

Indexes for User Access to Large Video Databases

Lawrence A. Rowe, John S. Boreczky, and Charles A. Eads

Computer Science Division - EECS

University of California Berkeley

Berkeley, CA 94720

(rowe@cs.berkeley.edu)

Abstract

Video-on-Demand systems have received a good deal of attention recently. Few studies, however, have addressed the problem of locating a video of interest in a large video database. This paper describes the design and implementation of a metadata database and query interface that attempts to solve this information retrieval problem.

Sample queries were collected by interviewing a variety of users. These queries guided the design of indexes that can be used to answer the types of queries users want to ask. Three types of indexes were identified: 1) bibliographic (e.g., title, abstract, subject keywords, genre, director, cast, etc.), 2) structural (e.g., segments, scenes, shots, transitions, sound effects, etc.), and 3) content (e.g., keyframe sequences, scripts, objects and actors in scenes, etc.).

A mixed-mode query interface was built that allows a user to select a set of videos and/or frame sequences and incrementally modify the answer set. The interface includes relational, hierarchical browsing, and keyword search operations.

1. Introduction

Many research and commercial groups are developing Video-on-Demand (VOD) systems that will allow users to play digital movies and videos. These systems are designed to deliver movies to a set-top device connected to a television in a home or to a multimedia desktop computer. Developers of these systems have focussed on the network and file system performance problems of servicing many concurrent users [7][9][20][23][29]. They have not addressed the problem of finding a video in a large collection because these VOD applications require access to relatively few distinct titles. For example, the SGI/Time-Warner experiment in Orlando will provide access to 250 movies. These applications are essentially on-line video rental stores as opposed to a video library that has thousands of hours of video material and hyperlinks to other material. A significant problem in any large library such as a video library is locating a desired set of video sequences.

This paper describes the design and implementation of a metadata database and query interface for searching a large video database. This research has four goals: 1) determine the types of queries users will perform, 2) investigate the indexes needed to answer those queries, 3) experiment with techniques needed to create these indexes from source material, and 4) compare user interfaces to access these indexes.

Our first problem was to characterize the types of queries that users want to ask. We collected sample queries by surveying a variety of users. At the same time we developed a data model to represent the indexes required to answer these queries. The following three types of metadata indexes were identified:

1. **Bibliographic Data.** This category includes information about the video (e.g., title, abstract, subject, and genre) and the individuals involved in the video (e.g., producer, director, and cast).
2. **Structural Data.** Videos and movies can be described by a hierarchy of movie, segment, scene, and shot where each entry in the hierarchy is composed of one or more entries at a lower level (e.g., a scene is composed of a sequence of shots and a segment is composed of a sequence of scenes [4]). A structural index can also be constructed for the audio track.
3. **Content Data.** Users want to retrieve videos based on their content. Examples of content indexes are: 1) sets of keyframes that represent key images in a video (e.g., frames that show each actor in the video or a sequence of images that depict the major segments and scenes in the video), 2) keyword indexes built from the sound track, and 3) object indexes that indicate entry and exit frames for each appearance of a significant object or individual.

The second problem addressed was the development of an easy-to-use ad hoc query interface. We built a mixed-mode interface that allows a user to select a desired set of videos and/or frame sequences from the database. The user can play one or

more selected videos, ask for more information about the videos, or incrementally modify the set to reduce the number of selected videos. The interface includes general relational operators (e.g., “select demonstration videos created by universities”), hierarchical browsers (e.g., show a list of topics from a course and allow the user to narrow the search to a list of subtopics within a topic), and keyword searches (e.g., “Select videos about ‘ATM networks’”).

These problems have been investigated by other researchers, although little work has been done on combining video indexing with video retrieval. The *Virtual Video Browser* [12] is a video retrieval system that stores limited bibliographic, actor, and scene information. O’Connor discusses the use of scene descriptions and keyframes as a surrogate for browsing video material in a library setting where access to the actual video is not always feasible [16]. The *QBIC* system allows users to query a small set of still images by color, texture, position, and shape of objects [15]. There are also a number of commercial CD-ROM movie databases [14][19][30]. These systems support computerized search of the information found in a typical movie guide published in book form [13]. In some cases, still images and short clips are included on the CD-ROM.

There has also been a great deal of work on using computers to automatically extract video content. The simplest and most common technique is video segmentation (i.e., detecting the boundaries between camera shots). Recent research in this area includes work on shot detection using compressed video [1] and work on automatic classification of shot transitions [31]. Swanberg has used a knowledge base to do automatic segmenting and scene classification for a highly structured type of video [28]. Research has also been done on object tracking and classification of object motion in video [11]. While most of this work will be useful in building the indexes we require, considerable progress will be needed before sophisticated content indexes (e.g., actors entering and leaving a scene) can be automatically created.

The system on which our work is built is the Berkeley Distributed VOD System. The goal of this system is to provide access to a video database that contains hundreds of hours of instructional and other videos both locally and across the Internet. Figure 1 shows the architecture of the Berkeley Distributed VOD System. The system is composed of a database, one or more video file servers (VFS), and one or more archive servers that manage tertiary storage devices (e.g., optical disk or tape jukeboxes). The database contains metadata and indexes about the videos stored in the system and tracks the location of video material cached on one of the VFSs. It is also used to schedule the movement of data between tertiary storage and the servers. A description of the storage management aspects of the system is available elsewhere [6].

FIGURE 1. Berkeley Distributed Video-on-Demand System architecture

This system will be accessed using an ad hoc query interface, called the *Video Database Browser* (VDB), that allows a user to query the database to locate interesting video material and then, if necessary, schedule it to be retrieved from the tertiary store. Our ultimate goal is to build a distributed system that will support archives at different geographic locations and use local VFSs to cache videos for playback. Users will be oblivious to where a video is stored or its format. To make this system a reality, we will also have to address problems of limiting access, protecting video data from being copied, and collecting usage fees. We are also developing a Mosaic interface to the system.

The remainder of this paper describes the design of the metadata database and query interface. Section 2 describes the queries that the system will accommodate. Section 3 describes the database schema and techniques that will be used to implement user queries. Section 4 describes the ad hoc query GUI. Section 5 discusses the implementation status and plans for further development.

2. Types of Queries

This section describes a survey of users we conducted to determine the types of queries that the system must support. These queries are categorized to determine the indexes needed in the database.

It is impossible to design a system to answer queries without determining the types of questions users will ask. We collected sample queries by surveying ten users from different backgrounds including: 1) faculty and students in Computer Science, 2) postgraduate biology researchers, 3) film librarians, and 4) the general population. Some of the questions these respondents asked are listed below.

Questions about course lectures:

1. Find lectures where the instructor is seated at a desk/standing at the board.
2. Which courses describe hash tables?
3. Show topics for CS 164 for spring 89 and fall 92.
4. Show me a list of all courses, who taught them, when they were taught, and course descriptions.
5. Explain this item "CS 164".
6. List guest lectures in a specific course.
7. Show the assigned homework for CS 186.
8. In CS 164, over the last 4 years, which professor had the highest rating.
9. Find all references to CPR, show lectures where its history is discussed, and show a demonstration.
10. Show a hierarchy of all the lectures you have and let me browse until I've narrowed it down to what I want.

Questions about motion pictures:

1. Show me the next diving scene in *The Abyss* after the Navy Seal dies.
2. Show me scenes where a person is hitting a baseball.
3. List all the comedies.
4. List the comedies directed by a director who usually makes horror movies.
5. List movies set in Los Angeles.
6. List movies with box office receipts over \$100 million.
7. List all the movies about dinosaurs.
8. What was the cost of making this movie?
9. List movies released between Dec. 25 and Dec. 31.
10. List the Oscar winners from 1987.
11. List the movies that got the highest critic ratings.
12. List movies with John Wayne acting and John Ford directing.
13. List all movies with kisses that last more than 5 seconds.
14. List movies that use the song "Happy Birthday."
15. Let me browse video box descriptions, like I'm browsing boxes in a video store.
16. List video releases for the last 3 years sorted by release date.
17. Let me browse a list of titles, pictures of actors and actresses, and all of the scenes from a movie.

We grouped these sample queries into four categories. The first category is queries about the movies as a whole, such as asking about Academy Award winners or favorite directors. This type of query can be answered with a bibliographic index. A bibliographic index includes "back-of-the-video-box" information such as title, director, subject, length, and genre. The second group is queries about topic content, such as items that are discussed by actors. These queries can be answered with a keyword index. A keyword index includes occurrence information for all non-trivial words that are associated with a movie,

such as spoken lines and words in the title and abstract. The third group is queries about sensory content, such as the appearance or location of objects. These queries can be answered with an object index. An object index contains information about the physical appearance and location within the movie of important objects such as actors, props, and scenery. The final category is queries about the data and metadata. For example, a user may be watching a movie and want to find out the name of an actor by selecting him or her in a scene and then asking to see a list of other movies in which they appeared.

Most questions people want to ask are about bibliographic information, likely because the only access that is commonly available for video data is bibliographic. There was some interest in using keyword queries for course lectures. For example, using a word index for lectures which are information-packed because it is similar to looking up words in a textbook index. A few people mentioned a desire to browse collections of videos and clips. They wanted an organized collection of items and a way to quickly scan them [8]. Visual content queries usually involved actors and their actions. Surprisingly, only people who were involved with multimedia work expressed an interest in asking detailed object or structural queries.

3. Database Schema

This section describes the database schema used to store video information and the methods we will use to answer user queries.

3.1. Database Overview

A comprehensive set of indexes is needed to satisfy the types of questions that users want to ask. We need four types of indexes: 1) bibliographic, 2) structural, 3) object, and 4) keyword. These four index types are described in further detail below.

The POSTGRES relational database management system is used to store the video indexes and to provide stable storage for coordination between components of the Distributed VOD System [27]. The indexes contain pointers to the location of the compressed video and audio data on tertiary and secondary storage devices. The video database is integrated into the Sequoia 2000 (S2K) database that contains images, text documents, and global change data sets [26]. The subset of the database that covers video material currently contains 62 classes¹, but this number will grow as different types of movies and movie formats are added and better query interface components are built. Table 1 shows the main classes in the database schema.

Index Type	Database Class	Contains Entry for Each
Document	DOCS	document
Bibliographic	VIDEO_BIB	video document containing bibliographic data
Structural	VSV_SHOT	shot in each video
	VSV_SCENE	scene in each video
	VSV_SEGMENT	segment in each video
Object	PEOPLE	person associated with any document
	OBJECT	item occurring in any document
	OBJ_INST	item occurring in a video
Keyword	KW_WORDS	keyword in any document

TABLE 1. Main classes in database schema

The central class of the database is the DOCS class that contains one entry for each document. The DOCS class contains the following attributes²:

<i>docid</i>	unique identifier of document
<i>docs_name</i>	name of the document (e.g., movie title)
<i>docs_creator</i>	name of the person who added the document to the database
<i>docs_comments</i>	free-form description of document
<i>docs_date</i>	date when added to the database
...	

Other classes reference a document using the unique ID of the document from the DOCS class (i.e., *docid*).

¹ Tables in POSTGRES are called classes.

² Columns in a POSTGRES database are called attributes.

A video can be stored in the database in one or more formats including Apple Quicktime, Microsoft Video-for-Windows, and UCB Script (UCBS). Each format uses several classes to represent information about the video so it can be queried or played. For example, useful parameters about the video such as image size and depth are extracted from the complex object used by most video storage representations to make the information more accessible.

In addition to allowing entire documents to be retrieved, we will provide access to subparts of documents, called *analytical indexing*. For videos or films, this type of indexing means allowing access to tracks (e.g., video and audio), frames or sets of frames, and frame sequences. These indexes will allow users to search for and view portions of a movie (e.g., “the airport scene from *Casablanca*”).

3.2. Bibliographic Indexes

As shown in the user survey most of the time video material is accessed through standard bibliographic information such as title and author as well as video-specific information such as genre¹ and media format. Bibliographic information is also available as MARC records so the video catalog can be accessed using other bibliographic interfaces such as WAIS [25] and the World Wide Web [3].

The main bibliographic data index (DOC_REFERENCE) contains a subclass for each type of document in the database. Some of the attributes in the VIDEO_BIB subclass are:

<i>ref_title</i>	title of document
<i>ref_abstract</i>	abstract of document
<i>ref_entry</i>	date of entry of bibliographic data
<i>ref_comments</i>	comments about bibliographic data
<i>ref_date</i>	release/creation date of video
<i>ref_org</i>	name of organization that produced the video
<i>vbib_length</i>	length in hours:minutes:seconds:frames
<i>vbib_genre</i>	category of video (e.g., comedy, documentary, etc.)
<i>vbib_cast</i>	text list of cast members (allows rapid access)
<i>vbib_director</i>	video director
<i>vbib_awards</i>	Golden Globe and Academy Award nominations and wins
<i>vbib_path</i>	series hierarchy
...	

There are other types of bibliographic information associated with a document such as subject material, geographic location, and supplemental material. Subject information is stored in a hierarchy of thesaurus entries as in the S2K data schema which allows information about narrower and broader terms and cross-listings to be accessed. For example, a user looking for documents about ‘computer animation’ might be interested in documents listed as ‘computer graphics’ (a cross-listing) or ‘animation’ (a broader term). Given the restricted nature of the video material we are making available in our pilot study, broad subject headings will be insufficient. For example, all lectures for a course fall under the same subject headings so they do not provide any help to a user trying to locate a lecture on a particular topic. We are using Library of Congress Subject Headings (LCSH) with extensions such as the computer topics covered in the course lectures.

For certain types of videos, such as course lectures, series information provides a more convenient method of broad subject access. A user might want to look at all course lectures, or all Computer Science lectures, or even all Berkeley CS 164 lectures. Because this information is hierarchical, only the path name from the root needs to be recorded for each video. For example, a CS 164 lecture might have the path “Movie:Educational:Lecture:Berkeley:CS:164” which will accommodate all of the above queries.

Movies are usually associated with a physical location, for example, motion picture settings and news story locations. To support queries on videos by geographical location, a world map with icons that represent where the video was filmed will be shown to the user. This type of two-dimensional display might also be useful for browsing documents based on other types of attributes. The geographic browser panel is based on the map widget included in the Lassen Text Browser [10].

Movies often have text information and other material such as poster images, press clippings, stills, and the shooting script that should be available to users. Course lectures can be supplemented viewgraphs and handouts. This type of auxiliary data will be linked to the DOCS class entry for the video.

¹ Genre is a type or category of movie (e.g., *King Kong* is a monster drama).

3.3. Structural Indexes

There is a great deal of research interest in document structure both for text documents and movies. Most research on video structure involves automatically detecting the boundaries between camera shots. Breaking a video into its component parts allows portions of a video to be indexed and retrieved. In addition, students of movie and video making want to study how a particular scene was presented (e.g., duration, time between camera cuts, transitions, audio effects, etc.).

There are two main types of video structure: visual and outline. Most videos contain aspects of both, although one usually predominates. Visual structure is based on grouping events that take place close together in time or space. It is most evident in videos that tell a story such as motion pictures, home movies, and television shows. Visual structure is a bottom-up hierarchy, with the camera shot at the bottom and scenes, segments, and movies built on top. Outline structure is based on the topics that are shown or discussed in a video. It typically consists of a hierarchy of main topics and several levels of subtopics. Outline structure is common in course lectures, seminars and conferences. Other types of videos, such as news programs, product videos, research demonstrations, and documentaries, have both types of structure.

The obvious visual structure entities for a movie are the movie as a whole and the shot. A shot is an unbroken sequence of frames from one camera. Thus, a scene that alternated between views of two people contains multiple shots. We also determine the types of transitions that occur between shots such as fades or wipes. Some of the attributes in the VSV_SHOT class which represents this information are:

<i>time</i>	start and end time in hours:minutes:seconds:frames
<i>description</i>	text description of shot
<i>timedesc</i>	text description of time of day depicted (e.g., daybreak, twilight, midday)
<i>docid</i>	document identifier
<i>opentrans</i>	type of transition that opens shot (e.g., fade, wipe, dissolve)
<i>closetrans</i>	type of transition that closes shot
<i>shottype</i>	type of shot (e.g, closeup, medium, full)
...	

A scene is defined as a collection of one or more adjoining shots that focus on an object or objects of interest. For example, a person walking down a hallway into a room is one scene, even though there might be different camera angles shown. Three camera shots showing three different people walking down a hallway might be one scene if the important object was the hallway and not the people. A collection of scenes that represents a separable component of a movie is defined as a segment. (Segments are sometimes called *sequences*). For example, each story on “60 Minutes” is a segment, as is a flashback in a motion picture. Attributes in the VSV_SCENE class which represents this information include:

<i>time</i>	start and end time in hours:minutes:seconds:frames
<i>description</i>	text description of shot
<i>timedesc</i>	text description of time of day depicted (e.g., daybreak, twilight, midday)
<i>docid</i>	document identifier
<i>focalpt</i>	object that defines shot (e.g., actor, locale)
<i>cast</i>	cast members that appear in the scene
<i>objects</i>	objects that appear in the scene
<i>location</i>	geographic location where the scene takes place
...	

The VSV_SEGMENT class contains similar attributes.

A structural schema for audio will be useful, since sound effects or different moods of background music often indicate transitions. These audio boundaries do not always correspond to visual boundaries. Changes in background music, appearances of songs, and sound effects can be time stamped and made available for queries. We are still working on this part of the schema.

3.4. Keyword Indexes

Keyword indexes are very similar to the indexes used in the Lassen Text Browser [10]. The WordNet database and software, developed by Princeton University, is used to find the stems of input words and place words in groups with their synonyms. Lassen adds an information retrieval system, based on the SMART system, that allows documents and queries to be represented by weighted vectors of keywords [22]. The vector representation allows for faster processing of keyword queries than would be possible using standard relational database methods. The results of the keyword search are ranked according to how well they match the query terms.

Because we want to access portions of a video, it is necessary to record the location of each occurrence of a keyword which requires a set of time stamps or time ranges for each keyword. These time stamps will allow temporal queries such as “Find all references to distributed computing and debugging within 2 minutes of each other.”

Because keywords can appear in different pieces of data associated with movies such as the shooting script, the title, the abstract, and movie reviews, the link between keywords and documents is complicated. The KW_WORDS class holds information about the keywords and includes the following attributes:

<i>word</i>	the stem of the keyword
<i>numdocs</i>	the number of documents that contain the keyword
<i>freq</i>	total number of occurrences of the keyword in all documents

The KW_ITEMS class stores information about the places that a particular keyword can appear. For example, the same word can appear in a title, abstract, and script of a video. There will be one entry in KW_ITEMS for each place containing the following attributes:

<i>docid</i>	unique identifier of document
<i>itemlen</i>	length of indexed item in words
<i>itemuni</i>	number of unique words in item
<i>entry</i>	when item was indexed
<i>indexer</i>	who created the index data
<i>attrvalptr</i>	pointer to exact location in database where text is stored (i.e., class, attribute, and tuple)

The KW_ITEM_LINK class links keyword occurrences to documents:

<i>kwitemid</i>	entry in KW_ITEMS
<i>wordid</i>	entry in KW_WORDS
<i>freq</i>	number of occurrences
<i>time</i>	array of occurrence times in hours:minutes:seconds:frames

To illustrate the use of these classes, consider the simple keyword query, “video transistors.” Figure 2 shows the classes and attribute values for one video retrieved by this query.

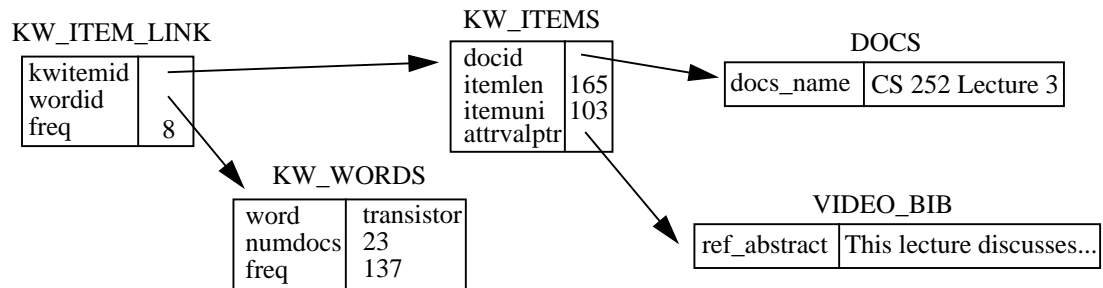


FIGURE 2. Classes involved in a simple keyword query

For this particular document, the keyword “transistor” appears in the abstract, so the KW_ITEMS class has an entry that describes and points to the abstract. The KW_ITEM_LINK class has an entry that links the keyword to that item and indicates that the keyword appears 8 times in the item. Notice that we do not create an index to each occurrence in abstract, just one entry to indicate that it exists. The viewer that displays the text to the user can either search for the occurrences or, if the document is large or complex such as a script, a separate index can be created. The specific video can be retrieved using the pointer contained in the KW_ITEMS class.

3.5. Object Indexes

Another type of content information available from the visual portion of a movie are the objects that appear on camera. Important objects that will be indexed include actors and actresses, objects important to the plot (e.g., the murder weapon), and visually interesting items (e.g., loud ties). For each object, we record the following information: 1) what type of object it is, 2) what it is doing (or what is being done to it), 3) visual characteristics such as color, texture, and shape, 4) a few key frames that are representative of the object’s appearance, 5) when the object is on camera, and 6) when the object is off camera but still present.

Objects are thus similar to keywords in that they have attributes and are associated with times or time ranges in a movie. The information retrieval techniques described for keywords will be applied to object queries.

There are several database classes that store object information. The PEOPLE class contains the following attributes:

<i>name</i>	name of individual
<i>sex</i>	male or female
<i>bio</i>	text biography
<i>mainjob</i>	person's normal position (e.g., actor, director, writer)
<i>numdocs</i>	number of documents person is associated with
...	

The OBJECT class which contains an entry for each significant object contains the attributes:

<i>objtype</i>	type of object (e.g., animal, building, explosion, etc.)
<i>name</i>	name of object
<i>description</i>	text description of object
<i>numdocs</i>	number of documents that object appears in
...	

The OBJ_INST class which contains information about the appearance of an object in a movie contains the attributes:

<i>location</i>	position on-screen
<i>action</i>	what object is doing or what is being done to it
<i>motion</i>	direction and speed of motion
<i>time</i>	array of occurrence intervals in hours:minutes:seconds:frames
...	

3.6. Schema relationships

Figure 3 shows an entity-relationship diagram for the database.

3.7. Postgres examples taken from sample questions

This section demonstrates how user queries can be answered by showing how a query is represented as a POSTGRES query using the POSTQUEL query language and the metadata database. It is interesting to note that it is easier to phrase these queries in English than in a database query language. Unfortunately, it is not as easy to use a natural language interface as a query interface [2].

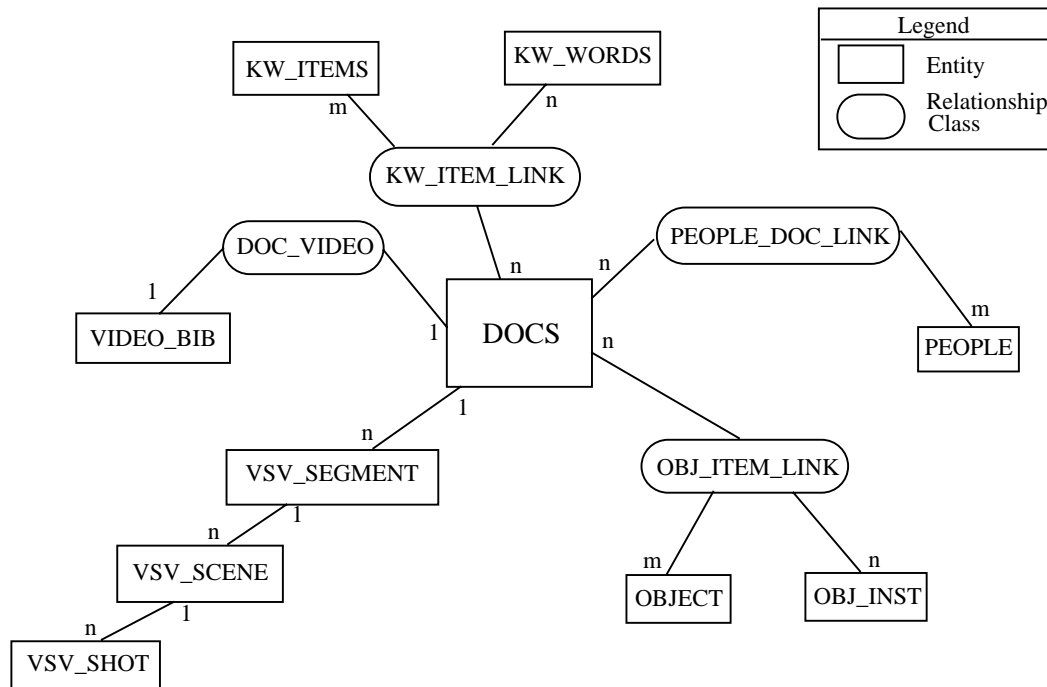


FIGURE 3. Entity-Relationship diagram of the database schema

The first query is “movies produced by Universal in the 1960’s in which Elvis did not appear.” The solution is:

```
retrieve (movie.all) from movie in DOCS, bib in VIDEO_BIB, dv in DOC_VIDEO
where dv.docid = movie.oid and dv.base_reference = bib.oid
and bib.ref_org = “Universal Pictures”
and bib.ref_date between [“Jan 1 1960”, “Jan 1 1970”]
and bib.vbib_cast !contains “Elvis Presley”
```

Note that a cast list is included in the VIDEO_BIB class to simplify the common query of checking the cast that appears in a movie. If an actor query is more complicated, such as retrieving the name of the portrayed character, the PEOPLE class must be accessed.

The second query is “movies in which Meryl Streep appeared in 3 or fewer scenes.” The first part of the solution retrieves a list of all movies in which Meryl Streep appeared into a temporary class:

```
retrieve into TEMP (movie.docid, movie.count=0)
from movie in DOCS, bib in VIDEO_BIB, dv in DOC_VIDEO
where dv.docid = movie.oid and dv.base_reference = bib.oid
and bib.cast contains “Meryl Streep”
```

Next, for each of movie, we count the number of scenes in which she appeared:

```
replace TEMP (count=count + 1) from vs in VSV_SCENE
where vs.docid = docid and vs.cast contains “Meryl Streep”
```

Finally, we retrieve the movies where the appearance count is 3 or less:

```
retrieve (movie.all) from movie in DOCS, temp in TEMP
where movie.oid = temp.docid and temp.count <= 3
```

The last query is “find movies that show John Wayne riding off into the sunset on a horse.” First we find movies where John Wayne rides a horse:

```
retrieve into TEMP (movie.docid, inst.time)
from movie in DOCS, obj in OBJECT, inst in OBJ_INST, link in OBJ_DOC_LINK
where link.inst = inst.oid and link.itemid = movie.oid and link.objid = obj.oid
and obj.name = “John Wayne” and inst.action = “riding horse”
```

Then, we find the scenes that occurred at dusk:

```
retrieve (movie.all) from movie in DOCS, temp in TEMP, vs in VSV_SCENE
where movie.oid = temp.docid and vs.docid = temp.docid
and temp.time overlaps vs.time
and vs.timedesc contains “dusk”
```

4. Video Database Browser

The Video Database Browser (VDB) interface allows a user to select a desired set of videos and/or frame sequences. This selection process will occur via the application of incremental reductions to a set of videos until the desired, or at least manageable, set of videos is obtained. The interface includes general relational operators, navigational operators to express structural queries, and keyword search operators.

Once a movie has been selected using the query interface, the Continuous Media Player (CMPlayer) [21] is used to play the video. The CMPlayer provides synchronized playback and random access to audio and MPEG and motion-JPEG video over local and wide area networks. VDB also allows a user to play movies on a local VFSs directly.

To handle the differing needs of the users of a video database, a flexible interface that provides multiple access points to any given piece of data is required. Result data can be displayed in different formats (e.g., a table that lists movie titles, a canvas that displays keyframes labeled with the movie title, etc.) or it can be passed to another tool (e.g., a video email composer, a video editor, etc.). For example, Figure 4 shows a whiteboard viewer that displays one keyframe per video. From this display a user can play a video or zoom into a storyboard viewer that shows one keyframe per scene. Another possible viewer that could be implemented is MIT frozen movies that represent frames over time as 3 dimensional solids and allow viewing along any slice [5].

Figure 5 shows the results of a query performed using the browser for hierarchical series data. The query is specified by pressing buttons in the directed graph that represents the hierarchy. The result set can be reduced by further restricting the path or calling up a panel to restrict the list to videos with a particular word in the title. Or, the set could be reduced by using a keyword search on the script of the selected videos.

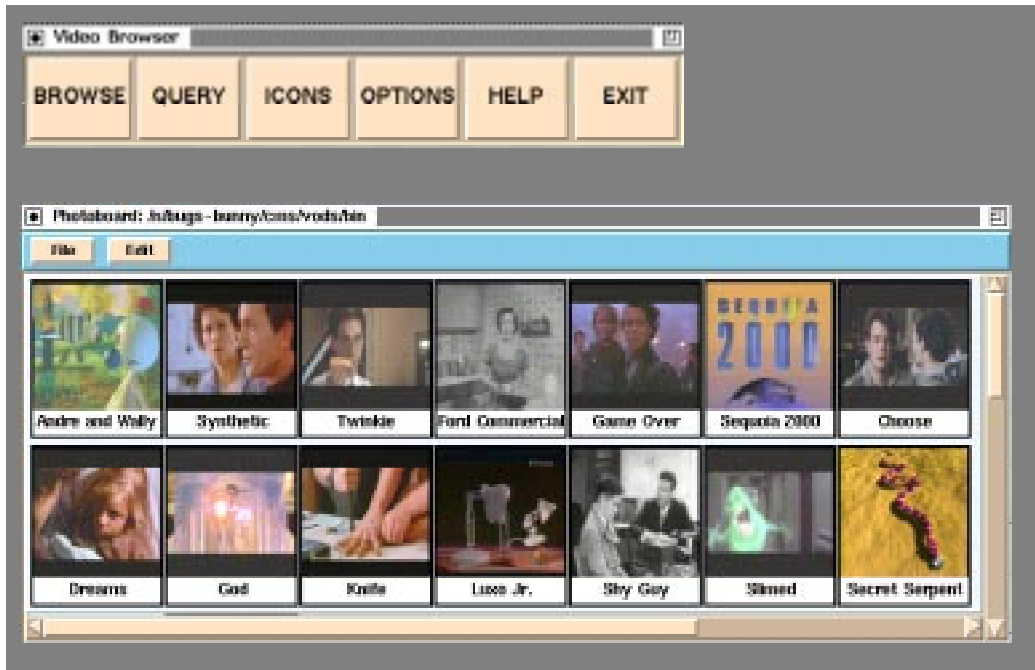


FIGURE 4. Button bar and whiteboard

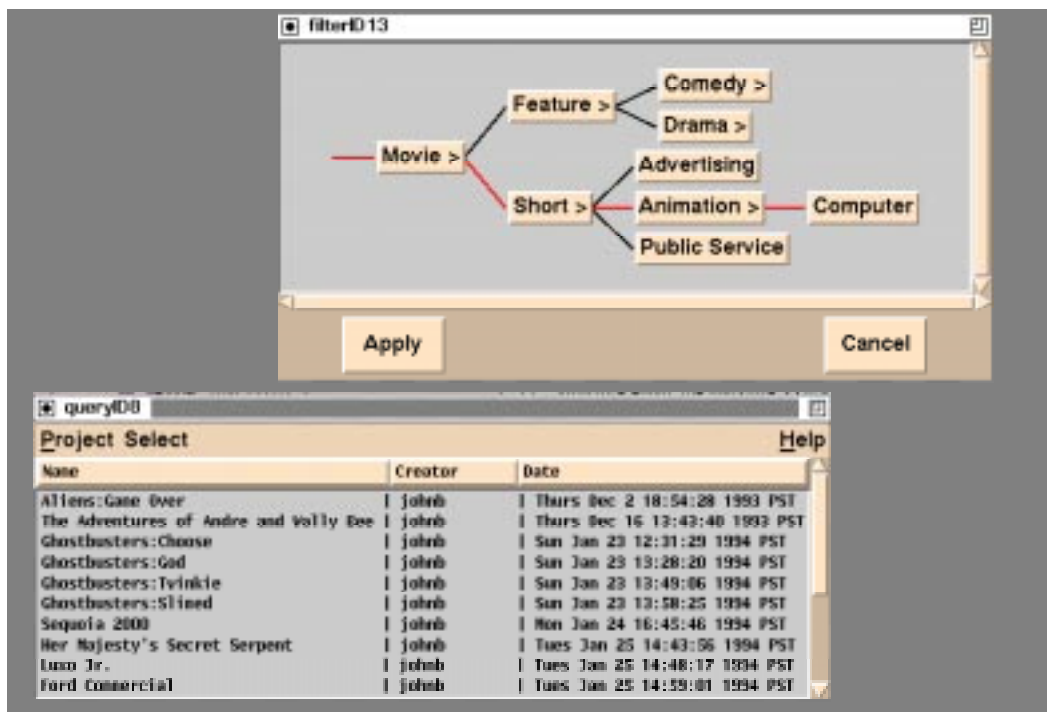


FIGURE 5. Series browser and query result

5. Current Implementation

The database currently contains approximately 1.5 hours of digitized video material in 20 short clips. We are digitizing approximately 1 hour of video per week. We hope to be digitizing 5-10 hours/week by Summer 1994. We are also continuing to refine the database schema and load metadata into the database. We have loaded the *rec.arts.movie* database which contains bibliographic data for over 25,000 movies, and we are currently loading the shooting script for a feature film.

The VDB is written in Tcl/Tk [17][18] with extensions such as a Tcl interface to POSTGRES and the icon display widget. The browser currently has 5500 lines of Tcl/Tk code. VDB currently supports bibliographic, series, and keyword queries. We need to complete the panels to specify relational queries (e.g., restrictions and projections) and to add other index-specific query interfaces. Lastly, we are designing a human factors experiment to test the GUI interface.

6. Acknowledgments

This research was supported by the National Science Foundation (Grant MIP 90-14940), Fujitsu Network Transmissions Systems, Inc., Hewlett-Packard Company, and Hitachi America, Ltd.

7. References

1. Arman, F., Hsu, A., and Chiu, M-Y., "Image Processing on Compressed Data for Large Video Databases", Proc. ACM Multimedia 93, Anaheim, CA, August, 1993.
2. Bell, J. and Rowe, L.A., "An Exploratory Study of Ad Hoc Query Languages to Databases", Proc. IEEE 8th Int'l Conf. on Data Engineering, February, 1992.
3. Berners-Lee, T.J., Cailliau, R., Groff, J.F., and Pollerman, B., "World-Wide-Web: The Information Universe," Electronic Networking: Research, Applications, and Policy, Spring 1992, Vol. 2, No. 1, pp. 52-58.
4. Davenport, G., Smith, T.G.A., and Pincever, N., "Cinematic Primitives for Multimedia", IEEE Computer Graphics & Applications, July 1991, pp. 67-74.
5. Elliot, E., "Multiple Views of Digital Video", MIT Media Lab, Interactive Cinema Working Paper, Massachusetts Institute of Technology, March 1992.
6. Federighi, C. and Rowe, L., "A Distributed Hierarchical Storage Manager for a Video-on-Demand System", 1994 IS&T/SPIE Symposium on Electronic Imaging: Science and Technology, San Jose, CA, February, 1994.
7. Haskins, R., "The Shark Continuous-Media File Server", Proc. IEEE COMPCON '93, San Francisco, CA, February 1993.
8. Hsu, A. and Arman F., personal communication, January, 1994.
9. Keeton, K. and Katz, R., "The Evaluation of Video Layout Strategies on a High-Bandwidth File Server", Fourth Int'l. Workshop on Operating Systems and Network Support for Digital Audio and Video, 1993.
10. Larson, R., "Classification Clustering, Probabilistic Information Retrieval and the Online Catalog," Library Quarterly, vol 61, April 1991.
11. Lee, S-Y. and Kao, H-M., "Video Indexing - an Approach Based on Moving Object and Track", 1993 IS&T/SPIE Symposium on Electronic Imaging: Science and Technology, San Jose, CA, February, 1993.
12. Little, T.D.C, Ahanger, G., Folz, R.J., Gibbon, J.F., Reeve, F.W., Schelleng, D.H., and Venkatesh, D., "A Digital On-Demand Video Service Supporting Content-Based Queries", Proc. ACM Multimedia 93, Anaheim, CA, August, 1993.
13. Martin, M. and Porter, M., *Video Movie Guide 1992*, Ballantine Books, New York, 1991.
14. Microsoft Corporation, "Microsoft Cinemania: User's Guide", Microsoft Corporation, Redmond, WA, 1992.
15. Niblack, W., Barber, R., Equitz, W., Flickner, M., Petkovic, D., and Yanker, P., "The QBIC Project: Querying Images by Content Using Color, Texture, and Shape", 1993 IS&T/SPIE Symposium on Electronic Imaging: Science and Technology, San Jose, CA, February, 1993.
16. O'Connor, B.C., "Access to Film and Video Works: Surrogates for Moving Image Documents", dissertation, University of California, Berkeley, Dec. 1984.
17. Ousterhout, J., "Tcl: An embedded command language", Proceedings 1990 Winter USENIX Conference, 1990.
18. Ousterhout, J., "An X11 Toolkit based on the Tcl language", Proceedings 1991 Winter USENIX Conference, 1991.
19. Paramount Interactive, "MovieSelect", Paramount Interactive, Palo Alto, CA 1992.

20. Rangan, P.V., Vin, H.M., Ramanathan, S., "Designing an On-Demand Multimedia Service", IEEE Communications Magazine, Vol. 30, No. 7, July 1992, pp. 56-64.
21. Rowe, L.A., and Smith, B.C., "A Continuous Media Player," Proceedings 13th Symposium on Operating System Support for Digital Audio and Video, La Jolla, CA, November, 1992.
22. Salton, G., Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer, Addison-Wesley, Reading, MA, 1988.
23. Sesek, E., *personal communication*, Aug. 1993.
24. Smith, B.C. and Rowe L.A., "An Application-Specific Ad Hoc Query Interface", IEEE Trans. on Semiconductor Manufacturing, Vol. 5, No. 4, pp. 281-289, November 1992.
25. Stein, R., "Browsing Through Terabytes", Byte Magazine, May 1991, pp. 157-164.
26. Stonebraker, M. and Dozier, J., "Sequoia 2000: Large Capacity Object Servers to Support Global Change Research", Sequoia 2000 Technical Report 91/1, University of California, Berkeley, CA, July 1991.
27. Stonebraker, M. and Kemnitz, G., "The POSTGRES Next-Generation Database Management System," Communications of the ACM, Vol. 34, No. 10, October, 1991, pp. 78-92.
28. Swanberg, D., Shu C.F., and Jain, R., "Knowledge Guided Parsing and Retrieval in Video Databases", 1993 IS&T/SPIE Symposium on Electronic Imaging: Science and Technology, San Jose, CA, February, 1993.
29. Tobagi, F.A. and Pang, J., "StarWorks *TM -- A Video Applications Server", Proc. IEEE COMPCON '93, San Francisco, CA, February 1993.
30. The Voyager Company, "Criterion Goes to the Movies", The Voyager Company, Santa Monica, CA, 1992.
31. Zhang, H.J., Kankanhalli, A., and Smoliar, S.W., "Automatic Partitioning of Full-motion Video", Multimedia Systems (1993) 1:10-28.