Multiscaling a Graph-based Dataspace

Matheus Silva Mota[1], Júlio Cesar dos Reis[1], Sandra Goutte[2], André Santanchè[1]

[1] Institute of Computing – UNICAMP, Brazil
{mota, julio.dosreis, santanche}@ic.unicamp.br
[2] Institute of Biology – UNICAMP, Brazil
froggologist@gmail.com

Abstract. Biologists increasingly need a unified view to understand and discover relationships among data elements scattered along data sources with different levels of heterogeneity. Existing approaches usually adopt ad-hoc *heavy-weight* integration strategies, requiring a costly upfront effort involving a monolithic chain of steps to handle specific formats/schemas, with low or no reuse. This article proposes the conception of a multiscale-based dataspace architecture, called *LinkedScales*. It departs from the notion of integration-scales within a dataspace, and defines a systematic and progressive integration process via graph-based transformations over a graph database. *LinkedScales* aims to provide a homogeneous view of heterogeneous sources, allowing systems to reach and produce different integration levels on demand, going from raw representations (lower scales) towards ontology-like structures (higher scales). We describe inner aspects of the architecture and its transformation process by introducing the Multiscale Transformation Graph, which tracks the transformation process among scales. Although the proposed framework can be applied to several scenarios, this work focuses on the biology domain addressing the organism-centric analysis scenario. Obtained results reveal the viability of the framework and its implementation to integrate relevant resources for the organism-centric scenario.

Categories and Subject Descriptors: H.2.0 [**Database Management**]: General; H.3.0 [**Information Storage and Retrieval**]: General

Keywords: Data Integration, Dataspace, Multiscale, Organism-centric Analysis

1. INTRODUCTION

Data-centric domains as biology are increasingly adopting different systems to produce, store and analyze datasets regarding specific processes and aspects of biological organisms – *e.g.*, experiments, descriptions, collections, simulations, *etc.* However, heterogeneity hampers the integrated exploration of knowledge across systems and research groups [Hey et al. 2009]. Therefore, integration remains a key issue since providing a "big picture" view of data may offer new perspectives and insights for researchers [Elsayed and Brezany 2010; Paton et al. 2012].

This research focuses on a specific integration paradigm known as Dataspaces [Franklin et al. 2005]. It advocates the advantages of an on-demand lightweight integration to comply with the dynamicity of modern environments, against the classic heavyweight upfront techniques. One of the advantages of on-demand integration is the ability of readily shaping the final outcome according to present needs.

A key issue with on-demand integration, addressed in this investigation, refers to the long chain of steps from source to target. In one extreme, biologists want to treat knowledge at a conceptual level, handling data in an integrated fashion. In the other extreme, there are several problem-relevant heterogeneous data sources, comprising files, DBs, ontologies, *etc.* Between both extremes, there might have a spectrum of intermediary integration steps, which are difficult to determine.

In this article, we propose an approach named *LinkedScales*, which aims at splitting the integration steps as discrete scales. Each scale encompasses common aspects and routines related to a specific integration step. The main objective of *LinkedScales* is to go from a source-related lower scale to a user-focused higher scale. Inspired by the layered software architecture, each scale offers to the immediate upper scale a pre-agreed model (interface), encapsulating a given type of heterogeneity of the lower scale. This investigation defines the different scales, formalizing them in a framework based on a graph model. In lower scales, we depart from a myriad of heterogeneous sources available. The upper scales enables to tailor the model according to specific needs, *i.e.*, the integration model fits the user needs, instead of the opposite.

We demonstrate the applicability of our proposal in the biological domain. In such dynamic context, reuse plays a key role and traditional on-demand solutions usually rely on ad-hoc techniques, implementing the entire integration chain. In our proposal, the encapsulation of scales in *LinkedScales* enables to customize only algorithms of a specific scale, reusing the remaining of the chain. Obtained results relying on real-world application scenarios experimenting the approach indicate the adequacy and usefulness of the *LinkedScales* proposal for organism-centric analysis.

The remaining of this article is organized as follows: Section 2 presents the problem in our research scenario and how existing work concerning data integration address it. Section 3 reports on the proposed *Linkedscales* framework. Section 4 details the formalization of the multiscale graph model. Section 5 describes implementation aspects and experiments showing a complete example to illustrate the solution. We also discuss its benefits. Section 6 wraps up the article with conclusions and presents future work.

2. FOUNDATIONS AND RELATED WORK

2.1 Challenges on organism-centric analysis for data integration

Organism-centric analysis refers to an usual approach conducted by biologists in which organisms – *i.e.*, species or taxonomic groups – are the central focus of the analysis and data are integrated around them. A common task faced by biologists conducting an organism-centric research refers to the construction of "views" of data, we call here *profiles* [Washington et al. 2009]. Profiles vary according to the focus of interest, but they can be seen as a subset of descriptive data of organisms selected for a research [Hedges 2002]. The construction of such profiles involves combining data usually fragmented in heterogeneous sources, requiring further efforts from biologists to collect and combine pieces coming from multiple repositories and several files with different formats.

Consider the example of profile illustrated in Fig. 1, defined by biologists interested in validating hypotheses regarding the evolution of "deafness" in frogs. Aiming at understanding why distant phylogenetic groups of frogs lack middle ear structures, biologists want to gather together as profiles data regarding morphological traits, habitat, reproduction mode, acoustics and phylogenetic trees of several species. Morpho-anatomical data would be required to examine whether miniaturisation in frogs lead to the loss of ear structures, while acoustic data would allow testing the co-evolution of mutism and deafness, *etc.* Based on such profiles, biologists might compare organisms in a systematic way and investigate conditions and associations related with the hypotheses.

Phylogenetic data for the target species of the genus *Brachycephalus* (shown in Fig. 1) can be found within the *TreeBASE*[1] repository – where scientists share their experimental data files – as a XML/Nexus file. It contains the phylogenetic tree reconstructed from DNA sequences from a study. Records from IUCN Red List[2] intended for conservation contains data regarding the species habitat in CSV format. Moreover, several phenotipic data can be found from Quaardvark System[3] in Excel format.

[1] http://treebase.org, [2] http://www.iucnredlist.org, [3] http://animaldiversity.ummz.umich.edu/quaardvark

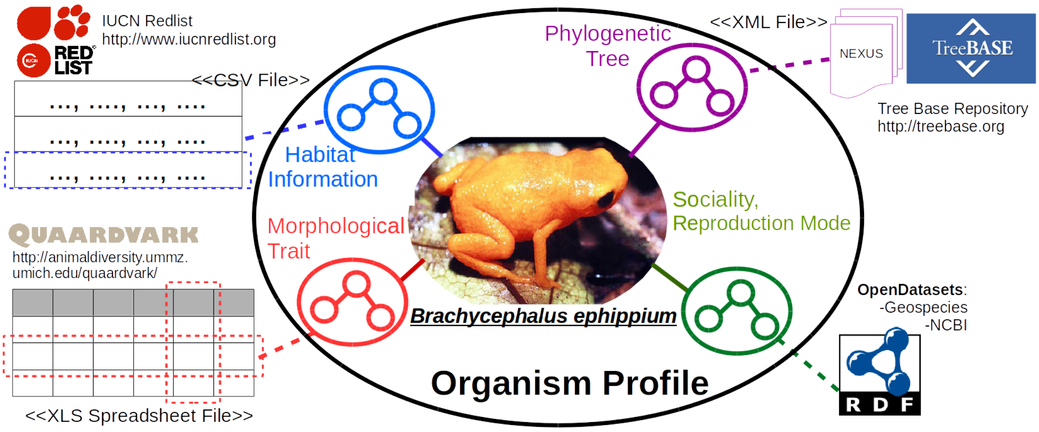


Fig. 1. Profile integrating characteristics scattered across several sources

In this scenario (Fig. 1), biologists spend a lot of time "cutting and pasting" data from each of the sources and organizing them in spreadsheets before any analysis. On the other hand, a systematic integration approach requires several steps of integration, due to the different types of heterogeneity, *i.e.*, different formats (CSV, Excel, Nexus), different structures (tables, trees), different schemas, *etc.* Therefore, the combination of different types of datasets may prove challenging, and the integration of missing data often result in a drastic data trimming and the partial use of the data available. Furthermore, such biological research has an intrinsic dynamism. For instance, biologists may discover during their investigations that other characteristics must be taken into account, which might require further efforts to reflect the new requirements and data on the profiles to make them up-to-date.

2.2 Upfront Data Integration *vs.* The "Pay-as-you-go" Integration

Motivated by the increasingly need of treating multiple and heterogeneous data sources, data integration has been the focus of attention in the database community in the past two decades [Hedeler et al. 2013; Hedeler et al. 2009].

Several data integration strategies have emerged, including federated databases, schema integration and data warehouses [Haas et al. 2002; Rahm and Bernstein 2001]. A common adopted approach relies on providing a virtual unified view under a global schema (GS) [Singh and Jain 2011; Kolaitis 2005]. Within GS-based systems, the data stay in their original data sources – i.e. maintaining their original schemas – and are dynamically fetched and mapped to a global schema under clients' request [Lenzerini 2002; Hedeler et al. 2013]. In a nutshell, applications send queries to a mediator, which relates them into several sub-queries dispatched to wrappers, according to meta-data regarding capabilities of the participating database management systems (DBMSs). Wrappers map queries to the underlying DBMSs and the results back to the mediator, guided by the global schema. Queries are optimized and evaluated according to each DBMS within the set, providing the illusion of a single database to applications [Lenzerini 2002].

The central drawback with such data integration strategy regards the big upfront effort required to produce a global schema definition [Halevy et al. 2006]. As in some domains different DBMSs may emerge and schemas are constantly changing, such costly initial step can become impracticable [Hedeler et al. 2013]. Moreover, several approaches focus on a particular data model (*e.g.*, relational), while new models also become popular [Elsayed et al. 2006]. As proposed in this investigation, our approach supports progressive small integration steps as an alternative to this classical all-or-nothing costly upfront data integration technique.

Since upfront mapping between schemas are labor intensive and scheme-static domains are rare, pay-as-you-go integration strategies have gained momentum. Classical data integration approaches might work successfully when integrating modest numbers of stable databases in controlled environments. Nevertheless, literature still lacks an efficient and definitive solution for scenarios in which schemas often change and new data models must be considered [Hedeler et al. 2013]. In a data integration spectrum, the classical data integration is at the high-cost/high-quality end, while an incremental integration based on progressive small steps starts in the opposite side. Such incremental integration can be continuously refined in order to improve the connections among sources.

The notion of *dataspaces* aims at providing the benefits of the classical data integration approach, but in a progressive fashion way [Halevy et al. 2006; Singh and Jain 2011; Hedeler et al. 2010]. The main argument behind the dataspaces proposal is that, in the current scenario, instead of a long wait for a global integration schema to have access to the data, users would rather to have early access to the data, among small cycles of integration – *i.e.*, if the user needs the data now, some integration is better than nothing.

Dataspaces approach of data integration can be divided in a bootstrapping stage and subsequent refinements. Progressive integration refinements can be based, for instance, on structural analysis [Dong and Halevy 2007], on users' feedback [Belhajjame et al. 2013] or on manual/automatic mappings among sources – if benefits worth such effort. Furthermore, several Dataspace platforms address a variety of specific scenarios, *e.g.*, SEMEX [Cai et al. 2005] and iMeMex [Dittrich et al. 2009] on the private information management context; PayGo [Madhavan et al. 2007] focusing on Web-related sources; and a justice-related dataspace [Dijk et al. 2013].

Although incremental integration approaches have already showed their potentialities, literature still lacks an architecture that systematizes the progressive integration steps and results according to integration aspects, providing provenance and reuse of partial results. Systematization, provenance and reuse are the three pillars of our *LinkedScales* proposal, introduced in next section.

3. LINKEDSCALES FRAMEWORK

LinkedScales refers to a framework that comprises a multiscale graph model – introduced here and formally detailed in the next section – and a data architecture which instantiate the model. It aims at bringing the proposal of multiscale to the data integration chain, systematizing and encapsulating the data regarding integration steps as graph-based scales.

In our approach, the modern tendency towards progressive integration [Halevy et al. 2006] evolves in progressive steps within a shared *"space"*, in which data of several steps coexist, even if not fully integrated. Over time, extra incremental integration steps are made within the space when benefits worth the efforts.

LinkedScales is based on an abstract model that organizes the progressive integration chain as a pile of scales, where the entities in an upper scale are built based on transformations over entities of a lower scale – the granularity and semantics of the entities vary according to the scale. The integration starts on the lowest scale, where all original data sources are ingested and transformed into graphs. Each subsequent scale from this point is a graph derived from the previous scale, taking advantage of the flexibility of graphs to logically represent different structures along the scales. This model allows representing operations within and across the scales as transformation procedures in graphs. Scales are interconnected by an orthogonal graph, supporting traceability among them – *i.e.*, it is possible to "track" sources/targets of transformations between scales.

In order to address a range of applications which share common integration concerns, we propose a *LinkedScales Primary Data Architecture*, defining a starting set of scales, based on previous experiences on data integration [Mota and Medeiros 2013; Bernardo et al. 2013; Miranda and Santanchè 2013].

Each scale of this data architecture emphasizes a different level of integration and its respective abstraction.

Fig. 2 presents an overview of the *LinkedScales Primary Data Architecture*. It depicts four different scales of abstraction aiming at going from the raw data sources (lower scales, containing more details about format and structure) to a conceptual scale (fewer details of format and structure, and focus on domain-specific concepts). From bottom to top, the scales are: (i) *Physical Scale*, (ii) *Logical Scale*; (iii) *Description Scale*; and (iv) *Conceptual Scale*. This primary data architecture was conceived to be extended, *i.e.*, further scales can appear on top of the conceptual scale to define additional domain-related views.

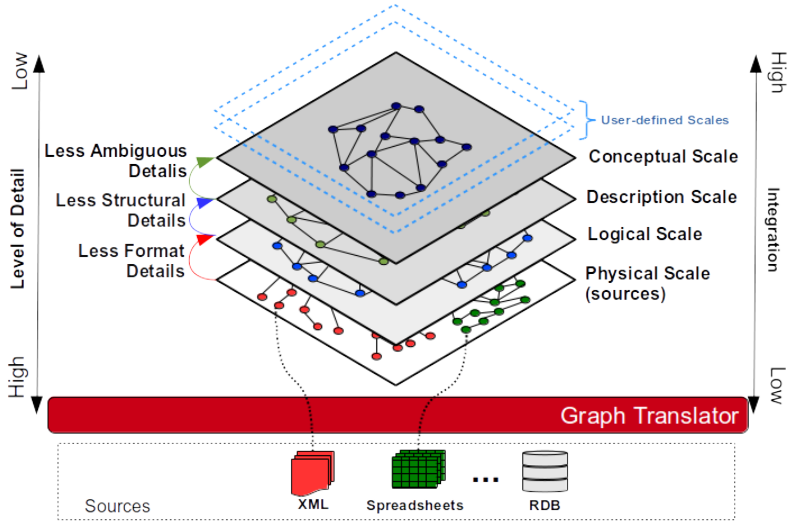


Fig. 2. Overview of the *LinkedScales Primary Data Architecture*

The lowest scale in Fig. 2 – ***Physical Scale*** – aims at representing the different data sources in their original physical format as a graph. The original raw data sources are transformed into a graph by an ingestion procedure (the Graph Translator in the figure) able to read several specialized formats – *e.g.*, Excel, CSV, relational tables, XML – and convert them to an equivalent graph representation. The original structure, format and content of the underlying data sources are reflected in a graph as far as possible. The role of this scale is to homogenize the physical representation, making explicit and linkable elements of the original data within sources.

Based on experiences of a previous work that explores a homogeneous representation model for textual documents independently of formats [Mota and Medeiros 2013], the next scale proposed is the ***Logical Scale***. It offers a common view to data inside similar or equivalent structural models represented in the previous scale. Tables and hierarchical documents are examples of structural models present in the sources containing data regarding organisms. In the previous scale, differences might exist in the representation of a table within a PDF, a table from a spreadsheet and a table within a HTML file, since they preserve specificities of their formats. Within the *Logical Scale*, format specificities disappear and the three tables are represented alike since they refer to the same structural model. This leads to a homogeneous approach to process tables, independently of the way that tables are represented in their original specialized formats.

The ***Description Scale*** emphasizes the content (*e.g.*, labels of elements within an XML document or values in spreadsheet cells) and their relationships. Since models represent relations among data elements in different ways – *e.g.*, a row in a table can represent data concerning the same entity while hierarchical relations in a document represent aggregations – the *Description Scale* reduces all logical models to a single unified one, to shift the focus towards the descriptive content, avoiding heterogeneous models concerns.

The unified model selected for this scale relies on the triple <resource, property, value>, which is usual in several meta-data standards as *Resource Description Framework* (RDF[2]). This scale only unifies the logical model, but still lacks essential properties of a semantic representation like RDF since it does not: distinguish entities, adopt controlled vocabularies to represent descriptive properties or make explicit the semantics of the elements using ontologies. This stands for the role of the next scale.

The highest scale of our data architecture, illustrated in Fig. 2, refers to the ***Conceptual Scale***. It integrates data of the lower scale in a semantic level, exploits the content and relationships between nodes to discover and to make explicit through ontologies their latent semantics. Entities are discovered, deduplicated and related to ontologies as instances of classes, or properties and their values. Therefore, a "textual graph" of the previous scale becomes a graph containing interrelated entities and their properties/values, with explicit semantics supported by ontologies. We also consider that predefined ontologies can be straightly interrelated to this scale, to be linked to the inferred entities.

4. MULTISCALE GRAPH MODEL

This section adopts a formal language to define aspects of the abstract model underlying the *LinkedScales* approach introducing our *Multiscale Graph Model*. It aims at facilitating the understanding of the involved concepts, but, it is not a full-fledged formal definition of the model. We organize three subsections, presenting first the preliminary definitions, followed by the transformation process and the orthogonal transformation graph.

4.1 Preliminary Definitions

As depicted in Figure 3, the *Multiscale Graph Model* contains a sequence of scales $(S_1, S_2, \ldots, S_n)$. It starts from an initial scale S_1 and each subsequent scale S_i is derived from a previous scale S_{i-1}.

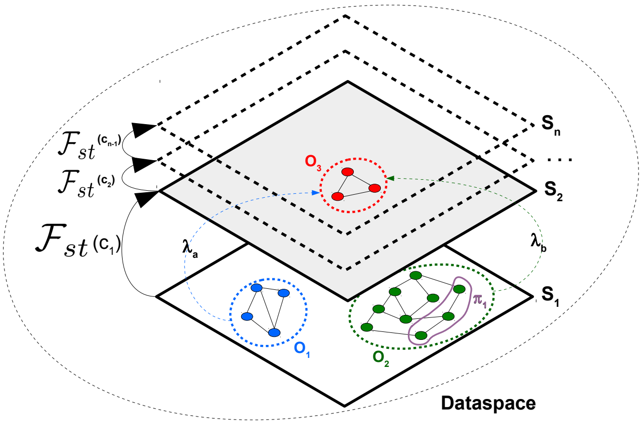


Fig. 3. *LinkedScales* graph model

[2] https://www.w3.org/RDF/

Inspired by the notion of *graph databases*, a ***Scale*** is defined as a finite, edge-labeled, directed graph [Wood 2012; Barceló Baeza 2013; Barceló et al. 2014; Cruz et al. 1987]. Formally, let Σ be a finite alphabet and $\mathcal{V}$ be a countably infinite set of node ids. A scale S over Σ is a pair (V, E), being V a finite set of ***nodes*** and E a finite set of ***edges***, where $V \subseteq \mathcal{V}$ and $E \subseteq V \times \Sigma \times V$. Furthermore, given any two scales $S_i = (V_i, E_i)$ and $S_j = (V_j, E_j)$, where $V_i \subseteq \mathcal{V}$ and $V_j \subseteq \mathcal{V}$, $V_i \cap V_j = \emptyset$.

Given a scale $S = (V, E)$ and two nodes $u, v \in V$ and a label $a \in \Sigma$, an edge $e \in E$ is a triple (u, a, v) indicating a link between u and v with a label a. A ***path*** π in a scale S is a set of edges in E connecting two nodes (initial and final) in V. Therefore, a path connecting a node v_1 and v_m is a sequence of edges $\pi = \{(v_1, a_1, v_2), (v_2, a_2, v_3), \ldots, (v_{m-1}, a_{m-1}, v_m)\}$, where any edge $(v_{i-1}, a_{i-1}, v_i) \in E$ and any end node of an edge in the path matches the initial node in the following edge . An empty path π is a triple (v, ϵ, v), where $v \in V$ and the label is the empty word ϵ; the length of such path, $|\pi| = 0$. The concept of path plays a key role in our transformation process.

A transformation between two scales is defined in terms of transformations of objects inside these scales, *i.e.*, objects are the atomic transformation units. An ***object*** is defined as a set of paths $O = \{\pi_1, \pi_2, \ldots, \pi_r\}$. An object O_h belongs to a scale S_i if all nodes/edges of the paths in O_h are nodes/edges of S_i. Figure 3 depicts three objects and a path, O_1, O_2 belongs to S_1, O_3 belongs to S_2, and the path $\pi_1 \in O_2$.

4.2 Transformation process

LinkedScales is represented as tuple $\mathcal{LS} = (S_i, \Omega, \mathcal{F}_{st})$, where S_i is a scale representing the initial state, $\Omega = \{C_1, C_2, \ldots, C_n\}$ is a sequence of transformation criteria and $\mathcal{F}_{st}$ is a function $\mathcal{F}_{st} : S_i \to S_{i+1}$ which derives a subsequent scale S_{i+1} by applying a transformation criteria C_i over a previous scale S_i. The **transformation** process comprises two steps: match and transform. The *match* step aims at finding paths in the subgraphs of a given scale, while the *transform* step addresses the production of a transformed subgraph in the upper scale. The example illustrated in Fig. 5 shows how an instance of a table (T_1) with a schema and two rows results in two entities (e_4 and e_5), each one containing three *paths* representing RDF-like triples. Such transformation is based on a pattern for matching paths in the input and for creating the corresponding nodes and vertices in the output.

The match and transform operations are encapsulated in the concept of criterion. A ***criteria*** $\mathcal{C}_\alpha$ is a set of criterion $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$. A ***criterion*** λ_a is a pair (m_a, t_a), where m_a is a *match* operation and t_a is a *transform* operation. Similarly to the *SELECT* operator in SQL, a ***match*** operator meets a set of objects of a given scale S_i, while the respective ***transform*** operator derives these objects to produce a graph in S_{i+1}. Algorithm 1 describes how the *scale transformation* function produces an upper scale based on a criteria set.

A pattern in the *match* operation is defined by a regular expression over the graph. Wildcards here are indicating the repetition of sub-patterns. While the wildcard *** indicates a repetition of a given subpath in a sequential disposition – *i.e.*, the beginning of a repeated subpath is connected with the end of the previous one – the wildcard **** indicates a repetition in a parallel disposition – *i.e.*, all repeated subpaths are connected to the same origin. The number of repetitions is constrained by adding the clause $[\alpha..\beta]$, where α and β are optionals minimum and maximum boundaries, respectively.

Fig. 4 visually illustrates the steps in a transformation that aims at producing RDF-like triples from an object representing a table (within the *Logical Scale*), also showing an example of objects matching the *Match* clause and the respective transformation (within the *Description Scale*). The regular expressions related to the input patterns are represented in the left side, using dashed boxes to define the scope of each wildcard. It also illustrates the main difference between the two regular expression wildcards for graphs.

The wildcard *** in π_x^* indicates that each matched object instance can have a sequential repetition of the subpath delimited by the dashed box. The wildcard **** in π_y^{**} indicates that each matched

Algorithm 1 Scale Transformation

```
procedure $\mathcal{F}_{st}(S_i, C_i)$        ▷ Produces a scale $S_{i+1}$ based on a scale $S_i$ and criteria $C_i$
    $S_{i+1} \leftarrow \emptyset$
    for each $\lambda \in C_i$ do

        $\mathcal{O} \leftarrow \lambda_{match}(S_i)$        ▷ Returns all matched objects in $S_i$

        for each object $O \in \mathcal{O}$ do
            $S_{temp} \leftarrow \lambda_{transform}(O)$
            $S_{i+1} \leftarrow (S_{i+1} \cup S_{temp})$
        end for each
    end for each
    return $S_{i+1}$
end procedure
```

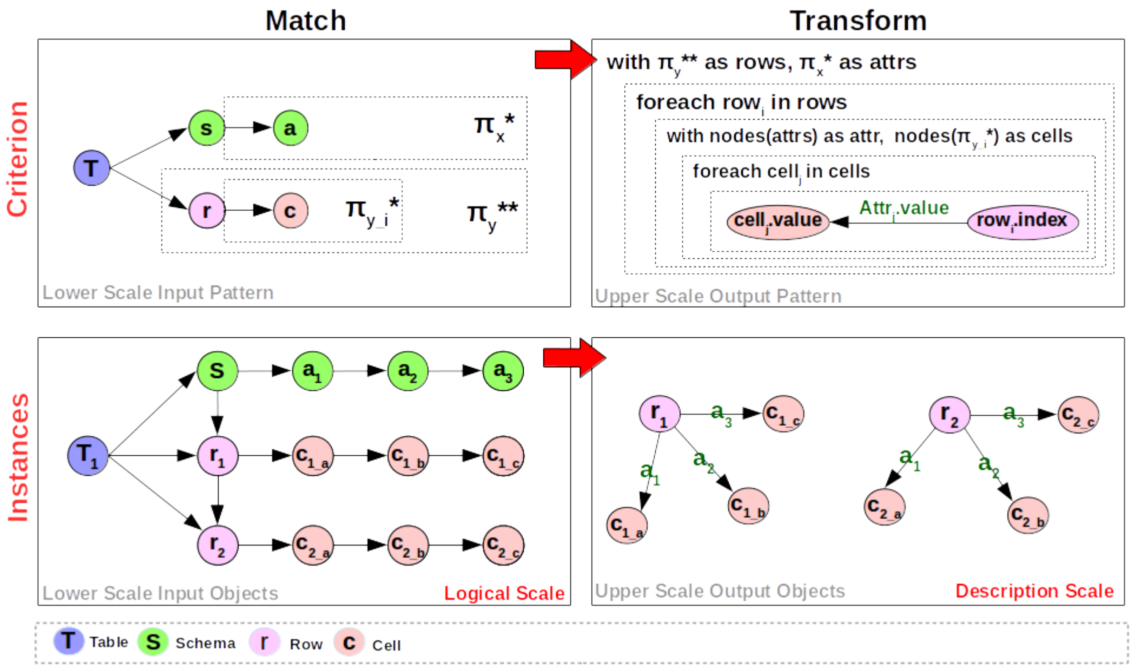


Fig. 4. Example of a match/transform process

instance can have a parallel repetition of the subpath delimited by the dashed box. The resulting subgraphs are connected to the same origin, i.e., the node T. The nested pattern $\pi^*_{y\_i}$ indicates a set of connected sequences of nodes c, where each sequence is connected to the respective origin r of the outer pattern.

Following the example of Fig. 4, the *match* pattern is applied to the lower scale containing a graph representation of a table. The pattern π^*_x matches the sequence of attributes of the schema in the row started by node s. The pattern π^{**}_y matches a set of rows started by nodes r; each row corresponds to a line of the table representing a tuple, formed by a sequence of cells matched by the nested pattern $\pi_{y\_i}*$.

The right box of Fig. 4 illustrates the ***transform*** step of a criterion, using a pseudocode inspired in the Cypher query language[3]. The "*with*" clause defines a scope, which comprises the set of instances matched by a given pattern. For example, the clause "*with* π_y^{**} *as rows*" means that all matched paths for the pattern π_y^{**} will be available in the inner scope of that clause, as instances of a variable *rows*. The inner "*foreach*" clause navigates through each path row_i of *rows*. Subsequently, the inner "*with*" uses the function *nodes*() to return only nodes from the path *attr* and the current row_i. The innermost "*foreach*" navigates through all the cells of the row and links the node corresponding to row_i with a node representing the value of the cell using the corresponding attribute label.

4.3 Multiscale Transformations and the Transformation Graph

For each pair of consecutive scales, there is an orthogonal graph linking the objects of the lower scale to the respective derived objects of the upper scale. The objects of the lower scale are subgraphs defined by the match clause of the criterion, as well as objects of the upper scale are the respective derived subgraphs. Such orthogonal graph is disjoint from the graph containing the data in the scales, and is called Multiscale Transformation Graph (MTG). The MTG fosters traceability of transformations along the integration scales, allowing analysis of provenance, reproducibility, reuse, *etc*.

MTG adopts elements of the *PROV Ontology* (PROV-O) [Lebo et al. 2013]. *Entities* are the sources/targets of transformation in PROV-O and they correspond to *objects* in our model (*cf.* Fig. 5). The transformations between an upper and a lower scales are represented as *Activities*, which correspond to a transformation criterion of our model.

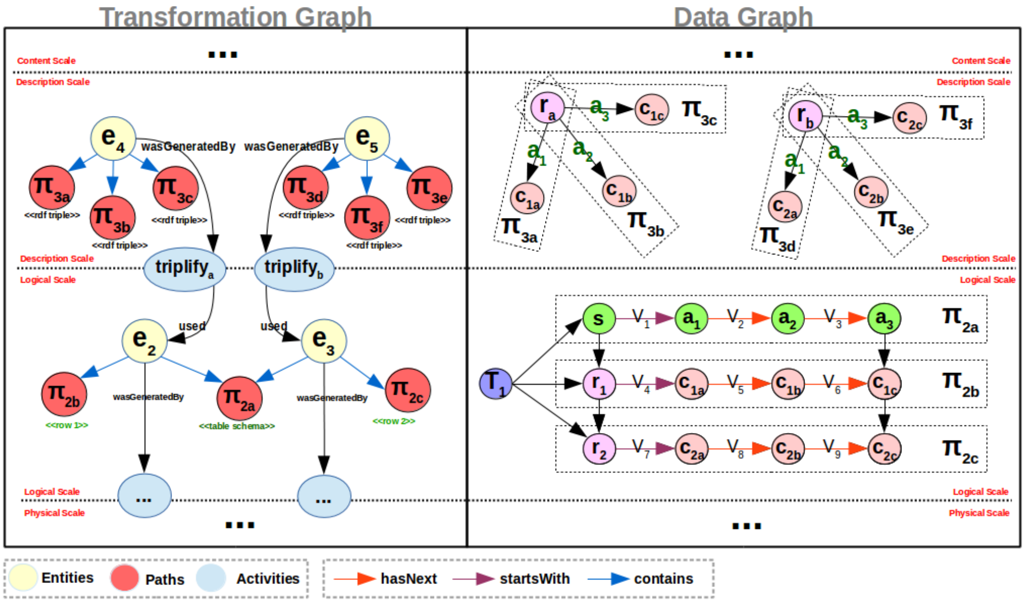


Fig. 5. Example of transformation between two scales and the corresponding MTG

The example illustrated in Fig. 5 shows how an instance of a table with a schema and two rows results in two entities (e_4 and e_5), each one containing three *paths* representing RDF-like triples. Such transformation is based on a pattern for matching paths in the input and for creating the corresponding nodes and vertices in the output.

[3] http://neo4j.com/docs/stable/cypher-query-lang.html

The right column in Fig. 5 (Data Graph) shows an example that meets the patterns specified in Fig. 4 and its respective MTG in the left column (Transformation Graph). The MTG indentifies two *entities* (e_2 and e_3) related to the lower scale based on two respective objects that includes three *paths* (π_{2a}, π_{2b} and π_{2c}). The path π_{2a} refers to schema s of the table. Similarly, paths π_{2b} and π_{2c} refer to the objects representing rows of the table.

The object and respective entity e_2 is composed by the path π_{2b} (the first tuple of the table) and path π_{2a} (the attributes of the schema). The object and respective entity e_3 shares with e_2 the path π_{2a} (attributes of the schema) and also refers to the second tuple of the table (π_{2c}). The entities e_2 and e_3 are the input for the transformation activities $triplify_a$ and $triplify_b$, which "triplificates" table rows. The $triplify$ activities produce objects represented by entities e_4 and e_5, which in turn refers to the paths of the output subgraphs.

5. EXPERIMENTAL SCENARIO: ORGANISM-CENTRIC ANALYSIS VIA LINKEDSCALES

In this section, we describe the implementation of the solution and evaluate its application in a biological scenario, exemplifying the transformation between the scales. We present the whole integration process in a practical scenario, going from the sources to the conceptual scale (organism profiles).

5.1 Implementing the Solution

Several elements and specific technical issues of the proposed framework have being implemented independently [Mota and Santanchè 2015]. In a nutshell, aspects related to the conceptual level were investigated in [Bernardo et al. 2013], while [Miranda and Santanchè 2013] studied how to extract triples as descriptions from different models. Furthermore, [Mota and Medeiros 2013] examinated the problem of handling a multitude of physical formats, converging to a homogeneous one.

Based on the previous implementations, we developed a unified architecture as a framework on top of the Neo4j graph database. A framework called *2graph* for converting resources to graphs in the *Physical Scale* was developed, currently supporting the conversion to graph of CSV, HTML, XML, XLS, XLSX, N3 RDF and ODS – this set of formats was defined as the most relevant formats for biologists in the organism-centric domain. The framework defines a specific module to convert each specialized format to a graph, and was built on top of DDEx [Mota and Medeiros 2013]. It can be extended by plugging new conversion modules.

The graphs of the Scales and the MTG are stored together within a Neo4j database, but logically separated by a different set of labels on nodes and edges. Similarly, nodes and edges from different scales are stored within the same graph but are logically sliced by properties indicating their scales.

The Neo4j database offers a specific graph query language (called Cypher) that supports both reading (match step) and writing (transform step) clauses. Cypher supports SQL update-like queries, which enables to combine reading and writing clauses to create new graphs resulting from a matched input. We are working to automatically map our generic transformation approach described in the previous section to Cypher queries.

Even though our proposal can be extended to other file formats, we are currently focusing on a set of formats defined by biologists as the most relevant for their work (discussed in Section 2.1), *i.e.*, spreadsheets (XLS, XLSX, ODS), HTML tables, CSV files, XML files and textual documents. We have developed a graph framework for ETL named *2graph*[4]. This framework is represented as the "*Graph Translator*" element in Fig. 2.

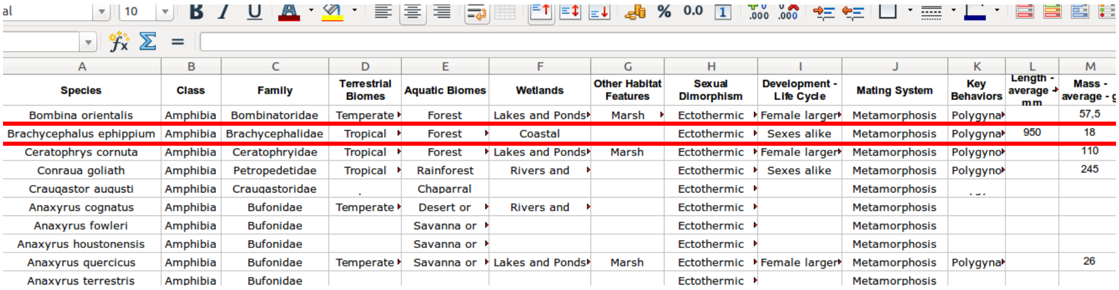
[4] Available at http://www.lis.ic.unicamp.br/ matheus/projects/2graph

5.2 Scenario and Experimental Procedure

In Section 2.1, we presented a scenario of an organism-centric data analysis, in which researchers dynamically produce profiles of living beings integrating characteristics scattered across several sources. The dynamical nature of this task and the heterogeneity of formats, models and schemas on the sources make the progressive incremental integration approach a powerful alternative.

In our investigation, we first collected data corresponding to the biologist's necessities in the scenario. We applied the implemented tools in these data analyzing the transformation results in each scale. We selected relevant examples to illustrate the findings.

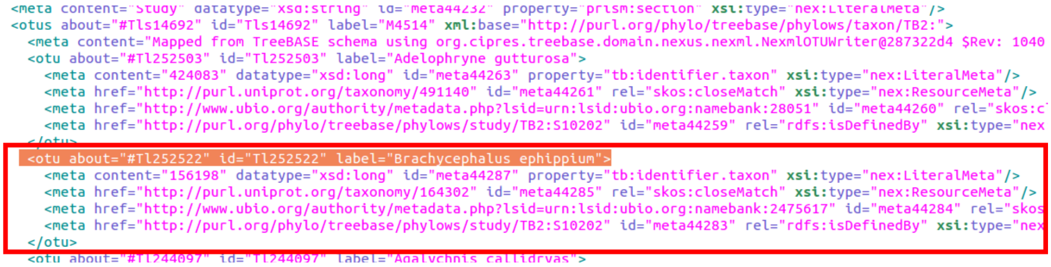
Consider Fig. 6 and Fig. 7 with excerpts of files to be integrated: an XLSX spreadsheet and an XML/NEXUS document, respectively. While the spreadsheet contains morphological traits, behavioral aspects, habitat characteristics *etc.* of several species, the XML/NEXUS file corresponds to the serialization of a phylogenetic tree.



A	B	C	D	E	F	G	H	I	J	K	L	M
Species	**Class**	**Family**	**Terrestrial Biomes**	**Aquatic Biomes**	**Wetlands**	**Other Habitat Features**	**Sexual Dimorphism**	**Development - Life Cycle**	**Mating System**	**Key Behaviors**	**Length - average - mm**	**Mass - average - g**
Bombina orientalis	Amphibia	Bombinatoridae	Temperate	Forest	Lakes and Ponds	Marsh	Ectothermic	Female larger	Metamorphosis	Polygyna		57,5
Brachycephalus ephippium	Amphibia	Brachycephalidae	Tropical	Forest	Coastal		Ectothermic	Sexes alike	Metamorphosis	Polygyna	950	18
Ceratophrys cornuta	Amphibia	Ceratophryidae	Tropical	Forest	Lakes and Ponds	Marsh	Ectothermic	Female larger	Metamorphosis	Polygyno		110
Conraua goliath	Amphibia	Petropedetidae	Tropical	Rainforest	Rivers and		Ectothermic	Sexes alike	Metamorphosis	Polygyno		245
Crauqastor auqusti	Amphibia	Crauqastoridae		Chaparral			Ectothermic		Metamorphosis			
Anaxyrus cognatus	Amphibia	Bufonidae	Temperate	Desert or	Rivers and		Ectothermic		Metamorphosis			
Anaxyrus fowleri	Amphibia	Bufonidae		Savanna or			Ectothermic		Metamorphosis			
Anaxyrus houstonensis	Amphibia	Bufonidae		Savanna or			Ectothermic		Metamorphosis			
Anaxyrus quercicus	Amphibia	Bufonidae	Temperate	Savanna or	Lakes and Ponds	Marsh	Ectothermic	Female larger	Metamorphosis	Polygyna		26
Anaxyrus terrestris	Amphibia	Bufonidae					Ectothermic		Metamorphosis			

Fig. 6. Excerpt of a XLS spreadsheet highlighting the row regarding the species *Brachycephalus ephippium*

Both resources contain data regarding the same set of organisms under investigation, being relevant to build organism profiles. While the red box of Fig. 6 highlights a row of the spreadsheet containing information about the species *Brachycephalus ephippium*, the red box in Fig. 7 points to an XML element – labeled as *OTU* (Operational Taxonomic Unit) – regarding the same species, representing its node in a phylogenetic tree.

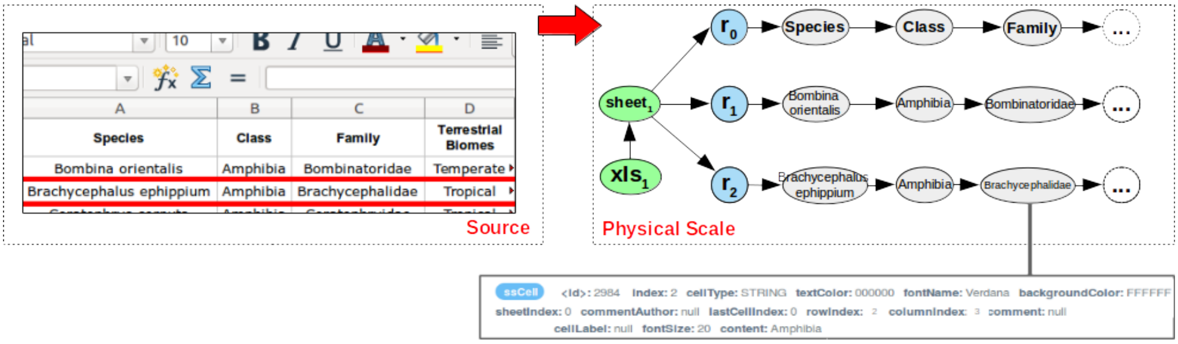


```
<meta content="Study" datatype="xsd:string" id="meta44232" property="prism:section" xsi:type="nex:LiteralMeta"/>
<otus about="#Tls14692" id="Tls14692" label="M4514" xml:base="http://purl.org/phylo/treebase/phylows/taxon/TB2:">
  <meta content="Mapped from TreeBASE schema using org.cipres.treebase.domain.nexus.nexml.NexmlOTUWriter@287322d4 $Rev: 1040
  <otu about="#Tl252503" id="Tl252503" label="Adelophryne gutturosa">
    <meta content="424083" datatype="xsd:long" id="meta44263" property="tb:identifier.taxon" xsi:type="nex:LiteralMeta"/>
    <meta href="http://purl.uniprot.org/taxonomy/491140" id="meta44261" rel="skos:closeMatch" xsi:type="nex:ResourceMeta"/>
    <meta href="http://www.ubio.org/authority/metadata.php?lsid=urn:lsid:ubio.org:namebank:28051" id="meta44260" rel="skos:c
    <meta href="http://purl.org/phylo/treebase/phylows/study/TB2:S10202" id="meta44259" rel="rdfs:isDefinedBy" xsi:type="nex
  </otu>
  <otu about="#Tl252522" id="Tl252522" label="Brachycephalus ephippium">
    <meta content="156198" datatype="xsd:long" id="meta44287" property="tb:identifier.taxon" xsi:type="nex:LiteralMeta"/>
    <meta href="http://purl.uniprot.org/taxonomy/164302" id="meta44285" rel="skos:closeMatch" xsi:type="nex:ResourceMeta"/>
    <meta href="http://www.ubio.org/authority/metadata.php?lsid=urn:lsid:ubio.org:namebank:2475617" id="meta44284" rel="skos
    <meta href="http://purl.org/phylo/treebase/phylows/study/TB2:S10202" id="meta44283" rel="rdfs:isDefinedBy" xsi:type="nex
  </otu>
  <otu about="#Tl244097" id="Tl244097" label="Agalychnis callidryas">
```

Fig. 7. Excerpt of a XML/NEXUS file highlighting the species *Brachycephalus ephippium*

5.3 Ingestion: From the original sources to the physical scale

The first step involves ingesting raw data from the input resources, converting them to a graph representation – see Fig. 10(A). The purpose of the *Physical Scale* is to solve a common initial issue in the data integration pipeline: homogeneous access. The mapping process preserves in the graph as much original format-related information as possible, without homogenization/standardization concerns. For instance, unlike a text-plain CSV file, proprietary spreadsheet formats have substantial extra information, such as, metadata, comments, text formatting, formulas, links, *etc.* In the current implementation, the ingested graphs are stored in a graph database and can be reached by a graph query language. The ingestion module in the system is conducted by the *2Graph* software, as described in Section 5.1.

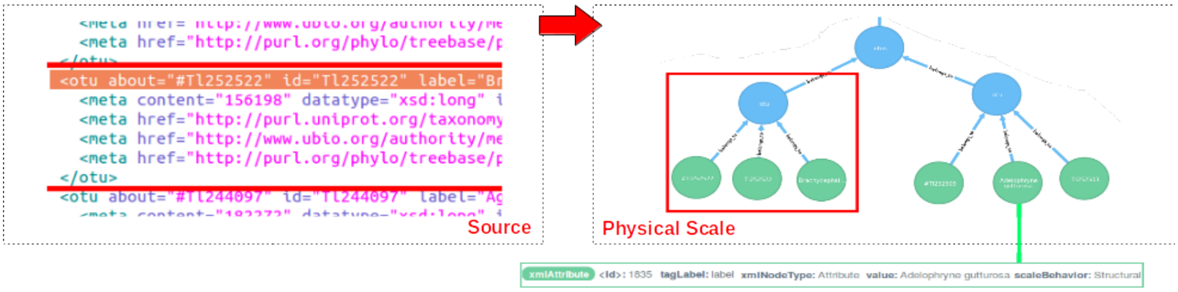


A	B	C	D
Species	Class	Family	Terrestrial Biomes
Bombina orientalis	Amphibia	Bombinatoridae	Temperate
Brachycephalus ephippium	Amphibia	Brachycephalidae	Tropical

ssCell <Id>: 2984 index: 2 cellType: STRING textColor: 000000 fontName: Verdana backgroundColor: FFFFFF sheetIndex: 0 commentAuthor: null lastCellIndex: 0 rowIndex: 2 columnIndex: 3 comment: null cellLabel: null fontSize: 20 content: Amphibia

Fig. 8. Graph Representation of an XLS file as a graph in the Physical Scale

Fig. 8 and Fig. 9 depict portions of the mapped graphs produced from the spreadsheet and XML/NEXUS presented in Fig. 6 and Fig. 7, respectively. The root node (green) in Fig. 8 represents a given XLS spreadsheet. It contains a single sheet, which has several rows (r_0 to r_2 in blue). Each row node points to its chain of cell nodes (gray). The box linked to the cell *Brachycephalidae* shows the variety of node properties, representing different aspects of the cell: location, content, format, *etc.* Similarly, the root node in Fig. 9 represents the XML resource itself, followed by an hierarchy representing the XML document. The highlighted red box represents the XML element *OTU*, previously presented in figure Fig. 7.



```
    <meta href="http://purl.org/phylo/treebase/p
  </otu>
  <otu about="#Tl252522" id="Tl252522" label="Br
    <meta content="156198" datatype="xsd:long" i
    <meta href="http://purl.uniprot.org/taxonomy
    <meta href="http://www.ubio.org/authority/me
    <meta href="http://purl.org/phylo/treebase/p
  </otu>
  <otu about="#Tl244097" id="Tl244097" label="Ag
```

xmlAttribute <Id>: 1835 tagLabel: label xmlNodeType: Attribute value: Adelophryne gutturosa scaleBehavior: Structural

Fig. 9. Graph Representation of an XML/NEXUS file as a graph in the Physical Scale

5.4 From the Physical to the Logical scale

Once the resources are represented as graphs in the *Physical Scale*, the integration process starts, and further scales are built on top of it as in a layered architecture. The subsequent *Logical* scale addresses the issue of handling a multitude of formats in a homogeneous logical structure. Transformations between the scales are based on criteria, which comprise a set of match/transform clauses, as detailed in Section 4.2. Fig. 10(A) to (B) illustrates the transformation from the Physical to the Logical scale.

While several formats organize their data as tables and relationships – *e.g.*, XLS, ODS, CSV and even an HTML table –, other organize the data as hierarchies – XML and JSON. Thus, it is possible to induce a common logical representation shared by several physical formats, which aligns or discards unmatched specificities.

Fig. 10 illustrates the XLS file in the *Physical Scale* and its corresponding representation in the *Logical Scale* as a table structure. While in the *Physical Scale* an XLS format is represented as a grid of cells, with specialized metadata concerning formulas, format *etc.* and no explicit schema – as usual in spreadsheets –, at the *Logical Scale* all *Tables* must look the same, *i.e.*, as illustrates Fig. 10, the first row of nodes connected to the *Table* node is an explicit Schema defined by its attributes.

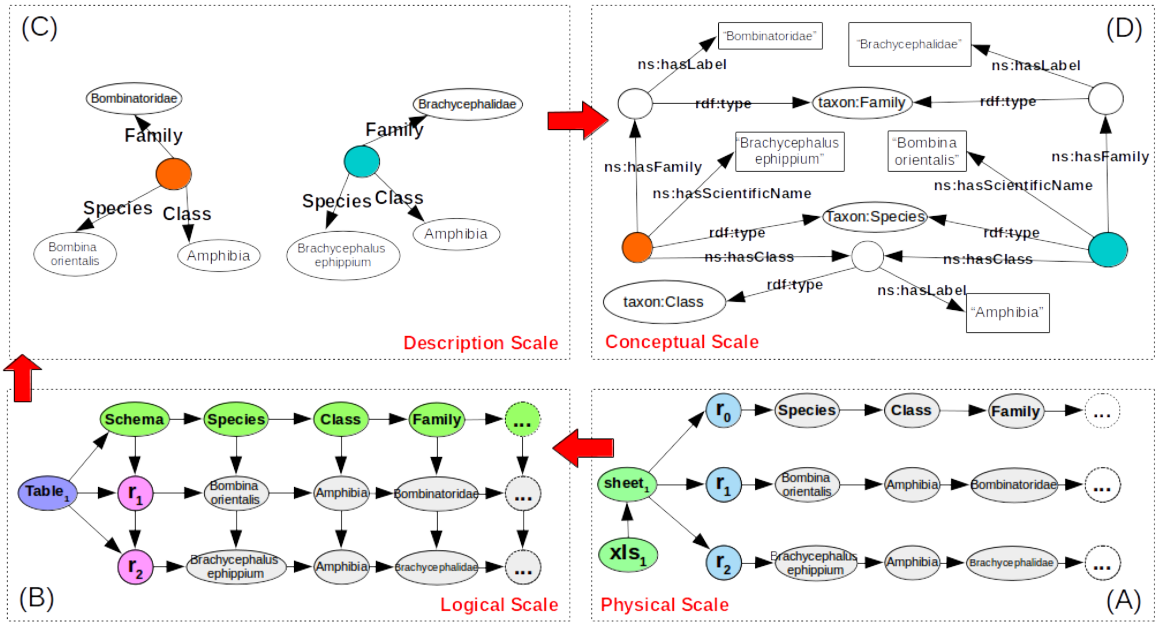


Fig. 10. All stages presented as a graph-based representation

The main benefit resulting from the effort of homogenizing multiple formats behind the same logical model is the possibility of reusing algorithms over the same logical structure, independently of its physical format – *e.g.*, the same algorithm can extract entities from tables coming from spreadsheets, CSV, relational tables and others. This transformation rise several challenges – e.g., schema recognition is not always trivial. Such challenges, however, are already widely discussed in the literature (including a previous work developed by us [Bernardo et al. 2013]) and are not subject of attention in this research.

5.5 From the Logical to the Description scale

The *Description Scale* aims at decoupling data from different logical structures and converges them to one single unified logical model. The unified model is based in the triple <resource, property, value>. It relies on RDF, but it still not a full fledged RDF, since it adopts only the RDF graph model to reduce all logical models to a single one. But the content of the nodes and edges are still plain text, lacking fundamental semantic concerns since it does not: distinguish entities, adopt controlled vocabularies to represent descriptive properties or make explicit the semantics of the elements using ontologies. These issues are addressed in the *Conceptual Scale*.

Initiatives found in literature stress different strategies for transforming a table or a hierarchy to triples, including a previous work developed by us [Bernardo et al. 2013]. This research do not focus on such problems and adopts a classical "triplification" strategy – as described in [Bernardo et al. 2013]. The transformation approach follows the same rationale of the previous section, to transform data represented as a *Table* in the *Logical Scale* to an RDF-based graph in the *Description Scale*.

The criterion applied in this transformation was described in Section 4.2 and illustrated in Fig. 4 (up), showing the match expression on the left and the transform process on the right. Fig. 4 (down) shows a materialization of the match/transform: each table row (r_1 and r_2) becomes a described instance, in which the descriptive attributes (a_1, a_2 and a_3) come from the *schema* row and their values (c_a, c_b and c_c) come from the rows content. Fig. 10(B) to (C) illustrates the transformation applied to our frogs example.

Although biologists still cannot handle data from previous scales in a conceptual and more integrated fashion, the *Description Scale* can be helpful to them, as it already allows some preliminary and meaningful analysis. For instance, spreadsheets regarding morphological traits usually adopts a cross-sheet way of organization. Such organization hampers an unified view of the traits of an organism, requiring more efforts from the biologists when conducting any initial analysis.

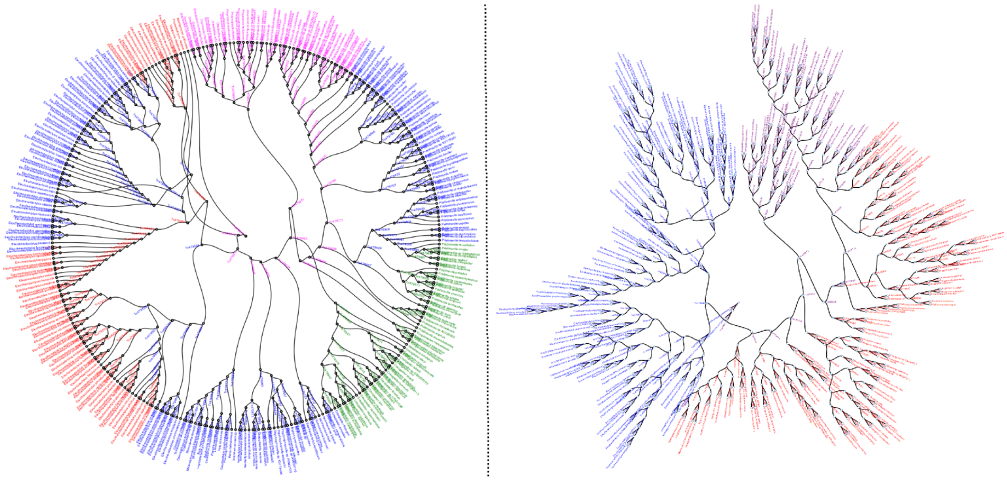


Fig. 11. Example of visualization of the Description Scale

At this stage of the investigation, *LinkedScales* enables to integrate XML files containing phylogenetic trees (from the *TreeBase* repository) with spreadsheets and CSV files regarding morphological traits (maintained by biologists). Based on the homogeneous models produced for the files in the *Logical Scales* (after being represented as a raw-format in the *Physical Scale*), species names mentioned on the tree and species names mentioned on the tables are linked using a simple string match.

Fig. 11 illustrates a visualization of output results corresponding to the initial outcome from the *Description Scale*. It shows the species following the phylogenetic tree provided by the XML file aggregated (colors) according to the tables in which the species are mentioned. Such tree enables the study of the evolution of traits across the phylogenetic group considered, but also correlates how closely related taxa are from one-another.

5.6 From the Description to the Conceptual scale

The *Conceptual Scale* achieves a full fledged RDF representation. The transformation from the Description to the Conceptual Scale involves applying algorithms like entity resolution and interconnection with ontologies to make explicit the semantics of the entities and properties involved in the description. Therefore, as illustrates Fig. 10(C) to (D), attributes are unified in the same RDF properties (*e.g.*, taxon:Species, taxon:Family); entities, like the class *Amphibia* and the family *Brachycephalus*, are unified.

6. CONCLUSION

In this article, we proposed an original framework named *LinkedScales*, based on the multiscale integration approach. Its architecture relies on graphs and systematizes in layers (scales) progressive integration steps based in graph transformations. *LinkedScales* is strongly related with the pay-as-you-go integration, slicing and encapsulating tasks concerned with the integration process in discrete scales. The approach is thus aligned with the modern perspective of treating several heterogeneous data sources as parts of the same dataspace, addressing integration issues in progressive steps, triggered on demand.

The designed solution is based on our Multiscale Graph Model, which was instantiated in our Primary Data Architecture able to be extended to several contexts. The proposal allowed a homogeneous perspective of data in each scale, encapsulating details about heterogeneities. In a nutshell, our approach is founded in three pillars: systematization, reuse and provenance.

The investigated experimental scenario demonstrated the overall potential benefits of *LinkedScales* to reach organism profiles. A significant part of the biological research work remains in an organism-centric perspective, which usually requires combining data regarding distinct aspects of organisms. However, relevant data is typically scattered among heterogeneous sources with different formats, structures and schemas, hampering the combination of data across sources to perceive information meaningfully and to systematically compare organisms. The solution proposed in the *LinkedScales* approach revealed its usefullness to the experimented analysis.

Future work involves conducting additional experimental evaluations to thoroughly examine the quality and scalability of data integration provided by the approach. Furthermore, a full-stack implementation integrating all the independent solutions in an unified system[5] will be developed.

Acknowledgments

Work partially financed[6] by FAPESP (#2014/14890-0), CNPq (#141353/2015-5), Microsoft Research FAPESP Virtual Institute (NavScales project), FAPESP/Cepid in Computational Engineering and Sciences (#2013/08293-7), CNPq (MuZOO Project), INCT in Web Science, FAPESP-PRONEX (eScience project), and individual grants from CAPES.

REFERENCES

Barceló, P., Libkin, L., and Reutter, J. L. Querying Regular Graph Patterns. *Journal of the ACM* 61 (1): 1–54, jan, 2014.

Barceló Baeza, P. Querying graph databases. In *Proceedings of the 32nd symposium on Principles of database systems - PODS '13*. Vol. 1777. ACM Press, New York, New York, USA, pp. 175, 2013.

Belhajjame, K., Paton, N. W., Embury, S. M., Fernandes, A. A., and Hedeler, C. Incrementally improving dataspaces based on user feedback. *Information Systems* 38 (5): 656 – 687, 2013.

Bernardo, I. R., Mota, M. S., and Santanchè, A. Extracting and semantically integrating implicit schemas from multiple spreadsheets of biology based on the recognition of their nature. *Journal of Info. and Data Manag.* 4 (2): 104, 2013.

Cai, Y., Dong, X. L., Halevy, A., Liu, J. M., and Madhavan, J. Personal information management with semex. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*. SIGMOD '05. ACM, New York, NY, USA, pp. 921–923, 2005.

Cruz, I. F., Mendelzon, A. O., and Wood, P. T. A graphical query language supporting recursion. *SIGMOD Rec.* 16 (3): 323–330, Dec., 1987.

Dijk, J., Choenni, S., Leertouwer, E., Spruit, M., and Brinkkemper, S. A data space system for the criminal justice chain. In *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*, R. Meersman, H. Panetto,

[5] For progress, refer to: http://linkedscales.lis.ic.unicamp.br

[6] The opinions expressed in this work do not necessarily reflect those of the funding agencies.

T. Dillon, J. Eder, Z. Bellahsene, N. Ritter, P. Leenheer, and D. Dou (Eds.). Lecture Notes in Computer Science, vol. 8185. Springer Berlin Heidelberg, pp. 755–763, 2013.

Dittrich, J., Salles, M. A. V., and Blunschi, L. imemex: From search to information integration and back. *IEEE Data Eng. Bull.* 32 (2): 28–35, 2009.

Dong, X. and Halevy, A. Indexing dataspaces. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. SIGMOD '07. ACM, New York, NY, USA, pp. 43–54, 2007.

Elsayed, I. and Brezany, P. Towards large-scale scientific dataspaces for e-science applications. In *Database Systems for Advanced Applications*. Springer, pp. 69–80, 2010.

Elsayed, I., Brezany, P., and Tjoa, A. Towards realization of dataspaces. In *Database and Expert Systems Applications, 2006. DEXA '06. 17th International Workshop on.* pp. 266–272, 2006.

Franklin, M., Halevy, A., and Maier, D. From databases to dataspaces: a new abstraction for information management. *ACM Sigmod Record* 34 (4), 2005.

Haas, L., Lin, E. T., and Roth, M. A. Data integration through database federation. *IBM Systems Journal* 41 (4): 578–596, 2002.

Halevy, A., Franklin, M., and Maier, D. Principles of dataspace systems. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART*. PODS '06. ACM, New York, NY, USA, pp. 1–9, 2006.

Halevy, A., Rajaraman, A., and Ordille, J. Data integration: The teenage years. In *Proceedings of the 32Nd International Conference on Very Large Data Bases*. VLDB '06. VLDB Endowment, pp. 9–16, 2006.

Hedeler, C., Belhajjame, K., Fernandes, A., Embury, S., and Paton, N. Dimensions of dataspaces. In *Dataspace: The Final Frontier*, A. Sexton (Ed.). Lecture Notes in Computer Science, vol. 5588. Springer Berlin Heidelberg, pp. 55–66, 2009.

Hedeler, C., Belhajjame, K., Paton, N., Campi, A., Fernandes, A., and Embury, S. Chapter 7: Dataspaces. In *Search Computing*, S. Ceri and M. Brambilla (Eds.). Lecture Notes in Computer Science, vol. 5950. Springer Berlin Heidelberg, pp. 114–134, 2010.

Hedeler, C., Fernandes, A., Belhajjame, K., Mao, L., Guo, C., Paton, N., and Embury, S. A functional model for dataspace management systems. In *Advanced Query Processing*, B. Catania and L. C. Jain (Eds.). Intelligent Systems Reference Library, vol. 36. Springer Berlin Heidelberg, pp. 305–341, 2013.

Hedges, S. B. The origin and evolution of model organisms. *Nature Reviews Genetics* 3 (11): 838ÂŊÂŊ849, 2002.

Hey, T., Tansley, S., and Tolle, K., editors. *The Fourth Paradigm: Data-Intensive Scientific Discovery.* Microsoft Research, Redmond, Washington, 2009.

Kolaitis, P. G. Schema mappings, data exchange, and metadata management. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. PODS '05. ACM, New York, NY, USA, pp. 61–75, 2005.

Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., and Zhao, J. Prov-o: The prov ontology. *W3C Recommendation* vol. 30, 2013.

Lenzerini, M. Data integration: A theoretical perspective. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS '02. ACM, New York, NY, USA, pp. 233–246, 2002.

Madhavan, J., Cohen, S., Dong, X. L., Halevy, A. Y., Jeffery, S. R., Ko, D., and Yu, C. Web-scale data integration: You can afford to pay as you go. In *CIDR*. www.cidrdb.org, pp. 342–350, 2007.

Miranda, E. and Santanchè, A. Unifying phenotypes to support semantic descriptions. *Brazilian Conference on Ontological Research – ONTOBRAS*, October, 2013.

Mota, M. and Medeiros, C. Introducing shadows: Flexible document representation and annotation on the web. In *Proc. of Data Engineering Workshops (ICDEW), IEEE 29th ICDE*. pp. 13–18, 2013.

Mota, M. S. and Santanchè, A. A. Conceiving a multiscale dataspace for data analysis. In *Proceedings of the Brazilian Seminar on Ontologies (ONTOBRAS 2015)* (2015-09-08), B. C. on Ontologies (Ontobras) (Ed.). Vol. 1442. CEUR, pp. 12, 2015.

Paton, N. W., Christodoulou, K., Fernandes, A. A. A., Parsia, B., and Hedeler, C. Pay-as-you-go data integration for linked data: opportunities, challenges and architectures. In *Proceedings of the 4th International Workshop on Semantic Web Information Management*. SWIM '12. ACM, New York, NY, USA, pp. 3:1–3:8, 2012.

Rahm, E. and Bernstein, P. A. A survey of approaches to automatic schema matching. *The VLDB Journal* 10 (4): 334–350, 2001.

Singh, M. and Jain, S. A survey on dataspace. In *Advances in Network Security and Applications*, D. Wyld, M. Wozniak, N. Chaki, N. Meghanathan, and D. Nagamalai (Eds.). Communications in Computer and Information Science, vol. 196. Springer Berlin Heidelberg, pp. 608–621, 2011.

Washington, N. L., Haendel, M. A., Mungall, C. J., Ashburner, M., Westerfield, M., and Lewis, S. E. Linking human diseases to animal models using ontologyÂŊbased phenotype annotation. *PLoS biology* 7 (11): e1000247, 2009.

Wood, P. T. Query languages for graph databases. *ACM SIGMOD Record* 41 (1): 50, apr, 2012.