IGI PUBLISHING
701 E. Chocolate Avenue, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.igi-global.com

ITJ 5259

This paper appears in the publication, Journal of Database Management, Volume 20, Issue 4
edited by **Keng Siau © 2009, IGI Global**

A Dynamic Grid File for High-Dimensional Data Cube Storage and Range-Sum Querying

Wen-Chi Hou, Southern Illinois University, USA

Xiaoguang Yu, Southern Illinois University, USA

Chih-Fang Wang, Southern Illinois University, USA

Cheng Luo, Coppin State University, USA

Michael Wainer, Southern Illinois University, USA

ABSTRACT

In this article, the authors propose to use the grid file to store multi-dimensional data cubes and answer range-sum queries. The grid file is enhanced with a dynamic splitting mechanism to accommodate insertions of data. It overcomes the drawback of the traditional grid file in storing uneven data while enjoying its advantages of simplicity and efficiency. The space requirement grows linearly with the dimension of the data cube, compared with the exponential growth of conventional methods that store pre-computed aggregate values for range-sum queries. The update cost is O(1), much faster than the pre-computed data cube approaches, which generally have exponential update cost. The grid file structure can also respond to range queries quickly. They compare it with an approach that uses the R-tree structure to store the data cube. The experimental results show that the proposed method performs favorably in file size, update speed, construction time, and query response time for both evenly and unevenly distributed data.*

Keywords: Data Management, Data Structure, Multidimensional Database, Query Processing

INTRODUCTION

A data warehouse is a large collection of integrated data, built to assist knowledge workers, such as executives, managers, analysts, etc., to make better and faster decisions. It is often required that data be summarized at various levels of detail and on various attributes to allow knowledge workers to analyze the data through a variety of views in on-line analytical processing (OLAP). Typical OLAP applications include product performance and profitability, effectiveness of sales programs or marketing campaigns, sales forecasting, capacity planning, etc. Data warehousing and OLAP have

DOI: 10.4018/jdm.2009062503

increasingly become a focus of the database industry.

OLAP systems generally support a multi-dimensional data model, which is also known as the data cube (Gray, 1997). Construction of a data cube is based on the set of selected attributes of the database. Certain attributes are chosen to be the measure attributes, i.e., attributes whose values are of interest, while some others are selected as dimension or functional attributes (Geffner, 1999). The values of the measure attributes are often aggregated according to the dimension attributes for analysis. The size of a data cube can be huge when the number of combinations of dimension attribute values is large.

The storage of data cubes is essential to OLAP. Much research (Agarwal, 1996; Beyer, 1999; Han, 2001; Morfonios, 2006; Xin, 2003; Zhao, 1997) has focused on the materialization of data cubes, that is, to pre-compute and store all possible combinations of multi-dimensional aggregates for fast multi-dimensional analysis. Some notable cube materialization algorithms proposed include ROLAP-based multi-dimensional aggregate computation (Agarwal, 1996, Morfonios, 2006), multi-way array aggregation (Beyer, 1999), BUC (Han, 2001), H-cubing (Xin, 2003), Star-cubing (Zhao, 1997), Minimal cubing (Li, 2004), etc. Since materializing data cubes are generally computationally intensive and space consuming, much effort has been devoted to reducing the computation and storage space of data cubes. These efforts include partial materialization of data cubes (Harinarayan, 1996), iceberg cube computation (Han, 2001; Xin, 2003; Zhao, 1997), computation of condensed, dwarf, and quotient cubes (Lakshmanan, 2002; Lakshmanan, 2003; Sismanis, 2002; Beyer, 1999; Wang, 2002), and computation of approximate cubes (Barbara, 1997; Cuzzocrea, 2006; Shanmugasundaram, 1999). While these pre-computed data cubes can be used to answer queries quickly, tremendous overhead is incurred in maintaining these pre-computed aggregate values as updates can propagate to a large number of relevant cells.

A range-sum query is used to compute the sum of the values of data cube cells that fall in the ranges specified by the query. It is very useful in finding trends and discovering relationships between attributes in the databases. Ho et al. (1997) proposed to compute prefix sums of data cube cells for range-sum queries. Although this method can respond to queries quickly, an update in the worst case can propagate to the entire prefix-sum cube. Therefore, it may not be suitable for data cubes that undergo frequent changes. Some efforts have been made to reduce the update propagations of prefix-sum cubes. Geffner et al. (1999) computed the relative prefix-sums to limit cascading updates to sub-cubes. More recently, they proposed (Geffner, 2000) to decompose the prefix-sum cubes recursively to control the cascading of updates. Although these measures can reduce the cost of updates to a certain degree, the cost can still increase exponentially with the number of dimensions of the data cubes. Chan et al. (1999) proposed a class of hierarchically organized prefix-sum cubes to reduce the update cost. However, the cost of update still increases exponentially with the number of dimensions. Gao et al. (2005) discussed efficient processing of range-sum queries over hierarchical cubes using parallel computing systems. In general, update propagation is a common problem for all pre-computed data cube approaches even though some improvements have been made. Note that all these approaches also require at least as much space as the original data cubes to store the prefix-sum cubes.

Instead of materializing pre-computed data cubes for range-sum queries, Hu et al. (2002) chose to store data cube cells in a slightly modified version of the R*-tree (Beckmann, 1990), called the DCA-tree. Updates to the data cube are accomplished by updating the corresponding points in the R*-tree.

Similar to Hu's approach (Hu, 2002) we attempt to design a spatial data structure to facilitate data cube storage and range-sum query processing. We propose a dynamic grid file structure as a natural fit to the structure of data cubes. The data cube space is partitioned into

Related Content