Extending Relational Databases to Support Content-based Retrieval of Medical Images

Myrian R.B. Araujo, Caetano Traina Jr., Agma Traina, Josiane M. Bueno, Humberto L. Razente
Computer Science Department - University of Sao Paulo at Sao Carlos - Brazil
[mrenata, caetano, agma, josiane, hlr]@icmc.usp.br

Abstract

This paper shows how to support images in a relational database, so it can fulfill the requirements to be used as the storage mechanism of a PACS. This support includes the ability to answer similarity queries based on the image content, providing fast image retrieval based on indexing structures. The main concept allowing this support is the definition of distance functions based on features, which are extracted from the images as they are stored in the database. An extension to the SQL language enables the construction of an interpreter that intercepts the extended commands and translate them to standard SQL, allowing to take advantage of any relational database server. We describe experiments made with a prototype implemented using these concepts, which allowed answering queries up to 20 times faster than using only existing relational servers.

1. Introduction and motivation

When applications deal with sets of images, the usual approach is to store all of them in a unique repository. A common example is the Picture Archiving and Communications System (PACS) based on the DICOM image format. This architecture is used in many hospitals with increasing acceptability [3]. In these systems, the images are tagged in the DICOM format, and stored in a core storage system. Images obtained from different kinds of exams and/or equipments are stored together, classified by tags set individually at each image. This approach is adequate to distribute images, because each image file has the identification of its acquisition, patient, and other pertinent data, so the information is never lost. However, this approach is not suitable to execute image retrieval operations, because each request for a subset of images tagged in a given way requires sequentially reading the full image set, filtering the asked images looking for the tag values at each image.

Medical application software is constructed to process images originating from specific equipment or kind of exams, so requests looking for an image set to be processed by a specific application is a very common operation. Reading large sets of full featured image files is a costly operation, so the PACS extract values from specific tags of each image as they are being stored in the system, into directory tables. When a query requests images based on tag values, the directories are used to select the required images, speeding up the query answering process. However, the choice of what tags to maintain in the directories is cumbersome, making the whole system inflexible, difficult to

---

1 This research has been partially supported by FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo, Brazil - grants 98/09880-1 and 98/03.209-6), and by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico, Brazil - grants 52.1685/98-6, 52.1267/96-0 and 860.068/00-7).
improve or expand, and hard to maintain. Moreover, operations of selecting images using content-based retrieval techniques are orthogonal to operations of selecting images using the tag-based approach, leading to duplication of efforts, thus further reducing efficiency and the flexibility of the system.

Relational databases stores large amount of information in tables, which are composed by attributes of various types, and their use lead to flexible, expansible and highly maintainable information systems. Unfortunately, relational databases do not support images adequately. They only enable storing images as “Binary Large OBjects - BLOBs” data, allowing images to be retrieved identifying the record where they are stored through other textual or numeric attributes that are stored together in the same record.

This paper shows how to improve a relational database to support images, so it can be used as the underlying storage mechanism of a PACS. It also shows that the resulting system presents not only the desired properties of flexibility, expansibility and high maintainability provided by relational systems; but also consistency of image distribution presented by PACS and the DICOM format and supports efficient content-based image retrieval operations.

The rest of this paper is organized as follows: section 2 gives an overview of concepts and previous related works. Section 3 presents the main ideas and describes the generic architecture of a system empowered to support images as a native data type. Concluding, section 4 provides some practical measurements obtained from a system implemented using those concepts.

2. Background

A distinctive characteristic of the relational database system when compared to storage mechanisms of a PACS is the partitioning of images into subsets. Whereas in a PACS there is no other way of storing information related to each image than using the tagging mechanism, in a relational system it is possible to link any kind of information to images. This is achieved storing a set of attributes together with each stored image. Therefore, images acquired by a specific equipment or to fulfill requirements for a given exam can be stored in a table specific for the equipment (or the exam), providing a natural way to select the images targeted by a given medical application software.

When a relational database is the core storage system of a PACS, attributes in the table replicate the values of tags in the images stored in DICOM format. A simple filter executed when the images are stored can extract relevant tags in the image file, copying them to the table attributes. This operation reproduces the directories of current PACS, but with the enhanced flexibility of relational databases to add or drop attributes in tables. Moreover, this approach enables the system to store images in any format, and not specifically DICOM, provided the necessary information can be obtained during the input operation from other sources, such as forms filled by the operator.

Despite easing the integration of clinical and administrative databases [5], this approach improves the efficiency and maintainability of the system. However, it adds little to the facilities provided to the end user - the physicians and health caretakers’ personnel. This is due to the selection engine in both approaches be based on comparisons using only numerical and textual attributes, and not using image contents. Nonetheless, using a relational database as the core storage system of a PACS makes it easier to exploit recent research results on an image content-based retrieval [7] [6].

To retrieve images based on content, it is necessary to provide an image as the reference for the comparisons. When retrieving records based on numeric or textual attributes, the comparisons are made using identity or relational operators, like retrieving patients whose name is 'equal to' “John Doe”, or whose age is 'larger than' 50 years old. When retrieving records based on image content, those operators are useless, as two identical images are almost impossible to occur and there is no ordering between images. There are two similarity comparison operators used to retrieve
Given three objects \( a, b \) and \( c \) in the object's domain, the triangular inequality property holds for the distance function \( d(x, y) \) if \( d(a, b) \leq d(a, c) + d(b, c) \).

The selection of images based on their contents is very time consuming, so the usual approach is to extract features using special image processing algorithms, called extractors. Such algorithms analyze one image and return a set of values, called the feature vector. They can be used to compare pairs of images, enabling a preliminary filtering pass over the set of images. Using sets of features makes the selection of images much faster than comparing every real image. Using this approach, whenever an image is stored in the database, a number of extractors obtain a set of features which are stored as numeric attributes, together with the image that is stored as a blob attribute.

To support image retrieval based on similarity of the image content, it is necessary to define what similarity mean. This is made through a distance function \( DF() \). A \( DF() \) is an algorithm that compares two images satisfying well-defined algebraic properties, and returns a nonnegative value that is smaller as more similar the two images are. A \( DF() \) is constructed using the extracted features. There can be one or more \( DF() \) defined over a specific set of images, such as similarity based on their histograms [2], number of object occurrences and object placement [4], etc.

3. A Relational database architecture to support images

Existing PACS were built to centralize the storage of images, providing a way to retrieve them based on well-defined protocols, and to distribute them to client workstations or processes. These systems are not designed to be integrated with other existing systems, so it is cumbersome to incorporate them to other applications already existing in a medical center, such as the administrative software or Digital Patient Record systems [1]. On the other hand, those systems are usually built using relational database systems, what makes easier to integrate different systems using the standard relational access language, the SQL (Structured Query Language). As relational database systems do not support the retrieval of images based on their content, we developed a layer atop the database system, that monitors the communication stream between the applications and the database server. This layer, called the Content-based Image Retrieval Core Engine - CIRCE, embraces an extended version of SQL, which supports both the storage of images as a new data type, and the indexing and retrieval of them based on their content, through similarity search operators. When designing the extensions, special attention was paid to minimize the modifications in the language, at the same time maximizing its power to express image retrieval conditions.

Image content-based support is built through a set of extractors, which create a feature vector for each image. The sets of feature vectors are in turn combined with a metric \( DF() \), so that comparisons between pairs of images can be approximated using comparisons between feature vectors. A metric \( DF() \) must obey the non-negativity, symmetry and the triangular inequality\(^2\) properties. Therefore, maintaining the images in a metric domain enables metric indexing structures to be used, greatly improving the speed of image retrieval. In metric domains, the so-called similarity queries most frequently used are the \( k \)-nearest neighbor and the range queries.

To support these kinds of queries, the SQL extensions created enable the definition of the following concepts: the definition of images as a native data type; the definition of the feature vectors and the corresponding \( DF() \); the similarity conditions based on the \( k \)-nearest neighbor and range operators; and the creation of index structures based on the feature vectors and \( DF() \). The main concept is to consider images as another data type supported natively by the DBMS. Therefore,

\(^2\) Given three objects \( a, b \) and \( c \) in the object's domain, the triangular inequality property holds for the distance function \( d(x, y) \) if \( d(a, b) \leq d(a, c) + d(b, c) \).
images are stored as attributes in any relation that requires them, and each relation can have any number of image attributes. Using a very simple example, consider a patient record stored in a relation named 'Patient' having two image attributes for its mugshots, called 'FrontView' and 'Profile'. Images are declared using the new data type 'image' in the table creation command.

The general architecture of CIRCE is shown in figure 1. This figure shows three conceptual databases - the ADB, IPV and IDD - although physically all of them constitute the same database. The ADB corresponds to the existing application database, such as the hospital administrative database or traditional patient record systems. Existing applications do not support images, so the attributes of the relations stored in the ADB are only numbers and/or texts. The ADB can be queried using either the standard SQL or the extended SQL. However, when an application is developed using image support, or when an existing one is expanded to support images, the application must use the extended SQL through CIRCE.

Each image attribute defined in a relation establishes a set of images disjoint from the set established by other attributes. If a query command uses only non-image attributes, the command is sent untouched to the ADB. Whenever an image attribute is mentioned in a query command, CIRCE uses the IPV and IDD databases, together with the metric indexing structures and the parameter extractors, to modify the query command, effectively implementing image retrieval by content. The indexing structure is the Slim-tree [8], one of the most efficient metric access method nowadays.

The IPV and IDD databases hold information about each image attribute defined in the application database. For each one a new relation is created in the IPV database, containing two attributes: a blob attribute storing the actual image, and an Image Identifier (ImId) created by the system. The identifier is a code number, unique for every image in the database, regardless of the relation or attribute where it is stored. Each create table command referencing image attributes is modified, so a numeric data type replaces the 'image' data type in the modified command sent to the ADB. The corresponding IPV database relation is named after the concatenation of the table and attribute names of the original image attribute. In this way, occurrences of this attribute in further query commands are intercepted by CIRCE and translated accordingly. For example, the patient table with two image attributes generates two relations in the IPV database named 'Patient_FrontView' and 'Patient_Profile', each one with two attributes: the image and the ImId.

The comparison of two images requires a metric distance function definition. This is done through the sole new command in the extended SQL: the 'create metric' command. It is part of the Data Definition Language (DDL) of SQL, and enables the specification of the $DF()$ by the domain specialist. Each metric $DF()$ is associated with at least one image attribute, and an image attribute can have any number of metric $DF()$. Image attributes that do not have a $DF()$ cannot be used in search conditions, so it is just stored and retrieved in the usual way. However, if one or more metric
$DF()$ is defined, it can be used both in content-based search conditions and in metric indexing methods.

A $DF()$ definition enrolls one or more extractors and a subset of the features retrieved by them. We consider as features only numbers or vectors of numbers. If the extractors used in a $DF()$ return only numbers, the feature vectors of every image stored in the associated attribute have the same quantity $N$ of elements. Thus, each feature vector is a point in a spatial domain of dimension $N$. However, extractors returning vectors may generate feature vectors with different quantity of elements, resulting in a not spatial domain. This domain can be metric if the $DF()$ is metric. To assure this property, we only allow the definition of $L_p$-norm $DF()$ over the elements of feature vectors. Those elements corresponding to vectors having the difference calculated as the double integral of the curve defined by the vectors [7].

When a $DF()$ is defined for an image attribute, the corresponding relation in the IPV relation is modified to add the elements of the feature vector as numeric attributes. The IDD database is the schema for image attributes, and stores information about the extractors and its parameters. This database also guides the system to store and retrieve the attributes in the IPV database. Whenever a tuple containing image attributes is stored, each image is processed by the set of extractors in the XP module, following the definitions retrieved from the IDD database. After that, its $ImId$ is created and stored, together with the image and the feature vector, in the IPV database. The original tuple is stored in the ADB, switching the images with the corresponding $ImId$ identifiers.

Indexes can be created for each image using the $DF()$ defined for them. If a new image is stored in an indexed attribute, the feature vector is used to index the image. When an image attribute is indexed, one Slim-tree will be created for each $DF()$ defined for this attribute. This enables the retrieval of images based on more than one similarity criterion, allowing the user to choose which criterion is required to answer each query. Each Slim-tree stores the $ImId$ and the feature vector extracted from each image. It allows the execution of the $k$-nearest neighbors and range search procedures using the feature vectors as input, and returns the set of $ImId$ that answer the query.

When a query must be answered, the feature vectors of the involved images are used in place of the real image. If the query refers to images not yet stored in the database ("image constants" expressed directly in the query), the same feature vectors associated to the image attribute being compared, which are already stored in the database, are extracted from this image and used to search the database. This feature vector is sent to the corresponding Slim-tree, which retrieves the $ImId$ of the images that answer the query. Using these $ImId$, the IPV database is used to retrieve the actual images that, in turn, are passed as the answer to the requester process.

4. A prototype implementation

Circe was implemented using the Borland C++Builder programming environment, and runs on MS-Windows NT operating system accessing an Oracle database server. In the experiments performed, it runs as a process in the same machine running the database server, thus reducing network traffic. The server machine was an IBM NetInfinity with 256MBytes of memory and SCSI disks, and a Pentium III 800MHz processor. The client machine was a Pentium III 750MHz processor with 256MBytes of memory, connected to the server through a 100MHz Ethernet link.

To measure the performance gain obtained, we modified an existing application that searches a

---

3 A $L_p$-norm distance function of two arrays $X(x_1, x_2, ...x_N)$ and $Y(y_1, y_2, ...y_N)$ is expressed as

$$DF(X, Y) = \sum_{i=1}^{N} w_i (x_i - y_i)^p$$

where $w_i$ is the weight of each attribute.
set of MR head tomographies looking for a specific disease. The original version retrieved the full set of tomographies in the Oracle server using standard SQL, and then pre-processed each image using a histogram-based filter. Using a set of 5049 MR head tomographies (~2GBytes of disk space), the total time to obtain the answer were approximately four hours. In the modified version, a DF() was created in Circe, using a metric histogram extractor [7]. The pre-processing phase of the application was removed, retrieving the image through a modified SQL command, using a range query to retrieve only highly probable images, using the metric histogram as the initial filter. Working on the same set of tomographies, the total time to obtain the answer dropped to ten minutes, yet selecting the same images as before. The time to extract the metric histograms and to create the Slim-tree index structure, operation that must be performed just once, took about 25 minutes.

However, more important than the impressive drop in the total processing time required, is the fact that the selection operations can be performed using a powerful and easy-to-use extension to the query language. In fact, the modification made in the testing application consisted essentially in the removal of part of the program, the one that performed the initial filtering pass. This not only simplifies the development of such programs, but provide a consistent way to guarantee that the same data is analyzed by similar algorithms.

5. Conclusions

In this paper we described how a powerful extension can be done with few modifications in the standard relational database query language - the SQL - and presented a pre-processor developed to support this extension. A prototype has been implemented and connected to the existing relational database used in a radiology center in a large hospital, which integrates MRI, CT and mammographies. The system provided a seamless image content-based retrieval system, put into use into a real environment, which achieved processing speedups of more than 20 times to answer similarity queries.

As far as the authors are concerned, this the first system to integrate content-based image retrieval with a relational database in an open architecture that can be integrated to existing operational (or administrative) databases. Moreover, it can accept the definition of new comparison methods and image analyzers at any time, in a seamless manner.

6. References