A Survey on Efficient Data Cube Computation Methodologies for Business Intelligence

[1]Miss. Vaidya Vijayshri Dattatray, [2]Mr. Nibe Abhishek Annasaheb
[1]Lecturer, [2]Lecturer
[1]Department of Computer Technology, [2]Department of Computer Technology
[1 2] P. Dr. Vitthalrao Vikhe Patil Polytechnic, Loni Ahmednagar Maharashtra-India

***Abstract*-Business Intelligence field [4] provides the solutions for multidimensional analysis using large amount of data from data warehouse. To make the decisions from the raw data available in data warehouse may take huge time. So to make interactive analysis from huge data, multidimensional data cubes can be used. The data cube contains the cells (cuboids) which are aggregation of the data. To access the whole data every time for analysis is too expensive. In this paper we can study different materialization techniques [6] which can compute the data cells at once only. So no need to access whole data, instead one can use these precomputed data cells (cuboids). For efficient data cube computation,[2][5] materialization can starts from base cuboid to apex (target) cuboid in such a way that intermediate cuboids takes less time and space to generate. In materialization we can compute all cuboids or specific cuboids required for analysis purpose.**

***Keywords:* business intelligence, data warehouse, multidimensional data cubes, cuboids, materialization**

1. INTRODUCTION

Data warehouse contains the multidimensional data cubes [7]. Data analysis depends upon the efficient data cube computation [2][5]. Data cube is a lattice of cuboids. Cuboid means aggregation of data, also referred as group-by's.[1] The set of group-by's forms a lattice of cuboids defining a data cube.
Taking the three attributes, city, item, and year, as the dimensions for the data cube, and sales in dollars as the measure, the total number of cuboids, or group by's, that can be computed for this data cube is 23=8.
The possible group-by's are the following: {(city, item, year), (city, item), (city, year), (item,year), (city), (item), (year), ()} where ()means that the group-by is empty (i.e., the dimensions are not grouped). These group-by's form a lattice of cuboids for the data cube, as shown in Figure.

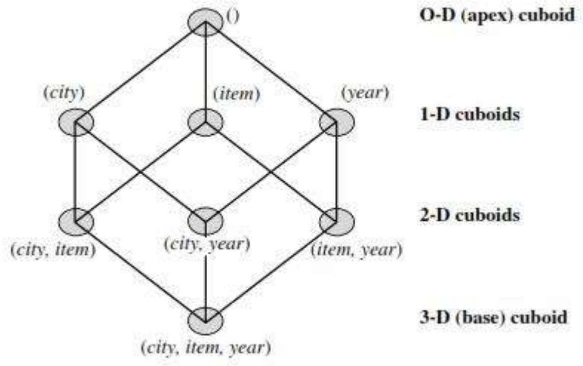

Fig 1: Lattice of cuboids

The base cuboid contains all three dimensions, city, item, and year. It can return the total sales for any combination of the three dimensions.
The apex cuboid, or 0-D cuboid, refers to the case where the group-by is empty. It contains the total sum of all sales.
The base cuboid is the least generalized (most specific) of the cuboids.
The apex cuboid is the most generalized (least specific) of the cuboids, and is often denoted as all.

2. MATERIALIZATION (PRECOMPUTATION OF DATA CUBE)

There are three choices for data cube materialization given a base cuboid:

a. No materialization:
In this technique, cuboids are not precomputed. This leads to computing expensive multidimensional aggregates that can takes more time and money.

b. Full materialization:
The technique can precompute all the cuboids available in multidimensional data cubes. The resulting lattice of computed cuboids is referred as the full cube.
This choice typically requires huge amount of memory space in order to store all of the precomputed cuboids.

Full Cube Computation:
For full cube computation, multi way aggregation method is used to compute full data cube. Aggregation is done by partition array into chunks and compute aggregate by visiting cube cells.

Ex: Consider a 3-D data array containing the three dimensions A, B, and C.

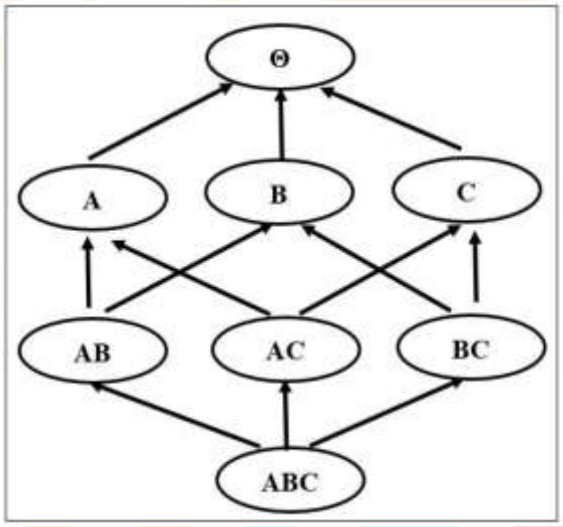

Fig 2: Full materialization

Full materialization [6] of the corresponding data cube involves the computation of all the cuboids defining this cube. The resulting full cube consists of the following cuboids:
i. The base cuboid, denoted by ABC (from which all the other cuboids are directly or indirectly computed). This cube is already computed and corresponds to the given 3D array.
ii. The 2-D cuboids, AB, AC, and BC, which respectively correspond to the group-by's AB, AC, and BC. These cuboids must be computed.
iii. The 1-D cuboids, A, B, and C, which respectively correspond to the group-by's A, B, and C. These cuboids must be computed.
iv. The 0-D (target) cuboid, denoted by all, which corresponds to the group-by (); that is, there is no group-by here. This cuboid must be computed. It consists of only one value. If, say, the data cube measure is count, then the value to be computed is simply the total count of all the tuples in ABC.

c. Partial materialization:
Selectively compute a proper subset of the whole set of possible cuboids.
Compute a subset of the cube, which contains only those cells that satisfy some user specified criterion.

It uses subcube where only some of the cells may be precomputed for various cuboids.
Iceberg Cube Computation:
Contains only those cells of data cube that meet an aggregate.
It is called iceberg, as it contains only some cells of full cube (tip of an iceberg). Purpose of this cube is to identify and compute those values which are required for decision support [3] queries.
Iceberg cube can be specified with an SQL query [8], as shown

compute cube sales iceberg as
select month, city, customer_group, count(*)
from salesInfo
cube by month, city, customer_group
having count(*) >= min_sup

3. STRATEGIES: DATA CUBE COMPUTATION: [7]

a. Sorting, hashing and grouping:
In cube computation, aggregation is performed on the tuples (or cells) that share the same set of dimension values.
Thus, it is important to explore sorting, hashing, and grouping operations to access and group such data together to facilitate computation of such aggregates.
Ex: To compute total sales by branch, day, and item, for example, it can be more efficient to sort tuples or cells by branch, and then by day, and then group them according to the itemname.

b. Simultaneous aggregation and caching intermediate results:
In cube computation, it is efficient to compute higher-level aggregates from previously computed lower-level aggregates, rather than from the base fact table. Simultaneous aggregation from cached intermediate computation results may lead to the reduction of expensive disk input/output (I/O) operations.
Ex: To compute sales by branch, for example, we can use the intermediate results derived from the computation of a lower-level cuboid such as sales by branch and day.

c. Aggregation from the smallest child:
If a parent cuboid has more than one child, it is efficient to compute it from the smallest previously computed child cuboid.
Ex: To compute a sales cuboid, Cbranch, when there exist two previously computed cuboids, C{branch,year} and c{branch,item}, it is obviously more efficient to compute Cbranch from the former than from the latter if there are many more distinct items than distinct years.

d. The Apriori pruning method:
The method can trim or delete the unwanted (infrequent) data. Apriori requires a priori knowledge to generate the frequent itemsets and involves two time consuming pruning steps to exclude the infrequent candidates and hold frequents. It is used to reduce the computation of iceberg cubes.

CONCLUSION

In this paper we have studied efficient data cube computation using different materialization methods and strategies. Materialization of cuboids is an essential query optimization technique for decision-support systems. The materialization in query optimization system can precompute all possible cuboids or required cuboids for decision making, which can act as business intelligence technique for fast decision making as well as saving the storage space.

REFERENCES

[1] S. Chaudhari and K. Shim. Including group- By in Query Optimization. In Proceedings of the Twentieth International Conference on Very Large Databases (VLDB), pages 354-366, Santiago, Chile, 1994.

[2] V. Harinarayan, A. Rajaraman, and J. D. Ullman Implementing Data Cubes Efficiently. A full version of this paper. At http: //db.stanford.edu/pub/harinarayan /1995/cube.ps

[3] A. Radding, Support Decision Makers with a Data Warehouse. In Datamation, March 15, 1995.

[4] A. Abelló, J. Darmont, L. Etcheverry, M. Golfarelli, J.N. Mazón, F. Naumann, T. Pedersen, S. B. Rizzi, J. Trujillo, P. Vassiliadis, et G. Vossen, « Fusion Cubes: Towards Self-Service Business Intelligence », Int. J. Data Warehous. Min., vol. 9, n 2, p. 66-88, 32 2013.

[5] Harinarayan V, Ullman RA (1996) Implementing data cubes efficiently. In: ACM SIGMOD international conference on management of data, ACM Press, New York, pp 205–216

[6] Stefanovic N, Han J, Koperski K (2000) Object-based selective materialization for efficient implementation of spatial data cubes. In: IEEE transaction on knowledge and data engineering, pp 938–958

[7] Antoaneta I, Boris R (2004) Multidimensional models constructing data cube. In: international conference on computer systems and technologies-CompSysTech'2004, vol 5, pp 1–7

[8] A. Gupta, V. Harinarayan, and D. Quass. Aggregate-Query Processing in Data Warehousing Environments. In proceedings of the 21st International VLDB Conference, pages 358-369, 1995.