Cluster Computing (2019) 22:783–803
https://doi.org/10.1007/s10586-018-2883-7

Designing data cubes in OLAP systems: a decision makers' requirements-based approach

Rahma Djiroun[1] · Kamel Boukhalfa[1] · Zaia Alimazighi[1]

Received: 30 October 2017 / Revised: 28 September 2018 / Accepted: 3 December 2018 / Published online: 13 December 2018

Abstract
Business Intelligence systems rely on an integrated, consistent, and certified information repository called the Data Warehouse (DW) that is periodically fed with operational data. In the decision-making process, the analyzed data are usually stored in the DW in the form of multidimensional cubes. These cubes are queried interactively by the decision makers, according to the online analytical processing paradigm. In larger companies with multiple subsidiaries, the frequent expression of new business needs requires the creation of new data cubes which generate a large number of cubes to be manipulated. The inevitable complexity and heterogeneity of data cubes make it difficult to design data cubes. The decision maker can precisely express his needs through a query in natural language which consists of a set of analysis indicators (measures, dimensions) separated by the *AND* operator. However, the decision maker's need may be incomplete. Indeed, he usually has a cube that represents part of his needs and he may want to complete it or enrich it with other cubes that are unknown to him. To deal with these situations, we propose in this paper an approach that addresses the problem of designing and constructing data cubes where the expressed need is scattered over more than one cube. Our goal is to enable decision makers to analyze all of their needs using just one cube. Our approach consists of two variants: a variant that is based on analysis indicators, and another based on the known cube. We present the validation of our approach by means of a tool, called "**Design-Cubes-Query**" that implements our approach and we show its use through a case study.

Keywords Business Intelligence · Cube design · OLAP · Multidimensional data cubes · Fusion · Drill-Across

1 Introduction

Business Intelligence (BI) is an area which is concerned with the development of methodologies, applications, and tools to collect data from internal systems and external sources, store them for analysis, and provide access to information so as to enable more effective strategic, tactical, and operational insights and decision-making.

With the fast development of business and social environments, decisions have to be made quickly, and the selection of an action plan must be based on reliable data, accurate predictions, and evaluations of the potential consequences [2]. In addition, these decisions must be taken in real time to be most effective. BI tools provide an effective solution for multidimensional online computing and analysis of large volumes of data. These data are stored in the Data Warehouse (DW) and materialised on multidimensional data cubes that are interactively queried by decision makers according to the online analytical processing (OLAP) paradigm [13].

OLAP systems allow decision makers to visualise and explore multidimensional data cubes by applying OLAP operators: Slice selects a subset of warehoused data, Roll-Up provides aggregate measures by moving up through the cubes hierarchy, Drill-Down is the opposite of Roll-Up; Drill-Across executes queries involving (i.e., across) more than one same-dimension cubes; etc.

In big companies with several subsidiaries, on the one hand separate data cubes are independently developed to

✉ Kamel Boukhalfa
kboukhalfa@usthb.dz

Rahma Djiroun
rdjiroun@usthb.dz

Zaia Alimazighi
zalimazighi@usthb.dz

1 Laboratory LSI, USTHB, BP 32, El Alia, Algiers 16111, Algeria

answer complex analysis needs. On the other hand, initiatives like Open Data[1] and Open Government[2] are pushing organizations to publish and share multidimensional data in data cube format [9]. However, the decision makers needs may be scattered over several cubes. They may compare different business process measures which are stored in different data cubes. Indeed, the decision maker will have to combine data from heterogeneous data cubes. Searching for the cubes that contain part of the users need in a large collection and merging some of them to construct relevant cubes is a very difficult task.

Generally, the decision maker (user) expresses his need through a set of terms (decision indicators) separated by operators like *AND*, *OR*, etc. In [8], an approach was proposed to find the relevant *Top-K* cubes, that answer a user's query expressed in natural language. However, the user's need can be complex and involve several cubes of several subsidiaries; i.e. the information related to the need is in this case scattered over several cubes. The approach cannot return any cube that entirely answers the user's need, but the search process detects cubes each of which contains part of the expressed need. The exploration of these cubes to analyse a phenomenon is a very tedious task and it is time and effort consuming. Indeed, the decision maker must navigate in the same time between multiple isolated cubes, which makes the global analysis prone to the risk of non-relevant or impossible decisions. The user actually looks for one cube that satisfies all the requirements instead of a set of cubes each of which would only partially satisfy them. Therefore, it is necessary to define a new approach to design a new cube from existing ones.

In BI applications, multidimensional modelling requires specialized design techniques. A lot of works have focused on the design of DWs, but there is no consensus on a design methodology yet [27].

Several approaches have agreed on a phase of conceptual design and one of logical design [12, 16, 18]. Others (e.g.,[12, 28]) also support a physical design phase which addresses all the issues that are specifically related to the set of tools used for implementation. Yet in other approaches, a phase of requirement analysis (e.g. [10]) was separately considered. Other approaches such as [2, 29] have focused on fusion in order to obtain a new DW.

Fusion of data cubes has already been investigated by considering the integration of dimensions and facts using the Drill-Across operator [19]. This operator aims to join data cubes which have common dimensions members. According to [20], a key requirement for crossing several multidimensional data cubes is that they must share dimensions. Shared dimensions must be the same (logical schema and instances). For this, users are often faced with the *non-conformity* problem when they want to combine heterogeneous cubes using non-shared dimensions. Many works have addressed the problem of non-conformity of dimensions to merge predefined cubes. In all these works, the user's need is accurate, i.e. the user knows exactly which cubes to merge and what he wants by the merge. On the other hand, the decision maker is not necessarily an OLAP expert and may not be familiar with OLAP tools. Therefore, it is not easy to exploit cubes via various operators such as *Drill-Across*.

The decision maker may have a clear need, in which case he expresses his requirements by expressing a set of decision indicators (measures, dimensions) that are separated by *AND* logical operator. In this context, we proposed in [25] an approach that designs and constructs new cubes that contain all the needs. The approach returns a cube constructed using user-defined measures and dimensions. The user sometimes has part of his need answered through data contained in a cube but may want to complete it or enrich it with other existing cubes so as to meet the entire need. In this paper, we propose an extension of [25] to consider this case. The present approach recommends, from the cube which is given as input, newly designed cubes that answer the users need. The approach seeks all the cubes that can be merged to construct a new cube by checking the merge conditions (common dimensions and conformities between dimensions). Our approach suggests to the decision maker a set of cubes that can be merged to the input one; by defining the measures and dimensions of each selected cube, the decision maker then selects from this set the cubes whose needs are satisfied. This returns only cubes that can be merged to the input one in such a way as to save time and gain in relevance. This paper is organised as follows. Section 2 presents a motivating example. Section 3 summarises some relevant works. In Sect. 4, we present some preliminary definitions. Section 5 describes our approach and Sect. 6 is dedicated to the implementation and tests. Finally, Sect. 7 concludes the paper and addresses some future works.

2 Motivating example

In this section, we present a case study that will be used in the rest of the paper to illustrate our approach. We present in Table 1 a subset of seven cubes: *Food Production* (C_1), *Agro-industrial Production* (C_2), *Irrigation* (C_3), *Agricultural Production* (C_4), *Pesticide* Cube (C_5), *Dairy Production* Cube (C_6) and *Weather* Cube (C_7). For each cube, we present the following multidimensional concepts: Fact, Measures with their Aggregation functions and Dimensions with their Levels. These cubes have different dimensions

[1] http://okfn.org/opendata/.

[2] http://opengovdata.org/.

Table 1 Pilot farm case study

Cubes	Facts	Measures [aggregates]	Dimensions [levels]
Food production cube (C_1)	Agricultural production for food	Cultivated area, input quantity [total, avg], output quantity [total, avg]	Technical operation, time [year season], product, location [parcel], production, crops
Agro-industrial production cube (C_2)	Agricultural production for industrial	Cultivated, area, input quantity [total, avg], output quantity [total, avg]	Technical operation, time [year, season], product, location [parcel, region], production, crops, transformation factory
Irrigation cube (C_3)	Irrigation	Quantity of water [total], water flux [rate], irrigated area [avg, total]	Time [hour, day, week month, season, year], crops, location [parcel, department, region], equipment
Agro-cultural production cube (C_4)	Crop production	Input quantity [total, avg], output quantity [total, avg], pesticide concentration [percentage, rate], pesticide flow [rate]	Technical operation, crops, pesticide, production, time [day, month, season, year], location [parcel, department, region], product, equipment
Pesticide cube (C_5)	Concentration of pesticide	Flux discharge, pesticide concentration [percentage, rate], pesticide flow [rate]	Crops, time [hour, day, month, year], parcel, equipment, pesticide, crops
Dairy production cube (C_6)	Milk production	Milk quantity produced [total, avg], input quantity [total], output quantity [total, avg]	Time [day, month, year], animal husbandry, product, production, operators
Weather cube (C_7)	Climatology	Pluviometry [total, avg], temperature [total, avg], humidity rate [total, avg], wind speed [total, avg]	Time [hour, day, week, month, season, year], station, localization [parcel, city, department, zone]

and measures, but also some common dimensions such as *production*, *parcel*, etc. Different decision makers can use these cubes (Agricultural Production Managers, Protection and environmental monitoring Experts, Hydraulic Managers and Farm Managers, etc.).

Let us suppose that the Environmental Monitoring Experts and Agricultural Production Managers want to have the crops, which require the less concentration of pesticide and produce the more quantity of product. The decision makers are interested in reducing the concentration of pesticide for agricultural products to reduce environmental risks.

A nave way to answer this query is to retrieve all cubes that analyse *Agricultural production* and all cubes that analyse *pesticide concentration*, then choose the *crops* cubes that have the highest *quantity of output* and the cubes having *crops* that require lower *pesticide concentration*. The decision makers must then manually analyse the two sets of cubes and compare the *crops* that have the greatest *output quantity* and require the least *pesticide concentration*. This manual process becomes increasingly laborious if the number of cubes is large.

The solution to this problem is to return, to the user, the cube that brings both the *quantity of output* (crop yield) and *pesticide concentration*. The idea is to combine these cubes in order to meet both needs simultaneously. This combination is possible by fusing these cubes with the common dimension *crops*. On the other hand, if the decision maker wants to increase or decrease the amount of water used for irrigation per parcel based on the weather, he would need to analyse the *amount of water* used for *irrigation*, *rainfall* and *temperature* for each *parcel* in the same time. However, this need is scattered over two cubes: *Irrigation* (C_3) and *Weather* (C_7). It would be interesting to merge these two cubes in a single one covering the user's need. The merge operation is based on the concept of dimensions' conformity. In this example, the fusion operation is not possible because cubes C_3 and C_7 have no shared dimensions. If we analyse the cubes C_3 (**Irrigation** (*Quantity of water*, *location* [parcel, *department*, *region*])) and C_7 (**Weather** (*Pluviometry* [*total*, *avg*] Temperature [*total*, *avg*] *localization* [parcel, city, department, zone]), we find that they have a common hidden dimension (location and localization). Both dimensions represent the same thing, but they have neither the same name nor the same levels of hierarchy. We show in our approach that it is possible to merge these two cubes after a conformity study of these two dimensions.

3 Related works

Due to the large number of internal and external cubes that are manipulated in a company, the data needed by decision makers may be scattered over several cubes. For example, the decision maker may need to compare different business process indicators that are stored in different data cubes. He should analyse each cube apart and then find a way to compare them. Comparing cubes is a tedious and complicated task. For this, existing solutions such as the Drill-

Across operator allow to merge cubes into one, helping the decision maker to visualise all needed data in a single cube. However, using this operator to merge cubes requires that these cubes have shared dimensions. In this work, we focus on the problem of designing and constructing cubes from a set of existing cubes on the basis of an expressed need, and seeking common dimensions that are not necessarily shared between these cubes.

In the literature, works that have addressed the cube construction problem may be classified in three categories: (1) cube design optimization [15], (2) cube design solutions [4, 5, 11] and (3) cube (attributes and instances) merging solutions (cube enrichment [2] and combining existing cubes [6, 16, 18]).

In order to derive a set of data cubes that answer the users frequent queries, there are two practical problems: the maintenance cost of the data cubes, and the cost of answering those queries. In [15] an approach is proposed to help the user decide which queries would be skipped and not taken into consideration. The authors focused on the optimization problem in data cube system design. Given the maintenance-cost bound, the query-cost bound and the set of frequently asked queries, the proposed system allows the determination of a set of data cubes that can answer the largest subset of the queries without violating the two bounds.

In some domains, such us social networks, bioinformatics, and chemistry, the graphs provide a powerful abstraction for modelling networked data. Ghrab et al. [11] propose a framework for building OLAP cubes from graph data and analysing the graph topological properties. The authors presented techniques for OLAP aggregation of the graph and discussed the case of dimension hierarchies in graphs.

The authors of [15] addressed the problem of integrating independent and possibly heterogeneous DWs. The authors provided a set of properties to solve the problem of matching heterogeneous dimensions. They proposed two approaches to deal with the integration problem. The first refers to different scenarios of a loosely coupled integration to identify the common information between data sources and perform join operations over the original sources. The second approach, which is based on the derivation of a materialized view built by merging the sources, refers to a scenario of tightly coupled integration. Thus, the authors developed a tool called *DaWaII* used to merge data marts developed autonomously by different designers of a telecommunications company.

In [1], the Drill-Across operator allows users to leap from one cube to another. The authors studied different kinds of object-oriented conceptual relationships between facts (namely Derivation, Generalisation, Association, and Flow) in order to Drill-Across them. They defined the Drill-Across operator, using UML relationships between dimensions and/or facts, to navigate between cubes even when no dimensions are shared. In the cube design community, Niemi et al. [23] presented a method to construct OLAP cubes based on the users example queries. From information stored into the DW, the user can pose a sequence of queries in order to construct a cube that contains all the information that is relevant to him. The proposed method makes it possible to improve the structure of existing cubes based on information about the posed queries. In their approach, the authors combined the cube design and query construction by considering a natural connection between OLAP cubes and queries.

In [2] a framework was presented to support cubes fusion in *self-service* BI so as to enrich the decision process with data that has a narrow focus on a specific business problem and a short lifespan. The fusion addresses the multidimensional cubes that can be dynamically extended in both their schema and their instances. Situational data and metadata are associated with quality and provenance annotations.

In [29] a new *OLAP-Overlay* operator was proposed so as to merge spatial data cubes (conforming spatial dimensions were not required). This operator is based on a Geographic Information System overlay operator that merges different layers using the topological intersection operator. The authors defined an algorithm that finds common instances (called members) between two dimensions by creating a new dimension that merges levels with the same members.

Various works have addressed the problem of conformed dimensions. In [24] a set of operators that coalesce data marts and perform *Drill-Across* operations on non-conforming dimensions were proposed. The authors suggested creating a new dimension by exploiting the **is–a** relationship. Intuitively, if two dimensions exhibit an **is–a** relationship with a common dimension, then they can be viewed as members of the parent dimension and can thus be combined in a meaningful way. In [26] a distinction was made between conformed dimension tables and conformed dimension attributes and the positive impact of relaxing the conformity requirement was discuss. The authors defined a method to measure the loss resulting from the join operation between conformed dimension attributes with dissimilar values. They extended the definition of the *Drill-Across* navigation operation to include in the analysis (selective) non-conformed dimension attributes.

In [28] the limits of the *Drill-Across* operator in improving the ability to connect two cubes that represent different aspects of the same reality were shown. The authors thus proposed a new operator called Drill-Across-link that introduces the explicit links that connect two cubes, and improves some of the *Drill-Across* operations.

A model of DW design using Data Mining (DM) algorithms (grouping, learning association rules) was proposed in [3]. The techniques used allowed the definition of dimension hierarchies according to the decision makers knowledge. The authors proposed a UML Profile to define a DW schema that integrates DM algorithms and a mapping process that transforms multidimensional schemata according to the results of the DM algorithms.

It was stated in related works that existing solutions are based either on attributes (logical schema) or instances. Unfortunately, the same attribute in two different cubes may not represent the same thing. For example, the attribute "category" may represent category of products, category of population, category of employees, etc. In the same way, instances may not represent the same reality. For example, *Jaguar* would be the animal or the Car Brand. We propose to combine schema and instances to check, using the conformity principle, if the common data represent the same concept. All the studied works consider that the user knows the cubes to merge. However, when the number of cubes is huge and the stored data is heterogeneous, it is hard to query these cubes. We compare related works according to some criteria: Conformity (the used levels), *Drill-Across* operator (the performed updates), Input data (how the query is defined and what data is needed by the approach), Output (what is the result of the approach), and Goal (what is the goal of the proposal) (see Table 2).

The existing works consider the conformity between three elements (dimensions, attributes and instances) to assess the conformity of dimensions when merging cubes. Nevertheless, these works do not provide details on the way to find similarity be it syntactic or semantic, etc. Also, none of the existing works considers all of these elements simultaneously. In [25], an approach was proposed which allows a user to express his need through a set of analysis indicators (measures, dimensions) which allow the study of the conformity based on syntactic and semantic similarity between all of the aforementioned elements.

4 Preliminaries

In this section, we present some preliminary definitions that lay the foundations for understanding our approach.

Definition 1 (*Collection*) A collection is a set of deployed cubes. We define C as a Collection of n cubes $C_1, C_2, \ldots, C_n$.

Definition 2 (*Structural Component* (SC)) The structural components represent the terms that describe the conceptual elements of a cube. A structural component refers to a Fact (F), a Measure [aggregate] ($M[A]$), Dimension (D) or a Level (L).

Definition 3 (*Cube*) A cube is a set of data constructed from a subset of a DW, organised and summarised into a multidimensional structure defined by a set of the structural components.

A cube C_j is represented by a set of Structural Components: a subject of analysis (Fact F), a set of Measures with or without aggregate functions ($M[A]$), analytical axes (Dimensions D) with a particular perspective, namely a Level of hierarchy (L), such as: $C_j = \langle F_j, M_j[A_j^+, D_j[L_j]^+\rangle$.

Example 1 The cube C_4 of Table 1 is composed of the Fact: Crop production; Measures: Input quantity, Output quantity, Pesticide concentration and Pesticide flow; Dimensions: *Technical operations*, *Product*, *Equipment*, *Pesticide*, *Time*, *Crops*, *Parcel* and *Production*, etc. (see Table 1).

Let C_j be a cube. We define $SC_D(C_j), SC_F(C_j), SC_L(C_j), SC_M(C_j), SC_A(C_j)$ the set of all instances of SC: *Dimension*, *Fact*, *Level*, *Measure* and *Aggregate* in C_j respectively. The set of all SC of a cube C_j [named $SC(C_j)$] is defined as: $SC(C_j) = SC_D(C_j) \cup SC_F(C_j) \cup SC_L(C_j) \cup SC_M(C_j) \cup SC_A(C_j)$.

Example 2 $SC_D(C_4)$ = *Technical operations*, *Product*, *Equipment*, *Pesticide*, *Time*, *Crops*, *Parcel*, *Production*.

Definition 4 (*Catalogue*) We define a Catalogue as the set of instances of all SC in the collection. Let $C = \{C_1, \ldots, C_n\}$ be the set of collection cubes; the Catalogue $SC(C)$ is defined as: $SC(C) = \{SC(C_1) \cup \cdots \cup SC(C_n)\}$.

Definition 5 (*Users Query* (Q)) A users query is expressed using n terms $(t_1, \ldots, t_n)$ in natural language and separated by the **AND** logic operators such as: $Q(t_1 AND t_2 AND t_3 AND\ AND t_n)$ where $t_i \in \{(M[A]), (D), (L)\}$, such as: $Q = \langle M_j[A]^+,\ D_j[L_j]^+\rangle$.

Example 3 Q_1 (*quantity of input product* **AND** *Pesticide concentration* **AND** *plot*) is an example of query.

Definition 6 (*Similarity*) The similarity is a function that quantifies the similarity between two terms. Two kinds of similarity are defined: syntactical similarity and semantic similarity [14].

Two terms are considered syntactically similar if they have a similar character sequence. Syntactical similarity is defined as String-Based similarity [14]. Syntactical similarity algorithms are based on the distance between terms (for example, *SimMetrics* package[3] proposes several

[3] https://github.com/Simmetrics/simmetrics.

Table 2 Related work

Research works	Cubes design				Cubes merging								Our approach
	[4]	[5]	[23]	[11]	[2]	[1]	[29]	[26]	[30]	[24]	[6]	[3]	
Conformity													
Dimension						x	x	x	x	x	x	x	x
Level						x			x			x	x
Instances and attributes					x			x		x			x
Drill-Across operator													
New operator(s)	x					x	x	x		x			x
New dimension(s)						x	x						x
New approach(s)									x		x	x	
Input													
Query type													
Cube name													x
Query answers		x	x								x		
Set of terms in NL													x
External sources					x								
Collection of cubes	x												x
Data warehouses	x	x	x		x	x		x		x	x		
Two cubes							x		x				
Graphs				x									
Output													
New cube	x	x	x	x		x	x	x		x			x
New data warehouse												x	
Enriched cube			x		x								
Integrated data mart									x				
New data mart											x		
Goal													
Performance		x	x								x	x	x
Relevance	x			x	x	x	x	x	x				x
Cost											x		x

distances: *Damerau–Levenshtein*, *Jaro*, *Jaro–Winkler*, *Needleman–Wunsch*, *Jaccard similarity*, etc.).

In Semantic Similarity, the terms are similar semantically if they reference the same thing, are opposite of each other, used in the same way, used in the same context or one is a type of another. Two kinds of algorithms are introduced for measuring the semantic similarity Corpus-Based and Knowledge-Based algorithms [14]. WordNet is the semantic network which is used in the area of measuring the Knowledge-Based similarity between terms, it is considered as a large semantic database of English [21].

Example 4 If we consider the previous query Q_1 (*quantity of input product* **AND** *Pesticide concentration* **AND** plot), the similar terms are defined as follow:

For the term "quantity of input product", the syntactically similar instances of structural components that are selected from the catalogue are ("output quantity", "input quantity", "consumed fuel quantity", "quantity of water", etc.).

For the term "plot", the semantic similar instance of structural component selected from the catalogue using *WordNet* is: ("parcel").

Definition 7 (*Reformulated query* Q') A reformulated query is a query, where each term t_i is replaced by a similar terms t'_i. The terms t'_i are instance of structural component of the catalogue. The reformulated query is defined by Q' (t'_1 **AND** t'_2 **AND AND** t'_n).

Example 5 The reformulated query Q'_1 for the query Q_1 is Q'_1 (*input quantity* **AND** *Pesticide concentration* [rate] **AND** *parcel*).

5 Our approach

The decision maker wants to analyse a given phenomenon in a single cube. He may have a clear need which he expresses by a set of decision indicators as measures and dimensions written in natural language. In this case, the query is composed of a set of terms separated by the *AND* operator.

Since the cubes collection can be large and subject to continuous updates, the user does not have a clear and complete view of the contents of the collection. Therefore, the terms used in its query may be ambiguous and sometimes obsolete.

To avoid the ambiguity associated with natural language problems, we propose to create a data structure called catalogue using OLAP cubes. The catalogue is used to perform an analysis of the query in order to form a clear unambiguous query. For this, we propose a preliminary step to prepare the data. This step consists of constructing the catalogue and refreshing it every time the OLAP schemas are updated.

On the other hand, the user may have part of his need in a known cube while he may want to complete or enrich it with data in other unknown cubes.

In this paper, we propose two variants to design and construct cubes according to the decision makers expressed need: an expressed need which is Based on Measures and Dimensions (**MDBV**) and an expressed need Based on Known Cube (**KCBV**). For the two variants, our approach returns a set of constructed cubes according to the decision makers need.

In this section, we present the main steps of our approach: (1) Preparation and definition of functional requirements, (2) Query Analysis, (3) Design and construction of cubes and (4) Ranking of constructed cubes. The architecture of our approach is shown in Fig. 1.

5.1 Preparation and definition of functional requirements

The preparation step consists firstly in generating the data cubes from the files in several formats (xl, csv). We parse these files to retrieve the structural components of a cube (fact, measures, dimensions, etc.) using the JDOM2 API to create the OLAP schema of a data cube. Secondly, we build a data structure, called catalogue, that is necessary for the querying, design and construction of cubes. In this step, our approach analyses the cubes OLAP schemas, extracts the different structural components of a cube and stores them in the catalogue. The latter represents all the instances of the multidimensional schema structural elements of the cubes.

Decision makers can express their analysis needs using terms. A term represents one or several elements of the catalogue. Indeed, our approach allows the user to select from a set of terms in the catalogue those that correspond to multidimensional concepts of deployed cubes. This avoids the problems of ambiguity that are related to natural language [10] and the difficulty to use complex computer languages (such as SQL, MDX) by decision makers [11]. The catalogue is necessary and will be used in the query analysis process. In order to make the OLAP platform (such as Pentaho, Oracle, etc.) independent, we propose, in Fig. 2, a meta-model to define all OLAP concepts that are used in this work.

In the proposed meta-model, a cube schema contains a fact that is described by several measures, where each measure is associated with several aggregate functions. In particular, we consider that a cube is composed of several dimensions with several levels. For example, the *Dairy Production* cube has the dimensions: *product*, *operator*, etc. The catalogue contains, among others, the element *product*. Using this catalogue, the decision maker can define a term product looking for all cubes about the *agricultural production*.

5.2 Query analysis process

The decision maker (user) expresses his need through a set of terms written in a natural language. These terms may not be accurate, making the expressed need ambiguous. We propose to perform an analysis of the query in order to guarantee that is clear and well-formed and to avoid the problems of ambiguity that are inherent to natural language [22]. The query analysis process provides the user, for each query term, with a set of multidimensional concepts (instances of structural components). These concepts are extracted from the catalogue [the set $SC(C)$] (see Sect. 4). For each term in the user's query, a mapping operation with multidimensional concepts is performed; it returns a set of concepts that are similar with the input term. This feature is especially important when the user has limited knowledge about the exact representation of the entities he is looking for.

The main idea for the query analysis process is to let the decision maker use natural language, then provide him with a set of similar instances of structural components that are proposed from the catalogue (3). We use string similarity with several distance functions such as *Cosine similarity*, *Jaro–Winkler Distance*, *Levenshtein Distance*, etc. In our approach, we performed a comparison between the results of these distances and we opted for the Levenshtein distance (as it gave the best results relative to the provided terms). The query analysis process is illustrated in Fig. 3.

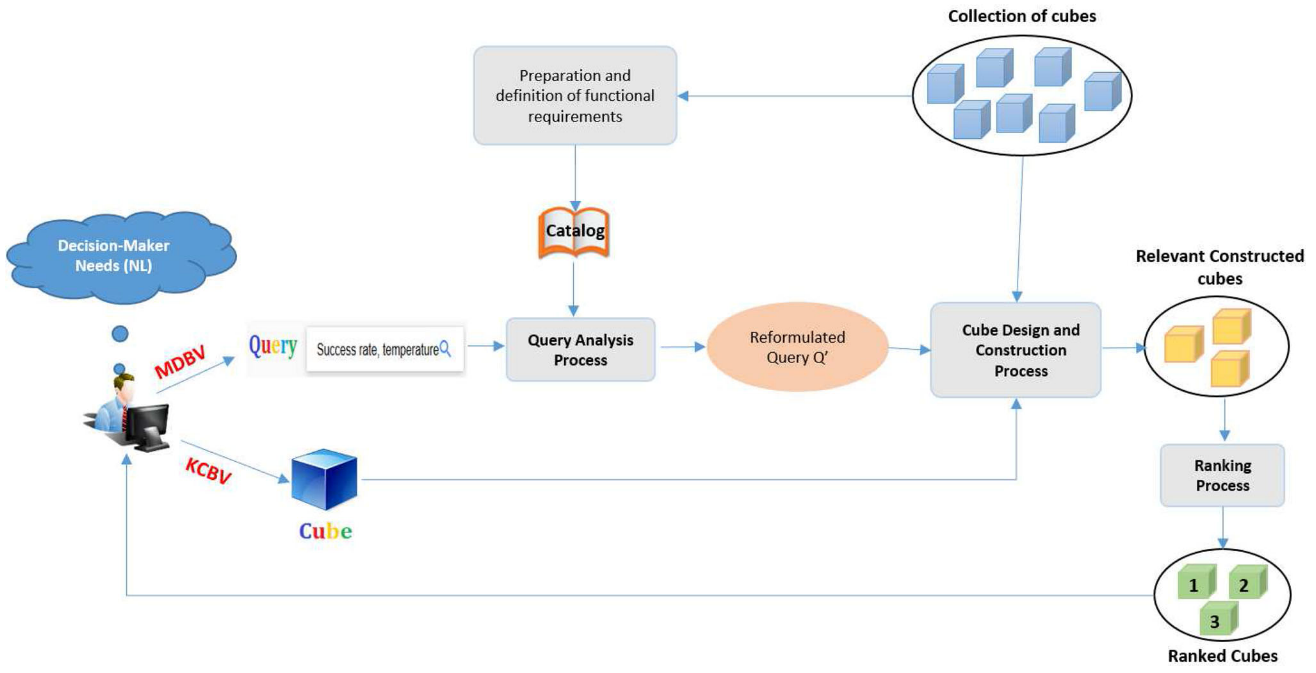

Collection of cubes
Preparation and definition of functional requirements
Catalog
Decision-Maker Needs (NL)
MDBV
Query
Success rate, temperature
Query Analysis Process
Reformulated Query Q'
Cube Design and Construction Process
Relevant Constructed cubes
KCBV
Cube
Ranking Process
1
2
3
Ranked Cubes

Fig. 1 General architecture of our approach

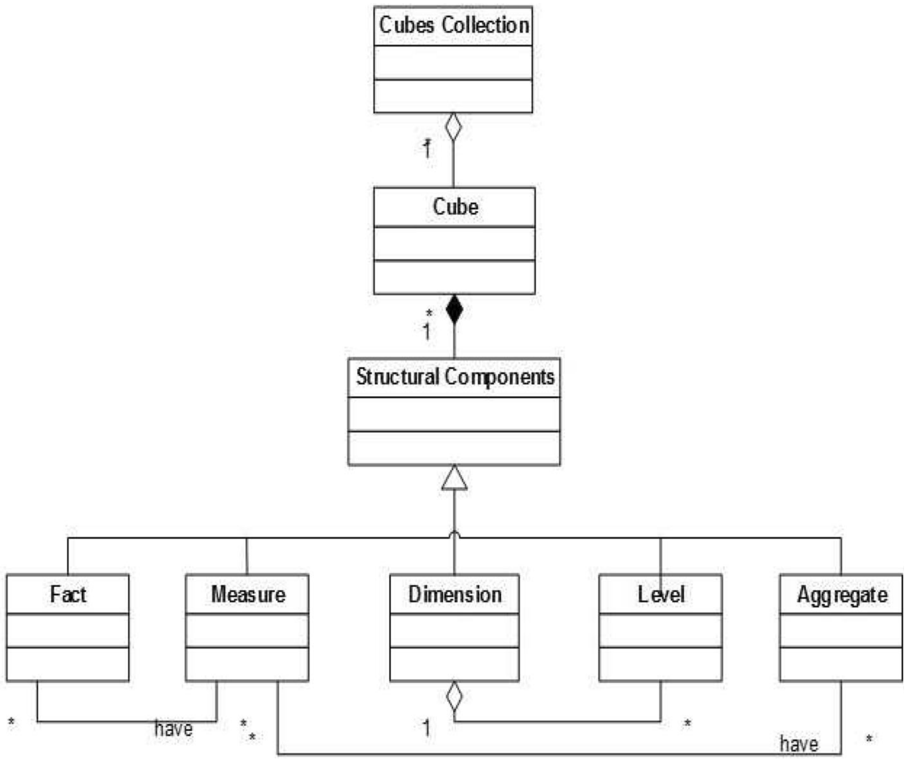


Fig. 2 Cube schema meta-model

However, since it considers only the syntactic aspect, the *Levenshtein distance* is not effective when the entered term is not similar to any term in the catalogue (set of cubes). Indeed, a term may have semantically similar terms in the catalogue but the analyser would not detect them. To deal with this problem, we propose to use external resources in order to conduct an analysis of semantic similarity and replace a term by its synonyms. As part of this work, we propose a simple solution that makes use of an ontology of synonyms such as *WordNet*. This resource

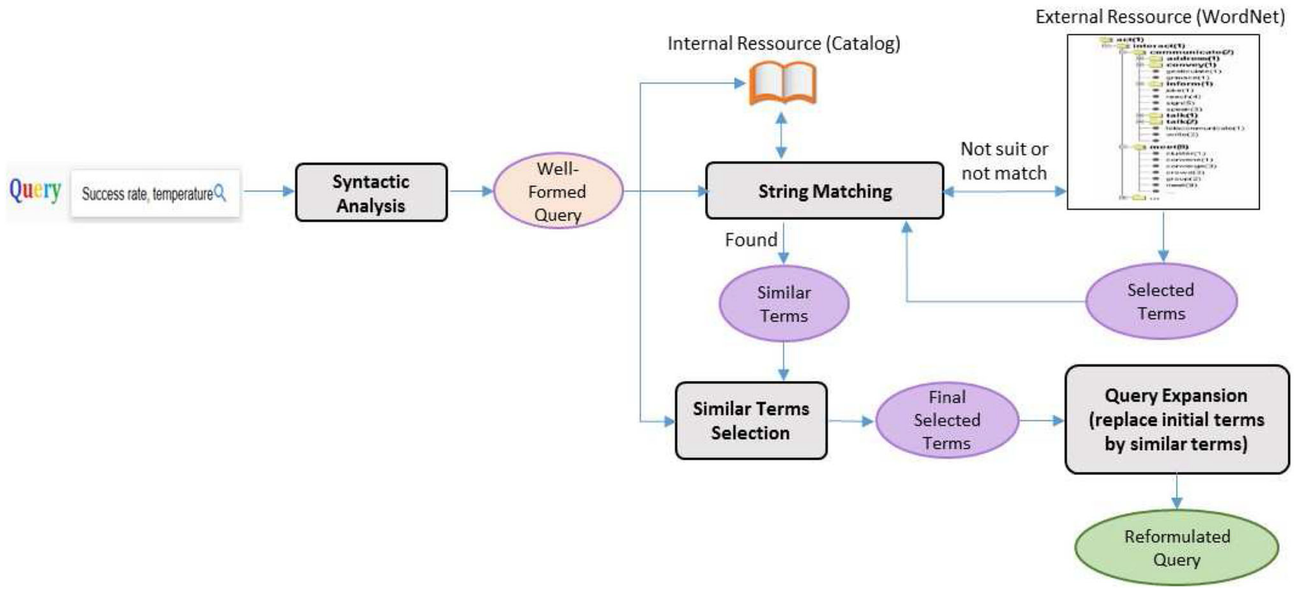


Fig. 3 Query analysis process

contains, for each term, a set of terms that have the same meaning. We note that this resource is used only in the case where no term in the catalogue is similar to a term entered by the user or when the proposed terms are not suitable for the user in terms of the query analysis process.

The query analysis process combines two resources, the internal (catalogue) and the external (*WordNet*) resource. The process consists firstly in selecting the similar terms from the internal resource (catalogue). If no term is appropriate for the user, the system provides a set of similar terms from the external resource (*WordNet*). The term selected from this resource will be compared with similar terms in the internal resource. This process allows the expansion of the search space so as to find more candidate cubes which never be returned using the original query . Clearly, these features can further improve the users search [19].

Example 6 Figure 4 shows the analysis of the initial user query *Q*: *amount of fuel consumption* **AND** *Plot* **AND** *Acreage*.

For the first term *amount of fuel consumption*, the analysis process provides the user with the following similar instances of structural components (*Rate of consumption*, *Consumption fuel quantity*, *Fuel*). Thereafter, the user selects the instance of structural component *consumption fuel quantity*. Since the first term is a measures instance, then the analysis process provides the terms (aggregate) *total* and *avg*. The user chooses *total*. The selected term (*total consumption fuel quantity*) is a compound term (Measure + Aggregate). For the term *plot*, the analysis process provides the user with a set of similar instances of structural components but no term is suits him. Then the analysis process provides from the external *WordNet* resource the following similar terms (*Parcel*, *Patch*, *Piece*, *Land*). Thereafter, the user selects the term *Parcel*. Then again, the analysis process provides the user with the following similar instances of structural components (*Parcel*, *Product*, *Production*, *Operator*, and *Pesticide*). The same process is repeated for each term. For the term *Acreage*, the analysis process finds no similar term in the catalogue, and then it automatically offers the user a set of similar terms from *WordNet*, and the above process is then repeated.

Example 7 The resulting reformulated query for *Q* of Example 2 is *Q′* where: *Q′* (*Total Consumption fuel quantity* **AND** *Parcel* **AND** *Treated Area*).

We consider also, in this paper, the case where the user has a cube containing part of his need. He can select this cube in which case we get all multidimensional concepts of this cube from the schema described in the XML file. The query analysis module is not activated in this case.

5.3 Cubes design and construction process

Our approach proposes two variants to design and construct cube according to the need expressed by decision maker: (a) constructing cubes based on measures and dimensions: the input of this variant is a query composed of a set analysis indicators (measures and dimensions) separated by the **AND** operator and (b) constructing cubes based on a known cube: in this variant, our approach takes as input a cube (name) from a set of deployed cubes of the collection.

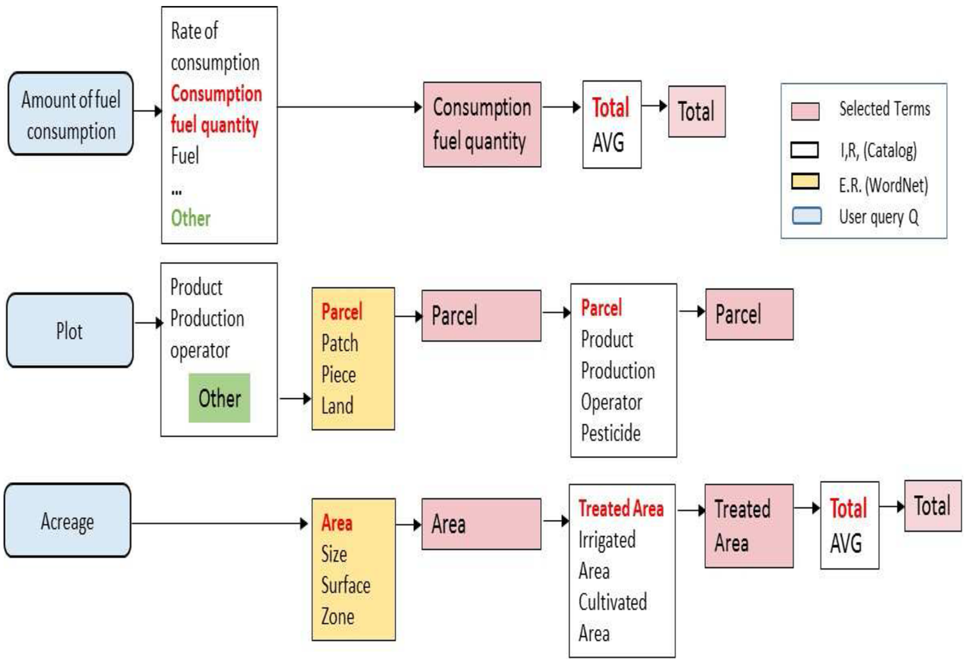


Fig. 4 Running example of query analysis step

The cubes design and construction process passes through four main steps: (1) candidate cubes search, (2) conformity check, (3) design of cubes (grouping), and (4) fusion (Fig. 5). For each step, we explain in detail the process for the two variants.

5.3.1 Candidate cubes search

In this step, we seek all candidate cubes that can be merged to satisfy the user's need.

5.3.1.1 Searching cubes based on analysis indicators In this case, we seek all cubes that partially satisfy the user query. The search process is to make a mapping between the query terms and the OLAP schemas of existing cubes.

A cube C_j ($C_j \in C$) in the collection is a candidate for fusion if the following conditions are verified:

- C_j must contain at least one measure referenced in the query: $SC_M(Q') \cap SC_M(C_j) \neq \emptyset$.
- C_j must contain all dimensions referenced in the query (if any): $SC_D(Q') \subseteq SC_D(C_j)$.
- The measures referenced by the query are scattered across multiple cubes: $\neg\exists C_j \in C \wedge SC_M(Q') \subseteq SC_M(C_j)$.

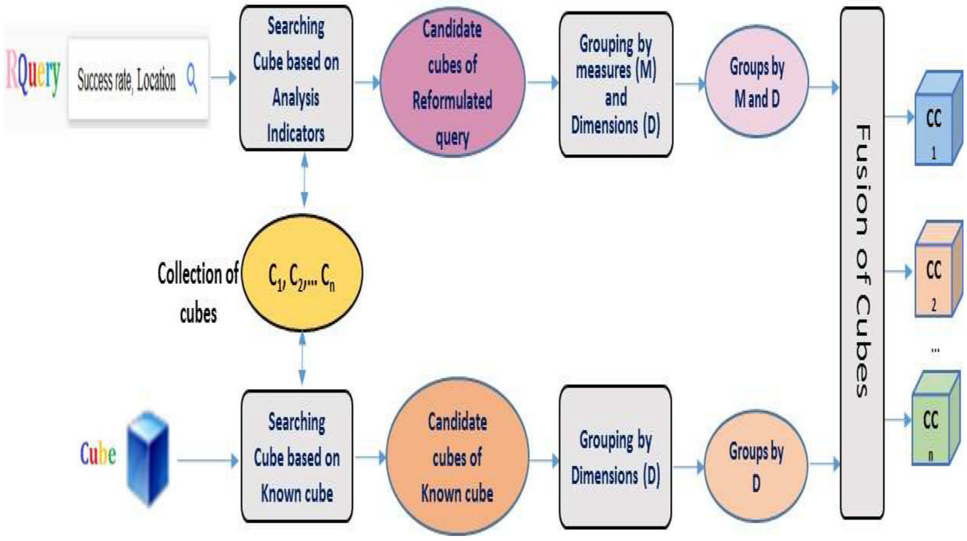

RQuery
Success rate, Location
Searching Cube based on Analysis Indicators
Candidate cubes of Reformulated query
Grouping by measures (M) and Dimensions (D)
Groups by M and D
Collection of cubes
C1, C2,... Cn
Cube
Searching Cube based on Known cube
Candidate cubes of Known cube
Grouping by Dimensions (D)
Groups by D
Fusion of Cubes
CC 1
CC 2
...
CC n

Fig. 5 Cubes construction process

Example 8 Let us suppose that a decision maker wants to analyse "the production quantity with the irrigation water quantity per parcel", the reformulated query is "output quantity and quantity of water and parcel". **The set of candidate cubes of this query is** $CC'_Q = \{C_1, C_2, C_3, C_4\}$.

5.3.1.2 Searching cubes based on a known cube

In this case, we seek all cubes of the collection that can be merged with the input cube to satisfy the user's need. These cubes must have at least one common dimension with the input cube.

The search process performs a mapping between all dimensions of the initial user's cube (C_i) and the OLAP schemas of existing cubes.

A cube $C_j(C_j \in C)$ in the collection (C) is a candidate for fusion if C_j must contains at least one common dimension with input cube (C_i).

Example 9 Let us suppose that a decision maker wants to analyse irrigation operation. He wants to have in a single cube all information that are related and influence this operation. The initial user's cube is "irrigation".

The set of candi dates cubes that can be merged with this cube is $C(C_i) = \{C_1, C_2, C_4, C_5, C_7\}$ with common dimensions *crops*, *time*, *location*, *equipment*.

To merge two or more cubes, they must have at least one common dimension. The potential for crossing several data cubes using the *Drill-Across* operator is closely related to the notion of common conformed dimensions [18]. We present in the next section, how we check the conformity of dimensions.

5.3.2 Conformity check

Our approach aims to construct a new cube from several cubes of the collection. These cubes may be heterogeneous and may have no shared dimension. After a deep study, we have concluded that unshared dimensions may become conform dimensions. Two dimensions are considered conformed if they represent the same thing (reality) [20]. We address the problem of conformity between dimensions by a (one-to-one) mapping between their hierarchy levels [30].

We propose a conformity check between dimensions, levels of hierarchy and instances. In this work, we determine the conformed dimensions using the similarity of two texts based on the semantic and syntactic information that they contain. We consider two similarity functions in order to have more generalised similarity. First, we consider string similarity using the *Jaro–Winkler* distance and semantic word similarity using the *WordNet* Ontology.

5.3.2.1 Syntactic similarity

We use the Jaro–Winkler distance to define the syntactic similarity between dimensions. The *Jaro distance* metric was introduced in 1989 by Matthew A. Jaro as a comparator that accounts for insertions, deletions, and transpositions [7].

The basic *Jaro* algorithm has three components which respectively aim to: (1) compute the string lengths, (2) find the number of common characters in the two strings, and (3) find the number of transpositions. The definition of common characters is that the matching characters must be within half the length of the shorter string.

The *Jaro distance* d_j between two strings is defined by the Formula (1):

$$d_j = \frac{1}{3}\left(\frac{m}{|S_1|} + \frac{m}{|S_2|} + \frac{m-t}{|m|}\right), \tag{1}$$

where $|S_1|$ and $|S_2|$ are the lengths of the two strings. m is the number of matching symbols and t is the number of transpositions.

Two characters are called matching if the one from the string S_1 coincides with the one from the string S_2 that is located not farther than [the distance defined by Formula (2)].

$$\frac{Max(|S_1|, |S_2|)}{2} - 1. \tag{2}$$

For each pair of matching characters with different sequence order the number of transpositions t is increased by Formula (1).

To say that two dimensions/levels are syntactically similar, we consider a threshold T. If the distance $D_j \geq T$, then the dimensions/levels are deemed syntactically similar.

Example

Let us consider the two dimensions "location" and "localization", apply the *Jaro–Winkler* distance. We get: $m = 5,\ |S_1| = 7,\ |S_2| = 15,\ t = 1,\ d_j = 1/3(5/8 + 5/12 + 1 - 1/5) = 0.613$.

5.3.2.2 Semantic similarity

In the previous step, we calculated the syntactic similarity between two dimensions/levels using the Jaro–Winkler distance. However, two dimensions/levels may be not syntactically similar while they are synonyms. In this case, we propose to calculate the semantic similarity between dimensions/levels. There is a relatively large number of semantic similarity metrics in the literature, ranging from distance-oriented measures computed on semantic networks or knowledge-based (dictionary/thesaurus) measures, to metrics based on models of information theory (or corpus-based measures) learned from large text collections. In this work, we focus on knowledge-based measures to determine the semantic similarity between dimensions and levels. To do this, we

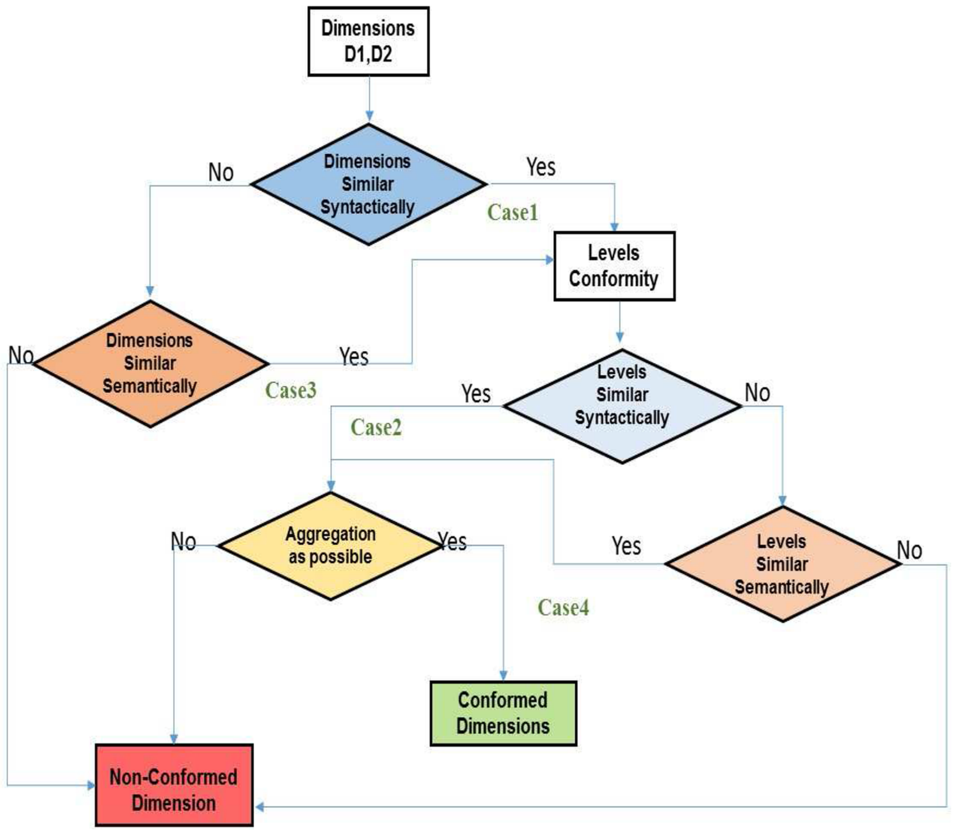


Fig. 6 The conditions to construct cubes

propose to use WordNet. WordNet is a lexical database that contains English nouns, verbs, adjectives and adverbs, organised in sets of synonym senses (synsets). The terms senses, synsets and concepts are used interchangeably. Synsets are connected with various links that represent semantic relations between them such as hyperonymy, hyponymy, synonymy, antonymy,...,etc. [17]. In our approach, we use the synonymy relation between two dimensions/levels.

Example

For the previous example between *location* and *localization*, if the threshold is fixed to 0.8, the system detects that the two dimensions are not syntactically similar whereas, they are synonyms using *WordNet*.

We check the conformity of the dimensions and the levels through the syntactic and semantic similarity by studying all the possible cases. From two dimensions, we define the cases where they are conformed and the cases where they are not. For the last case, we present the solution to make these dimensions conform. In this paper, we consider the cases listed below.

Case 1 The dimensions are syntactically similar (matching string) when they have the same hierarchy levels and when these levels are syntactically similar.

Example

D1: **Time** [**Year**, **Month**, **Day**]; *D2*: **Time** [**Year**, **Month**, **Day**] In this case, the two dimensions are conformed, so we use the *Drill-Across* merging operator which is widely cited in the literature.

Case 2 The dimensions are syntactically similar and they do not have the same hierarchy levels. In this case, we consider two sub-cases:

Case 2.1 The levels are syntactically similar but one level is more detailed than the other.

Example

D1: Time [Day, **Month**, **Year**]; *D2*: Time [**Month, Year**] In this case, we calculate the intersection between the two levels, and then we apply an aggregation operation (*Roll-Up*) to bring the two hierarchies at the same level of detail. The merge operation is made at this level. For this example, an aggregation is feasible by applying the Roll-up operator on the dimension *D1*: Time [Month, Year] then dimension *D1* and *D2* are conformed.

Case 2.2. The levels are not syntactically similar but semantically similar.

Example

D1: *Time* [**Year**, **Month**, **Day**]; *D2*: *Time* [**YY**, **MM**, **DD**]. In this case, we use external resources. In this work, we use the *WordNet* ontology and a glossary containing the most handled similar concepts. The glossary can be updated by adding new concepts. The procedure for checking the conformity is based on the calculation of similarity between the hierarchy levels.

Beyond a certain threshold of similarity, the dimensions are considered conformed (this threshold will be set empirically). In this case, the degree of similarity between the levels of these two dimensions, calculated using the glossary, is 100% (Year = YY, Month = MM, Day = DD). Therefore, the levels are conformed.

Case 3 The dimensions are not syntactically similar but have the same hierarchy levels.

Example

D1: **Period** [Year, Month, Day]; *D2*: **Time** [Year, Month, Day]

In this case, we use the same solution proposed in Case 2.2.

Case 4 The dimensions are not syntactically similar and do not have the same detail levels (at least one shared level).

Example

D1: **Period** [Year, Month, Day]; *D2*: **Time** [Year, Month] We combine in this case the solutions for cases 2.1 and 2.2. Figure 6 summarizes the conformity checking process between dimensions. In this paper, we check the conformity between instances using the intersection operator. If the intersection is empty, then we consider that the dimensions are not conformed.

We propose a set of algorithms to verify the conformity: between two dimensions (Algorithm 1) and between hierarchical levels (Algorithm 2). We calculate Jaro–Winkler distance between terms to test syntactic similarity using a threshold (Algorithm 3), and we used the *WordNet* ontology to check the semantic similarity (Algorithm 4). Finally, we study the opportunity of apply an aggregation (Algorithm 5).

Algorithm 1: Dimensions Conformity

```
Input: $D_1$, $D_2$, tow dimension;
Output: Conform Boolean;
1: Begin
2:    IF Syntactic Similarity ($D_1$, $D_2$) Then
3:        Conform:=Levels-Conformity ($L_i$, $L_j$);
4:    Else
5:            IF Semantic Similarity ($D_1$, $D_2$) Then
6:                Conform: =Levels-Conformity ($L_i$, $L_j$);
7:            Else
8:                Conform: = False;
9:            End IF;
10:   End IF;
11: End.
```

Algorithm 2: Levels Conformity

```
Input: $L_1$, $L_2$: two levels' lists for two dimensions $D_1$, $D_2$
Output: Conform Boolean.
Intermediate variables: $i$, $j$ String.
1: Begin
2:    For each level $i$ of $L_1$ and For each level $j$ of $L_2$
3:        Begin
4:            IF Syntactic Similarity ($i$,$j$) Then
5:                Conform: =Aggregation ($L_i$, $L_j$);
6:            Else
7:                IF Semantic Similarity ($i$,$j$) Then
8:                    Conform: =Aggregation ($L_i$, $L_j$);
9:                Else
10:                   Conform: = False;
11:               End IF
12:           End IF
13:   End
14: End.
```

Algorithm 3: Syntactic Similarity

```
Input: Two terms $t_1$, $t_2$ string, Threshold $T$ real.
Output: Similar Boolean.
Intermediate variables: $d$ real;
1 : Begin
2 :    $d$ := distance-calculation ($t_1$, $t_2$)
3:        IF ( $d >= T$ ) Then
4:            Similar: = True;
5:    Else
6:            Similar: = False;
7:        End IF
8: End.
```

Algorithm 4: Semantic-Similarity

```
Input: Two terms $t_1$, $t_2$ string, WordNet $DB$ Ontology
Output: Similar Boolean.
1: Begin
2:    Seek $t_1$ in WordNet $DB$
3:        IF found $t_2$ as a Synonym of $t_1$ Then
4:            Similar: = True;
5:        Else
6:            Similar: = False;
7:        End IF
8: End.
```

Algorithm 5: Aggregation

```
Input: $L_1$ , $L_2$ : two level lists of two dimensions $D_1, D_2$
Output: Conform Boolean
Intermediate variables: $L_c$: conform levels list
1: Begin
2:     IF $L_1 \cap L_2 = \emptyset$ Then
3:         Conform: = False;
4:     Else
5:         $L_c$: = $L_1 \cap L_2$ ;
6:         Applying Roll-up-operator ($L_c$);
7:         Conform: = true;
8:     End IF
9: End.
```

5.3.3 Cubes design (grouping)

After the conformity checking, the design of new cubes is performed by forming a set of groups. The cubes of the same group are merged to construct new cubes that contain the entire user's need.

5.3.3.1 Grouping by measures and dimensions This step takes as input a set of candidate cubes identified by the previous step and returns a set of groups (Fig. 7).

The cubes placed in the same group Gr must satisfy the following conditions:

- The measures of the query should be scattered over the cubes of the group $(\neg\exists C_j \in Gr \wedge SC_M(Q') \subseteq SC_M(C_j))$
- All measures of the query must exist in the cubes of the group $Gr(SC_M(Q') \subseteq \cup_{C_j \in Gr} SC_M(C_j))$.
- The cubes of the group must share at least one conformed dimensions $(\cap_{C_j \in Gr} SC_D(C_j) \neq \emptyset)$.
- If the query references dimensions, then all the cubes of the group must share all these dimensions $(SC_D(Q') \subseteq (\cap_{C_j \in Gr} SC_D(C_j)))$.

Example 10: For the query in Example 8, $CC'_Q = \{C_1, C_2, C_3, C_4\}$. The groups are shown in Table 3

5.3.3.2 Grouping by dimensions After the conformity verification, we obtain a set of candidate cubes for fusion. From this set of candidate cubes, we form the groups of cubes that have common dimensions. A cube can belong to several groups (Fig. 8). The cubes placed in the same group Gr must share at least one conformed dimension such as: $Gr(SC_M(Q') \subseteq \cup_{c_j \in Gr} SC_M(C_j))$.

Example 10 For the known irrigation cube (C_3) in Example 8, $C(C_i) = \{C_1, C_2, C_4, C_5, C_7\}$. The groups are shown in Table 4.

5.3.4 Fusion of cubes

5.3.4.1 Fusion by measures and dimensions Once the groups generated, we construct one cube per group.

The measures and dimensions of the new cube NC_m $(NC_m = C_1 \bowtie C_2 \bowtie \cdots \bowtie C_n)$ are defined as follows ($\bowtie$ is the fusion symbol):

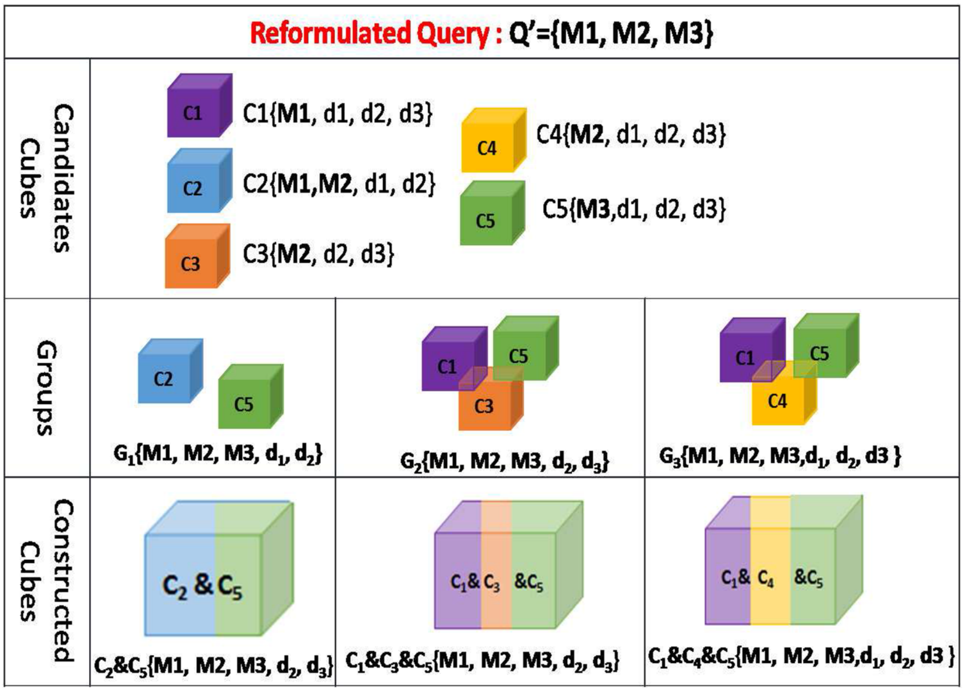


Fig. 7 Grouping by Measures and Dimension (Variant 1)

Table 3 Groups formed for the query in Example 8

Groups	Cubes	Measures	Common dimensions
Gr_1	C_1	Output quantity	Location [parcel], time [season, year], crops [crops]
	C_3	Quantity of water	Location [parcel], time [season, year], crops [crops]
Gr_2	C_2	Output quantity	Location [parcel, region], time [season, year], crops [crops]
	C_3	Quantity of water	Location [parcel, region], time [season, year], crops [crops]
Gr_3	C_4	Output quantity	Location [parcel, department, region], time [day, month, season, year], crops [crops], equipment [equipment]
	C_3	Quantity of water	Location [parcel, department, region], time [day, month, season, year], crops [crops], equipment [equipment]

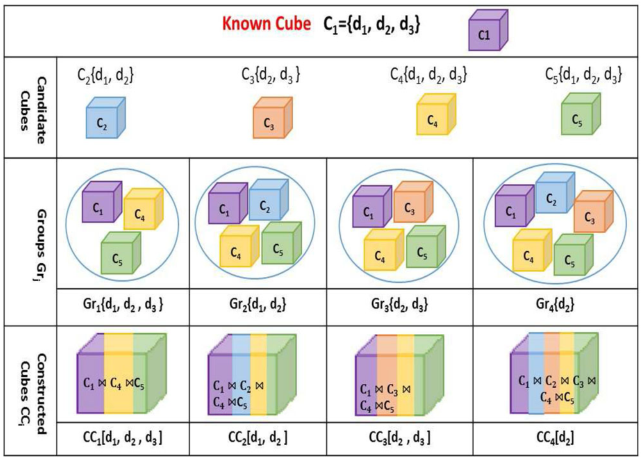


Fig. 8 Grouping by dimensions (Variant 2)

Table 4 Groups formed for the query in Example 8

Groups	Cubes	Common dimensions
GR_1	C_3	Location [parcel], time [day, month, season, year], crops [crops], equipment [equipment]
	C_4	Location [parcel], time [day, month, season, year], crops [crops], equipment [equipment]
	C_5	Parcel [parcel], time [day, month, season, year], crops [crops]
GR_2	C_3	Location [parcel], time [season, year], crops [crops]
	C_1	Location [parcel], time [season, year], crops [crops]
	C_2	Location [parcel], time [season, year], crops [crops]
GR_3	C_3	Localization [parcel, department, region], time [hour, day, week, month, season, year]
	C_7	Location [parcel, department, zone], time [hour, day, week, month, season, year]
GR_4	C_3	Time [day, month, year]
	C_6	Time [day, month, year]

– The measures of NC_m are the union of the measures of the merged cubes that are referenced in the query $Q'(SC_M(NC_m) = [\cup_{i=1}^{n} SC_M(C_i)] \cap SC_M(Q'))$
– The dimensions of NC_m are the intersection of the dimensions of the merged cubes $(SC_D(NC) = [\cap_{i=1}^{n} SC_D(C_i)] \cap SC_D(Q'))$.

Example 11 For the query in Example 8, the result of the construction is shown in Table 5.

5.3.4.2 Fusion by dimensions In this variant, we propose an interactive system with the decision maker. Once the groups are generated, the decision maker selects a group of

Table 5 Constructed cubes for query in Example 8

Constructed cubes	Measures	Dimensions
$C_1 \bowtie C_3$	Output quantity quantity of water	Location [parcel], time [season, year], crops [crops]
$C_2 \bowtie C_3$	Output quantity quantity of water	Location [parcel, region], time [season, year], crops [crops]
$C_4 \bowtie C_3$	Output quantity quantity of water	Location [parcel, department, region], time [day, month, season, year], crops [crops], equipment [equipment]

cubes from the cubes returned by our approach. In addition, he chooses the measures and dimensions that correspond to his need.

We construct one cube per group; the measures and dimensions of the new cube $NC_d(NC_d = C_1 \bowtie C_2 \bowtie \cdots \bowtie C_n)$ are defined as follow:

- The measures of NC_d are the union of the measures of merged cubes (d_m) chosen by the decision maker: $(SC_M(NC_d) = [\cup_{i=1}^{n} SC_M(C_i)] \cap SC_M(d_m))$
- The dimensions of NC_d are the intersection of the dimensions of merged cubes (d_m) chosen by the decision maker: $(SC_D(NC) = [\cap_{i=1}^{n} SC_D(C_i)] \cap SC_D(d_m)$.

5.4 Ranking of constructed cubes

The fusion step returns a set of constructed cubes in no specific order. To sort the resulting constructed cubes, we propose to order the cubes according to the number of cubes in the groups and the number of dimensions of the constructed cube. The sorting process is as follows: sort the groups according to the number of cubes in descending order. In the case where the groups have the same number of cubes, the process proceeds to sort them according to the number of dimensions of the cubes in ascending order. The process starts with the newly constructed cube from the group that contains the least number of cubes. In other words, it is the cube that requires less cubes for the merge operation. Then, for the groups which have the same number of cubes, we select the cube that contains more detailed information. The idea is based on the following assumption: the cube that contains more information is the most relevant to the user. On the one hand, the presence of several dimensions allows the user to make analyses that are more detailed in the returned cubes. The decision maker can perform aggregations to particular dimensions in order to keep only the dimensions that are referenced in the query.

Example 12 For the query in Example 8, the constructed cubes are $(C_4 \bowtie C_3)$ (seven common dimensions); $(C_1 \bowtie C_3)$ (six common dimensions) and $(C_2 \bowtie C_3)$ (four common dimensions). However, the decision maker can choose the first cube to make its analysis as he can exploit another cube.

6 Implementation and results

We present in this section, the tool that we have developed to implement our approach and some tests done to validate it.

We have developed a tool called "Design-Cubes-Query" implementing our approach. Our tool is based on a multi-tier architecture (Fig. 9). The Data tier aims to store multidimensional data, and it is implemented using Relational DBMS Postgres. The OLAP tier is implemented using the OLAP server Mondrian, where the cubes are defined. The Explorer tier is composed of a Cube exploratory tool, which allows the decision makers to explore the constructed cubes and visualise their instances. To show the features of our tool, we consider a set of 50 cubes downloaded from the statistical agro-environmental open-data cubes developed by FAO and pilot farm cubes.

The tool provides two easy interfaces, one for decision makers to allow them to define their need in natural language, view the constructed cubes, explore and visualise the data. The second interface is for IT-designer; it allows them, in addition to the functionality of the first interface, to load, update the collection of cubes and the catalogue and visualise the conceptual schema of the constructed cubes.

In addition, we have developed two variants: (1) the *Cube-Based Construction* Variant (**KCBV**) and (2) the Measure-Based Construction Variant (**MDBV**).

The Cube-Based Construction Variant (**KCBV**) (Fig. 10) is used if the user has a part of the need in a known cube. He starts by selecting the cube, then the system searches and displays all candidate cubes that can be merged with the first one. The candidate cubes are identified by automatically checking the dimension conformity. The system then displays the common dimensions between the cubes. The user selects among the common dimensions those that he wants to have in the constructed cube. For example, in Fig. 10, if the user selected the *Irrigation* Cube, then the system displays the cubes: *Food Production*, *Agricultural Production*, *Agro-Industrial*

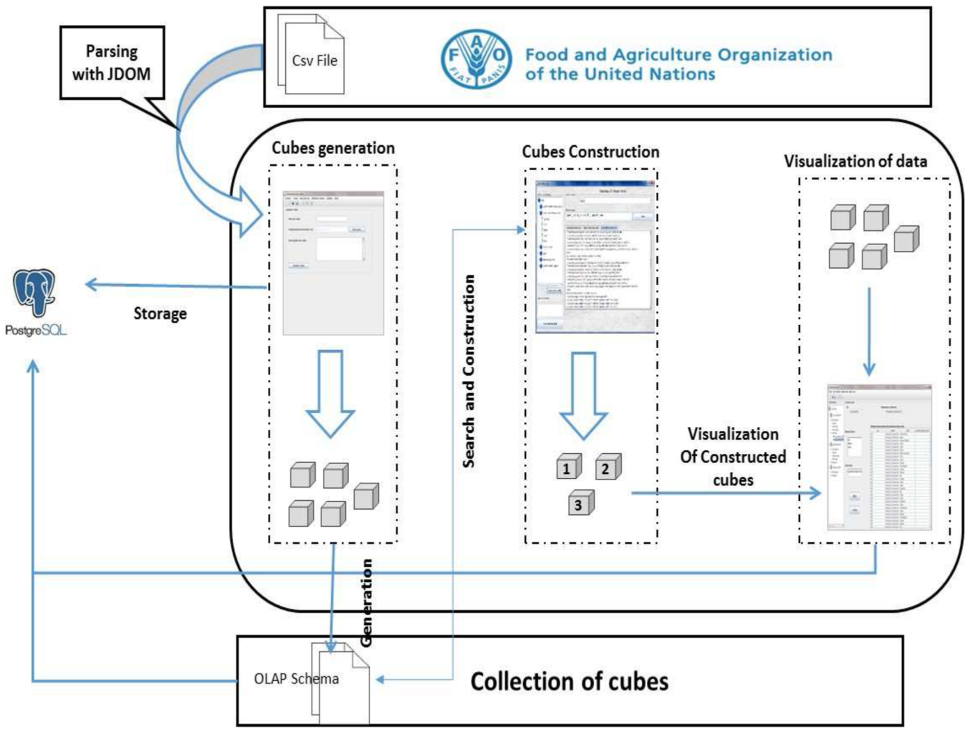
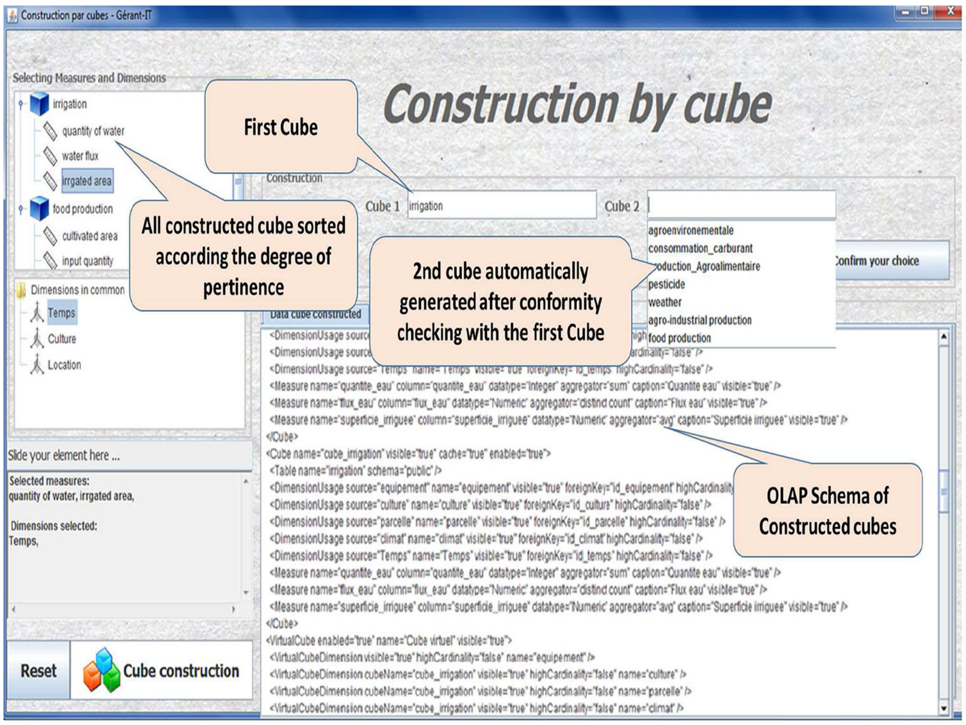


Fig. 9 Architecture of the *Design-cube-query* tool



Cube 1: irrigation

Selected measures:
quantity of water, irrgated area,

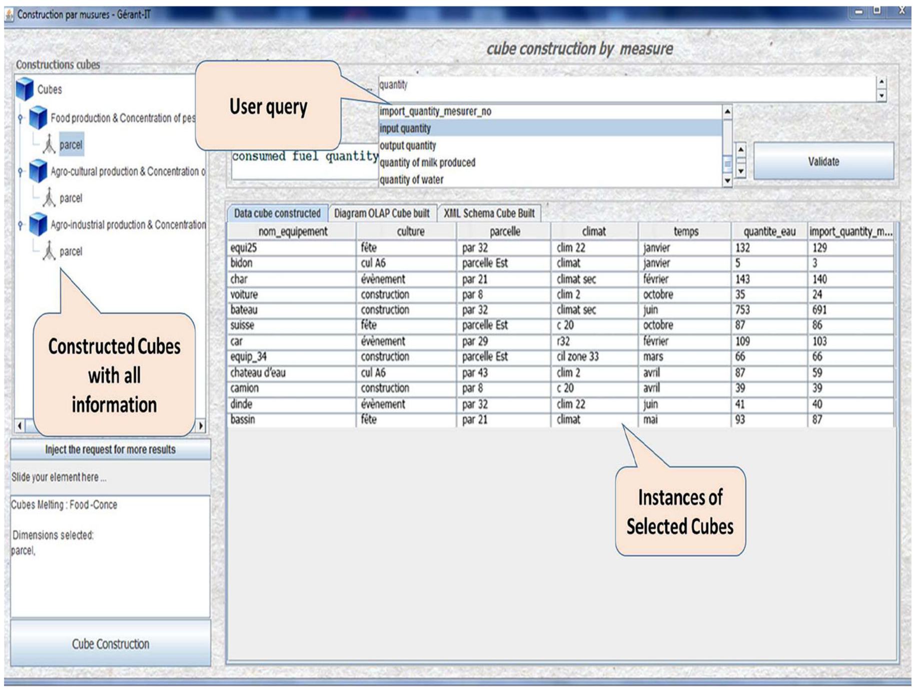
Dimensions selected:
Temps,

```
 <DimensionUsage sourc
 <DimensionUsage sourc                                                 ardinality="false" />
 <DimensionUsage source="Temps" name="Temps" visible="true" foreignKey="id_temps" highCardinality="false" />
 <Measure name="quantite_eau" column="quantite_eau" datatype="Integer" aggregator="sum" caption="Quantite eau" visible="true" />
 <Measure name="flux_eau" column="flux_eau" datatype="Numeric" aggregator="distinct count" caption="Flux eau" visible="true" />
 <Measure name="superficie_irriguee" column="superficie_irriguee" datatype="Numeric" aggregator="avg" caption="Superficie irriguee" visible="true" />
</Cube>
<Cube name="cube_irrigation" visible="true" cache="true" enabled="true">
 <Table name="irrigation" schema="public" />
 <DimensionUsage source="equipement" name="equipement" visible="true" foreignKey="id_equipement" highCardinality
 <DimensionUsage source="culture" name="culture" visible="true" foreignKey="id_culture" highCardinality="false" />
 <DimensionUsage source="parcelle" name="parcelle" visible="true" foreignKey="id_parcelle" highCardinality="false" />
 <DimensionUsage source="climat" name="climat" visible="true" foreignKey="id_climat" highCardinality="false" />
 <DimensionUsage source="Temps" name="Temps" visible="true" foreignKey="id_temps" highCardinality="false" />
 <Measure name="quantite_eau" column="quantite_eau" datatype="Integer" aggregator="sum" caption="Quantite eau" visible="true" />
 <Measure name="flux_eau" column="flux_eau" datatype="Numeric" aggregator="distinct count" caption="Flux eau" visible="true" />
 <Measure name="superficie_irriguee" column="superficie_irriguee" datatype="Numeric" aggregator="avg" caption="Superficie irriguee" visible="true" />
</Cube>
<VirtualCube enabled="true" name="Cube virtuel" visible="true">
 <VirtualCubeDimension visible="true" highCardinality="false" name="equipement" />
 <VirtualCubeDimension cubeName="cube_irrigation" visible="true" highCardinality="false" name="culture" />
 <VirtualCubeDimension cubeName="cube_irrigation" visible="true" highCardinality="false" name="parcelle" />
 <VirtualCubeDimension cubeName="cube_irrigation" visible="true" highCardinality="false" name="climat" />
```

Fig. 10 Cube-based construction variant

Production, *Pesticide* and *Weather*. In the Measure-Based Construction Variant (**MDBV**) (Fig. 11), the user expresses his need through a set of measures and dimensions. The query analyser provides the user, for each measure, with a similar set of measures from the catalogue and a set of dimensions in order to select those that suit him.

Example 13 Let the query be (*quantity of input product* **AND** *pesticide concentration* **AND** *parcel*). For the first measure *quantity of input product*, the query analyser provides the following similar instances of measures: *input quantity*, *output quantity*, *consumed fuel quantity*, etc. The same process is repeated to all measures and dimensions.


Construction par musures - Gérant-IT
cube construction by measure
Constructions cubes
Cubes
Food production & Concentration of pes
parcel
Agro-cultural production & Concentration o
parcel
Agro-industrial production & Concentration
parcel
User query
quantity
import_quantity_mesurer_no
input quantity
output quantity
quantity of milk produced
quantity of water
consumed fuel quantity
Validate
Constructed Cubes with all information
Inject the request for more results
Slide your element here ...
Cubes Melting : Food-Conce
Dimensions selected:
parcel,
Cube Construction
Data cube constructed
Diagram OLAP Cube built
XML Schema Cube Built
Instances of Selected Cubes

nom_equipement	culture	parcelle	climat	temps	quantite_eau	import_quantity_m...
equi25	féte	par 32	clim 22	janvier	132	129
bidon	cul A6	parcelle Est	climat	janvier	5	3
char	évènement	par 21	climat sec	février	143	140
voiture	construction	par 8	clim 2	octobre	35	24
bateau	construction	par 32	climat sec	juin	753	691
suisse	féte	parcelle Est	c 20	octobre	87	86
car	évènement	par 29	r32	février	109	103
equip_34	construction	parcelle Est	cil zone 33	mars	66	66
chateau d'eau	cul A6	par 43	clim 2	avril	87	59
camion	construction	par 8	c 20	avril	39	39
dinde	évènement	par 32	clim 22	juin	41	40
bassin	féte	par 21	climat	mai	93	87

Fig. 11 Measure-based construction variant (instances)

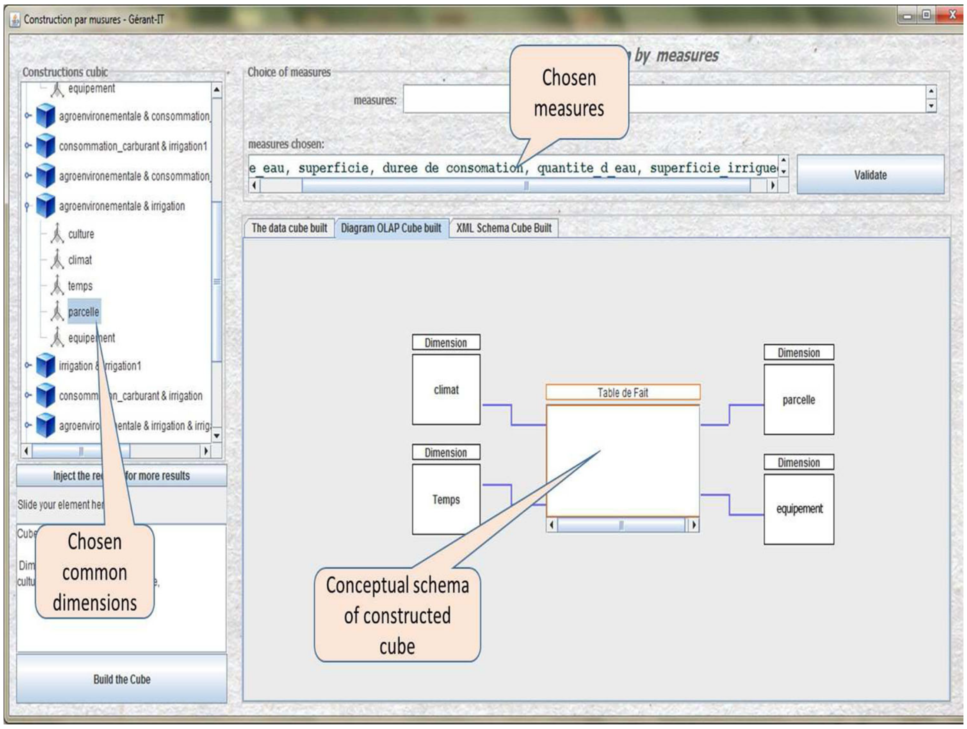

Construction par musures - Gérant-IT
by measures
Constructions cubic
equipement
agroenvironementale & consommation
consommation_carburant & irrigation1
agroenvironementale & consommation
agroenvironementale & irrigation
culture
climat
temps
parcelle
equipement
irrigation & irrigation1
consommation_carburant & irrigation
agroenvironementale & irrigation & irrig
Inject the request for more results
Slide your element here
Chosen common dimensions
Build the Cube
Choice of measures
measures:
measures chosen:
e eau, superficie, duree de consomation, quantite d eau, superficie irrigue
Validate
Chosen measures
The data cube built
Diagram OLAP Cube built
XML Schema Cube Built
Dimension
climat
Dimension
Temps
Table de Fait
Dimension
parcelle
Dimension
equipement
Conceptual schema of constructed cube

Fig. 12 Measure based construction variant (conceptuel model)

The system in this variant uses a search process to find the set of cubes that partially meet the need and that are candidates for merging. After the fusion, the system displays all the constructed cubes ordered according to the degree of relevance which was described in Sect. 5.4. The user can view and explore the cube that suits him (Fig. 11).

The tool offers the opportunity to see the conceptual model of the constructed cubes, in order to see the structural components of the new data cube (Fig. 12).

To validate our approach, we performed some experiments. For the needs of our tests, we consider a dataset composed of 50 cubes [8].

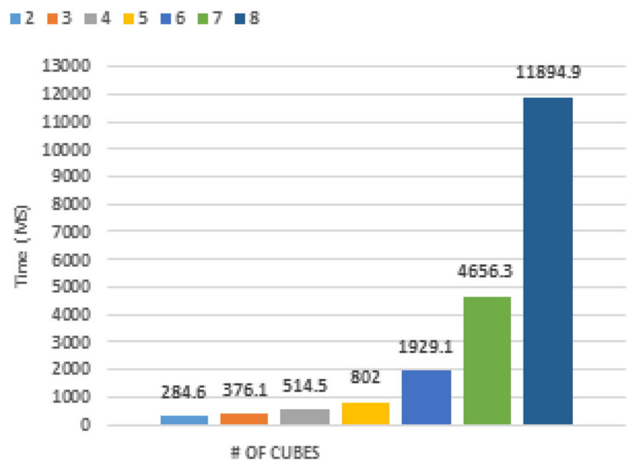


Fig. 13 Construction cost versus number of cubes

We want to analyse the construction cost of a cube with the number of cubes that are necessary to construct it. The construction process seeks partial cubes (i.e. that partially responding to the need), checks the conformity of dimensions, designs cubes, instantiates and displays their content for the decision makers.

We calculated the construction response time as a function of the number of partial returned cubes. The results show that when the number of cubes is reduced, the construction cost is negligible compared with the search time. From six partial cubes on, the construction becomes more expensive (see Fig. 13). This is due to the elapsed time for testing the conformity of the dimensions between the partial cubes and exploring the catalogue.

It should be noted that when the number of partial cubes that are required for the cubes construction increases, the cost of this construction increases. This confirms our hypothesis that was presented in the ranking process (Sect. 5.4) which sorts the resulting group cubes starting with the cube of a the group that requires the least number of partial cubes for the construction. This is due to the necessary construction cost which is not negligible when the number of cubes is important.

Let us now compare our approach cube construction cost with that of a naive approach. The naive approach seeks partial cubes (partially responding to the need) and displays them without fusion [13]. Our approach looks for partial cubes, checks the conformity of their dimensions, designs cubes, ranks them and instantiates and displays their content for the attention of the decision makers.

We compare the two approaches on real applications. We calculate the response time of each approach based on the number of returned partial cubes and we add the time needed for a decision maker to analyse the resulting cubes. To this end, we estimate that the analysis of a cube requires 1 min. Figure 14 shows the response time of both

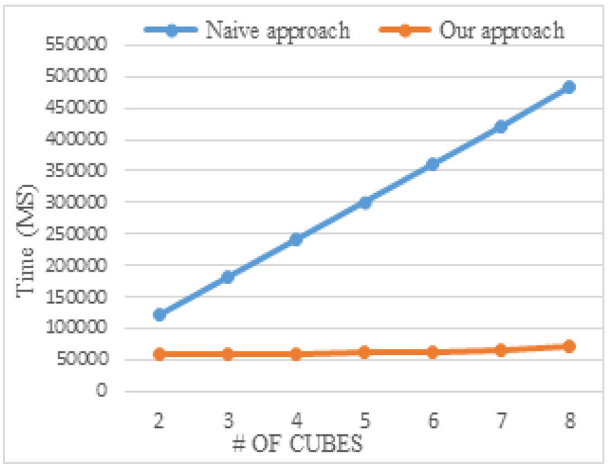


Fig. 14 Construction and analyse cost versus number of cubes

approaches by incorporating the analysis time of the resulting cubes. The results show that our approach is clearly better than the naive approach especially when the number of partial cubes increases. Indeed, our approach returns a single cube, which requires 1 min to analyse it, while the naive approach returns several partial cubes, which requires 1 min for the analysis of each cube.

7 Conclusion and future work

In this paper, we presented an approach for the construction of cubes from a set of existing cubes. This approach constructs new cubes according to a decision makers need, which is scattered over several data cubes. The approach aims to merge the cubes that containing only part of the decision makers need into one cube that integrates all the users need. The fusion of cubes is based on the conformity constraint between dimensions. For this, we used WordNet as an external resource to determine the similarity between dimensions. We have developed a tool called “Design-Cubes-Query” for the construction of relevant cubes in two variants, one per cube if the user wants to complete or enrich his need and another by measure if the need is accurate. As future work, we intend to use the notion of antonym to determine other semantic relationships between concepts. We propose to use domain ontologies instead of *WordNet* to determine the rate of similarity. We also plan to use DM techniques in the grouping step of our approach and integrate the users profile to better target the construction of cubes taking into account the decision makers preferences. Currently, many companies are opting for the *Cloud BI* solution. They deploy their data cubes in order to expand the use and exploitation of these cubes by several users. Our approach can be integrated as a service in the

Cloud BI technology to return a single cube containing the entirety of a users need.

References

1. Abelló, A., Samos, J., Saltor, F.: On relationships offering new drill-across possibilities. In: Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP, pp. 7–13. ACM (2002)
2. Alberto, D.: Fusion cubes: towards self-service business intelligence. Int. J. Data Warehous. Min. **9**(2), 66–88 (2013)
3. Bimonte, S., Sautot, L., Journaux, L., Faivre, B.: Multidimensional model design using data mining: a rapid prototyping methodology. Int. J. Data Warehous. Min. **13**(1), 1–35 (2017)
4. Boukraâ, D., Boussaïd, O., Bentayeb, F.: OLAP operators for complex object data cubes. In: ADBIS, pp. 103–116. Springer (2010)
5. Cheung, D.W., Zhou, B., Kao, B., Lu, H., Lam, T.W., Ting, H.F.: Requirement-based data cube schema design. In: Proceedings of the Eighth International Conference on Information and Knowledge Management, pp. 162–169. ACM (1999)
6. Chhabra, R., Pahwa, P.: Data mart designing and integration approaches. Int. J. Comput. Sci. Mob. Comput. **3**(4), 74–79 (2014)
7. Cohen, W., Ravikumar, P., Fienberg, S.: A comparison of string metrics for matching names and records. In: KDD Workshop on Data Cleaning and Object Consolidation, vol. 3, pp. 73–78 (2003)
8. Djiroun, R., Bimonte, S., Boukhalfa, K.: A first framework for top-k cubes queries. In: International Conference on Conceptual Modeling, pp. 187–197. Springer (2015)
9. Etcheverry, L., Vaisman, A., Zimányi, E.: Modeling and querying data warehouses on the semantic web using QB4OLAP. In: International Conference on Data Warehousing and Knowledge Discovery, pp. 45–56. Springer (2014)
10. Gardner, S.R.: Building the data warehouse: the tough questions project managers have to ask their companies' executives–and themselves–and the guidelines needed to sort out the answers. Commun. ACM **41**(9), 52–61 (1998)
11. Ghrab, A., Romero, O., Skhiri, S., Vaisman, A., Zimányi, E.: A framework for building OLAP cubes on graphs. In: East European Conference on Advances in Databases and Information Systems, pp. 92–105. Springer (2015)
12. Golfarelli, M., Rizzi, S.: A methodological framework for data warehouse design. In: Proceedings of the 1st ACM International Workshop on Data Warehousing and OLAP, pp. 3–9. ACM (1998)
13. Golfarelli, M., Rizzi, S., Biondi, P.: myOLAP: an approach to express and evaluate OLAP preferences. IEEE Trans. Knowl. Data Eng. **23**(7), 1050–1064 (2011)
14. Gomaa, W.H., Fahmy, A.A.: A survey of text similarity approaches. Int. J. Comput. Appl. (2013). https://doi.org/10.5120/11638-7118
15. Hung, E., Cheung, D.W., Kao, B.: Optimization in data cube system design. J. Intell. Inf. Syst. **23**(1), 17–45 (2004)
16. Hüsemann, B., Lechtenbörger, J., Vossen, G.: Conceptual Data Warehouse Design. Universität Münster, Angewandte Mathematik und Informatik (2000)
17. Islam, A., Inkpen, D.: Semantic text similarity using corpus-based word similarity and string similarity. ACM Trans. Knowl. Discov. Data **2**(2), 10 (2008)
18. Jindal, R., Taneja, S.: Comparative study of data warehouse design approaches: a survey. Int. J. Database Manag. Syst. **4**(1), 33 (2012)
19. Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. Wiley, New York (2011)
20. Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. Wiley, New York (2013)
21. Masuma, M.R., Losarwar, V.: Text classification and clustering through similarity measures. IJLTEMAS **5**(3), 91–94 (2016)
22. Nedelcu, B.: Business intelligence systems. Database Syst. J. **4**(4), 12–20 (2013)
23. Niemi, T., Nummenmaa, J., Thanisch, P.: Constructing OLAP cubes based on queries. In: Proceedings of the 4th ACM International Workshop on Data Warehousing and OLAP, pp. 9–15. ACM (2001)
24. Parimala, N., Pahwa, P.: Coalescing data marts. In: Proceedings of XVI International Conference on Computer and Information Science and Engineering, pp. 280–285 (2006)
25. Djiroun, R., Boukhalfa, K., Alimazighi, Z., et al.: A data cube design and construction methodology based on OLAP queries. In: 13th IEEE/ACS International Conference of Computer Systems and Applications, AICCSA 2016, Agadir, Morocco, pp. 1–8 (2016)
26. Riazati, D., Thom, J.A., Zhang, X.: Drill across & visualization of cubes with non-conformed dimensions. In: Proceedings of the Nineteenth Conference on Australasian Database, vol. 75, pp. 97–105. Australian Computer Society, Inc. (2008)
27. Rizzi, S., Abelló, A., Lechtenbörger, J., Trujillo, J.: Research in data warehouse modeling and design: dead or alive? In: Proceedings of the 9th ACM International Workshop on Data Warehousing and OLAP, pp. 3–10. ACM (2006)
28. Sabaini, A., Zimányi, E., Combi, C.: Extending the multidimensional model for linking cubes. In: EDA, pp. 17–32 (2015)
29. Bimonte, S., Schneider, M.: Merging spatial data cubes using the GIS overlay operator. J. Decis. Syst. **19**(3), 261–290 (2010)
30. Torlone, R.: Two approaches to the integration of heterogeneous data warehouses. Distrib. Parallel Databases **23**(1), 69–97 (2008)

Rahma Djiroun is a Ph.D. Student at the University of Science and Technology Houari Boumediene (USTHB), University of Algiers, Algeria. He received his Engineer and Magister Degree in Computer Science from the USTHB University of Algiers, Algeria, in 2002 and 2005, respectively. In 2005, he joined the Faculty of Electronics and Computing, USTHB University, as an Assistant Teacher. His current research interests include Data Warehouse, OLAP, Data Mining, Business Intelligence, Natural language, Cloud Computing. Recommendation Systems.

Kamel Boukhalfa is currently a Professor and Researcher at University of Sciences and Technology Houari Boumediene, Algiers (USTHB). He is a Member of the ISI Research Team at LSI Laboratory. He received the Engineer Diploma in Computer Science from USTHB in 1997, the Master Degree in Computer Science from the same university February 2002. He joined, in 2002, the Advanced Technology Development Research Center for the development of Tele-health Projects. He received the Ph.D. Diploma from Poitiers (France) and USTHB (Algeria) Universities in 2009. He received the accreditation to supervise research (HDR) from USTHB February 2011. His main research interests are data bases, data warehousing, Cloud Computing, data mining and optimization.

Zaia Alimazighi graduated from ENSEEIHT High School of Engineering in Toulouse in 1976, received her Ph.D. in Computer Systems from the University of Paris 6 in 1986, and her Doctorate in Computer Science from the University of Sciences and Technology (USTHB) in Algiers in 1999. She is a Full Professor in Computer Science at USTHB, Team Leader at the LSI Laboratory and Dean of the Electrical and Computer Science Faculty. Her current research concentrates on cooperative Information Systems modeling, inter-organizational Information Systems process modeling.