UNIVERSIDADE FEDERAL DO PARA CAMPUS TUCURUI

FACULDADE ENGENHARIA DA COMPUTAÇÃO

Mauricio Rodrigues Mendes – 201633840003

Valdeney André Matos Junior – 201633840029

Sistemas de Gerenciamento de Banco de Dados Orientados a Grafos

23 de junho de 2019

UNIVERSIDADE FEDERAL DO PARA CAMPUS TUCURUI

FACULDADE ENGENHARIA DA COMPUTAÇÃO

Sistemas de Gerenciamento de Banco de Dados Orientados a Grafos

A monografia apresentada à disciplina Banco de Dados, como requisito para obtenção de nota parcial, sob orientação do Prof. Marcos Amaris.

23 de junho de 2019

LISTA DE FIGURAS

Sumário

Sistema de Gerenciamento de Banco de Dados Orientado a Grafo

Valdeney André Matos Júnior N: 201633840029
Mauriocio Rodrigues Mendes N: 201633840002

Tucuruí, 2019

Resumo

Este trabalho aborda, como tema principal, os Sistemas de gerenciamentos de Banco de Dados (SGBD) Orientados a Grafos, apresentando suas principais características e funcionalidades bem como algumas definições; Uma breve comparação com SGBDs relacionais é realizada afim de elucidar algumas importantes diferenças entre os mesmos, uma breve introdução à teoria de grafos é utilizado para um melhor entendimento sobre o tema, uma vez que este é base teórica para a implementação de tal sistema. Através de diversas pesquisas realizadas em diferentes fontes, principalmente sites e livros, tenta-se explicar o tema de forma clara e concisa, elucidando pontos principais a respeito do assunto.

Palavras-chaves: SGBD, Grafos, Banco de Dados, Nós, NoSQL, Neo4j.

Introdução

Há muito tempo atrás povos com a necessidade de contabilizar seus rebanhos ou outros pertences para de alguma forma gerir melhor seus bens ou simplesmente garantir que estavam todos presentes criaram formas rudimentares de contagens, utilizando gravetos, ossos ou pedras, faziam suas analises de quantidades representando uma unidade daquilo que estivessem interessados, seja um rebanho, pescas ou produções agrícolas, com um pedacinho de pedra ou alguns riscos nos gravetos, dessa forma eles poderiam posteriormente consultar seus dados e verificar o quanto de peixes pescaram, o quanto de legumes colheram ou mesmo quantas ovelhas retornaram do pastoreio.

Percebesse com esse fato que desde os primórdios da humanidade a necessidade de guardar informações e consultá-las já se fazia presente. Nos dias atuais, um mundo altamente globalizado, interconectado, com um mercado altamente volátil e ditado por diversas grandes corporações, os dados se tornaram bens altamente valiosos para tais empresas e a necessidade de armazená-los e consultá-los levou a criação banco de dados mais complexos e por consequência a sistemas de gerenciamentos de banco dados mais sofisticados.

Os SGBD orientados a grafos como os demais, surgiu de uma necessidade específica que não poderia ser mais bem resolvido por outros modelos de BD. Neste trabalho

apresentamos uma breve visão geral sobre SGBD bem como a BD e enfocamos os SGBDS orientados a grafos, apresentando suas características definições e linguagens.

A tecnologia de banco de dados já foi descrita como sendo "uma das áreas de mais rápido crescimento na ciência da computação e da informação". Como campo, ela ainda é comparativamente jovem; os fabricantes e vendedores só começaram a oferecer sistemas de gerenciamento de banco de dados no final da década de 1960.

1 O que é um Banco de dados e um SGBD?

Segundo Korth, um banco de dados "é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico", ou seja, sempre que for possível agrupar informações que se relacionam e tratam de um mesmo assunto, posso dizer que tenho um banco de dados.

Um SGBD é basicamente um sistema computadorizado de manutenção de registros; em outras palavras, é um sistema computadorizado cuja finalidade geral é armazenar informações e permitir que os usuários busquem e atualizem essas informações quando as solicitar. As informações em questão podem ser qualquer coisa que tenha algum significado ao indivíduo ou à organização a que o sistema deve servir – ou seja, qualquer coisa que seja necessária para auxiliar no processo geral das atividades desse individuo ou dessa organização.

1.1 SGBD existentes no mercado:

SQLserve: Um dos maiores do mundo, sob licença da Microsoft; MySQL: Trata-se de um software livre, com código fonte aberto; FirebirdSQL: Possui código fonte aberto e roda na maioria dos sistemas Unix; Microsoft Access: É um Sistema Gerenciador de Banco de Dados que acompanha o pacote Office da Microsoft. Este SGBD tem poucas atribuições profissionais, sendo mais usado para aprendizagem, devido à sua interface amigável; mSQL: Sistema pequeno e que trabalha mais com o uso eficiente da memória. Foi criado pela Hughes Technologies Pty Ltd.; Neo4j: É um sistema de gerenciamento de banco de dados gráfico desenvolvido pela Neo4j, Inc.

2 SGBD Relacional

Criado por Edgar F. Codd, nos anos de 1970, começou a ser realmente utilizado nas empresas a partir de 1987. A abordagem relacional está baseada no princípio de que as informações em uma base de dados podem ser consideradas relações matemáticas e que estão representadas de maneira uniforme com o uso de tabelas bidimensionais.

Este princípio coloca os dados direcionados a estruturas mais simples de armazenamento de dados, que são as tabelas, e nas quais a visão do usuário é privilegiada. A abordagem relacional representa uma forma de descrever o banco de dados através de conceitos matemáticos simples: a teoria dos conjuntos.

Voltada, principalmente, a melhorar a visão dos dados pelos usuários, a abordagem relacional faz com que os usuários vejam o banco de dados como um conjunto de tabelas bidimensionais, originadas em linhas e colunas.

Definição Clássica: São conjuntos de dados vistos segundo um conjunto de TABELAS, e as operações que as utilizam são feitas por linguagens que o manipulam, não sendo procedurais, ou seja, manipulando conjuntos de uma só vez.

A linguagem padrão dos Bancos de Dados Relacionais é a Structured Query Language ou simplesmente SQL, como é mais conhecida.

3 SGBD não relacional (NoSQL)

De acordo com (OLIVEIRA; CURA, 2015 apud SADALAGE; FOWLER, 2012) a utilização de bancos de dados NoSQL (que não utilizam o modelo relacional com a linguagem de consulta SQL – Structured Query Language) tem crescido consideravelmente, porém escolher qual tipo de banco utilizar é uma tarefa complexa para os desenvolvedores, pois cada um dos bancos NoSQL utiliza uma lógica específica que funciona bem para determinado modelo de dados.

Cada banco de dados NoSQL segue um padrão específico voltado para uma situação, trazendo assim novas perspectivas, tais como: flexibilidade no modelo de dados, escalabilidade horizontal, novas linguagens de consulta apropriadas aos modelos de dados e diversas interfaces para acesso aos dados. (OLIVEIRA; CURA, 2015).

Os bancos de dados NoSQL não se enquadram totalmente no padrão ACID, por utilizarem as propriedades BASE (Basically Available, Soft-state, Eventually consistent), que não ficam verificando a consistência dos dados a cada transação por questão de eficiência. A responsabilidade para assegurar maior nível de consistência é transferido para a aplicação. (KOLOMICENKO, 2013).

Porém a desvantagem é que determinado banco NoSQL fica restrito ao modelo específico dele, sendo necessário adotar um SGBD específico para cada modelo. Então a solução para um determinado sistema que utilize vários modelos de dados seria um banco de dados Multi Modelo, que integra diversos modelos de bancos de dados NoSQL no mesmo SGBD. (OLIVEIRA; CURA, 2015).

4 Tipos de SGBD não relacionais

4.1 Orientados a documento:

Documentos representam a unidade básica neste tipo de tecnologia, sendo possível comparar os mesmos aos registros das tabelas convencionais.

4.2 Chave-valor:

Como o próprio nome sugere, os bancos que se encaixam nesta classificação são formados por conjuntos de chaves e seus respectivos valores. Cada um destes conjuntos, por sua vez, conta ainda com uma chave que funciona como um identificador único

4.3 Orientados a colunas:

Este tipo difere bastante do modelo relacional, em que uma linha representa um conjunto de dados relacionados (com cada um destes últimos correspondendo a colunas)

4.4 Orientados a grafos:

Bancos deste tipo empregam conceitos da teoria de grafos para a representação de relacionamentos entre diferentes conjuntos de dados. Uma das soluções mais conhecidas baseadas neste modelo é o Neo4j.

5 SGBD orientado a grafo

O sistema de gerenciamento de banco de dados relaciona possui uma grande utilização por meio empresarial e acadêmico devido fornece vários benefícios, mas com aplicações mais complexas, esse modelo pode gerar conflitos devido atualizações necessárias.

O Banco de Dados relacional, devido possuir várias interconectividades deu origem ao SGBD orientados a grafos na década de 80 como objetivo de suporte. O crescimento do SGBD orientado a grafo ocorreu com o crescimento da utilização das redes sociais como (Twitter, Facebook). Onde os dados estão todos em base de grafos onde os usuários são os vértices e as ligações entre eles são as arestas.

imagem 1

O SGBD orientado a grafo é composto por multigrafos, onde cada par de vértices pode ser ligado por mais de uma aresta, possuindo eficiência quando o usuário executa filtros complexos em propriedade nas arestas.

O SGBD orientado a grafo pode ser utilizado por vários outros tipos de sistemas, como sistemas de compras virtuais, aplicativos de relacionamento. e sistemas que analisam sites, analisando as arestas que incidem de cada um.

A implementação SGBD orientado a grafo são feitas de duas maneiras, quanto a consulta e armazenamento. Denominadas de nativos, possuem listas com a adjacência representada por: vértice-vértice. Dessa maneira o SGBD em grafo pode armazenar as listas por meio do modelo chave-valor (vértices representando as chaves e as adjacência o valor). Ou seja, em um disco ou na memória. Os classificados como não-nativos representam o meio relacional da forma: vértice-aresta-vértice.

O SGBD em grafo pode ser modelado de várias formas. A mais simples é onde todas arestas são do mesmo tipo de relacionamento e os vértices são do mesmo tipo. A outros tipos de grafos como exemplo: Grafos simples, Hipergrafos, Grafos de propriedade.

O SGBD orientado a grafo é composto por multigrafos, onde cada par de vértices pode ser ligado por mais de uma aresta, possuindo eficiência quando o usuário executa filtros complexos em propriedade nas arestas.

Figura 1 – Modelo de Grafo

O SGBD orientado a grafo pode ser utilizado por vários outros tipos de sistemas,

como sistemas de compras virtuais, aplicativos de relacionamento. e sistemas que analisam sites, analisando as arestas que incidem de cada um.

A implementação SGBD orientado a grafo são feitas de duas maneiras, quanto a consulta e armazenamento. Denominadas de nativos, possuem listas com a adjacência representada por: vértice-vértice. Dessa maneira o SGBD em grafo pode armazenar as listas por meio do modelo chave-valor (vértices representando as chaves e as adjacência o valor). Ou seja, em um disco ou na memória.Os classificados como não-nativos representam o meio relacional da forma: vértice-aresta-vértice.

O SGBD em grafo pode ser modelado de várias formas. A mais simples é onde todas arestas são do mesmo tipo de relacionamento e os vértices são do mesmo tipo. A outros tipos de grafos como exemplo: Grafos simples, Hipergrafos, Grafos de propriedade.

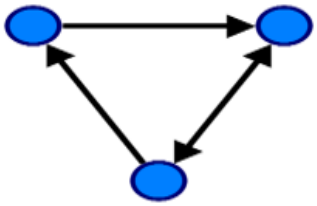


Figura 2: Grafos simples

É um grafo que não tem aresta paralelas nem grafos.

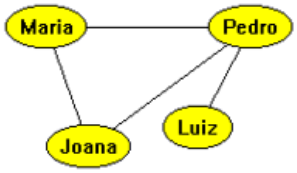
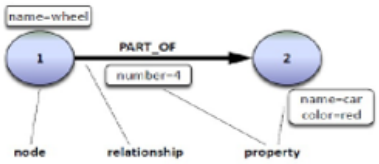


Figura 3: Hipergrafos

É a generalização do grafo, com suas arestas ligando quaisquer quantidades do vértice. Definida por um subconjunto conjunto não vazio V.



Fonte: (NEUBAUER, 2010)

Figura 4: Grafos com propriedade

Suas propriedades são definidas pelos vértices e arestas com atributos.

O banco de dados pode ser usado por vários tipos de algoritmos para consulta como exemplo:

Consultas de acessibilidade: serve para verificar se os vértices estão ligados por uma aresta.

Consultas adjacentes: vértices adjacentes são ligados por uma aresta, e arestas adjacentes compartilham o mesmo vértice. Baseado no princípio vertice-aresta, onde

Consultas de combinações de padrões: Pega o grafo padrão da consulta e pesquisa por subgrafos isomórficos.

Consultas de sumarização: Permite sumarizar ou operar na saída da consulta.

6 sistemas de banco de dados que utilizam a lógica de grafos:

6.1 InfiniteGraph

Banco de dados proprietário desenvolvido pela Objectivity no ano de 2010 com a linguagem de programação Java com o core em C++, e possui uma API (Application Programming Interface) em Java. Possui o modelo lógico baseado em grafo de propriedades, com uma arquitetura centralizada ou distribuída e mecanismo de replicação (com consistência eventual). O modelo físico de armazenamento é o orientado a objetos. (PENTEADO et al., 2014).

6.2 Neo4j

De acordo com (PENTEADO et al., 2014), desenvolvido pela empresa Neo Technology em fevereiro de 2010, foi implementado em Java (com versão proprietária e aberta), possui o modelo de grafos de propriedade. Quanto ao armazenamento físico é baseado em repositórios chave-valor. De acordo com (ROCHA, 2013): "Ele ainda pode ser instalado em um ambiente multiservidor ou pode ser executado embarcado em um programa Java".

6.3 OrientDB

Sistema aberto desenvolvido em Java pela empresa OrienTechnologies no ano de 2011, tendo sua linguagem de consulta baseada em Java, ou Gremlin da Tinkerpop ou ainda SQL adaptado. O modelo de grafo de propriedade é utilizado, o mapeamento é baseado lista livre de adjacências e o armazenamento físico com recursos de banco de dados de documentos e orientação a objetos. (PENTEADO et al., 2014).

6.4 Titan

Criado pela empresa Aurelius no ano de 2012, com código aberto feito em Java, a definição dos esquemas das bases e a manipulação dos dados é baseado na linguagem Gremlin. A modelagem lógica baseada em grafos de propriedades, o mapeamento é feito com lista de adjacências e utiliza pares chave-valor para armazenar fisicamente os dados atraves de bancos de dados específicos. (PENTEADO et al., 2014).

6.5 Trinity

Segundo (PENTEADO et al., 2014 apud SHAO; WANG; LI, 2013), é um banco de dados proprietário desenvolvido pela Microsoft Research, sendo um banco de dados distribuído com modelagem lógica baseada em grafos direcionados, nãodirecionados e

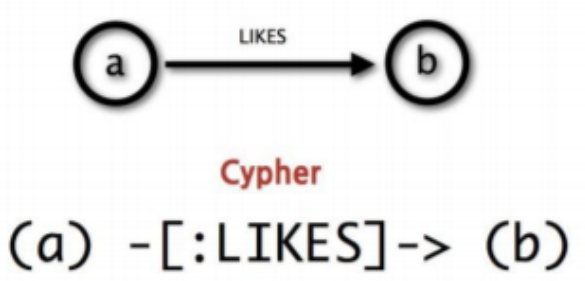
hipergrafos. Não tem suporte ao Tinkerpop, sendo que as linguagens de consulta e processamento são baseadas em uma engine sobre a plataforma Trinity, como a Trinity.RDF. O modelo físico adotado é o repositório chave-valor com mapeamento em lista de adjacência. (PENTEADO et al., 2014 apud ZENG et al., 2013).

7 Linguagem de consulta Cypher

Dando um maior enfoque ao Neo4j por ser o mais amplamente utilizado, será descrita sua linguagem de consulta, criação e modificação de grafos de maneira simples e breve, maiores informações poderão ser consultadas no arquivo de referência presente no site do programa.

O Neo4j suporta diversas linguagens de consultas para grafos na manipulação da sua base de dados, tais como Cypher, Gremlin, REST API, Lucene, entre outras A linguagem Cypher foi escolhida por ser uma linguagem simples, similar ao SQL. Com o Cypher podemos selecionar, inserir, atualizar ou excluir dados em uma base de dados gráfica. Mas não tem comandos para definição de esquema de grafos para a base de dados.

Essa é uma comparação do grafo e a sintaxe da consulta Cypher. No grafo podemos ver o nó “a LIKES b”, ou seja, o nó “a” está ligado ao nó “b” pelo relacionamento “LIKES”. No Cypher representamos os nós por parênteses “(a)”, o tipo do relacionamento por colchetes “[:LIKES]”.



5.png 5.png

Figura 2 – Cypher usando o relacionamento “LIKES”

A apresenta alguns exemplos de consulta usando a linguagem Cypher. O primeiro exemplo mostra a sintaxe da linguagem para uma consulta simples que retorne todas as 39 propriedades de todos os nós da base de dados. A figura 3 é uma consulta simples que retorna todas as propriedades de todos os nós da base de dados e por fim a terceira consulta retorna todas as propriedades de todos os nós da base de dados que possui um relacionamento denominado “LIKES” com outro nó.

```
//sintaxe da consulta
//MATCH (node) RETURN node;

//consultar todas as propriedades no nó “a”
MATCH (a) RETURN a ;

//consultar todas as propriedades das pessoas que o nó “a” possui um relacionamento de “LIKES”
MATCH (a) –[:LIKES] -> (b)
RETURN a;
```

6.png 6.png

Figura 3 – Exemplos de consulta utilizando Cypher

Para enquadrar os nós em grupos e conseguir fazer a distinção dos tipos de nós, por exemplo, Pessoas ou Empresas podemos nomeá-los por labels. A figura 4,apresenta uma consulta utilizando os labels, no exemplo seria retornado todos os nós que se enquadram com o label “Pessoa”.

7.png 7.png

```
//consultar o nó com label “Pessoa”
//MATCH (node:label) RETURN node ou seja:
MATCH (a:Pessoa) RETURN a ;
```

Figura 4 – Exemplos de consulta utilizando label (Cypher)

O comando CREATE é usado para instanciar novos nós. Os atributos do nó devem vir dentro de chaves “ ”. O agrupamento do nó, em um label, é opcional. A figura 5, apresenta um exemplo de criação do nó, agrupado ao label “PESSOA” com os atributos Nome e RG. O valor das propriedades do tipo string devem vir preenchidas entre aspas duplas (“ ”).

8.png 8.png

```
//sintaxe da criação do no
//create (node:label { Propriedade: “valor”, Propriedade: "valor"};

create (n:PESSOA {Nome: “Ana”, RG: “28.058.157-x”};
```

Figura 5 – Exemplos de criação de nós (Cypher)

Semelhante ao SQL a restrição da consulta no Cypher pode ser feita com a cláusula WHERE, a diferença é que o Cypher é case sensitive tanto para o nome da propriedade quanto para o valor da mesma. A figura 6, apresenta um resultado que retorna os atributos Nome e RG dos nós cujo atributo Nome seja IGUAL a “Ana”

9.png 9.png

```
// Restringir a consulta para selecionar todos os nós com a propriedade “Nome” igual “Ana”
MATCH (a)
WHERE a.Nome = “Ana”
RETURN a.Nome, a.RG
```

Figura 6 – Exemplos de consulta com restrições de retorno (clausula WHERE)

Para excluir ou alterar o valor da propriedade de um nó ou relacionamento, primeiramente temos que realizar a consulta que retorne o nó ou relacionamento para alteração ou exclusão e em seguida realizar a operação. O comando para realizar a alteração é o SET e para exclusão é o DELETE. O exemplo na figura representa a alteração e um exemplo de exclusão. O primeiro exemplo na figura 7 é uma alteração (SET), neste exemplo está sendo alterado a propriedade RG para “28.058.154-X” do nó com a propriedade Nome igual a “Ana”.O segundo exemplo mostra a exclusão (DELETE) do nó com propriedade RG igual a “28.058.154-X”.

Exemplos de alteração e exclusão utilizando Cypher

8 Teoria dos Grafos

Segundo (SZWARCFITR, 1988, p. 19), a Teoria dos Grafos surgiu em 1736, através do estudo e criação de um algoritmo para o problema da ponte de Königsberg de Euler,

```
// alterar o RG do nós com a propriedade "Nome" igual "Ana"
MATCH (a)
WHERE a.Nome = "Ana"
SETa .RG = "28.058.154-X";

// excluir o nó com a Propriedade RG = "28.058.154-X"
MATCH (a)
WHERE a.RG = "28.058.154-X"
DELETE a;
```

10.png 10.png

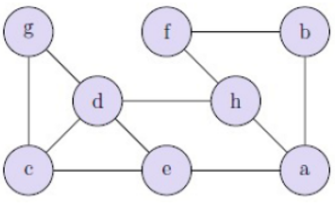
Figura 7 – Exemplos de alteração e exclusão utilizando Cypher

que consistia na obtenção de uma caminho entre determinados locais (passando por pontes existentes em duas ilhas) voltando para a região inicial sem passar duas vezes pela mesma ponte. Euler utilizou os grafos para chegar à conclusão de que somente tendo um par de pontes em cada região é que se cumpre este propósito. A Teoria foi com o passar do tempo sendo aperfeiçoada por De Morgan (1852), e por volta de 1930 por Kuratowski, König, Menger, entre outros cientistas.

Grafo é representado por G (V, A), onde V (Vértice) não pode ser vazio. O conjunto A (Aresta) é formado por pares de V não ordenados. Ficando desta forma (a 2 A): a = (v, w), onde o v e o w são vértices das extremidades. (SZWARCFITR, 1988, p. 35).

De acordo com (ROCHA, 2013), a ordem de um grafo é determinada pela quantidade de vértices que ele possui |V | e o tamanho equivale ao número de arestas do grafo |A|. O grau de um vértice é determinado pela quantidade de arestas conectadas ao mesmo e o caminho consiste em uma sequência de vértices (v1, ..., vn) conectados por arestas.

Os grafos podem ser representados de forma gráfica, como na Figura8, onde as arestas são equivalentes a linhas que unem os vértices que por sua vez são representados por pontos ou círculos. Sendo que se utiliza a definição grafo para a sua representação geométrica também para facilitação da exposição. (SZWARCFITR, 1988, p. 35).



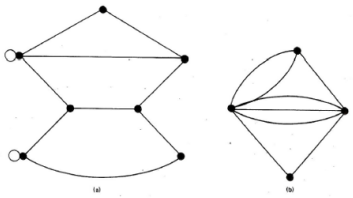
Fonte: (ROCHA, 2013)

Figura 8 – Exemplo de Grafo

Segundo (SZWARCFITR, 1988, p. 39), a distância entre dois vértices é o menor caminho entre eles d(v,w). Se uma aresta é do tipo a = (v, v) então será chamada de laço, pois as extremidades são formadas por um par idêntico de vértices, como ilustrado na

De acordo com (SZWARCFITR, 1988, p. 37), é possível existir mais de uma aresta

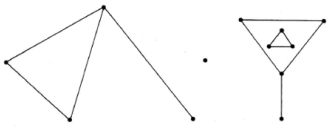
entre o mesmo par de vértices, sendo que esta estrutura se chama multigrafo, ilustrado na Figura9.



Fonte: (SZWARCFITR, 1988, p. 37)

Figura 9 – Grafos com laços e multigrafo

Um grafo onde existe um caminho entre todas os vértices do G é chamado de conexo, porém se não existe este caminho em um ou mais vértices, será chamado de desconexo, ou ainda totalmente desconexo se não houver nenhuma aresta entre os vértices do grafo. (SZWARCFITR, 1988, p. 38), como ilustrado na Figura10.



Fonte: (SZWARCFITR, 1988, p. 38)

Figura 10 – Grafo desconexo

Segundo (FURTADO, 1973, p. 2), nos grafos dirigidos há um vértice inicial e outro terminal na aresta (denotando direção). Pode-se atribuir designações ou valores aos vértices e às arestas, como pode ser observado na Figura11.

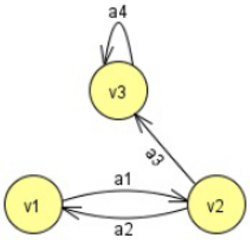


Figura 11 – Grafo Direcional

Diversas operações podem ser feitas em um grafo, tais como Inserção de algum elemento (aresta ou vértice), a Retirada de algum elemento, a Rotação que é a modificação de uma designação, Fusão e Redesignação que equivale a união de dois vértices e a Inversão que é alteração do sentido das arestas. (FURTADO, 1973, p. 3).

Pode ser feito também operações binárias, as quais são análogas às teorias dos conjuntos, como a União (G1 G2), a Interseção (G1 G2) e a Diferença (G1 – G2).

A operação de soma (G1 + G2) resulta em um grafo que possuí vértices com pares ordenados (x, y) com vértices x G1, y G2 e quanto às arestas, se (x, x') G1 e (y, y') G2 então as arestas ((x, y), (x, y')) e ((x, y), (x', y)) pertencem a soma. (FURTADO, 1973, p. 2).

O produto (G1 G2) resulta em um grafo que possuí vértices como os da soma e quanto às arestas, se (x, x') G1 e (y, y') G2 então as arestas ((x, y), (x', y')) pertencem ao produto. (FURTADO, 1973, p. 2).

A respeito dos subconjuntos dos grafos, tendo em vista um grafo G = (V,A), que G" = (V', A'), se V' = V e A' A então será um Grafo Parcial de G. Se V' V , e A' é o subconjunto de A formado pelas arestas de A cujas extremidades pertencem a V' então será um Subgrafo de G. Se V' V e V e A' é o subconjunto de A formado somente por um subconjunto das arestas de A que tenha as extremidades em V' então será um Subgrafo Parcial de G. (FURTADO, 1973, p. 2).

Considerações finais

Depois de uma análise dos SGBDGs foi utilizado o Neo4j para apoiar a implementação deste trabalho. A escolha recaiu sobre ele por ser um SGBDG que respeita as tão conhecidas propriedades ACID.

E por fim, foi apresentado a linguagem de consulta a grafos Cypher, linguagem utilizada pelo Neo4j similar ao SQL.

Graph-Oriented Database Management System

Valdeney André Matos Júnior N: 201633840029
Mauriocio Rodrigues Mendes N: 201633840002

Tucuruí, 2019

Abstract

The paper addresses, as the main theme, Database Management Systems (DBMS), its main and functional characteristics as well as some definitions; The DBMS is constituted by a method to elucidate the main results, since the graph theory is used for a better understanding on the subject, since this is a theoretical basis for the implementation of the system. Other sources of research are, mainly, websites and books, it is tried to explain the subject concisely, elucidating main points on the subject

Key-words: Database. Graphs. Neo4j.

[1] Aws Amazon Corporation. (23 de Junho de 2019). Banco de Dados NoSQL - Amazon Web Services. Fonte: aws.amazon.com: https://aws.amazon.com/pt/nosql/

[2] Date, C. J. (1989). Introdução a Sistemas de Banco de Dados. Em C. J. Date, Introdução a Sistemas de Banco de Dados (pp. 26-28). Rio de Janeiro: Editora Campos Ltda

[3] Groffe, R. (30 de Setembro de 2016). Banco de dados SQL: Uma visão geral. Fonte: IMasters: https://imasters.com.br/banco-de-dados/bancos-de-dados-nosql-uma-visao-geral

[4] Machado, F. N. (2014). Banco de Dados: Projeto e Implementação. Em F. N. Machado, Banco de Dados: Projeto e Implementação (pp. 39-42). São Paulo: Editora Érica

[5] Meirelles, M. (15 de Dezembro de 2015). Uma gentil introdução ao uso de banco de dados orientados a grafos com Neo4j. Fonte: Medium.com: https://medium.com/accendis-tech/uma-gentil-introdução-ao-uso-de-banco-de-dados-orientados-a-grafos-com-neo4j-ca148df2d352

[6] ELMASRI, R.; NAVATHE, S. Sistemas de banco de dados, 4a Edição. [S.l.]: ADDISON WESLEY BRA, 2005

[7] ELMASRI, R.; NAVATHE, S. Sistemas de banco de dados, 4a Edição. [S.l.]: ADDISON WESLEY BRA, 2005

[8] PENTEADO, R. R. et al. Um estudo sobre bancos de dados em grafos nativos. 2014.

[9] SZWARCFITR, J. L. Grafos e algoritmos computacionais, 2a Edição. Rio de Janeiro: Editora Campus, 1988