Multiresolution Data Aggregation and Analytical Exploration of Large Data Sets

Li Yang Mustafa Sanver
Department of Computer Science, Western Michigan University
{yang, msanver}@cs.wmich.edu

Abstract

Analytical processing of large relation data demands for a shared compact representation of data in multiple resolutions in order to efficiently facilitate the incurring data aggregation, data cube, and range queries. This paper addresses technical problems of multi-resolution data aggregation and investigates enabling technologies for efficient analytical processing of large data sets. In particular, the paper discusses visualization and data access techniques for interactive exploration of large data sets.

1. Introduction

Analytical processing of large relational data plays important roles in data warehousing and data mining. In these areas, relational data are commonly assumed as points distributed in high dimensional space. Therefore, the problem is how to support analytical processing, mining, and exploration of a large number of data points in high dimensional space effectively, efficiently, intuitively, and interactively.

By *large data*, we mean a data set that is too large to be loaded entirely into main memory. Over the years, data warehousing and data mining communities have developed efficient techniques to facilitate decision support on large data. One major idea is to materialize data aggregates and data cubes. However, the idea is not effective to restrain the exponential size of the data involved. For a relational data set with d aggregating attributes $A_1, \ldots, A_d$, multi-dimensional aggregation on the d attributes would have $\prod_{i=1}^{d} |A_i|$ cells, which constitute a base cuboid in the data cube on the d attributes. Assume each attribute A_i has C_i levels of concept hierarchy, the full data cube would contain $\prod_{i=1}^{d} C_i$ such cuboids. Although techniques have been proposed to reduce the size of the aggregated data, most of these techniques (1) support only a small number of dimensions, (2) ignore conceptual hierarchy, and (3) work in ad hoc ways and apply only to special situations or data distributions. In data mining, techniques have been developed for data clustering and data classification of large relational data. However, these techniques have the common drawback that they are query-dependent. They build their own unique data structures once for each query and such structures are generally of no use to answer further queries. Using raw relational data as input, these approaches need to scan the data set at least once for each query, which cause the computational complexities to be at least $O(n)$. In addition, decision support applications have additional concerns such as data integration and privacy protection. Existing privacy-preserving techniques (perturbation, k-anonymization, swapping) are ad hoc and also query-dependent.

The above observations suggest that large relational data in their raw format are rarely appropriate as data input for decision support applications. We have the following beliefs of analytical applications on large data sets:

- We believe that the key challenge is to create a common data representation that convert a large amount of relational data into forms that facilitate analytical applications. Such a data representation should be compact yet still comprehensive enough to answer most queries. Preparing such a common data representation may take time (ideally incremental to input data records). However, decision support should be more efficient using this data representation as input.

- We believe that such a common data representation should be available in multiple resolutions and *data resolution* plays an instrumental role in analytical applications. Applications should be offered a trade-off between precision and performance. Another big advantage of data resolution is that it provides a radical mechanism for privacy preservation: each user can be given permissions to access the data within a specific range of resolutions and, therefore, the privacy of individual data records is preserved. Existing data warehousing techniques do not support data resolution beyond concept hierarchy, yet concept hierarchy is not

[0]Research described in this paper is supported in part by National Science Foundation under Grant IIS-0414857.

fully honored in most data cube implementations.

- We believe that such a common data representation should support multidimensional indexing of aggregated data in order to support slicing and dicing, data selection, and range queries.

In summary, we think that decision support applications need a common representation of aggregated data. Such a data representation should be available in multiple resolutions and should provide a mechanism to index the multi-resolution data aggregates. In this paper, we discuss data aggregations piggybacked in internal nodes of a high-dimensional tree index as an intermediate data interchange mechanism between database and analytical applications. Because the number of entries of the aggregated data depends on the resolution rather than the number of original data records, it makes decision support tools scalable to large data sets.

As a fundamental change of input data format, multi-resolution data aggregation opens new challenges for research in OLAP, data mining, and data exploration. This paper investigates enabling techniques for these analytical applications. In particular, we focus on interactive visual exploration of large data sets, which is an important area that has not been adequately addressed by the data mining community.

2. Multi-Resolution Data Aggregation

At the core of multidimensional data analysis is efficient computation of data aggregates. As a primitive operation in SQL, the data aggregation (group-by) operation is extensively studied in database systems. Basic techniques for computing group-by's are sorting and hashing to organize the data by value and then aggregating with a sequential scan (often built into the sorting or hashing) of the organized data records. Because data reporting also needs subtotal and cross-tabulation, Jim Gray et al. [5] proposed new operators, *data cube* and *roll-up*. A data cube can be logically thought as the union of all group-by's each of which is obtained by grouping on a subset of aggregating attributes.

Efficient implementation of the data cube operation has received extensive study with many interesting approaches proposed. Data cube computation soon becomes prohibitive as the number of aggregating attributes increases (without considering class hierarchy, a full cube on d attributes has to compute 2^d group-by's). Iceberg cube [2] is a technique to compute only dense cells. Methods for computing iceberg cubes include BUC [2], H-cubing [6], and Star-cubing [11]. Researchers also paid attention on navigating data cubes, for example, generalizations along various roll-up paths [10] and cube transversals and closures [3].

The above techniques have not considered class hierarchy (never to mention multiple resolutions). A class hierarchy C_i on dimension D_i can be modeled as a lattice structure (in most cases it is simply a chain of layers with total order). A cube lattice on the dimensions $D_1, \ldots, D_d$ is then a product lattice $C_1 \times \cdots \times C_d$ where the partial order is defined as $c'_1 \times \cdots \times c'_d \preceq c''_1 \times \cdots \times c''_d$ if $c'_i \preceq c''_i$ for $c'_i, c''_i \in C_i$ and $1 \leq i \leq d$ respectively. Figure 1 gives example class hierarchies on three dimensions and shows a Hasse diagram of their cube lattice. The cube lattice with class hierarchy provides a way to aggregate data at multiple resolutions. Class hierarchy introduces two fundamental problems to efficient cube computation: (1) The number of cuboids in a cube lattice increases from 2^d to $\prod_{i=1}^{d} |C_i|$, assuming each dimension D_i has $|C_i|$ levels of concept hierarchy; (2) The data cube lattice is no longer a power set lattice and may require new strategies for lattice traversal.

In addition to multi-resolution data aggregation, range query is important in analytical applications. There do exist multi-dimensional index structures for data cubes: Ho et al. [7] used a quad-tree structure where each node contains the maximum measure value for a partition. An R*-tree is used to manage the minimum bounding hyperrectangles (MBRs) and the prefix sums of the dense regions. Roussopoulos and Kotidis [9] introduced cubetree which supports multidimensional range queries and bulk incremental updates. The cubetree structure is realized by a collection of packed R-trees. R-tree has the drawback that the overlap of MBRs in internal nodes grows with increasing dimensionality. Ester et al. proposed a DC-tree structure [4] for dynamic index maintenance. DC-tree supports concept hierarchy and has better performance when dimensionality increases.

For the past few years, we have focused on data access methods to support multi-resolution data aggregation. We have found that a partition-based high dimensional tree index offers a good vehicle to piggyback data aggregated at multiple resolutions, provided that the data have been aggregated according to the regions represented by internal nodes of the tree. Internal nodes of the tree cannot overlap with each other in order to make sure that every data point is aggregated exactly once at a given resolution. Therefore, the high dimensional index on which we choose to piggyback data aggregates must meet all of the following requirements: (1) it is a hierarchical data structure in order to carry data aggregates at multiple resolutions; (2) it provides a point access method (PAM) instead of a spatial access method (SAM); (3) regions represented by sibling nodes are disjoint with each other (no point is counted more than once); (4) the region represented by a node is totally covered by the union of regions represented by all of its child nodes (no point is uncovered). In short, the high dimensional index must be a partition-based PAM. There exist a few partitioned-based PAMs on secondary storage.

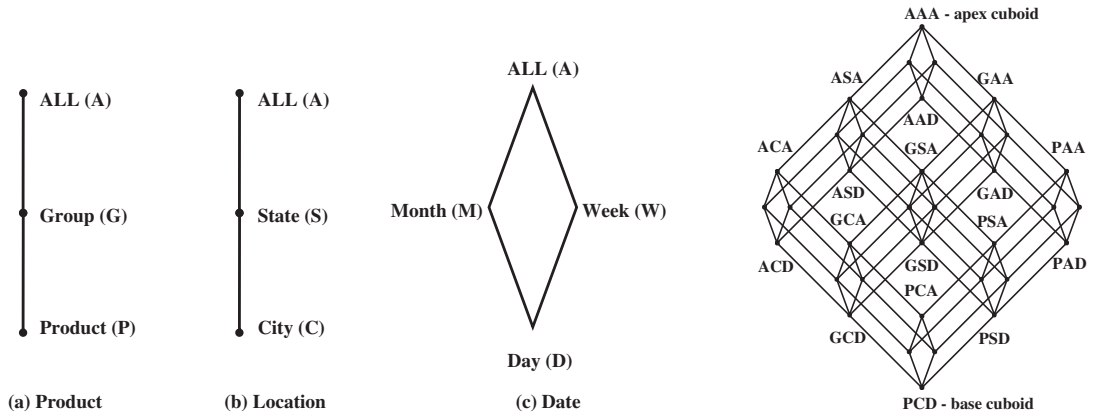


Figure 1. Lattices of attributes in three dimensions and a Hasse diagram of their product lattice.

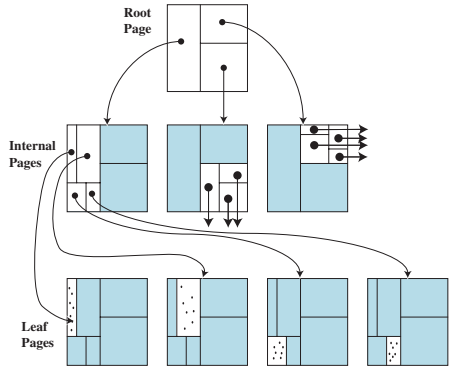


Figure 2. An example 2dB-tree.

kdB-tree [8] is a basic PAM that other PAMs (LSD-tree, Buddy-tree, hB-tree, and Bkd-tree) are based on. We have chosen kdB-tree as our primary data access method for its simplicity. We call such a tree index *data aggregation tree*. Figure 2 shows an example kdB-tree when k is two.

We have extended the kdB-tree structure by storing data aggregation values in internal nodes of the tree. Each disk page represents a hyperrectangle in high dimensional space and contains a collection of internal nodes that partition the hyperrectangle into smaller hyperrectangles. Each internal node has the format (Region, PageID, AggregateValues), where Region specifies the hyperrectangle represented by the node, PageID points to a child page representing the hyperrectangle, and AggregateValues represent a list of aggregate measures of all data points in the hyperrectangle. The user decides which data aggregate measures are kept in the node when building the tree.

This approach to multi-resolution data aggregation introduces a few research topics, the biggest one of which is probably the performance of analytical and range queries supported by the data aggregation tree. In principle, any partitioned-based PAM could be used to piggyback data aggregates. An important work we have to do is to carefully examine them for their performances for range queries, bulk loading, data insertion/deletion, and maintenance of piggybacked data aggregates. The original kdB-tree suffers from a cascade splitting problem of data insertion for the purpose of keeping the tree height-balanced. The problem causes unpredictable performance of data insertion. Another problem is bulk loading. Since the traditional sort-based approach is not applicable, high dimensional bulk loading are primarily buffer-based and sample-based. Techniques for bulk loading need to be studied together with node splitting strategy for loading large relational data sets.

An additional advantage of multi-resolution data representation is that it provides support to preserve the privacy of individual data records. Data resolution gives a new dimension for privacy preservation where permissions can be granted to users according to data resolutions. In this way, a multi-resolution data representation enables permissions to each user to access the data within a specific range of resolutions and, therefore, preserves the privacy of individual data records in a radical way. This approach for privacy preservation deserves further investigation.

Another concern is to decide a set of data aggregate measures that should be kept in each tree node. The set of aggregate measures should be enough to answer most user queries and still small enough in size so that each internal page keeps a high fan-out degree. Gray et al. [5] has classified aggregate measures into three categories: *distributive*, *algebraic*, and *holistic*. Distributive and algebraic measures in an index node can be directly computed from the corresponding distributive aggregate measures in its child nodes. An important decision is therefore to choose a set of distributive aggregate measures to be kept in tree nodes and to make clear what analytical and mining queries can be answered using the measures without accessing individual data records. In data clustering, for example, BIRCH

[13] uses cluster features (CFs) to summarize a data cluster and CF-trees to represent hierarchical cluster structures. CFs consist of count, sum, and squared sum of data points. Such a compact representation of data is powerful enough to compute cluster center, radius and diameter, L^1 and L^2 distances between cluster centers, average inter-cluster distance, average intra-cluster distance, etc. If we keep count, sum of attribute values, and sum of squared attribute values, we would be able to compute most linear and quadratic statistical functions directly from the aggregate measures.

3. Enabled Applications

Data aggregation tree may provide a common density representation of large relational data for a variety of decision making applications. Multi-resolution data aggregation offers a new format of data input and opens a new arena of research in data reporting, analytical processing, data mining, and data exploration. One important issue is how to process efficiently data aggregation queries, OLAP queries, and data mining queries as index-only queries on data aggregation trees. There are also other operations (for example, local magnification and brushing in visual data exploration) that issue more complex queries to be answered by accessing data aggregation trees.

Multi-resolution data aggregation brings unique opportunities to data mining algorithms and techniques. Specifically, we are interested in developing density-based and grid-based data mining techniques using multi-resolution data aggregation as data input. For this purpose, some existing data classification techniques as well as hierarchical and grid-based approaches for data clustering may potentially be extended to work with multi-resolution data aggregation. These areas deserve further investigation.

Issues remaining are how to reduce the I/O cost and how to optimize these queries with proper buffer management and pre-fetching strategy. One way to improve the response time of data access is to make database systems act in advance of the user action. Specific tasks include effective memory management and buffer replacement strategies, and data pre-fetching and caching. Existing techniques in query processing and buffer management may still be useful in the context of data aggregation tree.

4. Visual Data Exploration: A Case Study

Large relational data challenge visual data exploration in terms of both data size and dimensionality. A fundamental problem is the conflict between a large number of data records and the user's requirement for interactivity. For visual data exploration, data aggregation tree provides a good representation of data where data exploration queries can

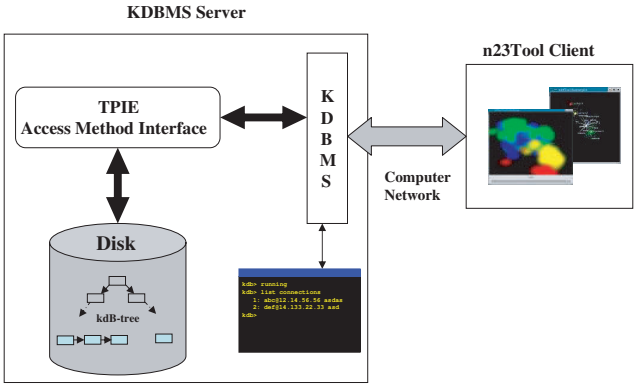
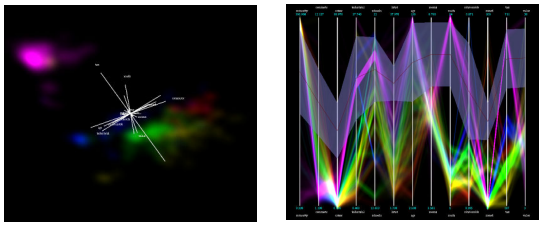


Figure 3. n23Tool system architecture.

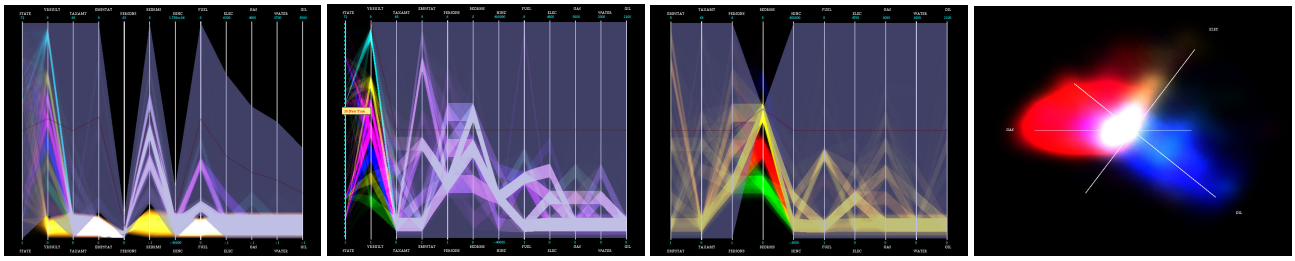


(a) Footprint splatting. (b) Density-based ||-coords.

Figure 4. Enabling visualization techniques.

be expressed as index-only queries. From the visualization perspective, data aggregates are high dimensional volume data where each data aggregation entry represents a high dimensional hyperrectangle with aggregate measures of data points in the hyperrectangle. To visualize aggregated data, we have combined [12] grand tour and footprint splatting and have extended parallel coordinates to a density-based version. Both techniques can visualize data by directly accessing data aggregates stored in internal pages of a data aggregation tree. Zooming is supported through accessing data aggregations in internal nodes at different levels of the tree.

To demonstrate the feasibility of this idea, we have developed a software system n23Tool for visual exploration of large relational data sets. Figure 3 depicts its current client/server system architecture. On the server side, we use kdB-tree as an external high dimensional index to organize multiresolution data aggregations, to support visual browsing and zooming, and to facilitate range queries for user visual interaction. The kdB-tree Management System (KDBMS) implements the tree index on top of TPIE [1], which provides a set of template classes and functions for efficient disk I/O. On the visualization client side, two visualization techniques, footprint splatting and density-based parallel coordinates, are extended and integrated to accept aggregated data. Figure 4 shows screen snapshots of the two visualization techniques on the Boston housing data set.



(a) Overview of all housing units. (b) Outliers eliminated. (c) Old houses in the state of NY. (d) Energy usages of 3-bedrooms.

Figure 5. Screen snapshots as we drill-down the 1% PUMS housing unit data of US Census 2000.

Opacity of each visual element (voxel in volume rendering and horizontal band in parallel coordinates) was assigned as a function of aggregate measures (usually count of data records) in the corresponding data hyperrectangle.

As an example, Figure 5 gives four screen snapshots of visualizing 1.25 million housing unit records in the 1% Public Use Microdata Sample files made public by the US Census 2000. Twelve variables (including four utility usage variables — electricity, gas, water, and oil) have been chosen. The first three screen snapshots visualize data in parallel coordinates, each of which shows a gray data selection band across all coordinates. Each data selection band specifies a query region. The query region is used to retrieve a subset of data, which are visualized in the next screen snapshot.

An important issue of future research is to evaluate existing multidimensional data visualization techniques for their effectiveness to convey multi-resolution data aggregation information. Other issues include interactive picking and brushing, distortion techniques, and so on.

5. Summary

Large relational data bring fundamental challenges to analytical data processing. This paper addresses these challenges and proposes to use multi-resolution data aggregation piggybacked on multidimensional tree index as a generic common representation of data for decision support applications. This paper further investigates techniques in analytical processing, data mining, and data exploration to take advantage of this multi-resolution data representation. In particular, it studies interactive visual exploration of large relational data, which is a key and somewhat neglected area in data mining and information visualization.

References

[1] L. Arge, O. Procopiuc, and J. S. Vitter. Implementing I/O-efficient data structures using TPIE. In *Proc. 10th European Symp. Algorithms*, pp. 88–100, Rome, Italy, Sept. 2002.

[2] K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cube. In *Proc. ACM SIGMOD Conf. Management of Data*, pp. 359–370, June 1999.

[3] A. Casali, R. Cicchetti, and L. Lakhal. Extracting semantics from data cubes using cube transversals and closures. In *Proc. ACM Inter. Conf. Knowledge Discovery and Data Mining*, pp. 69–78, Washington, DC, Aug. 2003.

[4] M. Ester, J. Kohlhammer, and H.-P. Kriegel. The DC-tree: A fully dynamic index structure for data warehouses. In *Proc. Inter. Conf. Data Engineering*, pp. 379–388, San Diego, CA, Mar. 2000.

[5] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*, 1(1):29–53, Mar. 1997.

[6] J. Han, J. Pei, G. Dong, and K. Wang. Efficient computation of iceberg cubes with complex measures. In *Proc. ACM SIGMOD Conf. Management of Data*, pp. 1–12, Santa Barbara, CA, June 2001.

[7] C.-T. Ho, R. Agrawal, N. Megiddo, and R. Srikant. Range queries in OLAP data cubes. In *Proc. ACM SIGMOD Conf. Management of Data*, pp. 73–88, Tucson, AZ, May 1997.

[8] J. T. Robinson. The K-D-B-tree: A search structure for large multidimensional dynamic indexes. In *Proc. ACM SIGMOD Conf. Management of Data*, pp. 10–18, Apr. 1981.

[9] N. Roussopoulos, Y. Kotidis, and M. Roussopoulos. Cubetree: organization of and bulk incremental updates on the data cube. In *Proc. ACM SIGMOD Conf. Management of Data*, pp. 89–99, Tucson, AZ, May 1997.

[10] G. Sathe and S. Sarawagi. Intelligent rollups in multidimensional OLAP data. In *Proc. Inter. Conf. Very Large Databases*, pp. 531–540, Roma, Italy, Sept. 2001.

[11] D. Xin, J. Han, X. Li, and B. W. Wah. Star-cubing: Computing iceberg cubes by top-down and bottom-up integration. In *Proc. Inter. Conf. Very Large Databases*, pp. 476–487, Berlin, Germany, Sept. 2003.

[12] L. Yang. Visual exploration of large relational datasets through 3D projections and footprint splatting. *IEEE Trans. Knowledge and Data Engineering*, 15(6):1460–1471, Nov./Dec. 2003.

[13] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proc. ACM SIGMOD Conf. Management of Data*, pp. 103–114, Montreal, Canada, June 1996.