



Universidade Estadual de Campinas
Instituto de Computação



Patrícia Raia Nogueira Cavoto

ReGraph: Bridging Relational and Graph Databases

ReGraph: Interligando Bancos de Dados Relacionais e
de Grafos

CAMPINAS
2016

Patrícia Raia Nogueira Cavoto

ReGraph: Bridging Relational and Graph Databases

ReGraph: Interligando Bancos de Dados Relacionais e de Grafos

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestra em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. André Santanchè

Este exemplar corresponde à versão final da Dissertação defendida por Patrícia Raia Nogueira Cavoto e orientada pelo Prof. Dr. André Santanchè.

CAMPINAS
2016

Agência(s) de fomento e nº(s) de processo(s): Não se aplica.

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

C316r Cavoto, Patrícia Raia Nogueira, 1983-
ReGraph : bridging relational and graph databases / Patrícia Raia Nogueira
Cavoto. – Campinas, SP : [s.n.], 2016.

Orientador: André Santanchè.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de
Computação.

1. Banco de dados. 2. Software - Desenvolvimento - Banco de dados. 3.
Ontologias (Recuperação da informação). I. Santanchè, André, 1968-. II.
Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: ReGraph : interligando bancos de dados relacionais e de grafos

Palavras-chave em inglês:

Databases

Software development - Databases

Ontologies (Information retrieval)

Área de concentração: Ciência da Computação

Titulação: Mestre em Ciência da Computação

Banca examinadora:

André Santanchè [Orientador]

Rodrigo Dias Arruda Senra

Ricardo da Silva Torres

Data de defesa: 04-02-2016

Programa de Pós-Graduação: Ciência da Computação



Universidade Estadual de Campinas
Instituto de Computação



Patrícia Raia Nogueira Cavoto

ReGraph: Bridging Relational and Graph Databases

ReGraph: Interligando Bancos de Dados Relacionais e de Grafos

Banca Examinadora:

- Prof. Dr. André Santanchè (*Orientador*)
Instituto de Computação - UNICAMP
- Dr. Rodrigo Dias Arruda Senra
EMC Brazil Research & Development Center
- Prof. Dr. Ricardo da Silva Torres
Instituto de Computação - UNICAMP
- Prof. Dr. Luciano Antonio Digiampietri (*Suplente*)
Escola de Artes, Ciências e Humanidades - USP
- Dr. Júlio César dos Reis (*Suplente*)
Instituto de Computação - UNICAMP

A ata da defesa com as respectivas assinaturas dos membros da banca encontra-se no processo de vida acadêmica do aluno.

Campinas, 04 de fevereiro de 2016

Acknowledgements

It was a hard task to finish this research and, indeed, it would not be possible without the supportive, kindness and lovely presence (even in my heart) of each one of you. I am really grateful for having you in my life.

André, the most supportive and competent advisor that I could have. Thank you for all the teachings, ideas, reviews and patience during this work. More important than that, thank you for being a friend, and for all the conversations and for the guidance. Rêgine Vignes thanks for the support in the biology domain and for the brilliant examples of analyses that could be performed over the data. Members of the dissertation committee, thank you for the valuable feedbacks.

Professor Claudia Bauzer Medeiros, for all the insightful ideas about this work. Friends from LIS, for the conversations, friendship and comments in the meetings that helped this work, specially to Lucas, Jaqueline, Jaudete, Matheus and Victor. And, of course, thanks to all the professors that contributed to my growth.

My mother, Sílvia, that, unfortunately, is not with me anymore. The woman who taught me to be confident and to believe in my dreams. The most responsible for each of the achievements in my life. Thank you eternally for everything that you did for me. I miss you so much.

Paulo, my beloved husband, thank you for the patience, kindness and support. Of course, I could not forget to thank you also for the insights and reviews of this work. You are awesome!

My father, Antonio, my aunt, Márcia, and my cousin – almost sister, Juliana, for always caring and supporting me in my life.

My friends João, Cris, Tati, Karina, Rodrigo, Letícia and Fujii for reminding me that, sometimes, we need some fun too.

Professor Juan, from PUC-Campinas, for the support with the schedule of my classes during my master's studies.

Finally, this work would not be possible without the data provided by the FishBase Consortium, the dedication of the Institute of Computing's staff, the technologies provided by the many competent open/free software projects used throughout this research and the support from the funding agencies: FAPESP/Cepid in Computational Engineering and Sciences (2013/08293-7), the Microsoft Research FAPESP Virtual Institute (NavScales project), CNPq (Mu-ZOO Project), FAPESP-PRONEX (eScience project), INCT in Web Science, and individual grants from CNPq. The opinions expressed in this work do not necessarily reflect those of the funding agencies.

Resumo

Redes estão em todos os lugares. Desde interações sociais: família, amigos, *hobbies*; passando pela computação: computadores conectados na Internet; e até mesmo na natureza: cadeias alimentares. Pesquisas recentes mostram a importância das conexões entre os dados e também da análise das redes para descobrir novos conhecimentos nos dados existentes. Além disso, os esforços para a disponibilização de dados padronizados na Internet – *Linked Open Data* e *Semantic Web* – têm proporcionado o crescimento de repositórios abertos de conhecimento na rede; a maioria utilizando o modelo de grafos RDF (*Resource Description Framework*). Contudo, muitos dados são armazenados em bancos de dados relacionais, cujo modelo não foi projetado para atender consultas com alto grau de transitividade nos relacionamentos. Por outro lado, o modelo flexível de grafos tem um ótimo desempenho nas análises envolvendo relacionamentos transitivos entre os dados e na topologia da rede, como por exemplo, em uma análise de componentes conexas. Portanto, nossa pesquisa é inspirada pela abordagem OLAP (*OnLine Analytical Processing*) para a criação de uma base especial orientada à análise dos dados com foco nas ligações e na topologia da rede, utilizando grafos. Nesta dissertação, nós apresentamos o ReGraph, um *framework* para mapear dados de uma base relacional para uma base de grafos, gerenciando a coexistência e evolução de ambas as bases, funcionalidade esta que não é contemplada pelos trabalhos relacionados. O ReGraph tem baixo impacto na infraestrutura existente, permitindo a geração de um modelo de grafos flexível e adaptado a cada esquema relacional mapeado. Utiliza um processo inicial de ETL (*Extract, Transform and Load*) para replicar os dados existentes no modelo relacional para o modelo de grafos. O serviço de sincronismo é responsável por refletir automaticamente as alterações realizadas no modelo relacional para o modelo de grafos. O *framework* também provê uma funcionalidade para anotação dos dados no grafo, que permite materializar inferências e incluir novas informações, possibilitando a conexão dos dados existentes no grafo local com outros grafos de conhecimento disponibilizados na *Web*. Neste trabalho, utilizamos o ReGraph para gerar o FishGraph, uma base de dados de grafos criada a partir da base relacional FishBase. Usando a base de dados FishGraph, realizamos experimentos envolvendo a análise das conexões entre milhares de chaves de identificação e espécies de peixes e conectamos estes dados com a DBpedia, criando anotações na base de grafos local que geraram novas informações a partir dos dados existentes.

Abstract

Networks are everywhere. From social interactions: family, friends, hobbies; passing through computer science: computers on the Internet; to nature: as food chains. Recent research shows the importance of links and network analysis to discover knowledge in existing data. Moreover, the Linked Open 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. However, a lot of data are stored in relational databases, whose model has not been designed to address queries with many transitive relations. On the other hand, the flexible graph model is suitable for data analysis focusing on links, their transitivity and the network topology, e.g., a connected component analysis. Therefore, 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 this dissertation, we present ReGraph, a framework to map data from a relational to a graph database, managing a dynamic coexistence and evolution of both, not supported by related work. ReGraph has minimum impact on the existing infrastructure, providing a flexible and tailored graph model for each relational schema. It uses an initial ETL (Extract, Transform and Load) process to replicate the existing data in the graph database. A scheduled service is responsible for automatically reflecting changes in the relational data into the graph, keeping both synchronized. ReGraph also provides an annotation functionality to materialize inferences and to support data enrichment, which enables linking the local database to global knowledge graphs on the Web. We have used the ReGraph framework to generate FishGraph, a graph database created from the FishBase relational database. Using FishGraph we developed experiments to analyze the connections among thousands of identification keys and species, and we have linked local data to DBpedia, creating annotations over the local graph and providing new knowledge from existing data.

List of Figures

2.1	FishBase tables	17
2.2	XML structure for the identification key problem	18
2.3	Graph model for the identification key problem	18
2.4	Proposed hybrid architecture	22
2.5	FishGraph model for identification keys and species classification	25
2.6	Annotation edges “share” between two keys and their properties	25
2.7	Distribution of the keys by species shared with another key	25
2.8	Clique of identification keys of size 10 and their respective species	26
2.9	All of the 863 connected components	27
2.10	Max Out-Degree key component with species colored by family	28
2.11	Hybrid architecture with FishBase, FishGraph and DBpedia	30
2.12	Species classification comparison between FishGraph and DBpedia	31
3.1	The ReGraph Framework	33
3.2	ReGraph Interface: Manual Mapping and Graph Model Viewer	36
4.1	Graph Model for Taxonomic Classification and Countries	40
A.1	Part of identification key 799 of teleostean families from East Africa	49
A.2	Part of identification key 799 represented by a tree structure	50
B.1	Synchronization Process detailed flow	52
C.1	ReGraph Screen of Automatic "Comparison" Annotation	54
C.2	Node “Species” after the Comparison Annotation Process with DBpedia	55
C.3	ReGraph Automatic "New" Annotation Screen	56
C.4	Node “Country” after the New Annotation Process with GeoNames	57

Contents

1	Introduction	11
2	FishGraph	14
2.1	Introduction and Motivation	14
2.2	Research Scenario	15
2.2.1	The FishBase Database	15
2.2.2	Why Graphs?	16
2.3	Theoretical Foundations and Related Work	18
2.3.1	Graph Databases	18
2.3.2	Database Models Integration	19
2.4	A Hybrid Architecture for the FishGraph	21
2.4.1	Hybrid Architecture: Relational and Graph	21
2.4.2	The FishBase Entities Analysis	22
2.4.3	Mapping Rules	23
2.4.4	From FishBase to FishGraph	23
2.5	Experiments and Results	24
2.5.1	Experiment 1: Identification Keys	24
2.5.2	Experiment 2: Species Classification	29
2.6	Conclusion	30
3	ReGraph	32
3.1	Introduction	32
3.2	The ReGraph Framework	33
3.2.1	The Mapping Module	34
3.2.2	The Graph Module	35
3.2.3	The Sync Module	35
3.3	Software Overview	36
3.4	Conclusions	37
4	Annotation-Based Method	38
4.1	Introduction	38
4.2	Related Work	39
4.3	ReGraph	39
4.3.1	The ReGraph Framework	39
4.3.2	From FishBase to FishGraph using the ReGraph framework	40
4.4	Automatic Annotation-Based Method	41
4.4.1	The Comparison Annotation Type	41
4.4.2	The New Annotation Type	42
4.5	Conclusions and Future Work	42

5 Conclusion and Future Work	43
Bibliography	45
A Identification Key Visual Example	49
B Synchronization Process	51
C Annotation-Based Method: Experiments	53
C.1 "Comparison" Annotation: Taxonomic Classification Scenario	53
C.2 "New" Annotation: Country Scenario	55

Chapter 1

Introduction

The analysis of data as a network, focusing on the relations, is becoming very important to discover knowledge in existing content. In addition, the use of data available in global graphs, as Linked Open Data and Ontologies, is growing fast and is bringing with it the popularization of the graph structure to represent information networks. Nevertheless, in different institutions and organizations we still have a big volume of data stored using the relational database model. This model was not designed to address requests focusing on the network of links and interactions. The flexible graph database model contrasts with the rigid relational model since this model is suitable for data analysis focusing on links and the network topology and it allows to store and maintain flexible data representations, which replace a rigid schema for a soft pattern.

In this context, several approaches arose to transform data from relational databases into structures connected as networks, mainly generating a database view modeled as a graph ([7] and [11]). With data in a view modeled as a graph, it is not possible to change nor add new data directly in the graph to perform specific analyses or to improve the generated knowledge. Likewise, other works were interested in connecting a given graph network with the Semantic Web, enriching the existent local knowledge ([1], [2], [30] and [33]). Most of these approaches start from an existent graph structure created to connect concepts with the Semantic Web, through the use of ontologies.

This work was developed over a practical scenario of FishBase¹, a fish knowledge base and a global information system encompassing several aspects of fishes, e.g., identification keys, taxonomic classification, ecosystems, predators, geographic locations, etc. FishBase currently has 33,200 species of fishes registered and these data are used by research scientists, fishery managers, biologists and enthusiasts.

Motivated by a joint research involving network-driven data analysis over FishBase, with biologists from the Muséum national d'Histoire naturelle in Paris, France, we defined the following requirements for this research:

- Connect data in a network structure: allows to map and connect data from FishBase, a relational database, with other relational database sources, through a network structure stored in a property graph database, in our case, using Neo4j;

¹<http://www.fishbase.org/> [Accessed: 2016-Jan-05]

- Keep the graph database synchronized with the relational database: after the data migration from a relational to a graph database, it is necessary to keep data in the graph database synchronized, reflecting all the changes executed in the relational database, managing a dynamic coexistence and evolution of both;
- Enrich the knowledge: allows to add new data into the local graph and to connect it with the Semantic Web increasing the existing knowledge;
- Perform analysis exploring the network topology: in a graph structure it is easy to perform network analysis, e.g., page rank, short path, components, cliques, etc., which is a hard task for relational databases.

Different from related work, in our approach, the local graph is a dynamic graph database evolving over time, aligned with the relational database source. It allows creating new manual and automatic annotations directly in the graph, even connected with global graphs available in the Semantic Web. These annotations stay consistent during the database evolution. In this work, we define as annotation any information in the graph not originated from the relational database, e.g., nodes, properties and edges created manually or automatically in the graph .

To validate the mentioned requirements, we proposed, implemented and tested a framework, named ReGraph², to map data from a relational to a graph database, providing a hybrid architecture that bridges both databases, keeping them connected, synchronized and in their native representations. ReGraph has minimum impact on the existing infrastructure, providing a flexible and tailored graph model for each relational schema.

Our research was 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. It applies an initial ETL (Extract, Transform and Load) process to replicate the existing data in the graph database. Later, a scheduled service is responsible for reflecting changes in the relational data into the graph, keeping both synchronized. ReGraph also provides an annotation functionality that allows users to add manually new information in the mapped graph, providing support to materialize inferences and data enrichment. Moreover, the framework can link local and global knowledge graphs through an automatic annotation-based method, creating annotations obtained from the Semantic Web using the specified Ontologies.

The first paper published of this work was “Arquitetura Híbrida de Integração entre Banco de Dados Relacional e de Grafos: Uma Abordagem Aplicada à Biodiversidade” (“Hybrid Architecture for Integration of Relational and Graph Databases: An Approach Applied to Biodiversity”), presented at the XIII Workshop de Teses e Dissertações em Banco de Dados, WTDBD 2014 [15], in which we presented our proposal.

The remainder of this dissertation is organized as a collection of other three published papers, included as chapters. Chapter 2 represents the paper “FishGraph: A Network-Driven Data Analysis”, presented at the 11th IEEE International Conference on eScience, eScience 2015 [12]. This paper establishes our application scenario and problem and

²<http://patricia.cavoto.com.br/regraph/> [Accessed: 2016-Jan-05]

argues in favor of the advantages of using graph databases on the network data analysis. The main contributions of the paper are:

- The proposal of an architecture to generate and maintain a graph database mapped from a relational database and further enriched;
- An experiment in which we have generated FishGraph, a graph database, from FishBase;
- Four application experiments involving network-driven data analysis over FishGraph that are complex to be performed in the relational database; and
- A software prototype for ReGraph to link FishGraph with DBpedia³.

Chapter 3 corresponds to the paper “ReGraph: Bridging Relational and Graph Databases”, presented at the 30o Simpósio Brasileiro de Banco de Dados, SBBD 2015, in the demo session [14]. The main contribution of the paper is:

- ReGraph, an implemented framework to map data from a relational to a graph database, managing a dynamic coexistence and evolution of both.

Chapter 4 represents the paper “Annotation-Based Method for Linking Local and Global Knowledge Graphs”, presented at the Seminário de Pesquisa em Ontologias do Brasil, ONTOBRAS 2015 [13]. The paper presents an approach to integrate the local graph database, generated by ReGraph, with global graphs on the Web. The main contributions of the paper are:

- An extension of the ReGraph framework to generate automatic annotations through the link of the local graph database with global graphs on the Web; and
- The proposition of an annotation method able to coexist with mapped data from the relational database. It introduces two possible annotation types (i) new: adds new information obtained from the Web in the local graph, as nodes, properties and/or edges; and (ii) comparison: compares local data with the related data on the Web, providing a status report of the comparison.

Finally, Chapter 5 concludes the dissertation, presenting our final remarks and future work.

³<http://www.dbpedia.org/> [Accessed: 2016-Jan-05]

Chapter 2

FishGraph: A Network-Driven Data Analysis

2.1 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 [6], 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 [24], 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, a database and information system for biological data storage of fish species [19].

In this chapter, 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. Fur-

thermore, 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 2.2 details the research scenario. Section 2.3 presents the theoretical foundations and related work. Section 2.4 presents our hybrid architecture and describes the migration process to integrate the FishBase data in FishGraph. Section 2.5 presents our experiments and their results. Section 2.6 presents our conclusions.

2.2 Research Scenario

2.2.1 The FishBase Database

Data in the FishBase system are stored in a relational database model. It currently has 33,000 registered species [18] 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 [19]. 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**¹: 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 Figure 2.1.

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 (Figure 1a). The next set of tables (Figure 1b) links the SPECIES with countries (COUNTRF) and ecosystems (ECOSYSTEMREF). There are 311 countries² and 1,019 ecosystems in the FishBase database. The last set of tables (Figure 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.

2.2.2 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:

¹See Appendix A for a visual example of the identification key.

²Islands are also counted as countries in FishBase.

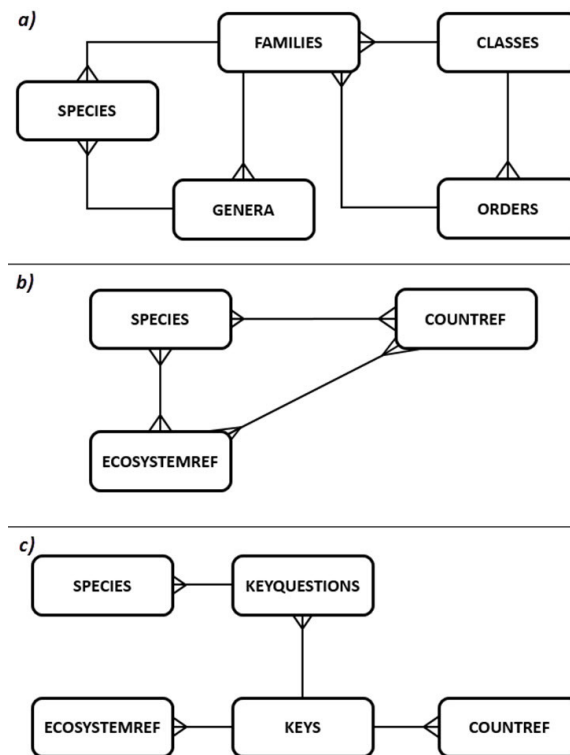


Figure 2.1: FishBase tables: a) species and their taxonomic classification; b) species, countries and ecosystems; and c) identification keys.

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 Figure 2.2).

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.

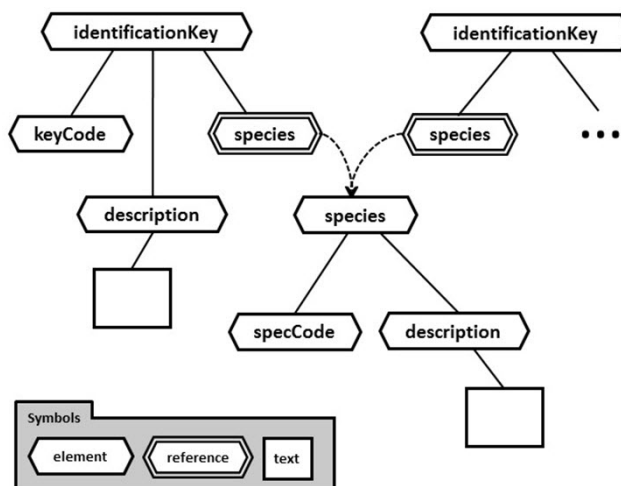


Figure 2.2: XML structure for the identification key problem.

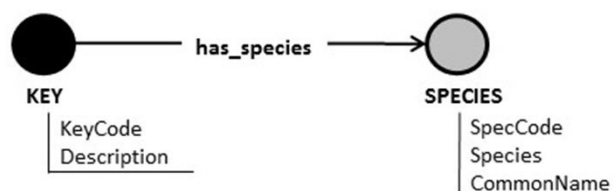


Figure 2.3: Graph model for the identification key problem.

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.

Figure 2.3 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.

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.

2.3 Theoretical Foundations and Related Work

2.3.1 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 [10]. 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 [35].

Have and Jensen [22] 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 [3].

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 [4]. 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 [3], as presented in Figure 2.3.

Based on the comparison of different graph database models [3] and the performance analyses of four graph databases [16], 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.

2.3.2 Database Models Integration

Raghavan and Garcia-Molina [31] 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:

2.3.2.1 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 [20] 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 [21] 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 [9] 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 [26]. The IBM DB2 RDF stores and retrieves data in an RDF graph format, expanding its relational database [23]. 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.

2.3.2.2 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 [11] defines an object-oriented extension of the SQL as a unified language. The D2R Map [7] 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 minimum 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.

2.3.2.3 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

Table 2.1: Related Work Comparison

Paper	a	b	c	d	e	f	g
Relational: [20], [21]	Manual	SQL	N	L	L	N	N
Relational: [9]	Both	SQL/SPARQL	N	L	M	Y	Y
Relational: [26], [23]	Auto	SQL/SPARQL	Y	H	M	Y	N
Integration Module: [11]	Manual	Own	Y	L	H	Y	Y
Integration Module: [7]	Both	SQL/SPARQL	N	L	M	Y	N
Layered: [36], [37]	Manual	Superior Layer	Y	H	H	Y	Y

Vodar architecture [36] integrates an IRS with an object-oriented database; De Vries and Wilschut [37] 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.

2.3.2.4 Summary

Inspired by the framework created by the W3C Incubator Group, RDB2RDF [32], in Table 2.1 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).

2.4 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.

2.4.1 Hybrid Architecture: Relational and Graph

Figure 2.4 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.

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

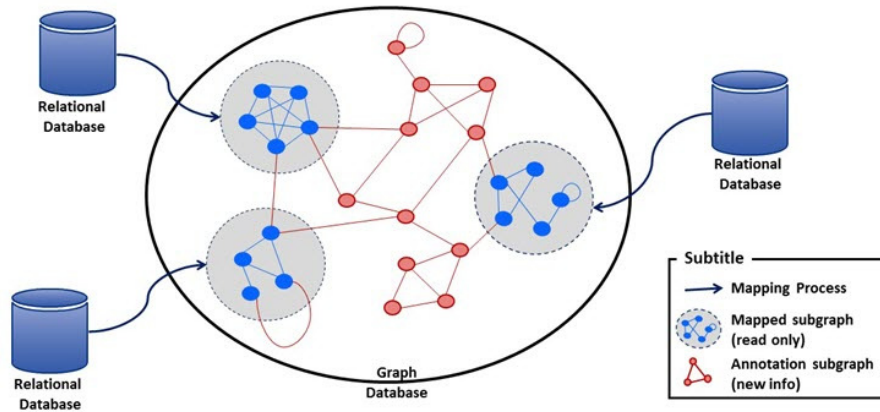


Figure 2.4: Proposed hybrid architecture.

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.

2.4.2 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, FAM-

ILIES, 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.

2.4.3 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'.

2.4.4 From FishBase to FishGraph

Our mapping model has been designed taking the "FishBase \rightarrow FishGraph" mapping problem as a starting point. Tim Berners-Lee [5] 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.

2.5 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 Figure 2.5. The diagram represents a graph modeling of the selected tables from FishBase (presented in Figure 2.1 and follows the same approach of Figure 2.3 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.

2.5.1 Experiment 1: Identification Keys

To study the network formed by data from FishBase, we connected FishGraph with Gephi³, 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 FishBase⁴ website.

2.5.1.1 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 Figure 2.6) 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.

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 Figure 2.7).

³<http://gephi.github.io/> [Accessed: 2016-Jan-05]

⁴<http://fishbase.org/keys/allkeys.php> [Accessed: 2016-Jan-05]

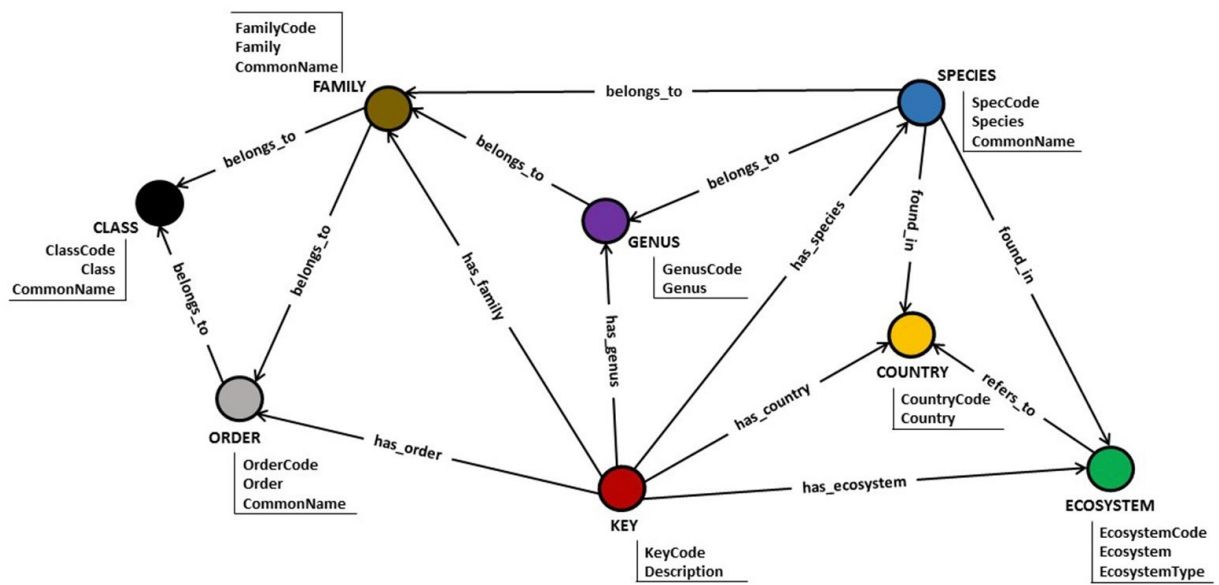


Figure 2.5: FishGraph model for identification keys and species classification.

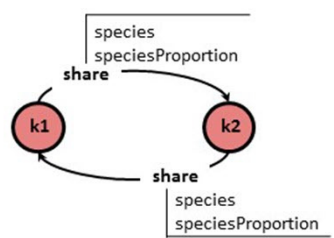


Figure 2.6: Annotation edges “share” between two keys and their properties.

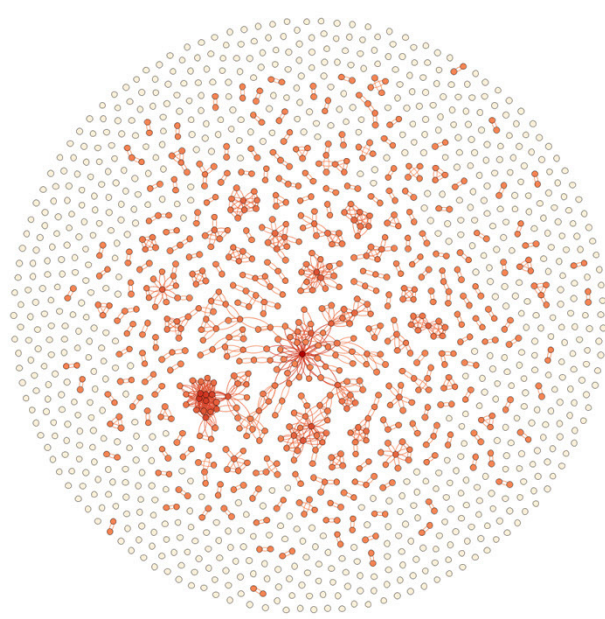


Figure 2.7: Distribution of the keys by species shared with another key.

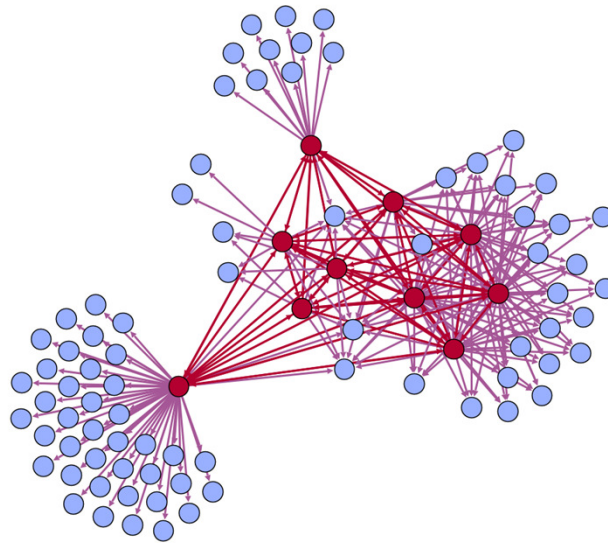


Figure 2.8: Clique of identification keys (red nodes) of size 10 and their respective species (blue nodes).

Over this set of data, we applied the Clique Percolation Method (CPM) [29] 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 Figure 2.8). The clique analysis was the starting point for the next steps.

2.5.1.2 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 [34] 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 [10].

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 Figure 2.9), 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

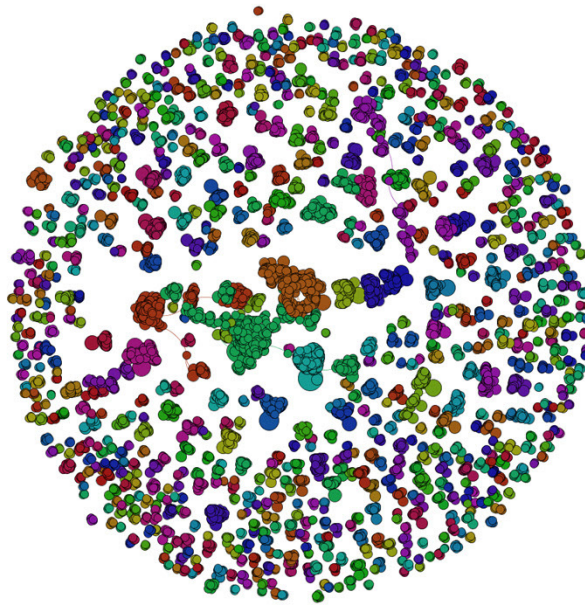


Figure 2.9: All of the 863 connected components (distinct colored).

the two keys could give more information about the species and make the identification process more efficient.

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.

2.5.1.3 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 Figure 2.10). 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*).

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.

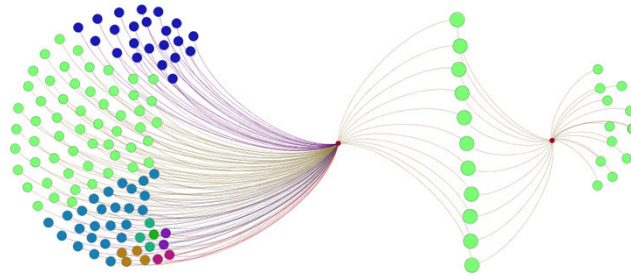


Figure 2.10: Max Out-Degree key component with species colored by family.

Table 2.2: 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

2.5.1.4 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 2.2 and Table 2.3).

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.

Table 2.3: Top Ten Countries

	Countries	Total Keys
1	China	688
2	Australia	614
3	Japan	596
4	Indonesia	557
5	Taiwan	556
6	Philippines	566
7	South Africa	506
8	Vietnam	505
9	India	482
10	Papua New Guinea	469

2.5.2 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 [8]. 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 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>;

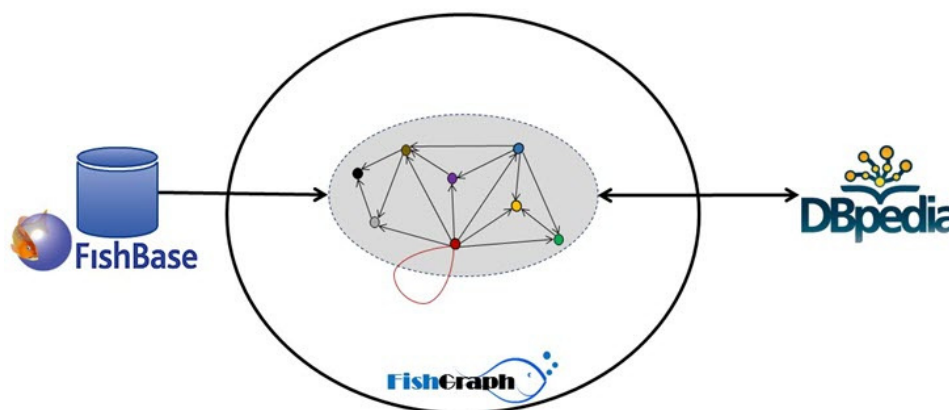


Figure 2.11: Hybrid architecture with FishBase, FishGraph and DBpedia.

- 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 of 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 Figure 2.11).

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 from http://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 Figure 2.12) 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.

2.6 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

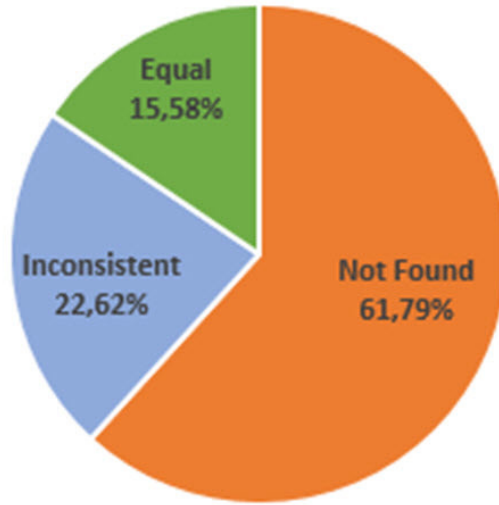


Figure 2.12: Species classification comparison between FishGraph and DBpedia.

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.

Chapter 3

ReGraph: Bridging Relational and Graph Databases

3.1 Introduction

In the data analysis context, links have been increasingly considered as important as the data elements that they connect. It is possible to analyze these links, and the existing latent semantics, through the analysis of the network topology formed by them. Graph databases are very effective in this type of analysis due to their flexible structure, which helps in defining the fitted model to each required analysis. Moreover, the OLAP (OnLine Analytical Processing) concept of creating an exclusive database for data analysis is a consolidated approach. This database is designed and tuned for the respective operations – over graphs, in our case – allowing users to manage data without impact in the original databases and to dynamically create information over the existing data.

Combining the graph database and the OLAP concept, we propose ReGraph, a framework that generates a graph database from a relational database, providing extra functionalities for analysis. Using a tailored mapping, an ETL process generates the graph database linked to the relational one, preserving both databases in their native forms. Beyond synchronizing the dynamic evolution of the relational database with the graph, our framework supports adding new data in the graph, annotating the mapped one. Annotations enrich the semantics in the existing data and can improve the network analysis performed in the graph database. As part of the ReGraph work, we have implemented FishGraph, a project in which we generate a graph database from FishBase, a relational database and information system for biological data storage of fish species. Through FishGraph, the relational and the graph databases interact with each other, maintaining their native forms. In Cavoto et al. [12] we present two scenarios in which a graph database can be a better choice for network analysis in the FishBase context and how annotations are helpful in improving this analysis.

Concerning the related work, there are several approaches for representing data as a graph structure starting from a relational database. The Database Graph Views [21] proposes an abstraction layer as a mechanism for creating views to manipulate graphs, independent of the physical arrangement where the original data is stored. The D2RQ

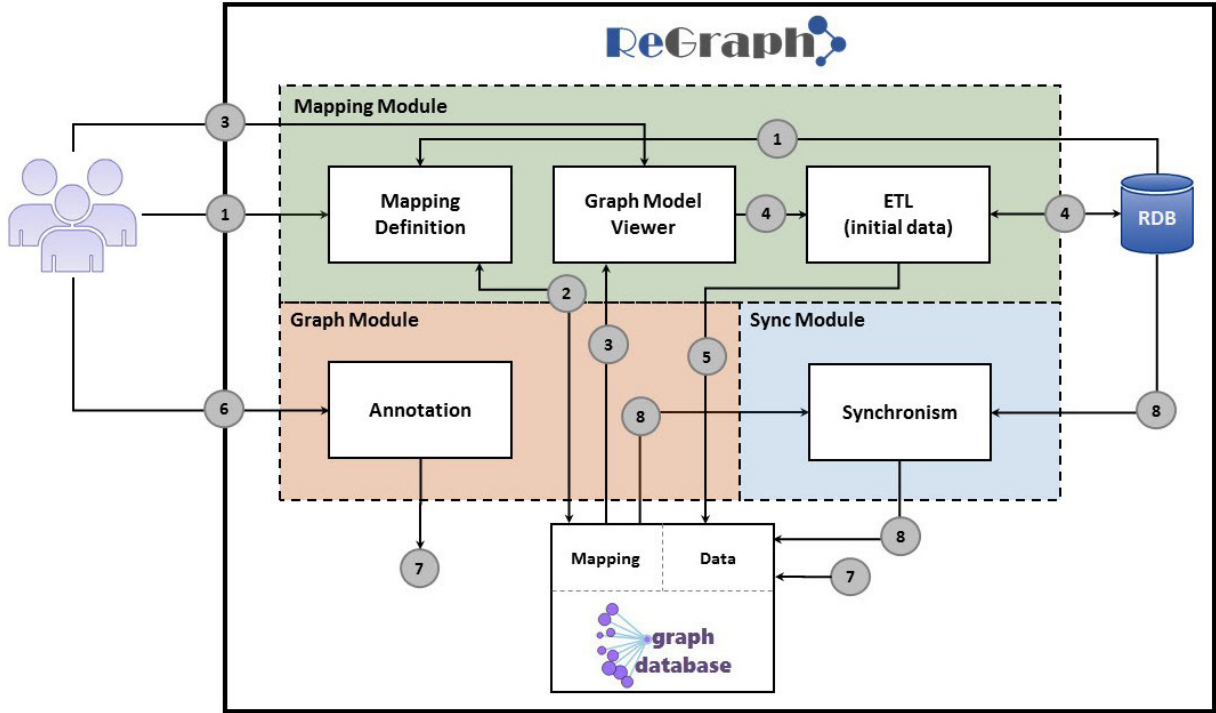


Figure 3.1: The ReGraph Framework.

Map [7] proposes the integration of ontologies in RDF with relational databases through a module that creates a virtual RDF graph. As both approaches do not materialize the graph database, they neither take fully benefit of the advantages in the network analysis offered by a database tuned for graph operations, nor the possibility of adding new data in the existing data. The Virtuoso RDF View [9] proposes mapping relational data to RDF. It operates exclusively over the RDF model and the Virtuoso relational database, restricting the use of other graph models and database management systems.

In this paper, we present the ReGraph framework, detailing the generation of the graph database starting from a relational database, keeping both in their native forms and interacting with each other, dividing tasks of management and analysis according to their specialties. Moreover, ReGraph allows the creation of new information by annotating the existing data and provides a dynamic evolution of the graph database, reflecting the changes executed in the relational database.

3.2 The ReGraph Framework

ReGraph is a framework that departs from a relational database, allowing to create and maintain within a graph database two connected subgraphs: mapped and annotation. It mainly addresses a Property Graph model, even though it can also work with the RDF model. Figure 3.1 illustrates an overview of the ReGraph framework that has three main modules: Mapping, Graph and Sync.

ReGraph keeps both databases in their native forms and does not require changes in the relational schema, which will generate a minimum impact on the existing infrastructure. Therefore, it allows to users to keep using the existing systems that operate over the

relational database and using the dynamically synchronized graph database for analysis, focusing on the relations among data elements.

To the best of our knowledge, ReGraph is the first framework that migrates data from the relational database to a property graph database, Neo4J, keeping the databases in their native forms and providing a dynamic evolution of both.

3.2.1 The Mapping Module

In the Mapping Module, through the Mapping Definition process, flow 1 in Figure 3.1, it is possible to map data from a relational database schema to a graph database. There are two mapping options:

1. **Automatic:** maps the relational database schema automatically to a graph database model. Tim Berners-Lee [5] discussed a set of mapping principles when converting relational databases to RDF (Resource Framework Description). The principles are as follows: a record in the relational database becomes an RDF node; a column name becomes an RDF predicate (a labeled edge); and a table cell becomes a value. As the target of ReGraph is a property graph, and the Tim Berners-Lee mapping considers an RDF graph, we have adapted the rules to our model as follows:
 - a table in the relational model becomes a node type, also called label;
 - each record becomes a unique node;
 - each table cell (except the foreign keys) of the record becomes a node's property;
 - each foreign key becomes an edge connecting the nodes generated by the involved records.
2. **Manual:** allows defining tailored mappings, according to their analysis requirements. For each table and column of a given relational database, it is possible to define a mapping to graph labels, nodes, properties and edges. Moreover, one can also map a relational database to an RDF graph model by keeping properties unselected – in this case, we are only considering the RDF graph model, since fully RDF mapping would require other RDF semantic representation concerns.

After the mapping definition, ReGraph saves this mapping in the graph database, creating it as a representation of the graph model, as shown in Figure 3.1, flow 2. The graph model viewer presents a representation of the mapping and provides a previous data organization in the graph database, flow 3 in Figure 3.1.

The last feature of this module is the ETL process, responsible for retrieving data from the relational database and inserting data into the graph database, generating the mapped subgraph, as specified in the mapping definition, Figure 3.1, flows 4 and 5. Each table and column mapped from the relational database receives a trigger that registers any insert, update or delete in a notification table, in order to perform the synchronism process in the future. All nodes and relationships generated from the relational database have a special property called “DataSource” defined as “Mapped”, which forbids changes in

their data directly in the graph database. Any required change in the mapped subgraph data must occur in the relational database and the synchronism process performs the respective changes in the graph database.

3.2.2 The Graph Module

The Graph Module allows the interaction with the graph database as annotations over the mapped subgraph, as shown in Figure 3.1, flow 6. While it is not allowed to change the mapped data in the graph, in order to maintain the consistency, it is possible to create new nodes and edges over this data as annotations, Figure 3.1, flow 7. The new annotation nodes and edges have the “DataSource” property, mentioned previously, defined as “Annotation”. The Annotation Graph aims to facilitate the network analysis, adding details over the existing data and enabling to materialize intermediary nodes and relations inferred from existing data.

In the end, there are two main distinct, but interconnected subgraphs: the mapped and the annotation. One cannot create nodes and edges in the annotation subgraph using the existing labels defined in the mapped subgraph. This rule aims to prevent the creation of redundant and conflicting data between the mapped and the annotation subgraphs.

3.2.3 The Sync Module

The Sync Module is responsible for maintaining the graph database synchronized with the relational database, providing a dynamic evolution of both, as shown in Figure 3.1, flow 8. It also keeps the graph database updated with the last changes to perform the network analysis. A scheduled service periodically reads the data registered in the notification tables, populated by the triggers created in the end of the ETL process, and runs the synchronism process, updating the graph database with the new information¹. There are three main rules applied in synchronism process:

1. **Delete:** there are two distinct policies that can be configured for deleted records in the relational database. The first one is the “Delete” policy, in which, when a mapped record is deleted in the relational database, the synchronism process reflects it in the graph by deleting the respective nodes, properties or edges. In this case, all the related annotations exclusively linked with the deleted nodes are also deleted. The second is the “Keep” policy, in which, when a mapped data is deleted in the relational database, the synchronism process performs only an update in the “DataSource” property, changing it from “Mapped” to “Deleted” and keeping the respective nodes and edges in the graph database. Moreover, it is not possible to perform changes in the “Deleted” nodes and edges. The “Keep” policy has direct impact in the Insert rule;
2. **Insert:** when a new record is inserted into a mapped table in the relational database, the synchronism process will check if these data already exists in the graph database as “Deleted” data. If these data do not exist, the synchronism process maps it

¹See Appendix B for a detailed flow of the synchronization process.

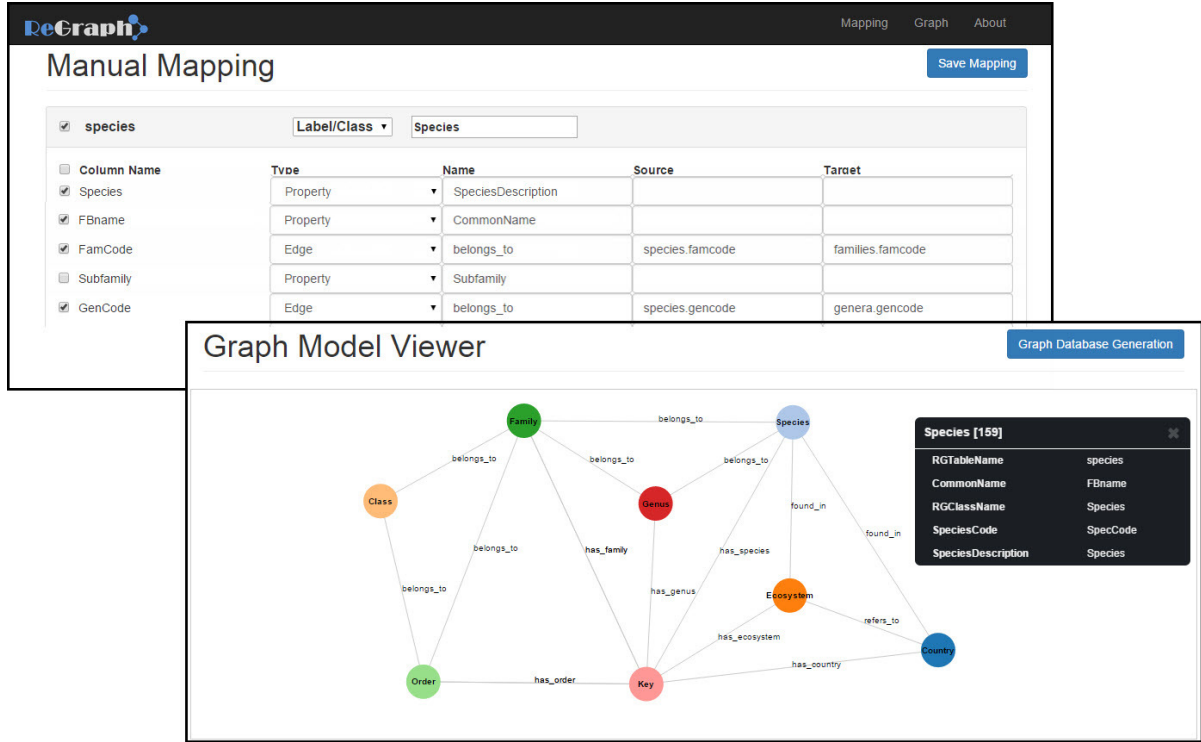


Figure 3.2: ReGraph Interface: Manual Mapping and Graph Model Viewer.

onto the graph database, according to the mapping configuration. Otherwise, the synchronism process will return the “DataSource” property to “Mapped” again, and update all the related data;

3. **Update:** when a mapped record in the relational database is updated, the synchronism process will update the respective data in the graph, according to the mapping definition.

3.3 Software Overview

The ReGraph framework has a web interface, providing easy access to the user through any browser. Figure 3.2 presents an overview of the interface, with an emphasis in the mapping and the graph model viewer screen.

In the first step, the user defines the general configuration, including information about the relational and the graph databases (server, instance, user and password), the mapping type from the relational to the graph database (automatic or manual) and the time interval among synchronism cycles.

The automatic mapping option lists all tables and columns of the relational database, and users can filter their interests removing tables or columns from the mapping, unchecking the respective tables/columns (except the primary and foreign keys). It helps in migrating only the necessary data to the graph database, e.g., fields containing detailed descriptions and big notes could be irrelevant in a network analysis. As in the automatic mapping, the manual mapping also lists all tables and columns from the relational

database. The user checks each table and column to be mapped, having the duty of defining the appropriate destination type, i.e., a label, node, property or edge.

After concluding the mapping, the “Save Mapping” button saves the mapping defined by the user and creates a graph database model that can be visualized by the user. It represents an abstract model of the graph database, presenting to the user a review of the mapping, as shown in Figure 2. At this point, it is possible to change the mapping in order to find the better model to fit an analysis. Whenever the user is satisfied with the mapping and model, it is possible to perform the first data load, carried by an ETL static process. After the mapping and the ETL processes, it is only possible to include tables and/or columns – it is not possible to remove tables and/or columns from the mapping. This rule aims to avoid inconsistencies in the graph database, considering the annotation subgraph. The ETL process also creates the triggers associated with the mapped data to perform the synchronism.

The synchronism process is transparent to the user. It is operated by a scheduled service running automatically on the server. This process is responsible for maintaining a dynamical evolution of the graph database, considering the changes executed in the relational database and following the rules explained in the previous section.

The Graph Module encompasses routines to compute new graph elements inferred from the existing ones or to enrich existing data with external resources, creating new labels, nodes, edges and properties. One example using the Graph Module is an interface with DBpedia, in which the user can compare a specific subgraph of the database with the DBpedia RDF content. It produces three kinds of annotation: (a) equal – data that are the same in the graph database and in DBpedia; (b) not found – data that exist in the graph database but does not exist in DBpedia; and (c) inconsistent – data in the graph database that has any difference compared to DBpedia. These annotations point data to be reviewed, which are presented in a report to the user.

3.4 Conclusions

In this paper, we present ReGraph, a framework bridging relational and graph databases. It offers to the user operations of automatic or manual mapping from a relational to a graph database, dynamically updating data in the graph according to the changes performed in the relational database. Moreover, the ReGraph infrastructure offers mechanisms to interconnect data available in the graph database with third party resources, as data available in DBpedia, allowing the enrichment of existing data. We are working to connect the graph database with other data sources on the Web. To the best of our knowledge, ReGraph is the first framework that maps data from a relational database to a property graph database, keeping the databases in their native forms and providing a dynamic synchronized evolution of both.

Chapter 4

Annotation-Based Method for Linking Local and Global Knowledge Graphs

4.1 Introduction

Real-world phenomena as biological processes, social networks and information systems have been increasingly modeled as networks, where nodes can represent individuals, computers, species, proteins, etc. and links the interactions among them. Recent research is pointing graphs as the fitted structure to store this kind of data, in which the relations among data elements are as important as the elements themselves. In the biology field, there are many uses for graphs, including metabolic networks, chemical structures and genetic maps [35]. The challenge is how to explore the network "behind" data available in existing information systems for analysis when data are stored in formats that do not valorize such network structure.

This challenge motivated our proposition of ReGraph, a framework inspired in the OLAP approach, which creates a special local graph database designed for network-driven analyses, aligned with an existing relational database. We applied ReGraph to taxonomic data from FishBase to create FishGraph [12].

In this paper, we present an automatic annotation-based method to link our local graph database to global graphs from the Semantic Web, applied to link FishGraph data with DBpedia. Our method contributes in the data quality analysis, in the enrichment of the local database and in building the Giant Global Graph.

This is a work in progress concerning how to relate data from a local graph, stored in a graph database, with global graphs. Different from related work, our local data repository is not a static set of documents or tags to be enriched, but a dynamic graph database. The annotated content evolves along the time, bringing challenges, addressed in this research, as how to manage this hybrid graph (local and global) maintaining its consistency during the evolution.

The remainder of the paper is organized as follows. Section 4.2 presents related work. Section 4.3 details our ReGraph framework. Section 4.4 presents our annotation-based approach to enrich data using ontologies. Section 4.5 presents our conclusions and future work.

4.2 Related Work

There are several contexts in which annotations are related to the Semantic Web resources (LOD and ontologies). The annotations are produced manually, semi-automatically or automatically, helping the improvement of information retrieval, knowledge reuse and information exchange [28]. There are works proposing annotations over wiki pages [27] and publishing personal notes as linked data in semantic blogs [17].

Several initiatives focus on how to reach semantic concepts to relate them to resources. In a survey of semantic search approaches, the authors present an overview and a classification of the existing methods for searching and browsing linked data and ontologies [25]. In Alm et al. [1], the authors propose a model to extract characteristic features from semantic annotations by importing the ontology concepts and their taxonomic relationships. Another work uses taxonomic distance measures to compute relatedness of the ontological annotations [30].

The work presented in Santos et al. [33] proposes an architecture to discover information sources through the use of semantic search techniques in a corporative metadata repository. The process begins with an initial keyword list, followed by the query reformulation process that expands this list, adding semantically related terms and creating a new query to run on semantic annotations.

In Amanqui et al. [2], the authors developed a semantic search application that uses semantic web key concepts for information retrieval. They have proposed an architecture for semantic search that maps concepts of the OntoBio domain ontology to a database from the National Institute for Amazonian Research (INPA), which has collections of insects, fishes, and mammals, totalizing over 16,500 species.

As mentioned before, this work differs from the above initiatives since it introduces a graph database perspective over the locally annotated data, which dynamically evolve along the time and must stay consistent.

4.3 ReGraph

As mentioned before, this method is an extension feature in our ReGraph framework, which provides a bridge integrating relational and graph databases, keeping both synchronized in their native representations. In this section, we briefly explain how the ReGraph framework works and the data conversion process from a relational to a property graph database.

4.3.1 The ReGraph Framework

The FishBase data is stored in a relational database. Besides the existing relational database, ReGraph produces a parallel property graph database (FishGraph), to perform network analyses and to link data with Semantic Web.

Starting from a relational database, ReGraph allows mapping its data into a property graph database, generating a mapped subgraph. It is also possible to further create manual and automatic annotations over these data, generating an annotation subgraph.

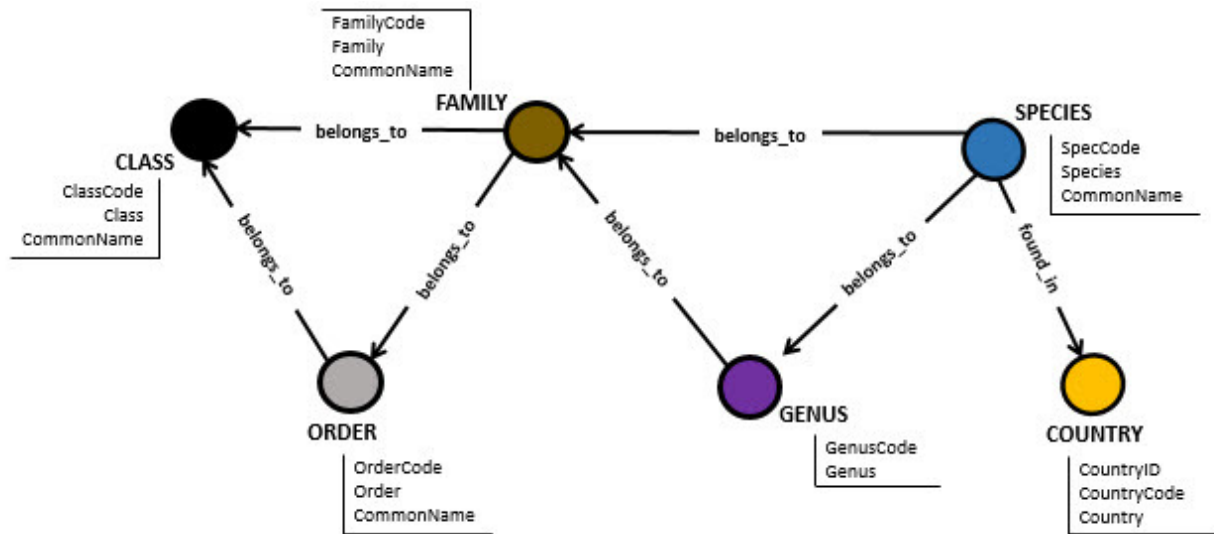


Figure 4.1: Graph Model for Taxonomic Classification and Countries.

Both subgraphs, mapped and annotation, are connected in the graph database. ReGraph keeps relational and graph databases in their native forms and has a synchronism module that reflects in the graph database changes executed in the relational database. The graph database is focused on the analysis on the relations among data elements.

4.3.2 From FishBase to FishGraph using the ReGraph framework

As previously mentioned, FishGraph concerns an application of ReGraph in the FishBase information system. We have mapped the taxonomic classification of fishes from FishBase to FishGraph - see details in [12]. The taxonomic classification of a species includes: Kingdom, Phylum, Class, Order, Family, Genus and Species. As FishBase has only species of fishes, it does not register Kingdom and Phylum, once that all fishes belong to the same Kingdom and Phylum. These data was compared to the taxonomic classification defined in DBpedia, generating a comparison annotation type.

In order to generate a new annotation type, we have selected also the table Country, representing countries where species are found. Figure 4.1 shows the graph model for the taxonomic classification and country data generated in the graph database, in which we have nodes and, associated with them, their respective properties and edges connecting them to each other.

We used the country information in the graph database to link them to GeoNames, a geographical knowledge base that covers all countries and contains over eight million placenames. Data retrieved from GeoNames generated new nodes and edges in the graph database, enriching it and bringing more details to the performed analyses. After the migration of the related data, we generated in the graph database 226,284 edges and 44,701 nodes, in which we have: 311 countries¹; 32,957 species; 10,790 genera; 572 families; 65 orders and 6 classes.

¹Islands are also counted as countries in FishBase.

4.4 Automatic Annotation-Based Method

Annotations can improve the understanding and the quality of the data adding extra information. We propose a method that allows creating automatic annotations over the existent data in a property graph database. These annotations will be created through a direct connection with existing ontologies and LOD, available on the Web, e.g., GeneOntology, GeoNames and DBpedia. In this section, we detail our automatic annotation-based method and the two distinct annotation types implemented: Comparison and New. Independently of the annotation type, local data is related to Web data through a match function that compares strings to find the proper resource.

A distinctive feature of our approach is to differentiate the annotation subgraph (produced here) from the mapped subgraph (mapped from the relational database). The mapped subgraph cannot be directly changed in the graph database, since it is the product of a one-way synchronization originated in the relational database. Synchronization rules avoid updates in the mapped subgraph that will create inconsistencies with the annotation subgraph.

4.4.1 The Comparison Annotation Type

The main goal in the Comparison annotation type is to record comparisons of data stored in the local graph database with third party sources available on the Web. To execute this type of automatic annotation, it is necessary to define the "subject query" that will return the data from the property graph database that will be subject to the comparison.

The order of the data returned by the subject query is determinant to the correct execution of the process: (i) the first value will be the identifier of the node, helping the annotation process; (ii) the second value will be the key matched with the ontology identifiers; it will be used by the match function to retrieve data on the Web; (iii) for each of the remaining values, it is necessary specify the direct path in the ontology to reach it, linking the returned values with the specific value in the ontology; it is possible to define two paths in the ontology for each value returned by the subject query.

The result of this comparison will produce an annotation over the first node returned by the subject query. This annotation is added in the graph database as a property of the node, in which there are three possible values, annotated automatically:

- **Equal:** indicate elements that have the same value in the graph database and in the external ontology. This kind of annotation can improve the quality and the confidence of the data, through a double check validation;
- **Not Found:** represent existing elements in the graph database that were not found in the referred ontology. It can indicate: data in the graph database have spelling mistakes; the specified data do not exist in the referred ontology; data were updated in one of the sources, and were not in the other; etc;
- **Divergent:** represent data that have a divergence compared to the referred ontology. It can indicate: incorrect data in the graph database or in the ontology. This value is defined as a recommendation to review data. In addition, a new node is

added, linked with the existing node, containing the exact data in the ontology for traceability.

4.4.2 The New Annotation Type

In the New annotation type, we produce new nodes, edges and/or properties, to improve the analysis and results. In this annotation type, it is necessary to specify in the "subject query" only two values: (i) the first one will be the identifier of the node, helping in the annotation process; (ii) the second one represents the key in the graph database matched with the respective identifier of a resource in the ontology; it is used by the match function to retrieve data on the Web. The second step is to define the ontology path to search.

Both data are the starting point to search in the ontology. For each information to be retrieved from the ontology and inserted in the graph database it is necessary specify: (i) ontology information: direct path in the ontology to retrieve the required information; (ii) annotation creation: how the annotation will be created in the graph database: as a node or property. The new node will be connected with the existing node by an edge that has its label also defined. In the property option, a defined property will be created over the existing node. In both cases, the value of the property will be the value found in the specified ontology.

4.5 Conclusions and Future Work

In this paper, we presented an automatic annotation-based method using ontologies, as an extension of our project ReGraph that connects a relational database with a property graph database, keeping both integrated, synchronized and in their native forms. It stands out for its flexibility in defining the ontologies and values that will be retrieved, compared and created, offering several possibilities to validate and enrich the graph database. Our method contrasts with the related work since it introduces a graph database perspective over the annotation-based connection between the local and global graphs. Annotations in the annotated subgraph stay consistent with the existing mapped subgraph, even after its evolution along the time.

We developed two distinct experiments to validate each proposed annotation type: Comparison and New². In the Comparison experiment, we compared almost 33,000 species of fishes from FishBase to validate their taxonomic classification with DBpedia. In the New experiment, we used the 249 countries³ in the graph database to retrieve their continent and information of GeonNameID and population from GeoNames.

Future work includes extending the functionality of ReGraph to allow retrieving data from other web formats and to save the link to the resource in the graph database as well as the "subject query" that generated it, helping in future repeated analysis and to track provenance.

²See Appendix C for details of these experiments.

³Islands are also counted as countries in FishBase.

Chapter 5

Conclusion

The relational database model is one of the most used to store, manage and retrieve data. However, network analyses in this model can be a hard and complex task. In addition, the rigid database schema does not allow to easily add new fields and/or relations in the current schema. In recent years, the necessity of constantly changing the schema and of performing network analysis are getting increasing attention and providing a new view over the existing data. The challenge is to create a network-driven data representation which coexists with the relational database, with a minimum impact in the current infrastructure.

The main contribution presented in this dissertation is ReGraph, a framework to map data from a relational to a graph database, providing a hybrid architecture that bridges both databases and managing the coexistence of them in their native representations. The proposed framework maintains the graph database synchronized with the relational database and allows the creation of annotations directly in the graph in manual or automatic ways. Automatic annotations can be generated from the Semantic Web. Related work, does not fulfill all these requirements.

In order to validate our proposal, we developed the ReGraph framework, generating FishGraph, a graph database obtained from FishBase, and we performed different network analyses in FishGraph. ReGraph was tested only using FishBase, but the framework allows to convert any MySQL¹ relational database to a graph database. To implement the diverse aspects of the framework, we employed a stack of technologies. Data were obtained from the MySQL relational database. To store the data as a graph, we adopted Neo4J². The processing module uses Java and Cypher languages. Finally, the Web user interface was implemented using the bootstrap³ framework and the jQuery⁴ library.

¹<https://www.mysql.com/> [Accessed: 2016-Jan-05]

²<http://neo4j.com> [Accessed: 2016-Jan-05]

³getbootstrap.com/ [Accessed: 2016-Jan-05]

⁴<https://jquery.com/> [Accessed: 2016-Jan-05]

Future Work

There are several areas in which we want to improve the proposal. The main ones are:

- **Annotation management:** the proposed framework allows creating any type of annotation without extra validation. It is necessary to avoid the creation of annotations that duplicate existing data in the relational source since it can generate inconsistencies in the data and in the synchronization process;
- **Multiple mappings:** one relational database could be mapped into many distinct graph models. A challenge is to provide the ability to generate distinct graph models departing from one relational database model;
- **Multiple and diverse relational databases:** the current framework can map only MySQL databases and does not allow more than one database source. It will be important to support multiple and diverse relational database sources in an integrated graph database. In addition, it will be necessary to consider the management of similar nodes, classified as “same as”, in the graph database;
- **Mapping:** even using ReGraph, manual mapping still requires technical knowledge from the final users. Make this step easier will popularize the use of the framework;
- **Graph algorithms:** a challenge is how to provide a library of predefined standard graph algorithms and how to map each of them to the users domain problems, generating valuable network analyses;
- **Visualization:** visualization of graphs with millions of nodes and edges is, currently, a big challenge in this research scenario. The improvement of the data visualization support will enrich the user experience allowing the dissemination of the ReGraph framework;
- **Impact and performance:** it is necessary to measure the real impact that the adoption of ReGraph will generate in the current infrastructure. Moreover, it is important to execute performance tests, comparing queries executed in the graph and in the relational database to identify the advantages of each model to specific queries.

Bibliography

- [1] Rebekka Alm, Dagmar Waltemath, Olaf Wolkenauer, and Ron Henkel. Annotation-Based Feature Extraction from Sets of SBML Models. In *10th International Conference on Data Integration in the Life Sciences*, DILS 2014, pages 81–95. 2014.
- [2] Flor K. Amanqui, Kleber J. Serique, Franco Lamping, Andréa C. F. Albuquerque, José Laurindo Campos dos Santos, and Dilvan A. Moreira. Semantic Search Architecture for Retrieving Information in Biodiversity Repositories. In *Proceedings of the 6th Seminar on Ontology Research in Brazil, Belo Horizonte, Brazil, September 23, 2013*, pages 83–93, 2013.
- [3] Renzo Angles. A Comparison of Current Graph Database Models. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering Workshops*, ICDEW '12, pages 171–177, 2012.
- [4] Renzo Angles and Claudio Gutierrez. Survey of Graph Database Models. *ACM Computing Surveys*, 40(1):1–39, 2008.
- [5] Tim Berners-Lee. Relational Databases on the Semantic Web. <http://www.w3.org/DesignIssues/RDB-RDF.html>, 1998 [Accessed: 2014-June-10].
- [6] Tim Berners-Lee. Giant Global Graph. <http://dig.csail.mit.edu/breadcrumbs/node/215>, 2007 [Accessed: 2015-July-10].
- [7] Christian Bizer. D2R Map – A Database to RDF Mapping Language. *12th International World Wide Web Conference*, pages 4–6, 2003.
- [8] Christian Bizer, Tom Heath, and Berners-Lee. Linked Data – The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.
- [9] C. Blakeley. Virtuoso RDF Views – Getting Started Guide. http://www.openlinksw.co.uk/virtuoso/Whitepapers/pdf/Virtuoso_SQL_to_RDF_Mapping.pdf, 2007 [Accessed: 2014-June-10].
- [10] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. Elsevier, 1976.
- [11] 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. In *Proceedings of the Fifth International Workshop on Research Issues*

- in Data Engineering, 1995: Distributed Object Management*, RIDE-DOM '95, pages 124–131, 1995.
- [12] Patrícia Cavoto, Victor Cardoso, Régine Vignes-Lebbe, and André Santanchè. Fish-Graph: A Network-Driven Data Analysis. In *11th IEEE International Conference on e-Science, e-Science 2015, Munich, Germany, August 31 - September 4*, pages 177–186, 2015.
- [13] Patrícia Cavoto and André Santanchè. Annotation-Based Method for Linking Local and Global Knowledge Graphs. In *Proceedings of the Brazilian Seminar on Ontologies (ONTOBRAS 2015), São Paulo, Brazil, September 8-11, 2015*.
- [14] Patrícia Cavoto and André Santanchè. ReGraph: Bridging Relational and Graph Databases. In *Proceedings of the 30th Brazilian Symposium on Databases 2015 (SBBD, 2015), Demos and Applications Session*, 2015.
- [15] Patrícia Raia Nogueira Cavoto and André Santanchè. Arquitetura Híbrida de Integração entre Banco de Dados Relacional e de Grafos. In *Proceedings of the 29th Brazilian Symposium on Databases 2014 (SBBD, 2014), Workshop on Thesis and Dissertations in Databases (WTDD, 2014)*, 2014.
- [16] 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. In *Proceedings of the 2010 International Conference on Web-Age Information Management*, Lecture Notes in Computer Science, pages 37–48, 2010.
- [17] Laura Dragan, Alexandre Passant, Siegfried Handschuh, and Tudor Groza. Publishing Semantic Personal Notes as Linked Data. *CEUR Workshop Proceedings*, pages 1–2, 2010.
- [18] FishBase Consortium. FishBase. <http://www.fishbase.org/>, 2015 [Accessed: 2015-July-05].
- [19] R. Froese and D. Pauly. *FishBase 2000: Concepts, Design and Data Sources*. ICLARM, 2000.
- [20] Robert N. Goldberg and Gregory A. Jirak. Relational Database Management System and Method for Storing, Retrieving and Modifying Directed Graph Data Structures, *United States Patents: US 5201046 A*. <http://www.google.com/patents/US5201046>, 1993.
- [21] Alejandro Gutiérrez, Philippe Pucheral, Hermann Steffen, and Jean-Marc Thévenin. Database Graph Views: A Practical Model to Manage Persistent Graphs. In *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB*, pages 391–402, 1994.
- [22] Christian Theil Have and Lars Juhl Jensen. Are graph databases ready for bioinformatics? *Bioinformatics (Oxford, England)*, pages 3107–3108, 2013.

- [23] IBM. DB2 NoSQL Support: DB2 RDF Store. <http://www-01.ibm.com/software/data/db2/linux-unix-windows/nosql-support.html>, 2014 [Accessed: 2014-February-26].
- [24] Martin Kleppmann. Should you go Beyond Relational Databases? <http://blog.teamtreehouse.com/should-you-go-beyond-relational-databases>, 2009 [Accessed: 2015-March-23].
- [25] Christoph Mangold. A Survey and Classification of Semantic Search Approaches. *International Journal of Metadata, Semantics and Ontologies*, pages 23–34, 2007.
- [26] Oracle Corporation. Oracle Spatial and Graph Developer’s Guide. <http://docs.oracle.com/cd/E16655{ }01/appdev.121/e17896.pdf>, 2013 [Accessed: 2014-February-26].
- [27] Eyal Oren, Renaud Delbru, Knud Möller, Max Völkel, and Siegfried Handschuh. Annotation and Navigation in Semantic Wikis? *CEUR Workshop Proceedings*, pages 58–73, 2006.
- [28] Eyal Oren, Knud Hinnerk Möller, Simon Scerri, Siegfried Handschuh, and Michael Sintek. What are Semantic Annotations? <http://www.siegfried-handschuh.net/pub/2006/whatissemannot2006.pdf>, 2006 [Accessed: 2015-July-23].
- [29] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society. *Nature*, 435(7043):814–818, 2005.
- [30] Guillermo Palma, Maria-Esther Vidal, Louiqa Raschid, and Andreas Thor. Exploiting Semantics from Ontologies and Shared Annotations to Partition Linked Data. *10th International Conference on Data Integration in the Life Sciences, DILS 2014*, pages 120–127, 2014.
- [31] Sriram Raghavan and Hector Garcia-Molina. Integrating Diverse Information Management Systems: A Brief Survey. Technical report, Stanford InfoLab, 2001.
- [32] Satya S Sahoo, Wolfgang Halb, Sebastian Hellmann, Kingsley Idehen, Ted Thibodeau Jr, Sören Auer, Juan Sequeda, and Ahmed Ezzat. A survey of current approaches for mapping of relational databases to RDF. *W3C RDB2RDF Incubator Group January*, 2009.
- [33] Veronica Dos Santos, Fernanda Araujo Baiao, and Asterio Tanaka. An Architecture to Support Information Sources Discovery Through Semantic Search. *2011 IEEE International Conference on Information Reuse & Integration*, pages 276–282, 2011.
- [34] Robert Tarjan. Depth-First Search and Linear Graph Algorithms. *12th Annual Symposium on Switching and Automata Theory (SWAT 1971)*, 1(2):146–160, 1971.

- [35] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. A comparison of a graph database and a relational database: a data provenance perspective. In *Proceedings of the 48th Annual Southeast Regional Conference*, ACM SE '10, pages 1–6, 2010.
- [36] Marc Volz, Karl Aberer, K. Böhm, and D. GMD-IPSI. An OODBMS-IRS coupling for structured documents. *IEEE International Conference on Data Engineering*, 0:34–42, 1996.
- [37] A. D. Vries and A. Wilschut. On the integration of ir and databases. In *Database Issues in Multimedia; Short Paper Proceedings, International Conference on Database Semantics (DS-8)*, pages 16–31, 1999.

Appendix A

Identification Key Visual Example

An identification key in FishBase is listed as a set of questions to be answered in the proposed order, following a path that leads to the species, family or order of the analyzed specimen. Figure A.1 shows part of the identification key 799 and its questions, obtained from the FishBase website¹.

Key to the Teleostean families from East Africa (sub-order <i>Trachinoidei</i>).				
Matthes, H. 1975. Key for the identification of the east African marine fishes to family level. Afr. J. Trop. Hydrobiol. Fish 4(1):93-120. Ref No [48587] Key No. [799]				
Note: n = 14				
Couplet	Character	Next	Prev	Link
1 a	One continuous dorsal fin.	2	(1)	
1 b	2 dorsal fins or dorsal fins clearly separated into 3 parts.	4	(1)	
2 a	Spines present in dorsals, sometimes feeble, pelvics present.	3	(1)	
2 b	No spines in dorsal or anal fin; eel-like; pelvics absent.	-	(1)	Apodocreeidia, Creediidae
3 a	Mouth extending beyond eye, with elongate maxilla; caudal fin rounded to subtruncate.	-	(2)	Opistognathus, Opistognathidae
3 b	Mouth reaching eye, lower jaw projecting; caudal fin pointed; first 2-3 dorsal rays filamentous, free.	-	(2)	Trichonotidae
4 a	Ventrals (=Pelvics) thoracic.	5	(1)	
4 b	Ventrals jugular.	6	(1)	
5 a	Mouth extending to eye level.	-	(4)	Parapercis, Pinguipedidae
5 b	Mouth huge, extending well past eye.	-	(4)	Chiasmodontidae
6 a	Pectorals reaching level of vent or beyond; mouth not extending past eye, either horizontal or nearly vertical.	7	(4)	
6 b	Pectorals not reaching level of vent; mouth oblique, extending past eye.	-	(4)	Champsodon, Champsodontidae
7 a	Eye large, snout ± pointed; mouth horizontal.	-	(6)	Bembrops, Percophidae
7 b	Eye small; snout wide, truncate; mouth vertical.	-	(6)	Uranoscopidae

Figure A.1: Part of identification key 799 of teleostean families from East Africa (sub-order *Trachinoidei*).

This set of questions can be organized in a decision tree data structure, in which the root is the identification key itself and each question is a node in the path. The leafs in this tree are species, families or orders. Figure A.2 shows the same identification key 799 from FishBase represented by a tree data structure.

¹<http://fishbase.org/keys/allkeys.php> [Accessed: 2016-Jan-05]

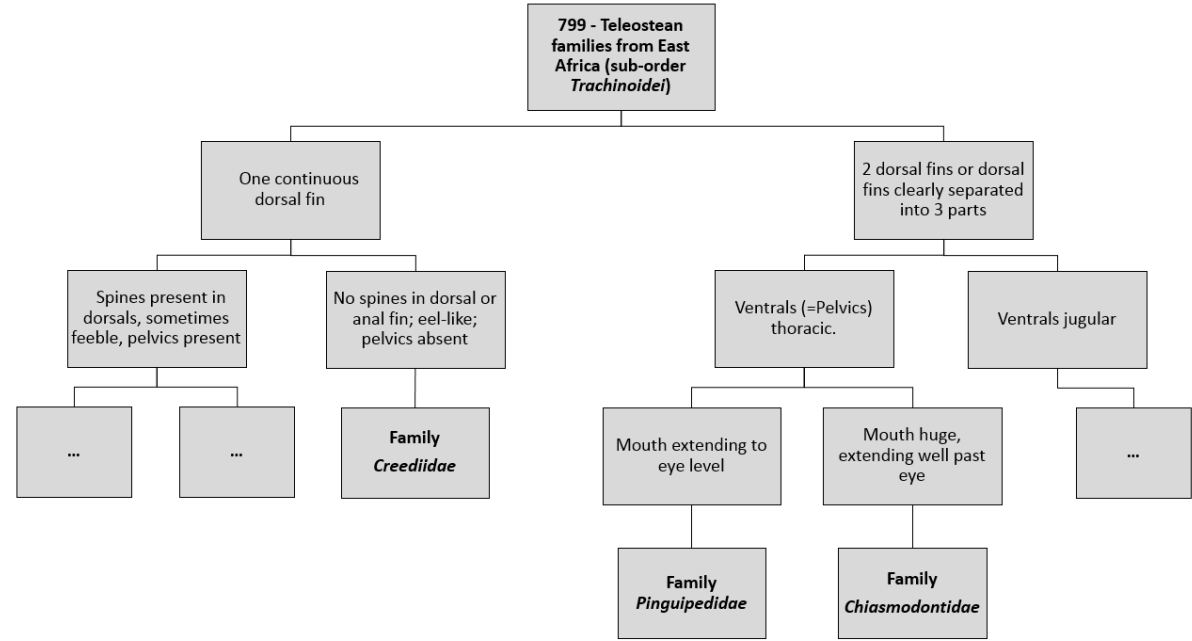


Figure A.2: Part of identification key 799 of teleostean families from East Africa (sub-order *Trachinoidei*) represented by a tree structure.

Appendix B

Synchronization Process

As mentioned in Chapter 3, Section 3.2.1, at the end of the ETL process, ReGraph generates triggers in the relational database over each of the mapped tables and fields. These triggers are responsible for any insert, update and/or delete over the mapped data. Once we have all the existent data in the graph database, the synchronization process is ready to start.

Figure B.1 shows the detailed flow of the synchronization, an asynchronous process. Step 1 happens when any user inserts, updates and/or deletes some mapped data in the relational database (flow 1). Triggers over these data record the event and the information in the Notification Table, a specific control table generated by the ReGraph framework (flow 2). These inserted data have its column status defined as "pending".

The ReGraph scheduled Sync Service reads the information in the Notification Table of those entries with the value in the column status different from "ok" and reads the mapping stored in the graph database (flows 3 and 4). With all the required information, the sync service performs the updates in the graph database (flow 5), following the rules detailed in Chapter 3, Section 3.2.3.

The last step of this process (flow 6) involves updating the status column of each of the entries in the Notification Table with two possible values:

- **ok:** if the update was successful, the status column receives the value "ok";
- **error:** if some problem occurs in the update step, the status column receives the value "error" and another column, observation, receives the description of the generated system error. Records with the status column "error" are always re-processed by the sync service.

The Sync Service runs according to the time interval defined by the user.

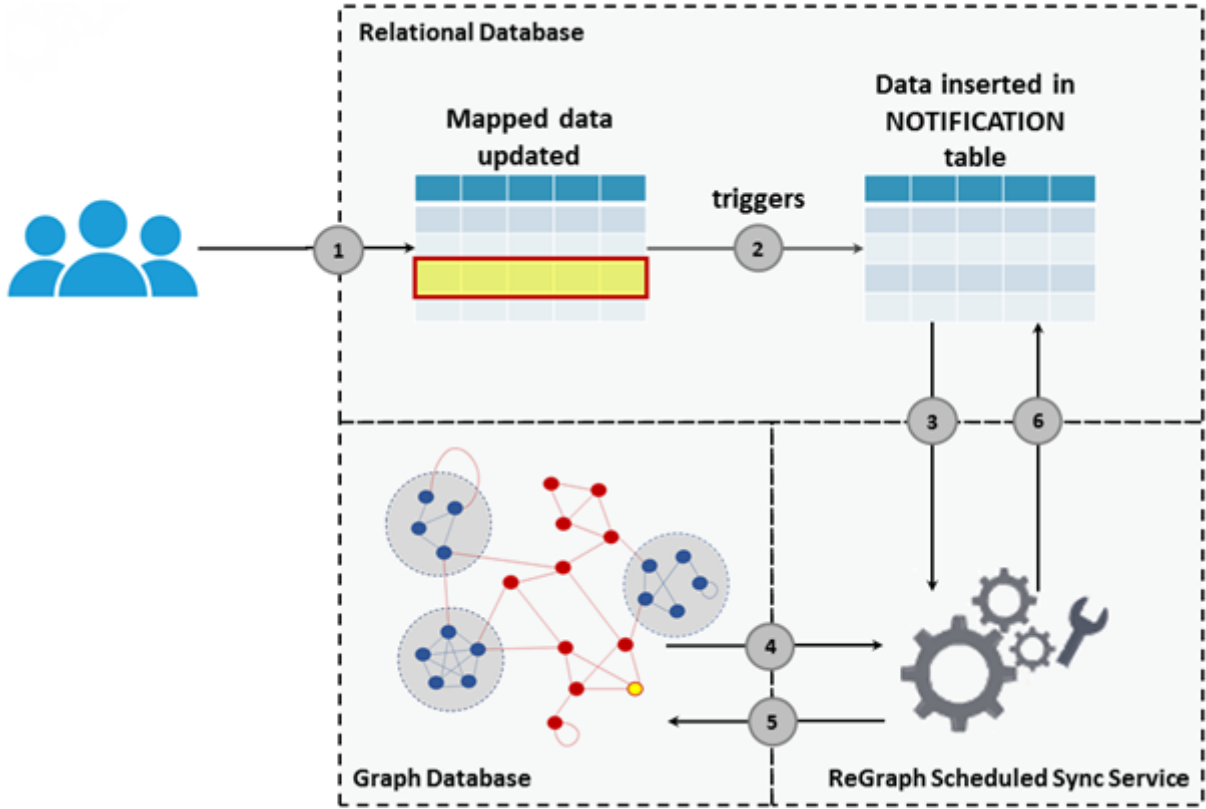


Figure B.1: Synchronization process detailed flow.

Appendix C

Annotation-Based Method: Experiments and Results

In Chapter 2, we presented a preliminary test result of an ad hoc routine to link data between FishGraph (available in a property graph model) and DBpedia, which was the basis to further generalize the solution and create the method detailed in Chapter 4. The previous preliminary routine, detailed in Chapter 2 was not generic and customizable and only compares data and presents results in the console, without producing annotations. Using our proposed annotation-based method, we extended the previous scenario producing a generic and customizable module, which allows defining the ontology, linking data with the Web resource and using annotations to improve local data and queries. More specifically, we added the following new features:

- **Ontology definition:** the definition of the ontology became flexible, allowing users to determine the ontology that will be used in the analysis. To access the ontology, it is necessary to inform the API that returns its content. In the current version, it must be a JSON (JavaScript Object Notation) format, a light and common standard highly used to exchange data on the Web;
- **Automatic annotations:** these annotations provide filters to the queries executed in the database, making easier to a specialist to validate the data and/or enrich the local graph;
- **Process generalization:** the process now can be fully customized through "subject queries", that defines which data from the graph will be used, and the specification of data to be added from the ontology into the graph.

C.1 "Comparison" Annotation: Taxonomic Classification Scenario

In Chapter 2, we developed an initial experiment integrating data from FishGraph with DBpedia to compare the taxonomic classification of the species. Starting from the Species node, we traversed FishGraph to get Genus, Order, Family and Class. Later, using text

ReGraph Mapping Graph

Automatic Annotation Run

Annotation Type:
 Comparison

Cypher Query:

```
MATCH (s:Species)-[r1]->(g:Genus)-[r2]->(f:Family)-[r3]->(o:Order)-[r4]->(c:Class)
RETURN s.SpecCode AS SpecCode, s.Species AS Key, s.Species AS Species, g.Genus AS Genus, f.Family AS Family, o.Order AS Order, c.Class AS Class
```

Ontology API:
 http://dbpedia.org/data/KEY_VALUE.json?jsoncallback=

Analysis Name:
 DBpediaTaxonomy

Related Values:

- http://dbpedia.org/resource/KEY_NAME
- http://dbpedia.org/ontology/genus
- http://dbpedia.org/ontology/family
- http://dbpedia.org/ontology/order
- http://dbpedia.org/ontology/class

- http://dbpedia.org/property/genus
- http://dbpedia.org/property/genus
- http://dbpedia.org/property/family
- http://dbpedia.org/property/order
- http://dbpedia.org/property/Class

Figure C.1: ReGraph Screen of Automatic "Comparison" Annotation.

labels, we defined the resources on DBpedia and we have retrieved the related information, using a DBpedia API. The final results obtained by our comparison showed that from the 32,957 species of fishes: 5,136 species have the same taxonomic classification in FishGraph and DBpedia (15.18%); 7,456 species have some divergence in genus, family, order or class (22.62%); and 20,365 species exist only in FishGraph (61.79%).

The first step of the process is to set that this analysis will be a "Comparison" annotation type. Figure C.1 shows the ReGraph screen with the specification to perform the taxonomic classification comparison using data from FishGraph and DBpedia.

We further need to define the subject query to retrieve the related data. As Neo4J uses the Cypher language to manipulate data in the graph, we defined the subject query to retrieve the required taxonomic classification using the direct hierarchy path as follows:

```
MATCH (s:Species)-[r1]->(g:Genus)-[r2]->(f:Family)-[r3]->
(o:Order)-[r4]->(c:Class)
RETURN s.SpecCode AS SpecCode, s.Species AS Key, s.Species AS
Species, g.Genus AS Genus, f.Family AS Family, o.Order AS
Order, c.Class AS Class
```

The first value returned by the subject query – SpecCode – is the unique identifier of the node that will be annotated at the end of the process.

The second value – Species – will be the key to search this resource in the referred Ontology. It is necessary to define the API from the ontology that will be used in the comparison with this value. In this experiment, we used the API from DBpedia: http://dbpedia.org/data/KEY_VALUE.json?jsoncallback=?. We used the reserved word KEY_VALUE in the specification. It is meant to be replaced with the Species value returned by the subject query to conduct the search in DBpedia. The following step is

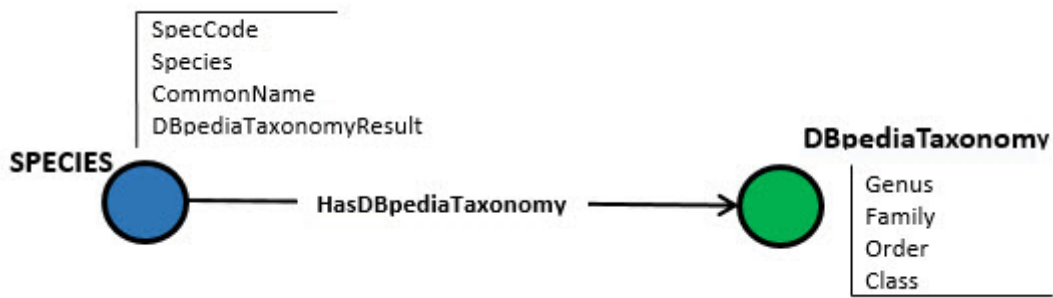


Figure C.2: Node “Species” after the Comparison Annotation Process with DBpedia.

to determine the name of the analysis, defined as DBpediaTaxonomy, in our case.

For each of the remaining values it is necessary to define the location of the resource in the Ontology. In DBpedia we can find the same resource in different places and, to improve our results, we allow the specification of two distinct locations for each of the values returned by the subject query. The resources definitions were:

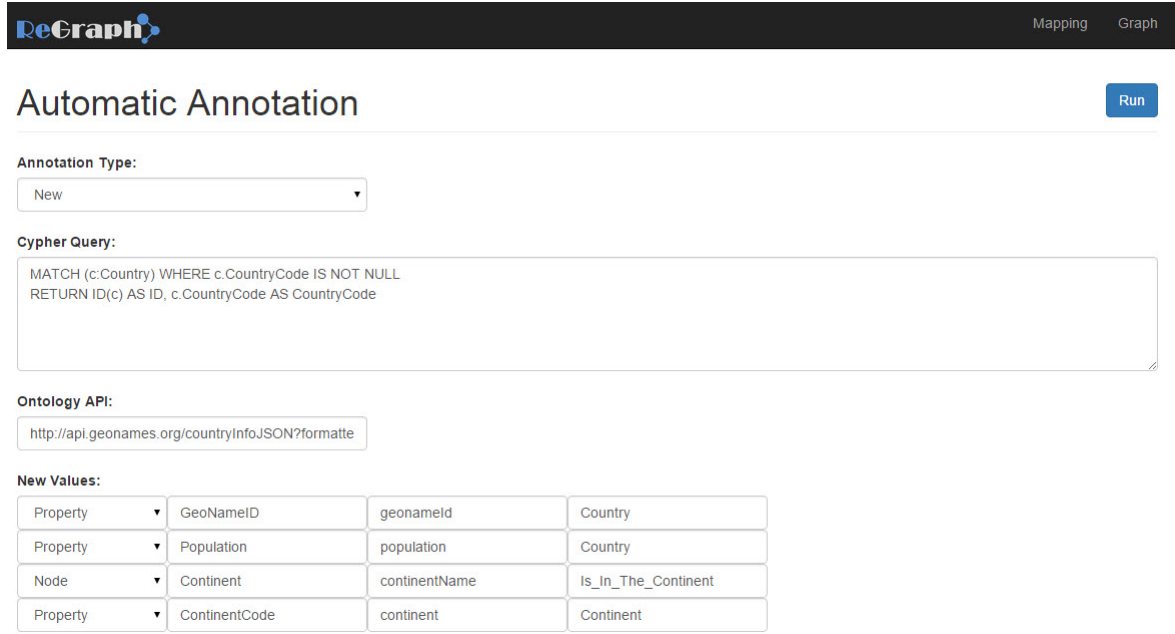
- Species: http://dbpedia.org/resource/KEY_NAME. Again, we need to use the reserved value KEY_NAME, that will be replaced with the Species value returned by the subject query;
- Genus: <http://dbpedia.org/ontology/genus> and <http://dbpedia.org/property/genus>;
- Family: <http://dbpedia.org/ontology/family> 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>.

At the end of this process, we added the property DBpediaTaxonomyResult in each of the Species nodes, with one of the three possible values: Equal, Not Found or Divergent. "Equal" nodes have the same taxonomic classification in FishGraph and DBpedia. "Not found" nodes represents species existent only in FishGraph. For nodes with the value "Divergent", we have generated a new node with the label: DBpediaTaxonomy and properties Genus, Family, Class and Order with the values found in DBpedia. The new node is connected with the original one by the edge HasDBpediaTaxonomy. Figure C.2 shows the Species node after the "Comparison" annotation process. The DBpediaTaxonomy node is applied only for nodes of Species with Divergent value in the DBpediaTaxonomyResult property.

C.2 "New" Annotation: Country Scenario

GeoNames provides a set of free web services¹ with several data from countries. It is necessary to create an account to access these web services and retrieve the required in-

¹<http://www.geonames.org/export/ws-overview.html> [Accessed: 2016-Jan-05]



Annotation Type:

New

Cypher Query:

```
MATCH (c:Country) WHERE c.CountryCode IS NOT NULL
RETURN ID(c) AS ID, c.CountryCode AS CountryCode
```

Ontology API:

<http://api.geonames.org/countryInfoJSON?formatted>

New Values:

Property	GeoNameID	geonameid	Country
Property	Population	population	Country
Node	Continent	continentName	Is_In_The_Continent
Property	ContinentCode	continent	Continent

Figure C.3: ReGraph Automatic "New" Annotation screen for the country node.

formation. Departing from the CountryCode property, our goal is to retrieve information related to the country and improve our local database, e.g., GeoNameID, population and the continent that it belongs. The first step to reach this goal is to define the annotation type as "New". Figure C.3 shows the ReGraph screen with the specification to perform the "New" annotation type for retrieving information about the countries from GeoNames.

In the "New" annotation type only two values are required: the identifier and the key that links the local data with the ontology. The subject query to retrieve country information from Neo4J was defined as follows:

```
MATCH (c:Country)
WHERE c.CountryCode IS NOT NULL
RETURN ID(c) AS ID, c.CountryCode AS CountryCode
```

FishBase has some places registered as countries, but they do not have a country code. In order to avoid mistakes, we excluded these countries in the subject query. We obtained a total number of 249 countries².

The next step is to define the API that will be used to retrieve the information. In this experiment, we used the country info API with the reserved word KEY_VALUE (http://api.geonames.org/countryInfoJSON?formatted=true&lang=it&style=full&country=KEY_VALUE) to obtain the specified data. The reserved word KEY_VALUE will be replaced with the value CountryCode returned by the subject query.

We then define each data element that will be obtained from the API and how this data will be created in the graph database:

- GeoNameID: will produce a property in the Country node called GeoNameID, with

²Islands are also counted as countries in FishBase.



Figure C.4: Node “Country” after the New Annotation Process with GeoNames.

the value retrieved from the geonameId data;

- Population: will produce a property in the Country node called Population, with the value retrieved from the population data;
- Continent: will produce a new node labeled Continent, with a property with the same name. The value for this property will be retrieved from continentName. The edge connecting Country and Continent will be Is_In_The_Continent;
- ContinentCode: will produce a new property in the node Continent called ContinentCode, with the value retrieved from the continent field.

Departing from the 249 existing countries³ in the graph database, only one was not found in GeoNames: AN - Neth Antilles. At the end of this process, we generated the defined properties, nodes and edges. Figure C.4 presents the Country node after the new annotation process with GeoNames.

³Islands are also counted as countries in FishBase.