2009 Fourth International Conference on Computer Sciences and Convergence Information Technology

Optimal Space and Time Complexity Analysis on the Lattice of Cuboids using Galois Connections for Data Warehousing

[1]*Soumya Sen,* [2]*Nabendu Chaki*
A. K. Choudhury School of Information Technology
University of Calcutta
Kolkata 700009, India
[1]iamsoumyasen@gmail.com, [2]nabendu@ieee.org

Agostino Cortesi
Dipartimento di Informatica
Università Ca' Foscari di Venezia
I-30170 Venezia, Italy
cortesi@unive.it

***Abstract*— In this paper, an optimal aggregation and counter-aggregation (drill-down) methodology is proposed on multidimensional data cube. The main idea is to aggregate on smaller cuboids after partitioning those depending on the cardinality of the individual dimensions. Based on the operations to make these partitions, a Galois Connection is identified for formal analysis that allow to guarantee the soundness of optimizations of storage space and time complexity for the abstraction and concretization functions defined on the lattice structure. Our contribution can be seen as an application to OLAP operations on multidimensional data model in the Abstract Interpretation framework.**

Keywords-Abstract interpretation, lattice, data warehouse, Galois connection, OLAP

I. INTRODUCTION

Data Warehouses and OLAP tools are based on multidimensional data model. This model views data in the form of a data cube. It is a hierarchical structure, used for basic OLAP operations to allow interactive mining at multiple levels of abstraction. A data cube allows data to be modeled and viewed in multiple dimensions. It is defined by dimension and facts. In general terms, dimensions are the perspective or entities with respect to which an organization wants to keep records. Each dimension may have a table associated with it, called a dimension table, which further describes the dimension. A multidimensional data model is typically organized around a central theme or fact. Facts are numerical measures. The fact table contains names of facts, or measures as well as keys to each of the related dimension tables. The data cube is a metaphor for multidimensional data storage. The actual physical storage of such data may differ from its logical representation.

In data warehousing literature, a data cube is often referred to as **cuboid**. Given a set of dimensions, we can generate a cuboid for each of the possible subsets of the given dimensions. The result would form a lattice of cuboids forming a data cube for the given dimensions. The cuboid that holds the lowest level of summarization is called the **base cuboid**. Going upper in the hierarchy, we reach to 0-D cuboid, which holds the lower level of summarization is called the **apex cuboid.** The apex cuboid is denoted by all. The following is a figure of 4-D data cube, where each cube represents different degree of summarization.

In figure 1, there are four different attributes labeled as 1, 2, 3, and 4. These attributes within a dimension table form a lattice. In the above lattice of cuboids, there exist multiple paths for summarization from a lower to an upper level cuboid. The alternate paths involve different amounts of storage space and different volume of computations. The objective of this paper is to design a framework for formal analysis leading towards detection of an optimal path for any two given valid pair of cuboids at different levels.

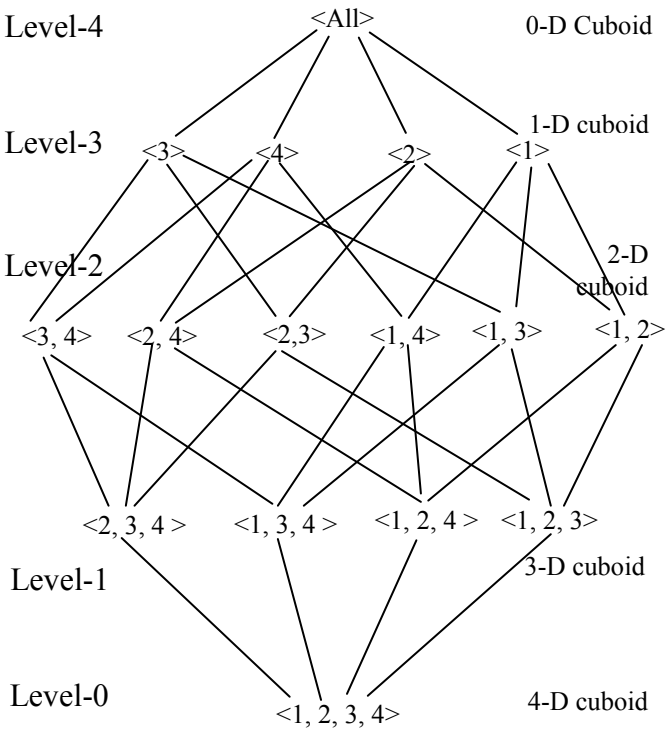

Figure 1. The Lattice of Cuboids

There have been a number of works on computational aspects of roll-up, drill-down and other OLAP operations [1, 2, 3, 6, 8, 9, 10, 11]. In [1], the hierarchy of group-by operations has been used to analyze the structure of CUBE computation. The focus had been on a special case of the aggregation problem - computation of the CUBE operator. The CUBE operator requires computing group-bys on all

978-0-7695-3896-9/09 $26.00 © 2009 IEEE
DOI 10.1109/ICCIT.2009.185

possible combinations of a list of attributes, and is equivalent to the union of a number of standard group-by operations. The algorithms in [1] extend sort-based and hash-based grouping methods with several optimizations aspects. They have used a heuristic based approach for allocation of cuboids.

Time-series data [3] often defines the hierarchy of multi-dimensional data. However, the hierarchy and dependencies between the cuboids is not necessarily on time series alone [6, 8, 11]. There has been work on the efficiency of OLAP operations on spatial data in a data warehouse [9]. In [10], efficiency of roll-up and drill-down on the data collection stored in the relational database management system has been considered. A method using labeling on a tree structure has been proposed and found to be advantageous over more conventional mapping tables to store the hierarchy. The approach however does not attempt to find an optimal solution amongst the alternates.

A pioneering work on the abstract interpretation for static analysis establishes it as a unified approach to apparently unrelated program analysis techniques [4]. More recently, abstract interpretation has also been used for software verification [5, 14]. However, the primary motivation of using abstract interpretation for analysis of OLAP operations as proposed in this paper originates from a couple of works reported in [12, 13].

This paper is organized as follows. In section 2 we present the definitions of Galois Connection and relevant terms before introducing a brief outline of Roll-Up and Drill-down, two major OLAP operations. In section 3, we establish a relation between these two OLAP operations and Galois Connection. Section 4 describes the proposed algorithm to traverse between two data cube of valid dimension. Section 5 illustrates an example using the proposed algorithm and section 6 state the basic properties on the model before we conclude in section 7.

II. DEFINITIONS AND OLAP OPERATIONS

A. Partial Ordering Relation:

A binary relation is partial ordering relation if it is reflexive, anti-symmetric and transitive. A set A, together with a partial ordering relation on A, is called partially ordered set and is denoted by (A,R). In the literature, a partially ordered set is also abbreviated as **poset.**

B. Upper and Lower Bounds

Let a and b be two elements in a partially ordered set (A,≤).
An element c is said to be an **upper bound** of a and b, if a ≤ c and b ≤ c.
An element c is said to be a **least upper bound** of a and b if c is an upper bound of a and b, and if there is no other upper bound d of a and b, such that d ≤ c.
An element c is said to be a **lower bound** of a and b if c ≤ a and c ≤ b, and an element c is said to be a **greatest lower bound** of a and b if c is a lower bound of a and b, and if there is no other lower bound d of a and b such that c ≤ d.

C. Lattice

A partially ordered set is said to be a **lattice** if every two elements in the set have a unique least upper bound and a unique greatest lower bound.

D. Galois Connection

Let C and D be complete lattices, and consider two functions **γ: D → C and α: C → D.** The tuple $G_{CD} = (\gamma, C, D, \alpha)$ is a Galois connection, if the following condition holds

$$\forall c \in C \text{ and } \forall d \in D: \alpha(c) \leq_D d \text{ if and only if } c \leq_C \gamma(d).$$

E. Tag Number and Tagged Value

We propose this concept in this section to be used in the algorithm described later in section 4 of the paper. In order to identify the dimensions for the cuboids uniquely, we associate an integer value with them. This value is in increasing order starting from 0. We define this number as the **tag number** for that dimension.

The tag numbers are taken as exponents in a sum of power of 2 to compute the **tagged value** for cuboids of specific dimensions at a level.

As for example, let's assume that the user provides the following four dimensions: A, B, C and D. Let, the integer values 0, 1, 2 and 3 be associated with A, B, C and D respectively. These are **tag numbers** for the respective dimensions.

The **tagged value** for a cuboid is calculated by taking the sum of **tag numbers** for associated dimensions taken into an exponent of 2. Thus **tagged value of** <A, B, D> would be $2^0+2^1+2^3= 11$ and that for <A, B, C> is $2^0+2^1+2^2 = 7$

F. Roll-Up and Drill-Down Operations

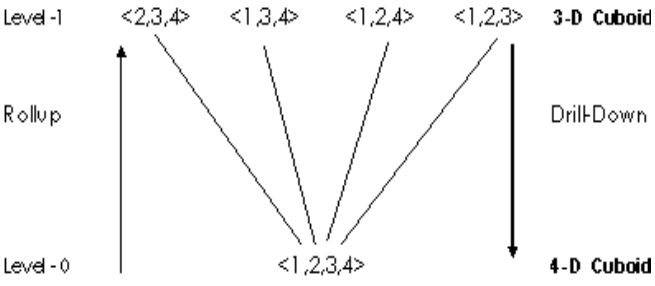

Figure 2. The Cuboid Structure Hierarchy

In Data Warehouse literature, a data cube is referred to as cuboid. Given a set of dimensions we can derive cuboids for

each of the possible subsets of the given dimensions. A partial order amongst the cuboids is observed in the sense that there exists cuboid Ci that may or may not be derived from some other cuboid Cj. The resulting Poset, in fact, would form a lattice of cuboids [7]. Each level of the cuboid going upper in hierarchy gives summarized data reducing one dimension from the previous level. The top level gives a single value which is a summarization of all the dimensions.

The two major OLAP operations that are applied amongst the cuboids are Roll-Up and Drill-down. Roll-up performs aggregation on a data cube, either by climbing up a concept hierarchy for a dimension or by dimension reduction. When roll up is performed by dynamic reduction, one or more dimensions are removed from the given cube.

Drill-down is the reverse of roll-up. It navigates from less detailed data to more detailed data. Drill-down can be realized by either stepping down a concept hierarchy for a dimension or introducing additional dimensions.

III. GALOIS CONNECTION ON CUBOID STRUCTURE

The Roll Up and Drill down operations may be expressed in term of concretization and abstraction function of a Galois connection.

Given a data cube, a family of **abstractions** may be defined as

$C_{i+1} = \alpha(C_i, k)$ where k is one of the existing (N-i) dimensions of any cuboid C_i in level i. Cuboid C_{i+1} will have remaining (N-i-1) dimensions after the abstraction operation α.

Given a data cube, a family of **concretization functions** may be defined similarly as

$C_i = \gamma(C_{i+1}, k)$ where k is one of the existing (N-i-1) dimensions of any cuboid C_{i+1} in level (i+1). Cuboid C_i will have (N-i) dimensions after the concretization operation γ.

Here, *0* is the lowest level where cuboid is of the form n-D cuboid and *n* is the highest level where cuboid is of the form 0-D cuboid.

Given a cube, any **abstraction** can be thought of as roll up. In roll up, as we go up in the hierarchy, the number of dimensions in cuboids reduces. This function focuses on more aggregated information. At the lowest level, total numbers of cuboids are multiplication of the value of all the dimensions, i.e. if there are four dimensions, each with C1, C2, C3, C4 value, then total number of cuboids = C1×C2×C3×C4. At the top level, total number of cuboid is always 1.

Given a cube, any **concretization** can be thought of as drill down operation. This function focuses on more detailed information, as we are going down of the hierarchy number of dimension in cuboid increases.

By means of these abstraction and concretization functions, we might formalize soundness conditions for the usual operations on the lattice of cuboids, namely "slice and dice", "pivot", "drill across", and "drill through".

The possible number of combinations of dimensions in each level r will be ${}^{n}c_{r}$, where n be the maximum dimension and r is the current dimension (ranging from 0 to n, where 0 is the dimension of the top level and n is the dimension at the lowest level).

IV. GENRATING OPTIMAL OPERATIONAL PATH

In section 3 we identified abstraction and concretization function on the lattice of data cube. Our aim is to derive an algorithm, which would use these two functions to obtain an optimization technique on intermediate storage space requirement and on time complexity of the lattice of data cube.

In the lattice of data cube, we may face a requirement to reach to a particular dimension of data cube from any other dimension of data cube. In this algorithm, we want to get a path, which requires the generation of minimum number of intermediate data cube, which eventually result in better space complexity. The less number of intermediate cube generation also implies better time complexity in this context, as less time would be required to generate less number of data cubes.

Starting from lower level of hierarchy we can move to upper level of hierarchy through number of ways to find a particular cuboid of certain dimension. The following is an example to show, how the proposed algorithm would work on the lattice of data cube.

e.g. Suppose in a lattice of cuboids we need to find a path which would generate minimum cuboids from <1,2,3,4> to <1,2>. Here we have two alternatives:

$$<1, 2, 3, 4> \longrightarrow <1, 2, 4> \longrightarrow <1, 2> \text{ and}$$

$$<1, 2, 3, 4> \longrightarrow <1, 2, 3> \longrightarrow <1, 2>$$

The proposed algorithm would pick up the path, which would require generation of smallest number of cuboids among all the possible paths.

Procedure SelectPath
Begin

Step 1: Ask the starting dimension pattern of cuboid and the target dimension pattern of cuboid from the user.

Step 2: Check whether the user given dimension patterns of the cuboids are valid or not.
/* At first check whether the starting dimension pattern of cuboid is a subset of the dimension of the base cuboid, then check whether target dimension pattern of cuboid is subset of starting dimension pattern */
If true
go to Step 3

else
Exit

Step 3: The tagged value of starting dimension pattern is stored in queue Q.

Step 4: Repeat Steps 4 to 8 until the target dimension pattern of cuboid exactly matches with the user given dimension of cuboid

Step 5: Apply abstraction function to every cuboid in the current level that contains the target cuboid as subset, to generate cuboid at $(i+1)^{th}$ level. The possible combination of dimension of cuboid in immediate upper level generated from each cuboid in current level is ${}^{n-i}c_{n-i-1}$, where i is the current level.

Step 6: From the abstraction function $C_{i+1} = \alpha(C_i, k)$ we choose that k which gives the minimum number of cuboids and contains the pattern of dimension given by user.
/* The number of cuboids in each pattern of dimension is obtained by multiplying the distinct value of each dimension of that cuboid */

Step 7: In the last step the k (dimension pattern) that has been chosen, for that the corresponding tagged value of a dimension pattern is stored into a queue Q.

Step 8: If the desired pattern is obtained
break from the loop
else
goto step 4

Step 9: Q contains the desired path

End.

V. ILLUSTRATION WITH EXAMPLE

Let's find the optimal path from <A, B, C, D, E> to <A, B>. Say, <A, B, C, D, E> be the base cuboid at the lowest level. We require a solution with minimal best space and time complexity. The key is to choose the path with minimum number of cuboids.
Let dimension A has 10 values, B has 15 values, C has 8 values, D has 12 values and the dimension E has 14 values.

A. Solution using proposed Algorithm

The algorithm at first verifies that both of <A, B, C, D, E> and <A, B> are valid patterns. We put the tagged value for C_0 = <A, B, C, D, E> i.e., **31** in the queue Q.
From C_0 = <A, B, C, D, E> one may generate:
<B, C, D, E> by abstraction $\alpha(C_0, A)$
<A, C, D, E> by abstraction $\alpha(C_0, B)$
<A, B, D, E> by abstraction $\alpha(C_0, C)$
<A, B, C, E> by abstraction $\alpha(C_0, D)$
<A, B, C, D> by abstraction $\alpha(C_0, E)$

Now <A, B, D, E>; <A, B, C, E>; <A, B, C, D> contains the target pattern <A, B>.

The number of cuboids for the pattern <A, B, D, E> is 10×15×12×14 = 25200.
The number of cuboids for the pattern <A, B, C, E> is 10×15×8×14 = 16800.
The number of cuboids for the pattern <A, B, C, D> is 10×15×8×12 = 14400.

In this step we choose <A, B, C, D> as this gives the lowest number of cube. The tagged value of C_1 = <A, B, C, D>, i.e. **15** is put in the queue Q. From C_1 = <A, B, C, D> one may generate

<B, C, D> by abstraction $\alpha(C_1, A)$
<A, C, D> by abstraction $\alpha(C_1, B)$
<A, B, D> by abstraction $\alpha(C_1, C)$
<A, B, C> by abstraction $\alpha(C_1, D)$

Now <A, B, D>, <A, B, C> contains the pattern <A, B>.

The number of cuboids for the pattern <A, B, D> is 10×15×12 = 1800, while that for the pattern <A, B, C> is 10×15×8 = 1200. We put the tagged value of C_2 = <A, B, C>, i.e. **7** in the queue Q. From C_2 = <A, B, C> one may generate

<B, C> by abstraction $\alpha(C_2, A)$
<A, C> by abstraction $\alpha(C_2, B)$
<A, B> by abstraction $\alpha(C_2, C)$

Now we get the target cuboid <A, B> and put the tagged value of <A, B>, i.e. **3** in the queue Q. The tagged values stored in queue are 31, 15, 7, 3.

Therefore, the desired path would be

<A, B, C, D, E> → <A, B, C, D> → <A, B, C> → <A, B>.

VI. PROPERTIES

Lemma 1: Consider a lattice of cuboids of 0 to n levels. Let **$d(x_i)$** denote any set of distinct dimensions selected arbitrarily from at level i of the lattice. Let $|k|$ be the value obtained by multiplying the number of values for each dimension in a particular cuboid. The total number of values in the cuboids at each intermediate level i starting from level 0 would be:

$$\sum_{j=0}^{{}^{n}C_{n-i}} \{|k| : k \in d(x_j)\}$$

Proof: The proof is by induction: the basic case is when i=n, i.e., at the top level, In this case, the total number of cuboids is always 1, independent of the distinct value of dimension. The inductive step is trivial.

Lemma 2: The number of dimensions for a cube in level i is n-i for the base cuboids on n dimensions.

Proof: The base cuboid in level 0 has n dimensions. Each cuboid at level 1 that are derived from the base cuboid by abstraction on a particular dimension are of remaining (n-1) dimensions. In fact, at each upper level, the dimension of the cuboids reduces by 1 while the level increases by 1. Thus, the lemma may be proved trivially by induction from the definition and structure of the lattice as introduced in section 1 of the paper.

Lemma 3: The number of distinct cuboid patterns generated in level (i+1) by abstractions on a given cuboid pattern in the current level i is ${}^{n-i}C_{n-i-1}$.

Proof: The statement is true for the unit cube at level n as the unit cube may be obtained by abstraction of any of the n one dimensional cubes at level n-1.

Let, *d* be the number of dimension of a cuboid in the current level. The immediate upper level would have the cuboids with dimension *d-1*, i.e. we need to choose any *(d-1)* dimension from the existing *d* dimension.

Thus the possible ways of choosing a cuboid for upper level from the cuboid at current level is ${}^{d}C_{d-1}$.

Again, from the statement of lemma 2, the number of dimensions of the cubes for current level i would be d=n-i.

By combining the last two statements, we get that the number of distinct cuboids at level i is exactly ${}^{n-i}C_{n-i-1}$.

Lemma 4: The number of cuboid patterns produced by concretization from a particular cuboid in the current level i is i.

Proof: By the statement of lemma 2, the number of dimensions in current level of cuboids is n-i. To generate the cuboid at immediate lower level we can add any one dimension from the unused dimensions, i.e., the dimension that is not present in current cuboid. Clearly the total unused dimensions are (n-(n-i))=i. We can choose any one of them to generate cuboid at lower level. So the number of cuboid patterns produced by concretization from a particular cuboid in the current level i is i.

Lemma 5: The total number of distinct cuboid patterns generated in level (i+1) by abstractions in level i is $T_{i+1} = (T_{i*}\,{}^{n-i}C_{n-i-1})/(i+1)$, where i = 0 to (n-2), $T_0 = 1$ and d be the number of dimensions of cuboid in current level.

Proof: In lemma 3, we have proved the number of cuboid patterns that can be generated in the immediate upper level from a particular cuboid in the current level is ${}^{n-i}c_{n-i-1}$. As there exists T_i number of unique dimension pattern of cuboids in current level, the total number of dimension patterns that is generated is $T_{i*}\,{}^{n-i}c_{n-i-1}$.

Now, each pattern in upper level is generated in (i+1) ways. Thus distinct dimension patterns in the upper level is generated in $(T_{i*}\,{}^{n-i}c_{n-i-1})/(i+1)$ ways.

VII. CONCLUSION

A data cube allows data to be modeled and viewed in multiple dimensions. It is defined by dimension and facts. In this paper, the techniques of abstract interpretations have been used to analyze the roll-up and drill-down operations on multidimensional data cube. This has lead to the derivation of optimal aggregation (roll-up) and counter-aggregation (drill-down) methodology defined on the lattice structure. The proposed algorithm is based on the two operations towards finding the optimized path to traverse between two data cube of valid dimension in terms of intermediate cuboid sizes. Abstract interpretation may also be useful when the dimension of the lattice of cuboids explodes, to keep the computational costs acceptable, by applying widening techniques [13]. This is a research topic that has not been investigated yet and represents the focus of our future work.

REFERENCES

[1] S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan and S. Sarawagi; "On the computation of multidimensional aggregates"; Proc. of 1996 International Conference on Very Large Data Bases VLDB'96.

[2] R. Agrawal and R. Srikant; "Fast algorithms for mining association rules"; Proc. of Int'l Conf. on Very Large Data Bases. VLDB'94.

[3] Chen, Y., Dong, G., Han, J., Wah, B. W., and Wang, J.; "Multi-dimensional regression analysis of time-series data streams"; Proc. of the 28th int'l conf. on Very Large Data Bases, pp. 323-334, 2002.

[4] Patrick Cousot, Radhia Cousot; "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints"; Proc. of the 4th Annual ACM SIGPLAN-SIGACT Symp. on principles of prog. lang., pp. 238—252, 1977.

[5] Patrick Cousot; "The Verification Grand Challenge and Abstract Interpretation"; Verified Software: Tools, Theories, Experiments, LNCS 4171, Springer-Verlag, Berlin, pp. 227—240, Dec. 2007.

[6] Gray, J., Bosworth, A., Layman, A., and Pirahesh, H.; "Data cube: A relational operator generalizing group-by, cross-tab, and roll-up"; Proc. of International Conf. on Data Engineering; IEEE Press. 1996.

[7] Jiawei Han, Micheline Kamber; "Data Mining: Concepts and Techniques" 2nd Ed., Morgan Kaufmann Series in Data Management Systems; Elsevier Science; ISBN 13: 978-1-55860-901-3, 2006.

[8] Xiaolei Li , Jiawei Han , Hector Gonzalez; "High-Dimensional OLAP: A Minimal Cubing Approach"; Proceedings of 2004 International Conference on Very Large Data Bases (VLDB'04).

[9] Dimitris Papadias, Panos Kalnis, Jun Zhang, and Yufei Tao; "Efficient OLAP Operations in Spatial Data Warehouses"; Proc. of SSTD 2001, Springer-Verlag LNCS 2121, pp. 443-459, 2001.

[10] Min Wang, Bala Iyer; "Efficient Roll-Up and Drill-Down Analysis in Relational Databases"; Proc. of 1997 SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery.

[11] Jef Wijsen, Raymond T. Ng, Day Price; "Discovering Roll-Up Dependencies"; Proc. of ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (1998).

[12] S.Bhattacharya, A.Cortesi; "A distortion-free watermark framework for relational databases"; Proc. 4th Int. Conference on Software and Data Technology (ICSOFT'09) Sofia, Bulgaria (2009).

[13] A.Cortesi; "Widening Operators for Abstract Interpretation"; Proc. 6th IEEE Int. Conf. on Software Engineering and Formal Methods, pp. 31-40, Cape Town, SA (2008).

[14] A.Cortesi, F.Logozzo; "Abstract Interpretation Based Verification of Non-Functional Requirements", Proc. COORDINATION 2005, Namur, LNCS 3454, pp 49-62 (2005).