Optimal Path Finding Algorithm For Massive Queries In Lattice of Cuboids

Prashant R	Suman Mann Maharaja Surajmal Institute of Technology, GGSIPU, New delhi-1100058, India

Abstract

Multidimensional analysis requires the computation of many aggregate functions over a large volume of collected data. To provide the various viewpoints for the analysts, these data are organized as a multi-dimensional data model called data cubes. Each cell in a data cube represents a unique set of values for the different dimensions and contains the metrics of interest. The different abstraction and concretization associated with a dimension may be represented as a lattice. The focus is to move up and drill down within the lattice using an algorithm with optimal space and computation. In the lattice of cuboids, there exist multiple paths for summarization from a lower to an upper level of the cuboid. The alternate paths involve different amounts of storage space and different volume of computations. This objective of this paper is to design an algorithm for formal analysis leading towards detection of an optimal path for any two given a valid pair of cuboids at different levels. The algorithm is proposed based on lowest first method for selection of the optimal path.

1. Introduction

A data warehouse is a repository of integrated information available for querying and analysis. The information in the data warehouse is stored in the form of the multidimensional model. The multidimensional model viewdata in the form of the data cube. Data cube computes the aggregates along all possible combinations of dimensions. It is defined by dimensions and facts. Dimensions are the entities with respect to which an organization wants to keep records [4]. Facts are the numerical measures/ quantities by which we want to analyze the relationship between dimensions. In general terms, we consider data cube as 3-D geometric structures, but in data warehousing, it is n-dimensional. The data cube is a metaphor for multidimensional data storage. Each cell of datacube shows a specific view in which users are interested. Given a set of dimensions, we can generate a cuboid for each of the possible subsets of the given dimensions which result in a lattice of the cuboid. Figure 1 shows a lattice of cuboid for the dimensions of time, product, market and supplier. For the number of dimensions, we may find 2n cuboid and the main challenge is to understand how the cuboids are related to each other[3]. As shown in figure1 different paths are available for a particular cuboid. These paths consider the different amount of storage space and time computation.

2. Definition and Properties of Lattice Theory

Some important definitions related to lattice are given below:

1. Lattice: A partial order set (POS) (A, ≤) is called a lattice if, x, y ε A, there exist sup(x, y) and inf (x, y) [27]. For supermum we use the symbol and for infimum we use the symbol .In the lattice theory, these operations are called binary operations. 2. POS: Partially ordered set is a set with a binary relation ≤ that, for any x, y, and z, satisfies the following conditions: (1) $x \leq x$ (reflexivity); (2) if $x \leq y$ and $y \leq x$, then $x = y$ (anti_symmetry); (3) if $x \leq y$ and $y \leq z$, then $x \leq z$ (transitivity).

If, for a POS (A, ≤), x, y ε A: x ≤ y or y ≤ x, then this set is called a linearly ordered set, or chain. 3. Supremum or Least Upper Bound: Let (A, R) be a POS. An element l ε A is called supremum of a and b in A if and only if i. aRl and bRl i.e. l is the upper bound of a and b.
ii. If an element l' ε A such that aRl' and bRl'then lRl'
That is if l' is another upper bound of a and b then l' is also the upper bound of l. Thus l is the least upper bound of a and b. 4. Greatest Lower Bound(GLB) or Infimum Let (A, R) be a POS. An element l ε A is called infimum of a and b in A if and only if i. lRa and lRb i.e. l is the lower bound of a and b. ii. If there exist one element l' ε A such that l' is also a lower bound of a and b then l'Rl that is if l'is also a lower bound of a, b then l' is the lower bound of l also i.e. l is the greatest of all lower bounds of a and b. That is if l' is another upper bound of a and b then l' is also the upper bound of l. Thus l is the least upper bound of a and b. 5. Roll Up & Drill Down: The roll-up operation performs aggregation on a data cube, may be climbing up a concept hierarchy for a dimension or by dimension reduction. Drilldown navigates from less detailed data to more detailed data. It is the reverse of roll-up operation. Drill down is realized by stepping down a concept hierarchy or adding a new dimension.

2.1 Some Characteristics of Lattices

If A is any lattice, then for any a, b, c ε L the following properties hold:

- $\overbrace{(a \cup a = a) \wedge (a \cap a = a)}^{Idempotent\ Law}$
- $\overbrace{(a \cup (b \cup c) = (a \cup b) \cup c) \wedge (a \cap (b \cap c) = (a \cap b)) \cap c}^{Associative\ Law}$
- $\overbrace{(a \cup b = b \cup a) \wedge (a \cap b = b \cap a)}^{Commutative\ Law}$
- $\overbrace{(a \cap (a \cup b) = a) \wedge (a \cup (a \cap b) = a)}^{Absorption\ Law}$

3. Lowest First Approach

We first store user input as source cuboid and target cuboid, we then create what we call a difference set which is a set containing attributes present in the target cuboid but not in the source cuboid. Once we have the difference set we sort the set based on the value of the each attributes. The optimal path then is to start from source cuboid and visiting each attribute in sorted fashion getting each level in order, So if we have a lattice of cuboids having attributes A with values 10, B with values 5, C with values 20, D with values 2, E with values 30 and we have to find optimal path from cuboid <A> to cuboid <A,B,C,D,E> then our difference set would be <B,C,D,E> which we sort based on values to get [D,B,C,E] and the path then comes out to be <A> => <A,D> => <A,D,B> => <A,D,B,C> => <A,D,B,C,E> which is optimal when matched with the brute-force answer.
The largest benefit of this approach is the time complexity of querying. For a single path finding operation the time complexity is O(nlogn) where n is the number of attributes but with a preprocessing of O(nlogn) for multiple queries the query time can be reduced to O(n) which is far lower than the typical exponential complexities.

4. Simple Mathematical proof

consider a lattice of cuboids having a total of 6 attributes Q, P, M, S, U given to us

that 0<=Q <=P<=S <=U <=M and we have to find optimal path from <Q> to <Q, P , S ,U , M>, Now a path is optimal if the intermediate cuboids generated while traversal have minimum aggregation so we write a generic expression for path value and try to optimize it, which is for the current case $A_1 * A_2 * A_3 + A_1 * A_2 + A_1$ and now we need to find an assignment order between the variables in out generic expression and the sorted difference set [P,S,U,M]which gives us the optimal value for the expression. To minimize the whole expression we minimize each term in it while making sure every move is valid, So starting from the last term we have

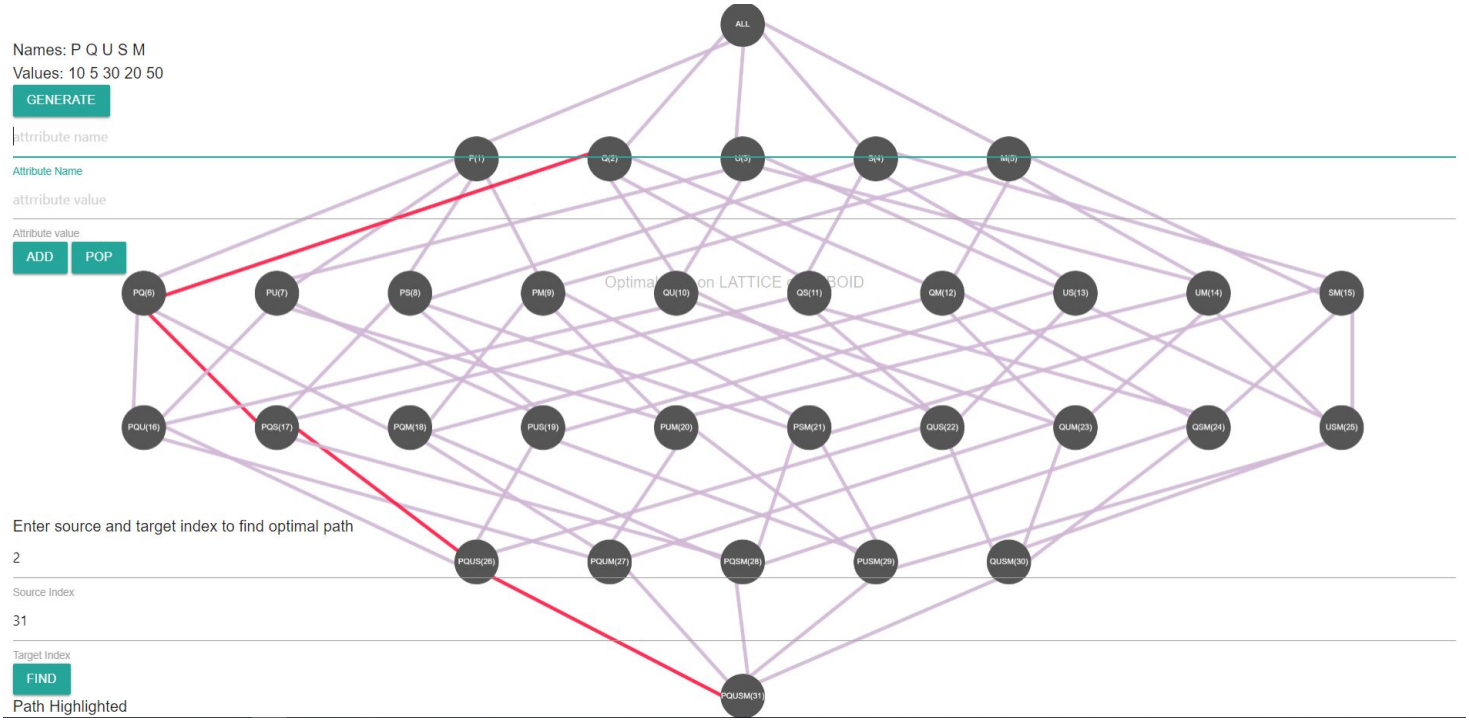
· A_1 is minimum when its value is P. So, A_1 = P (as P has lowest value in our difference set followed by S, U and M)

· $A_1 * A_2$ is minimum when its value is P*S and we already have A_1 =P so A_2 = S.

· $A_1 * A_2 * A_3$ is minimum when its value is P*S*U and we already have A_1 =P, A_2=S so A_3=U

and our optimal assignment order is A_1 =P, A_2=S, A_3=U and the path becomes <Q> => <Q,P> => <Q,P,S> => <Q,P,S,U> => <Q,P,S,U,M>
which gives us lowest possible value for any possible path from source cuboid to target cuboid.

Similarly we can prove its valid for any number of attributes which is easy to do using principle of mathematical induction.
Below is the optimal path highlighted generated by a web implementation of the above algorithm available at https://latticeofcuboid.ml/



5. Conclusion

In this paper we have proposed an algorithm which has a preprocessing time complexity of O(nlogn) and query time complexity of O(n).