

# Seamless integration of distance functions and feature vectors for similarity-queries processing\*

Marcos Vinicius Naves Bedo, Agma Juci Machado Traina and Caetano Traina Jr.

Institute of Mathematics and Computer Science - ICMC,  
University of São Paulo - USP,  
P.O. Box 668, São Carlos, SP, Brazil - 13560-970  
{bedo, agma, caetano}@icmc.usp.br

**Abstract.** Searching on complex data, such as medical images and financial time-series has attracted several research efforts in recent years. Feature extraction methods (FEM) and distance-functions (DF) play a crucial role when comparing such data, once they can help bridging the *semantic gap* between users' intent and the Content-Based Retrieval Systems (CBRS). Metric access methods (MAM), in turn, are important structures aimed at reducing the CBRS answer time. Integrating all these components into a single *framework* is quite challenging. In fact, such *framework* should provide flexibility to allow the database administrator (DBA) to add new FEM, DF and MAMs on the fly, as they are being required and developed, combining them according to the requirements of each CBRS application. An adequate way to create a *framework* is extending the Relational Database Management Systems (RDBMS) and the SQL language to allow integrating the definition of the FEM, DF and MAMs into the core search engine. In this paper we extend and improve the original strategy of the existing *middleware* SIREN into a new, complete similarity-enabled *framework* called SimbA. Based on a new approach, SimbA allows a much tighter and flexible handling of new complex data types, FEM, DF and MAM in a seamless abstraction. We adopted a SQL extension in order to represent the users' queries. Our extension of the **SELECT** command for similarity comparisons also provides a canonical query execution plan, which allows using data distribution estimates to generate cost-based optimization plans. To illustrate our architecture, we exemplify the developed concepts showing how to add a new complex data type (the financial time-series) as well as its corresponding FEM into our *framework*. Through the extended-SQL and by using complex predicates, we provide example applications to store and query two real datasets as relational tables in commercial RDBMS. The first one (DDSM\_ROI) is composed of mammograms, while the second (BM&F\_BOVESPA) contains weeks of the Brazilian stock exchange index. The experiments performed confirm that our approach can be used as a generic backend for CBRS, whereas maintaining all the desirable features of a RDBMS.

Categories and Subject Descriptors: H.2 [Database Management]: Miscellaneous; H.3 [Information Storage and Retrieval]: Miscellaneous; H.2.3 [Languages]: Miscellaneous

Keywords: distance functions, extended-SQL, feature extractors, similarity queries

## 1. INTRODUCTION

Nowadays, it is mandatory to store complex data such as images, movies, audio and time-series inside Relational Database Management Systems (RDBMS). The development of techniques to recover these data, as well as to mine the intrinsic knowledge embedded in them, have required many research efforts in recent years [Silva et al. 2013]. By following the relational model, the recovery of complex data should be treated as operations which extend those usually employed by RDBMS over traditional data [Zezula et al. 2006; Kaster et al. 2010].

When it comes to complex data, the queries do not require the identity nor the order relationship among the elements. In fact, semantically, it is not often meaningful to express the precedence

---

\* This research was supported by the scholarship grants from the São Paulo State Research Foundation (FAPESP) and by the Brazilian National Research Council (CNPq) and by CAPES.

Copyright©2013 Permission to copy without fee all or part of the material printed in JIDM is granted provided that the copies are not made or distributed for commercial advantage, and that notice is given that copying is by permission of the Sociedade Brasileira de Computação.

relation for these data types [Traina et al. 2003; Yang et al. 2012]. The identity operator can be applied accordingly to its universal definition, but the occurrence of two identical complex elements (e.g. two identical medical images) is very unusual. In the literature, the most solid approach to retrieve data by content is related to the use of similarity measures [Noh 2012; Bedo et al. 2013]. The similarity query paradigm relies on distance measures between the complex elements. The result set is composed of elements in the neighborhood according to a comparison perspective, based on a way to describe the complex data. Formally, the similarity query concept can be expanded for any data types, provided that a similarity measure is defined following the metric space requirements [Silva et al. 2010].

A metric space is defined by a pair  $\langle \mathbb{S}, d() \rangle$ , where  $\mathbb{S}$  is the domain of complex elements and  $d()$  is a metric distance function [Zezula et al. 2006]. The use of  $d()$  allows to measure the distances among the elements of  $\mathbb{S}$ . The values of distances can be compared in similarity operators. For instance, given a query element  $s_q \in \mathbb{S}$ , it is possible to recover all the elements which are not farther than a given radius  $\xi$ . This generates the selection operator known as *Range Query* ( $R_q$ ) [Jahirabadkar and Kulkarni 2014]. Another common selection operator retrieves the  $k$  nearest elements to  $s_q$ . This operator is referred as the  $k$ -Nearest Neighbor Query ( $kNN_q$ ) [Aha et al. 1991].  $R_q$  and  $kNN_q$  are the most employed similarity selection operators in the literature [Yianilos 1993; Silva et al. 2013]. Analogous reasoning can be applied to produce complex join operators as, for instance, the Range Join [Silva et al. 2013].

Similarity operators are employed as the basis of the Content-Based Retrieval Systems (CBRS) as, for example, the Content-Based Medical Image Retrieval System (CBMIR) and the Content-Based Audio Retrieval (CBAR). The original complex data are stored as Binary Large Objects (BLOB) inside RDBMS. A feature extractor method (FEM) is able to represent the original data as a feature vector (FV), allowing a distance function (DF) to measure the similarity among each pair of FVs [Chadha et al. 2012]. Another important point is related to the performance (time demanded) of the query execution. [As with traditional data, index structures may reduce the query answer-time by orders of magnitude \[Skopal 2010; Jr. et al. 2002\]. The Metric Access Methods \(MAM\) are structures capable of \*speeding up\* queries involving predicates with complex attributes, as in the case of metric spaces where only objects and their distances are available.](#)

Therefore, a generic similarity query system executor should be composed by operators able to handle and store complex data types and (1) feature extractors, (2) distance functions and (3) metric access methods. We call those three parts of the query systems as the *authorities* (formally defined in the next Section), who guide the complex data type creation and management. Moreover, a structured language should allow the database administrator (DBA) to employ and tune the aforementioned techniques for specific domains. Toward this aim, some prototypes have been developed to be integrated into a RDBMS. An interesting initiative is the *Similarity Engine Retrieval* (SIREN) [Barioni et al. 2006], a *middleware* able to perform similarity operations over images and audio. However, it does not provide the dynamics and flexibility to easily incorporate new complex data types and their authorities (FEMs, DFs and MAMs), which is an important requirement to support the growing development of experts CBRS (e.g. CBMIR [Bedo et al. 2012]).

In this paper we advance the state-of-the-art, systematically defining the similarity authorities and their relationships. To do so, we propose a new approach for similarity queries and its implementation as a new *framework*: the Similarity by Authorities - SimbA, which employs a powerful data-dictionary to handle new complex data types and their authorities on the fly. We employed the extended-SQL definition of the *middleware* SIREN [Barioni et al. 2006] in order to validate the queries submitted by the users. To illustrate our new architecture, we show how to add a new complex type (the "financial time-series") as well the adequate FEM and DF for this data type and how to combine them according to the applications requirements. For the completeness sake, to extend the SQL Data Manipulation Language (DML) commands, our solution is also able to provide a canonical query execution plan,

giving ground for further estimation of the query cost.

We evaluated our architecture through SQL DML commands using two distinct real datasets in example applications. The first one, denominated DDSM\_ROI is composed of regions of interest (ROI) of mammograms, cropped by specialists of the Clinical Hospital of the University of São Paulo (HCFMRP). We employ FEMs based on color and texture to represent the ROIs and several DFs to calculate the distances among them. The second dataset (BM&F\_BOVESPA) is composed of the index daily values of the Bovespa Stock Exchange covering every week from 2005 up to 2010. We employed a new FEM that we developed [Bedo et al. 2013] for this data type and the Euclidean distance to measure the similarities. The experiments showed that the proposed *framework* can be used as a generic backend for CBRS, providing the support for the specific complex data required by those applications, whereas maintaining all the desirable features of a RDBMS. The remaining of this paper is structured as follows: Section 2 presents the related work in the literature and the main concepts required. Section 3 introduces our proposed *framework*, while Section 4 describes example applications. Finally, in Section 5 we present the conclusions.

## 2. BACKGROUND AND RELATED WORK

In this section we present the definitions related to similarity queries and previous work in the literature. We describe how FEMs are employed to represent complex data in a domain well-suited for the comparison process performed by a DF, where the results are combined as a query operator result. Although those concepts are well-known, we present some of them in a slightly different way, analyzing them in an atomic execution perspective. Our goal is to illustrate the execution pipeline for similarity queries in a structured way, and how our proposed *framework* deals with their execution.

*Definition 2.1.* Feature Extractor Method - FEM: Given a data domain  $\mathbb{S}$  and a extractor domain  $\mathbb{F}$ , a feature extractor method  $FEM : \mathbb{S} \rightarrow \mathbb{F}$ , is a computational function able to represent the original complex data as its summarization in  $\mathbb{F}$ . The composition of two or more FEMs ( $FEM \circ FEM$ ) is also a FEM.

*Definition 2.2.* Feature Vector - FV: Given a specific element  $s_q \in \mathbb{S}$  and a FEM, the feature vector  $f_q \in \mathbb{F}$  is the concise representation of the original data element generated by the FEM in  $\mathbb{F}$ , which is also said to be a mapping of  $s_q$  in  $\mathbb{F}$ .

*Definition 2.3.* Metric Distance Function - DF: Given,  $f_i$ ,  $f_j$  and  $f_k \in \mathbb{F}$ ,  $\delta : \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{R}^+$  is a metric distance function, if it satisfies the following three properties:

- (1) Symmetry:  $\delta(f_i, f_j) = \delta(f_j, f_i)$ ;
- (2) Non-negativity:  $0 < \delta(f_i, f_j) < \infty$ ;
- (3) Triangular Inequality:  $\delta(f_i, f_j) \leq \delta(f_i, f_k) + \delta(f_k, f_j)$

*Definition 2.4.* Complex Data Descriptor - CDD: A complex data descriptor is a pair  $\langle \epsilon, \delta \rangle$ , where  $\delta$  is a (weighted) metric distance function and  $\epsilon$  is a FEM(s).

*Definition 2.5.* Data Type Authorities - DTA: The set of FEM, DF and Metric Access Methods (MAM) assigned to each particular data type is called the Authorities of the type.

*Definition 2.6.* Nearest-Neighbor Query -  $kNN_q$ : Given a query element  $f_q \in \mathbb{F}$ , a DF  $\delta$  and a number of neighbors  $k \in \mathbb{N}$ , the nearest-neighbor query answer is the unitary subset of  $F$  such that  $kNN_q(f_q, k) = \{f_n \in F \mid \forall f_i \in F; \delta(f_n, f_q) \leq \delta(f_i, f_q)\}$ .

*Definition 2.7.* Range-Query -  $R_q$ : Given a query element  $f_q \in \mathbb{F}$  and a maximum distance  $\xi \in \mathbb{R}^+$ , the answer  $R_q$  will be the subset of  $F$  such that  $R_q(f_q, \xi) = \{f_i \mid f_i \in F; \delta(f_i, f_q) \leq \xi\}$

## 2.1 Feature extractor methods

For each domain of complex data there are several methods proposed in the literature able to act as a FEM [Chadha et al. 2012; Sonka et al. 2007]. For images, generic features are usually based on color, texture, shape or their relationship. Besides, there is a large number of features specific for particular domains, such as for face recognition, fingertips, medical imaging, satellite or remote sensing imaging, etc. In the financial time-series case, statistical values are commonly extracted, such as mean and volume among others [Mills 2008]. The following Sections present concepts of some features that were employed in the implementation of our proposed *framework*.

**2.1.1 Color-based FEMs for images.** Most of FEMs employ color as the basis for the feature extraction. Usually, each pixel can be represented as a point in a proper color-scale system. Thus, taking their relative frequency is the main idea behind the histogram-based methods. The *equi-width* histogram [Ioannidis 2003] is sorted by the pixel-value (either gray or true-color scale) entries equally partitioned, where the sum of frequencies represents the cumulative probability. Formally, for a gray-color histogram  $h(b_k)$ , we can denote  $h(b_k) = n_k$  where  $b_k$  is the  $k$ -th pixel value and  $n_k$  is the number of pixels of intensity  $b_k$  on the image, considering  $b_k \in [0, L - 1]$  where  $L$  is the upper bound of the gray scale. Analogously, the color histogram can be expressed as a composition of single histograms taken by each color band [Yang et al. 2012]. Another histogram approach worthy to note is the one proposed in [Traina et al. 2003] and called as *metric-histogram*. Such technique can summarize an original histogram by joining the very similar frequencies in consecutive intensities. The correlations among the pixel-values are represented by their piecewise approximation in selected intervals of points (pixel value, frequency) of the original histogram. The set of chosen points produces a reduced version of the histogram requiring less space to be stored and being faster to compare.

**2.1.2 Texture and Shape-based FEMs for images.** There are several definitions of surface roughness and texture for image processing, but generically it can be described as a visual pattern, where there are several elements repeating. The texture-based FEMs can also be labeled as statistic (based on density) or structured (based on a previously segmented element). The fourteen characteristics proposed by Haralick et al. [1973] are able to express texture values such as entropy, variance and energy based on co-occurrences matrix of distances and angles. Another important FEM based on image contraction are the *wavelets* transforms. The main idea behind the wavelets descriptions is the reduction of information through a proper filter. Haar and Daubechies provide a basis for wavelets transforms for images [Mallat 2008]. Other techniques are also useful as image FEMs, such as the Zernike moments that are able to gather the shape distribution of an image [Sonka et al. 2007].

**2.1.3 FEMs for financial time-series.** Financial time-series are a particular realization of time-series, on which a graphical analysis can be performed. Several values can be obtained from a window-series such as Bollinger bands and the exponential moving average (EMA) [Mills 2008]. Both methods do not rely on a particular representation of the time-series, which is taken into account by other FEM methods, such as the *gap descriptor* (GAP). GAP is based on the *candlesticks* representation of a financial time-series, where the disruptions among two consecutive points (e.g. the closing price of share is not necessarily its opening price on the next day) are captured by four linear proposed functions. Although GAP only considers the intrinsic information of share movement (i.e. external variables as trending volume are discarded), the captured disruptions are adequate to represent trending and influence, suiting well for the financial time-series domain [Bedo et al. 2013].

## 2.2 Distance functions

To answer both similarity queries  $R_q$  and  $kNN_q$  it is imperative to measure the similarity among two feature vectors according to a defined CDD. This quantification is performed by a DF, which evaluates the similarity of a pair of elements as a distance expressed as a real value. The smaller the distance

value, the more similar the elements are. There are important functions that can measure similarity but not all of them respect the requirements of Definition 2.3. For example, Data Time-Warping (DTW) and the Jeffrey's Divergence do not satisfy the triangular inequality property [Rakthanmanon et al. 2012]. For the scope of this work, we will consider only metric distance functions, which allow using metric access methods. Perhaps, the most iconic family of DFs is the Minkowsky, which includes the Euclidean distance ( $L_2$ ). DFs of Minkowsky family compare each pair of FVs dimension by dimension, according to the following expression:  $L_p(X, Y) = \sqrt[p]{\sum_{i=1}^n |x_i - y_i|^p}$ , where  $X = \{x_1, x_2, \dots, x_n\}$  and  $Y = \{y_1, y_2, \dots, y_n\}$  are generated by the same FEM and the operator  $\{-\}$  is defined for each dimension of the FVs. A particular variation of  $L_1$  is the Canberra Distance, which is expressed as  $C(X, Y) = \sum_{i=1}^n \frac{|x_i - y_i|}{|x_i| + |y_i|}$  (when  $x_i = y_i = 0$  the fraction  $i$  is defined as 0). Another widely employed distance of the Minkowsky family is the Chebyshev member ( $L_\infty$ ). Notice that there are DFs able to compare FVs which are not defined over  $\mathbb{R}^n$  [Zezula et al. 2006]. For instance, the Metric Histogram Distance is able to obtain the similarity between two metric histograms, by calculating the area difference under their piecewise linear approximation [Traina et al. 2003].

### 2.3 Similarity queries support on RDBMS

The goal to integrate complex data into RDBMS is to allow the modeling of complex data using the entity-relationship paradigm. To query the stored data, a suitable language should also be adopted [Budíková et al. 2012]. Barioni et al. [2009] has defined an extension to SQL language, which is used by SIREN. Other prototypes are able to represent similarity queries as user-defined functions, such as the FMI-SiR [Kaster et al. 2010] and the PostgreSQL-IE [Guliatto et al. 2009]. Although the original proposal of SIREN allows definitions of CDDs by DDL commands, it lacks flexibility to include new complex data types and their authorities. On the other hand, FMI-SiR allows the integration of new FEMs, but lacks flexibility to add new complex data type. As a *framework* proposed to operate over ORACLE® architecture, FMI-SiR stores the complex data and its feature vectors as BLOBs (and the system ORDImage variations) in a single table, where the DBA is responsible for the creation and maintenance of the tables. PostgreSQL-IE, instead, has a proper data-dictionary to store images (of one specific type) and FVs  $\in \mathbb{R}^n$ , and allow the insertion of new FEM, yet it does not provide dynamic support for new complex data types or DFs. Figure 1 highlights the main characteristics of the aforementioned systems. Our proposed approach and its implementation was designed to fulfill all of those desirable characteristics.

	Support for new FEM	Support for new DF	Support for new MAM	Similarity Query by extended-SQL	Dynamic combination FEM and DF	Alternative Query execution	Independence of RDBMS
SIREN			■	■	■		■
FMI-SiR	■	■	■			■	
PostgreSQL-IE	■				■		
SimbA	■	■	■	■	■	■	■

Fig. 1. Comparative characteristics table among SIREN, FMI-SiR, PostgreSQL-IE and SimbA.

## 3. THE PROPOSED SIMBA FRAMEWORK

In this section we present our approach and the SimbA *framework*, designed to handle complex data storage and retrieval, which provides a flexible support to dynamically include new authorities for the complex data types. Our solution is based on a set of authorities' libraries. The integration of every atomic resources results in a *symbiotic* backend system: We take advantage of a user-selected

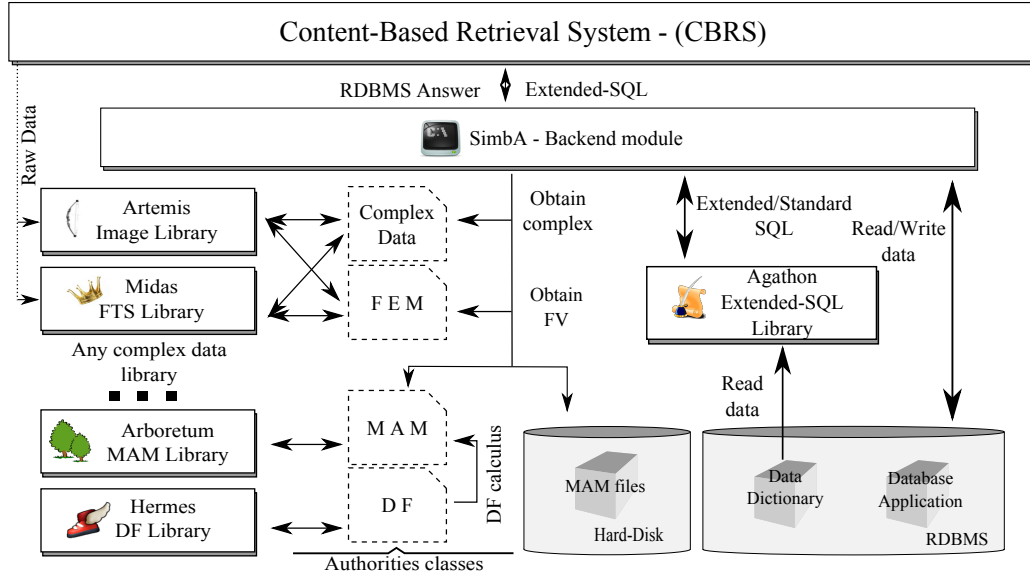


Fig. 2. Proposed SimbA architecture.

RDBMS system (we evaluated our proposal using Oracle®, Postgres® and Oracle MySQL®) to store the complex data and FVs as BLOBs and guarantee the ACID properties. On the other hand, we provide both syntactic and semantic interpretation for every SQL command. When a complex data is detected in a query command, the system performs all the required processing, translating the similarity-related command into a query expression tree and a regular SQL for the RDBMS input.

Figure 2 provides a high level view of the architecture of the SimbA *framework*. The definitions of FEM, DF and FV of Section 2 are mapped into the authorities libraries, which are responsible for the execution. The libraries are linked to SimbA as generic classes, which generalizes the entire concept for the system. For instance, from the image library perspective, the system is able to deal with several image data types (.jpg, .png, .dicom, etc). On the other hand, from a system perspective, there is no special format (such as indicated by a file extension), but only the complex object class representation.

The backend module can also be used to define the CBRS *schema* through DDL commands, which supports the extended-SQL. The first task performed by a DBA when configuring a database that stores complex data is to define how each pair of elements must be compared. The **CREATE METRIC** statement is the implementation of the CDD concept (Definition 2.4). More than one complex data descriptor may be needed for a given requirement analysis. Thus, the DBA should specify all required CDD related to the complex data. The second step in a *schema* definition is the specification of the tables. We propose to store the complex data into a separated table, linked with the original one by a foreign key. This solution allow that the original complex data and every system controlled FVs (Definition 2.2) specified by the CDDs be stored in any table in a way that is transparent to the users. For the RDBMS-portability sake and for security reasons, our *framework* always convert the BLOBs to an hexadecimal or base-64 representation, thus the DBA can dump/load the RDBMS without data loss and allowing data encryption. Removing the **METRIC** definition requires removing all related complex attributes in all internal tables of the *schema*, warranting the consistency of the definitions and related data stored in the database.

Figure 3 (a) illustrates how the linked tables are able to store the feature vectors and the complex data inside a RDBMS. For the **INSERT**, **DELETE** and **UPDATE** commands (Fig. 3 (a) Step 1), the backend validates the user input through the extended-SQL library ((Fig. 3 (a) Step 2). When a complex

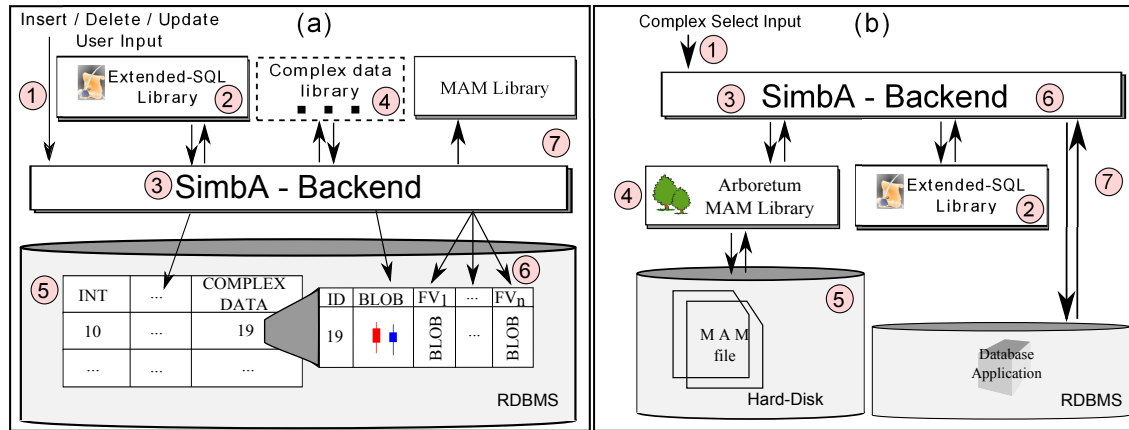


Fig. 3. Execution pipeline for (a) INSERT, DELETE and UPDATE commands (b) SELECT with complex predicates.

data is identified, the library requests the related FVs to the backend. Then, the backend employs the expert library to open the file and perform the extraction (Fig. 3 (a) Step 3) through the complex object class. The extended-SQL library provides a set of standard INSERT statements containing the ciphered blobs to the backend (Fig. 3 (a) Step 4), which finally persist the data into the RDBMS (Fig. 3 (a) Step 5 and 6). Notice that, when a complex data has to be stored, the system actually stores it in a controlled table, which is transparent to the user. This organization allows the extracted FVs to be stored in the same tuple of the complex element, but they are hidden from the users. The controlled and users' table are linked by a system ID. To improve the similarity query performance, a MAM is always assigned for each complex column definition. Thus the FVs are also inserted into the respective MAM through its expert library, and any query involving the complex data is always executed using the corresponding MAM before accessing the data elements.

In order to validate the statement, the data-dictionaries of the RDBMS and SimbA are consulted and if any non-conformity is found, an error is throw. In the case of the DELETE command, the CASCADE policy is always applied. In the case of delete/update of tuples involving complex data, an additional step is required. Once the MAM is not supported by the basic RDBMS, SimbA demands to the indexes library (through the MAM class) to remove/update the data on index file, which is stored outside the RDBMS (Fig. 3 (a) Step 7).

Figure 3 (b) illustrates the pipeline execution order for the SELECT command. First, the extended-SQL library validates the user query (Fig. 3 (b) Step 2). If a complex predicate is part of the query, the library demands the backend to solve the particular predicate (Fig. 3 (b) Step 3). Thereafter, the backend requests the MAM library to perform the retrieval operation (Fig. 3 (b) Step 4), which accesses the indexes files (Fig. 3 (b) Step 5) and returns the identifiers of the FVs, which satisfies the  $kNN_q$  or  $R_q$  predicate (the transparent table 'IDs' that links the users' and controlled' tables). In the next step, the backend rewrites the query using the standard SQL (Fig. 3 (b) Step 6), where the selection predicates are replaced by IN clauses containing the set of returned identifiers. Finally, the RDBMS receives and executes the standard SQL, providing the query result (Fig. 3 (b) Step 7).

An analogous workflow is employed to solve the complex JOIN predicates. The expert authorities libraries solve the similarity JOIN operation (i.e. RANGE, NEAREST or CLOSEST), returning a set of joined transparent 'IDs' pairs. Thereafter, the backend rewrites the query as a Cartesian product between the two original tables and a double IN list containing the pairs of elements returned. As the original query processing is independent of the RDBMS interpretation, our approach allows constructing a canonical query execution plan, in order to provide the first query-execution estimation. It allows the query optimizer to determine the best position in the query execution path where the similarity

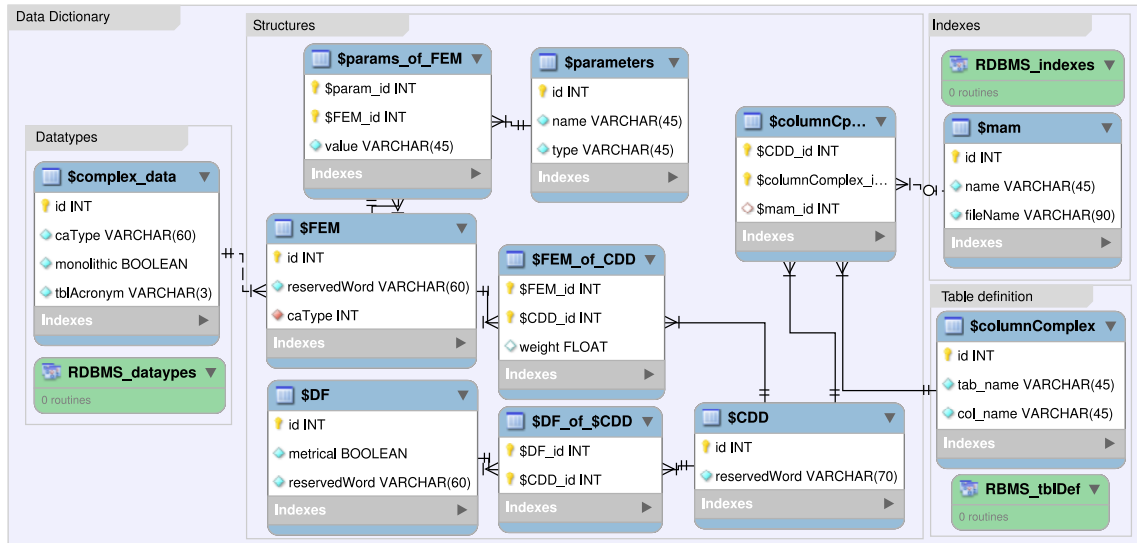


Fig. 4. Proposed SimbA data dictionary model.

operations should be executed, considering that our proposed solution act as a *middleware* and that the similarity-based operations are performed outside the RDBMS.

### 3.1 The proposed data-dictionary

The definitions of the FEM, DF, CDD, FV and MAMs and the assignment of them to each complex attribute in each relation were implemented by authority libraries, which are linked to SimbA as class representations. It is also very useful to model these concepts using an entity-relationship paradigm, allowing the DBA definitions to be persisted. Thus, as for the authority libraries, we define a new data-dictionary able to represent the structures as well to store new instances, providing the required dynamic insertion/deletion of authorities.

Figure 4 provides an overview of the proposed data-dictionary. The available SimbA data types are the traditional types supported by the RDBMS and the new DBA-defined complex types (previously, a data type library must be defined). The data-dictionary also unifies the traditional indexes and the MAM library to represent indexes for all data types. Finally, the system structures are able to store the relations among the similarity-query components, such as the user-defined CDDs and the tables, which contain complex-object columns. By adopting the transparent table strategy (Figure 3 (a)), the complete *schema* can be straightforwardly stored as a similarity-related meta database.

## 4. ADDING A NEW COMPLEX DATA TYPE AND RELATED AUTHORITIES

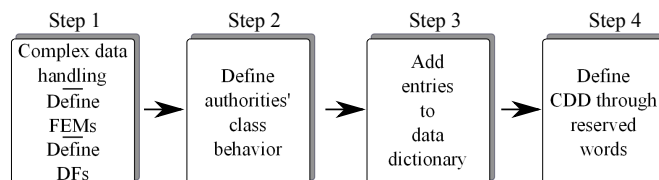


Fig. 5. Block-diagram for inserting a new complex data-type to be stored in RDBMS.



Our strategy employs the proposed architecture and data-dictionary to provide dynamicity of integrating new resources, as they are being developed. Figure 5 illustrates the workflow of how to incorporate a new complex data type and their authorities into a RDBMS. For each new complex data type that the DBA wants to store inside a RDBMS, the following steps are required:

*Step 1* - Implement data-handling procedures to open and write files (for the required file-formats) and the FEMs and DFs for the specific new data type by the corresponding authority library.

*Step 2* - Define the complex object behavior in the Complex Object generic class.

*Step 3* - Include the equivalent entries into Simba data-dictionary. To do so, the DBA should use the following pre-defined functions, which works as **STORED PROCEDURES**:

- *insert\_complex\_data* - This function stores the reserved word for the complex data type into the data-dictionary, its characteristics (e.g. **MONOLITHIC**, **SCALAR** , etc.) and the acronym to be used for the transparent tables.
- *insert\_fem* - This function stores the reserved word for the FEM developed and the related complex attribute. Default FEM parameters should also be provided.
- *insert\_parameters\_of\_fem* - This function stores the possible parameters employed for a specific FEM. It is only allowed to call the extractors with parameters that are stored in the data dictionary.
- *insert\_df* - This function stores the reserved word to represent a DF and its characteristics.
- *define\_fem\_df\_relationship* - This function allows a DF to be used over the FVs generated by an specific FEM.
- *insert\_mam* - This function stores the reserved word to be used as an index for the complex data type. A method implemented in the MAM library and the MAM generic class should be previously defined.

*Step 4* - Combine the CDD through extended-DDL commands.

Once fulfilled all these steps, the DBA can start the process of table creation, using the newly complex data type as the type of attributes that can be added to any table of the user's schema. The specified metrics will be used to compare the stored tuples.

## 5. EXAMPLE APPLICATIONS

We created a web interface and a *shell*, where the user can pose queries that include the extended-SQL constructions and visualize the results. Both runs over the Simba *framework*. In this Section we present two case studies using datasets encompassing complex data. The first dataset, called DDSM\_ROI is composed of more than 2000 images containing regions of interest of mammograms studies cropped by medical specialists of the University of São Paulo Clinical Hospital (HCFMRP). The second one is composed of daily values of all weeks of the IBOV index from BM&F\_BOVESPA starting on 2005 until 2010. Each point of the financial time-series is composed of four values: the opening, closing, highest and lowest values assumed by the stock index during each day.

### 5.1 The medical images analysis case

For DDSM\_ROI we created four tables, which compose a 'mammogram study' - the mammograms projections: LCC, LMLO, RCC and RMLO. Each table has one complex attribute (**STILLIMAGE**). Our *schema* allows comparing any image, considering distinct CDDs for the comparisons as shown in commands (1) and (2). It allows choosing the desired metric at each query, using two distinct perspectives. The mammogram images are stored in the four tables, each one following the same structure, as exemplified in command (3) for table **lccMammogram**. The other follow equivalent definitions.

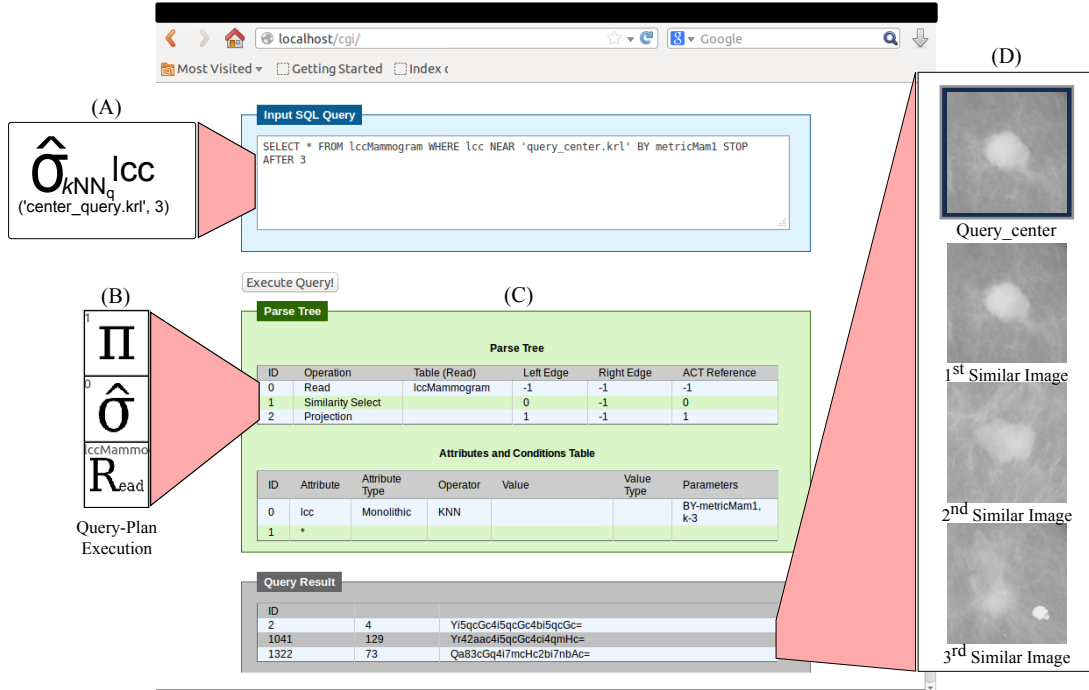


Fig. 6. Web interface with similarity-query results by command (4) involving a  $kNN_q$  predicate. The pop-outs in the figure are shown in the tool through navigation links: (A) the relational-algebra equivalent expression, (B) and (C) the constructed parse-tree, and (D) the query-results.

- (1) CREATE METRIC metricMam1 USING Chebyshev  
FOR STILLIMAGE (waveletshaarext (haar AS h));
- (2) CREATE METRIC metricMam2 USING Canberra FOR STILLIMAGE  
(histogramext (histogram AS hist 2), zernikeext (zernike AS zern) );
- (3) CREATE TABLE lccMammogram (id INTEGER, idStudy INTEGER, lcc STILLIMAGE,  
PRIMARY KEY (id), METRIC (lcc) USING (metricMam1 DEFAULT, metricMam2) );

The first CDD combines one metric, the Haar wavelets (FEM) and the Chebyshev distance function, while the second CDD employs two FEM: the *equi-width* histogram with weight 2 and the Zernike moments, and measures the composed similarity using the Canberra distance. In particular, for medical domains, it is important to determine the comparison parameter to be used, once each METRIC have distinct resolutions, and wrong comparisons disturb the expert's analysis. For instance, previous research on this dataset have shown that the first metric obtains better precision to retrieve images of 'mass' evidences, while the second one obtains better precision to retrieve images of 'calcification' evidences.

A similarity-query involving the  $kNN$  predicate can be expressed as shown in command (4). To illustrate the action of our *framework*, we also show command (5), which is the command converted to standard-SQL that the backend submits to the underlying RDBMS.

- (4) SELECT \* FROM lccMammogram  
WHERE lcc NEAR 'center\_query.krl'  
BY metricMam1 STOP AFTER 3;

```
(5) SELECT id, idStudy, IPV$lccMammogram_lcc.lcc AS lcc
FROM lccMammogram JOIN IPV$lccMammogram_lcc
ON lccMammogram.lcc = IPV$lccMammogram_lcc.lcc_id
WHERE lccMammogram IN (7, 1729, 406);
```

A medical CBMIR system can employ the interpretation and translation of extended-SQL as the *backend* to perform similarity queries. Figure 6 presents our developed web evaluation tool, which performs similarity queries using the SimbA *framework* through a Common Gateway Interface (cgi)<sup>1</sup>. The pop-outs of Figure 6 present the visualization of navigation links - Figure 6 (A) presents the equivalent relational-algebra expression for the input query, Figure 6 (B) and Figure 6 (C) show the parse-tree representation for the query-plan and, finally, Figure 6 (D) shows the query-results.

## 5.2 The financial time-series analysis case

We also analyzed our approach over a second dataset, now storing time series as complex attributes. It requires adding a new data type (the WEEK\_SERIES) to SimbA, following the steps described in Section 4. We created a table to store the related information, which describes the index behavior for every week. We created the following metric to compare the elements:

```
(6) CREATE METRIC gapAndEuclidean USING Euclidean
FOR WEEK_SERIES (gapext (gap AS g) );
```

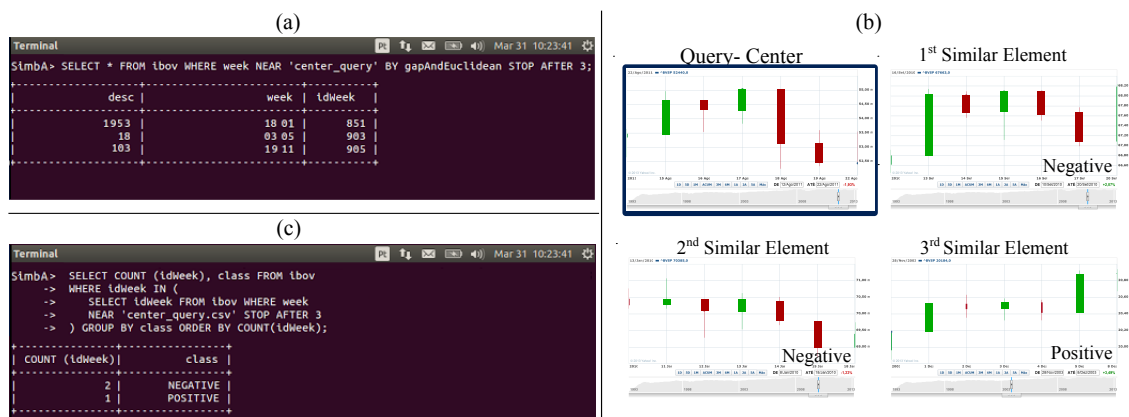


Fig. 7. (a) Similarity-query result over the SimbA *shell* (b) Visual representation of similar weeks (c) Result of grouping extended-SQL expression.

Figure 7 (a) exemplifies the query results for the 3 weeks most similar to the current one. The graphical representation of the result set is shown in Figure 7 (b). Several query variations can be requested in order to collect information of normalized tables. Another important question of this particular application is how to apply the similarity concept in order to obtain the classification of the current element based on the most similar weeks. Considering that one week is stored as a MONOLITHIC data type and that we can add a column to represent groups of stored weeks (for instance, a *boolean* column named 'class' that stores the week as positive or negative), it is now possible to pose a group-by command combined with the extended-SQL expression, such as:

<sup>1</sup><http://www.w3.org/CGI/>

```
(7) SELECT COUNT (idWeek), class FROM ibov WHERE idWeek IN (
      SELECT idWeek
      FROM ibov
      WHERE week NEAR 'center_query.csv' STOP AFTER 5
    ) GROUP BY class ORDER BY COUNT (idWeek);
```

The results of extended-SQL grouping commands are shown in Figure 7 (c). A last important aspect to be considered is query execution time. The wall-clock time spent to answer the query shown in Figure 6 took less than 1.2 secs in average, while to answer the question posed at Figure 7 (a) took less than 250 ms. in average. All the experiments were performed over an Intel Core i5 ® with 2,67 Ghz processor and 2 GB memory under Ubuntu 12.04.

## 6. CONCLUSIONS AND FUTURE WORKS

There are several researches based on the similarity search paradigm, but only few of them target the integration of the developed features into a RDBMS. This integration is required to bring the CBRS development as a standard software project. Aimed at reducing the gap among the development of similarity-based algorithms and its practical implementation into RDBMS, we presented an important contribution by systematically defining the similarity authorities concept application in RDBMSs.

We developed the SimbA to illustrate its implementation into the similarity search process. We showed how to extend the basic dictionary of a RDBMS, defining the structure for the corresponding complex data dictionary and architecture, to allow adding new complex data types and their authorities through experts libraries. Moreover, by using a few reserved words we defined a powerful way to deal with FEMs and DFs in extended-SQL, which was an open point in all the previous works. Unlike main competitors which are dependent on proprietary RDBMS technologies, SimbA is able to operate over almost any commercial RDBMS fulfilling the desired characteristics shown in Figure 1. Finally, we presented example applications of similarity queries over databases that include images and financial time-series, where the SimbA can play the backend role for CBRS systems, providing strong support for query execution.

Future directions for the *famework* include to provide support to query-execution analysis and optimization - once it is already possible to represent the query-execution plan, SimbA should determine which strategies can be applied to the faster query-execution. Another future direction is to provide support for other data types, such as XML, DNA sequence among others, developing the specific authorities' libraries.

## REFERENCES

- AHA, D. W., KIBLER, D., AND ALBERT, M. K. Instance-Based Learning Algorithms. *Journal of Machine Learning* 6 (1): 37–66, Jan., 1991.
- BARIONI, M. C. N., RAZENTE, H., TRAINA, A., AND TRAINA-JR., C. Siren: A similarity retrieval engine for complex data. In *Proceedings of the 32nd International Conference on Very Large Data Bases. VLDB '06*. VLDB Endowment, Seoul, Korea, pp. 1155–1158, 2006.
- BARIONI, M. C. N., RAZENTE, H. L., TRAINA, A. J. M., AND TRAINA-JR., C. Seamlessly integrating similarity queries in SQL. *Software: Practice and Experience* 39 (4): 355–384, 2009.
- BEDO, M. V. N., DOS SANTOS, D. P., KASTER, D. S., AND TRAINA-JR., C. A Similarity-Based Approach for Financial Time Series Analysis and Forecasting. In *International Conference on Database and Expert Systems Applications*. Vol. 2. Springer, Prague, Czech Republic, pp. 94–108, 2013.
- BEDO, M. V. N., PONCIANO-SILVA, M., KASTER, D., BUGGATI, P. H., TRAINA, A. J. M., AND TRAINA-JR., C. Higii: a perceptual medical CBIR system applied to mammography classification. In *Brazilian Symposium on Databases*. Vol. 1. SBBD'12, São Paulo - Brazil, pp. 22–28, 2012.
- BUDÍKOVÁ, P., BATKO, M., AND ZEŽULA, P. Query Language for Complex Similarity Queries. In *East-European Conference on Advances in Databases and Information Systems*. Springer-Verlag, Berlin, Heidelberg, pp. 85–98, 2012.

- CHADHA, A., MALLIK, S., AND JOHAR, R. Comparative Study and Optimization of Feature-Extraction Techniques for Content based Image Retrieval. *International Journal of Computer Applications* vol. 52, pp. 35–42, 2012.
- GULIATO, D., DE MELO, E. V., RANGAYAN, R. M., AND SOARES, R. C. PostgreSQL-IE: An Image-handling Extension for PostgreSQL. *Journal of Digital Imaging* 22 (2): 149–165, 2009.
- HARALICK, R., SHANMUGAM, K., AND DINSTEN, I. Texture Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics* 3 (6): 610–621, 1973.
- IOANNIDIS, Y. The History of Histograms (Abridged). In *International Conference on Very Large Data Bases*. Vol. 29. VLDB Endowment, Berlin, Germany, pp. 19–30, 2003.
- JAHIRABADKAR, S. AND KULKARNI, P. Algorithm to determine radius-distance parameter in density based clustering. *Expert Systems and Applications* 41 (6): 2939–2946, 2014.
- JR., C. T., TRAINA, A. J. M., FALOUTSOS, C., AND SEEGER, B. Fast Indexing and Visualization of Metric Data Sets using Slim-Trees. *IEEE Transactions on Knowledge and Data Engineering* 14 (2): 244–260, 2002.
- KASTER, D. S., BUGATTI, P. H., TRAINA, A. J. M., AND JR., C. T. FMI-SiR: A Flexible and Efficient Module for Similarity Searching on Oracle Database. *Journal of Information and Data Management* 1 (2): 229–244, 2010.
- MALLAT, S. *A Wavelet Tour of Signal Processing, Third Edition: the sparse way*. Academic Press, Waltham, MA, USA, 2008.
- MILLS, T. *The econometric modelling of financial time series*. Cambridge University Press, Cambridge [u.a.], 2008.
- NOH, S. Metric learning for nearest neighbor classification and its analysis. In *International Conference on Pattern Recognition*. IEEE, Tsukuba, Japan, pp. 991–995, 2012.
- RAKTHANMANON, T., CAMPANA, B. J. L., MUEEN, A., BATISTA, G. E. A. P. A., WESTOVER, M. B., ZHU, Q., ZAKARIA, J., AND KEOGH, E. J. Searching and mining trillions of time series subsequences under dynamic time warping. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, pp. 262–270, 2012.
- SILVA, Y. N., ALY, A. M., AREF, W. G., AND LARSON, P.-Å. SimDB: a similarity-aware database system. In *ACM SIGMOD/PODS Conference*. ACM, New York, NY, USA, pp. 1243–1246, 2010.
- SILVA, Y. N., AREF, W. G., LARSON, P.-Å., PEARSON, S., AND ALI, M. H. Similarity queries: their conceptual evaluation, transformations, and processing. *The VLDB Journal* 22 (3): 395–420, 2013.
- SILVA, Y. N., PEARSON, S., AND CHENEY, J. A. Database Similarity Join for Metric Spaces. In *International Workshop on Similarity Search and Applications*. ACM, New York, NY, USA, pp. 266–279, 2013.
- SKOPAL, T. Where are you heading, metric access methods?: a provocative survey. In *International Workshop on Similarity Search and Applications*. ACM, New York, NY, USA, pp. 13–21, 2010.
- SONKA, M., HLAVAC, V., AND BOYLE, R. *Image Processing and Machine Vision*. Thomson-Engineering, Mobile, AL, USA, 2007.
- TRAINA, A. J. M., JR., C. T., BUENO, J. M., CHINO, F. J. T., AND AZEVEDO-MARQUES, P. M. Efficient Content-Based Image Retrieval through Metric Histograms. *World Wide Web* 6 (2): 157–185, 2003.
- YANG, Y., NIE, F., LUO, J., ZHUANG, Y., AND PAN, Y. A Multimedia Retrieval Framework Based on Semi-Supervised Ranking and Relevance Feedback. *IEEE Trans. Pattern Anal. Mach. Intell.* 34 (4): 723–742, Apr., 2012.
- YIANILOS, P. N. Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. In *ACM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 311–321, 1993.
- ZEZULA, P., AMATO, G., DOHNAL, V., AND BATKO, M. *Similarity Search - The Metric Space Approach*. Advances in Database Systems, vol. 32. Springer Publishing Company, Incorporated, Berlin, Heidelberg, 2006.