Chabot: Retrieval from a Relational Database of Images

Virginia E. Ogle
Michael Stonebraker
University of California, Berkeley
{ginger, mike}@cs.berkeley.edu

Abstract

Chabot is a picture retrieval system for a database that will eventually include over 500,000 digitized multi-resolution images. We describe the design and construction of this system which uses the relational database management system POSTGRES for storing and managing the images and their associated textual data. For retrieval, Chabot uses tools provided by POSTGRES, such as representation of complex data types, a rich query language, and extensible types and functions. To implement retrieval from the current collection of 11,643 images, Chabot integrates the use of stored text and other data types with content-based analysis of the images to perform "concept queries".

1. Introduction

The Chabot project was initiated at UC Berkeley to study storage and retrieval from a large collection of digitized images. The images we use belong to the State of California Department of Water Resources (DWR), the agency that oversees the system of reservoirs, aqueducts and water pumping stations throughout California known as the State Water Project. DWR maintains a growing collection of over 500,000 photographs, negatives, and slides, primarily images of State Water Project facilities, but also many images of California natural resources. Some examples of these images are shown in Figure 1.

Over the years, as DWR has made its collection available to the public, it has found itself devoting increasing resources toward filling requests for prints and slides. The agency receives 100-150 requests a month from a variety of sources: other government agencies, regional magazines, encyclopedia, university libraries, wildlife organizations, and individuals. Requests vary from those where the ID number of the desired picture is already known, to very general requests for "scenic pictures" of California lakes and waterways. DWR keeps the slides that are requested most often in lighted display boxes for browsing; the rest of the collection is housed in archival containers and slide drawers.

To facilitate retrieval, DWR began a project last year to digitize all its images

using Photo-CD technology; several years ago the agency had begun to enter descriptive data about each image into a single-user personal computer style database. To process a request, the staff uses keyword look-up on the text descriptions stored in the database to find an ID number for the requested images. The ID is then used to locate the container or drawer where the print or slide is stored.

While an attempt is made to annotate each image with as much descriptive information as possible, keyword indexing for an image collection has significant limitations. Non-specific requests such as "find a scenic photo of Lake Tahoe" may entail looking through an unmanageably large set of images of Lake Tahoe to find the desired prints. Misspelled keywords ("azalia" for "azalea") thwart successful retrievals, even when close matches can be culled from a dictionary. And inaccurate descriptions are not helped by a dictionary: a photo of a bright red anemone in full flower with the stored description "close-up of a pansy" will never be retrieved using any spelling of the keyword "anemone". In some cases, a thesaurus might compensate for incorrect descriptions, but not for incomplete descriptions. Many images in the DWR collection are old and cannot be identified, so they are digitized and loaded into the database with minimal or no descriptive data.

Because of these types of problems, most of DWR's retrievals currently rely on having a staff member who is familiar enough with the collection to know where to find the desired prints. One of the goals of the Chabot project is to integrate image analysis techniques into the retrieval system so that requests for images do not depend solely on stored textual information. As a first step toward this goal, we have implemented a simple method for color analysis which we describe in Chapter 4. Using color information in the images in conjunction with textual information can help to find red flowers (the anemone and azalea pictures) and to locate scenic pictures of sunsets of Lake Tahoe.

Aside from the problem of depending on keywords to locate pictures, the current database system that DWR uses cannot support complex data types such as time, geographical location, and the images themselves. Nor does it allow the user to compose queries that combine several of the attributes of an image. Therefore, the Chabot project has the additional goal of providing DWR with a system that can store and search on diverse data types, and provide the functionality of an advanced relational database management system (DBMS) such as a high-level query language, query optimization, and flexible indexing. We use POSTGRES [6,9], an object-relational DBMS developed at the University of California, Berkeley. As we will describe, one of the advantages of using POSTGRES for this project is the ability to create user-defined functions and types. We use this feature

to perform run-time image analysis during the querying process.

Another consideration is that DWR would like to load and edit its database remotely and to enable browsing of its images by offsite users who are interested in ordering prints or slides. The current database does not meet the needs of the agency for on-line, multi-user access. Furthermore, it will not scale to accommodate the 500,000 images in the collection. The Chabot project was initiated to replace the existing system with a better system that includes:

- An advanced relational database for images and data
- Large-scale storage for images
- On-line browsing and retrieval of images
- A flexible, easy-to-use retrieval system
- Retrieval of images by content

In Chapter 2 we describe the motivation and goals for the project. Chapter 3 discusses current research in the field. Chapter 4 contains a description of the Chabot project, and in Chapter 5 we summarize the project and give our plans for further enhancements.

2. System Motivation and Goals

The design of Chabot is influenced by DWR's existing system for storing its metadata, by the types of requests it receives, and by the methods now in use for queries and updates.

Integration of Data Types: Each image is accompanied by a sizeable amount of metadata. Below is a sample entry for one image from DWR's existing database:

> 0162 A-9-98 6/1/69 SWP Lake Davis Lahontan Region (6) Grizzly Dam, spillway and Lake Davis, a scenic image. DWR 35 mm slide Aerial 2013 0556 18

In this example, "0162" is the first four digits of the CD number, "A-9-98" is the DWR ID, followed by the date the photo was taken (6/1/69), the category ("SWP"), the subject ("Lake Davis"), the location ("Lahontan Region (6)"), a description of the image, the organization ("DWR"), the type of film used, the perspective of the photo, the last eight digits of the Photo-CD, and finally, the number of the image on the Photo-CD.

DWR needs a DBMS that can support a variety of complex data types including text, numerical data, relative and absolute time, and geographical location. Retrievals should be possible on any combination of the complex data types that are associated with the images, as well as on the content of the images themselves.

Scalability and Storage Concerns: Since each of the multi-resolution Photo-CD images is from 4 to 6 MB in size, the entire database of 500,000 images and their associated text data will require in excess of 2.5 terabytes of storage. The desire for fast access for browsing images must be balanced with the need to minimize the cost to store the images. Therefore a multiple level storage plan is needed including a tertiary memory device to store images.

Simplicity of Use, Simplicity of Design: The browser needs to be simple enough for non-technical staff to use and it should also protect against accidental modification of data already contained in the database. The user interface should to be similar in structure to the existing system and as intuitive and self-documenting as possible. The design of the system should also be simple; it should use existing functions and established models both for ease of implementation as well as to simplify future modifications.

Flexible Query Methods: The retrieval system must be flexible enough to handle complex queries that combine several of the attributes of the image. To process a query such as "Find a picture of a sunset taken near San Francisco during 1994", the retrieval system must be able to search on multiple data types such as geographical location ("San Francisco"), time ("after 12/31/93 and before 1/1/ 95"), and content ("a sunset").

Querying by Image Content: Because of the size of the DWR collection, queries that are too general might return a result set of unmanageable size, as in the case of the "scenic picture of Lake Tahoe" example in Chapter 1. Therefore, we must take steps to increase the precision of retrievals, thereby reducing the set of images that a user must browse to find the images of interest. More importantly, since the primary data type of this database is the image, standard querying by stored descriptive data will not always yield satisfactory results. Therefore, the system must integrate stored textual information with image content information. Ideally, the user should be able to register a conceptual description like "sunset" with the retrieval system, which should respond by initiating the appropriate functions to analyze the content of the images stored in the database that meet the user's expectation of what constitutes a "sunset". Concepts should embody textual metadata about the image as well as image feature information. In Chapter 4 we will describe our implementation of "concept queries" like this.

3. Current Research

The problem of how to store large numbers of digitized images and retrieve pictures from such a collection is an active area of research that overlaps many

fields within computer science including graphics and image processing, information retrieval, and databases. The Chabot project takes a database approach to the problem, and we use a DBMS that allows us to incorporate image analysis and information retrieval tools into our system. Before we describe this process, we first discuss some of the current research.

Much work is underway in the area of image feature indexing, especially color indexing [10,11]. With this approach, features of the images such as dominant color, shapes, lines, and texture, are pre-computed and stored for later analysis. An index is created to provide quick access to the feature information. Runtime computations determine the degree of similarity between a sample image and other images stored in the collection. Usually a ranked list of matches is returned from queries. Given the unpredictable nature of the DWR queries, however, it is not likely that such an index can be made for our database that will be relevant for more than a small number of requests. Moreover, indexing presupposes similarity matching for retrievals ("Find other pictures that look like this one") and pre-identification of interesting features. DWR would like to "fish" from the database rather than present a sample image for matching, and image-by-image review to delineate content features is not feasible because of the large number of images in the collection.

The Photobook project [5] at the M.I.T. Media Lab seeks to circumvent the issue of pre-determined search criteria by storing enough information about each image so that run-time computations are possible. Images are classified at load time as having "face", "shape", or "texture" properties; some techniques have been developed to automate this process, such as foreground extraction. Once classified, the image is compressed by encoding salient semantic information according to its category, and these smaller encoded versions are used at query time both to reconstruct the image and also to compute any additional search criteria such as a color histogram. Photobook has been used to match faces in a collection of photographic portraits and to identify hand tools in a small collection of images. However, this project does not use an underlying relational database; moreover, static pre-analysis is not practical for our application.

One of the most closely related projects to Chabot is the QBIC project [1,3] at IBM Almaden, which uses image analysis to process queries for an image database. This project uses color, shape, and texture to match images in the database to a user's query, which has the form "find more pictures like this one". The user can make a sketch of a shape, select colors and color distributions from a color wheel, or select textures from a predetermined range. The system returns a ranked list of best matches to the user's query. However, the DWR application requires a stronger emphasis on the ability of the underlying relational database to handle

diverse types of textual metadata; support for integration of image features with text and other data types is essential. Most queries to the DWR collection rely on the textual metadata alone, or a combination of text and image features, but rarely on the image content alone. Moreover, queries that do include content-based criteria should not require the user to furnish a sample image for finding similar pictures in the database, so a courser granularity of image analysis is needed. For example, we would like to allow the user to "find pictures of a sunset at Lake Tahoe" by using textual information ("Lake Tahoe") and by defining the concept "sunset" as a range of predominant colors in the image.

In the DBMS community, image database research focuses on storage methods for large objects: spatial data such as geographical maps may be stored in structures such as R-trees [2]. Current work includes Digital Equipment Corporation's multimedia object support for Rdb [7], DEC's relational database. Multimedia object files are physically stored in segments on a WORM device and within the database as binary large objects (BLOBs), guaranteeing DBMS functionality such as transactions and concurrency for these objects. However, this project is not investigating ways to incorporate content-based queries to retrieve images.

The particular needs of our application require the services of a powerful relational database model as the foremost consideration because most retrievals from this collection are made using the stored textual data for each image, which encompasses diverse data types. The features that a relational DBMS provide - query optimization, complex types, a rich query language - become even more important as the size of the collection increases. We also require the flexibility to implement "concept" queries that use image content in conjunction with the text-based queries supported by the DBMS. In these ways Chabot differs noticeably from the systems described above.

4. Description of Chabot

Chabot includes a top-level user interface that handles both queries and updates to the database. Our querying mechanism retrieves images on the basis of stored textual data as well as on more complex relations among the stored data. As a first step towards integrating content analysis into the retrieval system, we have implemented a method for color analysis of the images. In this section we give the implementation details of the Chabot.

POSTGRES

To store the images and textual data, we are using POSTGRES. POSTGRES is particularly attractive for use with a database like Chabot; in addition to the standard relational database features, it provides features not found in traditional relational DBMS's, such as:

Object-oriented properties: Classes can be defined for objects in a POSTGRES database and attributes can be inherited among classes. The "Schema" section below explains this in more detail.

Complex types: POSTGRES provides a flexible assortment of data types and operators that are useful for a database like Chabot such as *time* (absolute and relative), variable-length arrays, and images. In addition, users can define new data types for a database, along with operators that are particular to the type. For example, a type "PhotoCD" can be defined that includes operators to manipulate the image at runtime.

User-defined indices: A secondary index can be defined using access methods specified by the user. The index can be implemented either as a B-tree or as an R-tree. Partial indices that include a qualifying operator can be extended incrementally. For image analysis, an index can be created for all pictures that are predominantly red, for example, using the stored color histograms for each image.

User-defined functions: Functions written in C can be registered with a POSTGRES database. The first time the function is invoked, POSTGRES dynamically loads the function into its address space; repeated execution of the function causes negligible additional overhead since it remains in main memory. For the Chabot database, we wrote a function that analyzes at retrieval time color histograms that have been previously computed and stored in the database.

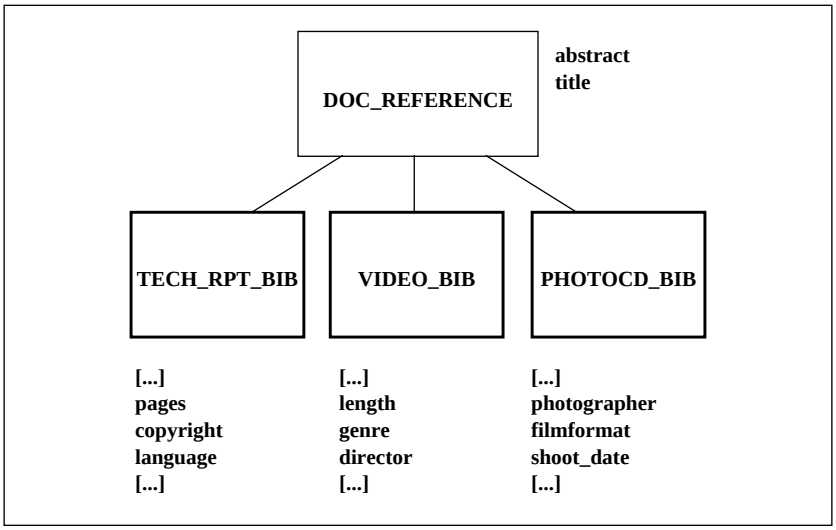
Storage

Each of our images is received in Photo-CD format in five different resolutions, ranging from a "thumbnail" (128x192 pixels) to the highest resolution of 2048x3072 pixels. The size of each image is from 4 to 6 MB. Since DWR's goal is to allow on-line access to both images and data, Chabot must provide reasonably fast browsing of the stored images over a network. A random access medium such as a magnetic disk that is fast enough for remote browsing is too expensive to store the large number of images we are storing; cheaper alternatives such as tape may be so slow that on-line browsing is virtually impossible. Our solution is to use a two-level storage scheme. We use magnetic disk for storing the thumbnail images and text needed for browsing the database and we archive the large multi-resolution image files on a tertiary device, a Metrum VHS-tape jukebox. The

Metrum holds 600 VHS tapes, each tape having a 14.5 GB capacity. With a total capacity of 10.8 TB, the Metrum is more than adequate as a repository for the DWR image library. The average time for the Metrum to find a tape, load it, and locate the required file is about 2 minutes - too slow for browsing a set of images but fast enough for filling a request from a DWR client once the desired image has been identified.

The Schema

The schema for the Chabot project was designed to fit with those of other research projects in progress at Berkeley -- a collection of technical reports and a video library. The image class in our database is called PHOTOCD_BIB, for "Photo-CD Bibliography", which inherits the attributes "title" and "abstract" from the DOC_REFERENCE class, which is shared by the technical report and video object classes. As shown below, the PHOTOCD_BIB class contains "bibliographical" information about the image object, such as the ID number, the name of the photographer, the film format, the date the photo was taken, and so on. A complete list of attributes for the PHOTOCD_BIB class is shown in Table 1 below.



Schema for technical report, video, and photo-cd classes

Most of the attributes for the image class are stored as text strings; there are two fields that have type *abstime*, the "shoot_date" of the photo and the "entry_date" that the information was entered into the database. These allow us to perform time-

relative searches, for example, “Find all shots of Lake Tahoe that were taken after January 1, 1994.”

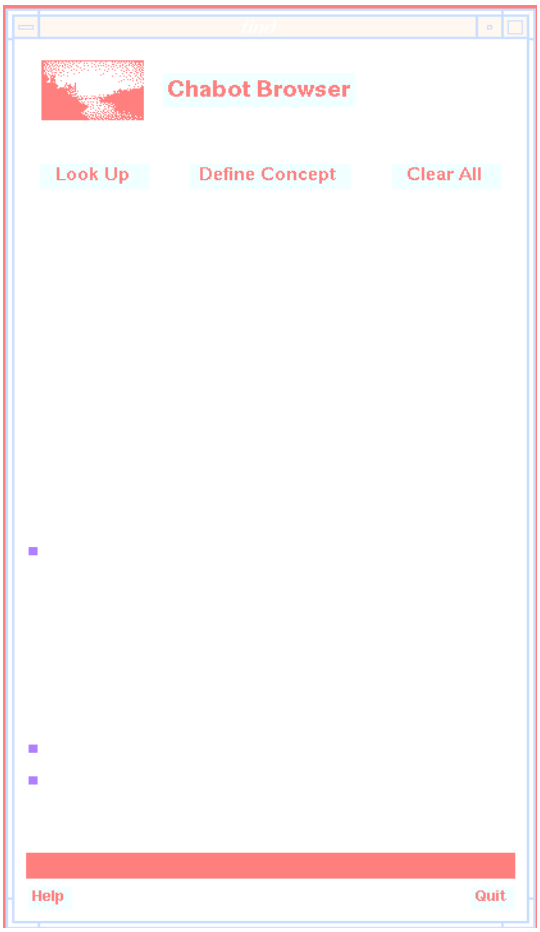
attribute	type	description
abstract	text	abstract (for documents)
title	text	title (of document)
comments	text	comments
disknum	text	Photo-CD number
imgnum	integer	image number on CD
id	text	DWR ID number
doc_type	text	nature, art, legal, etc.
copyright	text	copyright information
indexer	text	person creating db entry
organization	text	who commissioned photo
category	text	DWR category - “SWP”, etc.
subject	text	DWR subject - “The Delta”
location	text	one of 9 California regions
description	text	a description of the image
job_req_num	text	DWR job request ID
photographer	text	photographer
filmformat	text	“35 mm slide”
perspective	char16	aerial - ground - close-up
color	char	C (color) B (black & white)
orientation	char	H (horizontal) V (vertical)
histogram	text	color histogram
entry_date	abstime	date of db entry
shoot_date	abstime	date photo was taken
oid	oid	POSTGRES object ID

Table 1: Attributes for the PHOTOCD_BIB class

The User Interface

We have implemented a graphical point-and-click Motif-like interface for Chabot written in Tcl/Tk [4]. The interface is designed to prevent accidental corruption of data while browsing the database; the main screen gives the user three options: *find*, *edit*, and *load*. The database can be modified only via the *edit* and *load* screens and user authorization for these screens is required. The *find* screen is for running queries and for browsing the database.

An example of the current implementation for the *find* window appears below. The user can build queries by clicking on the appropriate buttons and typing text into the entry fields next to the search criteria. Pull-down menus, indicated by a downward pointing arrow next to the entry field, are provided for some search criteria, those that have limited options such as *Region*, *Film Type*, *Category*, *Colors*, and *Concept*. The user selects one or more of these fields and then clicks on the button labelled “Look Up” to initiate the query, and a Postquel query is constructed and issued to the database. Postquel is a query language written for



the *find* window

POSTGRES that is very similar to SQL. For example, using the search criteria from the *find* screen shown, the Postquel query would be:

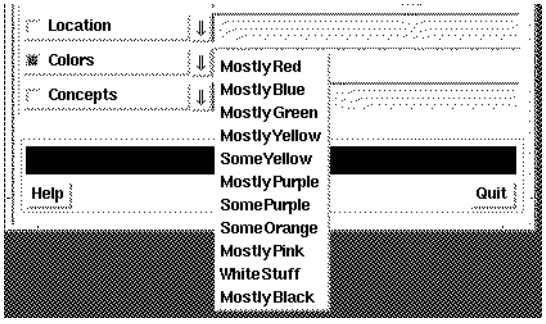
```
retrieve (q.all) from q in PHOTOCD_BIB where
      q.shoot_date>"Jan 1 1994" and
      q.location~"2" and
      MeetsCriteria("SomeOrange",q.histogram)
```

This query returns all images in the database that were taken after January 1, 1994 in the San Francisco Bay region, and that have some of the color orange in them. Figure 2 shows some of the images that were returned from the above query, such as orange poppies, pictures of fire, a sunset, and underwater marine life.

When a query is processed, the result set of data is displayed in a pop-up "Query Result" window; the user can then print the data, save it to a file, or click on a "Show Image" button, to display the selected images; up to 20 images can be displayed at once. In the example above, eight of the images were selected from the "Query Result" window and the resulting display is shown in Figure 2.

MeetsCriteria

To implement concept queries, we use two capabilities that POSTGRES provides: storage of pre-computed content information about each image (a color histogram) as one of the attributes in the database, and the ability to define functions that can be called at run-time as part of the regular querying mechanism to analyze this stored information. The function "MeetsCriteria" is the underlying mechanism that is used to perform concept queries. The example above shows how MeetsCriteria is used within a query. It takes two arguments: a color criterion such as "Some Orange" and a color histogram. The user selects a color criterion from a menu on the *find* screen, and a call to MeetsCriteria is incorporated into the query using the selected color. Colors implemented so far are shown in the example below:



For the histograms, we have experimented with quantizing the colors in our images to a very small number so that run-time analysis is speeded up. Our tests were conducted on histograms containing 20 elements that were computed using Floyd-Steinberg quantization. We have found that quantizing to as few as 20 colors allows us to find the predominant colors in a picture for the "Mostly" queries while still providing a glimpse of the minor colors for the "Some" queries. For example, a picture of a field of purple flowers having tiny yellow centers qualifies as "Mostly Purple", but we can also retrieve this picture using the search criterion "Some Yellow".

The POSTGRES query executor calls the function MeetsCriteria for each histogram in the database, checking to see whether it meets the criterion that is presented.

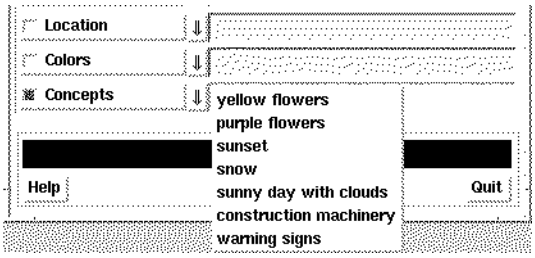
POSTGRES's query optimization facility is used to minimize the search set of histograms. The function returns *true* if the histogram meets the criterion, *false* if it does not. Although the method for finding histograms that meet the criterion varies according to which color is being checked, in general the algorithm employs two metrics: *compliance* and *count*.

Compliance: Each of the colors in the histogram is checked to see whether it complies with the values that have been pre-defined for the requested color. For example, in the RGB model the color white is represented by (255,255,255) for (red, green, blue); a color whose RGB values are all above 241 qualifies as "white" in our approach.

Count: As we check each color in the histogram for compliance, we keep two counts: the number of colors in the current histogram that have matched the criterion, and the number of pixels contained in the matching colors as a function of total pixels in the image. The former count is used when we are looking for "Some" colors; in the "Some Yellow" example, we get a *true* result if just one or two of the twenty colors in the histogram qualify as "yellow". We use the total pixel count for the "Mostly" matches: more than 50% of the total pixels of an image must be "red" in order for the image to meet the "Mostly Red" criterion.

Concept Queries

In addition to using color directly for content analysis, users can compose higher level content-based queries to the database that embody contextual information such as "sunset" and "snow". These queries are called concept queries. The *Concepts* selection on the *find* screen of the interface lists the concept queries that are available, each of which has been previously defined by the user:



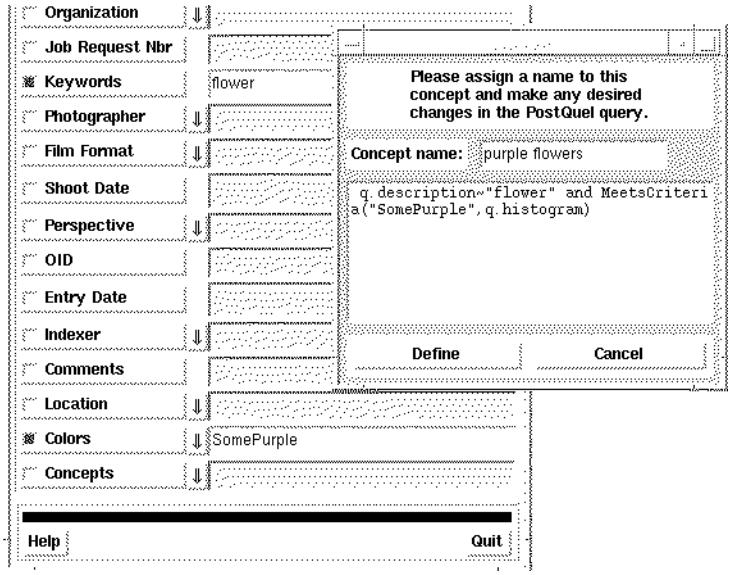
Selecting a concept from the pull-down menu generates a Postquel query that incorporates a combination of search criteria that satisfy the concept. Typically

MeetsCriteria is used in these queries for color analysis in combination with some other textual criteria. For example, when “sunset” is chosen from the *Concepts* menu, the following query is sent to the database:

```
retrieve (q.all) from q in PHOTOCD_BIB where
      q.description ~ “sunset” or
      MeetsCriteria(“MostlyRed”,q.histogram) or
      MeetsCriteria(“MostlyOrange”,q.histogram)
```

In this case, the user has defined the concept “sunset” as including images that have the stored keyword “sunset” associated with them, or images that have red or orange as their predominant color. Concept queries can be used in conjunction with other criteria. The query “Find pictures of Lake Tahoe at sunset” would be generated by choosing “sunset” from the *Concept* menu and setting the *Location* to “Lake Tahoe”.

Users can define new concepts and add them to the *Concepts* menu by first selecting criteria on the *find* screen that should be included in the new concept. Clicking on the “Define Concept” button on the find screen brings up a dialog box prompting the user for the name of the new concept, as illustrated below. The



Postquel query can be edited, after which the user presses the “Define” button to register the new concept. The query is written to a file in the user’s home directory, so that the new concept is immediately available, and future invocations of the browser will include it as well. The editing capability can also be used to add postquel constructs that may not be otherwise available, such as disjunctive

conjunctions. The user can edit the concept file, make copies of the file available to other users, and incorporate others' concepts in the file.

Testing

To test our content analysis, we measured the *recall* and *precision* [8] of some of the concept queries that are shown in the User Interface section. *Recall* is the proportion of relevant materials retrieved, while *precision* quantifies the proportion of retrieved materials that are relevant to the search. For each concept query, we identified by hand all the images in the collection that we thought should be included in the result set. We then tried various implementations of the concept using different combinations of content-based and stored textual data. We measured recall and precision for each implementation.

Table 2 shows the results from one of the test queries that is representative of our findings, the concept "yellow flowers". For this concept, we first identified 22 pictures in the collection that were relevant; we then implemented the "yellow flowers" function in seven different ways using different combinations of search criteria. As shown below, queries 1-3 used keyword search only, queries 4 and 5 used only content-based information, and queries 6 and 7 used a combination of keyword and content-based data.

#	keywords	color content	retrieved	relevant	recall	precision
1	"flower"	-	55	13	59.1	23.6
2	"yellow"	-	11	5	22.7	45.4
3	"flower" and "yellow"	-	5	4	18.1	80.0
4	-	SomeYellow (2)	235	16	72.7	6.8
5	-	SomeYellow(1)	377	22	100	5.8
6	"flower"	and SomeYellow (2)	7	7	31.8	100
7	"flower"	and SomeYellow(1)	15	14	63.6	93.3

Table 2: Query "Find yellow flowers" (relevant images = 22)

In this test, two different methods for finding yellow were tried. SomeYellow (2) means there were at least two yellow colors in a 20-element histogram.

SomeYellow (1) means that only one yellow color is needed for the picture to be counted as having “some yellow”. As shown for query 5, pictures can be retrieved with 100% recall if the color definition is broad enough, but the precision is too low: the 377 images retrieved from query 5 would require the user to browse nineteen screens of thumbnails (each screen displays 20 images) to find the pictures of yellow flowers. Using the coarse definition for yellow in conjunction with the keyword “flower” gives the best result: query 7 has a recall of 63.6% with a very high precision of 93%. Figure 3 shows the fifteen images that were retrieved from this query; only the image in the upper left corner of the group - a plant with pink stems and leaves but with only a small amount of yellow in its petals - was not considered relevant. Figure 4 shows the five images retrieved from query 3, where the keywords “flower” and “yellow” were used. The second picture in this group was not considered relevant.

Other examples of concept queries we tested are shown in Figures 5-8. The query “sunset on a lake” returned nineteen images, two of which were not relevant, when the keyword “lake” was used in conjunction with the concept “sunset” (Figure 5). When keywords alone - “sunset” and “lake” are used for this query, only nine images are retrieved (Figure 6). In the “find purple flowers” example (Figure 7), the keyword “flower” was used along with the content criteria “Some Purple”. Only the third picture in that group could be retrieved using keywords alone. In the “Antelope Lake on a sunny day with clouds” example (Figure 8), we show some of the 53 images that were retrieved by searching for “Antelope Lake” in the *Subject* field along with “White Stuff” (clouds) as a color criterion. In this experiment, most of the return set was not relevant. As shown in the Figure 8 sample, we did find clouds but we also got back white water and patches of light-colored sky. Only 10 of the total 53 images retrieved from this trial were relevant, out of 13 total relevant images. However, using textual information alone yielded worse results. Searching for the keyword “cloud” with the subject “Antelope Lake” returned just one image, which was relevant. And searching only on the subject field retrieved 120 pictures for a recall rate of only 8.3%, though all 13 of the relevant pictures were in the set.

In some of our tests, a good deal of experimentation was necessary to find the right combination of color content and keywords. For the “sunset on a lake” concept we made several passes through the database testing various colors and keywords before identifying all the necessary criteria that would result in the best recall and precision rates. Thus, the success of the concepts that users define will depend to some degree on their familiarity with the images in the collection. On the other hand, concepts like “yellow flowers” and “purple flowers” are more straightforward and are more easily implemented, especially if care is taken to

include some textual information in the concept along with the content-based criteria.

In summary, we found that retrieving images on keywords alone or on content alone produced unsatisfactory results. For example, recall and precision are in inverse proportion: when we retrieve a high percentage of the relevant images, as retrieving all “Mostly Red” images in order to find sunsets, we also retrieve many more images that are not sunsets. But if we more closely restrict the search criteria using carefully chosen keywords so that precision increases, fewer of the relevant images are retrieved. For our application, the best results are achieved when both content and some other search criteria were used, and this is the method we use to implement concept queries.

5. Conclusions and Future Work

We set out to integrate a relational database retrieval system with content analysis techniques that would give our querying system a better method for handling images. We have found that even the simple color analysis method we employ, if used in conjunction with other search criteria, improves our ability to retrieve images efficiently. We concluded that the best result is obtained when text-based search criteria are combined with content-based criteria and when a coarse granularity is used for content analysis. Our concept queries take advantage of this combination.

We are continuing to add images to our collection and we would like to re-run our tests on the collection after it has doubled or tripled in size. We expect that our color analysis technique will scale, but we are interested in discovering to what degree we have tuned our color definitions to the current body of images.

Implementation is underway to include techniques to improve our color analysis. We plan to divide each image into segments and compute and store a histogram for each of these areas. Since most of our images are outdoor shots, we would then be able to distinguish between colors in the lower (ocean, ground) and upper (sky) halves of the picture. Storing a histogram of the center diamond of the picture is another consideration; many of the photos were taken by a professional photographer, the “interesting” objects can often be found in the center of the picture. We would also like to experiment with adjusting our quantization factor according to the color distribution in individual images; images having a large number of different colors would be allotted a histogram with a larger than average number of elements.

We would also like to experiment with other content analysis techniques besides color, such as texture, shape, and line. For example, we would like to handle concept queries such as “Find a picture of Lake Anza with people swimming in it” using texture information, or “Find a picture of Chabot Reservoir when the water is low” using edge detection. POSTGRES provides an easy way to introduce new functions into the querying process by using its user-defined function facility; we plan to work with image analysis experts in developing new content analysis algorithms that can be registered with the database.

Since so many of our retrievals are based on the stored textual data rather than on the images, we would like to include some information retrieval techniques such as the use of a thesaurus and dictionary, and to investigate imposing restrictions on the size of the return set, which we expect will become increasingly important as we add more images to our collection. Although ranking the elements of the return set is usually associated with similarity matching, which is not used for Chabot’s retrieval system, we are interested in how ranking might be used to reduce the return set size as the collection becomes very large.

We plan to integrate the Chabot schema with those of the geographical and environmental datasets of other research projects at UC Berkeley such as satellite imagery, aerial photography, and environmental reports. One planned enhancement is to generate longitude and latitude coordinates for the DWR images using *GIPSY* [12] a system for extracting these coordinates from textual place names. Using this technique, a location name like “El Cerrito” that is attached to one of the DWR images can be associated with spatial data that contains longitudinal coordinates.

References

[1] Christos Faloutsos, Myron Flickner, Wayne Niblack, Dragutin Petkovic, Will Equitz, Ron Barber, "Efficient and Effective Querying by Image Content", IBM Research Report RJ 9453, August 3, 1993.

[2] Antonin Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching", *Proceedings of the 1984 ACM SIGMOD Conference on Management of Data*, Boston, Mass. June 1984.

[3] Wayne Niblack, Ron Barber, Will Equitz, Myron Flickner, Eduardo Glasman, Dragutin Petkovic, Peter Yanker, Christos Faloutsos, "The QBIC Project: Querying Images by Content Using Color, Texture, and Shape", IBM Research Report, RJ 9203, February 1, 1993.

[4] John K. Ousterhout, "Tcl and the Tk Toolkit", Addison-Wesley Publishing Company, 1994.

[5] Alex Pentland, Rosalind Picard, and Stan Sclaroff, "Photobook: Tools for Content-Based Manipulation of Image Databases." SPIE PAPER 2185-05 *Storage and Retrieval of Image and Video Databases II*, San Jose, CA. February 6-10, 1994.

[6] The POSTGRES Group, "The POSTGRES Reference Manual", Computer Science Division, University of California, Berkeley, January 1993.

[7] Mark F. Riley, James J. Feenan, Jr., John L. Janosik, Jr., T.K. Rengarajan, "The Design of Multimedia Object Support in DEC Rdb", *Digital Technical Journal*, Vol.5, No.2, Spring 1993.

[8] Gerard Salton, "Automatic Text Processing", Addison-Wesley Publishing Company, 1989.

[9] Michael Stonebraker, et al., "The Implementation of POSTGRES," *IEEE Transactions on Knowledge and Data Engineering*, March 1990.

[10] Markus A. Stricker and Markus Orengo, "Similarity of Color Images", *SPIE Proceedings* Vol. 2420, 1995.

[11] Michael J. Swain, "Interactive Indexing into Image Databases", *IS&T/SPIE International Symposium on Electronic Imaging: Storage and Retrieval for Image and Video Databases*, February 1993.

[12] Allison G. Woodruff and Christian Plaunt, "GIPSY: Georeferenced Information Processing SYstem", UC Berkeley Technical report UCB:S2K-94-41, March 1994.