European Journal of Operational Research 246 (2015) 582–596

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Innovative Applications of O.R.

Methods for solving the mean query execution time minimization problem

Marek Łatuszko*, Radosław Pytlak

Institute of Automatic Control and Robotics, Warsaw University of Technology, Św. Andrzeja Boboli 8, 02-525 Warsaw, Poland

ARTICLE INFO

Article history:
Received 18 June 2014
Accepted 21 April 2015
Available online 28 April 2015

Keywords:
Decision support systems
Heuristics
OLAP
View materialization
View selection problem

ABSTRACT

One of the most significant and common techniques to accelerate user queries in multidimensional databases is view materialization. The problem of choosing an appropriate part of data structure for materialization under limited resources is known as the view selection problem. In this paper, the problem of the mean query execution time minimization under limited storage space is studied. Different heuristics based on a greedy method are examined, proofs regarding their performance are presented, and modifications for them are proposed, which not only improve the solution cost but also shorten the running time. Additionally, the heuristics and a widely used Integer Programming solver are experimentally compared with respect to the running time and the cost of solution. What distinguishes this comparison is its comprehensiveness, which is obtained by the use of performance profiles. Two computational effort reduction schemas, which significantly accelerate heuristics as well as optimal algorithms without increasing the value of the cost function, are also proposed. The presented experiments were done on a large dataset with special attention to the large problems, rarely considered in previous experiments. The main disadvantage of a greedy method indicated in literature was its long running time. The results of the conducted experiments show that the modification of the greedy algorithm together with the computational effort reduction schemas presented in this paper result in the method which finds a solution in short time, even for large lattices.

1. Introduction

On-Line Analytical Processing (OLAP) is one of the most important technologies applied in modern decision support systems. OLAP provides users with ability to perform on-line, multidimensional analysis requiring the computation of many aggregating functions, frequently on large volumes of data. The query response time is considered as the main measure of system efficiency. Many methods are used to meet the performance demands, beginning from the base OLAP characteristic – the specialized multidimensional structure, through data indexing strategies, partitioning, or query optimizers. One of the common techniques is precomputing (materializing) the part of data cube aggregates. The user queries can retrieve data from prepared structures instead of making all calculations on the fly (Chaudhuri, Dayal, & Narasayya, 2011). The problem of choosing part of data structure for materialization under limited resources is universally known as the view selection problem, or as the warehouse view selection problem when used in designing data warehouses. The solution to this problem is not as simple as materializing the cells which are most frequently requested by users' queries, since cells are dependent on each other and some may be even not asked at all but their materialization will greatly facilitate calculation of other cells. In general case, the problem of selecting the right part of cube for materialization is NP-hard (Gupta, 1997).

To illustrate the concept of using materialized views to answer queries let us consider an example of simple star schema database (Kimball & Ross, 2002), which is derived from O'Neil, O'Neil, and Chen (2009). The schema consists of four dimension tables: CUSTOMER, SUPPLIER, PART and DATE and one fact table: LINEORDER. For simplification, assume that every dimension table has only one attribute, called the same as dimension name and fact table has only one measure Revenue. The database schema is presented in Fig. 1(a). For example, a user may be interested in revenue by customers and parts, which is expressed by the following SQL query:

```
SELECT C.Customer, P.Part, SUM(L.Revenue)
FROM dbo.LINEORDER L, dbo.CUSTOMER C, dbo.PART P
WHERE L.Customer=C.Customer and L.Part=P.Part
GROUP BY C.Customer, P.Part
```

The query could be answered directly from the schema tables, but if the view grouping data by customers and parts is

* Corresponding author. Tel.: +48 22 234 8497.
E-mail addresses: M.Latuszko@mchtr.pw.edu.pl (M. Łatuszko), R.Pytlak@mchtr.pw.edu.pl (R. Pytlak).

http://dx.doi.org/10.1016/j.ejor.2015.04.041
0377-2217/

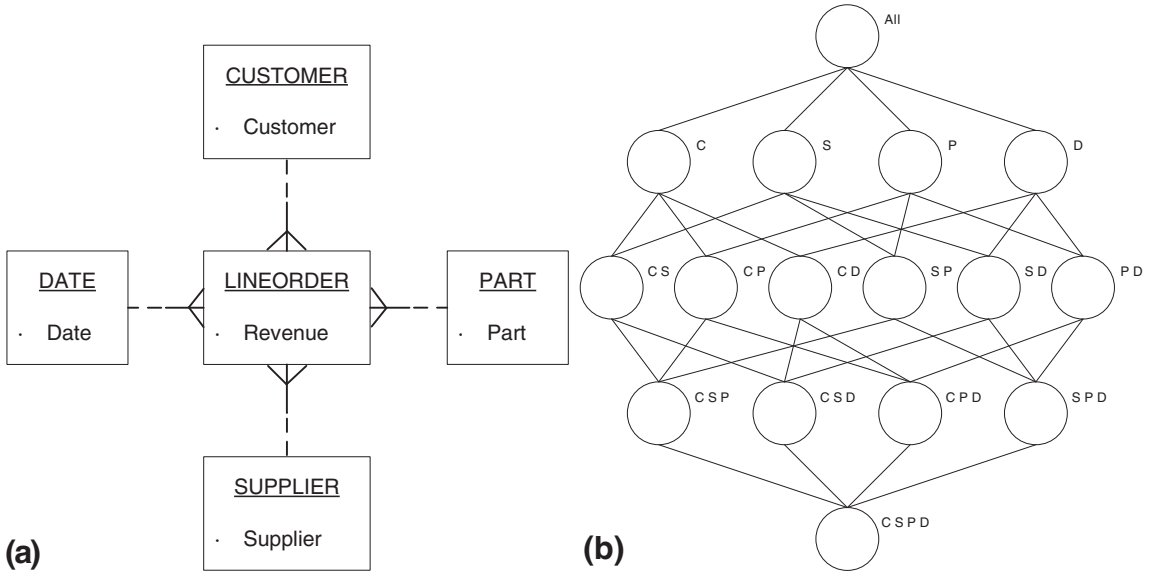

Fig. 1. Star schema example: (a) database diagram and (b) lattice diagram.

materialized, it could be used to save time on joins, groupings, and aggregations.

The above query could also be answered from the view which groups data by customers, parts, and suppliers (by summing revenue over all suppliers), or from the view which groups data by customers, parts, and dates (by summing revenue over all dates). The lattice framework introduced in Harinarayan, Rajaraman, and Ullman (1996) could be used to describe relations between all possible views which aggregate revenue by attributes from dimension tables. Views are represented as nodes in the lattice diagram. Every edge that connects two views means that a higher view (parent) can be calculated from a lower view (child). More precisely, if $\succ$ denotes a strong partial order between views,[1] $v_i \neq v_j$ and $v_i \succ v_j$ then there is an edge between two views v_i and v_j and there is no v_k such that $v_i \succ v_k \succ v_j$, $v_k \neq v_i$ and $v_k \neq v_j$.[2] All possible views with dependencies between them, for the considered example, are presented in Fig. 1(b). Each view is labelled according to the first letter of attributes in its GROUP BY (and WHERE) clause. The 'All' view contains one value which aggregates all the data (note, that it can be computed from any other view). The lowest view is created by joining all dimension tables with a fact table with grouping on all attributes. In this paper, the lowest view is called the base view. Every view could be computed from the base view.

The considered problem is as follows: giving the data cube, probability distribution of user queries and maximum space allocated for views materialization, find a set of views for materialization which minimizes the mean response time of user queries. As in most papers (Asgharzadeh Talebi, Chirkova, and Fathi, 2007 and antecedent papers, Asgharzadeh Talebi, Chirkova, and Fathi, 2013; Gupta, Harinarayan, Rajaraman, and Ullman, 1997; Harinarayan et al., 1996; Kalnis, Mamoulis, and Papadias, 2002; Li, Asgharzadeh Talebi, Chirkova, and Fathi, 2005; Shukla, Deshpande, and Naughton, 1998) the analysis is restricted to a data cube, which views form OR view graph (Gupta, 1997; Gupta & Mumick, 2005). The view graph is OR if every parent view can be computed from any of its children (as in the lattice defined above). It is also assumed that each query is always answered from one view (as in Asgharzadeh Talebi et al., 2007 and antecedent papers, Asgharzadeh Talebi et al., 2013; Gupta et al., 1997; Harinarayan et al., 1996; Kalnis et al., 2002; Li et al., 2005; Shukla et al., 1998) and based on that assumption in the rest of the paper the terms query evaluation time and view evaluation time are used interchangeably. When the cube views are described as the lattice, the view selection problem corresponds to selecting nodes from the lattice diagram. As stressed in Harinarayan et al. (1996) the storage space is also a good indicator of the time needed to create the cube.

The research presented here was motivated by the conclusions from Asgharzadeh Talebi et al. (2007) and antecedent papers. Intuitively, it may be supposed that for real size problems an exponential algorithm like Branch and Bound performs much worse with respect to the execution time than a polynomial time heuristic like a greedy algorithm. However, from Asgharzadeh Talebi et al. (2007) and antecedent papers it can be learnt that branch and bound actually performs better not only with respect to the quality of solution (which is obvious) but also with respect to the execution time.

The specific contributions of the paper are as follows:

(i) Different heuristics presented in literature and based on a greedy method are compared with widely used Integer Programming procedures.
(ii) The numerical experiments presented in the paper were done on large datasets since that minimizes the influence of untypical cases. A special attention was paid to large problems rarely considered in numerical experiments discussed so far in literature.
(iii) The presented paper is the first one in which different algorithms for the view selection problem are compared comprehensively with the help of performance profiles (Dolan & Moré, 2002).
(iv) Using the example of the lattice from Karloff and Mihail (1999) it has been shown that the popular greedy algorithm presented in Gupta and Mumick (2005) has no performance guarantee for the discussed view selection problem.
(v) Several modifications to the greedy heuristic from Gupta and Mumick (2005) have been proposed to construct the method which is better with the respect to the value of objective function and with respect to the computational time.

[1] $v_i \succ v_j$ for $v_i \neq v_j$ if and only if v_i can be answered using only the results of v_j – we say that v_i is calculated from v_j.

[2] In fact, in order to have the lattice it is also required that for any two views the least upper bound (supremum) and the greatest lower bound (infimum) must exist. However, after Harinarayan et al. (1996) it is postulated that: (1) there is a partial order between views; (2) there exists one view from which every other view could be determined.