UNIVERSIDADE DE BRASÍLIA
Faculdade do Gama

Sistemas de Banco de Dados 2

Tecnologias de Banco de Dados (TI-BD)

Sistemas de Banco de Dados NoSQL

João Pedro Sconetto – 14/0145940

Brasília, DF
2018

Definição da Tecnologia

De acordo com SADALAGE et al. (2013), o termo “*NoSQL*” teve a sua primeira aparição no final dos anos noventa, como o nome de um banco de dados relacional *open-source*, que para o mesmo é uma ironia, o Strozzi NoSQL. Liderador por Carlo Strozzi, esse banco de dados armazenava suas tabelas como arquivos ASCII, cada tupla representada por uma linha com campos separados por tabulações. O nome vem do fato de aquele banco de dados não usar SQL com uma linguagem de busca (*query language*). Ao invés disso, o banco de dados era manipulo por *shell scripts*, scripts de execução em terminais UNIX, que podem ser combinados com *pipelines* do próprio UNIX. Fora essa coincidência de terminologia, o NoSQL de Strozzi não têm nenhuma influência nos bancos de dados NoSQL descritos no livro de SADALAGE et al.

Movendo para a parte mais técnica da tecnologia, segundo o que podemos encontrar na Wikipédia, o ***NoSQL***, originalmente se referindo, como dito, a “*no SQL*” - “não *SQL*” ou “*non relational*” - não relacional, posteriormente foi estendido para *Not Only SQL* – Não Somente SQL, é um termo genérico para uma classe definida de banco de dados que fornecem um mecanismo para armazenamento e recuperação de dados que são modelados de formas diferentes das relações tabulares usadas nos bancos de dados relacionais. Esse tipo de banco de dado é datado do final da década de 1960, mas a priori não receberam o apelido de “*NoSQL*” até após a popularização do termo com a sua popularidade no início do século vinte e um, desencadeado pelas necessidades das empresas de Web 2.0 como Facebook, Google e Amazon. Os bancos de dados NoSQL são cada vez mais usados em *big data* e aplicações web de tempo real. Sistemas *NoSQL*, são chamados, também, de “*Not Only SQL*” para enfatizar que eles podem suportar linguagens de consulta semelhantes ao SQL.

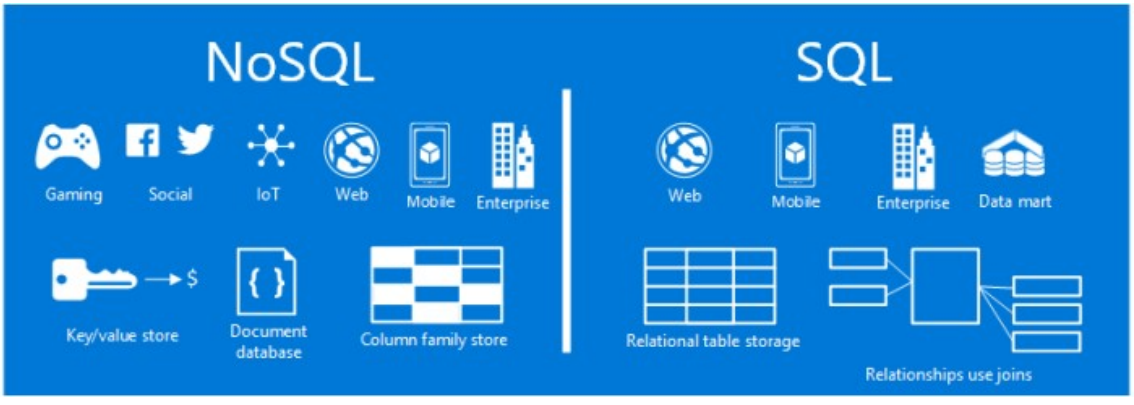

Imagem 1 – Visão geral entre bancos NoSQL e SQL, com seus usos e o tipo de funcionamento de cada. (2018 – Alex Souza).

Alguns motivos que levam o uso dessa abordagem são a simplicidade de projeto, escalonamento “horizontal” mais simples para *clusters* de máquinas (o que é um problema para bancos de dados relacionais), facilidade de

desenvolvimento, alta disponibilidade, com um controle mais refinado sobre a disponibilidade e resiliência. As estruturas de dados usadas pelos bancos de dados *NoSQL* (exemplo, chave-valor, coluna larga, grafo ou documento) são diferentes daquelas usadas por padrão em bancos de dados relacionais, tornando, em parte, operações mais rápidas em *NoSQL*. A adequação particular de um determinado banco de dados *NoSQL* depende do problema que se necessita resolver. Algumas vezes as estruturas de dados usadas por bancos de dados *NoSQL* também são vistas como “mais flexíveis” que tabelas de bancos de dados relacionais.

Tipos e Exemplos de Bancos de Dados *NoSQL*

A tecnologia de banco de dados *NoSQL* é marjoritariamente dividida em quatro tipos básicos, que dizem respeito a como, naquela tecnologia, é trabalhado o armazenamento, a manipulação e as demais tarefas envolvidas dos dados. Para essas quatro categorias há algumas divisões e subcategorias mas as básicas são: Banco de dados Chave-Valor (*Key-Value database*), Banco de dados de Armazenamento de Documentos (*Document-based database*), Banco de dados de Armazenamento em Colunas (*Column-based database*) e Banco de dados de Grafos ou Estrutura de Dados (*Graph-based database*).

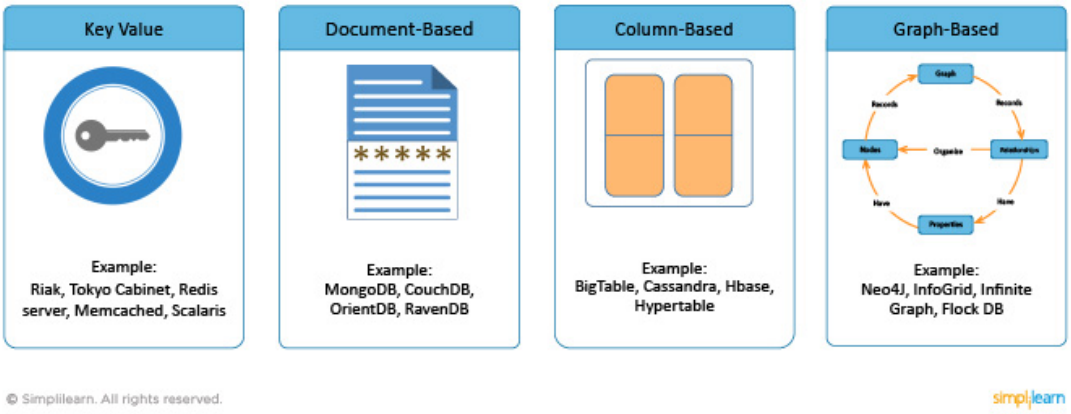

Imagem 2 – Os tipos de bancos de dados NoSQL. (2018 – SimpliLearn).

O banco de dados de valor-chave tem uma grande tabela *hash* de chaves e valores. Riak (se pronucia como REE-awk), Tokyo Cabinet, servidor Redis, Memcached (se pronuncia como mem-cached) e Scalaris são exemplos de um armazenamento de valor-chave. Os armazéns em memória de chave-valor são otimizados para cargas de trabalho de aplicativos de leitura pesada (como redes sociais, jogos, compartilhamento de mídia, e portais de P e R) ou cargas de trabalho com uso intenso da computação (como um mecanismo de recomendação). O armazenamento em cache na memória melhora o desempenho do aplicativo ao armazenar pedaços críticos de dados na memória para acesso de baixa latência.

O banco de dados baseado em documentos armazena documentos

compostos de elementos marcados. Exemplos: MongoDB, CouchDB, OrientDB e RavenDB. Os bancos de dados de documentos são projetados para armazenar dados como documentos, geralmente em formato JSON ou XML. Diferentemente dos bancos de dados relacionais tradicionais, o esquema de cada documento não relacional (NoSQL) pode variar, dando a você mais flexibilidade ao organizar e armazenar dados do aplicativo e ao reduzir o armazenamento exigido para valores opcionais.

No banco de dados de colunas cada bloco de armazenamento contém dados de apenas uma coluna, Exemplos: BigTable, Cassandra, Hbase e Hypertable. Os bancos de dados colunares são otimizados para colunas de leitura e gravação, ao contrário das linhas de dados. O armazenamento orientado a colunas para tabelas do banco de dados é um fator importante no desempenho de consulta analítica, pois ele reduz drasticamente os requisitos gerais de E/S e diminui a quantidade de dados que você precisa carregar do disco.

Um banco de dados de grafos (ou de estruturas de dados, também chamados de Banco de dados gráficos) é um banco de dados de rede que usa nós para representar e armazenar dados. Exemplos são Neo4J, InfoGrid, Infinite Graph e FlockDB. A disponibilidade de opções nos bancos de dados NoSQL tem suas próprias vantagens e desvantagens. Os bancos de dados de grafos tem seu armazenamento definido em vértices e links direcionados chamados de bordas. Grafos podem ser construídos em bancos de dados relacionais (SQL) e não relacionais (NoSQL). Vértices e bordas podem ter propriedades associadas a eles.

De acordo com a lista provida pelo Wikipédia, atualmente há várias abordagens para se classificar bancos de dados *NoSQL*, cada uma com categorias e subcategorias diferentes, sendo que algumas delas se sobrepõem. Uma classificação mais detalhada, baseada em Stephen Yen:

Tipo	**Exemplos notáveis desse tipo**
Cache de Chave-Valor	ArangoDB, Aerospike
Armazenamento Chave-Valor (Eventualmente-Consistente)	Oracle NoSQL Database, Dynamo, Riak, Voldemort
Armazenamento Chave-Valor (Ordenado)	FoundationDB, InfinityDB, LMDB, MemcacheDB
Servidor de Estruturas de Dados	Redis
Armazenamento de Tuplas	Apache River, GigaSpaces
Banco de dados de Objeto	Objectivity/DB, Perst, ZopeDB
Armazenamento de Documentos	ArangoDB, BaseX, Clusterpoint, Couchbase, CouchDB, DocumentDB, IBM Domino, MarkLogic, MongoDB, Qizx, RethinkDB
Armazenamento de coluna ampla	Amazon DynamoDB, Bigtable, Cassandra, Druid, HBase, Hypertable

Tabela 1 – Relação das categorias (e subcategorias) dos tipos de bancos de dados *NoSQL*, com alguns exemplos de tecnologias e ferramentas disponíveis.

Banco de dados correlacionais são independentes de modelo e, em vez de armazenamento baseado em linha ou coluna, usam armazenamento baseado em valor.

Imagem 3 – Logos de algumas ferramentas/sistemas de banco de dados *NoSQL* mais populares. (2018 – Símon Jakowicz)

Objetivos dos Bancos de Dados NoSQL

Conforme discutido no início deste trabalho, a tecnologia de da banco de dados *NoSQL* surgiu de uma necessidade de grandes empresas da Web, tentando solucionar alguns problemas que afetavam suas aplicações, mas centrado em performance e disponibilidade devido a grande demanda e com as evoluções da Web 2.0.

Uma pequena história contada pelo site *SimpliLearn* (2018) diz que, com a explosão das redes sociais, o conteúdo voltado para o usuário cresceu rapidamente e aumentou o volume e o tipo de dados produzidos, gerenciados, analisados e arquivados. Além disso, novas fontes de dados, como sensores, sistemas de posicionamento global ou GPS, rastreadores automáticos e outros sistemas de monitoramento geram grandes volumos de dados regularmente. Esses grandes volumes de conjuntos de dados, também chamados de *big data*, introduziram novos desafios e oportunidades para armazena, gerenciamento, análise e arquivamento de dados. Em adição, os dados estão se tornando cada vez mais semi-estruturados e esparsos. Isso significa que os bancos de dados relacionais (SGBDR) que requerem definição de um esquema inicial e referências relacionais são examinados. Para resolver os problemas relacionados a dados de grande volume e semi-estruturados, surgiu uma classe de novos produtos de banco de dados. Essas novas classes de produtos de banco de dados consistem em armazenamento de dados baseados em colunas, bancos de dados de chave/valor e bancos de dados de documentos. Juntos, eles são chamados de *NoSQL*. Os bancos de dados *NoSQL* consistem em diversos produtos, cada um com conjuntos exclusivos de recursos e proposições de valor.

Por fim, é certo afirmar que em sua concepção e evolução os bancos de dados *NoSQL* tentam solucionar o problema atual, conforme dito pelo *SimpliLearn*, dessa grande explosão da geração, consumo, gerenciamento e análise de dados, onde foi necessário uma tecnologia que fosse ágil e performática para fornecer os dados necessários a essa crescente demanda.

Vantagens do NoSQL

Durante a leitura da literatura acerca de bancos de dados e sistemas *NoSQL*, fica claro que durante as fases de planejamento para sistemas que irão usar o *NoSQL* como tecnologia de persistência, os engenheiros devem analisar e estudar qual das ferramentas vai melhor atender o projeto, pois cada um trabalha com um tipo e tem seus pontos fortes e fracos, com funcionalidades que podem ser muito útil no desenvolvimento dos mesmos. Existem alguns pontos que são comuns aos bancos *NoSQL*, e segundo Souza (2018), que podemos citar como vantagens são:

- **Flexibilidade** – Estruturas de dados intuitivas e flexíveis são funcionalidades que mais atraem desenvolvedores que trabalham em times de desenvolvimento ágil. A maioria dos bancos de dados *NoSQL* tem essas qualidades. A grande flexibilidade dos bancos de dados *NoSQL* é muito popular por suportar as práticas de desenvolvimento ágil, pois elimina a complexidade de mudanças dos bancos de dados gerando um bom suporte para adaptações rápidas;

- **Escalabilidade** – A maioria dos bancos de dados *NoSQL* são construídos para escalar horizontalmente, distribuindo os dados por *clusters* melhor que os SGBDs relacionais, que sofrem muito em performance quando executa consultas com "*joins*" em ambientes clusterizados. Como o *NoSQL* evita "*joins*" naturalmente, a performance das consultas permanece alta. A escalabilidade *NoSQL* aplica-se tanto ao crescimento dos dados quanto ao número de usuários agindo simultaneamente sobre os dados;

- **Disponibilidade** – A indisponibilidade de um banco de dados pode causar sérios prejuízos, incluindo perda de clientes. A maioria dos bancos de dados *NoSQL* oferecem eficientes arquiteturas de replicação de dados que proporciona aos *NoSQLs* maior disponibilidade. Ou seja, se um ou mais servidores (ou nós) cai um outro nó do *cluster* está apto a continuar o trabalho automaticamente sem perda de dados;

- **Raízes *open source*** – Muitos bancos de dados NoSQL tem raízes na comunidade *open source*. Talvez isso tenha sido fundamental para o rápido crescimento do seu uso e popularidade. Nota-se que as companhias que oferecem versões comerciais de bancos *NoSQL* com uma forte estrutura de suporte e serviços, estão ao mesmo tempo participando direta ou indiretamente de comunidades de bancos de dados *NoSQL open source*. Exemplos são Apache: Cassandra, IBM: CouchDB, MongoDB: MongoDB open source, entre outros;

- **Baixo custo operacional** – Devido ao peso do *open source* no *NoSQL*, o custo para iniciar a utilização desses bancos de dados também se torna muito baixo ou zero. É comum ouvir dizerem que a transição relacional para *NoSQL* diminuiu muito os custos enquanto obteve um desempenho melhor ou igual ao anterior. Grandes bancos de dados relacionais requerem computadores ou mainframes caros. Com o *NoSQL*, esse custo também diminui, pois este foi desenvolvido para trabalhar em ambientes distribuídos;

- **Funcionalidades especiais** – Também com o peso do *open source* sobre o *NoSQL*, muitos distribuidores de bancos de dados *NoSQL* oferecem algumas funcionalidades especiais para incentivar e atrair mais usuários. Índices específicos, capacidade de consulta de dados geoespaciais, replicação automática de dados, funcionalidades para sincronização, *APIs RESTful* são exemplos de funcionalidades especiais oferecidas pelos diferentes distribuidores *NoSQL*.

Desvantagens do NoSQL

Entre as diversas referências levantadas, há uma série de pontos onde o *NoSQL* têm revelado a suas fraquezas, a primeira a ser apontada é o seu problema com a consistência dos dados. Muitos armazenamentos *NoSQL* comprometem a consistência (no sentido do teorema CAP) em favor da disponibilidade, tolerância a partição e velocidade. O teorema de CAP, também chamado de Teorema de Brewer, afirma que é impossível que o armazenamento de dados distribuído forneça simultaneamente mais de duas das três garantias seguintes:

- Consistência: Cada leitura recebe a escrita mais recente ou erro;
- Disponibilidade (*Availability*): Cada pedido recebe uma resposta (sem erro) – sem garantia de que contém a escrita mais recente;
- Partição tolerante a falhas : O sistema continua a funcionar apesar de um número arbitrário de mensagens serem descartadas (ou atrasadas) pela rede de nós.

Em outras palavras, o teor do CAP afirma que, na presença de uma partição da rede, é preciso escolher entre consistência e disponibilidade. Observe que a consistência conforme definido no teor de CAP é bastante diferente da consistência garantida em transações de bases de dados ACID.

Barreiras para a grande adoção de armazenamentos *NoSQL* incluem o uso de linguagens de consulta de baixo nível (em vez de SQL, por exemplo, a falta de capacidade de realizar junções *ad-hoc* entre tabelas), falta de

interfaces padronizadas e grandes investimentos anteriores em bancos de dados relacionais existentes. A maioria dos armazenamentos *NoSQL* carece de transações ACID verdadeiras, embora alguns bancos de dados, como MarkLogic, Aerospike, FairCom c-treeACE, Google Spanner (embora tecnicamente seja um banco de dados NewSQL), Symas LMDB e OrientDB tornaram-as centrais em seus projetos. Alguns outros pontos são:

- O seu padrão de ausência esquema (*Schema-less*) é perfeito em algumas situações, no entanto, se um banco de dados é compartilhado entre aplicativos, não há como impedir que um aplicativo armazene dados inconsistentes. Existem soluções para isso:
    1. Encapsular todas as interações do banco de dados em um único aplicativo e integrá-lo aos seus outros aplicativos usando de *web services*.
    2. Usando diferentes famílias de colunas para diferentes aplicações em um banco de dados de família de colunas (*column-family database*), ou seções diferentes para cada aplicação em um banco de dados de documentos.
- Também é difícil determinar o que pode ser armazenado nas agregações sem verificar minuciosamente a aplicação. É sempre uma boa ideia mander um esquema (*schema*) implícito na documentação da aplicação.

Por fim, algo apontado por bancos *NoSQL*, que podem ser vistos como uma desvantagem se comparado a bancos de dados relacionais, a depender do ponto de vista, é a sua forma de tratar a segurança do próprio banco. Enquanto banco de dados relacionais, geralmente, temos usuários e delega-se permissões para os mesmos utilizarem o banco e os seu conteúdo, em bancos *NoSQL* geralmente a segurança do DB não é controlada por ela mesma, ao invez disso a segurança é uma responsabilidade do cliente acessando o banco de dados.

Uso em grandes projetos e empresas

É possível encontrar na comunidade diversas empresas que fazem uso de bancos de dados *NoSQL*, várias tem resultados bem positivos com o uso dessa tecnologia. Uma vez que a demanda dessas organizações/projetos exige softwares que sejam capazes de manusear grandes quantidades de dados, acessos, transações e pedidos de informação as tecnologias do *NoSQL* acabam por ganhar espaço. Segundo Allana (2018), no site de plubicação do Mundo DevOps, a lição que as grandes empresas que utilizam *NoSQL* deixam para as pessoas é a chance de saber que trata-se de algo confiável. Essas organizações precisam atender uma necessidade global e isso é possível apenas com boas práticas. Isso deixa para todos um exemplo de como as estratégias podem ser aplicadas. Lembre-se que se funciona nas grandes empresas que utilizam *NoSQL* também pode dar certo na sua empresa/projeto/time. Veja a seguir as organizações que optaram pelo uso de sistemas *NoSQL*:

Facebook

Atualmente o Facebook tem mais de 2,3 bilhões de usuários em todo mundo, ou seja, é praticamente um monopólio. É por essa razão que essa é uma das grandes empresas que utilizam *NoSQL* e conseguir ter ótimos resultados.
A necessidade para utilizar os bancos de dados *NoSQL* vem do desenvolvimento de vários tipos de funcionalidades. O objetivo claro era permitir que a fosse feita a leitura e gravação de forma assíncrona no próprio disco rígido.
Foi preciso ligar com os seguidores que estavam desatualizados, isso devia-se a novas instalação no servidor. Sem contar que em alguns outros casos, como por exemplo, aquelas situações em que o disco poderia até estar mais lento.

Google

Em 2015, o Google tinha mais de 2,2 bilhões de usuários, porém essa contagem parou, portanto, é complicado de dimensionar. Sem dúvidas, trata-se do maior buscador e talvez seja a empresa de tecnologia que mais cresce atualmente.
O Google está incluído na lista das grandes empresas que utilizam *NoSQL*, porque é uma empresa que inova a cada dia. Isso faz com que os seus bancos de dados precisem ser não relacionais, para garantir uma performance melhor.
Especialmente para conseguir fazer a manipulação dos dados que podem até apresentar algumas mudanças no seu formato. Outra vantagem é a escalabilidade, ou seja, a chance de ter uma plenitude de rendimento.

Twitter

O Twitter fecha essa lista das grandes empresas que utilizam *NoSQL* (há muitas outras mas esses são alguns exemplos) e hoje conta com mais de 336 milhões de usuários registrados. É uma das maiores redes sociais do mundo e

faz muito sucesso no Brasil, devido a sua interface bem intuitiva.
O número de usuários cresce a cada dia, ou seja, o armazenamento de dados acompanha essa curva de crescimento. O problema é que o serviço também deve acompanhar esse processo, permitindo assim um equilíbrio.
Os bancos de dados relacionais não terão essa capacidade e o Twitter inteligentemente optou pelo *NoSQL*. Esse fato permite eliminar o desequilíbrio, ou seja, é possível atender a necessidade com maior agilidade em todos os campos.

Referência Bibliográfica

[1] Wikipédia, a enciclopédia livre. **NoSQL**. 2018. Disponível em: <https://pt.wikipedia.org/wiki/NoSQL>. Acesso em: 14 set. 2018.

[2] IANNI, Vinicius. **NoSQL Tutorial: Introdução aos bancos de dados NoSQL**. 2012. Disponível em: <https://www.devmedia.com.br/introducao-aos-bancos-de-dados-nosql/26044>. Acesso em: 14 set. 2018.

[3] FERREIRA, Lauren. **As diferenças entre SQL e NoSQL: MySQL x MongoDB**. 2017. Disponível em: <https://medium.com/devtranslate/diferencas-entre-sql-e-nosql-51311f9069bd>. Acesso em: 15 set. 2018.

[4] SADALAGE, Pramod J.; FOWLER, Martin. **NoSQL Distilled – A Brief Guide to the Emerging World of Polyglot Persistence**. 1. ed. ISBN 978-0-32182662-6 (pbk. : alk. paper) -- ISBN 0-321-82662-0 (pbk. : alk. paper), 2013. 220 p.

[5] RODRIGUES VIERA, Marcos et al. Banco de Dados NoSQL. In: Simpósio Brasileiro de Banco de Dados, 2012, São Paulo. **Bancos de Dados NoSQL: Conceitos, Ferramentas, Linguagens e Estudos de Casos no Contexto de Big Data**. Minicursos: [s.n.], 2012. p. 1-30.

[6] SIMPLILEARN. Online Courses. **Introduction to NoSQL databases Tutorial**. 2018. Disponível em: <https://www.simplilearn.com/introduction-to-nosql-databases-tutorial-video>. Acesso em: 16 set. 2018.

[7] SOUZA, Alex. **Não Relacionais (NoSQL)**. 2018. Disponível em: <https://pessoalex.wordpress.com/dados/nao-estruturados/nosql/>. Acesso em: 15 set. 2018.

[8] JAKOWICZ, Simon. **NoSQL Essentials – Part 1 – Overview**. 2014. Disponível em: <https://www.jakowicz.com/nosql-essentials-part-1-overview/>. Acesso em: 15 set. 2018.

[9] Wikipédia, a enciclopédia livre. **Teorema de CAP**. 2018. Disponível em: <https://pt.wikipedia.org/wiki/Teorema_CAP>. Acesso em: 16 set. 2018.

[10] ALLANA. **Grandes empresas que utilizam NoSQL**. 2018. Disponível em: <https://www.mundodevops.com/grandes-empresas-que-utilizam-nosql/>. Acesso em: 16 ago. 2018.