Chabot: Retrieval from a Relational Database of Images

Virginia E. Ogle
Michael Stonebraker
University of California at Berkeley

Selecting from a large, expanding collection of images requires carefully chosen search criteria. Here's an approach that integrates a relational database retrieval system with a color analysis technique.

The Chabot project was initiated at UC Berkeley to study storage and retrieval of a vast collection of digitized images. These images are from the State of California Department of Water Resources (DWR), which 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 (SWP) facilities but also many images of California's natural resources (see Figure 1 for examples).

Since DWR made its collection available to the public, it has devoted increasing resources to filling requests for prints and slides. The agency receives 100 to 150 requests per month from various sources, including other government agencies, regional magazine or encyclopedia publishers, university libraries, wildlife organizations, and individuals. Requests vary from those that cite a specific ID number to general requests for scenic pictures of California lakes and waterways. DWR keeps the most frequently requested slides in lighted display boxes for browsing; the rest are housed in archival containers and slide drawers.

To facilitate retrieval, DWR began a project last year to digitize all of its images using Photo CD technology; so far, about 15,000 prints and slides have been scanned to Photo CD. Several years ago the agency began to enter descriptive data about each image into a single-user PC database. To process a request, the staff uses keyword lookup on the text descriptions stored in the database to find an ID number for the requested image. This number is then used to locate the container or drawer where the print or slide is stored.

However, DWR is facing some problems. First, although the staff is trying to annotate each image with as much descriptive information as possible, keyword indexing for an image collection has significant limitations. Nonspecific requests, such as "Find a scenic photo of Lake Tahoe," may entail looking through an unmanageable set of Lake Tahoe images to find the desired prints. Misspelled keywords, such as "azalia" for "azalea," thwart successful retrieval, even when close matches can be culled from a dictionary. And dictionaries offer no help for inaccurate descriptions: A photo of a bright red anemone in full bloom with the stored description "close-up of a pansy" will never be retrieved via any spelling of the keyword "anemone." A thesaurus might compensate for some incorrect descriptions, but not incomplete ones. 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.

As a result, most of DWR's retrievals rely on a staff member's being familiar enough with the collection to know where to find the desired prints. One goal of the Chabot project is to integrate image analysis techniques into the retrieval system so that image requests do not depend solely on stored textual information. As a first step, we have implemented a simple method for color analysis, which we describe in this article. By using both color and tex-

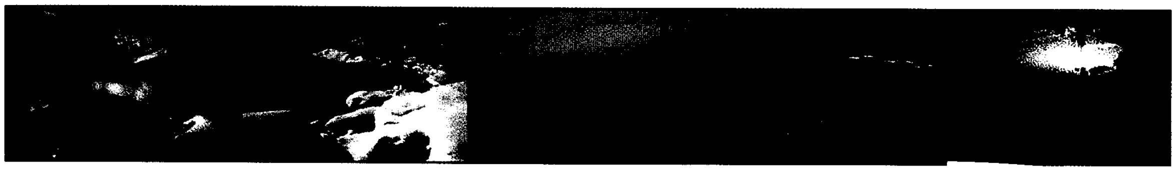


Figure 1. Random images from the State of California Department of Water Resources (DWR).

tual information for the images, we can locate pictures of red flowers (like anemones and azaleas) and Lake Tahoe sunsets.

Another problem is that DWR's database system cannot support complex data types such as time, geographical location, or the images themselves. Nor does it let the user compose queries that combine several attributes of an image. Therefore, the Chabot project also aims to provide DWR with a system that can store and search for diverse data types using the functionality of an advanced relational database management system (DBMS) that has features such as a high-level query language, query optimization, and flexible indexing. Postgres,[1,2] an object-relational DBMS developed at the University of California at Berkeley, allows for user-defined functions and types and thus serves this purpose. We use Postgres to perform runtime image analysis during the querying process.

DWR wants to load and edit its database remotely and enable browsing of its images by off-site users interested in ordering prints or slides. The current database does not meet the agency's needs for on-line, multiuser 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, and
- image retrieval by content.

SYSTEM MOTIVATION AND GOALS

Chabot's design was influenced by DWR's existing system of metadata storage, the types of requests it receives, and the query and update methods currently used.

Integration of data types

Each image is accompanied by extensive 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

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

DWR needs a DBMS that can support various complex data types, including text, numerical data, relative and absolute time, and geographical location. It must be capable of retrieving any combination of the complex data types associated with the images as well as the content of the images themselves.

Scalability and storage concerns

Since each multiresolution Photo CD image is 4 to 6 Mbytes, the entire database of 500,000 images and associated textual data will require more than 2.5 terabytes of storage. The desire for fast access to browse images must be balanced with storage cost considerations. Therefore, we need a multiple-level storage plan with a tertiary memory device for storing images.

Simplicity of use and design

The browser must be simple enough for nontechnical staff use yet must protect against accidental modification of data already in the database. The user interface should be similar in structure to the existing system and as intuitive and self-documenting as possible. The system design should use existing functions and established models to simplify implementation and future modifications.

Flexible query methods

The retrieval system must be flexible enough to handle complex queries that combine several image attributes. To process a query such as "Find a picture of a sunset taken near San Francisco in 1994," the retrieval system must be able to search on the basis of 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 DWR collection's scope, queries that are too general might return a result set of unmanageable size, as in the "scenic picture of Lake Tahoe" example. Hence, we must increase the precision of retrievals, thereby reducing the set of images through which a user must browse. More importantly, since this database's primary data type 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 could register a conceptual description like "sunset" with the retrieval system, and it would initiate the appropriate functions to analyze the content of the images stored in the database that meet the user's expectation of a "sunset." Concepts should embody textual metadata on the image as well as image feature information.

Current research

Determining how to store many digitized images and retrieve pictures from those image collections is an active area of research for many computer science fields, including graphics and image processing, information retrieval, and databases. The Chabot project takes a database approach to the problem. We use a database management system (DBMS) that allows us to include image analysis and information-retrieval tools in the query process.

Extensive work is under way in the area of image feature indexing, especially color indexing,[1,2] where image features such as dominant color, shapes, lines, and texture are precomputed 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. However, indexing presupposes similarity matching for retrievals (for example, "Find other pictures that look like this one") and pre-identification of interesting features. If the goal is to fish from the database rather than present a sample image for matching, indexing might not be useful. Moreover, image-by-image review to delineate content features is not feasible when the collection contains a large number of images.

The Photobook project[3] at the MIT Media Lab seeks to circumvent the issue of predetermined search criteria by storing enough information about each image to make runtime computations possible. Images are classified at load time as having face, shape, or texture properties; techniques have been developed to automate this process—for example, foreground extraction. Once classified, the image is compressed by encoding salient semantic information according to category. These smaller encoded versions are used at query time to reconstruct the image and to compute any additional search criteria such as a color histogram. Photobook has been used to match faces and to identify hand tools in a small collection of images.

One of the most closely related projects to Chabot is the QBIC (Query by Image Content) project[4,5] 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 query of the form "Find more pictures like this one." The user can sketch 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 query. However, some applications, such as the State of California Department of Water Resources (DWR) collection, may require the underlying relational database to handle diverse types of textual metadata and support integration of image features with text and other data types.

In the DBMS community, image database research focuses on storage methods for large objects. Spatial data such as geographical maps can be stored in structures such as *R*-trees.[6] Current work includes Digital Equipment Corporation's multimedia object support for its relational database, Rdb.[7] Multimedia object files are physically stored in segments on a WORM (write-once, read-many) device and in the database as binary large objects. This guarantees DBMS functionality such as transactions and concurrency for these objects.

The Chabot application's particular needs make a powerful relational database model a top priority, because most retrievals involve each image's stored textual data, which encompasses diverse data types. The features that a relational DBMS can provide—query optimization, complex types, and a rich query language—become even more important as collection size increases. Moreover, the flexibility to implement concept queries that use image content in conjunction with text-based queries is essential.

References

1. M.A. Stricker and M. Orengo, "Similarity of Color Images," *SPIE Proc.*, Vol. 2,420, 1995.
2. M.J. Swain, "Interactive Indexing into Image Databases," *SPIE Proc.*, Vol. 1,908, 1993, pp. 95-103.
3. A. Pentland, R. Picard, and S. Sclaroff, "Photobook: Tools for Content-Based Manipulation of Image Databases," *SPIE Proc.*, Vol. 2,185, 1994, pp. 34-47.
4. C. Faloutsos et al., "Efficient and Effective Querying by Image Content," Tech. Report RJ 9453, IBM Research, San Jose, Calif., 1993.
5. W. Niblack et al., "The QBIC Project: Querying Images by Content Using Color, Texture, and Shape," Tech. Report RJ 9203, IBM Research, San Jose, Calif., 1993.
6. A. Guttman, "*R*-Trees: A Dynamic Index Structure for Spatial Searching," *Proc. 1984 ACM SIGMOD Conf. Management of Data*, ACM, New York, 1984.
7. M.F. Riley et al., "The Design of Multimedia Object Support in DEC Rdb," *Digital Technical J.*, Vol. 5, No. 2, Spring 1993.

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 and complex relations among that data. As a first step toward integrating content analysis into the retrieval system, we have implemented a method for image color analysis.

Postgres

To store the images and textual data, we use Postgres, which is particularly attractive for a database like Chabot. Besides the standard relational database features, it provides features not found in traditional relational DBMSs:

- *Object-oriented properties*. Classes can be defined for objects in a Postgres database, and attributes can be inherited among classes.
- *Complex types*. Postgres provides a flexible assortment of data types and operators useful for a database like Chabot—for example, time (absolute and relative), variable-length arrays, and images. In addition, users can define new data types for a database, along with

Table 1. Attributes for the photocd_bib class.

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, and so on
copyright	Text	Copyright information
indexer	Text	Person creating db entry
organization	Text	Person who commissioned photo
category	Text	DWR category — SWP, and so on
subject	Text	DWR subject — "The Delta"
location	Text	One of 9 California regions
description	Text	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 and 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

operators tailored to those types. For example, a user can define a type "PhotoCD" that includes operators to manipulate the image at runtime.

- *User-defined indices.* A secondary index can be defined via user-specified access methods. The index can be implemented as either a *B*-tree or 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, via 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 that function into its address space. Since the function remains in main memory, additional overhead for repeated execution is negligible. For the Chabot database, we have written a function to analyze, at retrieval time, color histograms 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, 128×192 pixels, to the highest resolution, $2{,}048 \times 3{,}072$ pixels. Since DWR wants 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 fast enough for remote browsing, is too expensive to store the large number of images involved. But cheaper alternatives, such as tape, may be so slow that on-line browsing would be virtually impossible.

Our solution is to use a two-level storage scheme. We use a magnetic disk to store the thumbnail images and text needed for browsing the database, and we archive the large, multiresolution image files on a tertiary device, a Metrum VHS-tape jukebox. The Metrum device holds 600 VHS tapes of 14.5-gigabyte capacity each. With a total capacity of 10.8 terabytes, the Metrum is a more than adequate 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 images but fast enough for fulfilling 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 at UC Berkeley: technical reports and videos. The image class in our database is called photocd_bib, for "Photo CD bibliography," which inherits attributes from a doc_reference class shared by the technical-report and video object classes. Table 1 lists the photocd_bib class's attributes.

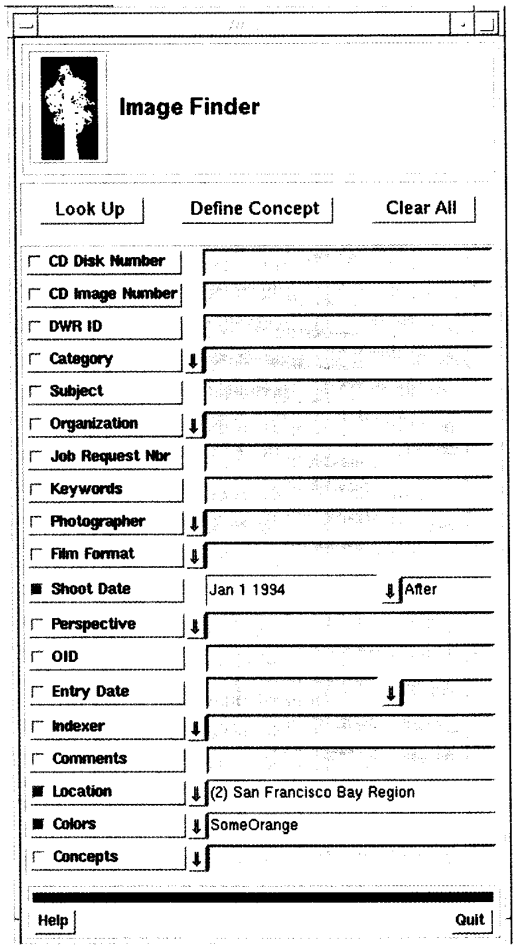
Most attributes for the image class are stored as text strings, including the color histogram. However, two fields have type *abstime* (absolute time): the photo's shoot_date and the information's database entry_date. These fields let us perform time-relative searches—for example, "Find all shots of Lake Tahoe that were taken after January 1, 1994."

The user interface

We have implemented a graphical point-and-click, Motif-like interface for Chabot written in Tcl/Tk.[3] The interface is designed to prevent accidental data corruption during database browsing. 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 used for running queries and browsing the database.

An example of the current implementation for the *find* window appears in Figure 2 on the next page. 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 search criteria that have limited options—for example, Region, Film Type, Category, Colors, and Concept. The user selects one or more of these fields and clicks on the "Look Up" button to initiate the query. A Postquel query is then constructed and issued to the database. Postquel is a query language similar to SQL (Structured Query Language) that is written for Postgres. As an example, using the search criteria from the *find* screen shown in Figure 2, 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)
```



☐ CD Disk Number		
☐ CD Image Number		
☐ DWR ID		
☐ Category		
☐ Subject		
☐ Organization		
☐ Job Request Nbr		
☐ Keywords		
☐ Photographer		
☐ Film Format		
■ Shoot Date	Jan 1 1994	After
☐ Perspective		
☐ OID		
☐ Entry Date		
☐ Indexer		
☐ Comments		
■ Location	(2) San Francisco Bay Region	
■ Colors	SomeOrange	
☐ Concepts		

Figure 2. The Chabot *find* screen for browsing the database.

This query returns all images in the database that were taken after January 1, 1994, in the San Francisco Bay area and that have the color orange in them. Figure 3 shows some of the images that met this query's criteria—orange poppies, fire, and interior shots.

When a query is processed, the resulting data is displayed in a pop-up "Query Result" window. The user can 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 images were selected from the "Query Result" window, with the resulting display shown in Figure 3.

MeetsCriteria

Implementing concept queries involves two Postgres capabilities: storing precomputed content information about each image (a color histogram) as a database attribute, and defining functions that can be called at runtime as part of the regular querying mechanism to analyze this stored information. The function MeetsCriteria is the underlying mechanism for performing concept queries. The sample Postquel query presented earlier shows how MeetsCriteria is used in a query. This function 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. Figure 4 shows the colors implemented thus far.

For the histograms, we experimented with quantizing the colors in our images to a very small number to accelerate the runtime analysis. We found that quantizing to as few as 20 colors let us find the predominant colors in a picture for the "Mostly" queries yet still provided 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."

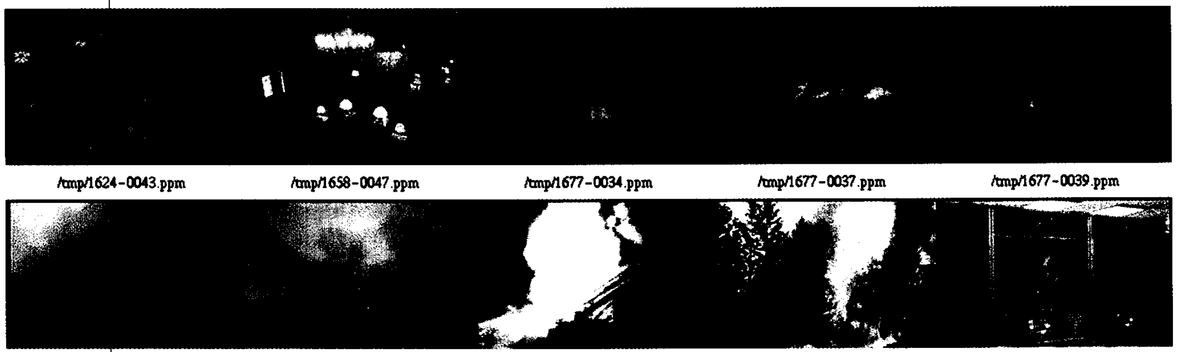


Figure 3. Partial results from "Some Orange" query.

U-M-I
BEST COPY AVAILABLE

The Postgres query executor calls the function MeetsCriteria for each histogram in the database, checking for compliance with the presented criterion. Postgres's query-optimization facility minimizes the histogram search set. 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 the color being checked, the algorithm generally employs two metrics: compliance and count.

Compliance. Each color in the histogram is checked for compliance with the requested color's predefined values. For example, in the RGB model, the color white is represented by 255, 255, and 255 for red, green, and blue, respectively; with our approach, a color whose RGB values are all above 241 qualifies as white.

Count. As we check each color in the histogram for compliance, we count the number of colors in the current histogram that match the criterion. We also count the number of pixels in the matching colors as a function of total image pixels. The former count is used when we are looking for "Some" colors. Thus, in the "Some Yellow" example, we get a true result if only one or two of the 20 colors in the histogram qualify as yellow. We use the total pixel count for the "Mostly" matches. For instance, more than 50 percent of an image's total pixels must be red for the image to meet the "Mostly Red" criterion.

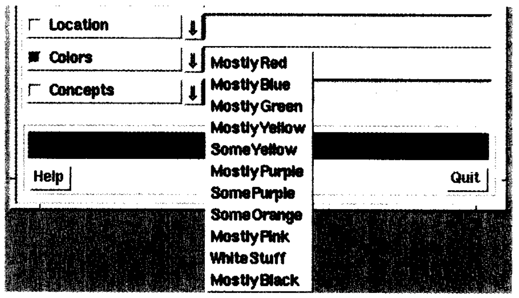


Figure 4. The *find* screen's Color menu for selecting color criterion.

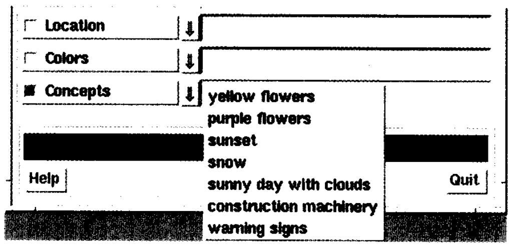


Figure 5. Available concept queries in the *find* screen's Concepts menu.

Concept queries

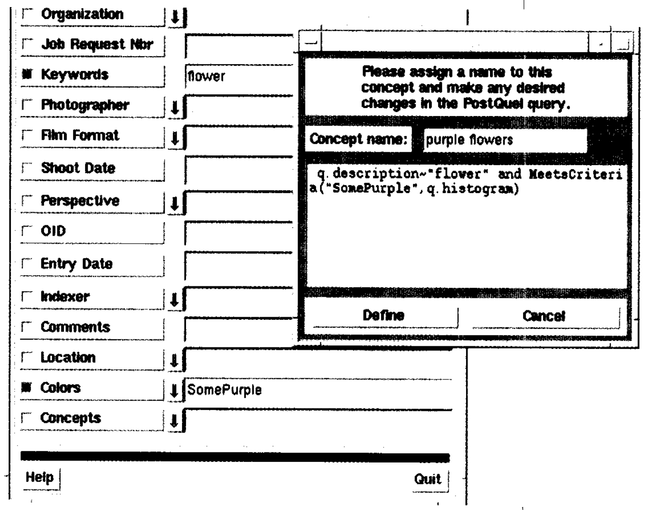
Besides using color directly for content analysis, users can compose higher level content-based queries to the database that embody contextual information such as "sunset" and "snow." These queries are called concept queries. The Concepts selection on the interface's *find* screen lists available concept queries, each one previously defined by the user (see Figure 5).

Selecting a concept from the pull-down menu generates a Postquel query. This query incorporates a combination of the search criteria satisfying the concept. The query typically employs MeetsCriteria for color analysis plus 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("MostlyYellow", q.histogram) or
        MeetsCriteria("MostlyPurple", 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 have red, yellow, or purple as their predominant color. Concept queries can be used in conjunction with other criteria. For example, we can generate the query "Find pictures of Lake Tahoe at sunset" by choosing "sunset" from the Concept menu and setting Location to "Lake Tahoe."

Users can define a new concept and add it to the Concepts menu by selecting from the *find* screen the criteria 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 new concept's name, as illustrated in Figure 6. After editing the Postquel query, the user clicks on the "Define" button to register the new concept. The query is written to a file in the user's home directory. Hence, the new concept becomes immediately available and will remain so for future browser invocations. The editing capability also lets the user add Postquel constructs that might not otherwise be available—for example, disjunctive conjunctions. The user can



Concept name: purple flowers

```
 q.description~"flower" and MeetsCriteri
a("SomePurple",q.histogram)
```

Figure 6. The "Define Concept" dialog box accessed from the *find* screen.

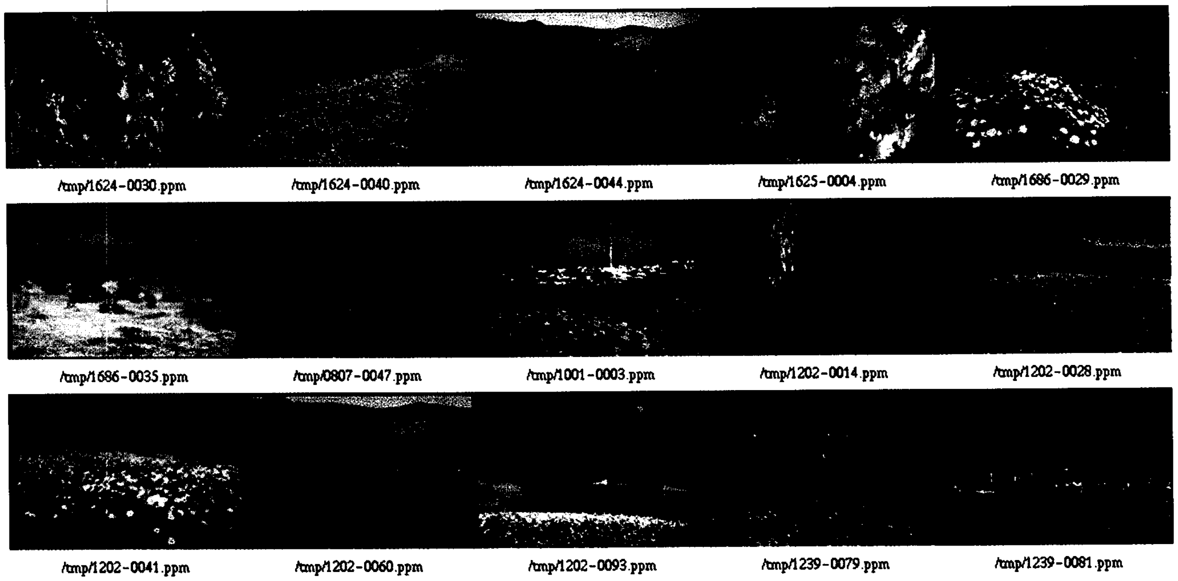


Figure 7. Return set using keyword "flower" and content "Some Yellow."

Figure 8. Return set using keywords "flower" and "yellow."

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[4] of some concept queries. 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 all the images we thought belonged in the result set. We then tried various implementations of the concept, using different combinations of content-based and stored textual data. And we measured recall and precision for each implementation.

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

In this test, two different methods for finding yellow were tried. "SomeYellow (2)" means at least two colors in a 20-element histogram are yellow. "SomeYellow (1)" means only one yellow color is needed for the picture to be counted as having "some yellow." As query 5 in Table 2 shows, pictures can be retrieved with 100 percent recall when the color definition is broad enough. However, the precision is too

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

No.	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	100.0	5.8
6	"flower"	SomeYellow (2)	7	7	31.8	100.0
7	"flower"	SomeYellow (1)	15	14	63.6	93.3

low. To find the pictures of yellow flowers, the 377 images retrieved from query 5 would require the user to browse 19 screens of 20 thumbnail images each. Using the coarser definition for "yellow" in conjunction with the keyword "flower" gives the best result: Query 7 has a recall of 63.6 percent and a high precision of 93.3 percent.

Figure 7 shows the 15 images retrieved from query 7. 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 8 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.

In some of our tests, a fair amount of experimentation was necessary to discover the right combination of color content and keywords needed to compose a successful query. An example is "Find a sunset on a lake." Figure 9 shows the sunsets retrieved from this query using text matching only; Figure 10 shows the results of text matching combined with color analysis. We expected sunsets to be "Mostly Red," but as shown in Figure 10, we also needed to look for "Mostly Purple" and "Mostly Yellow." Thus, the efficacy of some of the concepts we define depends somewhat on our familiarity with the collection's images. On the other hand, concepts like "yellow flowers" are relatively straightforward to implement, especially if some

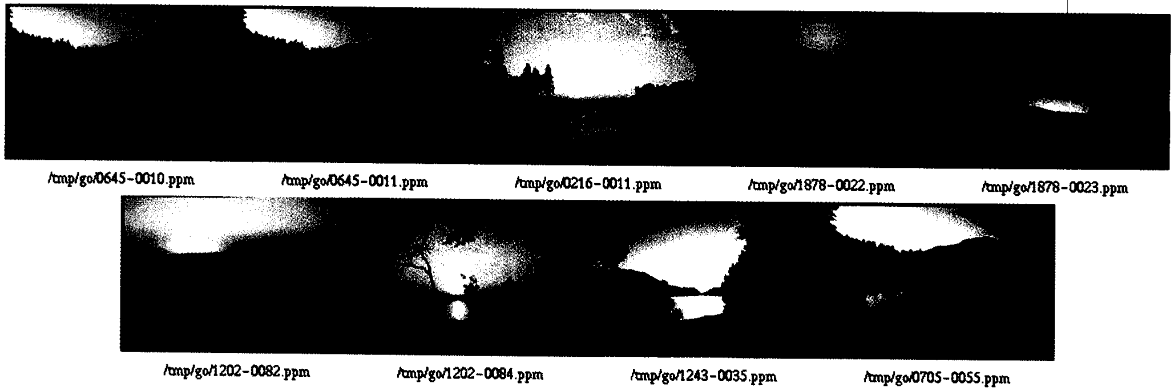


Figure 9. Return set using keywords "lake" and "sunset."

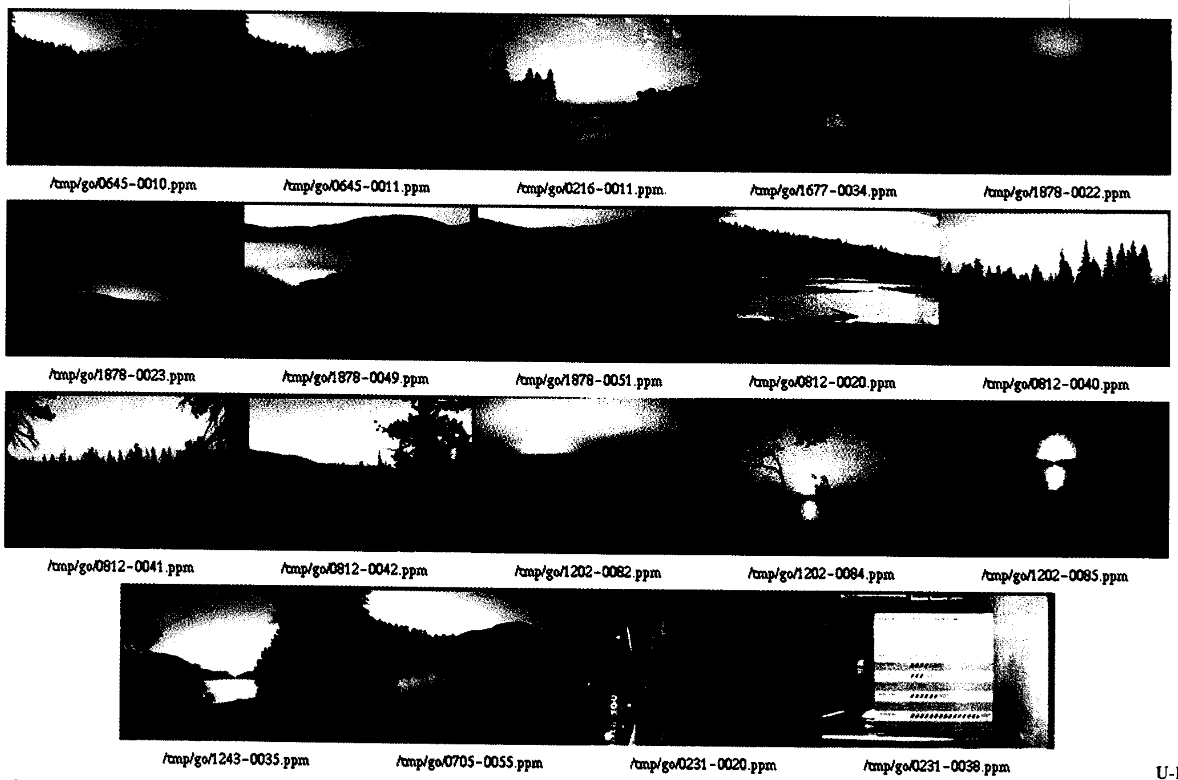


Figure 10. Return set using color analysis and keyword "lake."

textual information is included in the concept along with the content-based criteria.

In summary, retrieving images on the basis of keywords or content alone produces unsatisfactory results. For example, recall and precision are inversely proportional. When we retrieve a high percentage of the relevant images, such as retrieving all "Mostly Red" images to find sunsets, we also retrieve many more images that are not sunsets. But if we restrict the search criteria more closely so that precision increases, fewer relevant images are retrieved. For our application, the best results were achieved when both content and some other search criteria were used, and this is how we implement concept queries.

OUR GOAL WAS TO INTEGRATE A RELATIONAL DATABASE retrieval system with content analysis techniques that would give our querying system a better method for handling images. Our simple color-analysis method, if used in conjunction with other search criteria, improves our ability to retrieve images efficiently. 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 use such a combination.

The best retrieval result is obtained when text-based search criteria are combined with content-based criteria and when a coarse granularity is used for content analysis.

We are continually adding to our current collection of 11,643 images and plan to rerun our tests once the collection has doubled or tripled in size. We expect that our color analysis technique will scale, but we are interested in determining the degree to which we have tuned our color definitions to the current body of images.

We have begun implementing other color-analysis techniques and hope to investigate other content-analysis techniques, such as texture, shape, and line. Postgres provides an easy way to introduce new functions into the querying process through its user-defined function facility. We plan to work with image-analysis experts to develop new content-analysis algorithms that can be registered with the database.

Since so many of our retrievals are based on stored textual data rather than on the images, we plan to include some information-retrieval techniques such as thesaurus and dictionary use. We will also investigate the impact of imposing restrictions on the return set's size, which we expect will become increasingly important as we add more images to our collection. Ranking the return set's elements is usually associated with similarity matching, which is not used in Chabot's retrieval system. Nevertheless, we are interested in how ranking can reduce return set size as the collection becomes very large.

We also plan to integrate the Chabot schema with those of the geographical and environmental data sets of other research projects at UC Berkeley, such as satellite imagery, aerial photography, and environmental reports. One planned enhancement is to spatially locate the DWR images using the Georeferenced Information Processing System (Gipsy),[5] which generates longitude and latitude coordinates from textual place names. ▮

References

1. The Postgres Group, *The Postgres Reference Manual*, Computer Science Division, Univ. of California at Berkeley, Calif., 1993.
2. M. Stonebraker et al., "The Implementation of Postgres," *IEEE Trans. Knowledge and Data Eng.*, Mar. 1990.
3. J.K. Ousterhout, *Tcl and the Tk Toolkit*, Addison-Wesley, Reading, Mass., 1994.
4. G. Salton, *Automatic Text Processing*, Addison-Wesley, Reading, Mass., 1989.
5. A.G. Woodruff and C. Plaunt, "GIPSY: Georeferenced Information Processing System," Tech. Report UCB:S2K-94-41, Univ. of California at Berkeley, Calif., 1994.

Virginia E. Ogle *works for the Electronic Research Lab at UC Berkeley, where she is implementing the next phase of Chabot, called Cypress, for the university's digital library project (see URL http://elib.cs.berkeley.edu). Her research interests include multimedia databases, information retrieval, and user interfaces. Ogle received an MS degree in computer science from the University of California at Berkeley in 1995 and a BA degree in English literature from the University of Alabama.*

Michael Stonebraker *has been a professor of electrical engineering and computer science at the University of California at Berkeley since 1971. He was one of the principal architects of Ingres, Distributed Ingres, and Postgres. His interests include DBMS support for visualization environments and next-generation distributed DBMSs. Stonebraker is a founder of both the Ingres Corporation and Illustra Information Systems. He is a past chair of the ACM Special Interest Group on Management of Data and has been the keynote speaker at several recent conferences. He received the first ACM SIGMOD Innovations Award in 1992 and was named an ACM fellow in 1994.*

Readers can contact the authors at the Computer Science Division, University of California at Berkeley, 390 Soda Hall, Berkeley, CA 94720-1776; e-mail {ginger, mike}@cs.berkeley.edu. The Chabot retrieval system is accessible on the Web at URL http://elib.cs.berkeley.edu/cypress.html.