94-MS

5th International Hong Kong Computer Society Database Workshop

26th February, 1994
Hong Kong

Sponsors:

Borland (HK) Ltd
Hong Kong Computer Society
City Polytechnic of Hong Kong (Computer Science Department)
Epro Systems Ltd
Microsoft (HK) Ltd
Oracle Systems (HK) Ltd
Sybase Hong Kong Ltd
Informix Software (HK) Ltd
Computer Associates International Ltd

ISBN : 962-442-046-5

Publisher: Department of Computer Science, CPHK

PROCEEDINGS of

5th International Hong Kong Computer Society

Database Workshop on

NEXT GENERATION DATABASE SYSTEMS

February 25-26, 1994

Organized by

Hong Kong Computer Society, Database Special Interest Group

in cooperation with
Borland (HK) Limited

and

Sponsored by

City Polytechnic of Hong Kong(Computer Science Department)
Computer Associates International Limited
Sybase Hong Kong Limited
Informix Software(HK) Limited
Microsoft Hong Kong Limited
Oracle Systems Hong Kong Limited
EPRO Systems (HK) Limited

Edited by

Dr S L Hung
City Polytechnic of Hong Kong

Professor Francis Chin
University of Hong Kong

Compiled by

Dr Joseph Fong
City Polytechnic of Hong Kong

ORGANIZING COMMITTEE

Chairman:	Joseph Fong, City Polytechnic of Hong Kong
Vice Chairman:	Danny Ha, Databank Training&Consulting Services
Secretary:	Douglas Chung, Hong Kong Computer Society, DBSIG
Treasurer:	Penny Pang, Hong Kong Baptist College
Publication:	S.L. Hung, Hong Kong Computer Society
	Vincent Wong, City Polytechnic of Hong Kong
Registration:	Theresa Leung, Hong Kong Computer Society
Publicity:	Irene Kwan, Hong Kong Polytechnic
	Felix Wu, Jardine Metro Link Ltd, Hong Kong
Tutorial:	Q Li, HongKong University of Science&Technology
	K Karlapalem, Hong Konv Uni. of Sci.&Technology
Promotion:	Charles Tse, Hong Kong Computer Society, DBSIG
	Karan Wan, Hong Kong Computer Society, DBSIG
	Michael Tsui, Hong Kong Computer Society, DBSIG
	Raymond Lo, Lethman Brothers Asia Holding Ltd
Local Arrangements:	S. H. Tong, Hong Kong Baptist College
	Bonnie Lau, Chinese University of HongKong
	Handy Khow, HongKong Computer SocietyDBSIG
	Paul Kwan, HongKong Computer SocietyDBSIG

PROGRAM COMMITTEE

N V Balasubramanian, *City Polytechnic of Hong Kong, Hong Kong*
Don Batory, *University of Texas, USA*
Chris Bloor, *University of Sunderland, UK*
Sharma Chakravarthy, *University of Florida, USA*
Francis Chin, *University of Hong Kong, Hong Kong*
T.Y. Cheung, *City Polytechnic of Hong Kong, Hong Kong*
Stephen Faris, *Oracle Hong Kong Limited, Hong Kong*
Patrick Hui, *Borland (Hong Kong) Limited, Hong Kong*
S.L. Hung, *Hong Kong Computer Society, Hong Kong*
Laurie Kan, *Microsoft Hong Kong Limited, Hong Kong*
Ernest Lam, *Hong Kong Baptist College, Hong Kong*
John Lee, *Hong Kong Polytechnic, Hong Kong*
Gary Leung, *Sybase Hong Kong Limited, Hong Kong*
Kwong Sak Leung, *Chinese University of Hong Kong, Hong Kong*
Fred Lochovsky, *Hong Kong University of Science & Technology*
Akifumi Makinouchi, *Kyusyu University, Japan*
Y. Masunaga, *University of Library&Information Science, Japan*
Dennis McLeod, *University of Southern California, USA*
Sham Navathe, *Georgia Institute of Technology, USA*
Louiqa Raschid, *University of Maryland, USA*
Gunter Schlageter, *University of Hagen, Germany*
Allan Smith, *Computer Associates International Ltd, Hong Kong*
Peter Smith, *University of Sunderland, UK*
Stanley Su, *University of Florida, USA*
Wei Sun, *Florida International University, USA*
Graham Tate, *City Polytechnic of Hong Kong, Hong Kong*
Charles Tse, *Hong Kong Computer Society, DBSIG*
Anthony Wan, *Informix Software (Hong Kong) Limited, Hong Kong*
K.Y. Whang, *Korea Advanced Institute of Science & Technology*

FORWARD

The papers in this proceedings were presented at the 5th International Hong Kong Computer Society Database Workshop in Hong Kong on February 26, 1994. The aim of this workshop is to provide a forum for database researchers and practitioners to share views and experiences on the problems and development of the "Next Generation Database Systems". There were 26 submitted papers for the workshop (not including the 5 rejected late submissions). Each paper was reviewed by two to four program committee members. We had thorough discussions on each submitted paper in the program committee meeting on December 10, 1993. Based on the received comments and the interest of the workshop participants, 11 papers were accepted for presentation along with two keynote speeches and four invited presentations. Most of the accepted papers represent a preliminary report of ongoing research. It is anticipated that some of these reports can be polished and extended and will appear in recognized scientific journals.

The distribution of the submitted and accepted papers by geographic locations are as follows:

Geographic Locations	# of Submitted Paper	# of Accepted Papers
Asia	11	3
Australia	4	3
Europe	8	3
USA	3	2
Total	26	11

We are grateful to Professor Stanley Su of the University of Florida, Professor To-yat Cheung of the City Polytechnic of Hong Kong and the invited speakers for their contributions to this workshop and also to the sponsors who made this workshop financially viable and to the Hong Kong Computer Society for cooperation. Finally, we are also thankful to the program committee and the organizing committee who have worked hard to provide for the success of this workshop.

Francis Chin

Workshop Program Co-chairman

CONTENTS

MSQL: An SQL-based Relational Database Extension to Support Multimedia Data

Sha Guo[1], Wei Sun[1], Wei Li[1], Naphtali Rishe[1], Yi Deng [1], Qing Liu[2], Wei-Ping Zhang [2]

Abstract

In this paper, MSQL, an extension of SQL based on a conventional relational database system (RDBMS) and its hardware and software supporting platform, is proposed to support the modeling, presentation, representation, and manipulation of multimedia data. The focus of the study is placed on video and (possibly synchronized) audio data, because of their popular demand, many unique features and a great deal of resource consumption, although other types of data such as images, pictures, as well as ordinary structured texts are facilitated in an integral and uniform manner. Due to these unique features of multimedia data, a *property-based* multimedia data model is proposed so that multimedia data can be treated as high-level atomic objects in the extended RDBMS and can be manipulated/queried by using their *static/active properties* which capture the static/run-time behaviors of multimedia data. New multimedia data types and advanced multimedia data operations are provided, and real-time presentation and manipulation of video, audio, and image/graphics data are supported. Advantages of the proposed methodology by extending the SQL based on a RDBMS on a low-cost platform, in addition to those advantages possessed by the SQL and RDBMS, are its potentials of being widely and easily accepted. The hardware (an add-on card to a PC) and system software (MS Windows/DOS based) for the underlying mechanism to support high quality real time video and synchronized audio have been successfully completed, and we are currently in the prototyping stage of the MSQL system.

1 Introduction

Multimedia data have become increasingly popular and important in recent years due to advances in various hardware and software technologies such as faster CPU, wider communication bandwidth, bigger storage, audio/video capture/digitizing and playback devices with possible real-time compression and decompression, and standardization. Many multimedia products such as Intel's DVI (digital video interactive) technology [3, 10], among many others, are beginning to emerge in the commercial market. In a multimedia system, information which was previously represented in analog form for video and audio is now available in digitized form. This provides the possibility that all information can be presented, processed, and manipulated by computers. Efficiently and effectively manipulating multimedia data in a multimedia information system is challenging computer scientists, database system designers in particular, for novel ways of representing, modeling, handling, transferring, and presenting multimedia data in a database management system (DBMS).

Multimedia objects are in various forms such as video, audio, graphics, images as well as ordinary structural text. Real-time manipulation of video and (possibly synchronized) audio data [13] poses the most critical problem to a database management system due to their unique requirements such as an extreme large bandwidth on I/O and communication channels, how they are modeled and represented, and how database operations can be formulated and processed. For example, how queries on video and audio data in an extended RDBMS can be specified, processed, and optimized remains unclear. In this paper, we will focus on a study to support real-time video and audio, the most critical and resource-consuming data, in extending a RDBMS and SQL on a popular inexpensive PC platform. The significance of the proposed strategy can be briefly summarized as:

[1]School of Computer Science, Florida International University, Miami, Florida 33199. All correspondences shall be addressed to Dr. Wei Sun, email: weisun@fiu.edu.

[2]Milky Way Computer Corp., Beijing, China

- Video and audio data are supported in a uniform and integrated manner in the proposed and partially-prototyped MSQL system. Directly supporting high-quality real-time video and audio in a conventional relational database on an inexpensive platform has rarely been addressed.
- By extending the SQL on a conventional relational database system, it is expected that the resulting system can be widely and easily accepted, because of SQL's popularity, simplicity, dominance, and standardization. Current SQL does not directly support multimedia applications. The proposed MSQL has the following novel features:
  - a *property-based* multimedia data model;
  - new multimedia data types PICTURE, VIDEO, VOICE and their subtypes;
  - many advanced operations on multimedia data; and
  - new semantics associated with standard SQL statements when multimedia data are involved;
- A successfully-completed hardware and system software implementation of supporting high quality real time video and synchronized audio on a low-end PC platform running MS-Windows or DOS [12] has laid down a solid foundation for the proposed MSQL and for a full-scale multimedia DBMS in the future.
- By having the system on a inexpensive and popular PC platform, the proposed system could be very affordable to the general public.

Since a multimedia system collects, manipulates and presents large volumes of data, a multimedia database management system is an essential component of the multimedia system. This study is a first and important step toward a comprehensive multimedia DBMS and multimedia information system. The proposed MSQL is in its prototyping stage now, while some fundamental supporting mechanism for video and audio in both hardware and software have been successfully implemented in a collaboration with the Milky Way Computer Corp.

The following example illustrates our motivations why a multimedia DBMS and MSQL is useful and important, and will be used through the rest of the paper.

Example 1 *Consider a software product of a Data Flow Diagram (DFD) editor, a component of CASE tools. In order to help a user effectively learn how to use the editor, a multimedia instruction system containing video, audio, screen-dumps etc. is intended to be provided in addition to the traditional documents and user manuals. In organizing such a multimedia instructional system using a RDBMS, certain problems may be encountered. For example, there may not exist the corresponding multimedia data types to help represent multimedia data. And even these multimedia data can be implicitly stored in the RDBMS, say by using bitstreams, queries can hardly posed against them. Consider the following table of* VIDEODEMO *and its populated instance of two video segments:*

```
CREATE   TABLE  VIDEODEMO (
         Title          CHARACTER (30) NOT NULL,
         Productno      CHARACTER (10),
         Videoclip      BIT VARYING);
```

Title	*Productno*	*Videoclip*
'MenuFunction'	'DFD06'	**BITSTREAM**
'Overview'	'DFD06'	**BITSTREAM**

Given a simple query for all video:

```
SELECT    Title, Videoclip
  FROM    VIDEODEMO;
```

in addition to the problem how to represent the video data, presenting the video and its synchronized audio in real-time as the result of the answer to the query is not standard in a RDBMS: even if all video segments can be simultaneously presented in real-time on screen (in a table) like ordinary structural texts, it is not clear what is the right way to do this, because a simultaneous presentation of multimedia data such as their accompanying audio sometimes may not make sense at all. ■

The rest of the paper is organized as follows: In *Section 2*, a multimedia data model and new multimedia data types are introduced which explicitly support the representation of multimedia data in a DBMS. In *Section 3*, the semantics of typical MSQL statements are presented and how they can be supported are briefly discussed. *Section 4* introduces advanced expressions involving multimedia objects such as overlapping and concatenation operators, type conversions, etc. In *Section 5*, the fundamental supporting mechanism of multimedia data (for both hardware and system software) that have been developed on a low-end PC platform is described. Finally, the paper is concluded in *Section 6*.

2 Multimedia Data Model and Data Types

Multimedia data involve images/graphics, audio and video data in addition to ordinary structural data.

An image consists of an array of pixels with a color or grey-scale depth for each pixel. For example, a 8-bit color depth represents at most 256 colors. The more the color bits are used, the bigger the image file. There are many file formats for storing images/graphics, such as Postscript file format [1], Portable Bit Map (PBM) format, Graphics Interchange Format (GIF) [7], JPEG File Interchange Format (JFIF) [5], etc.

A segment of video consists of (continuous) video frames and (optional and possibly synchronized) audio. The amount of a video clip is determined by the resolution (the width and height of the video frame, and color-depth), number of frames, audio, and the compression ratio used. The visual effect of a video clip during its playback is determined by its resolution and the playback rate (frames per second, or fps). NTSC TV video (a TV standard used in North America) is refreshed at 30 fps and we call video clips at this rate or higher the real-time video (otherwise animated video where jerky movements may be observed by human eyes). The amount of storage for real-time digital video are enormous. Therefore, real time video must be compressed because of the constraints placed by current hardware such as communication bandwidth, computing speed, and storage capacity. Motion-JPEG [5] and MPEG [8] are two popular standards for video compression. JPEG (Joint Photographic Expert Group) [17] refers to the international digital image compression standard for continuous-tone still images of both grey-scale and color. Briefly speaking, using Motion-JPEG, the compression is made by representing each frame of a video clip (an image) in the JPEG format, and video data can be compressed by up to 1:40 without visibly affecting video quality for playback. MPEG is the international digital video compression standard for video signal and its associated audio. Two variants of MPEG, MPEG-I and MPEG-II, are practically used. MPEG usually delivers a better quality of video playback for the same compression ratio of the Motion-JPEG, although it needs much more computation power in compression. Audio requires much less space and computation than video. Thus, audio demands significantly less system resources.

It is possible that a multimedia object can be stored as a bit string. However, it seems that it is beneficial to have distinct multimedia data types, since strong-typed data in a DBMS bring up the flexibility and capability of efficient and effective representation and manipulation.

In MSQL, a multimedia object such as a segment of video or audio is treated as a high-level atomic object such that they can be queried/manipulated by SQL-like statements in MSQL. Clearly a multimedia object is not "atomic" by its nature. For example, information about the resolution is needed for a video object. In the proposed system, a multimedia object is an encapsulated complex object consisting of the "raw" data and *meta-data*. Meta-data is further divided into the static properties and active properties.

Static properties of a multimedia object: contain *physical properties* and *signatures* of the multimedia object:

Physical properties: physical properties are used to describe the physical information of a multimedia object. For example, for video data, its resolution, compression ratio, compression standard, etc., are typical physical features. Each multimedia data type has its own fixed pattern of physical properties.

Signatures: Signatures are in fact physical properties except that they contain abstractions of the multimedia object. An abstraction is likely obtained manually or semi-automatically, and is probably subjective. Because manual or semi-automatic abstraction may not be uniformly performed, there is no fixed pattern for the signatures. This mechanism thus facilitates easy expansion and customization. In our design, we use signature to represent certain interpretation of the multimedia object in order to support some operations on them. For example, in *Example 1*, the keywords {"Menu", "Functions", "Help", "DFD"} and {"Overview", "DFD", "High-Level"} are associated with the first and the second video clips as their signatures, respectively. Then a query like "Give me the video clips describing all menu choices and their functions" may be formulated and the first video clip will be retrieved as the result (a potential contend-based retrieval [4, 6, 14, 18]). We only provide a framework here, of course, more comprehensive and detailed studies need to be carried out.

Active properties of a multimedia object: Active properties capture the dynamic behaviors of a multimedia object, particularly how the multimedia object is presented. In our design, a *presentation shell*, an active property, is associated with each multimedia object which determines how the multimedia object will be presented. For example, for a video object, the shell will determine if to present it as an icon (in fact not physically presenting the video, and thus make it possible for "simultaneous" presentation) or fully present it with a VCR-like control panel that can be interactively controlled by a user, as shown in *Figure 1*. Different multimedia type may have different shell.

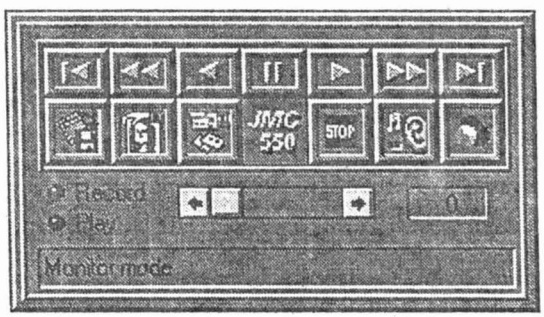

Figure 1: *Control Panel for Video Playback*

We note that all these properties are *encapsulated* in a "atomic" multimedia object. These concepts are basically borrowed and possibly "cloned" from an object programming paradigm. we elect to support the following atomic multimedia data types in MSQL:

PICTURE The PICTURE data type represents the bitmap of an image or two-dimensional shapes of graphics. Under the PICTURE type, there are different subtypes, which correspond to different formats of image/graphics. The following is an incomplete list of the picture subtypes to be supported.

- PS: the Postscript file format;
- GIF: the Graphics Interchange file format;
- PBM: the Portable Bitmap file format;
- JFIF: the JPEG File Interchange Format;

The *static properties* of an image could include a pointer to the file that stores the picture, its format type, the image size (width x height), the color depth used, a signature which is a set of polygons representing the skeleton of the image, etc. The *active properties* of the image could include presentation shell of a picture.

The PICTURE is a generic data type which can be used to represent any format of images/graphics. For an attribute of a subtype of PICTURE known by MSQL, its *active properties* could be pre-defined and triggered. For example, if the picture is in PS format, the presentation shell can automatically recognize it is in PS by examining the first a few lines of the "raw picture data", and thus properly present the picture.

VIDEO The VIDEO data type represents video frames with (optional) audio sequences. The following is a partial list of VIDEO's subtypes supported in MSQL.

- MJPEG: video clips of the Motion-JPEG standard format with audio sequences;
- MPEG-I: video clips of the MPEG-I standard format with audio sequences;
- MPEG-II: video clips of the MPEG-II standard format with audio sequences;
- AVI: represents video clips of the Microsoft AVI format;

The *static properties* of a video attribute could include file pointers which indicate the three files that store the video frames, the synchronized audio, and the synchronization information, respectively, the video format, the frame size (width x height), the color depth, the number of frames, a signature which consists of a set of keywords for the video clip, etc. The *presentation shell* of a video clip is a VCR-like control panel that supports interactive manipulation by a user as shown in *Figure 1*.

Similar to the picture type, the VIDEO is a generic data type which can be used to represent any format of video clips.

VOICE This data type is used to represent audio sequences. The property structure of a VOICE attribute may include the audio lengths in bytes and seconds, respectively, a pointer pointing to the content file, a volume and speed control presentation shell, etc.

3 SQL Statements on Multimedia Objects

In this section, we extend SQL to facilitate multimedia data. MSQL, like SQL, achieves the dual requirements of DDL (Dade Definition Language) and DML (Data Manipulation Language). Due to space limit, we will only elaborate on important and frequently used statements.

3.1 Data Description in MSQL

Like SQL, MSQL tables are created through the CREATE TABLE statement, and all MSQL data manipulation statements must operate on tables that have been created. Additional new types have been introduced in MSQL. In our sample database application, two tables that deal with screen-dumps (images/graphics) and video demos (video clips) are created.

```
CREATE   TABLE   SCREENDUMP (
         Title          CHARACTER (30) NOT NULL,
         Productno      CHARACTER (10),
         Image          PICTURE );
```

```
CREATE   TABLE   VIDEODEMO (
                 Title          CHARACTER (30) NOT NULL,
                 Productno      CHARACTER (10),
                 Videoclip      MJPEG );
```

The following is the sample data of the table *SCREENDUMP*. The three images describe the three basic components of a DFD, *flow, source*, and *process*. The sample data of the table *VIDEODEMO* is in *Example 1* with each **BITSTREAM** replaced by a **MJPEG PROPERTY**.

Title	*Productno*	*Image*
'Flow'	'DFD06'	**PICTURE PROPERTY**
'Source'	'DFD06'	**PICTURE PROPERTY**
'Process'	'DFD06'	**PICTURE PROPERTY**

3.2 Basic Data Manipulation in MSQL

In MSQL, operations on multimedia data are by their properties. In this section, we show the basic manipulations on multimedia data by examining the typical SELECT-FROM-WHERE statements.

An attribute in a SELECT-clause will cause the data under that attribute to be presented in the answer table. However, the presentation of multimedia data can be *exclusive* or *non-exclusive*. Audio data must be presented in an exclusive manner, namely, it does not make sense that multiple audio tracks are played simultaneously (unless multiple audio tracks are to be synthesized, which will be support in MSQL as a special operation in a multimedia expression to be discussed in *Section 4*, but this is not the general semantics of "displaying" the (audio) data). Video and synchronized audio are also exclusive, but video without synchronized audio or with its synchronized audio suppressed may be exclusive or non-exclusive. Hardware may also pose constraints on video to be presented in only exclusive manner (for example, only one video play window of the specified size can be supported by the hardware).

Consider that an attribute of the VIDEO type is in the SELECT-clause of an MSQL query. One way of presenting the video objects in the resulting table is to display one tuple each time, and the video value in the tuple is played in normal size in a display window. When the next tuple is specified, the video in the display window is replaced by that in the newly-selected tuple.

Example 2 *Consider the query*

```
SELECT    Title, Videoclip→NORMAL
  FROM    VIDEODEMO;
```

The presentation of the video clips in the resulting table is exclusive, which is specified by the presentation property NORMAL of video attributes. A VCR-like panel is also provided to allow users to control the playback of the current video, see *Figure 1*. ■

In statements of MSQL, a property of a multimedia data is represented as "DATA → PROPERTY', as the example of "Videoclip → NORMAL' shown in the above query.

Another method to present VIDEO values is to display each video as a small icon so that multiple video (icons) can be presented at once. Icons can be triggered by users to be played in full size with the presentation shell in a random manner. The presentation shell may also intelligently determines the best way to present video data (for example, when only one qualified video object is retrieved, the full motion video play as the retrieval result will be used).

Example 3 *Consider the query*

```
SELECT    Title, Videoclip→ICONS
  FROM    VIDEODEMO;
```

TITLE	VIDEOCLIP
'MenuFunction'	(icon) MJPEG VIDEO
'Overview'	(icon) MJPEG VIDEO

and the resulting table

The values of the column *Videoclip* appear as small icons in the corresponding fields of the resulting table which act as monitors (slow playback in icon size). A click on an icon will trigger its full size real-time playback. ■

4 Advanced Multimedia Expressions and Operations

In this section, we first show how to manipulate multimedia data by their static properties in multimedia expressions of MSQL. Then, the CAST expression of data format conversion is discussed. At last, we will provide two advanced operations for multimedia data, overlapping and concatenation.

4.1 Querying Multimedia Data by Their Static Properties

Static properties of multimedia data can be queried in MSQL. Again, due to space limit, we only use simplified self-explainable examples to illustrate the idea.

Example 4 *The following query is for the size of the images in the column* Image *of* SCREENDUMP*:*

```
SELECT   Title, Image→SIZE
  FROM   SCREENDUMP;
```

The resulting table is:

Title	*Image* SIZE
'Flow'	640X480
'Source'	320X240
'Process'	640X480

The second example computes the total length in seconds of all the video clips stored in the column *Videoclip* of *VIDEODEMO* as in the following statement:

```
SELECT   SUM(Videoclip→SECONDS)
  FROM   VIDEODEMO;
```

The statement results in 653 seconds in total. ■

4.2 CAST Expression

In SQL, CAST expression is provided to explicitly convert a numeric value to any other numeric data type, or any character string type, if it makes sense. This requirement of data conversions also exists for multimedia data types. For example, it is often required for a GIF image to be converted to a JFIF image to reduce space or for compatibility reason, or a MJPEG video clip converted to AVI format so that the video can be played in a system which supports only AVI format video data.

The syntax for CAST is as:

CAST (*value-expression* AS *data-type* SPEC *seconds*)

The *value-expression* computes a value to be converted, *data-type* specifies the data type of the conversion result. The SPEC is used to specify the length in seconds of the video clip converted from an image.

Example 5 *The following statement is a query of the video clips in* VIDEODEMO *which are returned as AVI format video data.*

```
SELECT   Title, CAST(Videoclip → ALL AS AVI)→ICONS
  FROM   VIDEODEMO
```

ALL indicates all of the properties of the MJPEG video values. In the resulting table, the MJPEG properties are converted to AVI properties. ■

The following is the rules of CAST expression in MSQL:

- Values of the PICTURE types, namely, PICTURE and all its subtypes, can be converted to any other PICTURE types;
- Values of the VIDEO types, namely, VIDEO and all its subtypes, can be converted to any other VIDEO types;
- ASCII text strings can be converted to audio sequences of the VOICE type.

4.3 Overlapping and Concatenation

In this subsection we will discuss the overlapping ("&") and concatenation ("+") operations which are applied to multimedia objects. Both operations are binary operations. However, they are overloaded or polymorphic operators. These two operators are dealing with multimedia data on the time dimension – a unique feature of multimedia data.

Concatenation ("source1 + source2"): where source1 and source2 are two multimedia data objects.

The concatenating different types of multimedia data has different semantics, since "+" is an overloaded operator, as explained in the follows.

- "voice1 + voice2": voice1 and voice2 are audio data of the VOICE type. This operation results in a new audio with *voice2* concatenated to (the end of) *voice1*. The length of the resulting audio data is the sum of that of voice1 and voice2.
- "video1 + video2": video1 and video2 are two video clips of the VIDEO types. This operation will append *video2* to *video1*. The number of frames of the result is the sum of that of *video1* and *video2*. Audio data associated in video clips are also concatenated in the same process of "voice1 + voice2". Some parameter may be set to allow the concatenation of video with different resolutions.
- "picture1 + picture2": picture1 and picture2 are images of the PICTURE types. The result of the operation is a new image of the same type with that of picture1. There are many ways to put two images into a single one. For example, the new image is formed by putting two source images side by side, see *Figure 2*. Suppose that the sizes of picture1 and picture2 are width1 x height1 and width2 x height2, respectively, then the new image is of the size (width1 + width2) x min(height1, height2). Other placements are also supported (via various parameter settings), and we do not exhaustive list them here.

Overlapping Operation:

Overlapping allows a larger degree of polymorphism than concatenation:

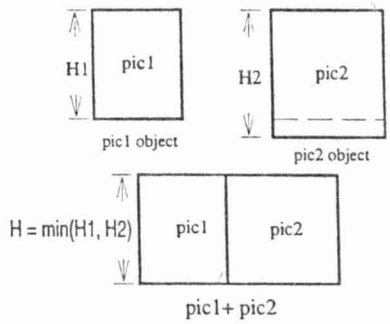

Figure 2: *Concatenation Operation: "picture1 + picture2"*

- "voice1 & voice2": voice1 and voice2 are mixed together as a single audio sequence, where voice1 and voice2 are two audio data of the VOICE type. An example is to synthesize both channels of a stereo sound into one channel.
- "video1 & video2": *video1* and *video2* are superimposed to form a single video clip. More precisely, each of the resulting video frames consists of both frames from video1 and video2 at the same time. Both accompanying audio sequences are also overlapped into a single sequence, as in the process of "voice1 & voice2".
- "image & voice": This operation will transform an image and an audio segment into a video clip (each frame consists of the image) and the synchronized audio. The number of frames is decided by the playback length of the given audio data. For example, if the playback of the given audio will last 10 seconds in normal speed, then the number of frames is 300 if 30 fps playback is used.
- "video & voice": A video clip is synchronized with the given audio data. If the given video clip has already had a synchronized audio, then the original audio is replaced by the given audio data.
- "voice & text": The text is converted to an audio sequence and mixed with the voice as the result. This basically adapts the sound synthesizing devices.

5 Supporting Real-Time Manipulation

In this section, we briefly describe the functional interface to support the real-time manipulation of multimedia objects. The hardware interface is the JMC550 image/audio/video real-time capture/playback devices (with compression and decompression) we developed on an IBM or compatible PC 386 or above platform running MS Window or DOS, and the software interface consists of Window and DOS based interactive utility programs and function libraries (dynamic linking library for Window, and static library for DOS).

As we know that video and audio are the two essential components of a multimedia computer system. Compared to audio data, video is significantly more difficult to handle than audio due to its high bandwidth requirement and complicated compression/decompression technique, particularly in a real time setting. For example, for NTSC video (30 frames per second with a frame resolution of 768x576 pixels by the true color 24 bits), a sustainable transmission rate of 31 Mbits/s is needed. Clearly, the direct use of digital video of TV-quality without a compression is not feasible on most of current computers. We choose to use JPEG standard, the first international digital image compression standard jointly adopted by CCITT and ISO. Currently, JPEG has been widely used in various applications. Its primary goal is to support both still and motion pictures.

5.1 The Hardware Design of the JMC550 Board

The high-level block diagram is illustrated in *Figure 3*.

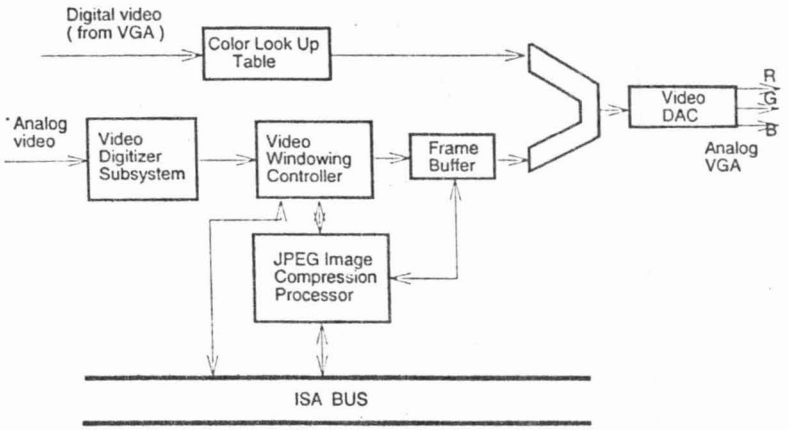

Figure 3: *Block Diagram of the Board*

The board contains a JPEG compression processor (C-Cube CL-550 chip), and a visual windowing controller (Chips and Technologies' PC Video chip). The compression processor is capable of compressing/decompressing the real-time video signals after the analog video signals are digitized (30 fps for NTSC and 25 fps for PAL). During a compression and decompression, the visual windowing controller takes care of window resizing, repositioning, clipping and buffering, creation and deletion of windows, and color indexing. When the analog video signal is supplied to the digitizer from a video source such as a video camera or VCR/laserdisk player, it is converted into a 8-bit YUV 4:2:2 signal, followed by a transformation into a 16 bit RGB signal by the luminance space transformation. The windowing controller then scales, clips and positions the RGB signal and stores them in the frame buffer, where it is mixed with the digital signal from the VGA adaptor (which is converted into 16 bit RGB signal via a color look up table [clut], and which is obtained from the feature connector of a VGA adaptor card). Finally, the mixed up signal is sent out to the monitor after the digital/analog conversion to be displayed on a SVGA/VGA monitor. During the compression and digitizing (store the video in a file), the 16 bit RGB video signal is sent to the ISA/EISA bus via the 16-bit video DMA. The decompression just reverses the above procedure.

Analog audio signal is separately converted into 8 bit digital signal by a 22KHz sampling, and stored in the disk via the audio DMA. During playback, the digital signal is converted back into analog signal and outputed to a speaker.

Audio and video files are separate, and there is a third file for their synchronization. Marks are placed in audio file and relevant information is kept in the synchronization file to indicate at which frame the audio should be played. After a video frame has been decompressed, audio will be compared with the video frame. If the audio lags behind the video, the frame will hold for one more frame. If the video lags behind the audio, the one frame will be skipped. The separation of audio and video makes it possible to support multilingual feature by just attaching different audio file to the same video file. Many other systems do not support the synchronization of video and audio, and mix both video and audio in the same digitized file. We have successfully written a few multilingual multimedia applications.

In summary, the JMC550 is a real time video compression (for digital capturing and recording) and decompression (for playback) card. It also supports synchronized audio. It accepts PAL or NTSC TV signals and in real time digitizes and compresses the video into a hard disk file at a rate of 30 fps for NTSC or 25 fps for PAL at a frame resolution of 320x240

and 16 color bits. It can also perform a real time playback in computer VGA/Super-VGA display with scalable size window up to the full screen. The video compression ratio varies from 12 to 400 times. The sustainable transfer rate is typically 200-400 KB/sec which makes a low-end personal computer with normal ISA-bus and IDE hard drive possible to serve as a real time high quality video/audio workstation.

5.2 The Function Libraries

JMC550 software has two different versions: MS-DOS (version 5 or 6) and MS-Windows 3.x. Each version of the software has two parts: the interactive utility program for interactive manipulation of video and audio and control of the card, and function library. The function library is static under DOS, but dynamic under Window (DLL), which is used to control the operation of JMC550. In *Figure 4*, the structure of a typical MS-Windows applications is shown.

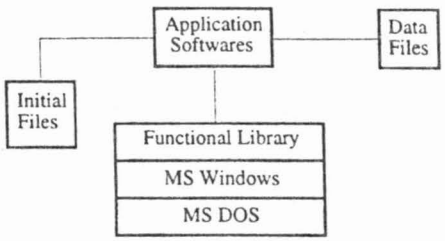

Figure 4: *The Structure of MS-Windows Applications*

The function library, consisting of a set of C subroutines (Microsoft C/C++ or Borland C/C++) provides the following functions:

1. recording, digitizing, and compressing video;
2. monitoring the video (a window becomes a monitor);
3. decompressing and playing back the video in a window or up to a full VGA display. VCR-like control functions are provided such as recording, pausing, playing, fast forward playing, backward playing, fast backward playing, rewinding, and playing starting at any frame;
4. freezing the video and saving/loading a frame in different formats such as TIFF, Targa, PCX, and DIB.
5. setting system parameters such as the input video mode (NTSC vs PAL), color (contrasts, brightness, and hue), audio on/off, etc.

The library is provided as the static linking library (for MS DOS) and dynamic linking library (for MS Window). A prototype of the proposed MSQL is under development by using the dynamic linking library using two popular PC RDBMSs Paradox for Window and Microsoft Access.

6 Conclusions

In this paper, SQL and a RDBMS is extended to support multimedia data. Many problems have been identified and novel features have been introduced. We have successfully completed the implementation of the underlying supporting hardware and software systems on a low-end PC, and the proposed MSQL is currently under prototyping. Although our study may still be rudimentary and incomplete at this stage, but we believe the proposed system is a first and important step of the practical development of a multimedia database management system and multimedia information system.

References

[1] Adobe Sys. Inc., *Encapsulated Postscript Files Specification Version 2.0*, 1989.

[2] Blattner, M. and Dannenberg, R. *Multimedia Interface Design*, ACM Press, 1992.

[3] Bunzel, M. and Morris, S. *Multimedia Applications Development Using DVI Technology*, Addison-Wesley, Reading Mass, 1992.

[4] Cakmakov, D. and Davcev, D. "Experiments in Retrieval of Mineral Information", *Proc. First ACM Int'l Conf. on Multimedia*, Anaheim, CA, August 1993, pp.57-64.

[5] C-Cube Microsystems, *JPEG File Interchange Format Version 1.02*, 1992.

[6] Chu, W. W., *et al.* "A Temporal Evolutionary Object-Oriented Data Model and Its Query Language For Medical Image Management", *Proc. 18th Int'l Conf. on Very Large Data Bases*, Vancouver, August 1992, pp.53-64.

[7] CompuServe Inc., *Graphics Interchange Format Version 89a*, 1990.

[8] Gall, D. "MPEG: A Video Compression Standard for Multimedia Applications", *Comm. of ACM*, Vol. 34, No. 4, April 1991, pp.46-58.

[9] Guo, M., Su, S., and Lam, H. "An Association Algebra for Processing Object-Oriented Databases", *Proc. 7th Int'l Conf. on Data Eng.*, Kobe, Japan, April 1991, pp.23-32.

[10] Harney, K., *et al.* "The i750 Video Processor: A Total Multimedia Solution", *Comm. of ACM*, Vol. 34, No. 4, April 1991, pp.64-78.

[11] Ishikawa, Y., Kitagawa, H., and Ohbo, N. "Evaluation of Signature Files as Set Access Facilities in OODBs", *Proc. 1993 ACM SIGMOD Int'l Conf. on Management of Data*, Washington, DC, May 1993, pp.247-256.

[12] Milky Way Computer Corp., *JMC-550 Image/Video/Audio Real-Time Compression Board User Manual*, 1993.

[13] Prabhakaran, B. and Raghavan, S.V. "Synchronization Models for Multimedia Presentation with User Participation", *Proc. First ACM Int'l Conf. on Multimedia*, Anaheim, CA, August 1993, pp.157-166.

[14] Rabitti, F. and Savino, P. "An Information Retrieval Approach for Image Database", *Proc. 18th Int'l Conf. on Very Large Data Bases*, Vancouver, August 1992, pp.574-584.

[15] Su, S. "Modeling Integrated Manufacturing Data with SAM*", IEEE Computer, January, 1986, pp.34-49.

[16] Teodosio, L. and Bender, W. "Salient Video Stills: Content and Context Preserved", *Proc. First ACM Int'l Conf. on Multimedia*, Anaheim, CA, August 1993, pp.39-46.

[17] Wallace, G. "The JPEG Still Picture Compression Standard", *Comm. of ACM*, Vol. 34, No. 4, April 1991, pp.30-44.

[18] Wu, J.K., *et al.* "Facial Image Retrieval, Identification, and Inference System", *Proc. First ACM Int'l Conf. on Multimedia*, Anaheim, CA, August 1993, pp.47-55.