An. Şt. Univ. Ovidius Constanţa Vol. 17(3), 2009, 197–208

A MODEL OF REGULAR SPARSITY MAP REPRESENTATION

Ina Naydenova, Zlatinka Covacheva and Kalinka Kaloyanova

Abstract

Sparse data causes the data explosion problem in precomputation process and decreases the performance of OLAP. The regular sparsity map is an object that saves information about specific empty domains of the OLAP hyper-cubes and enables business analysts to define rules and place data constraints over the multidimensional cube. The preserved information can be used for several purposes — data validation, storage consideration, user-interface improvements. In this paper we present an approach for a regular sparsity map representation. It allows implementation of set operations between a regular sparsity map and multidimensional domains and requires less storage and computational resources.

1 Introduction

Prior to the start of the Information Age in the late 20th century, business had to collect data from non-automated sources. Business then lacked the computing resources necessary to properly analyze the data, and as a result, companies often made business decisions primarily on the basis of intuition. The modern technologies of computers and networks have made data collection and organization much easier. However, the captured data needs to be converted into information and knowledge to become useful. This puts a new

Key Words: dimension, cube, OLAP, sparsity, map, set operations.
Mathematics Subject Classification: 68P05, 68P15, 68P20.
Received: April 2009
Accepted: October 2009
This reserch was supported by the project SUGrid (National Scientific Research Fund Reg. No. VU-MI-110/2005).

class of problems related to the data integrity and correctness, results reliability, performance of the analytical query etc. According to the independent OLAP Survey conducted by Nigel Pendse [8, 9] the top three problems in the field of Business Intelligence (BI) over the past years are slow query performance, poor quality data, and company policy.

In [4] we introduce a new object ("regular sparsity map") to the multidimensional data model, which model is widely used in BI applications. The map saves information about specific empty domains of multidimensional cubes and enables business analysts to define rules and place data constraints over the multidimensional cube. We also investigate the applicability of the defined object in three main directions — data quality, performance issues and human interface facilities.

With the purpose of map implementing and furthermore utilization, we develop a method of map preservation that allows implementation of set operations between a map and multidimensional domains and requires less storage and computational resources than the point-by-point based approach.

2 The regular sparsity map

To explain what a regular sparsity map is, first of all we will introduce some definitions.

2.1 Multidimensional data model definition

A popular conceptual model that influences the front-end tools, database architecture and design, and the query engines for OLAP is the multidimensional view of data in the data warehouse. In a multidimensional model, there is a set of numeric measures that are the objects of analysis. Examples of such measures are sales, revenue, profit, number of clients etc. Each of the numeric measures depends on a set of dimensions, which provide the context for the measure. For example, the dimensions associated with a sale amount can be the store, product, and the date when the sale was made. The dimensions together are assumed to uniquely determine the measure. Thus, the multidimensional data views a measure as a value in the multidimensional space of dimensions. Often, dimensions are hierarchical; time of sale may be organized as a day-month-quarter-year hierarchy, product as a product-category-industry hierarchy [2].

Regardless of common basis of the model there are a lot of formal definitions. One can find a multidimensional model specified as a set of dimensions, measures and data cubes with corresponding definitions of objects schemes, domains and applicable operators in [6] or using a multidimensional space

(Cartesian product of relational attribute projections) structured by the generalization order between tuples (cube lattice as a convex space is considered) in [1]. Since even tabular data, such as relations, can be thought of as multidimensional, many definitions incorporate this point of view: multidimensional database algebra, specified over table schemes, is defined in [3] and [7].

To define a regular sparsity map object we assume a simplified conceptual cube model that treats data in the form of $n-$dimensional cubes. The hierarchy between the various levels of aggregation in dimensions is of no interest to us.

- *Dimension* is a non-empty finite set;
- *Multidimensional space* S over dimensions $D_1, D_2, \ldots, D_n$ $(n \geq 1)$ is the Cartesian product $S = D_1 \times D_2 \times \cdots \times D_n$. It contains $n-$tuples $(x_1, x_2, \ldots, x_n)$, where $x_1 \in D_1, x_2 \in D_2, \ldots, x_n \in D_n$.
- *Rectangular domain* in multidimensional space S is a subset $M \subseteq S$, $M = A_1 \times A_2 \times \cdots \times A_n$, where $A_1 \subseteq D_1, A_2 \subseteq D_2, \ldots, A_n \subseteq D_n$;
- $\emptyset$ is a special value named "empty value";
- *Fact* F is a set, where $\emptyset \in F$;
- *Cube* is a function $C : S \to F$, where S is a multidimensional space, F is a fact;
- *Cell* in the cube $C : S \to F$ is a pair $c = (t, f)$, where $t \in S$, $C(t) = f$. The cell is *empty* if $f = \emptyset$ and *non-empty* otherwise;
- *Set of empty cells* in the cube $C : S \to F$ is the set $E(C) = \{t \in S \mid C(t) = \emptyset\}$, $E(C) \subseteq S$.

We might be building a cube for a supermarket, where one dimension (D_1) is geography (individual stores), another one (D_2) is time (months), another one (D_3) is customers and the last one is products (D_4). Measures in the observed fact (F) are the quantity sold and the revenue. If in "April 2008" customer "Andrew" bought "2 bars" of "chocolate" in store "Boyana" for "3 euro", then we have a non-empty cell (("Boyana", "April 2008", "Andrew", "chocolate"), ("2 bars", "3 euro")) in the cube. If in the same month he did not buy any "ice-cream" from this store, we have an empty cell (("Boyana", "April 2008", "Andrew", "ice-cream"), $\emptyset$).

2.2 Sparsity definition

Many cells in an OLAP cube are not populated with data. The more empty cells found in a cube, the sparser the cube data is. This is measured by the density coefficient.

Density coefficient of cube $C : S \to F$ is a ratio $\omega_C = \dfrac{|S| - |E(C)|}{|S|}$.

If we have 60 stores, 500 products, 70,000 customers and 12 months in a year, our cube has a potential $60 \times 500 \times 70,000 \times 12 = 25,200,000,000$ cells, but we might only have 360,000,000 non-empty cells in our measure (40 000 customers shopping 12 months a year, buying average on 25 products at 30 stores) making our cube $(360,000,000/25,200,000,000) \times 100 = 1.4\%$ dense.

2.3 Regular sparsity map definition

A closer scrutiny reveals that there could be some difference between empty cells in terms of the causes provoking the cell's emptiness. We divide the cube's sparsity into two types: *random* and *regular* sparsity. If a cell is empty because of the semantics of the modelled business area (the semantics enforces lack of value), then we witness "regular sparsity". If the cell is empty, but it is possible it had a value, "random sparsity" is what we have. Missing data in random sparsity usually expresses zero values, whereas the regular sparsity expresses inapplicable values. In [5] we point out several forms of regular sparsity (irrelevant dimensions, segmentation of dimensions, dimension changes over time).

To formally distinguish regular from random sparsity, we introduce the following definition:

Regular sparsity map of the cube $C : S \to F$ is the set $R_C \subseteq E(C) \subseteq S$.

A regular sparsity map (or shortly *map*) R_C determines the cells which are empty because of regular sparsity (business rules, formal requirements, natural dependences, etc.).

The set difference $E(C) \setminus R_C$ determines the cells which are empty because of random sparsity.

In the previous example we can observe random and regular sparsity. The store "Boyana" offers 3,000 products. "Andrew" has bought only 50 of them. For the rest 2,950 products we have empty cells because of the random sparsity (in fact their value is zero). For the 7,000 unavailable products we have empty cells because of the regular sparsity. If $Z \subset D_4$ is the list of available products in "Boyana", then $R_C = \{(d_1, d_2, d_3, d_4) \in S \mid d_1 = \text{“Boyana”}, d_4 \notin Z\}$.

3 The problem

In [4] we investigate the applicability of the regular sparsity map for development of business constraints enforcement module, non-additive data compression, composite dimension selection, automatic selection of relevant dimension elements etc. The applications development is related to the question of how the map could be represented. The utilization of the regular sparsity map requires a proper model that is convenient and easy for use by:

1) the people that will construct a map;

2) the software that will use the map in different applications.

From the humans' point of view the regular sparsity map is a set of business rules.

Our analysis shows that almost all applications of the regular sparsity map require an algorithm that performs set operations between a regular sparsity map and a multidimensional domain. So the software for an extraction of the regular sparsity map information has to be able to answer questions of the following type:

We have a regular sparsity map $R_C \subseteq E(C) \subseteq S$ of the cube $C : S \to F$, $S = D_1 \times D_2 \times \cdots \times D_n$.

An input rectangular domain I: $I \subseteq S$, $I = A_1 \times A_2 \times \cdots \times A_n$, $A_1 \subseteq D_1$, $\ldots$, $A_n \subseteq D_n$.

We are interested in which cells of the domain I $\{c = (t, f) | t \in I, C(t) = f\}$ are empty because of the regular sparsity: $I_E = I \cap R_C$.

We are also interested in which cells of the domain I $\{c = (t, f) | t \in I, C(t) = f\}$ are potentially not empty: $I_{NE} = I \setminus R_C$.

One solution is the regular sparsity model to store the set of tuples covered by the map (point-by-point approach). Then we can apply union or minus operation over tuples covered by the map and the tuples covered by the input domain I. Unfortunately, in real-life cases the number of empty cells in a map often exceeds 10^{13}. The performance of set operations depends on the cardinality of its arguments, so this solution is unsatisfactory: the case of 1.4% dense cube (the example above) requires 24,840,000,000 empty cells coordinates to be processed.

So our task is to find another representation of a regular sparsity map and a more efficient way to perform set operations with rectangular domains in a multidimensional space.

4 The map representation approach

To perform more efficiently set operations with rectangular domains, we propose an algorithm that works with dimension subsets instead of dimension elements. In our solution the input rectangular domain I is split into a set of rectangular sub-domains, each of which is entirely inside or outside the map. To facilitate the process of the input domain splitting, we represent the regular sparsity map R as a union of nonintersecting rectangular domains.

4.1 Business constraints representation

A convenient approach is the map to be constructed on a set of rules. Each rule describes a set of cells that should be empty. The union of these sets forms the regular sparsity map.

The simplest type of a rule that can be described by the users is in the following form:

(1) IF $D_1 \in \{d_{11}, d_{12}, \ldots\}$ AND $D_2 \in \{d_{21}, d_{22}, \ldots\}$ AND ... THEN the cell is EMPTY;

In such way the rule defines exactly one rectangular domain:

$$R_1 = A_1 \times A_2 \times \cdots \times A_n, \quad A_1 = \{d_{11}, d_{12}, \ldots\} \subseteq D_1, \ \ldots, \ A_n \subseteq D_n.$$

This form is flexible enough to describe all regular sparsity business constraints that we meet in our practice. But usually it is not so convenient for business analysts who are expected to define map rules.

Here we show how the model of business constraints description could be extended so that it would be more convenient and, at the same time, the rules could be transformed into the form (1).

Example of rule definition with different Boolean operations

Let us imagine that we have the following business constraints over the cube $C : S \to F$, $S = D_1 \times D_2 \times D_3$, where D_1 is the dimension Stores, D_2 is Products and D_3 Time, F is the quantity sold:

Store “Vitosha” does not offer cigarettes. Before January 2008, store “Vidim” did not offer peanuts. Store “Elemag” does not offer peanuts at all.

The business analysts define the existing constraints over the cube C with the help of the following rules:

Rule 1: IF Store = “Vitosha” THEN Product ≠ “cigarettes”.

Rule 2: IF (Time < “January 2008” AND Store = “Vidim”) OR (Store = “Elemag”) THEN Product ≠ “peanuts”.

The above rules describe dependences in the form of Boolean implication (IF A THEN B). For regular sparsity domains, the value of the implication is false. So, we are looking for an expression in the form of “A AND NOT B” to describe empty cells of the cube:

Rule 1) IF Store = “Vitosha” AND Product = ”cigarettes” THEN the cell is empty;

Rule 2) This rule can be transformed into two rules in the form (1). To separate them we are looking for an expression in a disjunctive normal form:

IF (Time < “January 2008” AND Store = “Vidim” AND Product = “peanuts”) OR (Store = “Elemag” AND Product = “peanuts”) THEN the cell is empty.

Rule 2.1: IF Time < “January 2008” AND Store = “Vidim” AND Product = ”peanuts” THEN the cube is empty;

Rule 2.2: IF Store = “Elemag” and Product = “peanuts” THEN the cell is empty.

Using hierarchies and other dimension attributes in rule description

Since the number of elements in one dimension may exceed hundreds, for the analyst the possibility for an easy choice of sets of dimension elements is important — by means of filters over different attributes, hierarchical relations etc. These are elements of the multidimensional model which we have not considered above, but here is an example which illustrates such possibilities:

In Alaska ice-cream is not sold. In the Region Alaska there are stores X,Y,Z. This relation is given with an additional stucture — a hierarchy or dimension over the dimension Stores. Analogously, ice-cream is a category of dozens of products — $p_1, \dots, p_n$.

Rule: IF Store.Region = “Alaska” THEN Product.Category ≠ “ice-cream”.

This rule will be transformed to: IF Store ∈ (X,Y,Z) THEN Product ∉ $(p_1, \dots, p_n)$.

4.2 The map construction

We are going to describe a process of regular sparsity map construction. At the end of this process the regular sparsity map is presented as a union of non-intersecting rectangular domains (segmentation approach). For this purpose, we will introduce what is a segmentation of the multidimensional space, how to extend a segmentation of the space by a rectangular domain and how to split a rectangular domain over a space segmentation.

Let S be a multidimensional space over dimensions $D_1, D_2, \dots, D_n$: $S = D_1 \times D_2 \times \cdots \times D_n$.

G is a segmentation of the multidimensional space S when:

$$G(D_i) = \left\{D_i^1, D_i^2, \dots, D_i^{p_i}\right\} : \quad D_i = \bigcup_{j=1}^{p_i} D_i^j \quad \text{and } D_i^j \cap D_i^k = \{\ \} \ \text{ for } \ k \neq j.$$

Extend a segmentation of the space by a rectangular domain:

Let M be a rectangular domain in the space S and let G be a segmentation of S:

$$M = A_1 \times A_2 \times \cdots \times A_n, \quad \text{where} \quad A_1 \subseteq D_1, A_2 \subseteq D_2, \ldots, A_n \subseteq D_n.$$

The segmentation G' is defined as the extension of G by M as follows:

$$\forall i = 1, \ldots, n: \qquad G'(D_i) = \left\{ D_i^j \mid D_i^j \in G(D_i),\ (D_i^j \cap A_i = \emptyset \vee D_i^j \subseteq A_i) \right\} \cup \left\{ D_i^j \cap A_i,\ D_i^j \setminus A_i \mid D_i^j \in G(D_i),\ D_i^j \cap A_i \neq \emptyset,\ D_i^j \setminus A_i \neq \emptyset \right\}.$$

In such way for every $D_i^j \in G(D_i)$: $D_i^j \cap A_i \neq \emptyset$ and $D_i^j \setminus A_i \neq \emptyset$ the splitting of D_i^j is performed by $D' = D_i^j \cap A_i$ and $D'' = D_i^j \setminus A_i$. The new segments D' and D'' replace the old segment D_i^j in the set of segments $G(D_i)$ of the dimension D_i.

If the result $G' \equiv G$, we say that G has been already extended by M.

Splitting of a rectangular domain over a space segmentation:

Let M be a rectangular domain in the space S and G be a segmentation of S and G has been already extended by M.

$$\text{split}(M, G) = \bigtimes_{i=1}^{n} \left\{ D_i^j \in G(D_i) \mid D_i^j \cap A_i \neq \emptyset \right\}.$$

Let R_C be a regular sparsity map of the cube $C: S \to F$, $S = D_1 \times D_2 \times \cdots \times D_n$ represented as a union of non-intersecting rectangular domains:

$$L = \{P_1, \ldots, P_t\}, \qquad \text{where} \quad R_C = \bigcup_{j=1}^{t} P_j \quad \text{and} \quad P_i \cap P_j = \{\ \},\ i \neq j.$$

At the beginning of the construction process the regular sparsity map is empty. The first rule adds a rectangular domain to the map. Then every time when a new rule is added to the map, the existing rectangular domains are split to non-intersecting rectangular domains.

(1) The map is empty: $L = \{\ \},\ R = \{\ \}$.

In this step we define the following segmentation of the multidimensional space S: $G(D_i) = \{D_i\}$.

(2) A new business rule is added.

The rule describes exactly one rectangular domain $B = A_1 \times A_2 \times \cdots \times A_n$.

(a) We extend the segmentation G by the rectangular domain B to G'.

(b) For every $P_i \in L$ we split the rectangular domain P_i over the segmentation G': $\quad \forall P_i \in L: \quad \mathrm{split}(P_i, G')$.

This procedure can increase the number of nonintersecting rectangular domains: $L = \{P_1, \dots, P_{t'}\}$, $t' \geq t$.

(c) We also split a rectangular domain B over G': $\mathrm{split}(B, G')$.

This action splits B to a union of nonintersecting rectangular domains according to the segmentation of dimensions in R_C.

$$\mathrm{split}(B, G') = \{B_1, \dots, B_m\} = \bigtimes_{i=1}^{n} \left\{ D_i^j \in G'(D_i) \middle| \, D_i^j \cap A_i \neq \emptyset \right\},$$

$$B = \bigcup_{i=1}^{m} B_i, \quad \text{where } B_i \cap B_j = \{\ \} \text{ for } i \neq j,$$

$$\forall i, j: \quad P_i = B_j \text{ or } P_i \cap B_j = \{\ \}.$$

(d) The union operation that should be applied between the map R_C and the rectangular domain B, is applied between the sets of pieces L and $\mathrm{split}(B, G')$. The assignment $L := L \cup \mathrm{split}(B, G')$ is functionally equivalent to the assignment $R_C := R_C \cup B$.

(3) Repeat Step 2 while there are new rules.

4.3 Set operations between a regular sparsity map and a rectangular domain

Let us have a regular sparsity map $R_C \subseteq E(C) \subseteq S$ of the cube $C: S \to F$, $S = D_1 \times D_2 \times \cdots \times D_n$ and an input rectangular domain $I: I \subseteq S$, $I = A_1 \times A_2 \times \cdots \times A_n$, $A_1 \subseteq D_1, \dots, A_n \subseteq D_n$. We are interested in the result of a union, intersection or minus operation between the input rectangular domain and the regular sparsity map: $I \circ R_C | \circ \in \{\cup, \cap, \setminus\}$.

To perform this operation we do the same steps as we do to add a rectangular domain to the map. The only difference is that the result of the operation is not preserved in the map definition.

Let G be a segmentation of a multidimensional space S already extended by the map rules.

According to our map representation $R_C = \bigcup_{j=1}^{t} P_j$, where $P_j = H_1^j \times H_2^j \times \cdots \times H_n^j$, $H_i^j \in G(D_i)$ and $P_i \cap P_j = \{\ \}$, $i \neq j$, $L = \{P_1, \dots, P_t\}$.

Then $I \circ R_C = \mathrm{split}(I, G) \circ L$.

5 Comparison with point-by-point approach

The segmentation approach enables us to operate with rectangular domains instead of points in a multidimensional space. Of course, in the worst case the rectangular domains in the map are points. In such case because of the time necessary for a map construction and preservation of space segmentations, the segmentation approach is even worse than the point-by-point approach. But in the real-life systems this is an impossible situation. Typically cubes have between 5 and 16 dimensions [10] and the number of cube cells exceeds 4E+17, whereas the number of rules usually is more than 10, but not more than 100.

This is why we do not try to formally prove the advantages of the segmentation approach but just describe and expose them.

Storage consideration

Point-by-point approach: We will preserve a map as a list of tuples in a multidimensional space S.

If the map is $R_C = \{(d_1, \ldots, d_n),\ d_i \in D_i\}$, then the storage magnitude is of order $n \cdot \theta \cdot |R_C|$, where n is the number of dimensions, θ is an average storage size for one identifier (of a dimension element or a dimension segment).

Segmentation approach: We will preserve a map as tuples of segment identifiers and the space segmentation as pairs (dimension element, segment identifier) for each dimension.

The storage magnitude is of order $n \cdot \theta \cdot |L| + 2\theta \cdot \sum\limits_{i=1}^{n} |D_i|$.

We do some computations in a real life information system. Here are the results for a typical OLAP cube from that system with 9 dimensions (see Table 1) and 6 business rules:

$n = 9, \quad |R_C| = 218,854,686,744, \quad |L| = 248, \quad \sum_{i=1}^{n} |D_i| = 650,$

$n \cdot |R_C| = 1,969,692,180,696, \qquad n \cdot |L| + 2\sum_{i=1}^{n} |D_i| = 3,532.$

Table 1. Sample cube dimensions

Dimension name	**Cardinality**
Company	6
Product	296
Channel	10
Geography	246
Gender	3
Partytype	3
Age	12
Branch of business	62
Deposit Size	12

Set operation computation
When the number of arguments of operations with sets is too great, in order to perform the entire operation in the operation memory one has to use algorithms which read and write information in the much slower external memory. On the one hand, this makes them less efficient by order than the operation which are performed entirely in the operation memory. On the other hand, the increase in the number of arguments leads to an increase in the number of necessary input-output operations, which is why these algorithms are strongly sensitive to the number of arguments.

With the point-by-point approach due to the large number of arguments — of order $|R_C|$ — it is very easy to exceed this critical threshold.

With the segmentation approach the number of arguments is of order $|L|$.

6 Conclusions

Sparsity refers to a natural phenomenon evident in all multidimensional data to some degree: not all of the cells in the logical cube will ever contain data. It is very common for a relatively small percentage of the possible combinations to actually store data. To provide a practical baseline expectation for sparsity, the authors in [10] examine data sparsity in a variety of industry-specific models including insurance claim analysis, telecom call analysis, revenue-per-user-analysis and a medical device company's sales analysis. The research shows that data density in all cases is significantly less than 1 percent - i.e., extremely sparse.

Cube sparsity causes the data explosion problem and has many impacts on the storage size, calculations, loading and query performance. The current methods for overcoming of data explosion work mainly on physical level and do not take into account the nature of sparsity. In [5] we discuss an idea of the regular sparsity map in order to avoid some difficulties related to the user requirements and the high-dimensionality of the requested reports. Then in [4] we formally define the "regular sparsity map" object and investigate the possible object application. Here, we present a model of map representation that allows practical implementation of set operations between a map object and rectangular domains over a multidimensional space. By virtue of this model we are in the process of regular sparsity map applications development.

References

[1] A. Casali, R. Cicchetti, L. Lakhal, *Cube lattices: a framework for multidimensional data mining*, in: Proceedings of the 3rd SIAM International Conference on Data Mining, San Francisco, California, USA, May 2003, pp. 304–308.

[2] S. Chaudhuri, U. Dayal, *An overview of data warehousing and OLAP technology*, SIGMOD Record, **26** (1997), 65–74.

[3] M. Gyssens, L. V. S. Lakshmanan, *A foundation for multi-dimensional databases*, in: Proceedings of 23rd International Conference on Very Large Data Bases, Athens, Greece, August 1997, pp. 106–115.

[4] I. Naydenova, *Regular sparsity map*, submitted to Information Technologies and Control magazine, Bulgaria, November 2008.

[5] I. Naydenova, K. Kaloyanova, *Some extensions to the multidimensional data mode*, in: Proceedings of the IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing, Sofia, Bulgaria, October 2006, pp. 63–68.

[6] T. B. Nguyen, A. M. Tjoa, R. R. Wagner, *An object oriented multidimensional data model for OLAP*, in: Proceedings of the First International Conference on Web-Age Information Management, Shanghai, China, June 2000, pp. 69–82.

[7] D. Pedersen, K. Riis, T. B. Pedersen, *A powerful and SQL-compatible data model and query language for OLAP*, in: Proceedings of the 13th Australasian Database Conference, Melbourne, Australia, January 2002, pp. 121–130.

[8] N. Pendse, *The OLAP Survey 6*, http://www.survey.com/olap/, 2006, 21 pp.

[9] N. Pendse, *The problems with OLAP*, DM Review Magazine, March 2007.

[10] J. Potgieter, *OLAP data scalability*, DM Review Magazine, October 2003. Available at: http://www.dmreview.com/dmdirect/20031031/7636-1.html

Ina Naydenova
Institute for Parallel Processing
Bulgarian Academy of Science, Sofia, Bulgaria
e-mail: naydenova@gmail.com

Zlatinka Covacheva
Higher College of Technology, Muscat, Oman &
Higher College of Telecommunications and Post, Sofia, Bulgaria
e-mail: zkovacheva@hotmail.com

Kalinka Kaloyanova
"St. Kliment Ohridski" University of Sofia
Faculty of Mathematics and Informatics, Sofia, Bulgaria
e-mail: kkaloyanova@fmi.uni-sofia.bg