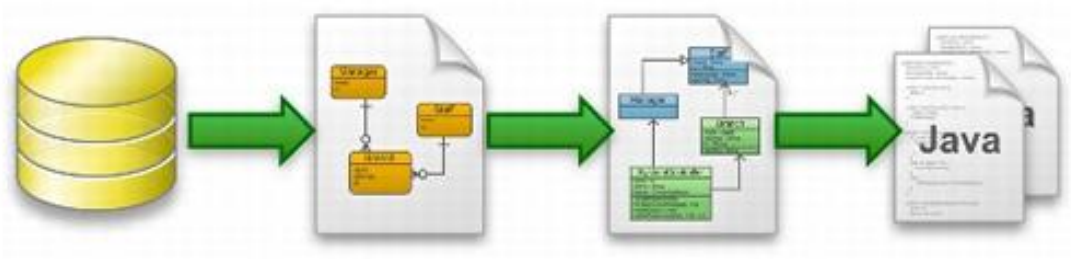
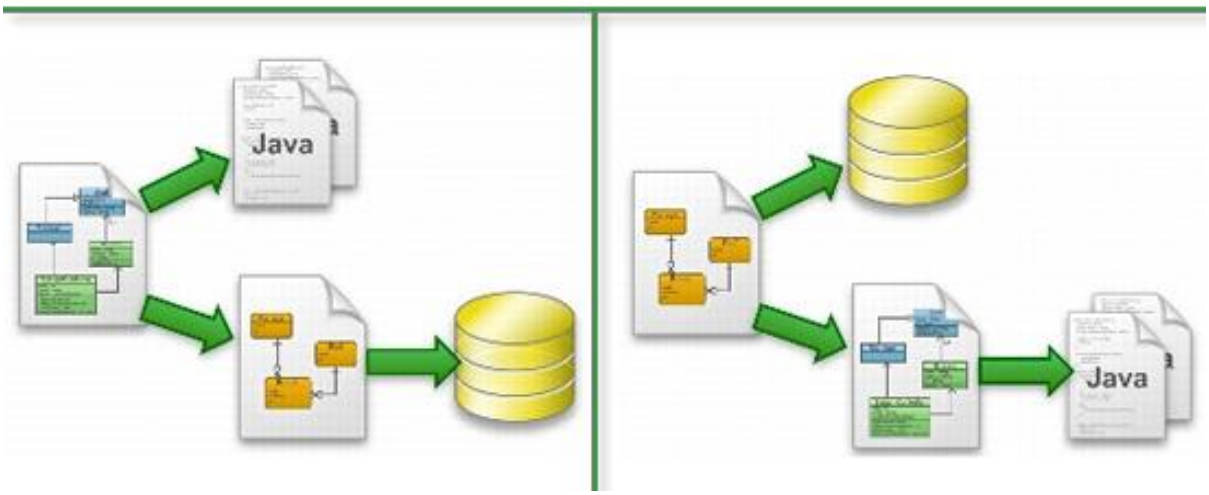
Desenvolvimento de Aplicações para WEB

Giovanne Prestes Dias 171029
Tiago Machado Maldonado 171047
Vinicius Tardelli Leite 132152

ORM e NoSQL

Mapeamento Objeto-Relacional (ORM)





O que é: ORM (Object Relational Mapper) é uma técnica de mapeamento objeto relacional que permite fazer uma relação dos objetos com os dados que os mesmos representam.

Ele faz o mapeamento da sua classe para o banco de dados e cada ORM tem suas particularidades para gerar o SQL referente a inserção do objeto que corresponde a uma tabela no banco de dados e realizar a operação. Utilizando um ORM, também se ganha produtividade, pois deixa-se de escrever os comandos SQL para deixar que o próprio ORM, faça isto por você.

Vantagens de se usar um ORM

- -Você escreve menos códigos e o programa fica muito mais produtivo;
- Seu código fica mais elegante;
- É mais fácil de dar manutenção no projeto;
- Melhora a padronização da sua aplicação.

Exemplos de ORM’s

- Django;
- Laravel;
- Hibernate;
- Ruby on Rails;
- Sequelize;
- NHibernate;
- Entity Framework;
- Codus
- Object Mapper;
- SubSonic;
- Entity Spaces;
- LLBLGen Pro;
- eXpress Persistent Objects.

Banco de Dados NoSQL

O que é:O termo NoSQL é bastante abrangente, pois envolve diversas ferramentas, tecnologias, estruturas de dados e arquiteturas. Esta nova *buzzword* representa muito mais um movimento, ou uma nova escola de pensamento, do que alguma tecnologia em particular.

O NoSQL surgiu da necessidade de uma performance superior e de uma alta escalabilidade, os atuais bancos de dados relacionais são muito restritos a isso, sendo necessária a distribuição vertical de servidores, ou seja, quanto mais dados, mais memória e mais disco um servidor precisa.

O NoSQL tem uma grande facilidade na distribuição horizontal, ou seja, mais dados, mais servidores, não necessariamente de alta performance.

As tecnologias NoSQL não têm como objetivo substituir os bancos de dados relacionais, mas apenas propor algumas soluções que em determinados cenários são mais adequadas. Desta forma, é possível trabalhar com tecnologias NoSQL e banco de dados relacionais dentro de uma mesma aplicação.

Características Bando de Dados NoSQL

Modelo de Dados

Bancos de dados não relacionais (NoSQL) normalmente não aplicam um schema. Uma “chave hash” é normalmente usada para recuperar valores, conjuntos de colunas ou documentos semiestruturados JSON, XML ou outros que contenham atributos de itens relacionados.

Propriedades

Bancos de dados NoSQL normalmente trocam algumas propriedades ACID de sistemas de gerenciamento de bancos de dados relacionais (RDBMS) por um modelo de dados mais flexível que escala horizontalmente. Essas características fazem dos bancos de dados NoSQL uma excelente opção em situações em que os RDBMS deparam com desafios de arquitetura e precisam solucionar uma combinação de gargalos de desempenho, escalabilidade, complexidade operacional e custos crescentes de administração e suporte.

Desempenho

Desempenho geralmente é uma função do tamanho do cluster do hardware subjacente, da latência de rede e da aplicação que faz a chamada.

Escala

Projetado para aumentar a escala "horizontalmente" usando clusters distribuídos de hardware de baixo custo para aumentar a transferência sem aumentar a latência.

APIs

APIs baseadas em objetos permitem que desenvolvedores de aplicações armazenem e restaurem facilmente estruturas de dados na memória. “Chaves hash” permitem que as aplicações procurem pares de chave-valor, conjuntos de colunas ou documentos semiestruturados contendo objetos de atributos de aplicação serializados.

Tipos de Bancos NoSql

Chave Valor

A ideia principal aqui é usar uma tabela hash na qual há uma chave única e um indicador de um dado ou de um item em particular.
O modelo chave-valor é o mais simples e fácil de implementar. Mas ele é ineficiente quando você somente está interessado em consultar ou em atualizar parte de um valor, entre outras desvantagens,esses tipos de bancos de dados são o que tem a maior escalabilidade e os e os que suportam maior carga de dados.

Exemplos :

- Berkeley
- DB
- Tokyo
- Cabinet
- Project
- Voldermort
- MemcacheDB
- SimpleBD

Grafos

Diferentemente de outros tipos de bancos de dados NoSQL, esse está diretamente relacionado a um modelo de dados estabelecido, o modelo de grafos. A ideia desse modelo é representar os dados e/ou o esquema dos dados como grafos dirigidos, ou como estruturas que generalizem a noção de grafos .
O modelo de grafos é mais interessante que outros quando informações sobre a inter-conectividade ou a topologia dos dados são mais importantes, ou tão importante quantos os dados propriamente ditos .
O modelo orientado a grafos possui três componentes básicos: os nós (são os vértices do grafo), os relacionamentos (são as arestas) e as propriedades (ou atributos) dos nós e relacionamentos.
Neste caso, o banco de dados pode ser visto como um multigrafo rotulado e direcionado, onde cada par de nós pode ser conectado por mais de uma aresta.
Exemplos:

- Neo4j
- InfoGrid
- Sones

- HyperGraphDB

Documentos

Este modelo permite armazenar qualquer documento, sem ter a necessidade de definir previamente sua estrutura. O documento e composto por uma inúmeros campos, com tipos de dados diversos, inclusive um campo pode conter um outro, possui uma estrutura semelhante a um arquivo XML.
Exemplos:

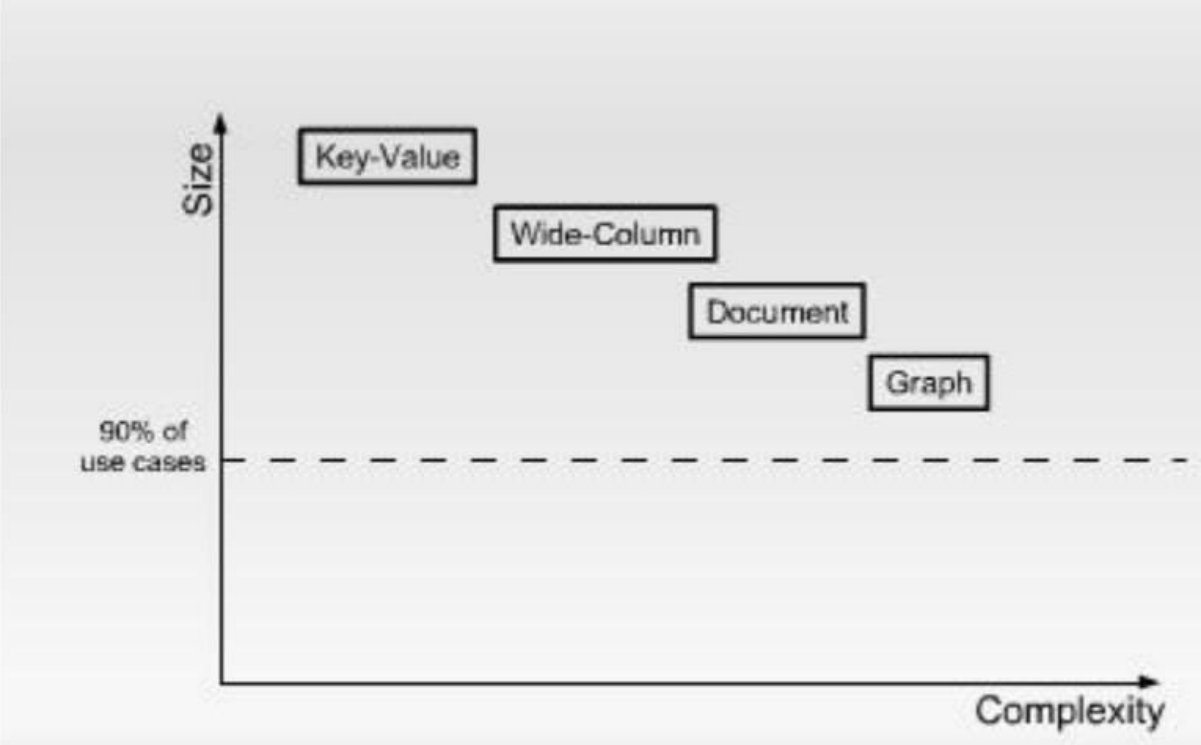
- AmazonSimpleDb
- ApacheCouchdb
- MongoDb
- Riak

Colunas

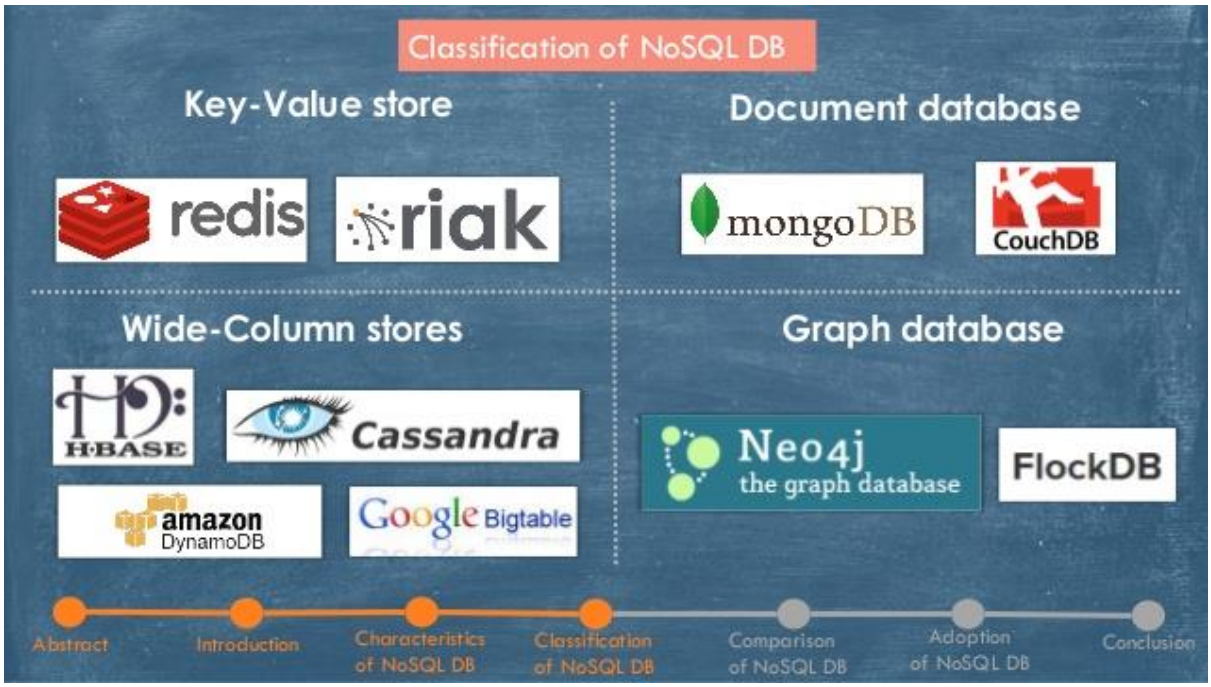
Esse modelo se tornou popular através do paper BigTable do Google, com o objetivo de montar um sistema de armazenamento de dados distribuído, projetado para ter um alto grau de escalabilidade e de volume de dados é composto por três componentes
Keyspace tem como função agrupar um conjunto de Famílias de Colunas. Semelhante a um banco de dados relacional.
Família de Colunas organizas as colunas. faz o uso de uma chave única, que traz flexibilidade ao modelo sem poluir as linhas com colunas nulas. Semelhante a uma tabela no modelo relacional.
Coluna, que é uma tupla composta por nome,timestamp e valor, onde os dados são realmente armazenados.
Duas características adicionais e importantes deste modelo são a forma de consulta, que pode ser executada apenas através da chave das linhas de uma família de colunas, e a necessidade de definir previamente o conjunto de colunas que podem ser armazenadas em cada família.
Exemplos:

- · Cassandra
- · GoogleBigTable
- · HBase
- · Hyperbase

Na imagem abaixo, podemos ver um gráfico demonstrando a diferença entre o tamanho da base de dados e a complexidade dos seus dados. Assim, podemos perceber que os bancos do tipo chave-valor conseguem suportam maior quantiadade dados, sendo que seus dados são mais simples, enquanto que os bancos do tipo grafo suportam menos, porem seus dados são mais complexos.



E na Imagem abaixo Temos a Logo de Alguns dos Banco de Dados NoSql apresentados anteriormente



Alguns exemplos de quando usar NoSQL

Logging/Arquivamento.

Ferramentas de Log-mining vêm a calhar porque elas conseguem acessar logs através de servidores, relacioná-los e analisá-los.

Insight em computação social.

Muitas empresas hoje fornecem a seus usuários a habilidade de atuar na computação social através de fóruns de mensagens blogs etc.

Integração de feed de dados externos.

Muitas empresas precisam integrar os dados oriundos de seus parceiros. Mesmo se as duas partes conduzirem várias discussões e negociações, as empresas têm pouco controle sobre o formato dos dados que chegam a elas.
Além disso, existem muitas situações em que esses formatos mudam muito frequentemente baseado na mudança das necessidades de negócios dos parceiros.

Sistemas de processamento de pedidos Front-end.

Hoje, o volume de pedidos, aplicações e serviços flutuando através de diferentes canais para comerciantes, banqueiros e seguradoras, fornecedores de entretenimento, logística etc é enorme. Esses pedidos precisam ser capturados sem nenhum interrupção sempre que um usuário final faz uma transação de qualquer parte do mundo. Depois disso, um sistema de reconciliação tipicamente os atualiza parasistemas de back-end, ao mesmo tempo que atualiza o usuário final com o seu status.

Serviço de gerenciamento de conteúdo empresarial.

O gerenciamento de conteúdo agora é utilizado através de diferentes grupos funcionais da empresa, como RH ou vendas. O desafio é congregar grupos diferentes utilizando estruturas diferentes de metadados em um serviço comum de gerenciamento de conteúdo.

Estatísticas/análises em tempo real.

Às vezes, é necessário usar o banco de dados como uma maneira de rastrear métricas de performance em tempo real para websites (visualizações de páginas, visitas únicas etc). Ferramentas como o Google Analytics são ótimas, mas não são em tempo real , às vezes, é útil construir um sistema secundário que fornece estatísticas em tempo real. Outras alterativas, como monitoramento de tráfego 24/7, são uma boa opção também.

SQL X NoSQL

Além de lidar com volumes extremamente grandes de dados dos mais variados tipos, soluções com o Big Data também precisam trabalhar com distribuição de processamento e *elasticidade*, isto é, suportar aplicações com volumes de dados que crescem substancialmente em pouco tempo.

O problema é que os bancos de dados "tradicionais", especialmente aqueles que exploram o modelo relacional, como o MySQL, o PostgreSQL e o Oracle, não se mostram adequados a estes requisitos, já que são menos flexíveis.

Isso acontece porque bancos de dados relacionais normalmente se baseiam em quatro propriedades que tornam a sua adoção segura e eficiente, razão pela qual soluções do tipo são tão populares: Atomicidade, Consistência, Isolamento e Durabilidade. Esta combinação é conhecida como *ACID*, sigla para o uso destes termos em inglês: *Atomicity*, *Consistency*, *Isolation* e *Durability*. Vejamos uma breve descrição de cada uma:

- Atomicidade: toda transação deve ser atômica, isto é, só pode ser considerada efetivada se executada completamente;
- Consistência: todas as regras aplicadas ao banco de dados devem ser seguidas;
- Isolamento: nenhuma transação pode interferir em outra que esteja em andamento ao mesmo tempo;
- Durabilidade: uma vez que a transação esteja concluída, os dados consequentes não podem ser perdidos.

O problema é que este conjunto de propriedades é por demais restritivo para uma solução de Big Data. A elasticidade, por exemplo, pode ser inviabilizada pela atomicidade e pela consistência. É neste ponto que entra em cena o conceito de **NoSQL**, denominação que muitos atribuem à expressão em inglês "*Not only SQL*", que em tradução livre significa "Não apenas SQL" (*SQL -Structured Query Language* - é, em poucas palavras, uma linguagem própria para se trabalhar com bancos de dados relacionais).

O NoSQL faz referência às soluções de bancos de dados que possibilitam armazenamento de diversas formas, não se limitando ao modelo relacional tradicional. Bancos do tipo são mais flexíveis, sendo inclusive compatíveis com um grupo de premissas que "compete" com as propriedades ACID: a *BASE* (*Basically Available*, *Soft state*, *Eventually consistency* - Basicamente disponível, Estado Leve, eventualmente consistente).

Não é que bancos de dados relacionais tenham ficado ultrapassados - eles são e continuarão por muito tempo sendo úteis a uma série de aplicações. O que acontece é que, geralmente, quanto maior um banco de dados se torna, mais custoso e trabalhoso ele fica: é preciso otimizar, acrescentar novos servidores, empregar mais especialistas em sua manutenção, enfim.

Via de regra, escalar (torná-lo maior) um banco de dados NoSQL é mais fácil e menos custoso. Isso é possível porque, além de contar com propriedades mais flexíveis, bancos do tipo já são otimizados para trabalhar com processamento paralelo, distribuição global (vários data centers), aumento imediato de sua capacidade e outros.

Além disso, há mais de uma categoria de banco de dados NoSQL, fazendo com que soluções do tipo possam atender à grande variedade de dados que existe, tanto estruturados, quanto não estruturados: bancos de dados orientados a documentos, bancos de dados chave/valor, bancos de dados de grafos, enfim.

Maiores Utilizadores dos Bancos de Dados NoSQL

- Twitter
- Facebook
- Digg
- Amazon
- LinkedIN
- Google
- Yahoo
- The New York Times

Dificuldades com NoSQL

Há alguns problemas que ainda hoje existem nessas bases de dados. O primeiro destes problemas é o como não há um padrão (e também não faz sentido existir um) a migração de um banco de dados para outro dentro de uma mesma categoria pode se mostrar uma tarefa bem trabalhosa.

Outro problema é a ausência de ferramentas de alto nível como as que encontramos no modelo relacional.

A curva de aprendizado também pode ser um problema. Como cada banco de dados apresenta a sua própria linguagem de consulta e manipulação, o desenvolvedor precisará adicionar cada vez mais e mais linguagens no seu cinto de utilidades.

Relação entre ORM e NoSQL

ORM para NoSQL é um termo pouco equivocado. As pessoas preferem chamá-lo de "ferramenta OM para NoSQL" ou talvez "ODM - ferramenta de mapeamento de armazenamento de dados de objetos". As estruturas ORM já existem há mais de 30 anos e é um padrão do setor de fato. As pessoas são muito claras sobre o que as ferramentas ORM devem fazer. Não há surpresas.

A chave aqui é deixar as pessoas esquecerem de se preocupar com as complexidades inerentes aos NoSQLs. Deixe-os fazer as coisas de uma maneira que já conhecem e se sentem à vontade. Por que não usar uma abordagem que existe para esse domínio do problema há décadas e provou sua utilidade.

Um bom caso de uso que advoga o uso de ferramentas ORM é a migração de aplicativos (criados usando a ferramenta ORM) do banco de dados RDBMS para NoSQL. (ou mesmo de um banco de dados NoSQL para outro). Isso requer (pelo menos em teoria) pouco ou nenhum esforço de programação no domínio comercial.

Aproximação

Existem algumas diretrizes que vale a pena compartilhar, que ajudam a desenvolver ferramentas ORM para NoSQL:

- **Mapear as notações na estrutura do ORM para estruturas no NoSQL com as quais é mais provável que estejam relacionadas. Exemplo é o @Embedded no JPA com Super column no Cassandra;**
- **Se a estrutura fornecer uma maneira de operar diretamente no banco de dados, aproveite-os. Exemplo é o mapeamento de consultas nativas no JPA com CQL no Cassandra;**
- **O NoSQL foi desenvolvido para desempenho. uma sobrecarga perceptível em relação aos drivers comuns será uma grande desvantagem para os usuários. Faça esforços especiais para manter a sobrecarga do ORM no mínimo, mesmo que isso exija noites sem dormir e semanas de brigas na discussão do design.**