FishGraph: A Network-Driven Data Analysis

Patrícia Cavoto*, Victor Cardoso*, Régine Vignes Lebbe§, André Santanchè*
*UNICAMP - Universidade Estadual de Campinas, São Paulo, Brasil
{patricia.cavoto, victor.roth.cardoso}@gmail.com
santanche@ic.unicamp.br
§ ISYEB - UMR 7205 – CNRS, MNHN, UPMC, EPHE
UPMC Univ. Paris 06, Sorbonne Universités, Paris, France
regine.vignes_lebbe@upmc.fr

Abstract—**There are a lot of data about biodiversity stored in different database models and most of them are relational. Recent research shows the importance of links and network analysis to discover knowledge in existing data. However, the relational model was not designed to address problems in which the links between data have the same importance as the data – a common scenario in the biodiversity area. Moreover, the Linked Data and Semantic Web efforts empowered the fast growth of open knowledge repositories on the web, mainly in the RDF (Resource Description Framework) graph model. The flexible graph database model contrasts with the rigid relational model and is also suitable for data analysis focusing on links and the network topology, e.g., a connected component analysis. Our research is inspired by the data OLAP (OnLine Analytical Processing) approach of creating a special database designed for data analysis; a network-driven data analysis using graph databases, in our case. Beyond an initial ETL (Extract, Transform and Load) approach, we are facing the challenge of migrating the data from the relational to the graph database, managing a dynamic coexistence and evolution of both, not supported by related work. This work is motivated by a joint research involving network-driven data analysis over the FishBase global information system. We present a novel approach to analyzing the connections among thousands of identification keys and species and to linking local data to third party knowledge bases on the web.**

Keywords-component; graph database; network topology analysis; biodiversity information systems.

I. Introduction and Motivation

Links among data elements are getting increasing attention. Besides the ever-growing Linked Data, which fosters the creation of a Giant Global Graph [1], links by themselves form networks and are a rich source of latent semantics, which can be discovered through the analysis of the network topology. Currently, there are various initiatives – such as GeneOntology, GeoNames, DBpedia and Bio2RDF – that share their information as graphs on the web, mainly in the RDF (Resource Description Framework) format.

Even with the growth in the use of the graph model for databases, we have big databases and systems that continue to use the relational model for data storage and retrieval. This model is appropriate to execute data maintenance transactions (insert, update and delete) and has been refined in the last decades to handle big volumes of data. However, in the analysis of data focusing on the network produced by links – in which relations are as important as the data – it is usually necessary to create complex and/or inefficient SQL queries to answer the problem, given the rigid data structure and restrictions that this model has in regard to analysis of transitive relationships [2], e.g., a query for identifying the environmental impact suffered from the extinction of a species. Graph databases are very effective in this type of analysis.

This research is inserted in the context of biodiversity information systems, more specifically in a collaborative research involving FishBase (http://www.fishbase.org), a database and information system for biological data storage of fish species [3].

In this paper, we present how we explored links for two sets of data from FishBase: identification keys, a biology mechanism to identify a specimen, and species classification, validating local data with DBpedia. Beyond straight links among species and identification keys – such as those provided by foreign keys in relational databases – the correlations form an interdependent network, whose topology analysis allows identifying the most relevant species in an area or finding inconsistencies in the keys. Due to the relational nature of data in FishBase, these analyses can be complex and/or inefficient to perform since the network is not explicit and because it requires a high-use of the JOIN statement. Furthermore, sometimes it is necessary to dynamically produce data – adding new fields and relations – to a specific analysis and, in this case, the relational model is not as flexible as the graph model.

In this research, we propose the coexistence of both models, relational and graph interacting with each other, dividing tasks of management and analysis according to their specialties. However, in some scenarios, transferring the entire database to a new model is a complex task, because it would require rewriting all of the systems accessing this database.

Therefore, the main goal of our research involves proposing a hybrid architecture that enables the integration of one or more relational databases with a graph database, minimizing the impact in the existing relational infrastructure, and enabling it to exploit network-driven analyses in the graph database. The current version, called ReGraph, works with an initial ETL data migration and operates over a dynamic synchronization, in which all the updates done in the relational database are reflected in the graph database. Moreover, we provide autonomy to data produced natively in the graph database, either for data analysis or to be able to link this information with third party knowledge bases on the web, enriching and validating the information in the graph database. This paper focuses on showing how our approach addresses biology information system analyses in the FishBase scenario. We produced an experiment migrating relevant data about identification keys and species from FishBase to a graph database. The analyses of the network produced by these links in the graph database is helpful to analyze relations among thousands of identification keys, to compare and check the inconsistencies between these keys and to analyze links with other sources on the web, allowing the discovery of new information and validating the existing data.

The remainder of the paper is organized as follows. Section 2 details the research scenario. Section 3 presents the theoretical foundations and related work. Section 4 presents our hybrid architecture and describes the migration process to integrate the FishBase data in FishGraph. Section 5 presents our experiments and their results. Section 6 presents our conclusions.

II. Research Scenario

A. The FishBase Database

Data in the FishBase system are stored in a relational database model. It currently has 33,000 registered species [4] whose data are distributed in 130 tables encompassing several aspects related to the study of fishes – e.g., identification keys, taxonomic classification, ecosystems, etc. – totaling over 2 million records [3]. All this information raises challenges to scientists who have difficulties analyzing some kinds of scenarios involving the examination of the network and links among elements. In this subsection, we will detail two problems we are facing that illustrate the typical scenario in which our approach will be beneficial. They also are the basis for practical experiments to test and validate our proposal:

1) Identification Keys: An identification key is a set of questions that guides scientists in the identification of a specific specimen. Each identification key in FishBase was produced independently. There are 1,668 identification keys for fishes and more than 24,000 questions related to these keys in FishBase. Since they are produced independently, there is a lot of overlapping information and distinct keys describing common species. Some keys have geographic or hydrographic boundaries, some keys have taxonomic limits, and other ones focus on a specific habitat or a development stage. Moreover, one must first select a specific key to begin the identification procedure, i.e., from an observation of an unknown fish, it is hard to decide which key to choose. In order to address these problems, we are exploiting the links among keys/species/locations to answer the following questions:

- Looking at the network formed by interconnections among keys and species – keys that share species and species that share keys – is it possible to find similarities among them?

- Is it possible to create groups (clusters) of these identification keys using the geographic location of the linked species? This will be helpful to identify inconsistencies between the analyzed keys and to offer a better access to choose a key among the 1,668 keys of FishBase.

2) Species Classification: FishBase has data on almost 33,000 species and their respective taxonomic classification, including genus, family, order and class. Cross-referencing these data with third party knowledge bases could be helpful in improving the accuracy of the data in FishBase.

The starting point to answer these questions is the FishBase relational model partially shown in Fig. 1.

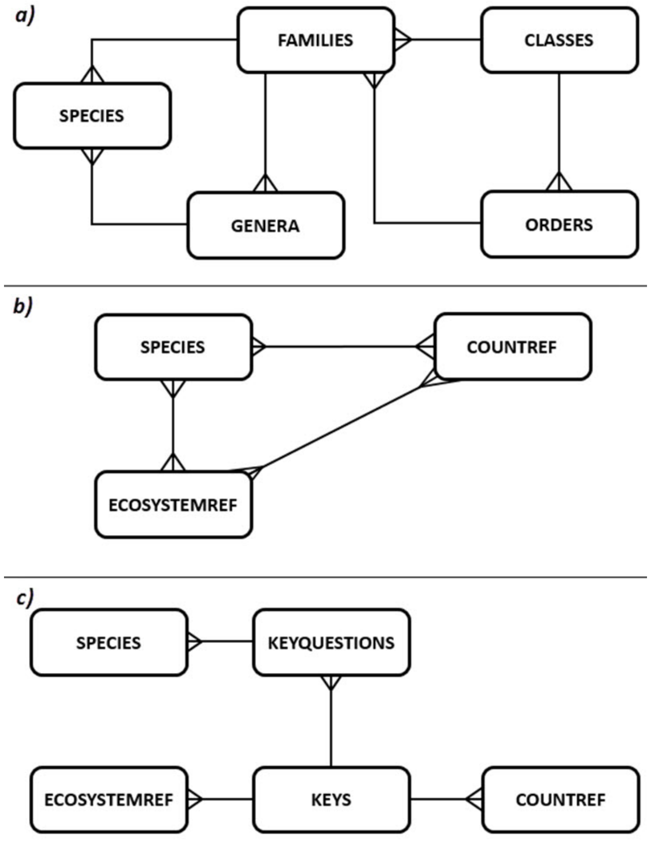

Figure 1. FishBase tables: a) species and their taxonomic classification; b) species, countries and ecosystems; and c) identification keys.

The main table used to address all the proposed analyses is the SPECIES table, in which we have all information about the fish species and their taxonomic classification (Fig. 1a). The next set of tables (Fig. 1b) links the SPECIES with countries (COUNTREF) and ecosystems (ECOSYSTEMREF). There are 311 countries and 1,019 ecosystems in the FishBase database. The last set of tables (Fig. 1c) has the link from SPECIES and location tables to identification keys (KEYS table). The KEYQUESTIONS table contains the questions used in the identification process to determine species identity. The KEYS table has also links to the GENERA, FAMILIES and ORDERS tables, used when it is not possible to determine a specific species inside a bigger taxonomic group.

We have migrated all this information from FishBase to a graph database.

B. *Why Graphs?*

One main argument of this work is that the graph model for databases has advantages when a network-driven analysis is involved. We will illustrate this argument here, addressing an identification key analysis scenario and confronting the following possibilities: implementing a solution in the current database model (relational); modeling the data in the XML format; and modeling the data as a graph. The analysis involves discovering similarities and inconsistencies among groups of identification keys, based on the species that they share.

Considering the first possibility, a relational database is not designed to query networks of links – JOINs in relational terms – since it requires transforming the query in the pairing of relations and multiplying the analysis of n-ary links in several combinations of binary links. Network analysis usually also requires traverse paths, involving – in relational terms – to transitively fetching nodes by successive JOINs. Furthermore, the table-like approach of the SQL language is not intuitive for this kind of problem.

XML is usually referred as an option for some network representations since it can connect pieces of data in a semi-structured way, forming a hierarchical network. However, modeling the data in the XML format will require a hierarchical representation of non-hierarchical data. It can be arranged when the network is simple, e.g., in our scenario, defining the identification key as the parent node and the species as the child nodes (see Fig. 2).

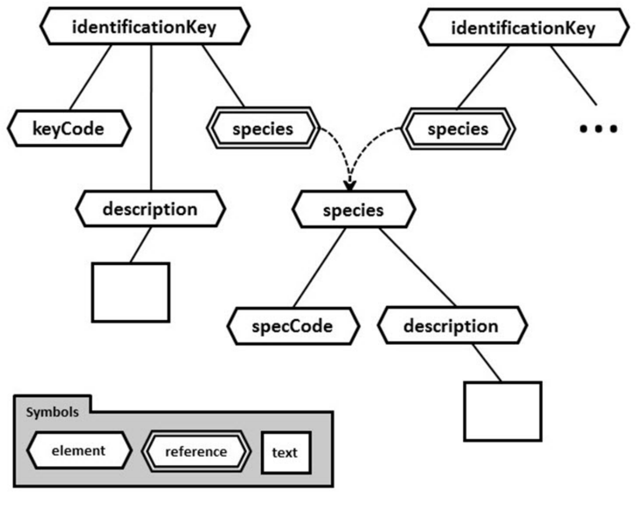

Figure 2. XML structure for the identification key problem.

XQuery (a query language for XML) can be applied to fetch the data hierarchically. Besides addressing only simple networks, these cases require modeling the data to fit a specific query and – as in the previous case – the hierarchical-like approach of XML/XQuery is not intuitive. Analyzing this scenario in reverse, in other words, finding a group of species and the identification keys that they share, would be harder or require a specific new inverted XML schema.

Both models, relational and XML, have rigid data structure and were not designed to address network-driven analysis, in which the connections have the same (or more) importance than the connected entities.

Fig. 3 presents our graph model to address the identification key problem, in which we have the nodes KEY and SPECIES and, below them, their respective properties.

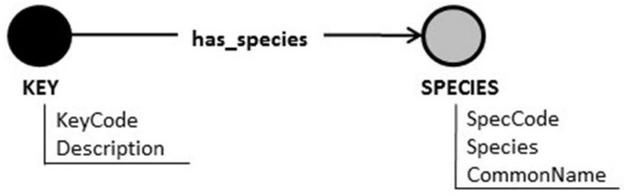

Figure 3. Graph model for the identification key problem.

This model allows a streamlined way to perform the analysis, given its flexibility. Moreover, the proposed scenario, in which we need to find the neighbors of a node, is a network-driven question typically analyzed in a graph. Even inverting the question, it is still fitting to analyze the network. In addition, the graph-like approach is more intuitive for network-like problems. Therefore, we chose the graph model to perform network-driven analyses.

III. THEORETICAL FOUNDATIONS AND RELATED WORK

A. *Graph Databases*

From the classic definition adopted by mathematics and computing, a graph is an ordered triple consisting of a non-empty set of vertices (also called nodes), a set of edges (links) and an incident function that associates each edge to an unordered pair of vertices [5]. Graphs are excellent to model complex connections, objects and their interactions – a very common scenario in scientific research. In the biology field, there are many uses for graphs, including metabolic networks, chemical structures and genetic maps [6].

Have and Jensen [7] present experiments in which the use of the Neo4J graph database in bioinformatics brought better overall performance results than using the PostgreSQL relational database. Their analyses were based on the execution of queries typically performed in bioinformatics: finding the neighboring vertices of a protein and its interactions; finding the best path score between two proteins; and finding the shortest path between two proteins.

Given the emphasis on explicit representation of large volumes of relations as edges, graph databases have an optimized infrastructure to manage transactions involving a large number of associations, such as, traversing a path, performing a breadth-first or depth-first search. The same transactions in relational databases would require, in some cases, many consecutive JOINs, which makes the query very costly. On the other hand, data consistency is one of the bases of relational databases and is still poor in graph databases [8].

Our proposal is to produce a hybrid information system combining the advantages of a relational database – to control data consistency and transaction operations – with a graph database for network-driven data analyses. Our application scenario is the FishBase information system.

Data stored in a graph database are typically represented by a directed graph model [9]. Several approaches have emerged to structure data inside this kind of database, according to the application scenario, e.g., simple graphs, hypergraphs and property graphs. A simple graph is based on nodes and edges; a hypergraph extends this concept, allowing the same edge to link any number of nodes; and the property graph allows it to create descriptive properties attached to nodes and edges [8], as presented in Fig. 3.

Based on the comparison of different graph database models [8] and the performance analyses of four graph databases [10], we have decided to use the Neo4J as our graph database management system. Neo4J implements a property graph data structure. The number of nodes and edges in this model tends to be lower when compared to the RDF graph model, which represents all properties/values as extra nodes/edges in the graph. Neo4J uses the Cypher declarative query language and provides graph algorithms packages, e.g., the Dijkstra's algorithm to determine the shortest path between two nodes and the traversal framework.

B. Database Models Integration

Raghavan and Garcia-Molina [11] classify architectures that integrate relational databases with other models. Influenced by this work, in this subsection, we define a classification of the related work focusing on the integration of relational data and graphs in three categories:

1) Graphs in the Relational Database Model

This architecture departs from the relational database model and extends it to support other data structures. There are two main groups in this architecture:

The first one creates new graph structures mapped on top of native relational implementations, extending the SQL language and strategies for optimizing the execution of queries when graphs are involved. Goldberg and Jirak [12] propose an enhanced relational model which allows for the storage, retrieval, and manipulation of directed graphs, by implementing new data types that can be included in the existing tables, avoiding to redesign of their schema or the migration of data. The Database Graph Views [13] proposes an abstraction layer as a mechanism for creating views to manipulate graphs, independent of the physical arrangement where the original data are stored. The Virtuoso RDF View [14] proposes mapping relational data to RDF. This group of approaches cannot fully benefit from the advantages of graph databases has in regard to network operations and design flexibility, since the queries still run over a relational database.

The second group implements a specialized data structure for graph models, besides relational models, expanding the existing relational database systems. In this group, there are leading manufacturers of commercial database management systems, such as Oracle and IBM. The Oracle Spatial Graph includes a rich set of features to address spatial and analytical data, which can be applied to social and semantic graphs. These features can be connected to the relational model, allowing it to use combined schemas with functions that support spatial data and graphs [15]. The IBM DB2 RDF stores and retrieves data in an RDF graph format, expanding its relational database [16]. This group of approaches has performance advantages compared to the previous one, however, it is still product dependent, i.e., solutions are tailored for specific database brands, requiring the adoption of the "whole package". It can involve complex migrations of data to adapt all the systems that use another database.

2) Integration Module Architecture

This architecture includes two or more types of databases kept in their native form. An independent module, that integrates all the databases, provides a unified access interface, using a third data structure as a support for queries and to combine the results. Each database source has a wrapper, with rules mapping and translating the native language to a unified structure. The Garlic approach [17] defines an object-oriented extension of the SQL as a unified language. The D2R Map [18] proposes the integration of ontologies in RDF with relational databases through a module that creates a virtual RDF graph.

This architecture brings many advantages to the user, who can interact with various data sources through a single interface and language. Changes in the data structure and schema of each database source are minimal or unnecessary. However, this architecture is very expensive to implement, given the complexity of the involved tasks, which include: development of a unified language among the models; use of wrappers responsible for performing the necessary translations between the models; and the adaptation of the systems to use the new language.

3) Layered Architecture

This architecture implements a database model (upper layer) that operates on top of another model (lower layer). The model in the upper layer is dependent on the one in the lower layer. The data access of both models is performed exclusively through the upper layer. The layered architecture is applied in Information System Retrieval (IRS). The Vodar architecture [19] integrates an IRS with an object-oriented database; De Vries and Wilschut [20] adopt the MonetDB column-oriented database. The challenge of a layered architecture is to map operators from the upper model to the lower one. We have not found works with the layered architecture integrating a relational model with a graph one.

4) Summary

Inspired by the framework created by the W3C Incubator Group, RDB2RDF [21], in Table I we classify the architectures and papers presented in this section, considering the following criteria:

a) Mapping type (Manual, Automatic, Both);

b) Language of data access;

c) Materialization of the integrated model in the database (Y-yes, N-no);
d) Required changes in the existing systems, schemas and data (L-low, M-medium, H-high);
e) Implementation effort (L-low, M-medium, H-high);
f) Coexistence of native models (Y-yes, N-no); and
g) Ability to integrate more than one database (Y-yes, N-no).

TABLE I. RELATED WORK COMPARISON

Paper	a	b	c	d	e	f	g
Relational: [12], [13]	Manual	SQL	N	L	L	N	N
Relational: [14]	Both	SQL / SPARQL	N	L	M	Y	Y
Relational: [15], [16]	Auto	SQL SPARQL	Y	H	M	Y	N
Integration Module: [17]	Manual	Own	Y	L	H	Y	Y
Integration Module: [18]	Both	SQL SPARQL	N	L	M	Y	N
Layered: [19], [20]	Manual	Superior Layer	Y	H	H	Y	Y

IV. A HYBRID ARCHITECTURE FOR THE FISHGRAPH

This section describes ReGraph, our proposed hybrid architecture, and how it was applied in the FishBase information system, generating the FishGraph model.

A. *Hybrid Architecture: Relational and Graph*

Fig. 4 presents a diagram of our hybrid architecture. It maintains the relational and graph databases in their native forms and connects them. The central part of the diagram is the graph database, which can map and connect one or more relational databases.

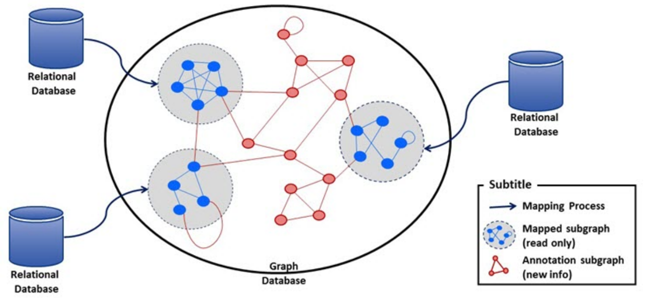

Figure 4. Proposed hybrid architecture.

The proposed architecture has the following features:

1) Mapping process: It is inspired by the OLAP approach of creating a special database designed for data analysis which replicates part of other existing databases. Data from the relational database are partially or completely mapped and replicated to the graph database. Data can be mapped automatically by the system or through a manual configuration, specifying how tables and fields should be migrated to nodes, properties or edges in the graph. The manual mapping also allows for the definition of rules for linking data from different sources;

2) Synchronization: Beyond a static ETL (Extract, Transform and Load) approach, the system manages a dynamic evolution of both databases. A service, scheduled in regular intervals by the user, will refresh the data. Changes in the relational database are captured by this service and updated in the graph database. Furthermore, we have the integrated coexistence of both models, keeping their native representation;

3) Mapped and Annotation subgraphs: To manage the consistency among databases, we devise two kinds of subgraphs in the graph database: Mapped and Annotation. The Mapped subgraph contains synchronized read-only data coming from the relational database, i.e., they cannot be modified in the graph database. The Annotation subgraph is produced directly in the graph database – without corresponding elements in the relational database. It enables it to produce new knowledge using the graph flexible structure. Besides enabling the enrichment of data with user annotations, without the requirement of changes in the relational schema and a specialist support, the annotation subgraph can also materialize intermediary results of a network-driven analysis process, as we will show in the practical experiments.

B. *The FishBase Entities Analysis*

FishBase does not have a complete documentation about the database model and schema. Therefore, in order to migrate the relevant data about identification keys and species taxonomic classification, we started analyzing the FishBase entities (tables). In this analysis, we checked the entities, their fields and records to determine the entities, the relationships and the database model to the relevant set of data.

This task resulted in the selection of the main entities, including the related data, and their relationships (previously explained in Section II): SPECIES, GENERA, FAMILIES, ORDERS and CLASSES, which help to determine the taxonomic classification of a species; and KEYS and KEYQUESTIONS, that have the information about the fishes' identification keys. With the objective of expanding our analysis, we also include entities related to the geographic location: COUNTRYREF and ECOSYSTEMREF, that allow connecting species and identification keys with countries and ecosystems. With the main entities defined, the next step was to select the relevant fields from each entity. To address the specific questions proposed, we decided to migrate only the main fields from each entity, such as identification fields and descriptions.

C. *Mapping Rules*

There are two possible mappings: manual and automatic. In the manual mapping, it is necessary specify each one of the tables and columns that will be migrated to the graph database and how they will be created (nodes, properties or edges). In the automatic mapping, all tables and columns will be migrated following the defined rules (detailed in the next subsection). We have tested two mapping features: the manual mapping process and the data synchronization process, using a Java program to perform both. The manual mapping works selecting the relevant data, detailed previously, and, later, running an ETL process that migrates all the selected data, which involves a big transformation load. In the data synchronization process, we created database triggers that will be fired after each insert, update or delete operation in a given table. When any of these events occur, a record is inserted in a notification table, in which we have the columns: primary key of the table that started the trigger; event (insert, update or delete); and a status representing whether this change was already applied to the graph. A Java program reads this notification table, filtering records with the status column defined as 'pendent', and performs the change in the graph database, maintaining the synchronism and updating the status column to 'done'.

D. From FishBase to FishGraph

Our mapping model has been designed taking the "FishBase → FishGraph" mapping problem as a starting point. Tim Berners-Lee [22] discussed a set of mapping rules from relational databases to RDF as follows: a record in the relational database becomes an RDF node; a column name becomes an RDF predicate (a labeled edge); and a relational database table cell becomes a value. The FishGraph will capture only the relevant data in the FishBase for the analysis. As we are working in a property graph, and the Tim Berners-Lee mapping considers an RDF graph, we have adapted the rules to our model as follows:

- an entity in the relational model becomes a node type in the graph, also called "class" in Neo4J;
- each record becomes a unique node in the graph;
- each table cell (except the foreign keys) of these records becomes a node property in the graph;
- each foreign key becomes an edge in the graph.

V. Experiments and Results

We applied the mapping rules described in the previous section to migrate the relevant data from FishBase to Neo4J, generating the FishGraph model presented in Fig. 5. The diagram represents a graph modeling of the selected tables from FishBase (presented in Fig. 1) and follows the same approach of Fig. 3 to describe property graph models. The KEYQUESTIONS table was used exclusively to link KEYS to the other nodes.

To validate the synchronism feature in our architecture, we have created a trigger to register changes in the SPECIES table. This trigger is fired after each insert, update or delete operation in this table and is responsible for maintaining the data synchronism.

A. Experiment 1: Identification Keys

To study the network formed by data from FishBase, we connected FishGraph with Gephi (http://gephi.github.io/), an interactive platform for analysis, visualization and exploration of networks.

Data for species, identification keys, ecosystems and countries generated 35,955 nodes in FishGraph. To do this analysis, we selected the species, ecosystems and countries that had links with identification keys. The resulting data had 10,403 nodes and 86,693 edges.

An identification key is a powerful mechanism to identify species. The right definition of an identification key is fundamental in achieving its goal. As in FishBase, most of the identification keys are produced independently, the analysis of the interconnection among them through species, ecosystems and countries can be helpful in finding inconsistencies. All the identification keys are available in the FishBase website (http://fishbase.org/keys/allkeys.php/).

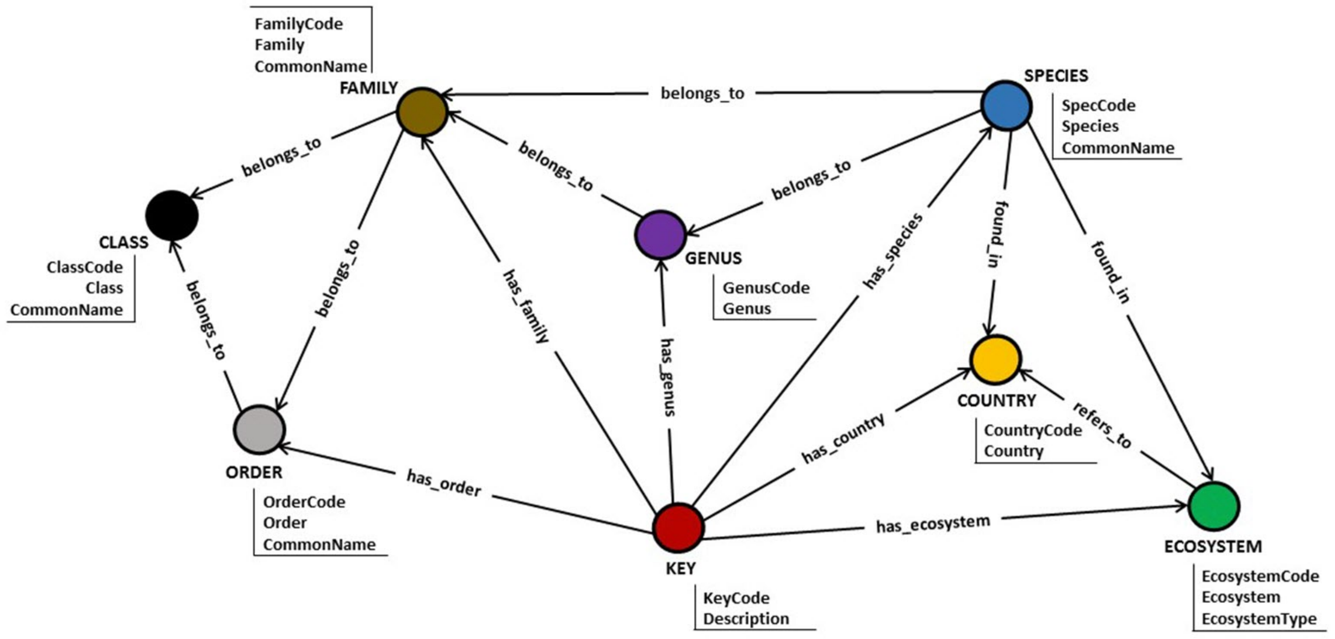

Figure 5. FishGraph model for identification keys and species classification.

1) Cliques Analysis

This analysis aggregates links among keys related to shared species. The resulting aggregation was materialized in the annotation subgraph, by the production of "Share" edges linking keys to keys, with the properties: "Species" indicating how many species each pair of keys share, and "SpeciesProportion" representing the percentage of shared species, when compared to the total of species in the respective key. Since the proportions change according to the node they refer to, the edges are directed and for each pair of keys *k1* and *k2* (see Fig. 6) there is an edge from *k1* to *k2* (with the proportion of *k1*) and an edge from *k2* to *k1* (with the proportion of *k2*). The annotation subgraph enables the creation of extra information without affecting the relational source, which would require a new table only to store the relations between the keys.

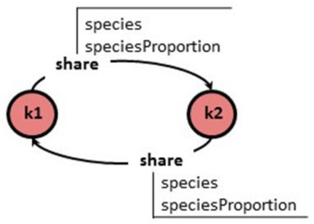

Figure 6. Annotation edges "share" between two keys and their properties.

The resulting network showed that we have a large number of keys that do not share species with other keys, on the border of the diagram, and some keys that are highly connected, on the center (see Fig. 7).

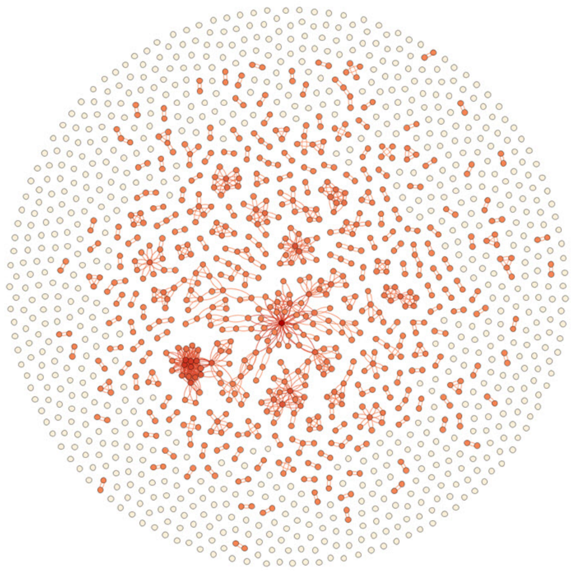

Figure 7. Distribution of the keys by species shared with another key.

Over this set of data, we applied the Clique Percolation Method (CPM) [23] for finding overlapping dense groups of nodes in networks. This method allows identifying all existing *k*-cliques in the network. A clique is a subset of vertices of a graph G, such that its induced subgraph is complete, i.e., every two distinct vertices in the clique are adjacent. A *k*-size clique indicates a clique that has *k* nodes and each node has *k*-1 edges connecting it to the other nodes. The two largest cliques in this network have size 10, indicating that there are 10 keys sharing at least one species (see Fig. 8). The clique analysis was the starting point for the next steps.

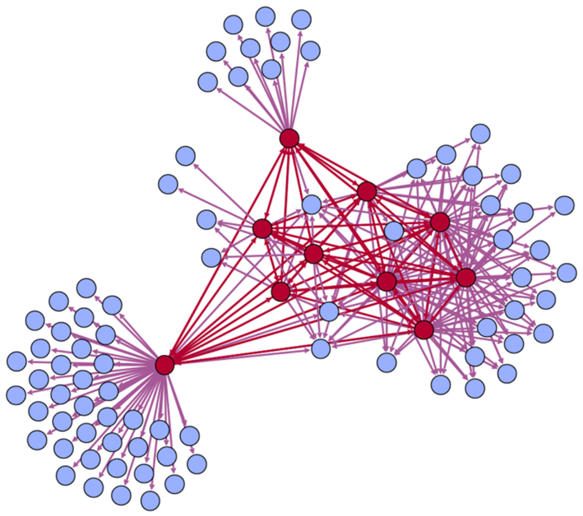

Figure 8. Clique of identification keys (red nodes) of size 10 and their respective species (blue nodes).

2) Connected Component Analysis

This analysis addresses the species and key nodes (1,365 keys and 8,112 species connected by 9,864 edges). In this network, we applied the Depth-First Search and Linear Graph Algorithms [24] to find connected components. A connected component in a graph G is a subgraph H of G in which, for each pair of nodes *u* and *v*, there is a path connecting *u* and *v*. Each connected component is independent [5].

In our analysis, a connected component represents keys and species that are connected. With this analysis, it is possible to find inconsistencies or redundancies in the keys definition. We identified 863 connected components (see Fig. 9), in which 31 have a unique key and a unique species.

This analysis can guide biologists in finding highly related and complementary keys. To illustrate that, we identified a component having two keys connected by only one species: *Erpetoichthys calabaricus*. The description of these two keys are quite similar:

1533 - Key to the genera of *Polypteridae* of West Africa.

1584 - Key to the genera of *Polypteridae* of Lower Guinea, West-Central Africa.

A complementary analysis in the FishBase database showed that each of the two keys have two questions each one describing different characters of the species. Unifying the two keys could give more information about the species and make the identification process more efficient.

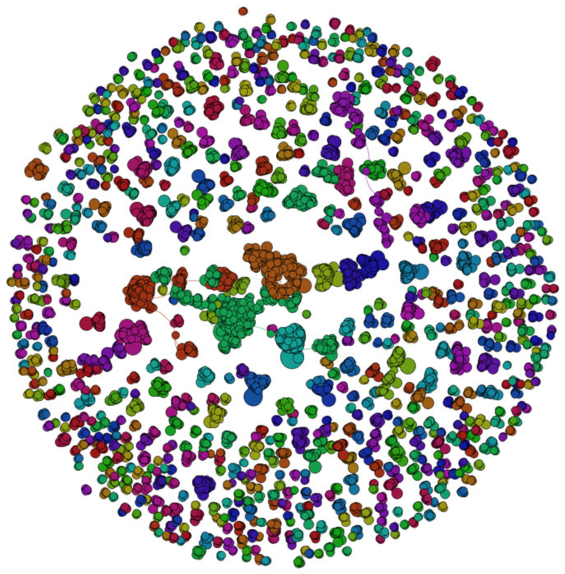

Figure 9. All of the 863 connected components (distinct colored).

We found components in which the keys share 100% of the species. To demonstrate that, we found a component with two keys and 18 shared species. A further analysis in the description and the questions of the two keys showed that they are versions of the same key:

475 - Key to species of *Hemipsilichthys*.

360 - Key to the species of *Hemipsilichthys*.

In this case, we could suggest deleting one key and keeping the more updated.

3) *Max Out-Degree Analysis*

In this analysis, we obtained the out-degree of each key, i.e., the number of links from keys to species. Confronting these data with the clique and component analyses, we found that the key with the max out-degree is part of a component with 149 edges and 140 nodes, in which there are 2 keys and 138 species. The two keys share only 11 species (see Fig. 10). The description of both keys is related to the same species:

205 - Key to the species of *scorpionfishes* (also, *lionfishes*, *rockfishes*, *stingfishes*, *stonefishes*, and *waspfishes*) (*Scorpaenidae*) occurring in the Western Central Pacific.

316 - Key to the species of Indo-Pacific *Scorpionfish* (Genus *Scorpaenopsis*).

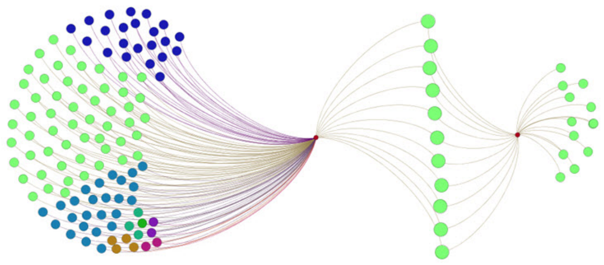

Figure 10. Max Out-Degree key component with species colored by family.

The questions of both keys are similar. All species in this component are part of the same taxonomic order and class, and there are 8 distinct families and 51 distinct genera. This analysis also suggests that it could be interesting to unify the keys.

4) *Ecosystems and Countries Analysis*

This analysis involves data from keys, ecosystems and countries. According to the FishBase data, FishGraph has keys directly connected to ecosystems and countries and keys indirectly connected to them through species referred by that key. Several keys do not have direct links to ecosystems and countries; therefore, we obtained this information indirectly through the respective connected species, producing new edges connecting them to the key, which are materialized in the annotation subgraph. The result was a dense and highly connected network.

Using the existing graph and the produced annotation subgraph, we applied a simple degree analysis of ecosystems and countries and obtained the top ten nodes of each one (see Table II and Table III).

TABLE II. Top Ten Ecosystems

	Ecosystem	**Total Keys**
1	South China Sea	507
2	East China Sea	459
3	North Australian Shelf	375
4	Agulhas Current	365
5	Peng-hu Island	345
6	Sulu-Celebes Sea	327
7	Caribbean Sea	304
8	Coral Sea and GBR	294
9	Red Sea	279
10	Insular Pacific-Hawaiian	268

TABLE III. Top Ten Countries

	Ecosystem	**Total Keys**
1	China	688
2	Australia	614
3	Japan	596
4	Indonesia	557
5	Taiwan	556
6	Philippines	533
7	South Africa	506
8	Vietnam	505
9	India	482
10	Papua New Guinea	469

The ecosystems 7 and 10 represent ecosystems in the American continent, but the analysis of countries does not show any country from this continent. We conclude here that there is a lack of relations between keys and countries in the FishBase database.

B. *Experiment 2: Species Classification*

The Semantic Web initiative – in which humans and computers can interpret web resources – has been increasingly supported by the scientific community. Linked Open Data (LOD) represents an important part of these initiatives, which provide large and growing collections of datasets, represented in open standard formats (including RDF and URIs). They are partially linked to each other and can be connected to domain knowledge, represented by the Semantic Web ontologies [25].

LOD provides an excellent environment for knowledge discovery.

Using the annotation subgraph, we have created a new property, named URI, for the species nodes. The URI (Uniform Resource Identifier) is a global unique identifier for a resource on the web, used by the Semantic Web to produce a distributed knowledge network, where any node can be linked to any other node on the Web, pointing to its URI. This URI property has the goal of preparing the FishGraph for the LOD and to linking its data to third party knowledge bases. We have adopted the DBpedia (http://www.dbpedia.org/) ontology to define the URIs of the properties for species, genus, family, order and class.

In this experiment, we compared data about species and their taxonomic classification, available in FishGraph, with the data of species available on DBpedia. The main goal is to compare data available in both sources and to present the results.

Some species in DBpedia have their taxonomic classification defined by properties connected to the DBpedia ontology (URI prefix http://dbpedia.org/ontology/) and other as literal text fields (URI prefix http://dbpedia.org/property/). In our analysis, we compared the FishGraph data with both values. The ontology/literal properties used in this analysis were:

- Genus: http://dbpedia.org/ontology/genus and http://dbpedia.org/property/genus;
- Family: http://dbpedia.org/ontology/family and http://dbpedia.org/property/family;
- Order: http://dbpedia.org/ontology/order and http://dbpedia.org/property/order;
- Class: http://dbpedia.org/ontology/class and http://dbpedia.org/property/class.

Species nodes in DBpedia are represented as Resources, fetched through the link http://dbpedia.org/resource/RESOURCE_NAME, in which RESOURCE_NAME is the name of the resource. The RESOURCE_NAME in species corresponds to its scientific name, composed by genus and species (GENUS_SPECIES). Therefore, the URI property of each species node in FishGraph was defined as http://dbpedia.org/resource/GENUS_SPECIES. Through the URI property, the related data can be obtained from DBpedia.

Departing from the 32,957 species nodes, we traversed the FishGraph to get genus, family, order and class. Each species and their taxonomic classification in FishGraph were compared to the related resource from DBpedia, using its API (see Fig. 11).

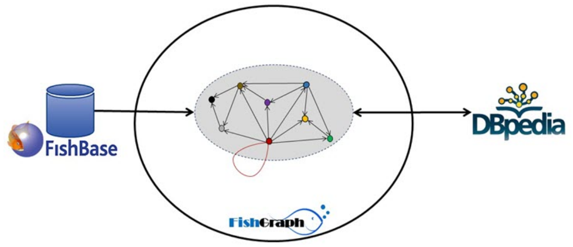

Figure 11. Hybrid architecture with FishBase, FishGraph and DBpedia.

There are reclassified species in DBpedia, which trigger a redirection that can cause a wrong interpretation of the results. For example, the species *Balistes vetula* in the DBpedia has the link http://www.dbpedia.org/resource/Balistes_vetula, which is automatically redirected by DBpedia to www.dbpedia.org/page/Queen_Triggerfish. In this case, it is not possible to fetch data from the initial resource. We verified that the total number of redirects was 5,183, approximately 15% of the species. Therefore, our system handles the redirection increasing the accuracy of the analysis.

The final results (see Fig. 12) showed that: 5,136 species have the same taxonomic classification in FishGraph and DBpedia (15.18%); 7,456 species have some inconsistency in genus, family, order or class (22.62%); and 20,365 (61.79%) species exist only in FishGraph. The resulting report of this analysis can support quality analysis and indicate species that may require review.

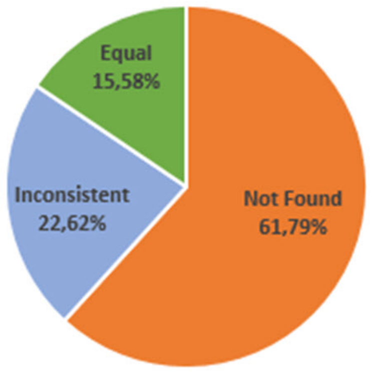

Figure 12. Species classification comparison between FishGraph and DBpedia.

VI. Conclusion

Nowadays, there are several biodiversity information systems maintaining big relational databases, with a lot of knowledge to discover. In this paper, we showed the importance of network-driven analysis for knowledge discovery, contrasted with the ineptitude of relational databases for such analysis. We proposed ReGraph, a hybrid architecture bridging relational and graph databases, with both models integrated and synchronized in their native representation. It has a low impact on the current infrastructure and on the information systems that access data in the relational database.

Our application scenario involves mapping data from FishBase to FishGraph, using ReGraph architecture, to perform a network-driven analysis of identification keys and to compare the taxonomic classification of species of fishes. We showed that a network analysis is a complex task to be performed in a relational database, given the required emphasis on the relations and the transitive aspects of the queries. To address this issue we mapped a large volume of data from FishBase to FishGraph, generating almost 36,000 nodes and 100,000 edges, in which we selected only the relevant data to analyze.

Our architecture contrasts with related work, since it generates a materialization of the graph mapping in a native graph database structure, in which it is possible to create new information and link the mapped data with new inserted data, in an annotation subgraph. We provide an automatic synchronization mechanism for data in the relational and graph databases. We have implemented also a mapping specification that supports automatic and manual mapping. This specification is directly related to the synchronization process, able to automatically infer the fields to be monitored in the relational database and the respective graph elements to be updated.

An additional contribution is the analysis presented to the proposed problems: identification keys and their species and species taxonomic classification, in which we applied graph algorithms, e.g., traversal path and clique percolation method. It resulted in recommendations for the review of identification keys and in a report comparing species in FishGraph and DBpedia. To generate this report, we analyzed data of species from DBpedia, to map all incomplete and divergent data in FishBase. Moreover, we intend to create a user interface, which will facilitate the performance of queries involving graph algorithms and data visualization.

Acknowledgment

Research partially funded by projects NAVSCALES (FAPESP 2011/52070-7), the Center for Computational Engineering and Sciences (FAPESP CEPID 2013/08293-7), CNPq-FAPESP INCT in eScience (FAPESP 2011/50761-2), INCT in Web Science (CNPq 557.128/2009-9) and individual grants from CNPq and CAPES. Thanks to FishBase.org and DBpedia, which provided the data used in this work.

References

[1] T. Berners-Lee, “Giant global graph,” Decentralized Inf. Gr., 2007.

[2] M. Kleppmann, “Should you go Beyond Relational Databases?,” 2009. [Online]. Available: http://blog.teamtreehouse.com/should-you-go-beyond-relational-databases. [Accessed: 23-Mar-2015].

[3] R. Froese and D. Pauly, FishBase 2000: concepts, design and data sources. Los Baños, Laguna, Philippines, 2000, p. 344.

[4] FishBase Consortium, “FishBase,” 2014. [Online]. Available: http://www.fishbase.org/. [Accessed: 23-Mar-2015].

[5] J. Bondy and U. Murty, "Graphs and subgraphs" in "Graph theory with applications," Ontario, Canada: Elsevier Science Ltd, 1976, ch.1, pp1–264.

[6] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen, and D. Wilkins, “A comparison of a graph database and a relational database,” ACM Southeast Regional Conference, pp. 1–6, 2010.

[7] C. T. Have and L. J. Jensen, “Are graph databases ready for bioinformatics?,” Bioinformatics, vol. 29, no. 24, pp. 3107–3108, 2013.

[8] R. Angles, “A Comparison of Current Graph Database Models,” Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference, pp. 171–177, 2012.

[9] R. Angles and C. Gutierrez, “Survey of graph database models,” ACM Comput. Surv., vol. 40, no. 1, pp. 1–39, Feb. 2008.

[10] D. Dominguez-Sal, P. Urbón-Bayes, A. Giménez-Vañó, S. Gómez-Villamor, N. Martínez-Bazán, and J. L. Larriba-Pey, “Survey of Graph Database Performance on the HPC Scalable Graph Analysis Benchmark,” Proceedings of the 2010 International Conference on Web-age Information Management, pp. 37–48, 2010.

[11] S. Raghavan and H. Garcia-Molina, “Integrating Diverse Information Management Systems: A Brief Survey,” IEEE Data Eng. Bull, vol.24, no. 4, pp. 44–52, 2001.

[12] Relational database management system and method for storing, retrieving and modifying directed graph data structures, by R. N. Goldberg and G. A. Jirak (1993, April 06). Patent US 5201046 A. [Online], Available: https://www.google.com/patents/ US5201046

[13] A. Gutiérrez, P. Pucheral, H. Steffen, and J. Thévenin, “Database Graph Views: A Practical Model to Manage Persistent Graphs.,” Proc. 20th Int. Conf. Very Large Data Bases, vol. 33, no. 1, pp. 1–20, 1994.

[14] C. Blakeley, “Virtuoso RDF Views - Getting Started Guide,” OpenLink Software, 2007. [Online]. Available: http://www.openlinksw.co.uk/virtuoso/Whitepapers/pdf/Virtuoso_SQL_to_RDF_Mapping.pdf. [Accessed: 10-Jan-2015].

[15] Oracle Corportation, “Oracle Spatial and Graph Developer’s Guide,” 2013. [Online]. Available: http://docs.oracle.com/cd/E16655_01/appdev.121/e17896.pdf. [Accessed: 26-Feb-2015].

[16] IBM, “DB2 NoSQL Support: DB2 RDF Store,” 2014. [Online]. Available: http://www-01.ibm.com/software/data/db2/linux-unix-windows/nosql-support.html. [Accessed: 26-Feb-2015].

[17] M. J. Carey, L. M. Haas, P. M. Schwarz, M. Arya, W. F. Cody, R. Fagin, M. Flickner, A. W. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J. H. Williams, and E. L. Wimmers, “Towards heterogeneous multimedia information systems: the Garlic approach,” Proc. RIDE-DOM’95. Fifth Int. Work. Res. Issues Data Eng. Object Manag., pp. 124–131, 1995.

[18] C. Bizer, “D2R Map – A database to RDF mapping language,” 12th Int. World Wide Web Conf., pp. 4–6, 2003.

[19] M. Volz, K. Aberer, K. Böhm, and D. GMD-IPSI, “An oodbms-irs coupling for structured documents,” IEEE Data Eng. Bull., pp. 34–42, 1996.

[20] A. De Vries and A. Wilschut, “On the integration of IR and databases,” Proc. 8th IFIP 2.6Working Conf. Database Semant., pp.16-31, 1999.

[21] S. S. Sahoo, W. Halb, S. Hellmann, K. Idehen, T. T. Jr, S. Auer, J. Sequeda, and A. Ezzat, “A survey of current approaches for mapping of relational databases to RDF,” W3C RDB2RDF Incubator Gr., January, 2009.

[22] T. Berners-Lee, “Relational Databases on the Semantic Web,” 1998. [Online]. Available: http://www.w3.org/DesignIssues/RDB-RDF.html. [Accessed: 10-Jan-2015].

[23] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, “Uncovering the overlapping community structure of complex networks in nature and society,” Nature, vol. 435, no. 7043, pp. 814–818, 2005.

[24] R. Tarjan, “Depth-first search and linear graph algorithms,” 12th Annu. Symp. Switch. Autom. Theory (swat 1971), vol. 1, no. 2, pp. 146–160, 1971.

[25] C. Bizer, T. Heath, and T. Berners-Lee, “Linked data-the story so far,” Int. J. Semant. Web Inf. Syst., vol. 5, pp. 1–22, 2009.