Um estudo para desenvolvimento de consultas com lógica nebulosa em bancos de dados de documentos

Leandro Brito do Nascimento Nogueira[1], **Luís Mariano Del Val Cura**[1]

[1] Programa de Mestrado em Ciência da Computação
Centro Universitário Campo Limpo Paulista (UNIFACCAMP)
Campo Limpo Paulista - São Paulo - Brasil

Klonoa51@gmail.com, delval@cc.faccamp.br

Abstract *Imperfect information is characterized by data that does not have complete meaning, being inaccurate or vague. Through the concepts of fuzzy sets theory, imperfect information can be better defined. In databases, an data is often presented, because not always the values are obtained in their correct form for registration. In particular, this paper deals with the occurrence of imperfect information in the Document Oriented Database Model. In recent years, this data model has once again gained prominence among database models whether it comes from XML or JSON formats and unstructured information. This article proposes the study of a technology to represent and queries the imperfect information and fuzzy data in Document Oriented Database Model. The project is in its initial phase, it is based in a Application Program Interface (API) related to an Internet Languages Program that transforms the fuzzy data for a Document Data Base MongoDB.*

Resumo. *Informações imperfeitas são caracterizadas por dados que não têm significado completo, sendo imprecisos ou vagos. Através dos conceitos da teoria dos conjuntos nebulosos, podemos definir as informações imperfeitas. Em bancos de dados estes dados não são apresentados, porque é difícil definir sua forma correta para registro. Este artigo trata da definição de informações imperfeitas no Modelo de Banco de Dados Orientado a Documentos. Esse modelo de dados voltou a ganhar destaque fundamentalmente por causa dos formatos XML ou JSON e de informações não estruturadas. Este artigo propõe o estudo de uma tecnologia para representar e consultar informações imperfeitas no Modelo de Banco de Dados Orientado a Documentos. O projeto está em fase inicial, ele é baseado em uma API (Application Program Interface) relacionada a uma Linguagem de programação da Internet que transforma os dados nebulosos para um banco de dados de documentos MongoDB.*

1. Introdução

O uso de banco de dados de documentos cresceu ao longo dos anos, fundamentalmente porque seus recursos são utilizados pelas novas tecnologias de *Big Data*. Essas novas tecnologias combinam a eficiência de sua arquitetura junto com a agilidade das consultas executadas. Esses bancos retornam os resultados das consultas em documentos XML ou JSON, um fator para a aceitação de uso pela comunidade.

Nós humanos temos a capacidade de avaliar e processar tipos de dados incertos e ambíguos. Termos de linguagem relativamente vagos como “rápido e lento” , “quente e frio” , ” jovem e velho” , etc. são usados para se julgar algum parâmetro que está sendo avaliado. Este tipo de informação é definida como informação imperfeita [Ma 2007].

Quando tentamos utilizar este tipo de informação para realizar uma busca em um banco de dados, em especial de documentos, o banco não consegue interpretar essa informação imperfeita como aconteceria com um dado preciso (que é possivel dentro da lógica booleana definir como verdadeiro ou falso e sim ou não). Essa informação pode ter vindo de um raciocínio baseado em termos imprecisos ou indefinidos porém familiarizado na rotina de quem teve o raciocínio. Que geralmente é o mais comun entre a sociedade, devemos primeiramente entender como vamos organizar esse conjunto de dados, sendo que os conjuntos nebulosos (*fuzzy*) definem o embasamento matemático para esse tipo de informação imperfeita [Zadeh et al. 1965].

Vários trabalhos exploram as definições nebulosas em bancos de dados semiestruturados como XML e JSON [Ma and Yan 2016, Marrara and Psaila 2016, Jin and Veerappan 2015]. Outros trabalhos apresentaram linguagens de consulta nebulosa sobre esses bancos como XQuery para XML [Thomson and Radhamani 2009].

Existem múltiplos bancos de dados de documentos: OrientDB [Tesoriero 2013], CouchDB [Lennon 2009], MongoDB [Suter 2012], dentre outros. Para este projeto utilizaremos o banco MongoDB.

O objetivo desta pesquisa é o tratamento de informação imperfeita por meio de consultas embasadas em conjuntos nebulosos, a bancos de dados de documentos. Não desejamos mudanças nas linguagens de consulta nesses banco de dados. Pretendemos neste projeto a definição de uma Interface de Aplicação (API) em uma linguagem com acesso à internet para realizar as transformações de consultas nebulosas em JSON para essas consultas. Esta interface deve ser responsável por todas essas transformações sem modificar o banco de dados de documentos. A Seção 2 revisita a conceituação de Banco de Dados Orientados a Documentos e apresenta uma revisão bibliográfica de trabalhos relacionados ao projeto sendo descrito. Na sequência, a Seção 3 apresenta e discute a caracterização de informação imperfeita para, na sequência, apresentar os conceitos básicos relacionados à Teoria de Conjuntos Nebulosos. Adiante temos na Seção 4 a interface de aplicação (API) bem como alguns techos da implementação inicial e testes realizados juntamente com o resultado atual da pesquisa, e por fim a conclusão e os trabalhos futuros.

2. Bancos de Dados Orientados a Documentos e Pesquisas Relacionadas

Bancos de Dados de Documentos utilizam documentos XML e JSON como forma de armazenamento e também como resultado das consultas. Este modelo de dados é bem eficiente e permite que cada documento tenha um conjunto de chaves e um conjunto de valores dos seus campos com delimitadores de tipo colchetes, chaves e parênteses. A estrutura de cada documento no banco de dados é flexível e pode ser mudada dinamicamente. No trabalho [Mehrab and Harounabadi 2018] foi apresentado a F-XML uma linguagem de consulta associada ao banco de dados nebuloso de documentos XML. Esta definição apresenta uma linguagem téorica não implementada.

Vários trabalhos tem explorado a inclusão de definições nebulosas em documentos como XML e JSON [Ma and Yan 2016, Marrara and Psaila 2016, Jin and Veerappan 2015] mas sem relação com bancos de dados de documentos. Em [Ueng and Skrbi 2012] é proposto a extensão da XQuery em uma implementação dos conjuntos nebulosos em banco de dados XML.

Em [Thomson and Radhamani 2009] também foram implementados os conjuntos nebulosos em comandos da XQuery. Em [Jin and Veerappan 2015] foi utilizada uma técnica similar a [Thomson and Radhamani 2009], porém nesse trabalho o autor usou valores de aproximação. É interessante mencionar um sistema para consulta de similiaridade em MongoDB usado em [Damaiyanti et al. 2017]. Na pesquisa de [Frozza and Mello 2020] é mostrado um meio de extender o JSON para conseguir lidar com dados espaciais. Nesse trabalho o que é mais interessante é a manipulação do JSON que é também a saída da consulta realizada no MongoDB.

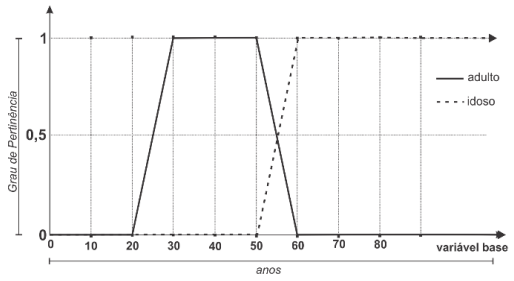
3. Informação Imperfeita e conjuntos nebulosos

Uma informação imperfeita é classificada em cinco tipos principais, sendo estes: imprecisão, incerteza, vagueza, ambiguidade e inconsistência [Ma 2007]. A informação imperfeita tem ausência de precisão e pode ter vários sentidos o que gera inconsistência na consulta. Utilizamos o seguinte cenário, existe um número "x" de usuários no banco, porém temos a seguinte consulta "selecionar os usuários idosos" . Percebe-se a falta de precisão no termo de busca, uma vez que a idade dos usuários não é exata. Por utilizar o termo "idoso" a idade pode ser 50,61 ou 90. Precisamos tratar essa informação para chegar no banco de dados de documento de uma forma objetiva e legível. Para interpretar a informação imperfeita vamos definir matematicamente o conceito de conjuntos nebulosos propostos por [Zadeh et al. 1965].

Seja U o conjunto de u elementos de um universo discreto ou contínuo. Um ***conjunto nebuloso*** F em U é caracterizado por uma ***função de pertinência*** $\mu_F(u)$, associada a cada um dos elementos do universo U com valores no intervalo $[0, 1]$, isto é, $\mu_F(u) \rightarrow [0, 1]$, $u \in U$. Adicionalmente, ***núcleo*** C de F é composto dos elementos de grau 1, ou seja, elementos exatos, expressos por: $Core(F) = \{u \in U | \mu_F(u) = 1\}$. O $\alpha - cut$ de F é um limiar , considerando elementos válidos definindo que são iguais ou maiores de um valor. Um $\alpha - cut$ pode ser definido a partir de qualquer valor acima de 0 até 1. Ele deve ter um valor entre 0 e 1, isto é: $F\alpha = \{u | \mu_F(u) \geq \alpha\}$ até $0 \leq \alpha \leq 1$. Elementos definidos por um $\alpha - cut$ com valores próximos a 1, são os elementos que melhor pertencem ao conjunto. De modo geral existem diferentes funções que podem ser utilizadas para obter um grau de pertinência, entre as funçõe mais comuns temos a função *trapezoidal* e a *função tringular* [Zadeh et al. 1965]. A função *trapezoidal* é expressada pela quadrupla (A, B, C, D), onde $C(F) = [B, C]\ and\ S(F) = [B - A, C + D]$.

Em muitos casos a definição ou interpretação dos dados imperfeitos está condicionada à subjetividade do usuário, tendo como base aspectos vivenciados por este. Fatores como local de nascimento e idade são exemplos de aspectos que afetam a interpretação dos dados. Para exemplificar a classificação de uma pessoa com idade de 52 anos é relativa, uma vez que, do ponto de vista de uma criança, a maioria dos adultos são classificados como idosos. O uso da teoria dos conjuntos nebulosos pode auxiliar na definição dos dados imperfeitos. Ao invés de forçar a atribuição de um valor exato ao dado imperfeito, pode-se matematicamente, classificar as suas possíveis interpretações quanto a sua relevância. Seguindo o exemplo dado, como afirmar que uma pessoa de 52 anos de idade é classificada como um adulto ou idoso?. Na Figura 1 aplicando a função trapezoidal de Zadeh [1965], pode-se classificar corretamente este dado. Os valores são colocados na função juntamente com as variável "x"(idade a ser classificada), "a"e "d"(idades com

grau de pertinência 1), ”b”e ”c”(idades com grau de pertinência 0). Nota-se que a idade proposta tem 0.2 de grau de pertinência com o termo idoso e 0.8 com o termo adulto, tornando está a classificação mais compatível.



$$trapezoidal(x;a,b,c,d) = max\left(min\left(\frac{x-a}{b-a},1,\frac{d-x}{d-c}\right),0\right)$$

$$trapezoidal_{adulto}(52;20{,}30{,}50{,}60) = max\left(min\left(\frac{52-20}{30-20},1,\frac{60-52}{60-50}\right),0\right)$$

$$trapezoidal_{adulto}(52;20{,}30{,}50{,}60) = 0{,}8$$

$$trapezoidal_{idoso}(52;50{,}60{,}80{,}90) = max\left(min\left(\frac{52-50}{60-50},1,\frac{90-52}{90-80}\right),0\right)$$

$$trapezoidal_{idoso}(52;50{,}60{,}80{,}90) = 0{,}2$$

Figura 1. Função trapezoidal proposta por Zadeh [1965].

4. Proposta de Interface de Programação de Aplicativos e Resultados

Neste artigo propomos uma Interface de Programação de Aplicativos (API) em Javacript que permita a associação de valores nebulosos para um Banco de Dados de Documentos. Nesta interface as funções definidas não estarão associadas a um banco de dados de documentos especifico concreto. No entanto, para sua implementação vamos utilizar o banco de dados MongoDB [Suter 2012] que é um software de código livre. O resultado das inserções aparecem no formato JSON utilizando um formato similar dentro de chaves “{ }”. Por exemplo, esta operação mostra a inserção dos diferentes campos associados a um usuário dentro de uma coleção chamada *user*.

db.user.insert ({name:”marcos”, age:65, size:{ h:14, w:21, uom: ”cm”} })

O operação a seguir mostra como é realizada uma consulta pelo campo *age* dentro da mesma coleção. O resultado deve oferecer um conjunto de documentos JSON que satisfaça a condição de possuir um campo *age* igual a 65.

db.user.find ({ age: 65 })

Neste caso vamos a apresentar como um exemplo a implementação da função trapezoidal nos conjuntos nebulosos.Inicialmente devemos definir uma função na API que vai estabelecer os valores da função trapezoidal para o atributo *age*. Nessa função devemos estabelecer os valores nebulosos para *adulto* e *idoso*. Por outro lado, na função *MontaQueryDifusa* podemos realizar uma consulta com valores nebulosos. Por exemplo:

MontaQueryDifusa({ age: ”adulto”}, 20, 40, 50, 60)

MontaQueryDifusa({ age: ”idoso”}, 50, 60, 90, 100)

Após o valor difuso passar pela conversão feita dentro da função trapezoidal, a consulta continua para o banco de dados de documentos MongoDB. Por sua vez, o resultado da consulta é retornado e a API deve modificar o JSON com os dados obtidos dessa consulta associados à função trapezoidal.

Assim, esses resultados podem ser mais relevantes que os que aparecem por padrão (exemplo um idoso que esta com 65 porém ainda pode ser considerado um adulto). Para exemplificar melhor o que a API vai fazer, será comentado um trecho do código na Figura 2.

```
function findNebuloso(consultadousuario){
    var idades = [];
    if(consultadousuario.age = "adulto"){
        idades = [20, 40, 50, 60];
    }else if(consultadousuario.age = "idoso"){
        idades = [50, 60, 90, 100];
    }
  //Trata a query para atender a consulta difusa do usuario
  TrataConsulta = montaQueryDifusa(consultadousuario,idades);
  //Faz a consulta para retornar os valores do banco utilizando a query
  retornoConsulta = findMongoDB(TrataConsulta);
  if(retornoConsulta != ''){
    return retornoConsulta;
  }else{
    return false;
  }
}

function montaQueryDifusa(consultadousuario,idades){
  //Prepara a variavel que vai receber o resultado query de busca com os valores difusos
  var query = {}

 preconsulta.forEach(function(item, index){
    query[consultadousuario[item]] = trapezoidal(idades[index],idades);
 });
  /*Aqui vou retornar o json ja montado com o resultado da consulta juntamente com o grau de associação
  de cada coleção*/
  return query;
}
```

Figura 2. Primeiro protótipo de código mostrando dentro da API como os dados fuzzy são normalizados antes de fazer a consulta

A Figura 2 mostra um protótipo de uma parte do código, vamos separar os campos que o usuário deseja pesquisar dos valores nebulosos respectivos associados a uma função trapezoidal. Esse valor vai ser armazenado em uma varíavel chamada *TrataConsulta* e pode conter vários campos pois o usuário pode buscar por quantos campos ele quiser dentro do banco de dados. Após o processamento, ela será chamada dentro de outra função que vai enviar a consulta ao banco de dados *FindMongoDB*. Porém, caso o usuário queira usar outro banco ele precisa definir o banco e a estrutura de conexão nas configurações da API. No final, se a consulta é realizada com sucesso, o banco de dados vai retornar um documento JSON que vai ser armazenado em uma variável de retorno RetornoConsulta. Essa variável vai receber um tratamento para que o documento JSON contido nela seja um documento JSON nebuloso, ou seja, dentro dele também vai conter o dado nebuloso pesquisado. Será colocado também o grau de pertinência que a função trapezoidal retornou. Um exemplo simples do resultado que pretendemos chegar com o retorno da API pode ser mostrado abaixo:

{name:"marcos", age:65, idoso:0.8 ,size:{ h:14, w:21, uom: "cm"}}

Outras alternativas de informação imperfeita também são manipuladas e tratadas na API mas não foram colocadas no artigo por causa do espaço.

A tecnologia precisa oferecer o mesmo tipo de método de consulta para outras plataformas. Note que estamos falando de métodos que podem ser modificados para ter o mesmo efeito em outros bancos de dados de documentos.

5. Conclusão

Neste artigo apresentamos uma Interface de Programaçao de Aplicativos (API) JavaScrip para a transformação de consultas em um Banco de Dados Orientado de Documentos Nebuloso. A interface transforma um conjunto de definições nebulosos em uma consulta em um banco de dados , neste caso o escolhido é MongoDB. Como resultado da consulta

são retornados documentos JSON nebulosos para o usuário final.

Devemos apresentar como resultado final uma ferramenta API pronta para ser utilizada com qualquer Banco de Documentos Nebuloso. Destacamos também, a possiblidade da tecnologia acessar o banco de dados de documento de forma online. Como trabalho futuro deve ser interessante modificar um Banco de Dados de Documentos para que consiga armazenar dados incompletos,

Referências

Ma, Z. and Yan, L. (2016) Modeling fuzzy data with XML: A survey, Fuzzy Sets and Systems, 301(15), pp 146-159.

Tesoriero, C. (2013) Getting Started with OriendDB., Packt Publishing, August 2013.

Lennon, J. (2009) Introduction to CouchDB Views, Beginning CouchDB, Appres, Chapter 7, pp 107-123.

Suter, R. (2012) MongoDB: An Introduction and Performance Analysis. Presented at Conference about Computer Science, HSR Hochschule für Technik Rapperswil, Sweden

Frozza, A. and Mello, R., (2020) JS4Geo: a canonical JSON Schema for geographic data, Geoinformatica.

Jin Y., and Veerappan, A (2010) Fuzzy XML Database System: Data Storage and Query Processing. IEEE International Conference on Information Reuse Integration, pp 318-321.

Ueng, P. and Skrbi, S. (2012) Implementing XQuery Fuzzy Extensions Using a Native XML Database, Proc of the 13th IEEE International Symposium on Computational Intelligence and Informatics (CIBTI 2012), pp 305-309.

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

Psaila, G. and Marrara, S. (2019) , A First Step Toward a Fuzzy Framework for Analyzing collections of JSON Documents, 16th International Conference on Applied Computing 2019, pp 19-28.

Marrara, S. and Pasila, G., (2016). Fuzzy Approaches to Flexible Querying in XML Retrieval. International Journal of Computational Intelligence Systems, 9(100), pp. 95-103.

Thomson E. and Radhamani, G. (2009) Fuzzy Logic Based XQuery operations for Native XML Database Systems, International Journal of Database Theory and Application, Vol. 2, No. 3, September 2009.

Mehrab, F. and Harounabadi, A. (2018) Apply Uncertainty in Document-Oriented Database (MongoDB) Using F-XML , Journal of Advances in Computer Research 9(3), pp 87-101.

Zadeh, L. (1965) Fuzzy sets. *Information and control*, 8, v. 8, n. 3, p. 338–353.

Damaiyanti, T. and Imawan, A. and Indikawati, F. and Choi, Y. and Kwon, J., (2017) A similarity query system for road traffic data based on a NoSQL document store, The Journal of Systems and Software, vol 127, pp 28-51.