Graph-Based ETL Processes For Warehousing Statistical Open Data

Alain Berro[1], Imen Megdiche[1], Olivier Teste[1]
[1] *IRIT UMR 5505, University of Toulouse, CNRS, INPT,*
UPS, UT1, UT2J, 31062 TOULOUSE Cedex 9
{*berro, megdiche, teste*}*@irit.fr*

Keywords: Statistical Open Data, RDF Graphs, Tables Extraction, Holistic Integration, Multidimensional Schema

Abstract: Warehousing is a promising mean to cross and analyse Statistical Open Data (SOD). But extracting structures, integrating and defining multidimensional schema from several scattered and heterogeneous tables in the SOD are major problems challenging the traditional ETL (Extract-Transform-Load) processes. In this paper, we present a three step ETL processes which rely on RDF graphs to meet all these problems. In the first step, we automatically extract tables structures and values using a table anatomy ontology. This phase converts structurally heterogeneous tables into a unified RDF graph representation. The second step performs a holistic integration of several semantically heterogeneous RDF graphs. The optimal integration is performed through an Integer Linear Program (ILP). In the third step, system interacts with users to incrementally transform the integrated RDF graph into a multidimensional schema.

1 INTRODUCTION

Crossing and analysing Statistical Open Data (SOD) in a data warehouse is a promising way to derive several insights and reuse these miner sources of information. Except that Open Data' characteristics make them unaffordable with traditional ETL (Extract-Transform-Load) processes. Indeed, an important part[1] of governmental SOD holds into spreadsheets disposing structurally and semantically heterogeneous tables. Moreover these sources are scattered across multiple providers and even in the same provider which hampers their integration.

To cope with these problems, we propose RDF-based ETL processes composed of three steps which adapt the extract, transform and load traditional ETL steps. The first step gives a solution to the automatic data table extraction and annotation problem. The second one gives a solution to the automatic holistic data integration problem. The third step focuses on an incremental multidimensional schema definition from the integrated RDF graphs. In this latter, we aim to maintain the re-usability of processed SOD.

RDF Graph-Based ETL Processes. Our ETL processes are based on manipulating RDF graphs. They are depicted in Fig.1 as follows:

[1]http://fr.slideshare.net/cvincey/opendata-benchmark-fr-vs-uk-vs-us

- Step 1: "Extraction and Annotation", it takes as input flat SOD spreadsheets and provides as output unified RDF instance-schema graphs. The main challenge is to automatically identify and restructure data from spreadsheets tables in order to be able to reuse them. We propose a workflow to automatically perform the ETL extraction phase. Each activity in the workflow extracts and annotates table components guided by a table anatomy ontology then constructs iteratively the vertices and edges of the output graph. Finally, users validate automatic annotations and the output graphs are stored in a staging area.

- Step 2: "Holistic Integration", it takes as input structural parts of several SOD RDF graphs and generates as output an integrated RDF graph and

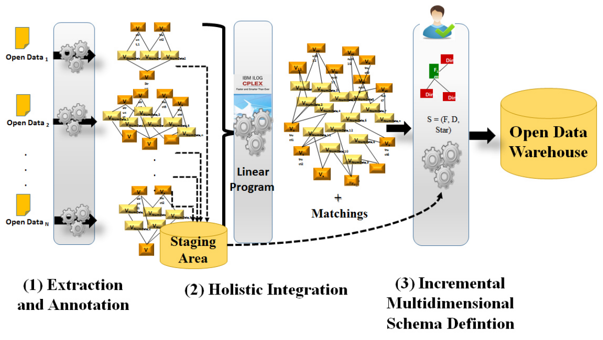


Figure 1: Graph-Based ETL Processes

the underlying matchings. The integrated graph is composed of collapsed and simple vertices connected by simple edges. Collapsed vertices reference groups of matched vertices. Not matched ones remain simple vertices. We propose a linear program to perform an automatic and optimal holistic integration. Users can validate and/or adjust the proposed solution. The linear program maximizes semantic and syntactic similarity between the labels of the vertices of the input graphs. It also resolves conflictual matching situations involving different graphs' structures by ensuring strict hierarchies (Malinowski and Zimányi, 2006) and preserving logical edges directions.

- Step 3: "Incremental Multidimensional Schema Definition", it takes as input an integrated graph and generates as output a multidimensional schema. An interactive process is established between users' actions and system. On the one hand, users' actions consist on incrementally identifying multidimensional components such as dimensions, parameters, hierarchies, fact and measures. On the other hand, system interacts with users' actions by transforming identified components into a multidimensional schema.

Motivating Example. The case study deals with an agricultural company in England aiming at launching a new project on cereal production. The company is interested on analysing cereal production activity in the previous years in UK. With a simple search on the UK governmental provider https://www.gov.uk/, the company retrieves relevant informations for this topic. For instance the link[2] provides several detailed sources for cereal production by cereal type, by farm size, by farm type, and soon. These sources suffer from the different problems presented above. They are: (i) scattered into different spreadsheets even in the same link, (ii) structurally heterogeneous since the spreadsheets can have one or several tables disposed randomly, and (iii) semantically heterogeneous due to the different analysis axis influencing the agricultural statistics.

The following sections describe the three steps of our approach respectively in sections 2, 3, 4.

[2]https://www.gov.uk/government/statistical-data-sets/structure-of-the-agricultural-industry-in-england-and-the-uk-at-june

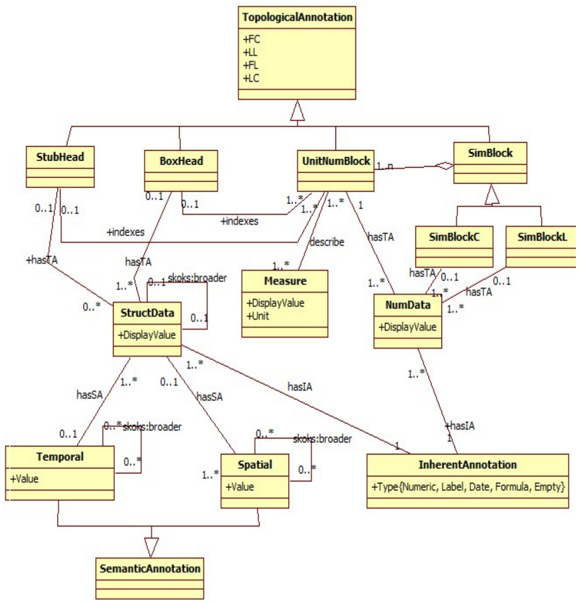


Figure 2: The Open Data Tables Annotating Concepts

2 EXTRACTION AND ANNOTATION

In this section, we describe a workflow performing extraction, annotation and transformation of flat open data tables into RDF instance-schema graphs.

2.1 The Tables Annotating Ontology

In order to annotate tables, we have extended the basic table anatomy described by (Wang, 1996). A table contains a body (a set of numerical data) indexed by structural data which are StubHead (row headers) and/or BoxHead (column headers).

We propose, as depicted in Fig.2, a formalisation of the ontology describing table annotations. We have constructed this ontology manually. It forms a support for the automatic construction of each Open Data source ontology as it will be detailed in the next section. Each table is composed of two types of data: (i) "StructData" which means concepts (sequence of words) able to express schema tables and (ii) "NumData" which means statistical or quantitative data. For each data type, we associate three overlapping annotation types as follows:

- Inherent Annotation (IA) describes the low level data types namely *Numeric*, *Label*, *Formula* and *Date*.

- Topological Annotation (TA) describes name and location of tables' parts. As depicted in Fig.2,

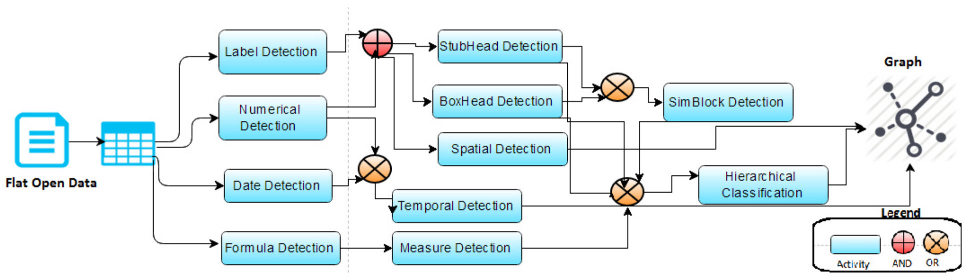


Figure 3: The Workflow Activities Description

the class "TopologicalAnnotation" has four properties FL, LL, FC, LC (First Line, Last Line, First Column, Last Column) representing respectively the landmarks of the component in the table. This class has different sub-classes: (i) "UnitNumBlock" which is the body of table composed it self by "NumData" and describes a measure, (ii) "StubHead" and "BoxHead" which are both composed of "StructData" and indexing "UnitNumBlock" and (iii) "SimBlock" composed of "UnitNumBlock". The "SimBlock" has two sub-classes: "SimBlockC" (a block composed of UnitNumBlock indexed by the same StubHead and different BoxHead) and "SimBlockL" (a Block composed of UnitNumBlock indexed by the same BoxHead and different StubHead).

- Semantic Annotation (SA) describes the semantic class of data, we focus on Temporal (Year, Month,..) and Spatial (Region, GPS coordinates,..) classes. The different types of concepts will be related with a "skos:broader"[3] relationship to express the order between types.

2.2 The Workflow Activities Description

We propose, as depicted in Fig.3, a workflow activities transforming input flat open data (encapsulating tables) into enrich annotated graphs. Each activity will perform, in the same time, the table components detection, ontology-guided annotation and graph construction.

The first workflow activity consists of converting flat open data into a matrix M which encodes low level cells' types namely Numeric, Label, Date and Formula. The vertices and relations for Inherent Annotations are created in G_i. Then, four activities, related to search all types of blocks, are executed. The numerical detection activity extracts all the UnitNumBlock, creates in G_i all the UnitNumBlock instances vertices with their landmarks properties, creates NumData instances vertices and finally makes relations (i.e annotates) between each NumData and the UnitNumBlock encapsulating it. The label (resp. date, formula) detection activity extracts all blocks containing concepts (resp. dates and formula) but does not add vertices into G_i.

Thereafter, BoxHead and StubHead detection activities use the detection results of numerical and label activities. To identify the BoxHead (respectively StubHead) of each UnitNumBlock, our algorithm consists of a backtracking search of the first line (respectively column) of labels situated above (respectively on the left) the UnitNumBlock beginning the search from the UnitNumBlock.FL (respectively UnitNumBlock.FC). BoxHead, StubHead and StructData instances vertices contained into these blocks are added to G_i. Then the relationships between these latter are created into G_i. Finally, indexing relationships between UnitNumBlock, StubHead and BoxHead are added into G_i.

Similar block detection activity can be executed after Stubhead or Boxhead activities. It groups disjoint UnitNumBlock having either the same BoxHead (SimBlockC type) or the same StubHead (SimBlockL type). Spatial detection activity uses the result of label and numerical detections. It uses predefined lists[4] to annotate instances present in the detected blocks. In G_i, the found instances are added and annotated with their appropriate types. Hierarchical relationships are expressed by the property skos:broder (a hierarchical link indicating that a concept is more general than another concept) between all spatial concepts.

Temporal detection uses date detection or numerical detection results. For temporal concepts, we refer to the generic temporal graph defined by (Mansmann and Scholl, 2007). We use regular expressions to identify instances of each temporal entity. The measure detection activity uses the date detection and relies on users to textually input measures names and units.

The last activity is the hierarchical relationships detection (skos:broader relations between StructData

[3]http://www.w3.org/2009/08/skos-reference/skos.html

[4]www.geonames.org

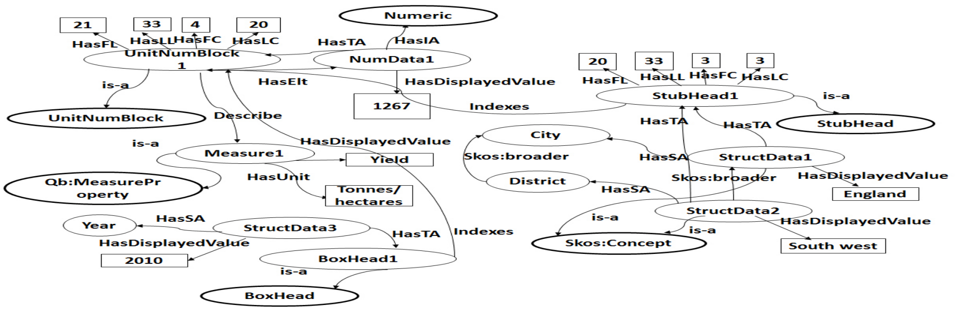


Figure 4: An excerpt from a resulting RDF Open Data Graph

as shown in Fig.2). Regarding the impact of complex hierarchies (Malinowski and Zimányi, 2006) in the summarizability problems, we focus on constructing graphs with non complex hierarchies by applying a set of rules (we will not detail them in this paper). We propose two complementary alternatives for hierarchical classification: (i) concept classification using annotations detected by the workflow activities and (ii) concept classification using data mining techniques to complement the first one. The first alternative is based on the arrangement of the numerical blocks which can determine hierarchical relationships between StubHead or BoxHead concepts. The second alternative concerns Stubhead or Boxhead concepts. We cross two conceptual classification techniques and we select the relevant classes resulting from them. The first technique is lattices (Birkhoff, 1967) which does not consider semantic aspects and generates formal contexts formed by concepts and attributes. The second technique is RELEVANT (Bergamaschi et al., 2011). It clusters a set of concepts and provide relevant attributes for each cluster.

An example of Open Data RDF graph An Open Data Extraction Tool (ODET) has been implemented to validate our approach (Berro et al., 2014). It performs the workflow described above. Fig.4 represents an excerpt of the output graph resulting from ODET for one source from our running example. This source contains statistics on wheat production per city, district and year. As depicted in Fig.4, the UnitNumBlock describes the yield measure which has as unit Tonnes/Hectares. The StubHead is composed of StructData, for instance we have StuctData1 with value "England" and StructData2 with value "South West". StructData1 has the semantic annotation city and StructData2 has the semantic annotation district. The relation skos:broader between city and district in the ontology has been instantiated in the instance-schema graph between StructData1 and StructData2. The BoxHead1, as an instance of the BoxHead class, is the topological annotation of the StructData3 which has as value 2010 and as semantic annotation Year.

3 HOLISTIC OPEN DATA GRAPH INTEGRATION

This section is devoted to present our holistic integration approach based on the integer linear programming technique.

3.1 Pre-Matching Phase

This phase takes as input a set of N graphs $G_i = (V_i, E_i)$ $i \in [1,N], N \geq 2$ representing only structural schema elements which are StubHead, BoxHead, spatio-temporal concepts and enrichment concepts. The output of this phase is composed of: (1) $\sum_{i=1}^{N-1}(N-i)$ similarity matrices and (2) N direction matrices representing the hierarchical relationships between structural vertices.

Similarity matrices computation : We compute $\sum_{i=1}^{N-1}(N-i)$ similarity matrices denoted $Sim_{i,j}$ of size $n_i \times n_j$ defined between two different graphs G_i and G_j, $\forall i \in [1, N-1]$, $j \in [i+1, N]$. Each matrix encodes similarity measures. We define the similarity measure as the maximum between two known similarity distances : (i) the syntactic measure Jaccard (Jaccard, 1912) and (ii) the semantic measure Wup (Wu and Palmer., 1994). As we deal with N input graphs, similarity measure are computed between all combination of all pairwise vertices concepts v_{i_k} and v_{j_l} belonging to two different input graphs G_i and G_j such as $i \in [1, N-1]$, $j \in [i+1, N]$.

$$Sim_{i,j} = \{sim_{i_k,j_l}, \forall k \in [1,n_i], \forall l \in [1,n_j]\}$$

Direction matrices computation : We compute a set of N direction matrices Dir_i of size $n_i \times n_i$ defined for each graph G_i, $\forall i \in [1,N]$. Each matrix encodes edges directions and defined as follows:

$$Dir_i = \{dir_{i_{k,l}}, \forall k \times l \in [1,n_i] \times [1,n_i]\}$$

$$dir_{i_{k,l}} = \begin{cases} 1 & \text{if } e_{i_{k,l}} \in E_i \\ -1 & \text{if } e_{i_{l,k}} \in E_i \\ 0 & \text{otherwise} \end{cases}$$

3.2 Matching Phase

The idea of the LP4HM matcher consist of extending the maximum-weight graph matching problem with additional logical implication constraints. These latter stand for constraints on matching setup and graph structure. Fig.5 depicts three abstract graphs G_1, G_2 and G_3, and similarities between graph vertices (not shown similarities are considered as null values).

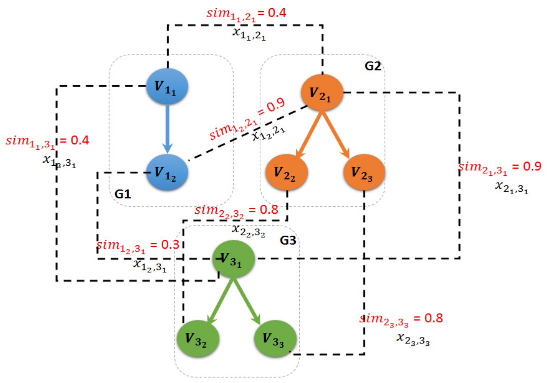


Figure 5: An example of holistic graph matching problem

Decision Variable We define a single decision variable which expresses the possibility to have or not a matching between two vertices belonging to two different input graphs. For each G_i and G_j, $\forall i \in [1,N-1]$, $j \in [i+1,N]$, x_{i_k,j_l} is a binary decision variable equals to 1 if the vertex v_{i_k} in the graph G_i matches with the vertex v_{j_l} in the graph G_j and 0 otherwise.

Logical Implication Constraints A logical implication (Plastria, 2002) is expressed as follows: for x_i a binary variable for all i in some finite set I and y a binary variable, if $x_i = 0$ for all $i \in I$ then $y = 0$ which is also equivalent to $y \leq \sum_{i \in I} x_i$.

The first type of constraints belongs to Matching Setup (MS). MS1 encodes the matching cardinality in particular we focus on 1:1 matching (Rahm and Bernstein, 2001). MS2 encodes the matching threshold in particular the model should not match vertices whose similarity is inferior than a given threshold.

MS1 (Matching Cardinality) Each vertex v_{i_k} in the graph G_i could match with at most one vertex v_{j_l} in the graph G_j, $\forall i \times j \in [1,N-1] \times [i+1,N]$.

$$\sum_{l=1}^{n_j} x_{i_k,j_l} \leq 1, \quad \forall k \in [1,n_i]$$

MS2 (Matching Threshold) We setup similarity threshold *thresh* in order to get better matching results. Our model encodes the threshold setup in the following constraint : $\forall i \times j \in [1,N-1] \times [i+1,N]$ and $\forall k \times l \in [1,n_i] \times [1,n_j]$

$$sim_{i_k,j_l}\, x_{i_k,j_l} \geq thresh\, x_{i_k,j_l}$$

The second type of constraints belongs to graph structure. GS1 constraint generates strict hierarchies and GS2 generates simple edge directions.

GS1 (Strict hierarchies) This constraint allow us to resolve non-strict hierarchy problem (Malinowski and Zimányi, 2006) which is an important issue in multidimensional schema definition. Hence, users will save time in resolving such problems in the multidimensional schema definition step. $\forall i \times j \in [1,N-1] \times [i+1,N]$ and $\forall k \times l \in [1,n_i] \times [1,n_j]$:

$$x_{i_k,j_l} \leq x_{i_{pred(k)},j_{pred(l)}}$$

GS2 (Edge Direction) The purpose of this constraint is to prevent the generation of conflictual edges. $\forall i \times j \in [1,N-1] \times [i+1,N]$ such as $i \neq j$ and $\forall k,k' \in [1,n_i]\ \forall l,l' \in [1,n_j]$

$$x_{i_k,j_l} + x_{i_{k'},j_{l'}} + (dir_{i_{k,k'}} dir_{j_{l,l'}}) \leq 0$$

Objective Function The objective of our model is to maximize the sum of the similarities between matched vertices. The objective function is expressed as follows

$$\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \sum_{k=1}^{n_i} \sum_{l=1}^{n_j} sim_{i_k,j_l}\, x_{i_k,j_l}$$

The resolution of the linear program of the example in Fig.5 gives the following solution:

- v_{1_2} matches with v_{2_1} with $sim_{1_2,2_1} = 0.9$
- v_{1_2} matches with v_{3_1} with $sim_{1_2,3_1} = 0.3$
- v_{2_1} matches with v_{3_1} with $sim_{2_1,3_1} = 0.9$
- v_{2_2} matches with v_{3_2} with $sim_{2_2,3_2} = 0.8$
- v_{2_3} matches with v_{3_3} with $sim_{2_3,3_3} = 0.8$
- the objective function value is 3.7

4 INCREMENTAL DATA WAREHOUSE SCHEMA DEFINITION

In this section, we describe an incremental process to define a multidimensional schema from the integrated Open Data RDF graphs. The system will interact with users' actions in order to generate a script to create the data warehouse schema and populate data, in the same time the integrated RDF graph will be refined with the QB4OLAP vocabulary proposed in (Etcheverry et al., 2014). QB4OLAP is more complete than RDF Data Cube Vocabulary [5] (RDF QB) to describe multidimensional schema. This twofold output is motivated by : (i) the interest of analysing populated data with OLAP tools, (ii) the interest to keep reuse of integrated Open Data tables in other scenarios for instance link them with available Statistical Open Data or Linked Open Data in general.

4.1 A Conceptual Multidimensional Schema and an Augmented RDF Graph

Two types of output are expected from our incremental process. The outputs are described as follows :

A Conceptual Multidimensional Schema. We will use the conceptual specifications of a generic multidimensional schema (a group of star schema) proposed in the works of (Ravat et al., 2008). This generic model is the most suitable according to the alternatives that can generate the integrated graph (for instance several facts which share the same dimensions). A multidimensional schema E is defined by (F^E, D^E, $Star^E$) such as :

- $F^E = \{F_1, \ldots, F_n\}$ a finite set of facts;
- $D^E = \{D_1, \ldots, D_m\}$ a finite set of dimensions;
- $Star^E$: $F^E \rightarrow 2^{DE}$ a function associating for each fact a set of dimensions.

A dimension $D \in D^E$ is defined by (N^D, A^D, H^D):

- N^D a dimension name,
- $A^D = \{a_1^D, \ldots, a_u^D\} \cup \{id^D, All\}$ a set of attributes,
- $H^D = \{H_1^D, \ldots, H_v^D\}$ a set of hierarchies.

A hierarchy denoted $H_i \in H^D$ is defined by (N^{H_i}, $Param^{H_i}$, $Weak^{H_i}$) such as:

- N^{H_i} a hierarchy name,
- $Param^{H_i} = < id^D, p_1^{H_i}, \ldots, p_{v_i}^{H_i}, All >$ an ordered set of attributes called parameters which represent the relevant dimension' graduations, $\forall k \in [1 \ldots v_i], p_k^{H_i} \in A^D$,
- $Weak^{H_i}$: $Param^{H_i} \rightarrow 2^{A^D - Param^{H_i}}$ is a function associating each parameter to one or several weak attribute.

A fact denoted $F \in F^E$ is defined by (N^F, M^F):

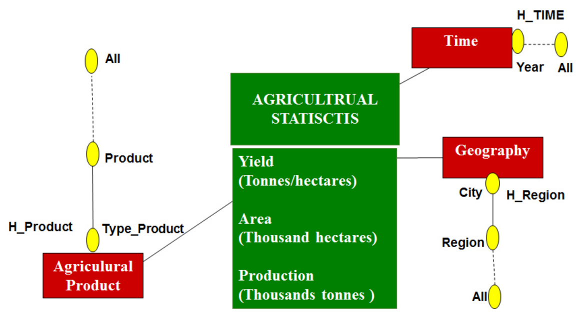
- N^F a fact name,
- $M^F = f_1(m_1^F), \ldots, f_w(m_w^F)$ a set of measures associated to an aggregation function f_i.

Example 1. The example depicted in Fig.6(a) is the expected multidimensional schema from the phase 3 of our process applied to the motivating example. For the sake of simplicity, fact F_i, dimension D_i and hierarchy H_i are cited by their names. The multidimensional schema E is composed of one fact and three dimensions. E = ({AgriculturalStatistics}, {Time, Geography, AgriculturalProduct}, {Star(AgriculturalStatistics)= {Time, Geography, AgriculturalProduct}) The dimensions are : (Time, {Year, All}, {HTime}), (Geography, {City, Region, All}, {HRegion}) and (AgriculturalProduct, {TypeProduct, Product , All}, {HProduct}). The fact is : (Agricultural-Statistics, {AVG(Yield), SUM(Area), AVG(Area), SUM(Production), AVG(Production)})

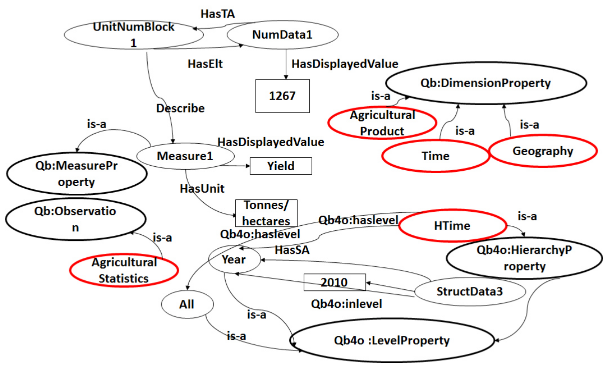
An Augmented RDF Graph. The integrated RDF graph resulting from the matching phase will be augmented with concepts belonging to the QB4OLAP vocabulary proposed in (Etcheverry et al., 2014). This latter uses the basic concepts of RDF QB and extends it with concepts representing hierarchies, hierarchies levels, aggregation functions, cardinalities. The correspondences between the multidimensional schema and the concepts of the RDF graph are as follows:

- E is an instance of $qb : dataStructureDefinition$.
- F^E is an instance of a $qb : Observation$ class.
- D^E is an instance of a $qb : DimensionProperty$ class.
- $Star^E$: $F^E \rightarrow 2^{DE}$ is an instance of $qb : DataSet$.
- A^D is an instance of $qb4o : levelProperty$
- The members of A^D are instances of $qb : leveMumber$.
- H^D is an instance of $qb4o : HierarchyProperty$.
- $Param^{H_i}$ is an instance of $qb4o : LevelProperty$.
- $Weak^{H_i}$ is an instance of a $qb : AttributeProperty$.

[5] http://www.w3.org/TR/vocab-data-cube/



(a) An example of Multidimensional Schema



(b) An Augmented RDF Graph

Figure 6: Expected outputs for the incremental schema definition step

- M^F is an instance of $qb : MeasureProperty$ and the associated aggregation function f_i is an instance of $qb4o : AggregateFunction$.

The Fig.6(b) shows an excerpt of an expected augmented RDF graph by following the correspondences explained above.

The processes we use to obtain the two expected results will be explained in the next section.

4.2 An incremental process to define multidimensional components

We propose an incremental process in which users should begin by identifying dimensions and their components (hierarchies, attributes, members) then they define the fact and its measures. Three actors interact in this process: (1) users who interact with the integrated graph via different functions to incrementally define multidimensional components, (2) graph transformation (GT) which interacts with user' actions in order to enrich the integrated graph with OLAP4QB objects and properties, (3) data warehouse script generator (SG) which interacts with user' actions in order to generate the script describing the multidimensional schema and from which the data warehouse will be populated with SOD.

Dimensions Identification. When a user add dimensions, the GT creates a new vertex with the dimension name as an instance of a qb:dimensionProperty and the SG creates a new table dimension without attributes. Then user add hierarchies to the dimension, but as hierarchies are composed of levels, users have to choose either to add level or to select a level from the graph. If user chooses the action add level, the GT creates a vertex with the hierarchy name instance of qb4o:hierarchyProperty, a vertex for the level instance of qb4o:levelProperty and a property qb4o:haslevel between the hierarchy and the level. If the user chooses the action select level form graph, this level had to be a StructData vertex, it will be changed to become an instance of qb4o:levelProperty and the property qb4o:haslevel will be created between this vertex and the hierarchy level. Then for both actions add level and select level from graph, user have to select members form the graph. The GT will collect members from graph which are StructData. These members are changed into instances of qb:levelMemeber. Properties qb4o:inlevel will be created between the concerned level and its members and visually these vertices will be collapsed in the label vertex. In the same time, the SG updates dimension with its attributes, creates hierarchies with levels and populates dimension attributes with their members. The users have to iterate the described process for all dimensions knowing that when a the task of dimension creation finish, the system can suggest to user potential dimensions. Indeed, since each NumData is related to a path composed of a succession of StructData, the GT deduces the NumData related to the paths belonging to the actual identified dimension and proposes to the users the non yet identified paths related to the NumData as a potential dimension.

Fact Identification. When all dimensions have been identified, the user can proceed to fact and measures identification. When the user creates a fact, the GT creates a vertex with the fact name as an instance of qb:observation. Then he should link dimensions to the fact, and select measures from the graph. So, the SG creates a fact table with its measures and relates it to the created dimensions. GT collects the NumData related to the selected dimensions and measures which are instances of the qb:dataset. Finally SG add these instances to the fact table.

Discussion and future work. In the process described above, the user intervention can be error-prone hence some improvements may be envisaged. For instance, we can inject some of the algorithms proposed by (Romero and Abelló, 2007) in order to semi-automate the identification of dimensions and facts. Moreover, even if we have resolved the problem of complex hierarchies in the first two steps of our ETL processes by generating strict hierarchies, we acknowledge that the summarizabiliy problem (Mazón et al., 2010) is not totally resolved especially completeness and disjointness integrity constraints (Lenz and Shoshani, 1997). Both the works of (Romero and Abelló, 2007) and (Prat et al., 2012) propose solutions applied on ontologies. Compared to (Romero and Abelló, 2007), the authors of (Prat et al., 2012) propose more explicit rules which seems straightforward to be added to our process. Since the construction of our process outputs are performed in parallel, the verification of the summarizabilty constraints in the ontology will be used to check the same constraints in the multidimensional model.

5 CONCLUSIONS

In this paper, we have presented a full RDF graph-based ETL chain for warehousing SOD. This ETL chain takes as input a set of flat SOD containing statistical tables and transforms them into a multidimensional schema through three steps. The first step automatically extracts, annotates and transforms input tables into RDF instance-schema graphs. The second step performs automatically a holistic graph integration through an integer linear program. The third one incrementally defines the multidimensional schema through an interactive process between users and system. The main contributions of our approach are: (i) the unified representation of tables which facilitates their schema discovery and (ii) the extension of the maximum weighted graph matching problem with structural constraints in order to resolve the holistic open data integration problem. In our future works, we aim to train a user study on our approach to measure the efficiency of automatic detections and integrations, and to measure the difficulties that users may encounter when they define incrementally the multidimensional schema from visual graphs.

REFERENCES

Bergamaschi, S., Guerra, F., Orsini, M., Sartori, C., and Vincini, M. (2011). A semantic approach to etl technologies. *Data and Knowledge Engineering*, 70(8):717 – 731.

Berro, A., Megdiche, I., and Teste, O. (2014). A content-driven ETL processes for open data. In *New Trends in Database and Information Systems II - Selected papers of the 18th East European Conference on Advances in Databases and Information Systems and Associated Satellite Events, ADBIS 2014 Ohrid, Macedonia*, pages 29–40.

Birkhoff, G. (1967). *Lattice Theory*. American Mathematical Society, 3rd edition.

Etcheverry, L., Vaisman, A., and Zimnyi, E. (2014). Modeling and querying data warehouses on the semantic web using QB4OLAP. In *Proceedings of the 16th International Conference on Data Warehousing and Knowledge Discovery, DaWaK'14*, Lecture Notes in Computer Science. Springer-Verlag.

Jaccard, P. (1912). The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50.

Lenz, H.-J. and Shoshani, A. (1997). Summarizability in olap and statistical data bases. In *Scientific and Statistical Database Management, 1997. Proceedings.*, pages 132–143.

Malinowski, E. and Zimányi, E. (2006). Hierarchies in a multidimensional model: From conceptual modeling to logical representation. *Data Knowl. Eng.*, 59(2):348–377.

Mansmann, S. and Scholl, M. H. (2007). Empowering the olap technology to support complex dimension hierarchies. *IJDWM*, 3(4):31–50.

Mazón, J.-N., Lechtenbrger, J., and Trujillo, J. (2010). A survey on summarizability issues in multidimensional modeling. In *JISBD*, pages 327–327. IBERGARCETA Pub. S.L.

Plastria, F. (2002). Formulating logical implications in combinatorial optimisation. *European Journal of Operational Research*, 140(2):338 – 353.

Prat, N., Megdiche, I., and Akoka, J. (2012). Multidimensional models meet the semantic web: defining and reasoning on OWL-DL ontologies for OLAP. In *DOLAP 2012, ACM 15th International Workshop on Data Warehousing and OLAP*, pages 17–24.

Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *VLDB JOURNAL*, 10.

Ravat, F., Teste, O., Tournier, R., and Zurfluh, G. (2008). Algebraic and graphic languages for OLAP manipulations. *International Journal of Data Warehousing and Mining*, 4(1):17–46.

Romero, O. and Abelló, A. (2007). Automating multidimensional design from ontologies. In *Proceedings of the ACM Tenth International Workshop on Data Warehousing and OLAP*, DOLAP '07, pages 1–8. ACM.

Wang, X. (1996). Tabular abstraction, editing, and formatting. Technical report, University of Waretloo, Waterloo, Ontaria, Canada.

Wu, Z. and Palmer., M. (1994). Verb semantics and lexical selection. In *In 32nd. Annual Meeting of the Association for Computational Linguistics, New Mexico State University, Las Cruces, New Mexico.*, pages 133–138.