Representação e Consulta da Informação Imperfeita em Bancos de Dados de Grafos Nebulosos

Bruno Ponsoni Costa

Dezembro / 2018

Dissertação de Mestrado em Ciência da Computação

Representação e Consulta da Informação Imperfeita em Bancos de Dados de Grafos Nebulosos

Esse documento corresponde à Dissertação apresentado à Banca Examinadora no curso de Mestrado em Ciência do Centro Universitário Campo Limpo Paulista.

Campo Limpo Paulista, 13 de dezembro de 2018.

Bruno Ponsoni Costa

Prof. Dr. Luis Mariano del Val Cura
(Orientador)

FICHA CATALOGRÁFICA

Ficha catalográfica elaborada pela
Biblioteca Central da Unifaccamp

C87r

Costa, Bruno Ponsoni

Representação e consulta da informação imperfeita em bancos de dados de grafos nebulosos / Bruno Ponsoni Costa. Campo Limpo Paulista, SP: Unifaccamp, 2018.

Orientador: Profº. Drº. Luis Mariano del Val Cura

Dissertação (Programa de Mestrado em Ciência da Computação) – Centro Universitário Campo Limpo Paulista – Unifaccamp.

1. Bancos de dados. 2. NoSQL. 3. Banco de dados de grafos. 4. Conjuntos nebulosos. 5. Informações imperfeitas. I. Del Val Cura, Luiz Mariano. II. Campo Limpo Paulista. III. Título.

CDD – 005.75

Agradecimentos

Agradeço primeiramente a Deus por proporcionar a oportunidade de concluir este programa de mestrado, pois sem ele nada seríamos. Aos meus pais, por me apoiarem na decisão de enfrentar este desafio. Em especial, ao meu pai *Hadide* que apoiou financeiramente em diversos momentos de dificuldade. Dedico está conquista à minha mãe *Dilza*, que mesmo sofrendo de *Alzheimer* continua a se preocupar com a segurança de todos e incentivar que novos desafios sejam alcançados. Agradeço ao meu orientador *prof. Dr. Luis Mariano del Val Cura* pela paciência e ensinamentos nestes anos, além da excelente orientação prestada durante todo o processo de desenvolvimento deste trabalho. Agradeço também a meu colega e amigo *Victor Martins de Sousa* que acompanhou desde o início, nestes milhares de quilômetros de viagens de carro. A *Anderson Francisco de Oliveira*, *Bruno do Amaral* e *Vagner Scamati*, pelos ótimos momentos de estudo, companheirismo nas dificuldades e nos momentos de descontração, sem eles tudo seria mais difícil. Agradeço também ao *Instituto Federal de São Paulo, Campus Registro* por permitir minha ausência nos dias de realização deste mestrado. Do mesmo modo, aos meus colegas e amigos da *Coordenadoria de Tecnologia da Informação*, que supriram minhas ausências, apoiaram e motivaram nos momentos difíceis, *Kelcey Baldoino*, *Marcio Teobaldino*, *Paulo César de Oliveira* e *Victor Calrquist*. Este trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior-Brasil (CAPES) - Código de Financiamento 001.

Dedicatória

Dedico está conquista à minha família, em especial à minha futura esposa Madisara Verônica Lima Hataka, que apoiou e incentivou desde o início, teve paciência com minhas queixas e o cansaço das viagens. Sem ela nada disso seria possível, espero que isso possa proporcionar orgulho à ela e aos nossos filhos que virão. Este é somente mais um passo, pois na vida o aprendizado é constante, se fizermos somente o que já sabemos, nunca nos tornaremos nada além do que já somos.

Resumo.*Informações imperfeitas são caracterizadas por dados que não possuem sentido completo, sendo imprecisos ou vagos. Estes dados são classificados em cinco tipos principais pertinentes à forma da imperfeição existente, sendo eles: imprecisão, incerteza, vagueza, ambiguidade e inconsistência. Em bancos de dados tradicionais, um dado imperfeito se apresenta com certa frequência, mas de modo geral não definimos os valores para registro. Com os conceitos da teoria dos conjuntos nebulosos, a informação imperfeita pode ser definida descrevendo para cada dado uma função de pertinência. Foram desenvolvidos trabalhos em bancos de dados relacionais, orientados a objetos e semiestruturados de tipo nebuloso, visando o aprimoramento da integração dos conjuntos nebulosos com os dados imperfeitos. Esta dissertação, trata da informação imperfeita no modelo de banco de dados de grafos. Exploramos as formas de inserção, representação e consultas da informação imperfeita no modelo de bancos de dados de grafos a partir de a ferramenta SUGAR. Esta ferramenta foi modificada para permitir a inserção de novos dados nebulosos no banco de dados de grafos. Além disso, a dissertação apresenta um caso de uso para o desenvolvimento de uma aplicação que integra estes novos conceitos. Como parte deste trabalho foram apresentados comandos para a linguagem Cypher do sistema de bancos de dados de grafos Neo4J.*

Palavras-chave*: Bancos de Dados, NoSQL, Banco de Dados de Grafos, Conjuntos Nebulosos, Informações Imperfeitas.*

Abstract.*Imperfect information is characterized by data that does not have complete meaning, being inaccurate or vague. These data are classified into five main types, pertinent to the form of the existing imperfection, being: imprecision, uncertainty, vagueness, ambiguity and inconsistency. In traditional databases, an imperfect data is presented with some frequency, but in general we do not define the values for registration. With the concepts of fuzzy sets theory, imperfect information can be defined by describing for each data a function of pertinence. Works were developed on relational, object - oriented and semi-structured nebulous-type databases, aimed at improving the integration of fuzzy sets with imperfect data. This dissertation deals with imperfect information in the graph database model. We explore the forms of insertion, representation and queries of imperfect information in the graph database model from the SUGAR tool. This tool has been modified to allow the insertion of new fuzzy data in the graph database. In addition, the dissertation presents a use case for the development of an application that integrates these new concepts. As part of this work we have presented commands for the Cypher language of the Neo4J graph database system.*

Keywords*: Databases, NoSQL, Graph Database, Fuzzy Sets, Imperfect Information.*

Sumário

Glossário

BDO *Banco de Dados de Objetos*

BDOO *Banco de Dados Orientados a Objetos*

DDDN *Documento de Definição de Dados Nebulosos*

DTD *Document Type Definition*

FOODB *Fuzzy Oriented-Object Database*

FOQL *Fuzzy Object Query Language*

FRDBM *Fuzzy Relational Database Model*

GEFRED *Generalized Model of Fuzzy Relational Databases*

NoSQL *Not Only SQL*

OID *Object Identification*

OQL *Object Query Language*

SQL *Structured Query Language*

SUGAR *System Based on Fuzzy Theory for (fuzzy) Graph Databases Querying*

XML *Extensible Markup Language*

XSD *XML Schema Definition*

Lista de Tabelas

Lista de Figuras

CAPÍTULO 1

Introdução

1.1. Contexto e Motivação

O pensamento humano é capaz de associar diferentes informações a fim de responder suas dúvidas. Em questionamentos feitos a partir de dados incompletos, imprecisos ou vagos, ocorre uma associação semelhante. Como consequência, obtemos no lugar de uma resposta exata um conjunto de respostas possíveis, onde podemos julga-las como aceitáveis ou não para a situação.

De modo geral, podemos classificar os referidos dados como *"dados imperfeitos"*, (Smets 1997). Dados imperfeitos são adquiridos no dia-a-dia de diversas formas, podendo ser através da audição, visão, tato, entre outros. Naturalmente, os dados imperfeitos acabam sendo relacionados a outros dados e fazendo com que se tornem uma informação útil.

Podemos tomar como exemplos de informações imperfeitas, as afirmações: *"Eu estou com muito calor"* ou *"Hoje está quente"*. Os termos sublinhados na afirmação *"muito"*, *"calor"* e *"quente"* são exemplos de termos que podem ser classificados como imperfeitos. O motivo disto é que sozinhos não descrevem de forma precisa o seu significado. O termo *"quente"* não define de forma precisa qual o grau da temperatura a que se refere.

Nesse sentido, ao analisarmos os termos *"calor"* e *"quente"* individualmente, podemos associá-los a um conjunto de possíveis temperaturas. Analisando o con-

texto geral destas afirmações, podemos considerar que eles fazem referência à temperatura de um dia qualquer. O termo "muito" atua ainda como um intensificador do termo *"calor"*, de modo a expressar valores superiores do conjunto de temperaturas.

Esta análise permite a compreensão geral da informação transmitida pelas afirmações. Todavia, pode-se afirmar a qual grau de temperatura estamos nos referindo ao utilizamos os termos *"muito calor"* e *"quente"*? É possível relacionar a expressão *"muito calor"* com *"dia quente"*, tendo como referência as poucas informações analisadas?

Nos sistemas de informação, a definição e tratamento de dados imperfeitos não decorrem de forma distinta dos exemplos mencionados (Smets 1997). Dados imperfeitos são captados a todo instante, necessitando um armazenamento correto nos bancos de dados para estarem disponíveis para consultas. Entretanto, nem todos os dados conseguem ser armazenados de forma perfeita, seja pela escolha da tecnologia de bancos de dados utilizada, ou ainda, pela semântica, ocasionando o descarte.

Vamos supor que necessitamos registrar dados referentes à temperatura diária, com base nas informações obtidas de diversos usuários. Utilizando o modelo relacional de bancos de dados, o conjunto das respostas destes usuários pode ser modelado e armazenado como uma relação, com domínios definidos para cada atributo:

$$R = \{\text{usuário, dia, período, temperatura}\}$$

Em particular, no atributo *temperatura* define-se o domínio como inteiro, pois espera-se a representação de valores exatos. No entanto, ao realizar o registro das informações nessa relação, chegaremos a seguinte coleção de tuplas, baseadas nas experiências dos diferentes usuários, que não se correspondem com esse domínio:

$$R_1 = \{\text{“João”, 1, “matutino”, “quente”}\},$$
$$R_2 = \{\text{“Pedro”, 1, “matutino”, “38 graus”}\},$$
$$R_3 = \{\text{“Maria”, 1, “matutino”, “muito quente”}\},$$
$$R_4 = \{\text{“Carlos”, 1, “matutino”, 39}\},$$
$$R_5 = \{\text{“Bento”, 1, “matutino”, “aprox.40”}\},$$
$$R_x = \{\ldots\}$$

Observa-se que diferentes visões podem ser coletadas para o caso de temperaturas. Em alguns casos, estes dados não podem ser representados em seus domínios de maneira tradicional. Se limitarmos a um valor inteiro, acarretará a impossibilidade do registro destas informações.

Segundo Petry (2012), para permitir representar e gerenciar os diferentes tipos de informações, os conceitos de lógica e conjuntos nebulosos foram incorporados em bancos de dados. A integração destas duas áreas ocasionou o surgimento dos bancos de dados nebulosos. Atualmente existem inúmeras pesquisas realizadas nos principais modelos de bancos de dados, o modelo relacional, orientado a objetos e documentos *XML*.

A teoria dos conjuntos nebulosos de Zadeh *et al.* (1965) surgiu como formalismo para modelar matematicamente dados imperfeitos. A partir dos conceitos dessa teoria, um dado imperfeito pode ser definido ao adicionar um valor que faça referência ao grau de participação ou pertinência deste elemento com seu conjunto completo de valores possíveis. Esta associação é denominada *"dado nebuloso"* e torna possível a sua interpretação através de operações nebulosas específicas.

Este trabalho visa a representação e consulta da informação imperfeita em bancos de dados de grafos nebulosos. Deste modo, faz-se necessário a apresentação de uma visão geral da ocorrência deste tipo de dado imperfeito em outros modelos de bancos de dados. Cada modelo de banco de dados possui características e restrições distintas que serão abordados no decorrer deste trabalho.

Inicialmente, a tentativa de representação e manipulação de dados nebulosos em bancos de dados ocorreu no modelo relacional Ma & Yan (2016). Considera-se a ocorrência da informação imperfeita em atributos e no relacionamento dos atributos de uma tupla. Abordamos ainda a ocorrência da informação imperfeita nos bancos de dados de objetos. Como consequência, aborda-se a ocorrência deste tipo de informação em atributos, objetos, classes, relacionamento de objetos/objetos ou objetos/classes e, por fim, classes/superclasses ou classes/subclasses.

A ocorrência da informação imperfeita e os dados nebulosos são apresentados também no modelo de bancos de dados baseados em documentos *XML*. Conside-

rando a flexibilidade já existente nos documentos *XML*, estes dados são incorporados por meio dos *Document Type Definition (DTD)* e *XML Schema Definition (XSD)*. Destinamos o terceiro capítulo para apresentar os conceitos que envolvem este tipo de informação nos diferentes modelos de documentos baseados em *XML*.

O modelo de bancos de dados de grafos ficou famoso no início dos anos 90, como alternativa para representar a complexidade das informações e as relações que as envolvem, baseando-se no modelo matemático de grafos, (Kaliyar 2015). Recentemente, houve o surgimento de novas aplicações de bancos de dados, conhecidas com *BigData*, com novas exigências para o armazenamento e o processamento da informação. Como resposta a estas exigências, novos modelos de dados estão sendo desenvolvidos, conhecidos como *NoSQL*, com gerenciadores de dados que atendem a estas exigências. Dentre os modelos de dados *NoSQL*, os modelos de dados baseado em grafos tem ressurgido como uma das alternativas para a modelagem e implementação de aplicações *BigData*.

Atualmente, os gerenciadores de bancos de dados de grafos comerciais não possuem recursos para a representação e consulta de informação nebulosa. Trabalhos recentes, como o de Pivert *et al.* (2016), têm realizado propostas com objetivo de incluir dados nebulosos e possibilitar a realização de consultas nebulosas em bancos de dados de grafos.

Quando desejamos representar uma informação imperfeita no modelo de grafos, fazemos uso de seus vértices e arestas, de modo que possa expressar a complexidade que este tipo de informação possui. Inclui também utilizar as propriedades que compõem um vértice, os rótulos de uma aresta ou a soma dos dois, possibilitando a análise da estrutura nebulosa do relacionamento dos dados que a constituem.

Com base nos estudos observados na área de banco de dados de grafos, nota-se que a maioria destes é direcionada ao desenvolvimento de consultas flexíveis. Estas propostas buscam incluir os conceitos da lógica nebulosa nas instruções, de modo a permitir a quebra da sintaxe tradicional de consulta. Por outro lado, somente alguns fazem referência à representação de dados nebulosos no modelo e ainda assim de forma superficial, sem apresentar uma forma concreta de como estes dados foram

obtidos. Nos trabalhos de Pivert *et al.* (2016), o objetivo foi a representação de consultas nebulosas em bancos de dados de grafos, mas representando, basicamente, a informação de forma perfeita. No nosso projeto, consideramos adicionalmente a inclusão dentro do banco de dados de grafos de informação imperfeita.

Considerando a complexidade para que os dados imperfeitos se transformem em informação útil, esta pesquisa visa modelar a representação, inserção e consulta de dados imperfeitos e dados perfeitos nos bancos de dados de grafos. Em conjunto com a lógica nebulosa, buscamos possibilitar que estes diferentes tipos de dados sejam tratados de forma igual pelo sistema. Busca-se ainda possibilitar uma representação visual, além de desenvolver instruções de consultas que sejam suficientemente compatíveis com o novo modelo proposto.

1.2. Objetivos

O objetivo geral deste trabalho é apresentar um modelo de inserção, representação e consulta da informação imperfeita em bancos de dados de grafos, utilizando os conceitos da lógica nebulosa. Este modelo deve se integrar de forma natural aos modelos tradicionais já existentes, que representam a informação perfeita e cujas consultas são baseadas na lógica booleana.

Especificamente pretende-se:

I. Propor um modelo de representação da informação imperfeita em banco de dados de grafos. Para este objetivo, deverá ser estendido um dos modelos lógicos de banco de dados de grafos da literatura, adicionando recursos para representar dados imperfeitos em arestas e vértices;
II. Definir um modelo de consulta sobre o modelo de representação proposto. O modelo de representação deve permitir que consultas nebulosas sejam feitas sobre dados imperfeitos ou não;
III. Propor uma extensão para a linguagem *Cypher*, do sistema de gerenciamento de bancos de dados de grafos *Neo4j*;
IV. Implementar um protótipo de sistema baseado no *Neo4j* para validar estas propostas.

1.3. Metodologia da Pesquisa

A metodologia desta dissertação segue os seguintes procedimentos metodológicos:

- **Revisão Bibliográfica:** A princípio buscou-se refletir sobre como a forma que as imperfeições nas informações contidas nos bancos de dados são encaradas. Para este fim realizou-se uma revisão sistemática sobre a representação da informação imperfeita nas principais estruturas de bancos de dados existentes atualmente, classificados em dados estruturados, semiestruturados e não estruturados. Por este motivo realizou-se uma pesquisa sobre os estudos existentes e as abordagens realizadas sobre o tema, através de bases de pesquisa como *ACM Digital Library*, *IEEE Xplore*, *Google Acadêmico*, entre outras. Foram utilizados *surveys* com boa aceitação e procura como forma de orientação, para relacionar os diversos estudos existentes de cada modelo a ser apresentado. A classificação resultante foi separada pela visão da ocorrência da informação nebulosa e formas de consultas da informação nebulosa;
- **Proposta de novo modelo:** Foram integradas as diferentes abordagens na bibliografia para desenvolvimento de uma nova proposta de gerenciamento de informação imperfeita para grafos. A referida proposta tem como parâmetros: *(a)* a possibilidade de representar dados imperfeitos no banco de dados e *(b)* a possibilidade de realizar consultas sobre qualquer tipo de dados utilizando operadores nebulosos;
- **Estudo das características da linguagem *Cypher* de Neo4j:** O objetivo do trabalho é a extensão da linguagem de definição de dados e de consulta de um sistema de gerenciamento de bancos de dados de grafos. Como parte desta fase, foi estudada em detalhes a sintaxe e semântica da linguagem *Cypher* do sistema *Neo4j*[1]. A extensão da linguagem deve incluir como recursos as ideias fundamentais da proposta de modelo e deve garantir a integração natural das novas extensões aos comandos já existentes em *Neo4j*;
- **Validação dos conceitos:** Para validar nossa proposta será desenvolvida uma aplicação protótipo que funcionará como uma camada intermediaria entre o usuário e o sistema *Neo4j*. Uma aplicação foi desenvolvida e utilizada

[1] *https://neo4j.com*

de modo a oferecer suporte entre o usuário e o banco de dados de grafos. As instruções deverão ser convertidas em instruções aceitas pelo sistema atual. Para tanto, o protótipo utilizará um *REPL* (*Read-Eval-Print-Loop*) disponibilizado pela comunidade do *Neo4j*, visto que o sistema oficial é proprietário e não possui código aberto. Um *REPL* consiste de um console simplificado baseado nas funções do sistema. No caso do *Neo4j* é disponibilizado uma aplicação denominada *"Rabbithole"*. Esta aplicação tem a característica de ser interativa com o desenvolvimento de novas funcionalidades, podendo ler expressões desenvolvidas, avaliar, executar e imprimir o resultado gerado. Isto a torna suficiente para o desenvolvimento inicial do protótipo. Com o funcionamento satisfatório do protótipo, tornam-se válidos os conceitos sobre a integração de diferentes tipos de dados em um sistema baseado em grafos, proposto neste trabalho.

1.4. Organização da Dissertação

Esta dissertação segue organizada nos seguintes capítulos.

No *Capítulo 1* são apresentados a contextualização e objetivos do trabalho.

O *Capítulo 2* introduz definições básicas sobre os conceitos de informações imperfeitas juntamente com a definição dos conceitos iniciais da teoria dos conjuntos nebulosos. A princípio, realiza-se uma análise da terminologia utilizada para descrever este tipo de informação. Pode-se assim, entender melhor a natureza e a característica de uma informação imperfeita. Posteriormente, apresenta-se um estudo da teoria dos conjuntos nebulosos como um método de definição de dados imperfeitos. Este capítulo tenta elucidar através de alguns exemplos, dados considerados imperfeitos e seus grupos de classificação frente à informação a qual está relacionada. Inclui também a definição do modelo matemático que oferece suporte a este tipo de informação.

No *Capítulo 3* são categorizadas as abordagens relacionadas aos modelos de estruturas de dados, com formas de representação e manipulação da informação nebulosa. Os diferentes tipos de estruturas de dados são apresentados como forma de elucidar abordagens através de estudos já realizados, a fim de indicar possíveis

caminhos. Apresenta as distintas representações nas estruturas de dados, bem como as formas de construção de instruções de consultas propostas pelas suas principais linguagens.

O *Capítulo 4* aborda os conceitos específicos dos bancos de dados de grafos, além de algumas abordagens já realizadas com objetivo de incorporar a informação nebulosa ao modelo. Este capítulo complementa os tipos de estrutura de dados definidos no capítulo anterior, contudo, visa destacar pontos importantes referentes ao modelo de grafos.

No *Capítulo 5* define-se o modelo proposto. Apresenta detalhes da idealização do modelo genérico de dados nebulosos em grafos. Relaciona o modelo proposto com à linguagem de consulta Cypher do Neo4j e apresenta conceitos iniciais para implementação da aplicação de validação.

O *Capítulo 6* apresenta os resultados obtidos durante a fase de implementação da aplicação proposta. Apresenta também os resultados obtidos pelo uso da aplicação desenvolvida em um caso de uso, como forma de validar o uso desta aplicação.

No *Capítulo 7* estão relacionadas as conclusões obtidas no decorrer do desenvolvimento deste trabalho. Além da apresentação de possíveis direcionamentos e propostas de desenvolvimento para trabalhos futuros.

CAPÍTULO 2

Informações Imperfeitas e Conjuntos Nebulosos

Neste capítulo abordamos o referencial teórico relacionado com o projeto de pesquisa proposto. Inicialmente é exposta uma classificação dos diferentes tipos ou formas em que se apresenta a imperfeição na informação. Na sequência são apresentadas as definições e conceitos fundamentais da teoria de conjuntos nebulosos que será o arcabouço matemático que nos permitirá modelar a informação imperfeita.

2.1. Classificação das Imperfeições nas Informações

O conceito de informação é indicado na literatura como uma reunião dos conhecimentos dos dados sobre um assunto ou pessoa, ou seja, a ação ou efeito de informar ou de se informar, (Dicio 2016). Deste modo a informação necessita ser completa e coerente para assim contribuir para que o usuário possa utilizá-la como uma fonte confiável de conhecimento e que possa contribuir nas tomadas de decisões. As informações imperfeitas existem em todos os lugares, possuem a característica de serem incompletas, vagas ou imprecisas, (Parsons 1996). Em particular para sistemas de banco de dados, é inevitável o armazenamento de informações imperfeitas, visto que boa parte dos dados não estão formatados de maneira correta para o seu perfeito armazenamento.

Um banco de dados é definido como uma coleção de dados relacionados que representam informações reais. Projetado essencialmente para organização e registro de informações, pode ter os mais variados propósitos, (Elmasri, Navathe, Pinheiro *et al.* 2005). Informações imperfeitas podem surgir em um banco de dados, seja pela

falta de conhecimento completo sobre esses dados, pela inserção de dados redundantes com múltiplos valores possíveis ou pela ausência de dados, registrando apenas segmentos de uma informação. Consideramos como segmento de informação um dado que não representa a informação completa individualmente, sendo necessário adicioná-lo a um contexto geral. De modo geral, os bancos de dados tradicionais não oferecem recursos para representar, gerenciar ou consultar informações imperfeitas. No decorrer da presente pesquisa, apresentam-se as abordagens para representação e gerenciamento da informação imperfeita em bancos de dados.

Em Ma & Yan (2007), Yan, Ma & Liu (2009) e Ma & Yan (2010) apresentam-se classificações para os trechos de informação, considerados como imperfeitos. A referida classificação foi baseada no estudo de Parsons (1996) e nos mostra a relevância que o gerenciamento de dados imperfeitos tem para as aplicações do mundo real. Podemos classificar estes segmentos em cinco tipos principais: *imprecisão*, *incerteza*, *ambiguidade*, *inconsistência* e *vagueza*. A classificação mencionada caracteriza as diferentes formas de como a informação incompleta aparece ou pode ser representada.

Cada um dos cinco tipos de informação incompleta será apresentado a seguir, junto com alguns exemplos para ilustrar as pequenas diferenças entre cada um deles.

2.1.1. Imprecisão

O termo imprecisão é descrito pelo dicionário como uma característica ou particularidade daquilo que é impreciso, que tem ausência de precisão, exatidão e/ou clareza. Seu conceito está geralmente ligado à definição de valores possíveis, (Dicio 2016).

Em um banco de dados, uma informação imprecisa pode se apresentar como um conjunto de valores possíveis para um determinado tipo de dado, definidos a partir de um conhecimento prévio ou suposição. Para exemplificar, supõe-se uma consulta para saber a idade de uma pessoa de nome *"João"*. Para um banco de dados tradicional, em geral o valor retornado seria a quantidade de anos de uma pessoa, o que corresponde a uma informação precisa. Todavia, a informação da idade poderia também ser armazenada de forma imprecisa, considerando mais de um valor para o registro ou através de uma descrição dessa idade. Por exemplo, esta idade

poderia ser apresentada como o conjunto {40, 50, 52, 60} de valores possíveis, por uma descrição aproximada como: *"João tem aproximadamente 50 anos"*, ou ainda como: *"João tem pelo menos 50 anos"*. Consideramos como um conjunto de valores possíveis, qualquer sequência de valores que possam ser associados logicamente ao sujeito que está sendo referenciado.

2.1.2. Incerteza

Uma informação que provoque incerteza deixa de ser confiável. O termo incerteza é definido na literatura como uma condição ou natureza do que é incerto, qualidade daquilo que incita dúvida ou indecisão, (Dicio 2016). O conceito de incerteza se aplica ao grau de verdade que pode ser atribuído a uma informação. Diferente da imprecisão, uma incerteza não é somente composta por uma série de valores, como também pela quantificação do quanto cada um desses valores pode ser verdadeiro.

Seguindo o exemplo citado, sobre a *idade de João*, um usuário de posse da mesma informação pode escolher qualquer um dos valores próximos a *"50 anos"* e tomar como sendo certa a informação. Na expressão *"estamos quase certos que a idade de João seja 50 anos"*, não é possível determinar qual a certeza sobre esta informação. Convertendo esta informação em um valor numérico, há a possibilidade de definir um pouco mais esta ideia, como *"estamos 90% certos que a idade de João seja 50 anos"*. Se analisarmos os demais valores com possibilidade de serem verdadeiros, é admissível a atribuição subjetiva de uma porcentagem referente a cada *idade de João*. Formando assim um conjunto de valores com sua porcentagem de certeza, como:

$$Idade = \{(40{,}80\%)\,,(50{,}90\%)\,,(52{,}89\%)\,,(60{,}80\%)\}$$

2.1.3. Vagueza

Uma informação vaga caracteriza-se por ser indefinida, possuir uma falta de clareza, (Dicio 2016). Geralmente uma informação vaga é apresentada por um termo que se relaciona diretamente com o conjunto da informação. Em banco de dados pode-se definir informações incompletas caracterizadas por termos indefinidos. Os fragmentos de informação vaga podem se apresentar com termos como: *"forte"*, *"gordo"*,

"magro", dentre outros; além de intensificadores que modificam o sentido de vagueza dos termos como: *"muito forte"*, *"pouco magro"*, entre outros. Estes termos vagos podem ser relacionados com conectivos lógicos como: *"e"* e *"ou"*, além da negação, os quais são utilizados para definir termos linguísticos mais complexos, resultando em: *"não forte"* ou ainda *"não gordo e não magro"*. A forma desta informação é considerada válida como descrição, mas se mostra de difícil compreensão de forma isolada.

Como exemplo, consideramos a expressão: *"João é alto"*. O termo *"alto"* não nos diz precisamente a altura que *João* possui, torna-se variável dependendo do contexto da informação. Neste caso, por exemplo, o padrão de altura de uma população. De outro modo, na expressão *"João está alto"*, a variação torna o significado da frase diferente da anterior, passando agora a ideia da *"posição"* em que *João* está. Desta maneira o termo *"alto"* tem interpretações distintas se se refere a estatura ou posição.

O termo classificado como *"Dependência Contextual"* determina que a informação exija outros dados, para que somente assim, quando suas partes estiverem alinhadas e completas, possa existir coerência e permita ser interpretada corretamente. Este tipo de informação carece de elementos complementares que façam com que o usuário tenha o entendimento completo de seu significado e assim atribuir o julgamento necessário para seu uso. O uso de termos como *"alto"*, aplicado no exemplo anterior *"João é alto"*, juntamente com o termo *"tamanho"*, determina a altura que *João* possui, seu tamanho corporal. Do mesmo modo, se relacionado ao termo *"posição"*, como no segundo exemplo *"João está alto"* seu sentido se altera, indicando a localização de *João*.

Um outro aspecto onde a vagueza se apresenta é no tratamento de valores nulos. Em bancos de dados tradicionais, o valor nulo (do inglês, *Null*) em um atributo indica a ausência do valor. Sendo assim, o valor Null pode ser considerado um tipo de vagueza, uma vez que indica um valor desconhecido, indefinido ou ainda, que realmente não exista um valor a ser atribuído. Como exemplo considera-se a relação $R_i = \{Nome,\ Idade,\ CNH\}$ que relaciona o nome de uma pessoa com sua

idade e carteira de habilitação. Podemos efetuar o registro da seguinte forma:

$$R_1 = \{\text{“João”, 30, 49176981618}\},$$
$$R_2 = \{\text{“Maria”, 15, } NULL\},$$
$$R_3 = \{\text{“José”, 19, } NULL\},$$
$$R_4 = \{\text{“Pedro”, } NULL, \text{ } NULL\},$$

Analisando os exemplos, R_1 segue o padrão definido corretamente. Em R_2, considerando a *“idade de Maria”*, sendo está menor de idade, o atributo *Null* é empregado por não existir um valor para seu registro. Em R_3, é de conhecimento que *José* possui a idade legal para possuir uma *CNH*, entretanto, desconhecemos se exista. Por fim, em R_4, não é possível determinar a existência ou não de uma *CNH*, pois desconhecemos a idade de *Pedro*, tornando impossível definir qual o tipo de informação cabe ao valor deste *CNH*.

Em bancos de dados tradicionais, a atribuição do valor nulo é utilizada, geralmente, quando o valor é desconhecido no momento do registro do dado ou ainda quando não se enquadra no conjunto de valores permitidos ao domínio no qual deve ser registrado. Por outro lado, a dependência contextual ocorre ao analisar-se o resultado da execução de consultas, onde a sequência de dados deve ser avaliada como um todo, para então serem compreendidos.

2.1.4. Ambiguidade

O termo ambiguidade é definido como uma qualidade daquilo que possui ou pode possuir diferentes sentidos, natureza do que é ambíguo. Se caracteriza por oferecer ao usuário um conjunto de possibilidades distintas para o mesmo valor, (Dicio 2016). Nesta definição, diferente da imprecisão e da incerteza, tem-se o valor real sobre um dado, mas interpretações distintas podem ser obtidas desta informação, sem que necessariamente estas interpretações estejam incorretas.

Em bancos de dados a ocorrência da ambiguidade se apresenta com a inserção de dados imperfeitos juntamente com dados perfeitos. Cita-se como exemplo, a informação *“João está velho”*. Para *Pedro* está informação induz que *“João tem mais de 60 anos”*. Todavia, para *Maria* está mesma informação induz que *“João*

tem mais de 70 anos". Pode-se determinar qual destas afirmações é a verdadeira? A informação deve ser clara, de modo que possa ser compreendida perfeitamente sem outras interpretações. Uma informação ambígua deve ser avaliada, pois algum detalhe, mesmo que pequeno, é capaz de influenciar ou esclarecer a diferença entre estes registros.

2.1.5. Inconsistência

O termo inconsistência indica a falta de lógica, de firmeza ou por provocar uma inconsistência de ideias, (Dicio 2016). Em banco de dados pode ser entendida como uma falha na sincronização dos dados, de modo que uma informação se apresente diferente, como resultado de uma consulta feita em locais distintos, em um mesmo banco de dados ou em bancos de dados diferentes.

Na primeiro caso, sem o controle devido, é possível que dados diferentes, referente a uma mesma informação, sejam inseridos em momentos distintos. Neste caso, esta dupla inserção de informações pode ocasionar problemas futuros, pois o usuário deixa de saber qual informação é verdadeira. Em um segundo caso, uma informação pode estar disponível em lugares distintos, podendo ser acessada e atualizada em um local e não nos demais. Como consequência o usuário não sabe em qual local a informação está correta.

Como exemplo, consideremos que *João* possua um registro de seu salário em uma empresa matriz, do mesmo modo que em sua filial. Este registro pode ser atualizado em algum momento pela empresa matriz e não repassado para sua filial. Em determinado momento, ao se consultar o *"salário de João"*, serão apresentados valores diferentes, sem que possamos determinar qual o valor real do salário de *João*. Em bancos de dados a inconsistência se agrava em aplicações que permitem armazenamento de forma distribuída, sendo necessária a sincronização dos dados contidos vindos de lugares distintos.

O conceito de inconsistência não está diretamente relacionado com o tipo de dado, como nos outros casos. Este está mais vinculado à modelagem de uma aplicação, às tecnologias de armazenamento, políticas de alterações e atualizações de registros, dentre outros fatores.

2.1.6. Considerações Finais

Abordamos até o momento os cinco tipos de informações imperfeitas. As informações imperfeitas necessitam de tratamento diferenciado do restante das informações, para que possam se tornar úteis.

2.2. Conjuntos Nebulosos

Os conjuntos nebulosos são aplicados em diversas abordagens relacionadas a manipulação da informação imperfeita e possibilitam a inclusão de valores que direcionam o sentido da informação, através do seu significado aproximado. Ao atribuir um coeficiente nebuloso a um dado imperfeito, pode-se comparar este dado com um universo de valores possíveis relacionados.

De acordo com Zadeh *et al.* (1965), a teoria de conjuntos nebulosos é uma extensão da teoria de conjuntos. Desta forma, integra conceitos de operações matemáticas básicas da teoria original, como *Intersecção*, *União*, *Diferença*, *Produto Cartesiano*, dentre outros. Inclui ainda as operações específicas utilizadas sobre relações, como *Seleção*, *Junção* e *Projeção*. Considerando que o foco da dissertação é a aplicação da teoria de conjuntos nebulosos à bancos de dados, serão expostos apenas os conceitos necessários para a compreensão da proposta.

A teoria de conjuntos tradicional, baseada na lógica booleana, considera que a relação de pertinência de um elemento a um conjunto só pode ser verdadeira ou falsa, ou seja, é aceitável somente que um elemento pertença ou não a um conjunto. A teoria dos conjuntos nebulosos de Zadeh *et al.* (1965), permite estender a noção de pertinência de um elemento a um conjunto a um valor entre *0* e *1* que representa o *grau de pertinência* do elemento ao conjunto. Este grau de pertinência introduz um novo tipo de relação mais flexível entre elementos e seus conjuntos. Desta forma, pode ser representado o fato de que um elemento satisfaz em certo grau a propriedade ou predicado que define o conjunto. Aplicando o conceito aos bancos de dados, um registro incompleto que seria descartado por não se encaixar nas definições tradicionais da teoria de conjuntos, poderia agora ser armazenado, com sua referência a um conjunto nebuloso ou a um valor aproximado.

Em Zadeh *et al.* (1965) são definidos os conceitos básicos dos conjuntos nebulosos que são apresentados a seguir.

Seja U um conjunto de elementos de um universo discreto ou contínuo, o qual possui um elemento qualquer denominado u. Um ***conjunto nebuloso*** F em U é caracterizado por uma ***função de pertinência*** μ_F representada por $\mu_F(u)$, que associa cada elemento de U com valores no intervalo $[0, 1]$, ou seja, $\mu_F : U \rightarrow [0, 1]$. Para exemplificar, em um conjunto de idades U, sendo $U : [1 \ldots 100]$, obtemos um subconjunto de idades I e seu respectivo conjunto nebuloso F a seguir:

$$I_{idade} = \{18,\ 19,\ 22,\ 25,\ 28\} \text{ e } F_{idade} = \{0.7,\ 0.9,\ 0.95,\ 0.8,\ 0.65\}$$

Um *conjunto nebuloso* F pode também ser representado como um conjunto de pares ordenados do universo U. O primeiro elemento do par é um elemento u de U e o segundo elemento do par seu grau de pertinência ao conjunto, ou seja, $F = \{(u, \mu_F(u)) \,|u \in U\}$. Deste modo, a representação do conjunto nebuloso para cada um dos elementos de U, é definida por:

$$F = \{(u_1, \mu(u_1)), (u_2, \mu(u_2)), (u_3, \mu(u_3)), \ldots, (u_n, \mu(u_n))\}$$

A expressão representa um conjunto limitado de elementos de U. Um conjunto ilimitado de elementos é definido pela expressão:

$$F = \int_{u \in U} u, \mu_F(u)$$

Assim, utilizando o exemplo apresentado anteriormente, conclui-se que:

$$F_{idade} = \{(18, 0.7), (19, 0.9), (22, 0.95), (25, 0.8), (28, 0.65)\}$$

O ***conjunto suporte*** de um *conjunto nebuloso* F é definido como um subconjunto do universo U, tal que: $Supp(F) = \{u | u \in U e u_n, \mu(u) > 0\}$, onde se considera unicamente os elementos de U com grau de pertinência com valor maior que *0*. Um ***ponto de inflexão*** sobre um conjunto nebuloso é o elemento u de U cujo valor de pertinência seja: $\mu F = 0,5$.

Um $\alpha - cut$ atua como um limiar de nível de elementos que são considerados aceitos como válidos, desde que iguais ou acima do valor estipulado pelo $\alpha - cut$, (Zimmermann 1996). Um $\alpha - cut$ do conjunto nebuloso F é composto pelos elementos

de U com valor de pertinência μ_F maior ou igual ao valor do α, onde: $F\alpha = \{u|\mu_F(u) \geq \alpha\}$ para $0 \leq \alpha \leq 1$. Um $\alpha - cut$ pode ser definido a partir de qualquer valor acima de *0* até *1*. Neste contexto, quanto mais perto do valor *1* um $\alpha - cut$ é definido, um maior valor de verdade é requerido do elemento.

A proposta da teoria dos conjuntos nebulosos de Zadeh *et al.* (1965) permitiu a definição formal de novas formas de representação e manipulação da informação imperfeita.

Para exemplificar, analisemos a seguinte afirmação, com informação imperfeita imprecisa: *"Maria tem entre 28 a 31 anos de idade"*. O conjunto de valores possíveis do universo U ao qual se refere *"a idade de Maria"* pode ser definido como $U = \{28, 29, 30, 31\}$. O conjunto de valores possíveis da idade de uma pessoa se classifica como um conjunto impreciso de valores de idade.

Adicionalmente a esse conjunto, pode-se adicionar uma informação imperfeita do tipo incerteza. Através da função de pertinência $\mu F(u)$ pode-se atribuir um grau de pertinência a cada elemento do conjunto. Assim, obtém-se $\mu F : U \rightarrow (0.7, 0.9, 0.95, 0.6)$, com os valores ou coeficientes de certeza respectivos. Este conjunto nebuloso pode ser considerado como *"valores verdade da idade de Maria"*, (Zadeh *et al.* 1965).

Pode-se então reescrever o *conjunto nebuloso* F, considerando o conjunto de pares com cada elemento do conjunto e a representação do seu grau de pertinência, (Ma & Yan 2016). Neste contexto, teremos então o retorno do conjunto nebuloso, com a seguinte expressão de valores aceitos como *"idades de Maria"*, sendo:

$$F_{IdadeMaria} = \{(28, 0.7), (29, 0.9), (30, 0.95), (31, 0.6)\}$$

2.2.1. Representação da Informação Imperfeita

Uma informação sendo perfeita ou imperfeita, deve ser compreensível, ou seja, deve possibilitar que um usuário possa interpretá-la com base em seus conhecimentos. A teoria dos conjuntos nebulosos apresenta meios matemáticos que aproximam uma informação imperfeita a um significado real. Deste modo, ainda que não seja totalmente compreendida, uma informação possuindo um significado aproximado, pode ser relacionada a um conjunto de respostas possíveis.

Existem diversas formas de representar este tipo de informação, tanto valores em forma numérica quanto textuais. Os conceitos sobre as informações imperfeitas se ampliam por meio de novas definições. Em Üstünkaya, Yazici & George (2007) apresenta as seguintes definições:

i. ***Valores nulos:*** Como definido anteriormente, os valores nulos são considerados um tipo de vagueza. Quando relacionados a um conjunto de elementos, representam a ausência de um elemento ou falta de informação sobre este. Classificam-se como *"indefinidos"*, *"desconhecidos"*, *"inexistentes"* ou *"sem informações"*;

ii. ***Intervalo de elementos:*** A representação de uma informação através de um intervalo de valores, tem objetivo delimitar os conjuntos de elementos possíveis. Para exemplificar, a representação da idade de uma pessoa pode ser definida através de um intervalo de valores, como: $[20 - 30]$;

iii. ***Conjunto de elementos:*** Diferente do tipo anterior, o conjunto de elementos define quais os elementos possíveis. Assim, seguindo o exemplo anterior, pode ser definido como: {20, 21, 25, 30};

iv. ***Valores relacionais ou de composição:*** Os valores relacionais ou de composição são definidos pela comparação entre dois ou mais conjuntos completos, podendo ser compostos também por subconjuntos. Este tipo de informação será melhor explicado adiante neste trabalho, através do conceito das ***Relações Nebulosas***;

v. ***Variáveis linguísticas:*** Definidas por termos linguísticos que são atribuídas pelo seu significado semântico. Os termos citados são utilizados para representar um dado exato desconhecido no momento. Os termos linguísticos que compõem uma variável linguística são denominados termos nebulosos. Termos nebulosos podem ser do tipo simples, como *jovem* ou *velho*, do tipo composto quando restringidas por termos do tipo *muito*, *pouco*, *mais ou menos*, ou ainda, do tipo composição quando composto de termos unidos *e*, alternados *ou* ou de negação *não*, (Chen & Jong 1997).

   Formalmente uma variável linguística é definida como uma quíntupla:

$$(V, T(V), U, G, S)$$

Sendo V a variável linguística propriamente, $T(V)$ representa um conjunto de termos de V, que correspondem a rótulos R dos valores de V. U representa o universo para a variável base u, que define os valores numéricos associados à V. G é a gramatica para gerar os termos linguísticos e S é a regra semântica, que busca associar o rótulo R contido em $T(V)$, com o significado $S(R)$.

Como exemplo da representação do valor da variável linguística *idade*, ou seja, V = "idade", definimos um conjunto U de valores enumerados *0* a *120*. Por intermédio de regras semânticas S associadas com a gramática G, temos a separação em faixas de valores de idades. Em outras palavras *"valores menores que 20"*, *"valores de 15 a 65"* e *"valores maiores que 60"*. Os rótulos R ou termos linguísticos são então utilizados como forma de representar estas faixas de valores, como *"jovem"*, *"meia-idade"* e *"idoso"*. O conjunto $T(V)$ representa a associação destes termos linguísticos à variável linguística, definidos assim como T(V) = {*jovem, meia-idade, idoso*}.

Analisando então a informação *"65 anos"*, onde u=65, pode-se associar este elemento tanto ao rótulo *"meia-idade"* quanto ao rótulo *"velho"*. Uma função de pertinência expressa quais elementos u estão associados aos rótulos R. O valor do rótulo R, do conjunto $S(R)$, relacionado por$T(V)$, é determinado como uma restrição da variável da base u, ou seja $R(u)$. Definimos então que $S(R)$ é o conjunto nebuloso dos elementos do universo de U cuja função de pertinência $R(u)$ expressa a semântica em relação ao seu rótulo R. A Figura 2.1, adaptada de Pedrycz & Gomide (1998), representa o exemplo dado anteriormente com a conversão de valores para a variável linguística *"idade"*. A área denominada *"definição simbólica"* representa a classificação textual dos termos linguísticos. A área denominada *"representação numérica"* representa o gráfico dos valores de idade, partindo de *0* na horizontal e a associação definida pela função de pertinência definida entre *0* a *1* na vertical. Entre estas áreas situa-se a *"regra semântica"* utilizada para definir a associação dos termos da definição simbólica com a sua representação numérica.

A relação de valores com seu grau de pertinência se torna simplificada, por assim dizer, quando considera-se um conjunto de elementos no qual é possível reali-

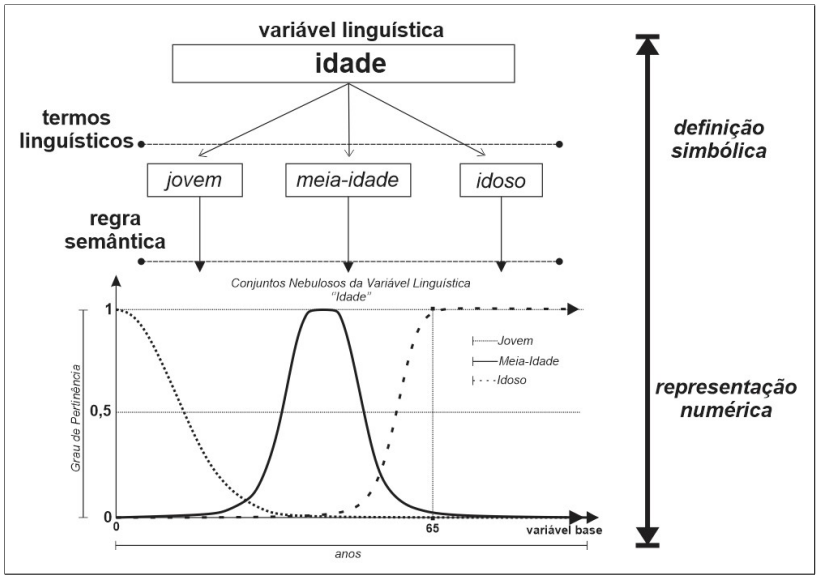


Figura 2.1. Definição da variável linguística *"idade"*.

Fonte: Adaptado de Pedrycz & Gomide (1998).

zar a enumeração subjetiva. A representação da função de pertinência da variável linguística *"idade"* e seus termos linguísticos, utiliza a escala de valores possíveis e sua representação no conjunto. Pode-se observar um exemplo destas definições por meio da Figura 2.2 destacada da Figura 2.1.

Observando a linha de valores do termo *"idoso"*, nota-se a curva de valores possíveis ao associar este termo linguístico a uma função de pertinência. Conforme Pedrycz & Gomide (1998), pode-se definir as variáveis linguísticas por meio de conjuntos nebulosos. Sendo possível realizar a conversão dos significados linguísticos destes termos em sua aproximação numérica e computável.

A representação da faixa de elementos relacionados a um termo é possível desde que se identifique quais os elementos que podem ser associados ao termo. Entretanto, na análise de variáveis linguísticas que não possuem limites bem definidos, não é possível definir com facilidade um conjunto base de elementos. A ausência dos limites dificulta a associação de um termo linguístico com um conjunto de valores possíveis.

Neste contexto, Zadeh *et al.* (1965) e Ma & Yan (2016) definem duas classes de variáveis linguísticas. O primeiro consiste na representação de elementos limi-

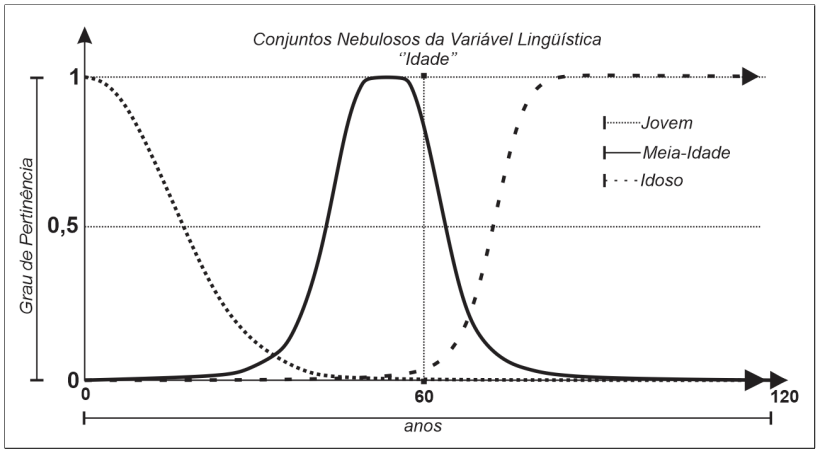


Figura 2.2. Variável linguística *"idade"*.

Fonte: Adaptado de Pedrycz & Gomide (1998).

tados ou discretos, onde temos um intervalo com um valor mínimo e máximo de elementos. O segundo é composto por um conjunto ilimitado ou contínuo de elementos. Este não é limitado a um intervalo, assim considera-se impossível serem computados de maneira formal.

É possível subdividir as duas classes de variáveis linguísticas referidas anteriormente em outros dois tipos. Em um é possível listar um número contável de elementos e em outra é composta por uma lista incontável de elementos, (Ma & Yan 2016). Em função disto, as variáveis linguísticas podem ser classificadas em quatro tipos de conjuntos nebulosos, sendo: (i) aqueles compostos por um conjunto finito de elementos; (ii) aqueles compostos por um conjunto limitado, mas com uma quantidade não enumerável de elementos; (iii) aqueles compostos por um conjunto ilimitado, mas com uma quantidade numerável de elementos possíveis e por fim, (iv) aqueles compostos por um conjunto ilimitado de elementos, Ma & Yan (2016). Como exemplo desta classificação, vamos supor as seguintes situações de representação:

- **Primeiro caso:** Na expressão *"A encomenda chegará em qualquer horário do dia 23"*, há uma informação imprecisa. Porém, é possível relacionar o conjunto entre horas, minutos e segundos fixos durante as *24 horas do dia*. Classifica-se a informação como um conjunto nebuloso limitado de elementos;
- **Segundo caso:** Na expressão *"O café está quente"*, temos um conjunto de temperaturas possíveis. Levando em consideração a temperatura de ebulição

da água, entende-se que a temperatura está abaixo de 100° *Celsius*. Entretanto, dentro deste limite existe um incontável número de temperaturas possíveis. A informação no caso é classificada como um conjunto nebuloso limitado, mas contendo uma quantidade não enumerável de elementos;

- **Terceiro caso:** Na expressão *"Dias futuros"*, existe uma quantidade ilimitada de dias, mas que podemos enumerá-los, sem necessariamente saber o número máximo de dias a que está se referindo. Classifica-se a informação como um conjunto nebuloso ilimitado de elementos, mas com uma quantidade numerável de elementos possíveis;
- **Quarto caso:** Para o dado *"tamanho"*, de forma isolada, há uma infinidade de valores possíveis com uma infinidade de possibilidades. Classificamos então como um conjunto nebuloso ilimitado de elementos.

As variações nos conjuntos nebulosos podem ser representadas de diferentes formas. Os modelos de representação das funções de pertinência mais utilizados são aquelas baseadas nas formas *"trapezoidal"* e *"triangular"*, Ma & Yan (2016). Estas funções representam os intervalos de valores nos conjuntos do universo descrito e o valor da referência de cada elemento, respectivamente.

Em Ma & Yan (2016) e Pedrycz & Gomide (1998), definem-se estas funções, sendo:

a) ***Função trapezoidal:*** Caracteriza-se pelo conjunto $\{a, b, c, d\}$. Os elementos deste conjunto representados por (a, d), são denominados como os elementos limites, representando o valor mínimo e máximo da função de pertinência deste conjunto. Os elementos representados por (b, c), são denominados o intervalo da função, ou seja, formam o conjunto de valores dos elementos com grau de pertinência próximo ao valor 1. Formalmente esta função é definida pelos parâmetros a, b, c, d, onde $a \leq b \leq c \leq d$, para cada valor de x do conjunto nebuloso C. Assim temos: $C(x; a, b, c, d) = max\left\{min\left[\frac{(x-a)}{(b-a)}, 1, \frac{(d-x)}{(d-c)}\right], 0\right\}$;

b) ***Função triangular:*** É identificada pelo conjunto $\{a, b, c\}$. Neste conjunto os elementos representados por (a, c) são considerados os elementos mínimo e máximo, respectivamente. O elemento b representa o valor do elemento

com grau de pertinência de valor próximo a 1. Formalmente é definida pelos parâmetros a, b, c, onde $a \leq b \leq c$, para cada valor x do conjunto nebuloso C. Portanto temos: $C(x; a, b, c, d) = max\left\{min\left[\frac{(x-a)}{(b-a)}, 1, \frac{(c-x)}{(c-b)}\right], 0\right\}$. De certa forma, a função triangular é considerada como um caso especial da função trapezoidal, por exemplo, se o valor atribuído ao elemento b é igual ao elemento c, ou seja, $b = c$, na função trapezoidal, então esta pode ser transformada em uma função triangular. É possível definir que a função triangular seja de fato uma função trapezoidal com a expressão do conjunto $\{a, b, b, d\}$.

A Figura 2.3 representa resultados baseados nas funções de pertinência trapezoidal e triangular. Observamos que a função trapezoidal é composta por intervalo de elementos dentro do universo apresentado, diferente da forma do modelo triangular que apresenta a aproximação do valor determinado.

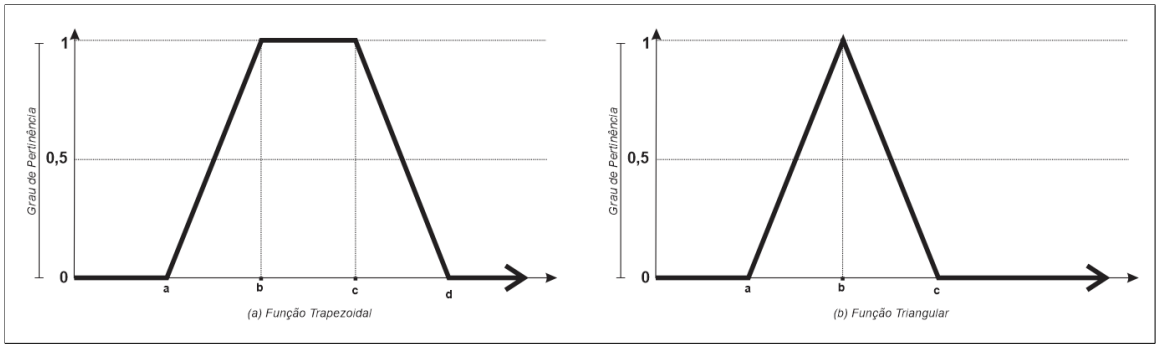


Figura 2.3. Representação gráfica das *funções de pertinência*.

Fonte: Elaborado pelo autor.

O conjunto de elementos definidos pelas funções mencionadas podem ser muito abrangentes, uma vez que podem existir uma grande quantidade de elementos entre os elementos que delimitam o conjunto. Um grau de pertinência pode ser obtido para cada elemento pertencente ao intervalo definido, mas nem sempre se faz necessário considerar todos estes elementos. Os elementos que possuam um grau de pertinência muito baixo em relação ao conjunto podem ser desconsiderados, em alguns casos, dependendo da interpretação que será feita. Deste modo, pensando em determinar de forma mais precisa quais os elementos são válidos, um delimitador pode ser utilizado, atuando com base na análise dos graus de pertinência obtidos.

Os valores de pertinência obtidos podem ser validados por meio de limitadores

(do inglês, *thresholds*), termo utilizado para estabelecer um limiar de aceitação, ou seja, um $\alpha - cut$. Como é possível ver na Figura 2.4, o uso do limiar estipulado em d para a função triangular indica os valores de aceitação. Assim, o conjunto de valores existentes entre os pontos x e y são os considerados como aceitáveis.

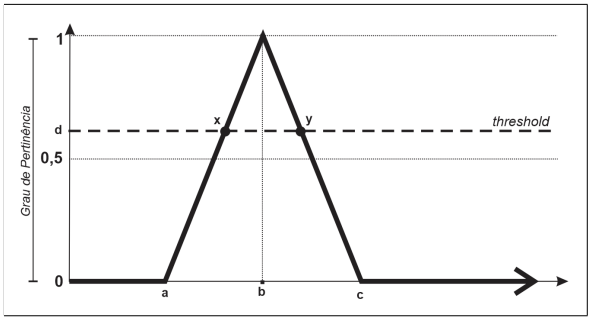


Figura 2.4. Função triangular limitada por um *limiar*.

Fonte: Elaborado pelo autor.

2.2.2. Função de Similaridade

Uma função de similaridade pode ser utilizada como ferramenta de mensuração da relação entre elementos de um conjunto ou entre conjuntos. Deste modo, pode-se implementar um algoritmo para definir um coeficiente que quantifica o valor de semelhança de um elemento em relação aos demais.

	Temperatura	*Fervendo*	*Quente*	*Morno*	*Normal*	*Frio*	*Gelado*
	Fervendo	**1.0**	0.8	0.6	0.4	0.2	0.0
	Quente	0.8	**1.0**	0.8	0.6	0.4	0.2
R_s:	*Morno*	0.6	0.8	**1.0**	0.8	0.6	0.4
	Normal	0.4	0.2	0.8	**1.0**	0.8	0.6
	Frio	0.2	0.4	0.2	0.8	**1.0**	0.8
	Gelado	0.0	0.2	0.4	0.6	0.8	**1.0**

Figura 2.5. Matriz de similaridade simétrica da variável linguística *"temperatura"*.

Fonte: Adaptado de Zadeh (1971).

Para exemplificar, utilizaremos o exemplo da Figura 2.5, baseado nos conceitos apresentados em Zadeh (1971). A figura demonstra a matriz de similaridade

para a variável linguística *Temperatura*, definido pelos termos linguísticos associados ao conjunto, sendo:

$$Temperatura = \{\text{Fervendo, Quente, Morno, Normal, Frio, Gelado}\}$$

Nota-se a existência de uma relação de valores de semelhança entre os termos. Uma comparação lógica sequencial é apresentada pela distribuição dos elementos, de modo a definir o coeficiente de similaridade base da relação entre estes termos.

Deste modo, conforme o exemplo, ainda que um tipo de dado seja considerado imperfeito, existe a possibilidade de obter o seu valor real. A abordagem para este tipo de análise pode ser realizada de duas formas: (i) atribuindo diretamente o conjunto de dados equivalentes e posteriormente, definir o seu valor e (ii) realizando a conversão destes valores e obtendo o coeficiente de similaridade.

Como exemplo, para definir o valor de uma temperatura que não possui um valor exato, é possível: (i) manter a relação da temperatura através do conjunto de seus termos linguísticos, digamos {Normal e Quente}, ou ainda, (ii) utilizar a função de similaridade para determinar o seu valor de pertinência.

Considerando que exista uma simetria entre os termos de um conjunto C, sendo (x, y) um par qualquer de elementos do conjunto, a expressão $Sim(x, y) = Sim(y, x)$ deve ser verdade. Os valores de semelhança máxima, ou reflexividade, são definidos por $Sim(x, x) = 1$, da mesma forma valores opostos definidos como a máxima exclusão são representados por $Sim(x, x) = 0$, por fim, a transitividade dos elementos deve satisfazer a condição, onde o máximo y, para o mínimo entre $Sim(x, y)$ e $Sim(y, z)$ deve ser pelo menos igual a $Sim(x, z)$.

Assim, os conceitos que a propriedade de similaridade de elementos deve possuir pode ser expressa por:

$$\text{Para } \forall\ x \in C, Sim(x, x) = 1 \qquad (\textit{Reflexividade});$$

$$\text{Para } \forall\ x, y \in C, Sim(x, y) = Sim(y, x) \qquad (\textit{Similaridade});$$

$$\text{Para } \forall\ x, y, z \in C, Sim(x, z) \geq max_y\left(min\left(Sim\left(x, y\right), Sim\left(y, z\right)\right)\right) \qquad (\textit{Transitividade});$$

Em Stasiu (2007) é realizada uma reflexão sobre os diversos tipos de funções

de similaridade que podem ser utilizadas. Uma função de similaridade é expressa por: $Sim(x, y) \rightarrow [0, 1]$. Isto é, definimos x e y como sendo dois elementos quaisquer do conjunto de elementos relacionados do universo proposto U. Entende-se então que x é igual a y em certo grau dentro do intervalo de $[0, 1]$.

A função de similaridade abordada nesta pesquisa apresenta alguns dos conceitos fundamentais do tema. A definição desta função é importante para o entendimento de uma das formas de manipulação das informações imperfeitas.

2.2.3. Relação Nebulosa

O conceito de relação nebulosa (do inglês, *Fuzzy Relation*) consiste em mensurar a relação existente entre os elementos de um conjunto ou pela comparação de elementos entre conjuntos. Neste contexto define-se que uma relação nebulosa pode ser caracterizada como um subconjunto de $X \times Y$, onde cada um dos pares (x, y) são associados a um *grau de pertinência* no intervalo $[0, 1]$, caracterizando a *relação de força* entre x e y (Zadeh 1971).

De forma geral, Zimmermann (1996) define a relação nebulosa para o conjunto de elementos que compõe X e Y, onde $R \subseteq X \times Y$ sendo o conjunto universo, deste modo temos: $R = \{((x, y), _R (x, y)) \,\|\, (x, y) \subseteq X \times Y\}$, que podem ser denominados como a relação nebulosa de $X \times Y$.

Os conceitos sobre as relações nebulosas se assemelham aos conceitos da relação de similaridade de Zadeh (1971). De fato considera-se que a relação de similaridade seja um caso especial de relação nebulosa, (Zimmermann 1996). Todavia, os conceitos sobre as relações nebulosas são mais abrangentes que as definições sobre as relações entre elementos de um conjunto. Uma vez que são considerados todos os elementos pertencentes ao conjunto, independentemente do domínio ao qual pertencem.

2.2.4. Distribuição de Possibilidades

A formulação da teoria dos conjuntos nebulosos e suas definições, permitiu a representação dos dados imperfeitos. Todavia, somente esta representação não satisfaz a necessidade de um método de julgamento mais eficaz. O uso da *"distribuição*

de possibilidades", oferece suporte adicional na avaliação de um elemento frente ao conjunto nebuloso ao qual pertence.

A função de distribuição de possibilidades proposta por Zadeh (1999), apresenta uma nova abordagem para a definição da semântica da informação nebulosa e os valores representativos de seus elementos. De modo geral a proposta de distribuição de possibilidades busca a validação de cada elemento de um conjunto atribuindo ao elemento um grau de possibilidade. Isto determina se de fato um elemento pode ser aceito, aceito até certo grau ou não aceito, quando relacionado a um determinado conjunto. Alguns dos trabalhos existentes na literatura assemelham a teoria de distribuição de possibilidades com a teoria de probabilidades, visto que as funções atuam sobre funções de conjuntos. Considera-se então que a distribuição de possibilidades seja uma teoria diferente para uma mesma situação ou ainda que estas teorias sejam complementos uma da outra.

De forma simplista, entende-se que a teoria da distribuição de possibilidades visa retornar o grau de possibilidade para que uma incerteza possa ocorrer. Portanto, ao atribuirmos uma incerteza x qualquer, devemos realizar a mensuração por meio de duas medidas, sendo uma do ponto de vista da *possibilidade de ocorrência* $\Pi(x)$ e outra sobre a *necessidade de ocorrência* $N(x)$. Estas medidas caracterizam respectivamente a possibilidade de ocorrência com a impossibilidade de ocorrência da situação oposta, sendo assim, $N(x) = 1 - \Pi(x)$.

Neste contexto, na expressão proposta por Zadeh (1999) de X ser F podemos considerar X sendo uma variável que obtém valores do universo U, com um elemento genérico u, de modo que $X = u|u \in U$. F representa uma restrição à variável X, sendo $R(X) = F$, este sendo um subconjunto de valores obtidos pela função de pertinência μ_F para os elementos u do universo U, onde $\mu_F : U \rightarrow [0, 1]$. Podemos definir o subconjunto nebuloso F obtendo os graus de pertinência dos elementos do universo pela expressão:

$$F = \{(u_1, \mu_F(u_1)), (u_2, \mu_F(u_2)), (u_3, \mu_F(u_3)), \ldots, (u_n, \mu_F(u_n))\}$$

O grau de pertinência obtido pela distribuição de possibilidades para um elemento u em X, caracteriza o quanto este elemento é compatível com o conjunto.

Com base no conjunto nebuloso F, define-se a distribuição de possibilidade $\prod_x$, sendo então $\prod_x = F$. Consideram-se os valores da distribuição de possibilidades como:

$$\prod_x = \{(u_1, \pi_F(u_1)), (u_2, \pi_F(u_2)), (u_3, \pi_F(u_3)), \ldots, (u_n, \pi_F(u_n))\}$$

Com base na equivalência mantida por $\pi_x = \mu_F$, define-se que $\pi_x(u)$ seja a possibilidade que X tenha o valor de u (Zadeh 1999), (Ma & Yan 2007), (Ma & Yan 2016).

Por fim, de posse dos valores de distribuição de possibilidades $\prod_x$, pode-se também definir a medida de necessidade N_x, através da função $N_x = 1 - \prod_x$.

Para uma melhor elucidação dos conceitos apresentados, vamos supor que obtemos a informação das médias finais dos alunos de uma escola. Neste contexto tem-se: *"Pedro tem 4 pontos, Maria tem uma pontuação alta e João tem 10 pontos"*. Assim temos precisamente a informação da pontuação dos alunos *"Pedro"* e *"João"*. Todavia, existe a impossibilidade de definir com certeza a pontuação de *"Maria"*, de modo que podemos aplicar os conceitos da função de distribuição de possibilidades.

Neste contexto, considerando o conceito do termo *"pontuação"*, sendo este formado por um conjunto de valores do universo U, onde $U = [0, 10]$. O termo *"alta"* atuando como um limiar, temos um subconjunto de U definido por $S = \{8, 9, 10\}$ de notas possíveis. É possível atribuir a medida de distribuição de possibilidade para os valores como a possibilidade da *"pontuação"* $= X$ ser realmente a *"pontuação de Maria"*, formalmente definido por $\mu_{pontuacaoalta}(u) = 1 - S(u; 8, 9, 10)$, onde:

$$\begin{aligned}
&S(u; a, b, c) = 0 && \text{para } u \leq a;\\
&S(u; a, b, c) = 2\left(\tfrac{(u-a)}{(c-a)}\right)^2 && \text{para } a \leq u \leq b;\\
&S(u; a, b, c) = 1 - 2\left(\tfrac{(u-c)}{(c-a)}\right)^2 && \text{para } b \leq u \leq c;\\
&S(u; a, b, c) = 1 && \text{para } u \geq c.
\end{aligned}$$

Com isso obtém-se a distribuição de possibilidades pela função, definida para a variável X, representando os valores tomados pela *"alta pontuação de Maria"* como sendo: $\prod_X = \{(8, 0.6), (9, 0.95), (10, 0.65)\}$. Neste exemplo aplicamos os valores de forma subjetiva, contudo, serve para demonstrar a possibilidade que um elemento

tem de ser aceito como verdadeiro. Por fim, podemos determinar que a afirmação *"Maria tem 9 pontos"* seja mais possível do que se afirmarmos que *"Maria tem 10 pontos"* ou *"Maria tem 8 pontos"*.

2.2.5. Considerações Finais

Neste capítulo buscou-se apresentar os conceitos fundamentais sobre a teoria dos conjuntos nebulosos. Além de conceitos de teorias relacionadas, que complementam a teoria dos conjuntos nebulosos. Estas se mostraram úteis para a representação e manipulação da informação imperfeita.

Por meio dos conceitos abordados, é possível compreender o real significado de um dado imperfeito. Com base na análise do elemento em questão e outros elementos do seu conjunto, pode-se alcançar o provável significado deste elemento de forma satisfatória. Pode-se, por exemplo, obter a similaridade entre termos de um dado imperfeito, como também, determinar qual a possibilidade que um valor tem de representar o valor real de um dado imperfeito.

No capítulo seguinte apresentamos algumas abordagens existentes para representação e manipulação de dados imperfeitos, utilizados nos principais modelos de bancos de dados. Este estudo tem por objetivo demonstrar a forma de representação dos dados imperfeitos realizada por cada um destes modelos. Além de abordagens referentes a lógica de inserção destes dados imperfeitos, como se relacionam com as limitações da estrutura e as formas de retorno dos dados inseridos para os usuários.

CAPÍTULO 3

Informações Imperfeitas em Bancos de Dados: Uma Visão Geral

Em aplicações do mundo real, a ocorrência das informações imperfeitas se torna comum, pois nem sempre a informação captada está na forma ideal para ser armazenada. Para cada um dos diferentes tipos de estruturas, que armazenam e gerenciam dados de diferentes fontes, existem também métodos específicos que buscam transformar a informação imperfeita em uma que possa ser armazenada corretamente.

Neste capítulo busca-se apresentar uma visão geral da forma em que a informação imperfeita é armazenada, manipulada e recuperada em diferentes estruturas de bancos de dados.

3.1. Conceitos de Banco de Dados

Um banco de dados é definido como uma coleção de dados relacionados, que possuam a característica de representar algum aspecto do mundo real, (Elmasri, Navathe, Pinheiro *et al.* 2005). Estes dados são considerados fatos conhecidos, podem ser registrados e devem possuir um significado implícito. Com seu registro passam a fazer parte de uma coleção, sendo logicamente coerente e com algum significado inerente.

O desenvolvimento ou modelagem de um banco de dados ocorre através de diferentes estágios, como levantamento de requisitos e análise da estrutura de dados que compõe o modelo.

A evolução das necessidades das aplicações levou ao surgimento de diferentes modelos lógicos de bancos de dados que passaram a representar melhor às necessidades dos usuários, oferecendo mais controle no tratamento destes dados. Dentre os modelos de bancos de dados mais importantes que apareceram neste período temos o modelo relacional e o modelo orientado a objetos.

3.2. Modelos de Banco de Dados Relacionais Nebulosos

A aplicação dos conjuntos nebulosos em banco de dados relacionais vem sendo estudada desde sua concepção inicial deste modelo de dados. Desde que passou a ser considerado o armazenamento de dados imperfeitos, os métodos e ferramentas que buscam oferecer suporte a estes dados vêm recebendo aprimoramentos. Diante disso foram aplicados conceitos pertencentes aos conjuntos nebulosos, em visões distintas para cada aspecto possível deste modelo, de modo a permitir a integração de dados nebulosos no modelo.

3.2.1. Bancos de Dados Relacionais

O modelo de bancos de dados relacionais foi proposto inicialmente por *Ted Codd*, da *IBM Research*, em um artigo publicado em 1970. De acordo com Elmasri, Navathe, Pinheiro *et al.* (2005), o modelo proposto por *Codd* atraiu atenção imediata por sua simplicidade e base matemática. Baseando-se no conceito de relação matemática, está representado por um arquivo plano de registros, semelhante a uma tabela de valores. Sua denominação de arquivo plano se deve a sua estrutura linear.

O modelo relacional é assim denominado por sua particularidade de representar uma coleção de relações. Uma relação é representada como uma tabela, possuindo linhas e colunas. Os valores são interpretados como eventos ocorridos e que descrevem o mundo real. Para facilitar a sua interpretação, são determinados nomes para estas relações e suas colunas, bem como a que tipo de dado essa coluna pode receber (Elmasri, Navathe, Pinheiro *et al.* 2005).

A terminologia formal definida para o modelo relacional é composta principalmente pelos termos *"tabela"*, *"tupla"*, *"coluna"*, *"atributo"* e *"domínio"*. Neste contexto podemos definir o termo *"tabela"* como uma relação de identificação geral

para a coleção de registros, as linhas de registros existentes como *"tuplas"*. O cabeçalho que identifica um agrupamento do dado ou uma referência para o conjunto de dados do mesmo grupo é denominado como *"coluna"*. Os dados registrados nas seções das colunas são denominados como *"atributo"* e o seu tipo de dado, que descreve os tipos de atributos que esta coluna podem receber, é denominado *"domínio"*, (Elmasri, Navathe, Pinheiro *et al.* 2005).

Cada tupla deve representar uma entidade única ou objeto. Neste contexto, o conjunto de tuplas de uma relação deve seguir a definição da teoria de conjuntos, onde seus elementos devem ser diferentes, (Elmasri, Navathe, Pinheiro *et al.* 2005).

Para realizar o acesso ou manipulação dos dados armazenados nos bancos de dados relacionais é utilizado por padrão instruções próprias, sendo estas de consulta ou manipulação. A linguagem de consulta *SQL* possui abrangência no que diz respeito a banco de dados relacionais, considerada como responsável, em grande parte, pelo crescimento do modelo relacional, a Linguagem de Consulta Estruturada (do inglês, *Structured Query Language (SQL)*) teve uma evolução ao longo dos anos.

Uma instrução construída nos moldes da linguagem *SQL* segue não somente as operações voltadas para o próprio modelo de relações, como também os princípios da teoria dos conjuntos, (Elmasri, Navathe, Pinheiro *et al.* 2005). Neste sentido foi introduzida a álgebra relacional com as operações especificas ao modelo de relações, definidas como: *"Seleção"*, *"Projeção"* e *"Junção"*, além daquelas pertencentes a teoria dos conjuntos como: *"União"*, *"Intersecção"*, *"Diferença"* e *"Produto Cartesiano"*. Estes princípios se relacionam e se complementam no desenvolvimento das instruções de consultas *SQL*, onde estas são utilizadas como parâmetros, norteando a sua execução.

3.2.2. Incorporação da Informação Imperfeita no Modelo Relacional

A integração da informação imperfeita e os dados nebulosos no modelo de dados relacional é definida como Modelo de Banco de Dados Relacional Nebuloso (do inglês, *Fuzzy Relational Database Model (FRDBM)*). Diversos trabalhos, sob os mais diferentes aspectos, foram realizados a fim de aprimorar o modelo. Em Ma & Yan (2007, 2010) são apresentados alguns destes trabalhos, como forma de revisão

do estado da arte.

3.2.3. Representação da Informação Nebulosa no Modelo Relacional

Os dados presentes nos bancos de dados relacionais, por definição devem seguir as limitações impostas aos domínios em que estes dados estão inseridos e suas relações. A aplicação dos conceitos da informação imperfeita e os conjuntos nebulosos busca superar algumas destas limitações, permitindo que novos tipos de dados sejam inseridos.

Em Ma (2007), abordagens distintas são apresentadas para a representação da informação imperfeita, sendo: (i) inserção de dados nebulosos como atributos dos bancos de dados e (ii) para cada tupla é atribuído um valor de pertinência, representando o conjunto destes domínios. O valor atribuído à tupla representa a análise de todos os dados existentes nos domínios desta tupla. Esta análise determina o coeficiente que caracteriza o grau de certeza da tupla, definido entre 0 e 1.

Existem inúmeros métodos de tratamento e conversão dos dados nebulosos, contudo, os métodos mais significativos foram aqueles baseados na *relação nebulosa*, por *similaridade*, *distribuição de possibilidades*, ou em alguns casos, uma combinação de um ou mais métodos, possibilitando assim alcançar novos paradigmas. Dentre estes, destacamos os trabalhos de Raju & Majumdar (1988), Buckles & Petry (1982) e Prade & Testemale (1984).

Os trabalhos realizados visam permitir a inserção de dados nebulosos nas relações das entidades e nos atributos dos domínios. Isto possibilita atribuir um significado a mais aos dados imperfeitos, além de uma maior abrangência de conteúdo.

A Figura 3.1 integra os principais resultados destas propostas, se assemelhando a um modelo ideal de representação da informação imperfeita e dos dados nebulosos. Como pode ser observado, diversos tipos de dados foram inseridos, sem se restringir à capacidade do domínio em suportá-los. O modelo hipotético representa a estrutura perfeita para armazenamento de dados no modelo relacional, pois comporta uma grande variedade de tipos de dados. Todavia, se torna complexa quando consideramos a manipulação de dados de diferentes naturezas. Vale ressaltar que os valores $\{3.000, 4.000, 5.000\}$, $\{8.000 - 10.000\}$, $1.000 - 3.000$, apresentados na Figura

3.1, tem significados iguais, somente foram representados de forma diferente para abranger as diferentes abordagens.

	Nome	Departamento	Experiência	Salário	Imposto	μ
1	João	Mecânica	10	Baixo	10%	0.70
2	Maria	Elétrica	15-20	Alto	{Médio/1, Baixo/0.6}	0.90
3	{Pedro}	{Financeiro}	{Pouca}	{Baixo, Alto}	{Baixo}	0.65
4	Juca	Manutenção	Aprox. Alta	Em torno de 1.500	*Desconhecido*	0.10
5	Beto	{Suporte}	Alta	{3.000, 4.000, 5.000}	$\{0.5/10, 1/15, 1/5\}_p$	0.20
6	Julia	Vendas	Média	{8.000-10.000}	1.000-3.000	0.85

Figura 3.1. Modelo genérico idealizado com conceitos de armazenamento da informação nebulosa.

Fonte: Elaborado pelo autor.

Separando as abordagens relacionadas quanto à representação de dados nebulosos como atributos, temos primeiramente a inserção de dados com múltiplos valores de Buckles & Petry (1982). Esta proposta consiste em registrar todos os valores da tupla como conjuntos, mesmo aqueles que possuem um único valor. Sua abordagem se baseia no modelo de similaridade dos elementos.

O registro de um dado de múltiplos valores pode ser feito através de conjuntos escalares de números, como $\{20, 25, 30\}$ ou ainda em conjuntos escalares de grandezas, como $\{fraco, mdio, bom, excelente\}$. Estes conjuntos podem ser compostos de um único elemento ou de uma sequência, sendo finitos ou infinitos. Dessa forma, os valores dos atributos podem fazer uma aproximação do seu valor real, quando não se é possível definir um único valor com a informação exata.

Para exemplificar, a Figura 3.1 representa em sua terceira tupla, o modelo de Buckles & Petry (1982), com exceção da coluna μ que definiremos posteriormente. Todos os atributos desta tupla passam a serem definidos como conjuntos de valores do domínio, mesmo aqueles atributos considerados como bem definidos ou precisos. Os atributos que teriam o valor desconhecido no modelo clássico, passam a ter um significado aproximado com base nestes valores.

Uma abordagem semelhante pode ser feita ao considerarmos uma função de possibilidade. Como visto em Prade & Testemale (1984) e Raju & Majumdar (1988), um atributo com múltiplos valores pode ser registrado juntamente com seu valor de

possibilidade. Um exemplo desta forma de registro pode ser visto na segunda e quinta linha da Figura 3.1, atributo do domínio *"Imposto"*. De forma semelhante, Prade & Testemale (1984) propõem o uso de conjuntos de dados nebulosos. Esses conjuntos podem ser representados por rótulos de valores, ou seja, o uso de intervalos como "$15 - 20$" ou termos linguísticos como *"Baixo"*, *"Aprox. Alta"* ou *"Desconhecido"*, que fazem referência a um conjunto de dados. Sua teoria se baseia na distribuição de possibilidades para os elementos.

Em Umano e Fukami (1994) encontra-se a definição e uso das variáveis linguísticas *"desconhecido"* e *"indefinido"*, aplicados ao conceito de informação nula. Indicam a possibilidade de associar algum valor dentre o conjunto e quando não se sabe se pode-se associar algum valor dentre o conjunto, respectivamente.

O problema encontrado na abordagem de valores múltiplos, se deve pela ocorrência de tuplas similares, também consideradas como ambíguas ou redundantes, pois sua interpretação pode variar. Atributos com múltiplos valores dificultam o estabelecimento da individualidade das tuplas da relação.

Em Buckles & Petry (1982), Chen, Vandenbulcke & Kerre (1992), Umano & Fukami (1994) e Ma, Zhang & Ma (2000) o problema de redundância é considerado e tratado do ponto de vista das funções de similaridade e possibilidade. Deste modo é possível verificar se há ocorrência de tuplas duplicadas e quais destas poderiam ser removidas ou mescladas. A remoção de tuplas deve ser executada com cautela, pois tuplas redundantes produzem resultados semelhantes, mas não necessariamente iguais. A principal diferença está no trabalho de Ma, Zhang & Ma (2000) utilizando da similaridade e da distribuição de possibilidades de forma conjunta, denominada como teoria de *possibilidade-estendida*. O método atua na análise semântica dos dados, partindo da relação de similaridade de um atributo com o seu domínio, juntamente com a possibilidade do valor nebuloso proposto estar dentro dos limites de aceitação definido.

A importância na remoção de tuplas redundantes afeta diretamente no conjunto de tuplas que são obtidas ao executar uma instrução de consulta, de modo que a quantidade de tuplas a serem retornadas e a dupla interpretação destes resultados

podem ocasionar em informações incorretas.

O segundo tipo de abordagem, onde para cada tupla é atribuído um valor de pertinência, representando o conjunto destes domínios. Conforme Figura 3.1, este conceito é representado pela adição da coluna μ. Em estudo complementar realizado por Raju & Majumdar (1988) e sustentado por Umano & Fukami (1994), foi determinado o seu uso, o considerando como um *"valor verdade"* atribuído à tupla ou à relação. Este valor é obtido pela comparação entre os valores dos domínios de cada tupla, definido como *"relação nebulosa"*.

O conceito de relações nebulosas é utilizado como forma de representar o significado da relação entre os domínios, ou seja, podemos definir a medida da associação destes domínios em cada tupla, através dos valores dos atributos. Raju & Majumdar (1988) e Umano & Fukami (1994), caracterizam esta relação como uma variante da função de pertinência, formalmente definida por: $\mu_r : U_1 \times U_2 \times \ldots \times U_n \rightarrow [0, 1]$, sendo μ_r a relação nebulosa obtida sobre um subconjunto do produto cartesiano obtida dos universos U_i, para $1 \leq \mathrm{i} \leq n$.

O conceito de relação nebulosa pode ser dividido em dois outros tipos, denominados *"tipo-1"* e *"tipo-2"*, Raju & Majumdar (1988) e Umano & Fukami (1994). Uma relação nebulosa do *tipo-1* consiste na análise das relações. Para exemplificar, comparamos os atributos existentes nas colunas *"Nome"* e *"Departamento"* da Figura 3.1. Se substituirmos a coluna *"Experiência"* pelo grau de pertinência obtido do cálculo da relação das duas primeiras colunas, o resultado pode ser interpretado como: *A média de experiência obtida pelo grau de relacionamento do funcionário com seu departamento* ou ainda, *"o valor verdade do funcionário pertencer ao departamento"*. Uma relação do *tipo-2* consiste na análise de todos os valores dos atributos, associando-os ao um grau de pertinência em relação ao domínio, mesmo os atributos exatos. Desta forma, a mensuração dos atributos é possível e determina o grau de verdade dos elementos de uma tupla.

As diferentes abordagens apresentadas demonstram formas de tratar uma informação imperfeita através de dados nebulosos. Um exemplo de modelo ideal para o modelo relacional é apresentado por Medina, Pons & Vila (1994). Sua proposta

apresenta um modelo genérico para os bancos de dados relacionais nebulosos, (do inglês, *Fuzzy Relational Database Model (FRDBM)*). Este modelo foi desenvolvido com objetivo de suportar as informações imperfeitas e os dados nebulosos. Este modelo integra algumas das abordagens apresentadas e adiciona novos conceitos, como o uso das distribuições de possibilidades ao invés da similaridade proposta por Buckles & Petry (1982). Isto faz com que possa determinar com mais exatidão a pertinência dos dados referenciados nos domínios, aplica ainda, o grau de possibilidade de verdade de uma tupla quando compara a outra similar.

3.2.4. Consulta da Informação Nebulosa no Modelo Relacional

A falta de flexibilidade do modelo tradicional fez com que novos paradigmas surgissem. Em Ma (2007) se classificam as principais abordagens de consultas utilizadas no modelo relacional: (*i*) consultas com termos exatos incidindo sobre a atributos imperfeitos, inseridos diretamente como dados de um banco de dados, (*ii*) consultas com termos imperfeitos atuando sobre atributos exatos, e (*iii*) a ocorrência de ambas. Os dois últimos se apresentam, geralmente, quando não há certeza do que o usuário busca, ou ainda, quando não é definido um número limite de respostas, (Bosc & Pivert 1992). Isto se deve por não conhecer os detalhes da estrutura de dados ou ainda, por não conseguir definir uma instrução que retorne aquilo que se deseja.

A incorporação dos dados nebulosos fez com que diferentes formas de consultas fossem estudadas. A principal abordagem consiste em estender a linguagem padrão de consulta *SQL*, visto que está linguagem é conhecida pela maioria dos desenvolvedores. Assim, o objetivo principal é integrar as novas funções, sem comprometer aquelas já existentes.

Para tornar uma consulta *SQL* flexível o bastante, a princípio, deve-se oferecer suporte os diferentes tipos de dados nebulosos. Estes podem ser utilizados como operandos ou operadores na definição de uma consulta. A extensão da linguagem *SQL* desenvolvida por Takahashi (1991) apresenta formas de suportar estes diferentes tipos de dados nebulosos. Deste modo, propõe-se inicialmente a separação através da classificação dos tipos de consultas. Isto torna possível a compreensão se-

mântica da sintaxe da consulta, fazendo com que seja identificado a finalidade desta consulta. Para exemplificar, pode-se classificar a execução de uma consulta como uma instrução de modificação ou ainda, como instrução de seleção. As diferenças nos tipos de consulta determinam quais atributos são realmente importantes para esta consulta. Diante disso, determina-se então, através do grau de pertinência, o quanto uma tupla retornada satisfaz aos requisitos da consulta utilizados. Semelhante a esta abordagem, o trabalho de Bosc & Pivert (1992, 1995) incorpora dados nebulosos permitindo que consultas flexíveis sejam desenvolvidas em busca de melhor representar a necessidade dos usuários. Além de possibilitar o uso de termos nebulosos como operandos, este possibilita também o uso como operadores. Desta forma, caso necessário, para atingir a necessidade do usuário, um operador como *"aproximadamente X"* pode ser utilizado no predicado da consulta.

Considerando as respostas obtidas pela execução das consultas nebulosas, o trabalho de Chiang, Lin & Shis (1998) define formas de avaliar a força destas respostas. O método possibilita determinar com mais exatidão se uma tupla retornada pela execução de uma consulta nebulosa está realmente relacionada com a resposta esperada. A definição do método parte do conceito que não somente seja avaliado a semântica de uma consulta, mas também, seja comparado com os critérios utilizados no desenvolvimento. Este método pode ser utilizado quando há ocorrência de dados nebulosos ou exatos no modelo.

Um método menos invasivo adotado em alguns trabalhos consiste em converter as consultas nebulosas para consultas exatas. Este método tenta interpretar uma instrução de consulta nebulosa, identificando a sua relação com os dados existentes. Posteriormente, o aproxima a uma consulta exata para que seja compreendido pela linguagem de consulta já existente. Os trabalhos de Chen & Jong (1997) e Ma (2007) apresentam formas deste tipo de conversão. O objetivo deste método é possibilitar que uma consulta seja desenvolvida de modo intuitivo para o usuário. Consequentemente, somente antes de sua execução o sistema realiza a interpretação, convertendo-as para a sintaxe original. Além desta conversão, os trabalhos utilizam limiares de resposta. Isto faz com que possa ser definido um limite para as respostas aceitáveis, além de auxiliar na interpretação dos termos nebulosos utilizados.

Por fim, em Medina, Pons & Vila (1994), considera o uso de dados nebulosos nos atributos e no desenvolvimento das instruções de consultas. Este apresenta por meio do seu sistema *GEFRED*, um modo de utilizar os dados nebulosos de uma instrução de consulta aplicando-os nas operações de um banco de dados relacional. Possibilitando a definição de limiares de forma flexível aplicados aos critérios da consulta.

3.2.5. Considerações Finais

Nesta seção apresentam-se conceitos básicos pertencentes ao modelo relacional. Estes conceitos foram idealizados para elucidar certos pontos, como a forma da concepção de sua estrutura, terminologias do modelo, além da sua linguagem de consulta padrão. Posteriormente buscamos apresentar algumas pesquisas relacionadas a incorporação de dados imprecisos ao modelo relacional, com objetivo de apresentar algumas das formas de tratamento, armazenamento e recuperação destas informações.

Podemos perceber a evolução das pesquisas ao longo dos anos, realizada em diversos trabalhos, com diferentes propósitos e teorias. A Tabela 3.1 apresenta os estudos realizados com as abordagens de manipulação e armazenamento da informação nebulosa e a Tabela 3.2 as abordagens com relação a formas de consultas.

Com a mesma importância dos métodos de armazenamento, os conceitos de instruções de consultas buscam aperfeiçoar as formas de transformar dados imprecisos em dados bem definidos, uteis para os usuários. Além de possibilitar uma abrangência maior de resultados com a aproximação da forma de consultar informações com base no conhecimento humano, deixando que instruções procedurais fixas sejam necessárias para se obter o resultado desejado.

Tabela 3.1. Comparações das formas de representação de dados nebulosos no modelo relacional.

Trabalhos Pesquisados	***Dados Nebulosos em Atributos (i)***	***Relação Nebulosa nas Tuplas (ii)***
Buckles & Petry (1982)	•	–
Prade & Testemale (1984)	•	–
Raju & Majumdar (1988)	•	•
Chen, Vandenbulcke & Kerre (1992)	•	–
Umano & Fukami (1994)	•	–
Medina, Pons & Vila (1994)	•	•
Ma, Zhang & Ma (2000)	•	–

Legenda: • Aborda; ○ Aborda em parte; – Não aborda.

Fonte: Elaborado pelo autor.

Tabela 3.2. Comparações dos tipos de consultas no modelo relacional.

Trabalhos Pesquisados	***Consultas com Predicados Nebulosos (i)***	***Consultas em Dados Nebulosos (ii)***
Buckles & Petry (1982)	•	–
Prade & Testemale (1984)	•	–
Raju & Majumdar (1988)	•	•
Chen, Vandenbulcke & Kerre (1992)	•	–
Umano & Fukami (1994)	•	–
Medina, Pons & Vila (1994)	•	•
Ma, Zhang & Ma (2000)	•	–

Legenda: • Aborda; ○ Aborda em parte; – Não aborda.

Fonte: Elaborado pelo autor.

Inúmeras outras abordagens podem ser relacionadas com o objetivo de aprimorar ainda mais a integração de dados nebulosos ao modelo relacional. O processamento destes dados é importante devido à complexidade dos sistemas que continuam surgindo. As formas de representação e consulta destes dados são fatores importantes para a conclusão deste trabalho.

3.3. Modelos de Bancos de Dados de Objetos Nebulosos

Com a popularização dos bancos de dados de objetos, diversos estudos foram realizados a fim de integrar a lógica dos conjuntos nebulosos ao modelo. Estes estudos aproveitam a flexibilidade que o modelo de objetos possui em comparação ao seu antecessor, o modelo relacional.

3.3.1. Bancos de Dados de Objetos

Separamos está seção para elucidar alguns conceitos básicos do modelo de banco de dados de objetos.

A necessidade de um sistema que melhor se adaptasse a evolução das aplicações, ocasionou no surgimento dos *Banco de Dados de Objetos BDO* (anteriormente conhecido como *Banco de Dados Orientados a Objetos (BDOO)*). Em virtude do fato que as aplicações não seguiram o modelo de registro de dados tradicional, em forma plana, como no modelo relacional.

As principais vantagens que o modelo oferece é a liberdade concedida aos projetistas dos *BDOs*, em desenvolverem a estrutura e as operações necessárias, sem a necessidade de seguir uma estrutura padronizada, além de integrar diretamente as aplicações de forma transparente (Elmasri, Navathe, Pinheiro *et al.* 2005).

As terminologias pertencentes aos bancos de dados de objetos são elucidadas em Elmasri, Navathe, Pinheiro *et al.* (2005), algumas destas são elencadas a seguir.

O termo *"objeto"* é utilizado para denominar uma instância da classe. Consiste de uma materialização momentânea de uma classe, possuindo todos os procedimentos que a compõe, sendo estes seus estados e comportamentos. Possuem *"atributos"* com valores para definir seus estados e *"operações"* que definem seus comportamentos. Por deixarem de existir ao término de sua utilidade, podem também ser denominados como *"objetos transientes"*.

Um *BDO* oferece suporte a manipulação dos valores de um objeto, isto faz com que este objeto se torne permanente, desde que possa ter sua estrutura de dados armazenada. Esta transformação permite o compartilhamento dos valores entre outras aplicações ou na necessidade de manter os dados após o término da aplicação.

Para isso requer o uso de procedimentos específicos, definidos como *"nomeação"* e *"acessibilidade"*. Uma *nomeação* é definida pelo ato de classificar um objeto através de um nome único. A *acessibilidade* transforma um objeto em persistente quando o seu alcance é realizado a partir de um objeto persistente, considera-se também a sequência de referências utilizadas.

O conceito de *"Classe"* ou *"Tipo"* em *BDOs* faz referência aos objetos em geral. É considerada uma abstração da ideia dos objetos. Para definir um tipo inclui-se também a forma da estrutura de dados que o compõe, como seus atributos e operações.

Os atributos em *BDOs* possuem a mesma definição de atributos no modelo relacional, contudo, são definidos como *"variáveis de instância"*. Podem ser do tipo comum, denominados como *"literais"*, ou ainda, possuir uma *"identificação de objeto"* (do inglês, *Object Identification (OID)*). Um *OID* é gerado automaticamente pelo sistema e possui uma identidade única, na qual permanece oculto ao usuário externo.

O *"encapsulamento"* faz com que os atributos ou operações existentes nos objetos se tornem ocultos para o usuário. Este conceito continua em discussão visto que a desvantagem com a ocultação é a necessidade de definir todos as operações e atributos públicos previamente. Tendo em vista o fato do usuário necessitar conhecer a estrutura para assim realizar as operações, um objeto oculto e sem acesso se torna sem utilidade para muitas aplicações, sendo considerado desnecessário ao sistema.

A *"Hierarquia"* e *"Herança"* tem grande importância nas linguagens de programação orientada a objetos. A herança pode ser utilizada para definir novos tipos ou classes, na qual pode ser aproveitada sua estrutura principal e adicionada a novos atributos e métodos. Pode-se sobrecarregar os métodos ou operações para definir novas funcionalidades específicas.

A *"sobrecarga de operações"* ou *"polimorfismo"* é utilizada para definir novas funcionalidades as classes que estão herdando funções da principal. Essa sobrescrita de operações ocasiona na alteração de funcionalidades definida pela classe original, tornando mais específica a classe referenciada.

Do mesmo modo que o modelo relacional, o acesso aos dados de um objeto é feito por meio de instruções. Uma instrução desenvolvida na linguagem de consulta padrão *OQL* (do inglês, *Object Query Language (OQL)*) pode seguir ou não o padrão de escrita. Em virtude de sua característica ortogonal em relação a sua estrutura. Isto é, possibilita a realização de consultas aninhadas, onde o resultado pode ser utilizado como valor de atributo ou na definição de outras consultas subsequentes, desde que não comprometa sua sintaxe.

3.3.2. Incorporação da Informação Imperfeita no Modelo de Objetos

Como vimos o modelo de objetos possui semelhanças e aprimoramentos em relação ao modelo relacional. Diversos estudos foram realizados a fim de aproveitar destas melhorias. Os trabalhos de Ma (2007) e Shukla *et al.* (2011) fazem uma revisão entre os diversos estudos existentes. Esta integração define o modelo de objetos como *Bancos de Dados Orientados a Objetos Nebulosos* (do inglês, *Fuzzy Oriented-Object Database (FOODB)*).

O conceito definido para o modelo *Fuzzy Oriented-Object Database (FOODB)* considera a existência de informações imperfeitas em todos os níveis do modelo, (Ma 2007). Desta forma, os atributos, classes, tipo de classes, objetos, a relação entre um objeto e sua classe geradora e uma classe com sua superclasse ou subclasse devem possuir meios de manipular este tipo de informação.

Ainda que a flexibilidade seja a característica marcante do modelo orientado a objetos, as diferentes abordagens apresentadas em Ma (2007) e Shukla *et al.* (2011) demonstram claramente a complexidade em integrar a informação imperfeita corretamente no modelo. Além disso, há ainda a necessidade de definir um padrão de modelo, no qual seja possível modelar os dados nebulosos da mesma forma que os dados perfeitos já utilizados. Isto inclui o desenvolvimento de instruções de consultas simples e claras para a manipulação destes dados.

A seguir destacamos os estudos que demonstram conceitos da representação de dados nebulosos e método de consultas no modelo de bancos de dados orientados a objetos. Estes estudos realizam abordagens que complementam ou atualizam alguns dos aspectos vistos no modelo relacional definidos anteriormente.

3.3.3. Representação da Informação Nebulosa no Modelo de Objetos

Como definido anteriormente, existem diferentes níveis em que a informação imperfeita pode se apresentar. Deste modo temos diferentes abordagens realizadas ao longo dos anos que integram os conceitos da teoria dos conjuntos nebulosos com a representação da informação imperfeita em um banco de dados de objetos.

Semelhante a forma em que são tratados os atributos do modelo relacional, os atributos do modelo orientado a objetos devem permitir a representação da informação imperfeita. Considerando o conceito de informação imperfeita com a ausência de uma informação, temos o trabalho de Zicari (1990). O autor propõe o uso de termos nebulosos do tipo *"indefinido"* ou *"desconhecido"* para representarmos uma informação que existe, mas no momento não pode ser inserida. O uso destes valores como atributos permite que objetos possam ser criados corretamente e classes definidas, mesmo que algumas de suas informações sejam desconhecidas. Os termos são utilizados no lugar do valor nulo comum. Sendo possível deste modo expressar mais significado ao atributo do que simplesmente o registro do valor nulo.

Outra forma de representar uma informação imperfeita em atributos consiste do uso de variáveis linguísticas, podendo possuir múltiplos valores ou não. Como definido em Yazici & Koyuncu (1997), Umano *et al.* (1998) e Bordogna, Pasi & Lucarella (1999), os atributos podem expressar um conjunto de informações, ou ainda, estarem representados por um único termo que expresse o conjunto aproximado de valores. Para exemplificar, um atributo *"Temperatura"* do objeto *"Cidade"* pode ser definido como: $cidade.temperatura = \{quente, morno\}$. Interpretamos esta informação como a temperatura da cidade é quente ou morno ou os dois.

Uma classe pode ser utilizada para instanciar objetos, ou ainda, para classificar objetos que possuem características semelhantes. Esta é considerada nebulosa quando possibilita a instanciação de objetos de forma flexível ou ainda, quando não define seus limites precisamente, (Zicari 1990), (Umano *et al.* 1998), (Bordogna, Pasi & Lucarella 1999) e (Ma, Zhang & Ma 2004). Um objeto instanciado a partir de classes nebulosas se torna nebuloso, pois não terá suas propriedades definidas com exatidão. Em contrapartida, a definição da semelhança entre objetos se torna

flexível, visto que os limites que definem está classe é desconhecido.

O conceito de classes nebulosas é apresentado inicialmente por Zicari (1990) e Umano *et al.* (1998). As referidas classes têm por finalidade definir uma estrutura de objetos e de atributos de forma flexível. Considerando que classes devem definir os atributos e comportamentos que um objeto terá, ou seja, suas propriedades, uma classe nebulosa permite que objetos sejam instanciados sem que estas propriedades sejam definidas a princípio. Desta maneira, atributos com dados nebulosos podem ser utilizados, sem comprometer o esquema estrutural definido pela classe.

Seguindo o exemplo anterior, um objeto instanciado por uma classe nebulosa do tipo *"Cidade"*, com atributos *"nome"* e *"temperatura"*, pode ser definido por *"Cidade("São Paulo", unk)"*. Neste exemplo, possuímos o valor do atributo *"nome"*, contudo, não possuímos o valor do atributo *"temperatura"*. Este esquema de estrutura de classe não compromete a instanciação do objeto quando não se conhece o valor de um dos atributos.

A Figura 3.2 a seguir, apresentada por Umano *et al.* (1998), ilustra a ocorrência da informação imperfeita no esquema estrutural da classe e os objetos instanciados por esta. Como pode ser observado, na classe *"human"*, *"male"* e *"female"*, o uso de dados nebulosos é possibilitado pelas definições existentes nestas classes. Os objetos instanciados utilizam dados nebulosos como valor de seus atributos, como visto no valor inserido para o atributo *"age"*. O uso de variáveis linguísticas com o seu valor de pertinência e termos linguísticos, são utilizados para descrever o valor deste atributo.

Como definido anteriormente, objetos possuindo o mesmo tipo de propriedades são semelhantes. Entretanto, está compreensão é modificada pelas classes nebulosas. Os conceitos de classes nebulosas de Bordogna, Pasi & Lucarella (1999) e Ma, Zhang & Ma (2004), definem que uma classe não necessita ter seus limites definidos, ou ainda, que não seja necessário definir um tipo especifico para esta classe. Uma classe definida por uma informação classificada como imperfeita tem seu significado alterado. Para exemplificar, a classe *"Carros"* é um exemplo de classe precisa, todavia, a classe *"Carros Velhos"* é considerada uma classe nebulosa, pois o tipo

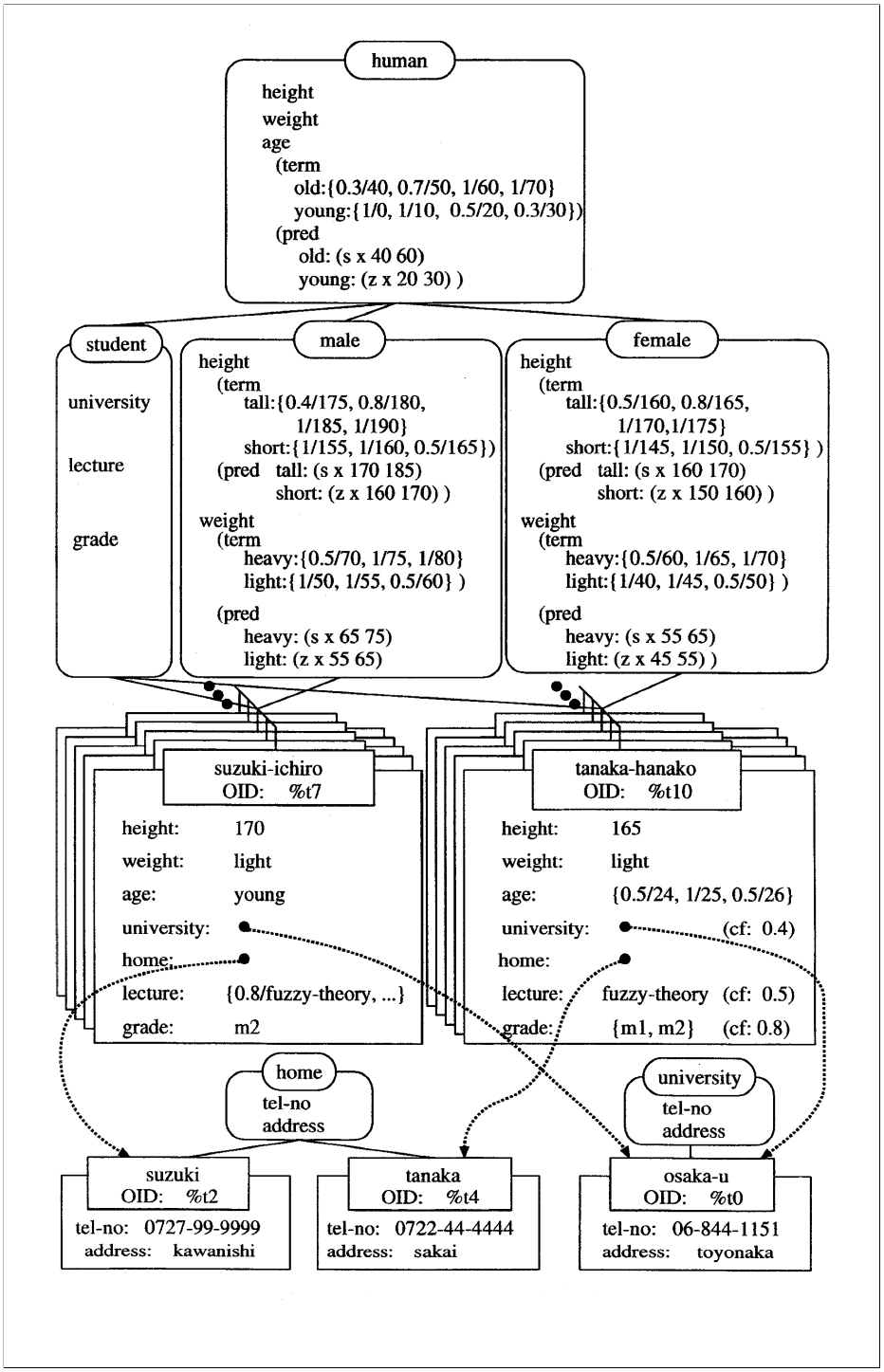


Figura 3.2. Representação de classes e objetos possuindo dados nebulosos.

Fonte: Adaptado de Umano *et al.* (1998).

"Velhos" que define a classe é nebuloso. Isto torna a relação de um objeto com esta classe, ou ainda, desta classe com sua superclasse ou subclasse, subjetiva à análise dos atributos que os compõe.

Neste sentido, a relação de um objeto com sua classe não pode ser determinada com precisão. Em George, Buckles & Petry (1993), George *et al.* (1996), Yazici & Koyuncu (1997), Bordogna, Pasi & Lucarella (1999) e Ma, Zhang & Ma (2004) os conceitos da relação nebulosa de um objeto com sua classe são definidos.

A relação nebulosa de um objeto com sua classe é definida com base na análise da relação deste objeto com sua classe. Em virtude do uso de classes nebulosas, o limite do tipo que esta classe classifica é indeterminado. Assim, a comparação de um objeto com sua classe geralmente é determinada através do valor de pertinência obtido da comparação dos atributos que as compõe.

Para determinar se um conjunto de objetos pertencem a uma determinada classe, deve ser considerado os atributos que identificam esta relação. Portanto, os atributos classificados como relevantes para a relação, determinam o valor de pertinência entre cada objeto e esta classe. Quanto maior o grau de pertinência, definido entre 0 e 1, mais forte é a relação entre o objeto e sua classe. Assim, ainda que um objeto seja instanciado a partir de uma classe não significa que este objeto a pertença completamente. Esta análise é essencial quando um objeto é instanciado a partir de mais de uma classe. Cada classe pode possuir um valor de pertinência diferente em relação ao objeto, tornando possível determinar qual destas classes define melhor este objeto.

Da mesma forma, a relação hierárquica de uma classe nebulosa com sua superclasse ou subclasse pode ser determinada. A avaliação deste tipo de relação se assemelha ao utilizado na definição da relação de objetos com suas classes. Em George, Buckles & Petry (1993), George *et al.* (1996), Yazici & Koyuncu (1997), Bordogna, Pasi & Lucarella (1999) e Ma, Zhang & Ma (2004) são definidos os conceitos que abordam a relação entre classes.

A definição de um valor de pertinência para representar a relação entre classes é definida como a forma de representar a intensidade ou força da ligação que uma classe possui com outra. Para exemplificar, utilizamos o modelo hierárquico da Figura 3.3, baseado no modelo de Bordogna, Pasi & Lucarella (1999). Vamos supor que exista definição dos esquemas de classes *"Projetos"*, *"Projetos Importantes"* e

"Projetos Comuns". Uma extensão da classe nebulosa *"Projetos Importantes"* é feita pela subclasse *"Projeto_01"*, onde o termo *"muito"* é utilizado para determinar a força da relação entre estas duas classes nebulosas. Uma segunda extensão é realizada pela subclasse *"Projeto_02"* da classe *"Projetos Importantes"* e definido a força desta relação como *"pouco"*. Os termos utilizados *"muito"* e *"pouco"* definem as relações como *"fortes"* e *"fracas"*, respectivamente. Desta forma é determinado que *"Projeto_02"* é uma instância fraca de *"Projetos Importantes"*, como também, é uma instância *"muito fraca"* de *"Projeto_01"*, uma vez que instâncias possuem certa relação entre estas, mesmo não declaradas explícitamente.

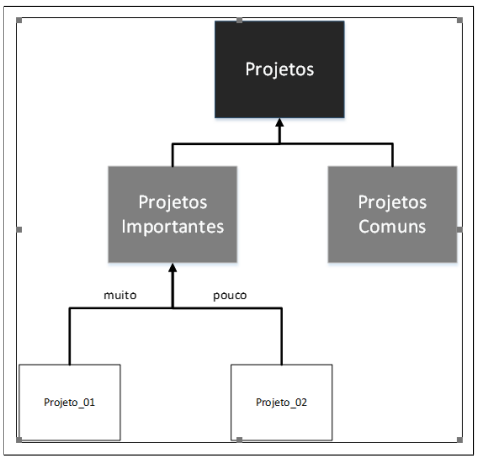


Figura 3.3. Representação hierárquica nebulosa.

Fonte: Adaptado de Bordogna, Pasi & Lucarella (1999).

3.3.4. Consulta da Informação Nebulosa no Modelo de Objetos

Similar a definição utilizada no modelo relacional, as instruções de consultas do modelo de bancos de dados de objetos devem possibilitar o desenvolvimento das instruções de consultas nebulosas. Para isto, a estrutura padrão da linguagem de objetos *OQL*, foi modificada a fim de incorporar os conceitos da teoria dos conjuntos nebulosos. As alterações mencionadas possibilitam a manipulação de dados imperfeitos, de forma similar aos conceitos utilizados na linguagem *SQL*.

A teoria dos conjuntos nebulosos integrada a *OQL*, possibilita: (*i*) consultas precisas realizadas em atributos contendo dados nebulosos, (*ii*) dados nebulosos no predicado das consultas e (*iii*) a junção destas abordagens.

Os diversos estudos existentes na literatura buscam apresentar novas formas de realizar consultas nos dados representados no modelo ou ainda, estender as definições da linguagem padrão. Uma destas abordagens, propõe uma extensão à linguagem, denominando como *FOQL* (do inglês, *Fuzzy Object Query Language (FOQL)*), de Nepal, Ramakrishna & Thom (1999). Denominação que melhor integra estes novos conceitos, pois utiliza a linguagem padrão como base para aplicação dos novos conceitos. Desta maneira, a estrutura padrão de uma *OQL* possibilita que as consultas nebulosas sejam realizadas em todos os níveis do modelo.

Uma sintaxe de consulta com as definições de uma *FOQL* pode ser definida baseada em *SQL*. Segundo a proposta de Ma, Zhang & Ma (2004), apresenta seguinte sintaxe:

SELECT $\langle lista\_de\_atributos \rangle$
FROM $\langle Classe_1\ WITH\ limiar_1,\ Classe_2\ WITH\ limiar_2, \ldots, Classe_i\ WITH\ limiar_i \rangle$
WHERE $\langle condição\_nebulosa\ WITH\ limiar \rangle$

Para exemplificar o uso desta sintaxe, realizando uma consulta nebulosa com base na classe nebulosa *"AlunosNovos"*, temos:

SELECT $\langle AlunosNovos.Altura \rangle$
FROM $\langle AlunosNovos\ WITH\ 0.5 \rangle$
WHERE $\langle AlunosNovos.Idade$=*"muito jovem" WITH 0.8*$\rangle$

Como resultado desta consulta, deverá ser retornado objetos do tipo *"AlunosNovos"* e seus valores para o atributo *"altura"*. Contudo, estes objetos devem possuir valor de pertinência de no mínimo *"0.5"*, definido por algum algoritmo que avalie a relação deste objeto com sua classe. Além disto, valores do atributo *"idade"* deste objeto devem ser aceitos como *"muito jovens"* dentro do limite *"0.8"*, conforme estabelecido.

Conforme exemplo dado, ao ser utilizado um valor nebuloso como parâmetro para um atributo, uma consulta nebulosa deve possuir métodos que tornem este valor comparável com outros valores do mesmo domínio. Desta forma, um dado nebuloso pode ser comparado com outro dado nebuloso, um dado preciso, um conjunto de dados precisos ou conjunto de dados nebulosos. Para tal fim são utilizadas as funções

de similaridade, variáveis linguísticas, distribuição de possibilidades, ou quaisquer outros métodos, que possibilitem a comparação.

Como exemplo do uso de variáveis linguísticas em consultas nebulosas que referenciam atributos de objetos, temos a proposta de Umano *et al.* (1998). Este utiliza os valores verdade de um atributo e seu predicado, definidos pela classe, para obter o valor de pertinência real, tornando-os comparáveis. Portanto, na definição da classe nebulosa *"AlunosNovos"*, deve existir um atributo de nome *"idade"* e o conjunto de valores possíveis para este atributo, sendo:

$$(idade\,(\%termo\,(jovem\{1/0, 1/10, 0.5/20, 0.3/30\})))$$

além de seu predicado, definido neste caso como: $(jovem\,(x)\ \ (zx2030))$. Este predicado faz com que um atributo nebuloso como *"jovem"* possa ser comparado com valores precisos ou conjuntos de valores, precisos ou nebulosos, a fim de determinar o valor de pertinência real da relação destes valores.

Abordagem similar é proposta em Na & Park (1996). Para o caso exposto anteriormente, uma variável linguística pode ser definida por meio de um termo restritivo e da distribuição de possibilidades. Assim, ao ser definido o valor *"jovem"* para uma variável linguística, podemos fazer uso de termos restritivos como *"muito"* ou *"mais ou menos"*, de modo a alterar a definição desta variável. Os valores reais relacionados a está variável passam a ser restritos pela definição semântica destes novos dados. Para exemplificar, suponhamos que a variável *"jovem"* seja delimitada pelos valores de *0* a *30*, como valores mínimo e máximo, respectivamente, das idades possíveis. Através da distribuição de possibilidades podemos obter o grau de pertinência de cada valor possível dentro deste limite. Com o dado restritivo atuando sobre esta variável, como *"muito jovem"*, os valores possíveis diminuem e se concentram nos valores mais próximos do mínimo definido.

Uma das características principais do modelo de objetos é a utilização de classes e objetos para representação dos dados. Consequentemente, consultas nebulosas devem possibilitar que dados nebulosos sejam definidos para expressar bem estas características.

Uma consulta nebulosa sobre objetos, sendo estes nebulosos ou não, possibi-

lita selecionar um conjunto de objetos de forma flexível. Isto ocorre em virtude do uso de limiares que, conforme definido em Nepal, Ramakrishna & Thom (1999) e Koyuncu & Yazici (2001), consideram a relevância dos atributos destes objetos. Por conseguinte o valor de pertinência determina a associação deste objeto em relação a outros objetos ou sua classe. Quanto maior o valor de pertinência, mais semelhança possui, sendo deste modo aceitos pelo limite definido na consulta. Este método de análise é também utilizado ao se avaliar as relações de classes com superclasses ou subclasses.

Para exemplificar, utilizando o exemplo dado anteriormente, o predicado $FROM\ AlunosNovos\ WITH\ 0.5$ faz referência ao limiar dos objetos instanciados pela classe *"AlunosNovos"*. Conforme definido por Nepal, Ramakrishna & Thom (1999) e Koyuncu & Yazici (2001), os atributos que definem a pertinência destes objetos são os atributos considerados como relevantes. A escolha dos atributos relevantes é determinada pela definição da condicional do predicado da consulta. O predicado do exemplo $WHERE\ AlunosNovos.idade = muito\ jovem\ WITH\ 0.8$ determina que o atributo *"idade"*, será responsável por determinar o quanto um objeto está relacionado com a classe. Estes objetos serão aceitos desde que está relação tenha no mínimo *"0.5"* de grau de pertinência.

3.3.5. Considerações Finais

O modelo de bancos de dados orientado a objetos nebulosos deve considerar a ocorrência da informação imperfeita em todos os níveis. As abordagens desenvolvidas conceituam a ocorrência deste tipo de informação e possibilitam o uso de dados nebulosos para fornecer mais significado a dados imperfeitos.

Abordamos aqui alguns dos trabalhos existentes que auxiliaram a nortear a execução do nosso trabalho. Os trabalhos citados foram selecionados pois utilizam alguns dos conceitos que podem ser empregados no modelo orientado a grafos, foco principal do nosso trabalho.

Relacionamos na Tabela 3.3 os trabalhos que representam a informação imperfeita, classificados pelo nível da ocorrência no modelo de objetos. As abordagens em que as consultas nebulosas estão melhor exemplificadas estão relacionadas na

Tabela 3.4, demonstrando também qual o tipo de consulta utilizada.

Tabela 3.3. Comparações das formas de representação de dados nebulosos no modelo de objetos.

Trabalhos Pesquisados	***Atributos***	***Classes***	***Objetos***	$\frac{Classe}{Objetos}$	$\frac{Classe}{Classe}$
Zicari (1990)	•	•	•	–	–
George, Buckles & Petry (1993)	–	–	–	•	•
George *et al.* (1996)	–	–	–	•	•
Yazici & Koyuncu (1997)	•	•	•	•	•
Umano *et al.* (1998)	•	•	•	–	–
Bordogna, Pasi & Lucarella (1999)	•	•	•	•	•
Ma, Zhang & Ma (2004)	•	•	•	•	•

Legenda: • Aborda; ○ Aborda em parte; – Não aborda.

Fonte: Elaborado pelo autor.

Tabela 3.4. Comparações de tipos de consultas no modelo de objetos.

Trabalhos Pesquisados	***Elementos Nebulosos no Predicado das Consultas***	***Consultas em Atributos Nebulosos***	***Abordagens Mescladas***
Umano *et al.* (1998)	•	•	•
Na & Park (1996)	•	•	•
Nepal, Ramakrishna & Thom (1999)	•	•	•
Koyuncu & Yazici (2001)	•	•	•
Ma, Zhang & Ma (2004)	•	•	•

Legenda: • Aborda; ○ Aborda em parte; – Não aborda.

Fonte: Elaborado pelo autor.

3.4. Modelos de Banco de Dados de Documentos *XML* Nebulosos

Nos últimos anos o modelo de bancos de dados baseado em documentos *XML* começou a ser melhor aproveitado, fazendo com que diversas questões fossem levantadas com relação a capacidade deste modelo tratar as informações imperfeitas.

Seguindo esta visão, a inclusão de dados nebulosos nos documentos *XML* passou a ser melhor investigado. Considerando a complexidade na manipulação das informações imperfeitas e os dados nebulosos, a modelagem destes dados, em uma

estrutura de dados semiestruturados ou não estruturados, se apresenta de forma atrativa.

3.4.1. Documentos *XML*

O modelo *XML* (do inglês, *Extensible Markup Language (XML)*) ou documento *XML* se enquadra na classificação de dados *"semiestruturados"* e *"não estruturados"*, sendo constituídos por uma sequência aninhada de elementos e atributos, (Elmasri, Navathe, Pinheiro *et al.* 2005).

Diferente dos modelos anteriores, o termo *"atributo"* se refere a uma descrição de um elemento da estrutura. O termo *"elemento"* é definido como a marcação de um item que compõe a estrutura, identificados pelo sinal $\langle \ldots \rangle$. A nomenclatura utilizada geralmente busca transmitir a ideia do que este elemento está se referindo. Vale ressaltar que para ser válido, o elemento deve possuir a definição $\langle / \ldots \rangle$ para identificar o término deste elemento. As marcações de elementos podem também ser denominadas *"tags"*, estão dispostos em diferentes níveis, sem qualquer limite de extensão em sua hierarquia.

Existem três tipos de classificações para os documentos *XML*, segundo Elmasri, Navathe, Pinheiro *et al.* (2005), são eles: (*i*) *XML* centrados em dados. Compostos por segmentos de dados, em sua maioria, geralmente estruturados ou semiestruturados; (*ii*) *XML* centrados no documento, compostos de trechos maiores de informações com poucos parâmetros; (*iii*) *XML* híbridos, os quais tem sua composição intercalada pelos dois anteriores, podendo ter ou não uma estrutura pré-definida, onde também são denominados *"documentos XML sem esquema"*.

A nomeação dos elementos dispostos no documento *XML* pode ser tornar complexa dependendo da quantidade de elementos e a profundidade que a estrutura atinja para representação das informações. Além do mais é permitido a nomeação dos elementos de modo subjetivo, sem qualquer padronização. Oferecendo uma liberdade maior ao desenvolvedor, contudo, limita o processamento, onde se faz necessário uma interpretação para que estes dados sejam válidos para outras aplicações. Neste contexto podemos realizar a definição destas estruturas utilizando um documento a parte que traduza este esquema de dados. Em Elmasri, Navathe, Pinheiro

et al. (2005) este documento é definido como *Documento de Definição de Tipo* (do inglês, *Document Type Definition (DTD)*) ou ainda arquivo de esquema *XML*. Um *DTD* mantém o padrão da estrutura de um *XML* comum, seguindo assim o modelo de árvore, com elementos aninhados. Pode especificar perfeitamente um documento *XML*, porém, possui desvantagens como a limitação dos seus tipos de dados, sua sintaxe especifica ou a ordenação dos elementos, os quais devem seguir a padronização para serem válidos.

Os problemas referidos fizeram surgir uma linguagem mais genérica que permitisse uma liberdade maior no seu desenvolvimento. A vista disso surge a *linguagem de esquema XML* ou simplesmente *esquema XML* (do inglês, *XML Schema Definition (XSD)*), (Elmasri, Navathe, Pinheiro *et al.* 2005). Similar ao *DTD*, um *XSD* possui características para realizar a elaboração deste documento. Permite que outras aplicações possam realizar a leitura dos dados corretamente, mesmo que o padrão esteja estipulado subjetivamente.

Duas linguagens de consulta se destacam no modelo *XML*, (Elmasri, Navathe, Pinheiro *et al.* 2005). O *XPath* tem por objetivo de realizar uma busca na estrutura do documento de forma a apresentar em seu resultado o caminho para os nós ou atributos de forma sequencial. Posteriormente o *XQuery* surgiu para oferecer mais suporte aos documentos *XML*, permitindo assim que consultas mais complexas fossem executadas, onde pudessem envolver mais de um documento.

3.4.2. Incorporação da Informação Imperfeita em Documentos *XML*

A estrutura flexível de armazenamento de dados de um documento *XML* oferece suporte a necessidade de inserção da informação imperfeita. Todavia, esta informação necessita de métodos de consultas, para que possam ser lidas e compreendidas por sistemas que necessitem destes dados.

Assim, destaca-se o trabalho de Ma & Yan (2016), o qual apresentada o estado da arte dos modelos de *documentos XML nebulosos*. Baseado neste trabalho, separamos algumas abordagens para exemplificar as formas de representação das informações imperfeitas e os métodos de consultas propostos.

3.4.3. Representação da Informação Nebulosa em Documentos *XML*

Do mesmo modo que nas abordagens anteriores, o modelo de dados *XML* é capaz de oferecer suporte a informação imperfeita através dos dados nebulosos. Este possui vantagens frente as demais abordagens, pois devido a sua estrutura de dados flexível possibilita representar um conjunto maior de tipos de dados.

A base da representação da informação imperfeita em um documento *XML* é feita através dos modelos de *DTD* e *XSD*, considerados como *DTD nebuloso* e *XSD nebuloso*, (Ma & Yan 2016). Estes modelos podem definir a forma como os elementos ou atributos destes elementos deverão representar as imperfeições das informações de um documento *XML*. Deste modo, Gaurav & Alhajj (2006), Ma & Yan (2007, 2016) e Panić, Racković & Škrbić (2014) classificam como *informação imperfeita de primeiro nível*, quando relacionados aos elementos e *informação imperfeita de segundo nível*, quando relacionados aos atributos destes elementos.

Uma das principais características de um documento *XML* é a capacidade de poder compartilhar diferentes tipos de informações com diversos outros sistemas, (Turowski & Weng 2002). A existência dos documentos de definição, como os *DTDs* e *XSDs*, informam a outros sistemas o modo de ler das informações contidas neste modelo. Portanto, nota-se a importância quanto a definição da forma lógica que um documento *XML* terá, para que desta maneira possa tratar as informações imperfeitas e os dados nebulosos corretamente.

Considerando a definição de documentos *XML* por meio de um *DTD*, diversos trabalhos foram realizados para incorporar a lógica nebulosa. Os trabalhos de Turowski & Weng (2002), Tseng, Khamisy & Vu (2005), Ma & Yan (2007) e Panić, Racković & Škrbić (2014) apresentam teorias para o modelo.

Um *DTD nebuloso* define como a informação nebulosa vai ser representada em um documento *XML* através de seus elementos e atributos. Esta informação pode ser representada por variáveis linguísticas, termos linguísticos, intervalo ou conjunto de valores precisos, entre outros. O exemplo desta representação é feito na Figura 3.4 de Turowski & Weng (2002), semelhante a abordagem de Tseng, Khamisy & Vu (2005). Como observado, no *DTD nebuloso* é proposto que exista um elemento do

tipo variável linguística, composto por termos linguísticos. Por sua vez, estes termos poderão conter um conjunto nebuloso em sua definição.

Para exemplificar, um documento *XML* poderá representar os dados nebulosos, através de um elemento do tipo variável linguística definida como *"idade"*. Para representar os valores desta variável, os elementos internos poderão ser definidos por meio de termos linguísticos. Desta forma, o termo *"jovem"* poderá ser representado por um conjunto nebuloso, sendo este composto por um par de dados, o valor real e o valor de pertinência deste valor.

⟨! **ELEMENT** *variavel_linguistica* (*termo_linguistico* ∗)⟩

⟨! **ATTLIST** *variavel_linguistica name* **CDATA #REQUIRED**⟩

⟨! **ELEMENT** *termo_linguistico* (*conjunto_nebuloso*)⟩

⟨! **ATTLIST** *termo_linguistico name* **CDATA #REQUIRED**⟩

Figura 3.4. Trecho de um *DTD nebuloso*.

Fonte: Adaptado de Turowski & Weng (2002).

Uma segunda abordagem na definição de um documento *XML* aplicado por um *DTD nebuloso* consiste das teorias de Ma & Yan (2007) e Panić, Racković & Škrbić (2014). Estes utilizam dos conceitos da distribuição de possibilidade para determinar a representação dos elementos e atributos. Desta forma, Ma & Yan (2007) inclui ao modelo dois novos elementos de definição, sendo eles denominados *"Dist"* e *"Val Pos"*. Conforme trecho do modelo ilustrado pela Figura 3.5, estes elementos determinam a distribuição de possibilidade através da *tag* ⟨**!ELEMENT** Dist⟩ e os valores de possibilidades obtidos pelo algoritmo, representados pela *tag* <***!ATTLIST*** *Val Pos*>.

De forma similar, Panić, Racković & Škrbić (2014) propõe a inclusão dos valores mínimo e máximo a funções de um elemento. Um trecho do modelo é ilustrado na Figura 3.6. As funções definidas neste trecho são utilizadas por algoritmos que determinam o valor de pertinência do elemento com base nos valores definidos por estas funções, limitadas pelos valores mínimo e máximo definidos.

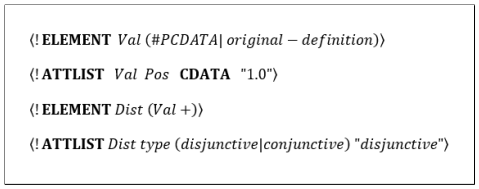


Figura 3.5. Trecho de um *DTD nebuloso*.

Fonte: Adaptado de Ma & Yan (2007).

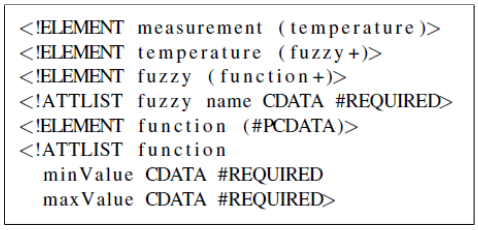


Figura 3.6. Trecho de um *DTD nebuloso*.

Fonte: Adaptado de Panić, Racković & Škrbić (2014).

O segundo tipo de abordagem relacionado a definição de um documento *XML*, consiste dos *esquemas XML* ou *XSD*. Este possui vantagem frente ao modelo *DTD*, pois conforme definido anteriormente, tem a estrutura mais flexível, possibilitando a definição um número maior de tipos de dados. Considerando a inclusão de informações imperfeitas e os dados nebulosos, o modelo *XSD* passou a ser denominado como *XSD nebuloso*, (Ma & Yan 2016).

Os trabalhos de Lee & Fanjiang (2003), Tseng, Khamisy & Vu (2005), Gaurav & Alhajj (2006), Üstünkaya, Yazici & George (2007), Oliboni & Pozzani (2008), Yan, Ma & Liu (2009) e Panić, Racković & Škrbić (2014) realizam propostas voltadas para o modelo *XSD*.

Diferente das abordagens realizadas para um *DTD*, a definição das informações perfeitas ou imperfeitas, utilizada em um *XSD* não se restringe a definição de elementos e atributos somente, pois possibilitam modelar o conjunto de dados por completo. Isto faz com que os dados nebulosos possam ser integrados, podendo ser especificados com mais profundidade e precisão do que em um *DTD*, (Tseng, Khamisy & Vu 2005).

Em muitos trabalhos, os documentos *XML* são determinados por meio dos registros de um banco de dados, como forma de extrair o conjunto de dados. A vista disto, um esquema *XSD* deve ser definido, onde as regras para esta conversão podem ser definidas. Os trabalhos de Lee & Fanjiang (2003) e Gaurav & Alhajj (2006) apresentam formas de realizar estas conversões, as aplicando em dados nebulosos do modelo relacional e de objetos, respectivamente. Em Yan, Ma & Liu (2009) temos

o mesmo tipo de abordagem realizada em Ma & Yan (2007), definindo as *tags Dist* e *Val Pos*, para o modelo *XSD*. Da mesma maneira, é definida a identificação da possibilidade para um elemento e a sua distribuição de valores possíveis através dos atributos.

Considerando os trabalhos apresentados, temos diferente tipos de abordagens para o mesmo modelo de dados, em virtude da flexibilidade que o modelo de documentos *XML* possui em modelar os dados nebulosos. Neste contexto, os trabalhos de Üstünkaya, Yazici & George (2007) e Oliboni & Pozzani (2008) apresentam propostas de desenvolvimento de um modelo genérico de incorporação de tipo de informação. Observando a granularidade existente ao definir uma informação imperfeita, estas propostas realizam o mapeamento dos dados nebulosos obtidos de forma mais precisa.

Realizando a comparação destes dois métodos de definição dos documentos *XML*, nota-se a vantagem em utilizar o *XSD* frente ao *DTD*, devido a liberdade na definição da estrutura de como o documento *XML* final irá representar as informações. Podemos observar estas diferenças na Figura 3.7, de Panić, Racković & Škrbić (2014). Aplicando o mesmo modelo utilizado no *DTD*, a definição dos elementos que irão representar a informação imperfeita pode ser expressa de maneira mais simples e com mais características. Podemos observar a definição do elemento identificado por *"measurement"* e sua composição complexa, definida por uma série de dados diferentes. Esta forma de definição de elementos era impossibilitada no modelo *DTD*.

Deste modo percebemos as vantagens do modelo de documentos *XML*. Sendo capazes de definir com precisão os aspectos das informações imperfeitas. Além disto, a possibilidade de ser interpretado pela maioria dos sistemas existentes, transforma este o modelo ideal para definição de dados complexos. Podemos observar na Figura 3.8 e Figura 3.9 a seguir, exemplos de documentos *XML nebulosos* obtidos através da definição de conceitos apresentados anteriormente. Estes modelos utilizam estruturas diferentes para representar as informações imperfeitas e os dados nebulosos. Isto demonstra as possibilidades que existem ao utilizar os conceitos do modelo de

```
<xs:schema id="Test" elementFormDefault="
    qualified" xmlns:xs="http://www.w3.org
    /2001/XMLSchema">
  <xs:include schemaLocation ="fuzzy.xsd"/>
  <xs:element name="measurement">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:
            string"/>
        <xs:element name="location" type="
            xs:int"/>
        <xs:element name="time" type="xs:
            dateTime"/>
        <xs:element name="temperature" type
            ="fuzzy"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figura 3.7. Trecho da definição de um documento *XML* por meio de um *XSD*.

Fonte: Adaptado de Panić, Racković & Škrbić (2014).

documentos *XML nebulosos* como suporte a informação imperfeita.

```
<measurement>
  <name>Weather bureau</name>
  <location>1</location>
  <time>2010-06-02 12:00</time>
  <temperature>
    <fuzzy name="cold">
      <function maxValue="10">1</function>
      <function minValue="10" maxValue
          ="20">-1/(Max-Min)*x + Max/(Max-
          Min)</function>
    </fuzzy>
    <fuzzy name="optimal">
      <function minValue="10" maxValue
          ="15">1/(Max-Min)*x - Min/(Max-
          Min)</function>
      <function minValue="15" maxValue
          ="20">-1/(Max-Min)*x + Max/(Max-
          Min)</function>
    </fuzzy>
    <fuzzy name="hot">
      <function minValue="10" maxValue
          ="20">1/(Max-Min)*x - Min/(Max-
          Min)</function>
      <function minValue="20">1</function>
    </fuzzy>
  </temperature>
</measurement>
```

Figura 3.8. Trecho de documento *XML*.

Fonte: Adaptado de Panić, Racković & Škrbić (2014).

```
<student SID = "20023056">
<sname>Tom Smith</sname>
<age>
<Dist type = "disjunctive">
<Val Poss = 0.4>23</Val>
<Val Poss = 0.6>25</Val>
<Val Poss = 0.8>27</Val>
<Val Poss = 1.0>29</Val>
<Val Poss = 1.0>30</Val>
<Val Poss = 1.0>31</Val>
<Val Poss = 0.8>33</Val>
<Val Poss = 0.6>35</Val>
<Val Poss = 0.4>37</Val>
</Dist>
</age>
<grade>95</grade>
<email>
<Dist type = "conjunctive">
<Val Poss = 0.60>TSmith@yahoo.com</Val>
<Val Poss = 0.85>Tom_Smith@yahoo.com</Val>
<Val Poss = 0.85>Tom_Smith@hotmail.com</Val>
<Val Poss = 0.55>TSmith@hotmail.com</Val>
<Val Poss = 0.45>TSmith@msn.com</Val>
</Dist>
</email>
```

Figura 3.9. Trecho de documento *XML*.

Fonte: Adaptado de Yan, Ma & Liu (2009).

3.4.4. Consultas da Informação Nebulosa em Documentos *XML*

Igualmente que para os modelos apresentados anteriormente, uma instrução de consulta é fundamental para interpretar os dados representados no modelo. Neste con-

texto, alguns estudos foram realizados a fim de contribuir com aperfeiçoamentos para a linguagem. Os trabalhos mais relevantes realizados na área se aproveitam das características da linguagem *XQuery*, escolhida pela grande variedade de funções e suporte à definição de instruções de consultas mais complexas.

Cita-se aqui os trabalhos de Üstünkaya, Yazici & George (2007), Ma, Liu & Yan (2010), Jin & Veerappan (2010) e Panić, Racković & Škrbić (2014), com importantes contribuições no desenvolvimento de consultas flexíveis permitindo o uso de dados nebulosos.

Considerando a forma em que uma instrução de consulta em um documento *XML* é feita, sendo esta através do caminho da relação de elementos e atributos, há a necessidade do uso de métodos para organizar os dados e a forma que será percorrido os caminhos. Portanto, uma consulta contendo dados nebulosos em sua sintaxe, deve realizar a análise dos atributos e elementos, contendo ou não dados nebulosos. Para Üstünkaya, Yazici & George (2007) a divisão em etapas para análise dos dados exatos e nebulosos é proposta. Para analisar as respostas corretamente, deve-se a princípio obter os dados exatos do critério e posteriormente, os dados nebulosos. Ademais, aplica o conceito de limiares para selecionar respostas possíveis como retorno das consultas.

Com base na consulta ilustrada pela Figura 3.10, busca-se *"Mostrar todos os alunos matriculados que possuam boas notas"*. Segundo a proposta de Üstünkaya, Yazici & George (2007), a princípio selecionam-se os *"alunos matriculados"*, sendo esta a informação exata e posteriormente, obtém-se os registros que atendam ao conceito de *"boas notas"*. Neste caso, para avaliar o conceito de *"boas notas"* deve-se obter a princípio o valor de pertinência dos atributos *"Excelent"* e *"Good"*. Por meio da função de similaridade, o limiar de aceitação é definido para este termo. O valor obtido limita o número de resultados para os que atendem a consulta definida.

```
for $b in input()/Subject_SimilarityTable/cell
where $b/@first_subject = "Excelent" and $b/@first_subject = "Good"
return $b/@similary_value";
```

Figura 3.10. Exemplo da consulta nebulosa.

Fonte: Adaptado de Üstünkaya, Yazici & George (2007).

Uma abordagem similar é proposta por Jin & Veerappan (2010) e Panić, Racković & Škrbić (2014) demonstrando consultas utilizando limiares de respostas. Adicionalmente, Panić, Racković & Škrbić (2014) define um método de classificar os caminhos de respostas com base no critério definido na consulta. Desta forma, é possível classificar respostas pelo seu grau de satisfação em comparação ao critério da seleção.

Por fim o trabalho de Ma, Liu & Yan (2010) define de forma matemática a relação dos critérios de uma consulta nebulosa, com os dados exatos e nebulosos que estão representados em um documento *XML*.

3.4.5. Considerações Finais

Buscamos apresentar alguns conceitos e abordagens fundamentais para o modelo de documentos *XML* nebulosos. Estes conceitos são importantes pois apresentam a forma de incorporar a informação nebulosa em estruturas que não necessitam de definição prévia. Esta característica torna mais atrativa o uso destes tipos de estruturas, visto que a flexibilidade supera muitas das limitações existentes nos modelos tradicionais.

O modelo de documentos *XML* nebulosos se sobressai aos demais, pois possui mais flexibilidade na definição dos dados que o compõe, em especial quando comparado com a forma de representar uma informação imperfeita. Ademais, o suporte a leitura por diversos outros sistemas possibilita a definição de estruturas que não necessitam ser exclusiva de um ou outro sistema. A Tabela 3.5 e Tabela 3.6 a seguir apresentam um resumo das abordagens selecionadas para o modelo de documentos *XML*.

Tabela 3.5. Comparações das formas de representação dos dados nebulosos no modelo de documento *XML*.

Trabalhos Pesquisados	***DTD***	***XSD***
Turowski & Weng (2002)	•	–
Lee & Fanjiang (2003)	–	•
(Tseng, Khamisy & Vu 2005)	•	•
Gaurav & Alhajj (2006)	–	•
Üstünkaya, Yazici & George (2007)	–	•
Ma & Yan (2007)	•	–
Yan, Ma & Liu (2009)	–	•
Oliboni & Pozzani (2008)	–	•
Panić, Racković & Škrbić (2014)	•	•

Legenda:* • *Aborda;* ○ *Aborda em parte;* – *Não aborda.

Fonte: Elaborado pelo autor.

Tabela 3.6. Comparações dos tipos de consultas do modelo de documentos *XML*.

Trabalhos Pesquisados	***Elementos Nebulosos no Predicado das Consultas***	***Consultas em Atributos Nebulosos***
Üstünkaya, Yazici & George (2007)	•	•
Ma, Liu & Yan (2010)	•	•
Jin & Veerappan (2010)	•	•
Panić, Racković & Škrbić (2014)	•	•

Legenda:* • *Aborda;* ○ *Aborda em parte;* – *Não aborda.

Fonte: Elaborado pelo autor.

CAPÍTULO 4

Bancos de Dados de Grafos Nebulosos

Os bancos de dados de grafos podem ser utilizados para armazenar boa parte dos diversos tipos de dados existentes, com a mais variada estrutura de informação. Em muitos exemplos, um grafo é utilizado para descrever o relacionamento de informações complexas. O objetivo é de facilitar o seu entendimento, pelo fato de tornar visível a complexidade destes relacionamentos. Neste contexto pode-se perceber as vantagens do modelo de bancos de dados de grafos, pois seu suporte à estrutura de dados complexos é de grande valia frente a manipulação de dados nebulosos.

4.1. Conceitos de Bancos de Dados de Grafos

Nesta seção separamos alguns conceitos básicos pertencentes ao modelo de bancos de dados de grafos, com objetivo de elucidar alguns aspectos fundamentais para melhor entendimento do modelo.

Um grafo é definido como uma estrutura composta por vértices e arestas, estes também definidos como relacionamentos, (Robinson, Webber & Eifrem 2013). A expressão para a topologia de um *grafo* G é definida por um conjunto de *vértices* V e *arestas* E, onde temos $G = (V, E)$. Cada aresta existente neste grafo é apresentada como a forma representativa de uma relação existente entre dois vértices. Um conjunto de vértices e arestas interligados é denominado como caminho de um grafo, (Penteado *et al.* 2014).

Bancos de dados de grafos podem ser utilizados para representar qualquer tipo de estrutura complexa de informação. A interconectividade é uma característica

importante para os dados armazenados em um grafo, considerada de valor igual ou superior quando se comparada a própria entidade. Esta relação representa a intensidade da conexão existente entre os dados de seus vértices, (Robinson, Webber & Eifrem 2013), (Kaliyar 2015), (Penteado *et al.* 2014), (Angles 2012) e (Angles & Gutierrez 2008).

O surgimento do modelo de bancos de dados de grafos se deu por volta do final da década de 80 e início da década de 90. Contudo, perdeu espaço para outras estruturas de dados, principalmente para os modelos geográficos ou semiestruturados, como o modelo baseado em *XML*, (Angles & Gutierrez 2008). Um banco de dados de grafos é composto por dados não estruturados, sendo então denominado *"Não Somente SQL"* (do inglês, *Not Only SQL (NoSQL)*). Esta classificação faz referência a estruturas de dados que não estão dispostas em registros sequencias ou planos como no modelo relacional tradicional. Possui uma característica estrutural dinâmica, pois pode ser modificada à medida que o modelo se amplia ou quando é necessário.

Nos últimos anos, o interesse nos bancos de dados de grafos voltou a crescer, como consequência da complexidade das aplicações existentes atualmente, que não é suportada perfeitamente pelos modelos de bancos de dados tradicionais. Para se adaptar, estes modelos realizam operações custosas, com inúmeros *"joins"*, com objetivo de interligar entidades e dados relacionados. Isto leva a uma grande dependência de dados e uma enorme sobrecarga de processamento, (Penteado *et al.* 2014).

Deste modo, o crescimento do modelo de grafos foi sustentado pela facilidade que possui em interagir com aplicações da vida real, as quais tem como principal característica a complexidade da interligação de seus dados. Um grafo pode facilmente ser modelado com base nos dados existentes e nas definições dos relacionamentos destes dados, (Angles & Gutierrez 2008) e (Kaliyar 2015).

O grafo de propriedades é o mais aceito como base do modelo para bancos de dados de grafos. Suas definições de atributos se assemelham ao conceito de atributos do modelo relacional. Tornando o modelo padrão para estruturas de bancos de

dados, (Rodriguez & Neubauer 2012). Neste modelo, em uma aresta ou em um vértice podem ser atribuídas propriedades ou atributos. Tais propriedades podem ser diferentes para cada classe ou tipo de vértice ou aresta.

Diferente dos modelos apresentados anteriormente, o modelo de grafos se destaca pelo foco nos relacionamentos. Estas relações podem continuar em expansão, sem a preocupação de comprometer a estrutura dos dados já existentes, (Robinson, Webber & Eifrem 2013). A característica principal dos relacionamentos formados neste tipo de estrutura é o de permitir o completo entendimento semântico existente. Partindo do seu vértice inicial até o seu vértice final, as arestas direcionadas determinam o fluxo do relacionamento da estrutura, sem outros vértices pendurados.

A Figura 4.1 de Robinson, Webber & Eifrem (2013) mostra uma estrutura de dados com interligações simulando o conceito de redes sociais. Utilizando o modelo de grafos de propriedades temos os vértices representando as pessoas e arestas representando seus relacionamentos. Rótulos são utilizados para apresentar uma descrição mais completa destes relacionamentos, juntamente com a direção das arestas, para identificar quais vértices estão sendo relacionados. Cada vértice possui ainda atributos utilizados para identificação e distinção de cada elemento.

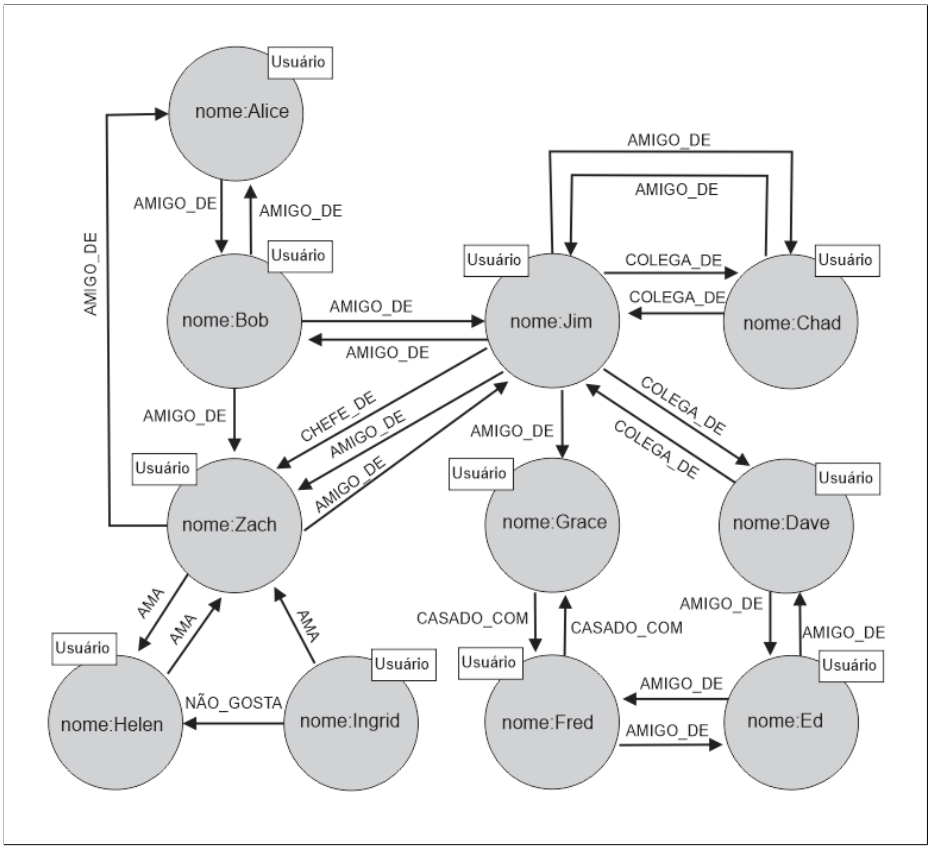


Figura 4.1. Grafo de propriedades simulando uma rede social.

Fonte: Adaptado de (Robinson, Webber & Eifrem 2013).

Atualmente existem diversos sistemas que oferecem recursos para manipulação dos bancos de dados de grafos. Em Angles (2012) e Kaliyar (2015) são realizados comparativos entre alguns dos principais sistemas existentes, sendo eles: *"AllegroGraph"*, *"DEX"*, *"HypergraphDB"*, *"InfiniteGraph"*, *"Infogrid"*, *"Sones"*, *"Trinity"*, *"Titan"* e *"Neo4j"*.

Semelhante aos bancos de dados apresentados anteriormente, o modelo de grafos possui diferentes linguagens para realização de suas consultas. Todavia, nota-se que a maioria dos trabalhos publicados fazem uso principalmente das linguagens *SPARQL* e *Cypher*.

O sistema *Neo4j* destaca-se dentre os sistemas existentes, considerado o sistema mais popular para bancos de dados de grafos. Este possui uma linguagem de manipulação proprietária e uma série de funcionalidades que auxiliam a manipulação dos dados e relacionamentos existentes em um grafo. Possui uma interface simples,

desenvolvida com o objetivo de ser de fácil compreensão para usuários iniciantes ou experientes. Em Robinson, Webber & Eifrem (2013) são apresentados conceitos da sua linguagem proprietária do sistema *Neo4j*, o *Cypher*.

A filosofia principal da linguagem *Cypher* consiste em ser simples, clara, de fácil leitura e entendimento por todos os seus usuários, (Robinson, Webber & Eifrem 2013). O *Cypher* busca eliminar a limitação existente em muitas linguagens, onde somente especialistas conseguem compreende-las e utiliza-las por completo. Neste contexto, sua sintaxe de consulta busca apresentar claramente a semântica da instrução desejada.

A Figura 4.2 adaptado de Robinson, Webber & Eifrem (2013) apresenta a sintaxe da instrução escrita em *Cypher*, juntamente com o subgrafo resultante da execução desta consulta. Nota-se que uma instrução em *Cypher* pode ser facilmente compreendida, pois se assemelha a um texto da linguagem comum. A sintaxe da instrução *Cypher* segue:

$$(Emil) < -[: CONHECE] - (Jim) - [: CONHECE] - > (Ian) - [: CONHECE] - > (Emil)$$

Deve ser lida da esquerda para a direita, onde teremos: *"Emil é conhecido por Jim, Jim conhece Ian e Ian conhece Emil"*. O uso da seta indica qual a direção do relacionamento, os parênteses indicam os vértices e os colchetes indicam os relacionamentos, ambos com seus rótulos de identificação.

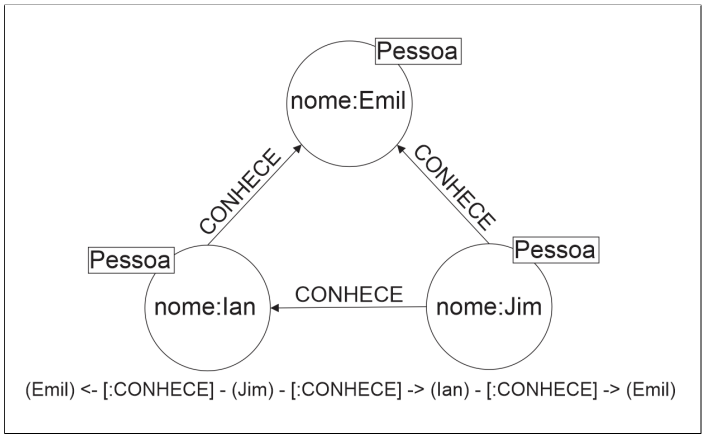


Figura 4.2. Exemplo da sintaxe de leitura utilizada pela linguagem *Cypher*.

Fonte: Adaptado de Robinson, Webber & Eifrem (2013).

De modo geral, uma consulta em um banco de dados de grafos é definida

pela sintaxe ⟨*padrão de caminho*⟩, ⟨*predicado*⟩ e ⟨*resultado esperado*⟩, sendo que:

- •⟨***padrão de caminho***⟩**:** é definido como uma sequência $v1, e1, v2, e2, v3, e3, \ldots$,-1$v_n$ alternadas dos vértices $v1, v2, v3, \ldots, v_n$ e arestas $e1, e2, e3, \ldots, e_n$;
- •⟨***predicado***⟩**:** é definido como um predicado booleano. Estabelece uma condição envolvendo atributos associados aos rótulos de vértices e arestas que fazem parte do ⟨*padrão de caminho*⟩;
- •⟨***resultado esperado***⟩**:** é definido como o modelo de apresentação dos resultados obtidos após a execução da consulta. Geralmente são utilizadas algumas das definições informadas no ⟨*padrão de caminho*⟩ ou no ⟨*predicado*⟩. A ausência desta definição resulta na apresentação de dados baseado nas informações do ⟨*padrão de caminho*⟩ definido.

A Figura 4.3 apresenta a sintaxe padrão de uma consulta na linguagem *Cypher*. Estão identificados os trechos referidos anteriormente.

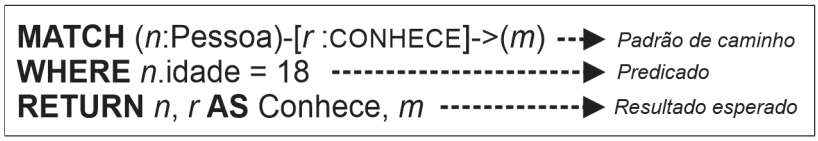


Figura 4.3. Exemplo de uma consulta na linguagem *Cypher*.

Fonte: Elaborado pelo autor.

A análise de um < *padrão de caminho* > pode conter outras definições, como o tamanho que este subgrafo possui, força de um caminho ou a distância entre dois vértices. Assim, como definido em Pivert *et al.* (2014), para um grafo G, sendo $G = (V, e)$, temos um caminho e, definido pela sequência $x_0 \rightarrow x_1 \rightarrow \ldots \rightarrow x_n (n \geq 0)$. Cada segmento é definido por $e(x_{i-1}, x_i) > 0$, onde $1 \leq i \leq n$, sendo n a quantidade de conexões do caminho. Para este caminho, pode-se avaliar:

a) **A força do caminho, expresso por:**

$$\text{Força}(e) = min_{i=1\ldots n}\ e(x_{i-1}, x_i)$$

Para exemplificar, com base na Figura 4.4, pode-se definir a força do caminho do vértice $v1$ até $v5$ pela expressão apresentada anteriormente. Deste modo, temos os caminhos $v1, v2, v3, v5$; $v1, v2, v4, v3, v5$; $v1, v2, v4, v5$. Se o

objetivo for o de obter a força destes caminhos, deve-se analisar cada segmento. Obtém-se então os valores de força dos caminhos: $v1, v2, v3, v5 = 3$; $v1, v2, v4, v3, v5 = 4$ e $v1, v2, v4, v5 = 3$. Deve-se observar que, para o exemplo em questão, todas as arestas possuem o mesmo valor, sendo este 1.

b) **O tamanho do caminho definido pela expressão:**

$$\text{Tamanho}(e) = \sum_{i=1}^{n} \frac{1}{e(x_{i-1}, x_i)}$$

Como exemplo, fazendo uso da Figura 4.4, objetiva-se determinar o tamanho do caminho, partindo do vértice $v1$ até o vértice $v3$. Podem ser obtidos os seguintes caminhos: $v1, v2, v3$; $v1, v2, v4, v3$; $v1, v2, v4, v5, v2, v3$; $v1, v2, v4, v5, v4, v3$. Com base na expressão é possível calcular o tamanho de cada um destes caminhos, sendo: $v1, v2, v3 = 2$; $v1, v2, v4, v3 = 3$; $v1, v2, v4, v5, v2, v3 = 5$; $v1, v2, v4, v5, v4, v3 = 5$. Conclui-se então que o menor caminho neste grafo é o caminho dos vértices $v1, v2, v3$, o qual utiliza somente duas arestas para interligar os vértices.

c) **A distância do caminho:**

$$\text{Distância}(x, y) = min_{todos\ caminhos\ x\ at\ y}\{\text{Tamanho}(e)\}$$

Para exemplificar, na Figura 4.4, objetiva-se determinar a distância do caminho de $v1$ até $v4$. Assim, temos os caminhos: $v1, v2, v4$; $v1, v2, v3, v5, v2, v4$; $v1, v2, v3, v5, v4$. Por meio da expressão, obtém-se:

$$v1, v2, v4 = 2;\ v1, v2, v3, v5, v4 = 4\ v1, v2, v3, v5, v2, v4 = 5;.$$

Observa-se que a menor distância deste grafo é o caminho passando por $v1, v2, v4$.

O resultado de uma consulta em um banco de dados de grafos, corresponde a todos os subgrafos $\{S1, S2, \ldots, Sm\}$ formados pelas sequências de rótulos dos vértices e arestas $v1, e1, v2, e2, v3, e3, \ldots, v_n$ que casam com o $\langle$*padrão de caminho*$\rangle$ e que os valores dos seus atributos satisfazem a condição booleana definida pelo $\langle$*predicado*$\rangle$.

Com exceção da definição do $\langle$*resultado esperado*$\rangle$, as definições do <*padrão de caminho*> e $\langle$*predicado*$\rangle$ não necessitam estar nesta mesma ordem. A linguagem *Cypher* possui uma estrutura maleável que não compromete sua execução. Permite a inversão ou intercalação da ordem dos grupos de definição a critério do usuário.

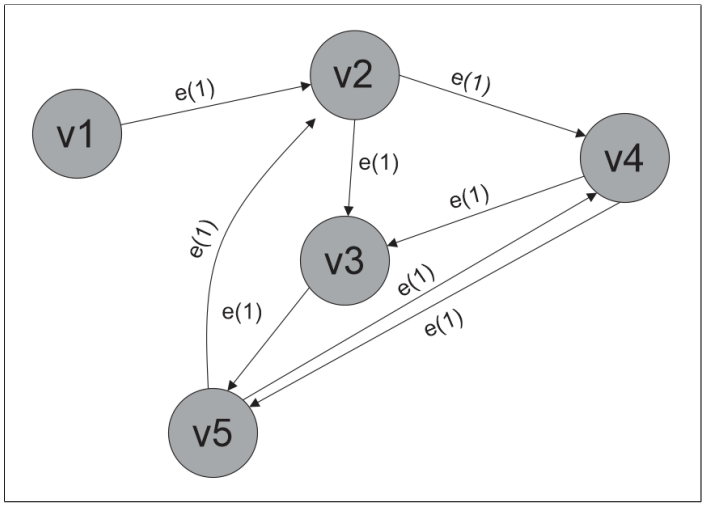


Figura 4.4. Grafo de propriedade.

Fonte: Elaborado pelo autor.

4.2. Modelos de Banco de Dados de Grafos Nebulosos

A integração do modelo de bancos de dados de grafos como suporte a informação imperfeita se apresenta com grandes possibilidades. A forma em que os dados são dispostos na estrutura contribui como forma de superar as limitações presentes nos outros modelos, pois restrições como formatação dos dados ou domínios, não necessitam ser consideradas em muitos casos. Em vista disto, torna-se ideal para a representação dos dados nebulosos, podendo armazenar diferentes tipos de dados.

Vale ressaltar que conforme demonstrado nas estruturas baseadas em *XML* e *BDO*, um modelo sem qualquer definição de estrutura de armazenamento de seus dados ou métodos de acesso, dificultam o desenvolvimento de instruções de consultas a fim de localizar ou relacionar seus dados.

4.2.1. Representação da Informação Nebulosa em Grafos

Como mencionado anteriormente, o crescimento do interesse pelos bancos de dados de grafos ocasionou também em um crescimento das pesquisas que buscam integrar os conceitos da lógica nebulosa com este modelo.

Na literatura existem diversos trabalhos publicados ao longo dos anos envolvendo os conceitos da aplicação da lógica nebulosa em grafos, denominado então como *"grafos nebulosos"*. Uma visão geral dos conceitos sobre grafos nebulosos é

apresentado em Sunitha & Mathew (2013).

Nos grafos nebulosos a informação nebulosa pode ocorrer tanto nos dados existentes nos atributos dos vértices, quanto na estrutura dos relacionamentos de um banco de dados de grafos, (Pivert *et al.* 2014) e (Pivert *et al.* 2016). Deste modo, indica que há a necessidade de permitir que dados nebulosos sejam inseridos como valores de propriedades de um vértice, bem como que um dado nebuloso seja utilizado para definir um relacionamento. Esta inserção faz com que a visão booleana da existência ou não de um determinado relacionamento deve ser reinterpretada. Podemos então analisar um caminho ou uma distância entre vértices de forma diferenciada, sendo influenciada pela definição destes dados nebulosos.

Na Figura 4.5, adaptado de Pivert *et al.* (2014), existe a ocorrência de dados nebulosos. Percebe-se que os relacionamentos dos vértices do grafo possuem valores que modificam a sua compreensão. O relacionamento do vértice *"Pierre"* com o vértice *"Serge"* é definido através de uma aresta rotulada como *"colaborador"* de valor *"0.3"*. Considerando as demais arestas com rótulos de *"colaborador"* e seus valores, identificamos que o maior valor atribuído é o valor *"1"*. Considera-se então que qualquer valor abaixo do valor *"1"* é um relacionamento de menor intensidade.

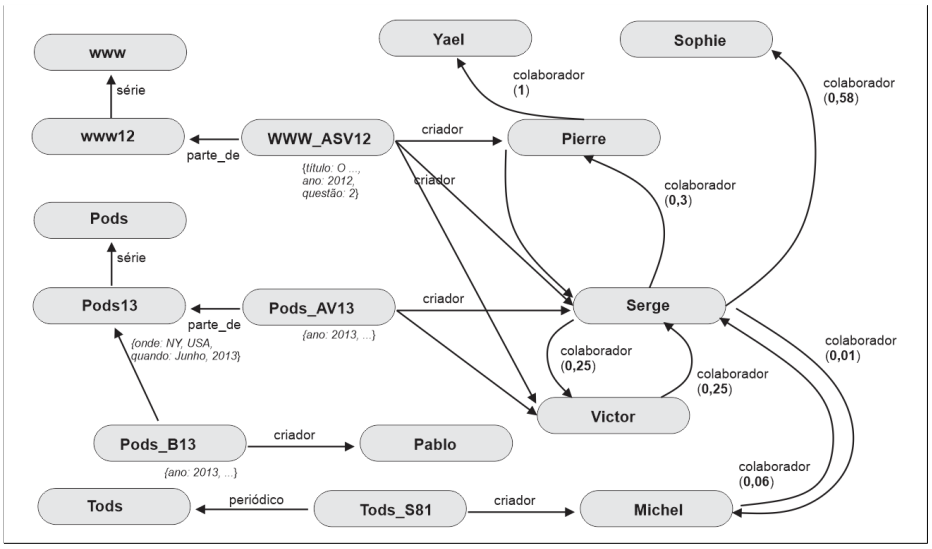


Figura 4.5. Modelo de representação do grafo nebuloso.

Fonte: Adaptado de Pivert *et al.* (2014).

Devemos considerar também no modelo de grafos nebulosos a sua capacidade

em oferecer suporte a ambos os tipos de dados, perfeitos e imperfeitos. O modelo determinado por Pivert *et al.* (2014) possui estes conceitos, oferecendo suporte equivalente tanto para a informação imperfeita quanto para a informação perfeita. A Figura 4.5, apresentada anteriormente, demonstra estes conceitos por meio de suas arestas.

Devido à natureza do modelo de banco de dados de grafos, a representação da informação complexa é simplificada. A maior complexidade consiste em oferecer métodos que possam relacionar um dado nebuloso com um dado preciso ou pelo menos possibilitar a sua interpretação. Portanto, grande parte dos estudos realizados com objetivo de incorporar a lógica nebulosa em bancos de dados de grafos são direcionados a execução de instruções de consultas. Estas consultas devem auxiliar os usuários a obter a informação requerida, independente da forma em que está informação está armazenada.

4.2.2. Consulta da Informação Nebulosa no Modelo Orientado a Grafos

Uma consulta no modelo de grafos é definida como uma forma de percorrer este grafo, entre os vértices e arestas existentes, retornando ao usuário uma lista dos vértices e arestas que atendam aos requisitos informados. Da mesma forma que um grafo comum, uma consulta em um grafo nebuloso deve seguir este conceito.

Assim, para possibilitar a incorporação da informação nebulosa, se faz necessário o desenvolvimento de métodos que atuem tanto em dados nebulosos quanto precisos. Uma instrução de consulta deve preferencialmente oferecer suporte na ocorrência de dados de ambos os tipos, seja no conteúdo dos vértices, em suas arestas ou na estrutura do grafo por completo, (Pivert *et al.* 2014) e (Pivert *et al.* 2016).

Deste modo, considera-se que exista a possibilidade de um dado nebuloso ocorrer em qualquer uma das partes que formam um grafo, como vértices, arestas ou em suas propriedades. Por consequência, devemos modificar a sintaxe da instrução de consulta a fim de que se torne flexível o bastante para atender a estes requisitos. Independente do sistema utilizado para manipulação de um grafo nebuloso, deve possuir meios de interpretar corretamente estas consultas.

Utilizando como base as principais linguagens do modelo de grafos, alguns

trabalhos apresentam extensões com novas funcionalidades. O objetivo destas extensões é possibilitar a incorporação da lógica nebulosa no modelo. Um dado nebuloso pode então ser aceito e compreendido pela linguagem que está sendo estendida.

Como exemplo de uma extensão para a linguagem *SPARQL*, Cheng, Ma & Yan (2010) apresentam sua linguagem denominada *f-SPARQL*. Em contrapartida Pivert *et al.* (2014) e (Pivert *et al.* 2016) apresentam uma extensão para o *Cypher*, denominada *FUDGE*. As extensões desenvolvidas para estas linguagens trazem aprimoramentos para a linguagem base, sem comprometer as funcionalidades já existentes. Desta maneira possibilitam que a sintaxe de escrita já conhecidades possam continuar sendo utilizadas, devendo somente adicionar os novos conceitos.

A linguagem *f-SPARQL* de Cheng, Ma & Yan (2010) foi desenvolvida para comportar uma série de regras de tradução. Estas traduções buscam converter uma instrução de consulta flexível na sua forma aproximada de uma consulta precisa. Considera-se uma instrução de consulta flexível ou nebulosa quando possibilita que seus usuários determinem suas preferências de consulta, diferente do método booleano. A Figura 4.6 a seguir, apresenta um trecho de uma instrução desenvolvida em *f-SPARQL*. As linhas iniciadas pela palavra *"FILTER"* indicam a preferência na seleção dos dados escolhida pelo usuário. Por meio deste são obtidos os dados que atendam aos requisito do campo *"idade"*, neste caso *"não muito jovem"* e *"não muito velho"*, como também do campo *"altura"*, sendo *"próximo a 175cm"*. Além disso, em ambos os casos os dados obtidos são selecionados a partir de limiares estipulados, como no caso *"0,9"* ou *"0,8"*. Estes atuam de modo a limitar a quantidade de resultados obtidos, uma vez que os termos imperfeitos utilizados podem obter uma grande quantidade de resultados.

De forma semelhante, mas direcionada para a linguagem *Cypher*, a linguagem *FUDGE* de Pivert *et al.* (2014) e Pivert *et al.* (2016), busca também traduzir as consultas flexíveis de seus usuários. Adicionalmente, seu trabalho apresenta a álgebra que torna possível a tradução destas consultas e a análise dos resultados para classificação. A Figura 4.7 apresenta uma instrução de consulta em *FUDGE* com suporte aos dados nebulosos. As linhas iniciadas pelas palavras *"DEFINEASC"*

e *"DEFINEDESC"* são palavras-chave para indicar ao sistema o modo de interpretação dos dados nebulosos *"short"* e *"recent"*. Com esta definição, os dados podem ser utilizados no restante da instrução. Posteriormente ocorre a conversão destes dados, tornando a instrução de consulta nebulosa em uma instrução de consulta precisa. Para exemplificar, a conversão ocorre quando no momento da execução, é identificado pelo sistema o uso de um dado nebuloso, como *"recent"*. Por meio do trecho de definição, no caso *"(2010, 2014)"*, o sistema converte a consulta nebulosa de *"year IS recent"* para *"year > 2010 AND year < 2014"*. Deste modo, uma consulta nebulosa pode obter os dados perfeitos armazenados.

```
#top-k FQ# with 30
SELECT ?X ?Age ?Height WHERE {
    ?X rdf:type Model
    ?X ex:hasAge ?Age  with 0.2.
    FILTER (?Age=not very young && ?Age=not very old) with 0.9.
    ?X ex:hasHeight ?Height with 0.8 .
    FILTER (?Height close to 175cm) with 0.8.
}
```

Figura 4.6. Instrução de consulta em *f-SPARQL*.

Fonte: Adaptado de Cheng, Ma & Yan (2010).

```
DEFINEDESC short AS (3,5)
DEFINEASC recent AS (2010,2014)
IN
MATCH
(ar1:Article)-[part_of.series]->(s1),
(ar2:Article)-[part_of.series]->(s2),
(ar1)-[:creator]->(au1:Author),
(ar2)-[:creator]->(au1:Author),
(au1)-[(contributor+)|Length IS short]->(au2:Author)
WHERE
s1.id=WWW AND s2.id=Pods AND ar2.year IS recent
```

Figura 4.7. Instrução de consulta em *FUDGE*.

Fonte: Adaptado de Pivert *et al.* (2014).

Observando as instruções da Figura 4.6 e Figura 4.7, nota-se que são apresentadas uma série de dados nebulosos. Estes atuam como operandos e operadores, definidos na forma de variáveis linguísticas, possuindo uma interpretação especifica para cada caso.

Em Cheng, Ma & Yan (2010) os dados nebulosos na forma de operandos são classificados em três tipos, sendo estes:

i. Dados nebulosos simples, representados por termos únicos ou atômicos. Como exemplo, temos o uso de palavras como *"curto"*, *"recente"* ou *"jovem"*;
ii. Dados modificadores, que atuam juntos aos termos simples. Estes alteram o significado dos dados nebulosos simples. Como exemplo, temos *"muito jovem"* ou *"mais recente"*;
iii. Dados compostos, que podem ser constituídos de dados simples, dados modificadores ou ambos. São dados que fazem a união através de um conectivo como *"e"*, *"ou"* ou *"não"*. Podem simplesmente unir, complementar ou oferecer alternância entre os dados. Como exemplo temos *"jovem ou muito jovem"* ou *"não muito jovem"*.

Para dados nebulosos utilizados como operadores, Cheng, Ma & Yan (2010) define:

i. O operador *"próximo de"* ou *"por volta de"*, estipulando os arredores de um valor. Como exemplo temos *"próximo de 30"* ou *"por volta de 40"*;
ii. O operador *"pelo menos"*, que aplica uma margem inferior de aceitação. Como exemplo temos *"pelos menos 20 anos de idade"*;
iii. O operador *"no máximo"*, que aplica uma margem superior de aceitação. Como exemplo temos *"no máximo 20 anos de idade"*.

Vale ressaltar que os operadores *"pelo menos"* e *"no máximo"* não considerados nebulosos dependendo da forma que são empregados. Deste modo, se considerarmos que, conforme exemplo dado, os limites não necessitam ser exatamente *20 anos*, mas também serão aceitos valores próximos, estes operadores se tornam nebulosos.

Um último ponto a ser abordado, consiste da análise do resultado de uma instrução de consulta nebulosa. Esta análise, feita em dados obtidos como resultado destas consultas, possuem duas abordagens, sendo: (i) a definição de limiares de aceitação das respostas e (ii) um método de classificação destes resultados.

A definição de limiares de aceitação de respostas consiste em atribuir um valor como forma de limitar a apresentação dos dados. Como definido anteriormente, uma consulta em grafos tem por objetivo obter um padrão de caminho. Desta forma, a aplicação dos limiares busca limitar a apresentação dos vértices e arestas que atendam aos requisitos.

Entretanto, vale ressaltar que a aplicação de um limiar de resposta pode ser utilizada em todos os trechos da instrução de consulta. Quando utilizado no trecho no padrão do caminho corresponde às arestas associadas a esse caminho. O predicado da consulta considera qualquer expressão nebulosa associada a propriedades de vértices e arestas e no resultado esperado, um filtro para apresentação destes resultados. Torna-se necessário a definição de qual a real interpretação que será utilizada no uso deste limiar.

Assim, para exemplificar, utilizando um limiar no padrão de caminho, entende-se que será mensurado o relacionamento dos vértices através de suas arestas. Por sua vez, a atribuição de um limiar no predicado de uma consulta indica que o valor dos atributos existentes nos vértices ou arestas será o responsável por limitar o resultado. Em Pivert *et al.* (2014) são apresentadas estas diferentes formas de interpretação.

O método der *"ranking"* apresentado em Cheng, Ma & Yan (2010) tem por objetivo classificar os dados obtidos, além de organizá-los para apresentação ao usuário. O método utiliza o valor de pertinência obtido na análise das condições definidas pelo usuário, comparadas com a soma de respostas possíveis. Assim, se os parâmetros desta consulta tem maior relevância sobre o padrão de caminho, o *ranking* é formado pelas respostas que possuam mais similaridade com os parâmetros definidos.

Um método de classificação também é abordado em Pivert *et al.* (2014) e Pivert *et al.* (2016). Utilizando a linguagem *FUDGE* é desenvolvida uma aplicação denominada *SUGAR* (do inglês, *System Based on Fuzzy Theory for (fuzzy) Graph Databases Querying (SUGAR)*). Além de aplicar os conceitos definidos para tradução de consultas nebulosas, está aplicação realiza a classificação dos resultados e os apresenta novamente ao usuário.

A Figura 4.8 ilustra o processo completo, partindo do momento de definição da consulta até seu retorno ao usuário. O módulo denominado *"Tradutor SUGAR"* realiza a conversão da consulta nebulosa *FUDGE*, transformando-a em uma consulta precisa *Cypher*. Posteriormente o envia para compilação pelo motor do *Cypher* original. Como retorno é obtido a lista de vértices e arestas que atenderam aos requisitos.

Estes por sua vez são direcionados para o módulo *"Calculadora de Pontuação SUGAR"*, para classificação e ordenação, e por fim, apresentados ao usuário.

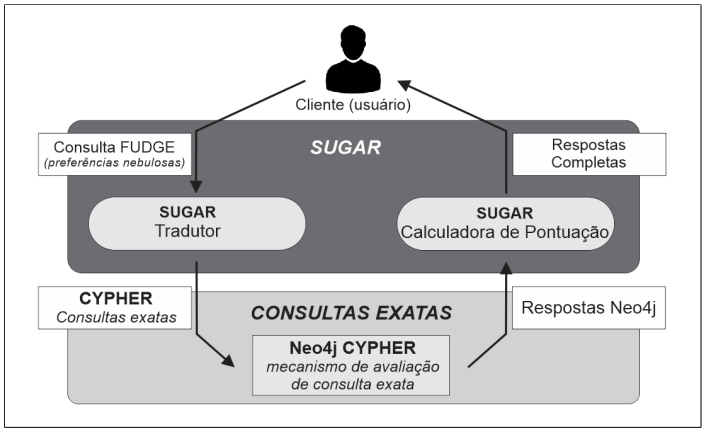


Figura 4.8. Arquitetura da aplicação *SUGAR*.

Fonte: Adaptado de Pivert *et al.* (2016).

Os trabalhos apresentados demonstram diferentes abordagens na incorporação da lógica nebulosa no modelo de grafos. Todavia existem inúmeras métodos que podem ser utilizados para a interpretação, em especial quando relacionamos dados imperfeitos com dados perfeitos. O trabalho de Castelltort & Laurent (2014) apresenta alguns dos desafios e oportunidades que devem ser considerados por aqueles que buscam realizar extensões na linguagem *Cypher*. Além de Pivert *et al.* (2016), que realiza um levantamento das propostas de extensão da linguagem *SPARQL*.

O trabalho de Castelltort & Laurent (2014) apresenta os diferentes níveis de modificação do sistema, como visto na Figura 4.9. A definição de níveis corresponde à profundidade da modificação ou integração necessária a ser realizada para que uma linguagem possa manipular os dados nebulosos. Segundo estes níveis temos:

i. Desenvolvimento de uma linguagem no nível acima a linguagem padrão. Está seria responsável pela formatação das instruções, convertendo-as para uma sintaxe suportada. Como exemplo temos a linguagem *FUDGE* apresentada anteriormente;
ii. Estender a linguagem *Cypher* adicionando novas funcionalidades. Estes conceitos se assemelham ao executado pela linguagem *f-SPARQL*;
iii. Aprimorar o núcleo do motor do banco de dados com um suporte melhorado

a dados nebulosos;

iv. Integrar conceitos dos itens (*ii*) e (*iii*), desenvolvendo uma estrutura mais robusta e completa.

Cada uma destas abordagens possui vantagens e desvantagens. Na escolha de uma abordagem, as vantagens e desvantagens devem ser consideradas. Conforme definido por Castelltort & Laurent (2014), quanto mais baixo o nível, mais conhecimento sobre o funcionamento da base da aplicação o desenvolvedor deve possuir. Todavia, as abordagens dos primeiros níveis são consideradas sem grande influência para o modelo como um todo.

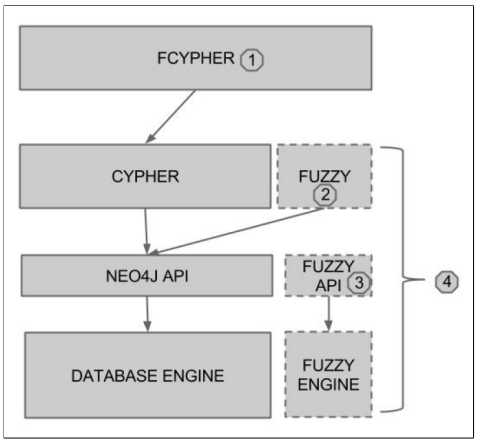


Figura 4.9. Níveis de modificação no sistema.

Fonte: Adaptado de Castelltort & Laurent (2014).

Como exemplo dos diferentes tipos de consultas que podem ser realizadas utilizando dados nebulosos, temos a Figura 4.10 de Castelltort & Laurent (2014). Observa-se que as consultas independem da posição de ocorrência do dado nebuloso. O desenvolvimento das instruções podem ser realizadas com base nas (1) propriedades dos vértices e arestas ou (2) na comparação entre vértices, ou ainda, (3) na seleção dos caminhos, com base na mensuração das arestas. Independentemente também do trecho da instrução a ser utilizado.

1) propriedades:

```
MATCH (h:Hotel)
WHERE CHEAP(price) > 0
RETURN h
ORDER BY CHEAP(h) DESC
```

```
MATCH (c:City)<-[:LOCATED]-(h:Hotel)
WHERE CLOSE(c,h) > 0
RETURN h
ORDER BY CLOSE(c,h) DESC
```

2) vértices:

```
MATCH (h1:Hotel),(h2:Hotel)
WITH h1 AS hot1, h2 AS hot2, SimilarTo(hot1,hot2) AS sim
WHERE sim > 0.7
RETURN hot1,hot2,sim
```

3) arestas:

```
MATCH (c1:customer)-[:KNOWS**almost3]->(c2:customer)
RETURN c1,c2
```

Figura 4.10. Exemplos de consultas nebulosas em *Cypher*.

Fonte: Adaptado de Castelltort & Laurent (2014).

Vale ressaltar que estas instruções de consultas propostas por Castelltort & Laurent (2014) são meramente conceituais. Este utiliza destes conceitos para nortear estudos posteriores.

4.3. Consideração Finais

Por meio dos trabalhos apresentados, pode-se observar que é possível a integração do modelo de bancos de dados de grafos com os dados nebulosos, uma vez que conceitos iniciais já foram testados e validados. Mais estudos devem ser realizados em busca de novas formas de aprimorar o modelo. Existem grandes possibilidades de integração com a lógica nebulosa, pois o modelo possui baixas restrições, quando comparado a outros modelos de bancos de dados existentes.

As Tabela 4.1 e Tabela 4.2 a seguir, relacionam os trabalhos abordados nesta seção. Relacionamos poucos trabalhos referentes a representação de dados nebulosos, pois o foco na maioria dos trabalhos consiste em aprimorar as linguagens de consultas. Entretanto, como mencionado neste trabalho, apesar da representação de dados ser facilitada, a forma de relacionar um dado nebuloso com um dado preciso é complexa. Isto demonstra a importância e a necessidade do aprimoramento das linguagens de consultas e manipulação.

Tabela 4.1. Comparações das formas de representação dos dados nebulosos no modelo de grafos.

Trabalhos Pesquisados	*Propriedades dos Vértices*	*Rótulos das Arestas*	*Estrutura do Grafo*
Pivert *et al.* (2014)	•	•	•
Pivert *et al.* (2016)	•	•	•

Legenda:* • *Aborda;* ○ *Aborda em parte;* – *Não aborda.

Fonte: Elaborado pelo autor.

Tabela 4.2. Comparações dos tipos de consultas do modelo grafos.

Trabalhos Pesquisados	*Elementos Nebulosos em Consultas*	*Consultas em Atributos Nebulosos*	*Estrutura Nebulosa*
Cheng, Ma & Yan (2010)	•	–	–
Castelltort & Laurent (2014)	•	–	–
Pivert *et al.* (2014)	•	•	•
Pivert *et al.* (2016)	•	•	•

Legenda:* • *Aborda;* ○ *Aborda em parte;* – *Não aborda.

Fonte: Elaborado pelo autor.

CAPÍTULO 5

Proposta da Pesquisa

Destinamos este capítulo para apresentar detalhes sobre o projeto de pesquisa proposto nesta dissertação. Além disso, são apresentados algumas conclusões obtidas da revisão que norteiam nosso projeto. Na sequência são apresentados detalhadamente os diferentes aspectos de nossa proposta.

5.1. Definição do Objetivo

Como definida na introdução, nosso objetivo é propor e implementar um modelo de inserção e representação de dados em bancos de dados de grafos, que suporte tanto dados nebulosos quanto precisos. Este modelo deve também permitir que as consultas sobre esses dados tenham a flexibilidade para incluir atributos nebulosos e que sejam obtidas respostas contendo tanto dados exatos quanto dados nebulosos. As consultas devem permitir também o uso de limiares sobre o grau de compatibilidade das respostas que possam ser definidos pelo próprio usuário.

Para atingir nosso objetivo desenvolvemos uma aplicação que permita manipular dados e relacionamentos nebulosos, podendo ser inseridos e manipulados em um grafo. A aplicação é responsável também pela tradução de consultas nebulosas, tornando-as compatíveis com a sintaxe padrão de consulta dos grafos.

Com o conhecimento obtido da revisão bibliográfica ao longo desta pesquisa, definimos certos pontos de vista que sustentam e orientam a nossa proposta:

a) O tipo da informação a ser inserida ou o termo nebuloso requisitado influencia diretamente no tipo de função que deve ser utilizada na consulta ou

recuperação dos dados;

b) O uso da função de similaridade, distribuição de possibilidades, variáveis linguísticas e relações nebulosas auxiliam na conversão e recuperação dos dados nebulosos. Estes recursos de representação da informação nebulosa oferecem a semântica necessária para suportar os dados e relacionamentos envolvidos;

c) O uso de um limiar estabelecido de forma subjetiva pelas necessidades dos usuários, oferece a possibilidade de um conjunto maior de respostas. O resultado torna-se mais próximo do que foi requisitado. Este método é mais atrativo aos usuários em comparação ao método booleano, pois em muitos casos, um resultado aproximado é melhor do que a ausência de resultados;

d) A forma de análise do relacionamento dos domínios contidos nas tuplas do modelo relacional e que permitem atribuir um valor nebuloso a uma tupla, se assemelha ao tipo de análise do relacionamento dos atributos dos vértices de um grafo, que permitiria também atribuir um valor nebuloso a um vértice;

e) O modelo de bancos de dados de objetos possibilita representar relações nebulosas perfeitamente, visto que o modelo possui muitas de suas definições baseadas em estruturas hierárquicas. Os conceitos apresentados na definição da relação de uma classe com os objetos instanciados ou da relação existentes entre estes objetos se assemelha a estrutura de um grafo. A lógica nebulosa aplicada ao modelo permite a definição das relações hierárquicas de forma flexível;

f) O modelo de documentos *XML nebulosos* tem os requisitos necessários para modelar os dados nebulosos. Este possui meios de definir boa parte dos conjuntos de dados nebulosos e a informação imperfeita relacionada de forma semântica. O compartilhamento das informações contidas em um documento *XML nebuloso* possibilita que seja definido uma estrutura de dados com especificações e funções que auxiliam a um sistema interpretar os dados contidos neste modelo. Isto torna o modelo ideal para definição de dados nebulosos de forma genérica;

g) O modelo de bancos de dados de grafos possibilita o desenvolvimento de

estruturas de dados relacionados, tornando visíveis suas relações complexas. O modelo de grafos de propriedades é o mais utilizado quando se objetiva construir uma base de informações. As aplicações que utilizam o modelo possuem ferramentas que podem ser utilizadas para obter as informações que estão contidas no modelo. Além do fato de seu principal gerenciador, o *Neo4j*, possuir uma linguagem de fácil compreensão e utilização.

Os pontos apresentados têm como objetivo nortear nosso trabalho e sua integração com o modelo de grafos.

Entre as propostas existentes atualmente, os modelos que possuem mais compatibilidade com o objetivo deste trabalho são os trabalhos de Pivert *et al.* (2016) e Castelltort & Laurent (2014).

Em Castelltort & Laurent (2014) são apresentados alguns apontamentos, com o objetivo de orientar uma extensão para a linguagem *Cypher*. Todavia, nenhuma implementação foi realizada. Deste modo, utilizamos os conceitos teóricos como suporte ao desenvolvimento deste trabalho. Em Pivert *et al.* (2016) apresenta-se a aplicação denominada SUGAR, baseado na linguagem estendida do *Cypher* denominada *FUDGE*. A extensão da linguagem e a aplicação apresentadas, são compatíveis com o objetivo deste trabalho. Todavia, são apresentados somente alguns dos conceitos nebulosos fundamentais utilizados na integração. Neste trabalho buscamos aprimorar estes conceitos.

Relacionam-se a seguir as comparações realizadas entre os tópicos abordadas por este trabalho e os trabalhos compatíveis existentes apresentados.

Tabela 5.1. Comparação entre as definições abordadas e os trabalhos existentes.

Item	*Tópicos abordados*	*Modelos*		
		A	***B***	***C***
1	*Extensão da linguagem Cypher.*	•	•	•
2	*Aplicação prática integrada à lógica nebulosa.*	–	•	•
3	*Uso de dados imperfeitos em instruções de consultas.*	○	•	•
4	*Instruções de inserção contendo dados imperfeitos.*	–	–	•
	Dados imperfeitos na estrutura do banco de dados			
5	*Representação de dados imperfeitos em um banco de dados de grafos.*	–	○	•
	Tipos de dados abordados			
6	*Valores Nulos.*	–	–	•
7	*Termos Linguísticos.*	•	○	•
8	*Intervalos de Valores.*	–	–	○
9	*Conjunto de Dados.*	–	–	○
10	*Expressões Nebulosas.*	○	–	•
11	*Limiares.*	–	–	•
	Aplicação com Base na Teoria dos Conjuntos Nebulosos			
12	*Variáveis Linguísticas.*	○	○	•
13	*Função de Similaridade.*	–	–	•
14	*Distribuição de Possibilidades.*	–	–	○
15	*Função Trapezoidal/Triangular.*	○	○	•
16	*Relação Nebulosa.*	–	–	○

Legenda 1: **Modelo A: Castelltort & Laurent (2014); Modelo B: Pivert *et al.* (2016); Modelo C: Proposta deste trabalho.**

Legenda 2:* • *Aborda;* ○ *Aborda em parte;* – *Não aborda.

Fonte: Elaborado pelo autor.

Item 1: Todos os modelos abordam as extensões para a linguagem *Cypher*. O *modelo A* propõe possíveis extensões. O *modelo B* apresenta aprimoramentos da linguagem com o *FUDGE*. O *modelo C*, busca integrar os conceitos apresentados pelos *modelos A* e *B*;

Item 2: O *modelo A* não apresenta aplicações, somente a formulação de uma teoria. O *modelo B* apresenta uma aplicação denominada *SUGAR*. Esta integra conceitos da lógica nebulosa por meio de novas funcionalidades incorporadas na linguagem *FUDGE* proposta. No *modelo C* é desenvolvido uma aplicação

nos moldes do *SUGAR*, aplicando ainda mais funcionalidades;

Item 3: O *modelo A* utiliza dados imperfeitos como exemplos, para demonstrar uma futura integração com os conceitos da lógica nebulosa. O *modelo B* e *C* utilizam dados imperfeitos diretamente na execução de consultas;

Item 4: Somente o *modelo C* possibilita o desenvolvimento de instruções de inserção que comportem dados imperfeitos;

Item 5: O *modelo A* não aborda a possibilidade de dados imperfeitos existirem em um banco de dados de grafos. O *modelo B* utiliza o conceito de arestas nebulosas. Em um grafo definido previamente, o valor de uma aresta é definido entre $[0, 1]$. Este dado nebuloso é utilizado posteriormente na análise desta estrutura. Com o *modelo C*, pode-se inserir dados nebulosos no decorrer do uso do banco de dados. Um tipo de dado imperfeito pode ser representado de maneira distinta de outra na estrutura;

Item 6: Somente o *modelo C* aborda o conceito dos tipos de dados imperfeitos de valores nulos;

Item 7: O *modelo A* faz uso de termos linguísticos na apresentação da teoria. O *modelo B* possibilita o uso de poucos termos linguísticos. O *modelo C* possibilita o uso de diversos termos linguísticos, definidos subjetivamente pelo usuário;

Item 8: Somente o *modelo C* aborda o conceito dos tipos de dados imperfeitos com intervalos de valores;

Item 9: Somente o *modelo C* aborda o conceito dos tipos de dados imperfeitos com conjuntos de dados;

Item 10: O *modelo A* aborda os conceitos de expressões nebulosas como possíveis propostas da extensão. O *modelo B* não apresenta propostas. O *modelo C* possibilita o uso deste tipo de dado;

Item 11: Somente o *modelo C* aplica o conceito de limiares. Estes são utilizados nas instruções de consultas, ampliando a definição dos dados imperfeitos. Em especial, é utilizado em termos linguísticos ou variáveis linguísticas, pois se relacionam perfeitamente com este tipo de dado;

Item 12: Todos os modelos abordam o conceito de variáveis linguísticas. Todavia, somente algumas variáveis linguísticas podem ser utilizadas pelo *modelo B*. No *modelo C* diversas variáveis linguísticas podem ser utilizadas;

Item 13: Somente é abordado pelo *modelo C*. Neste, uma função de similaridade é utilizada para mensurar o valor da relação existente entre dois termos linguísticos;

Item 14: Somente o *modelo C* apresenta contribuições com o uso da distribuição de possibilidades. Por meio deste é possível obter a relação de um valor exato com um conjunto representado por uma variável linguística;

Item 15: Os *modelos B* e *C* abordam o uso das funções trapezoidal/triangular. Com esta função pode-se obter o valor de pertinência de um termo linguístico;

Item 16: Somente o *modelo C* apresenta algumas definições no uso das relações nebulosas. Esta função é utilizada para mensurar a relação existente entre dois vértices.

5.2. Modelo Genérico de Inserção, Representação e Consulta de Dados

Diante do exposto, passaremos então à formulação do modelo que incorpora as informações imperfeitas e os dados nebulosos no modelo de grafos.

O modelo envolve três aspectos principais: (*i*) a inserção de dados nebulosos como valores das propriedades dos vértices e rótulos das arestas, (*ii*) a representação dos dados nebulosos na estrutura do modelo de banco de dados de grafos e (*iii*) as formas de consultas no modelo utilizando os dados nebulosos.

5.2.1. Inserção da Informação Imperfeita

De modo geral, o modelo proposto deve permitir a inserção da informação imperfeita, representada pelos dados nebulosos como valores nos atributos de vértices e arestas do grafo. Assim, é necessário o desenvolvimento de algoritmos que auxiliem, quando necessário, na interpretação desses dados nebulosos utilizados.

Adicionalmente, deve-se permitir a atribuição de uma função que atue como valor nebuloso de uma aresta. Este valor deve expressar a intensidade ou o valor de verdade da relação entre estes dois vértices. A finalidade desta abordagem é possibilitar que o valor da relação dos dois vértices ou do próprio vértice seja flexível, podendo ser alterado posteriormente, devido a alguma alteração dos valores das

propriedades.

5.2.2. Representação da Informação Imperfeita

Um grafo naturalmente demostra as diversas relações existentes entre dados simples ou complexos. Por este motivo, deve-se possibilitar a representação da relação de uma informação imperfeita com as demais informações perfeitas no modelo de grafos. Esta abordagem tem por objetivo tornar possível a visualização da complexidade existente na relação deste tipo de informação.

Levando em consideração os diversos tipos de sistemas que manipulam os bancos de dados de grafos, devem ser utilizados aqueles que possuam formas de apresentar visualmente como estes dados estão relacionados. Este requisito tem por objetivo facilitar a compreensão de como os dados estão relacionados.

5.2.3. Consultas com Informações Imperfeitas

Com relação a consulta em bancos de dados de grafos, o uso de dados nebulosos em uma instrução deve atuar sobre os dados existentes, sendo exatos ou nebulosos, de forma semelhante. Da mesma forma, o uso de uma instrução comum deve atuar sobre os mesmos tipos de dados. Assim, deve-se possibilitar o uso de termos nebulosos como valores de consultas, ou ainda, termos nebulosos como operadores destas consultas.

É necessário permitir também que por meio de funções específicas, os dados armazenados possam ser comparados entre si e as respostas obtidas possam ser classificadas através de algum método lógico de ordenação.

Por fim, o uso de limiares definidos subjetivamente pelo usuário deve ser permitido. Estes limiares devem atuar como forma de limitar o número de respostas obtidas, classificar estas respostas para ordenação ou flexibilizar a interpretação dos termos nebulosos utilizados ou existentes na estrutura.

5.3. Definição do Modelo Genérico para Bancos de Dados de Grafo

Seguindo os aspectos elencados anteriormente, definimos aqui os requisitos que satisfazem o modelo genérico de inserção, representação e consulta em um banco de dados de grafos.

Considerando a diversidade dos tipos de dados nebulosos e as suas definições variadas, propomos o armazenamento das definições de cada tipo de dado utilizando uma estrutura de dados separada do sistema. Esta abordagem permite definir os diferentes tipos de dados de forma independente do sistema de gerenciamento do banco de dados de grafos que será utilizado. Denominamos esta estrutura de dados como *"Documento de Definição de Dados Nebulosos"* ou *"DDDN"*. A Figura 5.1 apresenta o modelo de um *DDDN* geral, composto de elementos já definidos por um usuário. Em sua estrutura estão organizados o nível do grafo, a identificação do termo com os parâmetros relacionados e a interpretação que o sistema deve utilizar.

```
<?xml version="1.0" encoding="UTF-8"?>
<neo4j>
  <node>
  </node>
  <node_property>
    <idade type="INTEGER">
      <null_values>
        <dne/>
        <unk/>
        <ni/>
        <open/>
      </null_values>
      <linguistic_variable>
        <jovem type="TRAPEZOIDAL" a="10" b="20" c="30" d="40"/>
        <adulto type="TRAPEZOIDAL" a="18" b="20" c="40" d="55"/>
        <crianca type="DECREASING" a="0" b="0" c="10" d="15"/>
        <idoso type="INCREASING" a="50" b="70" c="0" d="0"/>
      </linguistic_variable>
      <interval>1</interval>
      <set>
        <criança/>
        <jovem/>
        <adulto/>
        <idoso/>
        <open/>
      </set>
      <fuzzy_expression>
        <pelo_menos scope="idade, max">
          case when tofloat(idade) > 0 then idade/max when a=~'>>:fz0.*' then 1 else 0 end as fz0
        </pelo_menos>
        <aproximadamente scope="idade, max">
          case when tofloat(idade) > (max - 5) and tofloat(idade) > (max + 5) then 1 when a=~'>>:fz0.*' then 1 else 0 end as fz0
        </aproximadamente>
      </fuzzy_expression>
    </idade>
  </node_property>
  <edge>
  </edge>
  <edge_property>
    <amigo>
      <fuzzy_expression>
        <bons_amigos scope="a, x">
          case when tofloat(a) > x then a when a=~'>>:fz0.*' then 1 else 0 end as fz0
        </bons_amigos>
      </fuzzy_expression>
    </amigo>
  </edge_property>
</neo4j>
```

Figura 5.1. Modelo de um *DDDN* geral.

Fonte: Elaborado pelo autor.

Com base na apresentação realizada no segundo capítulo desta dissertação, idealizamos a possibilidade da inserção dos seguintes tipos de dados nebulosos:

Valores nulos: Utilizaremos em nossa proposta os valores nulos para definir a ausência de um valor exato no momento da inserção. Para esta abordagem, seguiremos a classificação apresentada em Zicari (1990) para os tipos de valores nulos. Este tipo de dado imperfeito deve ser inserido como um dado comum no banco de dados do grafo, seguindo as definições para cada tipo de valor nulo. Não discorremos sobre o uso do referido tipo de dado como operandos das instruções de consultas. Contudo, utilizamos um algoritmo que se baseia no *DDDN* proposto para interpretar a ocorrência deste tipo de dado, caso se apresentem como resultado de uma consulta. Entre os valores definidos para este tipo de dados, temos:

- *Valores desconhecidos* — *unk;*
- *Valores inexistentes* — *dne;*
- *Valores indefinidos* — *open;*
- *Valores sem informação* — *ni.*

Intervalos: Considerando que intervalos definem o valor mínimo e máximo de possibilidade para um elemento, nossa proposta busca definir a inserção, representação e consulta deste tipo de dado. A inserção e representação será feita de forma direta, desta maneira nenhuma ação adicional será executada. Para as consultas a abordagem a este tipo de dados terá diferentes interpretações, considerando o uso e a ocorrência deste dado. Para exemplificar, vamos supor as seguintes situações:

1) *Insira o valor mínimo e máximo para a propriedade idade do vértice A, como 20 e 30, respectivamente.*
2) *O valor da propriedade do vértice B é de 10 a 50.*
3) *Selecione os vértices cujo valor da propriedade idade esteja entre 0 e 100.*
4) *Selecione os vértices cujo valor da propriedade idade seja 25.*

Analisando as situações, as situações (1) e (2) fazem referência a inserção e representação de intervalos no modelo. Consequentemente, serão inseridos os dados do atributo idade para os *vértices* A e B de $[20 - 30]$ e $[10 - 50]$, respectivamente. Nas situações (3) e (4) temos referências à consultas no modelo. Deste modo, em (3) serão selecionados os vértices possuindo os valores do atributo idade entre 0 e 100, ou conjuntos de dados neste intervalo. Da mesma forma, a execução de uma consulta

em (4) deverá retornar os vértices contendo os valores exatos das propriedades, como também os vértices que contenham o valor de sua propriedade definido como um intervalo. Essa definição será obtida através do *DDDN*.

Valores de composição: Utilizamos este tipo de dado nebuloso para definir, neste caso, funções que determinam a ligação entre vértices. Seguindo os conceitos da relação nebulosa, buscamos possibilitar que uma função seja definida quando há a impossibilidade de definir o valor exato, ou ainda, que este valor seja flexível mediante critérios estabelecidos no momento da consulta.

Suponhamos que o *vértices A* e o *vértices B* estão relacionados pela função de amizade, assim sendo: $(A) - [: AMIGO] - > (B)$. Suponhamos também, que esta relação se baseia pela propriedade idade existente em ambos os vértices e que hipoteticamente, quanto mais próximos ou iguais os valores destas propriedades nos vértices são, mais relação estes vértices possuem. Se o valor da propriedade idade de (A) é igual a 20 e a propriedade idade de (B) é igual a *adulto*. Considerando que a relação *Amigo* define a intensidade da ligação dos vértices e que esta intensidade é determinada pela análise do atributo idade. Podemos determinar então qual a intensidade da relação destes vértices, levando em consideração os diferentes tipos de valores dos atributos? A relação nebulosa, neste caso, analisa as definições utilizadas no *DDDN* para determinar a intensidade desta ligação no momento da consulta. Esta abordagem permite modificar o valor alterando os parâmetros utilizados.

Variáveis linguísticas: Nossa proposta utiliza os diferentes tipos de termos linguísticos como atributos para informação imperfeita. Adicionalmente, possibilitamos que estes termos sejam utilizados como operandos nas instruções de consultas. Assim, a inserção deverá ser feita de forma direta, bem como a sua representação no modelo. Em consultas em que estes dados sejam retornados, utilizamos um algoritmo para definir o valor da pertinência para tornar possível a comparação com outros valores. Quando utilizados como operandos, estes são utilizados como referência a um conjunto de valores.

Através do *DDDN* definimos o modo que as variáveis linguísticas serão interpretadas. Utilizamos a classificação de Takahashi (1991) e Chen & Jong (1997)

para definir os tipos de dados aceitos, entre eles estão:

- **Termos simples -** *jovem, idoso, alto, baixo;*
- **Termos compostos -** *muito jovem, mais ou menos baixo;*
- **Termos de composição -** *jovem e alto, não jovem e não baixo;*

Para exemplificar o uso dos termos nebulosos, vamos supor as seguintes consultas:

5) *Encontre os vértices cujo valor para o atributo idade seja considerado jovem.*
6) *Encontre os vértices onde valor do atributo idade seja mais ou menos jovem.*
7) *Encontre os vértices cujo valor para o atributo idade seja igual a 20 anos.*

Se em nossos registros existem valores exatos de idade como também valores nebulosos, como estes dados podem ser relacionados? Utilizaremos então o *DDDN* como suporte ao algoritmo de consulta.

Vamos considerar que o termo jovem é compatível com as idades de 0 a 20. Para as idades de 20 a 60 possui certa compatibilidade e também são incompatíveis acima destes. Podemos então obter o valor de pertinência através das expressões definidas em Takahashi (1991) e Chen & Jong (1997):

$$\mu_{jovem}(u) = \left\{ \begin{array}{ll} 1, & para\ \mu \leq 20 \\ \left(1 + \left(\frac{(u-20)}{15}\right)^2\right)^{-1}, & para\ 20 \leq u \leq 60 \\ 0, & para\ \mu \geq 60 \end{array} \right\}$$

$$\mu_{mais\ ou\ menos\ jovem}(u) = (\mu_{jovem}(u))^{\frac{1}{2}}$$

Deste modo, os vértices retornados contendo valores exatos para o atributo idade de (5) e (6), serão mensurados pela expressão definida, a fim de avaliar a compatibilidade com o termo *jovem* e *mais ou menos jovem*. A expressão define também os valores mínimo e máximo para o termo *jovem*, tornando possível comparar o valor exato da consulta (7) com qualquer termo nebuloso inserido como atributo.

Expressões nebulosas: As expressões nebulosas definidas aqui possuem semelhança com as definições utilizadas para as variáveis linguísticas. Em nossa proposta, utilizamos estas expressões como valores dos atributos dos vértices e arestas, bem como como operadores na execução de consultas. Entre os tipos aceitos estão:

•**Aproximação -** *Aproximadamente X;*

•**Valor mínimo -** *No mínimo X ou Pelo menos X.*

•**Valor máximo -** *No máximo X.*

Os exemplos a seguir determinam a forma do uso deste tipo de dado em duas situações:

8) *Para o vértice A, o valor do atributo idade é de 20 anos e para o vértice B, o valor do atributo idade é de aproximadamente 30 anos.*
9) *Selecione os vértices onde o atributo idade seja de aproximadamente 20 anos.*
10) *Selecione os vértices onde o atributo idade seja no mínimo de 25 anos.*

No exemplo (8) temos uma situação de inserção e representação do valor de um atributo contendo a expressão nebulosa. Buscamos utilizar o algoritmo de inserção juntamente com as definições do *DDDN* para este tipo de dado. Desta forma, podemos obter os valores aproximados obtidos pela expressão e inseri-los diretamente no modelo. Isto torna possível identificar visualmente a relação deste tipo de dado. O exemplo (9) e (10) as expressões nebulosas são utilizadas como operadores para os dados exatos definidos. Isto faz com que mais resultados possam ser obtidos, diferente da abordagem do modelo booleano. Tradicionalmente poderíamos somente definir limites para um valor, como *"igual a"* ou *"entre um e outro"*. Por meio de uma expressão nebulosa é possível realizar a consulta e obter valores relacionados associados ao que se deseja, mesmo não conseguindo expressar um valor exato.

Limiares de consultas: Propomos aqui o uso de limiares para execução de consultas. Esta abordagem atribui mais flexibilidade na obtenção de resultados, além de possibilitar o uso de termos nebulosos mais adequadamente. Como definido anteriormente, a estrutura de uma consulta básica em grafos é composta de três níveis, sendo estes o padrão de caminho, o predicado e o resultado esperado. Nossa proposta busca possibilitar o uso dos limiares em todos estes níveis.

A aplicação de limiares possui diferentes interpretações, dependendo do nível em que se apresenta. Assim, temos: (i) limiar no predicado indica que o limite aceito se baseia nas propriedades dos vértices ou das arestas; (ii) limiar no padrão de caminho indica que o limite aceito será baseado na estrutura do grafo, através

da intensidade das relações definidas pelas arestas ou pelo tamanho do caminho; e (*iii*) limiar no resultado esperado indica que a aceitação será feita após a execução, se baseando em todos os critérios definidos ou na combinação da estrutura com as propriedades.

Para exemplificar, vamos supor as seguintes situações:

11) *Selecione os vértices cujo valor da propriedade idade seja jovem com limiar 0,6.*
12) *Selecione a relação mais forte entre os vértices A e C com limiar 0,8.*
13) *Selecione o caminho mais curto entre os vértices A e M, com propriedade idade igual a adulto e limiar 0,7.*

No exemplo (12) avalia-se através do predicado as propriedades dos vértices. O uso do termo linguístico como operando e o limiar especificado limita a quantidade de elementos aceitos para o termo. Em (13) avalia-se todas as possibilidades do padrão de caminho obtido. Com base no valor obtido da aresta, a comparação dos valores de cada caminho entre os dois vértices definidos determina a sua força. A análise de força de um caminho foi definido anteriormente no capítulo 4.1 sobre grafos. Se o valor mínimo final deste caminho for de pelo menos $0,8$, o caminho é aceito. Por fim, em (13) temos a junção destas propriedades. O valor do limiar $0,7$ especificado, avalia tanto as propriedades dos vértices quanto o caminho obtido.

Em nossa proposta, o limiar aplicado no resultado em (*iii*) é definido pelo menor valor. Todavia, outras formas de avaliação podem ser definidas no *DDDN*.

Análise dos resultados: Nossa proposta utiliza o método de *"rank"* para análise dos resultados. Este método possibilita determinar as respostas mais satisfatórias, baseando-se pelo critério utilizado na consulta. Assim, quanto maior a compatibilidade com os critérios, mais alta é a sua posição no *rank*. Esta abordagem possibilita a organização dos resultados obtidos.

5.4. Proposta de Integração com a Linguagem *Cypher* do *Neo4j*

No modelo proposto, utilizamos os conceitos genéricos de lógica nebulosa, para descrever as propriedades nebulosas em um banco de dados de grafos. A seguir, são

apresentados os comandos da sintaxe da linguagem *Cypher* do sistema Neo4J que permitam definir um modelo de implementação. Deste modo, dados perfeitos e imperfeitos poderão ser inseridos e consultados igualmente. Nossa proposta se baseia na definição das instruções elencadas a seguir e relacionadas aos exemplos utilizados anteriormente no modelo genérico.

5.4.1. Documento de Definição dos Dados Nebulosos - *DDDN*

Os parâmetros definidos serão armazenados em um arquivo *XML*. A escolha do arquivo XML se deve pela sua integração com diferentes sistemas. Deste modo, independentemente do sistema base, a leitura dos dados não será comprometida. Como linguagem de base, estamos seguindo alguns conceitos da sintaxe da linguagem *FUDGE* de Pivert *et al.* (2014), apresentada anteriormente. Apresentamos aqui alguns exemplos de possíveis instruções que estão integradas em nosso projeto no sistema *Neo4j* pela linguagem *Cypher*. Utilizamos como tipo de dado uma variável linguística identificada pelo nome idade. Para tanto, temos:

$$DEFINE\ NODE\_PROPERTY\ linguistic\_variableidade\ AS\ jovem = \{10, 20, 30, 40\}$$
$$DEFINE\ EDGE\_PROPERTY\ linguistic\_variabletempo\ AS\ recente = \{2015, 2018\}$$

A execução destas instruções, incorporadas na linguagem *Cypher*, define os parâmetros demonstrados pelo trecho da Figura 5.2. Vale ressaltar que a definição de termos é subjetiva e não é atualizada automaticamente. Sendo assim, conforme o exemplo, a interpretação do termo *"recente"* deverá ser atualizada no futuro, se adequando a persepção do tempo no momento.

```
<?xml version="1.0" encoding="UTF-8"?>

<neo4j>
  <node_property>
    <idade type="INTEGER">
      <linguistic_variable>
        <jovem type="TRAPEZOIDAL" a="10" b="20" c="30" d="40"/>
        <crianca type="DECREASING" a="0" b="0" c="10" d="15"/>
        <velho type="INCREASING" a="50" b="65" c="0" d="0"/>
        <adulto type="TRAPEZOIDAL" a="15" b="20" c="40" d="55"/>
      </linguistic_variable>
    </idade>
  </node_property>
  <edge_property>
    <tempo>
      <recente type="INCREASING" a="2015" b="2018" c="0" d="0" />
    </tempo>
  </edge_property>
</neo4j>
```

Figura 5.2. Trecho do *Documento de Definição de Dados Nebulosos (DDDN)*.

Fonte: Elaborado pelo autor.

5.4.2. Inserção de Dados Nebulosos

Para realizar a inserção de dados a linguagem *Cypher* utilizará o algoritmo modificado para verificar a ocorrência de dados nebulosos. Caso existam, será selecionada a melhor interpretação conforme determinado no documento de definição de dados nebulosos. Desta forma, as instruções a seguir poderão ser utilizadas:

1. $CREATE\ (n : Pessoa\ \{nome : \text{“João”},\ idade\ IS\ jovem\})$
2. $CREATE\ (n : Pessoa\ \{nome : \text{“João”},\ idade : 18\})$

No exemplo, as instruções devem ser interpretadas de forma equivalente. A primeira instrução demonstra um exemplo da sintaxe para inserção de um dado nebuloso, já na segunda, representa a inserção de um dado exato, como é feito atualmente no modelo tradicional. Com a inserção do dado imperfeito é adicionado um *símbolo-chave* indicando que este é um dado imperfeito. Neste caso, o *símbolo-chave* $>>>$ indica ser um dado imperfeito do tipo variável linguística. Para cada tipo de dado imperfeito foi definido um símbolo único que o identifica para o sistema. Isto influencia na forma de manipulação posteriormente quando se deseja sua interpretação.

5.4.3. Consulta de Dados Nebulosos

Como requisito mencionado anteriormente, a consulta dos dados deve apresentar resultados exatos e nebulosos de forma equivalente. Seguindo os exemplos das instruções a seguir, estas abordagens são representadas por:

1. $MATCH\ (n : Pessoa) - [r : AMIGO] - > (m)\ WHERE\ n.idade = 18\ RETURN\ n, r, m$
2. $MATCH\ (n : Pessoa) - [r : AMIGO] - > (m)\ WHERE\ n.idade\ IS\ 18\ RETURN\ n, r, m$
3. $MATCH\ (n : Pessoa) - [r : AMIGO] - > (m)\ WHERE\ n.idade\ IS\ jovem\ RETURN\ n, r, m$
4. $MATCH\ (n : Pessoa) - [r : AMIGO] - > (m) WHERE\ \#APROX(n.idade; 18)\ RETURN\ n, r, m$

Na primeira instrução temos um exemplo de uma instrução comum, com parâmetros exatos. Em seguida, se apresenta uma instrução semelhante, com a variação do operador, de = para *IS*. A terceira instrução contém um operando nebuloso do tipo variável linguística *jovem*. Por último, é feito o uso de uma expressão nebulosa, referenciando o mesmo atributo dos anteriores e valor de operando equivalente.

Em todas as respostas das instruções propostas devem ser apresentado um conjunto de respostas semelhantes. Todavia, somente a primeira retornará valores exatos iguais ao requisito informado, não aceitando valores nebulosos. A diferença entre a primeira e a segunda instrução está na forma do operador. No uso do operador da segunda instrução, é indicado à aplicação que seus resultados não estão restritos somente a valores exatos, mas que serão aceitos também valores nebulosos.

Em instruções com operandos exatos, a princípio deve ser obtidos os resultados exatos. Posteriormente, deve-se utilizar o *DDDN* para obter as respostas possíveis relacionadas com o valor utilizado. Na execução de consultas utilizando os dados nebulosos a conversão é feita no início, com base nas definições do *DDDN*.

5.4.4. Uso de Limiares nas Consultas

A finalidade aqui é possibilitar o retorno de mais respostas do que o modelo booleano tradicional. O uso de limiares nas consultas tem por objetivo indicar a aplicação um *limite de valores aceitáveis*. Deste modo, conforme o exemplo a seguir de uma instrução com a aplicação do limiar, resultados obtidos que possuam limite inferior a $0,6$ serão descartados. O grau de pertinência é obtido pela comparação do operando *jovem* com os resultados obtidos na seleção do padrão de caminho solicitado.

$MATCH\ (n : Pessoa) - [r : AMIGO] - > (m)$
WHERE n.idade IS jovem WITH THRESHOLD ≥ 0.6 *RETURN n, r, m*

5.4.5. Análise dos Resultados

A representação dos resultados de uma consulta deve permitir a classificação pelo grau de satisfação destas respostas. Isso possibilita obter respostas possíveis e não somente aquelas que atendam perfeitamente aos requisitos. Para exemplificar, temos a instrução de consulta *Cypher*, com foco na apresentação dos resultados.

$MATCH\ (n : Pessoa) - [r : AMIGO] - > (m)$
WHERE n.idade IS 18 AND m.idade IS 20 RETURN n, r AS Amizade, m

Utilizando a instrução de consulta proposta, necessita-se classificar as relações pelo nível de amizade. A classificação é feita por critérios estabelecidos na

instrução, no caso, *n.idade IS 18 AND m.idade IS 20*, além da intensidade declarada na aresta, no caso, *r:AMIGO*. Quanto mais compatível os resultados são dos parâmetros declarados, neste caso *18*, *20* e o valor da aresta, mais alto no *rank* o resultado será apresentado. O *rank* é gerado pela análise da compatibilidade dos resultados de modo que entenda qual a real necessidade do usuário.

5.5. Proposta de Desenvolvimento da Aplicação

O modelo genérico proposto, se baseia nos conceitos apresentados, buscando a incorporação dos tipos de dados imperfeitos. Para aplicar os conceitos estabelecidos, faremos uso do *Cypher*, já que esta é considerada a principal linguagem do modelo de bancos de dados de grafos. Deste modo, para validar a ideia deste trabalho, integrando o modelo genérico com a linguagem *Cypher*, é proposto o desenvolvimento de uma aplicação, desenvolvida nos moldes da linguagem *Java*, que possa ser futuramente integrada ao sistema oficial *Neo4j*.

Conforme demonstrado na Figura 5.4, para validarmos a integração do nosso modelo nebuloso com o *Neo4j Cypher*, utilizaremos o algoritmo modificado pela aplicação para manipular os dados. Em vista disto, será disponibilizado na camada superior perante ao usuário, de modo a converter as instruções para a linguagem *Cypher* executar nas camadas inferiores.

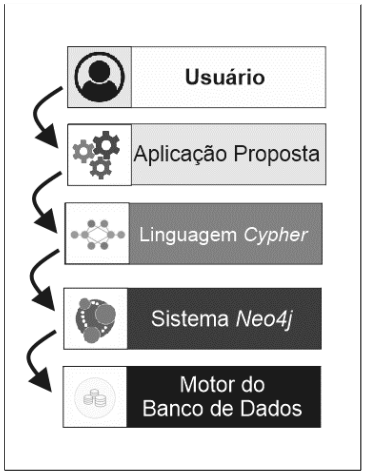


Figura 5.3. Arquitetura idealizada do modelo de aplicação.

Fonte: Elaborado pelo autor.

Por fim, desenvolvemos um fluxograma com as sequências de processos que

nossa aplicação executa. Conforme Figura 5.4, uma instrução de consulta utilizando dados nebulosos será analisada e convertida, tornando-a compreensível ao sistema e apresentando os resultados requisitados pelo usuário. Deste modo, conforme o exemplo, ao receber uma instrução qualquer, a aplicação analisa se esta contém algum tipo de dados nebulosos. Realiza a comparação com os dados definidos no *DDDN* tanto para valores referentes aos vértices quanto para arestas. Caso sejam compatíveis é feita a conversão em uma sintaxe compreensível pelo sistema. Adicionalmente, após a seleção dos dados, se forem definidos limiares de aceitação é executada a análise dos valores retornados, descartando aqueles que se encontrarem fora dos limites estabelecidos. Por fim os valores aceitos são ordenados pela proximidade com o grau máximo de aceitação, neste caso o *grau 1*.

O primeiro estágio serve para analisar a sintaxe da instrução verificando se existem dados nebulosos. Isto porque, caso o usuário não faça uso de dados nebulosos, a aplicação ignora a instrução, fazendo com que as demais funções já existentes no sistema não sejam afetadas. Conforme mencionado, se um dado nebuloso é declarado como parâmetro condicional de um vértice, por exemplo *"é jovem?"*, a aplicação tenta localizar se existe uma referência no *DDDN*. Ao ser encontrado, vamos supor que exista a definição *"jovem é maior que 18 e menor que 30"*, este dado é convertido com base nos valores armazenados no *DDDN*. Após sua conversão, os valores são retornados para a instrução, substituindo aqueles declarados inicialmente. Uma função similar é utilizada para verificar as arestas, se declarada na instrução. Por fim, é verificado se foi solicitado alguma classificação nebulosa, por exemplo o uso de um limiar de aceitação. Se existe tal definição, cada valor obtido como resultado da consulta é mensurado, comparando o que foi declarado na instrução com o resultado obtido. Deste modo, determina-se o *"grau de satisfação"*, que representa a compatibilidade da instrução de consulta com o resultado. Para cada resposta obtida é atribuído um coeficiênte que representa este *grau de satisfação*. Isto torna possível que o usuário que solicitou a execução da instrução compreenda a relação existente entre os dados apresentados. Vale ressaltar que em dados exatos o *grau de satisfação* é representado pelo valor *1* quando compatível e *0* quando incompatível.

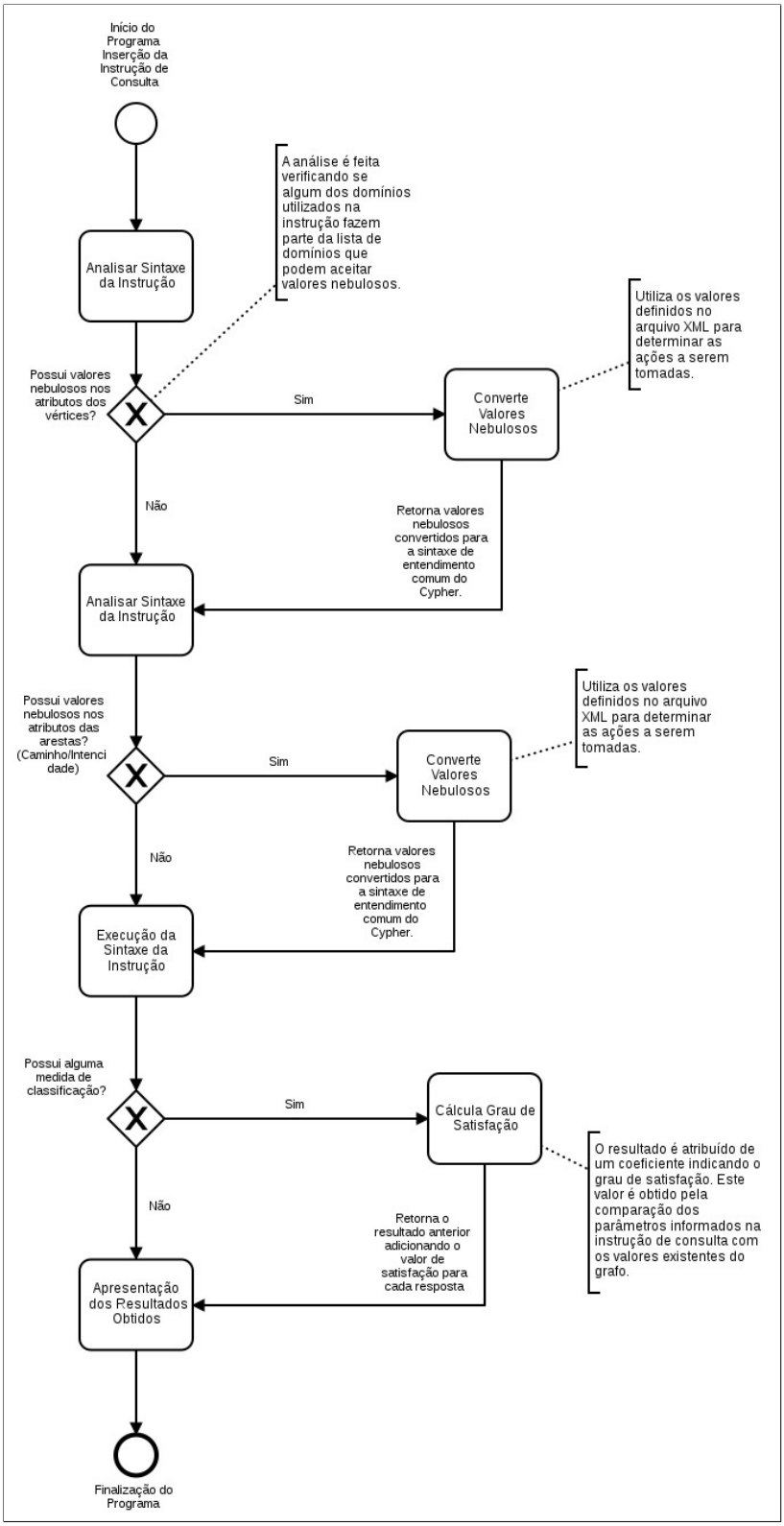


Figura 5.4. Fluxograma do processamento de uma instrução de consulta pela aplicação proposta.

Fonte: Elaborado pelo autor.

5.6. Considerações Finais

Com base nos requisitos levantados na análise dos modelos existentes, foi possível determinar os principais pontos que a aplicação deve contemplar. Apesar de existirem diversas formas de representar dados nebulosos, buscou-se um método que abordasse grande partes dos tipos existentes, sem comprometer o sistema atual do *Neo4j*. Além disso, buscou-se desenvolver uma aplicação maleável, que pudesse interagir com os usuários, sem que limitasse a um padrão previamente definido. Isto possibilita também que a aplicação possa ser aprimorada futuramente, abordando novos paradigmas ainda não explorados.

Com base no modelo genérico apresentado, é possível não só a integração com o sistema *Neo4j*, como também em outros sistemas gerenciadores de bancos de dados de grafos existentes, necessitando somente algumas modificações.

O modelo de inserção proposto apresenta uma nova visão para os sistemas de bancos de dados de grafos, uma vez que formas de inserção de dados nebulosos não são apresentados em trabalhos recentes. Além do mais, foi possível apresentar o uso de limiares de consulta, o que possibilita determinar resultados mais precisos em dados nebulosos. Por fim, a forma de classificar os resultados para o usuário possibilita que estes possam compreender a relação entre dados nebulosos e exatos, tanto aqueles utilizados na instrução quanto nos obtidos como resultados.

CAPÍTULO 6

Resultados e Validações

Como forma de validar a proposta deste trabalho, será apresentada uma aplicação integrada à extensão da linguagem *Cypher*. Esta aplicação é baseada nos conceitos do sistema oficial *Neo4j*. As etapas de desenvolvimento serão apresentadas a seguir, bem como os resultados obtidos. Ao final será feito a análise de validação por meio de um caso de uso proposto.

6.1. Desenvolvimento da Aplicação Proposta

A aplicação proposta foi desenvolvida com base em três principais abordagens. Cada uma destas abordagens atua em um segmento do processo e juntas apresentam o resultado esperado. São elas:

- *Desenvolvimento de uma estrutura de interpretação de dados imperfeitos;*
- *Desenvolvimento de um módulo tradutor de instruções;*
- *Desenvolvimento de um módulo interpretador de resultados.*

O desenvolvimento da aplicação foi feita por meio do *console* de desenvolvimento comunitário denominado *Rabbit Hole*[1]. Está aplicação utiliza a linguagem *Cypher* e permite a integração de conceitos do modelo da aplicação oficial do *Neo4j*. A escolha desta aplicação e não do sistema oficial se deve por ser possível o desenvolvimento de novos recursos internamente, o que não é permitido no sistema oficial disponibilizado. Deste modo, podemos modificar a estrutura de algumas funções de modo a adaptar para as funcionalidades propostas neste trabalho.

[1] *https://github.com/neo4j-contrib/rabbithole*

6.1.1. Configuração do Ambiente

A base de desenvolvimento da aplicação proposta foi idealizada a partir do modelo apresentado pelo projeto *SUGAR* de Pivert *et al.* (2016). Seguindo seus conceitos, definimos as novas funcionalidades da aplicação proposta.

O ambiente utilizado para desenvolvimento segue as orientações oficiais indicadas pela comunidade de desenvolvedores do *Rabbit Hole*. Desta forma, foi utilizado a plataforma de desenvolvimento *Java*[2] *Netbeans*[3] em conjunto com o suporte do sistema *Apache Maven*[4]. Neste ambiente é possível realizar as implementações e testes conforme necessidade.

Pode-se observar na Figura 6.1 a tela inicial da aplicação base, gerada pelo sistema *Rabbit Hole*. A aplicação já oferece inicialmente os recursos visuais necessários para construção dos dados e relacionamentos utilizados em um bancos de dados de grafos. Além do mais, possui todo o suporte das instruções da linguagem *Cypher* do *Neo4j*.

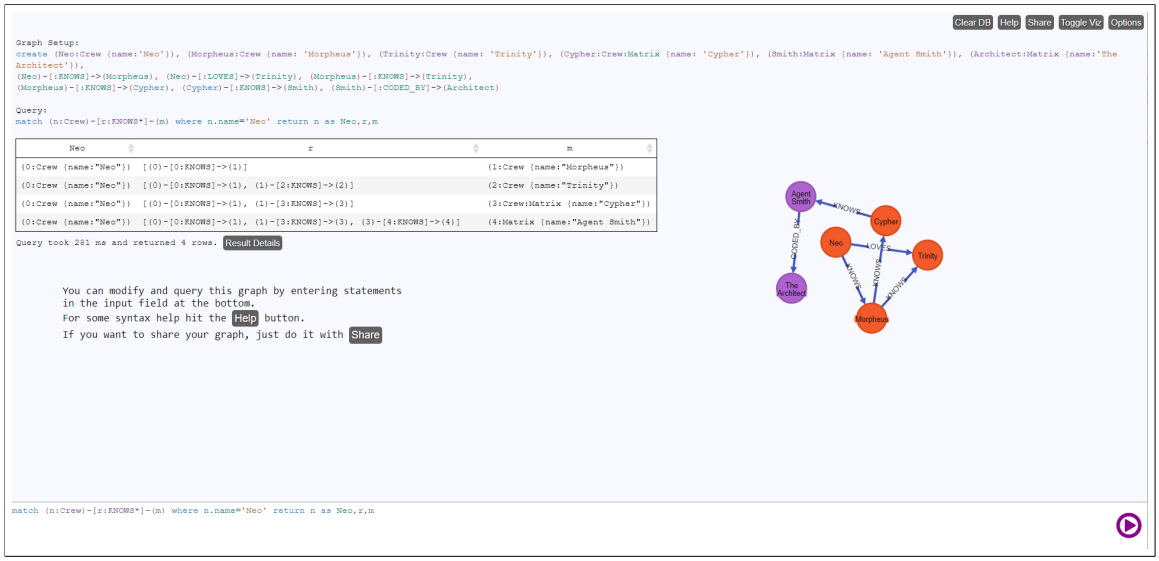


Figura 6.1. *Neo4j Console.*

Fonte: Elaborado pelo autor.

[2] *https://www.java.com*
[3] *https://netbeans.org*
[4] *https://maven.apache.org*

6.2. Etapas de Desenvolvimento

A aplicação foi dividida em etapas para que fosse possível avaliar o desenvolvimento do projeto. Cada etapa foi desenvolvida, testada e certificada para atender os requisitos propostos. Conforme o modelo conceitual da Figura 6.2, a aplicação foi dividida em dois módulos, de modo que fosse possível o desenvolvimento de forma independente. Isto faz com que um módulo não dependa do outro e sim que possa complementar aqueles já existentes.

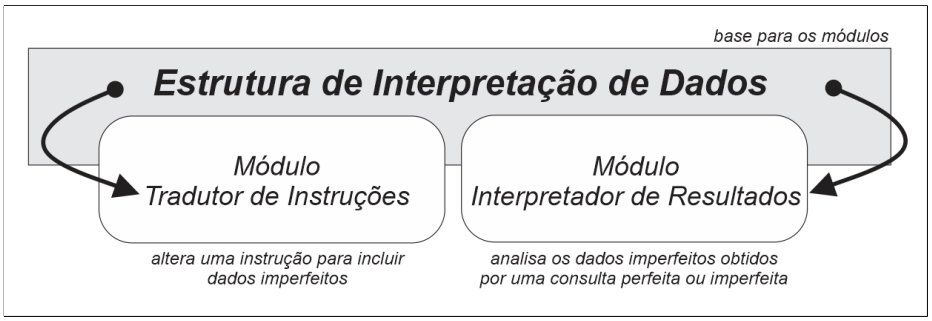


Figura 6.2. Ilustração da integração dos módulos da aplicação.

Fonte: Elaborado pelo autor.

6.2.1. Estrutura de Interpretação de Dados Imperfeitos

A estrutura de interpretação de dados imperfeitos foi idealizada para auxiliar o gerenciador do banco de dados de grafos a interpretar os dados imperfeitos utilizados ou obtidos. Independentemente da posição em que um dado imperfeito ocorra, este deve ser interpretado corretamente, oferecendo um conjunto de interpretações possíveis.

Ao serem consideradas as variadas interpretações que um dado imperfeito pode possuir, variando de indivíduo para indivíduo, houve a necessidade de ser incorporada esta variação à estrutura proposta. Deste modo, possibilita-se que o usuário defina de forma subjetiva cada dado imperfeito a seu entendimento.

Seguindo a sintaxe original do *Cypher* e as definições do *SUGAR*, foi desenvolvido uma variação para sua sintaxe de consulta. Fazendo assim com que as definições dos dados declaradas pelos usuários pudessem ser armazenadas em uma estrutura a parte. Desta maneira, incorpora-se o dinamismo para a base da interpretação para cada dado imperfeito e sua posição no grafo.

As definições dos dados imperfeitos são armazenados em uma estrutura de dados no formato de um documento *XML* pelas vantagens do modelo. Este possui compatibilidade com diferentes sistemas e é passível de leitura por qualquer usuário. Por meio da biblioteca *Dom4j*[5] *Java*, pode-se manipular a estrutura que contém os dados de forma satisfatória. Salienta-se que uma estrutura de dados baseada em outro formato de arquivo de dados também seria possível. Denominamos esta estrutura de dados como *Documento de Definição de Dados Nebulosos (DDDN)*.

Cada tipo de imperfeição encontrado em um dado possui uma forma de interpretação diferente. Logo, não é possível, até o momento, a definição de um tipo genérico de dado imperfeito. Assim, foi desenvolvida uma sintaxe de interpretação para cada tipo de dado. Conforme Figura 6.3, temos a relação dos tipos de dados com sua sintaxe de definição, que pode ser interpretada pelo sistema. Deste modo é possível o armazenamento no *DDDN*.

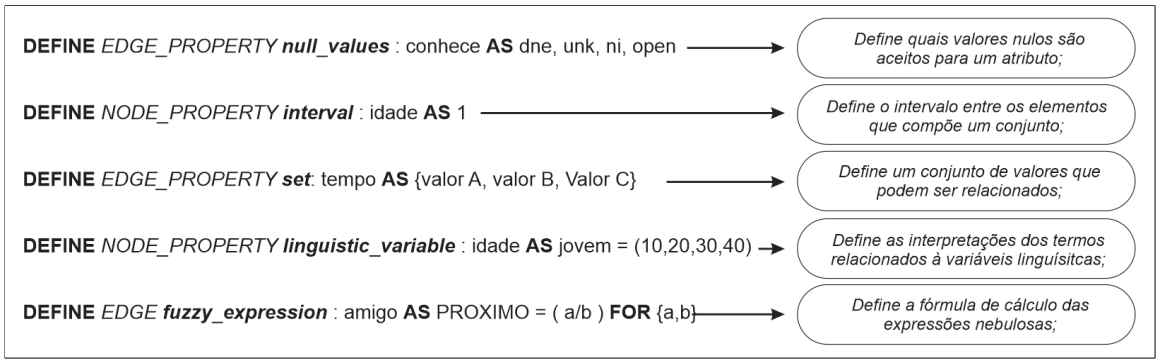


Figura 6.3. Sintaxe de definição dos tipos de dados imperfeitos.

Fonte: Elaborado pelo autor.

Como exemplo da formulação de uma interpretação para um dado imperfeito, faremos uso da sintaxe utilizada para um dado imperfeito do tipo variável linguística. Supondo-se então que será definida a interpretação do termo *"jovem"* da variável linguística *"idade"*, relacionado a uma das propriedades de um vértice. Conforme a Figura 6.4, temos a definição proposta utilizando a sintaxe desenvolvida. Pode-se observar que é utilizado o conceito de uma *função trapezoidal*. Deste modo, é passado o conjunto de valores relacionados à interpretação do termo imperfeito. Com isto é gerada a sua definição e a função pode ser utilizada para interpretar as ocorrências

[5] *https://dom4j.github.io/*

deste termo, sendo na execução de uma consulta ou na análise da comparação entre dados obtidos como resposta.

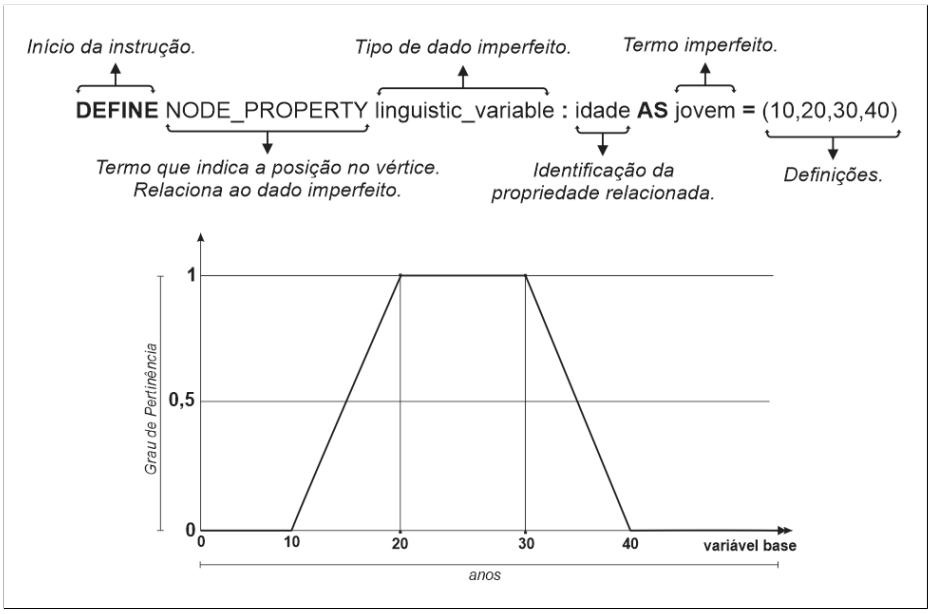


Figura 6.4. Instrução de definição de uma variável linguística.

Fonte: Elaborado pelo autor.

Um segundo exemplo consiste da formulação do tipo expressão nebulosa. Neste caso, é definido o retorno esperado desta expressão nebulosa. Assim, para exemplificar, será definida a função *"Próximo"* da aresta *"amigo"* existente entre dois vértices. Isto pode ser realizado por meio da instrução:

$$DEFINE\ EDGE\ fuzzy\_expression\ :\ amigo\ AS\ Proximo = (a/b)\ FOR\ \{a, b\}$$

Com esta definição, quando houver a ocorrência do termo *"próximo"*, esta poderá ser interpretada com base na declaração definida em (a/b) FOR $\{a, b\}$.

Para ambos os exemplos, o resultado será o armazenamento da interpretação na estrutura *DDDN*. Conforme demonstrado na Figura 6.5, apresenta-se o trecho do resultado obtido na execução das instruções de definição.

O algoritmo utilizado para a definição dos diferentes tipos de dados imperfeitos é apresentado na Figura 6.6. Observa-se o passo-a-passo para definição do dado imperfeito. Deve ser verificado se a definição do elemento já existe, pois assim evita-se a duplicidade das definições. Impede-se assim que diferentes interpretações

```xml
<?xml version="1.0" encoding="UTF-8"?>

<neo4j>
  <node_property>
    <idade type="INTEGER">
      <linguistic_variable>
        <jovem type="TRAPEZOIDAL" a="10" b="20" c="30" d="40"/>
      </linguistic_variable>
    </idade>
  </node_property>
  <edge>
    <amigo>
      <fuzzy_expression>
        <proximo scope="a,b">a/b</proximo>
      </fuzzy_expression>
    </amigo>
  </edge>
</neo4j>
```

Figura 6.5. Arquivo *DDDN* obtido a partir dos exemplos propostos.

Fonte: Elaborado pelo autor.

sejam utilizadas para um mesmo tipo de dado imperfeito. Do contrário, não poderia ser determinado com exatidão qual a real interpretação para este dado. A aplicação utilizaria sempre a primeira definição encontrada, ignorando as demais.

Algorithm 1 Definição de Dados Nebulosos (*DDDN.xml*)

Entrada: Instrução em Cypher
Saída: Elemento criado no arquivo *DDDN.xml*

Variáveis:
x ← `Instrução iniciada pela palavra 'DEFINE'`

1: **função** DEFINIÇÃO DADO IMPERFEITO(x)
2:
3: Verifica tipo da instrução x ▷ Identifica o tipo de dado imperfeito
4:
5: **se** não existe elemento da *"posição do grafo"* **então**
6: cria elemento para a *"posição do grafo"* ▷ ex.: propriedade do vértice
7: **fim se**
8:
9: **se** não existe elemento do *"nome do atributo"* **então**
10: cria elemento *"nome do atributo"* ▷ ex.: idade
11: **fim se**
12:
13: **se** não existe elemento do *"tipo imperfeito"* **então**
14: cria elemento *"tipo imperfeito"* ▷ ex.: variável linguistica
15: **fim se**
16:
17: **Grava** definições do dado imperfeito.
18:
19: **fim função**

Figura 6.6. Algoritmo para definição dos dados imperfeitos na estrutura do *DDDN*.

Fonte: Elaborado pelo autor.

6.2.2. Módulo Tradutor de Instruções

Com os dados imperfeitos definidos na estrutura de dados *DDDN*, o módulo tradutor de instruções passa a traduzir as instruções quando necessário. Isto porque ao encontrar um dado imperfeito na instrução de consulta, o módulo tradutor o converte, utilizando os parâmetros declarados pelo usuário, que estão armazenados no *DDDN*.

O desenvolvimento deste módulo faz uso das *expressões regulares*. Uma expressão regular utiliza padrões desenvolvidos para localizar as sequências de caracteres que sejam compatíveis com a sintaxe declarada. Deste modo, é possível localizar a ocorrência dos dados imperfeitos, já definidos pelo *DDDN* e converte-los em valores possíveis. Este método é necessário, pois do contrário seria necessário realizar alterações de baixo nível na estrutura do *Neo4j* e do *Cypher*.

Para ser interpretada corretamente uma instrução de consulta teve que ser dividida em partes. Analisando cada parte de forma individual é possível identificar se há a necessidade de modificações para que incorpore o conceito dos dados imperfeitos. Posteriormente, agrupa-se novamente os trechos desta instrução, para que assim a sua execução seja compreendida pelo *Cypher*.

O objetivo disto é incorporar o conceito de dados imperfeitos, de modo que o *Cypher* compreenda o que o usuário espera realmente realizar com a instrução, fazendo uso ou não de dados imperfeitos. Isto impacta diretamente na forma de interpretação, pois deve ser considerado que uma instrução de consulta possa conter dados imperfeitos, possa retornar dados imperfeitos ou ambas.

Neste contexto, temos ainda as variações nas formas das instruções de consultas do *Cypher*. Busca-se possibilitar que sejam executadas tanto consultas de seleção quanto consultas de inserção ou modificação. Deste modo, buscou-se considerar a relação destes tipos de instruções de consultas com o conceito dos dados imperfeitos. Pois um dado imperfeito utilizado em uma instrução de seleção possui diferente interpretação de um dado imperfeito utilizado em uma instrução de inserção.

Foram consideradas as principais palavras reservadas da linguagem *Cypher* para executar a separação de uma instrução. Assim, as palavras *"MATCH"*, *"CRE-*

ATE", *"WHERE"* e *"RETURN"*, separam as partes da instrução em análise. Conforme exemplo da Figura 6.7, é apresentado o algoritmo idealizado para uso do módulo tradutor de instruções. Com este, é possível converter uma instrução contendo ou não dados imperfeitos em uma instrução que possa ser compreendida pela interpretador da linguagem *Cypher* atual.

Algorithm 2 Módulo tradutor de instruções

Entrada: Instrução Nebulosa em Cypher
Saída: Instrução convertida para sintaxe *Cypher* original

Variáveis:
x ← `Instrução nebulosa em Cypher a ser traduzida`

1: **função** MÓDULO TRADUTOR(x)
2: n ← `Quebra a instrução em partes` ▷ *Vetor de elementos, separados pelos termos DEFINE, MATCH, WHERE ou RETURN.*
3: i ← `Cada elemento do vetor n` ▷ *Trecho da instrução a ser analisado, ex. MATCH (v1)-[]-(v2).*
4: nVetor ← `Número de elementos do vetor n`
5: t ← `Nova instrução` ▷ *Trecho da instrução convertido com base na interpretação feita pela aplicação.*
6:
7: **para** i ← Elemento do vetor **até** nVetor **faça**
8: **se** existe instrução *"DEFINE"* **então**
9: Interpreta instrução *"DEFINE"*
10: Remove a instrução
11: **fim se**
12:
13: **se** existe dados imperfeitos no vetor **então**
14: Interpreta a instrução ▷ *Utiliza os parâmetros declarados no DDDN.*
15: t ← `trecho da instrução com os novos parâmetros`
16: **fim se**
17: i ← `t` ▷ *Substitui trecho com a nova instrução, caso necessário.*
18:
19: **fim para**
20: **fim função**

Figura 6.7. Algoritmo do módulo tradutor de instruções.

Fonte: Elaborado pelo autor.

6.2.2.1. Instruções de Inserção

Em uma instrução de inserção, tem por objetivo inserir novos dados em uma estrutura. Originalmente, estes dados deveriam obrigatoriamente estar formatados em um padrão, para serem aceitos na estrutura. Em bancos de dados *NoSQL* está obrigação não existe, o que permite a inserção de quase qualquer tipo de dado.

Entretanto, a inserção de tipos variados de dados torna complexo para um sistema identificar qual a relação que existe entre estes dados. Por este motivo foi

desenvolvido o módulo tradutor de instruções para atuar junto a inserção de dados imperfeitos.

O módulo tradutor de instruções atua em uma instrução de inserção convertendo os dados imperfeitos utilizados em dados passíveis de interpretação. Deste modo, sem comprometer a semântica do dado informado, é adicionado um símbolo-chave indicando que o valor do dado é imperfeito e qual o seu tipo de dado. O módulo aplica diferentes símbolos-chaves para diferentes tipos de dados. A relação dos símbolos-chaves com os dados imperfeitos relacionados e seu tipo podem ser observados na Figura 6.8. Ao utilizar a sintaxe definida na área instrução de uso, o módulo aplica a tradução deste dado e insere o símbolo-chave corresponde ao tipo de dado imperfeito. Desta forma, é possível a sua identificação posteriormente quando necessário. Sem o uso deste método, seria ainda mais complexo para o sistema compreender a que este dado se refere. Isto porque um dado de valor *"jovem"*, por exemplo, pode ser considerado uma característica ou ainda, um dado imperfeito. Deste modo, indicamos ao sistema que a palavra *"jovem"* é semanticamente diferente do dado imperfeito *" jovem"*.

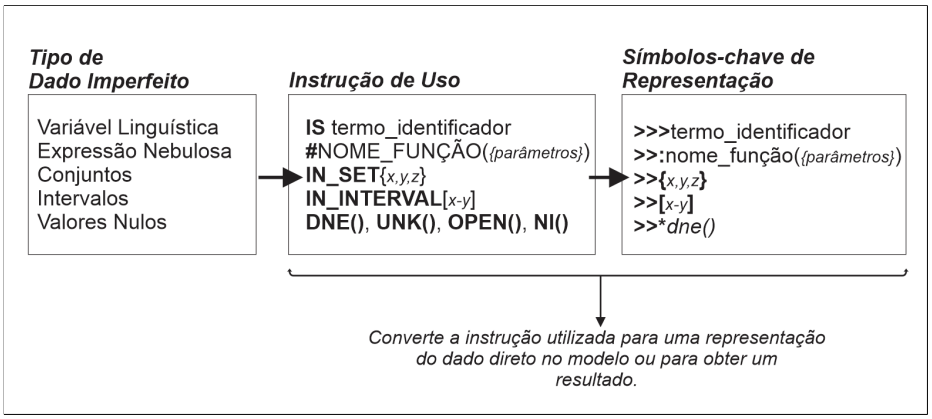


Figura 6.8. Símbolos-chave utilizados com a relação do tipo de dado imperfeito.

Fonte: Elaborado pelo autor.

Conforme exemplo da Figura 6.9, apresenta-se a tradução de uma instrução executada pelo módulo proposto. Isto faz com que o sistema possa identificar o uso de dados imperfeitos para uso posterior.

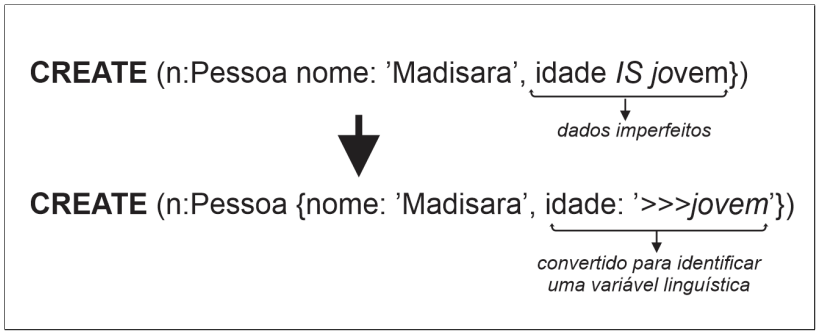


Figura 6.9. Execução do módulo tradutor de instruções.

Fonte: Elaborado pelo autor.

6.2.2.2. Instruções de Seleção

Neste tipo de instrução, o objetivo principal é selecionar os dados que atendam aos requisitos declarados. Fazendo uso de dados imperfeitos diretamente na instrução não se pode determinar com exatidão qual a intenção do usuário. Deste modo, o módulo tradutor de instruções modifica as instruções declaradas, com objetivo de delimitar os requisitos declarados, mantendo a correta relação com a intenção do usuário.

Para exemplificar o funcionamento do módulo de tradução de uma instrução de consulta contendo dados imperfeitos, serão utilizadas as instruções apresentadas pela Figura 6.10. Pode ser observado que estão identificadas as imperfeições no bloco do predicado, atribuído à propriedade dos vértices. Observa-se que no primeiro exemplo o operando é perfeito, no caso *"18"*, já no segundo, é utilizado um operando imperfeito, neste caso *"jovem"*. No terceiro exemplo, temos a atribuição do termo *WITH THRESHOLD 0.7*, indicando a definição do limiar de aceitação com valor de *0.7*.

Neste ponto o módulo tradutor deve entrar em ação. Efetuamos modificações na interpretação do *Cypher* para operandos e operadores. Neste caso, o termo *"IS"* associado a qualquer operando perfeito, indica ao módulo que esta é uma consulta perfeita, mas que permite que está consulta seja executada em dados imperfeitos e aceitos no seu retorno. O mesmo termo *"IS"* associado a um dado imperfeito definido no *DDDN* indica ao módulo que se trata de uma variável linguística.

O módulo tradutor possui diferentes tipos de identificadores, baseado no tipo de dado imperfeito que está sendo utilizado. Desta forma, é possível identificar o uso de variáveis linguísticas, expressões nebulosas, entre outras. Houve a necessidade de alterar os operadores e operandos das instruções, pois do contrário, qualquer instrução acionaria o módulo tradutor. Isto causou conflitos com as instruções já utilizadas pelo sistema, uma vez que o sistema não conseguiu compreender quando estavam sendo utilizadas instruções exatas ou as novas funcionalidades propostas. Assim, nos exemplos 4 e 5, as instruções serão executadas sem uso do módulo tradutor e sem comprometer a sintaxe original do *Cypher*.

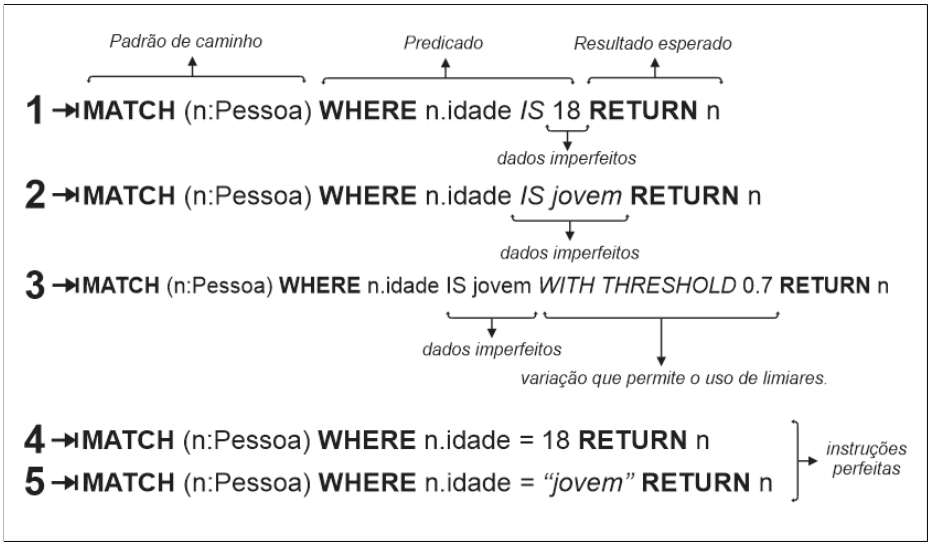


Figura 6.10. Exemplo de instruções *Cypher* com dados perfeitos e imperfeitos no predicado.

Fonte: Elaborado pelo autor.

A Figura 6.11 apresenta os resultados das traduções efetuadas para os exemplos de 1 a 3 apresentadas anteriormente. Nota-se que a sintaxe se tornou mais complexa, por este motivo estas traduções são executadas internamente no sistema, não sendo necessário que o usuário as desenvolva. A tradução dos dados imperfeitos é necessária para que a linguagem *Cypher* possa compreender o que o usuário está solicitando. Além disso, é incluída a expressão *">> .*"*, já existente no padrão do sistema, que em sua tradução literal, indica ao sistema *busque qualquer coisa que exista neste atributo*. Isto permite que dados de tipos diferentes possam ser obtidos, analisados e descartados ou classificados se compatíveis com o conjunto de resultados

esperados pelo usuário.

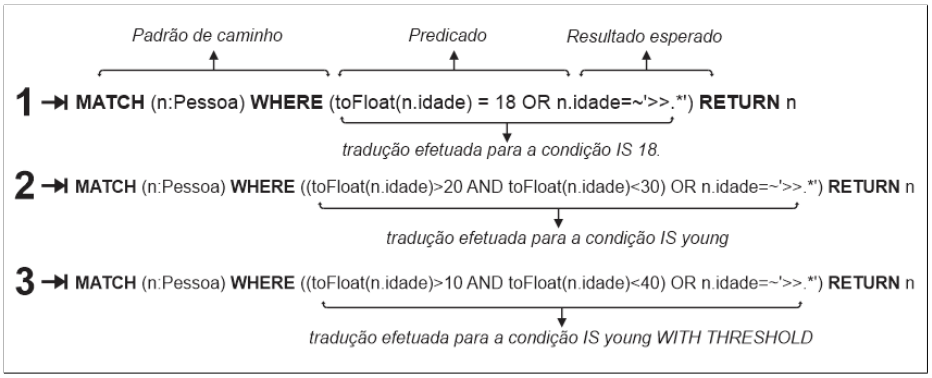


Figura 6.11. Conversão efetuada pelo módulo tradutor.

Fonte: Elaborado pelo autor.

6.2.2.3. Considerações Finais

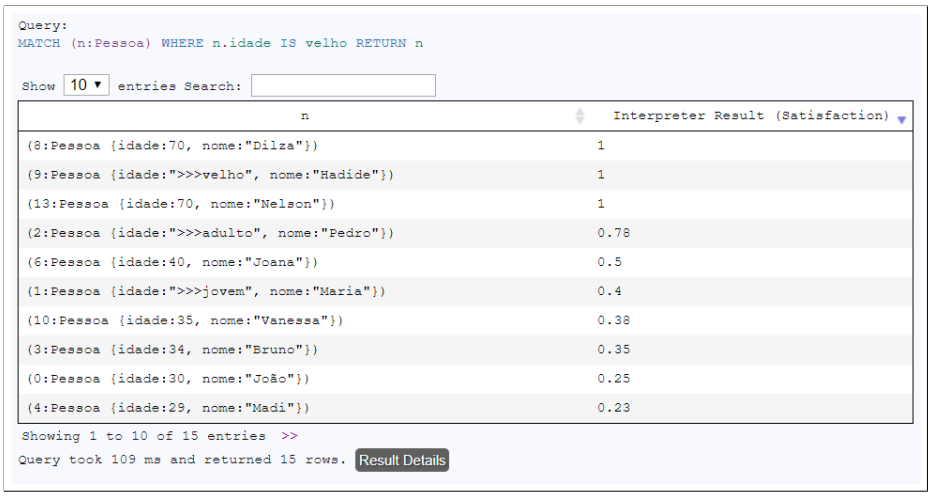
O motivo de realizar as traduções das instruções permite que seja possível obter um conjunto maior de respostas. Diferente do modelo booleano tradicional, este método possibilita a integração de diferentes tipos de dados.

Apesar destas traduções apresentarem sintaxes que poderiam ser inseridas diretamente na linguagem *Cypher*, dois pontos devem ser observados. O primeiro consiste da simplificação das instruções. Seria necessário um conhecimento maior da linguagem *Cypher* da parte de um usuário, para executar consultas complexas, utilizando diversos operadores. O segundo ponto consiste do uso do módulo interpretador. Sem os parâmetros utilizados pela sintaxe da proposta deste trabalho, a interpretação do conjunto de respostas obtidas poderia ser comprometida.

6.2.3. Módulo Interpretador de Resultados

Por meio da definição dos dados no *"DDDN"*, o módulo interpretador de resultados pode ser utilizado. Isto porque, com a definição dos dados, é possível realizar comparação dos dados utilizados na instrução de consulta com os dados obtidos como resultado. Este processo atribui um valor que representa a relação destes dados. Quanto mais próximo do valor 1, mais compatível é a resposta obtida com os parâmetros declarados.

O módulo possibilita a interpretação dos resultados obtidos, ainda que contenham dados imperfeitos. Além disto, se integra ao módulo de tradução, possibilitando ao usuário desenvolver uma instrução de consulta flexível e interpretada do mesmo modo. Os dados perfeitos ou imperfeitos declarados na instrução são comparados com os dados perfeitos e imperfeitos obtidos. Posteriormente é gerado o coeficiente pelo interpretador, definido como o grau de satisfação. Este coeficiente auxilia na compreensão da relação dos dados utilizados com as respostas obtidas. Conforme exemplo na Figura 6.12, apresenta-se uma instrução de consulta flexível, contendo dados imperfeitos, a relação de vértices obtidos como resposta e a atribuição do grau de satisfação.


Query:
MATCH (n:Pessoa) WHERE n.idade IS velho RETURN n

Show 10 entries Search:

n	Interpreter Result (Satisfaction)
(8:Pessoa {idade:70, nome:"Dilza"})	1
(9:Pessoa {idade:">>>velho", nome:"Hadide"})	1
(13:Pessoa {idade:70, nome:"Nelson"})	1
(2:Pessoa {idade:">>>adulto", nome:"Pedro"})	0.78
(6:Pessoa {idade:40, nome:"Joana"})	0.5
(1:Pessoa {idade:">>>jovem", nome:"Maria"})	0.4
(10:Pessoa {idade:35, nome:"Vanessa"})	0.38
(3:Pessoa {idade:34, nome:"Bruno"})	0.35
(0:Pessoa {idade:30, nome:"João"})	0.25
(4:Pessoa {idade:29, nome:"Madi"})	0.23

Showing 1 to 10 of 15 entries >>

Query took 109 ms and returned 15 rows. Result Details

Figura 6.12. Resultado obtido pela instrução de consulta imperfeita e seu grau de satisfação.

Fonte: Elaborado pelo autor.

O módulo realiza a mensuração dos dados independentemente do tipo do dado, desde que definidos no *"DDDN"*, como também que haja a possibilidade de realizar uma comparação lógica entre estes. Conforme exemplo apresentado pela Figura 6.12, as seguintes interpretações foram utilizadas:

- **Termo *"velho"* com valor preciso *"70"*:** Obtém-se o coeficiente de satisfação 1. Isto porque foi definido no *"DDDN"* que qualquer valor acima de *"60"* seja aceito como *"velho"*;
- **Termo *"velho"* com termo *"velho"*:** Obtém-se o coeficiente de satisfação 1. Pois os termos são iguais, não necessita de interpretação;

- **Termo *"velho"* com termo *"adulto"*:** Obtém-se o coeficiente de satisfação $0,78$. Uma vez que ambos os termos são variáveis linguísticas, utiliza-se a *função de similaridade* de *Jaccard*. Com esta função é possível obter o valor da relação entre dois conjuntos. Deste modo, compara-se o conjunto de idades para o termo velho com o conjunto de idades para o termo adulto;
- **Termo *"velho"* com valor preciso *"40"*:** Obtém-se o coeficiente de satisfação $0,5$. Relacionando o valor preciso 40 com o termo velho definido no *"DDDN"*, é possível fazer uso da *função trapezoidal*. Com esta função podemos obter o coeficiente que representa o quanto é possível que 40 faça parte do termo velho.

Outros tipos de dados podem ser manipulados conforme sua característica. Como exemplo, para o tipo de expressão nebulosa, a sua definição é composta por uma expressão definida pelo usuário. Assim, fazendo uso da biblioteca *Java Scripting Language*, torna-se possível validar expressões matemáticas. Desta forma, possibilita que o usuário desenvolva sua interpretação para uma expressão nebulosa como *"próximo a x"*.

6.2.3.1. Considerações Finais

O módulo interpretador de resultados é essencial em um modelo em que não são definidos parâmetros com exatidão. Por meio do módulo proposto é possível representar o quanto um dado utilizado em uma instrução está relacionado com o dado obtido como resultado.

Este método é necessário para apresentar ao usuário o *rank* de resultados aceitos, já que em alguns casos se torna complexo a análise dos dados obtidos. Além disso, permite a execução do filtro definido pelo usuário quando julgar necessário. O resultado apresentado pode ser limitado com base no limiar definido pelo próprio usuário, aumentando a precisão dos resultados.

6.3. Estudo de Caso Aplicado

Como forma de validar os conceitos abordados pela aplicação proposta, será utilizado o estudo de caso a seguir. O estudo de caso foi elaborado com base nos principais

conceitos do *Big Data*. Deste modo, é apresentado um estudo de caso onde os dados possuem diversidade de tipos, não estando corretamente formatados em suas estruturas formais.

6.3.1. Rede Social

Uma rede social consiste da análise de diversos tipos de relacionamentos entre indivíduos. A sua característica principal é compreender se dois indivíduos estão ou não relacionados de algum modo. Todavia, os modelos tradicionais possuem dificuldade em determinar as características dos diferentes tipos de informações baseados nestes relacionamentos.

Utilizando o modelo de banco de dados de grafos, é possível gerar o grafo da Figura 6.13. Esta estrutura possui vértices representando pessoas, cada um com suas características informadas por meio das propriedades destes vértices. Quando duas pessoas possuem algum tipo de relacionamento uma aresta é criada representando esta relação. Este relacionamento é identificado pelo rótulo *"AMIGO"*. Do mesmo modo que as propriedades dos vértices representam informações referentes as pessoas, nas propriedades das arestas estão informadas as propriedades deste relacionamento. Estas definem a intensidade do relacionamento, sendo por meio da atribuição de um valor estipulado entre 0 e 1, ou ainda, na atribuição de um dado imperfeito, quando não se pode determinar qual a intensidade deste relacionamento.

A partir do banco de dados de grafos inicial, é proposto uma série de casos de testes para verificar o funcionamento do módulo proposto neste trabalho, integrado ao sistema atual do *Neo4j*. Foram definidos a princípio alguns dados imperfeitos existentes neste banco. A relação de testes é apresentada a seguir:

1. **Definição inicial dos dados imperfeitos:** Pode-se observar na Figura 6.14, que estão sendo definidos os parâmetros para interpretação dos dados imperfeitos. Na seção *"Query"*, é apresentada a instrução *Cypher* utilizada. Em *"Result"*, apresenta-se um resumo dos parâmetros definidos, organizados no padrão *XML* da estrutura *DDDN*;

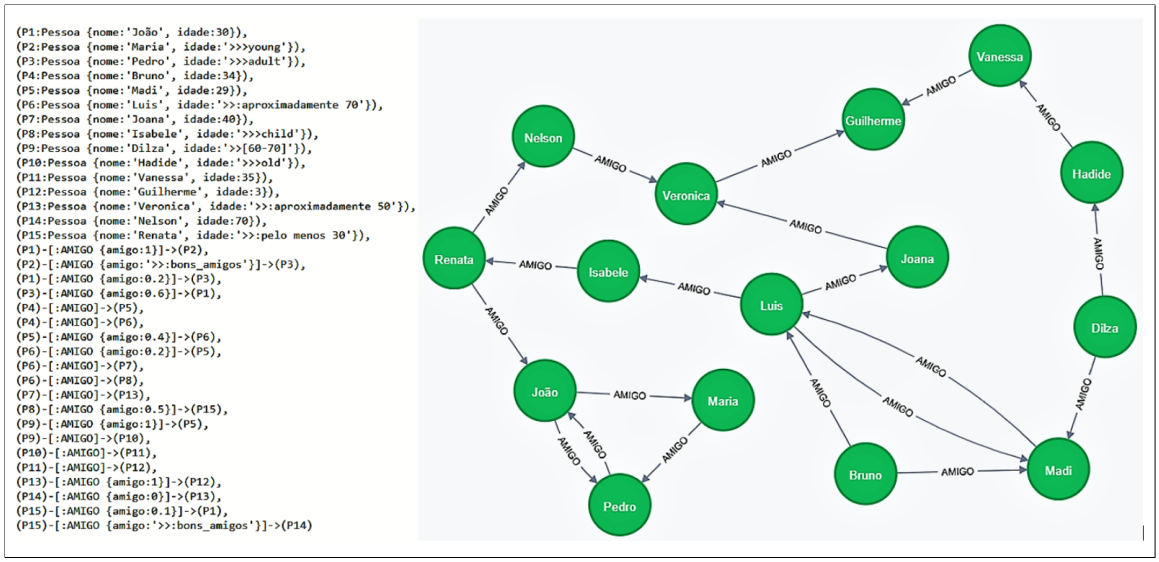


Figura 6.13. Estrutura de uma rede social no modelo de grafos.

Fonte: Elaborado pelo autor.

2. **Necessita-se inserir um registro no banco sobre uma pessoa de nome José, mas desconhece-se sua idade exata, somente sabe-se que é jovem:** Neste caso utilizou-se a instrução de inserção modificada para possibilitar a inclusão de dados imperfeitos. Conforme exemplo da Figura 6.15, ao ser inserido o valor proposto é feita a conversão para identificar o tipo deste dado, neste caso um termo linguístico para o atributo do vértice;
3. **Sabe-se que José e Pedro são bons amigos:** Nesta situação, define-se a relação entre dois vértices por meio de um dado imperfeito do tipo expressão nebulosa. Conforme exemplo da Figura 6.16, do mesmo modo que na inserção de um atributo do vértice, a instrução converte o dado utilizado de modo que possa ser identificado posteriormente;
4. **Qual a relação de pessoas jovens que existe neste banco de dados?** Neste caso, objetiva-se identificar os vértices que contém um valor de propriedade que seja compatível com o termo *"jovem"*. Este valor pode ser perfeito ou imperfeito. Conforme demonstrado pela Figura 6.17, retorna-se as respostas juntamente com o grau de satisfação ou compatibilidade com a instrução declarada;
5. **E se desejamos aumentar a exatidão das respostas do item anterior?** Para este exemplo foi adicionado um limiar de consulta, representado

pelo termo *"WITH THRESHOLD"*, conforme exemplo da Figura 6.18. O limiar foi aplicado para adicionar um limite de respostas aceitas, neste caso $0,2$. Qualquer resultado que obtenha o grau de satisfação menor que o valor estipulado é descartado.

6. **Quais os jovens que conhecem mais pessoas idosas?** Neste exemplo foi utilizado a combinação de dois termos linguísticos. Conforme Figura 6.19, são utilizados os termos *"jovem"* e *"idoso"* na instrução de consulta. O grau de satisfação obtido é pelo menor valor da combinação do *grau de satisfação* dos termos analisados individualmente.
7. **Qual a relação de pessoas que são boas amigas?** Neste exemplo, utilizou-se uma expressão nebulosa para avaliar a relação entre vértices. Conforme Figura 6.20, apresenta-se o resultado obtido pela interpretação da expressão nebulosa *"bons amigos"*, definido no *DDDN*. O resultado desta expressão define a intensidade da relação dos vértices que atendam ao requisito declarado.

Figura 6.14. Teste e resultado com a instrução de definição de dados imperfeitos.

Fonte: Elaborado pelo autor.

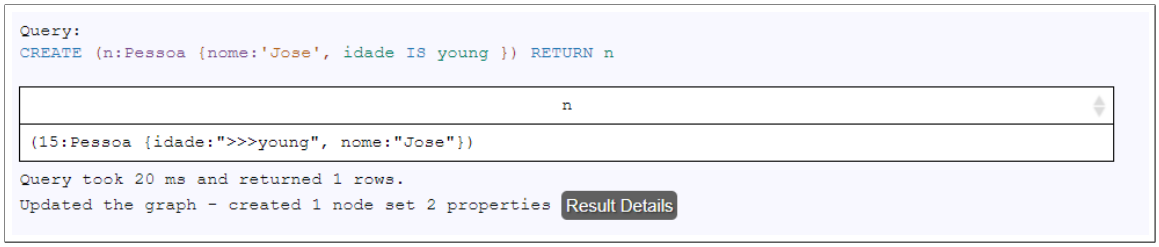


Figura 6.15. Teste e resultado com a instrução de inserção contendo dados imperfeitos.

Fonte: Elaborado pelo autor.

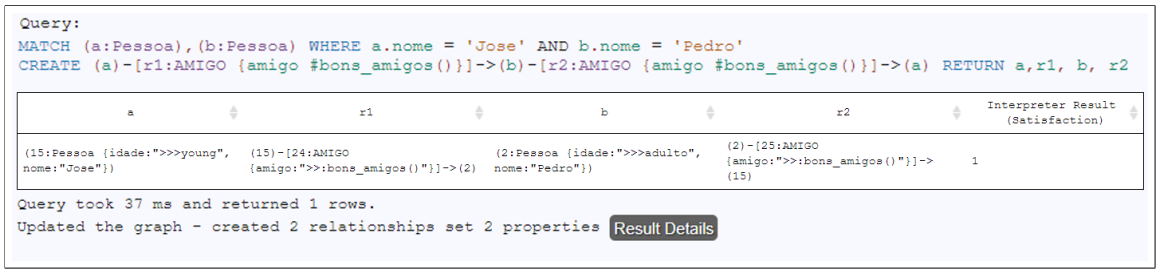


Figura 6.16. Teste e resultado da instrução de definição de uma aresta contendo dado imperfeito.

Fonte: Elaborado pelo autor.

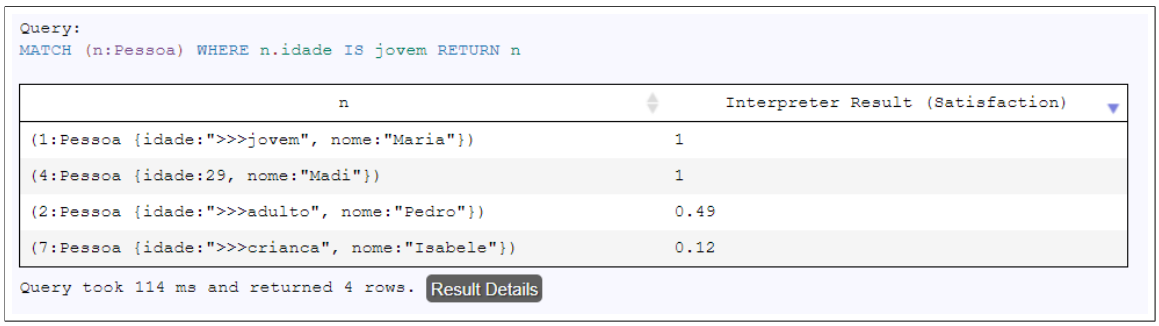


Figura 6.17. Teste e resultado da instrução de consulta por meio de um dado imperfeito.

Fonte: Elaborado pelo autor.

Query:
```
MATCH (n:Pessoa) WHERE n.idade IS jovem WITH TRESHOLD 0.2 RETURN n
```

n	Interpreter Result (Satisfaction)
(1:Pessoa {idade:">>>jovem", nome:"Maria"})	1
(4:Pessoa {idade:29, nome:"Madi"})	1
(2:Pessoa {idade:">>>adulto", nome:"Pedro"})	0.49

Query took 5 ms and returned 3 rows. Result Details

Figura 6.18. Teste e resultado da adição de um limiar de consulta.

Fonte: Elaborado pelo autor.

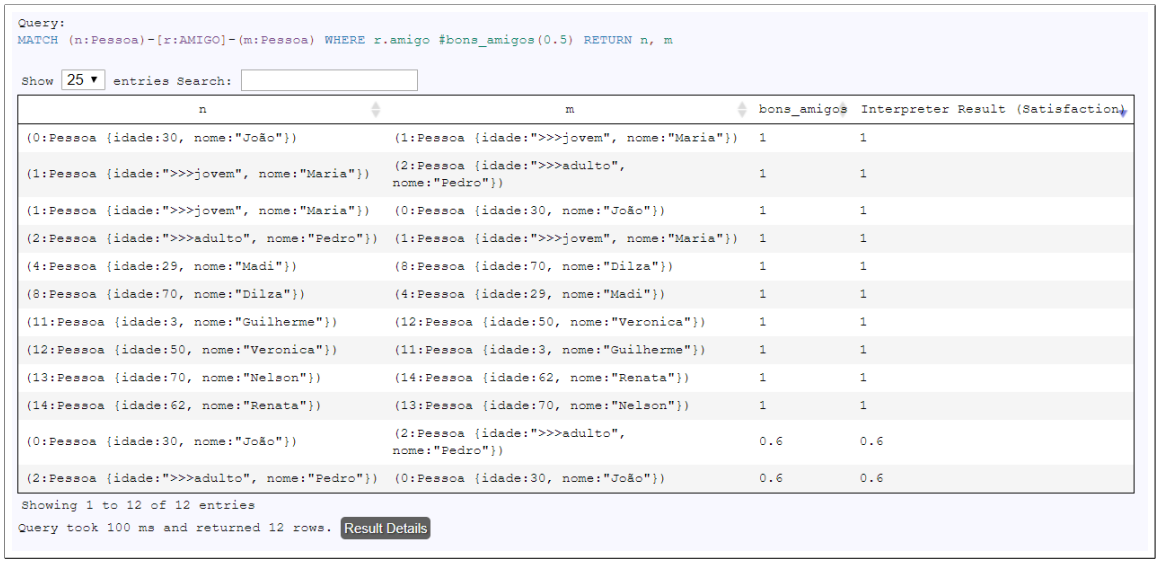
Query:
```
MATCH (n:Pessoa)-[r]-(m:Pessoa) WHERE n.idade IS jovem AND m.idade IS idoso RETURN COUNT(r), n, m ORDER BY COUNT(r)
```

COUNT(r)	n	m	Interpreter Result (Satisfaction)
1	(1:Pessoa {idade:">>>jovem", nome:"Maria"})	(0:Pessoa {idade:30, nome:"João"})	1
1	(4:Pessoa {idade:29, nome:"Madi"})	(8:Pessoa {idade:70, nome:"Dilza"})	1
1	(4:Pessoa {idade:29, nome:"Madi"})	(3:Pessoa {idade:34, nome:"Bruno"})	1
1	(1:Pessoa {idade:">>>jovem", nome:"Maria"})	(2:Pessoa {idade:">>>adulto", nome:"Pedro"})	1
2	(4:Pessoa {idade:29, nome:"Madi"})	(5:Pessoa {idade:60, nome:"Luis"})	1
1	(2:Pessoa {idade:">>>adulto", nome:"Pedro"})	(1:Pessoa {idade:">>>jovem", nome:"Maria"})	0.49
2	(2:Pessoa {idade:">>>adulto", nome:"Pedro"})	(0:Pessoa {idade:30, nome:"João"})	0.49
1	(7:Pessoa {idade:">>>crianca", nome:"Isabele"})	(5:Pessoa {idade:60, nome:"Luis"})	0.12
1	(7:Pessoa {idade:">>>crianca", nome:"Isabele"})	(14:Pessoa {idade:62, nome:"Renata"})	0.12

Query took 202 ms and returned 9 rows. Result Details

Figura 6.19. Teste e resultado da instrução de consulta contendo a combinação de termos linguísticos.

Fonte: Elaborado pelo autor.



```
Query:
MATCH (n:Pessoa)-[r:AMIGO]-(m:Pessoa) WHERE r.amigo ≠bons_amigos(0.5) RETURN n, m
```

Show 25 entries Search:

n	m	bons_amigos	Interpreter Result (Satisfaction)
(0:Pessoa {idade:30, nome:"João"})	(1:Pessoa {idade:">>>jovem", nome:"Maria"})	1	1
(1:Pessoa {idade:">>>jovem", nome:"Maria"})	(2:Pessoa {idade:">>>adulto", nome:"Pedro"})	1	1
(1:Pessoa {idade:">>>jovem", nome:"Maria"})	(0:Pessoa {idade:30, nome:"João"})	1	1
(2:Pessoa {idade:">>>adulto", nome:"Pedro"})	(1:Pessoa {idade:">>>jovem", nome:"Maria"})	1	1
(4:Pessoa {idade:29, nome:"Madi"})	(8:Pessoa {idade:70, nome:"Dilza"})	1	1
(8:Pessoa {idade:70, nome:"Dilza"})	(4:Pessoa {idade:29, nome:"Madi"})	1	1
(11:Pessoa {idade:3, nome:"Guilherme"})	(12:Pessoa {idade:50, nome:"Veronica"})	1	1
(12:Pessoa {idade:50, nome:"Veronica"})	(11:Pessoa {idade:3, nome:"Guilherme"})	1	1
(13:Pessoa {idade:70, nome:"Nelson"})	(14:Pessoa {idade:62, nome:"Renata"})	1	1
(14:Pessoa {idade:62, nome:"Renata"})	(13:Pessoa {idade:70, nome:"Nelson"})	1	1
(0:Pessoa {idade:30, nome:"João"})	(2:Pessoa {idade:">>>adulto", nome:"Pedro"})	0.6	0.6
(2:Pessoa {idade:">>>adulto", nome:"Pedro"})	(0:Pessoa {idade:30, nome:"João"})	0.6	0.6

Showing 1 to 12 of 12 entries

Query took 100 ms and returned 12 rows. Result Details

Figura 6.20. Teste e resultado do uso de uma expressão nebulosa em uma instrução de consulta.

Fonte: Elaborado pelo autor.

6.4. Considerações Finais

O uso da aplicação proposta deste trabalho se mostrou satisfatória no caso de uso demonstrado. Considerando os modelos existentes, não seria possível a aplicação dos conceitos utilizados.

Por meio da extensão da linguagem *Cypher* proposta neste trabalho, foi possível executar as instruções de forma mais simplificada ao usuário, possibilitando o uso de dados imperfeitos, tanto na execução como nos resultados obtidos. O método de análise consegue demonstrar ao usuário a compatibilidade dos resultados obtidos com os parâmetros declarados. Conforme Figura 6.21, é apresentado a pilha de rastreamento obtido da execução dos módulos propostos pela aplicação. Inclui-se também as formas de análise da comparação dos dados utilizados.

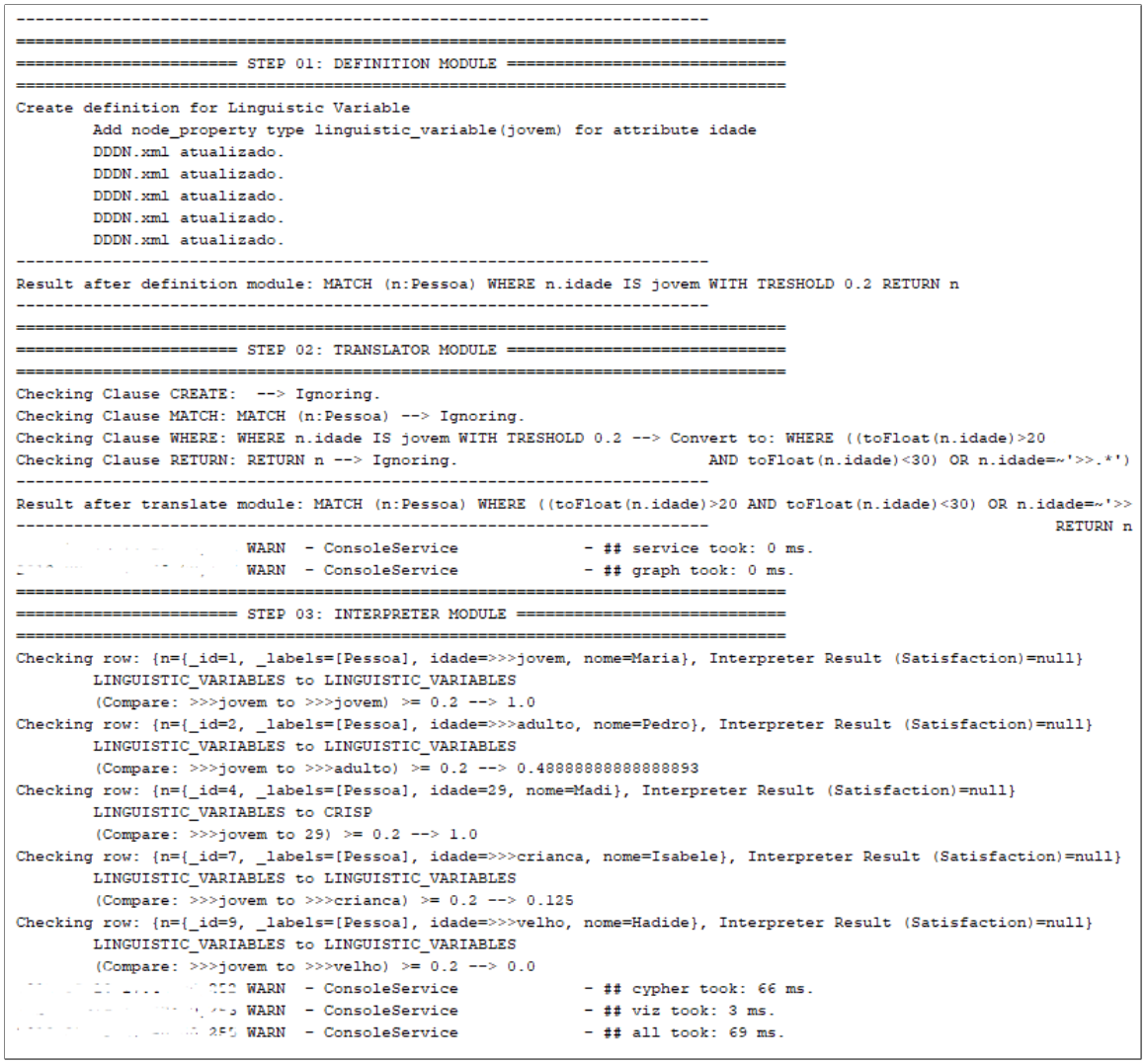


Figura 6.21. Pilha de rastreamento da execução dos módulos propostos no trabalho.

Fonte: Elaborado pelo autor.

Todavia, é clara a necessidade de continuar aprimorando o modelo e a implementação, visto que algumas funcionalidades já existentes no sistema acabam por ser comprometidas pelo uso de tipos variados. Em testes realizados no sistema, foi constatado que há necessidade de adaptar funções simples para que não ocasionasse erros na aplicação. Isto porque dados de tipos diferentes não conseguem se relacionar, em muitos casos, pois não existe uma definição geral para alinhar o entendimento entre estes.

Para exemplificar, executando uma função de comparação do tipo maior menor em um dado preciso do tipo número, como *"18"* e um dado imperfeito, como

"jovem", definido pela aplicação proposta, é apresentado um erro informando que o sistema não consegue fazer a relação entre estes, para definir qual o maior e qual o menor. Mesmo para um usuário, a relação entre estes dados é subjetiva, apresentando um série de respostas possíveis, mas sem obter uma resposta concreta ao final. Esta complexa relação teve que ser expressa até certo ponto na aplicação proposta. Entretanto, há a necessidade de modificações em baixo nível para proporcionar uma melhor integração entre estes tipos de dados.

CAPÍTULO 7

Conclusões e Proposições de Trabalhos Futuros

Neste capítulo serão apresentadas as conclusões obtidas com o desenvolvimento deste trabalho. São elencados os principais pontos observados e as dificuldades encontradas, além de oportunidades para trabalhos futuros que devem ser realizados para a continuidade deste trabalho.

7.1. Pontos Principais e Dificuldades Encontradas

A contribuição deste trabalho é a inclusão de informação imperfeita em um banco de dados orientado a grafos. Para esta contribuição, foram revisadas a representação da informação imperfeita em outros modelos de bancos de dados, de modo a garantir a harmonia entre o modelos de consultas perfeitos e imperfeitos. Foi necessário o estudo dos conjuntos nebulosos como forma de expressão matemática da informação nebulosa.

Como parte das contribuições, no trabalho foi estabelecido um modelo genérico de inclusão de informação imperfeita em bancos de dados de grafos. Este modelo foi concretizado, na linguagem *Cypher* do sistema *Neo4j* para o qual foram propostos novos comandos da linguagem. Adicionalmente, foi implementada uma modificação baseada no sistema *SUGAR*, na linguagem *Cypher* do sistema *Neo4j*, que permitiu a aplicação deste modelo. Como resultado final, foi apresentado um estudo de caso verificando a validade da proposta.

Neste trabalho foi apresentado uma das variadas formas de integrar o conceito da informação imperfeita nos bancos de dados de grafos. Pode-se observar

que o modelo aceita de forma satisfatória a definição deste tipo de dado. Todavia, funcionalidades tiveram que ser adaptadas para comportar dados de formatos diferentes.

7.2. Trabalhos Futuros

Conforme pode ser observado, o conceito de informações imperfeitas é vasto e possui ainda inúmeras lacunas a serem trabalhadas. Abordamos aqui alguns dos conceitos fundamentais da representação e consulta da informação imperfeita nos bancos de dados de grafos. Desta forma, deixamos um relação de possíveis trabalhos que podem ser realizados futuramente:

- *Definição de novos tipos de dados imperfeitos:* Abordamos neste trabalho os dados imperfeitos fundamentais. Contudo, existem inúmeras variações destes dados. Uma consulta na literatura indica a extensão deste assunto, pois cada indivíduo interpreta um dado de forma diferente. Deste modo, uma das possibilidade de continuação deste trabalho, consiste em aprimorar a definição dos dados. Assim, poderá ser desenvolvida uma forma genérica para o tratamento de tipos de dados imperfeitos;
- *Aprimoramento da interface com o usuário:* A aplicação proposta foi desenvolvida para se relacionar com as informações imperfeitas. Com isto, não foram explorados adequadamente todos os pontos da interação com o usuário. É possível desenvolver uma interface mais simples de definição de dados imperfeitos, bem como oferecer a usuários inexperientes, recursos para complementar ou indicar a forma correta do uso das definições propostas;
- *Aplicação na área de inteligência artificial:* Neste trabalho abordamos as definições da informação imperfeita de forma manual. Entretanto, uma das possíveis abordagens consiste em aplicar alguns dos conceitos pertencentes à inteligência artificial. Como a integração em aplicações que coletam e interpretam dados sobre a preferência ou constume do usuário, de modo a prever o que um usuário possa desejar ou a forma que este compreende o mundo. Isto porque, se a aplicação se integrar com as preferências de uso do usuário, estas definições poderiam ser adicionadas automaticamente.

Pode-se ainda, desenvolver métodos de definir a relação entre dois vértices de forma dinâmica, em tempo de execução, por meio da definição dos parâmetros relacionados declarados pelo usuário. Desta forma, a aplicação se tornaria ainda mais dinâmica, aumentando a interação com os usuários;

- *Definição do tipo imperfeito:* A aplicação proposta é executada em uma camada superior ao sistema. Todavia, seria possível o desenvolvimento de um tipo de dado específico para tratar de dados imperfeitos. Da mesma forma que existem os tipos de dado Integer para números e String para texto, seria possível a definição de um novo tipo de dado que integre varios tipos *"imperfeitos"*.

Para que o projeto tenha continuidade, o disponibilizamos em sua forma atual no *GitHub*[6], podendo ser acessado através do link:

https://github.com/brnpsony/unifaccamp-neo4j-informacoes-imperfeitas.

Esperamos que este projeto possa prosseguir, sendo aprimorado cada vez mais. Além de que possa se tornar uma ferramenta útil, contribuindo para melhorar a relação entre usuários e sistemas diversos.

[6] *https://github.com*

Referências Bibliográficas

Angles, R. (2012), A comparison of current graph database models, *in* 'Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference on', IEEE, pp. 171–177.

Angles, R. & Gutierrez, C. (2008), 'Survey of graph database models', *ACM Computing Surveys (CSUR)* **40**(1), 1.

Bordogna, G., Pasi, G. & Lucarella, D. (1999), 'A fuzzy object-oriented data model for managing vague and uncertain information', *International journal of intelligent systems* **14**(7), 623–651.

Bosc, P. & Pivert, O. (1992), 'Some approaches for relational databases flexible querying', *Journal of Intelligent Information Systems* **1**(3-4), 323–354.

Bosc, P. & Pivert, O. (1995), 'Sqlf: a relational database language for fuzzy querying', *IEEE transactions on Fuzzy Systems* **3**(1), 1–17.

Buckles, B. P. & Petry, F. E. (1982), 'A fuzzy representation of data for relational databases', *Fuzzy sets and systems* **7**(3), 213–226.

Castelltort, A. & Laurent, A. (2014), Fuzzy queries over nosql graph databases: perspectives for extending the cypher language, *in* 'International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems', Springer, pp. 384–395.

Chen, G., Vandenbulcke, J. & Kerre, E. E. (1992), 'A general treatment of data redundancy in a fuzzy relational data model', *Journal of the American Society for Information Science* **43**(4), 304–311.

Chen, S.-M. & Jong, W.-T. (1997), 'Fuzzy query translation for relational database systems', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **27**(4), 714–721.

Cheng, J., Ma, Z. & Yan, L. (2010), f-sparql: A flexible extension of sparql, *in* 'International Conference on Database and Expert Systems Applications', Springer, pp. 487–494.

Chiang, D.-A., Lin, N. P. & Shis, C.-C. (1998), 'Matching strengths of answers in fuzzy relational databases', *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **28**(3), 476–481.

Dicio, D. d. P. (2016), 'Disponível em:https://www.dicio.com.br', *Acesso em:(07 de junho de 2017)* .

Elmasri, R., Navathe, S. B., Pinheiro, M. G. *et al.* (2005), 'Sistemas de banco de dados'.

Gaurav, A. & Alhajj, R. (2006), Incorporating fuzziness in xml and mapping fuzzy relational data into fuzzy xml, *in* 'Proceedings of the 2006 ACM symposium on Applied computing', ACM, pp. 456–460.

George, R., Buckles, B. P. & Petry, F. E. (1993), 'Modelling class hierarchies in the fuzzy object-oriented data model', *Fuzzy sets and systems* **60**(3), 259–272.

George, R., Srikanth, R., Petry, F. E. & Buckles, B. P. (1996), 'Uncertainty management issues in the object-oriented data model', *IEEE Transactions on Fuzzy Systems* **4**(2), 179–192.

Jin, Y. & Veerappan, S. (2010), A fuzzy xml database system: Data storage and query processing, *in* 'Information Reuse and Integration (IRI), 2010 IEEE International Conference on', IEEE, pp. 318–321.

Kaliyar, R. K. (2015), Graph databases: A survey, *in* 'Computing, Communication & Automation (ICCCA), 2015 International Conference on', IEEE, pp. 785–790.

Koyuncu, M. & Yazici, A. (2001), A fuzzy database and knowledge base environment for intelligent retrieval, *in* 'IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th', Vol. 4, IEEE, pp. 2311–2316.

Lee, J. & Fanjiang, Y.-Y. (2003), 'Modeling imprecise requirements with xml', *Information and Software Technology* **45**(7), 445–460.

Ma, Z. (2007), A literature overview of fuzzy database modeling, *in* 'Intelligent Databases: Technologies and Applications', IGI Global, pp. 167–196.

Ma, Z., Liu, J. & Yan, L. (2010), 'Fuzzy data modeling and algebraic operations in xml', *International Journal of Intelligent Systems* **25**(9), 925–947.

Ma, Z. & Yan, L. (2007), 'Fuzzy xml data modeling with the uml and relational data models', *Data & Knowledge Engineering* **63**(3), 972–996.

Ma, Z. & Yan, L. (2010), 'A literature overview of fuzzy conceptual data modeling.', *J. Inf. Sci. Eng.* **26**(2), 427–441.

Ma, Z. & Yan, L. (2016), 'Modeling fuzzy data with xml: A survey', *Fuzzy Sets and Systems* **301**, 146–159.

Ma, Z., Zhang, W.-J. & Ma, W. (2000), 'Semantic measure of fuzzy data in extended possibility-based fuzzy relational databases', *International Journal of Intelligent Systems* **15**(8), 705–716.

Ma, Z., Zhang, W.-J. & Ma, W. (2004), 'Extending object-oriented databases for fuzzy information modeling', *Information Systems* **29**(5), 421–435.

Medina, J. M., Pons, O. & Vila, M. A. (1994), 'Gefred: a generalized model of fuzzy relational databases', *Information sciences* **76**(1-2), 87–109.

Na, S. & Park, S. (1996), A process of fuzzy query on new fuzzy object oriented data model, *in* 'International Conference on Database and Expert Systems Applications', Springer, pp. 500–509.

Nepal, S., Ramakrishna, M. & Thom, J. A. (1999), A fuzzy object query language (foql) for image databases, *in* 'Database Systems for Advanced Applications, 1999. Proceedings., 6th International Conference on', IEEE, pp. 117–124.

Oliboni, B. & Pozzani, G. (2008), Representing fuzzy information by using xml schema, *in* 'Database and Expert Systems Application, 2008. DEXA'08. 19th International Workshop on', IEEE, pp. 683–687.

Panić, G., Racković, M. & Škrbić, S. (2014), 'Fuzzy xml and prioritized fuzzy xquery with implementation', *Journal of Intelligent & Fuzzy Systems* **26**(1), 303–316.

Parsons, S. (1996), 'Current approaches to handling imperfect information in data and knowledge bases', *IEEE Transactions on knowledge and data engineering* **8**(3), 353–372.

Pedrycz, W. & Gomide, F. (1998), *An introduction to fuzzy sets: analysis and design*, Mit Press.

Penteado, R. R., Schroeder, R., Hoss, D., Nande, J., Maeda, R. M., Couto, W. O. & Hara, C. S. (2014), 'Um estudo sobre bancos de dados em grafos nativos', *X ERBD-Escola Regional de Banco de Dados* .

Petry, F. E. (2012), *Fuzzy databases: principles and applications*, Vol. 5, Springer Science & Business Media.

Pivert, O., Slama, O., Smits, G. & Thion, V. (2016), Sugar: A graph database fuzzy querying system, *in* 'IEEE International Conference on Research Challenges in Information Science (RCIS16) Demos'.

Pivert, O., Thion, V., Jaudoin, H. & Smits, G. (2014), On a fuzzy algebra for querying graph databases, *in* 'Tools with Artificial Intelligence (ICTAI), 2014 IEEE 26th International Conference on', IEEE, pp. 748–755.

Prade, H. & Testemale, C. (1984), 'Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries', *Information sciences* **34**(2), 115–143.

Raju, K. & Majumdar, A. K. (1988), 'Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems', *ACM Transactions on Database Systems (TODS)* **13**(2), 129–166.

Robinson, I., Webber, J. & Eifrem, E. (2013), *Graph databases*, "O'Reilly Media, Inc.".

Rodriguez, M. A. & Neubauer, P. (2012), The graph traversal pattern, *in* 'Graph Data Management: Techniques and Applications', IGI Global, pp. 29–46.

Shukla, P. K., Darbari, M., Singh, V. K. & Tripathi, S. P. (2011), 'A survey of fuzzy techniques in object oriented databases', *International Journal of Scientific and Engineering Research* **2**(11), 1–11.

Smets, P. (1997), Imperfect information: Imprecision and uncertainty, *in* 'Uncertainty management in information systems', Springer, pp. 225–254.

Stasiu, R. K. (2007), 'Avaliação da qualidade de funções de similaridade no contexto de consultas por abrangência'.

Sunitha, M. & Mathew, S. (2013), 'Fuzzy graph theory: a survey', *Annals of Pure and Applied mathematics* **4**(1), 92–110.

Takahashi, Y. (1991), 'A fuzzy query language for relational databases', *IEEE Transactions on Systems, Man, and Cybernetics* **21**(6), 1576–1579.

Tseng, C., Khamisy, W. & Vu, T. (2005), 'Universal fuzzy system representation with xml', *Computer Standards & Interfaces* **28**(2), 218–230.

Turowski, K. & Weng, U. (2002), 'Representing and processing fuzzy informationan xml-based approach', *Knowledge-Based Systems* **15**(1-2), 67–75.

Umano, M. & Fukami, S. (1994), 'Fuzzy relational algebra for possibility-distribution-fuzzy-relational model of fuzzy data', *Journal of Intelligent Information Systems* **3**(1), 7–27.

Umano, M., Imada, T., Hatono, I. & Tamura, H. (1998), Fuzzy object-oriented databases and implementation of its sql-type data manipulation language, *in* 'Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on', Vol. 2, IEEE, pp. 1344–1349.

Üstünkaya, E., Yazici, A. & George, R. (2007), 'Fuzzy data representation and querying in xml database', *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **15**(supp01), 43–57.

Yan, L., Ma, Z. & Liu, J. (2009), Fuzzy data modeling based on xml schema, *in* 'Proceedings of the 2009 ACM symposium on Applied Computing', ACM, pp. 1563–1567.

Yazici, A. & Koyuncu, M. (1997), 'Fuzzy object-oriented database modeling coupled with fuzzy logic', *Fuzzy Sets and Systems* **89**(1), 1–26.

Zadeh, L. A. (1971), 'Similarity relations and fuzzy orderings', *Information sciences* **3**(2), 177–200.

Zadeh, L. A. (1999), 'Fuzzy sets as a basis for a theory of possibility', *Fuzzy sets and systems* **100**(1), 9–34.

Zadeh, L. A. *et al.* (1965), 'Fuzzy sets', *Information and control* **8**(3), 338–353.

Zicari, R. (1990), 'Incomplete information in object-oriented databases', *ACM SIGMOD Record* **19**(3), 5–16.

Zimmermann, H.-J. (1996), Fuzzy control, *in* 'Fuzzy Set Theoryand Its Applications', Springer, pp. 203–240.