Tecnologia NoSQL

João Calhau m36764
José Pimenta m37158
Universidade de Evora, Portugal

5 de Maio de 2017

Resumo

Com a evolução da tecnologia, a necessidade de guardar informação tornou-se muito importante, com o número de dispositivos também a crescer e com os dispositivos cada vez mais fáceis de obter, surgiram algumas questões que precisam de ser tratadas. O acesso de informação por múltiplas pessoas ao mesmo tempo precisava de ser tratado, o maior tamanho de informação disponível necessitou de maior rapidez de acesso, maior rapidez de processamento de informação e informação em qualquer sitio, a qualquer altura. É aqui que entra o desenvolvimento de bases de dados NoSQL, que acabaram por facilitar a resolução destes problemas. [Cou17]

Keywords: NoSQL, Database, Documents, Graphs, Columns, Key-Value

1 Introdução

Com este artigo pretendemos verificar como bases de dados NoSQL podem beneficiar a evolução tecnológica e o aumento de informação que disso também advém, quais são os modelos que representam NoSQL, o que são, e quais poderão ser as diferenças entre eles, verificando os benefícios e malefícios deles. Seguidamente em mais profundidade iremos verificar o modelo chave-valor, mostrando algumas implementações desse modelo e como pode ser uma melhoria comparado com um modelo de bases de dados relacional.

2 NoSQL

NoSQL (Not Only SQL, não apenas SQL) é considerado uma alternativa viável a bases de dados relacionais SQL (Linguagem de Consulta estruturada), tendo como objetivo a manutenção de grandes quantidades de dados distribuídos, sobretudo em *Cloud* e *Web Applications*, dando maior importância a *Performance* e escalabilidade do que consistência de dados, providenciada por bases de dados relacionais. [RVB17]
NoSQL usa vários tipos diferentes de modelos, entre os quais modelo Orientado a Colunas, Orientado a Documentos, Orientado a Grafos e modelo Chave-Valor (o qual iremos falar em maior detalhe).

2.1 Modelo Orientado a Colunas

Modelo orientado a colunas transforma uma base de dados numa partição gigante vertical composta por colunas individuais que são guardadas separadamente. Ao guardá-las separadamente permitem uma eficiência de leitura maior do que bases de dados tradicionais, porque em vez de lerem linhas inteiras do disco (no caso de discos magnéticos a agulha ter de movimentar entre zonas de cada coluna) e descartar os dados das colunas que não são necessárias, leem apenas os atributos das colunas necessárias. Claro que isto não quer dizer que sejam perfeitas, aliás a este tipo de bases de dados está associado um maior custo de escrita em tuplos visto ter de aceder a múltiplas vezes a colunas diferentes. [ABH+12]

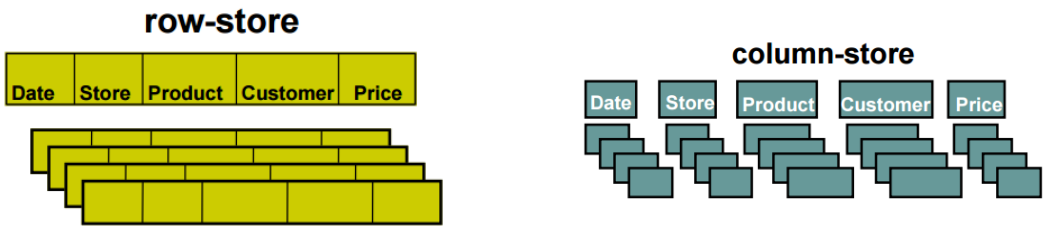

Figura 1: Comparação entre bases de dados por linha e bases de dados por coluna

2.2 Modelo Orientado a Documentos

No caso do modelo orientado a documentos, a informação é guardada de modo similar ao modelo de chave-valor, mas de modo mais complexo, pois permite guardar data com estrutura variável, podendo um documento conter um número variável de chaves e valores conforme necessário. Ao existirem documentos sem serem data relacional (como em SQL tradicional relacional), o armazenamento de documentos pode ser distribuído facilmente por vários servidores.
Uma das vantagens também deste tipo de modelos é que o esquema e estrutura de um documento ao não ser fixo, permite que a evolução e adaptabilidade ao longo do tempo seja muito mais fácil, pois a dita estrutura não tem de ser especificada antes desta ser utilizada, e assim o esquema é alterado quando se necessita sem problemas da ordem do tempo. [Inc17a]

2.3 Modelo Orientado a Grafos

Numa base de dados relacional, quando queremos resultados de duas tabelas temos de fazer conexões especiais entre ambas (*JOINS*) em bases de dados orientadas a grafos nada disso é preciso. Neste modelo cada nó de uma entidade ou atributo tem uma lista de relações que representa as relações com os outros nós (tal como num grafo). Estas relações estão organizadas por tipo e direção, que podem conter mais atributos. Quando queremos fazer uma operação de *JOIN*, a base de dados usa a lista e tem acesso direto aos nós conectados. Os modelos de dados resultantes destas relações são muito mais simples do que aquelas produzidas por bases de dados tradicionais. [Inc17c]

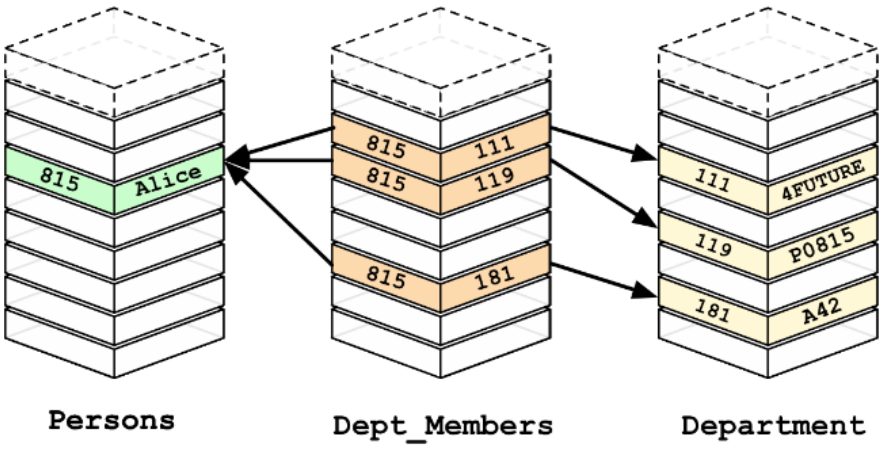

Figura 2: Modelo orientado a grafos (Neo4j) [Inc17c]

2.4 Modelo Chave-Valor

Modelo chave-valor é, provavelmente, a forma mais simples de sistemas de gestão de bases de dados (SGBD). Podem guardar pares chave-valor, mas só retornam valores quando sabemos a chave, visto os valores serem guardados de uma forma semelhante a uma Tabela de Hash ou um Dicionário, por causa disso, uma chave dever ser única para se poder associar a um valor (ou conjunto de valores). Este tipo de modelo costuma estar associado a boa escalabilidade, simplicidade de armazenamento, *arrays* associativos e estruturas de dados. [Ian16]

2.4.1 A Chave

A chave num modelo chave-valor tem de ser (ou deve ser em principio) única, de modo a que que o valor possa ser facilmente acedido pela chave que o identifica. O tipo de chave depende da base de dados em si e como se pretende organizar os dados, podendo a base de dados impor restrições. Para além disso, o tamanho da chave deve ser razoável, não sendo muito grande (causando problemas de performance), nem sendo muito curto, causando problemas de leitura e também de modo a ter uma quantidade considerável de chaves diferentes. [Ian16]

2.4.2 O Valor

O valor de um modelo chave-valor pode ser qualquer coisa, como texto, números, linguagens de *Markup* como HTML ou até código de programação. O valor pode ser uma lista, ou até outro par chave-valor. Dependendo do SGBD o tipo da chave pode mudar radicalmente, até pode permitir sermos nós próprios a definir o que pode ser um valor. [Ian16]

3 Redis

O Redis não é um base de dados com modelo chave-valor normal, é mais um servidor de estruturas de dados que suporta vários tipos de valores. O que isto significa é que em sistemas tradicionais chave-valor está associado a usar chaves Strings e guardar valores Strings, no Redis o valor não tem de ser necessariamente uma string, mas pode ser uma estrutura de dados mais complexa como por exemplo: [red15]

- Strings
- Listas - coleções de elementos String ordenados de acordo com a ordem de inserção.
- Sets - coleções de elementos String (únicos) desordenados.
- Hashes - mapas compostos por chaves associados a valores.

3.1 Redis - Chaves

As chaves do Redis são "binary safe", o que significa que qualquer sequência binária pode ser uma chave. Tal como referido anteriormente, o tamanho das chaves devem ter um tamanho considerável, nem muito grande nem muito pequeno, e serem fáceis de se ler, podendo ter um esquema em que separam-se palavras por pontos ou dois pontos. O tamanho em memória máximo são 512 MB. [red15]

3.2 Redis - Strings

Ao utilizar o Redis, alguns comandos são frequentemente usados, uns dos quais são o SET e o GET, na qual o SET é utilizado para substituir o valor de uma chave correspondente que já exista ou atribuir a uma chave nova. O valor máximo de valores que podem ser atribuídos é, tal como as chaves, 512 MB. O comando GET retorna o valor da chave correspondente. Um comando interessante é o comando GETSET, que atribui um novo valor a uma chave, retornando o valor antigo como resultado. Além destes comandos, existem entre outros, os comandos MSET e MGET que permitem executar SET ou GET de várias chaves ao mesmo tempo.
Em termos de existência de uma chave, esta pode existir atualmente na base de dados ou não, e para isso utiliza-se o comando EXISTS, podendo apagar-se com o comando DEL. Pode-se também verificar o tipo do valor correspondente à chave, através do comando TYPE. [red15]

3.3 Redis - Listas

As listas do Redis são implementadas via listas ligadas, o que quer dizer que até se tivermos milhões de elementos dentro de uma lista, inserção será sempre constante. O senão é que aceder a um elemento por index é rápido em listas implementadas com arrays mas não por listas ligadas.
Os comandos disponíveis para interagir com listas no Redis são: LPUSH e RPUSH para inserir á cabeça e á cauda, respetivamente, LRANGE para extrair os elementos dessa lista. Temos ainda LPOP e RPOP para remover o primeiro e o ultimo elemento da lista, respetivamente. [red15]

3.4 Redis - Sets

Os Sets no Redis são coleções desordenadas de Strings. Com o comando SADD podemos adicionar novos elementos ao set, com o comando smembers podemos ver os elementos que estão dentro do set (não ordenados). Para além de estes dois comandos podemos verificar se um elemento pertence ou não a um set utilizando sismember, que retorna um inteiro 1 ou 0 (1 em caso de pertencer, 0 em caso contrario). [red15]

3.5 Redis - Hashes

Os Hashes do Redis são exatamente o que esperaríamos de uma estrutura de dados de Hash, tem dois campos, a chave e o valor. Não há um limite prático para o que uma hash pode levar, a não ser memória, e por isso podemos usar os hashes de muitas maneiras diferentes nas nossas aplicações.
Os comandos que podemos utilizar para interagir com hashes em Redis são HMSET que insere um ou mais elementos num campo, HGET retorna um campo, HMGET retorna um array de valores de um campo. [red15]

4 Conclusão

Em conclusão, ao passo que numa base de dados relacional, guardar informação mais complexa (vídeos, imagens, gifs, entre outros) é mais complicado se não mesmo impossível, em bases de dados NoSQL, qualquer tipo de dados pode ser incorporado. As bases de dados relacionais também são mais difíceis de escalar, ao passo que as bases de dados NoSQL são preparadas para escalar automaticamente e transparentemente, à medida da quantidade de utilizadores. Finalmente, em questões financeiras, chega a ser mais barato manter e gerir uma base de dados NoSQL do que uma base de dados relacionais. Uma base de dados NoSQL está assim preparada para a grande quantidade de informação que existe atualmente, para além do diferente tipo de conteúdo que podem existir nessa dada informação. [inc17b]

Referências

[ABH+12] Daniel Abadi, Peter Boncz, Stavros Harizopoulos, Stratos Idreos, and Samuel Madden. The design and implementation of modern column-oriented database systems. *Foundations and Trends in Databases*, 5(3):197–280, 2012.

[Cou17] Couchbase. Why nosql database?, 2017.

[Ian16] Ian. What is a key-value database?, 2016.

[Inc17a] MongoDB Inc. Document databases, 2017.

[inc17b] MongoDB inc. Relational vs non-relational database, 2017.

[Inc17c] Neo Technology Inc. From relational to neo4j, 2017.

[red15] redislabs. An introduction to redis data types and abstractions, 2015.

[RVB17] Margaret Rouse, Jack Vaughan, and Barney Beal. Nosql (not only sql database), 2017.

[Str11] Christof Strauch. Nosql databases, Fevereiro 2011.