ASDeDaWaS: An Assistant System for the Design of Data Warehouse Schema

Nouha Arfaoui, Jalel Akaichi
BESTMOD
Higher Institute of Management
Bardo,Tunisia
e-mail: Arfaoui.nouha@yahoo.fr, Jalel.akaichi@isg.rnu.tn

Abstract—**Data Warehouse has the capacity to integrate data from different data sources for analyses purpose. Despite their importance, many data warehouse projects fail. As cause, we can mention, the poor communication between the developer/designer and the stakeholders, and the bad design that does not respond appropriately to the user requirements. Our work is set in the context of Enterprise Data Warehouse, and we propose a new methodology, Assistant System for the Design of Data Warehouse Schema (ASDeDaWaS). It ensures the design of the schema of the data warehouse taking into consideration the users' requirements and the available data sources, minimizing the computer-scientists intervention.**

Keywords-Data Warehouse Schema; Data Mart Schema; Schema Design; Schema Integration.

I. INTRODUCTION

Data Warehouse (DW) is the "heart of architecture environment and is the foundation of all decision support system processing" [7], since it provides an infrastructure that allows businesses to extract, clean and store vast amount of data. It is defined as "a subject-oriented, integrated, non-volatile, and time-variant collection of data in support of management's decisions" [17].

Concerning the warehousing projects, they are often characterized by their complexity and their huge costs [1] and they may fail during their achievements. According to [1][4][8][9][15], the causes of failure can be summarized as following:

- The nature of those projects requires long periods of development.
- The users' needs are generally poorly expressed by either designers or developers and they are not based on a common terminology.
- The absence of a good design that responds appropriately to the users' requirements.
- The users are, in many cases, not experienced with the technologies of DWs.
- The immaturity and complexity of the design methods and the lack of software tools that support these methods.
- The nonexistence of the right design that ensures the performance today and the scalability tomorrow.

The above difficulties lead to various problems such as the stopping of projects during their implementation, the exceeding of time and/or budget, and the rest.

In order to overcome the previous problems, we propose a new methodology, namely, **A**ssistant **S**ystem for the **De**sign of **Da**ta **Wa**rehouse **S**chema (ASDeDaWaS). It ensures the construction of the schema of the DW incrementally taking into consideration both the users' requirements and the available data sources. It focuses on each department separately, which facilitates the detection and the correction of possible problems and conflicts earlier. It reduces, also, the computer-scientists intervention through the automation of some tasks.

As working hypothesis, it is proposed to present the user requirement as a star schema because it is widely supported by a large number of business intelligence tools; also it has a simple structure, so it is easy to understand the schema. Concerning the data sources, it is proposed to deal with Entity-Relationship (ER) [14] database because it adopts the more natural view that the real world consists of entities and relationships; it incorporates some of the important semantic information and it can achieve a high degree of data independence [14].

As contributions, we propose in this work:

- Using an assistant system to facilitate the collection of users' requirements by exploiting the previous experiences.
- Using a new algorithm to cluster the schemas taking into consideration their semantic aspect.
- Automating the schema integration technique to merge the schemas to generate the logical schemas of the data mart (DM), and the final schema of the DW.

The outline of this work is as following:

- In the second section, we present the state of the art. We summarize some methods that use the mixed approach to design the DW.
- In the third section, we describe our proposed solution and we resume every step.
- In the fourth section, we start by detailing the first step that consists of collecting the users' requirements using an assistant system. The different requirements are modeled as star schemas.
- In the fifth section, the generated schemas are clustered using a new algorithm ak-mode which is an extension of k-mode. It takes into consideration the semantic aspect when clustering the schemas.
- In the sixth section, we propose the application of schema integration technique to ensure the

merging of different schemas existing within every cluster. The proposed technique is composed by schema matching and schema mapping.

- In the seventh section, we propose generating multidimensional schemas from Entity-Relationship (ER) databases.
- In the eighth section, we transform the conceptual schemas that were generated from the users' requirements to logical ones by adding the necessary information extracted from the multidimensional schemas. Using the logical DM schemas, we apply the schema integration technique to build the final schema of the DW.
- We finish this paper with a conclusion and future work.

II. RELATED WORK

Three main approaches have been proposed in the literature to conceive the DW: top-down, bottom-up and mixed.

Top-down starts from the description of the needs of all the users to construct the schema corresponding to the entire DW [5]. According to Ballard et al. [3], this approach has some disadvantages: it is a time-consuming process, it is difficult to collect the different agreement on the data definitions and business rules among all the different workgroups, departments, and lines of business participating. It can delay actual implementation, benefits and return-on-investment and it is length task.

Concerning the bottom-up, the construction of the global schema of DW starts from the different schemas of DMs that are built taking into consideration the requirements of the decision-making users responsible for the corresponding specific business area or process [5]. The problem with this approach is the redundancy and the inconsistency of the data between the DMs [3].

The mixed approach takes advantages of the two previous approaches [5]. It has the speed and the user-orientation of the bottom-up and the integration enforced by a DW in a top-down approach.

In the following, we present some work using the mixed approach to generate the DW and we start by "SelfStar" [6]. The proposed methodology is composed by four steps. It requires human intervention to validate the proposed schema until building the final DW schema. In what follows, we briefly present each step:

- First step: Extracting from the data source, that is expressed using the UML language, the candidate facts and showing them in the intermediate schema. Since the schema of the data source is not easy to be understood by a no-computer scientist user, the system presents a simplified representation automatically extracted from the schema of the source. The user selects the facts and the measures that correspond to his needs. He selects also the operations that will be applied on the measures.
- Second step: Generating the second schema by proposing the different dimensions that can be used with the extracted facts. The extraction of the dimensions is done using: a source described by an exploited schema containing classes and links, and an incomplete decisional base containing one or many facts.
- Third step: Generating a constellation schema (facts, dimensions, and all possible hierarchies). This step generates the candidate hierarchies for each chosen dimension. They are extracted from the classes that are related directly to the dimension using "1..N" link type, and from the attributes of the dimension excluding the attributes having distinct values. The temporal dimension is associated with a standard hierarchy: year, month, date.
- Fourth step: Generating the schema of the DW. In this step the decision makers choose the relevant parameters to their analysis. Then, the system generates the final schema and stores customization metadata for each user to reuse them later by attributing weights to the used classes of the source.

Romero and Abelló [10] propose an approach that uses the end-user information requirements which are expressed as SQL queries and the logical model of the data sources. The final result is a constellation schema. The automatic process is divided into four different stages:

- Concept labeling: It serves to build the multidimensional (MD) graph by applying the labeling standards. For each query, it extracts the MD knowledge.
- Multidimensional graph validation: Each MD-graph that has been generated in the previous step is validated in this stage by generating its multidimensional normal forms.
- Finding representative result: From the previous steps, more than one MD schema can be produced for a given query. Besides, the dimensional data could be considered as an alternative factless fact, although in most cases it will not be relevant to the end-user. This step serves to determine the representativeness of new alternatives and this is done according to some rules. Two sibling graphs differ only in the labeling of one node. Therefore, they have exactly the same labels except for one node, which is considered a factless fact that plays a role in one graph and a strict dimensional role in the other. In short, sibling graphs do not provide new interesting analytical perspectives. They are used to analyze the potential factual data that a dimension may contain. However, in most cases, the end-user would not be interested in this type of analysis.
- Conciliation: It validates each input requirement and generates a potential set of MD schemas for each query. Then, it normalizes MD graphs

Giorgini et al. [13] propose a mixed approach to build the DW. It starts with the requirement analysis that will be mapped next to conceptual level. This step requires the following tasks:

- Organizational modeling: It is centered on stakeholders. It identifies the facts. It is composed by three steps:
    - Goal analysis: Analyzing each goal of each actor in more details.
    - Fact analysis: Determining all the relevant facts and associating goals with facts.
    - Attribute analysis: Determining all the attributes that give a value when facts are recorded.
- Decisional modeling: It focuses on decision makers to extract their information needs. It is composed by four steps:
    - Goal analysis: Identifying the decision makers and establishing the dependencies between them.
    - Fact analysis: Identifying the facts that correspond to different objects of analysis and associating the goals.
    - Dimension analysis: Linking the fact to the dimensions according to the decisional goals of the decision makers.
    - Measure analysis: Associating the measures to each fact previously identified.

Once, they get diagrams that connect enterprise goals to facts, dimensions and measures, they move to the conceptual level by mapping the different elements determined previously. They use two types of frameworks: mixed design framework and demand-driven design framework.

- Mixed design framework: The requirements derived during organizational and decisional modeling are matched with the schema of the operational database in order to generate the conceptual schema for the DW. This is done by performing the requirement mapping, hierarchy construction and refinement.
- Demand-Driven Design Framework: The generation of hierarchies cannot be automatic; here we need the intervention of the designer. Indeed, through his skills and experiences, he can fruitfully interact with the domain experts to capture the existing dependencies between attributes.

Compared to the previous solutions, ASDeDaWaS follows all the necessary steps to generate the schema of the DW. It combines the mixed approach, which generates the logical schemas of the DM from the requirements and the available data sources, and the bottom up approach, which generates the DW schema from the DM logical schemas. Moreover, it offers help using an assistant system to facilitate the collection of the needs reducing the computer-scientists intervention. It can be applied to different departments having different information systems and different needs.

III. OVERVIEW OF ASDeDaWaS

In this section, we describe ASDeDaWaS briefly (Figure 1).

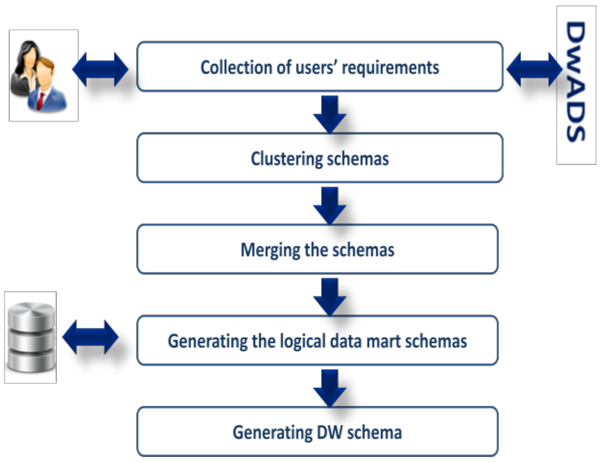

Collection of users' requirements
Clustering schemas
Merging the schemas
Generating the logical data mart schemas
Generating DW schema
DwADS

Figure 1. ASDeDaWaS steps.

It starts by collecting the requirements of the different users. It uses an assistant system DwADS (Data warehouse Assistant Design System) to facilitate the specification of the elements basing on the stored traces of the previous users. It defines, then, the possible facts, their measures, and the dimensions with their attributes. From the collected users' requirements, it generates the corresponding schemas that are represented as star.

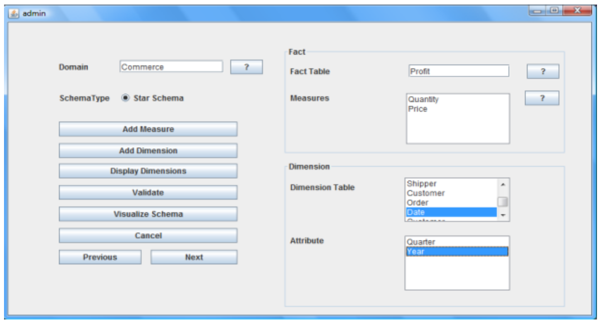
In the second step, it clusters the different schemas using a new algorithm ak-mode in order to get within one cluster the closest schemas. The new algorithm is an extension of k-mode. It takes into consideration the semantic aspect when making the comparison. Next, for each cluster, it generates the global schema. To achieve this goal, it uses the schema integration technique that is composed by schema matching and schema mapping. The schema matching extracts the semantically closest elements as well as the conflicts and presents them as mapping rules. Using the schema mapping it merges the different schemas to get at the end the global schema. This step allows the generation of conceptual schemas of the DM from the users' requirements. Then, the conceptual schemas are mapped to logical ones. This is done in two steps. In the first one, it extracts all possible multidimensional schemas from the databases. In the second step, it generates the logical schemas. Indeed, it updates the conceptual schemas by adding the necessary information extracted from the multidimensional schemas. Finally, by merging the logical schemas, it builds the final schema of the DW.

IV. COLLECTING OLAP REQUIREMENTS

In order to ensure a good design of DW, it is crucial to start by the requirements that specify "what data should be available and how it should be organized as well as what queries are of interest" [5]. In our case, we need to move through this step to extract the important objects of the multidimensional schemas (facts, measures, dimensions, and attributes). Despite its importance, not much attention has been paid to this phase causing the failure of 85% of the DW projects to meet business objects, and the no-development of 40% of the DW projects [12].

A. *The collection of requirements*

To collect the requirements, we give the freedom to the user to express his needs using an easy interface (Figure 2) where he specifies the different objects composing a star schema. This interface uses an assistant system DwADS that helps the user to choose the appropriate objects because the end users may find difficulties to specify their objects [5].



Domain: Commerce
SchemaType: ● Star Schema
Fact
Fact Table: Profit
Measures: Quantity, Price
Dimension
Dimension Table: Shipper, Customer, Order, Date
Attribute: Quarter, Year

Figure 2. The proposed interface to specify the users' requirements.

Once the user validates his schema, the set of the manipulated objects and the performed actions (add fact, create dimension, and the rest) are stored respecting their order over the time as a trace.

B. *The Proposed Assistant System*

Our assistant is based mainly on traces. Indeed, it starts by storing the traces of each user during his session, then, it suggests the useful elements after a comparison phase. The system extracts from the trace the objects through the use model and the actions through the observation model. Concerning the comparison step, it uses the episodes to detect the exact position of the user in order to extract next possible objects. This system occurs during the specification of requirements by suggesting to the user the possible elements used to build a first schema basing on the previous experiences.

DwADS performs two main tasks, as presented in Figure 3. The first one corresponds to the building of traces using the use model and the observation model. The second task is about exploiting the previous experiences using the episodes as a method of comparison. This task includes, also, suggestion of the possible next objects to manipulate.

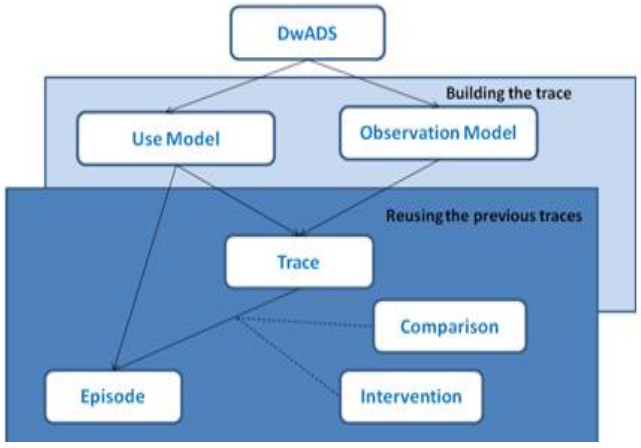


Figure 3. The DwADS composition.

The Use Model (Figure 4): It is used to isolate the objects from the current trace. The objects belong to the following categories (C): "C: Domain", 'C: Model", "C: FactTable", "C: Measure", "C: DimensionTable", "C: DimensionAttribute", and "C: Link". These various categories are linked into single schema through "Contextualization" link. The latter does not present the temporal aspect of the organization of different categories. It, only, shows them connected.

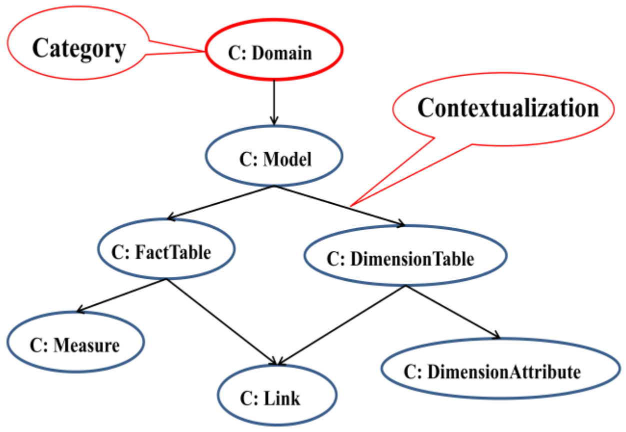
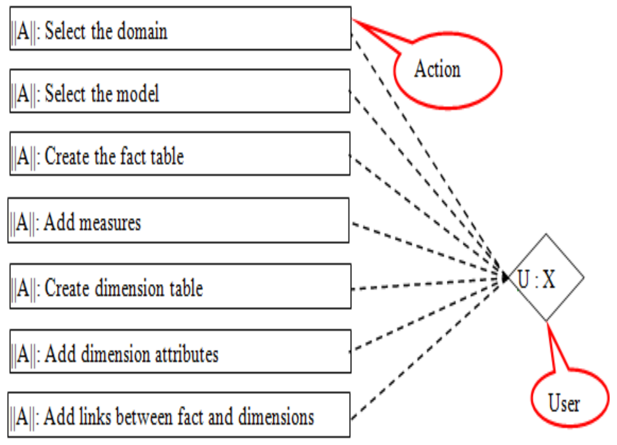


Figure 4. The structure of the use model.

For each requirement specification, the categories are instantiated which gives rise to many possible scenarios. For example the instantiation of "domain" can be "Commerce", "factTable" can be "Transaction", and the rest.

The observation Model (Figure 5): It encapsulates all the actions (||A: ||) handled by a single user during his session. It gives a vision on the use of the application and more precisely on how to deal with the existing objects extracted from the use model.



||A||: Select the domain
||A||: Select the model
||A||: Create the fact table
||A||: Add measures
||A||: Create dimension table
||A||: Add dimension attributes
||A||: Add links between fact and dimensions
Action
U : X
User

Figure 5. The observation model.

The observation model is instanced once the application is used. It gives different scenarios corresponding to the use of different objects. The scenarios present the actions used to instantiate the objects of the use model.

The trace is a succession of objects and actions over the time. It is built using the use model and the observation model. As example, in Figure 6 we have the trace corresponding to the creation of a star schema having one fact table “Transaction”, one measure “Gain” and three dimensions “Customer”, “Product” and “Seller”, with their attributes over the time.

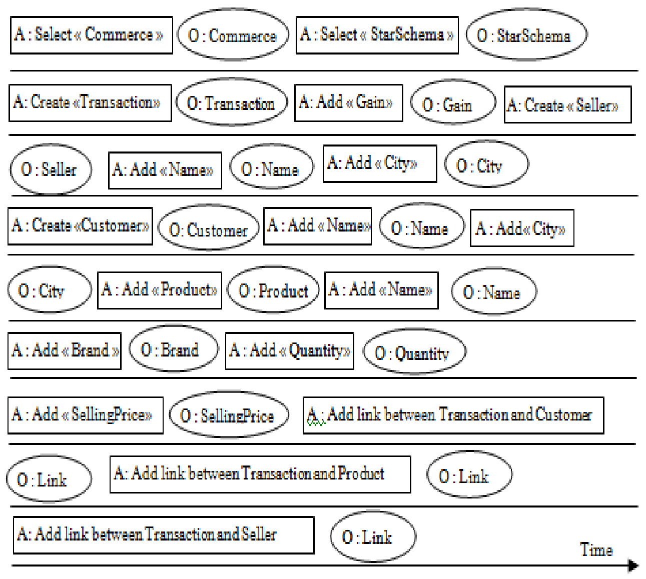


Figure 6. Example of trace corresponding to the construction of star schema.

Exploiting the previous experiences. Each new trace is stored. The set of existing traces in the database are exploited in order to assist the current user by extracting the useful objects. This exploitation is done through performing two tasks. The first one consists of comparing the trace of the current user with the previous traces in order to locate him. The second task is about making the necessary intervention by proposing the possible objects to use.

The comparison: To well exploit the previous traces, it is important to start by locate the user e.g., defining his last manipulated object to be able to predict the next possible objects. The location is done using the episodes that are extracted from the instantiation of the use model. For example, Figure 7 corresponds to three possible instantiation of the category “FactTable” that are “Transactions”, “Patient” and “Product”.

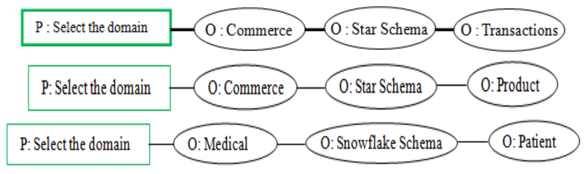


Figure 7. Example of episodes corresponding to the instantiation of the category “FactTable”.

Concerning the comparison, there are two cases:

- The system takes into consideration only the last manipulated object, example: “Transaction” (Figure 8).

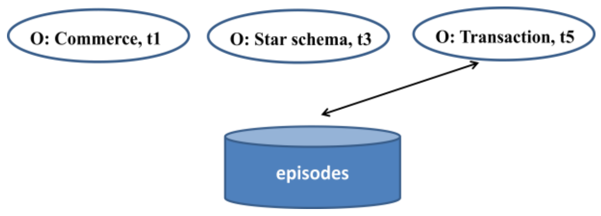


Figure 8. Example of comparing the last manipulated object.

- The system takes the whole trace into consideration, example: “Commerce, StarSchema, Transaction” (Figure 9).

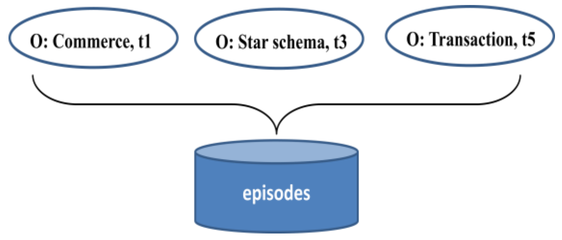


Figure 9. Example of comparing the whole trace respecting the order of objects over the time

.

The intervention: Once the system locates the user, it extracts from the database the set of traces containing the selected object or the set of ordered objects. The intervention can be done in two different ways:

- The system can intervene by suggesting one next object, example: “Seller”, “Country” and “Product” (Figure 10).

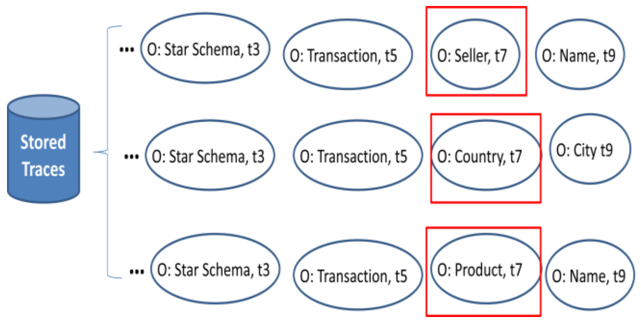


Figure 10. Example of intervention by suggesting one possible object.

- The system can intervene by suggesting the rest of the trace respecting the order of the objects, example: “Seller, Name”, “Country, City” and “Product, Name” (Figure 11).

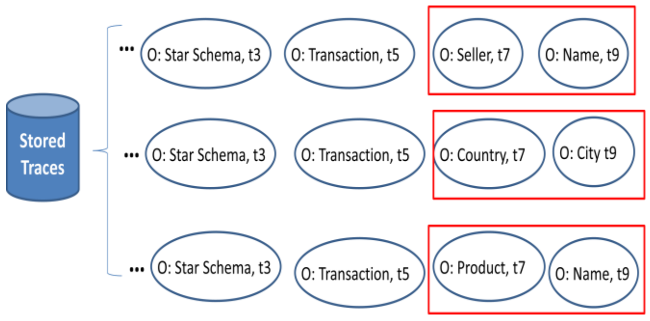


Figure 11. Example of intervention by suggestion the rest of trace.

C. *The structure of the generated schema*

At the end of this step, we get a set of schemas corresponding to the users’ requirements as example Figure 12.

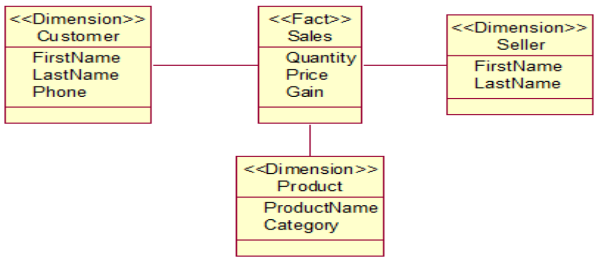


Figure 12. Example of user requirement presented as a star schema.

Each one is represented as star schema having the following structure:

- The fact table corresponds to the subject of analysis. It is defined by a tuple: **FN** and **MF { }** with:
    - **FN**: represents the name of the fact. e.g., “Sales”
    - **MF {m1, m2, m3, m4, …}**: corresponds to the set of measures related to the fact F, e.g., “Quantity, Price and Gain”.
- The dimension tables represent the axis of analysis. Each one is composed by: **DN** and **A{ }** with:
    - **DN**: corresponds to the dimension name, e.g., “Customer, Seller and Product”.
    - **A {a1, a2, a3, a4, …}:** presents the set of attributes describing the current dimension D, e.g., for the dimension “Customer” the attributes are “FirstName, LastName and Phone”.

V. CLUSTERING OLAP REQUIREMENTS SCHEMAS

At the end of the previous step, we get a set of schemas corresponding to the different requirements. In order to exploit them, we propose their clustering according to their domain using a new algorithm ak-mode that takes the semantics aspect into consideration.

Clustering is the unsupervised classification of patterns into groups called Clusters [2]. It involves dividing a set of data points into non-overlapping groups, or cluster of points [16], and this is exactly what we aim to do with OLAP Requirement Schemas (ORSs), i.e., grouping them with maximizing their similarity within one cluster and minimizing it between clusters.

The clustering proposes different algorithms. To choose the appropriate one, we compare them in term of “time complexity”, as presented in Table I.

TABLE I. CLUSTER ALGORITHMS FOR CATEGORICAL DATA.

Algorithm	Complexity	Coefficient
K-MODE	O (n)	Simple Matching
ROCK	O(kn^2)	Links
QROCK	O(n^2)	Threshold
COOLCAT	O (n^2)	Entropy
LIMBO	O (nLogn)	Information Bottleneck
MULIC	O (n^2)	Hamming measure

We can notice that the k-mode has O (n), which is the lowest complexity, cannot deal with our schemas because it does not take into consideration the semantic aspect of the elements; so, we extend it and we propose aK-Mode.

A. *The Extension of Simple Matching Dissimilarity Measure*

Let Sch1 and Sch2 be two schemas belonging to the same cluster.

Let Ci be the categories of elements existing in the schema with Ci = {fact, dimension, measure, attribute}.

When we calculate the similarity between the elements of the two schemas, we should take into consideration the following points:

- The identical: We use the same elements name in the two schemas.
  DeId (ei, ej) = 1 if ei and ej are identical and 0 if not.
- The synonymous: We use two different names that have the same meaning.
  DeSy (ei, ej) = 1 if ei and ej are synonymous, and 0 if not.
- The typos: We make mistakes when writing the name of the element. In this case, we calculate the degree of error. If it is low, we are in the case of typing error. If it is high we are in the case of two different words. In the following we only take into consideration the first case.
  DeTy (ei, ej) = 1 if ei and ej are the same with the existence of typing error, 0 if not.
- The post-fixe: We use post- fixes to design the same thing.
  DePost (ei, ej) = 1 if one the two elements is the post-fixe of the other, and 0 if not.
- The pre- fixe: We use pre-fixes to design the same thing.
  DePre (ei, ej) = 1 if one of the elements is the pre-fixe of the other, and 0 if not.

The degree of similarity between ei and ej (DeSim (ei, ej)) is measured by the numeric value {0} or {1}, and it is calculated as following formula (1):

$$\text{DeSim (ei, ej): Sch1 x Sch2} \rightarrow \{0, 1\}$$

$$\text{DeSim (ei, ej)} = [\text{DeId (ei, ej)} + \text{DeSy (ei, ej)} + \text{DeTy (ei, ej)} + \text{DePost (ei, ej)} + \text{DePre (ei, ej)}] \quad (1)$$

The new formula (2) of the simple matching (SM) dissimilarity measure is defined as following:

$$\text{Coef}_{SM}\text{ (sch1, sch2)} = [\ (\text{MaxD} - \text{CoefD}) / \text{MaxD}\ ] + [(\text{MaxM} - \text{CoefM}) / \text{MaxM}\ ] + [\ (\text{MaxF} - \text{CoefF}\ ) / \text{MaxF}] + [\ (\text{MaxA} - \text{CoefA}\ ) / \text{MaxA}\ ] \quad (2)$$

With:

- MaxD: It is the maximum number of dimensions existing in the two schemas.
- CoefD: It is the number of similar dimensions existing in the schemas using “DeSim”.
- MaxM: It is the maximum number of measures existing in the two schemas.
- CoefM: It is the number of similar measures existing in the schemas using “DeSim”
- MaxF: It is the maximum number of facts existing in the two schemas.
- CoefF: It is the number of similar facts existing in the schemas using “DeSim”
- MaxA: It is the maximum number of attributes existing in the two schemas.
- CoefA: It is the number of similar attributes existing in the schemas using “DeSim”

B. *The ak-Mode Algorithm*

The algorithm of aK-mode is described as following:
a) Define the ‘k’ number of existing domains.
b) Select ‘k’ initial modes. The initial modes correspond to the schemas that were selected randomly from each cluster.
c) Allocate a schema to the cluster whose mode is the nearest to the cluster, using the formula (2).
Update the mode of the cluster after each allocation.
d) After all schemas have been allocated to the respective cluster, retest the schemas with new modes and update the clusters.
e) Repeat steps (b) and (c) until there is no change in clusters.

VI. MERGING THE USERS’ REQUIREMENTS SCHEMAS

In this part, we generate the schemas of the DM from the existing clusters using the schema integration technique that combines the matching and the mapping. Compared to the others, our methodology does not require the pre-integration phase since the used schemas have the same model that was unified from the beginning.

A. *Schema Matching*

The schema matching is considered as one of the basic operations required by the process of data integration [11]. It is used to solve the problem related to the heterogeneity of the data sources by finding semantic correspondence between the elements of the two schemas. This phase is iterative; it takes two schemas as input each time to get as output a set of mapping rules in order to facilitate the merging task in the next step.

To ensure the effective schema matching, we focus on linguistic matching of names of schemas’ elements and according to Li et al. [18], it proceeds in three steps: normalization, categorization and comparison.

- Normalization: Different names design the same thing but they are written differently. They perform tokenization (e.g., parsing names into tokens based on punctuation, case, and the rest), expansion (identification of the abbreviation, acronyms, and the rest). So, we propose the use domain ontology, lenvenshtein name, and the rest.
- Categorization: It is to group the elements composing the schemas by categories: fact, dimension, measures, and attributes to reduce the number of one-to-one comparison eliminating the unnecessary comparisons.
- Comparison: A coefficient of linguistic similarity is calculated by comparing the tokens extracted from the names of the elements using the formula (1).

B. *Schema Matching Steps*

The schema matching serves to extract the mapping rules that will be used to facilitate the merging of schemas. Our proposed methodology is composed by the following steps:

- Categorization: It is to specify the category of each element. This can reduce the risk of error which provides a gain of time.
- Construction of the similarity matrix: It is about using a similarity matrix to find the closest elements. The cells contain the coefficient of similarity of the different elements belonging to the same category using the formula (1).
- Generation of the mapping rules: The rules visualize the conditional relationships between the instances of the categories. They are expressed as: **"If** Similar (X, Y) **then** Action (X, Y)**"**, with:
  - X and Y belong to the same category (fact, measure, dimension, or attribute).
  - Similar ( ): It is a function that specifies if the two inputs are similar or not.
  - Action ( ): It specifies the actions to perform. They can be union, or intersection.

The different rules are stored into rules database.

C. *Schema Mapping*

Once we extract the mapping rules; we move to the next where we apply those rules to merge the schemas. The schema mapping is a qua-triple M = (sch1; sch2; T; δ). "sch1" is the first schema, "sch2" is the second schema, "T" is the target schema, and "δ" is a set of formulas over <sch1, sch2; T>.

An instance of M is an instance of <s_1, s_2; t; δ_i> over <sch1, sch2; T; δ> that has a specific formula in the set δ_i. The formulas existing in δ_i correspond to one of the following functions:

- Union: R = union (ei, ej) implies that R contains all the components of "ei" and all components of "ej". It is applied when the two elements are identical.
- Intersection: R= intersection (ei, ej) implies that R contains the components that exist in "ei" and "ej". It is applied when the two elements are equivalent and not identical.

VII. GENERATING MULTIDIMENSIONAL SCHEMAS FROM DATABASE

In this section, we propose an algorithm to generate all multidimensional schemas from the data sources. This helps to construct the DM logical schemas by making the necessary modifications. We suggest working with Entity-Relationship (ER) model of the data source.

Our algorithm starts by extracting the potential facts and dimensions. For each fact, it extracts all possible measures. For each dimension, it adds the attributes.

Step 1: Normalize the ER model

Apply the 1NF, 2NF and 3NF to construct the ER normalized:

- First Normal Form (1NF): It is that there should be no nesting or repeating groups in a table.
- Second Normal Form (2NF): It is that the key attributes determine all non-key attributes.
- Third Normal Form (3NF): It is that the non-key attributes should be independent.

Step 2: Build the tree from ER model

From the ER model, we extract the entities (**Ef**) having n-ary relationships with other entities and those having numerical attributes. They represent the potential facts. Every **Ef** becomes to root of the tree. The number of trees corresponds to the number of **Ef** entities. From the ER, we extract the entities (**E**) that are directly linked to **Ef** corresponding to the potential dimensions.

Step 3: Transform the tree to multidimensional model

- The root of each tree becomes the fact table.
- The existing numeric attributes become the potential measures
- The measures are defined by an aggregation functions that are specified by the user.
- The nodes that are directly linked to the roots are transformed to dimensions keeping their attributes and their primary keys.
- The primary keys of the children nodes become foreign keys in the parents' nodes.

Figure 13 presents an example of multidimensional schema.

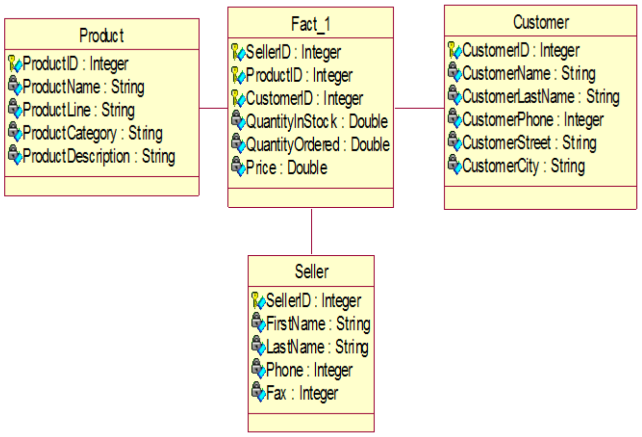


Figure 13. Example of multidimensional schema generated from an ER database.

It is composed by one fact table "Fact_1", three measures "QuantityInStock, QuantityOrdered, Price", set of keys defining the primary key of the fact table, three dimensions "Product, Customer, Seller". Each dimension has its primary key and a set of attributes.

VIII. THE DATA WAREHOUSE SCHEMA

In this section, we generate the schema of the DW. To realize this task, we need first to generate the logical schemas of the DM.

A. *Generating the Data Mart Logical Schemas*

The purpose of this step is to move from the conceptual schemas to the logical ones. At this level, we have two types of schemas. The first ones were generated from the requirements, they correspond to the **D**ata **M**art **U**ser **S**chemas (DMUS)s and they are modeled as star. The second ones were generated from the different databases, they correspond to the **D**ata **M**art **M**ultidimensional **S**chemas (DMMS)s and they are modeled as star schemas. The validation of DMUS is about adjusting the needs with databases so that we have the source from which we can extract data later.

In order to achieve this task, we compare the two types of schemas to extract the closest ones, then, we update the DMUS by adding the necessary information.

To compare the DM schemas, we start by classifying their elements into the following categories: fact, measure, dimension, and attribute. Using the similarity matrix, we extract the closest schemas. The updating task has as purpose transforming the conceptual schema to logical one. To achieve this task, we need human intervention. Indeed, we present the elements of two types of schemas and the final user specifies the necessary elements to keep. For example, Table II visualizes the elements extracted from Figure 12 and those extracted from Figure 13 to specify the elements of the final schema. For example, the attribute “FirstName”, extracted from the conceptual schema, has its corresponding attribute existing in the multidimensional schema. It has as type “String”. This attribute is added to the final schema with its type. The same process is applied to the rest of elements.

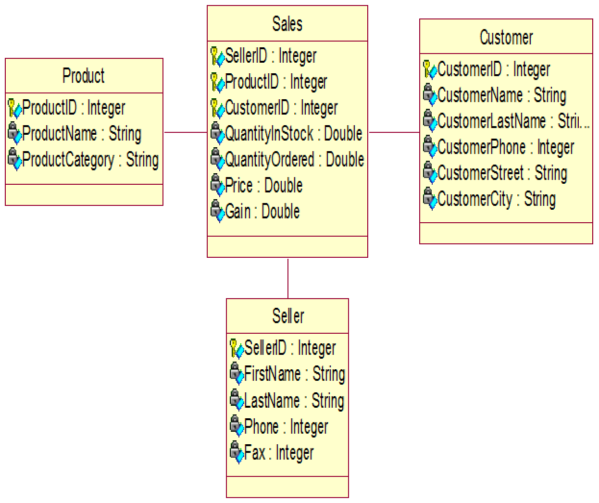


Figure 14. The logical schema of the Data Mart.

Figure 14 corresponds to the logical schema of the DM once the conceptual schema is updated.

B. *Generating the Data Warehouse Schema*

At this level, we have a set of logical DM schemas. To generate the final schema of the DW, we propose the use of schema integration technique as presented previously. It is composed by schema matching and schema mapping. This process is iterative. It takes every time two schemas as input. We stop when we get one final schema.

IX. CONCLUSION AND FUTURE WORK

The DW has the capacity to integrate huge amount of historical data for analysis purpose. It plays an important role with organizations. Despite their importance, many projects fail because of the absence of good design.

In this work, we proposed a new methodology to design the schema of the DW reducing the computer-scientists intervention. It takes into consideration the needs of each department separately to facilitate the detection of possible problems earlier, as well as existing databases to get at the end the best schema. Indeed, it starts by collecting the users’ requirements using an assistant system that exploits the previous experiences. Then, it clusters them using ak-mode to build first DM schemas that are generated using the schema integration technique. Besides, it revises those schemas to generate the logical DM schemas that serve at the end to build the final DW schema.

As future work, we propose dealing with other kind of data sources (UML, XML files, and the rest). We propose, also, taking into consideration the evolution of the schema of the database.

REFERENCES

[1] A. N. AbuAli, and H. Y. Abu-Addose, “Data Warehouse Critical Success Factors”, European Journal of Scientific Research, vol. 42 pp.326-335, 2010.

[2] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data Clustering: A Review”, ACM Computing Surveys, vol. 31, 1999, pp. 264-323.

[3] C. Ballard, D. Herreman, D. Schau, R. Bell, E. Kim, and A. Valencic, “Data Modeling Techniques for Data Warehousing”, International Technical Support Organization, 1998.

[4] E. Börger, “High Level System Design and Analysis using Abstract State Machines”, FM-Trends, 1998, pp. 1-43.

[5] E. Malinowski, and E. Zimanyi, Advanced Data Warehouse Design, From Conventional to Spatial and Temporal Applications, Springer Verlag Berlin Heidelberg, 2008.

[6] F. Abdelhédi, F. Ravat, O. Teste, and G. Zurfluh, “SelfStar: an interactive system for the construction of multidimensional schemas”, Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID), pp. 335-350, 2011.

[7] H. R. Nemati, D. M. Steiger, L. S. Iyer, and R. T. Herschel, “Knowledge warehouse: An architectural integration of knowledge management, decision support, artificial intelligence and data warehousing”. Decision Support Systems, vol. 33, Jun. 2002, pp. 143–16.

[8] M. Golfarelli, “From User Requirements to Conceptual Design in Data Warehouse Design”. In Data Warehousing Design and Advanced Engineering Applications Methods for Complex Construction, IGI Global, Hershey, 2009, pp. 1-16.

[9] M. Golfarelli, and S. Rizzi, “WAND: A CASE Tool for Data

Warehouse Design". In Demo Proceedings of 17th International Conference on Data Engineering (ICDE), pp. 7-9, 2010.

[10] O. Romero, and A. Abelló, "Automatic validation of requirements to support multidimensional design". Data Knowledge Engineering, vol. 69, pp. 917-942, 2010.

[11] P. A. Bernstein, and S. Melnik, "Meta data management". In Proceedings of the IEEE CS International Conference on Data Engineering. IEEE Computer Society, 2004.

[12] P. Andritsos, P. Tsaparas, R. J. Miller, and K. C. Sevcik, "LIMBO: Scalable Clustering of Categorical Data". In Proceedings of the 9th International Conference on Extending Database Technology (EDBT), 2004, pp. 123-146.

[13] P. Giorgini, S. Rizzi, and M. Garzetti, "Goal- oriented requirement analysis for data warehouse design". In Proceedings of the 8th International Workshop on Data Warehousing and OLAP (DOLAP), 2005, pp. 47 - 56 .

[14] P. P. S. Chen, "The Entity-Relationship Model-Toward a Unified View of Data". ACM Transactions on Database Systems, vol. 1, 1976, pp.9-36.

[15] S. R. Gardner, "Building the Data Warehouse". Communications of the ACM, Vol.41, 1998, pp. 52-60.

[16] V. Faber, "Clustering and the Continuous k-means Algorithm". Los Alamos Science, vol. 22, 1994, pp. 138-144.

[17] W. H. Inmon, "Building the Data Warehouse. John Wiley & Sons Inc, 2005.

[18] Y. Li, D. Liu, and W. Zhang, "A Generic Algorithm for Heterogeneous Schema Matching". International Journal of Information Technology, vol. 11, 2005, pp. 36-43.

TABLE II. TRANSFORMATION OF THE CONCEPTUAL SCHEMA TO LOGICAL ONE.

Category	**Conceptual level**	**Multidimensional level**		**Logical level**	
		Element	**Type**	**Element**	**Type**
Fact	Sales	Fact_1	-	Sales	-
FactKey	-	SellerID	Integer	SellerID	Integer
	-	ProductID	Integer	ProductID	Integer
	-	CustomerID	Integer	CustomerID	Integer
Measure	Quantity	QuantityInStock	Double	QuantityInStock	Double
		QuantityOrdered	Double	QuantityOrdered	Double
	Price	Price	Double	Price	Double
	Gain	-	-	Gain	Double
Dimension	Customer	Customer	-	Customer	-
DimensionKey	-	CustomerID	Integer	CustomerID	Integer
Attribute	FirstName	CustomerName	String	CustomerName	String
	LastName	CustomerLastName	String	CustomerLastName	String
	Phone	CustomerPhone	Integer	CustomerPhone	Integer
	-	CustomerStreet	String	CustomerStreet	String
	-	CustomerCity	String	CustomerCity	String
Dimension	Seller	Seller	-	Seller	-
DimensionKey	-	SellerID	Integer	SellerID	Integer
Attribute	FirstName	FirstName	String	FirstName	String
	LastName	LastName	String	LastName	String
	-	Phone	Integer	Phone	Integer
	-	Fax	Integer	Fax	Integer
Dimension	Product	Product	-	Product	-
DimensionKey	-	ProductID	Integer	ProductID	Integer
Attribute	ProductName	ProductName	String	ProductName	String
	Category	ProductCategory	String	ProductCategory	String
	-	ProductLine	String	-	-
	-	ProductDescription	String	-	-