2009 Ninth International Conference on Intelligent Systems Design and Applications

A Business Intelligence Process to support Information Retrieval in an Ontology-Based Environment

Filippo Sciarrone, Paolo Starace
Business Intelligence Division
Open Informatica srl
Via dei Castelli Romani, 12/A Pomezia, Italy
Email: {f.sciarrone,p.starace}@openinformatica.org

Tommaso Federici
Tuscia University
Viterbo, Italy
Email: tfederici@unitus.it

***Abstract*—In this paper we present a Business Intelligence process to dynamically develop multidimensional OLAP schemas to support Information Retrieval in an ontology-based environment. The particular aspect of our work consists in the integration of Information Retrieval techniques, such as the semantic indexing of non-structured documents with some dynamic management techniques of unbalanced hierarchies stored in a Data Warehouse. We show how to develop an ETL process to automatically build OLAP dimensions, inheriting the hierarchic structure of ontologies, with the goal of using the semantically indexed data to carry out multidimensional OLAP analysis. We experimented our system in a company environment with encouraging results.**

***Index Terms*—olap; business intelligence; etl;**

I. INTRODUCTION

Nowadays, the competitiveness of a company - in a dynamic market that constantly produces data - also depends on how fast the company manages to take decisions regarding its business mission. Suffice it to think about the Internet and its e-commerce web sites, or about a telephone company's user data that daily builds up. In such contexts, Business Intelligence (BI) offers a set of tools and systems that can play a key role in the strategic planning processes (e.g. [1], [2], [3], [4]). Besides, today's market also requires that data processing be also for semantics, hence for an automatic elaboration of concepts linked by ontologies, so as to achieve an extra competitive advantage in the business [5]. Moreover, in BI applications, Information Retrieval (IR) is a critical function because most BI applications today rely on traditional keyword searching for their primary retrieval mechanism [6].

In this article we present a BI-based process to support IR in managing and reusing data from an ontology-based datawarehouse, extending a preliminary version of the work presented in [7]. The main features of our proposal concern the development of a process to dynamically extract concepts from a semantically indexed database. In its entirety, the system actually helps semantic search for curricula from a government database of experts, through a concept-based search. Our approach allows us to design and build dimensions over ontologies. The basic idea is to use standard BI techniques to implement a new methodology, aimed at reusing predefined ontologies in a concept-based dictionary to develop multidimensional $OLAP$ (*On-Line-Analytical-Processing*) schemas. Dimensions are obtained from the structure of the ontologies in a dynamical way, namely, by defining only the root level of the very ontology and allowing the system to build the cube dimension automatically. The ontology tree is extracted from a *RDBMS*, the dimension is generated and it is once again memorized into a *RDBMS*. Should the ontologies change, the management engine will answer by modifying the structures existing before the previous execution. Other studies carried out in this field focused on different aspects of this problem. Some aim at extracting schemas without involving the human being. In this context, sometimes ontologies are used to describe the application domain [8], [9], to generate mediators [10] and to semantically describe data sources [11] to support and automate the definition of *Extract-Transform-Load* (ETL) processes. Besides, some frameworks have been defined for the same purpose [12], [13]. Another interesting project is the one illustrated in [8], aimed at the definition of ETL processes by means of a description in natural language. In all such cases, the use of ontologies occurs at a lower level in the application architecture with respect to our.

The paper breaks down as follows: Section II presents a detailed description of the *Dynamic dimension definition* process, ranging from the working hypotheses, to the treatment of *Bridge Tables* and the automatic generation of $OLAP$ dimensions. Section III explains the processes lying behind the integration of the indexed data, pointing out the most significant SQL code parts. Section IV gives an example of star schema through which our proposal was tested. Finally, Section V deals with the conclusions and sets the work to be done in future.

II. DYNAMIC DIMENSIONS DEFINITION

In this section we present the characteristics of the dynamic dimensions definition, from their extraction from the dictionary to the modeling into dimensional structures. This subprocess is the first part of the entire process that include

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.145

also indexed data integration into fact tables. The second part of the process is shown in Section III.

Subsection II-A shows the reference context and the operative restrictions. Subsection II-B illustrates the technique used to manage unbalanced hierarchies and subsection II-C explains the automatic dimension generation process. Finally, subsection II-D shows some problems related to the navigation on ontologies.

A. Working Hypotheses

To preliminary indexing the information system content we adopt a semantic indexing engine.The indexing process that it implements is a two-step process of non structured documents, divided in the three layers illustrated in Figure 1. During the first step, from the *Unstructured Docs* layer to the *Terms Set* layer, the engine, based on the API of the Open Source search engine *Lucene*, indexed each document, thus obtaining a set of index-terms. In the second step, from the *terms set* layer to the *ontologies* layer, these terms were contextualized and associated with the concepts of predefined ontologies.

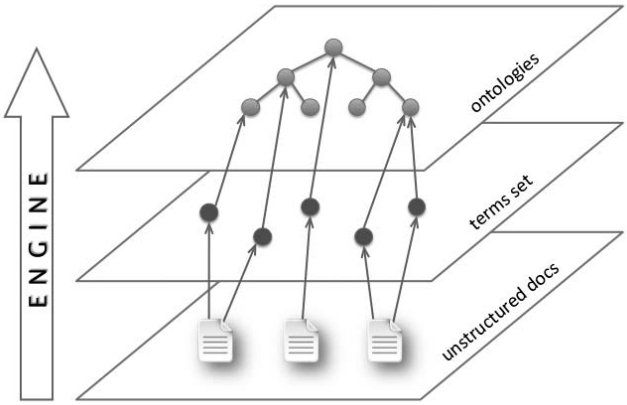

Fig. 1. The Two-Steps Indexing Process.

In order to correctly run our process, the following assumptions were imposed:

- The concepts included in the dictionary were exclusively linked by hypernymy and hyponymy relations;
- Each ontology was based on a hierarchic structure.

The aforesaid restrictions obviously entailed an experimentation that was to be limited to the context, albeit it could also go for other real cases.

B. Managing Unbalanced Hierarchies

When representing a hierarchy with a constant depth, one knows beforehand the type of structure that will be obtained by implementing the concept tree [14]. The rigid structure of constant depth hierarchies means that the number of levels is decided statically, when it is defined. In the case in point, an ontology is therefore to be considered as the number of non uniform levels, whose number may change and, above all, is not known beforehand. In the development of our solution it was deemed crucial to make sure that the ontological tree be extracted dynamically, so as to make the use of concepts as dimensions flexible and easily adaptable.

Representing an arbitrary and irregular hierarchy is an intrinsically hard task in a relational environment [15]. The approach required to manage this type of unpredictable hierarchy is that of including, in the records of the table in which the hierarchy is memorized, a recursive pointer going back to the parent concept, recursive on every dimension record of the concept. Despite such a solution offers a compact and efficient method to represent an arbitrary hierarchy, this kind of recursive structure cannot be used efficiently with the standard SQL language. The *GROUP BY* function cannot follow a recursive structure on the dimension. The adopted solution envisages the inclusion of a bridge table between the concept dimension and the facts table. In literature, Kimball suggests this method to manage the dimensions that recursively refer to records on the same table [15], [16]. The goal of the bridge table was to help the $OLAP$ engine aggregate data more quickly (that's why it is called *helper table*). A bridge table contains a record for each detectable path in the ontology tree, including a zero length path from a concept to itself. That's why there are generally many more records in a bridge table than in the concept dimension. To this aim, we firstly convert the recursive relational schemas into an unbalanced n-ary tree and secondly we convert it into a relational structure again (the second part of this translation process is illustrated in Figure 2).

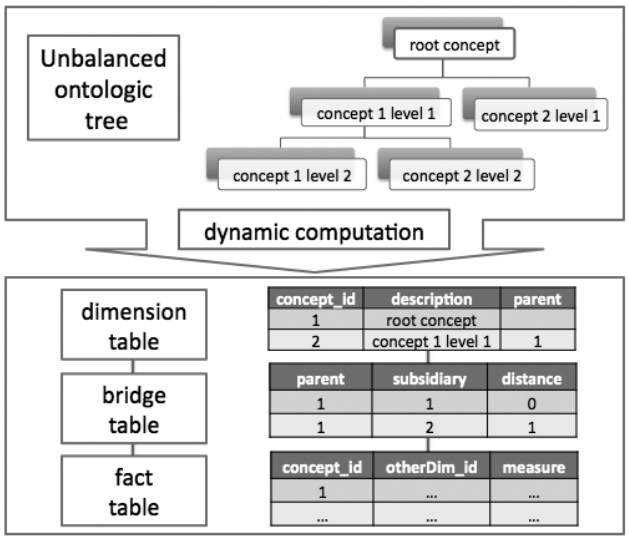

Fig. 2. The Second Step of the Translation Process.

C. Automatic dimensions generation

To uncouple the user from the manual definition of hierarchies, the system generate the dimension structure starting from the tree's root concept. By doing so, the user may actually

not know the logical structure of the hierarchy because the latter is defined automatically.

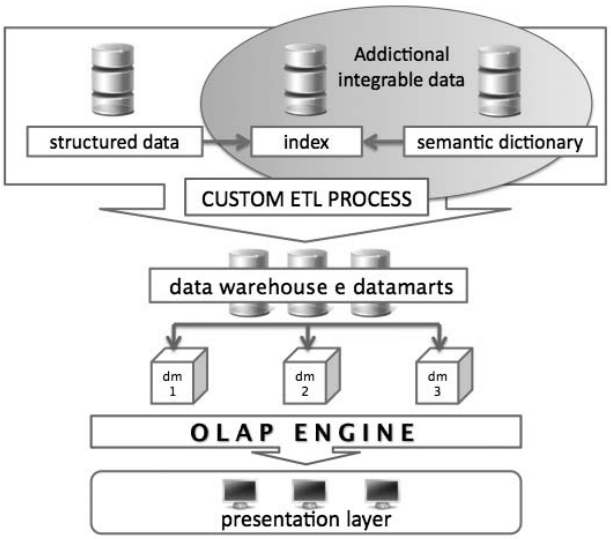

Fig. 3. The Overall Process.

Figure 3 shows the entire process performed by the overall system. In particular we developed a custom ETL module in order to integrate semantically indexed data with operational ones. The custom ETL process performs the following operations:

1) Builds the ontological tree, extracting it from the dictionary;
2) Defines the dimension table;
3) Includes the ontology nodes;
4) Defines the bridge table;
5) Includes a record for each identifiable path on the tree, along with the distance between the relevant nodes (including the 'zero length' path from a concept to itself).

Step (1) is made by recursively operating on the concept relations table included in the dictionary. Steps (2) and (3) are carried out by a function that extracts the tree (to do this one may choose any algorithm s/he wants, such as the depth-first or breadth-first visit). Even steps (4) and (5) are carried out recursively on the tree, making use of a function that is executed on every node.

D. Hierarchic Navigation

In order to ensure a hierarchic navigation it is necessary to bring ontology structure back to a tree structure, so this is a very important aspect that should be considered. The presence of navigable cycles on the structures is ruled out - even theoretically - from the typology of relation existing between the concepts. The relation between the nodes, namely, the hyponymy one (generally known as *part of*) ensures the non navigability of a hyponym concept towards a hypernym concept (possibly separated by several intermediate levels). We must therefore consider the management of *Directed Acyclic Graphs* (DAG). In theory, it is possible to make any DAG go back to a tree structure, but the operation could have a high cost. If such a possibility is taken into consideration, one must intervene by including restrictive hypotheses. It can be proved that the complexity of the desired transformation depends on the depth of the graph, but not on its width; indeed, it consists in the duplication of the shared hyponym node into two hypernyms and the entire relevant subtree. As the graph gets deeper there is an increase in the possibility of carrying out an exponential number of duplications of subtrees referring to conflicting nodes. As the width increases, so does the number of possible conflicts, but the complexity of the single transformations always depends exponentially on the depth of the single subtrees, not on their width. By making an assessment, with reference to the specific intervention context, of the relation between these two parameters, it is possible to obtain the actual cost of the transformation, hence assessing its impact on a computational level. Other approaches to the problem solution might include an intervention on the presentation logic used by the $OLAP$ engine. Dynamically modifying the ontology structures (giving a different weight to the arcs pointing at the same node, or cutting some of them), following the user's behavior while surfing, would solve the problem, but it would also increase the computational cost deriving from data aggregation, including the costs due to the computation of a new tree. Besides, it would introduce a non negligible approximation, due the modification of the tree structure.

III. INDEXED DATA INTEGRATION

In this section we illustrate the integration process of indexed data in a table of facts.

Some specific information is necessary to start the process, but by introducing some standardizations (for example as regards the names of dimensions and of facts, or the names of the columns and their suffixes) it is possible to reduce such information to a limited number of parameters. In this way, the definition of facts expresses a basic step to obtain multidimensional schemas.

The indexed data of a field can be blended with the operational data by means of simple "join" operations. It is sufficient to define the names of the operational tables and those of the involved columns; all the other parameter names can be calculated automatically.

An incremental example will be used to illustrate the automatic composition of the *join* function. The relevant database fields contain the external keys to the NON-TECHNICAL keys contained in the reference dimension. The code used to integrate one single dimension into a table of facts, in order to obtain a simple measure, is the following:

```
SELECT std_dimension_dt.std_dimension_id,
  op_fact_table.measure
FROM op_fact_table JOIN std_dimension_dt
  ON op_fact_table.column =
```

```
    std_dimension_dt.column;
```

It should be pointed out that:

- The names of the columns are the same ones on the basis of the examples shown above;
- The dimension column is not the technical key to the dimension (which is identified with the name *dimension_id*).

Finally, the measure will be aggregated according to the contents of the dimension:

```
SELECT std_dimension_dt.std_dimension_id,
  sum(op_fact_table.measure)
FROM op_fact_table JOIN std_dimension_dt
  ON op_fact_table.column =
    std_dimension_dt.column
GROUP BY std_dimension_dt.std_dimension_id;
```

By using this schematic approach to integrate a greater number of dimensions it is possible to generate complex queries for the definition of facts tables. Queries may be generated repetitively and incrementally. When an ontological dimension is included in a scheme, even the data referring to the search engine index must be aggregated. In order to do so, the indexed data will be initially joined with the operational data, and finally included in the dimensions. The resulting query is:

```
SELECT *
FROM index JOIN op_fact_table
  ON index.op_fact_id = op_fact_table.id;
```

The obtained result set exceeds the real needs, since the ”join” function goes for the entire index. Such function should actually exclusively concern the part of relevant data (which will be memorized beforehand in a table called *temp_index*):

```
SELECT *
FROM temp_index JOIN op_fact_table ON
  temp_index.op_fact_id = op_fact_table.id;
```

In this case, if a dimension deriving from an ontology is to be aggregated, we must use the concept id included in the previous query, including a reference to the ontology dimension as an external key. Hence, the completed query is the following:

```
SELECT ontology_dt.ontology_id,
  std_dimension_dt.std_dimension_id,
  sum(op_fact_table.measure)
FROM ((temp_index JOIN op_fact_table ON
  temp_index.op_fact_id = op_fact_table.id)
JOIN std_dimension_dt
  ON op_fact_table.column =
    std_dimension_dt.column)
JOIN ontology_dt ON
  temp_index.concept = ontology_dt.concept
GROUP BY std_dimension_dt.std_dimension_id,
  ontology_dt.ontology_id;
```

When using bridge tables one should take care in not counting facts too much. That’s why it is important to include only one record for the concepts that appear several times under the same subtree.

IV. Star Schema Example

In this section we illustrate a star schema example that include an ontological derived dimension. The ontology to be integrated in the schema provide a description of the *IT expert* concept and is shown in Figure 4. Consistently with the limitations mentioned in the previous paragraphs, the ontological graph results in an oriented n-ary tree. The resulting schema is shown in Figure 5.

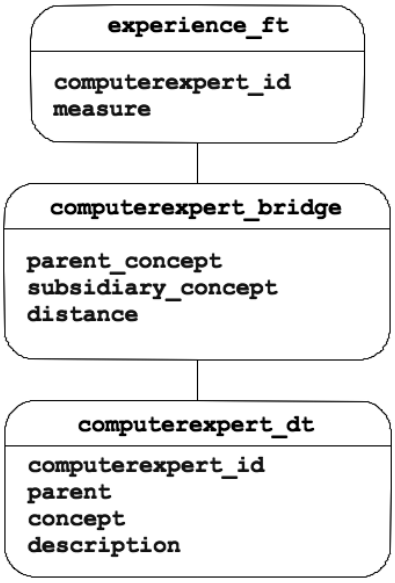

Fig. 5. The Star-Schema with Bridge Table.

The measure on which the aggregation is to be made is the number of candidates referring to the single concept. The resulting table will therefore be a factless fact table because it does not aggregate a number value but a summative conceptual value [17]. The dimensions that may be aggregated, both standard and dynamic ones, are not number-limited, and this makes the solution adaptable to any type of problem.

The result obtained from the navigation of the schema is shown in Figure 6. In the image it is possible to notice that the quality of the ontologies contained in the dictionary is the basic element for a good performance of the presented information. Therefore, the names in the attribute field have deliberately not been refined, so as to highlight this aspect. The sum of the leaves values does not always correspond to the value of top node because there may be concepts that refer

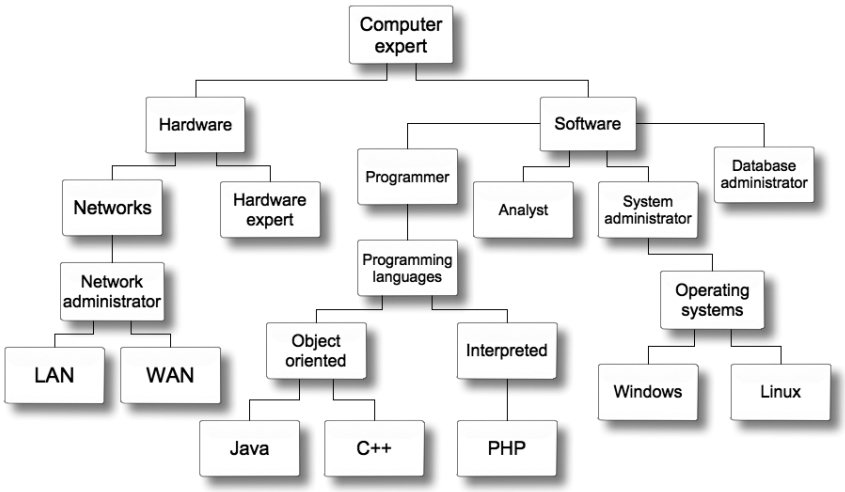

Fig. 4. The Computer Expert Ontology.

	Measures
ComputerExpert	**Expert**
- All ComputerExpert	2.412
- Hardware	429
Technician	33
+ Networking	349
Cyberspace	31
Internet	51
+ Network Administrator	121
LAN	39
WAN	45
- Software	1.942
+ System Administrator	368
- Database Administrator	205
+ Database programming languages	100
DBMS	31
Database	33
+ Programmer	1.252
Software Analyst	36
Project Manager	40

Fig. 6. The Resulting Pivot Table.

only to the parent node that you must add to the value of the sum.

V. CONCLUSIONS AND FUTURE WORK

In this article we proposed a BI-based process to support IR in an ontology-based environment, showing the theoretical implications together with a case study which the overall process is based on. Our process allows for the reuse of ontologies defined in semantic search engines dictionaries as $OLAP$ dimensions thus providing for a stable solution to integrate indexed data in a semantic context. To this aim, a two-steps process was implemented in such a way to let the overall system independent from the complexity of the ontologies. Our future work will focus on the resolution of problems related to the management of many-to-many relations. This aspect is now left to the user's capacity of developing consistent schemas, but it is our intention to introduce a management system based on weighted trees. As for the application testing, a first experimentation has been carried out and described: it was conducted in an industrial context and yielding positive results on the proposal's viability.

REFERENCES

[1] B. Liautaud, *e-Business Intelligence: Turning Information into Knowledge into Profit*. McGraw-Hill, October 2000.

[2] L. T. Moss and S. Atre, *Business Intelligence Roadmap: The Complete Project Lifecycle for Decision-Support Applications*. Addison-Wesley Information Technology Series, March 2003.

[3] H. J. Watson and B. H. Wixom, "The current state of business intelligence," *Computer*, vol. 40, no. 9, pp. 96–99, 2007.

[4] R. Wrembel, *Datawarehouses and OLAP:Concepts, Architectures and Solutions*. IGI Global, December 2008.

[5] N. Zhong, J. Liu, and Y. Yao, *Web Intelligence*. Springer Verlag Berlin Heidelberg, March 2003.

[6] R. C. LaBrie and R. D. S. Louis, "Dynamic hierarchies for business intelligence information retrieval," *International Journal of Internet and Enterprise Management*, vol. 3, no. 1, pp. 3–23, 2005.

[7] F. Sciarrone and P. Starace, "Ontological warehousing on semantically indexed data. reusing semantic search engine ontologies to develop multidimensional schemas," in *KDIR 2009: Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*. To appear, October 2009.

[8] A. Simitsis, D. Skoutas, and M. Castellanos, "Natural language reporting for etl processes," in *DOLAP '08: Proceeding of the ACM 11th international workshop on Data warehousing and OLAP*. New York, NY, USA: ACM, 2008, pp. 65–72.

[9] D. Skoutas and A. Simitsis, "Designing etl processes using semantic web technologies," in *DOLAP '06: Proceedings of the 9th ACM international workshop on Data warehousing and OLAP*. New York, NY, USA: ACM, 2006, pp. 67–74.

[10] T. Critchlow, M. Ganesh, and R. Musick, "Automatic generation of warehouse mediators using an ontology engine," in *Proceedings of the 5th INternational Workshop on Knowledge Represenation Meets Databases (KRDB '98): Innovative Application Programming and Query Interfaces, Seattle, Washington, USA, May 31, 1998*, ser. CEUR Workshop Proceedings, A. Borgida, V. K. Chaudhri, and M. Staudt, Eds., vol. 10, 1998, pp. 8.1–8.8.

[11] S. Toivonen and T. Niemi, "Describing data sources semantically for facilitating efficient creation of olap cubes," in *Poster Proceedings of the Third International Semantic Web Conference*, 2004.

[12] R. M. Bruckner, T. W. Ling, O. Mangisengi, and A. M. Tjoa, "A framework for a multidimensional OLAP model using topic maps," in *WISE (2)*, 2001, pp. 109–118.

[13] P. Spyns, R. Meersman, and M. Jarrar, "Data modelling versus ontology engineering," *SIGMOD Record (ACM Special Interest Group on Management of Data)*, vol. 31, no. 4, pp. 12–17, Dec. 2002.

[14] I. Y. Song, I. yeol Song, C. Medsker, E. Ewen, and W. Rowen, "An analysis of many-to-many relationships between fact and dimension tables in dimensional modeling," in *Proc. of the Internationall Workshop on Design and Management of Data Warehouses*, vol. 6, 2001, pp. 1–13.

[15] R. Kimball, L. Reeves, W. Thornthwaite, M. Ross, and W. Thornwaite, *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses with CD Rom*. New York, NY, USA: John Wiley & Sons, Inc., 1998.

[16] R. Kimball and M. Ross, *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (Second Edition)*. Wiley, April 2002.

[17] R. Kimball and J. Caserta, *The Datawarehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data*. Wiley, September 2004.