IGI PUBLISHING
701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.igi-global.com

ITJ 4544

This paper appears in the publication, International Journal of Data Warehousing and Mining, Volume 5, Issue 1
edited by **David Taniar © 2009, IGI Global**

Lossless Reduction of Datacubes using Partitions

Alain Casali, Aix-Marseille Universités, France

Sébastien Nedjar, Aix-Marseille Universités, France

Rosine Cicchetti, Aix-Marseille Universités, France

Lotfi Lakhal, Aix-Marseille Universités, France

Noël Novelli, Aix-Marseille Universités, France

ABSTRACT

Datacubes are especially useful for answering efficiently queries on data warehouses. Nevertheless the amount of generated aggregated data is huge with respect to the initial data which is itself very large. Recent research has addressed the issue of a summary of Datacubes in order to reduce their size. The approach presented in this paper fits in a similar trend. We propose a concise representation, called Partition Cube, based on the concept of partition and we give a new algorithm to compute it. We propose a Relational Partition Cube, a novel ROLAP cubing solution for managing Partition Cubes using the relational technology. Analytical evaluations show that the storage space of Partition Cubes is smaller than Datacubes. In order to confirm analytical comparison, experiments are performed in order to compare our approach with Datacubes and with two of the best reduction methods, the Quotient Cube and the Closed Cube.

Keywords: *concept lattices; datacubes; OLAP; partitions*

INTRODUCTION

In order to efficiency answer OLAP queries (Chaudhuri & Dayal, 1997), a widely adopted solution is to compute and materialize Datacubes (Gray et al., 1997). For example, given a relation r over the schema R, a set of dimensions Dim = $\{C_1, C_2, C_3\}$, Dim $\subseteq R$, a measure $M \in R$, an aggregate function f, the cube operator (Gray et al., 1997) is expressed as follows:

SELECT C_1, C_2, C_3, $f(M)$
FROM r
CUBE BY (C_1, C_2, C_3)

Dimensions are also called categorical attributes and r a categorical database relation. The given query achieves all the possible group-by according to any attribute combination belonging to the power set of Dim. It results in what is called a Datacube, and each sub-query

performing a single group-by yields a cuboid. Computing Datacubes is exponential in the number of dimensions (the lattice of the dimension set must be explored), and the problem worsens when very large data sets are to be aggregated. Datacubes are considerably larger than the input relation. (Ross & Srivastava, 1997) exemplifies the problem by achieving a full Datacube encompassing more than 210 millions of tuples from an input relation having 1 million of tuples. The problem is originated by a twofold reason: on one hand the exponential number of dimensional combinations to be dealt, and on the other hand the cardinality of dimensions. The larger dimension domains are, the more aggregated results there are (according to each real value combination). Unfortunately, it is widely recognized that in OLAP databases, data can be very sparse (Ross & Srivastava, 1997; Beyer & Ramakrishnan, 1999) thus scarce value combinations are likely to be numerous and, when computing entirely the Datacubes (full Datacubes), each exception must be preserved. In such a context, (1) approaches favour the efficency of OLAP queries to the detriment of storage space or (2) they favour an optimal representation of cubes but OLAP query performances are likely to be debased (Kotidis & Roussopoulos, 1998).

RELATED WORK

The approaches addressing the issue of Datacube computation and storage attempt to reduce at least one of the quoted drawbacks. The algorithms Buc (Beyer & Ramakrishnan, 1999) and Hcubing (Han, Pei, Dong, & Wang, 2001) enforce antimonotone constraints and partially compute Datacubes (iceberg cubes) to reduce both execution time and disk storage requirements. The underlying argument is that OLAP users are only interested in general trends (and not in atypical behaviors). With a similar argumentation, other methods use the statistic structure of data to compute density distributions and give approximate answers to OLAP queries (Pedersen, Jensen, & Dyreson, 1999; Vitter & Wang, 1999; Shanmugasundaram, Fayyad, & Bradley, 1999; Gilbert, Kotidis, Muthukrishnan, & Strauss, 2001).

The above mentioned approaches are efficient and meet their twofold objective (reduction of execution time and space storage). However, they are not able to answer whatever query (although OLAP queries are, by their very nature, ad hoc queries (Han & Kamber, 2001)).

Another category of approaches is the so-called "information lossless". They aim to find the best compromise between OLAP query efficiency and storage requirements without discarding any possible query (even unfrequent). Their main idea (see for details (Harinarayan, Rajaraman, & Ullman, 1996; Kotidis & Roussopoulos, 1999; Sellis, 2004; Theodoratos & Xu, 2004; Gupta & Mumick, 2005)) is to pre-compute and store frequently used aggregates while preserving all the data (possibly at various aggregation levels) needed to compute on line the result of a not foreseen query. They are mostly found in view materialization research.

The following five methods[1] also fit in the information lossless trend: the Dwarf Cube (Sismanis, Deligiannakis, Roussopoulos, & Kotidis, 2002), the Condensed Cube (Wang, Lu, Feng,&Yu, 2002), the CURE for Cubes (Morfonios & Ioannidis, 2006), the Quotient Cube (L. Lakshmanan, Pei, & Han, 2002; L. V. S. Lakshmanan, Pei, & Zhao, 2003) and the Closed Cube (Casali, Cicchetti, & Lakhal, 2003b; Li & Wang, 2005; Xin, Shao, Han, & Liu, 2006). They favor the optimization of storage space while preserving the capability to answer what ever query. The two latter compute the two smallest representations of a Datacube and thus are the most efficient for both saving storage space and answering queries like "Is this behavior frequent or not?". From these two representations, the exact data of a whole Datacube can be retrieved by performing a computation on line, because results of queries are not precomputed and preserved.