Efficient OLAP Operations in Spatial Data Warehouses

Dimitris Papadias, Panos Kalnis, Jun Zhang, and Yufei Tao

Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
{dimitris,kalnis,zhangjun,taoyf}@cs.ust.hk

Abstract. Spatial databases store information about the position of individual objects in space. In many applications however, such as traffic supervision or mobile communications, only summarized data, like the number of cars in an area or phones serviced by a cell, is required. Although this information can be obtained from transactional spatial databases, its computation is expensive, rendering online processing inapplicable. Driven by the non-spatial paradigm, spatial data warehouses can be constructed to accelerate spatial OLAP operations. In this paper we consider the star-schema and we focus on the spatial dimensions. Unlike the non-spatial case, the groupings and the hierarchies can be numerous and unknown at design time, therefore the well-known materialization techniques are not directly applicable. In order to address this problem, we construct an ad-hoc grouping hierarchy based on the spatial index at the finest spatial granularity. We incorporate this hierarchy in the lattice model and present efficient methods to process arbitrary aggregations. We finally extend our technique to moving objects by employing incremental update methods.

1 Introduction

Data warehouses are collections of historical, summarized, non-volatile data, which are accumulated from transactional databases. They are optimized for On-Line Analytical Processing (OLAP) [CCS93] and have proven to be valuable on assisting decision-making. The data in a warehouse are conceptually modeled as hyper-cubes [GBLP96] where each dimension represents some business perspective, like *products* and *stores*, and the cube cells contain a measure, such as *sales*.

Recently, the popularity of spatial information, such as maps created from satellite images and the utilization of telemetry systems, has created repositories of huge amounts of data which need to be efficiently analyzed. In analogy to the non-spatial case, a spatial data warehouse can be considered, which supports OLAP operations on both spatial and non-spatial data.

Han et. al.[HSK98; SHK00] were the first ones to propose a framework for spatial data warehouses. They considered an extension of the star-schema [K96] in which the cube dimensions can be both spatial and non-spatial and the measures are regions in space, in addition to numerical data. They focus on the spatial measures and propose a method for selecting spatial objects for materialization. The idea is similar to the algorithm of [HRU96], the main difference being the finer granularity of the selected

C.S. Jensen et al. (Eds.): SSTD 2001, LNCS 2121, pp. 443-459, 2001.

objects. In [ZTH99], an I/O efficient method is presented for merging spatial objects. The method is applied on the computation of aggregations for spatial measures.

In this paper we concentrate on the spatial dimensions. The fact that differentiates the spatial attributes from the non-spatial ones is that there is little or no a-priori knowledge about the grouping hierarchy. The user, in addition to some predefined regions, may request groupings based on maps which are computed on the fly, or may be arbitrarily created (e.g. an arbitrary grid in a selected window). Therefore the well-known pre-aggregation methods [HRU96; G97; GM99; SDN98] which are used to enhance the system performance under OLAP operations, cannot be applied.

Motivated by this fact, we propose a method which combines spatial indexing with the pre-aggregated results. We built a spatial index on the objects of the finer granularity in the spatial dimension and use the groupings of the index to define a hierarchy. We incorporate this implicit hierarchy to the lattice model of Harinarayan et. al, [HRU96] to select the appropriate aggregations for materialization. We study several algorithms for spatial aggregation and we propose a method which traverses the index in a breadth-first manner in order to compute efficiently group-by queries. Finally we employ incremental update techniques and show that our method is also applicable for moving objects.

Storing aggregated results in the index for non-spatial warehouses has been proposed by Jurgens and Lenz [JL98]. Lazaridis and Mehrotra [LM01] use a similar structure for on-line computation of approximated results which are progressivly refined. Yang and Widom [YW01] also employ an aggregation tree for incremental maintenance of temporal aggregates. None of these papers considers spatial objects.

There are numerous applications that benefit from our method. Throughout this paper we use an example of a decision support system for traffic control in a city. Some of the queries that can be answered efficiently are “which is the total number of cars inside every district”, or “find the road with the highest traffic in a 2km radius around every hospital”. One could also use data from other sources, like a map which groups the city based on the pollution levels, and group the traffic data based on the regions of the pollution map. Other application domains include network traffic control and congestion prevention systems for cellular communications, meteorological applications, etc.

The rest of the paper is organized as follows: Section 2 provides a brief overview of OLAP and a motivating example followed throughout the paper. Section 3 proposes the aR-tree which keeps aggregated information organized according to the spatial dimensions. Section 4 describes algorithms for query processing and Section 5 discusses update issues. Section 6 evaluates the approach experimentally, while Section 7 concludes the paper with future directions and potential applications.

2 Related Work and Motivating Example

In this work we assume that the multi-dimensional data are mapped on a relational database using a star schema [K96]. Let D_1, D_2, …, D_n be the dimensions (i.e. business perspectives) of the database, such as *Product*, *Store* and *Time*. Let M be the measure of interest; *Sales* for example. Each D_i stores details about the dimension, while M is stored in a fact table F. Each tuple of F contains the measure plus pointers (i.e. foreign keys) to the dimension tables (Figure 1a).