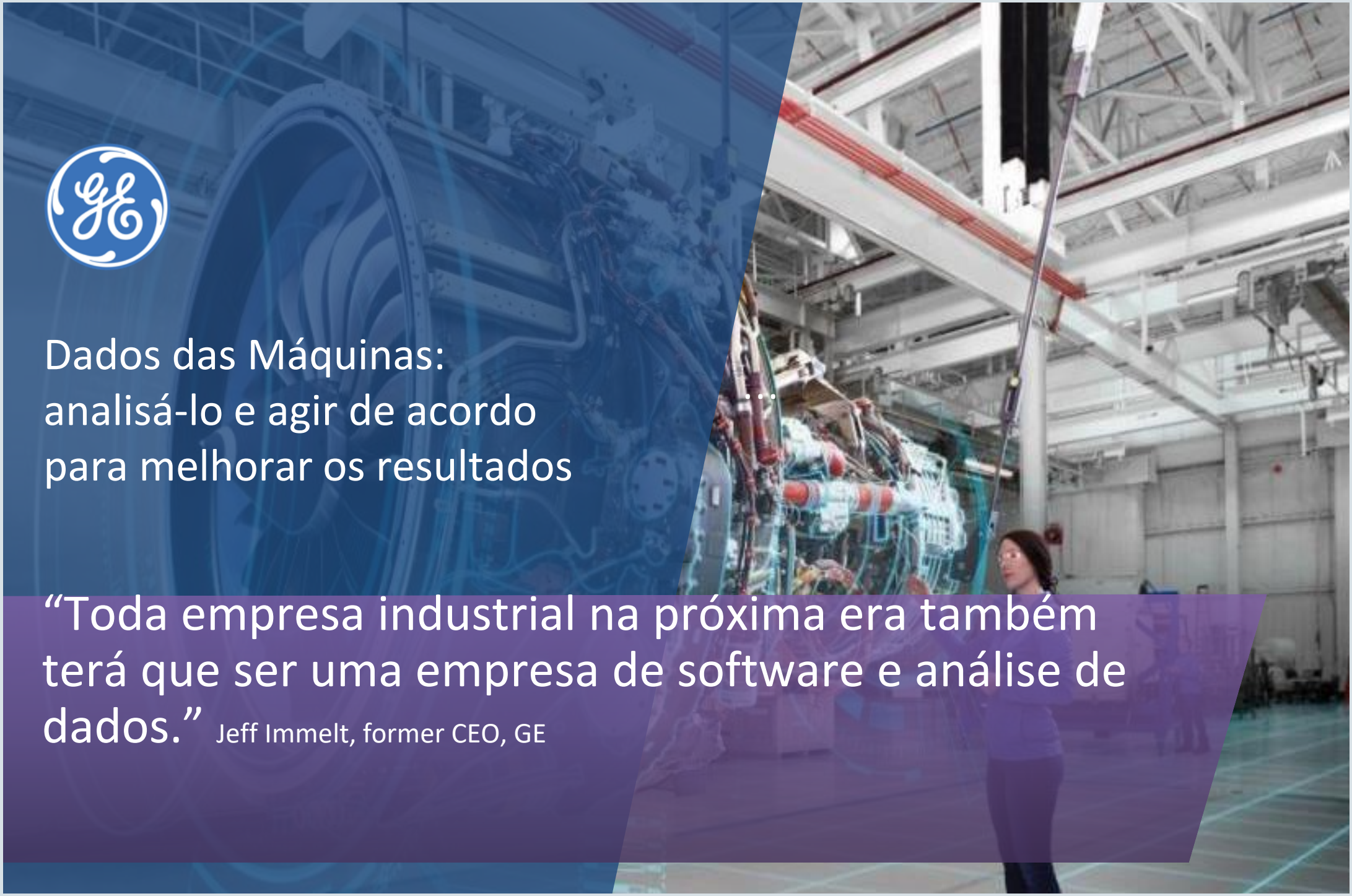
Bancos de Dados NoSQL

PROF.: EDSON GUIMARÃES

Agenda

- Historia
- O que é NoSQL
- Teorema CAP
- Do que abrimos mão.
- Tipos de NoSQL
- Modelos de Dados
- Quem utiliza e o que.



Dados das Máquinas:
analisá-lo e agir de acordo
para melhorar os resultados

“Toda empresa industrial na próxima era também terá que ser uma empresa de software e análise de dados.” Jeff Immelt, former CEO, GE

Do produto à plataforma: John Deere

- Consolidação agrícola
- Maior concorrência dos equipamentos
- Lealdade reduzida do parceiro

Produção de alimentos deverá crescer **60%** para alimentar uma população de **9B** pessoas em 2050

Lançou o portal MyJohnDeere, uma plataforma aberta e unificada, para ajudar fazendeiros a melhorar a eficiência operacional

$26B 2016

- Aumentar a venda de equipamentos em **18% YoY**
- Aumentar a qtde de parceiros
- Barreiras competitivas criadas

4

Data Insights e Gestão de Restaurantes

Visão 360º do cliente

Gerenciar os custos da mão de obra

Otimização do Menu & Localização

Detecção de Roubo & Fraude

Estruturado (Inside the Business)

- POS – O que vendemos?
- Supplier – Dispon. dos Produtos
- Contabilidade – Custos, Venda
- Mão Obra – Remuneração, Salarios, Gorjetas

Não Estruturado (Outside the Business)

- Midia Social – Likes, Tweets
- Perfil do Cliente
- Programa de Fidelidade
- Apps – Check-ins, Revisões
- Outros– Clima, Tráfego, etc

Pq Vc precisa de ambos

Dados estruturados Te conta o "**oque**", enquanto dados não estruturados Te conta o "**porque**". Usindo ambos Vc terá uma visão mais holística do cliente e do negócio

5

Gerenciando a Explosão de Dados

Diferentes Soluções para Diferentes necessidades

Fragmentação, Compatibilidade, Performance, Securança....

* 2017

Terminologia básica

- Um **banco de dados** (BD) consiste em um repositório contendo dados relacionados; já um sistema **gerenciador de bancos de dados** (SGBD) é um programa, pago, ou não, instalado e configurado;

- Um SGBD pode ser **proprietário** ou disponibilizado na modalidade **Open Source**;

História

- Bancos de Dados Relacionais – Esteio da Tecnologia
- Aplicações baseadas na Web causaram picos:
  - Especialmente aqueles voltados para vendas de varejo(diretas ao cliente);
- Os desenvolvedores começaram a confrontar os SGBDR com outros mecanismos de armazenamento,
  - Até mesmo incorporam alguns modelos diferentes para problemas específicos.

História

Algumas organizações atingiram a casa dos petabytes em volume de dados armazenados:

- O Facebook é um desses casos;
- Em 2011 este volume ultrapassou 30 petabytes aproximadamente 30 mil terabytes; https://zephoria.com/top-15-valuable-facebook-statistics/
- Menos de um ano antes o volume era da ordem de 20 petabytes;

No caso destes tipos de organizações, a utilização dos SGBD's Relacionais (SGBDR) tem se mostrado muito problemática e não eficiente;

O problema principal é combinar o tipo de modelo com a demanda por escalabilidade que é mais frequente.

História

- No caso do Facebook, se fosse utilizado um SGBDR e houvesse um crescimento no número de usuários, consequentemente haveria uma perda de performance;
  - Qual a solução?
  - Upgrade no servidor ou aumentar o número de servidores;
- Aumentando o poder do servidor:
  - Aumentar o poder do processador, memória e armazenamento;
  - Escalabilidade Vertical
- Aumentar o numero de servidores:
  - Escalabilidade horizontal.

Contextualização

- Ainda no campo dos exemplos, imagine um sistema que não sabe ao certo os campos de uma determinada entidade;
- Agora imagine esse sistema do dia para a noite sendo acessado por milhares e milhares de usuários;
- No dia seguinte, com a caixa de entrada do e-mail cheia com sugestões de alterações
  - As alterações são inovadoras e interessantes
  - Mas implicam em refatorar completamente o BD
- Bancos de Dados Relacionais se mostram burocráticos para estes tipos de problemas.

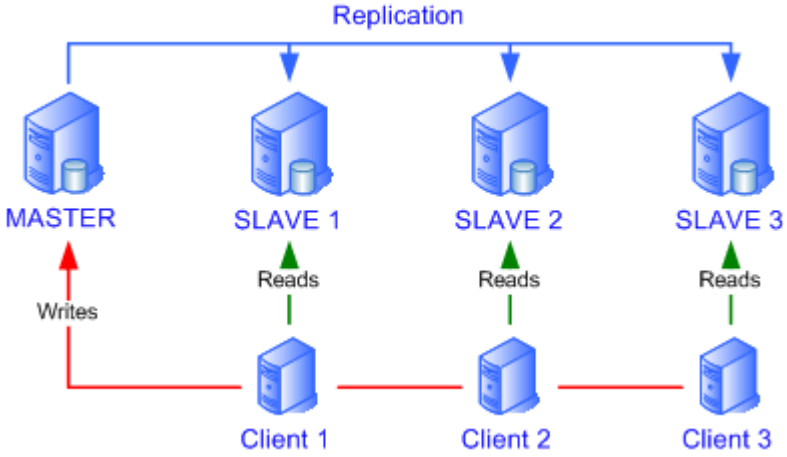
Contextualização

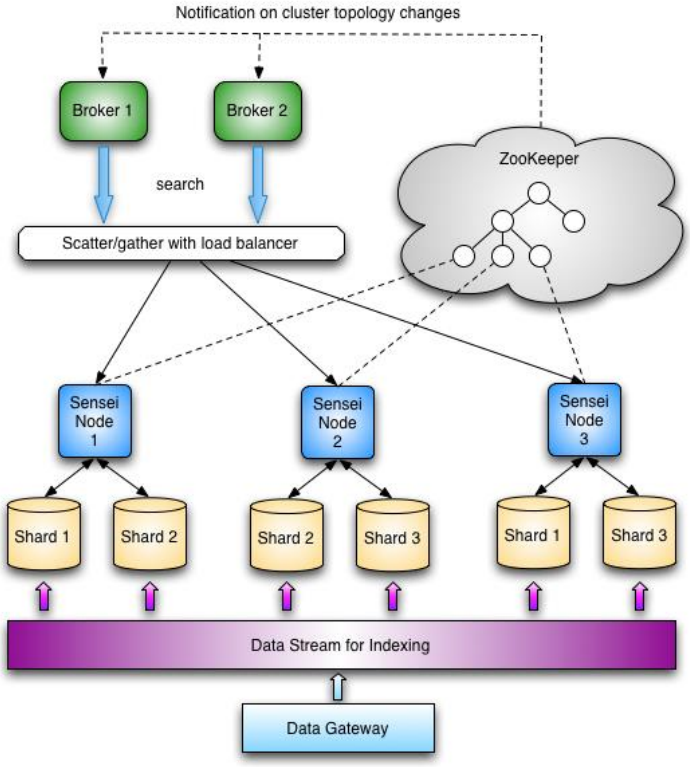
- Como esse intenso volume de dados vem aumentando, e dado a sua natureza estrutural;
- Os desenvolvedores perceberam a dificuldade de organizar dados no Modelo Relacional;
- É ai que entra os modelos NoSQL.

Aumentando de Tamanho

- Problemas de escalabilidade e quando as coleções são muito grandes;
- SGBDR não foram projetados para serem distribuídos;
- Começou-se a observar soluções de bancos de dados multi-nó;
- Conhecidas como ‘scaling out’ ou ‘horizontal scaling’;
- Diferentes abordagens como:
  - Mestre-escravo;
  - Sharding.

Escalabilidade(Master Slave e Sharding)





Escalabilidade em SGBDR – Mestre/Escravo

- Mestre-Escravo:
  - Todas as escritas são feitas no mestre. Todas as leituras são executadas nos bancos de dados replicados escravos;
- Leituras críticas podem estar incorretas assim como as escritas podem ainda não ter sido propagadas;
- Grandes conjuntos de dados podem apresentar problemas com a necessidade de aumentar a quantidade de escravos.

Escalabilidade em SGBDR – Sharding

- Partition ou sharding
  - Melhor balanceamento entre leituras e escritas;
  - Não transparente, a aplicação precisa estar consciente das partições;
  - Não da suporte a relacionamentos e junções entre as partições
  - Perda de integridade referencial entre as partições.

Outras formas de escalar SGBDR

- Replicação Multi-Master
- Apenas INSERT, sem UPDATES/DELETES
- Sem JOINs, reduzindo o tempo de resposta das consultas;
  - Isto envolve dados NÃO normalizados;
- Bancos de Dados em memória principal;

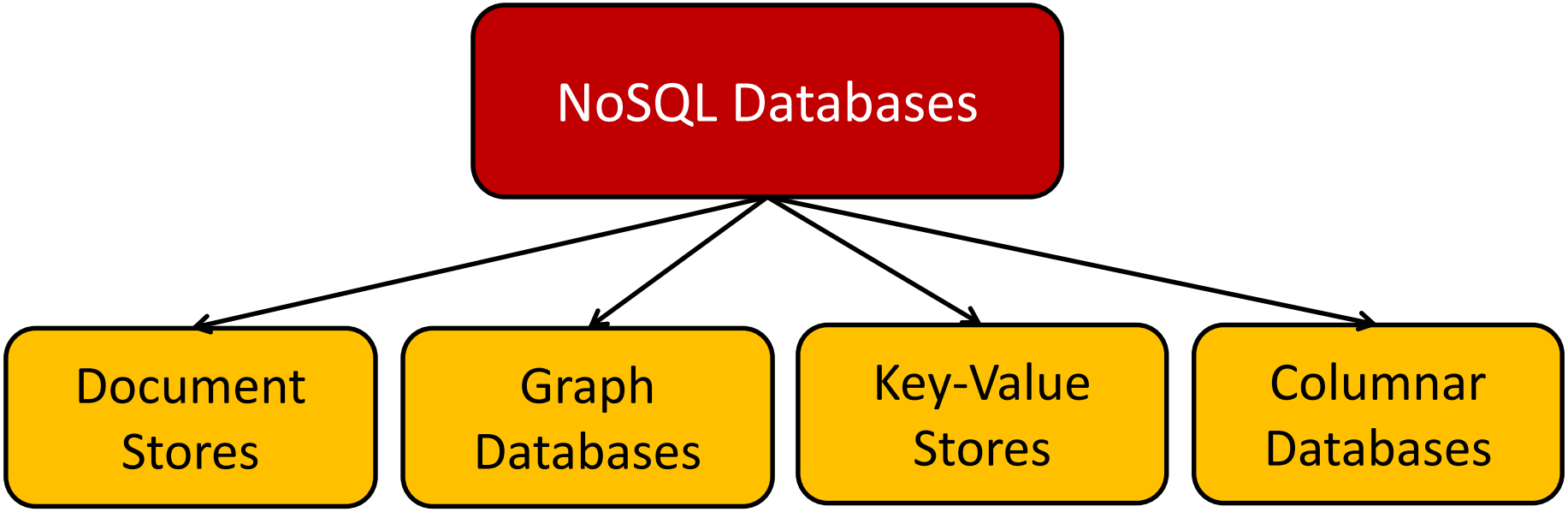
O que é NoSQL?

- Usualmente se diz Not Only SQL;
- Classe de sistemas de armazenamento de dados não- relacionais
- Também pode-se dizer No-Rel;
- Não requer um esquema fixo de tabelas e nem utiliza o conceito de junções;
- Todos os exemplos de NoSQL relaxam uma ou mais das propriedades ACID (vamos falar do teorema CAP em seguida).

Algumas implementações

- BigTable: Desde 2004 o Google investe num produto para suprir as necessidades da empresa:
- Baseado nos princípios de alto desempenho, escalabilidade e disponibilidade;
- Cassandra: Desenvolvido pelo Facebook e doado ao projeto Apache:
  - Foi concebido para lidar com grandes volumes de dados;
  - Em 2010 o Twitter adotou o Cassandra em detrimento do MySQL;
- CouchDB: Projetado pelo projeto Apache;
  - Orientado a documentos;
  - Projetado para suportar computação distribuída em larga escala

Algumas implementações (Tipos)



Por que NoSQL?

- Para o armazenamento de dados, um SGBDR não pode ser tudo e nem o fim de tudo;
- Assim como se tem diferentes linguagens de programação, precisamos ter outras ferramentas de armazenamento na caixa de ferramentas;
- Soluções NoSQL são mais aceitáveis para um cliente hoje do que a um ano atrás
  - Pense sobre a proposta de uma solução Ruby/Rails ou Groovy/Grails agora comparado a alguns anos atrás.

Como chegamos aqui?

- Explosão dos sites de mídias Sociais (Facebook, Twitter) com grandes necessidades de dados;
  - Chegada de soluções baseadas em nuvens como o Amazon S3 (simple storage solution);
  - Assim como mudança para linguagens dinamicamente tipadas (Ruby/Groovy), demandou mudança para dados dinamicamente tipados com mudanças frequentes de esquema;
- Comunidade Open-Source.

Dynamo e BigTable

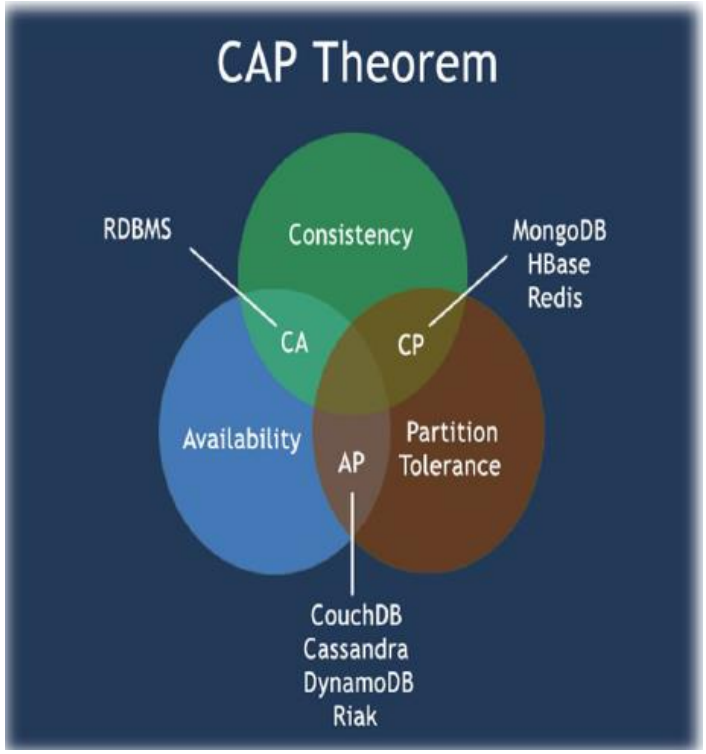
- Três artigos principais foram as sementes do movimento NoSQL;
  - BigTable(Google)
  - Dynamo (Amazon)
  - Protocolo Gossip (descoberta e detecção de erros)
  - Armazenamento de dados key-value distribuído;
  - Consistência eventual;
  - Teorema **CAP**(em seguida…)

A Tempestade Perfeita

- Grandes conjuntos de dados, aceitação de alternativas, e dados dinamicamente tipados vieram em conjunto em uma “tempestade perfeita”;
- Não é uma reação / rebelião contra os SGBDR
- O SQL é uma linguagem rica que não pode ser comparada com a lista atual de ofertas NoSQL.

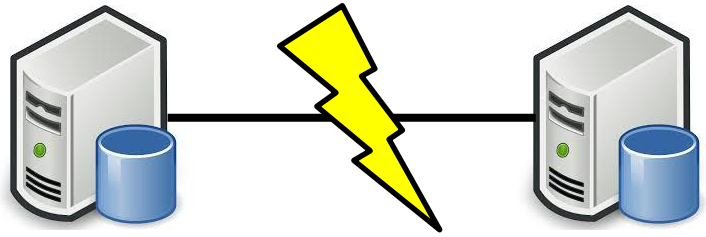
Teorema CAP

- Três propriedades de um sistema: **Consistency, Availability and Partitions**
- Você pode ter no máximo duas dessas três propriedades para qualquer sistema de dados compartilhado;
- Para crescer, você tem que dividir. Isso deixa consistência e disponibilidade para escolher
  - Na maioria dos casos, você deve escolher disponibilidade sobre consistência.



Disponibilidade

- Tradicionalmente pensa em como deixar o servidor/processo 99,999% do tempo disponível
- Entretanto, para um grande sistema de nós, é muito provável que em algum instante haja a queda de um nó ou uma interrupção de rede entre os nós;
  - Pretende um Sistema que seja resistente face a pertubações na rede



Modelo de Consistência

Um modelo de consistência determina regras para a visibilidade e ordem aparente de atualizações.

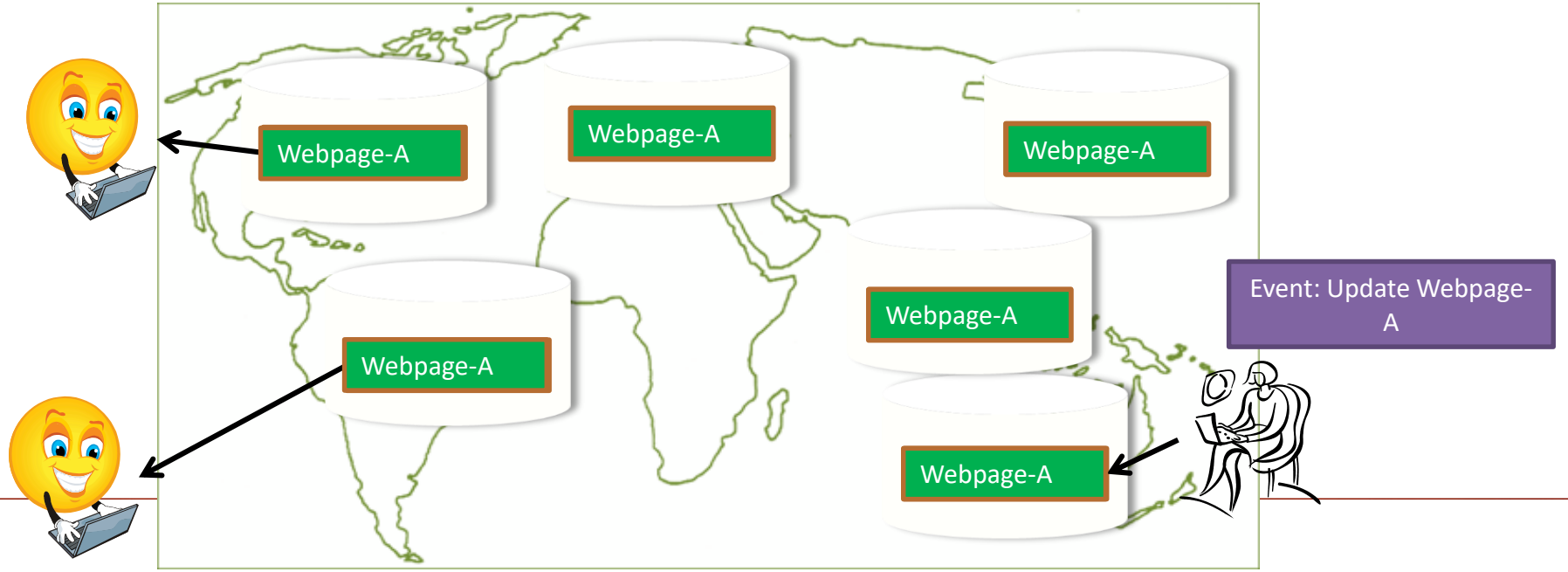
- Por Exemplo:
  - Linha X é replicada os nós M e N;
  - Cliente A escreve linha X no nó N;
  - Depois de algum período de tempo T decorrido.
  - Cliente B lê linha X do nó M
  - Será que o cliente B vê a gravação do cliente A?
  - A consistência é um continuo com compensações
  - Para NoSQL, a resposta seria: talvez
  - **CAP** Teorema afirma: Consistência Estrita não pode ser alcançado, ao mesmo tempo que disponibilidade e tolerância a partição.

Consistência Eventual

- Quando nenhuma atualização ocorre por um longo período de tempo, eventualmente, todas as atualizações serão propagadas pelo sistema e todos os nós serão consistentes;
- Para uma dada atualização em um dado nó, eventualmente, ou atualização alcança o nó ou o nó é removido do serviço;
- Conhecido como BASE (**B**asically **A**vailable, **S**oft state, **E**ventual consistency), em oposição ao ACID

Teorema CAP (Consistência Eventual)

- Banco é dito “Eventualmente Consistente se:
  - Todas as réplicas gradualmente se tornam consistentes na ausência de atualizações



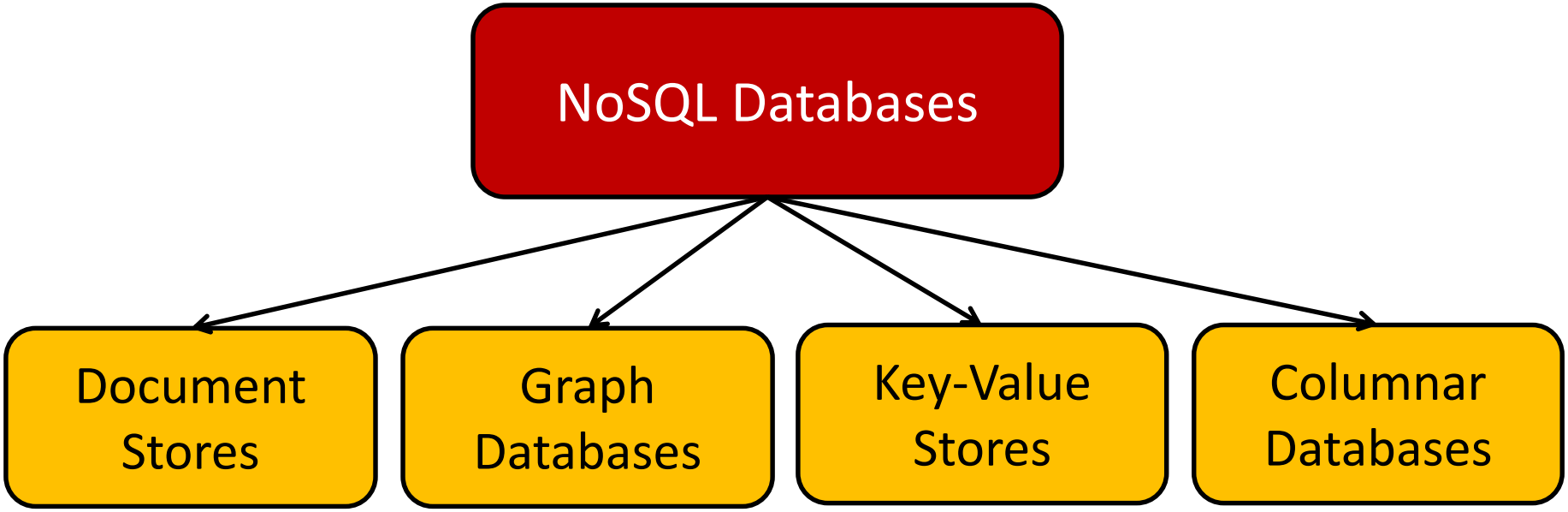
Propriedades ACID

- **ACID** (acrônimo de Atomicidade, Consistência, Isolamento e Durabilidade - do inglês: Atomicity, Consistency, Isolation, Durability)
  - é um conceito utilizado em ciência da computação para caracterizar uma transação em um Banco de Dados, entre outras coisas.
- **Atomicidade**: Trata o trabalho como parte indivisível (atômico).
  - A transação deve ter todas as suas operações executadas em caso de sucesso ou nenhum resultado de alguma operação refletida sobre a base de dados em caso de falha.
  - Ou seja, após o término de uma transação (commit ou abort), a base de dados não deve refletir resultados parciais da transação.

Propriedades ACID

- **Consistência**: A execução de uma transição deve levar o banco de dados de um estado consistente a um outro estado consistente
  - ou seja, uma transação deve respeitar as regras de integridade dos dados (como unicidade de chaves, restrições de integridade lógica, etc...).
- **Isolamento**: Em sistemas multiusuários, várias transações podem estar acessando o mesmo registro (ou parte do registro) no banco de dados.
  - Por exemplo, se um usuário tentasse alterar um registro e um outro estivesse tentando ler este mesmo registro.
- **Durabilidade**: Os efeitos de uma transação em caso de sucesso (commit) devem persistir no banco de dados mesmo em presença de falhas.
  - Garante que os dados estarão disponíveis em definitivo.

Quais os Tipos de NoSQL?

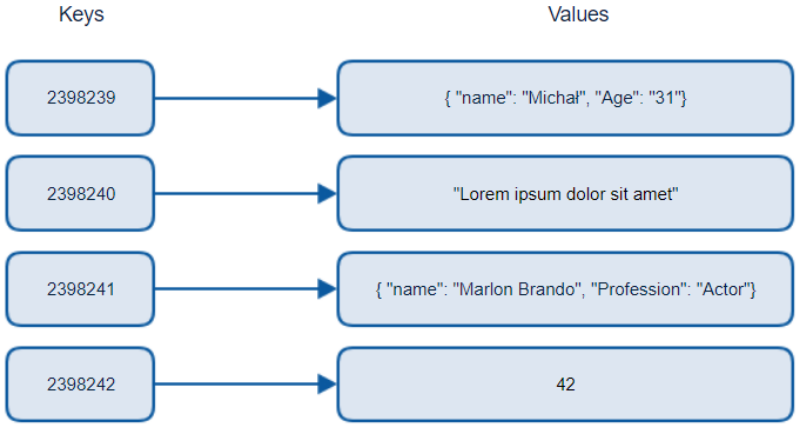


Quais os tipos de NoSQL?

- Soluções NoSQL se dividem em duas áreas principais:
  - Key/Value ou 'the big hash table'.
    - Amazon S3 (Dynamo)
    - Voldemort
    - Scalaris
  - Esquema-less, que vem em vários sabores: Baseado em coluna, Baseado em gráfico ou com base em documentos.
    - **Cassandra** (column-based)
    - **CouchDB** (document-based)
    - **Neo4J** (graph-based)
    - **HBase** (column-based)
    - **MongoDB** (document-based)

Key/Value

- **Pros:**
  - Muito rápido
  - Muito escalável
  - Modelo simples
  - Habilidade de distribuição horizontal;
- **Cons:**
  - Muitas estruturas de dados (objetos) não podem ser modeladas facilmente como pares de chaves e valores.



Schema-Less

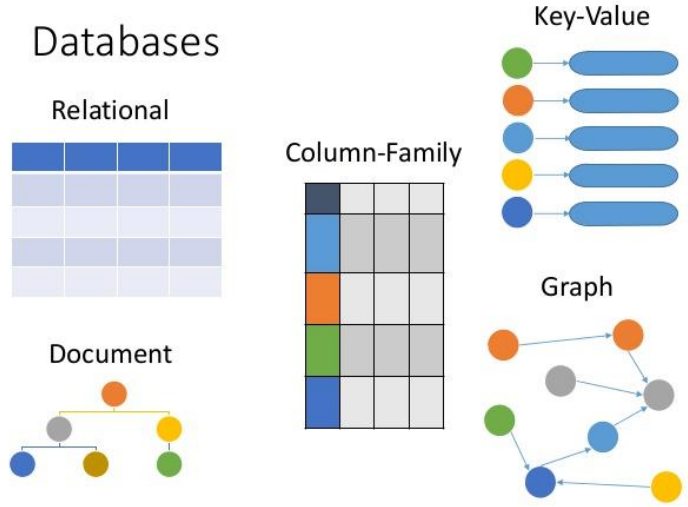
- **Pros:**
  - O modelo de dados Schema-less é mais rico que pares key/value
  - Consistência eventual;
  - Muitos são distribuídos;
  - Também oferecem excelente performance e escalabilidade;
- **Cons:**
  - Tipicamente sem transações ACID ou junções.

No Schema

People
name: John, **favorite_color**: blue
name: Robert, **favorite_color**: red, **favorite_smell**: oranges
name: 14, **owns_pants**: probably

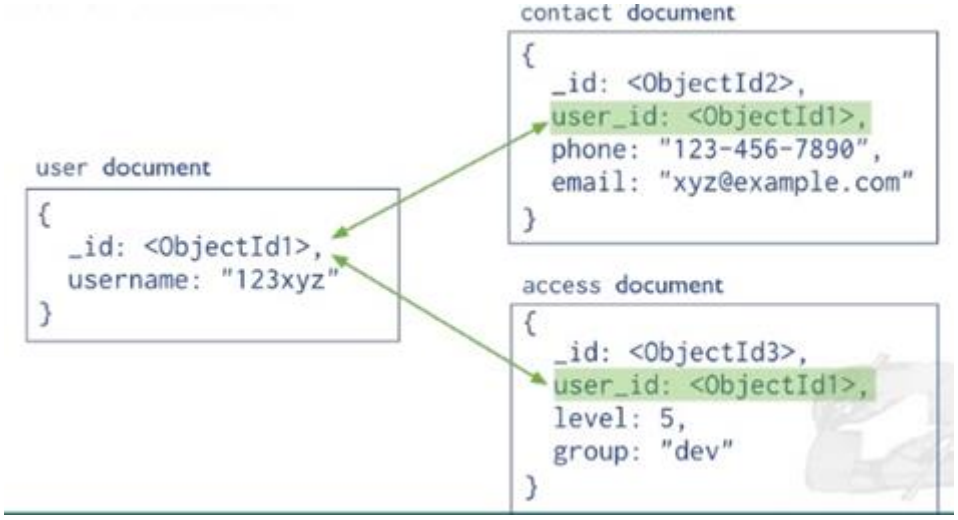
Modelos NoSQL

- Além dos modelos chave/valor temos tambem os seguintes modelos:
  - Orientado a Documentos;
  - Orientado a Coluna;
  - Orientado a grafos;



Orientado a Documentos

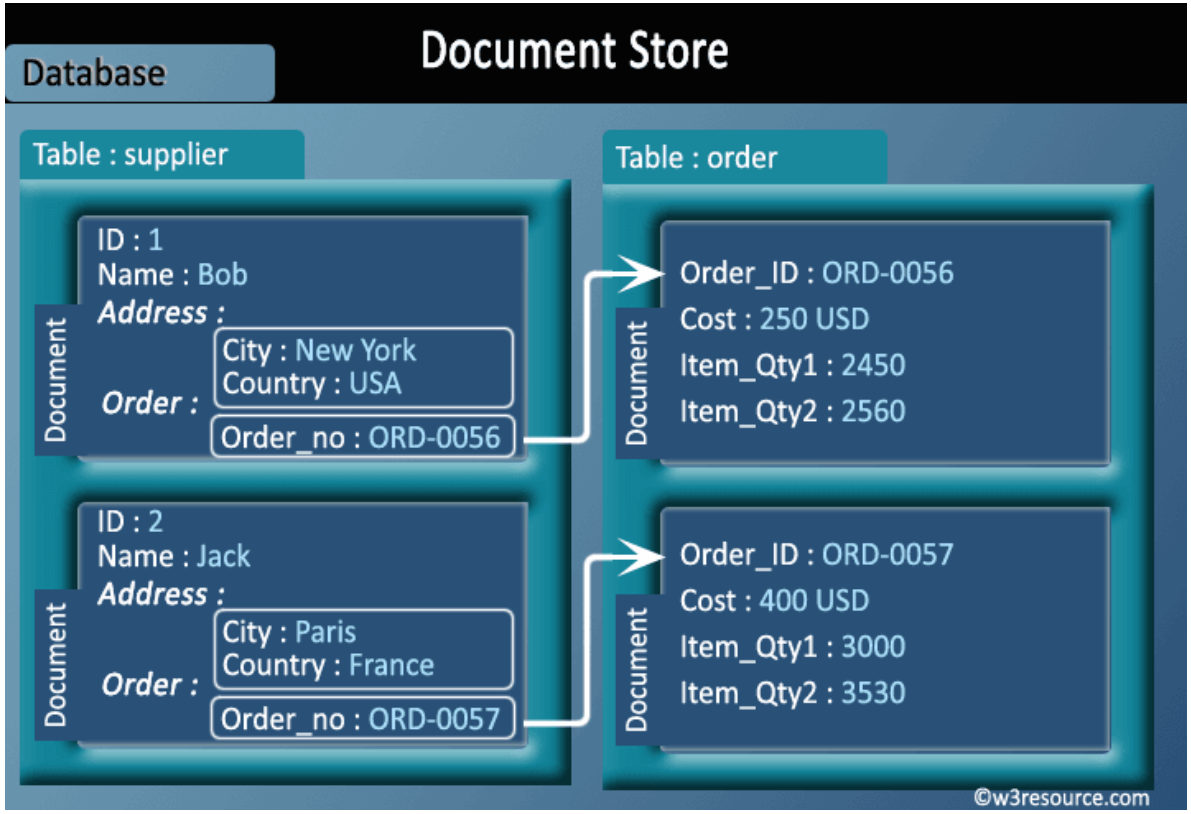
- Como o próprio nome sugere, este modelo armazena coleções e documentos.
- Explicando melhor, um documento, no geral, é um objeto identificador único e um conjunto de campos que podem ser strings, listas ou documentos aninhados.



Orientado a Documentos

- Diferente do banco de dados chave-valor onde se cria uma única tabela hash, neste modelo temos um agrupamento de documentos sendo que em cadaum deste documentos temos um conjunto de campos e o valor deste campo.
- Neste modelo temos ausência de esquema pré-definido (schema free).
- Isto significa que é possível que haja atualizações no documento, com a adição de novos campos, por exemplo, sem afetar adversamente outros documentos.
    - Ex: CouchDB, Mongo DB.

Orientado a Documentos



Orientado a Colunas

- Demonstra maior complexidade que o de chave-valor. Este tipo de banco de dados foi criado para armazenar e processar uma grande quantidade de dados distribuídos em diversas máquinas.
- Aqui existem as chaves, mas neste caso, elas apontam para atributos ou colunas múltiplas.
- Neste caso, os dados são indexados por uma tripla (coluna, linha e timestamp), a coluna e linha são identificadas por chaves e o timestamp permite diferenciar múltiplas versões de um mesmo dado.
- Como o próprio nome sugere, as colunas são organizadas por família da coluna.

Orientado a Colunas

- Vale destacar que as operações de escrita e leitura são atômicas, ou seja, os valores associados a uma linha são considerados em sua execução, independente das colunas que estão sendo lidas/escritas.
- O conceito associado a este modelo é o de família de colunas, com o objetivo de reunir colunas que armazenam o mesmo tipo de informação.
- Como exemplo, a Figura do próximo slide modela o conceito de família, onde o primeiro Nome e sobrenome são colunas pertencentes à família de colunas denominada “nome”.
- Da mesma forma, as colunas endereço, cidade e estado pertencem
- à família local.

Orientado a Colunas

Necessário rebuild da tabela a cada nova coluna

Key	Fname	Lname	State	Zip	Phone	Age	Sex	Golf
1	Bugs	Bunny	NY	11217	(718) 938-3235	34	M	Y
2	Yosemite	Sam	CA	95389	(209) 375-6572	52	M	N
3	Daffy	Duck	NY	10013	(212) 227-1810	35	M	Y
4	Elmer	Fudd	ME	04578	(207) 882-7323	43	M	Y
5	Witch	Hazel	MA	01970	(978) 744-0991	57	F	N

Orientado a coluna: Cria novo arquivo

Key
1
2
3
4
5

Fname
Bugs
Yosemite
Daffy
Elmer
Witch

Lname
Bunny
Sam
Duck
Fudd
Hazel

State
NY
CA
NY
ME
MA

Zip
11217
95389
10013
04578
01970

Phone
(718) 938-3235
(209) 375-6572
(212) 227-1810
(207) 882-7323
(978) 744-0991

Age
34
52
35
43
57

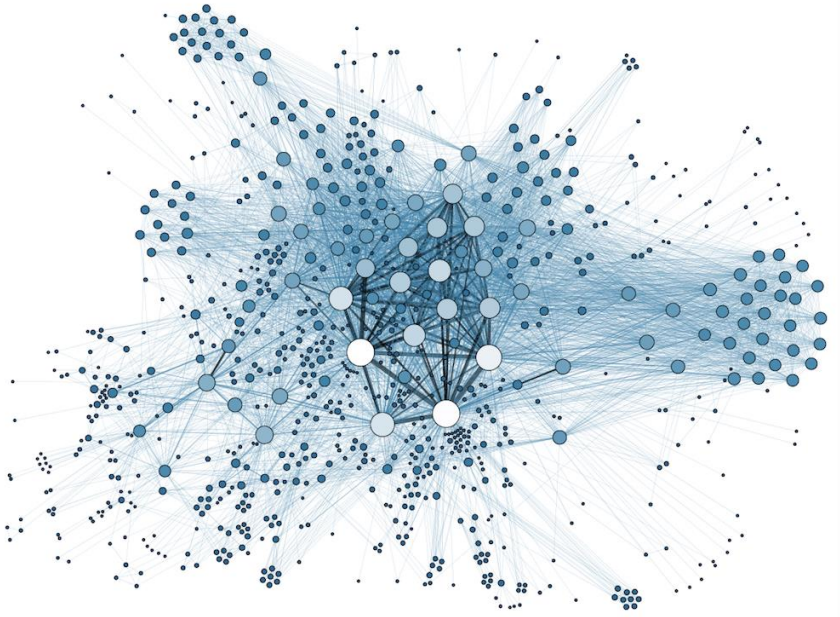
Sex
M
M
M
M
F

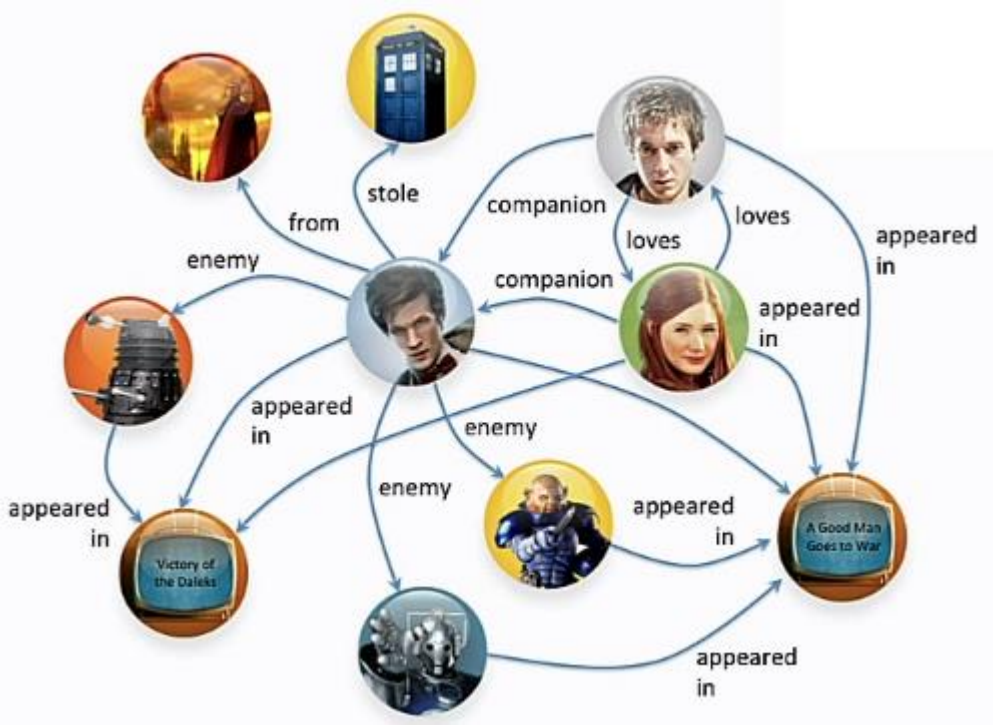
Golf
Y
N
Y
Y
N

Orientado a Grafos

- Orientado a Grafos: este modelo possui três componentes básicos: nós (vértices dos grafos), os relacionamentos (arestas) e as propriedades (conhecidos também como atributos).
- Este modelo é visto como multigrafo rotulado e direcionado, onde cada par de nós pode ser conectado por mais de uma aresta.
- A utilização deste modelo é muito útil quando é necessário fazer consultas demasiadamente complexas.
- O modelo orientado a grafos possui uma alta performance, permitindo um bom desempenho nas aplicações.

Orientado a Grafos

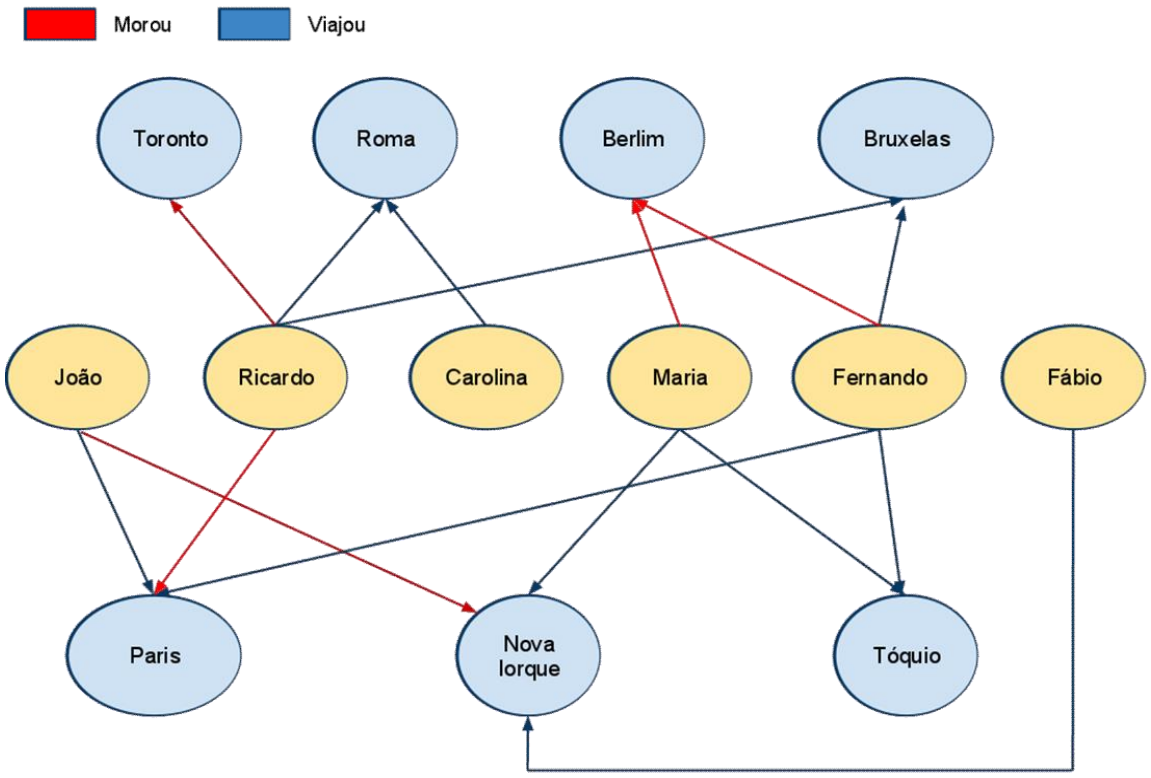




Orientado a Grafos

- Para exemplificar o que foi dito, podemos analisar a Figura 4 que representa uma aplicação que mantêm informações relativas à viagem.
  - Uma consulta pertinente seria: “Quais cidades foram visitadas anteriormente por pessoas que foram para Nova Iorque?”.
- No modelo de banco de dados relacional tal consulta poderia se mostrar complexa, pois não permitem que os dados sejam representados de uma forma natural.

Orientado a Grafos



Orientado a Grafos

- No exemplo da Figura do próximo slide, temos diversas pessoas:
  - João, Ricardo, Carolina, Maria, Fernando e Fábio que representam nós do grafo e estão conectadas a cidades que visitaram ou residiram.
  - Por exemplo: Ricardo viajou para Roma e Bruxelas e já residiu em Toronto e Paris.
  - A partir de cada cidade, precisamos dos relacionamentos de entrada que também sejam do tipo “viajou” e com isso encontramos pessoas que viajaram para o mesmo lugar que Ricardo, neste caso, Carolina e Fernando.

Orientado a Grafos

- Como exemplo, podemos citar o Neo4j que é um banco de dados open source.
  - O **Neo4J** trata-se de um banco de dados baseado em grafos desenvolvido em Java
- Além de possuir suporte completo para transactions, ele também trabalha com nós e relacionamentos

Modelos NoSQL

- Levando em consideração tudo o que foi dito, é fundamental ressaltar que nenhum modelo é superior a outro.

  Na realidade, o que ocorre é que um modelo pode ser mais adequado para ser utilizado em certas situações.

  Por exemplo, para a utilização de um banco de dados de manipulação de dados que frequentemente serão escritos, mas não lidos (um contador de hits na Web, por exemplo), pode ser usado um banco de dados orientado a documento como o MongoDB.

Modelos NoSQL

- Já aplicativos que demandam alta disponibilidade, onde a minimização da atividade é essencial, podemos utilizar um modelo orientado a colunas como o Cassandra.
- Aplicações que exigem um alto desempenho em consultas com muitos agrupamentos podem utilizar um modelo orientado a grafos.
- O importante é que no momento da criação do aplicativo os desenvolvedores utilizem a melhor solução que se encaixa no perfil desejado.
- Utilizar a solução adequada ao criar o banco de dados significa uma diminuição dos custos para a sua criação, assim como um banco eficiente no processamento de dados do ponto de vista das suas necessidades.

Vantagens Comuns

- Barato e fácil de implementar (open source)

  Os dados podem ser replicados para múltiplos nós (portanto, idênticos e tolerante a falhas) e podendo ser particionado

  - Nós com falhas podem ser facilmente substituídos;
  - Não tem apenas um ponto de falha;
- Fácil de distribuir
- Não requer um esquema
- Pode crescer ou diminuir
- Relaxa o requisito de consistência de dados (CAP)

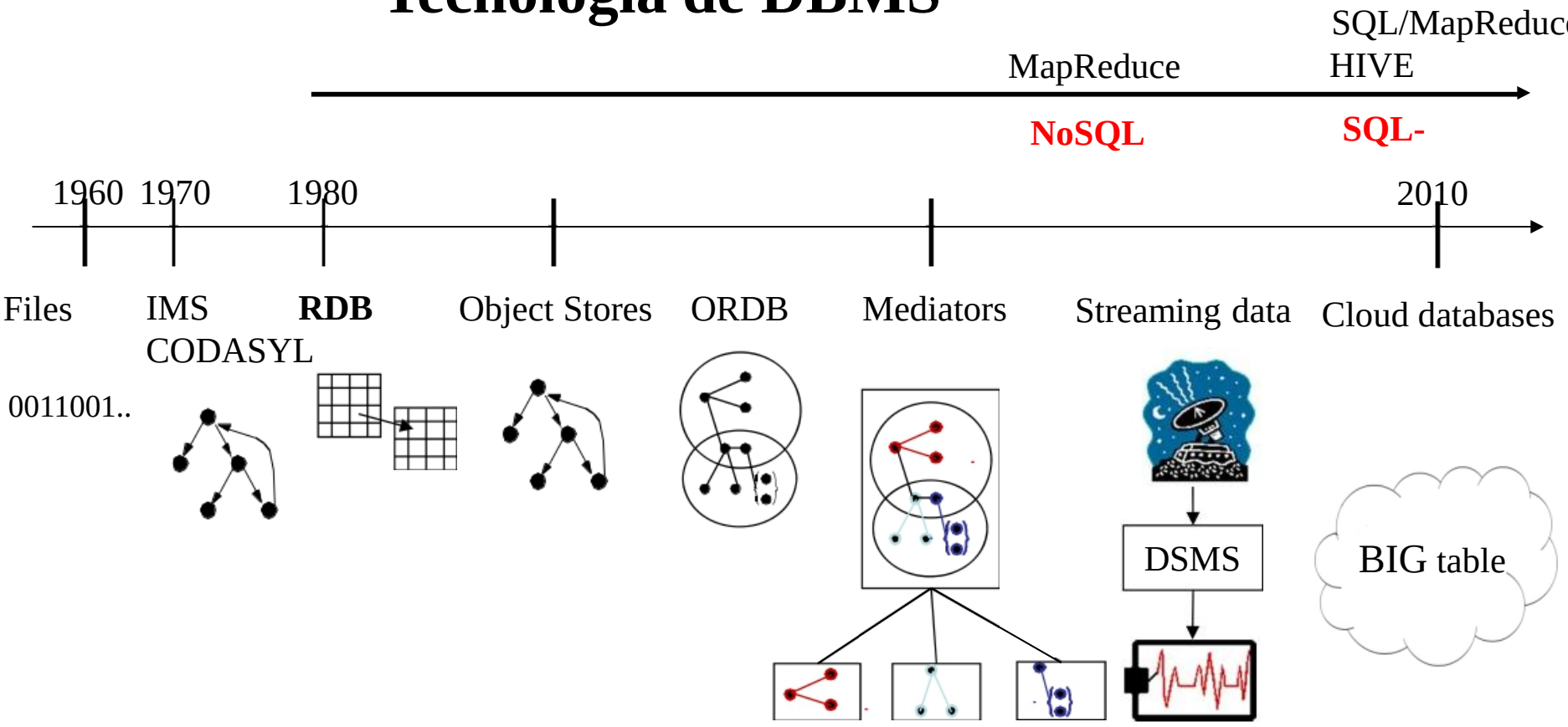
Do que estamos abrindo mão?

- joins
- group by
- order by
- Transações ACID
- SQL como linguagem de consulta às vezes frustrante, mas ainda poderoso
- Fácil integração com outras aplicações que suportam SQL

Tore Risch
Uppsala University, Sweden

UDBL

Evolução
Tecnologia de DBMS



Tipos de Suporte a DBMS

	Simple Data	Complex data
Query (SQL)	Relational DBMSs	Object-Relational DBMSs
No Query (NoSQL)	File systems (scalable) Storage managers	Object Stores

Classificação das Aplicações Modernas de DBMS

Complex Queries SQL, **New QL**	Business operations Business analytics Personal db	Multimedia search Custom Datatypes **Data streams** **Web analytics**
Simple Queries **SQL-**	**E-business**	**Web search**
No Queries **NoSQL**	Text, data logs Simple computations **Web documents**	CAD system Complex computations **Web surfing**
	Simple Data	Complex data

Tipos de Suporte de Banco de Dados

Complex Queries SQL, **New QL**	Relational DBMS	Object-Relational DBMS **Data Stream Mgmt. Syst.** **Virtuoso, Neo4J, Amos II**
Simple Queries **SQL-**	**Google App Engine (GQL)** **Amazon SimpleDB** **Microsoft Azure**	**Google search engine** **Facebook HIVE**
No Queries **NoSQL**	File systems **Content Mgmt. Syst.**	Object Stores **Google BigTable** **Yahoo Hadoop** **MongoDB,CouchDB**
	Simple Data	Complex data

Implementações

Resumo

- Os principais usuários dos bancos de dados NoSQL são sites de redes sociais como Twitter, Facebook, LinkedIn, e Digg;
- Uma das funcionalidades do Digg implementadas no Cassandra tem um banco de dados de 3 terabytes e 76 bilhões de colunas;

Nem todo problema é um prego e nem toda soluçao é um martelo.

Sugestões de Leitura

- Introdução aos bancos de dados NoSQL
  - http://www.devmedia.com.br/introducao-aos-bancos-de-dados-nosql/26044
- Banco de dados NoSQL: Um novo paradigma - Revista SQL Magazine 102
  - http://www.devmedia.com.br/revista-sql-magazine-102-25918/banco-de-dados-nosql-um-novo-paradigma-
- Bancos de dados não relacionais e o movimento NoSQL
  - http://blog.caelum.com.br/bancos-de-dados-nao-relacionais-e-o-movimento-nosql/

Bibliografia

- Base de dados de Bancos NoSQL
  - http://www.nosql-database.org/
- Banco de dados NoSQL: Um novo paradigma - Revista SQLMagazine 102
  - http://www.devmedia.com.br/sql-magazine-102-banco-de-dados-nosql-um-novo-paradigma-
- Bancos de dados não relacionais e o movimento NoSQL
  - http://blog.caelum.com.br/bancos-de-dados-nao-relacionais-e-o-movimento-nosql/

Perguntas???