MCIS 2009 Proceedings

Mediterranean Conference on Information Systems (MCIS)

2009

Data Errors And Relevant Dimension Values Detection With A Regular Sparsity Map

Ina Naydenova
Institute for Parallel Processing, Bulgarian Academy of Science, naydenova@gmail.com

Kalinka Kaloyanova
University of Sofia, Faculty of Mathematics and Informatics, kkaloyanova@fmi.uni-sofia.bg

Georgi Georgiev
University of Sofia, Faculty of Mathematics and Informatics, ggeorgiev@technologica.com

Pegruhi Melkonyan
University of Sofia, Faculty of Mathematics and Informatics, pmelkonyan@technologica.com

DATA ERRORS AND RELEVANT DIMENSION VALUES DETECTION WITH A REGULAR SPARSITY MAP

Naydenova, Ina, Institute for Parallel Processing, Bulgarian Academy of Science, Acad. G. Bonchev Str., Bl. 25-A, Sofia, Bulgaria, naydenova@gmail.com
Kaloyanova, Kalinka, University of Sofia, Faculty of Mathematics and Informatics, 5 J. Bourchier Str., Sofia, Bulgaria, kkaloyanova@fmi.uni-sofia.bg
Georgiev, Georgi, University of Sofia, Faculty of Mathematics and Informatics, 5 J. Bourchier Str., Sofia, Bulgaria, ggeorgiev@technologica.com
Melkonyan, Pegruhi, University of Sofia, Faculty of Mathematics and Informatics, 5 J. Bourchier Str., Sofia, Bulgaria, pmelkonyan@technologica.com

Abstract

Data warehouses require and provide extensive support for data cleaning. They load and continuously refresh huge amounts of data from a variety of sources so the probability that some of the sources contain “dirty data” is high. In this paper we present our regular sparsity map editor which can be used for the purpose of detection of specific data errors in the data warehouse systems. We also discuss how it can be used for a selection of relevant dimension elements.

Keywords: *Data Warehouse, Regular Sparsity Map, Cube, Data Cleaning*

1 INTRODUCTION

Information integration is one of the most important and problematic aspects of a Data Warehouse (Calvanese, De Giacomo, Lenzerini, Nardi and Rosati 2001). Anomalies and impurities in data cause irritations and avert its effective utilization, disabling high performance processing and confirmation of the results and conclusions gained by data interpretation and analysis (Muller and Freytag 2003). As a result, business intelligence systems experience low confidence and acceptance by users and consumers of downstream reports (Linsley and Dutta 2008).

In (Naydenova 2008) we introduce a new classification of multidimensional cube sparsity phenomena, define an object named “regular sparsity map” (RSM) and investigate the RSM applicability. The RSM saves information about empty domains of multidimensional cubes and provides analysts with the ability to define business rules and place data constraints over the multidimensional model. The map can be used at many stages of the business intelligence system life cycle (storage and performance consideration, user-interface improvements), but its primary function is to support the process of discovering inaccurate and inconsistent information.

We developed an editor for RSM creation and implement an algorithm that performs set operations between RSM and arbitrary multidimensional domains in the map space. We are going to briefly describe our implementation approach and discuss the data error and relevant dimension elements detection process. By virtue of RSM editor we are in the process of regular sparsity map applications development.

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. 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 measure can be the store, product, and the date when the sale was made. Often, dimensions are hierarchical; time of sale may be organized as a day-month-quarter-year hierarchy, product as a product-category-industry hierarchy (Chaudhuri and Dayal 1997). 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 hierarchies between the various levels of aggregation in dimensions are of no interest to us.

- *Dimension* is a non-empty finite set*;*
- *Multidimensional space S* over dimensions $D_{1,} D_2, ..., D_n$ (n>=1) is the Cartesian product $S = D_1 \times D_2 \times ... \times D_n$. It contains n-tuples (x1, x2, xn) where $x_1 \in D_1, x_2 \in D_2, ...x_n \in D_n$
- *Rectangular domain* in multidimensional space S is a subset $M \subseteq S$, $M = A_1 \times A_2 \times ... \times A_n$, where $A_1 \subseteq D_1$, $A_2 \subseteq D_2$, …, $A_n \subseteq D_n$;
- Ø is a special value named “empty value”;
- *Fact F* is a set, where Ø $\in$ F;
- *Cube* is a function C: S → F, where S is a multidimensional space, F is a fact;
- *Cell* in the cube C: S → F is a pair c = (t, f), where t $\in$ S, C(t) = f. The cell is empty if f = Ø and non-empty otherwise;
- *Set of empty cells in the cube C:* S → F is the set E(C) = {t $\in$ S | C(t) = Ø}, E(C) $\subseteq$ S.

We might be building a cube for a supermarket, where one dimension (D1) is geography (individual stores), another one (D2) is time (months), another one (D3) is customers and the last one is products (D4). 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”), Ø).

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 → F is a ratio $\omega_C = \frac{|S| - |E(C)|}{|S|}$

If we have 60 stores, 500 products, 70 000 customers and 12 months in a year, our cube has a potential 60×500×70000×12 = 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) ×100 = 1.42% dense.

Cube sparsity has many impacts on the storage size, loading and query performance in multidimensional databases. More information about the sparse data consequences and the relation between sparsity and exploding databases phenomenon can be found in (Pendse 2005).

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 one 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. In (Naydenova and Kaloyanova 2006) 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 (RSM) of the cube C: S → F is the set RC $\subseteq$ E(C) $\subseteq$ S.

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

The set difference E(C) \ RC 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 3000 products. "Andrew" has bought only 50 of them. For the remaining 2950 products we have empty cells because of the random sparsity (in fact their value is zero). For the 7000 unavailable products we have empty cells because of the regular sparsity. If Z $\subset$ D4 is the list of available products in "Boyana" then RC = {(d1, d2, d3, d4) $\in$ S | d1 = "Boyana", d4 $\notin$ Z}

3 RSM AND DATA CLEANING

In our experience the correctness of data is always a problem. Actually this is the problem which more often is an obstacle for the practical application of BI system. Usually, only after the data is loaded and the first results are obtained it is clear that there are defects in the data. The discovering and elimination of these defects is quite hard procedure because the input data is related with much dependence and passes through a number of transformations until it is presented to the multidimensional model. A second time data loading is often necessary and an execution of all the steps over again. The solution of this problem is the data to be verified on a possibly earlier stage of its processing.

The regular sparsity map describes constraints over the data in the term of multidimensional model, which is close to the concepts of the business analysts. At the same time it enables easy implementation of automatic data tests before receiving the results by the end users.

The development of a module for business constraints and dependence enforcement (CDE module) is an obvious application of the map information. The module can have the following functionalities:

- Validation of the regular sparsity map definition over a trusted data cube (cube without dirty data); this functionality is to be used immediately after the process of map construction, in order to check the correctness of the specified constraints;
- Errors detection and correction; after the validation of map definition, the constraints can be enforced over unverified data;

In our RSM Editor we implement base error detection functionalities of a CDE module. After the definition of a regular sparsity map the users can choose a trusted data cube and check the definition correctness. Then they can choose another cube and see the conflicts – cube cells that have to be empty according to the RSM definition, but they are loaded with non-zero values.

Example

Let us have a cube C varying over 9 dimensions: Company, Service, Sales Channel, Regions, Gender, Party type, Age, Branch of business, Clients. We have defined a RSM over C that specify the following rules:

1. If a company is MetLife Insurance then sales channel are Agents, Brokers and Direct sales.
2. If company is MetLife Insurance then relevant regions are Plovdiv, Sofia and Burgas.
3. If party type is organization then gender is unknown.
4. If party type is private person then branch of business is unknown.

Imagine that data of source OraLife migrates in new system. When the snapshot for April 2009 was loaded in the data warehouse system, the gender of all corporate clients loaded from an OraLife data source is set to Female (because of the migration or interface errors). The CDE module reports that the rule 3 is violated. The CDE module also reports that there is sales channel that violates rule 1. The further investigation shows that a new channel was created, but it was not entered in corresponding data warehouse dimension table.

According to the classification made by Rahm and Hong Hai Do (2000), the major data quality problems can be divided to schema- and instance-related problems. The scheme level problems are related with poor schema design, scheme translation and integration, while Instance-level problems refer to errors and inconsistencies in the actual data contents which are not visible at the scheme level. We believe that the RSM can support the process of Instance Level inconsistent data detection. The Rahm et al. (2000) discuss that in order to detect which kinds of errors and inconsistencies are to be removed a detailed data analysis is required. In addition to a manual inspection of the data or data samples, analysis programs should be used to gain metadata about the data properties and detect data quality problems. The constraints defined with a regular sparsity map are an additional source of metadata. Also the RSM based validation has the advantage that every modification in the RSM constraints immediately will be taken into account during the data cleaning process.

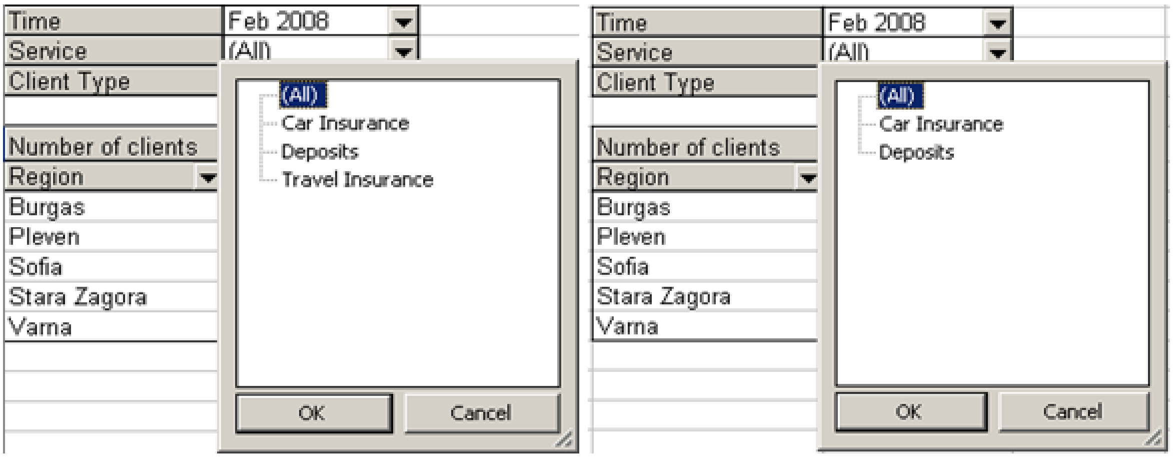
4 RSM AND RELEVANT DIMENSION ELEMENTS SELECTION

The regular sparsity map enables the feature for automatic restriction of user choice of dimension filters or parameters. When a business analyst selects some dimension values, the values of the other dimensions can be restricted to the set of meaningful tuples.

Let us imagine that at a certain moment in time t_k all the supermarkets in our company stop offering product p_j. We have a new rule: “When time > t_k then services ≠ p_j”. In the process of typical business intelligence slice and dice operation, the end user of the system can fix “Time” dimension to t_{k+1}. In fact, he or she is interested in a specific part of the entire cube. The application refers to the RSM API, preprocesses the user request and returns a reduced sub-cube without the p_j layer.

Example

If we have a business rule “After January 2008, we don’t offer travel insurance” and the user fixes the “Time” dimension to Feb 2008 the available choices of “Service” will be reduced to the available services over the month. The figure 1 illustrates the expected effect over the typical user interface of the BI tools. It demonstrates the available service choices without regular map filtering on the left side and reduced choices after the filtering on the right side.



Time: Feb 2008
Service: (All)
Client Type

Number of clients

Region
Burgas
Pleven
Sofia
Stara Zagora
Varna

Figure 25. Automatic selection of relevant dimension elements.

5 RSM EDITOR IMPLEMENTATION APROACH

The RSM 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 from:

- the people that will construct a map;
- 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. So in our editor the users define a map as a set of rules. Each rule describes a set of cells that should be empty.

The software that will use the map requires 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 the questions of the following type:

Let R_c is a regular sparsity map $R_c \subseteq E(C) \subseteq S$ of the cube $C : S \to F$,

$S = D_1 \times D_2 \times \ldots \times D_n$ and Q is an input rectangle domain (question) $Q : Q \subseteq S$,

$Q = A_1 \times A_2 \times \ldots \times A_n$, $A_1 \subseteq D_1$, …, $A_n \subseteq D_n$,

We are interested in which cells of the domain $Q\{c = (t, f) \mid t \in Q, C(t) = f\}$ are empty because of the regular sparsity: $Q_E = Q \cap R_c$

We are also interested in which cells of the domain $Q\{c = (t, f) \mid t \in Q, C(t) = f\}$ are potentially not empty: $Q_{NE} = Q \setminus R_c$

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 1013. The performance of set operations depends on the cardinality of it arguments so this solution is unsatisfactory: in the case of 1.42% dense cube (the example above) it requires 24840000000 empty cells coordinates to be processed.

So our task is to find other representation of a regular sparsity map and a more efficient way to perform set operations with rectangular domains in a multidimensional space. You can see our idea of RSM representation and set operation algorithm in (Naydenova, Kovacheva and Kaloyanova 2009) but in our RSM editor implementation we do some modifications. We present a map as a union of empty rectangular domains, but they are not necessarily non-intersecting as is pointed in (Naydenova et al. 2009). In our solution the input rectangular domain Q is spit to a set of rectangular sub-domains, each of which is entirely inside or outside the map.

This technique is used to detect non relevant dimension elements:

An input question Q is formed on the base of a user dimension selection. According to figure 1 the input question has the following form:

1. Time = February 2008, Region in (Burgas, Pleven, Sofia, Stara Zagora,Varna), Service in (Insurance, Deposits, Travel Insurance), Client Type = organization.
2. The dimension D_t, whose non-relevant elements are of interest to us, is specified as a target. According to figure 1 this is a Service dimension.
3. We apply the algorithms that split a question Q to a set of empty Q_E and potentially nonempty Q_{NE} rectangular domains.
4. The projection of the values of target dimension D_t in relation to all empty Q_E domains gives us the list of non-relevant dimension values.

   In the simplified example from figure 1 we receive only one empty domain:

   Time = February 2008, Region in (Burgas, Pleven, Sofia, Stara Zagora,Varna), Service= (Travel Insurance), Client Type = organization.

   So the “Travel Insurance” is a non-relevant value and we can remove it from the list of available services.

6 CONCLUSION AND FURTHER WORK

The sparsity of OLAP cubes is a phenomenon in multidimensional data that every designer and database administrator must consider. Sparse data causes the data explosion problem in the precomputation process and decreases the performance of OLAP. The current methods for overcoming of data explosion work mainly on physical level and don’t take account of the nature of sparsity. We introduce a regular sparsity map in attempt to look at sparsity from another angle – is there some useful information that sparsity can give us? With the help of an RSM editor and a map representation model we are going to implement others regular sparsity map applications. Also the CDE module can be improved with data correction functions, automatic generation of draft regular sparsity map definition by means of association rules data mining techniques, automatic generation of database constraints over the source data.

Acknowledgements: This paper is supported by the Sofia University SRF under Contract N 7/2009.

References

Calvanese, D., De Giacomo, G., Lenzerini M., Nardi, D.,Rosati, R. (2001).Data integration in data warehousing. International Journal of Cooperative Information Systems, 10(3), 237-271.

Chaudhuri S., Dayal U. (1997). An Overview of Data Warehousing and OLAP Technology. SIGMOD Record 26(1): 65-74.

Linsley, S., and Dutta, A. (2008). The Next Frontier for Data Warehouse Managers, DM Review magazine. Retrieved from http://www.information-management.com/infodirect/2008_58/10000674-1.html.

Muller, H., Freytag, J. (2003). Problems, Methods, and Challenges in Comprehensive Data Cleansing. Technical Report HUB-IB-164, Humboldt University Berlin, Germany.

Naydenova, I., Kaloyanova, K. (2006). Some Extensions to the Multidimensional Data Model. Proceedings of the IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing, Bulgaria, pp. 63 - 68.

Naydenova, I. (2008). Regular Sparsity Map. Submitted to Information Technologies and Control magazine, Bulgaria.

Naydenova, I., Kovacheva, Z., Kaloyanova, K. (2009). A Model of Regular Sparsity Map Representation. The 5th International Conference 2009 - Dynamical Systems and Applications, 15-18 June 2009, Romania.

Pendse, N. (2005). Database explosion, Business Application Research Center, Retrieved from http://www.olapreport.com/DatabaseExplosion.htm.

Rahm, E., Hong Hai Do. (2000). Data Cleaning: Problems and Current Approaches, IEEE Data Engineering Bulletin, Vol 23 No.4, pp.3-13.