UNIVERSIDADE FEDERAL DE SÃO PAULO

CAMPUS SÃO JOSÉ DOS CAMPOS

Trabalho Final de Aspectos de Implementação de Banco de Dados

ÉRIC FADUL CUNHA YOSHIDA - 148143

HENRIQUE GARCIA CAMPANHA - 150970

LIDYANE KÜSTER - 163897

LUIZ EDUARDO CASELLA - 156702

STELLA HADASSA ALVES VIEIRA - 156.587

ASPECTOS DA IMPLEMENTAÇÃO DE BANCO DE DADOS

Professora Doutora Daniela Musa

11/09/2024

Introdução

Este relatório detalha o desenvolvimento de um banco de dados de grafos utilizando o sistema de gerenciamento de banco de dados Neo4J, destinado a modelar as relações complexas entre personagens, locais, itens e afiliações no universo de Star Wars. O objetivo é permitir consultas intuitivas e eficazes sobre estas relações.

1. Tecnologias Utilizadas

Para a realização deste projeto, adotamos uma combinação de tecnologias que inclui:

- **Neo4J**: Um banco de dados de grafos que se destaca por sua eficiência em gerenciar e consultar relações complexas entre dados interconectados.
- **HTML, CSS e JavaScript**: Utilizados para desenvolver a interface gráfica em que o usuário interage com o banco de dados. O HTML estrutura o conteúdo da página, o CSS é usado para o estilo e o JavaScript gerencia a lógica interativa e as chamadas de banco de dados.
- **Neo4j JavaScript Driver**: Permite a integração direta entre o front-end da aplicação web e o banco de dados Neo4J, facilitando a execução de comandos Cypher através de interações do usuário na interface gráfica.

2. Como Funciona o SGBD Neo4j

Funcionamento do Neo4J

O Neo4J é um sistema de gerenciamento de banco de dados que utiliza a estrutura de grafos para armazenar e gerenciar dados. Ao contrário dos bancos de dados relacionais que utilizam tabelas, o Neo4J armazena informações em nós (nodes), arestas (relationships) e propriedades.

- **Nós**: Representam entidades, como personagens, locais e itens.
- **Arestas**: Representam as relações entre esses nós, como afiliações, relações familiares ou posse de itens.
- **Propriedades**: Informações adicionais que podem ser anexadas tanto aos nós quanto às arestas.

O uso de grafos permite uma modelagem mais intuitiva e direta das relações naturais existentes nos dados, facilitando consultas complexas que são intensivas e lentas em sistemas tradicionais, especialmente quando envolvem múltiplas junções.

Inserção de Dados

A inserção de dados no Neo4J é feita através da linguagem de consulta Cypher, que é desenhada especificamente para trabalhar com grafos. Cypher permite definir e manipular grafos com uma sintaxe que reflete visualmente a estrutura de um grafo.

Exemplo de Inserção de Dados: Para criar um personagem no banco de dados, usamos o comando `CREATE`:

```
CREATE (n:Character {name: 'Luke Skywalker', homeworld: 'Tatooine'})
```

Este comando cria um nó do tipo `Character` com propriedades para o nome e o mundo natal.

3. Ideia de Implementação

Optamos pelo uso do Neo4J para este projeto devido ao desafio de analisar e visualizar as complexas redes de relações no universo de Star Wars. A narrativa de Star Wars, repleta de interações entre personagens envolvendo alianças, conflitos familiares e evoluções pessoais ao longo de diversos episódios, apresenta uma oportunidade única para aplicar um banco de dados orientado a grafos. O Neo4J se destaca por sua capacidade de mapear e gerenciar eficientemente essas conexões complexas, tornando-o uma ferramenta excepcionalmente adequada para explorar como os personagens e eventos estão intrinsecamente ligados. A escolha desta tecnologia visa proporcionar uma compreensão mais profunda das dinâmicas internas desta saga icônica, permitindo uma exploração detalhada e sistematizada das suas ricas inter-relações.

4. Modelo de Dados

O modelo de dados adotado emprega a estrutura de grafos do Neo4J, detalhada a seguir:

Nós

Cada entidade do universo de Star Wars é representada como um nó com atributos relevantes. Por exemplo:

- **Personagens**: `(:Character {name: 'Luke Skywalker', homeworld: 'Tatooine'})`

- **Locais**: `(:Place {name: 'Tatooine'})`
- **Itens**: `(:Item {name: 'Lightsaber'})`

Relações

As relações entre os nós refletem as diversas interações e conexões narrativas:

- **Afiliações**: `(luke)-[:AFFILIATED_WITH]->(:Affiliation {name: 'Rebel Alliance'})`
- **Relações de Mestre e Aprendiz**: `(yoda)-[:TRAINED_BY]->(luke)`

Propriedades

Cada nó e relação possui propriedades que detalham informações adicionais, como a origem de um personagem ou a natureza de uma relação.

5. Etapas de Construção do Banco de Dados

a. Criação dos Personagens

Iniciamos o banco de dados criando nós para cada personagem com seus respectivos atributos como nome e mundo natal. Usamos o comando Cypher `CREATE` para estabelecer esses nós no banco de dados.

```
CREATE
(luke:Character {name: 'Luke Skywalker', homeworld: 'Tatooine'}),
...
(anakin:Character {name: 'Anakin Skywalker', homeworld: 'Tatooine'});
```

b. Definição das Afiliações

Após os personagens, definimos as afiliações principais como Rebel Alliance, Galactic Empire, entre outros. Estes também são criados como nós no banco.

```
CREATE
(rebel:Affiliation {name: 'Rebel Alliance'}),
...
(republic:Affiliation {name: 'Galactic Republic'});
```

c. Registro dos Locais

Os locais como Tatooine, Alderaan e outros são estabelecidos no banco como nós, identificando os principais locais onde os eventos ocorrem ou onde os personagens residem.

```
CREATE
(tatooine:Place {name: 'Tatooine'}),
...
(naboo:Place {name: 'Naboo'});
```

d. Inclusão de Itens Importantes

Itens significativos como Lightsabers e X-Wings são adicionados ao banco, permitindo relações de posse entre personagens e itens.

```
CREATE
(lightsaber:Item {name: 'Lightsaber'}),
...
(jedicane:Item {name: 'Jedi Cane'});
```

e. Estabelecimento de Relacionamentos de Afiliação

Relacionamentos são criados para conectar personagens às suas respectivas afiliações usando o comando CREATE com especificação do tipo de relação, como AFFILIATED_WITH.

```
MATCH
(luke:Character {name: 'Luke Skywalker'}),
...
(padme:Character {name: 'Padmé Amidala'})
CREATE
(luke)-[:AFFILIATED_WITH]->(rebel),
...
(padme)-[:AFFILIATED_WITH]->(republic);
```

f. Relacionamentos de Treinamento e Mestria

Estabelecemos quem treinou quem e, inversamente, quem é mestre de quem utilizando o tipo de relação TRAINED_BY e criando automaticamente o inverso MASTER_OF.

```
MATCH
(luke:Character {name: 'Luke Skywalker'}),
```

```
...
(anakin:Character {name: 'Anakin Skywalker'})
CREATE
(luke)-[:TRAINED_BY]->(obiwan),
...
(anakin)-[:TRAINED_BY]->(qui_gon);
```

g. Relacionamentos de Parentesco e Residência

Parentescos e residências dos personagens são mapeados para ilustrar conexões familiares e locais de habitação.

```
MATCH
(luke:Character {name: 'Luke Skywalker'}),
...
(anakin:Character {name: 'Anakin Skywalker'})
CREATE
(luke)-[:RELATIVE_OF]->(vader),
...
(anakin)-[:LIVED_IN]->(tatooine);
```

h. Atribuição de Itens aos Personagens

Finalmente, associamos personagens a itens específicos que possuem, refletindo posses como sabres de luz e naves.

```
MATCH
(luke:Character {name: 'Luke Skywalker'}),
...
(anakin:Character {name: 'Anakin Skywalker'})
CREATE
(luke)-[:OWNS]->(lightsaber),
...
(anakin)-[:OWNS]->(lightsaber);
```

6. Desenvolvimento da Interface de Usuário

Para proporcionar uma interação rica e envolvente com o banco de dados Neo4J, desenvolvemos uma página web que não só permite aos usuários consultar e inserir dados, mas também visualizar as complexas inter-relações do universo de Star Wars de maneira intuitiva e tematicamente apropriada.

Tecnologias Utilizadas

- HTML5: Estrutura a base do conteúdo da página, definindo os elementos como botões, seletores e áreas de entrada.
- CSS3: Fornece estilização visual para melhorar a experiência do usuário, incluindo cores, layouts e design responsivo que se adapta a diferentes tamanhos de tela.
- JavaScript: Gerencia a lógica de interação entre a interface do usuário e o banco de dados Neo4J, manipulando eventos do DOM e realizando chamadas ao banco de dados através do driver Neo4J JavaScript.

Implementação Detalhada

A página é caracterizada por um design temático de Star Wars, com fundo que evoca o espaço e cores que remetem à franquia. A interação do usuário é facilitada através de uma série de abas interativas que segmentam as funcionalidades da aplicação:

Tabs de Navegação

- Buscar Personagem: Permite aos usuários procurar detalhes específicos sobre os personagens, como planeta de origem e afiliações.
- Procurar Conexão: Encontra relações diretas ou indiretas entre dois personagens, como laços familiares ou mestre-aprendiz.
- Adicionar Personagem/Relacionamento: Facilita a inserção de novos personagens e o estabelecimento de novas relações entre personagens existentes.

Campos de Entrada e Botões

- Campos de texto e seletores para inserir ou selecionar informações específicas sobre personagens ou relações.
- Botões para submeter consultas, que interagem diretamente com o banco de dados e retornam informações que são então exibidas na página.

Visualização de Dados

- Elementos de exibição condicional que mostram detalhes sobre personagens ou conexões apenas quando relevantes, melhorando a clareza e a usabilidade da interface.

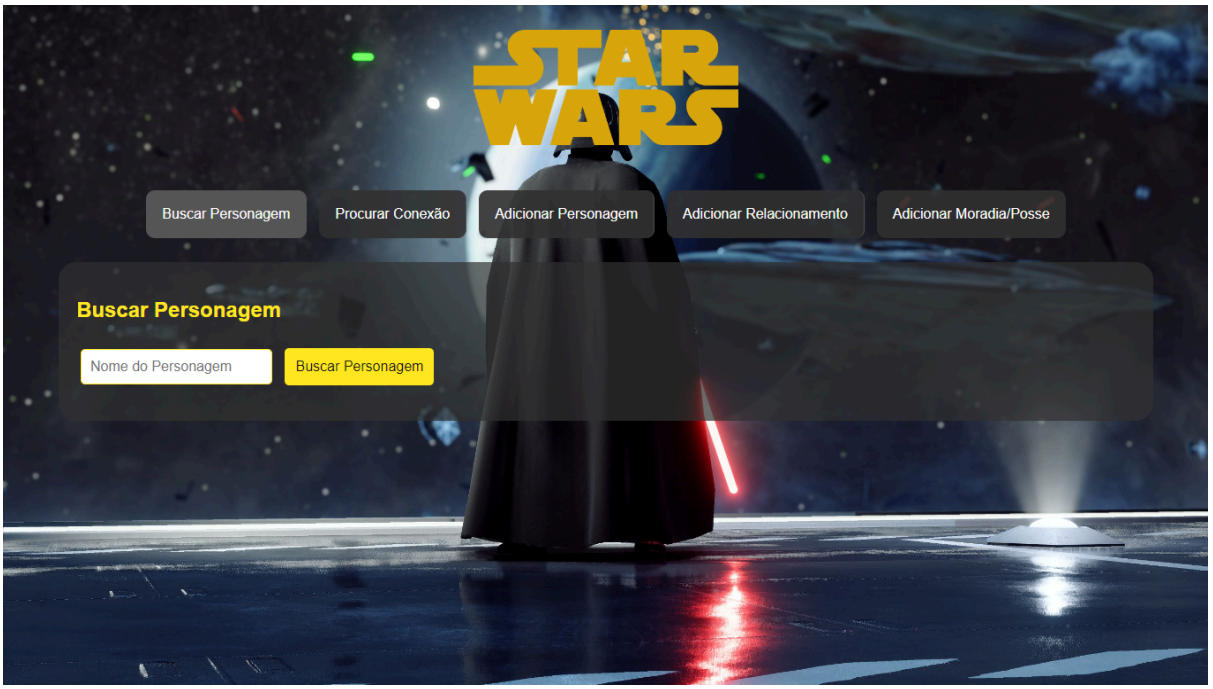
Código Exemplo: Busca de Personagem

```
function searchCharacter() {
    const characterName = document.getElementById('characterName').value;
```

```
  const query = `MATCH (character:Character {name: '${characterName}'}) RETURN character`;

  session.run(query)

    .then(function(result) {

      displayCharacterDetails(result.records[0].get('character'));

    })

    .catch(function(error) {

      console.error('Error fetching character details:', error);

    });

}
```

Este trecho de código JavaScript exemplifica como a aplicação captura o nome de um personagem inserido pelo usuário, executa uma consulta Cypher no Neo4J e então processa o resultado para exibir os detalhes do personagem.

A estética da página é inspirada no tema de Star Wars. A página é responsiva e projetada para ser fácil de usar, incentivando a interação contínua com o banco de dados.



7. Vantagens e Desvantagens do Banco Orientado a Grafo (Neo4J)

Vantagens do Uso do Neo4J

As vantagens destacadas do uso do Neo4J em nosso projeto incluem:

1. **Modelagem Intuitiva de Relações Complexas:**
    - Neo4J facilita a representação de relações complexas através de sua estrutura de grafos, tornando mais simples e direto o mapeamento das interações entre personagens, lugares e itens.
2. **Agilidade nas Consultas de Relacionamento:**
    - Devido à estrutura de grafos, o Neo4J permite realizar consultas de relacionamento de forma mais rápida e eficiente, especialmente em comparação com bancos de dados relacionais que requerem múltiplas junções.
3. **Visualização de Dados:**
    - A estrutura do Neo4J suporta visualizações gráficas dos dados, o que pode ser extremamente útil para entender as relações e padrões dentro dos dados complexos.
4. **Facilidade de Expansão de Esquema:**
    - O esquema no Neo4J é flexível, permitindo adições e mudanças sem a necessidade de reestruturar o banco de dados inteiro, o que é ideal para projetos que podem evoluir ou expandir com o tempo.
5. **Representação Natural de Dados e Flexibilidade para Mudanças:**
    - Neo4J oferece uma maneira mais natural de representar dados, refletindo como as informações são interconectadas na vida real. Isso adiciona uma camada de flexibilidade para adaptações e mudanças, facilitando a manutenção e atualização do banco.

Desvantagens do Uso do Neo4J

As desvantagens associadas ao uso do Neo4J incluem:

1. **Complexidade de Consulta:**
    - Embora poderoso, Cypher (a linguagem de consulta do Neo4J) pode ser complexo para usuários novos ou para aqueles acostumados apenas com SQL tradicional, resultando em uma curva de aprendizado acentuada.
2. **Gerenciamento de Transações:**

   - O gerenciamento de transações no Neo4J pode ser mais complexo que em bancos de dados relacionais, especialmente em cenários que requerem alta consistência e integridade de transação.
3. **Limitações de Integração:**
   - Integrar o Neo4J com outros sistemas e tecnologias pode apresentar desafios, especialmente se outras partes do sistema de TI não são otimizadas para trabalhar com grafos.
4. **Custo de Escalabilidade:**
   - Escalar um banco de dados de grafos pode ser custoso, tanto em termos de desempenho quanto financeiramente, particularmente quando o volume de dados e a complexidade das consultas aumentam.
5. **Overkill para Dados Simples:**
   - Para projetos com requisitos de dados simples, onde as interações são mínimas ou lineares, usar um banco de dados de grafos pode ser mais do que necessário, resultando em complexidade e custo desnecessários.
6. **Custo de Performance em Grandes Grafos:**
   - Embora o Neo4J seja eficiente para manipular relações complexas, o desempenho pode degradar conforme o tamanho do grafo e o número de relações crescem, requerendo otimizações e possivelmente hardware mais robusto.

Conclusão

A escolha do Neo4J para o mapeamento do universo de Star Wars provou ser extremamente eficaz. Este sistema de banco de dados orientado a grafos oferece uma maneira poderosa e intuitiva de visualizar e analisar as complexas redes de relacionamentos que permeiam a saga. Com ele, conseguimos não apenas compreender melhor as interconexões entre os personagens e eventos, mas também proporcionamos uma plataforma onde os fãs e pesquisadores podem explorar essas relações de maneira interativa e detalhada. A implementação do Neo4J neste contexto não apenas atendeu às nossas expectativas, mas também abriu novas possibilidades para a análise de dados narrativos em universos ficcionais complexos como o de Star Wars.