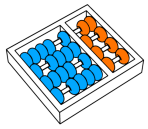
Universidade Estadual de Campinas
Instituto de Computação



Luiz Celso Gomes Júnior

Consulta e Gerenciamento de Redes Complexas

Querying and Managing Complex Networks

CAMPINAS
2015

Luiz Celso Gomes Júnior

Querying and Managing Complex Networks

Consulta e Gerenciamento de Redes Complexas

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

Supervisor/Orientador: Prof. Dr. André Santanchè

Este exemplar corresponde à versão final da Tese defendida por Luiz Celso Gomes Júnior e orientada pelo Prof. Dr. André Santanchè.

CAMPINAS
2015

Agência(s) de fomento e nº(s) de processo(s): FAPESP, 2012/15988-9; CAPES

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

G585q

Gomes Junior, Luiz Celso, 1979-
Querying and managing complex networks / Luiz Celso Gomes Junior. – Campinas, SP : [s.n.], 2015.

Orientador: André Santanchè.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Redes complexas. 2. Banco de dados. I. Santanchè, André,1968-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Consulta e gerenciamento de redes complexas
Palavras-chave em inglês:
Complex networks
Databases
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
André Santanchè [Orientador]
Carmem Satie Hara
Ricardo da Silva Torres
Lucas Francisco Wanner
João Eduardo Ferreira
Data de defesa: 26-10-2015
Programa de Pós-Graduação: Ciência da Computação

Universidade Estadual de Campinas
Instituto de Computação

Luiz Celso Gomes Júnior

Querying and Managing Complex Networks

Consulta e Gerenciamento de Redes Complexas

Banca Examinadora:

- Prof. Dr. André Santanchè
  IC/UNICAMP
- Prof. Dr. João Eduardo Ferreira
  IME/USP
- Profa. Dra. Carmem Satie Hara
  DInf/UFPR
- Prof. Ricardo da Silva Torres
  IC/UNICAMP
- Prof. Dr. Lucas Francisco Wanner
  IC/UNICAMP

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

Campinas, 26 de outubro de 2015

Then the sky spoke to me in language clear,
Familiar as the heart, than love more near.
The sky said to my soul, 'You have what you desire.

'Know now that you are born along with these
Clouds, winds, and stars, and ever-moving seas
And forest dwellers. This your nature is.

Lift up your heart again without fear,
Sleep in the tomb, or breathe the living air,
This world you with the flower and with the tiger share.'

(Passion – Kathleen Raine)

Acknowledgements

There is no easy in climbing the scientific knowledge tree. Indeed, it would be impossible without all the generous hands from people above and all the shoulders from giants of the past. I am humbled by, and grateful for being part of such community. From the early academic steps through the writing of this thesis, I have had the honor of being helped and guided by the most admirable beings one could find. To all of you, my deepest gratitude.

André, the competent, insightful, and supportive advisor, the thoughtful and present friend, the reference for generosity and honesty. Thank you for giving me direction, encouragement, and for being always so patient. Frank, a unique combination of academic excellence and kindness, thank you for all the lessons – I will always take them with me. Bernd, master of both serious and light moments, thank you for providing the nice environment and the guidance that were very important for my growth as a researcher. Luciano, the collaborator that became a mentor, thank you for all the teachings about our incredibly inspiring world. Claudia and Sérgio, earlier supervisors, thank you for making me realize what I wanted to be and for supporting me all along. Members of the thesis committee, thank you for the valuable teachings and feedback.

Friends from LIS, Alessandra, Bruno, Daniel, Eduardo, Ewerton, Fabrício, Fagner, Francisco, Hugo, Ivo, Jacqueline, Jaudete, Joana, Jordi, Leandro, Luana, Lucas, Lucas 2, Márcio, Matheus, Patola, Renato, Sávio, Senra, Victor, thank you for the conversations, ideas, meals and laughs shared. I am proud of being part of such united and talented group. Ive, thank you also for the silly, tender, and inspiring friendship. Friends from Lip6, Anne, Amine, Camélia, Clément, Hubert, Ndiouma, Olivier, Stéphane, Yifan, thank you for sharing so much of your world with me. Nelly and Andres, angels disguised as officemates, I cherish the memories of the time spent among you. Friends from Elabora, especially Léo and João, thank you for exposing me to some of the most interesting technical challenges, strengthening my passion for science and academic research. Longtime friends Alan, André, Andréia, Ayrton, Edna, Fabiano, Gilberto, Hednei, Hélder, Helena, Kaylla, Marcelo, Thiago(s), thanks for being always so present and supportive despite the distance. Friends from GEEU, capoeira, and yoga groups, thank you for making my life so much richer. Helen, little sister, big friend, thank you for being such a especial girl and for helping me grow as a person. The many important friends that I could not name for lack of space, I keep you all in my heart.

Finally, this work would not be possible without the support from the funding agencies[1], the dedication of the Institute of Computing's staff, and the technologies provided by the many competent open/free software projects used throughout the research.

[1]Work partially financed by CAPES and FAPESP [grant numbers 2012/15988-9, 2014/01419-8], FAPESP/Cepid in Computational Engineering and Sciences (2013/08293-7), the Microsoft Research FAPESP Virtual Institute (NavScales project), CNPq (MuZOO 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

Compreender e quantificar as propriedades emergentes de redes naturais e de redes construídas pelo homem, tais como cadeias alimentares, interações sociais e infra-estruturas de transporte é uma tarefa desafiadora. O campo de redes complexas foi desenvolvido para agregar medições, algoritmos e técnicas para lidar com tais tópicos. Embora as pesquisas em redes complexas tenham sido aplicadas com sucesso em várias áreas de atividade humana, ainda há uma falta de infra-estruturas comuns para tarefas rotineiras, especialmente aquelas relacionadas à gestão de dados. Por outro lado, o campo de bancos de dados tem se concentrado em questões de gestão de dados desde o seu início, há várias décadas. Sistemas de banco de dados, no entanto, oferecem suporte reduzido à análise de redes. Para prover um melhor suporte para tarefas de análise de redes complexas, um sistema de banco de dados deve oferecer recursos de consulta e gerenciamento de dados adequados. Esta tese defende uma maior integração entre as áreas e apresenta nossos esforços para atingir este objetivo. Aqui nós descrevemos o Sistema de Gerenciamento de Dados Complexos (CDMS), que permite consultas exploratórias sobre redes complexas através de uma linguagem de consulta declarativa. Os resultados da consulta são classificados com base em medições de rede avaliadas no momento da consulta. Para suportar o processamento de consultas, nós introduzimos a *Beta-álgebra*, que oferece um operador capaz de representar diversas medições típicas de análise de redes complexas. A álgebra oferece oportunidades para otimizações transparentes de consulta baseadas em reescritas, propostas e discutidas aqui. Também introduzimos o mecanismo *mapper* de gestão de relacionamentos, que está integrado à linguagem de consulta. Os mecanismos de consulta e gerenciamento de dados flexíveis propostos são também úteis em cenários além da análise de redes complexas. Nós demonstramos o uso do CDMS em aplicações tais como integração de dados institucionais, recuperação de informação, classificação e recomendação. Todos os aspectos da proposta foram implementadas e testados com dados reais e sintéticos.

Abstract

Understanding and quantifying the emergent properties of natural and man-made networks such as food webs, social interactions, and transportation infrastructures is a challenging task. The *complex networks* field was developed to encompass measurements, algorithms, and techniques to tackle such topics. Although complex networks research has been successfully applied to several areas of human activity, there is still a lack of common infrastructures for routine tasks, especially those related to data management. On the other hand, the databases field has focused on mastering data management issues since its beginnings, several decades ago. Database systems, however, offer limited network analysis capabilities. To enable a better support for complex network analysis tasks, a database system must offer adequate querying and data management capabilities. This thesis advocates for a tighter integration between the areas and presents our efforts towards this goal. Here we describe the Complex Data Management System (CDMS), which enables explorative querying of complex networks through a declarative query language. Query results are ranked based on network measurements assessed at query time. To support query processing, we introduce the *Beta-algebra*, which offers an operator capable of representing diverse measurements typical of complex network analysis. The algebra offers opportunities for transparent query optimization through query rewritings, proposed and discussed here. We also introduce the *mapper* mechanism for relationship management, which is integrated in the query language. The flexible query language and data management mechanisms are useful in scenarios other than complex network analysis. We demonstrate the use of the CDMS in applications such as institutional data integration, information retrieval, classification and recommendation. All aspects of the proposal are implemented and have been tested with real and synthetic data.

Contents

Chapter 1

Introduction

Interactions, explicit or implicit, between elements permeate the universe – from subatomic particles to galaxies interlocked by physical forces, from microorganisms to continental ecosystems through intricate ecological roles, from individuals to societies in the ever-changing characteristics of group dynamics. Since everything in the known universe is subject to the same laws of physics, several patterns arise from these interactions. When those interactions are explicitly represented, commonalities can be identified, the structures can be measured, and their behavior studied. These are among the goals of the research area of Complex Networks, also known as Network Science [22, 51].

The complex network field has developed steadily in the last decades, being applied to most areas of human knowledge, such as physics, biology, medicine, sociology, etc [21]. This development has also been fueled by advances in the largest human-made network: the Internet. The popularity of Google's PageRank algorithm and the ubiquitous online social networks have demonstrated the importance of techniques to derive knowledge from the structure of the networks.

The research community has introduced a wide range of measurements and their respective algorithms that assess diverse properties of the networks. For example, measurements of centrality (e.g. PageRank) identify important nodes, while clustering algorithms distinguish communities inside a network [22]. Given the size and complexity of the networks, deriving these measurements are computationally intensive tasks. Current information systems in this area can be divided in two main classes: (i) desktop-based analysis software, and (ii) distributed graph processing infrastructures.

Analysis software, used by most researchers of complex networks, are typically limited by main memory, have no widely agreed standards for data querying or exchange, and offer no means for data management. On the other hand, graph processing infrastructures have high technical and hardware demands that restricts their use to large companies. Furthermore, the infrastructures focus on algorithms and offer no flexibility in terms of querying and data management.

Although complex network tasks are data-intensive procedures, there is at the present little intersection with the area of Database Systems. Database Management Systems are ubiquitous nowadays, being used in diverse areas and scales, from data analysis in large companies to personal data management in mobile devices. Research in databases, fueled by large investments from the industry, has developed technologies now taken for granted

in aspects such as data querying, integrity, exchange standards, versioning, scalability, etc. Database systems, however, lack the support for network topology analysis which is required by complex network tasks.

The main goal of this research is to approximate the areas of complex networks and databases. Specifically, we propose the Complex Data Management System (CDMS), which is aimed at querying and managing complex networks. The system offers a query language that can express measurements that capture several aspects of the networks. A relational algebra is used to represent elements from the query and the required measurements. The system also offers data manipulation and query optimization capabilities. Our main hypothesis is that such system would allow a new type of interaction with complex networks, offering explorative querying and enabling a wide range of applications.

To validade our hypothesis, we proposed, implemented and tested several aspects related to different layers of a traditional data management system, namely query language, processing model, query optimization, and data management mechanisms. The proposed query language, called *in**, is an extension language that complements existing graph query languages with a ranking clause. The ranking clause is responsible for assessing and combining the complex network measurements. Since graph matching is a well studied topic, our focus for query processing is on the complex network measurements. We proposed two different models as means to encode the measurements: a spreading activation model, and a relational algebra. The thesis describes both models, although we favor the proposed relational algebra, called Beta-algebra, since it is more expressive and inherits several theoretical fundaments and optimization techniques developed in the databases field. We studied and proposed query optimization techniques for both models. As for data management, we introduced the *mapper* mechanism aimed at relationships in the networks.

This thesis is organized as a collection of four papers included as chapters. The second chapter is based on the article "Integrating Databases and Complex Networks" [29], submitted to the journal PLOS ONE. The text argues in favor of a tighter integration between the areas of complex networks and databases. In the article, we present the requirements and the architecture for the CDMS. We also introduce our query model and describe several queries in different task scenarios to demonstrate the usability of our approach. The main contributions described in the article are:

- A compilation of the similarities and differences of the complex networks and databases areas, as well as the advantages of their integration
- The description of the requirements and the architecture of an infrastructure for querying and management of complex networks
- An introduction to our Beta-algebra, using examples typical of complex networks and describing the relational and graph-based interpretations of the Beta-operator
- The demonstration of application scenarios with diverse types of data and queries, using real world and synthetic data

Chapter 3 corresponds to the paper "Beta-Algebra: Towards a Relational Algebra for Graph Analysis" [30], presented at the Fourth International Workshop on Querying Graph Structured Data, GraphQ/EDBT 2015. The paper describes the Beta-algebra in more details, presenting related work and motivating the proposal. The main contributions of the paper are:

- The proposal of the Beta-algebra, an extension to the relational algebra with an operator that can assess complex network measurements
- Suggestions of query optimization techniques in the new setting

Chapter 4 reproduces the article "The Web Within: leveraging Web standards and graph analysis to enable application-level integration of institutional data" [35], published in the journal Transactions on Large Scale Data and Knowledge Centered Systems. The article describes application scenarios for the CDMS that are beyond complex network analysis. The expressivity offered by a querying mechanism that can capture the properties of complex network can also be used in other applications, such as information retrieval, recommendation systems, and classification. In the article we demonstrate how the CDMS can be used in institutional scenarios, being the focus point of diverse tasks. The article also motivates and define the query language that we are using with the CDMS. The in* (in star) language is an extension language that, coupled with existing graph queries, introduces ranking based on a combination of complex network measurements. Being an early article, the text describes query processing based on a spreading activation formalism, which is now replaced by the Beta-algebra. The spreading activation model can be implemented using Beta-algebra, therefore the applications are a subset of the applications allowed by the Beta-algebra. We report query optimization experiments using the spreading activation model in [31]. The main contributions of the article are:

- Application of the proposed framework in other areas such as information retrieval and automatic classification, demonstrating the flexibility of the approach
- The definition of the *in** language that extends existing graph languages with ranking based on complex network measurements
- The proposal of a *spreading activation* model that allows the specification of complex network measurements
- The description of the *mapper* mechanism for relationship creation and manipulation, also integrated in the proposed query language
- Implementation and test of all proposed aspects, using real world and synthetic data

Chapter 5 is based on the paper "Query-based inferences in the Complex Data Management System" [32], presented at the workshop "Structured Learning: Inferring Graphs from Structured and Unstructured Inputs", SLG/ICML 2013. The paper discusses our first steps towards using our framework for machine learning tasks. In our proposal, a query becomes a classification model, with query results listing probable labels. The training phase is equivalent to parameter tuning for the query. The main contributions of the paper are:

- The hypothesis that, given a sufficiently expressive query model, a query can represent a learned model for classification tasks
- Tests of our hypothesis using real data from a nursing diagnosis research [42]. These results are presented in more details in [33].

Finally, Chapter 6 concludes the thesis, presenting final remarks and future work.

Chapter 2

Integrating Databases and Complex Networks

2.1 Introduction

Complex networks research has developed steadily since 1999 to become a well-established landmark in modern science [22, 51]. Inherently inter- and multidisciplinary, complex networks have thrived because of their generality to represent virtually any discrete system in terms of entities and relationships. Though a great deal of the initial investigations in this area were focused on the intricate topological structure of several theoretical and real-world networks, subsequent studies progressively embraced the dynamical aspects of networks, be it related to changes of connectivity along time or to the evolution of states associated to each node or even links.

Investigations in complex networks have frequently aimed at identifying specific topological or dynamical properties of nodes or sets of nodes [21]. For instance, hubs stand out as being particularly important because of their relatively high degree. At the same time, nodes with high betweenness centrality are key intermediators in a network. Yet another interesting structure are the vip nodes, i.e. nodes of relatively low degree which are attached to hubs.

Realizing the potential of deriving non-trivial properties of the networks and the development of related theories and techniques has let to advances in several fields, such as systems biology, neuroscience, communications, transportation, power grids, and economics [21]. Network analysis at this scale is, however, a challenging task. Complex networks research relies on the analysis of big and diverse datasets, employing a range of computationally intensive algorithms. Acquiring, adapting, and selecting data for each specific task are time-consuming processes. Complex networks researchers typically have to resort to *ad-hoc* workflows for each analysis task, leading to inefficiencies related to the development of new programs or scripts, training of personnel, lack of common infrastructures among collaborating groups, etc. The lack of broadly adopted data management infrastructures becomes a bottleneck in face of the ever-increasing demand for complex network analysis in a scenario of data-driven science and society.

This bottleneck manifests itself in underlying technical aspects as well. Since there is

no dedicated data management infrastructure for complex network analysis, researchers often employ toolkits that can only perform operations over data in main memory, severely restricting the maximum size of the dataset or demanding expensive supercomputers to tackle the task. Furthermore, the source data must be copied and changed for each different analysis task, leading to multiple, incompatible and difficult to manage copies of data. Offering access control, versioning, backups, and integrity validation becomes a challenge in this scenario.

The aforementioned shortcomings have been subject of intense research in the database community from the start. Database Management Systems (DBMSs) [23] are ubiquitous nowadays, offering convenient data querying, reliability and scalability. DBMSs – even considering current graph databases – lack, however, native support for tasks typical of network analysis. Furthermore, there is no query language that deals seamlessly with the sorts of emergent properties characteristic of complex networks.

In this work we argue and illustrate that a tighter integration between complex network and databases systems has the potential of greatly changing how network analysis is performed. Beyond providing a more adequate framework for researchers, a query-centric and database-backed interaction would also change the granularity in which researchers can work. By offering streamlined mechanisms for data selection and querying, a database system enables flexible analysis based on subsets of the network. The subsets can be specified in a query language, restricting the nodes, links or regions of interest. The queries can be easily changed for subsequent inquiries, enabling explorative analysis of the networks in shorter rounds of interaction.

This type of interaction would also allow non-experts to perform network analysis in their databases. Most organizations have network data alongside their operating data, and often need to resort to ad-hoc solutions to assess network characteristics. For example, a sales department that needs to analyze its network of customers often requires an expert to extract data from the database, build the network, and analyze it with specific tools. A database system that could perform network analysis in a similar fashion as other institutional systems would likely foster the application of complex network techniques by non-experts.

The objective of this paper is twofold: (i) to lay the case for a deeper integration between the areas of complex networks and databases, and (ii) to present our research efforts towards such integration. In Section 2.2 we contextualize the broad scenario, comparing typical approaches for complex network analysis and database management while introducing important concepts used throughout the paper. We also discuss the advantages of a database-centric approach to complex networks and describe our envisioned analysis workflow. Section 2.3 presents the challenges associated with designing a database system that offers mechanisms adequate for network analysis. Section 2.4 describes the system that we are implementing, the Complex Data Management System (CDMS), emphasizing our current processing and querying models and illustrating the assessment of common network measurements in our approach. Section 2.5 describes use cases for the system, showing how the approach can be used in different scenarios. The first scenario refers to network researchers analyzing food network data (as published in [2]). In the scenario, the researchers want to identify influential cuisines and strong reliance on ingre-

dients. To illustrate the use of network analysis in institutional applications, we created a scenario for restaurant recommendation. The scenario is a POI (point of interest) recommendation application that needs to integrate geographic data with a social network, hierarchical classification, and textual information, all represented in a unified graph. We show how an application can integrate reputation, distance and other measurements in a single database query that is evaluated on demand. Finally, Section 2.6 presents future work and concluding remarks.

2.2 Foundation and Related Work

Researchers in the complex networks field have developed a wide range of models and algorithms to analyze the dynamics of networks. The techniques utilize diverse strategies to derive measurements in the analysis, such as counting paths, matrix factorizations, structural analysis, etc [21]. More recently, researchers have been tackling issues related to networks that change in time and understanding the evolutionary behavior of the underlying processes.

Complex network researches typically rely on a combination of script programs, graph analysis, and mathematic software to perform the computations. The highly multidisciplinary field currently lacks common and standardized infrastructures (information systems) for the diverse data analysis tasks. Our goal is to enable a database-centric interaction for complex network research.

Database systems typically rely on a declarative query language for user interaction [23]. A declarative query language allows users to express what they need (declarative approach) as opposed to having to deal with details concerning how to fetch and process data (imperative approach). Furthermore, when processing the query, the system is free to perform under-the-hood optimizations. A query language enables a different type of interaction between users and data, allowing users to tweak the queries to suit their needs and allowing the system to make decisions in order to answer the request as efficiently as possible.

In our proposal, we aim at bringing the advantages associated with database systems to complex network data. We expect that the flexibility brought by a database-like query language and data management mechanisms will enable new forms of interaction for users analyzing complex networks data.

Complex networks are typically represented as graphs. There are many graph databases and languages available [64, 5]. Although the options available have advantages to offer in terms of data storage and management, they lack query capabilities that can capture the emergent properties of the underlying network. In our proposal, we aim at providing adequate querying mechanisms for complex networks research.

In terms of scalability, there are many frameworks for storage and processing of large graphs. Google's Pregel [48] and GraphLab [46] are examples of graph processing platforms that allow algorithms to run in multiple, distributed computers. The complexities in setting up these environments and learning the underlying models often restrict their

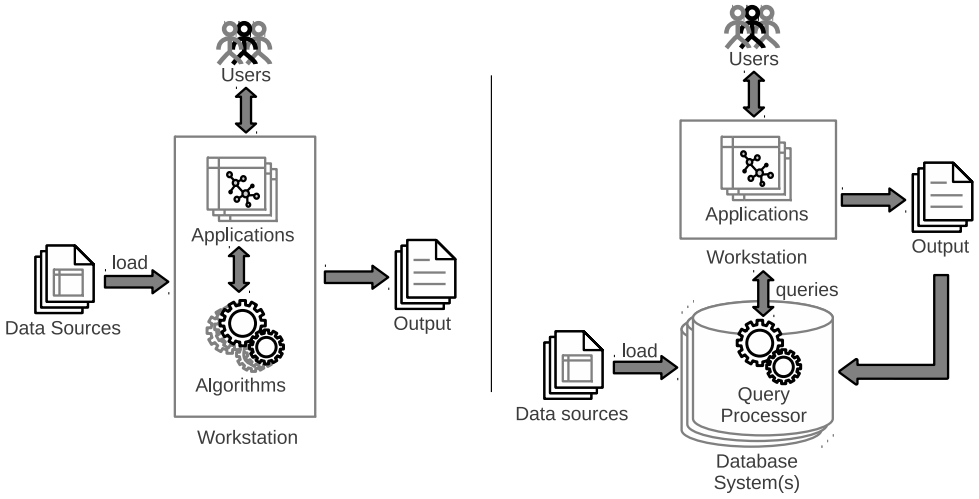


Figure 2.1: Typical scenarios for tasks using standalone software (left) and database-backed tasks (right).

application to large corporations. Distributed graph databases such as Titan[1] are also being developed, supporting a more traditional data management workflow. While these technologies do not aim at providing the type of network querying capabilities presented in this proposal, our framework could be adapted to run over these distributed infrastructures. Furthermore, in our declarative query setting, the users do not need to be aware that the queries are running in a distributed environment.

Visualization is an important aspect for network analysis [38, 53]. Being able to plot information associated with the dynamics of the network is typically an indispensable step in understanding the underlying data. Our proposal does not cover visualization aspects directly. Instead, we employ the same strategy adopted by the databases field, separating data querying and management from visualization issues. In our perspective, visualization software should be a layer on top of the proposed infrastructure.

2.2.1 Comparing network analysis software and database systems

Complex network analysis software and database systems are similar in basic fundamentals: both are data-driven fields, relying on large datasets to derive new information about underlying elements. Although database systems and complex network analysis share several related aspects, they have taken distinct paths in their development.

The main differences between the areas stem from the fact that database systems are general-purpose tools while complex network analysis are based on task-dependent sets of algorithms. Moreover, these differences are at the basis of the successful foundations of the areas. Database systems exploit the commonalities of the data management and query processing procedures to implement under-the-hood optimizations and to define standards for data models and query languages. Complex network analysis, on the other hand, rely on many specific techniques to meet the challenge of capturing the dynamics of the intricate data.

[1] http://thinkaurelius.github.io/titan/

	Complex Networks	Databases
Data	Networks/Graphs	Relational, Text, Graphs
Data management	Ad hoc	Standardized mechanisms for data manipulation and integrity validation
Interface	Ad hoc	Query-based
Basic operation	Assessment of properties based on topological structure of the links	Retrieval and aggregation of data based on attribute values
Data size	Often limited by memory	Highly scalable
System response	Off-line (hours or days)	On-line (seconds)
Interaction cycles	Long (changes in data structure or algorithms are time-consuming)	Short (queries are easy to change and re-execute)
Flexibility	High (any algorithm can be implemented)	Low (limited by query expressiveness)
Learning curve	Low (for each software)	Medium (need to know model and language)
Replicability and reuse	Low (each software uses specific standards and concepts)	High (standardized models and languages)

Table 2.1: Comparison of main characteristics of Database Systems and Complex Network Analysis.

Figure 2.1 shows the schematics of these two paradigms. On the left-hand side there is the typical workflow for data analysis based on ad hoc applications, a common scenario in complex network research. Data are loaded in one or more applications that run specific algorithms to generate results. The right-hand side of the figure depicts the typical workflow for database-backed analysis. Data are loaded in the database system (which can span several servers), users and applications issue queries to the database system, which are processed by the query processor. Query output can be conveniently used to augment the database.

Table 2.1 shows a comparison of diverse aspects of the complex networks and databases fields. Each aspect of the comparison is detailed below, focusing on typical characteristics and scenarios for either field.

Data: Complex networks are naturally represented as graphs, but there is usually no standardized model to serialize the data. Therefore, it is difficult to share data among researchers or use it in distinct applications. Databases, on the other hand, have a long history of data models and exchange standards. The relational model has prevailed over decades as the dominant model for databases. Other, more flexible models, such as semi-structured and graph have become popular recently. Besides standardizing the way data is modeled, database systems also provide mechanisms that validate the integrity of the data.

Interface: Software for complex network analysis typically offer graphical interfaces. Data input and manipulation in the software is handled through visual elements like menus and dialogs. A graphical representation of the network makes analysis intuitive, especially when the software offers flexible layout options. Database systems are built around the concept of declarative queries. They enable data selection and manipulation through commands in a standardized language. While a query mechanism may not be as intuitive at first as a graphical interface, it provides several advantages: tasks are repeatable, composable and easily documented, queries work across systems from different vendors,

the system can optimize query processing without involvement of the user, etc.

Basic operation: Network analysis is heavily based on assessing properties emerging from the interconnection among nodes. Although properties of the nodes can influence network dynamics, the analysis is mostly concerned with concepts such as diffusion over the topology, path traversals, shortest paths, and aggregations thereof. Databases have historically focused on operations (projection, aggregations, etc.) over *attributes* of the data elements (e.g. salary average). Although correlations among elements are often requested in queries (e.g. joins), the number of steps (degree of separation) between the nodes is typically short. This focus on more restricted sets of elements in the analysis makes query processing more predictable and prone to optimizations (although also too restrictive for network analysis tasks).

Data size: The complexity of the interconnections network data makes the algorithms computationally expensive and impose several challenges for memory management and distributed processing. Therefore, most network analysis software constrain data size to the limits of available main memory. This limitation is very restrictive, especially considering that typical complex network tasks have billions of nodes. Database researchers have been concerned with memory management and distribution issues for many decades. Many techniques and algorithms were developed for the simpler data models (e.g. relational) and have provided scalability to the most data-intensive tasks. More recently, these techniques have also been implemented for more flexible models, such as semi-structured and graph, enabling their application in complex network tasks.

System response: Due to the size as complexity of the datasets as well as the nature of the algorithms, complex network tasks are usually *off-line* processes: once a task is submitted there is no expectation that a response will be generated in a short period of time. Database systems are tailored to support OLTP (On-Line Transaction Processing), typically generating responses in seconds.

Interaction cycles: Once a response is produced in a complex network task, the researchers analyze the results and may identify issues or improvement opportunities that require a new run. The extra execution may require scripts and parameters to be changed or a restructuring of the dataset. These are by themselves time-consuming tasks that together with the execution times result in long interaction cycles. The declarative nature of database interaction allows for shorter analysis-execution cycles, relying heavily on the ability of queries to express subsets of data to be assessed and on the under-the-hood optimizations performed by the query processor. When a response to a database query is deemed inappropriate, preparing a new execution usually requires simple modifications to the query and a quick resubmission.

Flexibility: In terms of potential features, network analysis software are more flexible than database systems. A software developer for complex networks can implement any type of algorithm and expose its input parameters and present the output using any adequate interface component. Databases are usually restricted to small numbers and types of features. This restriction is important to enable standardization and to allow system designers to define and implement generic optimizations for query evaluation.

Learning curve: Network analysis software can be made intuitive through smart use of GUI (Graphical User Interface) components such as menus, dialogs, and assistants. A

new user can start performing simple to reasonably complex tasks without much training. In a database system, the user must be familiar with the underlying data model and query language for even the simplest tasks. Although learning these mechanisms is fairly straightforward, certain training is needed until a new user becomes familiar with the system.

Replicability and reuse: There is little to no reuse in traditional complex networks settings since there is no standardization in network analysis software. Therefore, although the learning curve might be gentle for a given package, users need to familiarize themselves with each new software. As for database systems, languages and models are well established and standardized, making it easy to transition between systems. It is also easier to replicate an experiment in another environment and to share data and queries among research groups.

2.2.2 Advantages of query-based, database-backed complex network analysis

Both network analysis software and databases systems have particular advantages and shortcomings in their typical modus operandi. We aim at striking a balance between the shortcomings and advantages in order to provide several mechanisms of database systems to be employed in network analysis scenarios. This approach can improve the network analysis workflow in several aspects, as described in the following paragraphs.

A straightforward advantage of backing complex networks with database systems is that of enabling a robust storage mechanism. Databases have been developed over the decades to provide consistency, scalability and availability in a centralized data management infrastructure. In a network analysis environment, these features facilitate data curation, simplifying how data is modified to create new versions of the database.

The maturity and wide adoption of database systems would also have an impact in performance and scalability. Using standardized query languages as input for processing tasks allows database systems to make decisions regarding the execution plan in order to optimize running time. For example, a query optimizer can decide on the order of execution of subtasks, combine parts of a query or multiple queries, replace operations with equivalent, more efficient options, employ indexes and caching, etc.

Furthermore, databases are designed to handle data that may not fit in main memory. The direct usage of file system's might lead to costly import/export data exchange protocols, whereas database solutions favor efficient cache policies and index structures for efficient data access. For even bigger datasets, distributed databases can be employed to balance the storage and processing load among multiple computers. Distributed databases also provide increased reliability through data replication.

While the advantages related to data management, performance, and scalability are compelling, we believe the most important benefits of the proposed integration are related to the flexibility provided by a query-based approach. A well-designed query system for complex networks has the potential to significantly change the traditional workflow of network analysis and enable types of analysis that are often too time-consuming to be viable in current settings.

The shorter interaction cycles common in database systems would enable exploratory analysis of complex networks. Researches can specify queries focusing on specific links or subgraphs, tweak the queries based on the responses, change focus, expand coverage, etc. This type of interaction facilitates the identification of, for example, localized phenomena or outliers in the network.

Representing complex network tasks as database elements also enable a higher level of modularity to complicated tasks that involve several steps. Database queries and stored procedures (automated routines that can be scheduled or triggered from queries) can represent each step of the task. The modular steps can then be rearranged, reused, or modified following convenient and standardized protocols.

With expressive data models and query languages it becomes simple to tackle richer network models including multigraphs with weighted edges. The query-based analysis of network dynamics can allow researchers to control, in a streamlined fashion, how different links influence the information flow in the graph. Researchers could also assess elements of the network based on combinations of scores derived by distinct measurements.

2.2.3 Data models

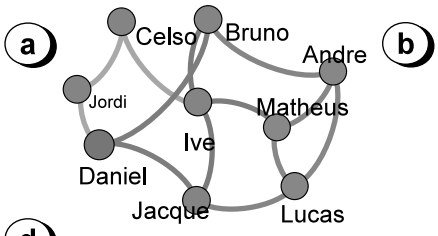
Query languages enable most of the advantages offered by DBMSs. By separating the specification of user's information needs from the instructions to retrieve and manipulate the data, databases systems are free to implement efficient and adaptable mechanisms to answer queries. Moreover, the systems can store and curate the data in a manner that is most adequate to the implemented mechanisms.

To offer this type of query-based declarative interaction with users, a database system must define an underlying data and query model. The data model determines how data are structured and organized. Examples of data models are the graph model (which is convenient for complex networks) [5], the semi-structured model (used for hierarchical data such as XML documents) [50], and the relational model (used in most commercial database systems) [17]. Since the graph and the relational models are central to the proposal in this paper, we will provide more details in the following paragraphs.

The graph data model is a natural option for representing complex network data. The graph model has relationships encoded explicitly as first class citizens, which allows flexible representations of a wide range of real-world data [55]. Many graph databases adopt a labeled multidigraph model to represent the underlying data.

A labeled multidigraph G is a directed multigraph with labeled vertices and edges. Formally, $G = (V, E, \Sigma_V, \Sigma_E, s_E, t_E, \ell_V, \ell_E)$ where V is a set of vertices and E is a set of edges, Σ_V and Σ_E are finite alphabets for the available vertex and edge labels, $s_E : E \to V$ and $t_E : E \to V$ are maps indicating the source and target vertex of an edge, ℓ_V and ℓ_E are maps describing the labeling of the vertices and edges. To add further flexibility, graph databases typically adopt the property graph model [55], which allows for vertices and edges to have data attributes (properties).

The relational model [17] was proposed by E. F. Codd in the 70's and is to this day the most used model to represent data in information systems. The model is based on the mathematical definition of relation. Briefly, given the sets $S_1, S_2, ..., S_n$, a relation R



b Person

id	name	type	age
14	Andre	person	32
15	Ive	person	23
16	Celso	person	29
17	Bruno	person	25
18	Daniel	person	26
19	Lucas	person	31
20	Jacque	person	22
21	Matheus	person	25
22	Jordi	person	27

c Knows

id	id_n	label
14	17	knows
14	19	knows
14	21	knows
15	16	knows
15	17	knows
15	20	knows
15	21	knows
16	22	knows
17	18	knows
18	20	knows
18	22	knows
19	20	knows
19	21	knows

d Friend of Friend

k1.id	k1.id_n	k1. label	k2.id	k2.id_n	k2.label
14	17	knows	17	18	knows
14	19	knows	19	20	knows
14	19	knows	19	21	knows
15	16	knows	16	22	knows
15	17	knows	17	18	knows
17	18	knows	18	20	knows
17	18	knows	18	22	knows

Figure 2.2: Social network and relational tables.

contains ordered tuples with attributes from $S_1, S_2, ..., S_n$. S_j is called the jth domain of R. In practice, relations are referred to as tables that contain rows (tuples). Tables have labeled columns based on the domain of the sets in the definition. For example, Figure 2.2b shows a table with nine rows and four columns. The domain of the column *age* is the natural numbers. The relational model can be used to represent graph data. In this case, node becomes a row and an extra table must be created to represent the links between nodes, such as in Figure 2.2c.

2.2.4 Relational Algebra

A data model must also provide a set of operations to manipulate the database [23]. In the case of the the relational model, the operations are defined in the relational algebra [17]. In this paper we extend the relational algebra to enable query and analysis of networks (Section 2.4.2).

The relational algebra contains basic set operations from set theory, such as UNION, INTERSECTION, SET DIFFERENCE, and CARTESIAN PRODUCT. The basic operations are extended with operations developed specifically for relational databases. The most relevant operations for this paper are SELECT, PROJECT, JOIN, and AGGREGATION, informally described next. Every relational algebra operation produces a new table that in turn can be the subject of other operations. Detailed descriptions off all operators can be found in references such as [23].

The $SELECT(\langle table \rangle, \langle condition \rangle)$ operation produces a new table containing the subset of the tuples in $\langle table \rangle$ that satisfies condition $\langle condition \rangle$. For example, using the table shown in Figure 2.2b, $SELECT(Person, \{age > 25\})$ returns a table containing all the rows with column age greater than 25.

The $PROJECT(\langle table \rangle, \langle attributelist \rangle)$ operation returns a table with tuples from $\langle table \rangle$ that are projected to contain only the attributes in $\langle attributelist \rangle$.

$PROJECT(Person, name)$ returns a table with only the attribute *name*. The operations can be composed such as in $PROJECT(SELECT(Person, \{age > 25\}), name)$.

Intermediate results can be given a table name for future reference. The previous

expression can be written as:

$OlderThan25 \leftarrow SELECT(Person, \{age > 25\})$

$Result \leftarrow PROJECT(OlderThan25, name)$

Attributes in a table can also be renamed with the same naming operation by using:

$\langle tablename \rangle(\langle attributelist \rangle) \leftarrow \langle table \rangle$

The expression $Result2(person\_name) \leftarrow Result$ renames the attribute *name* in the *Result* table. In ambiguous situations, the attribute name can be referred as *table.attribute* (e.g. Person.name). A table can be renamed inline when used as a parameter for an operation. For example:

$SELECT(Person\ AS\ p, \{p.age > 25\})$ renames table $Person$ to p.

$JOIN(\langle t1 \rangle, \langle t2 \rangle, \langle condition \rangle)$ is arguably the most important operator in the relational algebra. It is defined as a cartesian product (all possible combinations of tuples from tables $\langle t1 \rangle$ and $\langle t2 \rangle$) followed by a selection on the result (with the condition in $\langle condition \rangle$). The expression:

$JOIN(Knows\ AS\ k1, Knows\ AS\ k2, \{k1.id\_n = k2.id\})$

Returns the table in Figure 2.2d containing *friend of a friend* relationships (assuming, for simplicity, that the *knows* relationship is non-reflexive). Another important functionality for practical uses of the relational algebra is the ability to group tuples of a table and apply an aggregation over the grouped attributes.

$AGGREGATE(\langle table \rangle, \langle grouping\ \ attributes \rangle, \langle functions\ \ list \rangle)$ groups tuples in $\langle table \rangle$ according to the attributes in $\langle grouping\ attributes \rangle$. For each formed group, the aggregation functions in $\langle functionlist \rangle$ are applied to assess the result values. Aggregation functions can contain operations such as count (COUNT), sum (SUM), minimum (MIN), maximum (MAX), and average (AVG). For example, the expression $AGGREGATE(Person, type, \{AVG(age)\})$ returns a table containing one line with the the attribute value 25.6.

The SQL language was the first widely adopted database query language and is still the most used and influential query language. SQL is based in the relational algebra and allows simple combination of operators and aggregation functions. Our goal is to provide the CDMS with a similar language that combines declarative querying with graph-based analysis.

2.2.5 The envisaged workflow and application scenarios

Figure 2.3 depicts a typical workflow for our envisaged database-backed graph analysis. We illustrate the activities undertaken in each phase using the social network data in Figure 2.2.

The network load phase comprises all tasks performed to insert the data in the database. The initial data can be decoded in different formats, such as XML, comma separated files, etc. The data must be converted to a format that is compatible with the database system. Furthermore, it is often necessary to fix inconsistencies in the source data, for example, normalizing names as "last-name, first-name". These steps are common to most database systems and as such there is a great variety of tools and methods proposed and implemented in the last decades [36]. Once the network is imported, the

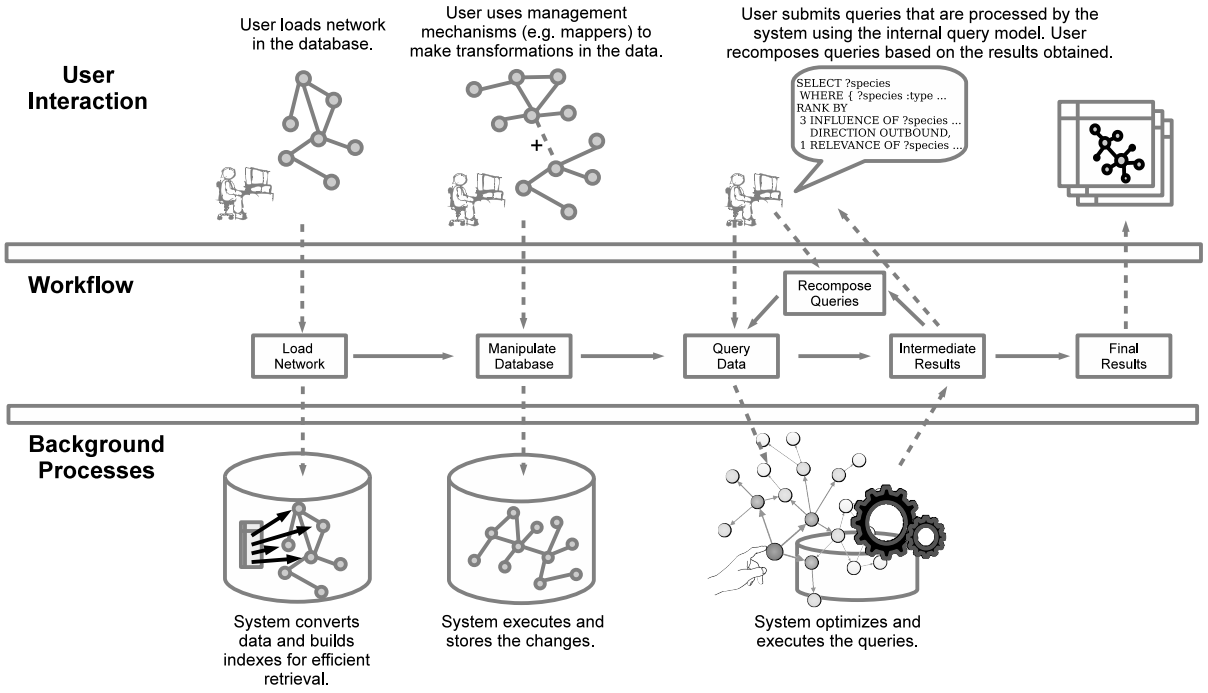


Figure 2.3: Comparison of main characteristics of Database Systems and Complex Network Analysis.

underlying database system can automatically (or driven by the administrator) execute procedures to optimize the performance of the queries over the data (e.g. by populating indexes or collecting statistics, see Section 2.3.3).

In the data manipulation phase, database users can transform the underlying network to suit the analysis needs. Using the social network example, the user may want to project the graph to create a friend-of-a-friend network. The projected graph can then be stored in the database for future use. A database system for complex networks must offer means to simplify these operations. We describe the operations implemented by our CDMS in Section 2.4.2.

The data querying and recomposition phase is the most important component in any database system. The query language allows users to have fast cycles of query submission, analysis of results, and query recomposition steps. In our social network example, users can compose queries to calculate metrics such as PageRank on the entire network or only over the friend-of-a-friend projection. Then they can choose to further investigate the most influential persons, retrieving their respective friendship network to calculate, for example, the average income of their friends. The underlying query processor is responsible for executing the queries in the most efficient way possible. To support this type of interaction, the CDMS must offer a declarative query language that can express the needed network analysis measurements. We describe our query language and operations in Sections 2.4.2 and 2.4.4.

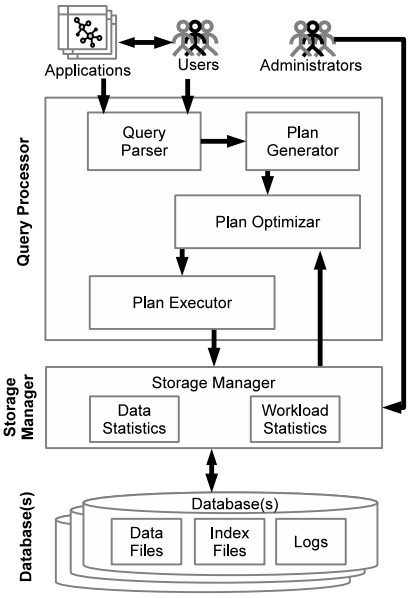


Figure 2.4: Typical architecture of a Data Management System (query return arrows omitted for clarity)

2.3 Designing a Complex Data Management System

There are several challenges in designing and implementing a database system for complex networks, from new requirements associated with the application scenario to adequate architectural elements. The new challenges and requirements define a new class of system that we refer to as Complex Data Management Systems (CDMS). A CDMS is a system that stores, manages and queries complex network data.

Complex network data in turn is characterized when relationships are defining elements of the data analysis. Any subset of a complex network can be characterized as complex data. Typically, these structures generate emergent behavior, which is determined by the complex interactions among their simple constituent elements.

2.3.1 Architecture

Figure 2.4 shows the typical architecture of a Database Management System (DBMS). The architecture for the CDMS follows the same blueprint. The difference is in how each part is specified and implemented (Section 2.4). Here we describe and contextualize each module of the architecture.

Users can access the system by issuing queries directly to the query processor or by interacting with network analysis software adapted to interface with the CDMS. Administrators manage the system, being responsible for tasks such as controlling access, tuning execution parameters, and performing backups. The query processor is the central part of the system. It receives queries from users and applications, processes the queries utilizing advanced transformation and optimization procedures, and returns the final results to the requester.

The storage manager is capable of reading the binary format files in which the database

is stored and can use the index structures to access information efficiently. The storage manager is also responsible for keeping statistics about the underlying data that can aid in optimization decisions.

The bottom layers of the architecture represent low level mechanisms for query processing and storage. Separating these elements from the core of the CDMS allows the selection of adequate infrastructures according to the task. For example, in large projects, the graph processing engine and graph database can be distributed over a network of computers, providing scalability and load balancing. In the following sections we describe the main elements of the architecture with more details.

2.3.2 Query model/language

A well designed query language is an important requirement for a CDMS. In a database system, the query language is the interface between the data and external users. A declarative querying approach is the typical choice for databases. A *declarative* query language allows users to specify their information needs without having to provide details about how the data should be fetched and processed. It is up to the query processor to define efficient access plans to answer the requests.

A query model for complex networks should offer means to select the target data (subgraphs), define and combine measurements, and specify appropriate stop conditions. For data selection, it is important to support features such as restricting the types of links used for analysis, definition of target subgraphs based on radii of neighborhoods, filtering of nodes based on properties or adjacency, etc.

It is important for a query to be able to precisely specify the intended measurements to assess the properties of the network. Measurements could also be combined, being it combinations of different measurements or of the same measurements assessed over variations of the graph (e.g. distinct link types).

Finally, the query language must allow modifications to the underlying data. A dataset may change for several reasons, for example, corrections, new samples, projections to simplify analysis, etc. These changes must be properly specified and handled, coupled with simplified materialization of results. Features like undos, transactions, versioning, and provenance can also be considered in this context. We do not focus on these aspects in the paper since current implementations already offer satisfactory solutions.

2.3.3 Query processor and optimizer

The query processor is responsible for answering requests as efficiently as possible. It relies on several modules do perform subtasks (Figure 2.4). The query parser translates the input query into an internal representation that can be read and manipulated by the other modules, usually a tree-like structure based on a formal algebra. The plan generator rewrites the query tree following equivalence rules to produce multiple candidate execution plans. The plan optimizer uses workload and data statistics to choose an efficient plan for execution by the plan executor. The plan executor then issues data retrieval operations to the storage manager to gather the necessary data to answer the query.

A typical query processor can explore diverse strategies to optimize the execution [23, 14]. Traditional approaches rely on (i) changing the order of operations to reduce the size and complexity of intermediate steps, (ii) replacing operations with equivalent ones that are less costly, (iii) identifying commonalities among the operations to avoid repeating analogous steps, and (iv) exploiting pre-processed summaries (caches) that are known to answer the query or parts of it.

2.4 An implementation for the CDMS

In this section we detail our current implementation of the CDMS.

2.4.1 Data model

To enable more flexibility in query composition and better support for value aggregation in the analysis, we adopt the relational model and an extension of the relational algebra for our query model.

In this paper we use a simple interpretation of graphs in the relational model. In this model, a labeled property multigraph G is represented in two relational tables, V (vertices or nodes) and E (edges or links). The tables contain attributes representing the properties for all nodes and edges (typically highly sparse tables). The V table is in the form $\langle id, a_1, a_2...a_i \rangle$, where id represents the node id in the database, and $a_1, a_2...a_i$ are node properties represented as attributes with a domain in the relational model. The model assumes that all possible properties are presented in the schema and non applicable properties are represented as null values. Figure 2.2b is an example of a valid V table.

The E table is in the form $\langle id, id_n, label, a_1, a_2...a_j \rangle$. id and id_n are ids representing the connected nodes for a given edge (the order implies the direction), $label$ is the label according to the labeled property graph model, and the properties are as in the V table. Figure 2.2c is an example of a valid E table.

This data model is the reference for the proposed algebra. However, as with any other model, it does not impose implementation constraints (e.g. it can be implemented over a pure graph database or use efficient structures to represent the sparse tables).

2.4.2 The beta algebra

We extend the relational algebra by adding the beta operator [30]. The new operator simplifies the specification of graph-based analysis, allowing the definition of diverse measurements to capture the properties of the network.

In the following paragraphs, the beta operator will be presented informally, in increasing complexity as parameters and sub-operators are introduced. Since the operator can be seen from either a graph or from a relational perspective, we will use equivalent terms interchangeably (e.g. join and traversal step). The semantics, however, is always relational.

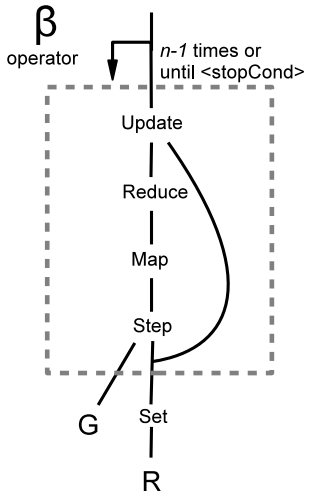


Figure 2.5: Simplified execution tree template

The beta operator

In its simplest form, the beta operator represented as $Beta(R, G)$ performs joins between the R relation, the E ($E \in G$) relation, and the V ($V \in G$) relation. For example, the operator can be used to find all friends of "Ive" in Figure 2.2. The expression $Beta(R, G)$, with $R \leftarrow Select(Person, \{id == 15\})$ and $G \leftarrow (Person, Knows)$ (meaning Person is the V and Knows is the E tables) returns the relation presented in Table 2.2.

id	Label	type	age	id_n	E.label	V.id	V.Label	V.type	age
15	Ive	person	23	16	knows	16	Celso	person	29
15	Ive	person	23	17	knows	17	Bruno	person	25
15	Ive	person	23	20	knows	20	Jacque	person	22
15	Ive	person	23	21	knows	21	Matheus	person	25

Table 2.2: Results for a simple beta expression.

In general, the beta operator performs recursive joins and is represented by $Beta(R, G, n)$. Where n is the number of iterations. Friends and friends-of-friends of "Ive" are obtained by $Beta(Select(Person, id = 15), G, n : 2)$.

The basic beta operator is equivalent to traversing the graph one step at a time. To increase flexibility in this step operation, we add more (optional) parameters to the operator, as follows: $Beta(R, G, n, follow, dir)$. $follow$ is a set that contains the valid edge labels to traverse (join) and $dir \in \{IN, OUT, BOTH\}$ determines the direction of the edge traversals.

To allow full control of the computation as the graph is traversed, we define four aggregation operations: *Set*, *Map*, *Reduce*, and *Update*, that are also specified as optional parameters to the beta operator. *Set* is a function that attributes a value to a new column before the join (traversal) operation is performed. *Map* calculates a new value for the neighbors based on values in the origin node. *Reduce* is a sub-operator that aggregates over values based on a *group_by parameter* (similar to relational aggregation). Finally, *Update* defines how new values are integrated with previously calculated values. Figure

```
set←{count ← 1}
map←{new.count ← current.count}
reduce←({count ← sum(count)},[id_n])
update← {current.count ← current.count + new.count}

R ← select(Person, {age > 30})
Count ← beta(R, G(Person,Knows), n:3, direction:BOTH, follow:[knows],
                    set, map, reduce, update)
```

Figure 2.6: Query that counts paths in the sample social network.

2.5 shows a simplified execution tree for the beta operator. The *Step* sub-operator is responsible for performing the join inside the operator. In the figure, an initial table R is subject to a cycle of sub-operators that will in practice simultaneously traverse the graph and aggregate the graph's properties according to the desired measurement.

For example, consider the query that calculates the total number of paths from people older than 30 to all other nodes in up to 3 steps. We represent the query as the expression in Figure 2.6. Lines 1-4 are used to define the expressions used in the sub-operation (set, map, reduce and update), determining the aggregations to be performed. Multiples expressions for each sub-operation are accepted (e.g. multiple *reduce* expressions). The expression in line 10 selects all person nodes with attribute *age* greater than 30. Line 11 invokes the beta operator over the table R for 3 iterations ($n : 3$).

Figure 2.7 shows a graph view of the execution of the first iteration of the operator. Nodes are labeled with the first letter of persons' names. *Set* initializes the *count* values for the nodes in the input table ("Andre" and "Lucas"). *Map* has the effect of forwarding the count values from the current nodes (current.count) to all neighbors (new.count). *Reduce* aggregates the counts for the neighbor nodes (i.e., counts originated from different source nodes). Finally, *Update* adds new derived counts (new.count) to previous totals (current.count).

Figure 2.8 shows the same execution of the first iteration of the operator, but in the relational perspective. *Set* creates and assigns the initial value to attribute *count* (represented as *ct*). *Step* makes the join with the neighbor nodes and *Map* assigns the current value of *count* to the joined table (which is equivalent to forwarding the value to all links to the neighbors). The non used extra attribute c is automatically included by the *Map* operator and is equivalent to the degree (query-specific, based on $follow$ and *direction* parameters) of the source node (see usage example in Figure 2.12). *Reduce* is a simple relational aggregation. The grouping parameter *id_n* instructs the sub-operator to aggregate exclusively over the nodes derived from the *Step* sub-operator. If the goal of query was to separate the counts for each of the origin nodes, the aggregation parameter would be *[id_n, id]*. Finally, *Update* joins the processed table (new) with the initial one after *Set* (current) and applies the specified expression.

The beta operator also offers an optional parameter that describes the conditions for the evaluation to stop. This can be used for finding the shortest paths between nodes or as a convergence condition. The parameter is a list of expressions in the form

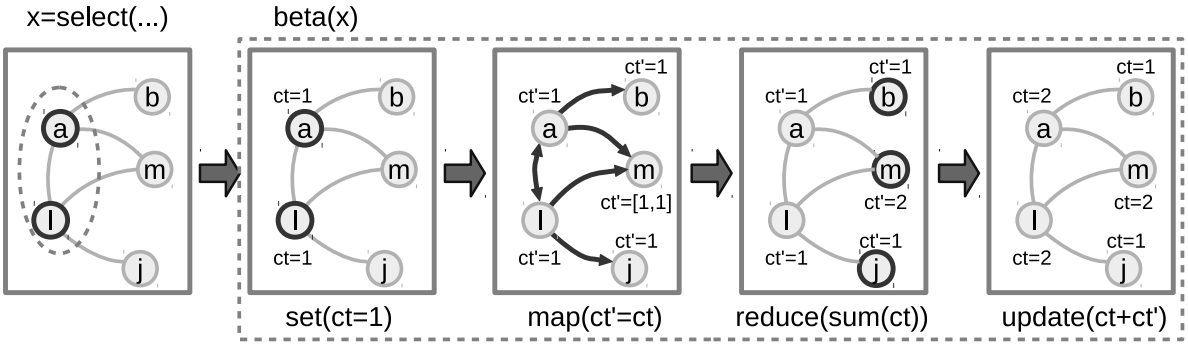


Figure 2.7: Simplified graph interpretation of the operator's execution. *ct* is the current count value, *ct'* is the new count value.

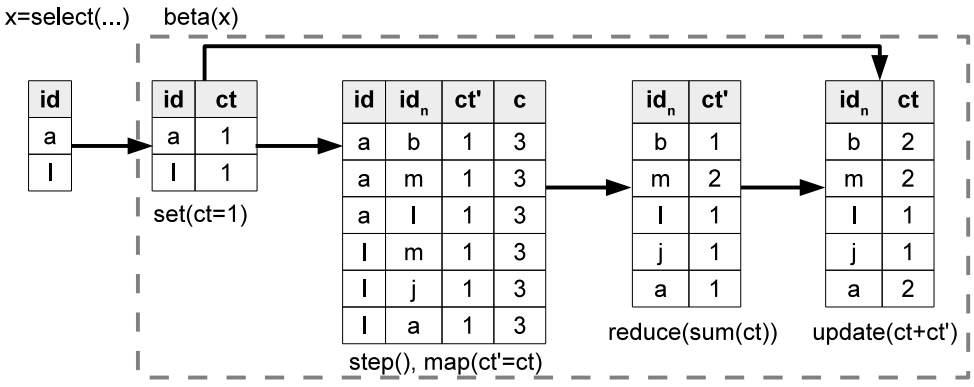


Figure 2.8: Simplified relational interpretation of the operator's execution. *ct* is the current count value, *ct'* is the new count value.

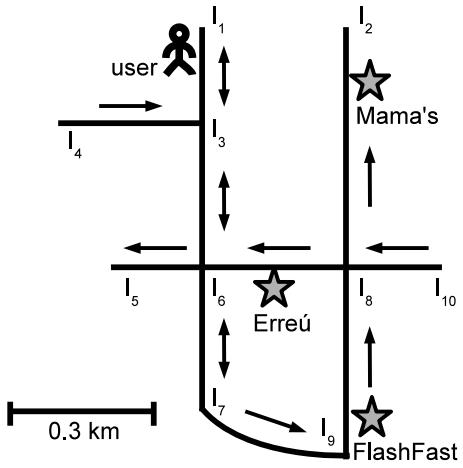


Figure 2.9: Map representation of the geographic data used in the running example

id	*id_n*	*minDist*	*maxDist*
3	1	0.20	0.80
3	2	1.10	1.10
3	3	0.0	1.20
3	4	0.30	0.90
3	5	0.50	1.10
3	6	0.30	0.90
3	7	0.60	1.40
3	8	0.60	1.60
3	9	1.00	1.00
3	10	0.80	0.80
3	11	1.00	1.20
3	12	0.50	1.10
3	13	0.80	1.20

id	*id_n*	*minDist*	*maxDist*
4	1	0.50	1.10
4	2	1.40	1.40
4	3	0.30	0.90
4	4	0.0	1.20
4	5	0.80	0.80
4	6	0.60	1.20
4	7	0.90	0.90
4	8	0.90	0.90
4	9	1.30	1.30
4	10	1.10	1.10
4	11	1.30	1.30
4	12	0.80	1.00
4	13	1.10	1.10

Table 2.3: Results for minimum and maximum distances query (see scale in Figure 2.9).

"*Every*{*condition*}" or "*Any*{*condition*}". The former stops the operator if *condition* (boolean) is true for all elements of the relation after the *update* sub-operator. The later stops the operator if *condition* is true for at least one tuple.

2.4.3 Assessing measurements with the beta-algebra

We now demonstrate the use of the beta-algebra for the assessment of several measurements, from typical distance and reputation measurements to more sophisticate methods that could benefit from our approach. To illustrate the execution of the queries we employ the data presented in Figure 2.2 and Figure 2.9.

Figure 2.9 represents a road network which is represented as a graph with intersections as nodes (I1, I2, etc.) and directed edges representing connecting streets (distance as weights, details in Section 2.5.2). The query in Figure 2.10 calculates minimum and maximum distances from points I3 and I4 in the map to all other points. The resulting relation is shown in Table 2.3.

```
set←{minDist ← 0.0, maxDist ← 0.0}
map←{new.minDist ← current.minDist + E.Weight,
                new.maxDist ← current.maxDist + E.Weight}
reduce←({current.minDist ← min(new.minDist, current.minDist),
                current.maxDist ← max(new.maxDist, current.maxDist)}, [id_n, id])
update← {current.minDist ← min(new.minDist, current.minDist),
                current.maxDist ← max(new.maxDist, current.maxDist)}

TableOrigins ← select(V, {id IN [3,4]})
Dist ← beta(TableOrigins, G(V,E), n:4, direction:BOTH, follow:[connects],
                  set, map, reduce, update)
Dist ← project(Dist, [id, id_n, minDist, maxDist])
Dist ← order_by([id, id_n], ASC)
```

Figure 2.10: Query that assesses minimum and maximum distances fro points I3 and I4 in up to four steps.

```
a)
AllDist ← Select(V, type IN [intersection, restaurant])
AllDist ← Beta(AllDist, E, n:4, direction:BOTH, follow:[connects], set, map, reduce, update)
AllDist ← Project(AllDist, [id, id_n, minDist, maxDist])
DistPerPoint ← Aggregate(AllDist, id, average(minDist))
DistPerPoint ← Project(DistPerPoint, [id, average])

b)
TableDistTotal = Aggregate(AllDist, {}, average(minDist))
```

Figure 2.11: Query that assesses average minimum and maximum distances fro points I3 and I4 in up to four steps (set, map, reduce, and update are defined as in the previous query).

To calculate average minimum distances from each point, a new aggregation query can be run over all points in the map. Figure 2.11a shows the code for the query while Table 2.4 shows the result relation. To calculate the average distance for the entire network, the query in Figure 2.11b will return the value 0.64.

A PageRank-like query can be composed in a similar fashion. Taking the data in the example social network, Figure 2.12 shows a query that sets all nodes with an initial value of 100 which is then iteratively mapped and aggregated for all neighbors. The resulting table for three iterations of the beta operator is shown in Table 2.5.

Combining relational algebra with graph analysis simplifies the specification of measurements that can emphasize or be constrained by a subset of the network. This type of interaction between measurements and subsets of the network is very important for explorative querying. In the next paragraphs we describe some of such measurements, from our own use and based on the literature.

White and Smyth [63] proposed and discussed different measurements that assess global properties biased towards a subset of especial nodes. For example, PageRank with

id	*average*
1	0.79
2	0.80
3	0.62
4	0.89
5	0.63
6	0.46
7	0.67
8	0.44
9	0.72
10	0.62
11	0.72
12	0.43
13	0.58

Table 2.4: Results for average distances query.

```
set←{rank ← 100}
map←{new.rank ← current.rank/c}
reduce←({rank ← sum(rank)},[id_n])
update← {current.rank ← new.rank}

TableOrigins ← select(V, {type == person})
TableRank ← beta(TableOrigins, E, n:3, direction:BOTH, follow:[knows],
                 set, map, reduce, update)
TableRank ← order_by([rank], ASC)
```

Figure 2.12: Query that assesses reputation in the sample social network.

Priors is an extension of the popular algorithm that includes a "back probability" of the random surfer returning to the initial set of especial nodes. Figure 2.13 shows a query that implements the measurement and applies to the social network in our example. In the query, we set the priors for nodes 14 and 16 in lines 4 and 5. The new formula in the reduce operation (line 9) takes the prior into account to assess the measurement. The results are shown in Table 2.6.

The flexibility offered by coupling the beta operator with relational algebra simplifies the definition of even more specific measurements. We have been extensively using a measurement for relative importance between nodes in the network based on the concept of relevance as generally applied in the area of information retrieval (e.g. web search engines).

Information retrieval, in general, measures the relevance of documents with respect to queries based on how many terms they share and how specific the terms are (similar to the TF*IDF approach). Our generalized measurement for nodes in a graph assesses relevance between a group of initial nodes R and a node v. Nodes are considered highly relevant when they have many paths connecting them and when the paths are short and specific. Intuitively, two nodes connected by a path that passes through a hub are not

```
vPerson ← Select(V, {type == person})
beta ← 0.25
total ← count(vPerson)
prior ← 1/total
vPerson ← UpdateTable(vPerson, {p = prior}, where: {id == 16 OR id == 14})

set←{rank ← prior}
map←{new.rank ← current.rank/c}
reduce←({rank ← (1-beta)*sum(rank) + V.p*beta},[id_n])
update← {current.rank ← new.rank}

Results ← Beta(vPerson, E, V:vPerson, n:10, direction:BOTH, follow:'knows',
                set, map, reduce, update)
Results ← OrderBy(Results, rank, DESC)
```

Figure 2.13: PageRank with Priors query.

id	*type*	*rank*	*id_n*
16	Celso	70.49	16
22	Jordi	73.84	22
21	Matheus	95.25	21
20	Jacque	98.03	20
17	Bruno	98.03	17
14	Andre	102.78	14
19	Lucas	102.78	19
18	Daniel	111.34	18
15	Ive	147.45	15

Table 2.5: Results for the reputation query.

id	*Label*	*p*	*rank*
14	Andre	0.5	0.19
16	Celso	0.5	0.18
15	Ive	0.0	0.13
21	Matheus	0.0	0.09
17	Bruno	0.0	0.09
19	Lucas	0.0	0.09
22	Jordi	0.0	0.09
18	Daniel	0.0	0.07
20	Jacque	0.0	0.06

Table 2.6: Results for the PageRank with Priors query.

```
gamma ← 0.8
set←{rank ← 1.0}
map←{new.rank ← gamma*current.rank/c}
reduce←({rank ← sum(rank)},[id_n])
update← {current.rank ← new.rank}

R ← Select(V, id IN [16,17])
I ← Beta(R, E, n:5, direction:BOTH, follow:['knows'], set, map, reduce, update)
I ← OrderBy(I, rank, ASC)
I ← Project(I, [V.id, V.Label, rank])
```

Figure 2.14: Query that assesses relevance between nodes 16 and 17 and all other nodes in the social network.

considered highly relevant – in the same sense that queries and documents containing an intersection of common words are not considered relevant. Our measurement is related to the proposals in [24] for communicability betweenness. Our measurement is defined by the formula:

$$I(v|R)^n = \begin{cases} 0, & \text{if } v \notin R \ \ \& \ \ n = 0 \\ 1, & \text{if } v \in R \ \ \& \ \ n = 0 \\ I(v|R)^{(n-1)} + \sum\limits_{u \in in(v)} p(v|u) I(u|R)^{n-1}\gamma, & \text{otherwise} \end{cases}$$

where $p(v|u)$ is the transition probability and $\gamma, 0 < \gamma < 1$ is a signal degradation parameter that favors shorter paths. Figure 2.14 shows an example of relevance query that assesses the relevance of the nodes in the example social network to the set of nodes [16,17]. The measurement is assessed for five steps ($n = 5$). The results are shown in Table 2.7.

V_id	*V_Label*	*rank*
17	Bruno	1.50
16	Celso	1.48
15	Ive	1.17
18	Daniel	0.77
22	Jordi	0.76
14	Andre	0.55
20	Jacque	0.45
21	Matheus	0.39
19	Lucas	0.32

Table 2.7: Results for the relevance query.

2.4.4 Query language

The beta-algebra is an abstraction for the type of query processing we are aiming at. Although it is the basis for our query processing mechanism, it is not intended to be used

```
START rest=node(*) WHERE rest.type = "restaurant" RETURN rest;
RANKED BY
a { 1 GEORELEVANCE OF rest TO node($ref)
      DEPTH 7 FOLLOW 'connects' DIRECTION both DESC
b { 2 REPUTATION OF rest
      DEPTH 6 FOLLOW ('likes', 'knows')
```

Figure 2.15: Query used in the example

by the users. The definition of a high-level query language that can take advantage of all elements of the proposed algebra is our next goal.

Currently, we are using a more restricted query language named "in*" (in star) that extends existing graph queries with ranking based on network measurements. Figure 2.15 shows an example of query in our language (see Section 2.5.2 for the semantics of the query). The graph-based aggregation is expressed in a RANK BY clause that combines the rankings obtained by different measurements (geo-relevance and reputation, in the example). The clause accepts measurements that aggregate values over graph traversals as in the Beta-algebra presented. The parameter *depth* determines the radius of the graph expanded by the beta operator (n). The parameter *follows* determines which types of links should be traversed by the algorithm. Details about the measurements and queries can be found in Chapter 4.

2.5 Use cases

In this section we present two usage scenarios for the proposed system. The first scenario encompasses several tasks for the analysis of the food network introduced in [2]. The second scenario is in the area of geographic POI (point of interest) recommendations. This scenario is based on synthetic data that combines diverse types of information. The cases provided were chosen based on the didactic properties and aim at showcasing diverse aspects of the proposed query-based interaction.

2.5.1 Food web analysis

The data used in this scenario was originally described in [2]. It contains information extracted from popular recipe websites (Epicurious, AllRecipes, etc.) spanning recipes, ingredients (and its chemical components), cuisines, and geographic regions. A schematic view of the organization of the network is shown in Figure 2.16[2]. In the figure, the label of the nodes represent the attribute "type" for nodes in the data graph, while labels in the edges represent the attribute "label" for the edges in the graph.

Data Cleaning

Most data analysis require an initial data transformation step. In the database industry, such step is referred to as data cleaning or, more generally, ETL (extraction, transformation, loading). In the case of the food web data that we used, a good deal of data

[2]Adapted from http://blog.bruggen.com/2013/12/fascinating-food-networks-in-neo4j.html

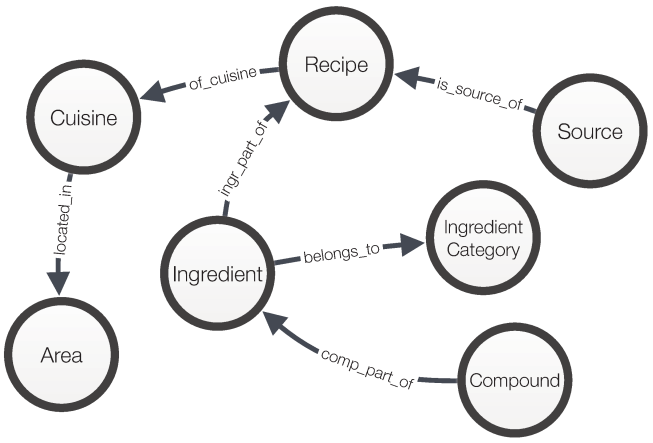


Figure 2.16: Schematic organization of the food network.

Cuisine ← Select(V, type == CUISINE)
CuisineRef ←
  Join(Cuisine AS Orig, Cuisine AS Ref, {$Similarity(Orig.name,\ Ref.name)\ >\ 0.61$
  $AND\ |Orig.name|\ >\ |Ref.name|$})

Figure 2.17: Algebraic expression for the first step of the data cleaning.

manipulation was done by the original authors of the research and also by the author of the website article that published the data as a graph.

To demonstrate a data transformation step using our algebra, we unified the cuisine designations in the dataset. The original data was extracted from crowdsourced recipe websites[3] that employ different classifications for cuisines. For example, some websites describe cuisines in more fine grained classes, such as 'East-African', while other use more general classes, such as 'African'. The first goal of this data cleaning step is to unify the cuisine names that refer to the same region. To accomplish that, we employ a self-join on a table with the cuisine nodes. The join condition employs a string similarity function, implying that only matches with similarity higher than a threshold will be output, and also a condition that favors smaller names as the final reference names for cuisines. The code is shown in Figure 2.17, the output table is shown in Table 2.8 (appendix). Subsequent joins, renaming and projections were applied to create links from recipes to the unified cuisine nodes (those steps are omitted here for brevity). As can be seen in the output, a few manual adjustments need to be made to create a proper reference table. The adjusted table in shown in Table 2.9 (appendix).

This simple data cleaning step is meant to further demonstrate the use of the relational algebra in the scenario and the advantages of dealing with networks as part of a central database. More advanced methods and tools have been used in the database community for several decades ref [36]. Our focus is on the analysis queries described next.

[3]Including www.epicurious.com, allrecipes.com

Orig.id	*Orig.name*	*Ref.id*	*Ref.name*
4101	East-African	4100	African
4103	North-African	4100	African
4104	South-African	4100	African
4105	West-African	4100	African
4107	chinese	4106	China
4108	east_asian	4163	asian
4110	japanese	4109	Japan
4112	korean	4111	Korea
4114	EasternEuropean_Russian	4113	Eastern-Europe
4116	Central_SouthAmerican	4119	South-America
4117	Mexican	4118	mexico
4119	South-America	4126	American
4126	American	4100	African
4126	American	4117	Mexican
4130	Southwestern	4162	western
4131	Scandinavia	4134	India
4131	Scandinavia	4135	Indian
4132	Scandinavian	4131	Scandinavia
4132	Scandinavian	4135	Indian
4135	Indian	4134	India
4141	Thailand	4140	Thai
4143	Vietnamese	4142	Vietnam
4145	Italian	4146	Italy
4157	Germany	4156	German

Table 2.8: Result table for the first step of the data cleaning.

Assessing cuisine influence

Having the cuisine nodes normalized, we now want to assess their relative influence. Since the cuisine nodes are not interconnected, we project the graph to materialize indirect connections. We traverse the graph through ingr-part-of and of-cuisine links and back (using the step operator) to determine the new links (co-occurrence of ingredients in a recipe from the cuisines). Since common ingredients such as sugar and milk are present in most cuisines, taking all ingredients into account would generate a complete graph. Therefore, we used the least common ingredients (that appear in fewer than 10 recipes) to redefine the graph for the step operations. The complete query for the task and the result table are shown in Figure 2.18 and Table 2.10 (appendix). The most influential cuisine according to the analysis is the American, followed by Jewish, Asian, French, and Mexican. The result reflects the fact that the base data for the study was extracted from US-based recipe websites.

To find the essential ingredients for certain cuisines, it is important to assess the pairwise relevance between cuisines and ingredients. The researches also want to understand the impact of a shortage of certain ingredients on the cuisines that rely on them.

We can use the *relevance* ranking to assess how correlated an ingredient is to the cuisines that use it. For example, the query in Figure 2.19 ranks cuisines based on their

id	*name*	*ref_id*	*ref_name*
4101	East-African	4100	African
4103	North-African	4100	African
4104	South-African	4100	African
4105	West-African	4100	African
4107	chinese	4106	China
4108	east_asian	4163	asian
4110	japanese	4109	Japan
4112	korean	4111	Korea
4114	EasternEuropean_Russian	4113	Eastern-Europe
4116	Central_SouthAmerican	4119	South-America
4117	Mexican	4118	mexico
4130	Southwestern	4162	western
4132	Scandinavian	4131	Scandinavia
4135	Indian	4134	India
4141	Thailand	4140	Thai
4143	Vietnamese	4142	Vietnam
4145	Italian	4146	Italy
4157	Germany	4156	German
4149	Spain	4150	Spanish_Portuguese
4154	France	4155	French
4158	Irish	4161	UK-and-Ireland

Table 2.9: Final CuisineReference table after manual adjustments.

relevance to *soy sauce*. The most relevant cuisines according to the metric are Korean and Japanese, followed by Chinese, Philippine, Indonesian, etc.

Using a similar query, researchers can identify the ingredients that are the most correlated with a cuisine. Those would be cuisines highly affected by a shortage of the ingredient. The researcher could then make even more specific queries to probe, for example, the impact of a shortage of a certain grain in different regions (food security assessment). To substantiate the analysis, the underlying data could be augmented with census and harvest forecast data, always taking advantage of the data management mechanisms provided by the database system.

2.5.2 Geographic-based recommendations

For our second usage scenario, we describe how the CDMS could be used by non-experts. The goal is to show how a query system capable of expressing network analysis information needs is useful in many applications. To this end, we manufactured a scenario with rich data and information requirements, inspired by geographic-based point-of-interest recommendation.

Geographic-based recommendations have become commonplace in our digital lives, accompanying and supporting us in real time on many daily tasks. Finding restaurants, locating businesses, planning touristic itineraries, suggesting routes, etc., are tasks that rely on the system's ability to recommend geo-coded elements based on a given initial

```
//Prepares subgraph for the rare ingredients
TableRareIngr ← Reduce(IngrPartOf, [id], {total ← count() })
TableRareIngr ← Select(TableRareIngr, {total < 10})
IngrPartOf ← TetaJoin(IngrPartOf, TableRareIngr)
TableE ← Union(IngrPartOf, OfCuisineRef)

//Projects the cuisine network
CoCuisine ← Step(Cuisine, TableE,
                    {in(of_cuisine).in(ingr_part_of).out(ingr_part_of).out(of_cuisine)})
CoCusine ← Select(CoCusine, {id > id_n})
CoCusine ← Unique(CoCusine)

//Executes the beta operator
TableResult ← Beta(Cuisine, CoCusine, n:3, direction:BOTH, set, map, reduce, update)
```

Figure 2.18: Algebraic expression that assesses influence based on the projected cuisine network (*set*, *map*, *reduce*, and *update* are set as in the PageRank query – Figure 2.12).

```
Cuisine ← Select(V, {type==CUISINE})
Result ← Beta(Cuisine, n:2, direction:INBOUND, set, map, reduce, update)
Result ← Select(Result, {id_n == 1283}) //selects soy sauce node
Result ← OrderBy(Result, rank)
```

Figure 2.19: Algebraic expression that assesses relevance to ingredient "Soy Sauce" (*set*, *map*, *reduce*, and *update* are set as in the Relevance query – Figure 2.14).

id	*name*	*rank*
4126	American	532.76
4122	Jewish	208.68
4163	asian	202.41
4155	French	163.47
4118	mexico	143.47
4146	Italy	141.99
4134	India	136.12
4100	African	119.19
4119	South-America	104.89
4161	UK-and-Ireland	104.89
4150	Spanish_Portuguese	90.37
4113	Eastern-Europe	86.31
4128	Canada	76.90
4129	Southern_SoulFood	76.90
4111	Korea	65.93
4102	Moroccan	58.12
4124	MiddleEastern	58.12
4109	Japan	52.17
4144	Greek	45.93
4147	Mediterranean	45.93
4162	western	40.49
4139	Philippines	35.07
4131	Scandinavia	30.28
4142	Vietnam	30.28
4140	Thai	17.99
4153	English_Scottish	15.67
4156	German	15.67

Table 2.10: Results for the cuisine influence query.

location.

In many situations, it is important to leverage other types of contextual information to produce the recommendations. Users may help guiding the processes by providing, for example, keyword queries or object types that help describe their information needs. Alternatively, the system may try to determine these preferences based on the users' profiles.

Our example query, for a location-based POI, is equivalent to "retrieve restaurants ranked by: geographic proximity of a starting point, relevance to the keyword query 'family-friendly restaurant', reputation in a social network, and relevance to the semantic concept 'Casual Dining'. The map corresponding to the geographic data used in our running example is shown in Figure 2.9.

Representing Topological information as graphs

To convert the transportation network into a graph, we created nodes to represent intersections and linked them respecting the permitted directions in their respective segments.

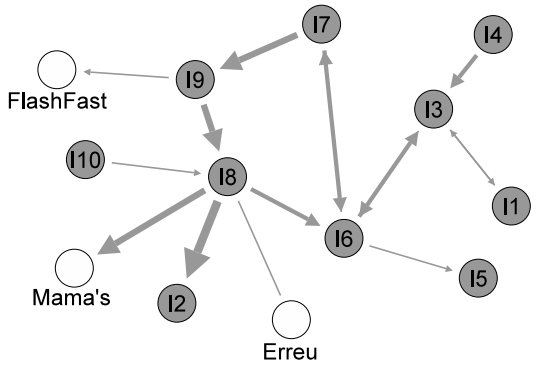


Figure 2.20: Road network and POIs represented as a graph. Edges thickness (weight) represents distance.

id	name	type	description
11	Mama's	Restaurant	Home-style cooking, comfort food, hand-made, and family friendly
12	Erreú	Restaurant	Rotisserie and grill, chicken dishes, whole food, family environment
13	FlashFast	Restaurant	Fastfood, sandwiches, soft drinks

Figure 2.21: Data contents for POIs

The distance between intersections (nodes) is encoded as the weight of the link. Points of interest also become nodes, with links connecting the intersections at the extremities of the closest segment.

Figure 2.20 shows the graphical interpretations of the map in Figure 2.9. Intersections and POIs are connected through links with label 'connects' (not shown for clarity). The weight of the links (the distances) is represented as the thickness of the arrows.

Textual data as graphs

Text documents can also be represented as graphs. Here we adopt a straightforward strategy: documents (in our case, points of interest) are represented as nodes; words in a document (excluding words that are too common) also become nodes that are linked to the node representing the original document.

Figure 2.21 shows a table with the content associated with each POI. To convert the textual data into a graph, the words in the description of the restaurants become nodes for our unified graph. The word nodes are linked to POI nodes containing that word. Likewise, keyword queries are converted into graphs and temporarily added to the database (the query is represented by a node linked to the nodes for its content words). In the CDMS, these and other data transformation operations are automatized by the *mapper* mechanism, which also plays an important role in query-time. Details of the mapper mechanism are described in [28].

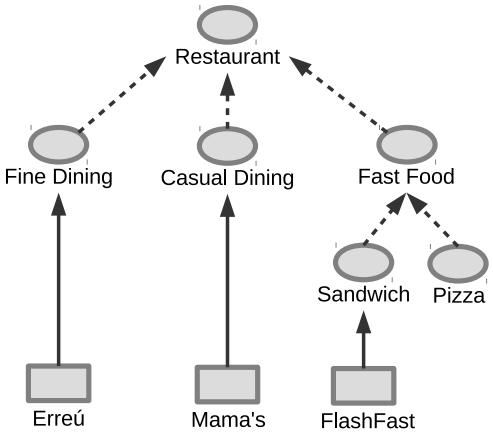


Figure 2.22: Ontology for the example showing specialization (dashed lines) and instanciation (full lines) relationships

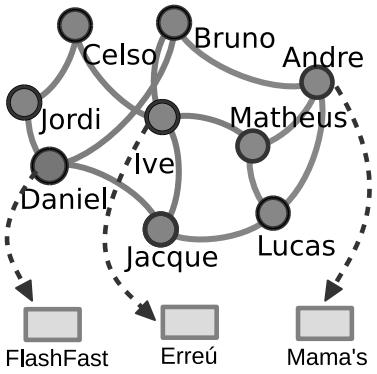


Figure 2.23: Social network for the example showing friendship/knows (full lines) and preferences/likes (dashed lines) relationships. The size of the node is proportional to its PageRank reputation.

Other information types

Several other important information sources for geographical recommendation can be easily represented as graphs. Semantic information from ontologies are already often represented as graphs (e.g. RDF[4]). The ontology used in our example is shown in Figure 2.22. In a recommendation systems those associations can be done by the restaurant owner, or by the company's staff.

Social networks are also naturally represented as graphs, and can include extra information such as user's preferences (i.e. likes), places visited, etc. The social network used for our example is shown in Figure 2.23. In the example the users manifest their preferences by creating the *like* link.

All this information (together with geographical and textual networks) can be integrated in a common graph. This graph can then be used to provide more contextualized recommendations, as shown in the next sections.

[4]http://www.w3.org/RDF/

```
START rest=node(*) WHERE rest.type = "restaurant"
RETURN rest RANKED BY
a 1 GEORELEVANCE OF rest TO node($ref) DEPTH 7
    FOLLOW 'connects' DIRECTION inbound DESC
b 1 RELEVANCE OF rest TO
    TokenMapper("family friendly restaurant")
    DEPTH 2 FOLLOW TokenMapper:hasToken,
c 1 REPUTATION OF rest
    DEPTH 6 FOLLOW ('likes', 'knows')
d 1 RELEVANCE OF rest TO node($ref)
    DEPTH 7 FOLLOW ('isa', 'instanceof')
```

Figure 2.24: Query used in the example

Querying

In this section we show how all the discussed elements are combined to compose geographic recommendation queries. Figure 2.24 shows the complete recommendation query that integrates geographical, textual, social, and semantic aspects of the integrated graph in our example.

The first line of the query (Figure 2.24) is responsible for selecting the nodes of interest. The query in the figure simply selects all restaurants, which are the POIs, in the database. A more elaborated query could include predicates with restrictions based on average price, ratings, etc.

In the “Ranked By" clause we combine several metrics that were built using our algebra. Each metric generates a score (between 0 and 1), and the scores are combined based on the weight defined at the beginning of the metric description (in this query all weights are 1, meaning that an average will be produced).

In Figure 2.24a, the geo_relevance metric is assessed between each restaurant and the node that represents the reference point (usually user’s location). The highest-scoring POIs are the ones closest to the user, respecting the road directions (not geographic distance). Table 2.11 (column a) shows the resulting scores for the example with only the geo_relevance metric in the ranking clause.

Figure 2.24b is the keyword part of the query. The mechanism that allows keyword queries in the system is outside the scope of this paper. More details can be found in [28]. Table 2.11 (column b) shows the resulting scores for the example with only the relevance metric for the keyword in the ranking clause. Mama’s restaurant, which has two of the keywords in the description, is the highest ranking result. Note that, as in most information retrieval systems, our metric favor words with high information content (i.e. less common words).

The third metric (2.24c) assesses the reputation of the POIs in the social network. The metric will favor restaurants that are liked by many people in the network, but will boost the score when these people are themselves highly influential. The results for the query with this metric only can be seen in Table 2.11 (column c).

The last metric (2.24d) assesses the relevance to the concept ‘Casual Dining’ in our ontology. The metric is also represented in Table 2.11. Table 2.11 also shows the results for the complete query in Figure 2.24. The result is shown in the column "a+b+c+d", representing an average of the scores for each metric (which is automatically calculated

and outputted by the query processor). Column "6a+b+c+d" shows results of the same query with a change on the weight of the geo_relevance metric. In this case, the weight is changed to 6, giving more priority for proximity. The result is a reorder in the ranking favoring POIs closer to the user. This example is meant to show how easily the recommendation can be tweaked by simple changes the query.

The example query is able to integrate diverse information types and provide non-trivial recommendations based on a few lines of our language. Many other parameters and predicates could be included to adapt the query to specific scenarios.

name	a	b	c	d	a+b+c+d	6a+b+c+d
Mama's	0.7	1.0	0.45	1.0	0.79	0.74
Erreu	0.85	0.35	1.0	0.27	0.62	0.75
FlashFast	1.0	0.0	0.62	0.08	0.42	0.74

Table 2.11: Scores for different combinations of the ranking metrics.

2.6 Conclusion

Complex network analysis has become an important requirement in a wide range of modern applications and research fields. Analyses are typically undertaken on software packages that offer no data management capabilities. Databases, on the other hand, offer adequate models and storage mechanisms for a variety of applications, but do not reach the level of expressivity necessary for capturing the emergent properties typical of network analysis tasks.

In this paper we presented our efforts towards integration of the databases and complex network areas. Our vision is that a database management system with adequate querying capabilities can offer several advantages for complex network analysis. We have presented our Complex Data Management System (CDMS), which offers a query processing model that can both capture a wide range of network properties and be integrated in a declarative query language.

The proposed model is based on a relational operator (Beta) that is parametrized and combined to build measurements than can be integrated as ranking clauses in a query language. We used examples and use cases based on real and synthetic data to showcase the capabilities of the model. Other use cases, from nursing diagnosis to enterprise data management, have been reported in [33, 35]. We believe that offering streamlined means to capture information about network dynamics has the potential to popularize this type of analysis and, in many cases, blur interdisciplinary lines.

We have now a working prototype of the system that runs the beta algebra over a graph database[5]. We are currently evolving the proposal in both theoretical and implementation fronts. We are finishing the formal definition of the algebra and assessing its expressivity (i.e. capacity of representing high-level information needs).

[5]Neo4J: http://neo4j.com/

In terms of system development, we are assessing query optimization techniques, testing different heuristics and the necessary supporting data statistics. For cases where a precise answer takes an impractically long time, we are developing a sampling-based version of the Beta operator. Running the operator over a sample of the data is acceptable in many situations, and can also serve as a testbed for the ones that require precise answers.

An important goal of our proposal is to enable highly scalable network querying. A natural step toward scalability is to implement a distributed version of our system. Distributed infrastructures divide data and processing among several computers connected through a network. We are planning on extending existing distributed graph processing frameworks with our query processing model, combining the flexibility of a query-based interface and the scalability of distributed execution.

Chapter 3

A Relational Algebra for Graph Analysis

3.1 Introduction

Graph analysis has become an important tool in diverse fields. Social, transportation, communication, and biological networks are examples of information often organized as graphs, which require specific tools and algorithms for proper data analysis.

The area of Complex Network Analysis [21] has advanced in the last decades and has produced several models, algorithms and techniques to study natural and human-made networks. In terms of database support, there has been a strong acceleration in the usage of graph databases and query languages, as well as in the development of the underlying algorithms and mechanisms. There is, however, still a big gap between graph query languages and the analysis techniques. Current graph query languages offer little support for the type of analysis required for complex networks. Such a gap is not present, for example, in traditional relational databases, which support query languages that offer aggregation operations that are the basis of more sophisticated multidimensional analysis.

Our goal is contributing towards bridging this gap. We aim at developing data management and querying mechanisms that offer a better support for network analysis. In this paper, we present our first steps towards creating an algebra that offers a graph-based aggregation operation. We expect this algebra to be the basis of more expressive queries, supporting declarative and interactive graph data exploration.

Our proposed algebra is based on Codd's relational algebra [17]. This has several advantages: (i) it provides a well established theoretical basis; (ii) it allows the combination of traditional relational operations alongside our proposed graph operation; (iii) it is a de facto standard in database research. Having an underlying algebra allows a better understanding of the semantics of the query language and, most importantly, allows the definition of rewriting rules for execution plan optimization. As an extra benefit, relational algebra compatibility also simplifies implementation in current relational database systems.

In simple terms, our goal with the algebra is to provide graph-based aggregation of values. The core of our proposal is the beta (β) operator, which encapsulates the

graph traversal procedure and allows parametric control over the aggregation of values. We see graph-based aggregation as a generalization of relational aggregation over sets. Considering that most useful relational aggregations perform joins before applying an aggregation operation, we adopt this pattern of first deriving relationships between the data (joins or graph traversals) and then aggregating the values as the basis for our new constructor.

Combining the advantages of relational query languages and graph analysis, the proposed algebra allows the construction of queries involving subgoals such as: obtain a subgraph based on given nodes properties and edge labels; calculate the reputation of the nodes in the subgraph; combine the reputation and the average distance to a given reference node in the general graph; order the resulting nodes based on the final score. In our envisioned scenario, such queries would be starting points for deeper explorative analysis, with goals such as: analyze the average node degree for the top-k nodes returned in the query; obtain the average value for a certain attribute for the bottom-k scoring nodes, etc.

This paper is organized as follows: Section 3.2 describes related work and fundamental concepts. Section 3.3 presents the definition of our algebra alongside with query and execution examples. Section 3.4 briefly describes our current approach for querying and initial query rewriting rules for execution plan optimization. Finally, Section 3.5 concludes the paper.

3.2 Related Work

There is great diversity of graph query languages, which have pushed the boundaries for more expressive constructors [64]. Graph query languages are often based on conjunctive regular path queries (CRPQs). CRPQs are the basis for several graph languages, such as GraphLog [18] and SPARQL[1]. Recent developments have extended CRPQs in order to allow constraints over path properties. These types of queries have been described as extended conjunctive regular path queries (ECRPQs) [8]. ECRPQs also allow paths to be returned as query results. These queries are all focused on data selection and support only the simplest cases of analysis.

Query languages such as in GID [61] allow ranking based on pre-calculated metrics of importance which capture the dynamics of a snapshot of the network. Our goal is to enable a higher level of on-demand analysis in graph query languages. To that extent, we are working towards an algebra that can handle graphs and aggregations over paths. The goal is to use this algebra to build more flexible query languages.

Several algebras that support graphs have been proposed. To our knowledge, the algebra that is closer to our goals is the alpha-algebra [1], which also serves as inspiration for the name of our operator. The alpha operator derives the transitive closure for tables that express self-relationships – e.g. CONNECTS(from, to, distance). The algebra supports aggregation and filtering over paths through the delta (Δ) attribute, which is an internal relation containing each path history in the result set. Conceptually, the alpha operator

[1] http://www.w3.org/TR/sparql11-query

has two main characteristics that make it unsuited for our needs: the operator always processes until reaching fix point, and there is a single point for value aggregation. In our algebra, we add more flexible stop conditions and split aggregation in four suboperations (Section 3.3). We also change the underlying model and add several elements for querying convenience. The changes allow more analysis algorithms to be represented in the algebra as well as providing more opportunities for optimization based on query rewritings (Section 3.4).

Frasincar et al. [26] propose an algebra for RDF, with some operators inspired by relational algebra. The algebra enables both querying and construction of RDF models. Although the algebra shares many of the goals in this paper, the focus is on the complete RDF-S model, which incurs considerable complexity when compared to our simpler graph model. Most importantly, the proposed algebra does not support graph-based aggregation, our main focus.

In a more recent and simpler proposal, Cyganiak [20] defines a relational algebra for the SPARQL language over the RDF model. The authors rely on a global reference table containing RDF triples (subject, property, object) as basis for the operations. We use a similar strategy for our underlying model as we employ global tables for nodes and relationships to represent the graph in the database. Their proposal, however, does not deal with aggregation or analysis issues.

The demand for graph analysis in large scale has motivated the development of several frameworks for distributed computation: Google's Pregel, the first NoSQL implementation in that scale, was followed by diverse proposals including GraphLab [46] and GraphX[2]. These models focus on parallelization of the API operations and do not provide declarative languages as means for data querying. GraphX shares some of our motivation since it aims at simplifying mixed analysis that include graphs and relations. The focus is, however, on general parallel computation and not on querying. Our goal is to provide a higher level, declarative language for graph querying and analysis, allowing a more interactive and explorative interface with the user. Although we are not concerned with distributed computation at the moment, we believe that would be a natural evolution for our framework.

The Complex Networks [21] field is a prominent area that would benefit from query-base graph analysis. Complex network formation is based on localized phenomena, which in a global scale determines emergent behavior that cannot be assessed based merely on the analysis of parts of the system. The researchers employ a variety of models and algorithms to derive knowledge from the structures. Among the frequently used algorithms are the well known PageRank [13] and HITS. Most of the analysis in the field is done using *ad hoc* applications with no database support.

The algebra that we propose in this paper has been developed in the context of our Complex Data Management System (CDMS) [35], which aims at providing a database-like framework for complex network analysis and management. Appropriate query languages (and underlying algebras) would allow a more fine-grained, exploratory and interactive interaction with the networks.

[2]https://spark.apache.org/graphx/

3.3 The Beta-Algebra

In this section we describe the requirements for our algebra and present the beta operator alongside example queries.

3.3.1 Requirements

The basic requirements for the proposed algebra are:

Allow traditional relational aggregation: given the widespread use and familiarity with relational database queries, it is important to build on and leverage this foundation. Moreover, a graph analysis workflow often contains routines that are typically relational (counting, ranking, statistics, etc).

Enable aggregation over path traversals: Most graph analysis tasks involve aggregating values along graph traversals. It is important to allow flexible and case-specific aggregation functions that are applied as the graph is traversed.

Preserve the closure of relational algebra: It is important that the aggregation operates over and produces relations. This allows compatibility with relational algebra properties studied for decades, as well as making the language more flexible and composable.

Support flexible selection of nodes of interest: To enable explorative analysis over graph data it is important to have efficient means to filter nodes and relationships that are to be part of the analysis. A declarative language would naturally allow this type of flexibility.

Offer means to express subgraphs to constrain the analysis: Currently, most complex networks analysis is done over graphs as a whole. We believe this is highly associated with a lack of convenient means to select a subgraph of interest and apply the analysis over the selection. An appropriate algebra needs to enable graph-based constraints over the execution of the algorithms.

Rewriting rules for cost-based optimization: One of the main advantages of building query languages over an underlying algebra is the possibility of rewriting the queries for faster execution. Including graph-specific operations allows the specification of semantically sound rewriting rules for the graph setting.

Other requirements, not covered in this paper but that we want to address soon are:

Include convergence criteria for operation termination: This would allow the implementation of complete versions of popular graph analysis algorithms (e.g. PageRank).

Provenance/Path information for query results: Like in the alpha algebra with its delta attribute, adding information about paths traversed by the operations allows more flexibility for query composition and allows optimizations such as avoiding loops in graphs.

3.3.2 Data model

In this paper we use a simple interpretation of graphs in the relational model. In this model, a labeled property multigraph[3] [55] G is represented in two relational tables, V (vertices or nodes) and E (edges or links). The tables contain attributes representing the

[3]a graph with labeled links and properties for nodes and links

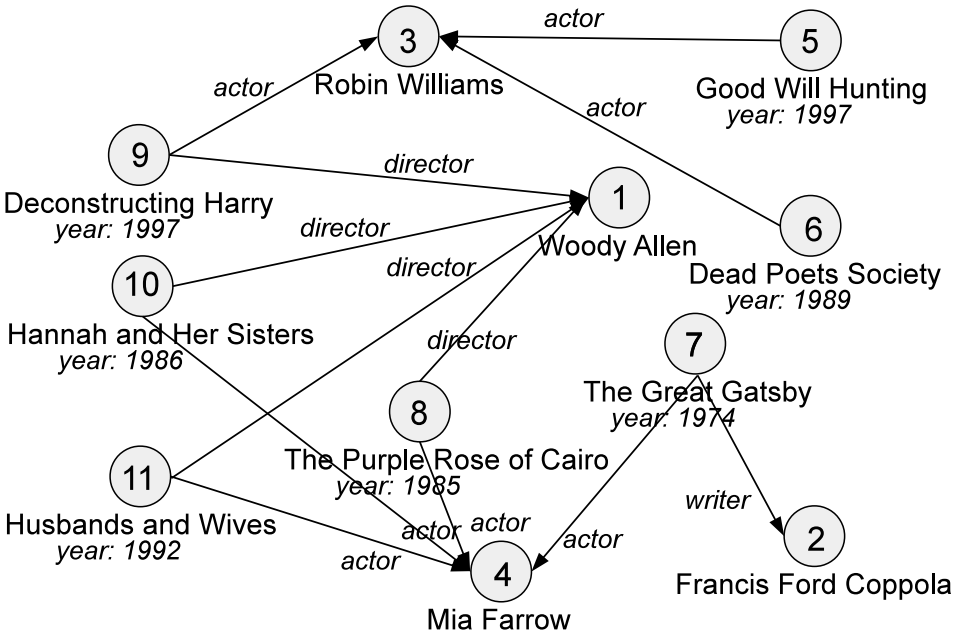


Figure 3.1: Subgraph from a movies dataset

properties for all nodes and edges (typically highly sparse tables). The V table is in the form $\langle node, a_1, a_2...a_n \rangle$, where *node* represents the node id in the database, and $a_1, a_2...a_n$ are node properties.

The E table is in the form $\langle source, dest, label, a_1, a_2...a_n \rangle$. *source* and *dest* are ids representing the connected nodes for a given edge (the order implies the direction), *label* is the label according to the labeled property graph model, and the properties are as in the V table.

This data model is the reference for the proposed algebra. However, as with any other model, it does not impose implementation constrains (e.g. it can be implemented over a pure graph database and use efficient structures to represent the sparse tables). Our own implementation stack does not include any traditional relational component.

In the following paragraphs, the beta operator will be presented informally, in increasingly complexity as parameters and suboperations are introduced. Since the operator can be seen from either a graph or relational perspective, we will use equivalent terms interchangeably (e.g. join and traversal step). The semantics, however, is always relational.

3.3.3 The beta operator

Much like the alpha operator, the beta operator assesses recursive relations in the database. However, the main goal is not to derive transitive closure. Instead, the focus is on data aggregation along the traversal of the relations. In its simplest form, the beta operator performs a single join between a single column source table and the table E. An union operation is then applied to aggregate the original and new nodes. For the sake of readability, we omit extra join, projection, and renaming operations required to maintain the original schema after the execution of the operator. In a graph interpretation, the beta operator augments an initial set of nodes with all of their neighbors. For

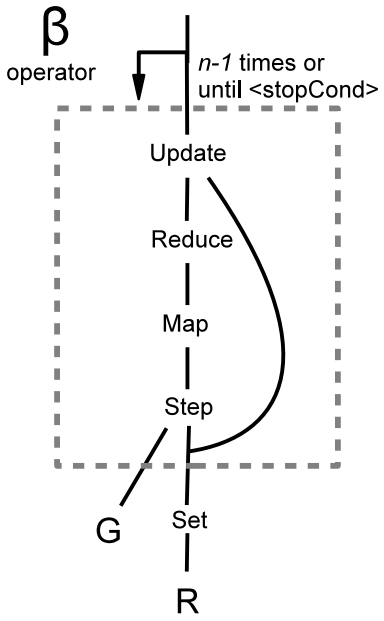


Figure 3.2: Simplified execution tree template

example, based on the graph in Figure 3.1, the beta operator applied to a source table containing one column with node ids $\{9, 10\}$ would produce $\{9, 10, 3, 1, 4\}$. This operation is represented as $\beta(\sigma_{id\in\{9,10\}}(V))$.

In general, we represent the beta operator as $n\beta_p(R)$, with $p = \langle s, dir, set, map, reduce, update, C\rangle$. Several parameters are used to control the behavior of the beta operator. s is the join condition, which accepts Boolean expressions just like its relational counterpart. n determines the number of recursive calls to the operator (i.e. consecutive joins). $dir \in \{inbound, outbound, both\}$ determines the order of the relations in the joins (or the direction of the graph traversal operations). The optional parameter *source* determines, for tables with more than one node column, the column over which the beta operator operates (the default is the fist column).

The operator keeps the algebra closed since (i) it always produces a table with at least the same columns as the input table, and (ii) it can be defined using standard relational algebra with aggregation. The design choices (such as encapsulating the table E inside the beta operator) are for convenience and to focus on what we think are the most important aspects of an aggregation operation. This aspect is inspired by the introduction of the relation join that despite being a redundant operation has directed the focus of the research on properties and optimizations of a central element of the model.

The most important elements of the beta operator are the aggregation suboperations. To allow full control of the computation as the graph is traversed, we define four operations: *set*, *map*, *reduce*, and *update*. *set* is a function that attributes a value to a new column before the join (traversal) operation is performed. *map* calculates a new value based on each node in the source relation. The new values are associated with the neighbors after the join operation. *reduce* is a function that aggregates over values for the same source node (equivalent to a group by). Finally, *update* redefines the aggregation values for the source nodes before the union. Figure 3.2 shows a simplified execution tree

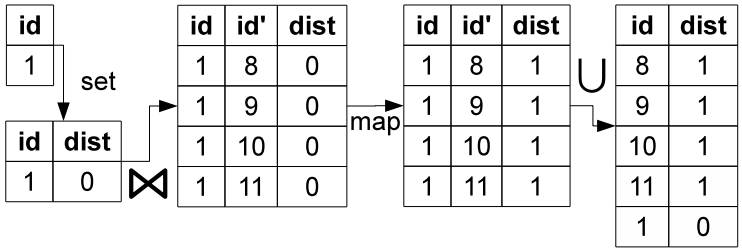


Figure 3.3: Snapshots of the resulting tables inside the beta operator for the first iteration of the example query.

for the beta operator. As an example, the query:

$\sigma_{type=movie}(5\beta(\sigma_{id=1}(V)))$,

with {set: dist=0, map: dist=dist+1,dir=both}, obtains distances to movie nodes that can be reached from the initial node 1 (director Woody Allen) in up to 5 steps. Figure 3.3 shows the partial tables after the suboperations for the first iteration of the query. The query:

$\sigma_{type=movie}(5\beta(\sigma_{id=1}(V)))$,

with {set: dist=0, map: dist=dist+1, reduce: dist=MIN(dist)} obtains the minimum distances in the same setting.

Another parameter is the optional graph constructor C. Its purpose is to specify the effective subgraph for the beta operator, constraining the search space for the traversals. The reference points for the constructor are the input nodes. C has its own parameters: *radius* is the maximum distance from the reference points; s is the edge selection expression (similar to the join condition in the beta operator). Nodes and links beyond the radius or that do not match the selection are ignored by the beta operator. The addition of the graph constructor in the algebra is also for convenience sake. The same effect could be obtained by a series of joins and selections over the E table. The query:

$5\beta(\sigma_{type=movie}(V))$,

with {set: $rank = 1$, map: $rank = rank/|e.out()|$, reduce: $SUM(rank)$, reset: $rank = 0$, C: $\{radius : 3\}$}, is a simplified PageRank algorithm executed for five iterations (not until convergence) over a graph of radius 3 around the source nodes in R. $|e.out()|$ represents the number outbound nodes (that can be obtained with traditional algebra aggregation). If the number of source nodes is known and its inverse is used in set function, the query obtains, for each node in the constructed graph from C, the probability that a random walker would stop at the node after five steps.

Other parameters and functionalities that we want to investigate are (i) specifying stop conditions for the beta operator, including simple test and convergence properties, (ii) recording path traversal information in a delta attribute, with functions that operate over it (similar to the alpha-algebra), and (iii) a modifier equivalent to the SQL *distinct*, that uses the delta attribute to avoid the computation of cycles.

3.4 Querying and Optimization

In this section we present our initial attempts with query language design and rule-based optimization.

3.4.1 Querying

We are developing the algebra presented here to support the query language that we have been developing as part of our CDMS system. The language that we are currently using is an extension of popular graph queries as shown in Figure 3.4. The language shows the type of queries we are envisioning, although it is less expressive than the algebra that we are proposing. We plan to design a more expressive language following the definition of our algebra.

In the initial language, the graph-based aggregation is expressed in a RANK BY clause. The clause accepts metrics that aggregate values over graph traversals as in the algebra presented. For example, Relevance is a generalization of the notion of relevance in Information Retrieval, attributing higher scores to elements that have multiple and more specific connections (paths). This metric can be represented in our algebra by a beta operator with aggregation functions $\{set : score = 1, map : score = (score * 0.9)/e.out(), reduce := SUM(score)\}$. Details about the metrics and queries can be found in [33].

Figure 3.4a shows a query that retrieves actors whose careers are strongly correlated (relevant) with the director Woody Allen (id 1). Based on the subgraph in Figure 3.1, Mia Farrow would have a much higher score than Robin Williams. Another interesting query, that includes traditional relational aggregation, would be to find the pair (actor, director) with the maximum mutual relevance. This type of query would be hard to express using current graph or relational queries.

Figure 3.4b shows a query that we used for a nursing diagnosis task. Possible diagnosis are ranked based on their connections with the symptoms identified in the patients. This query contains a combination of two different metrics (Relevance and Connectivity).

3.4.2 Rewriting rules

An important motivation for introducing a new algebra is to better understand the computation complexity and define rewriting rules for query optimization. This work is still ongoing and we will only show some first examples for illustration.

The first rule is about the bidirectionality of the analysis. A beta operation that starts on a group of nodes and selects another group of target nodes can be reversed (changing the directions of the allowed edges). Reversing the direction can, in certain cases, reduce the search space by avoiding dense regions of the graph. For example, a three step undirected traversal from node 1 to node 6 in Figure 3.1 visits 9 nodes, while the traversal from 6 to 1 activates only 4 nodes. This rule can be represented as:

$\sigma_a(\beta(\sigma_b(R))) \equiv \sigma_b(\overline{\beta(\sigma_a(R))})$, where $\overline{\beta}$ represents the β operator with inverted directions (parameter dir). We are omitting, for sake of simplicity, extra joins and renamings that would make the outmost selections equivalent. We have tested this strategy for the

a)
```
SELECT DISTINCT ?actor WHERE {
  ?film movie:director director:8501 .
  ?film movie:actor ?actor .
  ?film movie:initial_release_date ?date .
  FILTER ( fn:starts-with(?date, "199") ) }
RANK BY RELEVANCE OF ?actor TO director:8501
```

b)
```
START diag=node(*)
  WHERE has(diag.Type) and diag.Type = "Diagnose"
  RETURN diag
RANK BY
  1 RELEVANCE OF diag TO node($patient) ,
  2 CONNECTIVITY OF diag TO node($patient)
```

c)
```
SELECT DISTINCT ?product
 WHERE { ?product :type :Product .
  FILTER NOT EXISTS (:bob :purchased ?product) }
RANK BY RELEVANCE OF ?product TO :bob
 FOLLOW (:friendsWith, :purchased)
 DEPTH 3 DIRECTION BOTH
```

d)
```
SELECT ?species
 WHERE { ?species :type :MarineSpecies }
RANK BY
 3 INFLUENCE OF ?species FOLLOW :preysOn
   DIRECTION OUTBOUND,
 1 RELEVANCE OF ?species TO :CoralReefBiome
```

Figure 3.4: Query examples. a) extends SPARQL and b) extends Cypher

query in Figure 3.4a against a comprehensive movie database [37]. The execution was reduced to half the time of the baseline query. The initial results, using a different formalization, were published as a technical report [31]. We still have to assess the subset of aggregation functions that allow the use of this rule. In practice, this rule requires cost-based planning, which we have not implemented yet.

Another rule, regarding compositionability of operators, combines aggregation functions that are applied over the same data by different beta operators. It can be represented as:

$\sigma_a(\beta_p(\sigma_b(R))) \bowtie \sigma_a(\beta'_{p'}(\sigma_b(R))) \equiv \sigma_a(\beta_{p \bullet p'}(\sigma_b(R)))$, where p represents the tuple of aggregation functions for the operator and $p \bullet p'$ combines the respective functions. The functions in p and p' must not make conflicting operations over the same attributes. We expect this type of rewriting to be very common, as multiple metrics can be used for the same target nodes (as in the query b in Figure 3.4). We have not yet implemented this rule in the system.

We have also explored options for speeding up queries beyond rule-based rewritings. We have tested, with positive results, caching paths between nodes for metrics that need to traverse the paths in both directions. We also explored a few query approximation strategies for specific metrics. These experiments are also reported in [31]. Another possibility is to materialize graphs constructed from the parameter C for use in multiple beta operators in the same query (rewriting the execution tree to take advantage of the materialized graph).

3.5 Conclusion

Graph analysis has become an important requirement in a wide range of modern applications and research fields. This type of analysis is currently highly specialized, employing ad-hoc applications or complex distributed frameworks. Graph databases offer little support for graph-based aggregations that would allow for query-based analysis.

Here we presented our ongoing work on the beta-algebra, which is intended to allow graph-based aggregations for declarative query languages. The algebra extends the relational model to support graph traversals and allows the control of several aspects of the aggregations.

We have shown examples of the use of the algebra and how it fits in our broader goal of developing data management and querying mechanisms specific for graph/complex network analysis. Also, we presented initial tests and directions for query optimization based on execution plan rewriting, along with our preliminary experimental results.

Our algebra allows more expressive querying when comparing to pure relation aggregation and CRPQs. The increased expressiveness enables explorative analysis and more interactive data manipulation. It also enables seamless integration of relational and graph-based analysis, which is a common application scenario. We believe the algebra is a good basis to build expressive query languages as well as useful optimization strategies.

Ongoing and future work include the expansion of the algebra to allow more flexible stop conditions (e.g. convergence), accumulation of traversal history (such as with the delta attribute in alpha-algebra), definition and tests of rewriting rules, specification of a query language that can take full advantage of the algebra, and implementation of a distributed query processor.

Chapter 4

Application-level integration in the CDMS

4.1 Introduction

Digital data availability has grown to unprecedented levels and surpassed our capacity of storage and analysis. This has led to the Big Data and NoSQL movements, aiming at tackling the increasing demands for scalability. A parallel development regards the proportional increase in data complexity. Capturing and processing greater amounts of data produces information that is correlated in diverse and intricate ways. Furthermore, recent developments in processing power, modeling, and algorithms enable the implementation of systems that better explore the increased complexity of data. All these factors influenced and enabled the dissemination of social networks and initiatives like the Linked Open Data[1].

Realizing the potential of the relationships inside the interconnected data and developing the means for their analysis fueled the development of the areas of complex networks [22] and link mining [27]. Related techniques have been applied in several scenarios, such as systems biology, neuroscience, communication, transportation, power grids, and economics [21].

In this article we aim to show how the focus on relationship analysis is important for institutional data and applications. Assessing properties of how data is correlated is the basis for several tasks, such as document retrieval, item recommendation and entity classification. We, therefore, advocate the vision that institutional data can be seen as a big complex network and, most importantly, several modern and commonplace applications can be specified in terms of link analysis tasks. A unified, relationship-centric framework for data and applications can enable a new level of integration, encompassing data from diverse sources (e.g. structured and unstructured) and applications (e.g. information retrieval, machine learning, data mining). This application-level integration allows developers to rely on common models for data interaction, simplifying development of applications with information needs that span multiple querying paradigms.

Institutional data and applications have, however, several requirements that make it

[1] http://www.w3.org/standards/semanticweb/data

hard to apply link mining techniques directly. The size of the data, its dynamic nature, and heterogeneity are not considered in traditional approaches. The focus of complex network techniques is typically on homogeneous networks with a single type of relationship (e.g. social networks), employing off-line algorithms to assess snapshots of the data. Modern applications, on the other hand, favor online access to subsets of the data, and must handle data heterogeneity seamlessly.

Here we introduce the Complex Data Management System (CDMS), which aims at providing query-based interaction for complex data. The proposed query model allows users to specify information needs related to the topology of the correlations among data. It also offers management mechanisms that are more adequate to the increased importance of the relationships in the data.

The increased expressiveness in the new framework provides a better match to the requirements in the described institutional settings. The online query mechanism allows the composition of queries that explore diverse aspects of how data is correlated. This not only allows the same query model to be used in diverse application scenarios, but also allows queries that encompass concepts from multiple paradigms and data sources, such as in queries like "retrieve documents related to the keyword query 'US elections' and the topic *politics*, written by democrat journalists, ranked by relevance to the keyword query and reputation of the author".

To enable this type of interaction, the underlying institutional data must be integrated. Here we describe how we are fostering web standards to enable the required integration. Once the data is integrated, the CDMS is used to provide the proposed querying infrastructure. To tackle the integration of data models, we employ an RDF graph that interconnects data from diverse sources and models. The flexibility of graph models allows easy mapping from otherwise incompatible models (e.g. unstructured text and structured databases). Figure 4.1 contextualizes the elements in our proposal: several data sources are integrated in a unifying graph, which allows our framework to enable a more expressive interaction between users and data.

As for integration at the query and application level, we acknowledge the importance of the Information Retrieval (IR) and Databases (DB) fields – which dominate data-driven applications in current settings – and describe how our new query model, which leverages complex network analysis, unifies concepts from these areas. To enable our query model over the unifying graph, we reinterpret querying concepts from diverse areas into graph analysis tasks. We implement this model in a new query language called in* (in star), which is an extension grammar for existing languages such as SPARQL.

We also address architectural issues related to the integration process, introducing the concept of *mappers*, which aim at simplifying relationship management. Mappers are similar to *stored procedures* in databases, triggered when nodes are created to carry customized tasks such as adding appropriate relationships or even other new nodes. Our *mappers* are integrated in the query model for further flexibility.

We aim at contributing towards a more unified and expressive interaction between users and data through this relationship-centric querying and data management framework. Experiments with real data are presented to demonstrate the expressiveness and practicability of our framework.

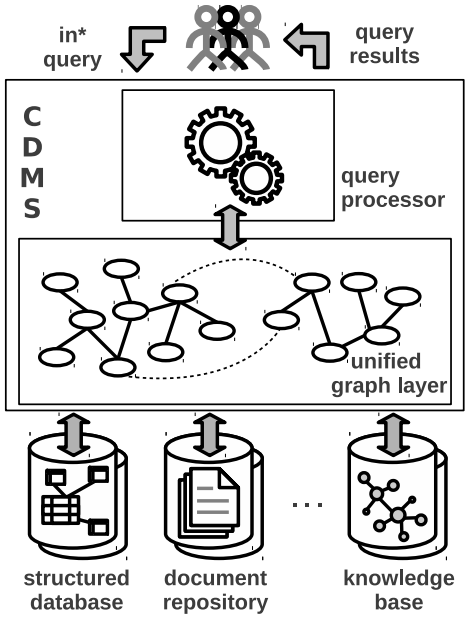


Figure 4.1: Architecture of a CDMS deployed in a data integration scenario

This paper is organized as follows: Section 4.2 discusses the new challenges for the current heterogeneous technological landscape. Section 4.3 introduces the Complex Data Management System, which is the basis for our integration approach. Section 4.4 describes the requirements for data access and model integration in our framework as well as issues related to query model integration, a fundamental concept in our proposal. Section 4.5 details our integrated query model and discusses usage scenarios. Section 4.6 introduces related data management issues and describes our *mapper* mechanism. Section 4.7 demonstrates experiments for our query language and the use of mappers in scenarios based on a large and interlinked database of movies. Section 4.8 contextualizes related work in respect to our proposal. Finally, Section 4.9 concludes the paper.

4.2 New challenges for institutional data and application integration

The new scenario of overwhelming accumulation of information has a profound impact on the way we organize and manipulate data. We have seen an increased specialization of database back-ends and data models to respond to modern application needs: text indexing engines organize data on the Web, standards and models were created to support the Semantic Web, Big Data requirements stimulated an explosion of data representation and manipulation models labeled under the NoSQL umbrella. This complex and heterogeneous environment demands unified strategies that enable data integration and, more importantly, cross-application, expressive querying.

Although data integration has been an active research topic for many decades, most proposals depart from environments that do not take into account the modern diversity of technological infrastructures. Federated databases, for example, usually adopt the relational model to integrate data sources, with limited capabilities when dealing with semi

or unstructured data. Similarly, in typical OLAP implementations, the benefits of integration are restricted by the adopted query model: data analysts may answer complex questions, but there is no direct benefit to other applications inside the institution. For example, Web developers cannot leverage the potential of the integration in their implementations of recommendation systems because they typically work on very different query models. Similar issues also appear in other contexts, such as the Semantic Web, which brings great benefits for data integration but querying capabilities do not match the diversity of Web applications.

A level of integration that covers a wide range of data models and, more importantly, data query models would not only allow applications to incorporate more relevant information, but would also allow more expressive queries that combine elements from different querying paradigms. For example, consider the following queries:

- retrieve documents related to the keyword query "US elections" and the topic *politics*, written by democrat journalists, ranked by relevance to the keyword query and reputation of the author;
- retrieve employees relevant to a given project ranked by their reputation among peers;
- retrieve profiles of people over 30 years old, ranked by similarity of hobbies on their profiles to hobbies on my own;
- retrieve products not yet purchased by the client Bob that are relevant to him.

These queries cover a broad range of data models (e.g. unstructured documents, relational, graph) and applications (CMSs, social networks, recommendation systems). The queries also combine concepts from diverse query models, such as relational predicates, keywords, ranking, and metrics of relevance and reputation. These and similar queries show up in many situations in typical institutions, both for internal, administrative purposes or for Web applications developed for external use. Answering these queries in current infrastructures typically demands substantial amount of resources and engineering to design ad-hoc subsystems.

To provide an overarching approach for querying, data model integration and query model integration must be tackled simultaneously. Querying is especially challenging, given the diversity of the data and the complexity of the information needs. The central observation underlying this article is that these issues can be mapped into complex network analysis tasks. Several tasks typically associated with the information retrieval and machine learning fields – including document retrieval, recommendation, and classification – draw inferences from how information pieces are correlated. Even though the correlations are often not explicit, it is intuitive to consider the data as a graph and notice the importance of the relationships and the underlying topology for each task. Our hypothesis is that an expressive query model that can capture topological properties in query time can be used to integrate these information needs in a single conceptual framework. We aim to show how the CDMS can be used in these scenarios, providing expressive querying and data management mechanisms that are appropriate to the heightened importance of relationships in the described scenarios.

4.3 Complex Data Management

The database framework used in our proposal is being developed to tackle issues associated with Complex Networks. In a complex network [22], the patterns defined by the interconnections are non-trivial, deviating substantially from cases where connections have the same probability (e.g. lattices or random graphs). The techniques developed for complex network analysis have become important resources in diverse fields such as systems biology, neuroscience, communication, transportation, power grids, and economics [21]. These areas deal with complex structures that requires specific techniques for analysis. In all cases, relationship analysis is a major aspect for knowledge acquisition. Typically, these structures generate emergent behavior, which are determined by the complex interactions among their simple constituent elements.

As a result of increased capacity of data storage and processing, these scenarios have come forth in other areas, such as enterprise data management, our focus on this paper. A typical institution nowadays stores and processes many textual documents alongside traditional structured data, communication and transaction records, and fast changing data about market and competition. These data are highly interlinked, by design or through intricate (and potentially imprecise) data analysis procedures such as named entity recognition, sentiment analysis, and recommendation systems.

Our CDMS is aimed at enabling querying and management of what we define as *complex data*. Complex data is characterized when relationships are central to data analysis. In these cases, the graph formed by data entities (nodes) and relationships (links) present properties typical of complex networks. The CDMS is aimed at providing adequate support for handling and querying complex data. It differs from typical DBMSs in four main aspects: (i) data model, (ii) query language, (iii) query evaluation, and (iv) data management mechanisms. Each of these items is described below.

- **Data model:** The data in target CDMS applications typically do not comply to predefined schemas. The high number and diversity of the relationships require a model where relationships are first-class citizens. Graph models are obvious choices in these settings. Their flexible modeling characteristics enable easy mapping of most types of data. Nodes with immediate access to neighbors is also an important feature for the type of computation involved. The CDMS framework adopts weighted edge-labeled property multigraphs to encode complex data. In this article, we leverage the RDF model to integrate institutional data.

- **Query language:** Our CDMS query language is intended to be flexible enough to allow correlation of data when little is known about how they are linked and organized. We developed a declarative query language that extends existing graph languages by introducing ranking based on a set of flexible correlation metrics. The ranking metrics proposed are: relevance, connectivity, reputation, influence, similarity, and context. The proposed language is designed as an extension for existing graph languages. In this article we show how SPARQL can be extended to enable the new query model.

- **Query evaluation:** Our abstractions for query evaluation fully support the query language while allowing for under-the-hood optimizations. We adopt a variation of the spreading activation (SA) model as our main abstraction for query evaluation. The model allows the specification of the ranking metrics that are the basis of our query language. The SA mechanism is based on traversing the network from a initial set of nodes, activating new nodes until certain stop conditions are reached. By controlling several aspects related to this activation flow, it is possible to infer and quantify the relationships of the initial nodes to the reached ones.

- **Data management mechanisms:** Relationship creation is an important and defining operation for the described application scenarios. For example, several text indexing tasks, such as topic modeling, derive relationships between the text and more general concepts. In machine learning applications, elements are associated with features or classification categories, for example. In our framework, the creation of relationships is encapsulated in mappers. Mappers are very similar to *stored procedures*. What sets them apart are (i) their integrated use in our ranking queries, and (ii) how they are hooked in the databases's API so that any new data that matches the mapping criterion is passed through appropriate mappers.

The CDMS offers an architecture where relationships are central elements of the database. It enables queries to tap into properties derived from topological characteristics of the underlying graph. CDMS's new query model and management mechanisms allow for new levels of expressiveness for several tasks, simplifying integration of data from diverse sources and allowing distinct applications to employ the same query model over the integrated data.

4.4 Data and query model integration

The level of integration that we aim at requires solutions to three main issues: (i) unified data access, so that queries have access to all data, (ii) unified data model, so that queries can reference data from diverse formats; and (iii) unified query model, so that applications can have a single interface for interaction with data. This level of integration allows applications to be based on the same underlying models to interact with data, what we call application-level integration.

4.4.1 The local unified graph

Institutions face similar challenges to that of the Web: data produced by diverse groups in distinct contexts must be integrated to allow for more capable and outreaching applications. Although several research and products were developed to address these issues, we argue that revisiting this problem through the perspective of the new developments in applications and standards of the Web would allow for a more adequate interaction with modern institutional data.

The Semantic Web initiative has advertised the benefits of treating the Web as an integrated Giant Global Graph (GGG) [9]. Similar benefits could be achieved inside

institutions by integrating all their data in a Large Local Graph (LLG). A LLG lacks the diversity and magnitude of the GGG, but it allows higher levels of control over data and local processing power, enabling better semantic integration among distinct data sources and more expressive querying. Another advantage of creating LLGs is that it facilitates transference of information to and from the GGG.

The framework proposed here assumes an underlying LLG. Although our solutions have interesting applications also in the context of the Semantic Web, we require levels of integration and processing power that are not currently available for the GGG. We, therefore, focus on institutional data but expect that in the future technological advances would allow similar interactions in a broader context.

A LLG is meant to integrate a broad range of data from an institution. Aggregation of external data from the GGG would also be important in many scenarios. Integrating data across domains and models is important to allow rich correlation queries between diverse data elements. The graph model is suiting for this scenario. Its simplicity and flexibility allows the representation of most of the popular data models [6, 10]. Figure 4.2 shows a graph containing data derived from documents and relational databases (more details on the mapping in Section 4.4.2).

Here we employ the RDF(S) model for the LLG for several reasons: it is a stable and popular model, it implements a flexible graph model, classes facilitate the mapping of other models (e.g. object, relational), integration with other standards (e.g. URI, XML), standardized query language (SPARQL), simplified data sharing, etc.

It is important to emphasize that the strategy to create the unified graph is environment-specific. Although we provide general guidelines on how data should be represented as nodes and edges, our framework assumes the data are converted and interlinked in a coherent graph. What we want to show in this paper, and our main contribution, is that popular query models can also be translated into graph concepts, employing graph analysis in query processing. To take full advantage of the model, users should be aware of the semantics of the elements composing the graph. In that regard, our strategy is similar to an OLAP environment, in which the query model assumes data are integrated in a multidimensional schema – according to whichever strategy is adequate for the specific environment.

4.4.2 Data model integration

There are several alternatives for mapping a given data model into graphs. Although our framework works independently of the strategy adopted, we provide guidelines on basic transformations of typical models.

Here we focus on the integration of text documents and the relational model. The mapping for other models, such as semi-structured or NoSQL variations, can be derived by similar approaches. There are several alternatives for mapping a relational scheme to an RDF graph [6, 10]. There is even a W3C working group[2] to define standards for these mapping languages. Here, to simplify the discussion, we assume that (i) table descriptions become RDF classes, (ii) rows become instances of their respective tables,

[2]http://www.w3.org/2001/sw/rdb2rdf/

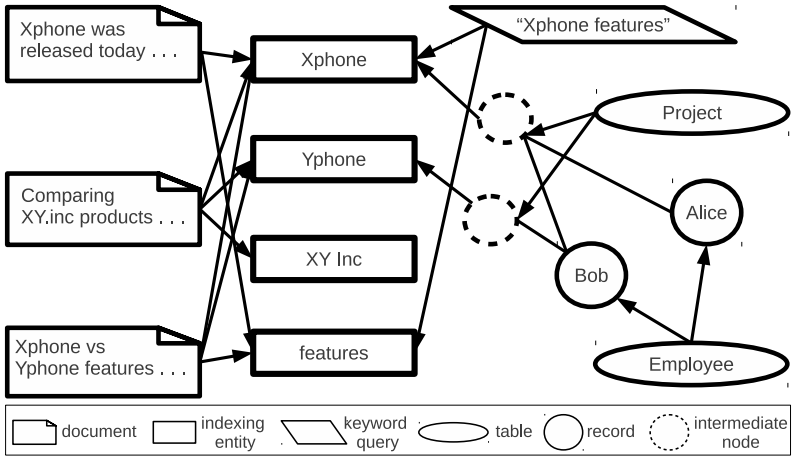


Figure 4.2: Data elements represented as a unified graph

with their primary keys as identifiers, (iii) columns become properties of the instances, with values corresponding to literals and foreign keys becoming explicit links to other instances.

Graph representation of documents for IR purposes is also possible. An inverted index (in the bag of words model) can be readily mapped into a graph that connects terms and documents. More modern schemes to index documents such as topic models [12] and explicit semantic analysis [49] also fit nicely into this strategy, bringing the benefits of reduced dimensionality (i.e. avoiding creating an unnecessarily large graph containing entire postings list), less semantic ambiguity, and more cognitive appeal.

In our framework, a keyword query is also represented as a (temporary) node in the graph. The same indexing strategy used for the stored documents is applied to generate the relationships of the query node (Figure 4.2). This graph representation of keyword queries allows them to be expressed alongside structured predicates in the queries (Section 4.5.3).

To simplify data management in the complex integrated graphs, our framework introduces mappers. Mappers play an important role in data model integration, being the mechanism that encapsulates the creation of relationships between elements of the graph.

Figure 4.2 shows a simplified example to illustrate diverse elements represented as a unified graph. News articles about products are mapped into entities according to mappers that implement an indexing/annotation technique (e.g. topic modeling, named entity recognition, etc). A keyword query is likewise mapped into these entities, using the same mapper in query time. Relational data from tables (Project, Employee) are also mapped into nodes in the graph and also connected to the entities. More details on the use of mappers to bridge data models are presented in Section 4.6.

4.4.3 Query model integration

Data access and model integration brings many benefits to institutions, providing a unified path for interaction with data. This interaction is, however, usually constrained by the data model and the query language employed for the integration. For example, in a typical OLAP setting, data are integrated in a data warehouse, but no direct benefit is

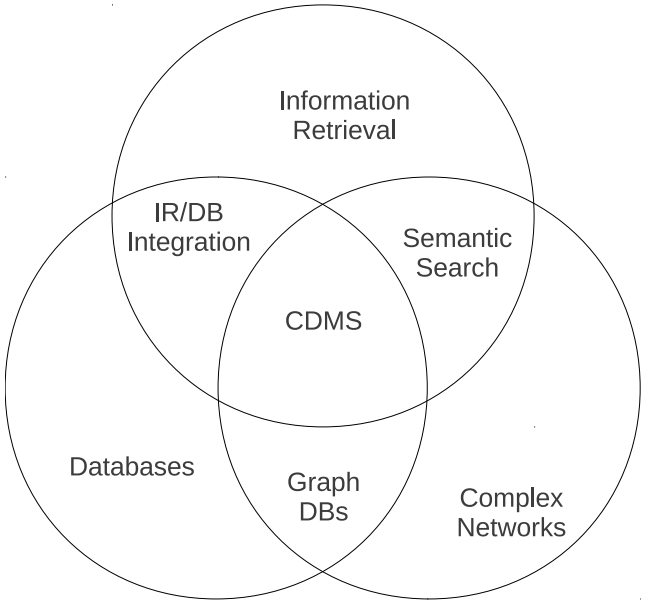


Figure 4.3: CDMS in the intersection of multiple areas

gained by applications such as institutional search engines. The problem is that there is a conceptual gap between the interaction language in the integration infrastructure (OLAP) and the languages used by the applications (keyword queries, SQL, etc.).

Our query model, on the other hand, is built on the assumption that integration should begin at the query or application level. The goal is to specify a query model that can express concepts from diverse interaction models in a unified and intuitive way. We focus on the applications related to the areas of databases, information retrieval, and complex networks (Figure 4.3). Our model can also be used in machine learning tasks, as discussed in [32].

The two main groups of models for data driven applications today are those associated with Information Retrieval and Database Systems. It is natural that these two areas attained such distinction over the last decades. They together cover a broad range of the data structuring spectrum – from unstructured data in documents to structured data in relations. Typical applications in IR include search engines, recommendation systems, social networks etc. Applications taking advantage of DBMSs are ubiquitous, being through traditional relational databases or the more recent models for document databases, XML and semistructured databases, graph databases and the NoSQL movement.

Complex networks, which have gained strong momentum in the last decade, is the third area completing our picture. Complex networks, whose techniques are often applied in typical IR and DB tasks, is an important area to cover in an integration framework. More importantly, we consider complex networks a fundamental piece to establish the basis of the integrated framework. To specify a query language that could be used in such a diverse scenario it is important to unify characteristics of the different interaction models.

Keyword queries and ranking are important concepts from IR, as other integration approaches have identified [62, 16, 4]. Significant research efforts have been dedicated to enable efficient ranking and keyword queries in a wider range of data models (e.g. rela-

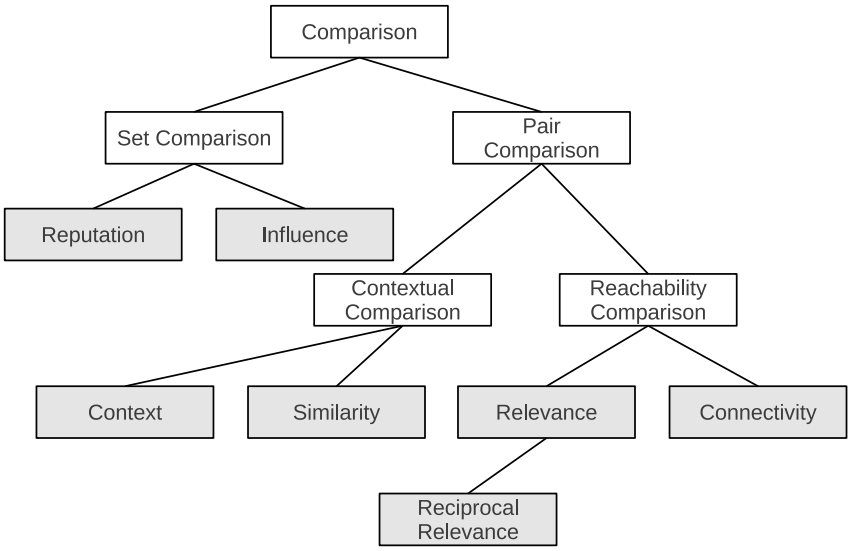


Figure 4.4: Taxonomy for the adopted ranking metrics

tional, XML). In databases, declarative languages offer effective means for online interaction with data. Furthermore, the declarative approach offers opportunities for transparent query optimization. Complex networks offer a range of techniques to assess important characteristics of the data based on the underlying connections. These techniques are employed in diverse scenarios, such as the use of relevance metrics (e.g. HITS, PageRank) for IR purposes.

Here we defined a query model that embodies characteristics from all the discussed areas, providing a declarative query language that can express structured predicates, keyword queries, network topology-aware metrics, and compose results (optionally) as ranked lists. The challenge is to enable all these features over the unified graph model (LLG) presented.

Declarative querying and traditional database concepts like selections, projections and aggregations are already provided by RDF query languages such as SPARQL. The remaining issues are related to enabling IR-like ranking metrics that now have to be reinterpreted in an RDF graph setting. To enable this extended querying mechanism, we reinterpret this topology-aware metric in a common graph processing model that we call Targeted Spreading Activation (TSA), described in the following section.

4.5 Ranking metrics and language integration

Correlating data is an important and defining characteristic for many of the applications we want to cover. To enable a high level of flexibility for correlations, we specify a set of ranking metrics which are influenced by information retrieval applications and complex networks concepts. The selection of the specific metrics aims at covering a wide range of applications while also being simple to use and understand. In the process of defining these metrics, we started with some popular metrics used in IR and then expanded the set according to the applications we wanted to cover. The set of metrics we define can be organized in the taxonomy presented in Figure 4.4.

The basis of our taxonomy is the concept of comparison. Our metrics are meant to compare elements in the graph and generate a score that represents the strength of the association. The peculiar aspect about our metrics is that the scores are generated based on analysis of the topology of the graph, in contrast to most ranking approaches that are based on attributes of the elements.

There are two main groups of comparisons. Set comparisons corresponds to comparisons among elements from a finite set. Reputation and Influence are the metrics in this category. They assess, using different strategies, how well a node performs as a hub for information. The definitions of the metrics, as well as details on their interpretation, are presented in the next section.

Pair comparisons are applied to individual pairs of nodes. They assess properties of the topology surrounding or connecting the two nodes. The similarity and context metrics, classified under contextual comparison, assess the commonalities in respect to elements (nodes or relationships) surrounding the comparing nodes. Relevance and connectivity, classified under reachability comparison, assess properties of the paths interconnecting the comparing nodes.

As far as we know, this is the first time that these metrics are considered and defined under the same conceptual framework. These metrics express cognitive processes or patterns that we use to assess correlation of entities in the real world, and which are the basis of many data-driven applications, as we intend to portray along the text. We now describe our metrics and define them from a graph analysis perspective.

4.5.1 Graph interpretation of the metrics

The translation of the ranking metrics to the unified graph strategy is a challenging task. Here we adopt a Spreading Activation (SA) [19] model for our novel interpretation of the metrics.

The Targeted Spreading Activation model:

Spreading Activation (SA) processes [19] were developed to infer relationships among nodes in associative networks. The mechanism is based on traversing the network from an initial set of nodes, activating new nodes until certain stop conditions are reached. By controlling several aspects related to this activation flow, it is possible to infer and quantify the relationships of the initial nodes to the reached ones.

This simple model has the fundamental requirements for the type of correlations we want to provide for complex data:

(i) it can derive correlations among any two sets of initial nodes and destination nodes; This is important to enable modeling of several correlation metrics, as described in Section 4.5.1.

(ii) the final value of the correlations decreases as the length of the contributing paths grows; This reflects the intuitive perception that closer elements are more correlated. The model allows tuning of this characteristic through a parameter for potential degradation.

(iii) the degradation of the potential imposes boundaries to query processing;

notation	description
$SA(N)$	a set of activated nodes after the execution of the spread activation process
$SA(M)_n$	value for the potential of node n after the execution of the spread activation with initial activated nodes M
a, t, d, c	respectively, initial activation potential, firing threshold, decay factor, maximum number of iterations (depth)
l	set of labels that determine valid nodes for traversal
dir	$dir \subset \{inbound, outbound\}$; set of directions for traversal
$\overline{dir}$	$\overline{dir} \cap \{inbound, outbound\}$; reversed directions of dir
$\overline{SA(m)_n}$	same as $SA(m)_n$ with reversed directions, i.e. $dir \leftarrow \overline{dir}$
$I(n)$	function that calculates the input potential of a node. $I(n) = \sum_{m \in ant(n)} O(i)$ in the default case
$O(n)$	function that calculates the output potential of a node. $O(n) = I(n) * d$ in the default case
$ant(n)$	set of antecedent nodes, i.e. nodes linked to n through relationships in l that follow the directions in dir
$sub(n)$	set of subsequent nodes, i.e. nodes linked to n through relationships in l that follow the directions in $\overline{dir}$
$p(SA(N))$	set of activation paths (for each node in $SA(N)$)
$\|S\|$	number of elements in set S

Table 4.1: Notation used in the definitions

(iv) it can be implemented as graph traversal patterns [55]; The processing of these patterns are centered in origin nodes, resulting in localized processing. The computation of these patterns requires less memory than global ranking metrics such as PageRank and HITS. This type of computation is supported by several graph database systems[3].

We tweak the basic SA model by adding mechanisms to (i) adapt the process to the labeled graph model used, (ii) consider relationship weights, (iii) add a more strict and predictable termination condition, and (iv) make the process aware of the target elements. The last point is key to the semantics of the SA process for querying complex data and also to improve optimization opportunities. We named the proposed SA variation as Targeted Spreading Activation (TSA).

The TSA model used here is defined by the parameters G, N, I, O, a, t, d, c, l, and dir described, alongside other definitions, in Table 4.1. A TSA process starts with origin nodes initially activated with potential a. Output potentials for each subsequent node are calculated by the function O. The output potential is spread through all relationships whose labels are in l that follow directions in dir. The potential for the reached nodes is calculated by function I. For the next iteration, the potential is spread to subsequent nodes, restarting the process, as long as the potential for reached nodes is higher than t and the number of iterations is lower than c.

[3]A good overview of applications and systems can be found in http://markorodriguez.com/2013/01/09/on-graph-computing/

Although simple in its definition, this is a very expressive model to build flexible correlation metrics. By specifying appropriate parameters and combining subsequent executions of TSAs, it is possible to define metrics that encompass concepts like relevance, reputation and similarity (Section 4.5.1). These metrics can be integrated in a declarative language with applications to a wide range of modern querying scenarios (Section 4.5.3).

Being the core of the querying process mechanism, the TSA process becomes the main target for query optimization strategies. Like with any other data or processing model, the practicability of TSA-based querying depends on architectural mechanisms to support data access optimizations and heuristics to provide approximate answers. Optimization issues were addressed in [31].

IR metrics according to the TSA model

In the TSA model, to assess the rank of the relationship of nodes according to a metric, an activation potential is placed at the target elements defined in the query. The potential is spread across the topology of the graph, losing or gaining strength based on the IR metric, length of the path, or properties of the traversed elements. The metric-specific definitions of the TSA processes are presented below.

Definition 1. $relevance(m, n) = SA(\{m\})_n$,
with $O(n) = \dfrac{I(n) * d}{|sub(n)|}$

Relevance between two nodes is a measure that encompasses correlation and specificity. Correlation is proportional to the number of paths linking the two nodes and inversely proportional to the length of the paths. Specificity favors more discriminative paths (i.e paths with fewer ramifications. It is easy to observe that this definition resembles the definition of relevance between queries and documents in a information retrieval setting. Traditional tf or idf term weighting can be readily emulated in our scheme when terms, queries and documents become nodes of a graph. Our definition is, however, a generalization of the concept that can be applied to any type of graph data and with any number or type of relationships in between m and n.

Definition 2. $rrelevance(m, n) = SA(\{m\})_n + \overline{SA(\{n\})_m}$,
with $O(n) = \dfrac{I(n) * d}{|sub(n)|}$

Reciprocal Relevance (RRelevance) between two nodes aggregates the relevance in both directions. In an information retrieval setting, it would be equivalent to aggregating tf and idf in the same metric.

Definition 3. $connectivity(m, n) = SA(\{m\})_n$

Connectivity between two nodes is a measure that assesses how interconnected two nodes are. The score is proportional to the number of paths linking the nodes in the network activated by the SA algorithm.

Definition 4. $reputation(n, N) = SA(N)_n$

Reputation of a node measures how effective it is as a hub for information flow. Here the nodes of interest are activated at the beginning and the ranking scheme favors nodes that are revisited in the sequence of the SA process. This is a simple but convenient interpretations in scenarios where the reputation cannot be pre-calculated due to high

update rates, variability in the types of relationships used for the queries, or need to bias the scores based on a set of initial nodes (as in [63]).

Definition 5. $influence(n) = |(SA(\{n\}))|$

Influence is a specialization of reputation where the only concern is the number of nodes reached from the origin. The topology of the graph – in/outdegree or cycles – do not influence the metric.

Definition 6. $similarity(m, n) = \frac{|p(SA(\{n\})) \cap p(SA(\{m\}))|}{|p(SA(\{n\})) \cup p(SA(\{m\}))|}$

Similarity measures the ratio of common relationships (same edge label linking common nodes) between two nodes.

Definition 7. $context(m, n) = \frac{|SA(\{n\}) \cap SA(\{m\})|}{|SA(\{n\}) \cup SA(\{m\})|}$

Context is a specialization of similarity where edge labels do not matter.

4.5.2 Semantics of ranking metrics in queries

Having the ranking metrics interpreted as graph analysis tasks, there is now the need of integrating these metrics in a declarative language. As opposed to creating an entirely new query language, we decided to leverage existing languages by defining an extension language that can be integrated into other languages. To that extent, we first define the semantics of the intended integration.

In our model, the proposed ranking metrics are intended to be used with graph query languages that offer: (i) means to reference individual nodes in the graph, (ii) selection of match variables, and (iii) query results as a set of tuples (or a graph representation of). These are basic components of graph languages like SPARQL and Cypher. A ranking metric can refer to:

- a single match variable (set of vertices), e.g."rank papers from EDBT 2013 according to first author *reputation*", where first author is the match variable in question (e.g. "SELECT ?firstAuthor ..." in SPARQL);
- a given vertex[4] and a match variable, e.g. "rank papers according to *relevance* of their first author (match variable) to the topic data integration (vertex)";
- two match variables, e.g."rank papers according to *relevance* of the first author to the topic in the first keyword of the paper".

Conceptually, the ranking metrics are applied to query results, generating a ranking value for each returned tuple. In practice, to speed up query processing, results would be approximate and the rank would be generated for some of the nodes based on access pattern heuristics.

4.5.3 Extending Declarative Queries

Having the ranking metrics interpreted as graph analysis tasks, it is possible to integrate them in a declarative query language. As opposed to creating an entirely new query language, we decided to leverage existing languages by defining an extension language.

[4]as defined previously, a keyword query would also be a node in the graph

```
1  ExtendedQuery::=RegularQuery RankClause
2  RankClause::='RANK BY' RankMetric ( ',' RankMetric )*
3  RankMetric::= Weight? ( UnaryRMetricDesc | BinaryRMetricDesc )
4  UnaryRMetricDesc::=UnaryRankMetric  'OF' MatchVariable
5          Modifiers*
6  BinaryRMetricDesc::=BinaryRankMetric 'OF' MatchVariable 'TO'
7          ( MatchVariable | Vertex | KWQuery) Modifiers*
8  UnaryRankMetric::=Reputation | Influence
9  BinaryRankMetric::=Relevance | Connectivity | Similarity | Context
10 Modifiers::=Follow | Depth | Direction | Weighted
11 Follow::='FOLLOW' EdgeSet
12 Depth::='DEPTH' INTEGER
13 Direction:=='DIRECTION' ( 'INBOUND' | 'OUTBOUND' | 'BOTH' )
14 Weighted:=='WEIGHTED'
15 KWQuery:=='KWQUERY' '(' String ')'
16 EdgeSet:=='(' Edge ( ',' Edge )* ')' . . .
```

Figure 4.5: Simplified BNF grammar for the proposed extension (terminators omitted)

A convenient way to integrate the ranking metrics into existing query languages is to add a "RANK BY" clause. The clause should enable an arbitrary combination of metrics that expresses the global ranking condition defined by the user. We encode the clause in the extension query language that we denominated in* (or in star). in* can be used to extend other languages, for example, extended SPARQL becomes inSPARQL by convention. More details about the language and its design principles can be found in [33].

Note that the extension causes query semantics and result interpretation to change, therefore, any extended language would be more adequately described as new language based on the syntax of the original language. This suggests an incidental meaning for an acronym like inSPARQL: recursively, "inSPARQL is Not SPARQL".

Figure 4.5 shows a simplified BNF grammar of the proposed extension. A ranking can be specified as mix of weighted ranking metrics (lines 2 and 3). Weights capture the relative importance of each metric. The scores generated by the metrics are normalized before the calculation of the final weighted score.

Ranking metrics are unary or binary. Unary ranking metrics are applied to a single match variable (lines 4 and 5). Binary ranking metrics can be applied to a match variable and a named vertex or between two match variables.

The language allows for modifiers (lines 10 to 14) to be applied to the ranking definitions. These modifiers define the parameters for the execution of the SA algorithm. FOLLOW specifies valid edges for the algorithm to traverse. DEPTH defines the maximum length for the traversal paths. DIRECTION sets the direction of traversal as outbound, inbound or both (default) edges. WEIGHTED makes edge weights influence the degradation of the activation potential (the potential is multiplied by the weight).

The combination of the IR-inspired metrics in a declarative querying setting enables a high level of flexibility and expressiveness for the applications to explore. In the next section we show and discuss some examples of queries that can be used for practical applications.

4.5.4 Applications

This section presents examples of queries in the extended SPARQL language. These queries are meant to demonstrate the expressiveness of the approach in a wide range of applications.

Search engines/CMSs: Figure 4.6a shows a possible implementation for a document retrieval query using topic modeling. The keyword query is expressed by the function KWQUERY [5] and the relevance is assessed as if the query was a node in the graph. The query also takes into account the reputation of the authors and the relevance of documents to the topic :Politics (assessed based on the connections between the query node and documents that are created by a Topic Modeling algorithm such as LDA). Data management aspects discussed in the next section would be interesting matches to implement novel CMS architectures like in Ngomo et al. [52]. Our metrics would also allow query answering based on the context of the user or a context defined by the user, implementing a query model such as the one proposed by [56]. Graph-based term weighting [11] could also be simulated in our query model.

Recommendation systems: Figure 4.6b shows a product recommendation query that finds products that the client Bob (with uri :bob) has not purchased. The query traverses Bob's friendship network to find products purchased by his friends that might be relevant to him. The spreading activation interpretation of this query evaluation also implies that products purchased by Bob, even though they do not appear in the results, will be traversed on the way to customers that have co-purchased these products, which in turn will activate other products from these customers.

Social Networks: Figure 4.6c shows a query that could be used for friend suggestion on a social network application. It ranks the top 5 persons over a given age based on the similarity of hobbies and movie preferences of user Alice.

Collaborative filtering: Figure 4.6d shows a query that filters posts from pages that friends of user Carol follow. The posts are ranked based on their influence in the network.

Decision support: Figure 4.6e shows a query that can be used to prospect for employees that would be good candidates to replace a manager (Charlie) in his post. The query favors employees strongly related to a (presumably important) product (yPhone) and also those that have professional contexts similar to the current manager.

Other applications: Similar queries could be used in several other scenarios, especially the ones with richly interconnected data and that require complex analysis of the correlations. Some examples are Semantic Web inference applications, were assessing correlations between classes and candidate instances can be complex [3]. The scientific domain is another interesting application field. For example, in a database with food network relationships, a query could identify relevant species or areas for conservation efforts.

[5] KWQUERY is a syntactical shortcut that represents an underlying mapper as in Section 4.6.1


```
SELECT ?doc, ?author
 WHERE { ?doc :type :BlogPost }
RANK BY (2 KW, 1 PoliticsTopic, 3 AuthorRep)
 2 RELEVANCE OF ?doc TO KWQUERY(``US elections'')
 1 RELEVANCE OF ?doc TO :Politics
 3 REPUTATION OF ?author
```


```
SELECT DISTINCT ?product
 WHERE { ?product :type :Product .
   FILTER NOT EXISTS (:bob :purchased ?product)}
RANK BY RELEVANCE OF ?product TO :bob
 FOLLOW (:friendsWith, :purchased)
 DEPTH 3 DIRECTION BOTH
```


```
SELECT ?person
 WHERE { ?person :hasAge ?age .
         FILTER (?age > 30)}
LIMIT 5
RANK BY SIMILARITY OF ?person TO :alice
 FOLLOW (:hasHobby, :favoriteMovie)
```


```
SELECT ?post, ?page
 WHERE { :carol :closeFriend ?closeFriend .
         ?closeFriend :follows ?page .
         ?page :publishes ?post .
         ?post :date ?date .
         FILTER(?date = "2012-12-12"^^xsd:date)}
RANK BY INFLUENCE OF ?post
```


```
SELECT ?employee
 WHERE { ?employee :worksFor :research}
RANK BY
 2 RELEVANCE OF ?employee TO :yPhone
 1 CONTEXT OF ?employee TO :Charlie
```

Figure 4.6: Examples of extended SPARQL queries (namespaces have been omitted)

```
MAP DISTINCT ?movie, ?label
WITH TokenMapper
WHERE{
   ?movie a movie:film .
   ?movie rdfs:label ?label }
```

Figure 4.7: MAP statement applying mapper *TokenMapper* to movies and their labels

4.6 Relationship Management in the CDMS

We have so far discussed our data and query models, with little focus on implementation or architectural aspects. The proposed query model implies new requirements for user interaction, query processing and data management. The CDMS is responsible for encompassing all these aspects in a coherent architecture.

The querying mechanism presented so far is based on the observation that relationship analysis is central to several applications and the basis for evaluating the metrics introduced here. Besides providing a query language that enables expressive correlation clauses, it is important to provide the CDMS with mechanisms to manage diverse aspects of relationship life cycle. Here we show how such mechanisms could provide better support for data integration tasks and increase the expressiveness of the query language.

4.6.1 Mappers

Relationship creation is an important and defining operation for the described application scenarios. For example, several text indexing tasks, such as topic modeling, derive relationships between the text and more general concepts. To support these types of task, an integrated framework must provide mechanisms to facilitate the creation of these relationships in the unified graph. The same type of mapping between source data and the unified graph is required for other types of data such as relational or semi-structured.

In our framework, the creation of relationships is encapsulated in *MAP* statements. Figure 4.7 shows an example (detailed in the experiments section) of such DML (Data Manipulation Language) query. The MAP statement (that triggers a *mapper*) is also encoded as an extension of a graph query language (SPARQL, in this case). The query selects all nodes of type 'film' and their respective labels. The selected elements are used to call the mapper *TokenMapper*.

Mappers are very similar to *stored procedures*. What sets them apart are (i) their integrated use in our ranking queries, and (ii) how they are hooked in the databases's API so that any new data that matches the mapping criterion is passed through appropriate mappers. Point (ii) is not yet supported by our framework. (i) is discussed in the experiment section.

Mappers are the mechanism that underlie the creation of the LLG. In a wider perspective, mappings are however not restricted to model transformations, but also allow transformations of data already in the unified graph, for example, to infer new relationships based on correlations between nodes. These ad hoc mappings are especially important for querying and analyzing data, enabling users to manipulate the underlying data at query time without the obligation to materialize the new relationships. For this

reason, our approach addresses query and mapping as an interdependent and symbiotic process of data analysis and exploration.

A mapping process stores metadata related to the creation of the relationships, which can be explored at query time. Since relationships are associated with their mappers, multiple mappers can be used for the same type of relationship. For example, multiple text indexing strategies can be used simultaneously, then queries can specify the strategy that best fits the information need or simply take advantage of the multiple connections created by the diverse mappers.

Metadata about creation time and usage statistics of the relationships can also be used in a more expressive version of the extended query language presented here. Query modifiers could refer to this metadata to favor *novelty* or *popularity* (also important concepts in IR) of the relationships.

4.7 Experiments

We now show experiments that aim at demonstrating what we envision as a typical usage scenario for our framework. The database used in the experiments is the Linked Movie Data Base (LinkedMDB) [37], which we think is a good representative for the type of unified graph we aim at. The database integrates data from several sources (FreeBase, OMDB, DBpedia, Geonames, etc). The process used to semantically integrate the distinct sources is similar to what is done in a typical Data Warehouse and precisely what we envision to be the workflow for the usage scenarios of our framework (i.e. integration of institutional data). The database contains 3,579,616 triples. The dataset has other important characteristics: (i) it encompasses the bulk of the production in an important area of human activity, (ii) data elements have clear semantics, (iii) data elements are organized based on several characteristics (type, genre, subject, etc.) and correlated in a complex graph topology. These characteristics support the applicability of our framework in real scenarios.

We implemented a basic mapper (TokenMapper) that maps input nodes into tokens present in their labels. The tokens are themselves stored as nodes in the graph database. The mapper receives as arguments the source node for the mappings and the text content to be mapped. This mapping uses a standard query analyzer (from Lucene's library) that simply lowercases, removes stop words, and tokenizes the text. We are using this strategy for its simplicity and didacticism. In a real scenario, more modern techniques such as NER (Named Entity Recognition), LSA (Latent Semantic Analysis) or ESA (Explicit Semantics Analysis) would provide more efficient and meaningful mappings.

The DML query used for our mapping is shown in Figure 4.7. We are mapping the label of movies to tokens. Executing the query triggers the mapping of each selected movie, generating the appropriate nodes for the tokens when needed. This same mapper can be used in the ranking clause, as will be shown below.

We now present analysis of queries to demonstrate the use of the metrics and mappers. At this point of development of our framework, we are focusing on the relevance and connectivity metrics, which we regard as having widespread applications and presenting

```
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?movie where{
   ?movie a movie:film .
   ?movie movie:initial_release_date ?date .
   FILTER ( fn:substring(?date, 1, 4) > "1990" ) .
   FILTER EXISTS { ?movie foaf:page ?page }
}
RANKED BY
 2 RELEVANCE OF ?movie TO
    TokenMapper("My favorite films are Matrix and Avatar")
    DEPTH 2 FOLLOW TokenMapper:hasToken,
 1 RELEVANCE OF ?movie TO movie:film_subject/461
    DEPTH 5 FOLLOW skos:subject
```

Figure 4.8: Query that ranks movies according to relevance to a preferences text and to the subject "Virtual Reality" (film_subject/461)

the biggest challenges for query processing.

The first query (Figure 4.8) retrieves and ranks movies relevant to a text stating movie preferences and that are also relevant to the subject 'Virtual Reality'. This type of query can be derived from user's profiles, for example. The query also selects elements based on structured predicates, specifying that returned movies should have a home page relationship and have been released after 1990.

The evaluation of this query can be divided in two phases: (i) graph matching and structured selection, and (ii) ranking based on topological properties. This separation in two distinct phases is only conceptual however. A query processor is free to combine the phases in any fashion to optimize the evaluation.

In the graph matching and selection phase, the triple pattern is matched against the data graph, the results are filtered according to the structured predicates. In the second phase, the multiple ranking criteria are evaluated.

The first ranking criterion is relevance to the text. To assess the scores for this metric, the query processor creates a temporary node and appropriate mappings are made using the specified mapper (TokenMapper). The spreading activation process is then executed to assess the correlation between each movie and the temporary node. The process is set to follow the relationships created by the mapper (hasToken). This is a typical mechanism for a keyword query type of interaction in our framework. A system-wide default keyword query mapper can be set so that queries can use the reserved KWQuery element, so that the parser automatically assigns the appropriate mapper and relationships to follow (as in Figure 4.6a).

The second ranking criterion assesses correlation between movies and the subject "Virtual Reality". The resulting scores from each criterion are normalized and aggregated, according to the specified weights, to generate the final score. The results (Table 4.2) show contributions from both ranking criteria. The movie Avatar is not directly related to the subject "Virtual Reality", owing its high ranking to the high tf*idf value of its name (note that tf*idfs are not calculated, this is an emergent property of the relevance metric

top 15	name
0.67	Avatar
0.67	Avatar
0.55	The Matrix
0.53	The Matrix Revolutions
0.53	The Matrix Reloaded
0.33	The Thirteenth Floor
0.33	EXistenZ
0.33	Lawnmower Man 2: Beyond Cyberspace
0.33	Storm Watch (aka Code Hunter)
0.33	Strange Days
0.33	The Lawnmower Man
0.33	Welcome to Blood City
0.21	The Favorite
0.19	The Matrix Online
0.19	The Matrix Revisited

Table 4.2: Top-15 ranked results for the first query

```
PREFIX film: <http://data.linkedmdb.org/resource/film/>
PREFIX resource: <http://data.linkedmdb.org/resource/movie/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
SELECT DISTINCT ?movie WHERE{
   ?movie a resource:film .
   ?movie resource:initial_release_date ?date .
   FILTER ( fn:starts-with(?date, "199")
}
RANKED BY
  CONNECTIVITY OF ?movie TO film:38145
  DEPTH 4 FOLLOW (skos:subject, resource:director)
```

Figure 4.9: Query that retrieves films correlated to 'The Silence of the Lambs' (film:38145)

which is not restricted to text-based rankings)[6]. All movies from the Matrix franchise have scores combining both criteria. Lower ranking results also provide insight into the interplay between the rankings for this query (e.g. 'The Favorite' matches the keyword query with a lower tf*idf-like score).

The second query (Figure 4.9) uses the connectivity metric to discover films correlated to 'The Silence of the Lambs'. The query specifies that the analysis should consider only the relationships 'movie:director' and 'skos:subject'. It is interesting to note that setting the modifier *depth* to 4 means that indirect correlations are also considered. For example, a film could receive a positive score even though it does not share a subject with 'The Silence of the Lambs', as long as it is correlated with a film that does share a subject.

The results for the query are shown in Table 4.3. The output is strongly biased towards scores generated by correlations through common subjects (which tend to form tighter clusters). If the user wants to increase the importance of the 'director' relationship to retrieve more movies correlated to the director of 'The Silence of the Lambs', the user

[6]The second Avatar record refers to a lesser known Singaporean film (introducing a *reputation* metric in the query would certainly lower its score).

top 10	name
1.0	The Silence of the Lambs
0.81	Man Bites Dog
0.80	Natural Born Killers
0.80	Butterfly Kiss
0.80	Freeway
0.79	Seven
0.79	Aileen Wuornos: The Selling of a Serial Killer
0.79	Serial Mom
0.79	Copycat
0.79	The Young Poisoner's Handbook

Table 4.3: Top-10 ranked results for the second query

top 10	name
1.0	The Silence of the Lambs
0.34	Philadelphia
0.34	Cousin Bobby
0.25	Man Bites Dog
0.25	Natural Born Killers
0.25	Butterfly Kiss
0.24	Freeway
0.23	Seven
0.23	Aileen Wuornos: The Selling of a Serial Killer
0.23	Serial Mom

Table 4.4: Top-10 ranked results for the first query

can separate the relationships into two ranking criteria. This type of user interactivity is another important advantage our declarative querying scheme. The results of splitting the rankings in such a way are presented in Table 4.4. The results are for a version of the query that used a 3:1 weight division favoring the *director* relationship.

4.8 Related work

We now discuss related work on data integration in various levels: from data access integration, through syntactic/semantic integration, and up to application or query model integration. Integration at any level is highly dependent on the lower levels.

4.8.1 Data access integration

The first level of integration must provide a unique access point for the data. This can be accomplished by basically two approaches: centralizing the data or connecting the data sources in an infrastructure that simulates a centralized repository. Centralized integration of institutional data is typically related to the deployment of data warehouses or data marts [40]. Data centralization approaches have also been proposed in the context

of the Semantic Web [37], and the DBpedia project[7] is a notable example of this type of approach.

The research on Federated Databases aims at providing a unified view of the data while maintaining the autonomy of the data sources [60]. In the context of the Semantic Web, Schwarte et al. [59] have proposed a federation layer for Linked Open Data. Schenk and Staab [58] have proposed a mechanism for the specification of views over Linked Data, enabling declarative federation of data sources.

Our framework is independent of the specific strategy chosen for data access integration. The requirement is that all interaction is done as if the data was integrated in a unified graph. Whether this integration is done through federation or physically integrating the data is an architectural decision based on expectations of performance and requirements for preserving the autonomy of data sources.

4.8.2 Data model integration

Data integration requires enabling data manipulation under a unified model. Federated databases frequently employ the relational model (common among data sources) for the integration. Data minig, which has application-specific requirements, favors the multidimensional model [41].

In the Semantic Web, the adopted unifying model is the RDF graph. The Resource Description Framework (RDF)[8] is a general-purpose language created for representing information about resources in the Web. The basic unit of information is a statement triple, which contains a subject, a predicate, and an object. All elements in a triple are identified by URIs (except for objects that can also be literal values). Triples can refer to each other, forming a graph. The advantage of the RDF model comes from its simplicity, enabling the representation of data from a wide range of domains.

There has been a substantial amount of research in mapping other data models into RDF [6, 10]. The W3C RDB2RDF Working Group[9] is defining languages and standards for mapping relational databases into RDF.

Besides having the data in a unified representation model, it is important to correlate data from the diverse sources into unified concepts. In the relational world, this process is know as record deduplication or linkage and is part of the ETL (Extraction Transformation Loading) workflow [36]. In the Semantic Web, the usual way to represent these correlations is the creation of *sameAs* relationships between entities. These relationships can be created manually or by automated processes. Hassanzadeh and Consens [37] employ several string matching techniques to correlate Linked Open Data from diverse sources to create an interlinked version of a movies database.

In this proposal, we assume that the institutional data is integrated in an RDF graph. This allows us to take advantage of other standardized technologies developed in the context of the WWW and the Semantic Web, such as universal identification through URIs, semantic integration through *sameAs* relationships, and the SPARQL query language.

[7] http://dbpedia.org/
[8] http://www.w3.org/RDF/
[9] http://www.w3.org/2001/sw/rdb2rdf/

4.8.3 Query model integration

Once data is integrated, it becomes possible to pose queries that could not be answered before, producing more valuable information for institutions and the public. The integration approaches, however, typically focus on integrating data under a specific query model, such as the relational or OLAP. This usually constrains the range of data models that can be integrate and, foremost, restricts direct querying of the integrated data from applications that use other query models.

Recently, there has been initiatives aimed at tackling integration at the application/query level. The research community has identified the interplay between the fields of Databases (DB) and Information Retrieval (IR) as a means to improve data integration and query expressiveness across applications [16, 4]. The drive to integrate the areas stems from the fact that they represent the bulk of data stored and processed across institutions. Furthermore, either field has been very successful by their own but still faces challenges when dealing with interactions typical to the other field.

The integration of the IR and DB areas has been an important topic in the agenda of the research community for many years. Following the initial identification of challenges and applications, several successful approaches were proposed and implemented [62]. Most prominent research focuses on keyword queries over structured data and documents, top-k ranking strategies and extraction of structured information from documents.

Keyword query research draws from the simple yet effective keyword query model to allow integrated querying over documents and structured data. Most of the frameworks match keywords to documents, schema and data integrated in a graph structure. The connected matches form trees that are ranked based on variations of IR metrics such as tf*idf and PageRank. Some of the research focus on optimizing the top-k query processing [44] while others implement more effective variations of the ranking metrics [47].

Keyword queries over structured data are intended for tasks where the schema is unknown to the user. The techniques are effective for data exploration, but there is no support for more principled interactions. There are conceptual and structural mismatches among queries, data and results that make returned matches hard to predict and interpret. Furthermore, the queries can only express relevance between the provided keywords and database elements. Our query model can represent many more correlation criteria that can be combined arbitrarily in user-defined expressions. More importantly, the queries can correlated any type of data in the graph database.

The research on Top-k queries focus on enabling efficient processing of ranked queries on structured and semi-structured data. Ranking is based on scores derived from multiple predicates specified in the query. The main challenge is to compute results avoiding full computation of the expensive joins. The proposals vary on adopted query model, data access methods, implementation strategy, and assumptions on data and scoring functions (see [39] for a contextualized survey).

Scoring functions enable ranking based on properties of data elements. There is, however, no simple means to rank results based on the context of elements or how they are correlated, typical requirements for IR-like applications and a defining characteristic of our SA-based ranking scheme.

This type of contextual ranking could only be implemented in an ad-hoc fashion through complex scoring functions. Since the query processor would be unaware of the semantics of the queries and the topology of the relationships, there would be no opportunity for the optimizer to make sensible execution plans. Furthermore, the relational model assumed in most research has no means to reference individual data elements, an important requirement for effective data correlation. Our focus is on offering predicates that allows ranking based on contextual metrics not readily available as attributes. The proposal described here is complementary to regular top-k querying. It is important to support both types of ranking, since they are recurrent to many applications.

Information Extraction refers to the automatic extraction from unstructured sources of structured information such as entities, relationships between entities, and attributes describing entities [57]. Information Extraction systems employ two main techniques to harvest relationships (or facts) from text: extrapolating extraction pattens based on example seeds [25] and employing linguistic patterns to discover as many types of relationships as possible, task known as Open Information Extraction [7]. Loading the extracted facts on a DBMS allows declarative querying over the data. This is a one-way, data-centric type of integration of DB and IR. The integration proposed here focuses on unified querying and data models. The framework proposed allows easy integration of Information Extraction system's output, maximizing the benefits of both approaches.

We argue that the mentioned approaches tend to focus on infrastructure issues related to extremes of enabling the type interaction present in one area over the data model of the other. In this paper we take a top-down approach to modeling the integration, questioning what are the main and defining properties of each area, and how to offer a unified, non-modal interaction over data and query models.

4.9 Conclusion

We showed how modern standards and technologies developed to solve integration issues on the Web can be applied in a unifying framework for institutional data. Representing the integrated data as a graph is a good strategy for data model integration. Our main contribution is on extending this type of integration to a higher level of abstraction, tackling integration of query models.

In our approach, the key to achieve more expressiveness at the query level is the combination of flexible metrics in a declarative model. Our query model redefines several metrics that rank entities based on the topology of their correlations. To the best of our knowledge, this is the first time the metrics presented are considered and formalized under the same model. Similarly, we are not aware of other ranking strategies that enable the level of expressiveness offered by the combination of our metrics and a declarative language. This combination allows data correlation queries that cover a wide range of applications. The introduced mappers play an important role as a data management mechanism to support this high level of integration.

As suggested by the query examples presented (Figure 4.6), it is possible to represent information needs that would require a level of data analysis that is beyond current imple-

mentations of typical DB or IR systems. In fact, answering the type of queries introduced here in a typical technological environment nowadays would require substantial engineering for the implementation of *ad-hoc* solutions. The expressiveness of the queries allowed by the extended languages sometimes blurs the line between declarative queries and data analysis. Given the computational requirements of such settings, it is important to introduce optimization mechanisms and heuristics to compute approximate answers. Our declarative querying scenario opens many opportunities for query optimization. Details about the mechanisms we are currently adopting are described in [31].

We expect query-level integration to become increasingly important as our technological landscape continues to diversify. We showed how our model can cover a broad range of models and applications. Our experiments indicate the practicability of our approach, especially regarding the use of mappers to simplify data integration and enable more expressive querying.

Chapter 5

Query-based inference in the CDMS

5.1 Introduction

The Databases (DB) and Machine Learning (ML) fields have evolved to become essential parts of computational infrastructures. Although by means of different mechanisms, both fields answer queries or make inferences based on digital data representing real world entities and concepts. Their tasks are, however, typically undertaken as separated processes. Data from databases indirectly feed learning models whose inferred data is in turn stored in databases. Guiding the processes are experts in one area or the other, developing *ad hoc* solutions to connect the ends, and rarely overlapping technical skills.

On another level, both DB and ML fields have developed to handle increasingly complex data in terms of how data elements are correlated. Several ML tasks have to discover complex interactions among features. These tasks usually adopt graph-based models such as decision trees, neural networks, and, becoming popular more recently, hidden markov models (HMM), conditional random fields (CRF), relational markov networks (RMN), etc. Databases, likewise, have adopted higher complexity in their models, allowing for more flexible representation of relationships. This trend is evident in models that attempt to overcome limitations of the relational models, such as the object-oriented or semistructured models, and especially, the graph models that experienced renewed interest in the last decade.

In this paper, we focus on demonstrating the benefits of a tighter integration among the DB and ML fields. The two main points of our proposal are (i) making feature extraction part of the data lifecycle, i.e. materializing features as data elements, and (ii) mapping the supervised learning of classification models into a query composition problem. To tackle these points, we employ a new database system being defined by the authors. The Complex Data Management System (CDMS) aims at enabling querying and management of data typical of *complex networks*. The system adopts a graph data model to represent data from structured or unstructured sources. The CDMS also offers mechanisms to simplify the management of relationship lifecycle and, most importantly, a flexible query model that can represent complex interactions among data elements. The query model allows parametrized definition of ranking metrics that are assessed based on properties of the topology of the underlying graph. This enables a level of expressiveness that can blur the line between querying and data analysis.

Our experiments demonstrate preliminary results based on a nursing diagnosis task over real data. We show how the classification task can be expressed as a declarative query in our query model. We also show how (manual) parameter tuning improves accuracy.

The main advantages of our approach are: (i) features become part of the database, evolving together with data, and (ii) the learned model is a declarative query, which gives insight into the important relationships among data and gives the opportunity for developers to tweak query parameters and scope with no need of retraining.

This paper is organized as follows: Section 5.2 describes related work in ML and complex networks. Section 5.3 defines the Complex Data Management System and introduces ML-related issues. Section 5.4 describes how ML tasks can be represented in our framework and presents experiments on real data. Section 5.5 concludes the paper.

5.2 Related Work

Relationship analysis and topological properties are central to both machine learning tasks and complex network analysis. If the outcome of a ML task depends on complex, non-independent interactions among features, it is important to employ adequate models that can capture the intricacies of the relationships. Decision trees, neural networks, HMMs, and CRFs [45] are models that aim at capturing these non-trivial associations in graphs representing features and outcomes. The more complex the graph model, the better its ability to capture non-linearities, and the bigger its computational costs.

More recent models have enabled collective reasoning, capturing associations among multiple instances of data and among related learning tasks. Examples of such models are the Markov Logic Networks [54] and Relational Markov Networks [15]. In this proposal, we also aim at capturing the correlations among elements and outcomes represented in a graph (the database itself). The most important distinction is that we want to represent the relevant correlations (i.e. the learned model) as a declarative query language.

The database framework used in our proposal is being developed to tackle issues associated with Complex Networks. In a complex network [22], the patterns defined by the interconnections are non-trivial, deviating substantially from cases where connections have the same probability (e.g. lattices or random graphs). The techniques developed for complex network analysis have become important resources in diverse applications such areas as systems biology, neuroscience, communications, transportation, power grids, and economics [21].

Our CDMS is aimed at enabling querying and management of what we define as *complex data*. Complex data is characterized when relationships are central to data analysis. In these cases, the graph formed by data entities (nodes) and relationships (links) present properties typical of complex networks. Our query language employs metrics that express correlations among data elements. These metrics are related to easily interpretable concepts such as relevance and reputation. Our hypothesis is that our language can, for some tasks, capture the same type of underlying patterns captured by graph-based learning models. Our data management mechanisms (e.g. *mappers*) can also be employed to simplify feature extraction and management.

5.3 Complex Data Management

The CDMS is aimed at providing adequate support for handling and querying complex data. It differs from typical DBMSs in four main aspects: (i) data model, (ii) query language, (iii) query evaluation, and (iv) data management mechanisms. Each of these items is described below, together with pertinent considerations related to ML.

5.3.1 Data model and data integration

The data in target CDMS applications typically do not comply to pre-defined schemas. The high number and diversity of the relationships require a model where relationships are first-class citizens. Graph models are obvious choices in these settings. Their flexible modeling characteristics enable easy mapping of most types of data. Nodes with immediate access to neighbors is also an important feature for the type of computation involved. The CDMS framework adopts weighted edge-labeled property multigraphs to encode complex data.

This graph data model is a convenient means to represent data from structured and unstructured sources. Mapping structured data to a graph is a straightforward process. Unstructured data, such as text documents, is a more challenging task. In our framework, it is convenient to represent individual unstructured elements as single nodes. The graph is then extended by extracting features from the added elements, which are themselves represented as nodes. Nodes representing the unstructured sources are linked to their extracted features through relationships created by *mappers*.

Figure 5.1 shows a simplified example to illustrate diverse elements represented as a unified graph. News articles about products are mapped into entities according to mappers that implement an indexing/annotation technique (e.g. topic modeling, named entity recognition, etc). A keyword query is likewise mapped into these entities, using the same mapper in query time. Relational data from tables (Project, Employee) are also mapped into nodes in the graph and also connected to the entities. More details on the use of mappers are presented in Section 5.3.4.

5.3.2 Query language

Our CDMS query language is intended to be flexible enough to allow correlation of data when little is known about how they are linked and organized. We developed a declarative query language that extends existing graph languages by introducing ranking based on a set of flexible correlation metrics. The ranking metrics proposed are: relevance, connectivity, reputation, influence, similarity, and context.

Our extension language is aimed at graph languages such as SPARQL[1] and Cypher[2]. We integrate the ranking metrics into existing query languages by adding a "RANK BY" clause. The clause enables an arbitrary combination of metrics that expresses the global ranking condition defined by the user.

[1]http://www.w3.org/TR/sparql11-query
[2]http://docs.neo4j.org

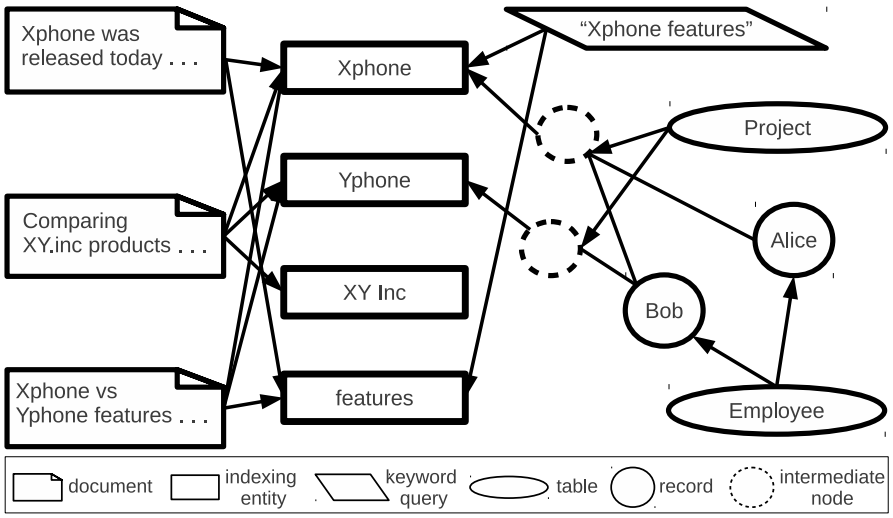


Figure 5.1: Data elements represented as a unified graph

a

```
SELECT ?doc
 WHERE { ?doc :type :News }
RANK BY
 2 RELEVANCE OF ?doc TO TokenMapper("XPhone features")
 1 RELEVANCE OF ?doc TO :ProjectAlpha
```

b

```
START diag=node(*)
  WHERE has(diag.Type) and diag.Type = "Diagnose"
  RETURN diag
RANK BY
  %r RELEVANCE OF diag TO node( %p ) WEIGHTED,
  %c CONNECTIVITY OF diag TO node( %p ) WEIGHTED
```

Figure 5.2: CDMS query examples (SPARQL and Cypher)

Figure 5.2a shows a typical query (in extended SPARQL) for data such as in Figure 5.1. The query retrieves documents relevant to the keyword query based on a simple tokenizer mapper (TokenMapper). Alternatively, the mapper could use more sophisticated strategies, such as topic modeling or entity recognition. The query also favors documents that are relevant to the project represented by URI :ProjectAlpha (prefixes are omitted). Figure 5.2b shows the query template (in Cypher) used in the classification task described in Section 5.4.

The raking metrics can specify several modifiers that restrict the subgraph used to assess the correlations. Among these modifiers are DEPTH, which limits the radius of the subgraph, and FOLLOW, which determines valid relationship labels to traverse. The tuning of these modifiers is directly related to the accuracy and computational efficiency of the queries.

The combination of the correlation metrics in a declarative querying setting enables a high level of flexibility and expressiveness for the applications to explore. We have employed our query language in information retrieval and recommendation tasks. Our goal here is to show how this query expressiveness can also be harnessed in order to represent some ML tasks. More details about the language, its design principles, and other applications can be found in [33].

5.3.3 Processing model

Our abstractions for query evaluation fully support the query language while allowing for under-the-hood optimizations. We adopt a variation of the spreading activation (SA) model as our main abstraction for query evaluation. The model allows the specification of the ranking metrics that are the basis of our query language. The SA mechanism is based on traversing the network from a initial set of nodes, activating new nodes until certain stop conditions are reached. By controlling several aspects related to this activation flow, it is possible to infer and quantify the relationships of the initial nodes to the reached ones.

Our SA model is represented as $SA(N)$, defined by the parameters G, N, I, O, a, t, d, c, and l. A SA process starts with the N nodes initially activated with potential a. Output potentials for each node are calculated by the function $O(n) = I(n) * d$ (in the default case). The output potential is spread through all edges whose labels are in l. The potential for the reached nodes is calculated by function $I(n) = \sum_{i \in in(n)} O(i)$. For the next iteration, the potential is spread, restarting the process, as long as the current potential for reached nodes is higher than t and the radius of the activated network is lower than c. $v(n)$ represents the final value for the potential of n at the end of the process. A detailed description of the SA process can be found in [33]. We show below how some of the metrics are represented as SA processes.

Def. 1. $relevance(n, m) = v(SA(\{n\})_m)$,
with $O(n) = \frac{I(n) * d}{|out(n)|}$

Def. 2. $connectivity(n, m) = v(SA(\{n\})_m)$

Def. 3. $influence(n) = |SA(\{n\})|$

Def. 4. $context(m,n) = \frac{|SA(\{n\}) \cap SA(\{m\})|}{|SA(\{n\}) \cup SA(\{m\})|}$

These metrics capture topological properties of the underlying relationships that are associated with cognitive processes, making them easily understandable by users. **Relevance** between two nodes is a measure that encompasses correlation and specificity. Correlation is proportional to the number of paths linking the two nodes and inversely proportional to the length of the paths. Specificity favors paths with less ramifications. It is easy to observe that traditional tf*idf weighting over data as in Figure 5.1 is an instance of this definition (for trivial paths of length two). **Connectivity** between two nodes is a measure that assesses how interconnected two nodes are. The score is proportional to the number of paths linking the nodes in the network activated by the SA algorithm. **Influence** is a specialization of **reputation** where the only concern is the number of nodes reached from the origin. The topology of the graph – in/outdegree or cycles – do not influence the metric. **Context** and **similarity** measure the ratio of common elements surrounding two nodes. Context is a specialization of similarity where edge labels do not matter. The definitions and usage of other metrics can be found in [33].

5.3.4 Relationship Management

Relationship creation is an important and defining operation for the described application scenarios. For example, several text indexing tasks, such as topic modeling, derive relationships between the text and more general concepts. In machine learning applications, elements are associated with features or classification categories, for example. In our framework, the creation of relationships is encapsulated in mappers. Mappers are very similar to *stored procedures*. What sets them apart are (i) their integrated use in our ranking queries, and (ii) how they are hooked in the databases's API so that any new data that matches the mapping criterion is passed through appropriate mappers.

Considering data as in Figure 5.1, a mapper is invoked whenever a document is added to the database. The mapper extracts features from the document – in this case, tokens or named entities. The features are materialized in the graph, with the new relationships marked as derived from their mapper. In query time (Figure 5.2a), the same mapper (TokenMapper) is invoked to extract features from the keyword query. The features and the keyword query are included as temporary nodes and relationships in the graph. Since all elements are represented in the same model, the query language can reference them as target for the correlation inferences.

5.4 CDMS for ML tasks

Our hypothesis is that the expressiveness of our query language coupled with the mapper mechanism can be an interesting way to represent some ML tasks. In our vision, learning could become indistinguishable from the natural process of database refinement and evolution. More data means better accuracy, i.e. more opportunities for exploring non-trivial patterns and for collective reasoning. Furthermore, refinements to the data, e.g. a classification mistake corrected by a database user, are immediately reflected in the data

to be used by inference queries. Once all elements are represented in the same database model, inferences can be specified as queries in our query language. Optimizations in this scenario become a combination of sampling methods from ML and query optimization plans from DBs.

In our setting, mappers are used to extract features, which by default become part of the database. In classification tasks, the mappers also accumulate information about the correlation between features and outcomes (e.g. posterior probabilities for a bayesian mapper).

5.4.1 Experiments

We tested our approach over real diagnose data compiled for a previously unrelated project with the faculty of nursing at our university [43]. The data consists of matrices SD (for Symptoms $\times$ Diagnoses), PS (for Patients $\times$ Symptoms) and PD (for Patients $\times$ Diagnoses). SD correlates 62 symptoms to 28 diagnoses, all defined in a domain-specific protocol. The strength of the correlations $sd_{i,j} \in \{0, 0.25, 0.5, 0.75, 1\}$ were determined based on consensus in committees of experts. Similarly, PS correlates 6 patients (clinical cases) and their symptoms. The strength of the correlations $ps_{i,j} \in \{0, 0.25, 0.5, 0.75, 1\}$ were also determined by experts. Experts also provided most likely diagnoses for each patient, with strength $pd_{i,j} \in \{0, 0.25, 0.5, 0.75, 1\}$ (these were used as gold standards). The matrices were consolidated in a graph with the strength of the correlations becoming weights for the edges. It is important to notice that the weights defined by experts could be easily produced automatically by mappers if we had enough training data.

Figure 5.3 shows a sample of the graph with the weights of the associations represented as the thickness of the edges. As an example, the patient represented by the node 'Case 1' presents the symptom 'Fatigue' that was classified as having a 1.0 pertinence to the diagnosis 'Caregiver Role Strain' and 0.5 for 'Risk for Falls'. The final graph has 96 nodes and 324 relationships. Our goal was to suggest diagnoses based on the assessment of our metrics in the graph. The template for the queries issued for each patient is shown in Figure 5.2b. $\%r$ and $\%c$ are weights for the relevance and connectivity metrics, $\%p$ is the id of the node of the patient. To represent an instance of the query template we use the notation Rr:cC, meaning that the query was specified with weight r for the relevance metric and weight c for connectivity.

We analyzed the result rankings based on the diagnoses suggested by the committees of experts. We used the diagnoses with total correlation (1) to each patient. We run the queries to assess, for the sets of diagnoses for each patient, how many ranking places were needed to cover all diagnoses. The patients had 1 to 3 diagnoses matching our criteria. The connectivity metric required on average 1.5 extra ranking positions to cover the diagnosis, outperforming relevance which required 2. We then ran the query with different combinations of weights for the metrics, as shown in Figure 5.4. Combining the metrics improved the performance, eventually reducing the required extra positions to 1.17 on average.

The analysis of these experiments suggest that characteristics from both metrics are

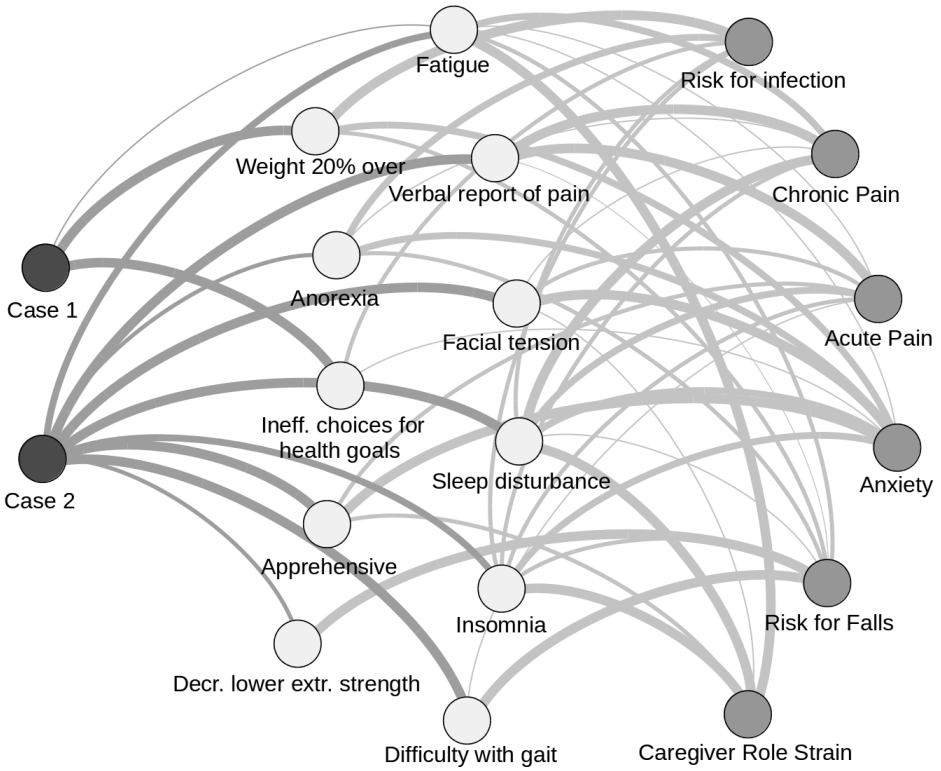


Figure 5.3: Sample subgraph of the dataset for the nursing diagnosis experiments

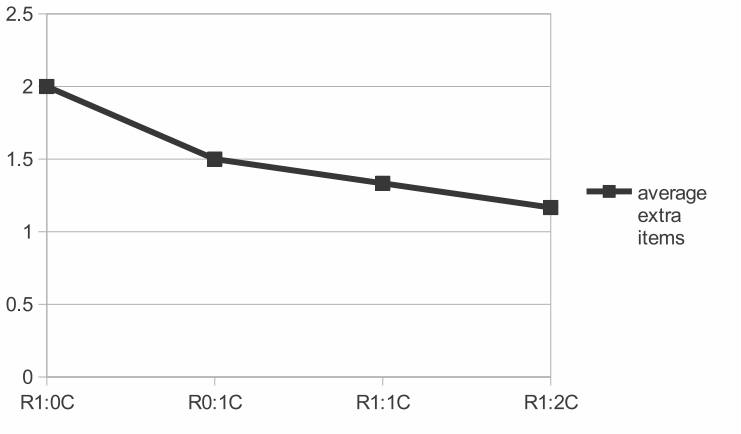


Figure 5.4: Average number of extra ranking items to cover the diagnosis from the expert

important to produce good performing rankings, a trait we conjecture as being pervasive across tasks. The accuracy of our diagnosis suggestions is very practical, especially considering that we only analyzed a fraction of the information available for the experts that defined the gold standard – they also had access to patient's clinical report and a list of patient-specific risk factors.

Most importantly, we were able to produce the diagnosis recommendations from a very simple declarative query, in a development effort of a few minutes. The simple, manual query specification process described here is equivalent to fitting a classification function with the metrics and weights as parameters. In practice, our goal is to automatize the query specification process. The query, with ranking metrics tuned to maximize the likelihood of observing the training data, becomes the classification model.

5.5 Conclusion

There are many challenges and open questions related to this proposal. The search space for query composition is very large, including combinations for metric types, their weights, spreading activation parameters, and relationships to be traversed. Furthermore, query evaluation for large subgraphs and high frequency relationships is inefficient. We have, however, preliminary evidence that the performance challenges may be tractable. In our experiments [34], we observed that good accuracy for the metrics can be achieve with low subgraph radii, and that relevant candidate values for SA parameters are in small ranges. Moreover, the declarative query setting offers opportunities for DB-like optimizations: we are proposing and testing a number of query optimization and approximation techniques. Using advanced sampling techniques will also contribute towards scalability.

Queries as materializations of learned models have several advantages compared to the abstract mathematical models learned in most ML tasks. The queries are easy for a user to interpret, tune and reuse. Moreover, the distribution of metrics, weights, and relationships provide insights on what are the most important patterns underlying the decision process. We also expect this approach to reduce, in many instances, the need for retraining as the database evolves.

We are just beginning to explore applications of our framework in learning tasks. We still need to assess the applicability of the language and the domain of tasks for which its expressiveness is adequate. Our model can only represent discrete features, but we believe it can be extended to include continuous features without a significant increase in complexity. This is an effort originated in the DB field that would greatly benefit from feedback and collaboration with ML researchers.

Chapter 6

Conclusion

This thesis presents data querying and management mechanisms for complex networks. The proposals are combined in the Complex Data Management System (CDMS), a system that offers a query-based and explorative interaction with the underlying complex networks. The CDMS makes the database the reference point for the analysis of complex networks, allowing researchers to focus on data curation efforts, ask queries based on desired parameters, and feed the results back to the database.

To enable the type of interaction we envisage, a CDMS must offer a query language and query processing mechanisms that supports the type of analysis undertaken in typical complex network settings. In this thesis those requirements are met with the Beta-algebra and the in* language. The Beta-algebra extends relational algebra (with aggregation) with an operator for graph-based aggregation of values. The proposed algebra allows the computation of diverse network measurements, capturing properties that indicate the importance of nodes, relevance between nodes, etc. The in* is a language that extends graph matching languages to support ranking based on combinations of measurements written in the Beta-algebra. Throughout this thesis we presented examples of the application of our language and algebra, using real and synthetic data.

We also discussed and tested query optimization and data management mechanisms. For query optimization, we developed heuristics that rely on database statistics to rewrite the queries. While we only implemented intra-operator strategies, we believe that inter-operator and inter-query approaches alongside approximation techniques can greatly improve response time. As for data management, we proposed the *mapper* mechanism, which is responsible for defining relationship between nodes, at insertion or query time. This mechanism is important for data consistency but also to support applications such as information retrieval and classification tasks.

To test our initial hypothesis, we applied the proposed framework in diverse scenarios, including integration of institutional data, diagnose, and geographic recommendation. We tested several aspects of the frameworks using real and synthetic data. The tests showed how our framework can provide a more fine-grained and flexible interaction with complex networks, allowing the type of explorative querying we aim at.

To implement the diverse aspects of the framework, we employed a stack of tech-

nologies. To store the data we used mainly the Neo4J[1] and the Virtuoso[2] databases. The processing module used the Gremlin language and other graph-related technologies from the Tinkerpop project[3]. The parser for the Beta-algebra was implemented using the ANTLR4 generator[4].

Future work

There are several areas in which we want to improve the proposal. Currently, the beta operator is implemented on top of another query processing stack (Blueprint's Gremlin, and Neo4j). A deployment-ready implementation would require the beta operator to be part of the overall query processing and optimization scheme. It is also important to consider distributed implementations of the operator, since the use cases are often too large for a single computer.

The in* language, while simple to use and implement, does not take full advantage of the expressive power of the Beta-algebra. We would like to design a language that exposes the operator and its parameters to the user (explicitly or implicitly), and that supports nested queries.

In terms of application scenarios, we would like to further explore information retrieval and machine learning tasks. For information retrieval, it would be interesting to introduce inverted indexes to the CDMS. The idea is that the index would represent the links that are currently made explicit. In this setting, traversal of word nodes would lead to document nodes, by means of a transparent index lookup. As for machine learning tasks, we would like to formalize the statistical properties of the classification tests that we made and compare them with traditional techniques such as Naive Bayes. We could then generalize to other models to determine the limits of our query approach.

In the long term, we believe that the CDMS could play an important role in a scenario of increasingly connected data. As we produce more data and find different means of connecting it, it becomes difficult to model and curate the data to guarantee consistency and to allow for meaningful queries. An expressive query language, capable of deriving knowledge with minimum or no awareness of the structure of the graph, would reduce data curation needs. We envision a scenario where data and links are stored in large scale, without the need for detailed modeling. It is then at query time that the users make sense of the data, issuing queries that analyze the relationships and dynamic properties on demand.

Funding

Work partially financed by CAPES and FAPESP [grant numbers 2012/15988-9, 2014/01419-8], FAPESP/Cepid in Computational Engineering and Sciences (2013/08293-

[1] http://neo4j.com
[2] http://http://virtuoso.openlinksw.com/
[3] http://tinkerpop.incubator.apache.org
[4] http://www.antlr.org

7), the Microsoft Research FAPESP Virtual Institute (NavScales project), CNPq (MuZOO 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.

Bibliography

[1] R. Agrawal. Alpha: An extension of relational algebra to express a class of recursive queries. *IEEE Trans. on Softw. Eng.*, 14(7):879, July 1988.

[2] Yong-Yeol Ahn, Sebastian E. Ahnert, James P. Bagrow, and Albert-László Barabási. Flavor network and the principles of food pairing. Sci. Rep. 1, 196;, 2011.

[3] Hugo Alves and André Santanchè. Abstract framework for social ontologies and folksonomized ontologies. In *SWIM*. ACM, 2012.

[4] Sihem Amer-Yahia, Pat Case, Thomas Rölleke, Jayavel Shanmugasundaram, and Gerhard Weikum. Report on the DB/IR panel at SIGMOD 2005. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 34(4):71–74, December 2005.

[5] Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Computing Surveys*, 40(1):1–39, February 2008.

[6] Sören Auer, Sebastian Dietzold, Jens Lehmann, Sebastian Hellmann, and David Aumueller. Triplify: light-weight linked data publication from relational databases. In *WWW*, 2009.

[7] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *IJCAI*, 2007.

[8] Pablo Barceló, Leonid Libkin, Anthony Widjaja Lin, and Peter T. Wood. Expressive languages for path queries over graph-structured data. *ACM Trans. Database Syst*, 37(4):31, 2012.

[9] T. Berners-Lee. Giant global graph. online posting, 2007. http://dig.csail.mit.edu/breadcrumbs/node/215.

[10] Christian Bizer. D2rq - treating non-rdf databases as virtual rdf graphs. In *Proceedings of the 3rd International Semantic Web Conference (ISWC2004)*, 2004.

[11] Roi Blanco and Christina Lioma. Graph-based term weighting for information retrieval. *Inf. Retr*, 15(1):54–92, 2012.

[12] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022, 2003.

[13] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[14] Nicolas Bruno and Surajit Chaudhuri. Automatic physical database tuning: A relaxation-based approach. In Fatma Özcan, editor, *SIGMOD Conference*, pages 227–238, 2005.

[15] Razvan C. Bunescu and Raymond J. Mooney. Collective information extraction with relational markov networks. In *ACL*, pages 438–445, 2004.

[16] Surajit Chaudhuri, Raghu Ramakrishnan, and Gerhard Weikum. Integrating DB and IR technologies: What is the sound of one hand clapping? In *CIDR*, pages 1–12, 2005.

[17] E. F. Codd. A relational model of data for large shared data banks. *Comm. ACM*, 13(6):377–387, June 1970.

[18] Mariano Consens and Alberto O. Mendelzon. Graphlog: a visual formalism for real life recursion. In *In Proceedings of the Ninth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 404–416, 1990.

[19] Fabio Crestani. Application of spreading activation techniques in information retrieval. *Artif. Intell. Rev*, 11(6):453–482, 1997.

[20] Richard Cyganiak. A relational algebra for SPARQL. Technical Report HPL-2005-170, Hewlett Packard Laboratories, May 21 2005.

[21] L. da F. Costa, O. Oliveira Jr, G. Travieso, F. Rodrigues, P. Boas, L. Antiqueira, M. Viana, and L. Rocha. Analyzing and modeling real-world phenomena with complex networks: A survey of applications. *Advances in Physics*, 60:329–412, 2011.

[22] L. da F. Costa, F. A. Rodrigues, G. Travieso, and P. R. V. Boas. Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242, 2007.

[23] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems*. Addison-Wesley, 6 edition, 2011.

[24] Ernesto Estrada, Desmond J. Higham, and Naomichi Hatano. Communicability betweenness in complex networks. *Physica A: Statistical Mechanics and its Applications*, 388(5):764 – 774, 2009.

[25] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in KnowItAll. In *WWW*, March 26 2004.

[26] Flavius Frasincar, Geert-Jan Houben, Richard Vdovjak, and Peter Barna. RAL: An algebra for querying RDF. *World Wide Web*, 7(1):83–109, 2004.

[27] Lise Getoor and Christopher P. Diehl. Link mining: a survey. *SIGKDD Explor. Newsl.*, 7(2):3–12, December 2005.

[28] Jr. Gomes, Luiz and André Santanchè. The web within: Leveraging web standards and graph analysis to enable application-level integration of institutional data. In *Transactions on Large-Scale Data- and Knowledge-Centered Systems XIX*, Lecture Notes in Computer Science, pages 26–54. Springer Berlin Heidelberg, 2015.

[29] Luiz Gomes-Jr, Bernd Amann, Luciano Costa, and André Santanchè. Integrating databases and complex networks. *submitted to PLOS ONE*, 2015.

[30] Luiz Gomes-Jr, Bernd Amann, and André Santanchè. Beta-algebra: Towards a relational algebra for graph analysis. In *GraphQ-EDBT*, 2015.

[31] Luiz Gomes-Jr, Luciano Costa, and André Santanchè. Querying complex data. Technical Report IC-13-27, Institute of Computing, University of Campinas, October 2013.

[32] Luiz Gomes-Jr, Rodrigo Jensen, and André Santanchè. Query-based inferences in the Complex Data Management System. In *Structured Learning: Inferring Graphs from Structured and Unstructured Inputs (SLG-ICML)*, 2013.

[33] Luiz Gomes-Jr, Rodrigo Jensen, and André Santanchè. Towards query model integration: topology-aware, ir-inspired metrics for declarative graph querying. In *GraphQ-EDBT*, 2013.

[34] Luiz Gomes-Jr and André Santanchè. The Web Within: leveraging Web standards and graph analysis to enable application-level integration of institutional data. Technical Report IC-13-01, Institute of Computing, University of Campinas, January 2013.

[35] Luiz Gomes-Jr and André Santanchè. The Web Within: leveraging Web standards and graph analysis to enable application-level integration of institutional data. *Transactions on Large Scale Data and Knowledge Centered Systems*, 2014.

[36] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2006.

[37] Oktie Hassanzadeh and Mariano Consens. Linked movie data base. In *Proceedings of the 2nd Workshop on Linked Data on the Web (LDOW2009)*, 2009.

[38] Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. In *IEEE Transactions on Visualization and Computer Graphics*, volume 6 (1), pages 24–43. IEEE Computer Society, 2000.

[39] Ihab F. Ilyas, George Beskales, and Mohamed A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys*, 40(4):11:1–11:58, October 2008.

[40] Claudia Imhoff, Nicholas Galemmo, and Jonathan G. Geiger. *Mastering Data Warehouse Design: Relational and Dimensional Techniques*. Wiley, 2003.

[41] Matthias Jarke, Maurizio Lenzerini, Yannis Vassiliou, and Panos Vassiliadis. *Fundamentals of Data Warehouses*. Springer, 2003.

[42] R. Jensen, M. Cruz, L. Gomes-Jr, and M. Lopes. Attributing fuzzy values to nursing diagnoses and their elements: The specialists' opinion. *International Journal of Nursing Knowledge*, 24:134–141, 2013.

[43] R. Jensen, P. S. P. Silveira, N. R. S. Ortega, and M. H. B. de M. Lopes. Software application that evaluates the diagnostic accuracy of nursing students. *Intl Journal of Nursing Knowledge*, 23:163–171, 2012.

[44] Kimelfeld and Sagiv. Finding and approximating top-k answers in keyword proximity search. In *PODS: 25th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 2006.

[45] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of 18th Int. Conf. on Machine Learning*, 2001.

[46] Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M. Hellerstein. Distributed graphlab: A framework for machine learning and data mining in the cloud. *Proc. VLDB Endow.*, 5:716–727, 2012.

[47] Yi Luo, Wei Wang 0011, Xuemin Lin, Xiaofang Zhou, Jianmin Wang 0001, and Keqiu Li. SPARK2: Top-k keyword query in relational databases. *IEEE Trans. Knowl. Data Eng*, 23(12):1763–1780, 2011.

[48] Grzegorz Malewicz, Matthew H. Austern, Aart J. C. Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *SIGMOD*, 2010.

[49] Shaul Markovitch and Evgeniy Gabrilovich. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007.

[50] Jim Melton and Stephen Buxton. *Querying XML: XQuery, XPath, and SQL/XML in context*. The Morgan Kaufmann series in data management systems. Morgan Kaufmann Publishers, 2006.

[51] M. Newman. The structure and function of complex networks. *SIREV: SIAM Review*, 45, 2003.

[52] Axel-cyrille Ngonga Ngomo, Norman Heino, and Martin Kaltenb. SCMS - Semantifying Content Management Systems. In *ISWC*, 2011.

[53] Georgios A. Pavlopoulos, Anna-Lynn Wegener, and Reinhard Schneider. A survey of visualization tools for biological network analysis. *BioData Mining*, 1, 2008.

[54] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[55] Marko A. Rodriguez and Peter Neubauer. The graph traversal pattern. *CoRR*, abs/1004.1001, 2010.

[56] Marko A. Rodriguez, Alberto Pepe, and Joshua Shinavier. The Dilated Triple. In *Emergent Web Intelligence: Advanced Semantic Technologies*, pages 3–16. Springer London, June 2010.

[57] Sunita Sarawagi. Information extraction. *Foundations and Trends in Databases*, 1(3):261–377, 2008.

[58] Simon Schenk and Steffen Staab. Networked graphs: a declarative mechanism for SPARQL rules, SPARQL views and RDF data integration on the web. In *WWW*, 2008.

[59] Andreas Schwarte, Peter Haase, Katja Hose, Ralf Schenkel, and Michael Schmidt. FedX: A federation layer for distributed query processing on linked open data. In *ESWC (2)*, volume 6644 of *Lecture Notes in Computer Science*, pages 481–486. Springer, 2011.

[60] A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[61] Ramakrishna Varadarajan, Vagelis Hristidis, Louiqa Raschid, Maria-Esther Vidal, Luis Daniel Ibáñez, and Héctor Rodríguez-Drumond. Flexible and efficient querying and ranking on hyperlinked data sources. In *EDBT*, pages 553–564, 2009.

[62] Gerhard Weikum, Gjergji Kasneci, Maya Ramanath, and Fabian Suchanek. Database and information-retrieval methods for knowledge discovery. *Communications of the ACM*, 52(4):56–64, April 2009.

[63] Scott White and Padhraic Smyth. Algorithms for estimating relative importance in networks. In *SIGKDD*, 2003.

[64] Peter T. Wood. Query languages for graph databases. *SIGMOD Record*, 41(1):50–60, 2012.