Information Systems 72 (2017) 117–135

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

CLAP, ACIR and SCOOP: Novel techniques for improving the performance of dynamic Metric Access Methods

Paulo H. Oliveira[a,*], Caetano Traina Jr.[a], Daniel S. Kaster[b]

[a] Department of Computer Sciences, Institute of Mathematical and Computer Sciences, Av. Trabalhador São-Carlense, 400, University of São Paulo at São Carlos, SP, Brazil
[b] Department of Computer Science, Rodovia Celso Garcia Cid, Pr 445 Km 380, University of Londrina, PR, Brazil

ARTICLE INFO

Article history:
Received 31 December 2016
Revised 22 June 2017
Accepted 10 October 2017
Available online 16 October 2017

Keywords:
Similarity queries
Metric Access Methods
Cutting Local Additional Pivots
Anticipation of Child Information

ABSTRACT

Constant technological advances in electronic devices have led to the growth of elaborated data such as large texts, time series, georeferenced imagery, genetic sequences, photos, videos and several other types of complex data. Differently from scalar, traditional data types such as numbers and strings, complex data do not present the order relation property, which allows identifying whether an element precedes another according to some criterion. Therefore, these data are usually compared by the similarity degree among them. The Metric Access Methods (MAMs) are recognized as well-suited to perform similarity queries over such kind of data more efficiently than other access methods. MAMs can be considered dynamic or static depending on the pivot type used to construct them. Pivots are often employed to narrow the search for data. Global pivots can be employed to look into elements in the whole dataset, thus they have a high impact in the process of pruning irrelevant elements, since a single global pivot can be used to discard a large amount of irrelevant elements. Nevertheless, MAMs based on global pivots may have their dynamicity compromised by the fact that eventual pivot-related updates must be propagated through the entire structure. Local pivots, on the other hand, allow the maintenance to occur locally at the price of a lower pruning ability. In this paper, we propose novel techniques for improving the performance of dynamic MAMs without harming their dynamicity, once that several applications handle online complex data and, consequently, demand efficient dynamic indexes to be successful. Specifically, our main contributions are three techniques: (i) CLAP, which consists of employing local additional pivots to reduce distance calculations; (ii) ACIR, which is combined with CLAP and anticipates information from child nodes to reduce unnecessary disk accesses; and (iii) SCOOP, which is combined with CLAP as an extended version of ACIR, anticipating a larger amount of information from child nodes. The techniques have been applied to a dynamic MAM and evaluated over real datasets ranging from moderate to high dimensionality and cardinality. The experimental results show that our techniques were able to reduce query execution time in up to 63% for point queries and up to 53% for queries retrieving multiple elements.

1. Introduction

In recent years, the technological advances in electronic devices have accelerated the generation of complex data. In this work, we use the term *complex data* to refer to data that cannot be represented by traditional types, such as numbers, characters, dates and short texts. Examples of complex data are large texts, time series, georeferenced imagery, genetic sequences, photos and videos.

The scalar data domains possess the order relation property, which allows identifying, for each pair of elements in the domain, whether one precedes the other according to some criterion. Based on this property, most of the index structures implemented in the current Relational Database Management Systems (RDBMSs) are able to efficiently perform queries. However, the order relation does not apply to most of the complex domains [1]. Since traditional index structures are based on this property, they are not suitable for complex data. Hence, the Metric Access Methods (MAMs) were developed to index complex data and to allow efficient similarity searches.

Several MAMs have been proposed, categorized in different ways depending on which factors are considered to structure the

* Corresponding author.
E-mail addresses: pholiveira@usp.br (P.H. Oliveira), caetano@icmc.usp.br (C. Traina Jr.), dskaster@uel.br (D.S. Kaster).

https://doi.org/10.1016/j.is.2017.10.003
0306-4379/© 2017 Elsevier Ltd. All rights reserved.

indexed data. These factors comprehend: response type, structure dynamicity, space partitioning and pivot type. Regarding response type, MAMs can be either exact or approximate. Approximate MAMs provide less accurate responses in favor of efficiency. As to structure dynamicity, MAMs can be either dynamic or static. Dynamic MAMs enable adding and removing elements at any time with no need for reconstruction. Static MAMs, on the other hand, require the prior existence of the whole dataset to be indexed and usually need to be reconstructed in face of updates. Considering space partitioning, the basic types include: ball partitioning [2], generalized hyperplane partitioning [2] and excluded middle partitioning [3]. Lastly, with respect to pivot type, MAMs can be based either on global or local pivots.

In this paper, pivots are elements that act as representatives of certain regions of the dataset. Their purpose is to prune irrelevant elements during query execution. It is said that a pivot is global when every element in the dataset can be referenced through it, whereas a pivot is local when only a portion of the dataset can be referenced through it. Because global pivots refers to every element in the dataset, they have a high impact in the process of pruning irrelevant elements, since a single global pivot can be used to discard large amounts of irrelevant elements. However, MAMs based on global pivots may have their dynamicity compromised by the fact that eventual pivot-related updates must be propagated through the entire structure. Local pivots, on the other hand, restrict the maintenance locally at the price of a lower pruning ability. Therefore, pivot type and structure dynamicity are directly related to each other.

In this paper, we address the challenge of improving the pruning ability of dynamic MAMs without harming their dynamicity. This is relevant for applications that manage online complex data and, consequently, demand dynamic and efficient index structures. We present novel techniques, applicable to hierarchical MAMs based on local pivots, which aim at reducing the number of distance calculations and disk accesses in similarity queries — two factors that determine the performance of MAMs. Specifically, our main contributions are as follows:

- The **CLAP** (***C****utting* ***L****ocal* ***A****dditional* ***P****ivots*) technique, which employs local additional pivots to reduce the number of distance calculations;
- The **ACIR** (***A****nticipation of* ***C****hild* ***I****nformation regarding* ***R****epresentatives*) technique, which employs CLAP and anticipates information from child nodes related to node representatives to reduce the number of unnecessary disk accesses;
- The **SCOOP** (***S****earching with* ***C****utting local pivots and informati****O****n anticipati****O****n of* ***P****ivots*) technique, which employs CLAP and anticipates information from child nodes regarding both node representatives and additional pivots.

The CLAP technique employs local additional pivots to reduce the uncertainty region in the search space. This is the region of the search space that may contain elements that are not in the answer, but nonetheless cannot be pruned until they are individually analyzed — which implies distance calculations. The ACIR and SCOOP techniques, in turn, anticipate information from child nodes into their parents to enable pruning the irrelevant elements before visiting the disk pages that actually store them. Unlike other approaches that employ multiple pivots to define regions in the search space, our approaches allow reducing distance calculations and disk accesses without impairing the index dynamicity.

We applied the CLAP, ACIR and SCOOP techniques to the MAM Slim-tree [4,5] and extensively evaluated them in a set of experiments over real datasets. The datasets vary both in the number of elements and in the number of attributes. The experiments also employed distance functions of different computational costs — from linear to quadratic. The results confirm the efficiency of the techniques, as they provided significant gains in execution time, number of distance calculations and number of disk accesses in similarity queries over all datasets evaluated. Partial results of this work regarding the CLAP and ACIR techniques have been published [6]. In this paper, we extend the anticipation-of-information mechanism, resulting in the SCOOP technique; provide in-depth details regarding the three proposed techniques, including SCOOP's construction and query algorithms; and implement the related approach *Nearest-Neighbor graph* (NN-graph), employed by the dynamic MAM M*-tree [7], over the same code base to enable a fair comparison. Our techniques have achieved notable gains over NN-graph, showing to be effective to improve the performance of dynamic hierarchical MAMs.

The rest of this paper is organized as follows. Section 2 covers concepts regarding similarity queries over complex data and Section 3 presents the related work. Section 4.1 describes the CLAP technique. Sections 4.2 and 4.3 describe the anticipation techniques ACIR and SCOOP. Section 5 presents the application of the proposed techniques to the MAM Slim-tree, describing the new node structures and the algorithms for insertion and for similarity queries. Section 6 describes the experiments and discusses the results. Finally, Section 7 concludes the work.

2. Background

2.1. Similarity queries over complex data

Most of complex domains do not possess the relation order. Therefore, relational operators (e.g. $<$, $\leq$, $>$ and $\geq$) cannot be employed on the elements of such domains to identify any precedence among them. It is also uncommon to employ the operators $=$ and $\neq$ on complex data, since it is quite unlikely that two complex elements are identical. Take for example two images. If a single pixel is different, then they are not equal. Hence, in complex domains, queries that consider the similarity degree among the elements make more sense [1].

To enable queries over complex domains, the elements of a dataset usually have characteristics extracted from their content. The extracted characteristics, known as *feature vector* or *signature*, are used in place of the raw data as the base for comparisons. The retrieval of complex data by means of such characteristics is called *content-based retrieval*. Usually, complex data are compared through dissimilarity relations between a pair of feature vectors. The comparisons are made by employing a *distance function* whose return value represents how dissimilar two feature vectors are. Queries performed over complex data are called *similarity queries*, since they retrieve from the dataset the most similar elements.

The criteria for selecting the most similar elements depend on the type of similarity query [8]. There are several types of similarity query, which include selections, joins and grouping similarity queries. The two most common types are the Range and the k-Nearest Neighbor queries [9]. The notation adopted in this paper considers an environment wherein the data is stored in a similarity-enabled RDBMS. Suppose a relation R containing an attribute S_j, sampled from a complex data domain. In the context, we describe both types of similarity query as follows.

Range query — Rq. Given a threshold ξ, a Range query retrieves every tuple t_i from relation R whose value s_i of attribute S_j, which refers to the feature vector of the element, satisfies the condition $\delta(s_i, s_q) \leq \xi$, where s_q is the value held by the provided element Q for the attribute S_j and δ is the distance function that returns the dissimilarity value between s_i and s_q. Considering R as a relation that stores images, an example of Range query is: "Select the images which are similar to image Q by up to 5 units".

***k*-Nearest Neighbor query — *k*-NNq.** Given an integer value $k \geq 1$, a k-Nearest Neighbor query retrieves k tuples t_i from relation