**ANA CAROLINA DA SILVA**
**JOÃO PAULO PIRES**

# BANCO DE DADOS RELACIONAL *VERSUS* BANCO DE DADOS ORIENTADO A GRAFOS APLICADOS A REDES SOCIAIS

**UNIVERSIDADE DO VALE DO SAPUCAÍ**
**POUSO ALEGRE**
**2013**

**ANA CAROLINA DA SILVA**
**JOÃO PAULO PIRES**

# BANCO DE DADOS RELACIONAL *VERSUS* BANCO DE DADOS ORIENTADO A GRAFOS APLICADOS A REDES SOCIAIS

Trabalho de conclusão de curso defendido e aprovado em ___/___/_____ pela banca examinadora constituída pelos professores:

_____
ROBERTO RIBEIRO ROCHA
Orientador

_____
MÁRCIO EMÍLIO C. V. AZEVEDO
Examinador

_____
CRISHNA IRION
Examinadora

Aos nossos familiares, amigos e todos os
que contribuíram com esta pesquisa.

**AGRADECIMENTOS**

Agradecemos primeiramente a Deus por nos guiar para o sucesso. Ao nosso orientador Roberto Ribeiro Rocha pelo excelente apoio em todas as fases do projeto, acreditando nos resultados positivos. A professora Joelma Pereira de Faria e ao professor José Luiz da Silva pelo grande auxílio textual, e a todos os demais professores responsáveis por nossa formação.

De Ana Carolina:

Agradeço a minha mãe Patrícia e ao meu pai Deusdedit (in memorian) pela minha educação A minha mãe por cuidar de mim todos os dias, não fosse por meus pais, não teria me tornado a pessoa que sou e é graças a eles mais essa conquista.

Aos meus irmãos agradeço por sempre acreditarem em mim e ao meu namorado Felipe pela paciência, o grande apoio e por podermos dividir essa etapa de nossas vidas juntos.

Sou muito grata também ao meu parceiro de projeto João Paulo por seu empenho, por acreditar que juntos faríamos um belo trabalho e por poder compartilhar essa vitória. Aos nossos colegas de classe, os quais tive o prazer de conhecer e dividir esses quatro anos de minha vida, levo cada um em meu coração.

Enfim, a todos que estiveram ao meu lado acreditando em mim, aos que me acompanharam e estão me acompanhando nesse momento, amigos, colegas de trabalho, familiares, muito obrigada a todos!

De João Paulo:

Agradeço aos meus pais por tornarem possível a concretização de mais esse projeto em minha vida.

Agradeço minha amiga e parceira de projeto Ana Carolina pela dedicação e trabalho em equipe e a todos os colegas de sala pelos bons momentos compartilhados.

"O sucesso é a soma de pequenos esforços - repetidos dia sim, e no outro dia também."
(Robert Collier)

PIRES, João Paulo; SILVA, Ana Carolina. **Banco de dados relacional *versus* banco de dados orientado a grafos aplicados a redes sociais**. Trabalho de Conclusão de Curso. Graduação. Sistemas de Informação. Universidade do Vale do Sapucaí. Pouso Alegre – MG. 2013.

## RESUMO

Esta pesquisa acadêmica mostra um estudo comparativo entre a modelagem de banco de dados relacional e orientado a grafos, utilizando as principais informações presentes em um ambiente de redes sociais. Para obter os resultados, foram utilizados os sistemas de gerenciamento de banco de dados PostgreSQL e Neo4J para o modelo relacional e orientado a grafos respectivamente. Foi desenvolvido um algoritmo na linguagem Java com o auxílio da ferramenta Eclipse para a inserção de dados aleatórios nos bancos. Com as bases de dados populadas, seis consultas foram executadas para busca de informações específicas nos bancos de dados utilizando as linguagens correspondentes SQL e Cypher. Esta pesquisa é do tipo exploratória, pois mostra as diferentes características dos bancos. Foram feitos comparativos do tempo de resposta de ambos os bancos de dados, tornando esta pesquisa de grande relevância técnica e teórica por contribuir com a exploração de novos recursos de persistência de dados. Com isso, os objetivos propostos no trabalho foram concluídos.

Palavras-chave: Banco de dados. Relacional. Orientado a grafos. Neo4J. Cypher. SQL.

PIRES, João Paulo; SILVA, Ana Carolina. **Relational Database *versus* Graph Database applied to social networks**. Trabalho de Conclusão de Curso. Graduação. Sistemas de Informação. Universidade do Vale do Sapucaí. Pouso Alegre – MG. 2013.

## ABSTRACT

This academic research shows a comparative study between the modeling of Relational Database and Graph Database using the main information present in social networks environment. To get the results, the database management systems PostgreSQL and Neo4J were used to the relational model and graph model respectively. An algorithm was developed in Java language with the help of Eclipse tool for inserting random data to the databases. With the databases filled, six queries were executed to search for specific information in the databases using the corresponding languages SQL and Cypher. This research is exploratory, because it shows the different characteristics of databases. Comparisons were made of the response time of both databases, what makes this research highly relevant technical and theoretical to contribute to the exploration of new resources for data persistence. Thus, the objectives proposed in this research were completed.

Keywords: Databases. Relational. Graph Database. Neo4J. Cypher. SQL.

**LISTA DE FIGURAS**

**LISTA DE TABELAS**

**LISTA DE ABREVIATURAS E SIGLAS**

| | |
|---|---|
| ACID | *Atomicity, Consistency, Isolation, Durability* |
| ANSI | *American National Standard Institute* |
| ASCII | *American Standard Code for Information Interchange* |
| API | *Application Programming Interface* |
| BASE | *Basically Avaliable, Soft state, Eventual consistency* |
| BSD | *Berkeley Software Distribution* |
| CAP | *Consistency, Availability e Partition tolerance* |
| DBA | *Database Administrator* |
| GB | *Giga Byte* |
| GRAPHDB | *Graph Database* |
| HTML | *HyperText Markup Language* |
| HTTP | *HyperText Transfer Protocol* |
| IBM | *International Business Machines* |
| IDE | *Integrated Development Environment* |
| JAX-RS | *Java API for RESTful Services* |
| JDK | *Java Developement Kit* |
| JSON | *JavaScript Object Notation* |
| JVM | *Java Virtual Machine* |
| NOSQL | *Not Only SQL* |
| RDF | *Resource Description Framework* |
| REST | *Representational State Transfer* |
| SEQUEL | *Structured English Query Language* |
| SGBD | *Sistema de Gerenciamento de Banco de Dados* |
| SPARQL | *SPARQL Protocol and RDF Query Language* |
| SQL | *Structured Query Language* |
| SSD | *Solid State Drive* |
| UNIVÁS | *Universidade do vale do Sapucaí* |

**SUMÁRIO**

# INTRODUÇÃO

O modelo de banco relacional está estabelecido como a principal arquitetura de dados e é utilizado nos mais diversos sistemas das mais diversas áreas desde sua criação em 1970. Suas características básicas são as tabelas, sua forma de garantir a integridade das informações a partir da restrição de integridade e seu processo de normalização com aplicação de regras sobre as tabelas do banco (Brito, 2010).

Os bancos de dados relacionais oferecem recursos que facilitam na hora do desenvolvimento de aplicações. Os dados são validados, verifica-se e garante-se a integridade dos mesmos, há controle de concorrência, as falhas são recuperáveis, oferece-se segurança, controla as transações e, dependendo do volume de dados e de sua complexidade pode se obter consultas otimizadas, dentre outros recursos que permitem ao desenvolvedor manter seu foco na aplicação (Brito, 2010).

Os benefícios oferecidos pelos bancos relacionais evidenciam sua grande utilização e domínio no mercado de gerenciamento de dados por tantas décadas. Porém, hoje utiliza-se a *Web* 2.0, termo que se refere a segunda geração da *World Wide Web* (Folha.com, 2006).

Na *Web* 2.0, o volume e o fluxo de dados são cada vez maiores, tornando a escalabilidade um problema para o banco relacional, principalmente por seu modelo estruturado. Há também o problema de desempenho. Para enfrentá-lo, existem duas alternativas: utilizar um servidor de melhor configuração e desempenho ou aumentar o número de servidores. Porém, se o número de usuários continua a crescer, nenhuma dessas soluções seria suficiente (Brito, 2010).

Com o crescimento da complexidade e da necessidade de respostas rápidas e confiáveis, os bancos de dados NoSQL (*Not only SQL*) ganham cada vez mais notoriedade. Dentre os diversos modelos aplicados, temos o banco de dados orientado a grafos, o qual baseia-se na teoria de grafos iniciada por Leonhard Euler, que em 1736 resolveu o problema das pontes de Königsberg (Ignatowicz, 2012).

A proposta do banco de dados NoSQL é exatamente atender ao gerenciamento de grandes volumes de dados, em especial os não-relacionais, que necessitam de disponibilidade e escalabilidade. O banco de dados NoSQL possui certas características que o tornam diferente do banco SQL e adequado para armazenar grandes volumes de dados (Lóscio; Oliveira; Pontes, 2011).

Bancos de dados orientados a grafos possuem: a) escalabilidade horizontal, pois de acordo com o crescimento do volume de dados é possível aumentar o número de servidores, apesar de nem todas as versões disponibilizarem, dependendo do banco está disponível apenas na versão paga; b) ausência de esquema (esquema flexível), uma vez que com a ausência de esquema se obtém uma melhora na escalabilidade e disponibilidade dos dados, porém não se garante a integridade dos mesmos e c) suporte nativo a replicação, já que outra forma de garantir escalabilidade, por meio da replicação nativa diminui o tempo na recuperação de informações.

Outra característica marcante de um banco de dados em grafos é a API simples para acesso aos dados. Seu objetivo está em oferecer uma forma eficiente de acesso, não importando como os dados são armazenados e sim como serão recuperados. De acordo com o teorema CAP, entre consistência, disponibilidade e tolerância, só se pode garantir duas dessas três propriedades,. No contexto da *web* costuma-se privilegiar disponibilidade e tolerância à partição, por esse motivo o banco orientado a grafos possui consistência eventual (Lóscio; Oliveira; Pontes, 2011).

O modelo de banco de dados orientado a grafos[1] é composto de: a) nós (representado graficamente por um círculo) que são os vértices de um grafo; b) relacionamentos (representados graficamente por linhas), que são arestas que conectam um vértice ao outro e c) propriedades, que são atributos dos vértices e dos relacionamentos (Ex.: amigo, parente, etc...). Pode-se visualizá-lo como um grafo de múltiplas arestas, os vértices podem se conectar por uma ou mais arestas ao mesmo tempo (Lóscio; Oliveira; Pontes, 2011).

Utilizando um modelo de grafos em consultas complexas, fica evidente sua vantagem em relação ao modelo relacional (que para uma situação deste tipo pode ser bem mais trabalhoso) e ainda ganha no desempenho de suas aplicações (Lóscio; Oliveira; Pontes, 2011).

Segundo Brites (2012), o Neo4j é um exemplo de banco orientado a grafos que mantém o benefício de transação do banco relacional. O NoSQL é uma solução alternativa no

[1] Os detalhes do banco de dados orientado a grafos serão apresentados na sessão 2.7.

gerenciamento de dados em que o banco relacional não é eficiente, sendo assim não é de intenção substituí-lo. De uma maneira prática, Brites (2012) mostra casos em que os bancos de dados NoSQL foram aplicados em empresas que possuíam sérios problemas em relação ao tratamento dos dados devido ao volume que os mesmos geravam e os resultados da implementação foram favoráveis.

Matsushita e Nguessan (2011) demonstraram que um banco de dados orientado a grafos se comunica perfeitamente com aplicações para dispositivos móveis que utilizam o *framework* iTi (Integração da Tecnologia da Informação) como meio de interação com o usuário e que a construção de uma rede social que utiliza o mesmo *framework* deve também seguir a modelagem de dados com grafos para que o intenso tráfego não atrapalhe no desempenho da aplicação.

Diana e Gerosa (2010) afirmam que, devido a uma grande quantidade de dados gerados, processados e armazenados na *web* 2.0, há a necessidade de se conhecer tecnologias de armazenamento que tratem também os dados não relacionais. Os bancos de dados NoSQL, apesar de serem recentes e ainda estarem sendo testados em ambientes reais de produção, possuem grandes chances de um futuro promissor. Um exemplo de que essa tecnologia não será passageira como uma simples moda, é o seu uso em operações de escala global como o Facebook e Twitter.

O objetivo geral desta pesquisa é criar um modelo de banco de dados relacional e orientado a grafos para execução de consultas baseadas em redes sociais, apresentando as diferenças entre as duas arquiteturas.

Os objetivos específicos desta pesquisa são:

- Apresentar o modelo de dados orientado a grafos, demonstrando as suas características;
- Utilizar servidores que forneçam recursos de modelagem e execuções de *scripts* para consultas às bases de dados.
- Com o resultado, comparar o desempenho, recursos necessários e complexidade de modelagem.

Atualmente, existem diversas ferramentas para a manipulação de dados baseados em grafos e uma delas é Neo4J, o qual será utilizado durante todo o desenvolvimento, apresentando suas vantagens, desvantagens e em qual contexto ela se enquadra.

Esta pesquisa irá demonstrar, em forma de resultados, qual a implementação de estrutura de dados mais adequada a se seguir em aplicações que utilizam o conceito de redes

sociais. Essas redes possuem indivíduos que podem, ou não, ter algum tipo de relacionamento com outro indivíduo, e esses mesmos indivíduos terão diferentes tipos de relações entre si. Essa complexidade dos sistemas atuais vem crescendo no decorrer dos anos. É preciso abstrair o mundo real para elaborar uma solução que será implementada em um *software*. A complexidade é encarada como um desafio pelos desenvolvedores que, agora, podem contar com a utilização de grafos em seus projetos. De acordo com Robinson, Webber e Eifren (2013, p.3):

> [2] Graph databases have been proven to solve some of the more relevant data management challenges of today, including important problems in the areas of social networking, master data management, geospatial, recommendations engines, and more.

É certo que, em todo o processo de desenvolvimento de sistemas, desde a análise de requisitos até a sua implantação, tanto o cliente quanto o desenvolvedor têm em mente um projeto em que os dados sejam entregues a quem os solicita de maneira ágil, confiável e correta. Mas depende somente do desenvolvedor adotar a melhor forma para que essas exigências sejam cumpridas com êxito. Para se chegar, então, a uma estrutura adequada de implementação, devem-se realizar estudos aprofundados em cada tecnologia, analisando os resultados de acordo com o que se pretende utilizar no sistema.

Apesar da grande importância do modelo de dados grafo, ele ainda é novo e poucas universidades o exploram, o que dificulta o acesso a materiais para pesquisa. Esta pesquisa trará novos recursos para que alunos e desenvolvedores ampliem seus conhecimentos e, ainda, contribuirá para o enriquecimento de conteúdo técnico de instituições de ensino que possuem cursos ligados à tecnologia da informação, pois com a disponibilização deste material, as disciplinas de estrutura de dados e banco de dados terão conteúdo para abordarem de maneira mais específica, a arquitetura de grafos, bem como o Neo4J, que é sua principal ferramenta de elaboração.

Diante das características positivas que o banco de dados com grafos possui, como a facilidade de modelar uma estrutura de dados que se assemelhe o máximo possível ao mundo real e da alta escalabilidade do mesmo, é que esta pesquisa torna-se relevante para uma aplicação prática sobre o assunto, aliado ao crescimento de recursos para buscas em redes sociais, como interesses pessoais, relacionamentos, ciclos de amizades e recomendações de páginas. Estas características são encontradas no Facebook que foi uma das redes sociais mais

[2] Bancos de dados orientados a grafos vêm provando a resolução dos mais relevantes desafios da gerência de dados, incluindo problemas importantes nas áreas de redes sociais, gerenciamento de dados, geoespacial, sistemas de recomendações, entre outros (Traduzido pelos autores).

acessadas em 2010 e estava presente em um a cada quatro sites acessadas nos Estados Unidos (Dougherty, 2010).

Após a divulgação desta pesquisa, desenvolvedores especializados em banco de dados, mais conhecidos como DBA's (*Database Administrator*) poderão obter conhecimento mais aprofundado no modelo de dados utilizando grafos e não se prenderão ao relacional, tornando as aplicações ainda mais precisas com relação à persistência. Ao final da pesquisa e a apresentação dos resultados obtidos, os DBA's também saberão onde, quando e como utilizar as estruturas com grafos bem como o modelo relacional, pois as vantagens e desvantagens de cada modelo nortearão o profissional a escolher a melhor solução para casos específicos.

Além do benefício técnico-acadêmico, é possível destacar a grande importância deste estudo para futuros profissionais de desenvolvimento de *software*, que não mais se limitarão a projetar sistemas que dependem de uma estrutura fixa, sem dinâmica e que demandam maior gastos com equipamentos como são os bancos de dados relacionais, mas poderão aplicar o conhecimento adquirido na prática em empresas que ainda hoje temem adotar novos padrões em suas aplicações por falta, muitas vezes, de qualificação de mão-de-obra.

Esta pesquisa está dividida em quatro capítulos. No capítulo 2 serão apresentadas as teorias e tecnologias aplicadas durante o desenvolvimento. No capítulo 3 serão explorados os processos, procedimentos e contexto da pesquisa. O capítulo 4, os resultados da implementação serão apresentados e discutidos.

# 2 QUADRO TEÓRICO

Nesse capítulo são apresentadas as teorias e tecnologias adotadas para o desenvolvimento desta pesquisa, demonstrando as características de cada uma, bem como sua utilidade e uso neste trabalho.

## 2.1 Teoria de grafos

A teoria de grafos é uma extensa matéria que abrange aspectos matemáticos e computacionais. Ela está presente em muitos problemas práticos do cotidiano, como por exemplo, a análise da hierarquia de uma empresa, o processo de seleção em um vestibular ou o cálculo de rotas para descobrir o percurso mais curto em uma viagem (Ignatowicz, 2012).

Em 1736, o matemático suíço Leonhard Euler resolveu um problema prático para a época. O algoritmo desenvolvido por Euler ficou conhecido como as sete pontes de Königsberg. A definição do problema era descobrir se era possível visitar todos os pontos da cidade atravessando as sete pontes somente uma vez. A partir de uma representação utilizando grafos **Fig. 1**, Euler provou que só era possível cruzar todos os pontos se a quantidade de pontes fossem um número par (Morais, 2013).

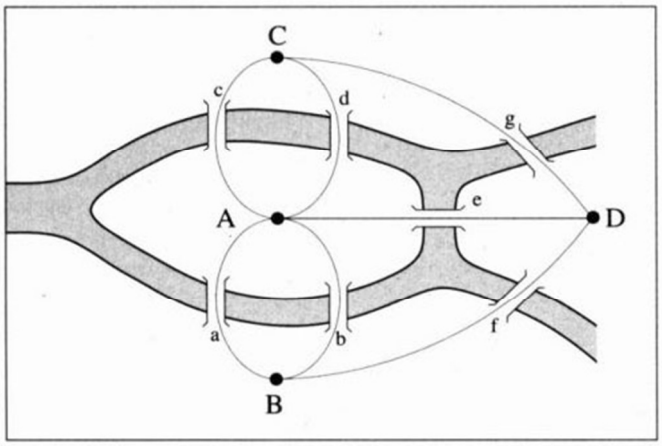


**Fig. 1** - Pontes de Königsberg. **Fonte:** Dantas (2010).

Um grafo G é composto por um conjunto de vértices V, que são interligados por arestas A. Sendo assim, os vértices poderão se conectar com arestas direcionais ou não. Os grafos que não possuem sentidos em uma aresta, por exemplo, $V^1$ --- $V^2$, os vértices $V^1$ e $V^2$ possuem uma relação sem uma direção; esse tipo de ligação é conhecida como relações

simétricas. Já utilizando um segundo exemplo: $V^1 \rightarrow V^2$, a aresta possui um sentido único, partindo de $V^1$ em direção à $V^2$, essa relação denomina-se relação não simétrica. Um vértice pode ser denominado sucessor nos casos em que há ligação não simétrica (ou grafo dirigido), ou seja, $V^2$ é sucessor de $V^1$. Já vértices antecessores são aplicados de maneira contrária ao sucessor, ou seja, $V^1$ é antecessor de $V^2$ (Morais, 2013).

Apresenta-se na **Fig. 2**, um grafo com três vértices conectados por arestas. Esse exemplo é representado matematicamente da seguinte maneira: $V = \{V^1,V^2,V^3\}$ e $A = \{(V^1,V^2) , (V^1,V^3) , (V^2,V^1) , (V^3,V^1)\}$, V é o conjunto de vértices e A o conjunto de arestas, como o grafo é não simétrico, o sentido em que os vértices são conectados é igual para todos (Feofiloff *et al*; 2011).

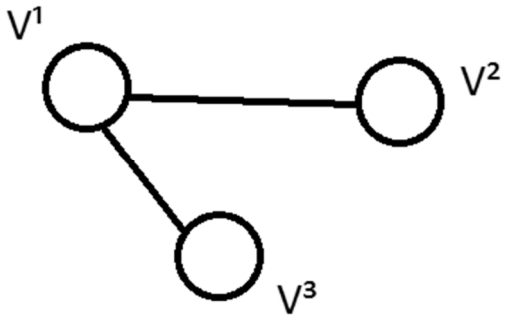


**Fig. 2** - Exemplo de grafo simétrico.
**Fonte**: Elaborado pelos autores (2013).

O número de arestas que cada vértice possui é denominado grau. Em vértices não simétricos, o grau é o número de arestas presentes, já em vértices simétricos, o grau é a soma de arestas de entrada e saída. Desta forma, se um ou mais vértices possuírem uma aresta que sai e chega ao mesmo vértice, serão considerados como grau 2. Nos casos em que não há arestas conectadas ao vértice (denominados isolados), o grau será igual a zero. Grafos que possuem o mesmo grau são denominados regulares. (Morais, 2013).

Segundo Feofiloff *et al*; (2011), denomina-se caminho, todo o grafo que possui vértices que admite uma permutação de forma que (v1v2, v2v3,...vn-1vn), desta maneira, os vértices são interligados entre si porém as extremidades não se conectam. Já um circuito é um grafo que possui um número de vértices maior ou igual a três e que o conjunto destes formam uma permutação (v1v2, v2v3...vn-1vn, vnv1). Neste caso, as extremidades do grafo se conectam.

Grafos também podem ser classificados quanto à forma em que os vértices são interligados entre si. Um conjunto que possui caminhos que conectem os vértices são chamados de grafos conexos, já grafos que possuem vértices que não se conectam são

denominados grafos desconexos (Morais, 2013). A **Fig.3** abaixo ilustra os dois tipos de grafos. É possível notar que os vértices X5 e X6 do grafo desconexo não possuem nenhum tipo de ligação com os demais vértices, diferentemente do grafo conexo.

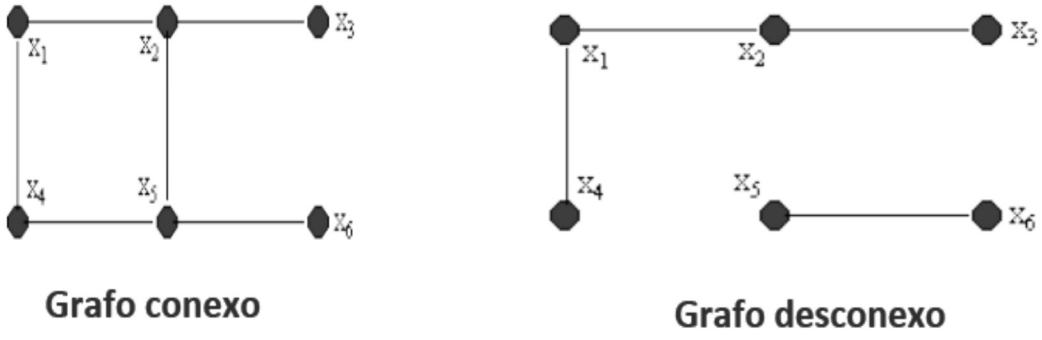


**Fig. 3** - Grafo conexo e desconexo. **Fonte:** Morais (2013).

A partir de um grafo principal, pode haver um ou mais grafos que utilizem os mesmos vértices ou arestas, esse conceito é conhecido como subgrafos. Feofiloff *et al*; (2011, p17) definem subgrafos como sendo "qualquer grafo H tal que V (H) <= V (G) e A(H) <= A(G). Um subgrafo H de G é próprio se V (H) != V (G) ou A(H) != A(G)".

Para Morais (2013), os subgrafos podem ser classificados como abrangentes ou induzidos. Nos casos em que os vértices do grafo original forem totalmente empregados, mas não possuirem todas as arestas que interliguem os mesmos, o conjunto é denominado abrangente; já se, somente partes do total de vértices do grafo original serem mantidos mas as ligações forem as mesmas, o subgrafo é chamando de induzido.

Para se obter informações específicas ou realizar cálculos nas ligações entre os vértices, as arestas (ou arcos) podem trazer valores associados a elas. Esta característica é conhecida como grafos ponderados (Morais, 2013).

## 2.2 Redes Sociais

Entende-se por rede social da internet, uma rede que não conecta apenas computadores, mas também pessoas. A comunicação realizada por meio do computador permite amplificar a capacidade das pessoas de se comunicarem e se manterem conectadas (Recuero, 2009).

Schelp (2009) define as redes sociais da internet como uma página na rede utilizada para a publicação on-line de um perfil de si próprio, com fotos e informações pessoais, sendo possível encontrar e montar uma lista de amigos que tenham perfis no mesmo site. Como na

vida real, em uma praça ou clube é um espaço virtual para compartilhamento de informações e conhecimento, postagem de fotos e vídeos, troca de experiências de vida e do cotidiano, compartilhamento de gostos pessoais e em comum.

O estudo das redes sociais vem de longo tempo, em todo o século XX uma das mudanças na ciência é o estudo da sociedade pelo conceito de rede (Recuero, 2009).

A partir do matemático Leonard Euler com a publicação de seu artigo sobre o enigma das pontes de Königsberg, conforme item 2.1, aliado ao surgimento do conceito de rede, os estudos sobre redes sociais ganharam força (Recuero, 2009).

Assim como na teoria de grafos, as redes sociais possuem dois elementos: os vértices e as arestas. Os atores (pessoas ou organizações) são os vértices da rede, e suas conexões (interações) as arestas (Recuero, 2009).

- Atores: Como se trata do mundo virtual os atores podem ser um perfil no Twitter ou no Facebook. Cada página representa um nó na rede, mesmo quando mantida por um grupo de pessoas, como as comunidades do Facebook por exemplo. O que permite as redes sociais estarem no mundo virtual é a forma como o ator se apresenta, seu individualismo, mostrando que há alguém falando por meio da rede (Recuero, 2009).
- Conexões: As conexões da rede, suas arestas, são as interações sociais entre os atores, seus laços sociais. Por meio dos rastros deixados pelos atores é que se permite perceber suas interações. Comentários, "curtir" e "compartilhar" do Facebook, "seguir" do Twitter, são exemplos de rastros, eles são percebíveis independentes do tempo e do espaço em que ocorreu a interação, só deixarão de ser quando o ator excluir o comentário ou a página, caso ao contrário ficarão para sempre representados no espaço virtual (Recuero, 2009).

O que caracteriza as redes sociais e torna suas interações diferentes é o fato de se usar uma plataforma informatizada, a princípio para as pessoas e atores se encontrarem virtualmente com outras pessoas conhecidas na vida real. Antigamente as pessoas procuravam salas de bate papo, por exemplo, para se conhecerem, trocavam informações virtualmente e marcavam encontros presenciais, mantendo uma relação também off-line. Já nas redes sociais as pessoas buscam conhecidos da vida real no mundo virtual e acabam mantendo a relação na maior parte das vezes on-line (Aguiar, 2007).

Segundo Aguiar (2007), os primeiros sites do gênero Redes Sociais surgiram por volta de 1990 baseados nas ligações diretas entre colegas em uma sala de aula na escola, e indiretas com os amigos dos amigos e conhecidos.

Para Machado e Tijiboy (2005), o conceito de rede social que representa a atualidade e com o desenvolvimento de novas ferramentas de alta tecnologia a partir da internet, vem se fortalecendo cada dia mais, atingindo várias áreas de conhecimento como a econômica, científica, cultural entre outros (Machado; Tijiboy, 2005). São tantas pessoas conectadas, independentes do tempo e do espaço, que aumenta a capacidade de difusão da informação (Recuero, 2009). Na área econômica as redes sociais são de interesse das empresas como um espaço de propaganda, onde vão negociar e vender seus produtos e serviços (Machado; Tijiboy, 2005).

A rede e a comunicação mediada pelo computador têm sido utilizadas também para dar início a movimentos sociais e culturais como, por exemplo, a luta dos direitos humanos e ambientalistas. Já no campo da educação podem ser exploradas como um espaço para debates e argumentação. Ainda uma grande parte de pessoas não são adeptas ao uso do computador e das redes sociais para se comunicarem, mas, contudo não param de crescer o número de conectados e isso vem se tornando um hábito na vida cotidiana dessas pessoas (Machado; Tijiboy, 2005).

Os brasileiros são os que mais fazem parte dessa gama de internautas e colaboradores para o aumento de usuários nas redes sociais. Schelp (2009, s. p.) diz que: “Em nenhum outro país as redes sociais on-line têm alcance tão grande quanto no Brasil, com uma audiência mensal de 29 milhões de pessoas”.

Entre os países que mais utilizam as redes sociais para reunir o maior número de contatos em sua lista de amigos, além do Brasil, estão também a Hungria e Filipinas (Schelp, 2009).

Para melhor entendimento dos sites de Redes Sociais citados nesta pesquisa, apresenta-se, a seguir, um breve histórico do Facebook e Twitter.

- Facebook: O americano Mark Zuckerberg desenvolveu o sistema Facebook enquanto estudava em Harvard, motivado a criar uma rede para jovens que estão saindo do ensino médio e entrando para a faculdade, devido a esse período representar grandes mudanças de vida, já que nos Estados Unidos, a maioria dos estudantes acaba mudando de cidade. A princípio o sistema só era permitido para estudantes de Harvard. O Facebook é considerado o sistema de

maior privacidade, pois permite que apenas usuários da mesma rede, amigos, vejam o perfil uns dos outros (Recuero, 2009).

- Twitter: Jack Dorsey, Biz Stone e Evan Williams apresentaram o Twitter como projeto da empresa Odeo em 2006. O site permite que seus usuários publiquem textos pequenos, de no máximo 140 caracteres, respondendo o que estão fazendo no momento. Sua estrutura se dá por seguidores e usuários sendo escolha do dono do perfil. A principal característica do Twitter é oferecer sua API para que ferramentas interajam com o sistema (Recuero, 2009).

## 2.3 Banco de dados relacional

Banco de dados é uma coleção de dados com informações importantes para uma organização ou empresa (Silberschatz*al*; 2006). Segundo Machado e Abreu (2004), informação é o conhecimento do que se vai analisar enquanto dado é sua representação em registro, que pode ser em uma folha de papel como também em um banco de dados, ou em qualquer outra forma que se possa consultar posteriormente.

Um Sistema de Gerenciamento de Banco de Dados (SGBD) contém essa coleção de dados que se relacionam entre si e vários programas utilizados para acessá-los. Além disso, é preciso definir uma estrutura de armazenamento, oferecer um mecanismo para manipulá-los e garantir que as informações estão contidas em segurança, podendo ser acessadas apenas por pessoas autorizadas e de forma a manter sua integridade (Date; 2003).

Bancos, linhas aéreas, universidades, transações de cartão de crédito, telecomunicação, finanças, vendas, revendedores *on-line*, indústria e recursos humanos são algumas aplicações que utilizam SGBD, pode-se perceber que é grande o seu campo de atuação, chegando a quase todas as empresas (Silberschatz *et al*; 2006).

Em 1960 surgiu o primeiro SGBD comercial a partir dos sistemas de arquivos de armazenamento em disco utilizados na época e que não permitiam o controle de acesso por mais de um usuário ou processo (Takai *et al*; 2005). Nas últimas décadas do século XX ocorreu um grande aumento no uso de bancos de dados nas empresas, e, a princípio, eram poucas as pessoas que tinham acesso direto, a maioria o fazia indiretamente. Com o advento da internet no final da década de 1990 possibilitando mudanças na interface dos sistemas de manipulação dos dados permitindo uma melhor interação dos usuários com o banco, aumentou o número de pessoas com acesso direto (Silberschatz *et al*; 2006).

Os SGBD utilizam vários tipos de modelos para descrever sua estrutura de armazenamento de dados: modelo hierárquico, modelo de redes, modelo relacional e modelo orientado a objetos (Takai *et al*; 2005). Neste trabalho será utilizado o modelo relacional, pois a função de um SGBD é de suma importância no desenvolvimento de sistemas de informações e hoje em dia este modelo tem sido o mais utilizado (Bonfioli, 2006).

O modelo de banco de dados relacional foi criado em 1970 por Edgar F. Codd e sua utilização de fato nas empresas teve início em 1987. Este modelo se baseia no princípio de que as informações contidas em uma base de dados podem ser como relações matemáticas, sendo representadas uniformemente e em tabelas bidimensionais (Machado; Abreu, 2004).

Segundo Silberschatz *et al*; (2006, p.5) o modelo relacional

> Usa uma coleção de tabelas para representar os dados e as relações entre eles. Cada tabela possui diversas colunas, e cada coluna possui um nome único. O modelo relacional é um exemplo de um modelo baseado em registros. Os modelos baseados em registros recebem esse nome porque o banco de dados é estruturado em registro de formato fixo de vários tipos.

Para os matemáticos, uma relação é definida como subconjunto de um resultado cartesiano de uma faixa de domínios, o que é praticamente a definição da tabela de um banco relacional. A diferença se dá nos atributos, pois em um banco de dados utilizam-se nomes, enquanto na matemática são números inteiros (1, 2, 3...), sendo as tabelas relações, as linhas são tuplas nos termos matemáticos (Silberschatz *et al*; 2006).

Outro conceito da teoria relacional matemática utilizado é o de que os usuários não precisam ter conhecimento de onde e como os dados estão, conceito principal de transparência (Machado; Abreu, 2004).

O modelo de banco de dados relacional se mostrou o mais adequado e flexível para a solução de diversos problemas no nível de criação e execução da base de dados. Este modelo não possui um caminho de acesso aos dados previamente definido (Takai *et al*; 2005). Seu destaque está por ser um modelo simples e que facilita para os programadores na hora do desenvolvimento (Silberschatz *et al*, 2006). A **Fig. 4** abaixo mostra um exemplo de tabelas do modelo relacional.

| Id_Cliente | Nome | CPF | Endereco |
|---|---|---|---|
| 1 | Ana | 12345678910 | Rua B |
| 2 | João | 10987654321 | Rua A |
| 3 | Vitor | 65748392019 | Rua C |

| Contrato | Saldo |
|---|---|
| 1 | 100,00 |
| 2 | 300,00 |
| 3 | 50,00 |

| Id_Cliente | Num_Contrato |
|---|---|
| 1 | 2 |
| 2 | 3 |
| 3 | 1 |

**Fig. 4 -** Tabelas do modelo relacional Cliente – Contrato. **Fonte:** Elaborado pelos autores baseado em Takai *et al* (2005).

As seguintes vantagens do banco de dados relacional são responsáveis por este ser o banco mais utilizado: a) a total independência dos dados; b) a visualização de diversos dados simultaneamente; c) o maior controle dos dados por estarem centralizados; d) os dados são armazenados uma única vez e podem ser compartilhados permitindo reduzir o espaço no armazenamento e compartilhamento; e) o controle da replicação dos mesmos; f) a manutenção da persistência e plenitude dos dados garantindo a segurança dos mesmos; g) o estabelecimento de padrões e facilidade no acesso aos dados; h) a centralização dos dados que possibilita determinar padrões de documentação e nomenclatura (Costa, 2011).

Como já abordado, não existe um caminho determinado para saber onde estão os dados dentro do banco e o usuário nem precisa saber. Então é necessária uma forma de identificá-los. Isso é permitido a partir dos valores dos campos da tabela, que devem ser únicos permitindo que se identifique determinada linha, ou seja, dentro do banco de dados, em uma mesma tabela, não se pode ter duas linhas com valores exatamente iguais em todos os campos (colunas) (Silberschatz *et al*, 2006).

Sendo assim, pode-se acessar a tabela pelo valor de um campo, mesmo se este não for designado chave. As chaves são utilizadas nas consultas ao banco, mas não são responsáveis pelo acesso e nem ordenação do mesmo. Chave primária é utilizada como identificação de uma linha da tabela. A Chave estrangeira serve para junção de duas ou mais tabelas. Uma chave não pode ter valor zero ou estar em branco e seu valor deve ser único, não podendo haver duas chaves primárias iguais em uma tabela e nem duas chaves estrangeiras com a mesma referência. A chave estrangeira de uma tabela é chave primária de outra, então neste

caso seu valor deve ser o mesmo em ambas as tabelas, tanto como primária quanto como estrangeira (Machado; Abreu, 2004).

Segundo Silberschatz *et al* (2006) a estrutura do banco de dados pode ser modelada de forma gráfica a partir do diagrama de entidade-relacionamento (E-R), que é composto por:

- Retângulos, que são a representação das entidades;
- Elipses, representando os atributos;
- Losangos, que são os relacionamentos;
- Linhas, utilizadas para a ligação das entidades com seus atributos bem como das entidades com os relacionamentos.

Na **Fig. 5** temos um exemplo de diagrama E-R dentro do contexto das redes sociais.

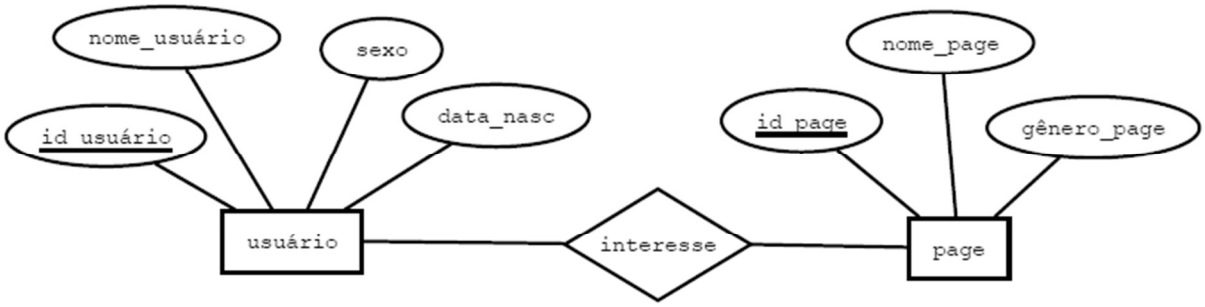


**Fig. 5** - Exemplo diagrama entidade-relacionamento. **Fonte:** Elaborado pelos autores baseado em Silberschatz *et al* (2006).

Podemos observar que id_usuário e id_page são atributos e estão sublinhados, diferente dos demais (nome_usuário, sexo, data_nasc, nome_page e gênero_page), isso se dá pelo fato de que ambos são chave primária de suas respectivas entidades, ou seja, id_usuário é atributo e chave primária da entidade usuário, bem como id_page é atributo e chave primária da entidade *page*.

Com o diagrama E-R é possível uma melhor compreensão da estrutura do banco por parte do usuário do mesmo, pois ele não exige o conhecimento de como os dados serão armazenados. Ultimamente o diagrama E-R tem sido muito utilizado na fase em que a base de dados é projetada (Takai *et al*; 2005).

Com o diagrama E-R podemos estabelecer restrições as quais o banco deverá respeitar. Dois tipos mais importantes de restrição são o mapeamento de cardinalidades e a existência de dependências (Silberschatz *et al*; 2006).

O mapeamento de cardinalidades é utilizada para mostrar com quantas entidades uma outra entidade pode se relacionar. Os relacionamentos podem ser:

a) Um para um: uma entidade x pode se relacionar apenas com uma entidade y, e esta entidade y só pode se relacionar com uma entidade x;

b) Um para muitos: uma entidade x pode se relacionar com inúmeras entidades y, porém uma entidade y só pode se relacionar com uma entidade x;

c) Muitos para um: uma entidade x pode se relacionar apenas com uma entidade y, porém uma entidade y pode se relacionar com inúmeras entidades x;

d) Muitos para muitos: uma entidade x pode se relacionar com inúmeras entidades y, e uma entidade y pode se relacionar com inúmeras entidades x.

Ainda na **Fig. 5**, podemos dizer que a entidade usuário possui um relacionamento de interesse com a entidade *page*. Aplicando o conceito de mapeamento de cardinalidade, as entidades usuário e *page* seguem a instrução de muitos para muitos, já que a entidade usuário pode se relacionar com inúmeras entidades *page*, e vice versa. O que determina o correto mapeamento de cardinalidade do conjunto de relacionamentos é a sua situação no mundo real, a qual está sendo modelada (Heuser; 2001).

A segunda restrição importante é a dependência de existência, que nada mais é do que quando uma entidade y depender da existência de uma entidade x. Então y é dependente da existência de x. Outro termo que também podemos utilizar é entidade dominante x e entidade subordinada y (Heuser; 2001).

Um sistema de banco de dados oferece uma linguagem de definição de dados para discriminar as tabelas e uma linguagem de manipulação de dados para consulta e atualização dos dados, ambas compõem uma linguagem única, a linguagem SQL, que tem sido amplamente utilizada e que será detalhada neste trabalho (Date; 2003).

Enfim, o modelo relacional é destaque dentre os demais, sendo utilizado pela maioria dos bancos de SGBD, suprindo de maneira satisfatória as necessidades de aplicações convencionais no quesito tecnologia de banco de dados e seu desenvolvimento (Bonfioli, 2006). Contudo, quando se trata de aplicações mais complexas este modelo possui suas limitações que serão tratadas no decorrer desta pesquisa.

## 2.4 SQL

O SQL é uma linguagem muito utilizada que teve início a partir do modelo relacional. A princípio, nomeada SEQUEL, foi desenvolvida pela IBM em 1974 e, após algumas alterações de melhorias e um grande projeto chamado System R, passou a ser chamado SQL.

Seu sucesso foi tamanho pela forma fácil de consulta e manipulação dos dados tornando a sua utilização cada vez maior. Foi em 1982 que a SQL tornou a linguagem padrão de banco de dados relacional pela ANSI (Machado; Abreu, 2004).

Como apresentado anteriormente, o banco de dados relacional é composto por tabelas com linhas e colunas. Seus dados não são ordenados e, para localizá-los, utiliza-se um atributo que assume a função de chave primária da tabela. O que tornou o sistema relacional popular foi sua facilidade em manipular os dados e seu fácil entendimento. Dessa maneira a linguagem SQL surgiu para que fosse possível a manipulação do banco de dados.(Machado; Abreu, 2004).

Segundo Silberschatz *et al* (2006), o SQL é dividido em conjunto de comandos, dois deles são: Linguagem de Definição de Dados (DDL) e Linguagem de Manipulação de Dados (DML). A DDL permite criar tabelas no banco, enquanto a DML é utilizada para se excluir e modificar dados. O SQL possui limites para garantir a plenitude dos dados e autorização.

Valores inteiros (*int*), reais (*real*), um caractere (*char*) e uma cadeia de caracteres (*varchar*) são alguns tipos de domínios internos da linguagem SQL (Silberschatz *et al*, 2006).

As tabelas do banco de dados são definidas a partir do comando *create table*. Depois de criadas, quando ainda vazias, utiliza-se o comando *insert* para inserção dos dados. O comando *delete* é responsável por apagar uma ou mais linhas da tabela. Para excluir a tabela pode-se fazer uso do comando *drop table* e para alterá-la, o *alter table* (Silberschatz *et al*, 2006).

Uma das principais operações de consulta da linguagem SQL se dá por meio do comando *select*, e por palavras-chaves que o compõe (*from* e *where*). O *select* determina as colunas a serem selecionadas, o *from* indica as tabelas e *where* as linhas. Para ordenação das informações na tabela se utiliza o comando *order by* e, para agrupá-las, o *group by*. Como um banco de dados não possui somente uma tabela e elas podem se relacionar. Para consultas em que é necessária a visualização de duas ou mais tabelas ao mesmo tempo, o comando *join* é utilizado para a junção das mesmas permitindo a seleção de dados (Machado; Abreu, 2004).

Segundo Machado e Abreu (2004) a linguagem SQL possui vantagens e desvantagens. Como vantagens esta linguagem é independente de fabricante, ela é oferecida pelos SGBD's, sendo assim não é preciso se preocupar caso se queira trocar de SGBD. Além disso a linguagem também oferece a portabilidade entre computadores, podendo ser utilizada tanto em um computador pessoal quanto em um computador de grande porte. A utilização da linguagem SQL também traz a vantagem da redução de custos com treinamento, pois com a

sua portabilidade podemos movimentar uma aplicação de um ambiente a outro sem a necessidade de reciclar a equipe de desenvolvimento.

O inglês estruturado de alto nível da linguagem SQL é outra de suas vantagens, por ser composta de um conjunto de sentenças em inglês simples e de rápido e fácil entendimento. Também possui a vantagem de suas consultas serem interativa, ela oferece respostas ao usuário, relativa a questões muitas vezes complexas, em segundos. Por fim, a linguagem SQL possui a vantagem de definição dinâmica dos dados, permitindo alterar e excluir as estruturas de dados dinamicamente e com a maior flexibilidade.

Além dessas vantagens a SQL, possui algumas desvantagens; uma delas está no fato da padronização acabar inibindo a criatividade na hora do desenvolvimento por deixar o desenvolvedor preso a ela, sem poder alterar ou até mesmo melhorar a aplicação.

## 2.5 PostgreSQL

Como visto no subitem 2.3 deste trabalho, é de fundamental importância o papel do SGBD no desenvolvimento de um sistema, portanto a escolha deste SGBD também deve ser eficiente, pensando sempre na compatibilidade com as tecnologias atuais. Em meio a diversos SGBD's o PostgreSQL se destaca pelo seu código aberto e seus recursos avançados (Souza *et al*; 2011).

A primeira versão do PostgreSQL foi lançada em 1989, desenvolvido em um projeto da Universidade de Berkeley na Califórnia pela licença Berkeley Software Distribution (BSD) de código aberto. O objetivo principal de seu desenvolvimento foi a necessidade de um SGBD que entendesse vários tipos diferentes de dados e que descrevesse suas relações (Souza *et al*; 2011).

Hoje, seu código é livre e, dentre seus desenvolvedores, a maior parte são voluntários que se relacionam por meio da internet. Há também um grupo responsável pela sua validação. Empresas grandes como a Fujitsu, NTT Goup, Skype, Hub.org, Red Hat e SRA são financiadoras do PostgreSQL que, além disso, recebe doações. É utilizado por multinacionais, órgãos governamentais e universidades. Recebeu vários prêmios como melhor sistema de banco de dados *Open Source*. O PostgreSQL possui recursos para consultas complexas, triggers, transações e controle de concorrência. Permite adição de novas funcionalidades e é de fácil utilização (Souza *et al*; 2011).

## 2.6 Bancos de dados NoSQL

Os bancos de dados que não possuem uma estrutura fixa, nos quais os dados são interpretados de diferentes maneiras a fim de se obter uma maior escalabilidade, são denominados NoSQL. A crescente necessidade de dinâmica durante o desenvolvimento de sistemas e a busca por soluções que demandem um menor investimento de recursos fizeram com que diversos bancos de dados NoSQL fossem desenvolvidos de maneira *open-source* (código aberto). Atualmente, muitos desenvolvedores contribuem para o movimento NoSQL, aprimorando ou criando novos recursos para os bancos já existentes.

Brito (2010, p2) afirma que "O propósito, portanto, das soluções NoSQL não é substituir o modelo relacional, mas apenas em casos nos quais seja necessária uma maior flexibilidade da estruturação do banco". Diante das diferentes exigências na elaboração de um *software,* há casos em que somente o banco de dados relacional não é capaz de atender a demanda de informações que se pretende gerar após a expansão dos dados.

Empresas como o Google, Amazon, Facebook e posteriormente o Twitter utilizaram e ainda utilizam os bancos de dados NoSQL: *BigTable*, *Dynamo* e *Cassandra* respectivamente para suprir a grande demanda de dados que trafegam pelo sistema. Além dos bancos de dados citados, entre 2009 e 2010 outras soluções como *CouchDB*, *MongoDB*, Neo4j e *Inforgrid* surgiram com o propósito de atender diferentes arquiteturas de dados (Diana; Gerosa, 2010).

Diferentemente do banco de dados relacional que utiliza a propriedade ACID, em que a principal característica é a consistência e, por esse motivo, não suporta o escalonamento horizontal, os bancos de dados NoSQL oferecem a propriedade BASE que, ao contrário da ACID, não garante a consistência, pois não há a garantia de que os dados atualizados serão apresentados em sua última versão (teorema CAP) ela pode eventualmente existir. No entanto, a propriedade BASE é caracterizada pela disponibilidade, ou seja, a principal ideia é a escalabilidade onde os dados podem ser distribuídos por diversos servidores evitando a dependência de uma aplicação a somente um servidor, garantindo que os dados estarão sempre disponíveis (Brito, 2010).

O NoSQL possui diferentes tipos de estrutura de dados que são chamados de núcleos. São eles: *Wide Column Store, Document Store, Key Value Store, Eventually Consistent Key Value Store, Object Databases, Graph Databases, Grid Database Solutions, XML Databases.* Cada núcleo possui um tratamento específico para os dados, desta forma, os bancos podem ser orientados por diferentes núcleos (IMaster, 2010).

Brito (2010, p3) descreve os núcleos (ou categorias) de bancos de dados NoSQL que mais se destacam como:

- CHAVE-VALOR: coleção de chaves de valor único que são associadas ao conteúdo (dado). Citando como exemplo o *Amazon Dynamo;*
- DOCUMENTOS: conjunto de documentos no formato JSON, onde cada documento é tratado como um objeto único que possui campos com os respectivos valores. O *CouchDB* é um dos bancos mais utilizados.
- COLUNA: formado por colunas que possuem um conjunto de informações como nome, conteúdo e *timestamp* e que podem ser organizadas em uma família de colunas (semelhante a uma tabela), além de vetores cujos valores são as próprias colunas. Os bancos de dados *Cassandra* e *BigTable* são alguns exemplos.
- GRAFOS: dados distribuídos em forma de vértices e arestas que possuem atributos tanto nas arestas quanto nos vértices. Neo4J e InfoGrid são exemplos de bancos de dados que implementam essa arquitetura.

Esta pesquisa irá aplicar a estrutura de grafos (*Graph Databases*) durante seu desenvolvimento. No subcapítulo seguinte esta estrutura será apresentada com mais detalhes.

## 2.7 Banco de dados orientado a grafos

Como descrito anteriormente, grafo é um conjunto de pontos que são chamados de vértices e que são interligados entre si por arestas. Com base nesta forma de raciocínio, aliada à busca dos desenvolvedores NoSQL, por diferentes meios de se modelar uma estrutura de dados, é que surgiram os bancos de dados orientados a grafos, cuja estrutura e funcionamento serão descritos neste subcapítulo.

Robinson *et al*. (2013) definem o banco de dados orientado a grafos como um mecanismo de armazenamento que possibilita a modelagem de uma estrutura que implemente a teoria de grafos, além da execução de pesquisas em busca de um registro, retornando resultados com maior desempenho.

A principal ideia da utilização de uma estrutura de dados baseada em grafos é a facilidade de abstração do problema, ou seja, identificar os principais pontos que deverão ser tratados na aplicação e representá-los por círculos ou retângulos e então, conectá-los de acordo com a necessidade através de linhas. Desta forma, a implementação física do diagrama

realizar consultas como: “Quais são as pessoas que vivem em Pouso Alegre e que estudam na UNIVÁS?” (Neotechnology, 2013).

O planejamento é a principal forma de garantir que o banco de dados baseado em grafos ofereça os recursos necessários para a aplicação. Para Robinson, *et al*; (2013), o processo de planejamento pode ser direcionado por diferentes fatores, como por exemplo o custo, desempenho, redundância e carga de dados. Os fatores irão depender do tipo da aplicação que se pretende desenvolver, além da quantidade de dados que irão ser inseridos à estrutura, número de usuários e suas atividades. As características são:

- CUSTO: utilização de *hardware* que suporte a demanda, pois o sistema de gerência do banco de dados utiliza o disco rígido para o armazenamento dos arquivos de sistema e a memória principal para a manipulação dos dados no grafo. Qualquer alteração de *hardware* para a obtenção de desempenho ou escalabilidade são custos que devem ser levados em conta.
- DESEMPENHO: alteração do *cache* nos arquivos de sistema (que podem variar de 2GB à 200GB) e a adição de dispositivos SSD’s ou *flash drive* são meios de se obter uma maior agilidade nas transações da base de dados. O cálculo de memória a ser alocada, bem como a capacidade de armazenamento são atribuídos pelos desenvolvedores que necessitam ter total conhecimento da aplicação a ser desenvolvida.
- REDUNDÂNCIA: para se obter disponibilidade de informação, é necessário estipular o número de *clusters* que trabalharão em conjunto. Em aplicações críticas esse número deve ser no mínimo dois.
- CARGA: otimização para suportar o número de requisições que a aplicação fará à base de dados. É um dos fatores mais importantes durante o *upgrade* de um servidor; deve-se levar em conta o tempo médio em que o servidor responde a uma requisição enviada pelo cliente e a quantidade de acessos simultâneos.

Atualmente existem ferramentas disponíveis para a modelagem, gerência e auxílio na elaboração de algoritmos em grafos como: *InfoGrid*, *HyperGraphDB*, *BigData* e *Neo4J*. Todos os bancos citados possuem a principal característica de manipulação de dados orientados a grafos, no entanto, o Neo4J é o mais utilizado e possui uma rede maior de colaboradores (Imaster, 2010).

## 2.8 Neo4J

Como descrito pela Neotechnology (2013), o Neo4J é uma das principais ferramentas para elaboração de banco de dados orientado a grafos existentes atualmente. É desenvolvido de maneira *open source* (código aberto) e é mantido pela empresa Neotechnology com o auxílio de uma comunidade de colaboradores desde 2003.

A Neotechnology (2013) oferece três diferentes edições do Neo4J que atendem diferentes tipos de consumidores, são elas:

- *COMMUNITY:* sistema *open-source* com os recursos básicos necessários para a implementação de banco de dados orientado a grafos. São indicados para testes e avaliações. O serviço é disponível para download gratuito no *site* oficial da empresa.
- *ADVANCED:* disponível sob licença comercial, possui recursos extras para monitoramento avançado.
- *ENTERPRISE:* além do recurso avançado para monitoramento, essa versão também oferece backup online e serviços de escalonamento horizontal (*Clustering*), a edição não é gratuita.

Para aplicações cujo código fonte não é aberto e em que há a necessidade de implementação do Neo4J, as edições *Advanced* e *Enterprise* oferecem recursos extras, além de suporte técnico oferecido pela Neotechnology.

Segundo Robinson *et al*. (2013), o Neo4J possui duas principais formas de estruturar e manipular o banco de dados são elas:

- *EMBEDDED:* nesta versão, o Neo4J é incorporado à aplicação, sendo assim, os dados são manipulados dentro do próprio código fonte (inserção, alteração e consultas). O Neo4J possui API's para a linguagem Java que permite ao profissional de desenvolvimento de *software* elaborar algoritmos para consultas transversais à estrutura criada, além de controle de *indexes*.
- *SERVER:* a versão *server* oferece recursos para comunicação *web* utilizando o protocolo HTTP. Com requisições feitas por documentos no formato JSON, é possível realizar a comunicação entre estações clientes e base de dados por meio da API REST. Esta característica torna a execução do banco de dados independente de plataforma. A comunicação do servidor com aplicações Java

também pode ser feita com extensões do servidor (*Server extensions*) utilizando a API Java conhecida como JAX-RS.

A versão *Server* do Neo4J possui uma interface de gerenciamento que pode ser acessada pelo navegador *web*. Todas as informações e recursos necessários para a manipulação do banco de dados ficam disponíveis na janela principal. Os principais recursos da interface são: criar vértices, bem como suas propriedades; criar arestas e suas propriedades; obter informações sobre o servidor, além do *shell* de comandos, onde é possível executar os comandos *Cypher* ou *Gremlin* (Neotechnology, 2013).

Como visto anteriormente, no NoSQL a consistência é eventual (ACID). Apesar do Neo4J ser construído com base na arquitetura NoSQL, ele implementa o paradigma ACID, garantindo a integridade dos dados, desta forma, nota-se uma grande vantagem em sua utilização se comparado com os demais bancos de dados (Matsushita, Nguessan. 2011).

O funcionamento do Neo4J segue os padrões gerais dos bancos de dados orientados a grafos, apresentados no subcapítulo anterior. Para encontrar um nó ou um relacionamento em específico a partir de uma propriedade como, por exemplo, o nome, a ferramenta também utiliza índices que podem ser criados pelo administrador do banco de dados ou automaticamente, após a ativação deste recurso no arquivo de configuração do Neo4J. Os índices irão varrer todos os vértices a procura do valor da propriedade informada pelo usuário e então, retornará o vértice resultante da pesquisa (Neotechnology, 2013).

## 2.9 Cypher

Para a modelagem e consultas em um banco de dados é necessário utilizar uma linguagem específica para manipular os dados e a sua estrutura. Como visto no subcapítulo anterior, o Neo4J possui suporte às linguagens *Gremilin* e *Cypher*. Nesta pesquisa, a linguagem *Cypher* será aplicada e sua definição será descrita a seguir.

Taylor e Jones (2012) apresentam a linguagem *Cypher* como sendo uma alternativa encontrada pela Neotechnology no que se refere à manipulação dos dados no grafo, afim de tornar a compreensão da modelagem mais próxima à visão humana. Em muitos casos a API Java necessita de mais linhas de código para a execução dos algoritmos de pesquisa. A linguagem *Gremilin* por sua vez, segue um padrão, sem dinâmica, prescritiva, desfavorecendo o uso da linguagem em casos onde há a necessidade de consultas complexas. Já a linguagem

SPARQL, utilizada em alguns casos para manipular dados RDF, é projetada para diferentes modelos de dados.

Cypher é uma linguagem de fácil entendimento, pois se assemelha a linguagem humana. As expressões de consulta utilizam conceitos de entrada e saída de informações de vértices, baseando-se em seus relacionamentos. Cypher foi projetado para atender tanto DBA's quanto outros profissionais que precisam realizar consultas em um banco de dados orientado a grafos. Por ser autoexplicativa, a linguagem torna as pesquisas complexas possíveis de serem executadas pelo profissional de banco de dados sem que o mesmo tenha um conhecimento aprofundado em estruturas de dados (Neotechnology, 2013).

Robinson *et al*. (2013) afirmam que a linguagem Cypher é composta por três cláusulas que compõe uma pesquisa (*query*) simples, são elas:

- *START*: Define um ou mais pontos iniciais, eles podem ser obtidos utilizando índices;
- *MATCH*: Executa operações de entrada e saída de nós, utilizando caracteres ASCII;
- *RETURN*: Especifica quais objetos serão apresentados após a execução do *MATCH*.

A **Fig. 7** apresenta um exemplo simples de consulta utilizando comandos Cypher com base em uma estrutura de grafos já criada, que tem como objetivo encontrar as peças de teatro que o autor Willian Shakespeare escreveu. Neste exemplo, para iniciar a navegação no grafo, o comando *START* é utilizado para encontrar um vértice específico que servirá como ponto de partida e então, armazená-lo na referência *bard*. Para encontrar tal vértice, é possível criar índices que retornarão elementos do grafo a partir de uma ou mais propriedades, neste exemplo, utilizou-se a propriedade *firstname* e *lastname* do índice *authors.* Com a referência criada, é preciso analisar as arestas que chegam ou saem de determinados vértices, as arestas são graficamente representadas na linguagem Cypher por linhas direcionais (- >) ou não (--). O comando *MATCH* realiza tal navegação analisando as arestas que possuem o tipo (*WROTE*) e que saem do vértice *bard,* e então, armazena o resultado na referência *plays.* Desta forma, a referência *plays* contêm as peças escritas por Willian Shakespeare, o resultado é retornado ao usuário por meio do comando *RETURN*. (Robinson *et al*. 2013).

```
START bard=node:authors('firstname:"William" AND lastname:"Shakespeare"')
MATCH (bard)-[:WROTE]->(plays)
RETURN plays
```

**Fig. 7 -** Exemplo de código *Cypher* I. **Fonte:** Robinson *et al* (2013).

Para Robinson *et al*. (2013), alguns comandos utilizados em consultas que utilizam a linguagem SQL em bancos de dados relacionais são semelhantes aos comandos *Cypher* como por exemplo:

- *WHERE*: Critérios de seleção após a reunião de dados preliminares;
- *CREATE*: Utilizado na criação de vértices e arestas (*nodes* e *relationships*);
- *DELETE*: Remoção dos elementos do grafo;
- *SET*: Utilizado na configuração de propriedades aos elementos do grafo;
- *WITH*: Utilizado para a divisão de um ou mais comandos, formando partes distintas de uma mesma *query*.

Para exemplificação, o vértice (V1) possui uma aresta (A) que o interliga de maneira não simétrica ao vértice (V2), esse cenário pode ser criado com o seguinte código: *CREATE (V1)-[:A]->(V2);*.

A execução do código *Cypher* presente na **Fig. 8** abaixo é um exemplo de utilização do comando *WITH* e *SET*. Após a definição do vértice principal, é realizada uma busca para encontrar arestas que saem do vértice V armazenando-as no identificador A. O comando *WITH* realiza a contagem de arestas armazenando seu valor em C. O comando SET por sua vez atribui o valor C à propriedade das arestas presentes em A (Robinson *et al*. 2013).

```
1 START V=node(1)
2 MATCH V--A-->()
3 WITH A, count(*) as C
4 SET A.cont = C
```

**Fig. 8** - Exemplo de código *Cypher* II. **Fonte**: Elaborado pelos autores (2013).

Em resumo, a linguagem Cypher possui características similares à linguagem SQL com comandos que auxiliam o DBA ao realizar consultas e torna a compreensão da teoria de grafos mais fácil para quaisquer usuários.

## 2.10 Java

Java é uma plataforma de desenvolvimento e uma linguagem de programação orientada a objetos, lançada em 1996 pela *Sun* com a capacidade de desenvolver aplicativos tanto para *desktop* quanto *web*, entre outros. Permite a vantagem de programar utilizando uma única linguagem mesmo em diferentes plataformas (Sobral; Claro, 2008).

Os programas desenvolvidos em Java podem ser *Applets*, aplicativos ou *Servlets*. *Applet* é um tipo de programa carregado junto com páginas HTML, já os aplicativos necessitam que na máquina esteja instalado um interpretador (JVM). *Servlets* são classes para serem interpretados por um servidor *web* (Sobral; Claro, 2008).

Para o desenvolvimento em Java é necessário um *kit*, o JDK – Java Developement Kit que a própria Oracle oferece. O JDK é composto por um compilador, interpretador e utilitários. Sua utilização é feita primeiramente escrevendo o programa em um editor de texto ou IDE's que são ambientes de edição, compilação e testes de código-fonte como, por exemplo, o Eclipse. Logo após, o programa deve ser compilado gerando um arquivo com a extensão (*.class)* (Sobral; Claro, 2008).

A linguagem Java se caracteriza por ser orientada a objetos facilitando sua manipulação, além de familiar por possuir sintaxe parecida com a linguagem C++. Permite multiprocessamento (*multithread*), linhas de execução de um mesmo processamento executadas em um mesmo intervalo de tempo são *threads*. Outra característica é sua portabilidade, por poder ser executado em qualquer plataforma (Sobral; Claro, 2008).

O objetivo da linguagem Java não são os sistemas pequenos, seu foco é uma plataforma para o desenvolvimento de sistemas de médio a grande porte, com uma equipe de desenvolvedores que pode estar sempre mudando e crescendo. É normal que começar a desenvolver uma aplicação utilizando a Java seja mais trabalhoso do que em outras linguagens script, mas o que se tem de trabalho no início é facilitado na hora de realizar alterações no sistema, devido sua linguagem ser orientada a objetos além de ser madura (Caelum, 2013).

Uma vantagem de se utilizar a plataforma Java é a grande quantidade de bibliotecas gratuitas que podem ser utilizadas em diversos trabalhos como relatórios, gráficos entre outros. Como cada linguagem é apropriada para uma determinada área, a utilização da Java é melhor para o desenvolvimento de aplicações que tenham tendência a crescer (Caelum, 2013). Enfim, contudo a Java sendo brevemente utilizada neste trabalho ela foi escolhida por ser uma linguagem de fácil manipulação e pela sua biblioteca que nos será de grande auxilio.

# 3. QUADRO METODOLÓGICO

Neste capítulo serão apresentados o tipo de pesquisa, contexto e os processos realizados para a obtenção dos resultados da pesquisa, bem como a maneira em que as ferramentas foram empregadas.

## 3.1 Tipo de Pesquisa

A pesquisa realizada é do tipo exploratória. Ela é utilizada para se obter maior conhecimento sobre determinado assunto ainda novo no mercado. De acordo com Santos (2013, p.1) o objetivo de uma pesquisa exploratória:

> É familiarizar-se com um assunto ainda pouco conhecido, pouco explorado. Ao final de uma pesquisa exploratória, você conhecerá mais sobre aquele assunto, e estará apto a construir hipóteses.

Seguindo este conceito, esta pesquisa apresentará a conclusão dos participantes com base nas definições teóricas disponibilizadas por empresas e demais estudos acadêmicos sobre a aplicação prática de bancos de dados distintos, e que, por esse motivo, atendem diferentes exigências, gerando resultados que podem, ou não garantir a qualidade ideal para determinado *software*.

Os dados utilizados em pesquisa exploratória são colhidos a partir de pesquisas bibliográficas e neste trabalho também serão realizadas reuniões entre os participantes e o orientador. Os dados são coletados após o desenvolvimento dos modelos de banco relacional e orientado a grafos. Os bancos são carregados com informações fictícias, então são feitas as consultas e a partir dessas consultas, colhidos os dados de tempo de resposta e outros dados importantes para o resultado do projeto.

O objetivo é obter uma base de conhecimento no banco de dados orientado a grafo, aliado com o conhecimento de banco de dados relacional vistos durante o curso e apresentá-lo como uma solução no tratamento de dados como grafos, aplicando-o no contexto das redes sociais mostrando de forma prática as diferenças entre os dois modelos.

## 3.2 Contexto da pesquisa

Há um aumento excessivo de dados na internet. Diversas informações são transmitidas pela rede e nem sempre podem ser armazenadas de uma mesma forma, como em tabelas no banco relacional. Surge então a necessidade de armazenar estes dados de uma maneira mais dinâmica sem que se percam. A utilização de um banco de dados orientado a grafos pode ser a solução para esse tipo de problema.

O Neo4j é um dos bancos de dados orientado a grafos mais conhecidos e vem sendo muito estudado no momento. É uma tecnologia nova que oferece muitas melhorias em *performance* e escalabilidade no tratamento de dados representados como grafos. Após um estudo mais aprofundado no assunto, sua aplicação é necessária para a coleta dos resultados. O contexto das redes sociais não poderia ser mais apropriado para tal desenvolvimento, visto que se trata de algo atual e que movimenta uma grande quantidade de dados.

As redes sociais são *sites* da internet onde pessoas criam um perfil de si mesmas com foto e informações pessoais, e criam laços de amizade com outras pessoas que tenham perfil no mesmo *site*. A partir desses laços é permitida a interação entre elas. Toda e qualquer movimentação realizada na rede social gera dados, sendo assim, cada comentário ou mensagem enviada deve ser armazenado adequadamente.

Com a facilidade de acesso à internet atualmente, o número de pessoas "conectadas" no mundo virtual gerando dados a todo o momento é grande, ou seja, são bilhões de registros a se armazenar. Neste contexto se torna imprescindível a utilização de um banco de dados eficiente no tratamento de um grande volume de dados de forma apropriada.

## 3.3 Instrumentos

Esta pesquisa foi desenvolvida utilizando como contexto principal os dados presentes em redes sociais. Os instrumentos utilizados foram as pesquisas de funcionalidades nas redes sociais *Facebook* e *Linkedin,* além de reuniões entre os participantes para a definição da estrutura de dados utilizada.

Com o surgimento da ferramenta *Graph Search* (Busca Social) desenvolvida pelo *Facebook*, as perguntas comumente feitas no cotidiano do meio social foram representadas na *web* a fim de se obter exatidão nas respostas, além de retornar a informação que o usuário necessita (Facebook, 2013). A ferramenta *Graph Search* foi utilizada nesta pesquisa como

fonte para a coleta de informações, por meio de exemplos fornecidos pelo próprio recurso e também com a exploração do mesmo por parte dos participantes.

Após a análise do comportamento das redes sociais, foi observado que o modelo adotado pela rede *Linkedin* fornece recursos como: rede de contatos, grupos de usuários, empresas e listas de empregos atrelados às empresas cadastradas (Linkedin, 2013). Já a rede *Facebook* possui características como rede de contatos, grupos e utiliza o conceito de *pages* (páginas) que representam escolas, empresas e interesses diversos (Facebook, 2013).

Com reuniões entre os participantes desta pesquisa baseadas na análise realizada anteriormente, foram definidos cinco principais tipos de dados a serem manipulados pelas bases de dados. Essas informações foram definidas pela semelhança entre ambas as redes sociais analisadas[3], são: a) Usuários, b) Escolas e Empresas, c) Cidades; d) Grupos; e) interesses.

Para a realização da avaliação prática, uma rotina de consultas foi desenvolvida pelos participantes baseando-se na estrutura definida anteriormente. As consultas[4] envolveram informações com dois ou mais tipos de dados, obtendo como resposta, o dado retornado após o processamento dos bancos de dados, e foram definidas após a observação dos participantes sobre a ferramenta *Graph Search* do *Facebook*.

Para que a execução das consultas fossem realizadas, foram inseridos dados fictícios em ambas as estruturas. A quantidade de dados utilizada foi: 1.000.000 de usuários, 50 escolas, empresas, interesses e cidades além de 30 grupos. A quantidade foi definida mantendo-se entre a limitação de *hardware* utilizada na pesquisa e o uso de ao menos 0,4% da quantidade de usuários ativos na rede *Linkedin* que é de 238 milhões (Congo, 2013).

Foram utilizados ao todo 100 relacionamentos para cada usuário envolvendo todos os tipos de dados citados anteriormente. A quantidade foi definida após a exploração das funcionalidades da rede *Facebook* pelos participantes e utilizando até 50% da média de amigos por jovem ativos na rede que é de 300 amigos (Madden, *et al*; 2013).

## 3.4 Procedimentos

O desenvolvimento desta pesquisa seguiu um cronograma com as principais tarefas e foram realizadas de maneira progressiva de acordo com os estudos em cada tecnologia por meio de reuniões presenciais e à distância.

[3] Os tipos de dados serão explorados na seção 3.4.1
[4] A descrição das consultas será apresentada na seção 3.4.5

O ambiente de implementação foi direcionado ao sistema operacional *Windows Seven* após a instalação e configuração das ferramentas necessárias para a execução prática da pesquisa. Ambos os bancos de dados, alvos desta pesquisa, foram instaladas localmente.

### 3.4.1 Modelagem de dados

Como descrito anteriormente, os resultados dos instrumentos desta pesquisa foram a quantidade e os tipos de dados que serão trabalhados sobre uma estrutura. As informações definidas para a modelagem dos dados foram descritas como:

1) Usuário: é considerado o objeto principal da rede. Para cada usuário é possível manter relacionamentos dos seguintes tipos: a) conhecer um ou mais usuários que, por sua vez, podem manter vínculos de amizades ou familiares; b) estudar em um local; c) trabalhar em um local; d) residir em um único local; e) participar de um ou mais grupos de usuários; f) possuir um ou mais interesses.
2) Escolas e Empresas: possuem o tipo de relacionamento "estudar" ou "trabalhar" com um ou mais usuários.
3) Cidades: são relacionadas com os usuários com o tipo "residir".
4) Grupos: possuem somente relação do tipo "participar" com um ou mais usuários.
5) Interesses: assim como grupos, são relacionados com um ou mais usuários com o tipo "gostar".

A partir da definição das principais informações geradas nas bases de dados, foi realizado o esboço dos modelos de dados objetos de estudo desta pesquisa que serão apresentados nas seções seguintes.

### 3.4.2 Modelo relacional

O modelo relacional desta pesquisa é composto por cinco entidades nomeadas usuarios, grupo, tipos_rel_page, page e tipos_rel_contato, e três relacionamentos: contatos, usuario_grupo e usuario_page. No diagrama ER elas são representadas conforme a **Fig. 9**. Os relacionamentos estão representados por losangos, os retângulos são as entidades e as elipses representam os atributos tanto das entidades quanto dos relacionamentos.

O losango usuario_grupo faz a representação do relacionamento entre os usuários e os grupos da rede. Esse relacionamento é de muitos para muitos, ou seja, um usuário pode participar de mais de um grupo, assim como um grupo contém vários usuários. Nesta pesquisa a entidade grupo possui dois atributos: id e nome, e o relacionamento usuario_grupo possui os atributos usuario_id e grupo_id.

A entidade usuarios é composta pelos atributos: id, nome, data_nasc, sexo e email, e também se relaciona com a entidade page de atributos id, nome e genero_tipo_ramo por meio da tabela relacionamento usuario_page. A entidade page é utilizada para o armazenamento de páginas criadas dentro da rede, essas páginas podem representar uma cidade, escola, empresa ou algo que seja de gostos em comum, como estilo musical, filmes, séries, programas de TV e etc.

Sendo assim é necessário informar o tipo de relacionamento que se tem com essa página, neste caso pode ser: residir, quando a page se referir a uma cidade; estudar para páginas de escolas; gostar para assuntos em geral, entre outros. Para isso criou-se a entidade tipos_rel_page que possui dois atributos: id e nome. Dentro da tabela relacionamento usuario_page o atributo tipos_rel_page_id faz referência a essa tabela; essa referência será melhor detalhada ainda nesta seção. Outros atributos do relacionamento usuario_page são os usuario_id e page_id.

As interações entre os usuários da rede são permitidas através do relacionamento contatos, composto pelos atributos: data_rel, usuario_id, usuario_id_amigo e tipos_rel_contato_id, sendo que esta última se refere a tabela entidade tipos_rel_contato, de atributos: id e tipo. Os relacionamentos entre os usuários da rede podem ser de amizade (amigo) ou parentesco (família), portanto é um relacionamento de muitos para um, pois cada contato pode ter apenas um tipo de relacionamento, ou de amizade ou de parentesco, enquanto que o relacionamento do tipo amizade pode estar para vários contatos, assim como o do tipo parentesco.

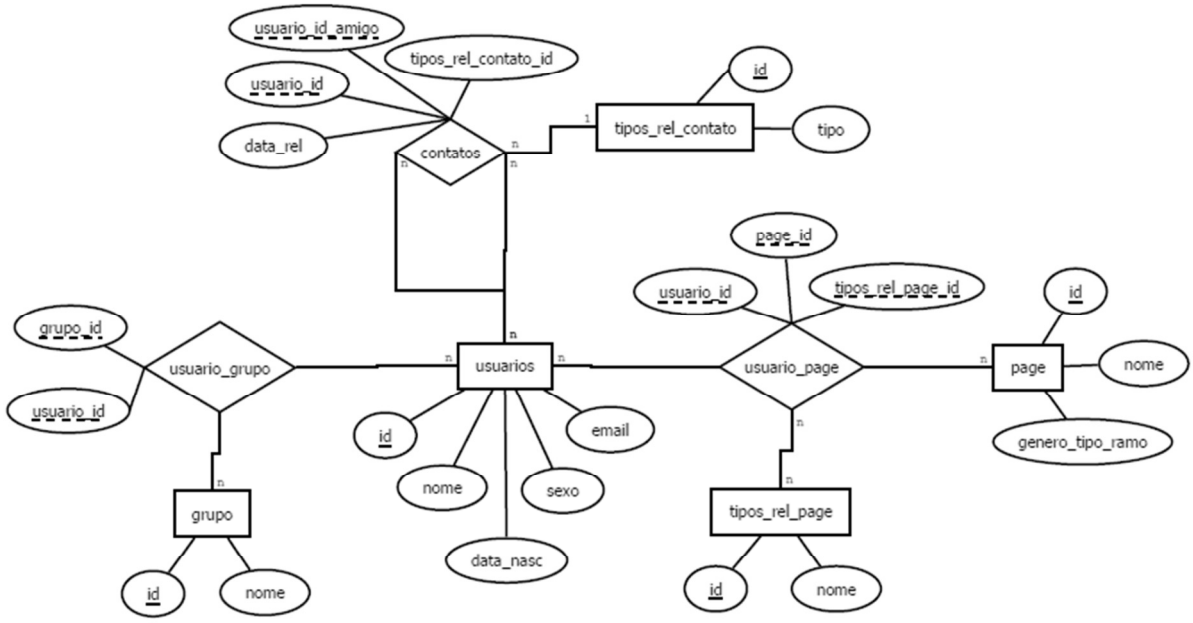


**Fig. 9** - ER Estrutura Modelo Relacional. **Fonte:** Elaborado pelos autores (2013).

A **Fig. 10** representa o modelo físico do banco relacional desenvolvido. Com este modelo pode-se visualizar os tipos de cada atributo e as chaves das tabelas. No modelo relacional desenvolvido os atributos podem ser do tipo *integer*, *varchar* e *char.*

Os atributos do tipo *integer* são de números inteiros (Ex.: 1, 2, 3..), esse tipo está sendo utilizado para os atributos de identificação da tabela (id, usuario_id, grupo_id, usuario_id_amigo, tipos_rel_contato_id, tipos_rel_page_id).

O *varchar* é utilizado para o armazenamento de uma cadeia de caracteres. Nesta pesquisa são atributos do tipo *varchar* o nome das entidades/relacionamento grupo, usuarios, tipos_rel_page e page. O atributo tipo da tabela tipos_rel_contato, genero_tipo_ramo da tabela page, data_rel da tabela contatos e data_nasc da tabela usuarios também são *varchar*. Para armazenar apenas um caractere, como F para sexo feminino e M para sexo masculino utiliza-se *char*.

Na **Fig. 10** notam-se as chaves primárias e estrangeiras do banco acompanhadas por PK, FK e PFK. Os atributos PK são atributos chave primária que são utilizadas para identificação da tabela. Quando o atributo é seguido de FK ele é uma chave estrangeira, utilizada para fazer referência a um atributo chave primária de uma outra tabela. PFK são atributos chave primária e estrangeira ao mesmo tempo.

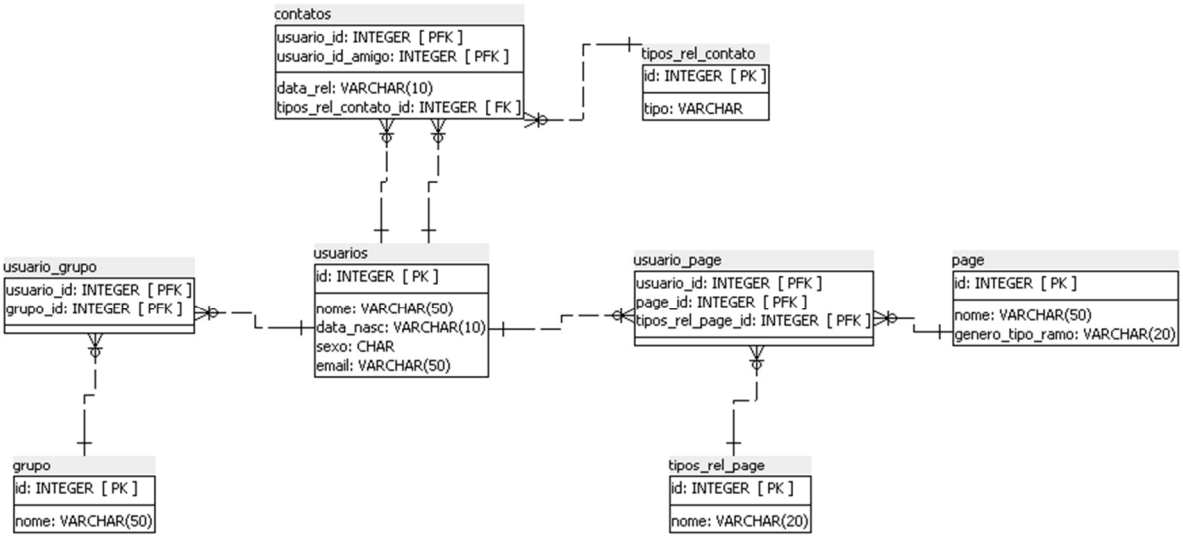


**Fig. 10** – Modelo Físico do Banco Relacional**. Fonte:** Elaborado pelos autores (2013).

No PostgreSQL criou-se as tabelas utilizando a linguagem SQL. Conforme já citado para a criação das tabelas é utilizado o comando *create table*. Iniciou-se pelas entidades e em seguida foram criados os relacionamentos. A entidade usuario foi criada com a SQL da **Fig. 11**.

O comando *create table* deve ser seguido pelo nome da tabela, entre parênteses acrescenta-se os atributos. O atributo id é a chave primária da entidade, por isso é seguido de *primary key not null*, os demais atributos seguem a mesma regra: primeiro o nome do atributo e em seguida seu tipo (*integer*, *varchar* ou *char*). Para os atributos do tipo *varchar* acrescenta-se um valor limite de caracteres entre parênteses.

```
CREATE TABLE usuarios (
id INTEGER PRIMARY KEY NOT NULL, nome VARCHAR (50) NOT NULL,
data_nasc VARCHAR (15), sexo CHAR, email VARCHAR (50));
```

**Fig. 11** - SQL de criação das tabelas no banco relacional. **Fonte:** Elaborado pelos autores (2013).

Após criadas as entidades e os relacionamentos no PostgreSQL, deu-se início à inserção de dados fictícios para validação das consultas. Para a inserção dos dados utilizou-se o comando *insert* já mencionado nesta pesquisa.

O comando *insert into* deve ser seguido pelo nome da entidade ou relacionamento no qual estão sendo inseridos os dados. Entre parênteses relaciona-se os atributos da entidade/relacionamento. O comando *values* indica os valores a serem inseridos, no segundo parênteses deve-se colocar os valores na mesma sequência dos atributos a que se referem. A **Fig. 12** apresenta o comando SQL para a inserção de dados na entidade usuarios.

```
INSERT INTO usuarios (id, nome, data_nasc, sexo, email)
VALUES (1, 'Usuario1', '23/04/1987', 'M', 'usuario1@hotmail.com');
```

**Fig. 12** - SQL de inserção dos dados nas tabelas do banco relacional. **Fonte:** Elaborado pelos autores (2013).

Conclui-se então a modelagem do banco relacional com a criação e alimentação das tabelas, lembrando que os dados foram inseridos manualmente apenas para posterior validação das consultas que são abordadas no sub capítulo 3.4.5 deste projeto. As demais SQL de criação e inserção dos dados estão nos apêndices 1 e 2.

### 3.4.3 Modelo do grafo

Diferentemente do modelo relacional, o banco de dados orientado a grafos foi estruturado utilizando vértices que representaram a massa de dados e arestas que são os relacionamentos entre vértices.

No modelo de banco de dados orientado a grafos, as colunas das tabelas do modelo relacional foram tratadas como propriedades dos vértices ou das arestas. O **Quadro 1** apresenta os tipos de vértices que foram inseridos à estrutura de dados e suas respectivas propriedades:

**Quadro 1** - Vértices e propriedades.

| ***VÉRTICES*** | ***PROPRIEDADES*** |
|---|---|
| *Usuário* | • Name;<br>• Tipo;<br>• Data_nasc;<br>• Sexo;<br>• Email; |
| *Cidade* | • Name;<br>• Tipo; |
| *Empresa* | • Name;<br>• Tipo; |
| *Escola* | • Name;<br>• Tipo |
| *Interesses* | • Name;<br>• Tipo; |
| *Grupos* | • Name; |

| |
|---|
| • Tipo; |

**Fonte**: Elaborado pelos autores (2013).

Em todos os vértices, a propriedade “name” contém como valor, o nome do objeto a ser manipulado. Já a propriedade “tipo” possui a descrição do tipo do vértice que está sendo tratado. Ambas as propriedades serão utilizadas como índices para a obtenção da informação específica dentro do grafo.

Inicialmente, as arestas foram definidas após a fase de coleta de dados, estas arestas foram: a) conhecer; b) gostar; c) estudar; d) trabalhar; e) participar; f) residir. Na aresta “conhecer” existe a possibilidade de esta ligação possuir tipos diferentes, ou seja, um ou mais vértices podem possuir uma relação de amizade ou família com outro vértice. Desta forma, a propriedade “tipo” cujos valores são: “amigo” ou “família”, foi vinculada com arestas do tipo “Conhecer”.

Após a definição da estrutura do grafo, foi elaborado um conjunto inicial de nove vértices e doze arestas, a fim de representar a estrutura criada e verificar a consistência dos resultados. A **Fig. 13** ilustra um diagrama contendo os dados inseridos ao banco para permitir a elaboração e execução das consultas.

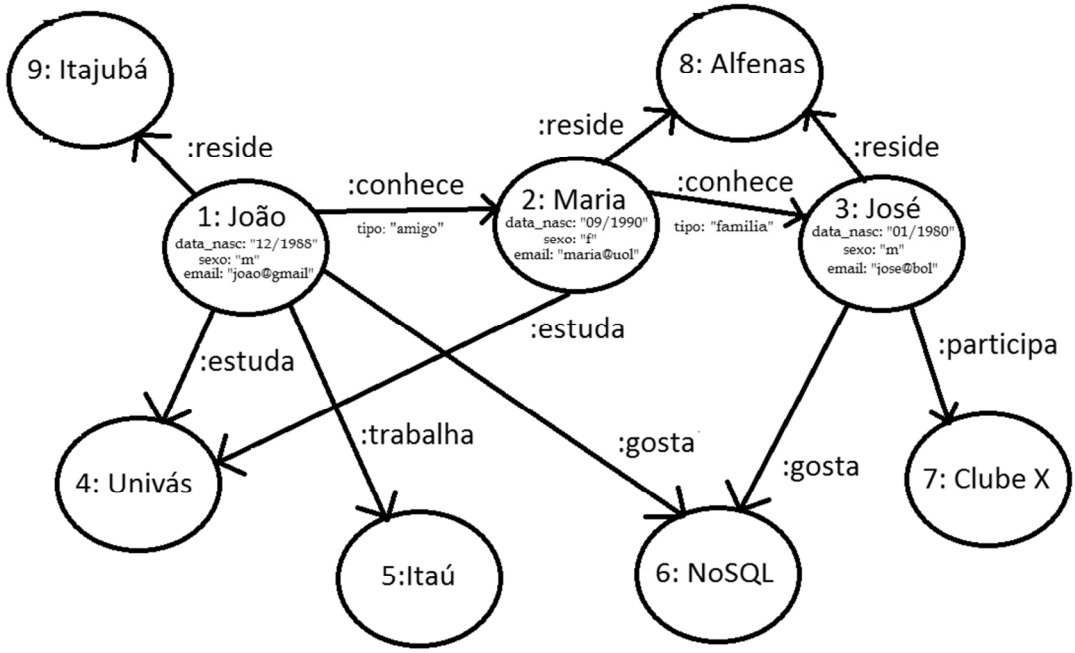


**Fig. 13** – Grafo de exemplo/teste das ferramentas. **Fonte:** Elaborado pelos autores (2013).

Os comandos Cypher presentes na **Fig. 14** foram executados no console do Neo4J para a inserção dos vértices de testes.

```
CREATE (joao{name:'joao' data_nasc:'12/1988', sexo:'m', email:'joao@gmail', tipo:'usuario'});
CREATE (maria{name:'maria' data_nasc:'09/1990', sexo:'f',email:'maria@uol', tipo:'usuario'});
CREATE (jose{name:'jose' data_nasc:'01/1980', sexo:'m', email:'jose@bol', tipo:'usuario'});
CREATE (univas{name:'univas', tipo:'escola'});
CREATE (itau{name:'itau' , tipo:'empresa'});
CREATE (noSql{name:'noSQL' , tipo:'interesse'});
CREATE (clubex{name:'clube x', tipo:'grupo'});
CREATE (alfenas{name:'alfenas', tipo:'cidade'});
CREATE (itajuba{name:'itajuba', tipo:'cidade'});
```

**Fig. 14** - Comandos Cypher para criação de vértices. **Fonte:** Elaborado pelos autores (2013).

A **Fig. 15** ilustra os comandos para a ligação por arestas entre três dos vértices criados anteriormente. Utilizando a linguagem Cypher, foi possível criar arestas de diferentes tipos em uma mesma linha de código. No exemplo, o comando START busca os vértices desejados e o comando CREATE cria as arestas entre eles.

```
START joao=node(1),maria=node(2),univas=node(4)
CREATE joao-[:conhece {tipo:'amigo'}]->maria-[:estuda]->univas;
```

**Fig. 15** - Comando Cypher para criação de arestas**. Fonte**: Elaborado pelos autores (2013).

Em resumo, o comando START, assim como na navegação do grafo, define um ou mais vértices que servem como ponto de partida. Desta forma, o comando CREATE cria as arestas com base nestes pontos.

### 3.4.4 Configurações

As ferramentas utilizadas para a implantação das bases de dados foram instaladas e configuradas de maneira a atender o contexto desta pesquisa.

#### 3.4.4.1 PostgreSQL

Para a utilização do PostgreSQL é necessário o *download* do executável do programa (.exe) disponível no site oficial (Postgresql, 2013). Ao acessar o site, clicar na aba *download* onde seleciona-se o sistema operacional que, neste trabalho é o Windows. Após clicar no sistema operacional, clicar em *download* novamente. O próximo passo é escolher a versão do PostgreSQL desejada e clicar na referente ao sistema operacional utilizado, a versão que utilizamos é a 8.4, em seguida o *download* se inicia automaticamente. Finalizado o *download*

Os comandos SQL utilizados na elaboração das consultas foram: *select*, *from*, *join*, *on*, *where* e *and*. Os comandos *select* e *from* são utilizados para selecionar os campos da tabela a qual se deseja retirar informação. Para a junção de tabelas utilizou-se os comandos *join* e *on*. O comando *where* é a condicional da consulta e o *and* é utilizado quando se tem mais de uma condição.

A **Tabela 1** contêm os códigos para a pergunta "Quais são seus contatos?" Em SQL, para buscar as informações de quem são os contatos utilizou-se uma consulta aninhada, ou seja, uma consulta SQL dentro de outra consulta SQL. Na primeira consulta utilizou-se o comando *select* para selecionar o campo id da tabela usuarios, onde (*where*) o campo nome é igual a "Maria", este SQL retornará o id do usuário para a condicional *where* da segunda consulta.

Na segunda consulta selecionou-se (*select*) os campos nome e email da tabela contatos com junção (*join*) a tabela usuarios para (*on*) o campo usuario_id_amigo da tabela contatos iguais ao campo id da tabela usuarios onde (*where*) o campo usuario_id da tabela contatos é igual ao valor retornado pela primeira consulta.

Para o código Cypher, a linha 1 define o vértice inicial de nome "Maria" que é utilizado para início da navegação presente na linha 2. O resultado é apresentado ao console no final da linha 3.

**Tabela 1** - Consulta 1.

| | | |
|---|---|---|
| Quais os contatos da Maria? | SQL: | *1. SELECT u.nome, u.email*<br>*2. FROM contatos c*<br>*3. JOIN usuarios u ON c.usuario_id_amigo=u.id*<br>*4. WHERE c.usuario_id=(*<br>*5. SELECT id from usuarios*<br>*6. WHERE nome='Maria');* |
| | Cypher: | *1. START usuario=node:node_auto_index(name='Maria')*<br>*2. MATCH usuario-[:CONHECE]->contato*<br>*3. RETURN contato;* |

**Fonte:** Elaborado pelos autores (2013).

A **Tabela 2** contêm os códigos para a pergunta "Quais os contatos que moram em determinada cidade, que também trabalham em determinada empresa?".

Para tal pergunta executada com comandos SQL, a consulta dividiu-se em duas: CONSULTA1 e CONSULTA2. A primeira consulta (CONSULTA1) foi utilizada para buscar os contatos do "João" que moram em "Itajubá", a segunda consulta (CONSULTA2) buscou os contatos que trabalham no "Itaú". Ambas as consultas armazenam seus resultados em uma tabela temporária, ao final faz-se a junção das duas tabelas temporárias e retorna-se apenas os contatos que estão presentes em ambas.

No código Cypher, definiu-se um vértice inicial através do comando START que, com o auxílio do índice automático, retornou o vértice cuja propriedade “name” é igual a “João”. A linha 2 mostra o processo de navegação por arestas do tipo “CONHECE” que saem do vértice inicial, armazenando na referência “contato”, os vértices que recebem as arestas do mesmo tipo. Prossegue-se a navegação, desta vez colhendo os vértices que recebem arestas do tipo “RESIDE” a partir da referência “contato” criada anteriormente e armazenando tais vértices na referência “cidade”. Com o comando “WHERE”, foi possível filtrar o resultado obtido com a navegação, armazenando somente vértices cuja a propriedade “name” é igual a “Itajubá”.

Ainda no comando Cypher, a linha 3 realiza uma nova navegação (*traversal*) pelo grafo a partir nos vértices presentes na referência “contato” criada pela navegação anterior. Desta forma, a referência “emp” contêm os vértices que recebem arestas do tipo “TRABALHA” e são posteriormente filtrados pela propriedade “name” cujo valor é “Itaú”. O comando “RETURN” retorna o resultado, apresentando o mesmo no console do Neo4J.

**Tabela 2** - Consulta 4.

| | | |
|---|---|---|
| Quais os contatos do João que moram em Itajubá e que trabalham no Itaú? | SQL: | *1. SELECT CONSULTA1.*<br>2. FROM<br>3. (<br>4. SELECT u.nome, u.id<br>5. FROM contatos c<br>6. JOIN usuarios u ON c.usuario_id_amigo=u.id<br>7. JOIN usuario_page r ON u.id=r.usuario_id<br>8. JOIN tipos_rel_page trp ON trp.id=r.tipos_rel_page_id<br>9. JOIN page p ON p.id=r.page_id<br>10. WHERE c.usuario_id=(<br>11. SELECT id<br>12. FROM usuarios<br>13. WHERE nome='Joao') AND trp.nome='Live' AND p.nome='Itajuba'<br>14. )<br>15. CONSULTA1<br>16. JOIN (<br>17. SELECT u.nome, u.id<br>18. FROM contatos c<br>19. JOIN usuarios u ON c.usuario_id_amigo=u.id<br>20. JOIN usuario_page r ON u.id=r.usuario_id<br>21. JOIN tipos_rel_page trp ON trp.id=r.tipos_rel_page_id<br>22. JOIN page p ON p.id=r.page_id<br>23. WHERE c.usuario_id=(<br>24. SELECT id<br>25. FROM usuarios<br>26. WHERE nome='Joao') AND trp.nome='Work' AND p.nome='Itau'* |

| | | |
|---|---|---|
| | | 27. *) CONSULTA2 ON CONSULTA1.id = CONSULTA2.id* |
| | Cypher: | 1. *START usuario=node:node_auto_index(name='João')*<br>2. *MATCH usuario-[:CONHECE]->contato-[:RESIDE]->cidade WHERE cidade.name="Itajubá"*<br>3. *WITH contato MATCH contato-[:TRABALHA]->emp*<br>4. *WHERE emp.name="Itaú" RETURN contato;* |

**Fonte**: Elaborado pelos autores (2013).

A **Tabela 3** apresenta os códigos para a pergunta: "Quem é da minha família?". No SQL, essa consulta também é aninhada, sendo assim, foi feita a junção das tabelas contatos, usuarios e tipos_rel_contato. A primeira consulta retorna o id do usuário "Maria" para a condicional da segunda consulta que por fim traz o nome dos usuários que são familiares de "Maria".

O comando Cypher é semelhante à primeira pergunta, no entanto, a linha 3 filtra arestas pela propriedade "tipo" que por sua vez, deve possuir valor igual a "familia".

**Tabela 3** - Consulta 2.

| | | |
|---|---|---|
| Quem é da família do usuário Maria? | SQL: | 1. *SELECT u.nome*<br>2. *FROM contatos c*<br>3. *JOIN usuarios u ON c.usuario_id_amigo=u.id*<br>4. *JOIN tipos_rel_contato t ON c.tipos_rel_contato_id = t.id*<br>5. *WHERE c.usuario_id=(*<br>6. *SELECT id*<br>7. *FROM usuarios*<br>8. *WHERE nome='Maria') AND t.tipo='família';* |
| | Cypher: | 1. *START usuario=node:node_auto_index(name='Maria')*<br>2. *MATCH usuario-[r:CONHECE]->contato*<br>3. *WHERE r.tipo="familia" RETURN contato;* |

**Fonte:** Elaborado pelos autores (2013).

A **Tabela 4** possui os códigos para a pergunta: "Quais os contatos que moram em determinada cidade?". Em SQL, fez-se a junção das tabelas contatos, usuarios e tipos_rel_page.

A consulta realizada em Cypher se assemelha às anteriores, no entanto, a linha 2 realiza a navegação por duas arestas diferentes, já a linha 3 filtra o resultado pela propriedade "name" dos vértices retornados do tipo cidade.

**Tabela 4 -** Consulta 3.

| | | |
|---|---|---|
| Quais os contatos do João que moram em Alfenas? | SQL: | 1. *SELECT u.nome*<br>2. *FROM contatos c*<br>3. *JOIN usuarios u ON c.usuario_id_amigo=u.id*<br>4. *JOIN usuario_page r ON u.id=r.usuario_id*<br>5. *JOIN tipos_rel_page trp ON trp.id=r.tipos_rel_page_id*<br>6. *JOIN page p ON p.id=r.page_id*<br>7. *WHERE c.usuario_id=(SELECT id*<br>8. *FROM usuarios*<br>9. *WHERE nome='Joao') AND trp.nome='Live' AND p.nome='Alfenas';* |
| | Cypher: | 1. *START usuario=node:node_auto_index(name='João')*<br>2. *MATCH usuario-[:CONHECE]->contato-[:RESIDE]->cidade*<br>3. *WHERE cidade.name="Alfenas" RETURN contato;* |

**Fonte:** Elaborado pelos autores.

A **Tabela 5** apresenta os códigos da pergunta "Do que os amigos gostam?". Em SQL, utilizaram-se dois *selects*.

Em Cypher, o comando MATCH da linha 2 navega pelo grafo colhendo os vértices cujas arestas chegam do vértice inicial criado na linha 1. O filtro é realizado na linha 3, utilizando a propriedade "tipo" da aresta CONHECE.

**Tabela 5** - Consulta 5.

| | | |
|---|---|---|
| Do que os amigos do usuário João gostam? | SQL: | 1. *SELECT u.nome, p.nome*<br>2. *FROM contatos c*<br>3. *JOIN usuarios u ON c.usuario_id_amigo=u.id*<br>4. *JOIN usuario_page r ON u.id=r.usuario_id*<br>5. *JOIN tipos_rel_page trp ON trp.id=r.tipos_rel_page_id*<br>6. *JOIN page p ON p.id=r.page_id*<br>7. *WHERE c.usuario_id=(*<br>8. *SELECT id*<br>9. *FROM usuarios*<br>10. *WHERE nome='Joao') AND trp.nome='Like';* |
| | Cypher: | 1. *START usuario=node:node_auto_index(name='João')*<br>2. *MATCH usuario-[r:CONHECE]->contato-[:GOSTA]->interesse*<br>3. *WHERE r.tipo="amigo" RETURN interesse;* |

**Fonte**: Elaborado pelos autores (2013).·.

A **Tabela 6** contêm os códigos para a pergunta "De quais os grupos os amigos participam?". Realizou-se a junção das tabelas: contatos, usuarios, tipos_rel_contato e usuario_grupo para o modelo relacional.

O código Cypher é semelhante à pergunta anterior, no entanto, foram utilizadas arestas do tipo CONHECE e PARTICIPA, conforme linha 2.

**Tabela 6 -** Consulta 6.

| | | |
|---|---|---|
| Quais os grupos que os amigos do usuário João participam? | SQL: | 1. *SELECT u.nome, g.nome*<br>2. *FROM contatos c*<br>3. *JOIN usuarios u ON c.usuario_id_amigo=u.id*<br>4. *JOIN tipos_rel_contato trc ON trc.id=c.tipos_rel_contato_id*<br>5. *JOIN usuario_grupo ug ON u.id=ug.usuario_id*<br>6. *JOIN grupo g ON g.id=ug.grupo_id*<br>7. *WHERE c.usuario_id=(*<br>8. *SELECT id*<br>9. *FROM usuarios*<br>10. *WHERE nome='Joao') AND trc.tipo='amigo';* |
| | Cypher: | 1. *START usuario=node:node_auto_index(name='João')*<br>2. *MATCH usuario-[r:CONHECE]->contato-[:PARTICIPA]->grupo*<br>3. *WHERE r.tipo="amigo" RETURN grupo;* |

**Fonte**: Elaborado pelos autores (2013).

Na próxima sessão, será apresentado o algoritmo de inserção de dados, que teve como objetivo popular as bases de dados antes da coleta dos tempos de resposta das consultas aqui apresentadas.

### 3.4.6 Inserção de dados

Para que fosse possível executar as consultas, foram elaborados duas classes para inserção de dados fictícios aos bancos de dados. A linguagem Java foi utilizada juntamente com a IDE Eclipse, cuja configuração está descrita na seção **3.4.4.3**, que contém as informações necessárias para que haja a comunicação entre os bancos de dados com o código Java.

Os códigos foram escritos com o objetivo de inserir dados aleatórios aos bancos de dados, respeitando os tipos de relações entre eles[6]. A quantidade de dados que o algoritmo insere é dinâmica, ou seja, definiram-se constantes que representam o total de registros bem como as relações entre eles para cada tipo de dado. Desta forma, foi possível popular as bases de dados com quantidades diferentes.

Na classe GraphDB, que possui os códigos de inserção para o modelo orientado a grafos, conforme ilustrado na **Fig. 20**, definiu-se o objeto graphDb para a manipulação no banco de dados. Os objetos do tipo Node representam os vértices do grafo, onde é possível executar métodos como criar, apagar e adicionar propriedades a eles. Para armazenar tais

[6] Os tipos de dados e suas relações foram descritas na seção 3.4.1

ambas as arquiteturas foram inseridos um total de 1.000.230 registros[7] ao modelo relacional e orientado a grafos, sendo que estes registros representam os tipos de dados definidos no início da implementação[8]. Deste total, foram realizadas 98.000.000 de ligações (ou relacionamentos) para cada registro[9]. A **Tabela 7** contém a média do tempo de resposta para cada consulta em ambos os bancos de dados com esta primeira quantidade.

**Tabela 7** - Tempo médio de execuções.

| **Consultas** | **Tempo médio de execução no modelo relacional (ms)** | **Tempo médio de execução no modelo orientado a grafos (ms)** |
|---|---|---|
| Quais são seus contatos? | 301,33 | 10,33 |
| Quem é da família? | 304,78 | 11,89 |
| Quais os contatos que moram em determinada cidade? | 317,11 | 12,56 |
| Quais os contatos que moram em determinada cidade, que também trabalham em determinada empresa? | 530,56 | 14,11 |
| Do que os amigos gostam? | 313,78 | 45,6 |
| Quais os grupos que os amigos participam? | 301,67 | 24,00 |

**Fonte**: Elaborado pelos autores (2013).

Com o resultado desta primeira análise, os bancos de dados foram remodelados, desta vez com menos dados. Foram inseridos 100.036 registros e 11.100.000 ligações entre eles. A **Tabela 8** apresenta a média do tempo de resposta das consultas para cada modelo de banco de dados com esta segunda quantidade.

**Tabela 8 -** Tempo médio de execuções com menos dados**.**

| **Consultas** | **Tempo médio de execução no modelo relacional (ms)** | **Tempo médio de execução no modelo orientado a grafos (ms)** |
|---|---|---|
| Quais são seus contatos? | 53,67 | 11,56 |
| Quem é da família? | 54,00 | 9,0 |
| Quais os contatos que moram em determinada cidade? | 21,00 | 12,89 |
| Quais os contatos que moram em determinada cidade, que também trabalham em determinada empresa? | 60,56 | 17,00 |
| Do que os amigos gostam? | 59,11 | 25,44 |

[7] No banco de dados orientado a grafos, estes registros representam os vértices.
[8] Os tipos de dados foram descritos na seção 3.4.1
[9] No banco de dados orientado a grafos, estes relacionamentos representam as arestas.

| | | |
|---|---|---|
| Quais os grupos que os amigos participam? | 55,33 | 21,78 |

**Fonte:** Elaborado pelos autores (2013).

Para obter as médias nas duas análises, cada consulta foi executada 10 vezes nos banco de dados de maneira sequencial, ou seja, em cada execução, registrou-se o tempo de resposta e após a décima execução foi calculado a média. No entanto, devido ao *cache* que cada banco de dados implementa, o tempo de resposta da primeira execução tende a ser incerto e varia de acordo com cada *hardware*, sendo assim, o tempo da primeira execução foi descartado da média, que foi obtida entre a segunda e a décima execução.

Para inserir os dados nas duas análises o algoritmo[10] de inserção foi utilizado, alterando-se apenas a quantidade de dados.

[10] A discussão do código presente no algoritmo foi discutida na sessão 3.4.6

Realizando o *commit* dos dados, evita-se que a memória virtual exceda o limite imposto anteriormente e reduz o tempo final do procedimento. Os comandos "tx.success()" e "tx.finish()" são utilizados para realizar o *commit* dos dados presentes na memória para o banco de dados. Após o *commit*, inicia-se uma nova transação por meio do método "beginTx()" (Rocha, 2013).

## 4.2 Execuções das consultas

As consultas definidas durante o desenvolvimento desta pesquisa[11], mostrados na **Tabela 9**, foram executadas em ambos os bancos de dados após a modelagem, definição e inserção de dados às mesmas.

**Tabela 9**: Consultas.

| | |
|---|---|
| **Consulta 1** | Quais os contatos da Maria? |
| **Consulta 2** | Quem é da família do usuário Maria? |
| **Consulta 3** | Quais os contatos do João que moram em Alfenas? |
| **Consulta 4** | Quais os contatos do João que moram em Itajubá e que trabalham no Itaú? |
| **Consulta 5** | Do que os amigos do usuário João gostam? |
| **Consulta 6** | Quais os grupos que os amigos do usuário João participam? |

**Fonte**: Elaborado pelos autores (2013).

Inicialmente, cada consulta foi executada 10 vezes de maneira sequencial em cada base de dados a fim de se obter as médias dos tempos de execução e compará-los, levando-se em consideração os valores de *cache* que cada sistema utiliza.

Como já descrito por Robinson *et al*. (2013), a recomendação feita pela Neotechnology ao se trabalhar com grafos de grande dimensão, é utilizar *flash drives* ou mídias de armazenamento ágeis para a garantia de desempenho, o tempo de resposta em muitos casos depende também de sistema operacional e memória. No entanto, para suprir a deficiência de *hardware*, o Neo4J trabalha com *caches* que buscam otimizar consultas em grafos, relacionando identificadores de vértices e arestas com as respectivas propriedades. O *cache* armazena essas informações no decorrer da navegação no grafo, sendo assim, o desempenho é obtido a partir de sucessivos *traversals*. Esta característica é conhecida como "cache quente" (Ignatowicz, 2012).

[11] As consultas foram as mesmas descritas na sessão 3.4.5

**Tabela 10** - Tempo médio de execuções com mais dados.

| Consultas | Tempo médio no modelo relacional (ms) | Tempo médio no modelo orientado a grafo (ms) |
|---|---|---|
| **Consulta 2** | 315,75 | 23,25 |
| **Consulta 5** | 346,75 | 73,25 |
| **Consulta 6** | 323,75 | 32,5 |

**Fonte:** Elaborado pelos autores (2013).

O desempenho de consultas realizadas em bancos de dados NoSQL é uma das principais características abortadas por Brito (2010) e Robinson *et al*. (2013). Este ganho no tempo para a execução de consultas pode ser explicado pela ausência de esquema, ou estrutura fixa, fazendo com que a modelagem de dados se adapte ao problema, e não o contrário. A necessidade de chaves primárias e estrangeiras imposta pelo modelo relacional para que haja consistência dos dados é outro fator que, dependendo do problema, afeta o desempenho.

De acordo com Brito (2010, p. 2), em se tratando de SGDB relacional, "Com o crescimento do número de usuários, o sistema começa a ter uma queda de *performance*". A fim de se observar tal comportamento, as consultas também foram executadas nas bases de dados com um menor número de dados. Foram inseridos um total de 100.000 usuários que possuíam 54 amigos e 6 familiares cada. Também foi criado um total de 4 registros de escolas, empresas e cidades, além de 4 interesses que foram posteriormente relacionados aleatoriamente com cada usuário.

A **Tabela 11** apresenta o tempo médio de resposta para cada consulta executada 10 vezes de maneira sequencial. Novamente, a média foi obtida desconsiderando a primeira execução em ambos os bancos de dados.

**Tabela 11** - Tempo médio de execução com menos dados.

| **Consultas** | **Tempo médio no modelo relacional (ms)** | **Tempo médio no modelo orientado a grafos (ms)** |
|---|---|---|
| **Consulta 1** | 56,6 | 11,56 |
| **Consulta 2** | 54 | 9 |
| **Consulta 3** | 21 | 12,89 |
| **Consulta 4** | 60,56 | 17 |
| **Consulta 5** | 59,11 | 25,44 |
| **Consulta 6** | 55,33 | 21,78 |

**Fonte:** Elaborado pelos autores (2013).

Com a inserção e execução de consultas com aproximadamente 100.000 dados, foi observado que o banco de dados orientado a grafos obteve novamente maior desempenho, ficando entre 56% e 83% a frente do modelo relacional. No entanto, a diferença entre as arquiteturas se reduziu com a menor quantidade de dados inseridos. Com os resultados, comprova-se o estudo realizado por Brito (2010).

### 4.3 Complexidade

Com a modelagem e elaboração de consultas, ficou evidente a diferença de complexidade em ambos os bancos de dados quando se trabalha em um cenário de redes sociais. Nitidamente, percebe-se que as redes sociais são grafos, onde cada dado, seja pessoa, grupo ou página em geral, são tratados como vértices e a relação entre eles como arestas. Desta forma, não houve dificuldades em modelar o banco de dados orientado a grafos, diferentemente do banco de dados relacional, que necessitou de uma visão mais sistemática sobre cada tipo de dado e organizá-los em tabelas, para então relacioná-las por meio de chaves primárias e estrangeiras.

Como descrito por Lóscio, Oliveira e Pontes (2011, p.8): "A vantagem de utilização do modelo baseado em grafos fica bastante clara quando consultas complexas são exigidas pelo usuário". Tal afirmação pode ser percebida na **Tabela 12**, a codificação das consultas na linguagem Cypher não exigiu grande conhecimento técnico se comparado ao modelo relacional, cujas consultas elaboradas em SQL tornaram-se extensas na medida em que a complexidade aumentava, necessitando conhecer por completo a modelagem do banco para realizar validações de chaves primárias, estrangeiras e consulta aninhada.

**Tabela 12** - SQL X Cypher.

| | | |
|---|---|---|
| Quais os contatos da Maria? | SQL: | *1. SELECT u.nome, u.email*<br>*2. FROM contatos c*<br>*3. JOIN usuarios u ON c.usuario_id_amigo=u.id*<br>*4. WHERE c.usuario_id=(*<br>*5. SELECT id from usuarios*<br>*6. WHERE nome='Maria');* |
| | Cypher: | *1. START usuario=node:node_auto_index(name='Maria')*<br>*2. MATCH usuario-[:CONHECE]->contato*<br>*3. RETURN contato;* |

**Fonte:** Elaborado pelos autores (2013).

Já em Cypher, com uso de tipos de dados com nomes semânticos, é possível navegar entre os vértices por meio de arestas direcionadas, desta forma, elimina-se o esquema fixo, tornando a consulta mais dinâmica.

# CONCLUSÃO

Esta pesquisa foi concluída pelos acadêmicos após a realização de estudos teóricos sobre cada tecnologia e a implementação prática utilizando ferramentas específicas. Os objetivos deste trabalho foram alcançados ao final da coleta dos resultados e a apresentação destes com fundamentação teórica.

O banco de dados orientado a grafos possui modelagem dinâmica, não é dependente de estrutura fixa e sua arquitetura é de fácil compreensão. A ferramenta Neo4j, utilizada para a modelagem e manipulação de dados orientados a grafos, mostrou-se ágil, simples e útil, por fornecer recursos como índices, console de comandos, interface de gerenciamento *web* e uma extensa documentação, tornando-a de grande importância para a comunidade de desenvolvedores NoSQL.

Percebeu-se ainda que a linguagem Cypher possui uma sintaxe simples, utiliza caracteres para representar arestas e desta forma, favorece a compreensão do código sem que haja a necessidade de grande conhecimento técnico, pois a codificação se assemelha a forma como o ser humano lê o grafo.. As consultas elaboradas em Cypher se destacam pela utilização de poucas linhas de código se comparada com a linguagem SQL do modelo relacional.

Com a modelagem de dados baseados em redes sociais, notaram-se grandes diferenças entre o banco de dados Neo4J e o PostgreSQL. O tempo médio de resposta foi consideravelmente menor no modelo de dados orientado a grafos, que se manteve entre 38% e 97% mais ágil que o banco de dados relacional. Dentre outros fatores, a linguagem Cypher se mostrou ideal no tratamento de dados orientados a grafos, sendo uma das principais características do banco Neo4J. No entanto, a integridade de dados é um fator que deve ser levado em conta, pois com o esquema fixo de tabelas presentes no modelo relacional, pode-se garantir maior segurança.

Com este resultado, conclui-se que em se tratando de sistemas que possuem uma estrutura de dados que se assemelhem a redes sociais e demais grafos, o Neo4J é a ferramenta ideal, tendo como principal vantagem a *performance* e a dinâmica na modelagem dos dados. Contudo, é necessário analisar o tipo de sistema para definir a melhor solução de banco de dados. Conclui-se também que a arquitetura de dados NoSQL pode e deve ser utilizada em conjunto com o modelo relacional para alcançar a qualidade dos sistemas computacionais, favorecendo tanto o cliente quanto o desenvolvedor de *software*.

**REFERÊNCIAS**


AGUIAR, S. **Redes Sociais na Internet: Desafios à Pesquisa**. 2007. Disponível em: < http://www.sitedaescola.com/downloads/portal_aluno/Maio/Redes%20sociais%20na%20internet-%20desafios%20%E0%20pesquisa.pdf> Acesso em: 25 abr. 2013.

ALVES, Gabriel Costa. CARVALHO, Danilo Otávio de. **Análise de problemas de performance em aplicações JEE com o uso de ferramentas livres** – Pouso Alegre/MG: Univás/2012.

BONFIOLI, G. F**. Banco de Dados Relacional e Objeto-Relacional: Uma Comparação Usando PostgreSQL.** 2006. Disponível em: < http://www.bcc.ufla.br/monografias/2005/Banco_de_dados_relacional_e_objeto_relacional_uma_comparacao_usando_postgresql.pdf> Acesso em: 28 abr. 2013.

BRITES, C. M. **Banco de Dados NOSQL Conceitos e Aplicabilidade** . 2012. Disponível em: <http://ged.feevale.br/bibvirtual/Monografia/MonografiaClaudiomiroBrites.pdf> Acesso em: 24 fev. 2013.

BRITO, R. W. **Bancos de Dados NoSQL x SGBDs Relacionais: Análise Comparativa** . 2010. Disponível em: <http://www.infobrasil.inf.br/userfiles/27-05-S4-1-68840-Bancos%20de%20Dados%20NoSQL.pdf> Acesso em: 29 mar. 2013.

CAELUM. **Apostila java e orientação a objetos**. 2013. Disponível em: <http://www.caelum.com.br/apostila-java-orientacao-objetos/o-que-e-java/> Acesso em: 23 jul. 2013.

CLARO, D. B., SOBRAL, J. B. M. **Programação em Java**. Santa Catarina: Copyleft Pearson Education, 2008.

CONGO, M; **Radar Tecnológico**. 2013. Disponível em: < http://blogs.estadao.com.br/radar-tecnologico/2013/08/01/linkedin-lucra-32-mais-e-chega-a-238-milhoes-de-usuarios/> Acesso em: 12 ago, 2013

COSTA, E. R. da. **Banco de Dados Relacionais**. 2011. Disponível em: < http://www.fatecsp.br/dti/tcc/tcc0025.pdf> Acesso em: 28 abr. 2013.

DANTAS, A. **Folguedos, pontes e navegador de posicionamento global por satélite.** 2010. Disponível em: < http://autoentusiastas.blogspot.com.br/2010/08/folguedos-pontes-e-gps.html > Acesso em: 22 abr. 2013

DATE, C. J. **Introdução a Sistemas de Bancos de Dados**. Tradução de Daniel Vieira. 8ª Ed. Rio de Janeiro: Elsevier, 2003.

DIANA, M. D.; GEROSA, M. A. **NOSQL na Web 2.0: Um Estudo Comparativo de Bancos Não-Relacionais para Armazenamento de Dados na Web 2.0** . 2010. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wtdbd/2010/sbbd_wtd_12.pdf> Acesso em: 24 fev. 2013.

DOUGHERTY, H. Pesquisa de Marketing. **Facebook.com generates nearly 1 in 4 page views in the US**. Disponível em: < http://www.experian.com/blogs/marketing-forward/2010/11/19/facebook-com-generates-nearly-1-in-4-page-views-in-the-us/> Acesso em 08 abr. 2013.

FACEBOOK. **Central de ajuda Facebook.** 2013. Disponível em < https://www.facebook.com/help> Acesso em: 13 ago. 2013

FACEBOOK. **Introducing Graph Search.** 2013. Disponível em < https://www.facebook.com/about/graphsearch> Acesso em: 13 ago. 2013

FEOFILOFF, P.; KOHAYAKAWA, Y.; WAKABAYASHI, Y. **Uma Introdução Sucinta à Teoria dos Grafos.** 2011. Disponível em: <http://www.ime.usp.br/~pf/teoriadosgrafos/texto/TeoriaDosGrafos.pdf>. Acesso em 22 abr. 2013.

FOLHA.COM. **Entenda o que é a Web 2.0**. 2006. Disponível em: <http://www1.folha.uol.com.br/folha/informatica/ult124u20173.shtml> Acesso em: 29 mar. 2013.

HEUSER, C. A. **Projeto de Banco de Dados**. 4ª Ed. Porto Alegre: Sagra Luzzatto, 2001.

IGNATOWICZ, E. **Neo4 o que? Uma visão prática do banco de dados orientado a grafos.** Palestra realizada dia 30 de agosto de 2012. Disponível em: <http://www.infoq.com/br/presentations/neo4j-grafos > Acesso em 01 mar. 2013.

IMASTER. **NoSQL - você realmente sabe do que estamos falando?.** 2010. Disponível em: <http://imasters.com.br/artigo/17043/banco-de-dados/nosql-voce-realmente-sabe-do-que-estamos-falando>. Acesso em: 17 abr. 2013.

LINKEDIN. **Linkedin**. 2013. Disponível em: <http://www.linkedin.com> Acesso em: 29 ago. 2013.

LÓSCIO, B. F.; OLIVEIRA, H. R.; PONTES, J. C. S. **NoSQL no desenvolvimento de aplicações Web colaborativas**. 2011. Disponível em: <http://www.addlabs.uff.br/sbsc_site/SBSC2011_NoSQL.pdf> Acesso em: 29 mar. 2013.

MACHADO, F.; ABREU, M. **Projeto de Banco de Dados: Uma Visão Prática**. São Paulo: Érica, 2004.

MACHADO, J. R.; TIJIBOY, A. V. **Redes Sociais Virtuais: Um Espaço para Efetivação da Aprendizagem Cooperativa.** 2005. Disponível em: < http://www.inf.ufes.br/~cvnascimento/artigos/a37_redessociaisvirtuais.pdf> Acesso em: 25 abr. 2013.

MADDEN, M; LENHART, A; CORTESI, S; GASSER, U; DUGGAN, M; SMITH, A; BEATON, M; **Teens, Social Media, and Privacy.** 2013. Disponível em: < http://www.pewinternet.org/Reports/2013/Teens-Social-Media-And-Privacy/Summary-of-Findings.aspx> Acesso em: 28 ago. 2013.

MATSUSHITA, R.S.I.; NGUESSAN, D. **Framework para integração de serviços móveis baseado em rede social.** Disponível em <http://www.fatecsaocaetano.edu.br/fascitech/index.php/fascitech/article/view/55/54> Acesso em: 15 mar. 2013.

MORAIS, J.M. **Teoria de grafos**. 2013. Disponível em: <http://www.laps.ufpa.br/jefferson/Graduacao/ProjetoAlgoritmosI/aula_grafos_ufpa_08022012.pdf > Acesso em 22 abr 2013.

NEOTECHNOLOGY. **The Neo4j Manual v1.9.M04**. Disponível em: <http://docs.neo4j.org/chunked/1.9.M04/index.html > Acesso em: 01 mar. 2013.

NEOTECHNOLOGY. **What is a graph database?**. Disponível em: <http://docs.neo4j.org/chunked/milestone/what-is-a-graphdb.html> Acesso em: 20 abr. 2013.

ROCHA, R, R; **Algoritmos de Particionamento e Banco de Dados Orientado a Grafos.** 2013. Universidade Federal de Itajubá. Dissertação de mestrado.

RECUERO, R**. Redes Sociais na Internet**. Porto Alegre: Meridional, 2009.

ROBINSON, I.; WEBBER, J.; EIFREN, E. **Graph Databases.** E-book. California: O'Reilly, 2013.

SANTOS, C. G. J. **Tipos de pesquisa**. 2013. Disponível em <http://www.oficinadapesquisa.com.br/APOSTILAS/METODOL/_OF.TIPOS_PESQUISA.PDF> Acesso em: 16 ago. 2013.

SCHELP, D. **Nos Laços (Fracos) da Internet**. 2009. Disponível em: <http://veja.abril.com.br/080709/nos-lacos-fracos-internet-p-94.shtml> Acesso em: 21 abr. 2013.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados**. Tradução de Daniel Vieira. 5ª Ed. Rio de Janeiro: Elsevier, 2006.

SOUZA, A. C.; AMARAL, H. R.; LIZARDO, L. E. O. **PostgreSQL: uma alternativa para sistemas gerenciadores de banco de dados de código aberto**. In: Anais do Congresso Nacional Universidade, EAD e Software Livre. 2012.

TAKAI, O. K.; ITALIANO, I. C.; FERREIRA, J. E. **Introdução a Banco de Dados**. USP, 2005.

TAYLOR, A; JONES, A. **Cypher Query Language**. 2012. Disponível em: <http://www.slideshare.net/graphdevroom/cypher-query-language> Acesso em: 26 jul. 2013.

# APÊNDICES

# ANEXOS