Automating Multidimensional Design from Ontologies

Oscar Romero
Universitat Politècnica de Catalunya
Dept. Llenguatges i Sistemes Informàtics
c/ Jordi Girona 1-3, E-08034, Barcelona, Spain
oromero@lsi.upc.edu

Alberto Abelló
Universitat Politècnica de Catalunya
Dept. Llenguatges i Sistemes Informàtics
c/ Jordi Girona 1-3, E-08034, Barcelona, Spain
aabello@lsi.upc.edu

ABSTRACT

This paper presents a new approach to automate the multidimensional design of Data Warehouses. In our approach we propose a *semi-automatable* method aimed to find the business multidimensional concepts from a domain *ontology* representing different and potentially heterogeneous data sources of our business domain.

In short, our method identifies business multidimensional concepts from heterogeneous data sources having nothing in common but that they are all described by an ontology.

Categories and Subject Descriptors

H.4.2 [**Information Systems Applications**]: Types of Systems—*Decision support*; H.2.1 [**Database Management**]: Logical Design

General Terms

Algorithms, Design, Theory

Keywords

OLAP, Multidimensional Design, Ontologies

1. INTRODUCTION

Many methodologies and approaches have been presented to design multidimensional DWs in the literature. However, these approaches are typically carried out manually by DW experts over the organization data sources to identify relevant multidimensional knowledge contained in the sources.

In the last years, a few research efforts have tried to automate the design of multidimensional databases in order to free this task of being (completely) performed by an expert, and ease the whole process. To do so, these approaches start from a detailed analysis of the data sources to determine the multidimensional concepts in a reengineering process. However, all of them carry out this reengineering process from relational OLTP (*On-Line Transaction Processing*) systems, overlooking other data sources.

This paper presents a new approach to automate the multidimensional design of DWs. In our approach we propose a *semi-automatable* method aimed to find the business multidimensional concepts from an *ontology* representing our business domain. As shown in figure 1, our input ontology may represent different and potentially heterogeneous data sources. Thus, this method will point out our business multidimensional concepts (i.e., giving rise to conceptual schemas) contained in data sources of our domain having nothing in common but that they are all described by the same domain ontology.

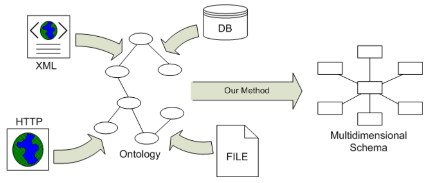


Figure 1: Our method main idea

This work extends a short version [14] by thoroughly analyzing the whole process and justifying the method feasibility. To our knowledge, this is the first method addressing this issue from ontologies and in general, automating the process from non-relational data sources. Hence, we do believe this work opens new interesting perspectives. For instance, we can extend the DW and OLAP concepts to other areas like the *Semantic Web* research area [16], where ontologies play a key role to provide a common vocabulary. One consequence would be that despite the DW design has been typically guided by data available within the organization, we would be able to integrate external data from the web into our DW to provide additional up-to-date information about our business domain.

This approach raises new challenges with regard to current automated modeling. We cannot assume anymore that data sources are implemented over relational databases as current methods do, and we need to focus on the input ontology representing our data sources. Ontologies are semantically richer than relational schemas metadata, and despite, in some cases, we will need to extract missing knowledge by means of data samples, our design process will be guided by knowledge contained in the input ontology, avoiding to perform exhaustive pattern searches over data.

Section 2 discusses the related work underlining automatable approaches. Section 3 sets the foundations of our method that is presented in section 4.

2. RELATED WORK

According to Winter et al. [18], existing approaches to design multidimensional DWs can be classified within a *supply-driven* or *demand-driven* framework. Supply-driven approaches start from a detailed analysis of the data sources to determine the multidimensional concepts in a reengineering process. Most of the methodologies presented in the literature follow this paradigm. For instance, [9, 7, 10] among others. Oppositely, demand-driven approaches focus on determining the user multidimensional requirements (as typically performed in other information systems) to later map them onto data sources as, for instance, [18, 6].

In the literature, partially automatized approaches [7] and those fully automatizing the DW design process [10, 13, 15] always start from a thorough analysis of the relational sources, within a supply-driven framework. These approaches share three main general restrictions [6] not suiting them for multidimensional design over ontologies (and not allowing us to smoothly compare them to our proposal): (1) they exclusively work over relational sources, (2) the complexity of the source schemas must not be high and (3) they mainly work with a *table granularity*. That is, each table in the relational sources is considered as a whole and it is assigned either a fact or a dimension role and therefore, the role of each table attribute is overlooked. However, as already discussed [5], a table, a relationship or even an attribute may be playing a fact/dimension role. Thus, these methods need a certain degree of normalization in the relational schema to work properly. Otherwise, they do not help much in the design process. Nevertheless, this is the case not only of many organizations but also of those applications/areas not using the relational technology.

[10] thoroughly analyzes the data sources, assuming that the database does not contain composite keys, deriving valuable metadata such as functional and inclusion dependencies and key or cardinality information, in order to point out potential *snowflake schemas* [11]. To infer this metadata, they access the instances and perform data mining techniques, hardly extensible to our proposal. Moreover, since they are looking for snowflake schemas, they completely rely on “foreign keys” - “candidate keys” relationships to identify functional and inclusion dependencies. Thus, this method demands a good degree of normalization and, for instance, it cannot face *degenerated dimensions* [11]. Finally, this approach can present problems in terms of complexity due to the high number of permutations computed when looking for inclusion dependencies.

[13] proposes a supply-driven method to be validated by means of a demand-driven process. In this approach, they automatically propose some potential multidimensional schemas that are validated by end-user requirements expressed in terms of MDX queries. We also presented a supply/demand-driven method [15] to point out potential multidimensional schemas from end-user requirements expressed in terms of SQL queries over relational sources. In fact, the method presented in this article is an extension of this previous one from a wider perspective; i.e., without needing to provide end-user requirements beforehand.

Finally, [17] proposes a semi-automated method to design multidimensional DWs from XML schemas. Despite it follows a supply-driven framework, this work was, unlike previous approaches, completely oriented to design *Web Warehouses* [4]. However, from our point of view, data sources from the web should complement the business data already available in the organization and be integrated, altogether, in a single and detailed view of our business domain. Moreover, our approach goes one step beyond since ontologies are more expressive and powerful than XML schemas, providing us with detailed knowledge of the data sources domain and powerful reasoning services easing the process.

3. METHOD FOUNDATIONS

Since our goal is to generate multidimensional schemas in an automated way, this section aims to concisely define and point out those criteria our proposal will be based on; that is, those criteria allowing us to identify multidimensional concepts. Multidimensionality pays attention to two main aspects; placement of data in a multidimensional space and correct summarizability of data. Therefore, our method looks for meaningful conceptual schemas with orthogonal **Dimensions** fully functionally determining **Facts** and free of summarizability problems. The following criteria are addressed to guarantee those schemas proposed by our method accomplish these premises:

- **[C1] The Multidimensional Model**: Multidimensionality is based on the fact/dimension dichotomy. We consider a **Dimension** to contain a hierarchy of **Levels** representing different granularities (or levels of detail) to study data, and a **Level** to contain **Descriptors**. On the other hand, a **Fact** contains **Measures**. One **Fact** and several **Dimensions** to analyze it give rise to a multidimensional schema. These are those concepts we will try to identify along our process.

- **[C2] The multidimensional space arrangement constraint**: **Dimensions** of analysis arrange the multidimensional space where the **Fact** of study is depicted. Each instance of data is identified (i.e., placed in the multidimensional space) by a point in each of its analysis **Dimensions**. Conceptually, it embraces that a **Fact** must be related to each analysis **Dimension** by a many-to-one relationship. That is, every instance of data is related to, at least and at most, one instance of an analysis **Dimension**, and every **Dimension** instance may be related to many instances of data.

- **[C3] The Base integrity constraint**: We denote by **Base** a *minimal* set of **Levels** functionally determining a **Fact**. That is, two different instances of data cannot be spotted in the same point of the multidimensional space. Moreover, **Dimensions** giving rise to a **Base** must be *orthogonal* (i.e., functionally independent) [1]. Otherwise, we would use more **Dimensions** than strictly needed to represent data, and it would generate empty meaningless zones in the space. The concept of **Base** would result in a “primary key” in a relational implementation of an OLAP tool.

- **[C4] The summarization integrity constraint**: Data summarization performed must be correct, and we warrant this by means of the three necessary conditions (intuitively also sufficient) [12]: (1) **Disjointness** (the sets of objects to be aggregated must be disjoint), (2) **Completeness** (the union of subsets must constitute the entire set), and (3) **Compatibility** of the **Dimension**, kind of measure being aggregated and the aggregation function. Compatibility

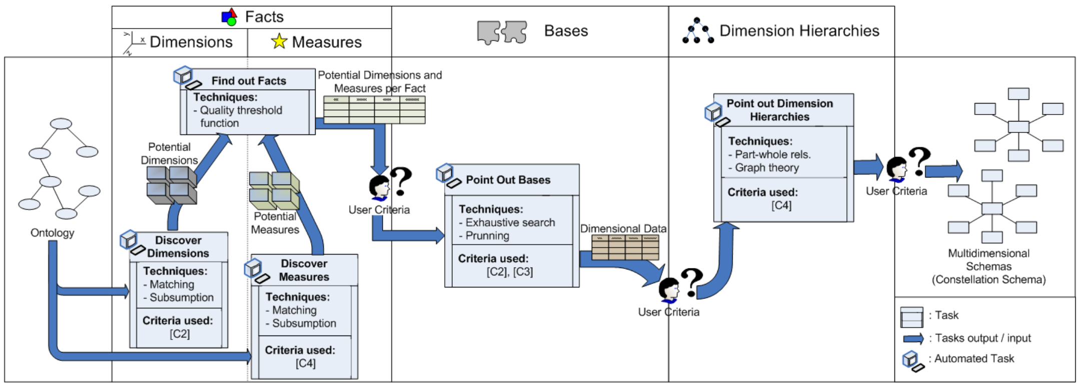


Figure 2: Method Overview

must be satisfied since certain functions are incompatible with some **Dimension**s and kind of measures. For instance, we cannot aggregate `Stock` over `Time` **Dimension** by means of sum, as some repeated values would be counted. However, compatibility will not be automatically checked in our method unless additional metadata was provided.

Finally, bearing in mind that our method input would be an ontology, we also assume the following premise:

- The ontology is expressed in an ontology language providing basic reasoning tools such as *subsumption*, allowing us to work with taxonomies of concepts. Hence, we do not ask to state all possible relationships between concepts in the ontology, since our method will be able to infer them through reasoning. For instance, OWL (*Web Ontology Language*), a W3C recommendation, fits properly for our purposes.

4. OUR METHOD

This section presents a detailed view of our method and how it applies the criteria exposed in section 3. Figure 2 depicts a schematic overview of our method: along three well-differentiated tasks, it points out those concepts presented in [C1] to give rise to multidimensional schemas.

As a previous remark, notice this is a fully supply-driven method. In each step our approach automatically looks for a specific multidimensional concept. At the end of each step it asks for the end-user multidimensional requirements (the only part not being automatable) in order to restrict results got and guide next steps:

- The first task looks for potential subjects of analysis (i.e., **Facts**). Those concepts with most potential **Dimensions** and **Measures** are good candidates. At the end of this task, we ask the user to choose his/her subjects of interest among those concepts proposed by the method as potential **Facts**. The rest of the tasks will be carried out once for each **Fact** identified (i.e., each **Fact** will give rise to a multidimensional schema).

- The second task points out sets of concepts likely to be used as **Base** for each **Fact** identified. **Bases** are compound of concepts labeled as potential **Dimensions**. In short, we look for concepts being able to univocally identify objects of analysis (i.e., factual data).

- The third task gives rise to **Dimension** hierarchies. For every concept identified as a **Dimension**, we conform its hierarchy of **Levels** from those concepts related to it by typical whole-part relationships.

Finally, notice our method input would be an ontology. For instance, consider the ontology extracted from *ResearchCyc*[1] presented in figure 3 that will be used as example along this paper. For the sake of comprehension, it is depicted in UML (*Unified Modeling Language*) notation in spite of using any ontology language like OWL. This subset of *Cyc* corresponds to a given domain; a sales event: a *product* is *bought* in a *specific place* (where the *event occurs at*) by a specific *monetary value*, and where a *seller* and a *buyer* are involved in the transaction.

4.1 Pointing out Facts

In the literature we can find different approaches to point out **Facts** but most of them are hardly automatable. In fact, to point out **Facts** is one of the most difficult steps in the design process, and it is usually done manually [13]. In our approach, we consider a concept to be a potential subject of analysis if it is related to as many potential **Dimensions** and **Measures** as possible. So that, our aim in this section is twofold; (1) discover potential analysis **Dimensions**, and (2) point out potential **Measures**.

4.1.1 Discovering Potential Dimensions of Analysis:

According to [C2], a concept is a potential **Dimension** of analysis if it is related to a **Fact** by a one-to-many relationship; that is, every instance of data is related to one, and just one, of its instances. Hence, we can express our multidimensional pattern to look for potential **Dimensions** as follows:

$$\mathcal{F} \sqsubseteq = 1r.\mathcal{D}, \quad where \;\; r \equiv (r_1 \circ \ldots \circ r_n)$$

Notice it is expressed in "Description Logic" (DL) notation [2] (which OWL is based on), where r and $\mathcal{D}$ are variables,

[1]The *Cyc* ontology is a commercial knowledge repository developed to capture and represent common sense. A full version of Cyc, called ResearchCyc, has been released for the scientific community (http://www.cyc.com).

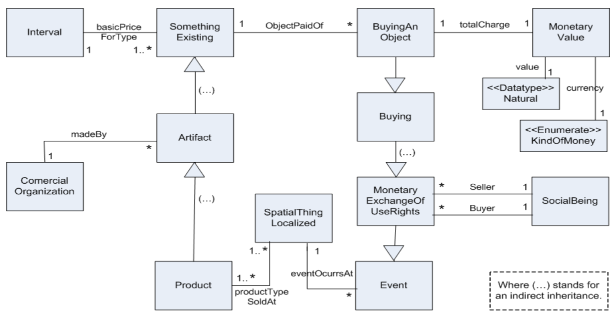


Figure 3: An ontology extracted from *ResearchCyc*

and $\mathcal{F}$ the ontology concept we are trying to identify as a potential **Fact**. About the terminology used, we consider a *class* to be a unary predicate (i.e., $\mathcal{D}$ and $\mathcal{F}$), and a *property* (i.e., r) as a binary predicate expressing a relationship between two classes. Briefly, the $\sqsubseteq$ symbol stands for *subsumption*, the basic inference on classes in DL. Subsumption (i.e., $\mathcal{A} \sqsubseteq \mathcal{B}$) is the problem of checking if the *subsumer* ($\mathcal{B}$) is considered more general than the *subsumee* ($\mathcal{A}$). That is, if the subsumee can always be considered a subset of the subsumer. $\equiv$ stands for a logic *equivalence* and can be defined as a specific kind of subsumption, that is: $\mathcal{A} \sqsubseteq \mathcal{B}$ and $\mathcal{B} \sqsubseteq \mathcal{A}$. $\circ$ stands for properties *composition*. That is, for the two properties $\mathcal{R}$ and $\mathcal{S}$ we have that if $\{a, b\} \in \mathcal{R}$ and $\{b, c\} \in \mathcal{S}$, then $\{a, c\} \in (\mathcal{R} \circ \mathcal{S})$. Finally, $= 1$ stands for *functionality* that is a specific *number restriction* where, in our case, the number of individuals belonging to class $\mathcal{D}$ related to a given individual of the class $\mathcal{F}$, through the property r, must be exactly one. Thus, we are looking for classes ($\mathcal{D}$) such that every instance of a given **Fact** ($\mathcal{F}$) is related, directly or by composition through a set of properties (r), to, at least and at most, one of its instances.

To find those concepts fitting as potential **Dimensions**, we will evaluate the multidimensional pattern against each ontology concept to discover if it is a potential **Fact**. Unfortunately, we may not take advantage of generic reasoning algorithms provided by DL in order to compute this multidimensional pattern since they are not decidable when considering property chains. Moreover, it may not be expressible in some ontology languages like OWL DL. Nevertheless, it is feasible to find an algorithm to compute our multidimensional pattern with a reasonable complexity degree for functional properties. As a demonstration, we propose an ad hoc algorithm to compute this pattern (see figure 4). As discussed later, our algorithm is mainly based on multiplicities among concepts. Therefore, we suppose that the ontology provides multiplicities since nowadays, ontology languages, such as OWL 1.1 [8], allow to state them in the ontology:

Let $\mathcal{M}$ be a matrix of N×N elements where N is the number of concepts in the ontology. In each row we represent a concept and its to-one relationships with the rest of the concepts. Conceptually, a cell of $\mathcal{M}$ is ticked if we can find a to-one (i.e., a functional and complete property) path between both concepts. In other words, if that column represents a potential **Dimension** for that row. Since $\mathcal{M}$ is a sparse matrix, the function *create matrix* implements it as a *vector* of *lists* (step 1). That is, every position in the vector represents a concept, and its list contains its functional

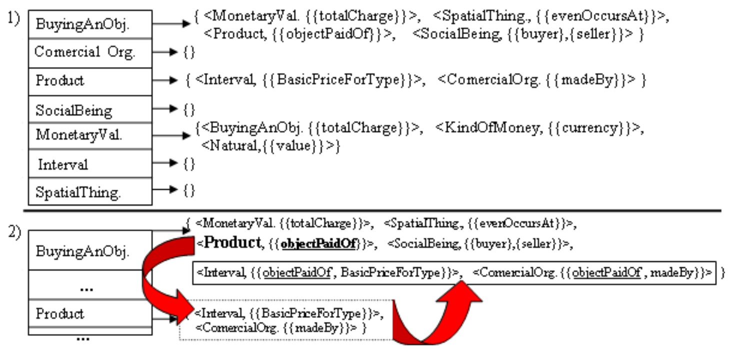
```
typedef list <properties> path
typedef tuple < concept, list<path> > paths_to_concept
typedef tuple < concept, list<paths_to_concept> > func_depend
function create_matrix returns Matrix
    1. vector< list<func_depend> > M;
    2. bool converge = false;
    3. initialize(M);
    4. first_iteration(M, ontology);
    5. while(not converge)
        (a) propagate_path(M, converge);
    6. return M;
void propagate_path (Matrix ↨M, bool ↨converge)
    1. bool converge = true;
    2. foreach concept c in M do
        (a) func_depend ini_depend = M<c>;
        (b) foreach concept p in M<c> do
            i. M<c> ∪ (M<c<p>> ∘ M<p>);
        (c) if(ini_depend ! = M<c>) then
            i. converge = false;
```

Figure 4: An algorithm to look for Dimensions

dependencies (i.e., its *potential* **Dimensions**) as well as the property path between both concepts (see the *func_depend typedef* declaration). Lists are created and initialized to the empty list in step 3.

Step 4 looks for direct to-one relationships for each concept. Conceptually, it would represent the first iteration of our process. This step may be implemented through generic reasoning algorithms provided by DL. Specifically, the *matching* inference algorithm would fulfill our objectives [3]. This algorithm was conceived to match concepts against patterns to support the pruning of large concept descriptions discarding unimportant aspects. More precisely, given a concept pattern $\mathcal{D}$ (i.e., a concept description containing variables) and a concept description $\mathcal{C}$ without variables, the matching problem asks for a substitution σ (of the variables by concept descriptions) such that $\mathcal{C}$ is subsumed by $\sigma(\mathcal{D})$. For instance, considering the pattern "*the research interest of those people being only interested in a single research area*" (expressed in DL as $\forall$`research-interests.X`), and the concept description "*those people only having cats as pets and being only interested in AI research*" ($\forall$`pets.Cat` $\sqcap$ $\forall$`research-interests.AI`), the matching algorithm finds the scientific interests (in this case Artificial Intelligence) described in the concept (i.e., assigns to X the value `AI`).

Since step 4 looks for direct relationships, we can overlook composition in this step. Therefore, the multidimensional pattern is simplified since we do not longer need to consider r as a property composition but as a single property. We can get rid of the variable r performing matching for every isolated property. Hence, matching perfectly fits to our purpose to evaluate the multidimensional pattern for to-one relationships directly related to a **Fact**. With this approach, taxonomies of concepts would be automatically considered since matching uses subsumption to look for a proper substitution. Potential **Dimensions** pointed out in


1)
BuyingAnObj.
Comercial Org.
Product
SocialBeing
MonetaryVal.
Interval
SpatialThing.
{ <MonetaryVal. {{totalCharge}}>, <SpatialThing., {{evenOccursAt}}>, <Product, {{objectPaidOf}}>, <SocialBeing, {{buyer},{seller}}> }
{}
{ <Interval, {{BasicPriceForType}}>, <ComercialOrg. {{madeBy}}> }
{}
{<BuyingAnObj. {{totalCharge}}>, <KindOfMoney, {{currency}}>, <Natural,{{value}}>}
{}
{}
2)
BuyingAnObj.
...
Product
...
{ <MonetaryVal. {{totalCharge}}>, <SpatialThing, {{evenOccursAt}}>,
<Product, {{objectPaidOf}}>, <SocialBeing, {{buyer},{seller}}>,
<Interval, {{objectPaidOf, BasicPriceForType}}>, <ComercialOrg. {{objectPaidOf, madeBy}}> }
<Interval, {{BasicPriceForType}}>,
<ComercialOrg. {{madeBy}}> }

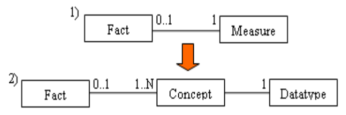
*Note: From all concepts in every taxonomy of concepts, we have only depicted the most relevant concept due to space limitations. That is, Event would have the same list associated than BuyingAnObj.

Figure 5: Example of propagation of to-one paths by composition

this first iteration are added to the proper list in the vector. According to our example, figure 5.1 depicts the vector (i.e., the matrix) $\mathcal{M}$ after performing step 4 over the ontology introduced in figure 3.

Steps 5 and 5a propagate the to-one paths pointed out, by composition. That is, if two concepts, $\mathcal{A}$ and $\mathcal{B}$, are related by a direct to-one relationship, and $\mathcal{B}$ is also related through a to-one relationship to a concept $\mathcal{C}$, $\mathcal{A}$ functionally determines $\mathcal{C}$ as well. Also notice, that to avoid recurrence problems in one-to-one relationships, we must not follow the same property backwards. Following the previous example, figure 5.2 depicts how these steps work. There, potential **Dimensions** of those concepts functionally determined by a given concept $\mathcal{C}$, are also considered potential **Dimensions** of $\mathcal{C}$ (see function *propagate_path*). Moreover, note that the path list has been properly updated (i.e., by concatenating both path list) to let us know the path from $\mathcal{C}$ to its potential **Dimensions**.

Finally, this algorithm converges if we can assure that if in a given iteration vector $\mathcal{M}$ is not updated then, in the following iteration it would not be updated either. It can be guaranteed since, in the worst case, if P is the maximum number of functional properties chained in the ontology, in each iteration steps 5 and 5a would propagate, at least, one property. That is, the path length is *strictly increasing* and at most, in P iterations we would have propagated all chained properties in the ontology. Thus, steps 5 and 5a will not be able to propagate any other property in next iterations. About each iteration, in the worst case, we have an exponential computational cost with regard to the number of functional properties chained. However, it is not a strong limitation, since in a real world case, it is rather difficult to find several to-one chained relationships not accepting zeros. For instance, in our example, from 10 functional relationships, the biggest to-one path is of size 2, and our algorithm would converge in just 2 iterations.



Multiplicity	Fact - Dim.	Fact - Measure
Min. Left	0	0
Max. Left	N	1
Min. Right	1	1
Max. Right	1	N

Figure 6: Multiplicities looked for

4.1.2 *Pointing out Measures:*

In this step we look for **Measures** (i.e., factual data). Typically, **Measures** are numeric attributes allowing data aggregation. In our method, we consider any numeric *datatype* (i.e., those allowing data aggregation by its own nature) to be a **Measure** of a given **Fact** $\mathcal{F}$ if, according to [C4], it preserves a correct data aggregation from $\mathcal{F}$; that is, if they are conceptually related by a one-to-one relationship (see figure 6.1). (1) The to-one multiplicity in the **Measure** side forces each **Fact** instance to be related to just one **Measure** value, and forbidding zeros we preserve *completeness*, (2) whereas the to-one multiplicity in the **Fact** side preserves *disjointness*. Nevertheless, notice that we can allow zeros in the **Fact** side still preserving *disjointness* but assuming that that **Measure** may not be related to any **Fact** instance.

In practice, (3) we can relax this conceptual one-to-one relationship asking for concepts related to a **Fact** by a many-to-one relationship (see figure 6.2). In this case, all those (many) **Measures** values related to a given **Fact** instance need to be aggregated by some *compatible* aggregation function, prior to be inserted in the DW; giving rise to the conceptual one-to-one relationship needed. Therefore, the datatype could be either directly related to the **Fact** or by properties composition, as shown in figure 6.2. In the first case, since a datatype does not have an *object identifier* (i.e., *oid*) we may allow any multiplicity in the **Fact** side without violating *disjointness*. In the second case, the path between the **Fact** and the concept which is directly related to the datatype must accomplish the aggregation conditions discussed above.

Notice that not all these relationships could be detected through the matrix $\mathcal{M}$ computed previously. In fact, we would need two different matrixes; one to store those to-one paths between concepts allowing zeros, and another one storing to-many paths forbidding zeros. However, for the sake of readability and comprehension, they have not been introduced before. These matrixes do not modify the complexity degree of the algorithm, since they can be computed with equivalent algorithms to the one presented to compute matrix $\mathcal{M}$; whereas $\mathcal{M}$ can be computed as the intersection of both matrixes (i.e., those to-one paths not allowing zeros). Finally, note that the baseline algorithm to find **Measures** is essentially the same than the one used to discover potential **Dimensions**: as shown in the right-side table of figure

Concept	Dimensions	Measures
BuyingAnObj.	Interval, ComercialOrg., MonetaryVal, Value (datatype), Seller, Buyer, KindOfMoney, SpatialThing.	Value (Natural)
MonetaryVal.	Interval, ComercialOrg., BuyingAnObj, Value (datatype), Seller Buyer, KindOfMoney, SpatialThing.	Value (Natural)
Product	Interval, ConercialOrg.	-
Others	-	-

Figure 7: Multidimensional knowledge pointed out

6, both algorithms aim to propagate one-to-many relationships allowing zeros in the **Fact**-side. Hence, same considerations about the computational complexity of pointing out **Dimensions** are extensible to find **Measures**.

Once we have pointed out potential analysis **Dimensions** and **Measures** for each concept (see figure 7), we propose to the user those concepts being potential **Facts**. The more potential **Dimensions** and **Measures** a concept has, the higher we rate it to be a good candidate as a potential **Fact**. Thus, we define f as a function that, given the number of potential **Dimensions** and **Measures** related to a concept c, it evaluates c as a promising **Fact**. This quality function will prune those concepts not reaching a threshold that would be provided by the user. Potential **Facts** not pruned are ordered according to its f value and presented to the user that must choose those **Facts** making more sense for him/her. In our example `BuyingAnObject` and `Monetary-Value` are clear candidates, but it would be reasonable to expect the user to just select `BuyingAnObject` as his/her **Fact** of interest.

All in all, as discussed, the baseline algorithm computational complexity to point out **Facts** would be exponential. For every concept we look for its potential **Dimensions** and **Measures** (both being exponential with regard to the maximum length of proper relationships we are looking for). Let N be the number of concepts in the ontology; c the maximum functional *connectivity* (i.e., direct to-one relationships from a concept); and l the maximum chain of functional properties. In the worst case (i.e., when the ontology conforms a *clique* of to-one properties), we will need *N-1* steps (i.e., the value of l in a clique) to propagate the to-one paths and determine that every concept is a potential **Dimension** of any other. In a smart implementation of this algorithm, we have been able to raise a $\Theta(N \times c^l)$ complexity assuring that each element (i.e., each cell of matrix $\mathcal{M}$) will be computed in one step. That is, propagating the functional dependencies from the *end of the functional paths* to the *beginning.*

Nevertheless, we would like to underline that this is a theoretical upper bound since it is highly unlikely to find an ontology where all concepts have maximum connectivity and all its functional paths are of maximum length. Moreover, the algorithm cost is exponential w.r.t. l, and in real world cases it is rare to find large to-one property chains.

4.2 Looking for potential Bases

The second step points out possible **Bases** for each identified **Fact** among those concepts related to it and labeled as its potential **Dimensions** of analysis.

According to [C3], we need to find a set of **Dimensions** identifying the **Fact** unequivocally. **Bases** must contain orthogonal **Dimensions**, and a set of potential **Dimensions** will be considered a *feasible* **Base** if they are able to identify

```
function seek_bases (Fact F, Matrix M) returns Set<Base>
  1. int i=1; Set<Concept> Base;
  2. Set<Base> Bases = {}, Combinations =
     Get_Potential_Dimensions(F,M), Candidates_Sets;
  3. while(Combinations != ∅)
     (a) Candidates_Sets = {};
     (b) Base = Get_First_Combination(Combinations);
     (c) while(Base)
           i. if(Feasible_Base(Base)) then
                A. Bases += Base;
          ii. else if(Intermediate_Bases(Base)) then
                A. Candidates_Sets += Base;
         iii. Combinations −= Base;
          iv. Base = Get_Next_Combination(Combinations);
     (d) i++;
     (e) Combinations = Generate_Combinations_by_Size(i,
         Candidates_Sets, M);
  4. return Bases;
```

Figure 8: A smart algorithm to look for Bases

all instances of a **Fact**:

$$\prod |\mathcal{D}_i| \geq |\mathcal{F}|$$

That is, the **Fact** cardinality must be lower than or equal to the product of the cardinalities of those **Dimensions** giving rise to a **Base**.

In our approach, given a set of potential **Dimensions**, they are evaluated to be a feasible **Base** as follows. First, by means of data samples from our involved data sources, we estimate the potential **Dimensions** cardinalities (i.e., every $|\mathcal{D}_i|$). As introduced in [C2], a **Fact** is related to **Dimensions** by many-to-one relationships. So that, if any many-side of these relationships provides numerical multiplicities and we know the **Dimensions** cardinalities, we can accurately estimate the **Fact** size as $|\mathcal{D}_i| \times multiplicity$; since every instance of $\mathcal{D}_i$ is exactly related to *multiplicity* **Fact** instances. If two different cardinalities are inferred from the same **Fact** then either the ontology or the sources sampled are not sound and must be reconsidered. If multiplicities available are ranged (i.e., *minMultiplicity..maxMultiplicity*) we estimate the lower bound of the **Fact** size as $max(|\mathcal{D}_i| \times minMultiplicity_i)$. Notice we do not set an upper bound with regard to *maxMultiplicity* since it would only assure we are selecting **Bases** giving rise to dense **Cubes**; which would be considered later.

Otherwise, if the ontology does not provide numerical multiplicities for the **Fact** - **Dimension** relationships, we will also need to sample our data sources in order to estimate the **Fact** cardinality; however it is important to sample it from the same sites as **Dimensions** to avoid incoherent results.

After that, we start combining potential **Dimensions** to point out those combinations identifying the **Fact**. A naive solution would embrace to check all possible combinations between potential **Dimensions**. However, it would not only be expensive (i.e., exponential with regard to the number of potential **Dimensions**) but also inappropriate since, according to [C3], **Dimensions** conforming a **Base** should be orthogonal, and many concepts have functional dependencies as explained in subsection 4.1. Figure 8 presents a smarter and efficient algorithm to find all feasible **Bases**. Essentially, this algorithm selectively generates sets of potential **Dimensions** likely to be **Bases**, according to the following pruning rules based on [C2] and [C3]:

Proposition 1. *Let $\mathcal{B}$ and $\mathcal{W}$ be sets of potential* **Dimensions** *of a given* **Fact** *$\mathcal{F}$. If $\mathcal{B} \subseteq \mathcal{W}$ and $\mathcal{B}$ is known to be a* **Base** *of $\mathcal{F}$ then, $\mathcal{W}$, despite functionally determining $\mathcal{F}$, is not considered a* **Base** *of $\mathcal{F}$ since it is not minimal; there exists a subset of $\mathcal{W}$ (i.e., $\mathcal{B}$) functionally determining the* **Fact**.

Corollary 1. *Let $\mathcal{B}$ be a set of potential* **Dimensions** *fully functionally determining $\mathcal{F}$. According to proposition 1, this set would only be generated as a combination if all its subsets were generated and proved not to be* **Bases** *of $\mathcal{F}$. Otherwise, $\mathcal{B}$ would not be minimal and, therefore, it would not be a* **Base** *of $\mathcal{F}$.*

Proposition 2. *Let a and c be two concepts such that $\{a\}$, $\{c\}$ have been proved not to be* **Bases** *of $\mathcal{F}$. According to [C3], $(\{a\} \cup \{c\})$ would be generated as a combination iff, a and c are orthogonal.*

Corollary 2. *Let $\mathcal{B}$ be a set of potential* **Dimensions** *and a and c two concepts. According to proposition 2, $\{a, c\}$ was generated at the end of the first iteration iff a and c were orthogonal. Moreover, according to corollary 1, $\mathcal{B}$ is generated iff all its subsets were generated and refuted as* **Bases** *of $\mathcal{F}$. Consequently, if $\mathcal{B}$ was generated, we can say that $\forall a, c \in \mathcal{B}$, a and c are orthogonal. Otherwise, $\{a, c\}$ would have not been generated and $\mathcal{B}$ neither.*

Proposition 3. *Let $\mathcal{B}$ be a set of potential* **Dimensions** *and c a concept such that $c \in \mathcal{B}$. Let $\mathcal{I}$ be the set of intermediate concepts giving rise to the many-to-one path between $\mathcal{F}$ and c. We say that $\mathcal{B}$ might be a* **Base** *of $\mathcal{F}$ iff, for each $\{c_i\} \in \mathcal{I}$, $\mathcal{B} - \{c\} + \{c_i\}$, namely the intermediate* **Bases** *of c, has been proved as a* **Base** *of $\mathcal{F}$; since each intermediate concept determines c and then, if $\mathcal{B}$ is a* **Base** *then $\mathcal{B} - \{c\} + \{c_i\}$ should also be a* **Base**. *Otherwise, we can assure it would not be a* **Base**.

With these premises, our algorithm drastically reduces the searching space. Mainly, it generates (*i+1*)-sized sets from those *i*-sized *candidate sets*. An (*i+1*)-sized set is generated if, in the previous iteration, all its *i*-sized subsets have been proposed as *candidate sets* (see step 3e). When it is not able to generate new (*i+1*)-sized sets the process ends (step 3). But how are sets pointed out as *candidate sets*? According to propositions stated above.

As stated in corollary 1, if a set $\mathcal{B}$ is proved to be a feasible **Base**, all those sets containing $\mathcal{B}$ must not be generated (i.e., in step 3(c)i current **Base** is not proposed as a *candidate set*). For instance, according to figure 5, if {`Product, SocialBeing`} has been proved to be a **Base** of `BuyingAnObject`, any other set containing it (for instance, {`Product, SocialBeing, SpatialThing-Localized`} would not be generated). According to proposition 3, step 3(c)ii only considers as *candidate sets* those sets whose *intermediate* **Bases** were proved to be **Bases**. Realize the importance of this rule since, something refuted as a **Base** invalidates all those sets having it as an *intermediate* **Base** (i.e., they are not added to the *candidate sets*). In addition, also notice that, as a prerequisite of this pruning rule, *i*-sized combinations generated in a given iteration must be treated in a proper order to avoid checking it more than once. For instance, following our example, {`Interval, SocialBeing, SpatialThing-Localized`} could be a **Base** of `BuyingAnObject`, iff {`Product, SocialBeing, SpatialThing-Localized`} (i.e., an *intermediate* **Base**) is a **Base** of `BuyingAnObject`.

Finally, the last prune rule is applied in step 3e of the first iteration of the algorithm. According to proposition 2, only those *2*-sized sets whose concepts are orthogonal (i.e., not appearing in its potential **Dimensions** list in the matrix $\mathcal{M}$ -see subsection 4.1-) must be generated. For instance, if {`BuyingAnObject`} and {`MonetaryValue`} were potential **Dimensions** of a given **Fact**, they could not give rise to {`BuyingAnObject, MonetaryValue`} since each one appear in the functional dependencies list of the other.

Once **Bases** have been detected, we sort them according to the product value of multiplicities (or cardinalities) of participating **Dimensions**. Values closer to the **Fact** cardinality will give rise to more dense (not sparse) **Cubes** resulting in more efficient designs and, in the end, more intuitive **Cubes** to be handled by the user. Next, the user must choose those **Bases** making more sense for him/her. Moreover, it would be necessary to allow the user to invalidate an, a priori, valid **Base**. If it happened, the process would be launched again to propose other **Bases** pruned as well as to discard some selected, according to the feedback provided: which feasible **Bases** are **Bases** indeed, and which not.

4.3 Giving rise to Dimension hierarchies

In the previous step (see 4.2) we have pointed out **Dimensions** of analysis, but we still need to shape their hierarchies in order to allow summarizability of data; one of the multidimensionality principles.

Dimension hierarchies must guarantee a correct summarizability of data (see [C4]). Thus, in this step we look for to-one relationships (also known as "Roll-up" relationships) giving rise to hierarchies allowing a correct data aggregation: the to-one multiplicity preserves *disjointness* of aggregated data, and forbidding zeros we also preserve its *completeness.*

Starting from each concept identified as a **Dimension**, a directed graph following all to-one relationships paths is depicted, using the matrix $\mathcal{M}$ built in subsection 4.1. Notice that, at this moment, we cannot differentiate the role played by each graph node (i.e., concepts); either as a **Level** or as a **Descriptor**. However, two specific cases can give us more information about it. (1) If any concept is placed in more than one graph, we consider it to be a **Level** (since it seems interesting to show data at this granularity level), and we relate those **Levels** by semantic relationships to denote their conceptual relationship. (2) One-to-one relationships in a graph are depicting either a conceptual relationship between two different **Dimensions** or an attribute of a **Level** (i.e., a **Descriptor**). Then, if the *ending* concept (notice the graph is directed) was identified as a **Dimension**, both concepts would be properly related according to (1). Otherwise, we consider it to be a **Descriptor** of the *initial* concept.

Finally, directed graphs deployed are presented to the user as **Dimension** hierarchies, altogether with those semantic relationships between **Dimensions** pointed out. With these premises, in most cases we have not been able to iden-

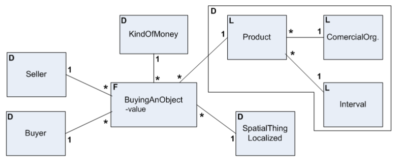


Figure 9: Resulting multidimensional schema

tify each graph concept either as a **Level** or a **Descriptor**. However, this is sound, since it is up to a design decision to spot each concept as an attribute of an existing **Level** or as a new **Level**; giving rise to *star* or *snowflake* schemas [11]. Consequently, this differentiation should be made by the user, if he/she is interested in aggregating data at this level, when stating his/her requirements.

Eventually, after carrying out all the method steps and asking the user for his/her requirements at the end of each step, for each **Fact** identified, we would get a multidimensional schema like the one depicted in figure 9.

5. CONCLUSIONS

In this paper we have presented a semi-automated method to point out multidimensional concepts from an ontology representing our business domain. In our approach, we use ontologies as well as reasoning tools provided by ontology languages to look for multidimensional patterns. Currently, we have carried out a complete simulation of a real case study to validate the algorithm. Moreover, we have also presented an in depth theoretical study of the algorithms complexity to justify its feasibility.

We believe this paper to be the first to address the issue of semi-automatic multidimensional design from ontologies. Up to now, traditional approaches were typically carried out manually and were restricted to work over relational sources. Conversely, our approach is able to integrate information from heterogeneous data sources describing their domain through ontologies. One of the most promising areas where to apply our method is the Semantic Web, giving rise to new possibilities like extracting and integrating external data into our DW. In fact, since data warehousing embraces, essentially, data integration, and the web is the biggest source of data available, we can consider the web as the source of the biggest DW to be exploited.

6. ACKNOWLEDGMENTS

This work has been partly supported by the Ministerio de Educación y Ciencia under project TIN 2005-05406.

7. REFERENCES

[1] A. Abelló, J. Samos, and F. Saltor. **YAM**^{2} (Yet Another Multidimensional Model): An extension of UML. *Information Systems*, 31(6):541–567, 2006.

[2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[3] F. Baader and R. Küsters. Matching Concept Descriptions with Existential Restrictions. In *Proc. of 7th Conf. on Principles of Knowledge Representation and Reasoning*, pages 261–272. M. Kaufmann, 2000.

[4] S. S. Bhowmick, S. K. Madria, W. K. Ng, and E.-P. Lim. Web Warehousing: Design and Issues. In *Proc. of ER '98 Workshops on Advances in Database Technologies*, volume 1552 of *LNCS*, pages 93–104. Springer, 1999.

[5] L. Cabibbo and R. Torlone. A Logical Approach to Multidimensional Databases. In *Proc. of 6th Int. Conf. on Extending Database Technology*, volume 1377 of *LNCS*, pages 183–197. Springer, 1998.

[6] P. Giorgini, S. Rizzi, and M. Garzetti. Goal-oriented Requirement Analysis for Data Warehouse Design. In *Proc. of 8th Int. Workshop on Data Warehousing and OLAP*, pages 47–56. ACM Press, 2005.

[7] M. Golfarelli, D. Maio, and S. Rizzi. The Dimensional Fact Model: A Conceptual Model for Data Warehouses. *Int. Journal of Cooperative Information Systems*, 7(2-3):215–247, 1998.

[8] I. Horrocks, O. Kutz, and U. Sattler. The Even More Irresistible SROIQ. In *Proc. of 10th Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 57–67. AAAI Press, 2006.

[9] B. Hüsemann, J. Lechtenbörger, and G. Vossen. Conceptual Data Warehouse Modeling. In *Proc. of 3rd Int. Workshop on Design and Management of Data Warehouses*. CEUR-WS.org, 2000.

[10] M. R. Jensen, T. Holmgren, and T. B. Pedersen. Discovering Multidimensional Structure in Relational Data. In *6th Int. Conf. on Data Warehousing and Knowledge Discovery*, volume 3181 of *LNCS*, pages 138–148. Springer, 2004.

[11] R. Kimball, L. Reeves, W. Thornthwaite, and M. Ross. *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses.* John Wiley & Sons, Inc., 1998.

[12] H. Lenz and A. Shoshani. Summarizability in OLAP and Statistical Data Bases. In *Proc. of 9th Int. Conf. on Scientific and Statistical Database Management*, pages 132–143. IEEE, 1997.

[13] C. Phipps and K. C. Davis. Automating Data Warehouse Conceptual Schema Design and Evaluation. In *Proc. of 4th Int. Workshop on Design and Management of Data Warehouses*, volume 58, pages 23–32. CEUR-WS.org, 2002.

[14] O. Romero and A. Abelló. Generating Multidimensional Schemas from the Semantic Web. In *Proc. of CAiSE Forum'07, Poster*, volume 247, pages 69–72. CEUR-WS.org, 2007.

[15] O. Romero and A. Abelló. Multidimensional Design by Examples. In *Proc. of 8th Int. Conf. on Data Warehousing and Knowledge Discovery*, volume 4081 of *LNCS*, pages 85–94. Springer, 2006.

[16] T. Berners-Lee and J. Hendler and O. Lassila. The Semantic Web. *Scientific American*, 284(5), 2001.

[17] B. Vrdoljak, M. Banek, and S. Rizzi. Designing Web Warehouses from XML Schemas. In *Proc. of 5th Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK 2003)*, volume 2737 of *LNCS*, pages 89–98. Springer, 2003.

[18] R. Winter and B. Strauch. A Method for Demand-Driven Information Requirements Analysis in DW Projects. In *Proc. of 36th Annual Hawaii Int. Conf. on System Sciences*, pages 231–239. IEEE, 2003.