Toward Reducing Data Cubes

Sami Naouali
Military Academy of Fondok J'Did
Tunis, Tunisie
snaouali@gmail.com

ABSTRACT:*.This paper presents an approach toward reducing multidimensional data in a data warehouse and giving the analyst more facilities when querying data cubes. This reduction concerns the number of dimensions as well as the huge amount of facts in the data warehouse. To make it possible, this approach combines the potential of the Principal Component Analysis which is a purely statistical approach with the Rough Set Theory which is an Artificial Intelligence one. In fact, The Principal Component Analysis is used, here, to reduce dimensions in the data warehouse without loosing information. So it computes a subset of the most important and pertinent dimensions. This subset of data dimensions is used by the Rough Set Theory in order to delete superfluous and redundant facts. The interest of this approach is to help user to focus on the most important information of the data warehouse which is a very hard task when working on the whole multidimensional database.*

Keywords: Data Warehouse, Principal Component Analysis, Rough Set Theory, Data Cubes, Multidimensional Data

Received: 8 March 2012, Revised 23 April 2012, Accepted 29 April 2012

1. Introduction

Data warehouses are multidimensional databases where data are covering long periods of time (more than 10 years according to [1]). The multidimensional aspect of data warehouses is important since it allows user to analyse and understand his activity according to several axis such as customer, provider, localisation, time, etc. Also, storing data covering long periods of time (by regularly providing the data warehouse with new data) is essential in order to extract knowledge and to apply data mining techniques (extracting patterns, computing association rules, clustering, etc.). So, the size of a data warehouse is growing increasingly and analyzing such amount of data becomes extremely difficult.

So, numerous research works were proposed to reduce data in a data warehouse. All of these works focused on making reductions by aggregating data. So, [2] proposed to aggregate old data to a higher granularity level without deleting them from the multidimensional database. [3] proposed to delete data older than a pre-defined limit. In [4], authors proposed to automatically aggregate old data to a higher level of granularity than new data. So, data granularity level depends on original data. [5] also proposed to aggregate old data but proposed to let the data warehouse administrator decide about their granularity level.

In [6], authors proposed to delete data that no longer affect materialized views of the data warehouse. [7] proposed to work only on samples and aggregated data by giving approximate answers to user queries.

In our approach, we don't suggest to delete old data, because data warehouses are built to compute patters from old and new data. Also, avoiding some details in data may hide important analysis. That's why we first proposed in [8] to provide approximations of user queries, and then we propose, in this paper, an hybrid approach toward reducing data in a data warehouse to allow users to work more efficiently on a subset of data with closely the same information as in the original cube.

2. Background and Adaptation of Used Techniques to the Multidimensional Data Framework

2.1 Principal Component Analysis

The Principal Component Analysis (PCA) was introduced by Karl Pearson in 1901 [9]. It offers a statistical technique commonly used for finding patterns in data of high dimensions. It was successfully used in face recognition and image compression. That's why we try to use it in the context of data warehouses, where facts are described according to several dimensions (in order of 12 to 20 [10]).

In a PCA context, data must be quantitative, continuous and are given in a matrix with individuals as lines and variables as columns. In the data warehouse's context, individuals are facts and variables are dimensions. The PCA is then used to extract patterns so that we can compress data by reducing the number of dimensions without much loss of information. To extract these patterns, we use the PCA to find relationships between facts by evaluating their similarities and also between dimensions by evaluating their relationships. We use in this task statistical concepts such as covariance matrix, eigenvectors and eigenvalues, etc. (more details are given later in this paper).

2.2 The Rough Set Theory

The Rough Set Theory (RST) was introduced by Z. Pawlak at the beginning of the 80s. It offers a theoretical foundation for handling vague concepts and badly defined borders [11]. It was an important evolution in the inductive learning from incoherent data and was applied in a wide variety of applications such as medicine, industry, finance, commerce, etc.

The RST is mainly based on two notions: (1) the indiscernibility that expresses the degree of similitude between objects, and (2) the approximation that allows a description of a given set of objects (called a concept) based on possible flaw in data.

In a RST context, an information system is constituted of U that is a non-empty finite set of objects called the *universe* and A that is a non-empty finite set of attributes. In order to make approximations and data mining tasks, these attributes are divided into two disjoint subsets: C for condition attributes used to describe objects and D for decision (classification) attributes used to partition the universe into several concepts (groups of objects) in which objects will be approximately classified.

In a multidimensional framework, U is the multidimensional database, objects are facts and attributes are dimensions. So characteristic dimensions describe facts and decision dimensions partition the database into several fact groups (concepts).

Then, two facts are *indiscernible* with respect to P, i.e. a subset of condition dimensions, if they have the same value for each dimension in P. So, the multidimensional database can be partitioned into equivalent classes, each class contains facts that are indiscernible from each other with respect to P.

For each concept X, we can compute two principal approximations: the lower approximation, i.e. Lower(X), witch is a subset of the concept, and the upper approximation, i.e. Upper(X), which is a superset of the concept. The lower approximation is constituted of the union of equivalent classes that are totally included in the concept, while the upper approximation is constituted of the union of equivalent classes that have intersection with the concepts. That's why the lower approximation contains only facts that are indiscernible with each other and belong actually to the described concept, while the upper approximation may contain facts that are not necessarily elements of the described concept but are indiscernible with one or more facts from the described concept.

3. Algorithm

The following algorithm for multidimensional data reduction presents our approach to help data warehouse's user to identify the most important dimensions that resume the whole system content without loss of pertinent information. Then, instead of working on a specific concept, our approach proposes both: (1) a restricted approximation of this concept (the lower view) that contains facts that are indiscernible with each other and belong actually to that concept, and (2) a more large approximation (the

upper view) with facts that belong partially to this concept but are indiscernible with one or more facts of that concept. In fact, some concepts are large and badly defined, so working on a subset with more restricted data is important and may be really useful for the analyst. Also, concepts may be much more restrictive and few data belong to them, so tolerate and accept less restrictions on such concepts may be useful to understand and analyse them. This is what we try to achieve with computing two approximations for each user concept.

Algorithm DWH_Reduction

1: **Input**
2: FT: fact table with n dimensions and k measures
3: dec: decision dimension with m different values (so m concepts: $c_1,\ldots,c_m$)
4: D: set of the (n-1) remaining dimensions
5: **begin** { DWH_Reduction}
6: facts ← ComputeFacts(FT)
7: **for all** d ∈ D **do**
8: d ← d - avg (d) /*we subscribe the average across each dimension from this dimension*/
9: **end for**
10: M ← compute_covariance_matrix (facts)
11: eigenValues ← compute_eigenvalues (M)
12: eigenvectors ← compute_eigenvectors (M)
13: eigenvectors ← compute_order(eigenvectors) //by eigenvalues from highest to lowest
14: characteristic_dimensions ← {d ∈ D /eigenvalue (d) > avg (eigenvalues(D))}
15: facts ← new_facts_set(facts, characteristic_dimensions∪{dec})
16: Eq_Classes ← compute_equivalent_classes (facts, characteristic_dimensions)
/* Eq_Classes = $Cl_1,\ldots,Cl_k$ *are groups of indiscernible facts with respect to characteristic dimensions**/
17: **for all** c_i in $c_1,\ldots,c_m$ **do**
18: LowerView_ c_i ← ∅
19: UpperView_ c_i ← ∅
20: BoundaryRegionView_c_i ← ∅
21: examples_ c_i =compute_examples_set(facts,dec,c_i) // the set of facts with dec = c_i
22: **for all** Cli in $Cl_1,\ldots,Cl_k$ **do**
23: **if** ($Cl_i \subseteq$ examples_ c_i)
24: **then** LowerView_ c_i ← LowerView_ $c_i \cup Cl_i$
25: **end if**
26: **if** (Cl_i) ∩ examples_ $c_i \neq \emptyset$)
27: **then** UpperView_ c_i ← UpperView_ $c_i \cup Cl_i$
28: **end if**
29: **end for**
30: BoundaryRegionView_c_i ← UpperView_ c_i - LowerView_ c_i
31: **end for**
32: PositiveView_dec ← LowerView_ c_1U…U LowerView_ c_m
33: NegativeView_dec ¬ UpperView_ c_1U…U UpperView_ c_m
34: **end** { DWH_Reduction}

The algorithm has two main steps. In the first step (lines 1-13), an adaptation of the principal component analysis[1] is performed in order to use it with multidimensional data. This first step computes the most important dimensions that describe the whole database content so that less important dimensions can be ignored (line 14). Here, we consider only dimensions with eigenvalues greater than the overage of all dimensions eigenvalues.

After deriving the new data set (line 15), we compute all equivalence classes (line 16). Each class contains facts that are indiscernible with respect to the subset of dimensions computed in the first step of this algorithm. Then, knowing that the set of distinct values of the analysed dimension *dec* is $(c_1,\ldots,c_m)$, we compute for each analysed concept c_i, its set of examples. This set includes facts with *dec* = c_i (line 21). Then, the union of equivalence classes entirely included in this set of examples is the lower view of ci, while the union of equivalence classes that have intersection with this set of examples is the upper view of ci (lines 22-29). The algorithm computes also the boundary region view of c_i (line 30). This view includes facts indiscernible with one or more examples of c_i but belong at the same time to another concept, that's why identifying them is certainly important for the analyst. To finish, we compute the positive view of *dec* with all facts that belong without ambiguity to a specific concept of dec, and the negative view with facts that are indiscernible with each other but can refer to other concepts (lines 32 and 33).

Finally, it's up to the analyst to make restrictions or to enlarge the set of data answering its queries.

4. Conclusion

This paper proposes an approach toward reducing data in a data warehouse in order to focus only on important data with almost the same information as in the original database. This approach is based on an adaptation of the Principal Component Analysis to the multidimensional framework in order to highlight the most important dimensions of the data warehouse. Then, the Rough Set Theory is used to keep only a reduced set of fully matching facts according to the subset of most important dimensions.

References

[1] Inmon, B., Strauss, D., Neushloss, G. (2008). Architecture for the Next Generation of Data Warehousing. *DW 2.0,* Elsevier Press.

[2] Skyt, J., Jensen, C.S., Pedersen, T.B. (2001). Specified-based Data Reduction in Dimensional Data Warehouses, TimeCenter TechReport TR-61.

[3] Jensen, C. S. (1995). Vacuuming, in the TSQL2 Temporal Query Language, R. T. Snodgrass editor, Chapter 23, p. 451-462, Kluwer Academic Publishers.

[4] Chen, Y., Dong, G., Han, J., Wah, B., Wang, J. (2002). Multi-dimensional regression analysis of time-series data streams, In Proceedings Int. Conf. VLDB'02, p. 323-334, Hong Kong, China.

[5] Boly, A., Hebrail, G. (2007). Forgetting data intelligently in data warehouses, RIVF 2007, p. 220-227.

[6] Gupta, H., Mumick, I. (2005). Selection of Views to Materialize in Data Warehouse, *In*: IEEE Transactions on Knowledge and Data Engineering, TKDE, p. 24-43.

[7] Chaudhuri, S., Das, G., Motwani, R., Narasayya, V. (2001). A Robust Optimisation-Based Approach for Approximate Answering of Aggregate Queries , *In*: Proceedings of the ACM SIGMOD, p. 295-306, Santa Barbara, California, USA.

[8] Naouali, S., Missaoui, R. (2005). Flexible Query Answering in Data Cubes, *In*: Proceedings 7th International Conference, DAWAK'2005, p. 221-232, Copenhagen, Denmark.

[9] Person, K. On Lines and Planes of Closest Fit to Systems of Points in Space, Philosophical Magazine, 2 (6) 559-572.

[10] Kimball, R., Margy, R. (2002). The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd ed., Wiley. ISBN 0-471- 20024-7.

[11] Pawlak, Z., Grzymala-Busse, J., Slowinski, R., Ziarko, W. (1995). Rough Sets, *Communications of ACM* 38, p. 88-95.

[1] Witch is usually used with matrix