Chapter LXXVII
A Pagination Method for Indexes in Metric Databases

Ana Valeria Villegas
Universidad Nacional de San Luis, Argentina

Carina Mabel Ruano
Universidad Nacional de San Luis, Argentina

Norma Edith Herrera
Universidad Nacional de San Luis, Argentina

INTRODUCTION

Searching for database elements that are close or similar to a given query element is a problem that has a vast number of applications in many branches of computer science, and it is known as proximity search or similarity search. This type of search is a natural extension of the exact searching due to the fact that the databases have included the ability to store new data types such as images, sound, text, and so forth.

Similarity search in nontraditional databases can be formalized using the metric space model (Baeza-Yates, 1997; Chavez, Navarro, Baeza-Yates, & Marroquín, 2001; Chávez, & Figueroa, 2004). A metric space is a pair \((X, d)\) where \(X\) is a universe of objects and \(d: X \times X \rightarrow \mathbb{R}^+\) is a distance function that quantifies the similarities among the elements in \(X\). This function \(d\) satisfies the properties required to be a distance function:

- \(\forall x, y \in X, \quad d(x, y) \geq 0\) (positiveness)
- \(\forall x, y \in X, \quad d(x, y) = d(y, x)\) (symmetry)
- \(\forall x, y, z \in X, \quad d(x, y) \leq d(x, z) + d(z, y)\) (triangle inequality)

A finite subset \(U \subseteq X\), which will be called a database, is the set of objects where the search takes place.

A typical query in a metric database to retrieve similar objects is the range query, denoted by \((q, r)\). Given a query \(q \in X\) and a tolerance radius \(r\), a range query retrieves all elements from the database that are within a distance \(r\) to \(q\); that is \((q, r)_q = \{x \in U : d(x, q) \leq r\}\).

Range queries can be trivially answered by examining the entire database \(U\). Unfortunately, this is generally very costly in real applications. In order to avoid this situation, the database is preprocessed by using an indexing algorithm. An indexing algorithm is an off-line procedure whose
1. **Hyperplane criterion**: This is the most basic one and the one that best expresses the idea of compact partition. Basically, if \( c \) is the center of the class \([q]\) (i.e., the center closest to \( q \)), then the query ball does not intersect \([c]\) if \( d(q,c)+r < d(q,c)-r \). In other words, if the query ball does not intersect the hyperplane that divides both its closest center \( c \) and \( c_r \), then the ball is totally outside the class of \( c \).

2. **Covering radius criterion**: This tries to bound the class \([c]\) by considering a ball centered at \( c \) that contains all the elements of \( U \) that lie in that class. The covering radius of \( c \) for database \( U \) is \( cr(c) = \max_{u \in [c]} d(c,u) \). Therefore, we can discard \([c]\) if \( d(q,c)+r > cr(c) \).

Regardless of the indexing algorithm used, the total time to evaluate a query (similarity search) can be split as \( T = \# \text{ of distances evaluations} \times \text{complexity} (d) + \text{CPU extra time} + \text{I/O time} \), and we would like to minimize \( T \). Based on the assumption that evaluating \( d \) is so costly, most of the research on metric spaces has focused on algorithms to discard elements reducing the number of distance evaluations, paying no attention to I/O cost; one exception is the M tree designed specifically for secondary storage. However, if the index does not fit into main memory, the I/O cost plays an important role and, consequently, the performance criteria will be both the number of distance evaluations and the number of disk page reads (I/O cost).

We begin presenting a brief explanation of indexes on secondary storage. After that, we introduce our proposal in detail and give an application example. Finally, the conclusions are presented.

**INDEXES IN SECONDARY STORAGE**

The organization of the memory of the most computer systems is based on a hierarchy of memory