information

Article

LOD for Data Warehouses: Managing the Ecosystem Co-Evolution

Selma Khouri [1,2,*] and Ladjel Bellatreche [2]

1 Laboratoire LCSI, Ecole Nationale Supérieure d'Informatique, Alger 16309, Algeria
2 LIAS/ISAE-ENSMA, The University of Poitiers, 86960 Futuroscope CEDEX, France; bellatreche@ensma.fr
* Correspondence: s_khouri@esi.dz

Received: 9 June 2018; Accepted: 11 July 2018; Published: 17 July 2018

Abstract: For more than 30 years, data warehouses ($\mathcal{DW}$s) have attracted particular interest both in practice and in research. This success is explained by their ability to adapt to their evolving environment. One of the last challenges for $\mathcal{DW}$s is their ability to open their frontiers to external data sources in addition to internal sources. The development of linked open data ($\mathcal{LOD}$) as external sources is an excellent opportunity to create added value and enrich the analytical capabilities of $\mathcal{DW}$s. However, the incorporation of $\mathcal{LOD}$ in the $\mathcal{DW}$ must be accompanied by careful management. In this paper, we are interested in managing the evolution of $\mathcal{DW}$ systems integrating internal and external $\mathcal{LOD}$ datasets. The particularity of $\mathcal{LOD}$ is that they contribute to evolving the $\mathcal{DW}$ at several levels: (i) source level, (ii) $\mathcal{DW}$ schema level, and (iii) $\mathcal{DW}$ design-cycle constructs. In this context, we have to ensure this co-evolution, as conventional evolution approaches are adapted neither to this new kind of source nor to semantic constructs underlying $\mathcal{LOD}$ sources. One way of tackling this co-evolution issue is to ensure the traceability of $\mathcal{DW}$ constructs for the whole design cycle. Our approach is tested using: the LUBM (Lehigh University BenchMark), different $\mathcal{LOD}$ datasets (DBepedia, YAGO, etc.), and Oracle 12c database management system (DBMS) used for the $\mathcal{DW}$ deployment.

Keywords: co-evolution; data warehouse; LOD; external data; design cycle; persistence

1. Introduction

Endowing data with semantics is a crucial task for many applications. Motivated by the emergence of ontologies and linked open data ($\mathcal{LOD}$) in different fields (e.g., Cyc, DBpedia, Freebase, and YAGO), the amount of semantic data is rapidly increasing in various domains (e.g., the New York Times, BBC, and Thomson Reuters semantic data) and their involvement in different data centric systems is growing. Business intelligence (BI) systems are no exception. Data warehouses ($\mathcal{DW}$s) are data management systems at the core of BI applications, used in small, medium, and large organizations for decision support. $\mathcal{DW}$s integrate operational data coming from heterogeneous sources, and organize data using a multidimensional perception that identifies central subjects of analysis (called the *Facts*) according to different dimensions of analysis, to be exploited by OLAP (on-line analytical processing) techniques and tools.

The $\mathcal{DW}$ community has undergone an important event materialized by the spectacular development of Semantic Web technologies that are intensively used in designing and exploring $\mathcal{DW}$s (for more details, refer to the survey paper [1]). Many studies have used ontologies (an ontology is defined as a specification of a conceptualization) [2] in order to facilitate $\mathcal{DW}$ design. Unlike the traditional vision of $\mathcal{DW}$ construction which concentrates on internal sources, the new generation of $\mathcal{DW}$s integrates both internal and external sources. Among external sources, recent research efforts have identified the interest of integrating the $\mathcal{LOD}$ in designing $\mathcal{DW}$. As part of the Big

Data landscape, $\mathcal{LOD}$ initiatives bring new and publicly available semantic data on the web that are valuable for data analytics inside $\mathcal{DW}$ systems. The value of $\mathcal{LOD}$ for $\mathcal{DW}$s manifests in their capabilities to answer requirements that are unsatisfied from internal sources, and to enrich the multidimensional expressiveness of the $\mathcal{DW}$ model. $\mathcal{LOD}$ keep adding value to $\mathcal{DW}$s, since they are continuously changing (new concepts and knowledge). For instance, Yago has evolved from Yago to Yago3, passing by Yago2. Therefore, the added-value of $\mathcal{DW}$ has to be managed continuously through an evolution process.

Traditional evolution approaches for $\mathcal{DW}$ manage the evolution of internal sources (usually conventional relational databases). The use of external semantic data poses new challenges related to their evolution management: (i) $\mathcal{LOD}$ sources bring a new kind of heterogeneity since they are presented using the semantic formalism, based on the Resource Description Framework (RDF) (https://www.w3.org/RDF/) language using the triple format to organize data in the form of $< subject, predicate, object >$). The set of triples form a graph. (ii) The evolution events of $\mathcal{LOD}$ sources (e.g., addition, deletion, renaming, etc.) are gathered from the web using the triple format, which requires specific management tasks to identify the impact of such evolution events on the $\mathcal{DW}$ system. (iii) Finally, different scenarios have been proposed in order to integrate $\mathcal{LOD}$ sources in the $\mathcal{DW}$ that can be done at the schema level or at the query level. Contrary to traditional $\mathcal{DW}$s that manage evolution either from a source perspective or from a requirements perspective, ignoring the interrelated artifacts composing the $\mathcal{DW}$ design cycle, $\mathcal{LOD}$ can be integrated at different steps of the $\mathcal{DW}$ cycle, possibly impacting different artifacts. Such integration scenarios are conducted during the core phases of $\mathcal{DW}$ design, namely: *requirements definition*, the *extract–transform–load* (ETL) phase composed of a set of ETL processes used to transform heterogeneous data in the $\mathcal{DW}$ representation, and the *deployment phase* which implements the $\mathcal{DW}$ system. In this context, the $\mathcal{DW}$ system can be seen as a set of components inherently intertwined with each other. This idea of managing evolution between different types of software artifacts or different representations of them is called "co-evolution" in the software engineering field.

Our goal in this study was to define the impact of changes on a $\mathcal{DW}$ system populated by internal and external sources, by identifying design constructs (in each design phase) impacted by the changes, and to propagate these changes to the final $\mathcal{DW}$ system. Forward and backward evolution strategies are required to manage changes. The identification of design constructs is a significant contribution of our work, since evolution events may concern any of these interlinked constructs of different design levels. Our approach exploits the correlations between constructs in order to identify which part of the $\mathcal{DW}$ is sensitive to evolution events. The identification of change impact on the $\mathcal{DW}$ design-cycle requires an explicit representation of the $\mathcal{DW}$ design artifacts, their relationships, and their traces.

To the best of our knowledge, this work is the first study that manages co-evolution in the context of $\mathcal{DW}$ fed by $\mathcal{LOD}$ sources. Moreover, the availability of the ontology model allows evolution to be addressed from the conceptual level, where the understanding of the evolution problem is easier to follow and to propagate to the other design levels. The main contributions of this work are (Figure 4): (1) We describe a $\mathcal{DW}$ traceability model that uniformly covers the most relevant $\mathcal{DW}$ constructs and artifacts. In order to define traces semantically, we extend the ontology meta-model by the design model. (2) We present an event-based approach for managing the co-evolution of the whole cycle, including a co-evolution mechanism for the annotation of $\mathcal{DW}$ constructs impacted by evolution events at the semantic level. Evolution events may concern primarily $\mathcal{LOD}$ or other design constructs. Regarding $\mathcal{LOD}$ changes, evolution is managed in asynchronously, which we consider to be the most suitable approach since the $\mathcal{LOD}$ receive flows of changes, and only some of them are relevant for the $\mathcal{DW}$. (3) We demonstrate the efficiency and feasibility of our approach through a case study related to the University domain.

Motivating Example

We illustrate our purpose by the following example:

Example 1. *Let us consider a $\mathcal{DW}$ used for analysing the research publications of a university. The number of publications is analysed according to each department and according to the type of publication (conference or journal). Information related to publications are collected from internal sources related to the university researchers, and the characteristics of publications (impact factor, type, editor, etc.) are collected from $\mathcal{LOD}$ like Thomson Reuters datasets. Inspired by DBpedia (one of the most popular $\mathcal{LOD}$ portals), we assume that $\mathcal{LOD}$ portals provide the set of changes periodically in the form of N-Triples files containing added, modified, and deleted triples. Identifying the impact of these changes when dealing with large amounts of triples is a particularly difficult and error-prone task if it is not automated. An ontological view is required for identifying the impact of these changes: (1) at the conceptual level (i.e., which concepts are concerned by these changes and are they involved in any ETL process feeding the $\mathcal{DW}$?); (2) at the requirements level (i.e., which requirements are concerned by these changes and are they still satisfied after applying the changes?); and (3) at the physical level (i.e., which ontological instances in the database management system (DBMS) are concerned with the changes?). Note that the reasoning skills of the ontology can also be used for inferring new traces of the impact of events evolution.*

This paper is organized as follows: Section 2 overviews the related work related to $\mathcal{DW}$ design (from internal and external sources) and $\mathcal{DW}$ evolution. Section 3 gives the background and essential concepts for the comprehension of our approach. Section 4 presents the $\mathcal{DW}$ traceability model underlying the proposed approach. Section 5 describes the proposed co-evolution approach. Section 6 presents the case study (related to the University domain) used to evaluate the feasibility and effectiveness of our approach. The last section concludes the paper.

2. Related Work

This section contains two main parts: it begins by presenting the main studies proposed for semantic $\mathcal{DW}$ design from internal and external sources, then it presents the main studies proposed for managing $\mathcal{DW}$ evolution.

2.1. $\mathcal{DW}$ Design from Internal and External Sources

Conventional Semantic approaches. Figure 1 illustrates the typical design phases of a semantic $\mathcal{DW}$ design. Following the conventional DW design [3], a set of sources and a set of user requirements are considered as inputs. The requirements definition phase allows the definition of the set of concepts that are adapted to the $\mathcal{DW}$ context. The selected concepts form the $\mathcal{DW}$ model. It can be represented by a local $\mathcal{DW}$ ontology (DWO), representing an implementation-independent schema for the $\mathcal{DW}$. The $\mathcal{DW}$ model is annotated with multidimensional concepts. A multidimensional view identifies *Fact* concepts, which are central concepts to analyze and identify *Dimension* concepts representing perspectives of analyzing the facts (e.g., analyzing the number of publications (fact) for each university (dimension)). This schema can be represented according to a specific conceptual design model. The ETL phase includes three steps: the extraction of data from sources, their transformation, and their loading into the target DWO schema using mapping assertions defined between target and source schemas. Mapping assertions are defined using ETL operators (e.g., union, aggregate, convert, etc.), forming an ETL workflow. We consider ETL processes that define each concept of the DWO according to the concepts of sources (Global as View approach). For example, the concept *University* is fed using the ETL process formed by the *Union* of concept *University* from Source S1 and concept *School* from Source 2 ($\text{University}_{DWO} \equiv \text{Union}(\text{University}_{S1}, \text{School}_{S2})$). The deployment design allows the implementation of the $\mathcal{DW}$ according to the storage model of the database management system (DBMS) chosen. Managing the co-evolution of the $\mathcal{DW}$ requires identifying the source of changes and predicting the impact of changes at any design level.

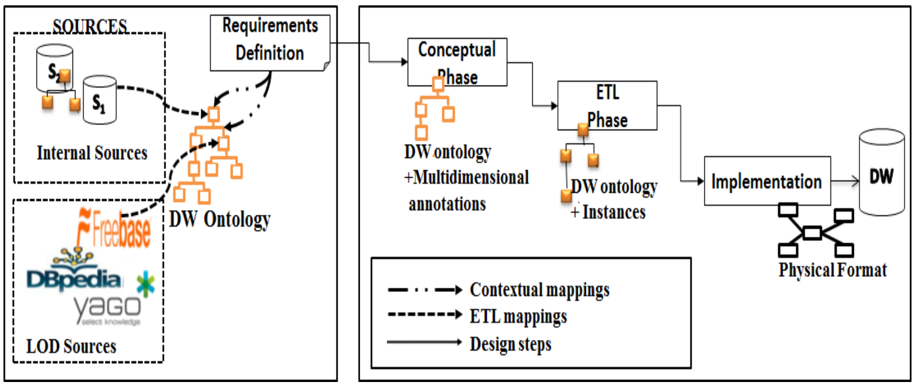


Figure 1. Data warehouse ($\mathcal{DW}$) design process. ETL: extract–transform–load; $\mathcal{LOD}$: linked open data.

In this process, the usage of ontologies in $\mathcal{DW}$ design impacts all the design phases [4]. The cited approaches used the ontology schema. Ontologies were first introduced in DW design to facilitate the process of integrating sources, by exploiting their sharing and descriptive characteristics [5]. The use of ontologies was then extended to different design levels: they have been used for analyzing users' requirements [6], for facilitating the definition of $\mathcal{DW}$ schema [7–10], for automating the ETL process [11,12], and for documenting the integration process [13]. Semantic databases, which are sources containing their own ontology schema and data, have been involved in $\mathcal{DW}$ design as candidate sources [14]. These studies used semantic web technologies in a closed world where semantic sources are internal to the environment of the organization. The integration of $\mathcal{LOD}$ in $\mathcal{DW}$ design was introduced with the vision of opening the internal environment to external sources by using open semantic spaces published on the web [1].

$\mathcal{LOD}$-driven approaches. Tim Berners Lee, the founder of the web, outlined a set of best practices, known as the "linked data principles" (https://www.w3.org/DesignIssues/LinkedData.html) for publishing and connecting structured data on the Web: (1) use Uniform Resource Identifiers (URIs) as names for things. (2) Use HTTP URIs so that people can look up those names. (3) When someone looks up a URI, provide useful information using the standards: RDF language and SPARQL query language. (4) Include links to other URIs, so that they can discover more things. Linked data principles provide guidelines on how to use standardized Web technologies to set data-level links between data from different sources [15]. Over the last three years, an increasing number of data providers have begun to adopt the linked data principles, creating a global data space containing billions of assertions about geographic locations, scientific publications, media, clinical trials, etc. [15].

We identify two main types of approaches that combine LOD and OLAP. The first category *OLAP2LOD* deals with multidimensional data published in the web as linked data, usually based on the RDF Data Cube vocabulary (QB) (https://www.w3.org/TR/vocab-data-cube/) recommendations. This category of approaches aims to achieve business analytics at the web scale, and different studies have been proposed to manage issues related to such analysis (e.g., query management and optimization issues) [16,17]. The second category of approaches follows a *LOD2OLAP* approach that manages the incorporation of $\mathcal{LOD}$ in the existing $\mathcal{DW}$ environment. Such approaches deal with the issue of opening the $\mathcal{DW}$ frontiers to external data like $\mathcal{LOD}$. Our approach follows this second category. Two main solutions have been proposed for the co-habitation between $\mathcal{LOD}$ in the $\mathcal{DW}$ environment that can be made *before* or *during* querying.

In *before querying* solutions, the cohabitation can be made at the schema level or at the ETL level. Concerning the first category, the consideration of $\mathcal{LOD}$ is proposed for several purposes, such as *reparation of missing information* in internal sources as done in [18], or the *unification of cubes*—the internal relational cube translated into RDF data, with RDF cube defined from the $\mathcal{LOD}$ as proposed in [19]. The ETL process was not considered in both studies. The second category of approaches handle this issue like [20] which proposes an incremental fetching and storing of $\mathcal{LOD}$ and other external resources on-demand. An ETL process is defined to load external data. Motivated by the fact that some external sources have a limited time access, a process determines the set of extractions from external data that cover the missing information with a minimum cost. Deb Nath et al. [21] propose a programmable semantic ETL framework focusing on the integration of external $\mathcal{LOD}$ datasets. Internal relational resources are then linked to external resources by calculating similarity measures. Kämpgen and Harth [22] propose an approach to interpret linked data as a multidimensional model and to automatically load the data into the $\mathcal{DW}$. This approach does not consider internal sources. In the studies presented, the $\mathcal{DW}$ is generally considered as operational, except in the work of [23] which proposes an approach to construct a $\mathcal{DW}$ from scratch, considering both internal and $\mathcal{LOD}$ sources using a common ETL process.

In *during querying* solutions, we can cite the work of [24], which proposes an OLAP (on-line analytical processing) framework taking as inputs raw data from relational databases and semantic web data of different formats. A set of specialized OLAP operators is defined for use in queries. These operators can operate over multiple semantic sources and merge the outputs. Saad et al. [25] formalize a multidimensional model in terms of RDF data structures following a conceptual constellation model. Conventional OLAP operations are then translated into SPARQL queries to fulfill the target requirements. Kämpgen et al. [26] define common OLAP operations on input linked data modeled using the QB (https://www.w3.org/TR/vocab-data-cube/) vocabulary. A set of OLAP queries are transformed into SPARQL queries to analyze the cube. Results are propagated to OLAP clients.

The cited studies generally focus on specific design issues (multidimensional identification, $\mathcal{LOD}$ remote time access, $\mathcal{LOD}$ schema definition, ETL). These studies illustrate the variety of scenarios proposed for incorporating $\mathcal{LOD}$ sources. Contrary to internal sources, $\mathcal{LOD}$ sources can be integrated in the $\mathcal{DW}$ system at different steps of its design cycle (requirements, ETL, physical schema). This consolidates the solution of the co-evolution approach we are proposing in this paper, which considers the $\mathcal{DW}$ as an ecosystem composed of interrelated artifacts.

2.2. DW Evolution Management

In this section, we review the most important studies that manage evolution in conventional and semantic $\mathcal{DW}$s. Data-centric systems evolution has recently gained research attention [27], due to its great significance and practical importance. Vassiliadis et al. consider schema evolution as a process that can severely impact the life cycle of data-intensive software projects, as schema updates can drive dependent applications to crash or deliver incorrect data to end users [27]. Three main approaches for managing $\mathcal{DW}$ evolution have been proposed in the literature [28]: view maintenance, $\mathcal{DW}$ evolution, and $\mathcal{DW}$ versioning. View maintenance was first studied at the beginning of data warehousing, when $\mathcal{DW}$s were considered as collections of materialized views, defined over sources. These studies ignore the ETL phase, which is an integral part of current $\mathcal{DW}$s. Versioning consists of adapting the $\mathcal{DW}$ schema by maintaining the history of previous schemas. Our study tackles the $\mathcal{DW}$ *evolution* issue which considers that only one version of the $\mathcal{DW}$ is maintained. Many studies have proposed how to manage evolution for conventional $\mathcal{DW}$s. For instance, Quix et al. [29] propose an approach to $\mathcal{DW}$ evolution using complementary meta-data that track the history of changes. Some studies propose to study the evolution of the $\mathcal{DW}$ multidimensional schema [30]. Fan et al. [31] present a framework for managing schema evolution in $\mathcal{DW}$s using a schema transformation-based approach to handle the evolution of the source and the $\mathcal{DW}$ schema. Hoang [32] proposes a methodology that handles requirements evolution in the context of on-demand $\mathcal{DW}$s. Papastefanatos et al. [33] discuss the problem of performing what-if analysis for changes that occur in the sources schema. Jovanovic et al. [34] present an evolution approach based on a meta-data repository that integrates the $\mathcal{DW}$ data and meta-data. A historized meta-data repository is proposed to support schema changes. Studies like the one of Manousis et al. [28] provide a survey of the most relevant evolution approaches in conventional $\mathcal{DW}$s. Most of the studies manage evolution at the physical level—they manage evolution either from sources or from requirements. Few studies approach the concept of *co-evolution* by considering several design levels at once. One such study was conducted by Papastefanatos et al. [35], where they consider relations, queries, views, and ETL activities. The study presents a method for performing impact prediction for the adaptation of ETL workflows to evolution events occurring at their sources. Munoz et al. [36] propose a set of design measures related to ETL design used to predict the effort associated with the maintenance tasks of ETL processes. El Akkaoui et al. [37] propose a model-driven framework that supports the maintenance of ETL models (based on Business Process Model and Notation BPMN) according to data source evolution. Papastefanatos et al. [38,39] propose a set of metrics to predict the vulnerability of relational $\mathcal{DW}$ modules to future changes. The above studies manage evolution from internal sources usually having a relational formalism.

The evolution of semantic $\mathcal{DW}$s remains an open issue, and efforts achieved in the context of conventional $\mathcal{DW}$s should be revisited for semantic $\mathcal{DW}$ design. Regarding the evolution management of semantic-driven $\mathcal{DW}$s, few recent studies have been proposed. Romero et al. [40] proposed to manage the evolution of a mediator integration system using ontologies as intermediary schema. Unlike $\mathcal{DW}$ systems, mediators do not manage ETL or the persistency of integrated data. Jovanovic et al. [41] present a semi-automatic method for managing the evolution of events occurring on requirements. Their approach uses a domain ontology to automatically explore relationships between concepts. These studies do not consider semantic datasets (neither internal nor external sources), and no study considers the evolution of design artifacts and their correlations. Even if these studies use ontologies, they consider them as intermediate schema to unify the sources and the target $\mathcal{DW}$.

3. Background

In this section, we provide some basic definitions related to co-evolution management and some related to $\mathcal{LOD}$ sources.

3.1. Co-Evolution Management

Co-evolution is a term originating in biology that has been borrowed in software development to state that modification in one representation should always be reflected by corresponding changes in other related representations to ensure the consistency of all involved software artifacts [42]. In sharp distinction to traditional software systems, data-intensive system evolution has hardly been studied throughout the entire lifetime of the data management discipline [43].

Co-evolution management follows a set of well-defined activities. Starting with a consistent modeling ecosystem in its original version, the executed phases are as follows [44]:

- change detection: consists of identifying all applied changes in the model,
- impact analysis: consists of the determination of the impacts of the applied changes on diverse kinds of artifacts,
- change propagation: consists of the actual propagation of changes to ultimately migrate the artifacts (to version v1),
- Validation: this step is optional, it consists of the validation of the migrated artifacts.

In this paper, we adapt these activities to the $\mathcal{DW}$ system fed by internal and external sources.

3.2. LOD, Ontologies, and Knowledge Bases

One interesting characteristic of $\mathcal{LOD}$ is that they are structured by ontologies that define the sense of their knowledge bases (KBs) [45]. Ontologies are usually formalized in Ontology Web Language (OWL) standard, formalized using a description logic (DL) formalism. In DL terminology, a $\mathcal{KB}$ 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, the TBOX is defined as 5-tuples: $< C, R, L, Def(R), Ref(C) >$, where:

- C: a set of concepts of the TBOX
- L: a set of literals (e.g., numbers, strings, 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 as 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.
- 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(Student)$\rightarrow$($\sqsubseteq$, Person $\sqcap$ ($\exists$ follows.Course))).

Instances of the ABox that represent the extensional part of a source can be formalized as follows: $< I, Pop, Format_{TBOX}, Format_I >$, where:

- I: the set of instance (the ABOX)
- Pop: C $\rightarrow 2^I$ corresponds to the concept instantiation.
- $PhyFormat_I$: represents the physical format used for the instances part I.
- $PhyFormat_{TBOX}$: represents the physical format used for the model part $TBOX$.

For example, for $\mathcal{LOD}$ sources, the schema and instances are stored using the same graph-based representation. $\mathcal{LOD}$ representations are based on the RDF language (Resource Description Framework: https://www.w3.org/RDF/). RDF is a set of 4-tuple $< s, o, p, g >$ forming an RDF quad, where its subject s has the property p, and the value of that property is the object o, and the graph label is g. Note that the graph label g can be omitted, in which case the triples are considered as part of the default graph of the RDF data-set. Examples of statements from our case study are: (Student#1, follows, Course#1), (Student#1, type, Student), (Student, subClassOf, Person).

Note that DL formalism also captures the knowledge of conventional conceptual schemas of relational databases like Entity/Relationship (ER) or Unified Modeling Language (UML) class diagram [46], which makes our representation generic for conventional internal sources and external sources. Different studies advocate defining the ETL process at this conceptual level, in order to facilitate the integration process [47], which is the approach we follow in this paper.

In the following, we use this formalization as a basis for defining the $\mathcal{DW}$ design model that underlies our proposed co-evolution management approach.

4. $\mathcal{DW}$ Traceability Model for Managing Co-Evolution

We propose a metamodel presenting the co-dependent artifacts used in DW design considering both internal and external sources, namely: requirements, ETL, and DW model (conceptual artifacts and physical storage artifacts). The model helps to identify the parts of the overall configuration that are most prone to being affected by the changes. Several approaches have been proposed to detect and manage traces in the context of $\mathcal{DW}$s. We assume that traces are available and correctly maintained.

In our work, we use the traceability model that we proposed in [48], the goal of which is to keep traces of the $\mathcal{DW}$ during its design cycle. The model is illustrated in Figure 2 as a UML class diagram. The model represents each design phase (framed in a box in the figure) including its main design artifacts. The central part of the model is related to the ontology model *OntologyResource* class. Based on the previous formalization (Section 3.2), this class subsumes three main classes: *Concept*, *Role* and *Instance*. Note that the *OntologyResource* class in the model designates the resources of sources (internal or external) and those of the DWO. Each ontology has a unique identifier (the URI), which is associated to each resource name composing the resource identifier. These resources are identified to satisfy some given requirements (*Requirement* class). The set of resources are alimented using an ETL workflow composed of a set of ETL operators that defines the OutputSet (an OntologyResource of the DW) in terms of an expression InputSet (DL expression over resources of internal and external sources). Skoutas and Simitsis [11] identified ten generic ETL operators for integrating internal data sources into the $\mathcal{DW}$ system, namely: Extract, Retrieve, Filter, Union, Aggregate, Convert, Join, Sort, Merge, and Split. We introduce new ETL operators for managing $\mathcal{LOD}$ sources that cover the integration scenarios cited in Section 2.1: *context* operator (used for identifying $\mathcal{LOD}$ concepts to extract), *synchronize* (for synchronizing the internal ETL flow with the external flow), and the *query* operator for unifying the result of an ETL flow required by a given with the internal $\mathcal{DW}$. We classify these operators into five main classes: *Source operators*, *Transform operators*, *Store operators*, *Flow operators*, and *Query operators*. The set of operators of each group is defined as enumerations in the model. Finally, the set of resources used are translated to their physical format using a set of defined translation rules.

The traces between design artifacts are illustrated using the trace links presented by the stereotype $<< \mathit{Trace} >>$. Each TraceLink involves the source of the trace and its target. For example, concept University (OntologyResource class) is the source of a trace, and the ETL process alimenting this concept is the target. Some *TraceLinks* require the definition of trace rules (*TraceRules* class) if the target element is generated from its source using defined rules (e.g., translation from the ontology layout to the $\mathcal{DW}$ physical format). Source and target trace elements are defined in the model for each design phase. These traces facilitate the identification of change impact. For example, concept Organization can be considered as important since it subsumes other concepts (University, Academic press, Faculty, etc.). One requirement can be considered as sensitive if it involves different concepts of the $\mathcal{DW}$. We followed a meta-modeling approach for the definition of this traceability model. The ontology meta-model (class *OntologyResource*) of the $\mathcal{DW}$ is connected to the traceability model as presented in Figure 3 (upper part). This figure is illustrated using Protege editor, where the ontology meta-model is extended with the traceability model described. The ontology illustrated in this figure is a fragment of Lehigh University BenchMark (LUBM) benchmark ontology that will be used in the case study section. The adoption of a meta-modeling approach is explained by the fact that traceability data are considered as meta-data. Additionally, this makes the approach independent of any particular modeling technique.

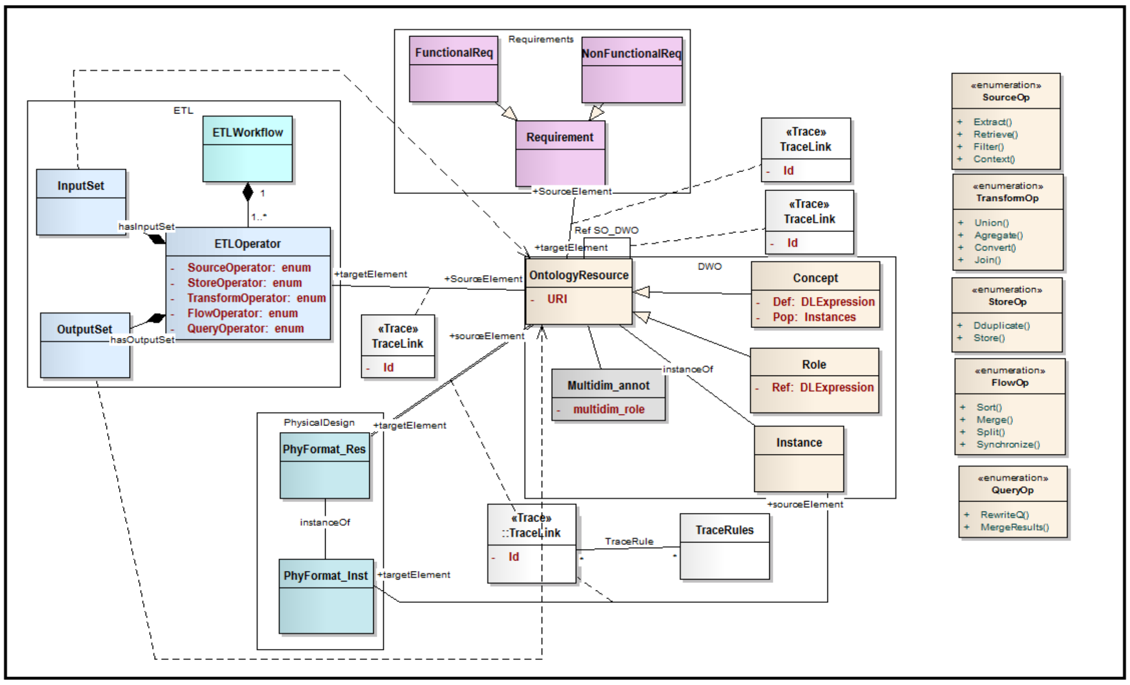


Figure 2. $\mathcal{DW}$ traceability model for managing co-evolution.

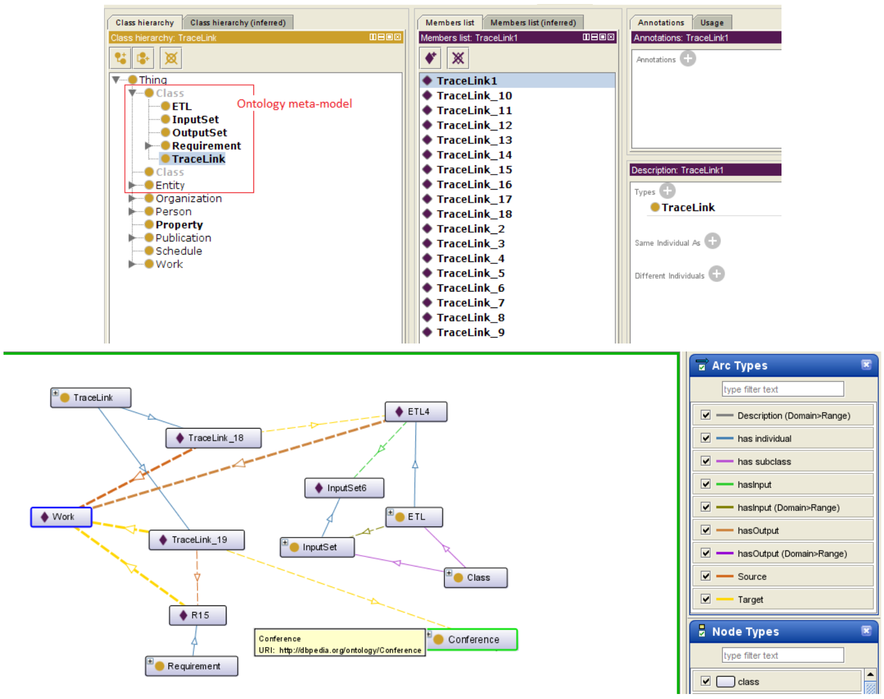


Figure 3. Traceability model and trace links (Protege editor).

Note that the DBMS may keep track of meta-data and some ETL tools also keep track of the ETL processes, but it is achieved at the physical level and is not centralized in a single model. Extending the ontology model by the traceability model allows traces to be defined in a common model and allows semantic identification of all correlations between design constructs of different levels (ontology, ETL, requirements, etc.). This yields a conceptual representation of event changes (concepts, relationships, specializations), when the final $\mathcal{DW}$ model normally lacks these concepts. The ontology is also used to reason on traces, and ontology editors offer powerful visualization tools (Figure 3).

5. Evolution Management Approach

Our general proposed approach is illustrated in Figure 4. The approach identifies the changes of $\mathcal{LOD}$ datasets and their impact on the $\mathcal{DW}$ constructs at different design levels. Two main types of changes are identified for $\mathcal{LOD}$ sources [16,49]: simple and complex changes. Simple changes (addition, deletion, and modification of triples) are meant to capture fine-grained evolution events. Complex changes are defined on top of simple changes to capture more coarse-grained changes (e.g., schema changes, groupings of triple additions). Because it is unrealistic to capture all possible evolution types of complex changes that can be defined, simple changes can be considered as a "default" set of changes for describing evolution types [49].

For each of the above events, the approach manages co-evolution by annotating $\mathcal{DW}$ constructs (DWO fragment) affected by the event and by applying the changes. We propose a co-evolution management approach that follows these steps:

1. Parsing the input files: The $\mathcal{LOD}$ datasets that we analyzed and their different versions reflecting evolution events are available as ntriples files. Note that the files can be available in other formats serializing RDF data (e.g., N3 and Turtle). Our approach first considers a parser that analyses the input files (i.e., ntriple files of changes). The files concerning each type of change (add, delete, modify) are provided separately. The files can be visualized by the designer using Protege editor.

In these files, the set of changes concerning the dataset and those concerning the schema are declared without distinction as triples. This first step of our approach aims to semantically define the set of changes by distinguishing the data from schema changes. For example, the "add" file can include (American_Journal_of_Business, academicDiscipline, Business) as a triple of type instance addition and (publisher, type, Property) as a property addition. Because such triples are not intuitive enough for human designers, our approach parses the files in order to identify the concepts and all their instances that are concerned by the changes. This identification is achieved using a java script that reads the files using the OWL API (http://owlcs.github.io/owlapi/).

The result of this parsing is a set of changes that belong to the space of potential events comprising the Cartesian product of two sub-spaces: the space of hypothetical actions (addition, deletion, and modification) over the space of resources of the $\mathcal{LOD}$ knowledge base (Concept, Role, and Instance). Note that annotation properties related to Wikipedia (like WikiPageExtracted, WikiPageID, label, ...) are considered as annotations. they are reflected only in the DWO ontology model, and are not propagated.

Once the concept representing the first impact of change management is identified, it is propagated to the other design artifacts. Our approach identifies the concepts that are concerned with the changes in the DWO using the ETL processes. That is, instances of the meta-concept *ETLProcess* in the ontology (representing the ETL workflow composed of a set of ETL operators). The identification of the ETL processes impacted by the change is the first step, because that is what links the $\mathcal{DW}$ schema with its sources. Note that we consider a Global as View (GaV) approach for defining the ETL processes, which is the approach usually followed in $\mathcal{DW}$ design. The GaV approach advocates for defining the concepts of the target schema (the DWO) in terms of the sources' schemas (internal and external sources). We also assume that the mappings between sources and target data stores are achieved at the schema level (not at the instance level). After identifying the set of ETL processes concerned by the evolution event, the set of artifacts concerned by this event are then identified using a change propagation mechanism described in the next step.

The changes concerning design constructs (requirements, ETL) can also be defined in ntriples or OWL files, since our model defines all these constructs as ontological concepts. In this case, the files must have the same meta-model extension by the traceability model. The files are parsed similarly to match between concepts of the input files and concepts of the DWO. A simpler approach is to manually define these changes by the designer, since they occur less frequently than $\mathcal{LOD}$ evolution events.

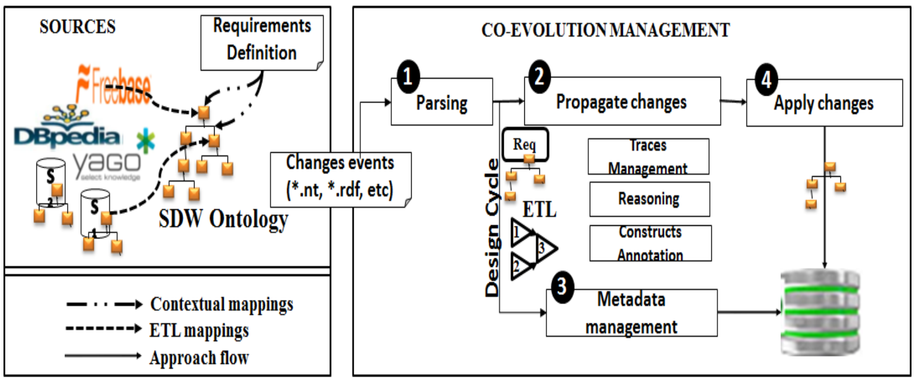


Figure 4. Proposed approach.

2. Propagating the changes: The core step of our approach consists of propagating the identified evolution events. We propose a concept-flow analysis algorithm entitled *PropagateChange* (presented in Algorithm 1) to identify the concepts concerned by the changes. When an evolution event is tested, the impact of the event is automatically computed throughout the ontology subset affected by the event. After the identification of the first impact of the evolution event on the DWO (the ETL processes concerned) in the previous step, Algorithm 1 propagates this impact. The propagation is achieved by identifying the set of concepts C of the DWO that are involved in the ETL processes identified. Then, the set of concepts related to C are also identified (*Usage (C)* in Algorithm 1, line 2).

Example: Assume that the following ETL processes are identified:

$$\text{Publisher}_{DWO} = \text{Publisher}_{Dbpedia} \; \textit{Union} \; \text{AcademicPress}_{Yago}.$$

An evolution event on AcademicPress will identify this ETL process as the first impact and then Publisher$_{DWO}$ as the second impact. The set of concepts related to Publisher$_{DWO}$ are identified (e.g., Publication). The set of requirements involving these concepts are identified. For instance, requirement R1 analyzes the number of publications for each university, and it thus involves concept Publication in its parameters.

To do so, the algorithm uses trace links (instances of class TraceLink in Figure 2). We identify two types of trace links: defined and inferred links. Reasoning on trace links is based on a transitive rule used to identify new traces. This rule states 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. The rule is formalized in Semantic Web Rule Language (SWRL) as follows:

```
Source(?T1, ?X), Source(?T2, ?Y), Target(?T1, ?Y), Target(?T2, ?Z) -> Target(?T1,?Z).
```

In this rule, Source(?T, ?X) (resp. Target(?T,?Y)) defines that a trace link T has design element X as source (resp. element Y as target). Source or target elements can be any TraceElement, that is, any instance of the meta-concepts extending the ontology meta-model as illustrated in Figure 3 (DWO concepts, requirement concepts, ETL concepts, etc.). The algorithm relies on the results of an ontological reasoner (we used Pellet reasoner) to get inferred trace links. The impact of the evolution event is thus propagated to all design constructs of the concepts impacted by the change (Algorithm 1, lines 3–4). This final subset of a concept impacted by the change is consequently identified by including both its immediate and its transitive consumers. The impacted concepts are added into a queue to be processed in the next step for applying the changes (Algorithm 1, line 5). For each processed concept, its meta-concept is identified to apply the adapted evolution process (Algorithm 1, line 9).

Algorithm 1 Change Propagation Algorithm

```
Input: (1) an event e over a concept C,
(2) the ontology model extended by the traceability model,
(3) Q: Queue  of concepts concerned by the event ($Q = \emptyset$)
Output: the fragment of the ontology impacted by the event
Enqueue (Q, usage(C))
for each TraceLink trace involving C do
    Get concepts C′ included in trace
    Enqueue (Q, C′)
end for

while $Q \neq \emptyset$ do
    C = Dequeue Q
    Identify meta-concept of C % ETL process, Requirement or DWO concept
    ApplyChange (C,event e)
end while
```

We used Protege editor to visualize the fragment of the DWO concerned by the evolution events. Protege offers different visualization plugins. We chose the OntoGraph plugin, which provides a visualization in the form of a graph. Figure 3 (lower part) illustrates the ontology fragment involved with concept Work (i.e., research works) in LUBM ontology. Note that in the figure the concept Conference is provided from DBpedia $\mathcal{LOD}$, and other concepts defining design constructs are also identified (by their meta-concept). Note also that this mechanism propagates the changes regardless of the first impact, and consequently allows a forward and a backward strategy propagation of changes. Backward strategy refers to the ability to follow the trace links from a specific artifact back to its sources, from which it has been derived. Forward strategy refers to following the trace links to the artifacts that have been derived from the artifact under consideration. In this step, the impact of evolution events is propagated to the main design artifacts (source and target).

3. Applying the changes: At this step, the DWO can be reshaped to adjust to the new semantics incurred by the event. The list of changes is presented to the designer so that she can validate or reject some changes. This last step applies changes using the following rule: `ON <event> TO <construct> THEN <policy>`. The combination of events and annotations determines the policy to be followed for the handling of a potential change. For each annotated concept, the corresponding evolution policy is applied. For some cases, designer intervention is required to decide what should be done. We distinguish the following scenarios:

(i) Event in the DWO: This scenario concerns changes in DWO concepts (to be stored in the $\mathcal{DW}$). We define the evolution policy for each change type and for each design level as follows. We distinguish the schema level from the instance level. Table 1 summarizes the change policies required for each type of change.

If the event is an *addition* of a concept or a role from the $\mathcal{LOD}$ knowledge base, two types of additions are distinguished: adding a new concept C in the $\mathcal{LOD}$ knowledge base, or adding a sub-concept to concept C alimenting the $\mathcal{DW}$ (i.e., extending the hierarchy of concepts). In the first case, no traces are identified because the new concept cannot be included in any ETL process. If the designer chooses to include this concept, this scenario requires redefining the design, which has been treated in existing studies (Section 2.1). We focus on the second scenario of addition. In this case, the ETL traces are identified. This addition event is then propagated to the design artifacts detected by Algorithm 1 (requirements and DWO). The new concept is added to the identified requirement parameters. For instance, requirement R1 analyzes the number of research works for each university, and thus involves ResearchWork in its parameters. The addition of a new type of ResearchWork will impact this requirement. The new concept is added as an instance of the meta-concept “TargetElement” of the “TraceLink” concerning R1. The new concept may include a set of instances, which requires the re-execution of the identified ETL traces (processes).

For example, the ETL process ResearchWork$_{DWO}$ $\equiv$ Union (Work$_{S1}$,Work$_{S2}$), and Source S2 adds a new concept C (subclass of Work) in its hierarchy. After identifying the traces, the impacted ETL processes must consider this new concept. The maintenance of the ETL process at the implementation level will be described in scenario (iii).

If the event requires adding instances *Inst*, the approach first identifies the set of classes C, such as (*Inst rdf:type C*). The class is described in the input file if it is well-formed, otherwise it is possible to retrieve it from the SPARQL endpoint of the $\mathcal{LOD}$ dataset. Then, the same process as described previously is achieved: the ETL processes involved, the set of classes of DWO, and the set of requirements concerned by the change are identified as traces. The ETL processes identified are re-executed to load the new instances.

Modification events may concern the schema level (concepts and role) or the instance level. The modification of concepts and roles requires the identification of traces concerning the ETL processes, the set of DWO concepts, and the set of requirements. The concept/role modified is updated in these traces. In cases where the type of role changes, the update of the ETL process requires the validation of the designer and its re-execution. The modification of instances (Inst) requires first identifying the set of classes C (*Inst rdf:type* C) of the $\mathcal{LOD}$ knowledge base, then follows a similar process described for concepts and roles.

Deletion events require more careful treatment since they can impact some requirements and thus analysis possibilities. The validation of the designer is required for these events. A deletion event of a concept or a role first identifies the set of traces (ETL processes, DWO, and requirements) impacted by the event. The concept/role is then deleted from ETL processes and from the requirement parameters. The modified ETL processes are first validated by the designer and are then re-executed. The deletion of instances is easier to achieve. It requires the identification of traces and the re-execution of ETL processes concerned by the event.

Table 1. Change management policies. DWO: $\mathcal{DW}$ ontology.

Event on Construct	Policy
Add Concept (in hierarchy)/Add Role	Identify ETL traces (ETL activities) Identify DWO traces Identify requirements traces Add Concept/Role to requirements parameters Re-execute ETL activities
Add new Concept	Redefine Design
Add Instance	Get Class (rdf:type) Identify ETL traces (ETL activities) Identify DWO traces Identify requirements traces Re-execute ETL activities
Modify Concept/Role C	Identifiy ETL traces (ETL activities) Identify DWO traces Identify requirements traces Update C in traces if type of Role changes: Re-execute ETL activities (requires validation)
Modify Instance	Get Class (rdf:type) Identify ETL traces (ETL activities) Identify DWO traces Identify requirements traces Re-execute ETL activities
Delete Concept/Role C	Identify ETL traces (ETL activities) Identify DWO traces Identify requirements traces Delete ETL activities containing C (requires validation) Delete C from requirements Re-execute ETL activities
Delete Instance	Get Class (rdf:type) Identify ETL traces (ETL activities) Identify DWO traces Identify Requirements traces Re-execute ETL activities

(ii) Event in requirements constructs: An addition event in the requirements indicates the addition of a new requirement (performed by the designer). In this case, the requirement and its definition (defined in input file) are added in the traceability model. If the requirement needs new concepts, this is managed in the first scenario (i). Modification events are handled similarly by modifying related concepts in the traceability model. A deletion event is handled by the deletion of the requirement in the traceability model (without the deletion of the concerned concepts). For all the events, the traces are automatically updated using the reasoning rule defined.

(iii) Event in ETL processes: An evolution event in the ETL process may concern any ETL activity (input, output, or ETL operator) performed by the designer. All types of events require the re-execution of the new/modified ETL process. This can be achieved in two ways: (i) the re-execution of the ETL process concerned. Each ETL scenario is coded as PL\SQL stored procedures in the DBMS (Oracle in our case). (ii) The use of the Protege Plugin for Oracle Database (https://protegewiki.stanford.edu/wiki/Protege_4_Plugin_for_Oracle_Database), which provides the ability to view, edit, and save ontologies in Oracle Database with Protege Desktop (Figure 5). This approach can be more suitable, as it provides a visualization tool associated to traces. Note that this option may not be available for all semantic DBMSs, which do not provide plugins for ontology editors.

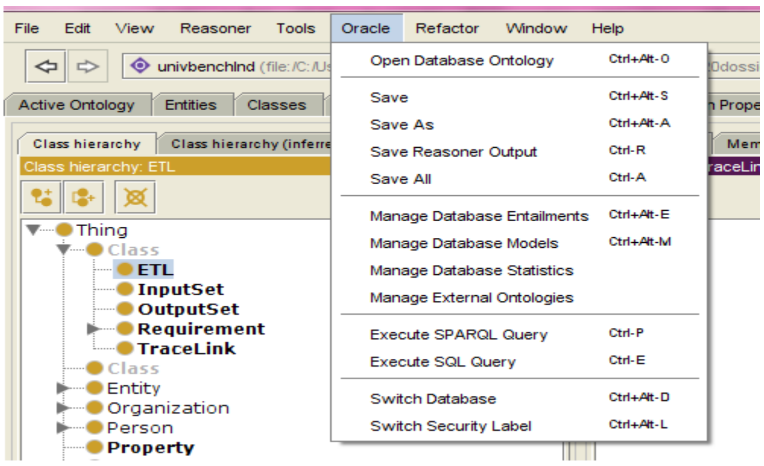


Figure 5. Protege plugin for the Oracle semantic database management system (DBMS).

The deletion of an ETL process requires the validation of the designer if it concerns its output concept, since this will imply a concept in the $\mathcal{DW}$ that will not be alimented by instances. The consistency of the DWO is checked using a reasoner. Note that evolution events may also concern a new translation rule. Our model stores the translation rules. If new rules are needed (in the case of a new storage design model), they can easily be stored in the proposed model to be executed. However, these events require the implementation of the rules to be executed in the DBMS. These evolution events are not implemented in our approach, but they are supported by the proposed model.

6. Case Study and Experiments

We conducted a set of experiments using the LUBM benchmark (http://swat.cse.lehigh.edu/projects/lubm/) related to the University domain. LUBM provides an ontology schema called Univ-Bench which describes universities, departments, the activities that occur at them (courses, research activities, programs, publications, etc.), and the actors related to these activities (students, teachers, researchers, etc.). LUBM also provides a data generator tool (called UBA) used to create data over Univ-Bench ontology in the unit of a university. UBA provides synthetic data with varying size. We used these data-sets to simulate internal sources. We considered Univ-Bench as the DWO (Figure 6). We also considered $\mathcal{LOD}$ as external sources, which were: DBpedia (http://wiki.dbpedia.org/), Yago (https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/), Scholarlydata (http://www.scholarlydata.org/dumps/), and Thomson Reuters (https://permid.org/download) $\mathcal{LOD}$. We used the 14 queries of LUBM to express requirements. We defined ETL processes for each $\mathcal{DW}$ concept from an internal source and from at least one $\mathcal{LOD}$ source.

The $\mathcal{LOD}$ datasets are usually available as ntriples files. DBpedia extracts structured information from Wikipedia, interlinks it with other $\mathcal{LOD}$, and freely publishes the results on the Web using Linked Data and SPARQL. DBpedia release processes are heavy-weight, and releases are sometimes based on data that are several months old. DBpedia-Live solves this problem by providing a live synchronization method based on the update stream of Wikipedia. The set of changes are provided as ntriples files (http://live.dbpedia.org/changesets/) that we considered as input files to test our approach. We considered 60 files (20 files for addition events, 20 for deletion, and 20 files for modification events) chosen so that they included concepts defined in Univ-Bench ontology. The distribution of events in these files is almost 80% events related to instances and 20% related to schema changes. In order to balance the set of events, we considered Bdpedia and other

$\mathcal{LOD}$ (Yago, Scholarlydata and Thomson Reuters) by downloading datasets (ntriples) related to the University domain (e.g., Author, AcademicJournal, AcademicPress, ResearchProject, Book) matching with Univ-Bench conceptualization, that we considered as triples randomly concerned by the three types of events (addition/deletion/modification). For instance, we considered the AcademicPress concept and its instances, downloaded from Yago $\mathcal{LOD}$ as a new concept enriching the hierarchy of Organization concept of Univ-Bench ontology (schema addition event). All the steps of our approach were implemented in a Java script that uses OWL API for interacting with the ontology and the $\mathcal{LOD}$. We implemented this minimal prototype only for defining the input sources and running the propagating and evolution algorithms. We used Protege editor to visualize the DWO and Oracle Database Management System (RDF Semantic Graph 12c version (https://docs.oracle.com/database/121/RDFRM/title.htm)) to implement the $\mathcal{DW}$ (Figure 7).

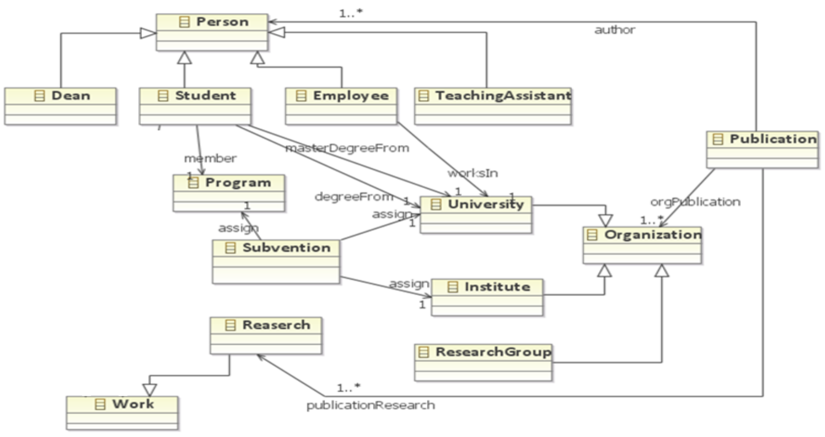
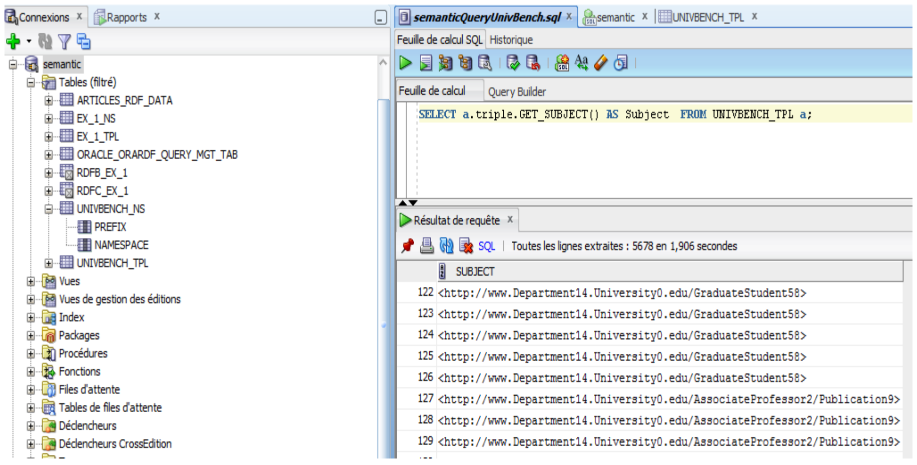


Figure 6. DWO schema (Lehigh University BenchMark, LUBM).


Connexions × Rapports ×

semantic
Tables (filtré)
ARTICLES_RDF_DATA
EX_1_NS
EX_1_TPL
ORACLE_ORARDF_QUERY_MGT_TAB
RDFB_EX_1
RDFC_EX_1
UNIVBENCH_NS
PREFIX
NAMESPACE
UNIVBENCH_TPL
Vues
Vues de gestion des éditions
Index
Packages
Procédures
Fonctions
Files d'attente
Tables de files d'attente
Déclencheurs
Déclencheurs CrossEdition

semanticQueryUnivBench.sql × semantic × UNIVBENCH_TPL ×

Feuille de calcul SQL Historique

Feuille de calcul Query Builder

```
SELECT a.triple.GET_SUBJECT() AS Subject  FROM UNIVBENCH_TPL a;
```

Résultat de requête ×

SQL | Toutes les lignes extraites : 5678 en 1,906 secondes

```
    SUBJECT
122 <http://www.Department14.University0.edu/GraduateStudent58>
123 <http://www.Department14.University0.edu/GraduateStudent58>
124 <http://www.Department14.University0.edu/GraduateStudent58>
125 <http://www.Department14.University0.edu/GraduateStudent58>
126 <http://www.Department14.University0.edu/GraduateStudent58>
127 <http://www.Department14.University0.edu/AssociateProfessor2/Publication9>
128 <http://www.Department14.University0.edu/AssociateProfessor2/Publication9>
129 <http://www.Department14.University0.edu/AssociateProfessor2/Publication9>
```

Figure 7. $\mathcal{DW}$ implementation in Oracle semantic DBMS.

The first experiment shows the amount of triples (inserted, deleted, or modified) in the $\mathcal{DW}$ after applying the approach for change management. The purpose of this experiment was to illustrate the impact of the evolution management at the instance level of the $\mathcal{DW}$. Figure 8 shows the amount of triples asserted, the amount of triples inferred (using the ontology reasoner Pellet), and the amount of triples inferred using the reasoning rule we defined. We could assess the effort gain of a designer using the highlighting of affected concepts compared to the situation where this process had to be performed manually.

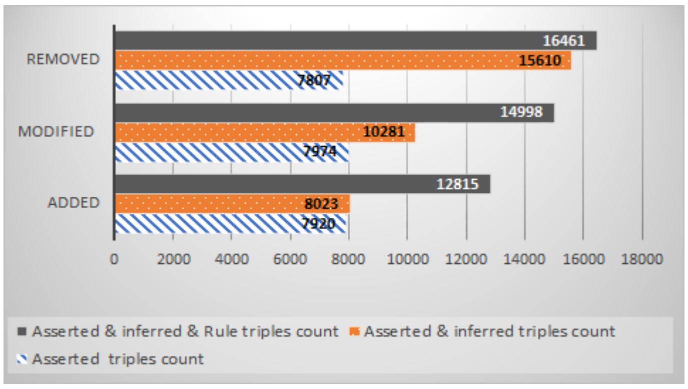


Figure 8. Amount of triples (asserted and inferred) affected by the changes.

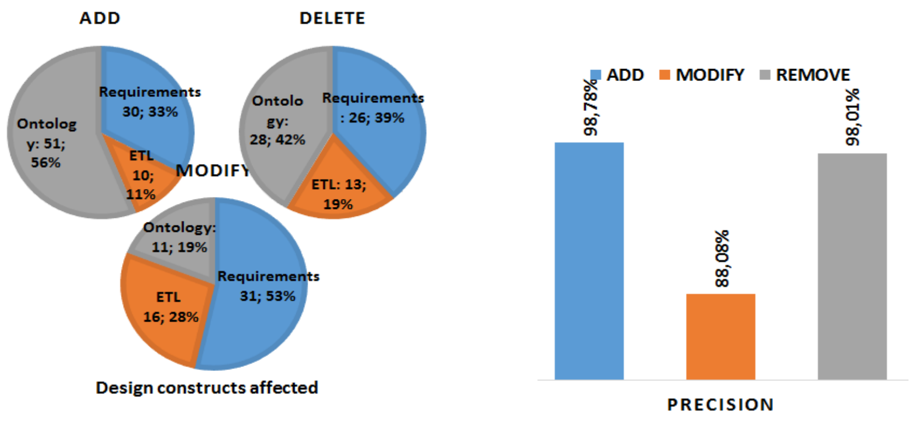
Figure 9a illustrates the amount of nodes (number of nodes and percentage) impacted by changes at different design levels: requirements, DWO, and ETL. We considered the set of concepts (including the traces) and roles (attributes and relationships) as nodes. The goal of this experiment was to illustrate the impact of co-evolution management at the schema level of the $\mathcal{DW}$, and the number of design constructs affected by changes. The designer knowledge of these constructs and their visualization is an integral part of the co-evolution management.

Figure 9b illustrates the precision of the process proposed for managing the changes. The precision was calculated as follows: the number of instances affected by the changes/total number of instances that should be affected (from the $\mathcal{LOD}$). Since there is no "standard" for the precision measure in our study context, we used study [50] to define this measure. We adapted the proposed precision measure to our context of $\mathcal{DW}$ co-evolution management. We can conclude from the figure that the $\mathcal{DW}$ could be effectively adapted to different kinds of events. Exceptions regarding the three types of events are due to the fact that for some concepts from the $\mathcal{LOD}$ that had to be added/deleted/modified, our process did not find a match with the corresponding concepts of the DWO.

As explained in the previous section, the process for propagating the changes starts by projecting the change events on the ETL processes, because they describe the link between the sources and the $\mathcal{DW}$. During the parsing of files, our matching is strictly terminological (term matching). For instance, for the event <Event Add on Construct Concept>, the process tries to find if concept C is used in the ETL processes and then identifies the DWO concepts and the requirements concerned by this event. Because this matching is terminological, some concepts were missed (e.g., AcademicJournal is part of the evolution events that should be detected as a subclass of Publication). The matching can be enriched with other types of ontology matching mechanisms (semantic, conceptual, etc.), which is planned in further research.

Finally, we analyzed the set of requirements that were impacted to check if they can be satisfied after change propagation. The set of requirements can be translated to queries reflecting the analysis performed on the $\mathcal{DW}$. We noticed that even for requirements where deletion events occurred, they could be answered after applying the changes. This is because the set of requirements are those of LUBM, referring to concepts from local sources that we extended to reference concepts from external sources. In the case of addition, modification, or deletion, the results of queries may change but requirements remain satisfied.

Note that comparing these results with other state-of-the-art papers is difficult to achieve because there is no study managing $\mathcal{DW}$ co-evolution, and in the context of $\mathcal{DW}$ evolution, most studies focus on the automation of the evolution process and not the precision quality of the results. In the context of *database* evolution management, we consider [50] as the most detailed study—their precision results were above 90%.



(**a**) (**b**)

Figure 9. (**a**) Design constructs affected by the changes and (**b**) Precision of the evolution process.

7. Conclusions

Building a $\mathcal{DW}$ with internal and external data published in $\mathcal{LOD}$ formats requires managing the evolution of $\mathcal{DW}$s that deal with "open world" sources and their specific characteristics. We proposed and implemented a co-evolution management approach that identifies the impact of changes semantically, for all design stages of the $\mathcal{DW}$. Our approach relies on a coherent maintenance of traces. The goal of the approach is to provide a complete and semantic view of the real impact of the evolution event on the $\mathcal{DW}$. Our approach was tested using different $\mathcal{LOD}$, LUBM benchmark, Protege editor to visualize the impact of changes, and Oracle semantic DBMS to implement the $\mathcal{DW}$. Our future research will include: (i) automatic detection of the most relevant changes in $\mathcal{LOD}$ that can be *synchronously* triggered in the $\mathcal{DW}$. (ii) Management of fine-grained changes like constraints and multidimensional events. (iii) Managing the evolution events on the decisional application level (OLAP queries and operations). (vi) Managing different versions of the $\mathcal{DW}$ to track the most relevant changes.

Author Contributions: Conceptualization, S.K.; Funding acquisition, L.B.; Methodology, S.K.; Writing-original draft, S.K.

Funding: This research received no external funding.

Acknowledgments: This research work was 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 (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) of Nouvelle Aquitaine region. The FEDER-PLAIBDE project is directed by the lead manager of aYaline (https://www.ayaline.com/) company, and the laboratories partners: L3i (http://l3i.univ-larochelle.fr/) laboratory of La Rochelle University (France) and LIAS laboratory of ENSMA (France).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Abelló, A.; Romero, O.; Pedersen, T.B.; Berlanga, R.; Nebot, V.; Aramburu, M.J.; Simitsis, A. Using semantic web technologies for exploratory OLAP: A survey. *IEEE Trans. Knowl. Data Eng.* **2015**, *27*, 571–588. [CrossRef]
2. Gruber, T.R. A translation approach to portable ontology specifications. *Knowl. Acquis.* **1993**, *5*, 199–220. [CrossRef]
3. Golfarelli, M. Data Warehouse Life-Cycle and Design. In *Encyclopedia of Database Systems*; Springer: New York, NY, USA, 2009; pp. 658–664.
4. Ali, S.M.; Wrembel, R. From conceptual design to performance optimization of ETL workflows: Current state of research and open problems. *Int. J. Very Large Data Bases* **2017**, *26*, 777–801.

5. Calvanese, D.; Giacomo, G.; Lembo, D.; Lenzerini, M.; Poggi, A.; Rosati, R.; Ruzzi, M. Data Integration through *DL-Lite*$_A$ Ontologies. In Proceedings of the International Workshop on Semantics in Data and Knowledge Bases, Nantes, France, 29 March 2008; pp. 26–47.
6. Khouri, S.; Bellatreche, L.; Jean, S.; Ait-Ameur, Y. Requirements driven data warehouse design: We can go further. In Proceedings of the International Symposium On Leveraging Applications of Formal Methods, Imperial, Corfu, Greece, 8–11 October 2014; pp. 588–603.
7. Khouri, S.; Boukhari, I.; Bellatreche, L.; Jean, S.; Sardet, E.; Baron, M. Ontology-based structured web data warehouses for sustainable interoperability: Requirement modeling, design methodology and tool. *Comput. Ind.* **2012**, *63*, 799–812.
8. Romero, O.; Abelló, A. Automating multidimensional design from ontologies. In Proceedings of the ACM Tenth International Workshop on Data Warehousing and OLAP, Lisbon, Portugal, 9 November 2007; pp. 1–8.
9. Mazon, J.N.; Trujillo, J. Enriching data warehouse dimension hierarchies by using semantic relations. In *Flexible and Efficient Information Handling*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 278–281.
10. Anderlik, S.; Neumayr, B.; Schrefl, M. Using Domain Ontologies as Semantic Dimensions in Data Warehouses. In Proceedings of the International Conference on Conceptual Modeling, Florence, Italy, 15–18 October 2012; pp. 88–101.
11. Skoutas, D.; Simitsis, A. Ontology-based conceptual design of ETL processes for both structured and semi-structured data. *Int. J. Semant. Web Inf. Syst.* **2007**, *3*, 1–24. [CrossRef]
12. Skoutas, D.; Simitsis, A.; Sellis, T. Ontology-driven conceptual design of ETL processes using graph transformations. *J. Data Semant.* **2009**, *5530*, 120–146.
13. Simitsis, A.; Skoutas, D.; Castellanos, M. Representation of conceptual ETL designs in natural language using Semantic Web technology. *Data Knowl. Eng.* **2010**, *69*, 96–115. [CrossRef]
14. Khouri, S.; Bellatreche, L. DWOBS: Data warehouse design from ontology-based sources. In Proceedings of the International Conference on Database Systems for Advanced Applications, Hong Kong, China, 22–25 April 2011; pp. 438–441.
15. Bizer, C. The emerging web of linked data. *IEEE Intell. Syst.* **2009**, *24*, 87–92. [CrossRef]
16. Meimaris, M.; Papastefanatos, G.; Vassiliadis, P.; Anagnostopoulos, I. Efficient Computation of Containment and Complementarity in RDF Data Cubes. In Proceedings of the Extending Database Technology, Bordeaux, France, 15–16 March 2016; pp. 281–292.
17. Meimaris, M.; Papastefanatos, G.; Vassiliadis, P.; Anagnostopoulos, I. Computational methods and optimizations for containment and complementarity in web data cubes. *Inf. Syst.* **2018**, *75*, 56–74. [CrossRef]
18. Abelló Gamazo, A.; Gallinucci, E.; Golfarelli, M.; Rizzi Bach, S.; Romero Moral, O. Towards exploratory OLAP on linked data. In Proceedings of the 24th Italian Symposium on Advanced Database Systems, Lecce, Italy, 19–22 June 2016; pp. 86–93.
19. Etcheverry, L.; Vaisman, A.; Zimányi, E. Modeling and querying data warehouses on the semantic web using QB4OLAP. In Proceedings of the Data Warehousing and Knowledge Discovery, Munich, Germany, 2–4 September 2014; pp. 45–56.
20. Baldacci, L.; Golfarelli, M.; Graziani, S.; Rizzi, S. QETL: An approach to on-demand ETL from non-owned data sources. *Data Knowl. Eng.* **2017**, *112*, 17–37. [CrossRef]
21. Deb Nath, R.P.; Hose, K.; Pedersen, T.B. Towards a Programmable Semantic Extract-Transform-Load Framework for Semantic Data Warehouses. In Proceedings of the ACM Eighteenth International Workshop on Data Warehousing and OLAP, Melbourne, VIC, Australia, 19–23 October 2015; pp. 15–24.
22. Kämpgen, B.; Harth, A. Transforming statistical linked data for use in OLAP systems. In Proceedings of the 7th International Conference on Semantic Systems, Graz, Austria, 7–9 September 2011; pp. 33–40.
23. Berkani, N.; Bellatreche, L.; Benatallah, B. A value-added approach to design BI applications. In Proceedings of the International Conference on Big Data Analytics and Knowledge Discovery, Porto, Portugal, 5–8 September 2016; pp. 361–375.
24. Matei, A.; Chao, K.; Godwin, N. OLAP for Multidimensional Semantic Web Databases. In *Enabling Real-Time Business Intelligence*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 81–96.
25. Saad, R.; Teste, O.; Trojahn, C. OLAP manipulations on RDF data following a constellation model. In Proceedings of the 1st International Workshop on Semantic Statistics, Sydney, Australia, 21–25 October 2013.

26. Kämpgen, B.; O'Riain, S.; Harth, A. Interacting with Statistical Linked Data via OLAP Operations. In Proceedings of the Extended Semantic Web Conference, Heraklion, Greece, 27 May–31 June 2012; pp. 87–101.
27. Vassiliadis, P.; Zarras, A.V.; Skoulis, I. Gravitating to rigidity: Patterns of schema evolution—And its absence—in the lives of tables. *Inf. Syst.* **2017**, *63*, 24–46. [CrossRef]
28. Manousis, P.; Vassiliadis, P.; Zarras, A.; Papastefanatos, G. Schema Evolution for Databases and Data Warehouses. In *European Business Intelligence Summer School*; Springer: Barcelona, Spain, 2015; pp. 1–31.
29. Quix, C. Repository Support for Data Warehouse Evolution. *DMDW* **1999**, *99*, 14–15.
30. Kaas, C.; Pedersen, T.B.; Rasmussen, B. Schema evolution for stars and snowflakes. In Proceedings of the 6th International Conference on Enterprise Information Systems, Porto, Portugal, 14–17 April 2004; pp. 425–433.
31. Fan, H.; Poulovassilis, A. Schema evolution in data warehousing environments—A schema transformation-based approach. In Proceedings of the International Conference on Conceptual Modeling, Shanghai, China, 8–12 November 2004; pp. 639–653.
32. Hoang, D.T.A. Impact Analysis for On-Demand Data Warehousing Evolution. In Proceedings of the European Conference on Advances in Databases and Information Systems, Vienna, Austria, 20–23 September 2011; pp. 280–285.
33. Papastefanatos, G.; Vassiliadis, P.; Simitsis, A.; Vassiliou, Y. What-if analysis for data warehouse evolution. In Proceedings of the International Conference on Data Warehousing and Knowledge Discovery, Regensburg, Germany, 3–7 September 2007; pp. 23–33.
34. Jovanovic, V. Data Warehouse and Master Data Management Evolution. A Meta-Data-Vault Approach. *Issues Inf. Syst.* **2014**, *15*, 14–23.
35. Papastefanatos, G.; Vassiliadis, P.; Simitsis, A.; Vassiliou, Y. Policy-regulated management of ETL evolution. In *Journal on Data Semantics XIII*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 147–177.
36. Muñoz, L.; Mazón, J.N.; Trujillo, J. A family of experiments to validate measures for UML activity diagrams of ETL processes in data warehouses. *Inf. Softw. Technol.* **2010**, *52*, 1188–1203. [CrossRef]
37. El Akkaoui, Z.; Zimányi, E.; Mazón López, J.N.; Trujillo Mondéjar, J.C. A BPMN-based design and maintenance framework for ETL processes. *Int. J. Data Warehous. Min.* **2013**, *9*, 46–72. [CrossRef]
38. Papastefanatos, G.; Vassiliadis, P.; Simitsis, A.; Vassiliou, Y. Design Metrics for Data Warehouse Evolution. In Proceedings of the International Conference on Conceptual Modeling, Barcelona, Spain, 20–24 October 2008; Volume 5231, pp. 440–454.
39. Papastefanatos, G.; Vassiliadis, P.; Simitsis, A.; Vassiliou, Y. Metrics for the Prediction of Evolution Impact in ETL Ecosystems: A Case Study. *J. Data Semant.* **2012**, *1*, 75–97. [CrossRef]
40. Nadal, S.; Romero, O.; Abelló, A.; Vassiliadis, P.; Vansummeren, S. An Integration-Oriented Ontology to Govern Evolution in Big Data Ecosystems. In Proceedings of the Workshops of the EDBT/ICDT 2017 Joint Conference, Venice, Italy, 21–24 March 2017.
41. Jovanovic, P.; Romero, O.; Simitsis, A.; Abell, A.; Mayorova, D. A requirement-driven approach to the design and evolution of data warehouses. *Inf. Syst.* **2014**, *44*, 94–119. [CrossRef]
42. Mens, T.; Wermelinger, M.; Ducasse, S.; Demeyer, S.; Hirschfeld, R.; Jazayeri, M. Challenges in software evolution. In Proceedings of the Principles of Software Evolution, Lisbon, Portugal, 5–6 September 2005; pp. 13–22.
43. Skoulis, I.; Vassiliadis, P.; Zarras, A.V. Growing up with stability: How open-source relational databases evolve. *Inf. Syst.* **2015**, *53*, 363–385. [CrossRef]
44. Etzlstorfer, J.; Kapsammer, E.; Schwinger, W. On the Evolution of Modeling Ecosystems: An Evaluation of Co-Evolution Approaches. In Proceedings of the 5th International Conference on Model-Driven Engineering and Software Developmen, Porto, Portugal, 19–21 February 2017; pp. 90–99.
45. Karagiannis, D.; Buchmann, R.A. Linked open models: Extending linked open data with conceptual model information. *Inf. Syst.* **2016**, *56*, 174–197. [CrossRef]
46. Calvanese, D.; De Giacomo, G.; Lenzerini, M.; Nardi, D.; Rosati, R. A Principled Approach to Data Integration and Reconciliation in Data Warehousing. In Proceedings of the International Workshop on Design and Management of Data Warehouses, Heidelberg, Germany, 14–15 June 1999; p. 16.
47. Berkani, N.; Bellatreche, L.; Khouri, S. Towards a conceptualization of ETL and physical storage of semantic data warehouses as a service. *Clust. Comput.* **2013**, *16*, 915–931. [CrossRef]

48. Khouri, S.; Berkani, N.; Bellatreche, L. Tracing data warehouse design lifecycle semantically. *Comput. Stand. Interfaces* **2017**, *51*, 132–151. [CrossRef]
49. Roussakis, Y.; Chrysakis, I.; Stefanidis, K.; Flouris, G.; Stavrakas, Y. A flexible framework for understanding the dynamics of evolving RDF datasets. In Proceedings of the International Semantic Web Conference, Bethlehem, PA, USA, 11–15 October 2015; pp. 495–512.
50. Manousis, P.; Vassiliadis, P.; Papastefanatos, G. Automating the adaptation of evolving data-intensive ecosystems. In Proceedings of the International Conference on Conceptual Modeling, Hong Kong, China, 11–13 November 2013; pp. 182–196.