A Prototyped NL-Based Approach for the Design of Multidimensional Data Warehouse

Abeer Alzahrani, Mohamed Alqarni, Jamel Feki
College of Computer Science & Engineering
University of Jeddah
Jeddah, Saudi Arabia
Email: {Aalzahram2341.stu, alqami, feki}@uj.edu.sa

Abstract—Organizations are more and more interested in the Data Warehouse (DW) technology and data analytics to base their decision-making processes on scientific arguments instead of intuition. Despite the efforts invested, the DW design issue remains a great challenging research domain. The design quality of the DW depends on several aspects, as the requirement gathering. In this context, we propose a Natural Language (NL) based design approach, which is twofold; first, it facilitates the involvement of the decision-makers in the DW design process; indeed, NL can encourage the decision-makers to express their requirements as English-like sentences conform to NL-templates. Secondly, our approach aims to generate semi-automatically a DW schema from a set of requirements gathered as analytical queries compliant to the NL-templates. This design approach relies on (i) two easy-to-use NL-templates to specifying the analysis components, and (ii) a set of five heuristic rules for extracting the multidimensional concepts from the requirements. We demonstrate the feasibility of our approach by developing the prototype Natural Language Decisional Requirements to DW Schema (NLDR2DWS).

Keywords—Data Warehouse, Multidimensional schema, NL-templates, Decisional requirements.

I. INTRODUCTION

Data is essential for organizations; it is the secret of the success as the well-founded decisions rely on the effective analysis of data rather than intuition. Decisional data is often organized as a Data Warehouse (DW) which is the central component of modern decisional systems of organizations. DW has become really a promising technology for the managers. In this context, merging, collecting, organizing and synthesizing data is crucial for the DW design process [1]. Although several researchers have been addressing the DW design issues such as the design approaches and software tools [2], elicitation of user requirements, as well as the effective design of the decisional system, these issues still need more investigations [3] and are at the heart of the DW design and modeling concerns [2]. In other words, decisional requirements merit to be defined precisely and clearly [4].

In this context, this research aims to help DW designers elaborating the DW model relying on decisional requirements. More accurately, it proposes a Natural Language (NL) NL-template based design approach, which is twofold; first, it facilitates the involvement of decision-makers in the early step of the DW design by using NL as a natural means to encourage them to specify their requirements as query-like English sentences. Secondly, the approach aims to help the generation of a DW schema from gathered requirements.

For the requirements specification, we propose two NL-templates. Regarding the semi-automatic generation of the DW schema, we define five extraction rules for identifying the multidimensional concepts from requirements compliant to our NL-templates. Finally, as the terms –i.e., words- in the user's requirements are susceptible to linguistic issues such as ambiguity, we define a cleaning process and then apply it on the cleaned concepts to build the DW model. In fact, we have elected templates as they can guide the requirements specification by avoiding/reducing issues due to different structures in requirements formulations and, we have privileged the NL because it is close to end-users.

This paper is organized into six sections. Section 2 introduces the general context of this research. Section 3 gives an overview of the DW design approaches, completed with a discussion of the related works. In Section 4, we briefly describe our proposed approach for generating a DW schema from requirements written according to NL-templates. Section 5 discusses the foundation of the suggested templates, and defines the proposed NL-templates. Furthermore, we set five extraction rules and illustrate with a meaningful example. Section 6 presents our NLDR2DWS prototype and evaluates it. Finally, Section 7 concludes the paper.

II. GENERAL CONTEXT AND BASIC CONCEPTS

As we are interested in developing a DW approach based on the decision-makers requirements and on using NL, we will give an overview of some recent research works. In the literature, there are two main categories of DW design approaches namely
Bottom-up and Top-down; a third Hybrid approach has stemmed from the combination of them.

Before introducing these approaches, let us remember that a DW schema is designed according to the multidimensional model [3] built around two main concepts: Fact and Dimension. The fact concept models the subject to analyze (i.e., business activity); it is composed of attributes called measures/indicators. As an example, in the Commercial domain, the Sale and Supply are two facts. The Sale fact may have the measures Quantity-Sold, Amount-of-Sale, Unit-Price. They are fundamental to analyze the business activity (e.g., sum-up the Amount-of-Sale) and predict the future sales. Such analyses perform according to Dimensions like the Product, Time, and Customer. In DWs' sing, a dimension models an axis for recording and analyzing the fact measures that are at the intersection of all dimensions. In other words, each measure is functionally dependent from all the dimensions of its fact. Each dimension has attributes organized semantically into hierarchy (ies); each attribute at a given level in the hierarchy is called Parameter. For instance, the Time dimension has the parameters Day, Month, and Year; we organize them semantically into the following hierarchy called H_Time: Day → Month → Year, where the arrow (→) denotes a functional dependency (One-to-One relationship and One-to-Many in the reverse direction), we read each Day belongs to one Month that belongs to one Year. Figure 2 exemplifies a star schema that illustrates the multidimensional concepts.

The Bottom-up DW design approach starts by studying the data model of the Data Source (DS) intended to load the DW; it classifies the components of the DS data-model (generally a relational database) into entities and relationships using a reverse engineering technique. This classification helps to elaborate the DW multidimensional model because, in the literature, the entities serve to build the dimensions whereas the relationships build the facts. This approach was initially suggested and widely used in practice by Ralph Kimball [3] as well as in several research works [4][5][6].

The Top-down approach is originally due to Bill Inmon [7]; it starts from the decision-makers requirements from where it identifies the facts, and then for each fact its dimensions and parameters. The result is a DW schema.

Actually, neither the first approach nor the second produce a completely convincing DW schema; indeed, a Bottom-up approach produces a DW schema closely related to the data model of the DS, i.e., a large DW schema that may have much more facts/dimensions than the decision-makers need.

Inversely, Top-down approaches may produce a DW schema closely related to the users' requirements; it may be incomplete when the requirements are not exhaustive or are ambiguous, or need data not existing in the DS. The third category of approaches is a compromise that aims to benefit from the advantages of top-down and bottom-up approaches while avoiding the shortcomings of each one [8][9][10][11].

Even this hybrid approach has cons; indeed, it requires from the DW designer skills in the design of the operational systems for understanding the DS data-model, along with skills in gathering the requirements of the future DW users. How to collect requirements? What format of specification? Is it free NL or template-guided sentences? How to solve semantic ambiguities due to natural language? ...

In this DW design context, and in an attempt to bypass some of the above problems, we have elected a Top-down NL-based approach for the specification of the decision-makers requirements; more accurately, this specification will be driven by NL-templates defined in accordance with the common format of decisional needs known as On-Line Analytical Processing (OLAP) requirements. Using NL-templates has many benefits; it facilitates the decision-makers involvement in the DW design process; in addition, it encourages them to express their requirements as English-like sentences. In the next section, we review the pertinent recent works related to the context of our proposal.

III. RELATED WORK

This section reviews some recent and pertinent papers related to top-down DW design approaches.

In [12], the authors tried to simplify the complex task of DW design; they suggest the Star Schemas from requirements (SSReq) approach for generating a DW schema from business requirements. They focused on the requirements specification phase neglected in some approaches. They define a NL-based template to allow business users to express their needs as NL-like queries. Their approach relied on three steps: i) Business requirements elicitation; ii) requirements normalization; and iii) generation of Multidimensional schemas. On the one hand, their template has difficulties that face decision-makers when writing complex requirements, mainly when they are not familiar with the DW concepts and OLAP needs; this may lead later to ambiguities in the identification of multidimensional concepts. On the other hand, their requirement normalization step does not solve ambiguities such as synonyms. In fact, we believe the simpler and shorter the template, the better the conceptual results. Furthermore, we should emphasize the pre-processing of requirements to identify synonyms, hyperonyms... and then solve these issues by enabling the DW designer to intervene.

Other authors in [11] have focused on using a decisional ontology to support the decision-makers requirements specification. They present a NL goal-based template to express the requirements and enhance the involvement of the stakeholders. Their approach automates the reasoning about the decision-making knowledge to overcome the lack of domain knowledge ontology and allows systematic requirements elicitation. In an attempt to involve the decision-makers, the
authors in [13] define a NL-based template and a process for requirement validation. They defined three steps to remove any confusion in the NL queries: i) Syntax checking and Part of Speech (PoS) tagging, ii) Mapping and disambiguation, and iii) Generation and verification. The first step extracts the noun phrases from the query to determine the facts and dimensions; for any syntactic nonconformity with the pattern, the user is alerted. The second step identifies PoS of the extracted noun phrase, and then performs the tagging process to solve the PoS ambiguities. Finally, in step 3, a set of matching and expansion rules is defined to determine the multidimensional type using an Extended Data Dictionary (EDD). Note that the use of the NL-template is helpful in the specification and verification phase; however, the EDD is domain-dependent and therefore difficult to elaborate or possess in practice, which limits the usage of the approach.

The requirement-driven approach named DW Requirement Model (DWRM) was proposed in [14] and the authors of the paper NL Why-Question modeling [15] were inspired by DWRM linguistic patterns. Once again, the authors have used a model relying on NL formalism that brings an advantage but inherits semantic ambiguities because of the diversity of writing styles; by using linguistic patterns, they overcome this confusion. The main limitation is their formalism is compatible with the common and frequent request writing style. However, the approach does not deal with the problem of identifying attributes of hierarchies although they are crucial for the DW design.

The approach in [16] generates automatically DW schemas from business keys based on NL. The main limitation is that users' business keys are free syntax, i.e., not conform to templates, which can lead to ambiguities. The main drawback of the software tool developed is its limitation to creating a star schema from users' business keys reduced to two nouns assumed as facts. In the same extension, the authors in [17] adopted an ontology-based hybrid methodology to produce a DW schema and developed a tool for entering the different goals, contexts, and measures identified in the requirement analysis task. The limitation of the approach is decision-makers must be familiar with the multidimensional concepts and DW modeling. TABLE I summarizes these approaches according to a set of criteria we have identified.

Finally, we note the absence of theoretical foundation for the correctness of the suggested patterns/template-based works. Does a decisional requirement need one complex template or simple ones? Does a collection of several simple requirements are equivalent to a complex one? Moreover, do we actually need more than one template? The first contribution of this paper answers these matters.

Based on the related work, we can claim there is still a real need for further investigation in the DW design methodology. More accurately, we tackle two main tasks: i) Requirement gathering and ii) Automatic generation of DW schema.

**TABLE I. TOP-DOWN WORKS COMPARISON**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Involvement of users decision-makers in the design</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Use of Natural Language Pattern</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Use of more than one Pattern</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Use of Simple Patterns</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Use of a semantic resource</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Involvement of decision-makers in the elicitation phase</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Theoretical foundations for NL-templates</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Heuristics/Algorithms for fact construction</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Heuristics/Algorithms for measures identification</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Heuristics/Algorithms for dimensions construction</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Heuristics/Algorithms for dimensional attributes identification</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Heuristics/Algorithms for hierarchy construction</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Automation degree</td>
<td>Semi</td>
<td>Semi</td>
<td>Full</td>
<td>Semi</td>
<td>Full</td>
<td>Full</td>
<td>Semi</td>
<td>Semi</td>
</tr>
</tbody>
</table>
The more precise and well-structured the requirements, the better the quality of the DW schema, and the easier the automatic generation of the schema. We propose, in this paper, a semi-automatic design approach based on NL-templates; the use of NL-templates in conjunction with extraction rules will permit easier and efficient locating/identifying the multidimensional concepts along with the role of each concept in the DW schema. Furthermore, to the best of our knowledge and for the first time in the literature, we will justify the use of simple NL-templates by relying on properties taken from the DW literature and usually used as DW schema-constraints; this distinguishes our work from the existing ones. Besides, we formalize these properties. Our approach defines rules to extract the multidimensional concepts from requirements and automate the rules to derive a DW schema. The following section details our approach.

IV. OVERVIEW OF THE PROPOSED APPROACH

The design of a DW is a complex, difficult and tedious task [18] [19] [20]; it requires skilled persons in design approaches and, in On-Line Transaction Processing (OLTP) and OLAP systems. Therefore, involving the decision-makers reveals a challenge since they completely ignore the DW design approaches. On the other hand, the concept of template has demonstrated its efficacy in many domains; a template refers to a preformatted format for problems specification. We have elected NL-templates to help users expressing their analytical requirements in a readable format; i.e., as natural language sentences; this helps bypassing the difficulties in gathering the requirements and facilitates extracting the multidimensional components [21]. In addition, this involves the decision-makers in the DW design process. Figure 1 depicts our NL template-based approach for the specification of OLAP requirements and generation of multidimensional DW schemas.

This approach has four components hereafter explained.

Requirements Acquisition: for entering OLAP requirements by the decision-maker according to defined NL-Template.

Extraction of Multidimensional Components: extracts, from a collection of requirements, the facts and their measures, the dimensions and their attributes.

Cleaning of Multidimensional Components: cleans the collection of each category of the extracted elements by converting into uppercase, standardizing names, solving synonyms, removing redundancy...

Semi-automatic Construction of the DW Schema.

V. NL-PATTERN FOUNDATIONS AND DEFINITION

As our approach has a twofold objective, first, help the decision-makers expressing their analytical needs, and secondly automate the extraction of the multidimensional components from requirements, we build the structure of the NL-templates around unambiguous keywords (as verbs, functions...) familiar to end-users. Each NL-template component plays a precise role in identifying what the user wants to analyze (facts, measures) and according to what criteria (i.e., dimensions and hierarchies).

The proposed templates are query-like English sentences and allow decision-makers to write a wide range of requirements [12], either as short or Complex analytical queries. Before introducing our NL-templates, we clarify the meaning of Short and Complex queries along with our intuition and the theoretical properties supporting them.

A Complex OLAP-query (C-query for short) is a decisional query that encompasses several multidimensional components (i.e., fact, measures, dimensions, parameters, and conditions) at a time; as query Q1: Analyze the Amount of sales by Client-Country, Client-city, Product-Category and Year of sale. In Q1, the bold terms are multidimensional components; for instance, Amount is a measure for the Sale fact, and Client-Country, Client-city... are parameters (i.e., detail levels of analysis). Although not very complex Q1 is difficult to write by a novice decision-maker.

A short query (S-query) is simple to write by decision-makers even when they are not familiar with the DW concepts and OLAP analysis. In addition, S-queries are very helpful and efficient for the extraction of multidimensional components; moreover, a short query is subject to fewer ambiguities when identifying the role of each of its terms.

Naturally, replacing a C-query by an equivalent collection of S-queries is possible, and inversely. To justify this, we define two novel properties P1 query decomposition and P2 query re-composition. They rely on four constraints (definitions 1 to 4) taken from the DW literature. Let us use the following notation.

Query Q = (F₀, D₀)
Where:
- F₀: the fact in the query Q
- D₀: a non-empty set of dimensions of fact F₀, such as:
  - F₀ = (F₀Name,M₀)
  - F₀Name: the name of the fact F₀
  - M₀ = (m₁₀,...,mₖ₀): a non-empty set of n measures of F₀ and
  - D₀ = (d₁₀,...,dₖ₀): a non-empty set of k dimensions in Q such as
    - dᵢ₀ = (dᵢNameᵢ,Aᵢ₀) ∀ i ∈ {1..k}
- \( A_{Q_1} = \) a non-empty set of attributes of dimension \( d_0 \)

Note that in the generic notation above, we replace the letter \( Q \) with \( C \) or \( S \) to denote a Simple or a Complex query respectively.

P1. Query decomposition. Any Complex query \( C \) can be broken down into an equivalent collection of \( h \) simple-queries \( S_1, S_2, \ldots, S_h \) having the same fact \( F \) as \( C \), without loss of information.

The decomposition of \( C \) into \( S_1, S_2, \ldots, S_h \) is without loss of information if the \( h \) subqueries have the same fact as \( C \) and their measures and dimensions covers all measures and dimensions of \( C \). Formally, if and only if the decomposition respects the following conditions:

\[
\forall i \in [1,h], \quad F_{S_i} = F_c \wedge (M_{S_i})_c \subseteq M_c \wedge (D_{S_i})_c \subseteq D_c
\]

P2. Query composition. Given a collection of \( h \) simple queries on the same fact \( F \), we can use all-or-part of their multidimensional components to write a collection of complex queries on the same fact \( F \) without loss of information, and without respecting necessarily the additivity constraint of measures of \( F \).

Note that the composition must satisfy the same conditions as the decomposition, but in the reverse direction. Accordingly, the decomposition of the \( C \)-query \( Q_1 \) (above) is equivalent to the following four simple queries on the same fact \( \text{sales} \) as \( Q_1 \):

\begin{align*}
S_1 & : \text{Analyze the Amount of sales} \text{by Client-Country.} \\
S_2 & : \text{Analyze the Amount of sales by Client-City.} \\
S_3 & : \text{Analyze the Amount of sales by Product-Category.} \\
S_4 & : \text{Analyze the Amount of sales by Year of sale.}
\end{align*}

In this decomposition, each Simple query \( S_i \) uses one dimension; this shortens writing the requirements by users. Note if \( Q_1 \) has several measures each measure can be alone or combined with other measures in each \( S_i \).

Splitting a complex query \( Q \) into an equivalent collection \( S_1, S_2, \ldots, S_h \) of \( h \) short queries will facilitate the expression of requirements without assistance of IT persons. Therefore, this motivated us to define a first NL-template (cf., syntax T1).

Inversely, the equivalence \( C = S_1, S_2, \ldots, S_h \) in property P2 states that we can re-compose a \( C \)-query \( C \) from its simple sub-queries \( S_1, S_2, \ldots, S_h \) since all components in \( C \) are also in the sub-queries. This is important for the design; it means that the design starting from \( C \) or from \( S_1, S_2, \ldots, S_h \) builds the same star schema.

We base these properties on definitions from [22] [23][24] initially used as DW schema constraints. We define them hereafter.

---

1. An attribute \( b \) is said to be functionally dependent on attribute \( a \) (\( a \neq b \)) if and only if for each value of \( a \) it corresponds only one value of \( b \) at any time (\( b \) is not necessarily the same in time). For example, each \( \text{Client_Id} \) is associated with only one \( \text{Client_Name} \).

www.ijcit.com

116
for specifying facts, measures, dimensions and dimensional attributes.

**Simple NL-Template**

$$\text{OLAP-Verb Analysis-process}\quad \text{By D-name } [\{a_1 < \ldots < a_n\}]$$

(T1)

**Complex NL-Template**

$$\text{OLAP-Verb } [S\text{-function } \{\text{measure}\}] \text{ measure}\quad \text{of Analysis-process}\quad \text{By D-name } [\{a_1 < \ldots < a_n\}]$$

{where | when} condition

(T2)

In these templates,

- **OLAP-Verb**: is a verb that decision-makers use in OLAP-requirements specification; e.g., Analyze, Examine
- **S-function**: is a statistical function (e.g., Min, Max, Average, Count) to aggregate the numeric measures and help in their identification.
- **Analysis-process**: is the subject (i.e., the fact representing the activity) to analyze.
- **By**: reveals the presence of a dimension name.
- **D-name** [\(\{a_1 < \ldots < a_n\}\)]: is a dimension name followed by an optional list of its attributes, preferably ordered semantically from the lowest to the highest attribute (e.g., 
  Prod-ID < Sub-Categ < Category).
- **Condition**: is a condition on the dimensional-attributes 
  \(\{a_1, \ldots, a_n\}\) specified after the D-name in the same query. It can use the logical operators as well as the comparison operators.
- \(\{\}, \{\} \text{ and } |\) denote respectively an optional part, mandatory part, or an alternative (OR).

Note that the statistical functions are optional.

**TABLE II. EXAMPLES OF SIMPLE REQUIREMENTS**

<table>
<thead>
<tr>
<th>Query#</th>
<th>Simple Queries (SQ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQ1</td>
<td>Analyze Sales by Product</td>
</tr>
<tr>
<td>SQ2</td>
<td>Analyze Sales by Time</td>
</tr>
<tr>
<td>SQ3</td>
<td>Analyze Sales by Client</td>
</tr>
</tbody>
</table>

**TABLE III. EXAMPLES OF SIMPLIFIED LONG REQUIREMENTS**

<table>
<thead>
<tr>
<th>Query#</th>
<th>Examples of Simplified Long Queries (LQ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LQ1</td>
<td>Analyze Sales by Client where Id &gt; 123 and &lt; 386</td>
</tr>
<tr>
<td>LQ2</td>
<td>Examine Total Amount of Sales by Client when City = &quot;Jeddah&quot;</td>
</tr>
<tr>
<td>LQ3</td>
<td>Analyze Sales by Client where Country = &quot;USA&quot;</td>
</tr>
<tr>
<td>LQ4</td>
<td>Analyze Amount of Sales by Time when Year = 2016 or Year = 2017</td>
</tr>
<tr>
<td>LQ5</td>
<td>Analyze Sales by Time when Month-no = 2</td>
</tr>
<tr>
<td>LQ6</td>
<td>Analyze Sales by Time when Month-name = &quot;APRIL&quot;</td>
</tr>
<tr>
<td>LQ7</td>
<td>Analyze Sales by Product where Id = 22</td>
</tr>
<tr>
<td>LQ8</td>
<td>Study Sales by Product where Color = &quot;Green&quot;</td>
</tr>
<tr>
<td>LQ9</td>
<td>Analyze Sales by Product where Category = &quot;Toy&quot;</td>
</tr>
<tr>
<td>LQ10</td>
<td>Analyze Sales by Product where Name = &quot;Pram&quot;</td>
</tr>
<tr>
<td>LQ11</td>
<td>Analyze Sales by Product where Subcategory = &quot;Boy toys&quot;</td>
</tr>
</tbody>
</table>

Figure 2 shows the schema we construct using the components extracted from the requirements in TABLES II and III. We have assumed the DW designer has organized the dimensional attributes into hierarchies based on his knowledge of the DW business-domain. Next, we define the rules to identify the multidimensional elements from requirements.

**A. Rules for the generation of the DW schema from requirements**

We define extraction rules for identifying the DW schema components (facts, measures, as well as dimensions and their attributes) from the requirements [29]. We adopt the following notation:

- **Simple-requirement**: stands for a requirement written according to the simple NL-template T1.
- **S Req**: a collection of Simple-requirements.
- **L Req**: a collection of Long-requirements.
1) Facts Construction

A fact is a focus of interest in the decision-making analysis process [4]. Facts construction is the process of finding out the facts from requirements, we conduct it through three phases i) Facts Extraction, ii) Cleaning, and iii) Facts Setting.

a) Facts Extraction. This phase extracts facts firstly from \( S_{req} \) to build a first collection \( F_3 \) of potential facts, and secondly from \( L_{req} \) to build a second collection \( F_4 \) of facts. We define two rules \( FR1 \) and \( FR2 \) to apply on \( S_{req} \) and \( L_{req} \) respectively.

**FR1**: In a Simple-requirement, any noun located after the OLAP-verb is a candidate fact; we insert it into \( F_3 \).

By applying the rule \( FR1 \) to \( S_{req} \) in TABLE IV, we obtain the redundant collection of facts \( F_3 = \{ \text{DIST} ^ {\text{iby}} \text{Item}, \text{DISTRIBUT} ^ {\text{by}} \text{items}, \text{DISTRIBUTION}, \text{Distributions}, \text{RETURNED} _ {\text{item}}, \text{RETURNED} _ {\text{ITEM}}, \text{Returned} _ {\text{Items}}, \text{Returned} _ {\text{products}}, \text{ordered} _ {\text{items}} \} \).

<table>
<thead>
<tr>
<th>Query #</th>
<th>Simple Requirements (( S_{req} ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQ1</td>
<td><strong>Study</strong> DISTRIBUTION by Item</td>
</tr>
<tr>
<td>SQ2</td>
<td>Analyze distributions by time</td>
</tr>
<tr>
<td>SQ3</td>
<td>Study DISTRIBUTION by items</td>
</tr>
<tr>
<td>SQ4</td>
<td>Evaluate Distributions by retailers</td>
</tr>
<tr>
<td>SQ5</td>
<td>Examine Distributions by Retailer</td>
</tr>
<tr>
<td>SQ6</td>
<td>Analyze RETURNED_item by TIME</td>
</tr>
<tr>
<td>SQ7</td>
<td>Examine Returned_item by items</td>
</tr>
<tr>
<td>SQ8</td>
<td>Analyze RETURNED_ITEM by Items</td>
</tr>
<tr>
<td>SQ9</td>
<td>Evaluate Returned_item by RETAILERS</td>
</tr>
<tr>
<td>SQ10</td>
<td>Analyze Returned_items by items</td>
</tr>
<tr>
<td>SQ11</td>
<td>Evaluate Returned_items by Retailer</td>
</tr>
<tr>
<td>SQ12</td>
<td>Analyze Returned_products by Retailer</td>
</tr>
<tr>
<td>SQ13</td>
<td>Analyze Returned_products by item</td>
</tr>
<tr>
<td>SQ14</td>
<td>Analyze ordered_items by product</td>
</tr>
</tbody>
</table>

We continue the facts extraction from the \( L_{req} \) using rule \( FR2 \).

**FR2**: In a Long-requirement, any noun located immediately before the keyword By is a candidate fact; we insert it into the \( F_4 \).

By applying the rule \( FR2 \) to \( L_{req} \) in TABLE V, we obtain \( F_4 = \{ \text{Distributions}, \text{Distribution}, \text{distribution}, \text{distributions}, \text{DISTRIBUTION}, \text{Returned} _ {\text{item}}, \text{Returned} _ {\text{ITEM}}, \text{Ret} \text{urned} _ {\text{item}}, \text{RETURNED} _ {\text{ITEM}}, \text{manufacturing} \} \).

<table>
<thead>
<tr>
<th>Query#</th>
<th>Long Requirements (( L_{req} ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>LQ1</td>
<td>Analyze Max Quantity of Distributions by time when sale_period = &quot;end of the year&quot; and week = 3</td>
</tr>
<tr>
<td>LQ2</td>
<td>Study amounts of DISTRIBUTION by time when Promotion_period = &quot;summer&quot; and day = 6</td>
</tr>
<tr>
<td>LQ3</td>
<td>Evaluate AVERAGE unit_price of distribution by time when month = 11</td>
</tr>
<tr>
<td>LQ4</td>
<td>Study Amounts of distribution by time when month = &quot;June&quot; and sale_period = &quot;new year&quot;</td>
</tr>
<tr>
<td>LQ5</td>
<td>Analyze SUM QUANTITY of distribution by time when quarter = &quot;Third&quot; and years = 2013</td>
</tr>
<tr>
<td>LQ6</td>
<td>Analyze Unit_Price of distributions by time when semester = &quot;first&quot;</td>
</tr>
<tr>
<td>LQ7</td>
<td>Analyze Max Qty of Distributions by Time when quarter = 3 and day = 29</td>
</tr>
<tr>
<td>LQ8</td>
<td>Study Quantity of Distribution by item when subcategory = &quot;Phones&quot; and name = &quot;Samsung&quot;</td>
</tr>
<tr>
<td>LQ9</td>
<td>Study total dist_amount of DISTRIBUTION by item when category = &quot;Electronics&quot;</td>
</tr>
<tr>
<td>LQ10</td>
<td>Analyze Unit_price of Distribution by item when subcategory = &quot;kitchen appliance&quot; and origin = &quot;USA&quot;</td>
</tr>
<tr>
<td>LQ11</td>
<td>Examine Unit_price of distribution by item when category = &quot;Appliance&quot;</td>
</tr>
<tr>
<td>LQ12</td>
<td>Study distribution by RETAILER when City = &quot;Jeddah&quot;</td>
</tr>
<tr>
<td>LQ13</td>
<td>Examine MIN unit_price of Distribution by retailers when region = &quot;west&quot; and CITY = &quot;Jeddah&quot;</td>
</tr>
<tr>
<td>LQ14</td>
<td>Study TOTAL Dist_amounts of Distribution by Retailer when NAME = &quot;extra&quot; or city = &quot;Riyadh&quot;</td>
</tr>
<tr>
<td>LQ15</td>
<td>Examine MIN unit_price of Distribution by retailers when Region = &quot;North&quot; or name = &quot;extra&quot;</td>
</tr>
<tr>
<td>LQ16</td>
<td>Analyze Max quantities of Returned_item by Time when Sale_P eriod = &quot;New year&quot; and year = 2019</td>
</tr>
<tr>
<td>LQ17</td>
<td>Study total amounts of Returned_item by Time when Promotion_period = &quot;Summer&quot; and month = 7 and week = 4</td>
</tr>
<tr>
<td>LQ18</td>
<td>Analyze RETURNED_qty of Returned_items by time when week = 3</td>
</tr>
<tr>
<td>LQ19</td>
<td>Examine Amounts of returned_item by time when day = 6</td>
</tr>
<tr>
<td>LQ20</td>
<td>Evaluate MIN unit_price of returned_item by time when month = 11</td>
</tr>
<tr>
<td>LQ21</td>
<td>Evaluate MAX unit_price of returned_item by time when month = &quot;June&quot;</td>
</tr>
<tr>
<td>LQ22</td>
<td>Study SUM Quantity of _returned_item by time when quarter = &quot;fourth&quot; and month = &quot;December&quot;</td>
</tr>
<tr>
<td>LQ23</td>
<td>Analyze Unit_price of _returned_item by time when semester = &quot;second&quot;</td>
</tr>
<tr>
<td>LQ24</td>
<td>Analyze MAX AMOUNTS of _returned_item by time when year = 2019</td>
</tr>
<tr>
<td>LQ25</td>
<td>Analyze Quantities of _returned_items by item when Name = &quot;Extra&quot; and subcategory = &quot;laptops&quot;</td>
</tr>
<tr>
<td>LQ26</td>
<td>Study total amount of Returned_items by item when category = &quot;Electronics&quot;</td>
</tr>
<tr>
<td>LQ27</td>
<td>Examine Returned_qty of _returned_items by item when subcategory = &quot;iPad&quot; or origin = &quot;USA&quot;</td>
</tr>
<tr>
<td>LQ28</td>
<td>Study amount of Returned_items by Retailer where city = &quot;JEDDAH&quot;</td>
</tr>
<tr>
<td>LQ29</td>
<td>Examine Average unit_price of Returned_item by retailers where region = &quot;North&quot;</td>
</tr>
<tr>
<td>LQ30</td>
<td>Analyze RETURNED_QTY of RETURNED_ITEM by retailer where NAME = &quot;eddy&quot; or ZIP = 6667</td>
</tr>
<tr>
<td>LQ31</td>
<td>Study total Amounts of manufacturing by retailer where city = &quot;Damman&quot;</td>
</tr>
</tbody>
</table>
After this extraction, we continue building the facts by applying the Cleaning phase of our approach.

b) Cleaning

Note that the collections $F_S$ and $F_L$ obtained so far may overlap, have synonyms, or uncommon elements. This Cleaning phase solves the issues for which we develop a four-step Cleaning method applicable to facts as well as measures and dimensions. It deals with redundancy, synonyms and antonyms. These steps apply in the following order:

i. Convert into capital all elements in $F_S$ and $F_L$. This is to avoid the case-sensitivity problem in comparisons.

ii. Replace with singular each element in $F_S$ and $F_L$.

iii. Find synonyms if any, by using WordNet as an open-source semantic resource. We highlight the most frequent synonym encountered (as a default to keep) to the DW designer and allow him to select which synonym is better appropriate for the business domain of the DW under construction.

iv. Eliminate the redundancy in each collection to obtain two cleaned sets noted $F_{SC}$ and $F_{LC}$.

v. Purge the sets $F_{SC}$ and $F_{LC}$. Each element in $F_{LC} \setminus (F_{SC} \cap F_{LC})$ must be either removed from $F_{LC}$ or moved to $F_{SC}$ if it is not recognized as a synonym for an element in $F_{SC}$.

In step v), we can consider $F_S$ (and then $F_{SC}$) as a reference collection of facts so that only the facts in $F_S$ will be acceptable during the entry steps of the Long-requirements.

For instance, we clean the fact collections $F_S$ and $F_L$, previously extracted, through the above steps as follows:

i. Convert into capital all elements gives

$$F_S = \{\text{DISTRIBUTION, DISTRIBUTIONS, DISTRIBUTION, DISTRIBUTIONS, RETURNED\_ITEM, RETURNED\_ITEM, RETURNED\_ITEM, RETURNED\_ITEM, RETURNED\_ITEM, RETURNED\_ITEM, RETURNED\_PRODUCT, ORDERED\_ITEM}\},$$

and

$$F_L = \{\text{DISTRIBUTION, DISTRIBUTION, DISTRIBUTION, DISTRIBUTION, RETURNED\_ITEM, RETURNED\_ITEM, RETURNED\_PRODUCT, ORDERED\_ITEM, MANUFACTURING}\}.$$ 

ii. Replace with singular produces

$$F_S = \{\text{DISTRIBUTION, DISTRIBUTION, DISTRIBUTION, DISTRIBUTION, RETURNED\_ITEM, RETURNED\_ITEM, RETURNED\_PRODUCT, ORDERED\_ITEM}\},$$

and

$$F_L = \{\text{DISTRIBUTION, DISTRIBUTION, DISTRIBUTION, DISTRIBUTION, RETURNED\_ITEM, RETURNED\_ITEM, RETURNED\_PRODUCT, ORDERED\_ITEM, MANUFACTURING}\}.$$ 

iii. Find synonyms

In $F_S$ RETURNED\_ITEM and RETURNED\_PRODUCT are synonymous, we elect RETURNED\_ITEM.

iv. Eliminate redundancy in each collection

$$F_{SC} = \{\text{DISTRIBUTION, RETURNED\_ITEM, ORDERED\_ITEM}\},$$

and

$$F_{LC} = \{\text{DISTRIBUTION, RETURNED\_ITEM, MANUFACTURING}\}.$$ 

v. Purge the sets $F_{SC}$ and $F_{LC}$

As we note there is an element (MANUFACTURING) in $F_{LC}$ not in $F_{SC}$ we warn the user to add simple and may be Long queries for this fact or remove it. In the next that follows, we assume the designer has dropped the fact; therefore, the result is:

$$F_{SC} = \{\text{DISTRIBUTION, RETURNED\_ITEM, ORDERED\_ITEM}\},$$

and

$$F_{LC} = \{\text{DISTRIBUTION, RETURNED\_ITEM}\}.$$ 

After these two phases, we end with the Facts Setting phase.

c) Facts Setting

We compare the two cleaned collections of facts $F_{SC}$ and $F_{LC}$ to build a Final set of facts $F_{Final}$. The comparison of two sets leads to consider at least two cases: $F_{SC} \cap F_{LC} \neq \emptyset$ or when $F_{SC} \cap F_{LC} = \emptyset$. However, the cleaning step has simplified the problem so that we have now only the following situations: i) $F_{LC} \subseteq F_{SC}$ this means that some facts accepted in the Short-requirements are unused within the Long-requirements. Therefore, we warn the DW designer with the unused fact(s). In the second situation ii) $F_{LC} \not\subseteq F_{SC}$ all facts are common; we accept them all. For example, the fact ORDERED\_ITEM in $F_{SC}$ is not common with $F_{LC}$ (i.e., $F_{LC} \not\subseteq F_{SC}$), we warn the DW designer with this vacant fact. Assume the designer abandons this fact, the final set of facts is $F_{Final} = \{\text{DISTRIBUTION, RETURNED\_ITEM}\}$. Next, we will complete our approach with the identification of measures.

2) Measures Identification

It aims to find attributes [4] optionally preceded by an aggregation function in the Long-requirements. We define the following rule MR for the extraction of measures.

MR: Any noun or sequence of nouns, in a requirement $l \in L_{Req}$ located after an aggregate function and/or before “of” is a candidate measure for the fact extracted from $l$ using rule FR2.

By applying the rule MR on queries in TABLE V, we obtain the measures in TABLE VI. After we have identified the measures, we may encounter the same problems as the facts; hence, we apply the same Cleaning steps as for the facts.

Cleaning of Measures. The final set of measures, obtained for each fact, after capitalizing, replacing with singular, solving synonyms, and eliminating the redundancy is:

\[
\text{DISTRIBUTION measures} = \{\text{QUANTITY, AMOUNT, UNIT\_PRICE}\} \quad \text{and} \\
\text{RETURNED\_ITEM measures} = \{\text{QUANTITY, AMOUNT, UNIT\_PRICE}\}.
\]

3) Dimensions Determination

Dimensions are composed of attributes called parameters (i.e., Analysis levels) according to which we aggregate the measures of the fact. The determination of dimensions is driven by the by keyword. For this purpose, we define the rule DR.

DR: In a Long-requirement $l \in L_{Req}$, any noun located after the by keyword would be a candidate dimension for the fact extracted from $l$ using rule FR2.
The application of the rule \( DR \) on queries in TABLE V gives the three dimensions depicted in TABLE VI. Once again, we purify the collection of dimensions by applying the same Cleaning steps; we obtain the following final cleaned sets of dimensions:

DISTRIBUTION dimensions = \{TIME, RETAILER, ITEM\} and RETURNED_ITEM dimensions = \{TIME, RETAILER, ITEM\}

We continue to extract for each dimension its attributes useful for the construction of hierarchies.

4) Extraction of dimensional attributes and Hierarchies construction

Hierarchy construction builds hierarchies of dimensions. The semantics is a key issue for ordering the attributes into hierarchies since this semantics is Business-domain dependent, and therefore requires Human skills.

a) Dimensional attributes extraction.

Dimensional attributes come from the Long requirements, they are preceded by Where or When. We define the rule \( HR \) and then illustrate it on our running example.

\( HR: \) In a Long requirement \( L_{req} \), a noun located after the keyword Where or When is a candidate dimensional attribute, for the dimension extracted from \( L_{req} \) using rule \( DR \).

Applying \( HR \) on the set \( L_{req} \) in TABLE V, we extract the dimensional attributes depicted in TABLE VI where we have conventionally named a dimensional attribute as the concatenation of its dimension with the underscore ("_") and the attribute name extracted from requirements.

Following our approach, we perform the same Cleaning steps as for the facts, and we obtain the results below:

TIME dimension attributes = \{SALE_PERIOD, QUARTER, YEAR, ROMOTION_PERIOD, DAY, WEEK, MONTH, SEMESTER\}
ITEM dimension attributes = \{NAME, ORIGION, CATEGORY, SUBCATEGORY\}
RETAILER attributes \{NAME, ZIP, CITY, REGION\}

Since our approach is semi-automatic, it asks the DW designer to classify manually the extracted attributes into parameters and weak attributes, associate the parameters with weak attributes then organize them into hierarchies. We delegate the semantic organization to the DW designer relying on his knowledge of the Business-domain of the DW.

In addition, for each dimension, we generate an Identifier (as a surrogate key) when no Id is encountered. Note that for the TIME dimension the DW designer manually renamed the MONTH attribute to MONTH_NAME and added the new attribute MONTH_NAME.

In our running example, we have parameters and weak attributes the DW designer uses to construct the hierarchies:
• Four hierarchies for the TIME dimension:
  
  TIME_H1: TIME_ID < TIME_WEEK
  TIME_H2: TIME_ID < TIME_PROMOTION_PERIOD
  TIME_H3: TIME_ID < TIME_SALE_PERIOD
  TIME_H4: TIME_ID < TIME_DAY < TIME_MONTHNO
            < TIME_MONTH_NAME < TIME_QUARTER < TIME_SEMESTER < TIME_YEAR

• One hierarchy for the ITEM dimension:
  
  ITEM_H1: ITEM_ID (ITEM_NAME, ITEM_ORIGIN) < ITEM_SUBCATEGORY < ITEM_CATEGORY

• One hierarchy for the RETAILER dimension:
  
  RETA_H1: RETAILER_ID (RETAILER_NAME) < RETAILER_ZIP < RETAILER_CITY < RETAILER_REGION

Finally, we obtain two facts (DISTRIBUTION and RETURNED_ITEM) having three common dimensions (TIME, ITEM, and RETAILER); this is typically a Constellation schema as depicted in Figure 3. A constellation has multiple facts sharing common dimensions [30]. The obtained DW schema is able to answer complex queries as “Total Amount and Quantity (measures) returned by City (parameter) of RETAILER (dimension) and by ITEMS (dimension) from a given Category (parameter) of items during the third Quarter of the Year 2019 (parameters of the TIME dimension).

A. Software Environment

Since the NLDR2DWS future users are decision-makers not familiar with the DW technology, we made sure that the software is simple for use by non-IT persons. We have opted for Python as an environment that includes libraries for the design of graphical interfaces, and as a means to access a linguistic resource. Mainly, we have used three libraries:

• wxPython facilitates the creation of robust and greatly functional graphical user interface programs [31].
• Natural Language Tool Kit (NLTK): A comprehensive efficient tool in NL Processing domain [32] to access and explore lexical resources such as WordNet [33] that we have used to find out semantic relationships among concepts [32]: Synonymy (as customer and client) and Antonym (when two words have opposite meanings as Sell and Purchase). This improves the consistency of the design result. Indeed, we warn the user with these situations and we ask him to rectify, if necessary.
• SQLite Database Library: An open source code to bind with Python [34].

B. NLDR2DWS Presentation

We built our framework components in two main complementary interfaces: Simple NL-Interface and Long NL-Interface. The user starts with the Simple NL-interface (cf. Figure 4) to enter facts and dimensions and then validate them. The second interface (cf. Figure 5) is for entering complementary components for the validated components. This stepwise method is an incremental process for entering requirements.

Figure 4. Interface for Entering Requirements using the Simple NL-Template

VI. THE NLDR2DWS PROTOTYPE

To prove the feasibility of our DW design approach and evaluate it, we have implemented a software prototype called NLDR2DWS (NL Decisional Requirements to DW Schema) that supports it. It produces a DW schema from entered requirements as per the defined templates. We have built a benchmark of 30 queries in the Supply Chain Management business-domain and tested it. NLDR2DWS checks some constraints on the obtained DW schema to verify i) the Non-Isolated-fact constraint that guarantees every fact must be linked to two dimensions at least, and ii) the presence of the Minimal hierarchy in each dimension.

Figure 3. Constellation Schema Built from Requirements in Tables IV and V

www.ijcit.com
Once we have extracted the schema components, we can visualize them as a list or in a tabular format (cf. Figure 6). After that, the DW designer can edit the schema components manually: he can rename, remove and/or rarely add new measures or weak attributes (cf. Figure 7).

C. Evaluation

The results show that NLDR2DWS is able to extract the multidimensional concepts from the requirements and alert the user with the synonym and antonym ambiguities that could be solved by the DW designer relying on his expertise of the DW business-domain. However, in certain circumstances, some issues remain hard to identify; this is mainly with composed words that require the implementation of additional features. We identify the following:

- Synonyms problem. From the business domain viewpoint, some words are synonymous; in such a case, WordNet was unable to detect, for instance, the facts RETURNED_PRODUCT and RETURNED_ITEM should be synonymous in the Supply Chain Management (SCM) domain. In this case, a fine look to the result by the designer is necessary to identify and decide which one is the most appropriate.

- Redundancy/Abbreviation. The use of abbreviations raises ambiguity as in queries LQ1, 5, 7, 8 where QUANTITY and QTY should be equivalent, but the result is two different measures; manual editing is necessary.

- Inclusivity problem. This occurs when decision-makers use the same word differently. As an example, in queries LQ3 and LQ4, MONTH is identified as an attribute for the TIME dimension. However, in LQ3 MONTH designates the MONTH number (as MONTH=11), and in LQ4 it indicates the name (MONTH="June"). NLDR2DWS detects this issue; the DW designer will manually rename...
the MONTH attribute to MONTHNO and add a new attribute MONTH_NAME associated with MONTHNO.

These are semantic difficulties; the DW designer should edit the design result through the interface in Figure 7.

VII. CONCLUSION

We have proposed a semi-automatic design approach for entering analytical requirements according to one Simple template from which we identify the facts, and one Complex for measures, dimensions and analysis levels. We have established the templates relying on two properties: decomposition and re-composition defined on requirements. We have based these properties on constraints issued from the literature of the DW conceptual domain. The approach accepts a collection of OLAP requirements conformed to NL-templates, extracts the multidimensional concepts from these requirements, and then applies three phases i) Fact construction, ii) Measures Identification, and iii) Dimension determination; it generates a DW schema. We have defined five extraction rules. For feasibility, we have developed the prototype NLDR2DWS that implements the different steps of the approach. Actually, we have conducted some experiments on a set of analytical requirements in two different domains: students’ registration deanship and Supply Chain Management. NLDR2DWS produces quasi-automatically a respectable DW schema, and therefore the results are very promising under further extensions. Really, the DW designer should intervene to solve some problems due to synonyms, antonyms, and to complete the design with the build of the dimensional hierarchies representing the levels of analysis for the fact measures.

For the short term, expected extensions deal with improving the design quality of the DW schema by emphasizing additional Schema constraints like the Additivity of measures to guarantee that the fact measures are summarizable according to one dimension at least. This requires enriching the schema with the data type of measures and answer the following question: Do the sum of a measure by each dimension is a meaningful value? In the same direction, we consider to include the Hierarchy constraint for checking that all hierarchies of a dimension D must start from the identifier of D.

For the long-term, we intend to build a domain-semantic resource as a dictionary from the OLTP database tables in order to check whether the facts and dimensions are compliant to the Business domain of the DW; also, it could be used to organize semi-automatically the extracted dimensional attributes into hierarchies. Ultimately, the Model Driven Architecture (MDA) paradigm[35]36 is extremely interesting to automate rapidly the implementation of the DW using the Query/View/Transformation (QVT) Language [37].

REFERENCES


124

International Journal of Computer and Information Technology (ISSN: 2279 – 0764)
Volume 9 – Issue 5, September 2020

warehouse.” 2011.


