Consolidation of BI Efforts in the LOD Era for African Context

Selma Khouri
Ecole nationale Supérieure d'Informatique, BP 68M,
16309, Oued-Smar, Algiers, Algeria
LIAS, ISAE-ENSMA France
s_khouri@esi.dz

Ladjel Bellatreche
LIAS/ISAE-ENSMA, France
Téléport 2 - 1 avenue Clément Ader - France
bellatreche@ensma.fr

ABSTRACT

During the last few years, we assist to spectacular increase of Business intelligence and analytics (BI&A) software revenue in the Middle East and Africa (MENA) totaled $245 million in 2014, a 12 percent increase from 2013 revenue of $219 million, according to Gartner, Inc. The Linked Open Data (LOD) era will contribute positively in keeping this dynamic. LOD datasets complete internal sources by new and relevant information for decision making. This integration in the data warehousing landscape has become a necessity. Recent studies conducted mainly in Europe have been proposed for this direction. Similarly, to first studies of conventional DW design, LOD driven approaches focused on data issues (like integration and multidimensionality) and ignored the importance of functional and non-functional requirements. This issue is a precondition for the success of BI&A projects in Africa. This continent is living an interesting phenomenon related to the multiplication of Open Data initiatives. Based on our experience on developing BI&A projects, our origin and knowing the European and African contexts, we propose a requirement-driven approach for designing semantic data warehouses from internal and LOD datasets, by considering requirements incrementally. All phases of our approach is formalized allowing a traceability of requirements. Experiments are conducted that show the impact of incorporating exploratory requirements of the target warehouse. A case study analyzing book sales transactions is given.

CCS CONCEPTS

• **Information systems** → **Database design and models**; **Storage management**;

KEYWORDS

Requirements engineering, African Context, Data Warehouses, Semantic Web, Linked Open Data

ACM Reference Format:
Selma Khouri and Ladjel Bellatreche. 2018. Consolidation of BI Efforts in the LOD Era for African Context . In *Proceedings of The Symposium on Software Engineering in Africa (SEiA2018).* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3195528.3195529

SEiA2018, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5719-7/18/05…$15.00
https://doi.org/10.1145/3195528.3195529

1 INTRODUCTION

The data technology industry is generating billions of dollars each year to efficiently manage available data. This industry is facing an opening of frontiers with the open data movement, where several entities (e.g., governments, education, development corporations and commercial enterprises) are looking for added-value. This objective is done by integrating and exploring internal and external sources freely published in the web. The Linked Open Data (LOD) movement, as part of the Big Data landscape, is a set of design principles for sharing machine-readable data on the Web, using semantic web standards for usage of public administrations, business and citizens[1].

The particularity of this global movement is that it has been rapidly adopted by developing countries. One of the main reasons of this adoption is the colossal efforts done by these countries on developing networked systems and Internet services for companies, organizations and citizen. In a such context, data become easier to generate and to collect. In African context, Open Data initiatives has been intensively promoted through different projects, like the Open Data Kit[2], a suite of tools that enables users to collect their own rich data. Fields of application of such project include decision support for pediatrics patients in Tanzania, documenting war crimes in the Central African Republic or HIV treatment program in western Kenya [2]. Another example comes from the Open Data for Africa platform[3], an initiative of the African Development Bank Group (AfDB), which is the first regional organization in Africa that has adopted an open data approach, to boost access to infographics data necessary for managing and monitoring development results in African countries[4]. Algeria is involved in this project through the open data portal http://algeria.opendataforafrica.org/. In 2012, the Open Data in Developing Countries project[5] is launched, to critically examine the impact of open data in these countries.

Different data-driven systems are facing this transition from closed, formal world towards open and networked world of organization. Business Intelligence and Analytics (BI&A) platforms are no exception to this transition phenomenon. In BI&A platforms, Data warehouses (DWs) are the data storage technologies that integrate, unify, manage, mine and transform data into strategic decisions. DWs are complex systems by the fact that they involve various resources (data, hardware components, storage media, programming languages, visualization tools, etc.), diverse actors (designers, analysts, architects, programmers, end users) and application software

[1] https://www.w3.org/DesignIssues/LinkedData.html
[2] https://opendatakit.org/about/deployments/
[3] http://www.afdb.org/en/knowledge/statistics/open-data-for-africa/
[4] https://www.europeandataportal.eu/sites/default/files/library/201302_open_data_in_developing_countries.pdf
[5] http://www.opendataresearch.org/emergingimpacts

accessing and updating data (commercial and Open Source software). Integrating the LOD in the landscape of DWs has become a necessity for African companies to add value to their business.

The study of this paper is motivated by our experience in the DW field, our origin (Africa) and the double cap of the authors in the sense she did her Ph.D. degree with joint supervision between Algeria and France. In her engineering school ESI in Algeria, the number of engineering projects[6] related to DWs has increased these last five years from 15% to 45% in 2018, proposed by private and public companies such as Sonatrach[7] and Ooredoo. One of the reasons of this explosion is the important student exchange studying DW between African countries and the Rest of the World. For instance, During her stays in France (ENSMA), she met different PhD students coming from Africa and working on DWs[8]. Some of these students went back to their home countries to develop this technology through their training, research and startups creation.

Faced to this favorable context, we could like to launch a think tank about the development of DW projects in Africa: *do we reproduce the European experience or do we adapt it to the African context?*
One of the main causes of DW projects failure is the user requirement management [9]. Reproducing the same experience represents a high risk in Africa. Based on our experience, we advocate the adaptation by integrating the local context. This integration passes mainly through the understanding of the functional and non functional requirements related to quality and efficiency of local services and applications.

The first task for the adaptation of DW findings is to analyze DW experience. In conventional DW design studies, two main approaches have been proposed: supply driven approaches that consider mainly data sources for constructing the DW system, starting from a detailed analysis of data sources to identify the relevant data to consider. Requirements definition and analysis have been overlooked in the first DW projects mainly because data warehousing emerged from industrial projects, which focused on implementation and optimization issues. Recognizing the importance of users' requirements, a second category of requirements-driven approaches appeared, that start from determining the information requirements of business users and map requirements to data sources. This category of approaches extended the design cycle by the requirements design phase. The design cycle established includes five main phases [26]: requirements definition, conceptual design, ETL, logical and physical design. The first phase conducts the whole design, and identifies all relevant data to consider according to users' requirements. The conceptual design phase provides an abstract view of the DW system, where concepts are organized into a multidimensional perception, which identifies a central concept to analyze (for example the sales of a company) according to different dimensions of analysis (such as suppliers, products, etc.). The Extract-Transform-Load (ETL) phase extracts data from sources and load them to the target DW schema according to defined mappings. The logical and physical design defines the implementation of the DW following a storage schema, a DBMS chosen and its deployment according to a given platform. The provided DW must satisfy a set of users' requirements. Demand-driven approaches bring requirements to the foreground, thus ensuring that the DW will be tightly tailored to the users needs [11].

Afterwards, semantic web technologies consolidated DW current research and innovation by providing *Value*-added DW systems. The global landscape of DW design using semantic Web technologies includes two main generations. The first generation of semantic DW systems have used domain ontologies to enrich the design process [15]. An Ontology is defined as an explicit and consensual specification of conceptualization [12]. In conventional DW systems from ontologies (internal to a company), a set of data sources are identified, an ontology (existing or constructed) is defined to assign semantics to data. Ontology-based appoaches for DW design also started by mainly considering sources, then different approaches followed and considered requirements. One main contribution of the authors in this field was the valorization of requirements engineering in DW field using semantic web technologies [14].

LOD environment revitalized DW field by bringing new approaches, new processes (ETQ for Extract-Transform-Query instead of conventional Extract-Tranform-Load), new terms (self-service BI), new vocabularies (QB and QB4OLAP). As any software project, the design process is always conducted by the fulfilment of user's requirements. However, this lesson seems to be forgotten in the new approaches facing the LOD environment, where requirements definition phase is omitted, and requirements are either ignored or assumed defined. In this paper, we try to turn experience and lessons learnt into value by considering SDW design under requirements perspective. We claim that requirements should be precisely identified, formalized and linked to following design artifacts. We show that the availability of semantic resources facilitates these tasks. We believe that the requirments definition issue is important for any warehousing project, but particularly for african projects, where both fileds (DW and Open Data) are focusing real attention.

To conduct our study, we claim that SDW design cycle should be revisited to consider challenges brought by the emergence LOD and external resources in the DW landscape. We propose a requirement-driven approach that supports this design cycle, that consists in selecting and automatically linking available data sources (internal and external) that best fit the user requirements. We analyze and demonstrate the impact of users' requirements on the whole design cycle, by considering BI&A benchmark (Star schema Benchmark) and LOD sources (Dbpedia[9] and Yago[10]). To cope with the different design scenarios (SDW design from scratch or existing SDW), we propose an incremental design that considers new requirements and integrate them into the SDW. The approach is illustrated in figure 1, it starts from projecting users' requirement on internal and external sources. A multidimensional schema defining each requirement is defined. The integration process starts by unifying MD schemas and matching them with existing MD schema (existing SDW). An ETL process is defined for integrating internal and external concepts. The ETL process unifies new concepts incrementally, the loading step is decided by the designer. This process is supported by an

[6] https://sites.google.com/a/esi.dz/pfe/home
[7] http://www.sonatrach.dz/
[8] https://www.lias-lab.fr/node/302?lang=en

[9] wiki.dbpedia.org/
[10] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/#c10444

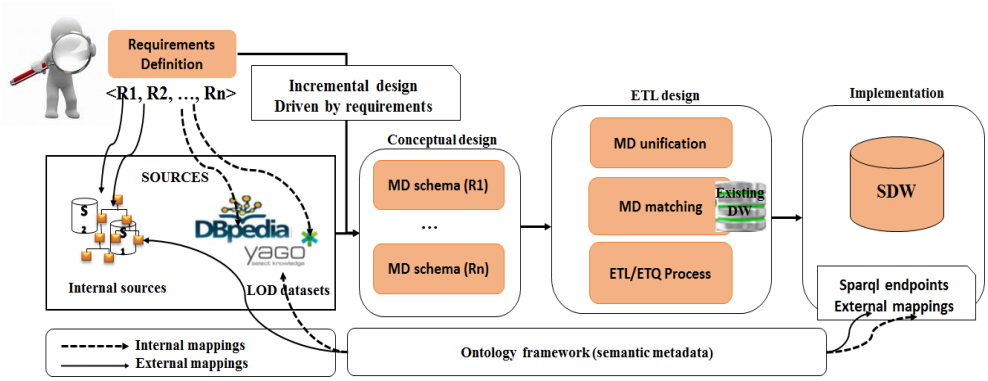

Figure 1: Proposed approach

ontological framework storing traces of the design decisions of each requirement.

This paper is organized as follows: section 2 presents the related works addressing DW design in the presence of semantic web technologies and particularly in LOD environment. Section 3 provides a formalization of SDW design revisited. Section 4 presents the ontological framework supporting the design cycle revisited. Section 5 presents the requirements-driven approach proposed, considering different construction scenarios. Section 6 illustrates the impact of requirements on the design cycle using a set of experiments. Finally, section 6 draws the conclusions.

2 RELATED WORK

The first generation of SDW systems considered ontologies during the design process. Ontologies have gained broad recognition in different domains and fields, thanks to their capabilities to representing explicitly the meaning of concepts and to reason on them. An ontology is defined as an explicit and consensual specification of conceptualization [12]. Conceptualization refers to an abstract model of some domain knowledge in the world that identifies that domain's relevant concepts [7]. Because ontologies satisfy the consensual criterion, the various conceptualization corresponding to various database schemas may be connected to a domain ontology. This sharing characteristic has been exploited by academician and industrials to define semantic data integration systems. Data integration systems are formalized as follows: $< G, S, M >$ where G is the global schema, S the set of sources and M are the mappings between G and S. In semantic DW design, G is represented by a domain ontology covering the set of sources. In a context of LOD environment, G is extended to include concepts borrowed from LOD schemas. Ontologies have been first used to facilitate the integration of sources in DW systems [6]. Their usage has been then extended to explicit requirements specifications [9], as a first level of conceptual design, to annotate the DW conceptual schema by their multidimensional role [14], to document the ETL design [24] or to automate it [25].

The second generation of SDW systems incorporate the use of LOD sources to provide exploratory analysis of data. LOD initiative aims to facilitate data sharing and enables data to be delivered in both machine- and human-readable formats. Linked Data is based on defining links (i.e. relationships) between related datasets of others in order to create new knowledge. XML-based schemas have been proposed as the technological paradigm of the last decades for facilitating data sharing[11]. However, when these schemas evolve, information systems using them (as data providers or data consumers) need to be adapted accordingly. To support the evolution of schemas over time without requiring all the data consumers to be changed, Resource Description Framework (RDF) emerged as the new paradigm for data exchange, and became a standard of linked data initiative[12]. DW approaches considering LOD datasets formed the second generation of SDW approaches.

We distinguish four categories of approaches for designing DW systems using LOD sources: (i) from scratch approaches which are approaches designing a new DW system by considering internal and external sources. (ii) Schema on write approaches, which consider an existing OLAP (including a DW) system and integrate new LOD data and materialize these data in the existing DW. (iii) Schema on query approaches, which consider an existing DW system and integrate new LOD data and do not materialize these data in the existing DW. The two last approaches often require an incremental ETL (for schema on write) - ETQ (for schema on query) process. (vi) Schema on read approaches which identify multidimensional structures on existing LOD data without considering a DW system. For example, [21] proposes a statistical framework to automatically discovering multidimensional conceptual patterns that summarize linked datasets. [20] proposes an OLAP framework to model heterogeneous semantic web data into multidimensional structures. [23] formalizes a multidimensional model in terms of RDF data structures following a conceptual constellation model. [17] defines common OLAP operations on QB data cubes. These approaches either rely on mediation systems with an OLAP layer (data are materialized on the web), or consider a DW structure but dot not deal with integration issues. We focus on studies considering a full DW structure.

Concerning "from scratch" approaches, [4] proposes a method for constructing a new SDW considering internal semantic sources and external data sources (Yago). The method generates a SDW

[11] https://joinup.ec.europa.eu/sites/default/files/inline-files/D4.3.2_Case_Study_Linked_Data_eGov.pdf
[12] https://joinup.ec.europa.eu/sites/default/files/inline-files/D4.3.2_Case_Study_Linked_Data_eGov.pdf

storing RDF triples. The set of concepts extracted from Yago forms an initial schema that is completed by concepts of local ontologies of sources. ETL operators are used to aliment the defined schema. [8] proposes a programmable semantic ETL framework from internal relational sources and linked data sources. The framework generates a semantic DW, composed of an ontology and its instances (RDF triples), where data are semantically connected with other internal and/or external. [16] proposes an approach to interpret Linked Data reusing QB as a multidimensional model and to automatically load the data into a data warehouse used by common OLAP systems. QB is a standard that allows to publish statistical data on the web using the RDF standard.

concerning "schema on write" approaches, [3] proposes to incrementally fetch and store data on-demand, i.e., as they are needed during the analysis process. The proposed approach considers an existing DW and external datasets (heterogenous like LOD, gene datasets or tweets). A dice abstraction of a cube is defined and used to identify missing facts. Based on the result of dice abstraction, an ETL process is defined in order to load external data. An optimization process is used to minimize the total cost of integration. [1] considers a DW system where missing data are identified, and completes these data by external datasets. The approach considers as sources an internal ontology (in RDF format) covering the internal sources and external LOD datasets formalized in QB4OLAP. QB4OLAP is an RDF vocabulary that covers multidimensional model components. The study focuses on the identification of functional dependencies from LOD datasets for building roll up hierarchies that are added to the existing DW schema. [10] proposes a multidimensional analysis framework using QB4OLAP over an existing ROLAP cube (schema and data).

Concerning "schema on query" approaches, [22] defines a framework for alimenting from LOD datasets an existing ROLAP cube where missing data are identified. A unified cube (UC) is defined to unify the existing DW schema and LOD datasets (chosen subset). The cube is not materialized, the framework helps to answer to queries on the fly when missing data are identified.

All these approaches do not consider the impact of requirements on the design cycle process, and do not consider all covered scenarios (design from scratch scenario and existing DW extension). Our approach proposes an incremental design which fits both scenarios, it covers the whole SDW design cycle under the requirements perspective.

3 SDW DESIGN CYCLE REVISITED

The formalization of the SDW design cycle revisited is based on the conventional design cycle, and a thorough analysis of existing studies considering LOD environment. Note that the framework formalization must contain the model-based and the process-based facets. We focus on the model-based aspect, since the process facet is mainly conducted by the design approach considered (detailed in the next section). We formalize the design cycle in a generic way to cover existing design efforts. The design cycle is formalized as follows: DW: <Sources, RM: Requirements modeling, CM: Conceptual Modeling, ETL: ETL modeling, IMP: SDW implementation using logical and physical Modeling>. The details of each component are described as follows:

- Sources: <Type, Schema, Instances, Formalism>. Sources are defined by their schema, a set of instances, a formalism (like relational or OWL) and a type. We distinguish five main types of sources: (i) internal conventional (usually structured), (ii) internal semantic (like local ontologies), (iii) internal conventional DW system, (vi) internal semantic DW system, (v) external semantic source (like QB vocabulary).
- RM: <Requirements, Relationships, Formalism>:
  - Requirements: is the set of requirements collected from users and validated. Following the modeling approach adopted, each requirement can be decomposed into elements. For instance, in the goal-oriented approach (which is the approach usually adopted in DW design), goals are characterized by an identifier, they are defined by results to achieve and criteria influencing the goal achievement $< Id, Result, Criteria >$.
  - Relationships: define different types of relationships between the goals, such as conflict, equivalence or refinement relationships.
  - Formalism: is the formalism used for analyzing the set of requirements (like textual, UML, etc.)
- CM: <C, R, I, L, Def(R), Pop(C), Multidim >, the conceptual modeling is based on the semantic model (internal ontology considered and/or subset of LOD schema). This model is thus based on the standards defining ontologies: Description Logic (DL), which is the formalism underlying OWL standard. In DL terminology, a knowledge base is composed of two components: the TBOX (Terminological Box) stating the intentional knowledge, also called the ontology schema and the ABOX (Assertion Box) stating the extensional knowledge or the instances. Formally, a $\mathcal{KB}$ is defined as 5-tuples: $< C, R, I, L, Def(R), Pop(C) >$, where:
  - C: a set of concepts of the TBOX
  - L: a set of literals (like numbers, string, or dates)
  - R: a set of relationships between concepts or between a concept and a datatype. For some $\mathcal{KB}$s, every literal is considered an instance of its datatype.
  - Def: R $\rightarrow$ C$\times$ (C $\cup$ L) corresponds to the relation definition, where Def(r) = (dom(r), ran(r)) for r $\in$ R. *dom(r)* is the source of r and *ran(r)* is the target of r.
  - Ref: C $\rightarrow$ (Operator, Exp(C,R)). Ref is a *function* defining terminological axioms of a DL TBOX. Operators can be inclusion ($\sqsubseteq$) or equality ($\equiv$). Exp(C,R) is an expression over concepts and relationships using constructors of description logics such as union, restriction, etc. (e.g., $\mathcal{R}$ef(Customer)$\rightarrow$($\sqsubseteq$, Person $\sqcap$ ($\exists$ hasOrder.Book))).
  - I: the set of instance (the ABOX)
  - Pop: C $\rightarrow 2^I$ corresponds to the concept instantiation.
  - Multidim: C $\cup$ R $\rightarrow$ Role. Multidim is a function that defines the multidimensional role of concepts and roles.

  When data are published on the web (LOD), they are represented as a set of triples or statements, following the Resource Description Framework Schema (RDFS) which is a W3C standard for knowledge representation and interchange. Each statement takes the form s $\in$ (U $\setminus$ L) $\times$ R $\times$ U, and usually for most statements s, s $\in$ I $\times$ R $\times$ (I $\cup$ L), such as U = I $\cup$ R $\cup$ L $\cup$ C [19]. Examples of statements from our case

study are: (customer#1, hasOrder, Book#1), (Customer#1, type, Customer), (Customer, subClassOf, Thing).

- ETL: <G, S, M>, where:
  - S is the set of sources are the first component formalized.
  - G is described by CM the conceptual model (as defined previously)
  - M is the set of mappings that must be identified. M is formalized as follows: <MapSchemaG, MapSchemaS, InputSet, OutputSet, Interpretation, Sense of mapping> where: *MapSchemaG* and *MapSchemaS* are respectively the mappable schema of the global schema and of the source schema. *OutputSet* is an element (usually a class) of MapSchemaG, and *InputSet* is an expression over source schemas. This expression is built using a set of ETL/ETQ operators, for example: aggregate operator, convert format operator, load operator (in case of ETL process), etc. Generic operators are detailed in [25]. Two main approaches define the sense of mappings: GaV (Global as View) approach where elements of target schema are described in terms of the source schemas and LaV (Local as View) approach where elements of source schemas are described in terms of the target schema. GaV variant is usually chosen because it is straightforward to populate the DW concepts.
- IMP: defines the implementation of the SDW, IMP is formalized as follows $<$ $PM_{Element}$, I, Pop, SM_{CM}, SM_I, Platform>:
  - $PM_{Element}$: the constructs of the physical data schema. For a SDW implemented using Oracle DBMS, it would be a table of triplet (*subject*, *predicate*, *object*).
  - I: presents the set of *instances* of the SDW system.
  - $Pop : PM_{Element} \rightarrow 2^I$ is a *function* that relates each physical element to its instances. Note that I and Pop function are obtained by the ETL algorithm, which aliments each class by its corresponding instances according to the defined mappings.
  - SM_{CM}: represents the *Storage Model* chosen for the CM. Note that if a semantic DBMS is chosen, it allows the storage of instances and also of their semantic schema.
  - SM_I: represents the *Storage Model* chosen for the instances part I_i, which can be the same as SM_{CM} (like in *Oracle* semantic DBMS [28]) or different from SM_{CM} (like in *IBM Sor* semantic DBMS [18]). In this case, two types of translation rules are used.
  - $Platform$: is the platform used for deploying the SDW. We focus in our approach on the centralized platform.

4 ONTOLOGICAL FRAMEWORK AS SUPPORT TO THE DESIGN CYCLE

Before presenting the steps of the approach, we first present the ontological framework underlying the design cycle formalization. The steps of our approach rely on this framework. The ontological framework defines the design cycle formalization at the ontology metamodel level. The ontology metamodel (corresponding to the CM component in our formalization) is extended by other components (Requirements, ETL, etc.) and form semantic metadata. We illustrate

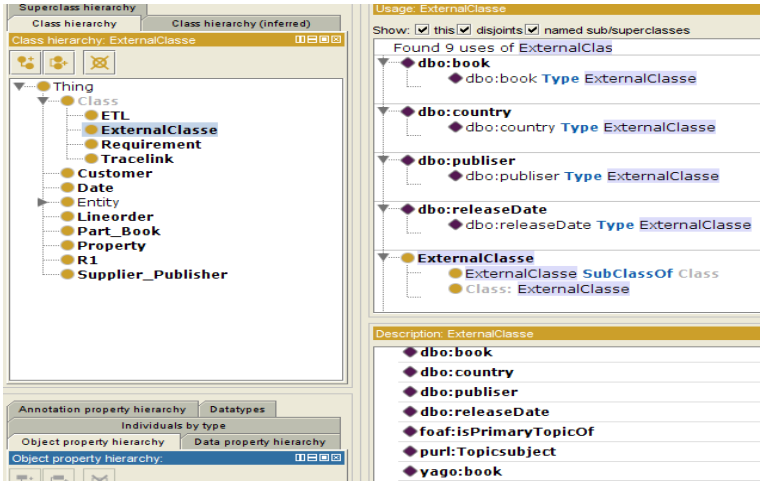

Figure 2: Extending the ontology meta-model (using Protege editor)

this process using Protege[13], an ontology editor broadly used in semantic web community.

Figure 2 illustrates this process. The meta-concepts are first made visible. For instance, the requirements model extends this meta-schema by the creation of a new meta-concept (*Requirement*). Each new requirement is defined as an instance of this meta concept. The ETL model extends the ontology model, using *Mapping*, *InputSet* and *OutputSet* meta-concepts. Note that in the ontology repository, Protege can store different ontology models. Each ontology has a unique URI (source ontology URIs, LOD dataset URIs and the DW ontology URI). This process is done for the whole design cycle components and forms a traceability model. This traceability model has been validated in [13]. Having this model allows retrieving traces when needed. For example, in case where data of a given requirement have not been materialized (ETQ and not ETL), time consuming tasks (like the multidimensional annotation and ETL processes) satisfying this requirement can be easily retrieved and re-executed without effort, if they are already stored.

For illustrating our approach, we use the following case study, based on Star Schema Benchamrk[14] (SSB), a benchmark dedicated to DW systems. We chose this benchmark for the following reasons: (i) SSB presents a set of (13) realistic OLAP queries that span the tasks performed by an important set of Star Schema queries and (ii) SSB provides a generic DW schema related to product sales that can be adapted to any case study. We chose to precise this schema for Book sales. We implemented this schema in Protegé Ontology editor, the dataset (instances) provided by SSB is considered as internal data.

A requirement example provided by SSB is *to provide revenue volume for lineorder transactions (sales of books) by customer nation and supplier nation and year within a given region, in a certain time period*. We adapted the set of requirements so that each requirement requires external resources. For example, for the previous requirement, the decision maker requires also publishers of

[13] https://protege.stanford.edu/
[14] https://www.cs.umb.edu/~poneil/StarSchemaB.PDF

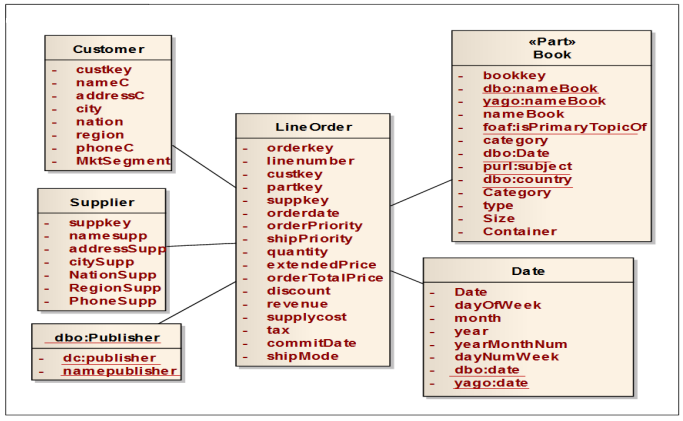

Figure 3: SSB schema extended

the books sold, and their country of origin. Different information are borrowed from external resources (mainly DBepdia[15] and Yago[16]) as illustrated in figure 2 (eg. dbo:book, dbo:country, yago:book, etc.). Based on the multidimensional annotation algorithm (that will be presented in the next section), external information are added to SSB model as new attributes of existing SSB classes, and as new classes (eg. *dbo:Publihser*). Note that DBpedia includes external datasets like FOAF vocabulary, the DBLP bibliography and the RDF Book Mashup. Figure 3 illustrates SSB schema adapted to our case study. The external resources are underlined, they are showed using the prefix of the dataset (dbo: for DBpedia, yago: for YAGO, foaf: for Foaf, ect.). The name of entities as provided in SSB are showed as stereotypes (eg. $< Part >$). In the figure, some attributes are considered as data-value attributes and other attributes (eg. suppkey in Lineorder concept) are object properties (target of the property is a concept).

5 SDW DESIGN : REQUIREMENTS DRIVEN APPROCH

The proposed approach presents a requirement-driven system for managing SDW design lifecycle in LOD environment. The proposed approach comprises four main stages: (i) Requirements definition, (ii) Conceptual design, (iii) ETL design and (vi) SDW implementation.

5.1 Requirements definition

The first design phase starts from requirements elicitation and analysis. In ontology based integration systems, three main architectures are identified [27] (i) unique ontology covering all sources, (ii) multiple ontologies where each source defines its local ontology. Ontologies are unified by matching mechanisms. (iii) Shared ontology where each source defines its local ontology from a shared and existent ontology. As DWs are considered as integration systems, SDW design followed these architectures, where a global ontology covers the set of defined data sources. The global ontology is either existing or constructed according to the architecture defined (the global ontology construction is out of the scope of this study). In LOD environment, the global ontology is extended by LOD schemas. We define the global ontology covering the sources as an internal ontology (IO), and we consider LOD schemas as a possible extension of IO. For the first stage of requirments definition, three main scenarios are identified for each Requirement Ri $\in$ Requirements (as defined in the design cycle formalization for RM):

(1) $Map(Ri) \subset IO$: the IO covers the requirements, all concepts needed by Ri are found in the internal ontology. Note that $Map(Ri)$ corresponds to the projection (mapping) of Ri on the ontology. As formalized previously, each requirement can be represented as <*Id*, *Result*,*Criteria*>. Each element of Ri must be satisfied by some concepts of the ontology. For instance, assuming that requirement R_1 analyses the sales of books by year. This requirement requires concepts *Book* and *Year*, it also requires the role *TotalSale*. For satisfying this requirement, the SDW must contain the data that are instances of these concepts.

(2) $Map(Ri) = IO$: the IO corresponds exactly to the users' requirements.

(3) $Map(Ri) \supset IO$: the IO does not fulfill all users' requirements. The designer will need to extend the IO to satisfy all her/his requirements. In a conventional SDW design, the concepts required by requirements and that are not found in the IO are either rejected or stored with null values. In a context of LOD environment, missing concepts are explored in LOD schemas. We refer to concepts explored from LOD datasets as O_{LOD}, which is the subset of concepts identified from LOD schemas for satisfying Ri. The identification of LOD concepts is based on the exploration of LOD schemas using visual interfaces. Most LOD datasets propose explorer interfaces to identify concepts, for example Dbpedia browser[17] or Yago browser[18].

Note that this approach is available even when existing SDW is available, and when new or exploratory requirements are required. In the case of designing a new SDW from scratch, the subset of concepts from $Oi \cup O_{LOD}$ forms the target SDW schema. In the case of an incremental design, where the SDW is already implemented, O_{LOD} will extend the SDW ontology schema, either by new concepts or by new instances (in case where the concept is already existing but does not cover all necessary data).

5.2 Conceptual design

The conceptual design phase identifies the schema required for satisfying the set of requirements. The multidimensional structure of this schema is also identified. To fulfill users' requirements, the identification of multidimensional concepts is based on the analysis of requirements. Considering the proposed structure of requirements $<$ *Id*, *Result*, *Criteria*$>$, we consider 'result' concepts as central concepts to analyze (the facts) and 'criteria' concepts as dimensions for the fact. Additional dimensions are identified based on their relationships with fact concepts, by considering many-to-one relationships.

[15] wiki.dbpedia.org/

[16] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/#c10444

[17] http://dbpedia.org/sparql

[18] https://gate.d5.mpi-inf.mpg.de/webyagospotlx/Browser?entityIn=concept&codeIn=eng

The multidimensional role of concepts and properties are discovered and stored as ontological annotations. The annotation concerns only new requirements (design from scratch or new exploratory requirements), and non-annotated concepts. In case of an incremental design, some concepts may appear in new requirements, we conserve their initial annotation (since it corresponds to other requirements). We propose algorithm 1 for multidimensional annotations. The result of this algorithm is a multidimensional schema representing each given requirement. For instance, the result of applying the algorithm on SSB requirements provides the multidimensional model illustrated in figure 3.

Algorithm 1 Algorithm for multidimensional annotation

Inputs: a set of *Requirements*, MappingList $Map(R)$.
Output: Multidimensional schema for each requirement (extracted from $Oi \cup O_{LOD}$ (that we call $\mathcal{ODW}$)

```
/* For each new requirement Ri, identify the corresponding concepts and annotate them */
for all R_i ∈ Requirements do
    Elemt := Identify-Result-Element (O_ODW, R_i);
    if Elemt ∈ C (/*Elemt is a Concept*/) then
        ODW := Add-Annotation (Elemt, Fact);
    else
        if Elemt ∈ R (/*Elemt is a Role*/) then
            ODW := Add-Annotation (Elemt, Fact-Attribute);
        end if
    end if
    Elemt := Identify-Criteria-Element (O_ODW, R_i);
    if Elemt ∈ C (/*Elemt is a Concept*/) then
        ODW := Add-Annotation (Elemt, Dimension);
    else
        if Elemt ∈ R (/*Elemt is a Role*/) then
            ODW := Add-Annotation (Elemt, Dimension-Attribute);
        end if
    end if
    /* Identify new dimensions depending on relationships roles (in Oi and O_LOD schemas)*/
    for all C_i ∈ Facts do
        Rel_Ci := ObjectProperty(C_i);
        if maxCardinality.OnObjectProperty(P_Ci) ≡ 1 then
            Dimension_List := Class(ObjectProperty(Domain(C_i) or Range(C_i)));
            ODW := Add-Annotation(Dimension_List, Dimension);
        end if
    end for
    /* Extract the subgraph from Oi ∪ O_LOD that elements are annotated */
    ODW := Extract-Schema (Oi ∪ O_LOD, Elemt_annotated ).
end for
```

5.3 ETL design

the ETL design is formalized by $< G, S, M >$ triplet, where G is the target schema (i.e. the global schema is the unification of multidimensional schemas obtained previously), S is the set of sources and the mappings between these schemas. Two scenarios are identified:

- Design from scratch: the ETL process is executed in order to aliment this schema from the set of sources using defined mappings.
- Incremental design: in case where an existing multidimensional schema, the ETL process must first include fact and dimension matching, before executing the ETL process. Note that this scenario corresponds to the case where requirements are not satisfied by internal sources ($Map(Ri) \supset IO$) and require the exploration of LOD datasets. The necessary concepts identified may match with existing concept (in the DW) but complete their attributes (for example, dateOfPublication for books). A matching process is necessary to identify overlap concepts. We detail this scenario since the first scenario is included in this one.

5.3.1 Unification of multidimensional schemas. This first step integrates multidimensional (MD) schemas of requirements. The unification identifies similar facts and unifies them in the same schema (fact shared by the set of dimensions of MD schemas concerned). If one dimension is shared between different MD schemas, they form the so-called constellation schema.

5.3.2 Fact and Dimension Matching. This step allows matching facts, matching dimensions, and complementing the MD schema of the SDW. This process identifies the overlap between concepts of the requirements MD and concepts of the SDW MD (that are already materialized). We start by Facts which are the core MD concepts. The approach looks for potential matches between facts in the current MD and the DW MD schema at hand. The matching is based on the names of concepts and attributes. Since the approach is defined at the ontological level, semantic matching can be used using synonymy and equivalent relationships. The same process is achieved for dimensions. After matching facts and dimensions, the approach conforms the dimensions producing the MD space of the merged fact. The final MD schema is finally obtained first by groupings concepts containing an equivalent MD knowledge. A set of external mappings (Map_{LOD}) is thus identified that will be used in the next step. If no matching is identified for an external concept, it is considered as a new concept that must extend the SDW MD schema. Note that the obtained MD schema is not materialized yet. Nevertheless, the complete information starting from requirement to MD schema is preserved in the ontology metamodel to assists further integration steps. The materialization process is decided by the designer.

5.3.3 ETL process. The ETL algorithm is executed using mappings defined between obtained MD schema and the source schemas. The set of mappings is discovered in two steps:

- if a concept of requirement R_i is satisfied by internal ontology (IO), the IO covers the sources and includes identified mappings Map_{IO}. We proposed and validated ETL algorithm based on ontological internal resources defining different integration scenarios [5].
- if a concept of requirement R_i is satisfied by external resources, we use the external mappings identified in the previous matching step Map_{LOD}. Note that we consider mappings following the GaV approach where concepts of the target schema are defined in terms of concepts of sources (which is the most followed approach in DW design). New mappings are identified for unifying LOD datasets into the SDW MD.

The proposed ETL algorithm includes three steps: extraction, transformation and loading. The extraction of instances from LOD datasets is achieved using Sparql endpoints proposed by most known LOD datasets. For example, the extraction of instances of Books from Dbepdia dataset is executed using the following Sparql query on the following endpoint http://dbpedia.org/sparql:

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX
rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?s ?p WHERE {
?s rdf:type  <http://dbpedia.org/ontology/Book> ;
```

```
   dc:publisher   ?p .
}
```

The following example illustrates the extraction of books, their publishers, date publication and country of origin:

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX  foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?a ?d ?p ?t ?co WHERE {
?a rdf:type  <http://dbpedia.org/ontology/Book> ;
<http://dbpedia.org/ontology/releaseDate>   ?d ;
dc:publisher   ?p ;
foaf:isPrimaryTopicOf   ?t ;
<http://dbpedia.org/ontology/country>   ?co . }
```

These endpoints provide interfaces for querying LOD datasets using Sparql, which is the standard query language of LOD initiative. We invoked these endpoints from java program implementing the ETL process. The results of the query can be retrieved in RFD format or another format like JSON or XML.

The transformation step is based on ETL operators [25] defines ten generic conceptual operators typically encountered in an ETL process, which are:

(1) **EXTRACT(C,S)**: extracts, from incoming record-sets, the appropriate portion; (2) **RETRIEVE(C)**: retrieves instances associated to the class C from source S; (3) **MERGE(C,C')**: merges instances belonging to the same source; (4) **UNION (C,C')**: unifies instances whose corresponding classes C and C' belong to different sources S and S' respectively; (5) **JOIN (C, C')**: joins instances whose corresponding classes C and C' are related by a property; (6) **STORE(S,C, I)**: loads instances I corresponding to the class C in target data store S, (7) **DD(I)**: detects duplicate values on the incoming record-sets; (8) **FILTER(C,Cond)**: filters incoming record-sets, allowing only records satisfying condition Cond; (9) **CONVERT(C,C')**: converts incoming record-sets from the format of the element C to the format of the element C'; (10) **AGGREGATE (C, F)**: aggregates incoming record-set applying the aggregation function F (COUNT, SUM, AVG, MAX) defined in the target data-store.

The loading step is decided by the designer. We provide in the experimentation section some relevant indicators for helping the designer during this process. The ETL algorithm is formalized as follows in 2.

5.4 SDW implementation

When considering a 'from scratch' design scenario, this step should include the choice of a semantic DBMS for storing the SDW. For the second scenario, the SDW is already implemented. The loading concerns new data. This process (in both scenarios) requires the translation of ETL process into physical workflow. This translation consists in translating conceptual ETL operators into SPARQL queries. Each query result is represented through an RDF dataset. The following example translates the Aggregate operator. For having the total number of books contained in Yago dataset, the following query is executed on yago sparql endpoint https://linkeddata1.calcul.u-psud.fr/sparql:

```
PREFIX
```

Algorithm 2 Semantic ETL Algorithm

Input: $\mathcal{SDW}$ MD (schema), Map_{IO}: set of internal mappings, LOD dataset (subset schema considered).
Output: $\mathcal{SDW}$ (schema + instances).

for all Mapping M $\in Map_{IO}$ **do**
  Execute mappings
end for
for all Class C $\in C_{LOD}$ (/*C_{LOD} are external concepts identified in the final MD schema*/) **do**
  identify target class (C_{DW}) (*/*identified in the previsous step "Fact and Dimension matching" */*)
  Instances (C_{DW})= RETRIEVE Instances (C_{LOD}) /* Retrieve instances associated to C_{DW} */
  if format of C is different from format of C_{DW} **then**
    CONVERT (C, C') /* Convert instances from format of C to that of C_{DW} */
  end if
  if C_{DW} presents aggregation constraint defined by F **then**
    AGGREGATE (C,F) /*Aggregate instances of C using function F*/
  end if
  if C_{DW} presents filter constraint **then**
    FILTER (C_{DW}, Condition) /* Filter instances of C_{DW} depending on Conditions */
  end if
end for
if more than one class Cs_{LOD} is identified in the same LOD source **then**
  MERGE (Instances (Cs_{LOD})) /* Merge instances associated to Cs_{LOD}*/
end if
if classes Cs_{LOD} have the same super class **then**
  UNION (Instances (Cs_{LOD})) /* Unify instances incoming from other sources */
end if
if classes Cs_{LOD} are related by same property **then**
  JOIN (iNSTANCES (Cs_{LOD})) /* Join incoming instances*/
end if
if (exists Instances(C_{DW}).duplicate) $\vee$ (Instances(C_{DW}).NullValues) **then**
  Filter(C_{DW}, DD(Instances(C_{DW}))
  Filter(C_{DW}, NullValues(Instances(C_{DW}))
end if
if Loading validated for C_{DW} */* Loading is decided by the designer */* **then**
  Store (C_{LOD}, C_{DW})
end if

```
rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://yago-knowledge.org/resource/>
select  (count(?p) AS ?c) "
where {
?p  rdf:type  :wordnet_book_106410904}
LIMIT 100
```

6 EXPERIMENTS

In this section, we carry out some experimental assessments to show the added value of considering requirements in the SDW design process.

Our experiments are based on SSB benchmark as illustrated in section 4. SSB schema is extended by external resources from DBpedia and Yago as illustrated in figure 3. We used the set of SSB queries as requirements using the ontology metamodel. Each requirement is linked to the set of internal concepts and the set of external concepts required for satisfying the requirement. For example Requirement (R#1) requires internal concepts (*Date*, *LineOrder*, *Year*, etc) and requires external concepts (*dbo:book*, *yago:book*). The ETL process integrating internal and external concepts is defined as instances of ETL metaclass, and linked to the required requirement using internal and external mappings. The sparql query translating the ETL operators is defined in the query attribute of each ETL operator.

Our evaluations were performed on a laptop computer (HP 650) with an Intel(R) CoreTM i3-2328M CPU 2.20 GHZ and 6 GB of RAM and a 500 GB hard disk. We use Windows10 64bits. We use Oracle Database 12c release 2 that offers RDF semantic features to store the SDW.

To conduct the experiments, we started by implementing in a JAVA application the design steps detailed in the previous section

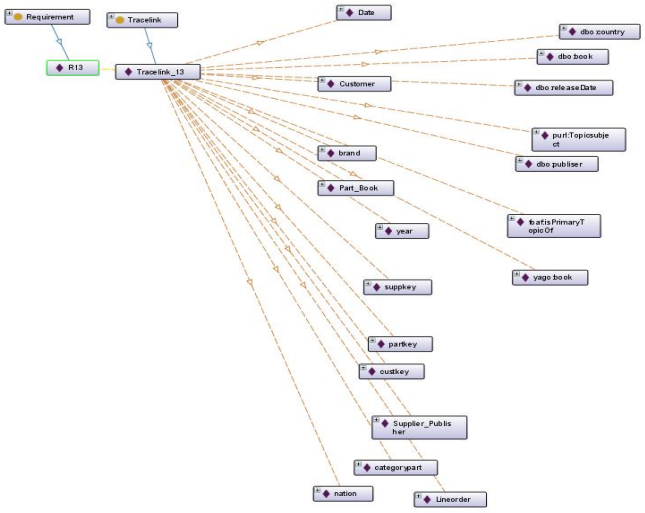

Figure 4: Impact of an exploratory requirement on a SDW schema

(multidimensional annotation and the ETL algorithms). The access to ontologies and external resources is achieved using JENA[19] API. The external resources are extracted from their datasets using their defined sparql endpoints.

We conducted three types of experiments to illustrate the impact of incorporating new exploratory requirements on the SDW. The first experiment illustrated in figure 5 (left part) illustrates the impact of exploratory requirements at the *schema* level. It shows the amount of nodes (internal and external) and the amount of mappings involved in each requirement. This first experiment identifies which requirements impact the most the SDW schema. For example, requirement (R4, R11, R12 and R13) have bigger impact on the SDW schema, and must be treated more carefully. For calculating the amount of nodes, we cumulated the set of traces using the metamodel defined in the different design levels (concepts, attributes, relationships, ETL elements, etc.). We did not consider the instances of classes in order to have an idea of the impact of each requirement at the schema level. Using protege, the designer can visualize the impact of each requirement with the detailed set of concepts. For instance, figure 4 illustrates the traces of requirement (R13) on the SDW schema, using internal and external nodes. This graph is defined using Ontograph (visualization plugin in Protege editor). We used the following reasoning rule to propagate the traces from an input node (for example requirement Ri) to the rest of nodes. The rule defines a transitivity rule indicating that: *if object x is related to object y and object y is in turn related to object z, then object x is also related to object z.*

```
Source(?T1, R1), Source(?T2, ?Y),
target(?T1, ?Y), target(?T2, ?Z)
 -> Source(?T2, R1), target(?T1, ?Z)
```

The second experiment (figure 5, right part) illustrates the impact of exploratory requirements at the *instance* level. For each requirement, the algorithm calculates the number of RDF triples to be

[19]https://jena.apache.org/documentation/ontology/

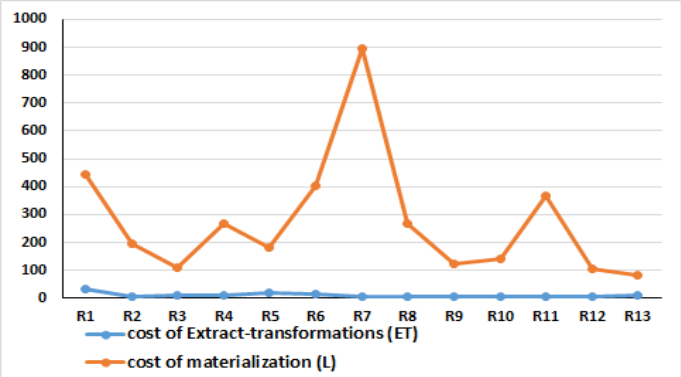

Figure 6: Impact of incorporating requirements: execution time of Extract-Transform and Load

inserted and the number of triples inferred. Oracle provides different reasoning mechanisms for identifying new triples based on existing ones. We used "RDFS" rule base for the reasoning mechanism. This information can guide the designer for having the impact of incorporating new requirement on the data of the SDW.

The last experiment (figure 6) illustrates for each requirement: the execution time of extracting-transforming concerned data and then the execution time of loading data. The figure shows that the cost of materialization can be important for some requirements. The execution time is not exactly correlated with the number of triples, since some requirements require data that need more inferences (with existing data), which increases the execution time of reasoning and thus the total time of materialization. The designer may decide to not materialize data of those requirements which materialization cost is important and which ET (extract-transform) time is low. The experiments provide the designer relevant indicators on the impact of incrementally adding new requirements in the SDW.

7 CONCLUSION

In this paper, we have launched a think tank about the reproduction vs. adaptation of the experience of constructing DW project in the Linked Open Data Era. Based on our experience on conducting projects in this area and our origin and knowing the North Africa context, we militate for adaptation of the experience by integrating the local context. With such motivation in mind, we proposed a SDW design approach conducted by requirements that represents one of the most important phases of any BI&A project related to Africa. We provided a formalization of SDW design cycle including its main phases: requirements definition, conceptual design, ETL design and implementation phases. An ontological framework supporting this design cycle and keeping traces of design decisions has been discussed. Keeping traces is important in this context of LOD environment, where data are not always materialized in the SDW and where many design decisions and time-consuming tasks may be lost (like integration of external data with internal data, mapping definition or multidimensional annotation). We defined a requirement-driven approach that considers requirements incrementally. The incremental design aims covering many design scenarios (design from scratch scenario and existing SDW scenario). Different

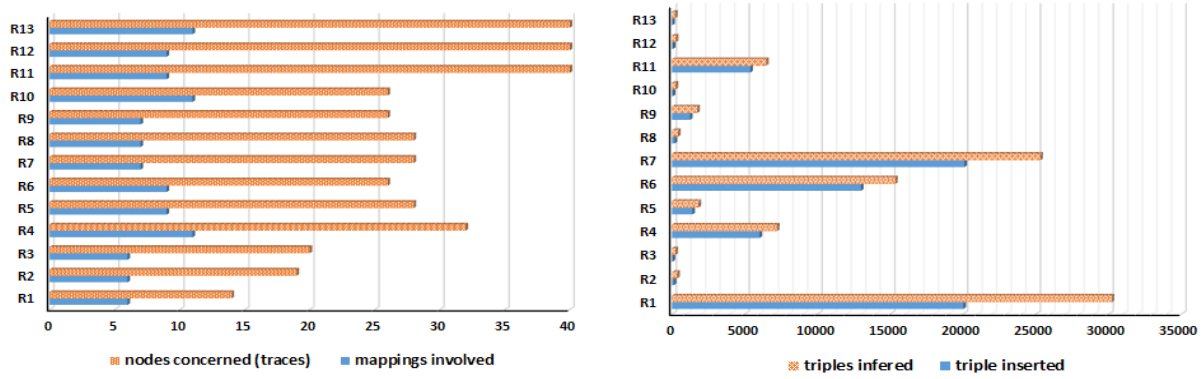

Figure 5: Impact of incorporating requirements: nodes concerned, mappings involved, triples inserted and inferred

experiments are conducted to illustrate the impact of incorporating exploratory requirements in the design process, at different design levels (schema and instances).

We are currently deploying this approach in our training courses to build DW projects and evaluate the satisfaction of users.

8 ACKNOWLEDGMENTS

This research work is carried out thanks to the support of the European Union through the PLAIBDE (Plateforme Intégrée Big-Data pour les Données Entreprise) project of the Program FEDER-FSE[20] of Nouvelle Aquitaine region. The FEDER-PLAIBDE project is directed by the lead manager of aYaline[21] company, and the laboratories partners: L3i[22] laboratory of La Rochelle University (France) and LIAS laboratory of ENSMA (France).

REFERENCES

[1] Alberto Abell Gamazo, Enrico Gallinucci, Matteo Golfarelli, Stefano Rizzi Bach, and scar Romero Moral. 2016. *Towards exploratory OLAP on linked data*. 86?93.
[2] Yaw Anokwa, Carl Hartung, Waylon Brunette, Gaetano Borriello, and Adam Lerer. 2009. Open source data collection in the developing world. *Computer* 42, 10 (2009).
[3] Lorenzo Baldacci, Matteo Golfarelli, Simone Graziani, and Stefano Rizzi. 2017. QETL: An approach to on-demand ETL from non-owned data sources. *Data & Knowledge Engineering* 112, Supplement C (Nov 2017), 17?37.
[4] Nabila Berkani, Ladjel Bellatreche, and Boualem Benatallah. 2016. A value-added approach to design BI applications. In *International Conference on Big Data Analytics and Knowledge Discovery*. Springer, 361–375.
[5] Nabila Berkani, Ladjel Bellatreche, and Selma Khouri. 2013. Towards a conceptualization of ETL and physical storage of semantic data warehouses as a service. *Cluster Computing* 16, 4 (2013), 915–931.
[6] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. 1999. A principled approach to data integration and reconciliation in data warehousing. In *Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99)*.
[7] Isabel F. Cruz and Huiyong Xiao. 2005. The Role of Ontologies in Data Integration. *Jounal of Engineering Intelligent Systems* 13, 4 (2005), 245–252.
[8] Rudra Pratap Deb Nath, Katja Hose, and Torben Bach Pedersen. 2015. *Towards a Programmable Semantic Extract-Transform-Load Framework for Semantic Data Warehouses*. ACM, 15?24.
[9] Zouhir Djilani and Selma Khouri. 2015. Understanding user requirements iceberg: Semantic based approach. In *Model and Data Engineering*. Springer, 297–310.
[10] Lorena Etcheverry, Alejandro Vaisman, and Esteban Zimnyi. 2014. *Modeling and querying data warehouses on the semantic web using QB4OLAP*. Springer, 45?56.
[11] Paolo Giorgini, Stefano Rizzi, and Maddalena Garzetti. 2005. Goal-oriented requirement analysis for data warehouse design. In *Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*. ACM, 47–56.
[12] T.R. Gruber. 1993. A translation approach to portable ontology specifications. *In Knowledge Acquisition* 5, 2 (1993), 199–220.
[13] Selma Khouri, Nabila Berkani, and Ladjel Bellatreche. 2017. Tracing data warehouse design lifecycle semantically. *Computer Standards & Interfaces* 51 (2017), 132–151.
[14] S. Khouri, I. Boukhari, L. Bellatreche, S. Jean, E. Sardet, and M. Baron. 2012. Ontology-based structured web data warehouses for sustainable interoperability: requirement modeling, design methodology and tool. *Computers in Industry* (2012), 799–812.
[15] Selma Khouri and Bellatreche Ladjel. 2010. A methodology and tool for conceptual designing a data warehouse from ontology-based sources. In *13th international workshop on Data warehousing and OLAP (DOLAP)*. ACM, 19–24.
[16] Benedikt Kmpgen and Andreas Harth. 2011. *Transforming statistical linked data for use in OLAP systems*. ACM, 33?40.
[17] Benedikt Kmpgen, Sen O?Riain, and Andreas Harth. 2012. *Interacting with statistical linked data via OLAP operations*. Springer, 87?101.
[18] Jing Lu, Li Ma, Lei Zhang, Jean-Sébastien Brunner, Chen Wang, Yue Pan, and Yong Yu. 2007. SOR: a practical system for ontology storage, reasoning and search. In *Proceedings of the 33rd international conference on Very large data bases (VLDB '07)*. VLDB Endowment, 1402–1405.
[19] Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. 2014. Yago3: A knowledge base from multilingual wikipedias. In *CIDR Conference*.
[20] Adriana Matei, Kuo-Ming Chao, and Nick Godwin. 2015. *OLAP for multidimensional semantic web databases*. Springer, 81?96.
[21] Victoria Nebot and Rafael Berlanga. 2016. Statistically-driven generation of multidimensional analytical schemas from linked data. *Knowledge-Based Systems* 110 (2016), 15?29.
[22] Franck Ravat, Jiefu Song, and Olivier Teste. 2016. Designing multidimensional cubes from warehoused data and linked open data. In *Research Challenges in Information Science (RCIS), 2016 IEEE Tenth International Conference on*. IEEE, 1–12.
[23] Rafik Saad, Olivier Teste, and Cssia Trojahn. 2013. *OLAP manipulations on RDF data following a constellation model*.
[24] Alkis Simitsis, Dimitrios Skoutas, and Malú Castellanos. 2010. Representation of conceptual ETL designs in natural language using Semantic Web technology. *Data & Knowledge Engineering* 69, 1 (2010), 96–115.
[25] Dimitrios Skoutas and Alkis Simitsis. 2006. Designing ETL processes using semantic web technologies. In *9 th ACM international workshop on Data warehousing and OLAP (DOLAP)*, Vol. 10. 67–74.
[26] Alejandro Vaisman and Esteban Zimányi. 2014. *Data warehouse systems*. Springer.
[27] Holger Wache, Thomas Voegele, Ubbo Visser, Heiner Stuckenschmidt, Gerhard Schuster, Holger Neumann, and Sebastian Hübner. 2001. Ontology-based integration of information-a survey of existing approaches. In *IJCAI-01 workshop: ontologies and information sharing*, Vol. 2001. Citeseer, 108–117.
[28] Z. Wu, G. Eadon, S. Das, E.I. Chong, V. Kolovski, M. Annamalai, and J. Srinivasan. 2008. Implementing an Inference Engine for RDFS/OWL Constructs and User-Defined Rules in Oracle. In *24th International Conference on Data Engineering (ICDE)*. 1239–1248.

[20] http://www.europe-en-poitou-charentes.eu/Les-projets-soutenus-en-Poitou-Charentes/Travaux-de-recherche-pour-la-creation-d-une-Plateforme-integree-Big-Data-pour-les-donnees-entreprise-PLAIBDE-PC470
[21] https://www.ayaline.com/
[22] http://l3i.univ-larochelle.fr/