Adding Knowledge Extracted by Association Rules into Similarity Queries

Mônica R. P. Ferreira[1,3], Marcela X. Ribeiro[2], Agma J. M. Traina[1], Richard Chbeir[3], Caetano Traina Jr.[1]

[1] Computer Science Dept., ICMC-Univ. of São Paulo, São Carlos-SP, Brazil
{monika, agma, caetano}@icmc.usp.br
[2] Computer Science Dept., Univ. Fed. de São Carlos, São Carlos-SP, Brazil
marcela@dc.ufscar.br
[3] LE2I Laboratory UMR-CNRS Univ. of Bourgogne, Dijon, France
{Monica.Ferreira, richard.chbeir}@u-bourgogne.fr

Abstract. In this paper, we propose new techniques to improve the quality of similarity queries over image databases performing association rule mining over textual descriptions and automatically extracted features of the image content. Based on the knowledge mined, each query posed is rewritten in order to better meet the user expectations. We propose an extension of SQL aimed at exploring mining processes over complex data, generating association rules that extract semantic information from the textual description superimposed to the extracted features, thereafter using them to rewrite the queries. As a result, the system obtains results closer to the user expectation than it could using only the traditional, plain similarity query execution.

Categories and Subject Descriptors: H.2.3 [**Database Management**]: Languages—*Query languages*; H.2.8 [**Database Management**]: Database Applications

Keywords: association rules, content-based retrieval, query rewriting, similarity queries, SQL extension, user expectation

1. INTRODUCTION

Retrieving complex data, such as images, audio, time series, and genomic sequences, from a large dataset is best solved comparing the data elements by similarity, as two identical elements seldom occurs. However, similarity queries over complex data are both slow to evaluate and hard to define. They are slow to evaluate because the operations executed by the software to obtain the answer involves handling large amounts of data, due to the small amount of opportunities that exist to prune parts of the dataset from evaluation, as it is often possible over traditional data. They are hard to define because describing what the user expects requires a lot of specifications.

If the user has a small set of images that meets his/her expectations, it would be interesting to say: "*I am interested in images like those!*" Suppose the user's database stores photos and other related data, including several types of photos, such as interior of buildings, nature, animals, people, etc. Now, let us assume that the user is interested in images from the nature. If he/she wants to find images similar to a given image X at hand "but from the nature", it is very hard to specify what "from the nature" means. However, if he/she has a small collection of images "from the nature", it could say: "*Find images similar to X that are like my images from nature*". In this paper we present a technique, based on association rules mining integrated with similarity query answering, that allows the user to express queries in this way.

This work has been supported by FAPESP, CAPES, CNPq, STIC/Amsud, CNRS and Microsoft Research.

In the beginning, relational database management systems (RDBMSs) supported only numbers and small character strings, called traditional data, which can be compared using exact matching ($=$ and $\neq$) and relational ($<$, $\leq$, $>$ and $\geq$) operators, also called traditional operators. Traditional data can be mined using traditional data mining techniques, as association rules. Nowadays, with the growing interest in complex data, many of the traditional RDBMSs are being extended to support these new kinds of data. Furthermore, traditional data mining techniques are also being extended to discover knowledge from this kind of data [Han and Kamber 2006].

As the conventional operators are not applied over complex data, the notion of similarity is emerging naturally as the way to compare pairs of elements in complex domains, pushing the degree of similarity among data elements as the most important factor [Faloutsos 1997]. Therefore, a distance function d, which is the basis to create a metric space, should be defined. Formally, a metric space is a pair $M = \langle \mathbb{S}, d \rangle$, where $\mathbb{S}$ denotes the universe of valid elements, and d is a function $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ that expresses a "distance", or a dissimilarity between elements of $\mathbb{S}$, which satisfies the following properties: (*i*) symmetry: $d(s_1, s_2) = d(s_2, s_1)$; (*ii*) non-negativity: $0 < d(s_1, s_2) < \infty$, if $s_1 \neq s_2$ and $d(s_1, s_1) = 0$; and (*iii*) triangular inequality: $d(s_1, s_2) \leq d(s_1, s_3) + d(s_3, s_2), \forall s_1, s_2, s_3 \in \mathbb{S}$. Complex data can be queried by similarity, and the two most known similarity query types are the range and the k-nearest neighbor. The range query returns the elements closer than or at the same distance to a similarity threshold from a query element (also called the query center); the k-nearest neighbor (k-NN) query returns the k elements nearest to the query center.

The integration of traditional data mining techniques to RDBMS creates a useful analytical tool, and researchers are studying how to provide data mining query commands as SQL extensions. Several query languages for data mining have been proposed [Imielinski and Mannila 1996]. The first language proposed was `DMQL` (the Data Mining Query Language), aiming at expanding the *DBMiner* data mining system to mine different kinds of knowledge, such as association, characteristic, discriminant and classification rules, from relational databases [Han et al. 1996].

Meo et al. [1996] proposed the `MINE RULE` operator, which extends the semantics of association rule mining allowing the specification of rule forms. The operator mines association rules from the attributes of a relation and stores each discovered rule based on a minimum support and confidence into a new relation produced by the operator execution.

The `MineSQL` language expresses rule queries and aids the user in rule generation, storage and retrieval, using the new statement `MINE` and a new data type `RULE` [Morzy and Zakrzewicz 1997]. The language allows users to express different types of queries to discover association, characteristics, classification and discriminant rules [Morzy and Zakrzewicz 1997].

Imielinski and Virmani [1999] proposed `MSQL` (Mining SQL), which is the language used in the *DataMine* system. In this rule-based query language, the authors include the `GetRules` and the `SelectRule` commands in the SQL language to generate propositional rules from the database as well as to retrieve rules from the database, which is created to store some of the previously generated rules [Imielinski and Virmani 1999].

Pereira et al. [2003] implemented the primitives `ASSOCIATOR TO` and `EXTRACT IN` in the selection clause, which generates and stores in a new relation every attribute subset whose size is defined by a user parameter, and extracts into another relation the attribute values that are in a pattern set, respectively. The authors also unified these primitives in a unique SQL operator, called `DESCRIBE ASSOCIATION RULES`, to efficiently support association-based tasks [Pereira et al. 2003].

In Netz et al. [2001], the authors presented `OLE DB for DM` (OLE DB for Data Mining), an interface for Microsoft SQL Server, which is an API (Application Program Interface) that provides a `DMX` (Data Mining Extensions) language to create, modify and work with data mining models, in a syntax similar to SQL. This language is a major step towards the standardization of data mining language primitives [Han and Kamber 2006]. In fact, after the proposal of `OLE DB for DM`, it has been implemented

as part of the Microsoft SQL Server, and was first released as a product in its 2005 version. All the authors of the original paper were since them researchers with Microsoft Research.

Our approach follows the syntax proposed by Netz et al. [2001]. However, to the best of our knowledge, no previous work has addressed the problem of using data mining to improve the quality of similarity query answers - that is, the previous approaches can only be used over traditional data. In the same way, there is no data mining language for knowledge extraction on similarity queries. In this context, this paper proposes an extension of SQL aiming at mining complex data, generating association rules and using them to improve the quality and speed of similarity queries.

The remainder of this paper is structured as follows. Section 2 presents the background and related concepts useful for this work. Section 3 describes its main contribution, showing both how to extend a similarity-enabled query processor to embody data mining algorithms to help in query rewriting and optimization as well as how to extend SQL to allow the users to define data mining models to improve the query quality. Section 4 reports experiments on real datasets obtained from the web-based Flickr image repository, showing that the quality of similarity queries are much improved by the proposed technique. Finally, Section 5 concludes the paper.

2. BACKGROUND

The extension of traditional RDBMS to support similarity queries makes sense only if it can support both complex and traditional data. In order to enable similarity queries in traditional RDBMS, Barioni et al. [2009] proposed SIREN, a similarity retrieval engine that evaluates query commands expressed in an extension of SQL to support similarity queries and executing them.

SIREN is a service implemented between a RDBMS and the application software, which intercepts and analyzes every SQL command sent from an application, treating the similarity-related constructions and references to complex data. When similarity-related constructions are found, SIREN executes the similarity-related operations and rewrites the command, calling the underlying RDBMS to execute the traditional operations. On the other hand, if the command has neither similarity-related operations nor complex data, SIREN is transparent and the command is directly relied to be executed in the RDBMS [Barioni et al. 2009].

Figure 1 shows the SIREN's architecture, whose modules are analogous to those of a RDBMS [Barioni et al. 2009]. The Query Compiler module performs the lexical and syntactical analysis [Barioni et al. 2006]. The Query Optimizer module generates alternative query plans employing traditional and similarity-based algebraic rules to rewrite them [Ferreira et al. 2009]; it evaluates each plan to estimate its execution cost using suitable selectivity and cost models [Baioco et al. 2007] and chooses the plan with the minimum expected computational cost to be executed. The SIREN Feature Extractors module encompasses a set of feature extraction algorithms employed to extract the features used to represent and to compare complex data. The SIREN Indexer module deals with appropriate index structures, called metric access methods, developed to answer similarity queries [Traina et al. 2002].

One way to integrate traditional RDBMS and data mining techniques is to incorporate mining tools into the database engine, in a way that extends both the SQL language to include new syntactical constructions to handle data mining operations, and the catalog of the underlying data mining model supported. In this integration schema, called a *tight coupling*, the data mining subsystem is treated as one functional component of the RDBMS, which performs several other functions, such as query processing, indexing schemes, the storage requirements and management issues affected by the mining techniques [Thuraisingham 1999] [Han and Kamber 2006]. In order to discover knowledge from complex data, traditional data mining techniques and algorithms should be extended to operate over these data directly.

Association rule mining is one of the most important tasks in the data mining field, and it has

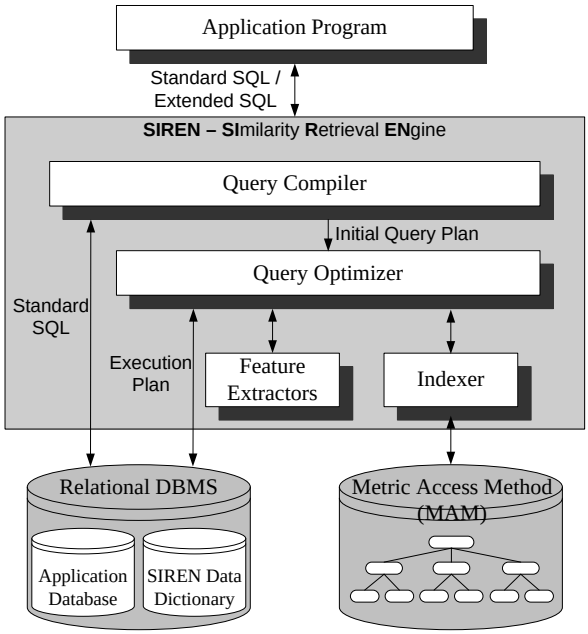

Fig. 1. SIREN architecture.

been extensively studied and applied to market basket analysis. The problem of mining association rules was firstly stated in Agrawal et al. [1993], as follows. Let $I = \{i_1, \ldots, i_n\}$ be a set of data items (stored as attribute values). A set $X \in I$ is called an itemset. Let R be a relation with transactions t involving elements that are subsets of I. An association rule is an expression of the form $X \rightarrow Y$, where X and Y are itemsets. X is called the body or antecedent of the rule. Y is called the head or consequent of the rule. The *support* is the ratio between the number of transactions of R containing the itemset $X \cup Y$ and the total number of transactions of R. *Confidence* is the fraction of the number of transactions containing X that also contain Y. The problem of mining association rules, as it was firstly stated, consists of finding association rules that satisfy the restrictions of minimum support sup_{min} and confidence $conf_{min}$ specified by the user.

Mining association rules from complex data is a more complex process than traditional ones. Although in this paper we are focusing on a extension of association rule mining algorithms to mine image datasets, the same concepts can be employed to extract rules from other types of complex data [Jiang et al. 2008], as long as adequate feature extractors are employed.

Content-based image retrieval (CBIR) systems use information extracted from images to search for similar images in the visual feature space. Their current development state and trends are discussed in [Datta et al. 2008]. Image mining employs image processing algorithms to extract relevant features from the images, organizing them into feature vectors [Ribeiro et al. 2008]. The feature vectors are employed in place of the images to represent them as transactions, and therefore they are the key information about the images to be handled in the association rule mining process. As feature vectors used to represent images have a large amount of elements, the "curse of dimensionality" problem arises and the significance of each feature decreases. In order to discretize the continuous values that compose the feature vectors representing each image, and also to avoid the curse of dimensionality, Ribeiro et al. [2008] proposed an algorithm called Omega. Omega is a supervised algorithm that prepares the feature vectors for the association rules mining, performing simultaneously data discretization and feature selection. It pre-processes complex data for subsequent association rule mining. The rules are mined employing the well-known Apriori [Agrawal and Srikant 1994] algorithm over the pre-processed data, restricting the rule bodies to be composed of feature indexes and the corresponding intervals,

and the head to be composed only of an image class. Minimum confidence values are set to be high (greater than 97%).

The format of the association rules to be mined follows.

$$f_{r_1}[l_{1_0} - l_{1_1}], \ldots, f_{r_n}[l_{n_0} - l_{n_1}] \rightarrow Class_{R_1}, \ldots, Class_{R_m} \quad (sup, conf)$$

A rule indicates that the images having the features $f_{r_1}, ..., f_{r_n}$, respectively in the closed intervals $[l_{1_0} - l_{1_1}], ..., [l_{n_0} - l_{n_1}]$ tend to be in classes $Class_{R_1}, ..., Class_{R_m}$. The number of features in the body is $1 \leq n \leq max_{feature}$, where $max_{feature}$ is the largest amount of features that can be extracted by the corresponding extractor. The number of classes in the head is $1 \leq m \leq max_{class}$, where max_{class} is the number of classes found in the relation. The values sup and $conf$ indicate, respectively, the support and the confidence for the rule. The support is the fraction of the images that satisfies the rules. The confidence is the fraction of the images containing the body (antecedent) of the rule that also contains the head (consequent).

In this work, we incorporate association rule mining in the similarity query processing. To allow association rule mining over any kind of attributes, including the continuous ones, we couple SIREN, association rule mining algorithms and Omega. As the SIREN architecture is analogous to those of the RDBMS, the incorporation of Omega and Apriori mining algorithms was done in the SIREN query optimizer module instead of on the RDBMS one. Therefore, both SIREN and the RDBMS continue working in the same way, and SIREN remains transparent for the traditional operations, handling only similarity-related operations. Moreover, in this paper, we use the association rule mining techniques presented above to rewrite similarity queries and to make them more semantically adequate to meet the users' expectation.

3. THE PROPOSED METHOD: ADDING ASSOCIATION RULE MINING IN SIMILARITY-EXTENDED SQL

SIREN is a similarity retrieval engine working on top of RDBMSs that extends SQL to enable expressing and evaluating similarity queries. However, to answer similarity queries targeting images that meet the users' expectations, where users' expectations are expressed as a set of examples, like illustrated in Section 1, two new resources must be developed: first, SIREN needs to be expanded to execute data mining processes, whose results can be used to rewrite similarity queries; and second, the SQL extension must have new commands added to allow expressing how to use the data mining processes. In this section, we describe how both resources can be developed.

3.1 SIREN Data Mining Module

As described in Section 2, one way to seamlessly integrate RDBMS and data mining techniques is to incorporate mining tools into the database engine. Accordingly, we extended the SIREN query optimizer module to tightly couple data mining algorithms, thus allowing similarity queries to benefit from automatically mined rules, improving both query quality and performance.

The SIREN query optimizer module is based on query rewriting (see Figure 2), which rewrites similarity queries using similarity algebra [Ferreira et al. 2009]. The optimizer explores several algebraically equivalent alternative plan, which holds equivalent results but with different execution costs, choosing the one with the minimum expected cost. The part of the queries corresponding to non-similarity based operations are relied to the underlying RDBMS.

The data mining module is developed as a new functional component of SIREN (see Figure 2). It extracts knowledge from the multimedia database, and generates mining rules, correlating semantic information from the textual description to the low-level extracted features. In this paper, we implemented the Apriori and Omega algorithms in the SIREN data mining module. Taking advantage of

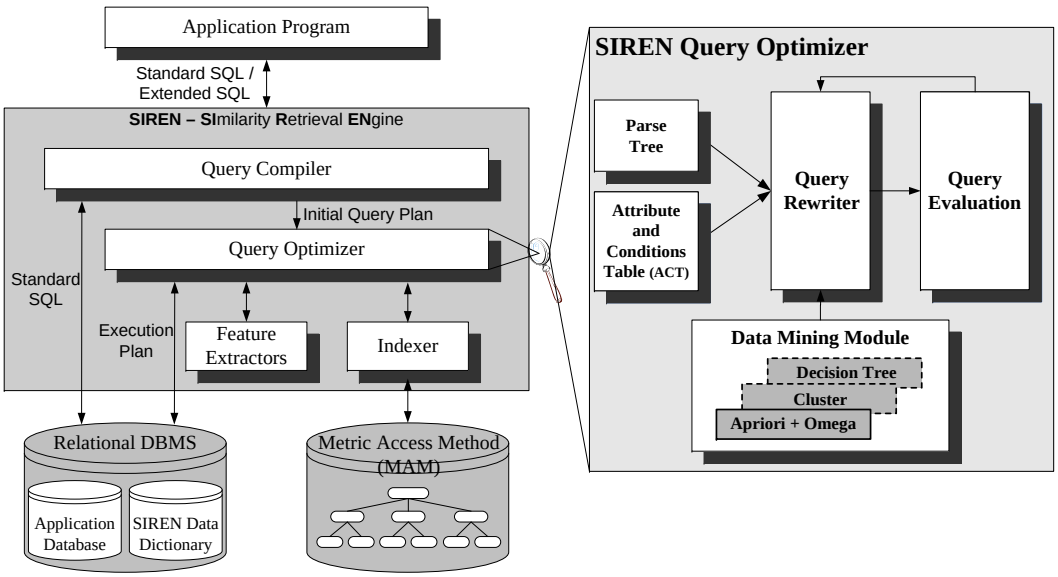

Fig. 2. SIREN optimizer architecture with the data mining module.

the similarity algebra already supported by the SIREN query rewriter, the rules are used by SIREN to rewrite the queries. Although the similarity algebra allows SIREN to obtain several equivalent plans (every one obtaining the same result, perhaps at a distinct computational cost), applying the mined rules usually leads to distinct results. In fact, the idea is to mine the data in order to either simplify the query execution, enabling to find approximate answers faster, or to better follow the users' expectations, enabling to find better answers. Thus, by combining the similarity algebra and data mining techniques, SIREN is able to improve both the efficiency and the efficacy of similarity query answering.

The data mining module acts in a way similar to the indexer module: whenever a new data mining model is created, the associated relation is processed following the model definition, mining the corresponding rules. Thereafter, whenever a similarity query is posed, the optimizer uses the rules to evaluate if any of them can be used either to improve the query quality (better tailoring the query execution to the mined user expectations), or to improve the query execution efficiency (using the rules as screening predicates to prune part of the data that is known to have no interesting data).

3.2 Defining the Data Mining Model

As presented in Section 1, some authors proposed SQL-based language extensions for data mining. However, none of them extracts knowledge from complex data – that is, they mine only traditional data. In this paper, we show how to extend SQL to allow expressing data mining tasks in a way that they can be executed over complex data. Moreover, we focus on association rules mined over images, and our approach also allows that other data mining algorithms can also be employed to extract rules, either from images or from other complex data type as long as adequate feature extractors are available.

One problem occurring when association rule mining is included in similarity queries is how to define data mining models in SQL. A data mining model provides the specification of particular data structures, which are stored in the database catalog, constraining the data sets associated with these structures. Thus, the syntax of the specification commands should follow the `DDL` (Data Definition Language) command style. Following that style, we propose to include two commands in SQL to handle a Data Mining Model (DMM): the `CREATE MINING MODEL` and the `DROP MINING MODEL` com-

mands. A new data mining model is defined by the CREATE MINING MODEL statement. The syntax in EBNF (Extended Backus-Naur Form) notation follows.

```
<create_mining_model_statement>::= CREATE MINING MODEL <model_name>
                                   ON <relation_name> (<attrib_name>)
                                   [WHERE <predicate>]
                                   [METRIC <metric_name>]
                                   USING <data_mining_alg_name> [(<parameter_list>)]
```

The syntax is similar to the traditional CREATE INDEX syntax. The CREATE MINING MODEL command creates a new data mining model called <model_name> over the complex or traditional attribute (<attrib_name>) of the relation (<relation_name>), based on the data mining algorithm (<data_mining_alg_name>). The <model_name> parameter must be unique. The optional clause METRIC specifies which metric (<metric_name>) associated to the complex attribute will be used to create the data mining model. The optional clause WHERE <predicate> allows defining that only the elements in the relation that meets the specified <predicate> are employed to evaluate the mining algorithm.

Each data mining algorithm indicated in the <data_mining_alg_name> clause must be individually developed and integrated to the SIREN data mining module (including a new algorithm requires recompiling SIREN). In this paper, we describe only the Apriori and the Omega algorithms, both already implemented as a coupled, single <data_mining_alg_name> algorithm. The parameters for the data mining algorithm are optional and depends on the particular algorithm specified. All required parameters are included in the <parameter_list> in the <data_mining_alg_name> clause. Whenever the <parameter_list> of an algorithm is modified, a new data mining model can be created. In this way, it is possible to execute the composition between different models. Updating the 'training' database does not automatically changes the existing DMM: an explicit SET MODIFICATION UPDATE command (which is explained in Section 3.3) must be issued to re-evaluate the rules.

The syntax to use the Apriori and Omega algorithms in the CREATE MINING MODEL statement is as follows. For other data mining algorithms, their meaning are different, but the syntax is similar.

```
<data_mining_alg_name>::= APRIORI(<attribute_class>, <nclass>, <sup>, <conf>
                                  [, <minint>, <maxmrgint>, <maxkpatt>])
```

Apriori and Omega algorithms have both required and optional parameters. The required parameters are: the attribute employed to classify the images (<attribute_class>), the number of classes (<nclass>), and the value of minimum support (<sup>) and minimum confidence (<conf>) for the association rule mining algorithm. The optional parameters are those employed to tune the Omega algorithm: the minimum interval size (<minint>), the maximum acceptable inconsistence to merge consecutive intervals (<maxmrgint>), and the maximum acceptable inconsistence to retain an attribute (<maxkpatt>), as described in Ribeiro et al. [2008].

As a running example to illustrate the proposed commands, suppose that the user has a database of images already tagged with a set of keywords describing the images. Therefore, whenever a new image is stored in the database, the user tags the image with a number of keywords. Several images can be assigned to a given keyword, as well as each image can have several keywords assigned. In the experiments section, we show queries posed against a database like this, extracted from the Flickr website[1]. We are employing Flickr as a good example of the applicability of our approach.

A relation useful for this example can be defined by the following command:

[1]The Flickr website is available at <http://www.flickr.com/>

Example 1

```
CREATE TABLE WondersWorld AS (
          ImagId INTEGER,
          Tag VARCHAR(50),
          Description VARCHAR(120),
           . . . -- Other attributes in general
          Training CHAR,
          Image STILLIMAGE,
          METRIC (Image) USING (Histogram, Texture DEFAULT));
```

The important attributes in this relation are: the `Tag` attribute, that stores the tags assigned to each image; the `Training` attribute, used to select the subset of images to be used to train the association rule miner (as it will be described later); and the `Image` attribute, a complex attribute that, in this example, stores the images, specified with one of the complex data types defined in the similarity-enabled SQL extension available in SIREN. The `METRIC` clause is also a SIREN extension, that here enables `Image` attributes to be compared using metrics based on either (color) `Histogram` or `Texture`, the latter being the default one. That is, as more than one metric is defined, the default one is used in a query, unless another is explicitly stated. In this paper, the metric `Texture` is used in every example.

For instance, let us use an image database obtained from the Flickr website. Suppose that this database stores several photos together with their corresponding information, such as the tags and other descriptions from the fourteen wonders, seven of the ancient and seven of the new world, plus the complex of Giza pyramid, the last remaining wonder of the ancient world. Suppose also that the user is only interested in photos that share a specific characteristic, for which he/she has an initial training subset already marked in an attribute of the relation. Now, the user must create a data mining model to evaluate its training subset. Assuming that texture is adequate to discriminate among images that the user is interested in or not, the following command can be issued.

Example 2

```
CREATE MINING MODEL WondersWorldAssociationRulesModel
   ON WondersWorld (Image) METRIC texture
   WHERE Training= 'True'
   USING APRIORI (Tag, 15, 1, 100, 2, 0.1, 0.42);
```

This command creates a data mining model using the Apriori algorithm to extract association rules from the `Image` attribute of `WondersWorld` relation, which has its parameters specified following the `APRIORI` keyword. The Apriori algorithm processes only the subset of tuples from the relation where the attribute `Training` has the value `'True'`, thus the corresponding subsets of `Images` are `Tags` corresponding to the training subset. In Example 2, the Apriori algorithm is executed with a minimum of 1% as the threshold for support and 100% for confidence. The attribute used to classify the `WondersWorld` relation is `Tag`, and this relation has 15 classes, that is, seven of new and ancient wonders world plus the complex of Giza. The Omega algorithm is executed using a minimal threshold for interval size of 2, a maximum threshold for merging consecutive intervals of 0.1 and a maximum threshold for keeping an attribute of 0.42. Those thresholds were defined following Omega recommendations [Ribeiro et al. 2008]. Using our evaluation database, this data mining model generates a total of 1,799 rules. An example of the kind of association rule mined by this data mining model follows.

```
70[836.1-877.6] 8[155.7-156.5] → ChichenItza (1.0, 100.0)
```

As mentioned in Section 2, the rule body is composed of feature indexes and their corresponding intervals. The rule shown indicates that images having the 70^{th} and the 8^{th} features with values in

the closed intervals $[836.1 - 877.6]$ and $[155.7 - 156.5]$, respectively, tend to be images of `ChichenItza`. The values 1.0 and 100.0 indicate, respectively, the support and the confidence values of the rule. The support of 1.0 indicates that 1% of the training images (four images) have the 70^{th} and the 8^{th} feature values in $[836.1 - 877.6]$ and $[155.7 - 156.5]$, respectively, and all of them are in the class `ChichenItza`. The confidence of 100.0 indicates that 100% of the training images which has 70^{th} and the 8^{th} feature values in intervals $[836.1 - 877.6]$ and $[155.7 - 156.5]$, respectively, are in the class `ChichenItza`.

An existing data mining model can be dropped with the `DROP MINING MODEL` statement:

```
<drop_mining_model_statement> ::= DROP MINING MODEL <model_name>,
```

where `<model_name>` is the name of data mining model to be dropped. Example 3 presents the command to drop the data mining model defined above.

Example 3

```
DROP MINING MODEL WondersWorldAssociationRulesModel;
```

3.3 Setting Data Mining Model

When a data mining model is created, the parameters of the model are also specified and the database is processed by the specified mining algorithm to generate the corresponding rewriting rules, thus the data mining model becomes ready to be used. However, real query rewriting is not enabled until the user explicitly assigns it to his/her own query `MODIFICATION` environment. This environment embodies a set of query modifications that can be employed together to rewrite similarity queries. In order to control what data mining model are available to change similarity queries, the `SET MODIFICATION` statement must be issued. Its syntax is as follows.

```
<set_modification_statement> ::= SET MODIFICATION [ADD | REMOVE | UPDATE]
                                      [ALL | <model_name>]
```

The `SET MODIFICATION` statement is used both to enable or disable using a specified data mining model in the user's `MODIFICATION` environment (using the `ADD` or `REMOVE` clauses), or to update existing data mining models (using the `UPDATE` clause). When `SET MODIFICATION ADD <model_name>` is issued, the current set of rules mined by the `<model_name>` model is added to the user's `MODIFICATION` environment. When `SET MODIFICATION UPDATE <model_name>` is issued, the data mining model associated to the `<model_name>` model is re-evaluated (for example, due to changing data in the relation or due to updates in the `<predicate>` attribute of the relation). When `SET MODIFICATION REMOVE <model_name>` is issued, the current set of rules mined by the `<model_name>` model is removed from the user's `MODIFICATION` environment. The `ALL` option is used to add, remove or update all set of rules mined by the user in the user's `MODIFICATION` environment.

Whenever there are rules enabled in the user's `MODIFICATION` environment, every query issued are rewritten following them. Therefore, the `SET MODIFICATION` command allows the user to control when queries should be modified, and which rules must be employed to modify each query. Whenever the user adds two or more models in his/her `MODIFICATION` environment, the SIREN query rewriter uses a conjunction of the models to rewrite the queries. Figure 3 shows a data flow of how the two commands `CREATE MINING MODEL` and `SET MODIFICATION` respectively prepares and enables the rules to be employed for query rewriting.

Example 4 adds the data mining model defined in Example 2 into the user's `MODIFICATION` environment, enabling every subsequent similarity query to be rewritten according to the rules mined by the corresponding data mining model.

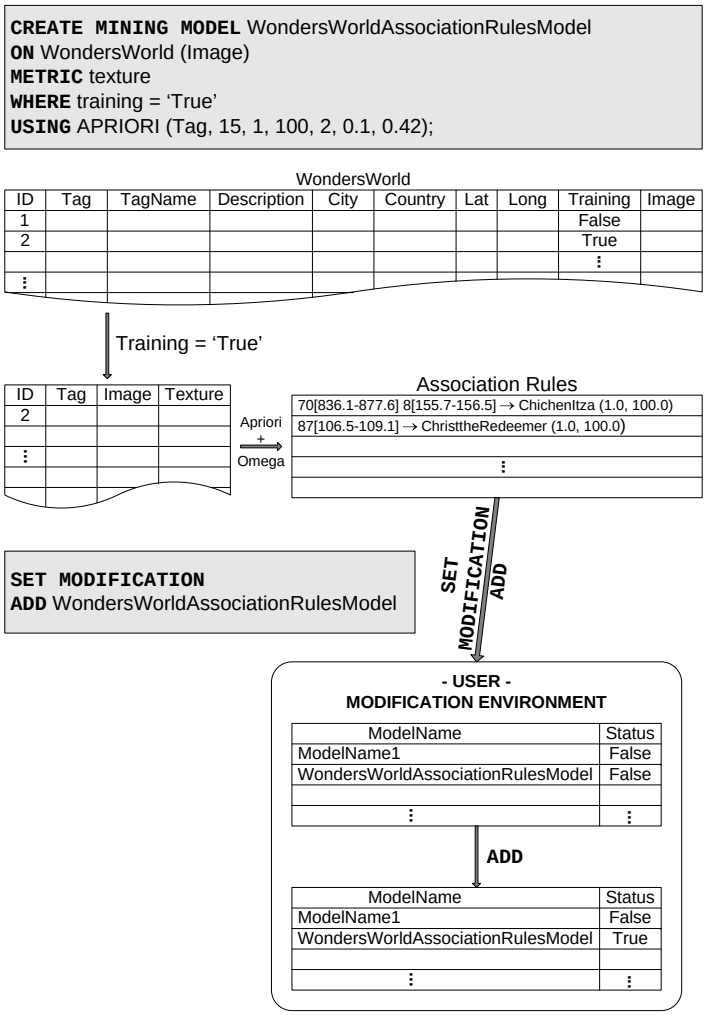

Fig. 3. Data flow showing how the mined rules are enabled using the `CREATE MINING MODEL` and `SET MODIFICATION` commands.

Example 4

```
SET MODIFICATION ADD WondersWorldAssociationRulesModel;
```

Whenever the user poses a similarity query after this command has been executed, the query is first rewritten to take into account the rules mined by the data mining models enabled in the user's `MODIFICATION` environment. Thereafter, queries like the one presented in Section 1 can be posed by the user. Assuming that the `WondersWorld` relation was defined and loaded, the query stated in that section is equivalent to: "*Find images in the WondersWorld relation that are similar to X and are like my training images*", in this case assuming that the training images are "from the nature".

Figure 4 summarizes the whole process of preparing and posing queries either using or not using the mined rules, over a database already loaded with images. In the first step (Figure 4, Actions ①), the user sends interesting images to the Omega and Apriori algorithms. In our running example, they are the images having the `Training` attribute set to `'True'`. The association rules are mined and stored, together with the interesting images, in the database. Notice that the interesting images must be previously stored in the database, possibly together with any other images, regardless of them being interesting or not. The interesting images can be stored in the same relation or in any other relation

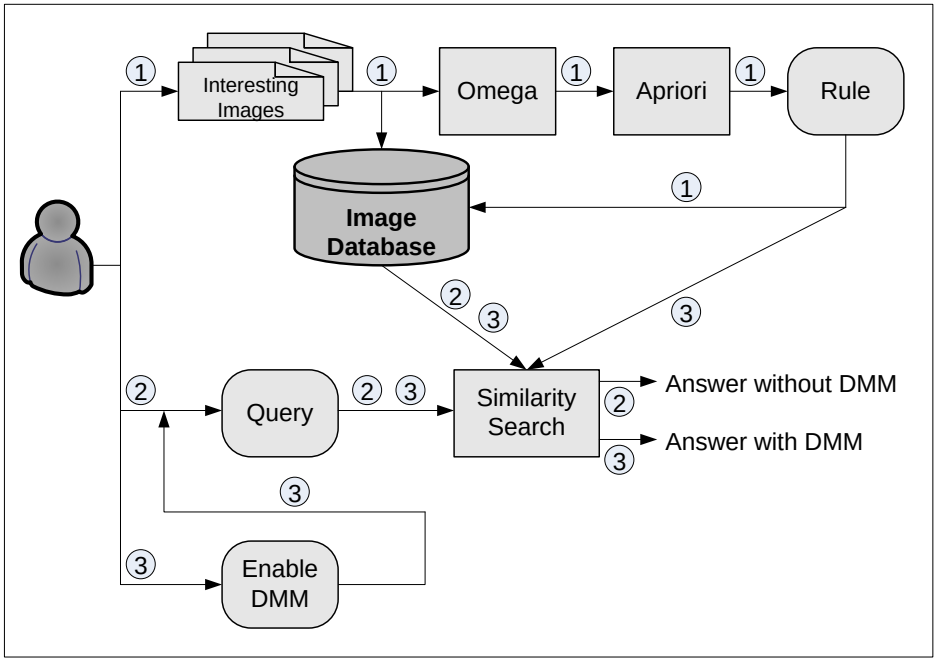

Fig. 4. Processes to prepare and execute similarity queries taking care of data mining models taking int account the users' expectations.

having the same attribute structure; in both cases, the attribute `Training` is used to identify which image must be used to generate the rules. After retrieved, the rules are also stored in the database, as part of the system catalog. Notice that a DMM is attached to the `METRIC <metric_name>` employed to search the database, thus it can be applied to any attribute in any relation sharing the same image domain and this the same rules mined. When searching for similarity with DMM disabled (Figure 4, Actions ②), the "similarity search" engine uses only the image database to answer the query; otherwise, when the DMM is enabled (Figure 4, Actions ③), the "similarity search" engine uses the rules together with the image database to evaluate the queries.

4. EXPERIMENTAL EVALUATION

In this section, we present experiments comparing SIREN executing a similarity query both using association rules to rewrite queries and not using them. SIREN and its data mining module are implemented in C++, and the experiments were evaluated using an Intel Core 2 Quad 2.83GHz processor with 4GB of main memory, under the Windows XP operational system. To evaluate the efficiency and efficacy of our technique, we employed the following measures: the time spent to execute the queries and the quality of the answer. The RDBMS used to process the traditional part of the query was Oracle 9i.

The experiments were performed using a database created with 1,798 images extracted from the Flickr website, together with the following information: tags, a short description, the city with the corresponding latitude and longitude and the image. They are images from the fourteen wonders, seven of the ancient and seven of the new world, plus the complex of Giza pyramid, the nowadays remaining wonder from the ancient world. The relation is created using the command shown in Example 1. There are 100 images from each of the fourteen distinct wonders and 100 more for the current Giza pyramids, retrieved from Flickr, together with the descriptions that people in general stores in that website. No structure or consistency are expected to exist in this tagging system. From the 1,798

images, 1,500 had the attribute `Training` set to `'False'`. The remaining 298 are images from all of the fourteen wonders and the current Giza pyramids, all of them have the attribute `Training` set to `'True'`, and correspond to examples of the kind of images that the users expects to obtain from each wonder. It is guaranteed that there are at least 20 images of each fifteen wonders that meet the user's expectations, so they were employed to train the DMM.

When the command shown in Example 2 is issued, the 298 images that express the users' expectations (`Training = 'True'`) are retrieved from the `WondersWorld` relation and submitted to the Omega and Apriori algorithms. They process the images and generate the rules to be employed by the query rewriter and optimization modules to process further queries. A total of 1,799 rules were generated. The number of images submitted for training should not be very large, as the Apriori and Omega algorithms do not scale well. Therefore, we suggest selecting from 20 to 200 images of each class as an amount adequate to create the rules. We claim that these figures are adequate for practical purposes, both because it was experimentally found as enough to generate useful rules, and because users seldom have many images already classified. For the example shown, it takes in average 460 milliseconds to create a DMM. It is worth remembering that creating a DMM is performed only once, and further queries that use it are not affected by this number. Thereafter, the rules generated by a DMM can be applied over very large image sets, so although using a small amount of training images, our technique is scalable to very large datasets.

Just executing the `CREATE MINING MODEL` command does not enable SIREN to modify the queries (remember that the user must enable query rewriting using the `SET MODIFICATION ADD` command). Therefore, initially it was submitted several k-nearest neighbor queries, such as the following:

Example 5

```
SELECT ID, Image
    FROM WondersWorld
    WHERE Image NEAR 'C:\ChichenItza121.jpg' STOP AFTER k;
```

where k is the number of images to be retrieved, which varies from 1 to 20 for each query center. Ten queries with distinct query centers were posed for each value of k.

Following, it was submitted several range queries like the following one:

Example 6

```
SELECT ID, Image
    FROM WondersWorld
    WHERE Image NEAR 'C:\ChristtheRedeemer09.jpg' RANGE r;
```

where r is the range radius of images to be retrieved, varying from 0.1 to 0.8 for each query center. Again, ten queries with distinct query centers were posed for each value of r.

Finally, the DMM-driven query rewriting was enabled (issuing the command shown in Example 4) and the same sets of kNN and range queries were issued again. Figure 5 shows an example of ten images that have the `Training` attribute set to `'True'`, that is, some of the images submitted to the association rule mining algorithms Apriori and Omega. The figure shows also examples of the results obtained when the Example 5 is executed both with DMM disabled and enabled, using $k = 10$ over the test database (that is, over the full database regardless of the `Training` attribute setting).

Table I shows the average values obtained for the range queries and Table II shows the average values obtained for the k-nearest neighbor queries. Both tables show the average number of relevant images obtained, the average number of non-relevant images obtained and the average time (in milliseconds) to execute one corresponding query, varying values of range r in the range queries, and varying values of k in the kNN queries. The same sets of measurements were performed for both kinds of queries,

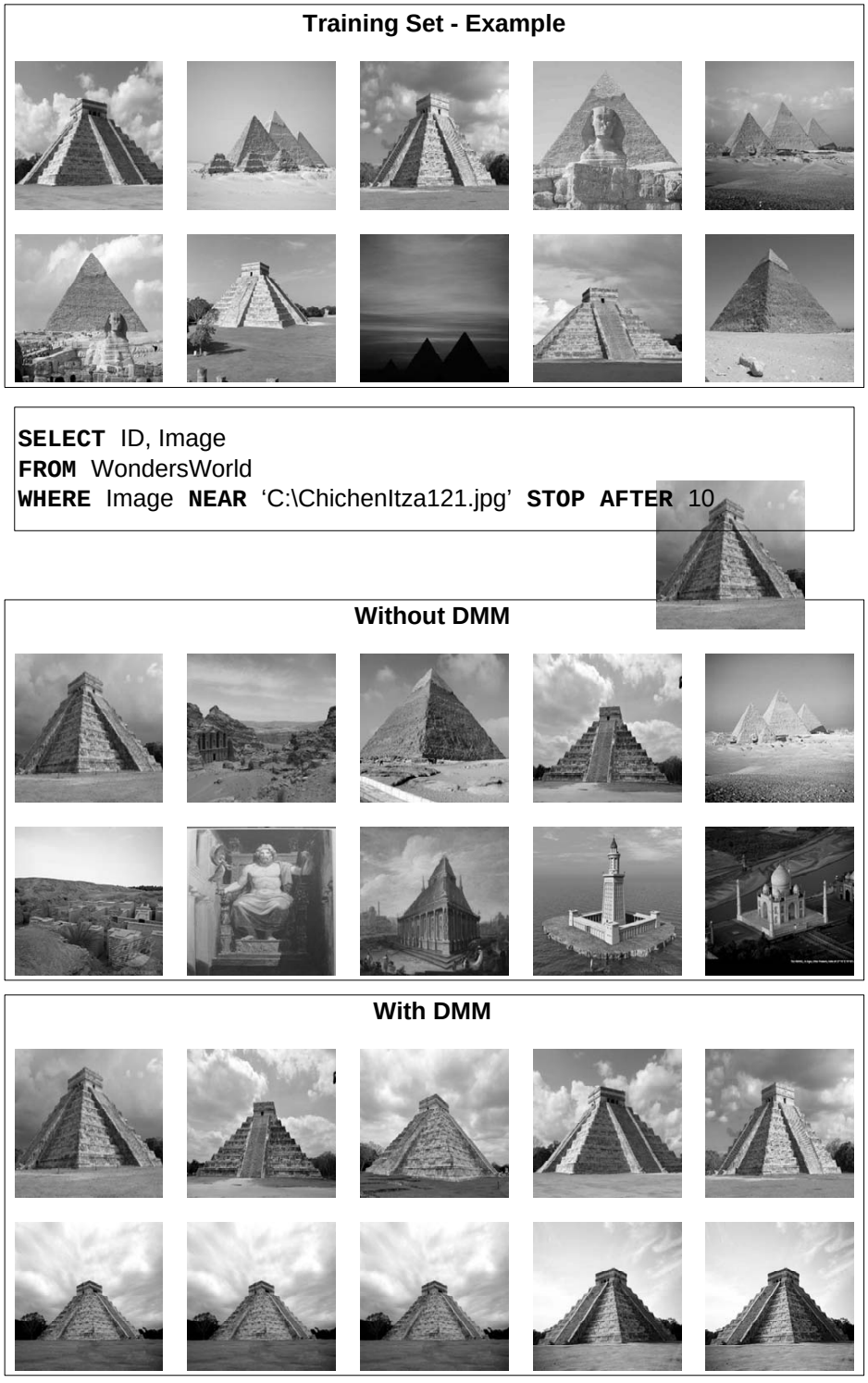

Fig. 5. Results of 10-NN over Example 5 enabling and disabling the use of a DMM. The training set example was obtained from the `WondersWorld` relation with the attribute `Training` = 'True' and used to generate the rules.

enabling the query rewriting using the rules mined by the data mining model (the values shown as "With DMM" in both tables) and disabling them (the values shown as "Without DMM" in both tables). "Without DMM" corresponds to the plain execution of the traditional similarity query, that is, without DMM rule-based query rewriting.

Analyzing the range query results shown in Table I, it can be seen that for the same range radius, with the query rewrite disabled, SIREN returns several images in which the user is not interested in. For example, with a range radius $r = 0.2$ it retrieves only 3 interesting images but 22 non-interesting; with a range radius $r = 0.8$ all the 20 interesting images are returned, but other 188 non-interesting images are retrieved together. When the query rewrite is enabled, for the same range radius SIREN always returns just the interesting images within the given distance. Table I also shows that the time required to execute queries for both enabling or disabling the query rewriting does not change significantly. This is due to the fact that the query processing in SIREN is always fast, and the communication between SIREN and Oracle take the most significant time.

Table I. Results from several range values - range queries (average)

Average Range	Without DMM			With DMM		
	Relevant	Non-Relevant	Time (ms)	Relevant	Non-Relevant	Time (ms)
0.1	1	0	47	1	0	42
0.2	3	22	47	3	0	47
0.3	7	54	47	7	0	47
0.4	12	85	52	12	0	52
0.5	15	113	47	15	0	47
0.6	19	141	47	19	0	47
0.7	19	165	47	19	0	47
0.8	20	188	47	20	0	47

Analyzing the k-nearest neighbor query results shown in Table II, it can be seen that, again, for the same number of relevant images retrieved with the query rewrite disabled, SIREN also returns several images that are not relevant to the user. It is important to remember that kNN and traditional predicates are not commutative. Therefore, if the user asks for a kNN query with the option for query rewriting disabled, it will be returned k images, although possibly not every image will be interesting, therefore the number of interesting images returned is at most k, but it is often less than k relevant images. To perform this experiment with the query rewritten disabled, we repeated the experiment with higher values of k, until the desired number of interesting images were obtained. Column "Non-relevant" from the "Without DMM" experiment reports the average number of k required to obtain the corresponding number of interesting images.

Table II shows, for example, that to retrieve 2 interesting images, an average of 12 images including non-interesting ones should be asked for; that is, the kNN must be issued asking for, at least, $k = 12$. To retrieve all the 20 interesting images, it should be asked for an average of 215 images. When the query rewrite is enabled, only the required number of images must be effectively asked, that is, k can be set exactly to the desired value. As it occurs regarding range queries, the time to process both queries are equivalent.

5. CONCLUSION

In this paper we presented a novel technique to improve the quality of similarity queries over image databases, in order to obtain answers that better fulfills the users' expectations. Our technique also provides an intuitive way for the user to express what he/she intends to obtain, as the users must just submit a small subset of images as examples of what they consider interesting. The subset can be selected from the image dataset using predicates using any traditional attribute assigned to the images. For example, it can be just marked as interesting or not interesting in a boolean attribute.

Table II. Results from several k values - kNN queries (average)

Average kNN	Without DMM			With DMM		
	Relevant	Non-Relevant	Time (ms)	Relevant	Non-Relevant	Time (ms)
1	1	3	47	1	0	42
2	2	12	42	2	0	42
3	3	22	47	3	0	47
4	4	27	37	4	0	47
5	5	30	42	5	0	47
6	6	33	47	6	0	47
7	7	38	47	7	0	47
8	8	46	47	8	0	47
9	9	58	41	9	0	41
10	10	67	47	10	0	42
11	11	71	42	11	0	47
12	12	75	42	12	0	47
13	13	85	47	13	0	47
14	14	98	47	14	0	42
15	15	108	36	15	0	47
16	16	118	42	16	0	47
17	17	123	47	17	0	47
18	18	127	42	18	0	47
19	19	141	42	19	0	47
20	20	215	42	20	0	47

The training images are submitted to train an association rule mining algorithm - in this paper we used the Apriori together with the Omega algorithm to discretize the usually continuous values of features automatically extracted from the images (texture attributes were used in the reported example).

After mining the rules of what the user regards as interesting/important, similarity queries can be posed in the form "*Find images similar to this given one and that are like those of my preference*". The experiments performed using the SIREN prototype show that a training set of only 298 images was enough to return correct answers for similarity queries performed over an image dataset with 1,798 images divided into 15 different subjects.

Our approach uses association rule mining integrated with the query processor of an interpreter a similarity-extended version of SQL. The proposed technique uses the mined rules to rewrite the similarity queries posed, which are thereafter optimized by the query optimizer of the similarity-enabled query engine. Nonetheless, only two new commands are required for the language extension, which provides respectively a simple way to specify data mining models, and a way for the user to control when to enable or disable the use of the mined rules. Despite having a small impact on the language syntax, using only two new commands have a large impact on the query results, as it can affect many query constructs — in fact, any command including a similarity predicate referring to the affected attribute can have its results improved. Our approach is also non-invasive to the user, in the sense that the user can enable or disable the query rewriting, thus allowing the queries to retrieve just the images that meet the user's interest, or allowing just the plain similarity query to be executed. It also allows non-expert users to take advantage of the query rewritten resource, as the user does not need to change the query by himself/herself: the system is able to perform the rule-based query optimizations previously tuned by a domain specialist, without any further user's intervention.

REFERENCES

Agrawal, R., Imielinski, T., and Swami, A. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data Conference*. Washington, D.C., United States, pp. 207–216, 1993.

Agrawal, R. and Srikant, R. Fast algorithms for mining association rules. In *Proceedings of the International Conference on Very Large Data Bases*. Santiago de Chile, Chile, pp. 487–499, 1994.

Baioco, G. B., Traina, A. J. M., and Traina, Caetano, J. MAMCost: Global and local estimates leading to robust cost estimation of similarity queries. In *Proceedings of the International Conference on Scientific and Statistical Databases Management*. Banff, Canada, pp. 6–16, 2007.

Barioni, M. C. N., Razente, H. L., Traina, A. J. M., and Traina, Caetano, J. SIREN: A similarity retrieval engine for complex data. In *Proceedings of the International Conference on Very Large Data Bases*. Seoul, South 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.

Datta, R., Joshi, D., Li, J., and Wang, J. Z. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys* 40 (2): Article 5(1–60), 2008.

Faloutsos, C. Indexing of multimedia data. In *Multimedia Databases in Perspective*. New York, NY, United States, pp. 219–245, 1997.

Ferreira, M. R. P., Traina, A. J. M., Dias, I., Chbeir, R., and Traina Jr., C. Identifying algebraic properties to support optimization of unary similarity queries. In *Proceedings of the Alberto Mendelzon International Workshop on Foundations of Data Management*. Arequipa, Peru, pp. 1–10, 2009.

Han, J., Fu, Y., Wang, W., Chiang, J., Gong, W., Koperski, K., Li, D., Lu, Y., Rajan, A., Stefanovic, N., Xia, B., and Zaiane, O. R. DBMiner: A system for mining knowledge in large relational databases. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Portland, OR, United States, pp. 250–255, 1996.

Han, J. and Kamber, M. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2006.

Imielinski, T. and Mannila, H. A database perspective on knowledge discovery. *Communications of the ACM* 39 (11): 58–64, 1996.

Imielinski, T. and Virmani, A. MSQL: A query language for database mining. *Data Mining and Knowledge Discovery* 3 (4): 373–408, 1999.

Jiang, B., Pei, J., Lin, X., Cheung, D. W., and Han, J. Mining preferences from superior and inferior examples. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Las Vegas, NV, United States, pp. 390–398, 2008.

Meo, R., Psaila, G., and Ceri, S. A new SQL-like operator for mining association rules. In *Proceedings of the International Conference on Very Large Data Bases*. Mumbai (Bombay), India, pp. 122–133, 1996.

Morzy, T. and Zakrzewicz, M. SQL-like language for database mining. In *Proceedings of the Symposium on Advances in Databases and Information Systems*. St. Petersburg, Russia, pp. 311–317, 1997.

Netz, A., Chaudhuri, S., Fayyad, U., and Bernhardt, J. Integrating data mining with SQL databases: OLE DB for data mining. In *Proceedings of the IEEE International Conference on Data Engineering*. Heidelberg, Germany, pp. 379–387, 2001.

Pereira, R. T., Millán, M., and Machuca, F. New algebraic operators and SQL primitives for mining association rules. In *Proceedings of the IASTED International Conference on Neural Networks and Computational Intelligence*. Cancun, Mexico, pp. 227–232, 2003.

Ribeiro, M. X., Ferreira, M. R. P., Traina Jr., C., and Traina, A. J. M. Data pre-processing: a new algorithm for feature selection and data discretization. In *Proceedings of the International Conference on Soft Computing as Transdisciplinary Science and Technology*. Cergy-Pontoise, France, pp. 252–257, 2008.

Ribeiro, M. X., Traina, A. J. M., Traina Jr., C., Rosa, N. A., and Marques, P. M. d. A. How to improve medical image diagnosis through association rules: The IDEA method. In *Proceedings of the IEEE International Symposium on Computer-Based Medical Systems*. Jyväskylä, Finland, pp. 266–271, 2008.

Thuraisingham, B. *Data Mining: Technologies, Techniques, Tools, and Trends*. CRC Press, 1999.

Traina, Caetano, J., Traina, A. J. M., Faloutsos, C., and Seeger, B. Fast indexing and visualization of metric datasets using slim-trees. *IEEE Transactions on Knowledge and Data Engineering* 14 (2): 244–260, 2002.