Chapter II

Computing Dense Cubes Embedded in Sparse Data

Lixin Fu, The University of North Carolina at Greensboro, USA

Abstract

In high-dimensional data sets, both the number of dimensions and the cardinalities of the dimensions are large and data is often very sparse, that is, most cubes are empty. For such large data sets, it is a well-known challenging problem to compute the aggregation of a measure over arbitrary combinations of dimensions efficiently. However, in real-world applications, users are usually not interested in all the sparse cubes, most of which are empty or contain only one or few tuples. Instead, they focus more on the “big picture” information—the highly aggregated data, where the “where clauses” of the SQL queries involve only few dimensions. Although the input data set is sparse, this aggregate data is dense. The existing multi-pass, full-cube computation algorithms are prohibitively slow for this type of application involving very large input data sets. We propose a new dynamic data structure called Restricted Sparse Statistics Tree (RSST) and a novel cube evaluation algorithm, which are especially well suited for efficiently computing dense sub-cubes imbedded in high-dimensional sparse data sets. RSST only computes the
aggregations of non-empty cube cells where the number of non-star coordinates (i.e., the number of group by attributes) is restricted to be no more than a user-specified threshold. Our innovative algorithms are scalable and I/O efficient. RSST is incrementally maintainable, which makes it suitable for data warehousing and the analysis of streaming data. We have compared our algorithms with top, state-of-the-art cube computation algorithms such as Dwarf and QCT in construction times, query response times, and data compression. Experiments demonstrate the excellent performance and good scalability of our approach.

Introduction

Given n dimensions \( D_1, D_2, \ldots, D_n \), where domain values of \( D_i \) are in \( 0..C_i-1 \), \( C_i \) is the cardinality of \( D_i \), and a measure \( M \), the data cube problem is to compute the aggregation of \( M \) over any subset \( Q \subseteq D_1 \times D_2 \times \ldots \times D_n \). Domain values other than integers are converted into integers. For example, strings and ordinary domains are mapped into integers starting from 0. Real numbers can be discretized into ranges. SUM, COUNT, and MIN/MAX are typical aggregation operators. In this chapter, we mainly focus on COUNT and SUM operators; others can be implemented similarly.

Data cube facility is essential in data warehousing and OLAP (Gray et al., 1997). Because of the importance of efficient data cube computation and exploration, numerous significant studies have been performed. However, the state-of-the-art algorithms and systems do not scale well in terms of I/O bottleneck. This problem is important because in the data warehousing and analytical data processing environment, the data sets are often very massive. Almost all current technologies build certain structures (e.g., materialized views, indexes, trees, etc.) before answering user queries. The running times of setting up the structures for such large data sets will be dominated by I/O operations. Current top algorithms either entirely ignore the problem of I/O efficiency or do not handle it well. They require multiple passes for large data sets, which makes the data cube computation prohibitively slow or infeasible.

On the other hand, although the input data sets are very large and very sparse (i.e., the vast majority of cubes are empty), the high-level views with few group-bys contain dense data—the typical targets of user navigation. In data cube applications, users (mostly data analyzers and top managers) are usually not interested in the low-level, detailed data, which is the typical focus of daily transactional processing operations. The analytical users are more interested in the highly aggregated, “big picture” information. For example, in a warehouse storing car sales data, a manager may not be so interested in a query like “What is the total sale for golden, Honda Accord
Incremental Algorithm for Discovering Frequent Subsequences in Multiple Data Streams
www.igi-global.com/article/incremental-algorithm-discovering-frequent-subsequences/58635?camid=4v1a