Document Warehousing Based on a Multimedia Database System

Hiroshi Ishikawa, Kazumi Kubota, Yasuo Noguchi, Koki Kato, Miyuki Ono, Naomi Yoshizawa, and Yasuhiko Kanemasa
Software Laboratory, FUJITSU LABORATORIES LTD.
4- 1- 1 Kamikodanaka, Nakahara-Ku, Kawasaki, 2 1 l-8588 Japan
hiro@flab.fujitsu.co.jp

Abstract

Nowadays, structured data such as sales and business forms are stored in data warehouses for decision makers to use. Further, unstructured data such as emails, html texts, images, videos, and office documents are increasingly accumulated in personal computer storage due to spread of mailing, WWW, and word processing. Such unstructured data, or what we call multimedia documents, are larger in volume than structured data and precious as corporate assets as well. So we need a document warehouse as a software framework where multimedia documents are analyzed and managed for corporate-wide information sharing and reuse like a data warehouse for structured data. We describe a prototype document warehouse system, which supports management of simple and compound documents, keyword-based and content-based retrieval, rule-based classification, SOM-based clustering, and XML data query and view rules.

1. Introduction

Nowadays, structured data such as sales and business forms are stored in data warehouses for decision makers to use. Further, unstructured data such as emails, html texts, images, videos, and office documents are increasingly accumulated in personal computer storage due to spread of mailing, WWW, and word processing. Such unstructured data, or what we call multimedia documents, are larger in volume than structured data and precious as corporate assets as well. Traditionally, they are handled as bulk data or BLOB (binary large objects) and are subject to no interpretation by DBMS. So we need a software framework where multimedia documents are analyzed and managed for corporate-wide information sharing and reuse like a data warehouse for structured data. We propose a document warehouse realizing such a framework.

Document warehouses need to satisfy the following requirements at least.

(1) Document files and folders managed by document warehouses must be independent of physical details such as names and locations. Such attributes are often changed, which makes document sharing difficult.

(2) Document warehouses must provide users with methods for analyzing contents of documents for further processing. They must enable users to abstract keywords from texts and feature data from images and videos.

(3) Document warehouses must provide a query facility based on attributes of documents and keywords contained by documents, needless to say. Browsing alone is insufficient because a large amount of media data take long time to play.

(4) Document warehouses must also provide a content-based retrieval facility. Generally, images and videos cannot be attached appropriate keywords in advance. We need an alternative approach to retrieving such media data. We must provide a facility which allows users to retrieve documents by expressing feature data such as colors, layouts, and motions. Such a content-based retrieval facility enables inexact match in contrast to exact match facilitated by keyword-based retrieval.

(5) Document warehouses must allow users to refine queries in a step-wise fashion by combining both keyword- and content-based query facilities of related multimedia documents. Bidirectional access of interrelated documents is also needed.

(6) Document warehouses must provide a facility for automatically classifying documents according to user-supplied viewpoints such as keywords and features. In particular, we must allow overlapping classification where a single document belongs to different groups. In an ordinary file system, the user could overcome this issue by making aliases, but this inflexible approach imposes unnecessary efforts upon users.

(7) Document warehouses must provide a facility for automatically clustering documents based on similarity of features associated with documents. This helps as a first step for further document analysis.

(8) Document warehouses must be able to manage compound documents, which are groups of relevant data such as texts, images, and videos as components.

0-7695-0071-4/99 $10.00 © 1999 IEEE

Compound documents must be handled like simple documents.

(9) Document warehouses must allow users to retrieve documents such as XML data scattered over WWW sites and reuse them in applications for their custom needs. A view facility for packaging such actions on documents is mandatory like relational views.

(10) Document warehouses must support business rules for associating documents with business processes. This is necessary for document-based business solutions.

The above requirements are of course not comprehensive. In fact, version control and scalability are also important issues, but they are not addressed in this paper due to space limitation. We have already proposed a multimedia database system based on an object-oriented database (OODB) for general networked multimedia applications [4]. Document warehouses in this paper customizes and enhances the multimedia database system for use in corporate-wide document sharing and reuse. Please note that document warehouses are an integration of database technology and text, image and video processing. In the following section, we describe our approach to document warehouses based on a multimedia database system.

2. Implementation

2.1 Basic Concepts

In this subsection we describe basic concepts, that is, simple and compound documents, document folders, and classification. To address all the issues described in Section 1, we assume a multimedia database system which enables flexible and efficient acquisition, storage, access and retrieval, and distribution and presentation of large amounts of heterogeneous media data [4]. We take a realistic approach based on OODB [3], which is more suitable for description of media structures and operations than a traditional relational database (RDB). We have enhanced OODB by supplying Event-Condition-Action (ECA) rules [2].

First, we define documents in document warehouses. In document warehouses, simple documents are implemented as objects in terms of OODB. A simple document may be any type such as text, image, and video. Document objects have attributes such as date and creator. Document objects also have attributes representing physical information such as file descriptors. Documents are referenced through object identifiers. Therefore, documents can be unchangeably accessed even though physical attributes are changed when documents are updated or moved. In other words, documents in document warehouses are more logical than ordinary files in that they are independent of physical aspects. Document warehouses provide dedicated analyzers for multimedia documents. Document analyzers abstract and attach keywords and feature data to documents, which are used in retrieval, classification, and clustering, as described in detail later.

Document folders in document warehouses are defined as a set of documents. Of course, the user can manually insert documents into such folders. In addition, the user can attach classification rules to document folders. Such classification rules are implemented as an index whose entry is a pair of folder names (identifiers) and keywords. They are many-to-many relationships. Thus, one document, in general, can belong to more than one folder. When documents are checked in document warehouses, documents are automatically classified into relevant folders according to user-supplied classification rules if any. For example, doc1 is classified into three folders by applying three rules (See Figurel). The user can browse documents associated with keywords attached to folders by just opening the folders.

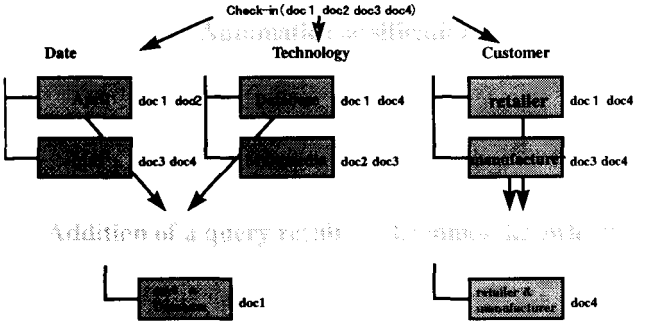

Figure 1. Classification and folder query

Classification rules are implemented as a special case of event-condition-action (ECA) rules [2] supported by a multimedia database as a kernel of document warehouses. Thus, in a case of classification rules, an event is check-in, a condition is keyword check, and an action is insertion. ECA rules can be used to implementing business rules. In this case, actions implement business processes.

As folders are sets of documents, set operations such as union, intersection, and difference can be performed upon folders. A user's query intersecting two folders is evaluated to produce a new folder containing documents which belong to both of them. For example, "April and database" folder contains a set of documents related to database technology and created in April. Further, set operations can help users to discover important documents as common knowledge. We can create a set of common documents by intersecting "retailer" and "manufacturer" folders (See Figure 1).

In contrast to simple documents, a compound document is implemented as a special folder containing related documents such as texts and images as its components.

Compound document folders are also document objects and given object identifiers like simple documents. They are handled similar to simple documents. For instance, we can obtain texts and images within a hard-copy document by using OCR systems. Then they are analyzed by dedicated analyzers, such as keyword abstractors and feature abstractors, and are checked in document warehouses as compound document folders. They are inserted into document folders. They are retrieved based both on keywords and contents. For instance, we assume that compound documents have both keywords of component texts and features of component images. If the user retrieves documents by keywords of texts, the system returns images associated with matching texts at the same time and vice versa.

In some applications, a compound document consists of a form and an email. If a form is filled in, an email is sent to a person in charge. These business rules are implemented by attaching ECA rules to compound documents.

2.2 Approach to texts

Now we concretely discuss an approach to text data management, focusing on HTML page management. First, mapping between text contents and formats such as HTML, SGML is necessary. We resolve such heterogeneity by using polymorphism of objects [3]. Moreover, we need to allow users to acquire texts and reorganize them for further distribution. To this end, we provide HTML page management by databases including storage and retrieval. The system abstracts keywords from texts of collected HTML pages automatically and stores keywords and URL associated with texts into databases. Either HTML texts themselves, or only their file names and URL are stored in databases. In-line images as components of HTML texts are also stored by databases. The system adds URL, file names, titles, anchor character strings (i.e., links), data types (e.g., GIF, JPEG) as default keywords. The user can delete or add favorite keywords. We prefer the recall ratio to the precision ration of keyword-based retrieval. Relatively-addressed links (i.e., URL) are transformed to absolutely-addressed links. The users can retrieve pages or components by a wide variety of keywords and drag and drop retrieved pages and components into work pages to create new home pages in a WYSIWYG fashion. Content-based retrieval of texts is facilitated by using a full text search engine. Of course, if classification rules specifying correspondence among folders and keywords are attached to folders, then texts are classified into appropriate folders based on keywords analyzed in the above way.

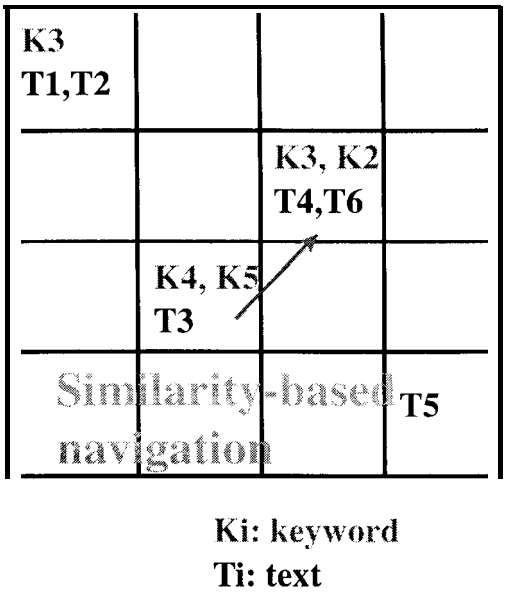

Figure 2. Retrieval map as a result of SOM-based clustering

Now we describe logical clustering by using texts as an example. Ease of data acquisition through WWW, however, makes the size of collected data unmanageable for the user. Keyword-based retrieval alone is not sufficient. So we logically cluster texts to enable similarity-based navigational search for document exploration. The system automatically abstracts keywords from collected HTML or SGML texts. Then the system chooses most frequent 100 keywords contained by a set of texts and places each text in the information space of 100 dimensions ranging from having the corresponding keyword to not having it. The system uses a *Self-Organizing Map* (SOM) [6] technique to logically cluster a set of collected texts into the given number of groups in the retrieval space of manageable dimensions. The system displays the structured map by using 2-D or 3-D graphics such as VRML. The user can retrieve texts by navigating a 2-D or 3-D user interface (See Figure 2). We use SOM because it helps to reduce high dimensions of characteristic vectors and map data into a 2-D or 3-D GUI.

More and more XML texts have been published on WWW. XML texts are of course documents in our sense. However, some kinds of XML texts should be viewed as records or objects of databases and be rather called XML data. In other words, XML data scattered over various WEB sites can be distributed databases in general terms. Therefore, we must provide a query facility over distributed XML data. We devise an XML data query language and an XML data view facility based on the query language for customizing XML data and packaging business rules.

The user defines XML data view rules by specifying

target and condition parts of a query. The basic unit of an XML data query is an extended tag expression. It is an URL plus document names followed by a series of tag names. In the condition part, the user compares extended tag expressions with constant values or other extended tag expressions. The latter case is similar to RDB join predicates. In the target part, the user specifies how to bind query results to client applications. The target specification includes inserting to folders, listing, embedding to spread sheets, and other report generation. For example, this rule compares financial results of 1996 and 1997 stored in local and official sites company by company and displays the results as lists (See Figure 3).

Condition:

	A	B	C
1	URL/Report.Accounts		
2	Year	=	1997
3	Company	=	A4.Company
4	local/Report.Accounts		
5	Year	=	1996
6	Company		

Target:

1997 Accounts		
Company	Sales	Increase
A1.Company	A1.Sales	A1.Sales - A4.Sales

Figure 3. XML data view rule

In our temporary implementation, XML data view rules are specified like spread sheet applications. The XML data view rules are parsed and translated into an XML data query (See Figure 4). The system further translates the query into local and global queries and optimizes an execution sequence of access methods. The index manager creates and maintains appropriate indices for each tag such as hash or B-tree. The system uses meta data expressed as XML data, such as data size and update time, for optimization if any.

```
select A1.Company, A1.Sales, A1.Sales-A4.Sales
from        A1: URL/Reports.Accounts,
            A4: local/Reports.Accounts
where A1.Year = 1997 and A4.Year = 1996
      and   A1.Company= A4.Company
```

Figure 4. XML data query

2.3 Approach to videos and images

Logical video contents need semantic description of contents, that is, what are recorded. Moreover, logical contents are recorded in several ways, that is, in CODEC such as MPEG and Motion JPEG, and in quality such as frame sizes and rates. Long-duration play of whole video streams is not always required. Users should rather have partial access to video streams in order to jump to only their necessary portions. We allow users to access to video streams with uniform interfaces independent of CODEC by using polymorphism of objects.

To facilitate interactive retrieval of multimedia, we enable users to flexibly and efficiently access partial data such as sub streams of videos by temporal information (e.g., temporal intervals), keywords, and other related information. This technique of subsetting a large amount of media data is analogous to RDB views. A stream view selects a subset of logical contents by specifying a time interval. Keywords are attached to views for keyword-based retrieval. Feature data, such as figures, colors, and motion directions, are attached to frames of streams corresponding to views for content-based retrieval. Logical contents have several physical streams of different quality, which are chosen for appropriate QOS control in playback.

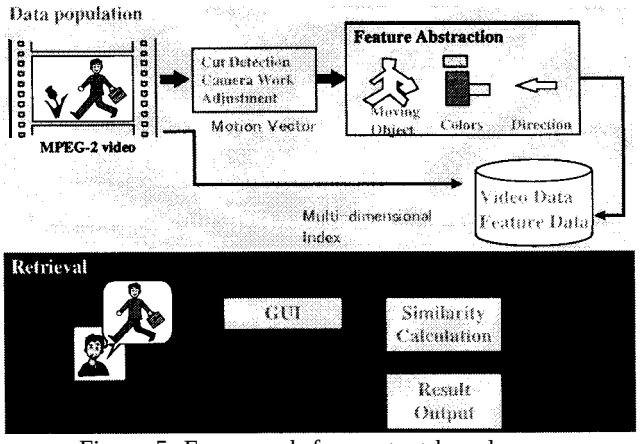

Figure 5. Framework for content-based query

We describe methods for feature analysis [5] (See Figure 5). We use a light-weight technique to segment scenes and recognize moving objects for content-based retrieval of stream data. First, the system can detect scene cuts by using differences in successive frames, such as motion vectors of macro blocks of MPEG and colors. Thus, in MPEG coding, we abstract motion vectors of macro blocks by taking advantage of similarity between successive frames and make motion compensation by using such motion vectors. Motion compensation, however, becomes difficult at the point between successive different scenes. At the point, macro blocks with no motion compensation become dominant. So we can detect cuts by

checking such macro blocks. To enhance the precision of cut detection, we use difference of colors between successive frames as well.

Then the user can define views of streams (i.e., sub streams) by attaching keywords to such cut scenes. Keywords of stream views enable association between video data and other media data such as texts. Further, the user can define new views recursively by combining existing stream views. The user can retrieve sub streams corresponding to views by using such keywords. The system also chooses representative frames within a scene and abstracts feature data and stores them into databases. Please note here that matching with representative frames can reduce the recall ratio of a content-based query since feature data, such as colors and layouts, may change even within a single scene.

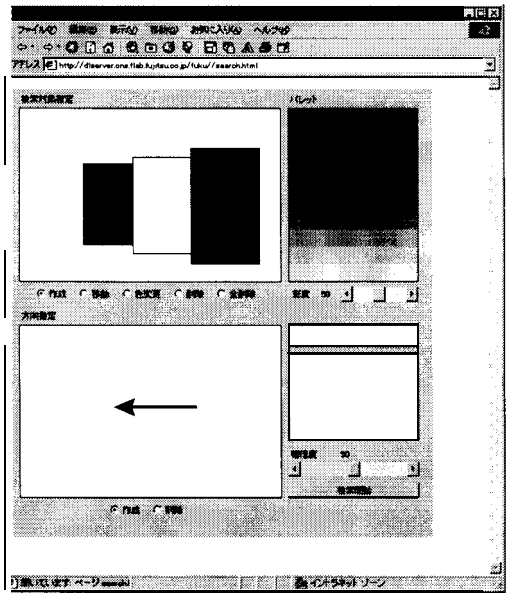

Figure 6. GUI for content-based retrieval of videos

The system can also detect moving objects by using motion vectors of MPEG. The system decreases the number of colors to more accurately recognize moving objects. The system also compensates camera work such as zoom and pan and detects moving objects by using motion vectors of MPEG. The system stores motion directions in addition to figures and colors associated with moving objects as a multi-dimensional index such as R-tree and Quad-tree. The system allows the user to do content-based retrieval by using this multi-dimensional index. Thus, the user can illustrate a sample of user-specified colors, figures, and motion directions through GUI (See Figure 6). A sample figure consists of several parts like a car. The system abstracts feature data from the user-specified sample. The system uses the largest part, such as a rightmost green rectangle, as a search key to Quad-tree indices. Thus, the system selects an appropriate index for the user-specified direction and searches the primary key part against the index. The system evaluates the other parts as additional conditions of a query by using the selected index. The retrieval results are ranked based on the similarity of matches and are tiled to display. The system tiles still images for representative frames of retrieved video scenes in order of similarity ranks. Then the user can choose one representative frame and play an associated scene (See Figure 7). The system allows the users to retrieve video sub streams containing user-specified moving objects without any interference from the background information because the system can distinguish the moving objects and the backgrounds unlike other approaches such as QBIC [1].

Figure 7. Result of content-based video retrieval

Next we describe an approach to images as components of documents. The system abstracts feature data such as colors and layouts from images like videos. First, the system divides a given image into several regions representing different colors by reducing the number of colors of the image. Next the system approximates each region by a rectangle (i.e., a bounding box) and store its color, size, and location within the whole image as a multi-dimensional index.

The user can retrieve images by specifying a probe consisting of colored rectangles through GUI similar to that of video retrieval. In fact, the user cannot specify only motion in retrieval of images unlike retrieval of videos. Of course, the system allows the user to retrieve images by specifying attributes and keywords. Classification rules can

be extended for accommodating feature data of images and videos instead of keywords of texts. The system also allows logical clustering of images and videos, based on their feature data.

The system stores features associated with moving objects and still images as a multi-dimensional index such as R-tree and Quad-tree. We have implemented these two methods and have selected the Quad-tree method. The index creation time of the Quad-tree method is shorter than that of the R-tree method because the former doesn't reconstruct trees in insertion. In a situation, such as content-based retrieval, where the minimum search range is fixed, the search time of the two methods is comparable. We have implemented a three-dimensional index to represent three primitive color coefficients of figures. In our current implementation, we have not included a dimension for the direction of moving objects. Instead, we use eight indices for eight motion directions such as, up, down, left, right, and their middles. Each index has three dimensions corresponding to three primary color coefficients of objects moving in the same direction. In still image retrieval, we use only one index.

2.4 Approach to multimedia query

In real situations, single use of a retrieval facility such as keyword-based or content-based retrieval facilities is insufficient for retrieving necessary multimedia documents only. Thus, we need integrated use of various facilities. We take two complimentary approaches to this issue. One approach is that we allow users to refine queries in a step-wise fashion by applying keyword-based retrieval and content-based retrieval facilities in succession. This approach is useful if texts are related to whole video streams. Another approach is that we relate texts and video streams portion by portion and allow bidirectional direct access. This is useful if audio data of video streams can be transcribed into texts and parts of texts and video streams can be interrelated on a time basis.

We discuss about synergy between mutimedia data and that between OODB and document management. All data handled by document warehouses are represented as objects managed by OODB. Both keyword-based and content-based retrieval of media data such as texts, images, and videos, produces a subset of objects. Clustering of media data from both a keyword and feature point of view also makes a subset of objects. Therefore, combination of such facilities helps to refine search results.

The user can relate texts, images, and videos either by attaching explicit links, making compound documents, or time-based joining. In particular, we cut video data by a minute and translate audio data of a little longer than a minute into texts by using a voice-recognition system. We relate video and text data based on time codes.

We directly use query processing of underlying OODB for attribute-based retrieval of media data. Keyword indices of text data and content-based indices of image and video data are B-trees or Quad-trees implemented as applications of OODB although we don't use query processing of OODB directly. Further, we could add a full-text retrieval engine as an application of OODB.

3. Conclusion

In this paper, we have proposed a document warehouse for corporate-wide multimedia document sharing and reuse, based on an object-oriented multimedia database system. We have implemented a prototype document warehouse system to verify the proposed approach. This prototype supports management of simple and compound documents, keyword-based and content-based retrieval, rule-based classification, SOM-based clustering, and XML data query and view rules.

Our system is unique in its enabling technologies: Its document model is based on objects and folders, and its classification uses ECA rules, and its content-based retrieval focuses on object motion. In particular, its XML data query and view rule facility is totally new. Further, our document warehouse framework integrating the various facilities has no comparable work to our knowledge. It has been applied to in-house applications and has proved to be effective. We plan to enhance the functionality and performance of our system in order to make the system applicable to industrial applications.

References

[1] Flickner, M. et al: Query by Image and Video Content: The QBIC System, IEEE Computer Vol.28, no.9, 1995, pp.23-32.

[2] Ishikawa, H., et al.: An Active Object-Oriented Database: A Multi-Paradigm Approach to Constraint Management, Proc. the 19th VLDB Conference, pp.467-478 (1993).

[3] Ishikawa, H., et al.: An Object-Oriented Database System Jasmine: Implementation, Application, and Extension., IEEE Trans. Knowledge and Data Engineering, vol. 8, no. 2, pp.285-304 (1996).

[4] Ishikawa, H., et al.: An Extended Object-Oriented Database Approach to Networked Multimedia Applications. Proc. IEEE 14th Intl. Conference on Data Engineering, (1998).

[5] Kato, K., Kondo, A., and Ishikawa, H. : Multimedia Database InfoServer - Script and Video Playback (in Japanese), Proc. 7th Data engineering Workshop, 1996, pp. 109-114.

[6] Kohonen, T.: ***Self-Organizing Maps,*** Springer-Verlag (1995).