing systems) and that it can be easily adapted into other languages (International scope and applicability). Some of **Dublin Core** metatags (placed in head section of webpage) are:

- **Title** the same as that given in the < TITLE > tag
- **Creator** person/organization responsible for the intellectual content. LastName, FirstName
- **Description** some search engines use this description as the summary of a page in the displayed search results
- **Subject** a list of keywords that describe the content of the page
- **Date.Created** YYYY-MM-DD
- **Identifier** URL of the page. The uniquely identifier for the page.

etc..

- The ISO 11179 standard addresses the specification and standardization of registration of data elements.
- The Meta Content Format (MCF) addresses the abstraction, standardization and representation of the structures used for organizing information.

3.6 Some points to remember while generating metadata

Some problems which should be kept in mind while generating metadata [13] :

- *Not doing it !* The costs of not creating metadata are much bigger than the costs, costs of not creating metadata: loss of information with staff changes, data redundancy, data conflicts, liability, misapplication, and decisions based upon poorly documented data.
- Don't try to cover all of the data resources with a single metadata record. A good rule of thumb is to consider how the data resource is used as a component of a broader data set or as a stand-alone product that may be mixed and matched with a range of other data resources.
- Human review matters. The whole process of creating metadata should not rely solely on automated tools.
- Assessments of consistency, accuracy, completeness, and precision about data are quite important. The methods for controlling the data quality include field checks, cross-referencing, and statistical analysis, etc.
- Metadata should be recorded throughout the life of a data set, from planning (entities and attributes), to digitizing (abscissa/ordinate resolution), to analysis (processing history), through publication (publication date).

Chapter 4

Multimedia database support by SQL and other packages

As we have seen the challenges faced due to multimedia objects (*Section 1.2*) Therefore a DBMS must provide domain types for such types of data to deal with integration of multimedia data. Here are basic datatypes provided for multimedia data :

- **Large Object Domains** these are long unstructured sequences of data often of two kinds
 - **Binary Large Objects** BLOBs, which are an unstructured sequence of bytes
 - **Character Large Objects** CLOBs, which are an unstructured sequence of characters

file references instead of holding the data, a file reference contains a link to the data we will see the use of OLE in Access.large objects at best allow you to extract sections or to concatenate them file references mean that the DBMS has no access the data at all.

Here is example on how to define table with large objects i.e LOB

```
CREATE TABLE grape
( grape_name VARCHAR2(30) ,
  grape_txt CLOB DEFAULT EMPTY_CLOB() ,
  picture BLOB DEFAULT EMPTY_BLOB() ,
  CONSTRAINT prim_grape PRIMARY KEY (grape_name) )
```

here grape_text is the the description of grape and is stream of characters so is stored as CLOB. Whereas picture is stored as BLOB. Here the image will be stored in database itself. Other way is we can store it as BFILE. Then we can have used : picture BFILE ,

Now while inserting data we have to do as :

```
INSERT INTO grape (grape_name, picture) VALUES ('chardonnay', BFILENAME('PHOTO_DIR', 'chardonnay.jpg' ))
```

4.1 Large Object Types in Oracle and SQL3

Oracle and SQL3 support three large object types:

BLOB : The BLOB domain type stores unstructured binary data in the database. BLOBs can store up to four gigabytes of binary data.

CLOB : The CLOB domain type stores up to four gigabytes of single-byte character set data.

NCLOB : The NCLOB domain type stores up to four gigabytes of fixed width and varyingwidth multibyte national character set data.

These types support the following operations:

Concatenation : making up one LOB by putting two of them together.

Substring : extract a section of a LOB

Overlay : replace a substring of one LOB with another

Trim : removing particular characters (e.g. whitespace) from the beginning or end

Length : returns the length of the LOB

Position : returns the position of a substring in a LOB

Upper and Lower : turns a CLOB or NCLOB into upper or lower case

4.2 SQL/MM

Within the international SQL standards efforts, SQL/MM is the responsibility of the committee ISO/IEC JTC 1/SC 32/WG 4 SQL/Multimedia and application packages. The current parts of SQL/MM are:

- SQL/MM Part 1: Framework
- SQL/MM Part 1: Framework
- SQL/MM Part 2: Full Text
- SQL/MM Part 3: Spatial
- SQL/MM Part 5: Still image
- SQL/MM Part 6: Data mining

here is an example of media table for still Images defined as per SQL/MM standards :

SI_MEDIA Table Definition :

```
CREATE TABLE PM.SI_MEDIA(  
    PRODUCT_ID NUMBER(6),  
    PRODUCT_PHOTO SI_StillImage,  
    AVERAGE_COLOR SI_AverageColor,  
    COLOR_HISTOGRAM SI_ColorHistogram,  
    FEATURE_LIST SI_FeatureList,  
    POSITIONAL_COLOR SI_PositionalColor,  
    TEXTURE SI_Texture,  
    CONSTRAINT id_pk PRIMARY KEY (PRODUCT_ID));  
COMMIT;
```

Oracle interMedia ("interMedia") contains the following information about object types that comply with the first edition of the ISO/IEC 13249-5:2001 SQL MM Part5:StillImage standard (commonly referred to as the SQL/MM Still Image standard):

4.3 Oracle's interMedia

The capabilities of interMedia audio, image, and video include the storage, retrieval, management, and manipulation of multimedia data managed by Oracle8i. Data for this section is taken from *Oracles intermedia Webpage*

Oracle interMedia supports multimedia storage, retrieval, and management of:

BLOBs stored locally in Oracle8i and containing audio, image, or video data

BFILEs, stored locally in operating system-specific file systems and containing audio, image, or video data

URLs containing audio, image, or video data stored on any HTTP server such as Oracle Application Server, Netscape Application Server.

Oracle also stores metadata including:

source type, location, and source name

MIME type and formatting information

characteristics such as height and width of an image, number of audio channels, video frame rate, pay time, etc.

interMedia provides the *ORDAudio*, *ORDImage*, and *ORDVideo* object types and methods for:

- manipulating multimedia data source attribute information
- extracting attributes from multimedia data

- getting and managing multimedia data from Oracle interMedia, Web servers, and other servers.
- performing a minimal set of manipulation operations on multimedia data (images only). It provides functions to describe color Histogram, Texture, Average Color (As per SQL/MM standards)
- performing description attribute manipulation, file operations (open, close, trim, read, and write) on the source, comments attribute manipulation, and processing commands (*processAudioCommand* and *processVideoCommand*) to operate on the multimedia data (*interMedia audio and video* only).

The properties available are:

ORDImage : the height, width, data size of the on-disk image, file type, image type, compression type, and MIME type

ORDAudio : the format, encoding, number of channels, sampling rate, sample size, compression type, and audio duration.

ORDVideo : the format, frame size, frame resolution, frame rate, video duration, number of frames, compression type, number of colors, and bit rate.

Chapter 5

Epilogue

5.1 Conclusion

We saw how multimedia data should be stored using advanced data structures and with aid of metadata in order to make retrieval and search process easier. This report presented an approach on how images can be stored using data structures like quad-trees and can be searched then. These data structures help us to extract features from data like image, video so that we can perform content based queries. We also discussed the advantage and disadvantage of these data structures. But sometimes the indexing and searchin process consume lots of time in case of large database. So we need help of metadata to make that process faster which do not require to extract features and information from data itself. This report also presented how metadata is generated and it mentioned several issue that have to be tackled. It was also seen that how metadata can be classified so that depending upon context better use of it can be made. We also saw how metadata standards can help us exploiting use of metadata. Finally we discussed some of features provided by *sql/mm* and *oracle's intermedia* to support multimedia database

5.2 Future Scope

These data structure can be made more useful(faster retrievaland more accurate) by developing better search heuristic (similarity function) for states. More work is required in direction of interoperability and standards as no. of multimedia application are increasing tremendously. Also functionality of metadata can be extended by better technique for automatic metadata creation and harvesting metadata created by human[12].

Bibliography

- [1] V. S. Subrahmaniam : *Principles of Multimedia Database System*, Morgan Kaufmann Publishers ,1998.
- [2] F.G.B. De Natale and F. Granelli : *Structure-Based Image retrieval using as structured color descriptor* Workshop on Content-Based Multimedia Indexing (CBMI'01), pages 109-115, Brescia (Italy), sept. 2001.
- [3] Joseph Kuan and Paul Lewis : *Fast k nearest neighbour search for R-tree family* In Proc. of First Int. Conf. on Information, Communication and Signal Processing, pages 924–928, Singapore, 1997. 2.2
- [4] Christos Faloutsos *Indexing Multimedia Database* Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 22-25,
- [5] Walid G. Aref and Ihab F. Ilyas : *An extensible index for spatial database* Proceedings of the 13th International Conference on Scientific and Statistical Database Management, July 18-20, 2001, George Mason University, Fairfax, Virginia, USA
- [6] Antonm Guttman: *R-trees A dynamic index structure for spatial database* Proceedings ACM SIGMOD Conference, pp.47-57, Boston, MA, 1984.
- [7] Sherry Marcus and V.S. Subrahmanian: *Foundation of multimedia database system* Volume 43 Journal of ACM (May 1996)
- [8] Sussane Boll, Wolfgang Klas and Amit Sheth : *Overview on using Metadata to Manage Multimedia Data* book:Multimedia Data Management pages 1-24
- [9] Yuchai Zhou: *An approach to building Metadata for Georeferenced Multimedia Data* GEOG 5905 : Master's Research Workshop, Carleton University
- [10] Reshma Suvarna, K. Seluk Candan¹, Huan Liu, Jong Wook Kim: *Structure-based Mining of Hierarchical Media Data, Meta-Data, and Ontologies* The 5th International Workshop on Multimedia Data Mining (MDM/KDD2004)
- [11] Robert Garcia and Oscar Celma: *Semantic Integration and Retrieval of Multimedia Metadata* Proceedings of the 5th International Workshop on Knowledge Markup and Semantic Annotation (SemAnnot 2005) to be held with ISWC 2005, Galway, Ireland, 7 November 2005.

- [12] Jane Greenberg: *Metadata Extraction and Harvesting: A Comparison of Two Automatic Metadata Generation Applications*. Journal of Internet Cataloging, 6(4): 59-82.
- [13] Yu Deng: *The Metadata Architecture for Data Management in Web-based Choropleth Maps* Department of Computer Science, University of Maryland.
- [14] Kris Cardinaels, Michael Meire, Erik Duval: *Automating Metadata Generation: the Simple Indexing Interface* Proceedings of the 14th international conference on World Wide Web.
- [15] Ji-Rong Wen, Qing Li, Wei-Ying Ma, Hong-jiang Zhang: *A Multi-paradigm Querying Approach for a Generic Multimedia Database Management System* SIGMOD Record, Vol. 32, No. 1, March 2003
- [16] Kris Cardinels, Michael Meire and Erik Duval: *Automatic Metadata Generation :the Simple Indexing Interface* international conference on World Wide Web May,2005
- [17] Peter van Oosterom: *Spatial Access Methods* Chapter in Geographical Information Systems Principles, Technical Issues, Management Issues, and Applications (edited by Longley, Goodchild, Maguire en Rhind), Wiley pages 385-400 (vol.1), 1999.