www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 2 Issue 11 November, 2013 Page No. 3278-3281

Applications of Video Data Base Management System (Vdbms)

For Video Query Processing

[1]Dr. D.S.Kushwaha & [2]Ayushi Kushwaha

[1]Institute of Engineering & Technology, Lucknow-226021(India)
drkushwaha@rediffmail.com
[2]Dr Ram Manohar Lohia National Law University Lucknow-226012 (India)

ABSTRACT

With the increase use of video data sets for multimedia-based applications has created a demand for database technology that is capable of storing digital video in the form of video databases and providing efficient methods for handling the content based query and retrieval of video data. Traditional methods for handling data storage, query processing and retrieval cannot be extended to provide this functionality for video. In this paper, we have described the VDBMS which is a research platform for video database, a system that supports comprehensive and efficient database management for digital video. Video query processing presents significant research challenges, mainly associated with the size, complexity and unstructured nature of video data. This paper also presents the video query processing functionality of VDBMS using several query processing algorithms. It also focuses on how VDBMS supports query processing for content-based query, search and retrieval of video data. In this paper, we have addressed query processing issues in two contexts, first as applied to the video data type and second as applied to the stream data type. Our paper also describes two query operators for the video data type which implement the rank-join and stop-after algorithms. Lastly this paper also presents the stream query processing framework of VDBMS.

KEYWORDS: ***Video database management system, query processing, content based retrieval, rank-join algorithm, stop- after algorithm, stream query processing.***

1. INTRODUCTION

In today's age, more information is being generated than at any other time in human history. Information has become somewhat of a commodity and an asset, critical to the success of all types of organizations, from large corporate entities to the US armed forces. Information is not only plain text but also media rich sources containing audio and visual stimuli. Digital video is a medium with extremely high resolution and very rich information content. Along with metadata such as title, data, etc., video provides other detailed information including object motion, time elapse event occurrences. As a consequence of the large amount of video generated today, an efficient and effective means of managing and retrieving the data must be developed. In an effort to provide efficient and effective storage and management of digital video data, much research has been accomplished towards developing a VDBMS. A VDBMS is a software system that manages a collection of video data and provides content-based access to users. As in other DBMS, the goal of a VDBMS is

to provide an environment both convenient and efficient for retrieving and storing video information in a database.

The VDBMS, video database management system was designed to support a full range of functionality for video as a well-defined abstract database data type, with the goal of providing video-based applications with all the powerful functionality generally provided by database management systems [1, 2]. VDBMS processes video for content representation and uses the indexed content to support video query, search and retrieval [8].The most useful environments for advancing research and development in video databases are those that provide complete video database management, including(1)video pre-processing for content representation and indexing,(2)storage management for video, meta data and indices,(3)image and semantic-based query processing,(4)real-time buffer management, and(5)continuous media streaming. VDBMS system components include a video pre-processing toolkit, a query processor, a stream manager, and a search-based buffer management system [1,2,8,6,13].

This paper described selected VDBMS query interface in section 2. This paper addressed VDBMS video query processing in two contexts, first as applied to the video data type (VDT) and then as applied to the stream data type (SDT). Section 3 describe VDBMS query processing for the VDT and present VDBMS query operators that implement the rank-join and stop-after algorithms operating on video as a VDT. Section 4 describes the stream query processing framework of VDBMS.

2. THE QUERY INTERFACE

A video object consists of features such as frame resolution, number of bits per pixel, compression information and duration and color models. A frame is a data object. In traditional databases, a frame can be compared to a row/tuple. A data object can qualify as a frame when it has basic features like colour, texture and shape defined by the MPEG-7 standard for multimedia data. A shot is a contiguously recorded series of frames that represent continuous action in space and time. Shots that are related in time and space can be combined to form an episode. Figure 1 provides a graphical depiction of the hierarchical abstraction of video data.

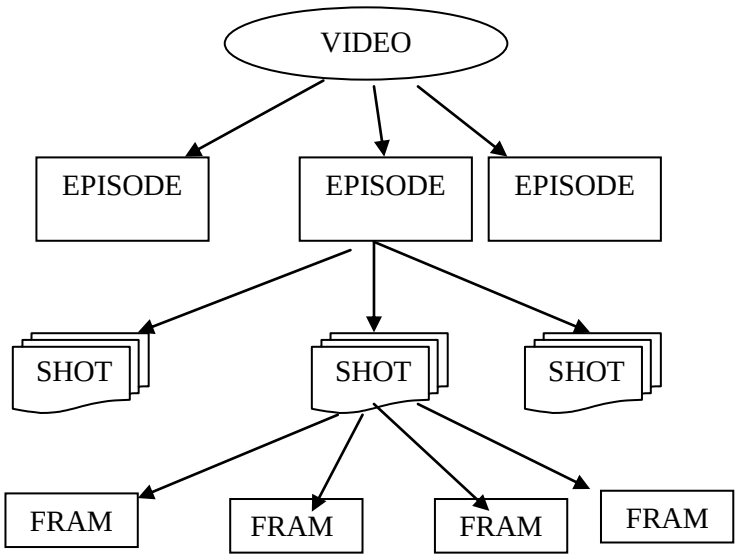

Figure 1. Hierarchical Abstraction of Video Data

As stated, a video is a sequence of frames and the video frame sequence can be divided into shots using image processing techniques or by a domain expert. VDBMS processes video for content representation and uses the indexed content to support video query, search and retrieval. VDBMS image processing for content representation and indexing supports (1) video

shot detection, (2) feature extraction per frame and per shot, (3) key frame extraction, with one key frame representing each detected shot [24]. The video, with its image and semantic-based content representation, are stored in VDMBS to support query and retrieval though the query interface.

A VDBMS query interface client supports content- based query, search, and retrieval and real-time streaming for the VDBMS video database server. Users access the VDBMS query interface using the Windows-based client shown in Figure 2.The client connects to the VDBMS system which resides on a Sun Enterprise 450 machine with 4UltraSparc II processors located at Purdue University. VDBMS functionality has been tested against more than 500 hours of medical videos obtained from the Indiana University School of Medicine. The medical videos are digitized, compressed into MPEG1 format, processed off-line by the VDBMS pre-processing toolkit to generate image and content-based meta-data, and then stored together with their meta-data in the VDBMS database.

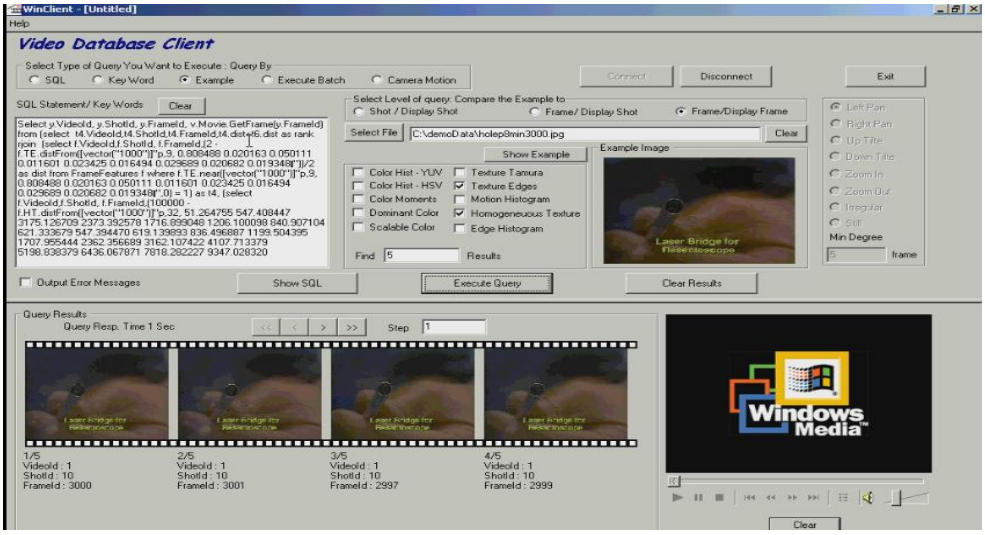

Figure 2. A VDBMS query interface

2.1 Video Query Language

Video query language is similar to SQL in structure. The language can currently be used for spatio-temporal queries that contain any combination of directional, topological, 3D-relation, object- appearance, trajectory-projection and similarity-based object-trajectory conditions. As a work in progress, the language is being extended so that it could support semantic video queries as well in a unified and integrated manner. The language has four basic statements for retrieving information:

SELECT video FROM all [where condition];
SELECT video FROM videolist WHERE condition;
SELECT segment FROM range WHERE condition;
SELECT variable FROM range WHERE condition;

Target of a query is specified in select clause. A query may return videos (video) or segments of videos (segment), or values of variables (variable) with/without segments of videos where the values are obtained. Aggregate functions, which operate on segments, may also be used in select clause. Variables might be used for object identifiers and trajectories. Moreover, if the target of a query is videos (video), users may also specify the maximum number of videos to be returned as a result of a query. If the keyword random is used, video fact-files [12] to process are selected randomly in the system, there by returning a random set of videos as a result. In from clause range of query is specified, which may be either the entire video collection or a list of specific videos. Query conditions are given in

where clause. In our query language, condition is recursively defined, and consequently, it may contain any combination of spatio-temporal conditions. For example, users can specify the query to play the first shot of the fourth scene of the third episode of the current video object:

SELECT S4.play ()

FROM current Video V, V.Episode E1, E1. Scene S3, S3.Shot S4

WHERE E1.Epnum=3

AND S3.ScNum=4

AND S4.ShNum=1;

3. VDBMS QUERY PROCESSING FOR THE VIDEO DATA TYPE

3.1 Rank Join Operator

In multi-feature image similarity queries, users generally present a sample image and query the database for images "most similar" to the example based on some collection of visual features. Results must be determined according to a combined similarity order [4,5]. NRA-RJ is a new binary pipelined rank query operator that determines an output global ranking from the input ranked video streams based on a score function. It advances Fagin's optimal aggregate ranking algorithm[3] by assuming no random access is available on the input streams. The output of NRA-RJ thus serves as valid input to other operators in the query pipeline, supporting a hierarchy of join operations and integrating easily into the query processing engine of any database system.

A new VDBMS query operator that encapsulates the rank-join algorithm is its GetNext operation; each call to GetNext returns the next top element from the ranked inputs. The GetNext operator is the core of the rank-join operator. The implementation of GetNext () support:

- Source input steam arriving as output of another NRA-RJ operator with range of possible scores (no random access, exact scores unknown).
- Number of requested object not known in advance, rather it increases for each call to GetNext.

Therefore GetNext operation can easily integrated into query pipelines to support hierarchy of join operations, nested views and wider range of query execution plans.The major disadvantage of this operator is computational overhead as the number of pipeline stages increased.

3.2 The Rank-Join Algorithm

Consider stored video metadata that describes low-level visual features such as color histogram, texture and edge orientation. The features are extracted for each video frame during pre-processing and stored in separate tables in the database. Each feature is then indexed using a high-dimensional index for faster query response. If a user is interested in the k video frames most similar to a given query image based on color, the database system should rank the frames according to their similarity to the color information extracted from the given image, and present only the k most similar frames to the user. The database system can use the high- dimensional index to perform an efficient nearest-neighbor search [4] and produce the nearest k neighbors. We call this simple ranking query a single-feature or a single-criteria ranking query, and no joins are required to answer the query. A database system supporting approximate matching merely ranks the tuples according to how nearly they match the query image.

A more complex similarity query occurs when a user is interested in finding the k most similar frames to a given query image based on both color and texture. In this case, the database system must obtain a global ranking of frames based on both color and texture similarities to the query image. We refer to this type of query as a multi-feature ranking query. Unlike for single feature ranking queries, it is not clear with multi-criteria ranking how the database system should combine the individual rankings of the individual criteria, even if the notion of approximate matching is supported [3]. In current database systems, the only way to evaluate the

query in the previous example is as follows: First, the feature tables are joined on the tuple key attributes. Then, for each join result, the similarity between the tuple features and the query features are quantized and combined into one similarity score. Finally, the results are sorted on the computed combined score to produce the top-k results. Two expensive major operations are involved: joining the individual inputs and sorting the join results. When using traditional join operators to answer a ranking query, an execution plan with a blocking sorting operator on top of the join is unavoidable. If the inputs are large, the cost of this plan can be prohibitively expensive.

Figure 4 shows how rank join operators are used to retrieve the two video shots that are most similar to an input image based on colors and texture, and that contain the keyword "kidney" in the their annotation for the following multimedia database table:

- Video Table (VideoId, VideoName, Annotations).
- Feature Table (VideoId, Color, texture, Edge orientation).

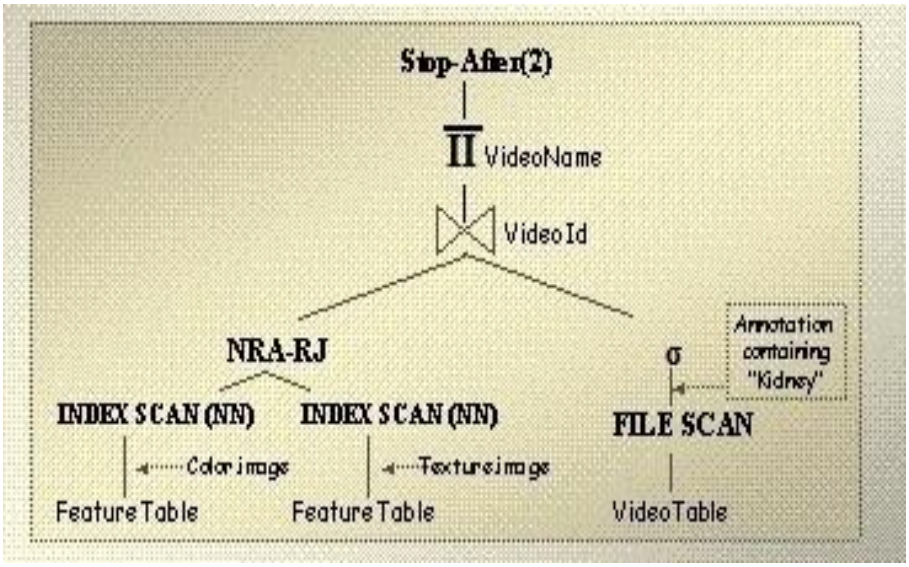

Figure-4, NRA-RJ: Complex Multi-features Queries

3.3 The Stop-After Algorithm

As NRA-RJ is a pipelined operator, it does not specify the number k of desired outputs. Therefore the Stop-After query operator was used to limit the output of the similarity queries. The number of reported answers to k in NRA-RJ is limited by applying the Stop-After query operator [7,6], which is implemented in VDBMS as a physical query operator Scan-Stop. The Stop-After appears on top of the query plan. The Scan-Stop does not perform any ordering on its input.

4. VDBMS QUERY PROCESSING FOR THE STREAM DATA TYPE

Since videos can be considered as a long sequence of frames delivered over time, one can model video as a stream of frames. With this view of video data, a multitude of fine-grained and incremental video operations can be introduced. Whereas the offline and bulk processing of video is widely deployed to process stored videos, the incremental and frame-level processing of video would be advantageous in scenarios such as the following:

(a) Video is delivered on-line as an infinite stream, and the responsiveness of video processing is important, for example, while tracking moving objects in surveillance applications.

(b) Storage space is limited and it is not feasible to keep multiple copies of the same video (the original video and processed versions). In this case the processing of video upon request and streaming the resulting, or processed, video is considered a space-

efficient approach.

The VDBMS design uses MPEG-7 standard which can handle content with various modalities such as the image, video, audio, speech, graphics, and their combinations. The Motion Picture Expert Group, or MPEG, meets under the International Standards Organization (ISO) to create and maintain standards for digital video and audio compression. MPEG video compression is a block-based encoding scheme. Specifically, the standard defines a compressed bit stream, which implicitly defines a decompressor. The video stream consists of a header, a series of frames, and an end-of-sequence code. Because the stream is temporally compressed (most frames build upon previous frames), there are periodic Intra-Pictures, or I frames. I frames provide full images to be used as periodic references, and allow random access to the video stream. Other frames are predicted, using either a preceding I frame (creating a P frame) or a combination of preceding and following I frames (creating a B frame). The order and interspersion of the I, B, and P frames are determined by the encoder. A typical sequence would be I-B-B-P. However, the order of pictures in the data stream is not the order of display. For example, the previous sequence would be sent as I-P-B-B.

This section describes the stream query processing framework of VDBMS. This is followed by stream query operations and their applications on video streams. Finally, a brief description of the interface between query operations and the underlying streams is explained.

4.1 Stream Query Processing Framework of VDBMS

Consider a stream to be an infinite sequence of data items, where items are appended to the sequence over time and items in the sequence are ordered by a timestamp. Accordingly, each stream data item as a tuple < v, t> where v is a value (or set of values) representing the data item content, and t is the time at which this item joined the stream. The data content v can be a single value, a vector of values or NULL, and each value can be a simple or composite data type. Time t is our ordering mechanism, and the time stamp is the sequence number implicitly attached to each new data item. The time stamp may be assigned to the data item at its source or at the query processor [11]. As an example application of the stream model, the single data item in a video stream can be defined as the < frame, t>, where frame is an abstract data type representing frame content and t is the timestamp assigned at the query processor. Note that the frame data type includes different attributes such as FrameID, size, type (I, P or B frames for MPEG video), headers and binary content.

Some of the traditional SQL operations, such as selection and projection, have semantics similar to the relational model when applied to the processing of data streams. Selection operations select stream data items that satisfy a predicate condition (Boolean expression), much the same as selection in the relational model. Projection is also similar to its relational model equivalent, where a mapping function is repeatedly executed for each stream data item. These two operations are directly applicable to video processing when viewing video as streams of frames. As an example, a selection operation could select frames that satisfy the selection condition: "select I-Frames from the video stream" and a projection operation function Low Resolution() could be applied to every frame to produce video streams with reduced (lower resolution) quality. This may be important for applications which stream video through network links with slow bandwidth.

The binary form of the join operation finds the correlated items in two data sources. For a binary join, a data item from one source (the outer) is compared against all data items in the other source (the inner) to produce the matching pairs. This definition is clearly applicable when the data sources are non-streams or if the inner-stream is a non-stream data source. For all stream data sources, iteration on all items on the inner stream is not possible since the

stream is assumed to be infinite. Therefore, for joining two stream data sources, a restricted form of the join referred to as window-join is used [10]. In the window-join, only part of the data stream (a window) is considered for the join. An example SQL query that includes W-join is the tracking of objects that appear in multiple data streams from multiple cameras. For an object Obj that requires w time units to travel between two monitoring cameras, the query is posed:

SELECT A.Obj
FROM Camera1 A, Camera2 B WHERE similar (A.Obj, B.Obj) WINDOW w

Where similar () is a user-defined function that determines when two objects captured by different cameras are similar. This function can be considered as an equality predicate on object identifiers, A.Obj = B.Obj.

In the video stream processing application domain, the following types of joins are applicable:

(a) Joining a video stream with a non-stream data source, for example when searching for matching frames between a video stream generated by a monitoring camera and a stored database of images.

(b) Window-join for two data streams, for example when tracking objects that appear in video data streams from two monitoring cameras. The objects are identified in each data stream and the maximum time for the object to travel through the monitoring devices defines an implicit time window for the join operation.

A special type of the JOIN operation is the OUTER-JOIN, where tuples from left, right or either streams are always produced as output, regardless of whether they satisfy the join condition.

5. CONCLUSION

Video-based applications require strong video database support, including efficient methods for handling content-based query and retrieval of portions of video data. This paper has described video query processor that support video-based operations for search by content and streaming, new video query types, and the incorporation of video methods and operators in generating, optimizing and executing query plans. This paper also described VDBMS database support for video query processing in two contexts: first as applied to the video data type and then as applied to the stream data type. The VDBMS query capability was designed to support a full range of functionality for video processing, based on the development and integration of video as an abstract database data type (VDT). The paper also explained two query operators for the VDT which implement the rank-join and stop-after algorithms. This paper then considered video data when as streams of consecutive image frames, and expressed video query processing as continuous queries over video data streams. Lastly, this paper explained stream data type (SDT) and its functionality to support general data streams.

6. REFERENCES

[1] Aref,W.,& etal,A video database management system for advancing video database research. In Proc.of the Int Workshop on Management Information Systems. Nov 2002. Tempe, Arizona.

[2] Aref,W.,& etal. A distributed server for continuous media. In Proc. 18th Conf. on DataEng. San Jose, CA.Feb.2002.

[3] Fan, J., & etal, A.An improved isotropic color edge detection technique.Pattern Recognitio nLetters.pp.1419-1429,2001.

[4] Fagin, R.,& etal Optimal aggregation algorithms for middleware. In Proc. PODS'01 Santa Barbara, CA. May 2001

[5] Hellerstein, J., & etal, generalized search trees for database systems. In VLDB'95, Proc.of 21st International Conf .on Very Large DataBases. September 11-15. Zurich, Switzerland.1995.

[6] Hammad,M.,& etal A. Search- based buffer management policies for streaming in continuous media. In Proc.IEEE Intl.Conf.on Multimedia & Expo .Lausanne, Switzerland. Aug.2002.

[7] J.,Hacid,& etal A Model-based video classification for hierarchical video access. Multimedia Tool sand Appls..Vol.15.Oct 2001.

[8] Natsev,A.,& etal Supporting incremental join queries on ranked inputs. In Proc. Of 27th Conf. on Very Large DataBases. Rome, Italy. 2001.

[9] Smith, J., Sequentiality and prefetching in database systems.ACM Trans. on Database Systems.3(3).pp.223-247.September 1978.

[10] Storage Manager Architecture. Shore Documentation, Computer Sciences Department. UW-Madison,June1999

[11] S. Madden,& etal The design of an acquisitional query processor for sensor networks. In *Proc. of the SIGMOD Conf.*2003.

[12] www.cs.purdue.edu/icds.

[13] www.cs.bilkent.edu.tr/ bilvideo