Mestrado em Engenharia Informática

UNIVERSIDADE
DE ÉVORA

Tarefa NoSQL

Modelo Orientado a Grafos Neo4j

Disciplina:
Tópicos Avançados de Bases de Dados
Docente :
Irene Rodrigues
Elaborado por:
João Pedro Amaral Dias 42055

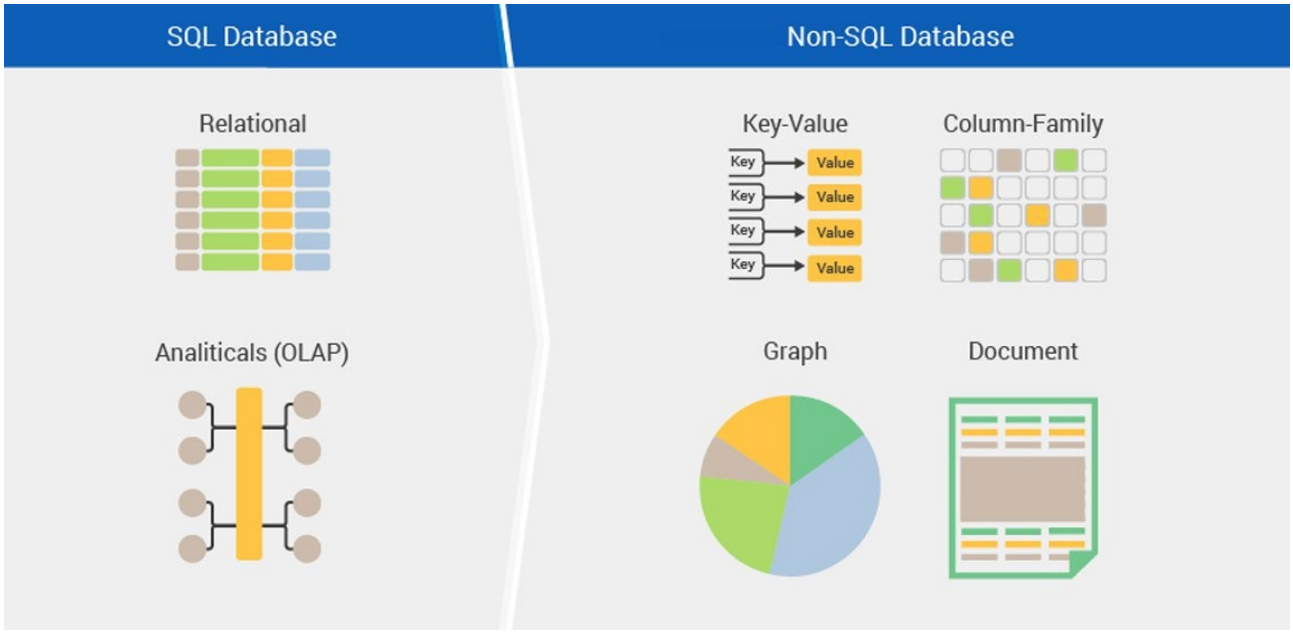
1. NOSQL

Uma base de dados NoSQL é um modelo de bases de dados não relacional que veio oferecer novas alternativas ao uso das tradicionais base de dados relacionais. Esta solução não modela os seus dados através da criação de tabelas (relações), mas sim através de outros modelos como o orientado a grafos, orientado a documentos, orientado a colunas ou até mesmo o modelo chave-valor.

Apesar de ser denominada por NoSQL, isto não significa que esta classe de base de dados não suporte a tradicional liguaguem SQL mas sim que suporta mais métodos de consulta de dados para além do SQL, passando a ser reconhecido mais recentemente por Not Only SQL.

As base de dados NoSQL existem desde cerca de 1960 mas apenas na ultima decada começaram a ser cada vez mais reconhecidas e utilizadas. Isto deveu-se ao aparecimento da necessidade de gerir enormes conjuntos de dados (Big Data), como é o caso das redes sociais onde utilizar base de dados relacionais não é uma escolha viável. Daí o uso de diferentes leques de bases de dados NoSQL para estes tipos problemas ser a aposta mais acertada, pois estas bases de dados tem foco na performance e na escalabilidade.

Isto não significa que as bases de dados NoSQL venham substituir por completo as bases de dados tradicionais, pois existem várias situações onde implementar uma base de dados relacional é a escolha mais acertada.



2. Modelo Orientado a Grafos

O modelo orientado a grafos é um dos tipos de modelos da classe NoSQL a par do orientado a documentos e o modelo chave-valor. Mas a vantagem desde modelo é a maneira como simplifica as relações entre dados, ou seja a maneira como simplifica os tradicionais “joins” usados nos modelos relacionais.

Neste modelo para podermos inserir os dados não é preciso criar tabela, chaves primárias nem outro tipo de estrutura pré-definida. Para criar dados basta criar um nó de um tipo, preencher as suas propriedades e caso seja necessário fazer a ligação a outro nó.

Por exemplo podemos criar um nó do tipo aluno inserindo nesse mesmo nó a informação respetiva a apenas esse aluno. Depois de o nó estar criado, a qualquer momento é possível fazer a ligação a outro nó, por exemplo, disciplina. A ligação entre nós deve ter um significado como, por exemplo, “frequentou em 2014”. Este pequeno gráfo traduz nos a informação de que aquele aluno frequentou aquela disciplina no ano de 2014. No futuro e com um grafo mais complexo se quisermos pesquisar quais as disciplinas que aquele aluno frequentou em 2014, basta procurar pelos arcos que “saiem” do nó aluno para o nó disciplina com o ano 2014.

Este método simplifica bastante a relação entre tipos diferentes de dados. Para o exemplo ilustrado anteriormente se tivessemos a trabalhar numa base de dados relacional iriamos de ter uma tabela aluno e uma tabela disciplina juntamente com uma tabela para relacionar as duas. Para consultar informação teriamos de usar um “join” na query, que para base de dados com um vasto número de dados é uma operação que se torna bastante cara.

3. Neo4j

O Neo4j, criado em 2007, é uma base de dados que segue o modelo orientado por gráfos da classe NoSQL. Apesar de ser um modelo de base de dados não relacional é conhecido por seguir as permissas ACID (Atomicidade, Consistência, Isolação, Durabilidade), sendo esta uma das razões que o torna a base de dados com grafos mais popular do mundo.

Neo4j foi totalmente desenvolvido em Java e atualmente todos os dados inseridos nesta base de dados são armazenados em nós, arcos ou atributos, tornando-o um modelo de fácil compreensão. Os nós e os arcos podem ter mais que um atributo que o caracteriza e contém uma label para serem facilmente identificados.

A pesquisa nesta base de dados é feita pela linguagem Cypher desde a versão 2.0 do Neo4j. Esta linguagem recentemente passou a suportar indices tornando as consultas de dados muito mais rápidas. A sintaxe é declarativa e inspirada na linguagem SQL. Em baixo encontra-se um exemplo de uma query em Cypher que procura uma pessoa chamada Jennifer que goste da tecnologia grafos.

MATCH (p:Person {name: “Jennifer”})-[rel:LIKES]→(g:Technology {type : “Graphs”})
*RETURN **

Pode se interagir com uma base de dados Neo4j através de uma aplicação desktop ou aplicação browser. A aplicação desktop é usada para gerir uma instancia local da base de dados enquanto que a aplicação browser contém uma interface para fazer querys à base de dados e visualizar os dados.

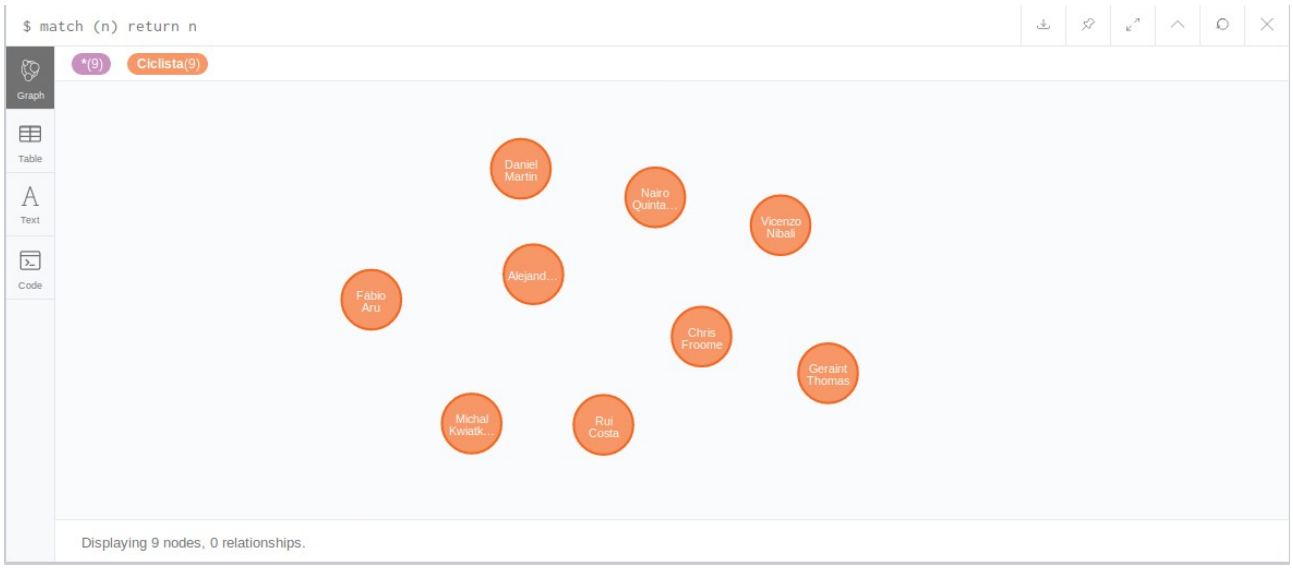
3.1 Base de dados em Neo4j

De modo a ser testado este modelo orientado por grafos, será implementado uma base de dados em Neo4j. Esta base de dados irá conter informação de vários ciclistas, as suas respectivas equipas e a sua classificação em multiplas provas.

Numa primeira fase irão ser criados todos os nós referentes aos ciclistas, através do comando:

CREATE (n:Ciclista {nome:"Chris Froome"}) RETURN n ,

onde o atributo nome do nó ciclista vai ser diferente para cada ciclista. Após inserirmos alguns nós ficamos com o seguinte grafo.



O mesmo será feito para a criação dos nós das várias equipas e das várias corridas a serem realizadas.

Após termos inseridos os nós podemos adicionar mais atributos a qualquer um deles, como por exemplo adicionar a idade a alguns ciclistas ou a data de cada corrida. Um exemplo desta alteração é exemplificada na seguinte query em Cypher.

MATCH (n:Ciclista {nome:"Chris Froome"}) SET n.idade = 33 RETURN n

Para concluir o nosso grafo falta apenas adicionar as relações entre nós. Neste caso iremos adicionar uma relação do nó ciclista para o nó equipa de modo a ilustrar que um certo ciclista pertence a uma equipa. Se nesta relação adicionarmos um atributo de data de inicio e outro de data fim, podemos criar várias relações entre o mesmo nó ciclista e diferentes nós equipa, mostrando assim o percurso do ciclista pelas várias equipas.

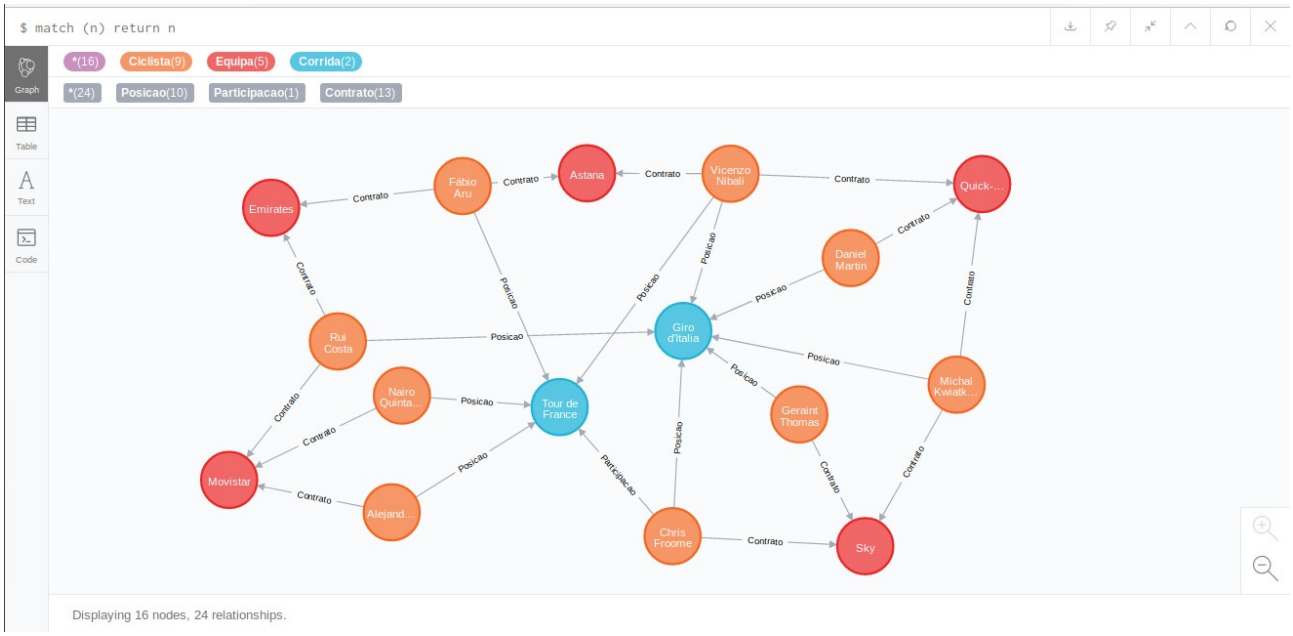
MATCH (c:Ciclista),(e:Equipa) WHERE c.nome = "Chris Froome" AND e.nome = "Sky"

CREATE (c)-[r:Contrato {ano_inicio:2007}]->(e) RETURN r

Será também adicionada uma relação entre os nós ciclistas e os nós corrida com o atributo lugar de modo a definir a posição de um certo ciclista na respetiva corrida.

MATCH (c:Ciclista),(e:Corrida) WHERE c.nome = "Chris Froome" AND e.nome = "Tour de France" CREATE (c)-[r:Posicao{lugar:1}]->(e) RETURN r

O nosso modelo orientado a grafos vai ficar da seguinte forma.



Com a base de dados definida podemos executar querys para fazer pesquisa de dados ou até mesmo fazer a ligação a uma aplicação exterior que poderá vir buscar dados a esta base de dados.

Exemplo de uma query que retorna todos os ciclistas que ganharam uma corrida.

MATCH (c:Ciclista)-[r:Posicao {lugar:1}]->(e:Corrida) RETURN c

Resultado:

Maio 2019