Information Systems 60 (2016) 34–49

Contents lists available at ScienceDirect

Information Systems

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

Aggregated 2D range queries on clustered points ☆

Nieves R. Brisaboa [a], Guillermo De Bernardo [b], Roberto Konow [c], Gonzalo Navarro [c], Diego Seco [d,*]

[a] University of A Coruña, Campus de Elviña, A Coruña, Spain
[b] Enxenio S.L., Baños de Arteixo, A Coruña, Spain
[c] DCC, University of Chile, Beauchef 851, Santiago, Chile
[d] University of Concepción, Edmundo Larenas 219, Concepción, Chile

ARTICLE INFO

Article history:
Received 25 February 2016
Received in revised form
4 March 2016
Accepted 5 March 2016
Recommended by: D. Shasha
Available online 16 March 2016

Keywords:
Compact data structures
Grids
Query processing
Aggregated queries
Clustered points

ABSTRACT

Efficient processing of aggregated range queries on two-dimensional grids is a common requirement in information retrieval and data mining systems, for example in Geographic Information Systems and OLAP cubes. We introduce a technique to represent grids supporting aggregated range queries that requires little space when the data points in the grid are clustered, which is common in practice. We show how this general technique can be used to support two important types of aggregated queries, which are ranked range queries and counting range queries. Our experimental evaluation shows that this technique can speed up aggregated queries up to more than an order of magnitude, with a small space overhead.

1. Introduction

Many problems in different domains can be interpreted geometrically by modeling data records as multi-dimensional points and transforming queries about the original records into queries on the point sets [1, Chapter 5]. In 2D, for example, orthogonal range queries on a grid can be used to solve queries of the form "*report all employees born between y_0 and y_1 who earn between s_1 and s_2 dollars*", which are very common in databases. In the same way, other aggregated range queries (e.g., top-*k*, counting, quantile, majority, etc.) have proved to be useful for data analysis in various domains, such as Geographic Information Systems (GIS), OLAP databases, Information Retrieval, and Data Mining, among others [2]. In GIS, aggregated range queries can facilitate decision making [3] by counting, for example, the number of locations within a specific area for which the values of pollution are above a threshold. Similarly, top-*k* range queries on an OLAP database[1] of sales can be used to find the sellers with most sales in a time slice. In this example, the two dimensions of the grid are seller ids (arranged hierarchically in order to allow queries for sellers, stores, areas, etc.) and time (in periods of hours, days, weeks, etc.), and the weights associated to the data points are the amount of sales made by a seller during a time slice. Thus, the query asks for the

☆ Funded in part by European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant agreement no. 690941, by Millennium Nucleus Information and Coordination in Networks ICM/FIC P10-024F (Chile), by MINECO (PGE and FEDER) Projects TIN2013-46238-C4-3-R and TIN2013-46801-C4-3-R (Spain), and also by Xunta de Galicia (GRC2013/053) (Spain). A preliminary partial version of this paper appeared in *Proceedings SPIRE 2014*, pp. 215–226.

* Corresponding author. Tel.: +56 41 2204692; fax: +56 41 2221770.
E-mail address: dseco@udec.cl (D. Seco).

http://dx.doi.org/10.1016/j.is.2016.03.004
0306-4379/

[1] The support of more than two dimensions is essential in OLAP databases. We discuss the extension to multi-dimensional structures in the conclusions.

k heaviest points in some range $Q=[i_1,i_2]\times[t_1,t_2]$ of the grid.

The approach of modeling problems using a geometric formulation is well-known. There are many classical representations that support the queries required by the model and solve them efficiently. Range trees [4] and *kd*-trees [5] are two paradigmatic examples. Some of these classical data structures are even optimal both in query time and space. However, such classical representations usually do not take advantage of the distribution of the data in order to reduce the space requirements. When dealing with massive data, which is the case of some of the aforementioned data mining applications, the use of space-efficient data structures can make the difference between maintaining the data in main memory or having to resort to (orders of magnitude slower) external memory.

In this work we consider the case where we have clustered points in a 2D grid, which is a common scenario in domains such as Geographic Information Systems, Web graphs, and social networks. There are some well-known principles that hold in most scenarios of that kind. Two examples are Tobler's first law of geography [6], which states that near things are more related than distant things, and the locality of reference for time-dependent data. This is also the case in Web graphs [7], where clusters appear when the Web pages are sorted by URL. We take advantage of these clusters in order to reduce the space of the data structures for aggregated 2D range queries.

The K^2-tree [8] (a space-efficient version of the classical Quadtree) is a good data structure to solve range queries on clustered points and it has been extensively evaluated in different domains [9–11]. We introduce a general technique to extend this data structure in order to support aggregated range queries. We then illustrate its potential by instantiating the technique in two emblematic cases: range counting and ranked (MAX/MIN) queries within a 2D range.

The paper is organized as follows. First, we introduce basic concepts and related work in Sections 2 and 3, respectively. In Section 4 we describe the general technique to extend the K^2-tree to solve different aggregated range queries on grids. Two paradigmatic examples of such queries are described in Section 5 (ranked range queries) and Section 6 (range counting queries). Section 7 presents an exhaustive empirical evaluation of the proposed solutions. Finally, Section 8 concludes and sketches some interesting lines of future work.

2. Basic concepts

2.1. Aggregated queries on clustered points

We consider two dimensional grids with n columns and m rows, where each cell a_{ij} can either be empty or contain a weight in the range $[0,d-1]$ (see Fig. 1). For some problems, we will omit the weights and just consider the non-empty cells, which can be represented as a binary matrix (Fig. 2) in which each cell contains a 1 (if there is a weighted point in the original matrix) or a 0 (in other case).

Let t be the number of 1s in the binary matrix (i.e., the number of weighted points). If we can partition the t points into c clusters, not necessarily disjoint and $c \ll t$, we will say that the points are clustered. This definition is used by Gagie et al. [12] to show that in such case a Quadtree needs only $O(c\log u+\sum_i t_i \log l_i)$ bits, where $u=\max(n,m)$, and t_i and l_i are the number of points and the diameter of cluster i, respectively.

A range query $Q=[x_1,x_2]\times[y_1,y_2]$ defines a rectangle with all the columns in the range $[x_1,x_2]$ and the rows in $[y_1,y_2]$ (see Fig. 3). An aggregated range query defines, in addition to the range, an aggregate function that must be applied to the data points in the query range. Examples of aggregated queries are $\mathcal{COUNT}(Q)$, which counts the number of data points in the query range, $\mathcal{MAX}/\mathcal{MIN}(Q)$, which computes the maximum (alt. minimum) value in the query range, and its generalization top-k, which retrieves the k lightest (alt. heaviest) points in the query range. These top-k queries are also referred to in the literature as ranked range queries. For the range query q in Fig. 3 the result of $\mathcal{COUNT}(q)$ is 6, $\mathcal{MAX}(q)$ returns 7, $\mathcal{MIN}(q)$ returns 1, and the top-3 heaviest elements are 7, 4, and 3.

There are other interesting data-analysis queries on two-dimensional grids. For example, $\mathcal{QUANTILE}(Q,a)$ returns the a-th smallest value in Q, and $\mathcal{MAJORITY}(Q,\alpha)$ retrieves those values in Q that appear with relative frequency larger than α. These and other queries have been studied by Navarro et al. [13], who introduce space-efficient data structures with good time performance. We restrict ourselves to an emblematic subset of these queries, and propose data structures that are even more space-efficient when the points in the set are clustered.

2.2. Rank and select on bitmaps

Two basic primitives used by most space-efficient data structures are rank and select on bitmaps. Let $B[1,n]$ be a sequence of bits, or a bitmap. We define operation $rank_b(B,i)$ as the number of occurrences of $b\in\{0,1\}$ in $B[1,i]$, and $select_b(B,j)$ as the position in B of the j-th occurrence of b. B can be represented using $n+o(n)$ bits [14,15], so that both operations are solved in constant time. These operations have proved very efficient in practice [16]. In addition, when the bitmaps are compressible, it is possible to reduce the space and still support these operations in constant time [17].

2.3. Wavelet tree and discrete grids

An elegant generalization of rank and select queries to an arbitrary alphabet Σ of size σ is provided by the wavelet tree [18]. Given a sequence S over the alphabet Σ, the wavelet tree supports rank, select and access in $O(\log\sigma)$ time with $n\log\sigma+o(n\log\sigma)$ bits. The wavelet tree is a complete binary tree, in which each node represents a range $R\subseteq[1,\sigma]$ of the alphabet Σ, its left child represents a subset $R_\ell\subset R$ and the right child the subset $R_r=R\backslash R_\ell$. Every node representing subset R is associated with a subsequence S' of the input sequence S composed of the