UNIVERSIDADE FEDERAL DE ITAJUBÁ

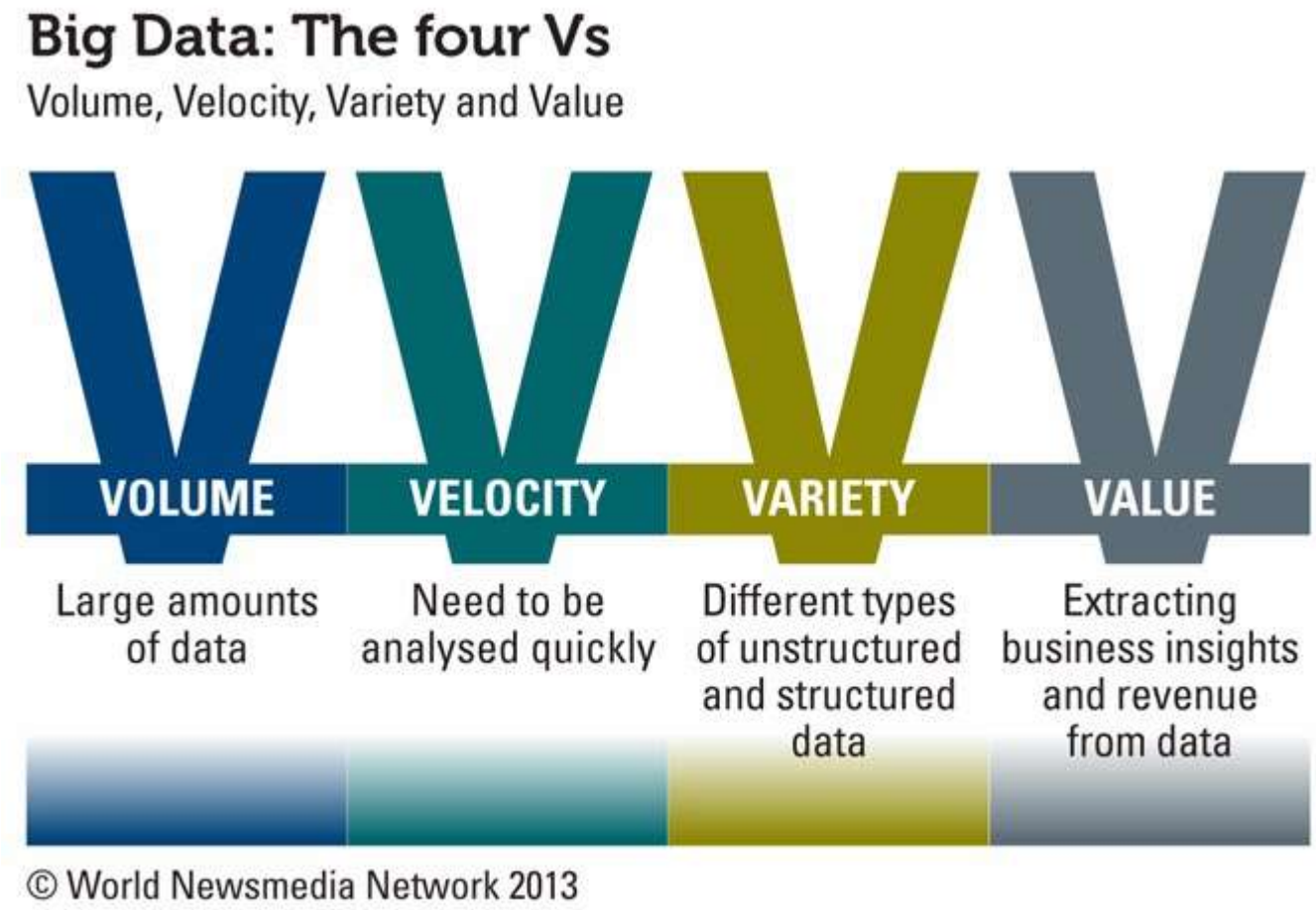
Banco de Dados II

COM 231

NoSQL
Aula 15

Vanessa Cristina Oliveira de Souza

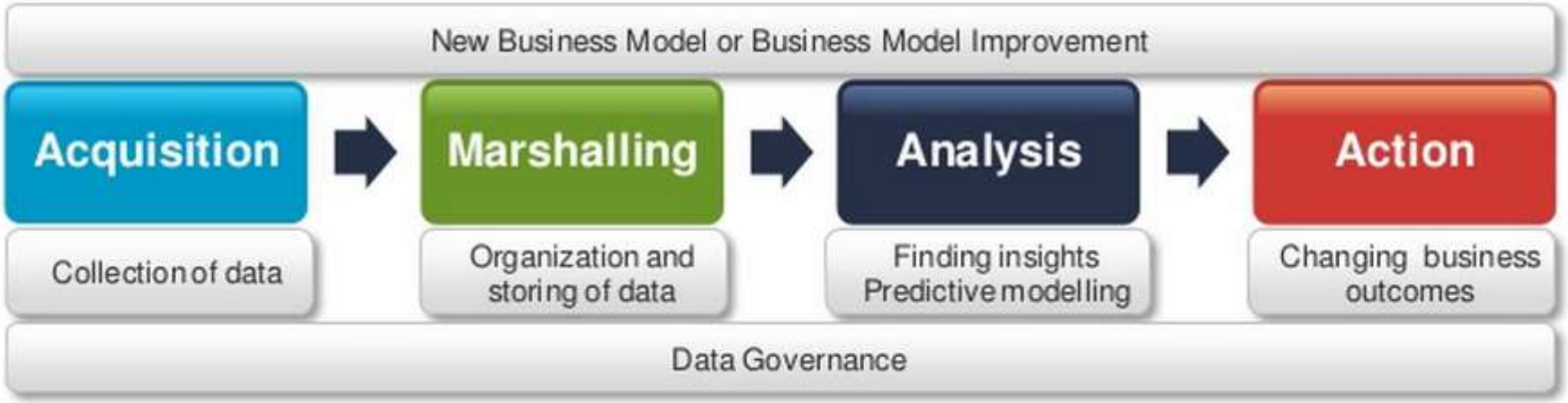
O que é Big Data?



O que é Big Data?

We have developed a Big Data strategy, methodology and delivery capability to help clients take advantage of big data:

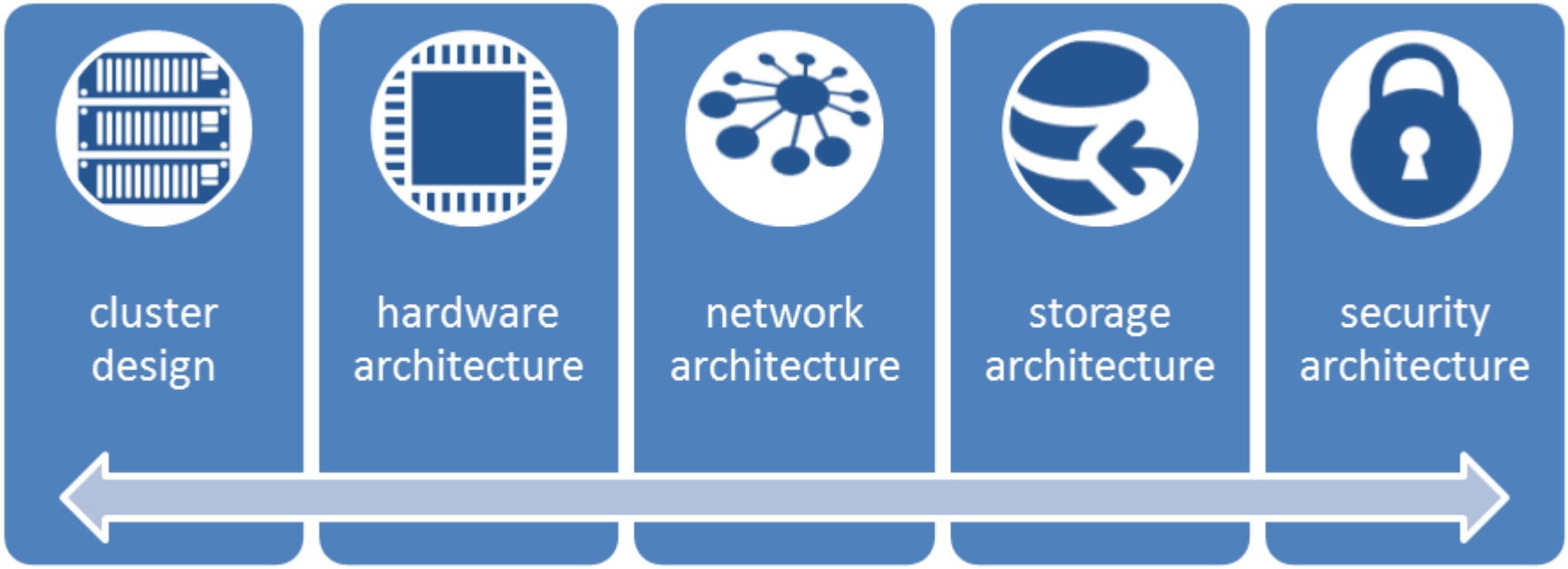
- **Big Data Process Model**



http://pt.slideshare.net/capgemini/understanding-big-data-14598849

O que é Big Data?

- ✓ Armazenamento



Tecnologias Associadas ao Big Data

- Sistemas Distribuídos
- Computação em Nuvem
- Virtualização
- Armazenamento

UNIVERSIDADE FEDERAL DE ITAJUBÁ

BIG DATA
X
BANCOS DE DADOS

Big Data x SGBDR

- Basicamente, os SGBDR não dão correto suporte aos 3Vs do Big Data
  - Volume
  - Velocidade
  - Variedade

Big Data x SGBDR

- Basicamente, os SGBDR não dão correto suporte aos 3Vs do Big Data
  - Volume
  - Velocidade
  - Variedade

Volume, Velocidade → Escalabilidade

Variedade → Esquema Rígido

Limitações dos bancos relacionais para armazenamento de *Big Data*

- **Leitura e escrita lenta**
  - com o aumento do tamanho dos dados, é possível provocar bloqueios e outros problemas de concorrência, o que leva a um rápido declínio na eficiência da leitura e escrita.

- **Capacidade limitada**
  - bancos de dados relacionais atuais não suportam um volume grande de dados no motor de busca.

- **Dificuldade de expansão**
  - mecanismos de correlação entre múltiplas tabelas, devido ao uso de chaves estrangeiras existentes em bancos de dados relacionais, não oferecem um bom mecanismo para a escalabilidade, sendo este um fator crucial para atender aos requisitos da *Big Data*.

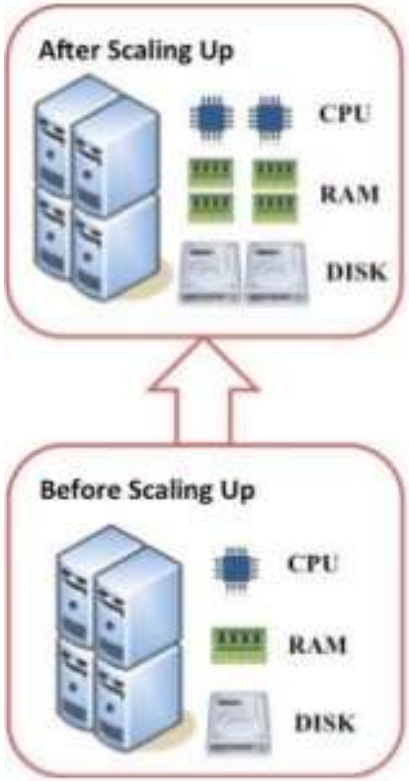
- **Dificuldade de indexação**
  - devido a grande quantidade de dados, indexá-los torna-se impraticável, pois o espaço ocupado pelos índices pode chegar a tamanhos críticos, o que leva a uma perda de desempenho.

Escalabilidade

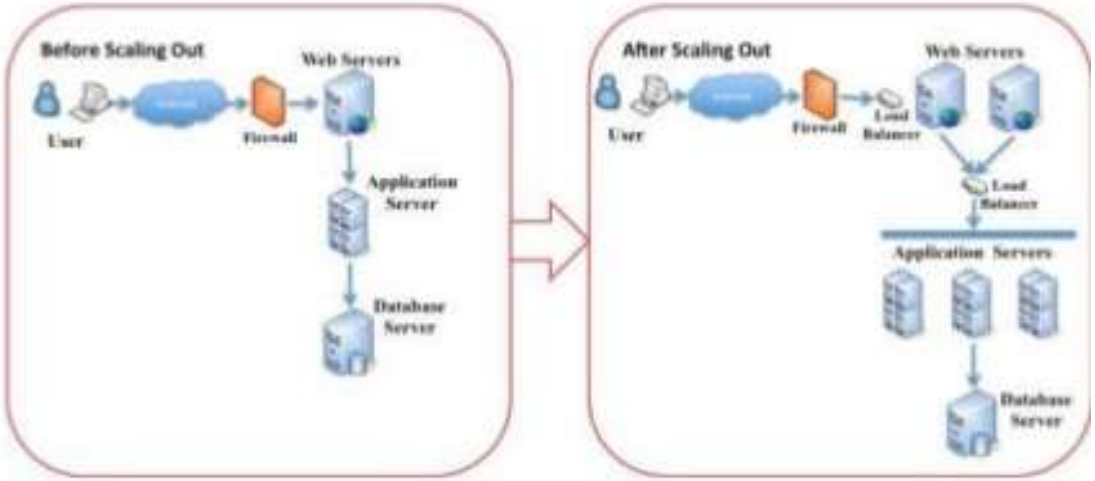
- Escalar significa quão bem uma solução particular se encaixa em um problema enquanto o escopo do problema aumenta.
- A medida que o banco cresce, o servidor precisa aumentar sua capacidade de processamento, armazenamento, memória, etc.
- **Problema**
  - O aumento de utilização de recursos computacionais causado pelo aumento de acessos e/ou aumento do tamanho físico do banco.
- **Soluções**
  - Escalabilidade Vertical
  - Escalabilidade Horizontal

Escalabilidade

> Vertical (Scaling Up)



> Horizontal (Scaling Out)



Qual é o problema?

- Apesar da riqueza de recursos, os SGBDs relacionais tendem a aumentar a complexidade de utilização com o aumento do fluxo de dados.

- Eles não foram criados para trabalhar em ambientes distribuídos e o escalonamento, em geral, se dá pelo aumento da capacidade da máquina servidora (escalonamento vertical).

- Acontece que ao aumentar muito o fluxo de dados, o uso de *cluster*s torna-se inevitável e o desempenho dos SGBDs relacionais cai.

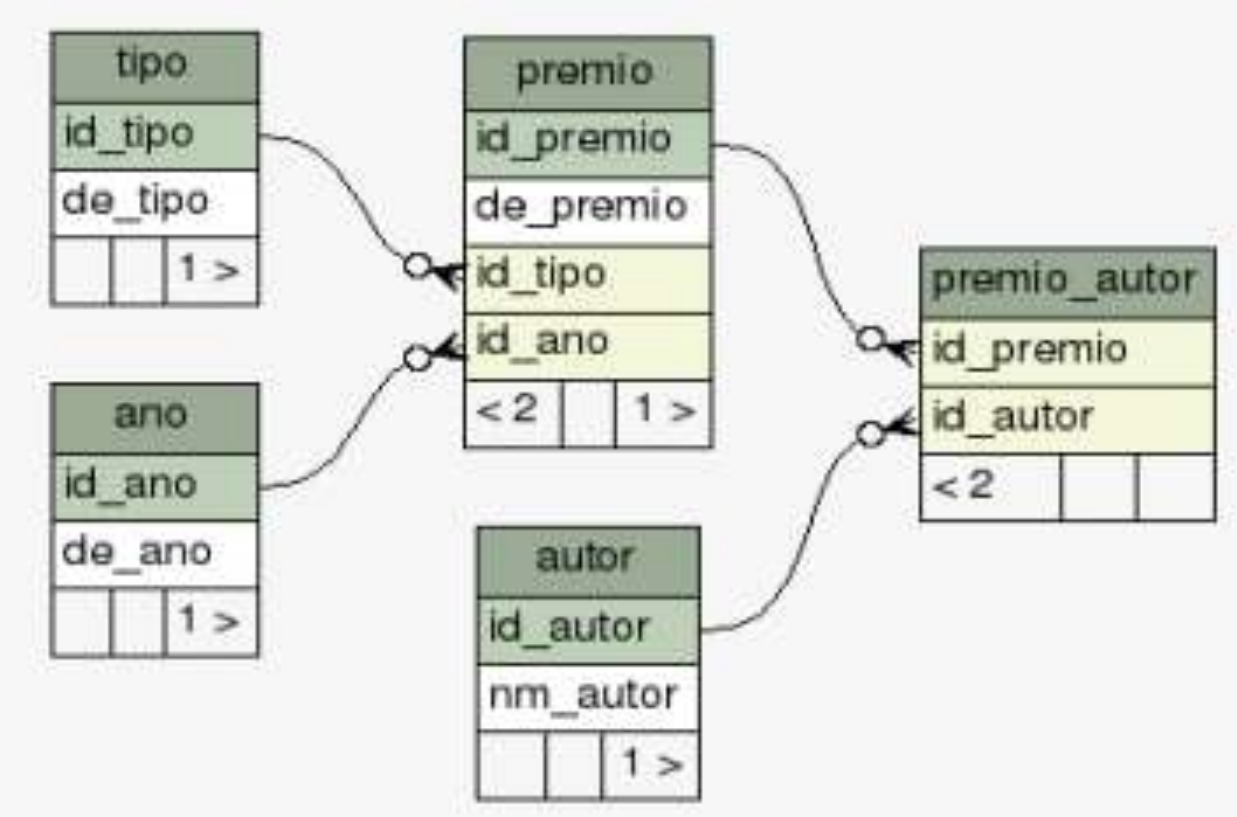
Qual é o problema?

- Um banco de dados relacional baseado em **ACID** não consegue facilmente escalar de forma horizontal devido sua **arquitetura** baseada em **tabelas**.

Esquema Rígido

- Um banco de dados relacional possui esquema definido, rígido, cuja alterações apresentam forte impacto na solução implementada.
  - Aplicações ficam ‘presas’ ao esquema que o modelo relacional pode representar
  - São criadas entidades no modelo que não existem no mundo real

Esquema Rígido



Outras Motivações...

- Mapeamento objeto-relacional é caro
  - Hoje, a programação orientada a objetos é o paradigma de programação mais prevalente. Uma aplicação persistindo objetos em um SGBDR requer um mapeamento entre o modelo do objeto e do modelo relacional. Definir isso toma tempo e processamento.
- O pensamento "One Size Fits All" é falho
  - Este pensamento quer dizer que os SGBDRs são vistos como um instrumento geral que pode lidar com todos os requisitos que uma aplicação pode ter no gerenciamento de dados. Uma vez que diferentes aplicações podem ter diferentes exigências sobre a consistência, desempenho e assim por diante, este pensamento pode ser falho.
- ACID não é sempre necessário
  - Em algumas aplicações, transações ACID não são necessárias, o que significa que pode se perder alguns *trade-offs* como desempenho.

Evolução dos modelos de Bancos de Dados

Modelo de Dados Hierárquicos
Primeiro modelo de dados a ser reconhecido. Usa uma estrutura de árvores onde cada registo é considerado uma coleção de campos ou atributos.

Modelo de Dados Relacional
Sucessor do modelo Hierárquico. Baseia-se no conceito de Entidades e Relacionamentos.

Melhorias nos SGBD's
Os Sistemas Gerenciadores de Banco de Dados começam a ser melhorados devido a grande aceitação dos usuários.

Modelo de Dados NoSQL
Surgem as primeiras alternativas aos modelos relacionais baseados em documentos, chave-valor ou famílias de colunas.

Modelo de Dados NoSQL
As bases de dados NoSQL começam a ser reconhecidas devido ao alto poder de performance e escalabilidade.

Evolução dos modelos de Bancos de Dados

UNIVERSIDADE FEDERAL DE ITAJUBÁ

NOSQL

NoSQL

- Não há definição rígida para o termo NoSQL - *Not Only SQL*
- Surgiu em 1998 com a criação de uma aplicação de banco de dados que não oferecia uma interface SQL, mas que ainda era baseado no modelo relacional.
- Mais tarde o termo passou a representar soluções que promoviam uma alternativa ao Modelo Relacional.

NoSQL

- **Principais Características:**
  - Não utilizam o modelo relacional e nem a linguagem SQL;
  - são projetados para rodar em *clusters* (sistemas distribuídos);
  - tendem a ser *open source*;
  - não possuem um esquema fixo, permitindo a persistência em qualquer registro.

NoSQL

O propósito das soluções NoSQL não é substituir o modelo relacional, mas sim serem utiizadas quando houver necessidade de maior **flexiblidade** na **estruturação** do banco de dados.

NoSQL

- Não é uma nova tecnologia, é uma nova abordagem
- Propõe um modelo alternativo de banco de dados
- Não é relacional
- Não respeita as propriedades ACID

NoSQL

Proposta!

Quebrar as correntes e a ditadura da base de dados relacional, junto com as propriedades ACID!

http://pt.slideshare.net/renatomolina/no-sql-renato

NoSQL

Persistência Poliglota

Utilizar diferentes armazenamentos de dados em diferentes circunstâncias.

Em vez de escolher o banco de dados relacional mais utilizado por todos, precisamos entender a natureza dos dados que estamos armazenando e como queremos manipulá-los.

NoSQL

- Soluções NoSQL costumam buscar atingir objetivos como baixa **latência**, alta **performance** e **escalabilidade**.

- Para tanto seguem um conjunto de propriedades chamado BASE
  - ***B**asically **A**vailable*
  - ***S**oft-state*
  - ***E**ventual consistency*

NoSQL

- ***B**asically **A**vailable*
  - Basicamente Disponível ou Disponibilidade Básica
  - *O sistema estará disponível basicamente todo o tempo*
  - Determina que o sistema deve estar disponível na maior parte do tempo, mas que subsistemas podem estar temporariamente inoperantes.

NoSQL

- ***S****oft-state*
  - Estado leve e consistente
  - *O estado do sistema não precisa estar sempre consistente*
  - Os desenvolvedores devem criar mecanismos para consistências de dados.

NoSQL

- ***E**ventual consistency*
  - Consistência Eventual
  - *O sistema torna-se consistente em um determinado momento*
  - Garante que se certo registro não sofre atualizações, eventualmente estará consistente e todas as leituras retornarão a última atualização do valor.

NoSQL

BASE – ACID alternative

- **B**asically **a**vailable: Nodes in the a distributed environment can go down, but the whole system shouldn't be affected.

- **S**oft State (scalable): The state of the system and data changes over time, even without input. This is because of the eventual consistency model.

- **E**ventual Consistency: Given enough time, data will be consistent across the distributed system.

NoSQL

ACID vs. BASE

ACID	BASE
☐Strong Consistency	☐Weak Consistency
☐Isolation	☐Availability first
☐Focus on "commit"	☐Best effort
☐Nested transactions	☐Approximated answers
☐Less Availability	☐Aggressive(optimistic)
☐Conservative (pessimistic)	☐Simpler!
☐Difficult evolution (e.g. schema)	☐Faster
	☐Easier evolution

NoSQL

Armazenam e replicam dados em **sistemas distribuídos**, muitas vezes através de *data centers*, para alcançar escalabilidade e confiabilidade.

NoSQL

- Por ser distribuído, podem haver conflitos:
  - De gravação : 2 clientes tentam gravar os mesmos dados ao mesmo tempo
  - De leitura-gravação : um cliente lê dados inconsistentes durante a gravação de outro cliente.

- Abordagens:
  - Pessimista: bloqueiam registros de dados para evitar conflitos.
  - Otimistas : detectam conflitos e os resolvem

NoSQL

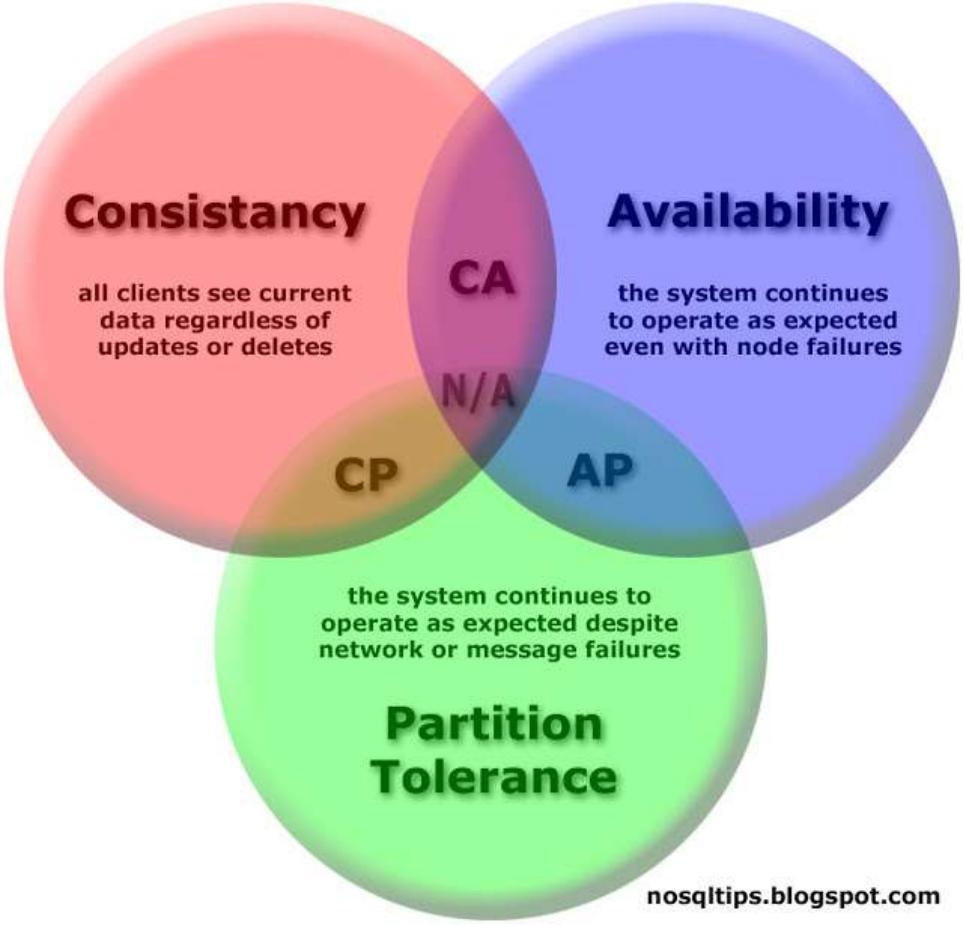
- Consistência
  - Às vezes é necessário sacrificar a consistência
  - Sempre é possível projetar um sistema para evitar inconsistências, mas, muitas vezes, é impossível fazê-lo sem sacrificar outras características.
  - Consequentemente, muitas vezes é necessário balancear a consistência com algo diferente.

NoSQL

- Consistência
  - Diferentes domínios têm diferentes tolerâncias à inconsistência, logo, precisamos levar essa tolerância em consideração ao tomarmos nossas decisões.

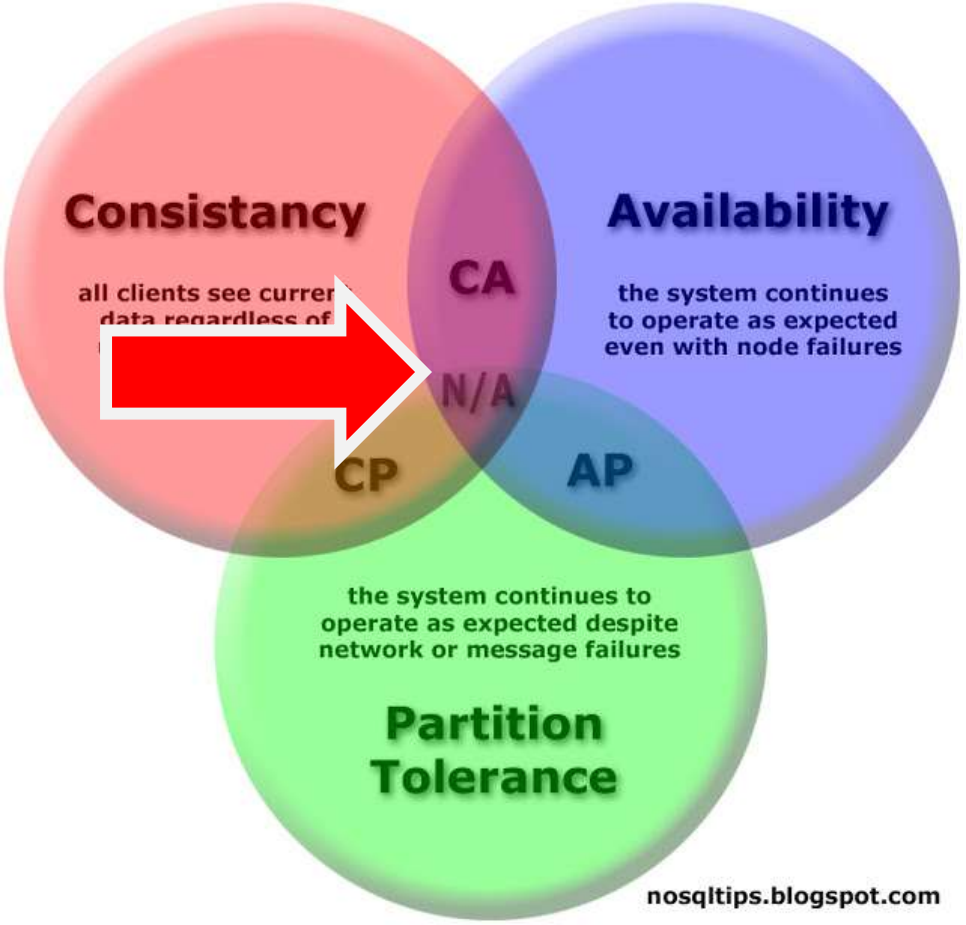
NoSQL

- Teorema CAP
  - Brewner 2000



NoSQL

- Teorema CAP
  - Brewner 2012



NoSQL

- Teorema CAP

NoSQL

- Teorema CAP

NoSQL

- Teorema CAP

Solicitação para reserva do quarto

NoSQL

- Teorema CAP

Comunicação com o Nó de Bombaim

Solicitação para reserva do quarto

NoSQL

- Teorema CAP

NoSQL

- Teorema CAP

NoSQL

- Teorema CAP

NoSQL

- Teorema CAP

NoSQL

- Teorema CAP

NoSQL

- Teorema CAP
  - Soluções
    - Designar um nó como mestre
    - Permitir que ambos os nós continuassem aceitando reservas, mesmo que a conexão caísse.
  - Questões
    - Quão tolerante sua aplicação pode ser quanto a leituras desatualizadas?
    - Quanto tempo sua aplicação tolera inconsistência?

UNIVERSIDADE FEDERAL DE ITAJUBÁ

MODELAGEM NOSQL

Modelagem NoSQL

- Modelagem representa uma **abstração** da realidade
  - Representar o mundo real em um determinado modelo conceitual e físico (de armazenamento)

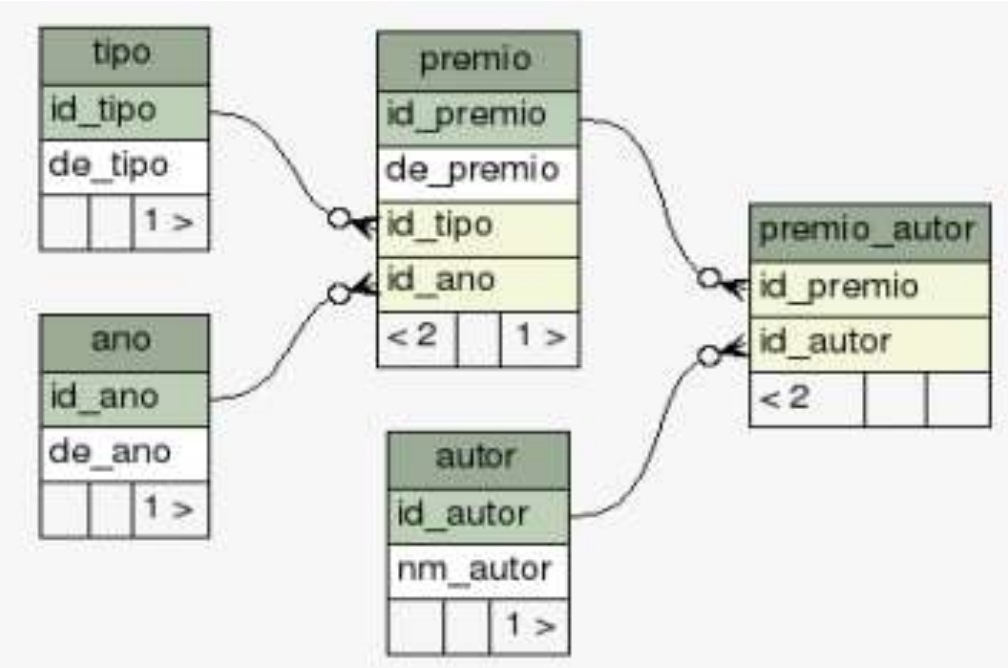
- Problemas do Modelo Relacional
  - Normalização
    - Evita redundância
    - Espalha os dados pelo disco rígido -> maior tempo para realizar consultas e realizar alterações
  - Esquema Rígido
    - Utilização de valores Nulos
    - Grande impacto no caso de alteração do esquema
  - Problemas de semântica

Modelagem NoSQL

- Schema-less
  - Não há esquema pré-definido
- Modelagem baseada nas consultas da aplicação
- Dados agregados
  - Redundância
  - Incorporação

Modelagem NoSQL

- Modelo Relacional
  - A modelagem é decidida pela estrutura de dados



Modelagem NoSQL

- Modelo NoSQL
  - A modelagem é **<u>focada nas consultas</u>** que serão executadas nos dados

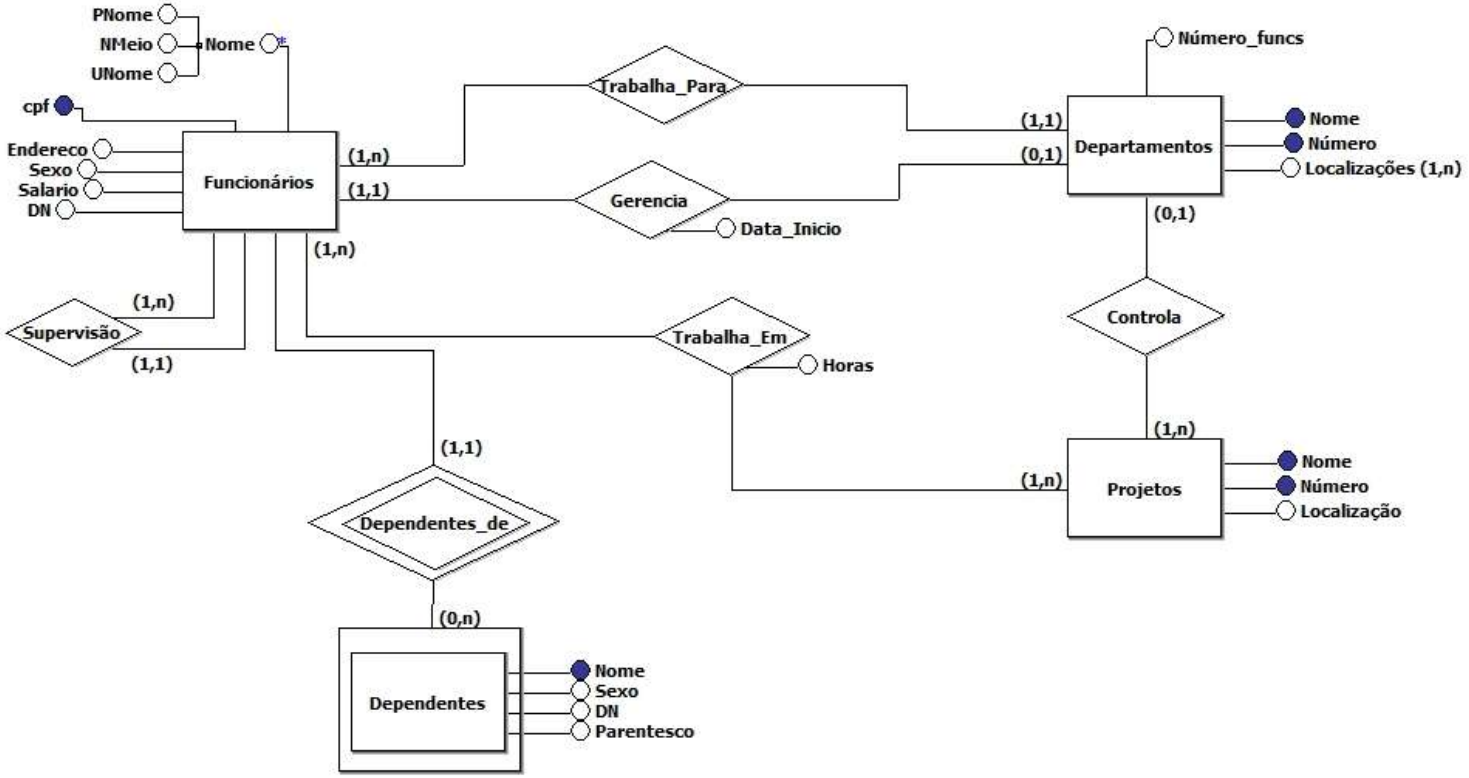
Premio
Id
Descrição
Ano
Tipo
Autores Nome do Autor

Modelagem NoSQL

- Em contraste com o banco relacional, o NoSQL procura agrupar dados agregados juntos no disco
  - Reduz o número de requisições ao disco
  - Melhora tempo de acesso

Modelagem NoSQL

- Modelo Relacional
  - A modelagem é decidida pela estrutura de dados



Modelagem NoSQL

- Embora esta nova geração de bancos de dados seja livre de esquema, a importância do modelo de dados será sempre entender e demonstrar o armazenamento de dados.

- O modelo para Big Data fornece uma maneira visual para gerenciar as fontes de dados, e cria uma arquitetura para que otimiza a reutilização de dados e reduz os custos de computação.

Modelagem NoSQL

- Diferente do modelo relacional, os bancos NoSQL possuem diversos modelos de armazenamento.
  - A modelagem dos dados também depende do modelo de armazenamento utilizado

Modelagem NoSQL

How I Learned to Stop Worrying and Love NoSQL Databases*

Francesca Bugiotti[1], Luca Cabibbo[2], Paolo Atzeni[2], and Riccardo Torlone[2]
[1]CentraleSupélec & INRIA & Université Paris-Sud and [2]Università Roma Tre

Abstract. The absence of a schema in NoSQL databases can disorient traditional database specialists and can make the design activity in this context a leap of faith. However, we show in this paper that an effective design methodology for NoSQL systems supporting the scalability, performance, and consistency of next-generation Web applications can be indeed devised. The approach is based on NoAM (NoSQL Abstract Model), a novel abstract data model for NoSQL databases, which is used to specify a system-independent representation of the application data. This intermediate representation can be then implemented in target NoSQL databases, taking into account their specific features.

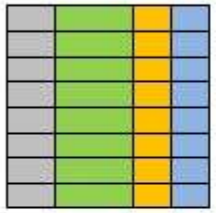
UNIVERSIDADE FEDERAL DE ITAJUBÁ

MODELOS DE DADOS NOSQL

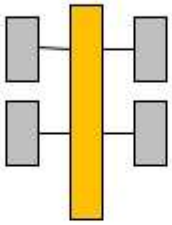
Modelos de Dados

After NoSQL

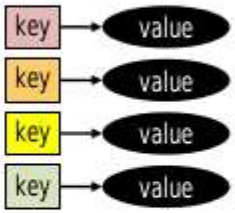
Relational



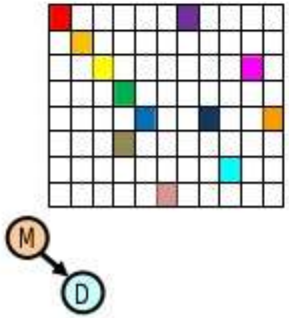
Analytical (OLAP)



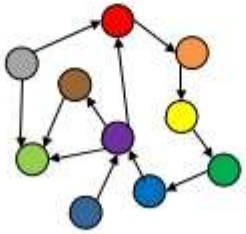
Key-Value



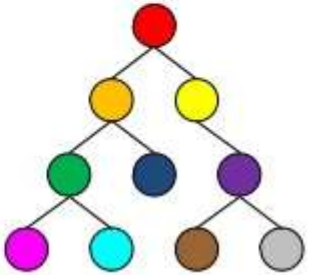
Column-Family



Graph



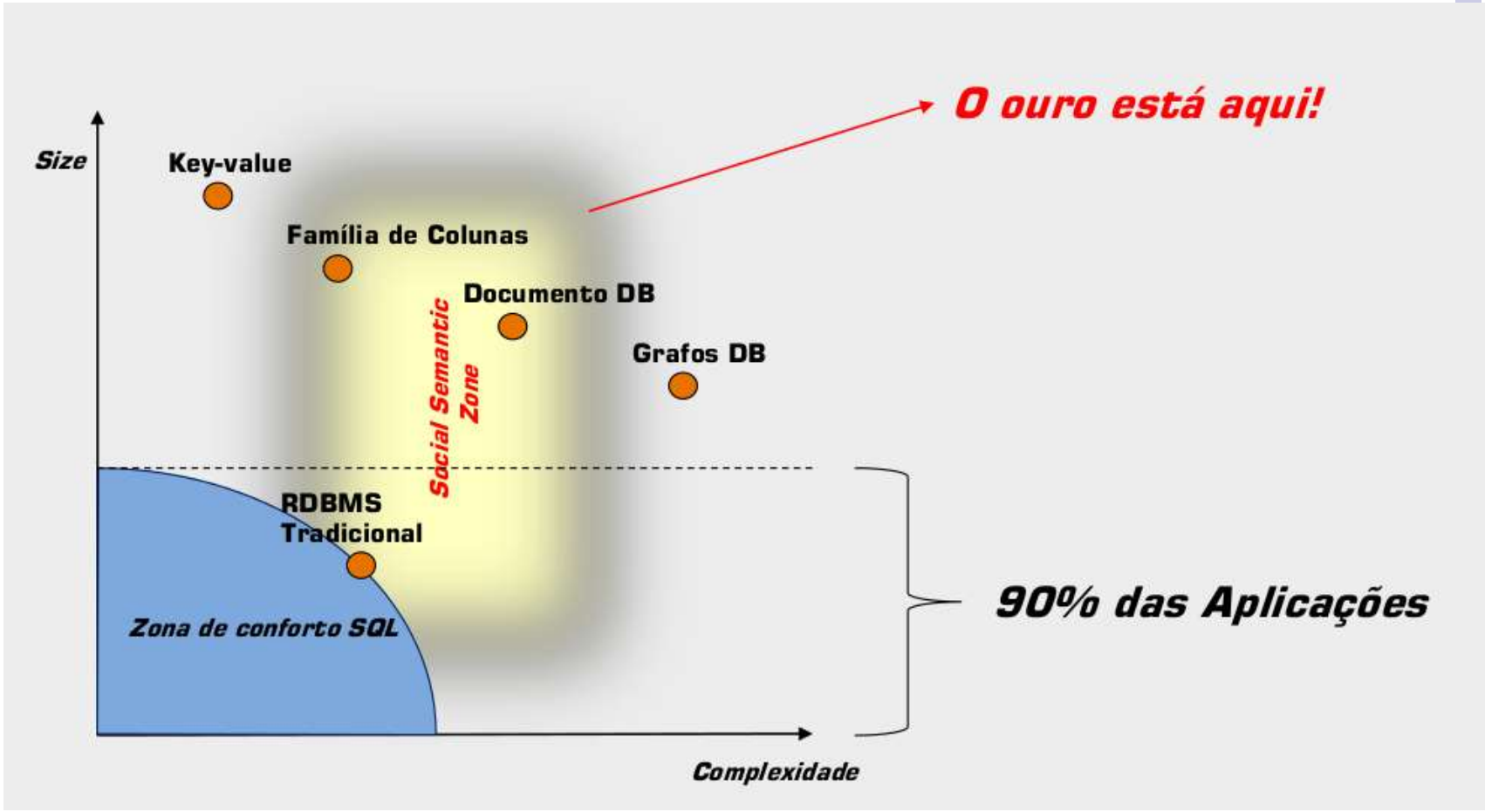
Document



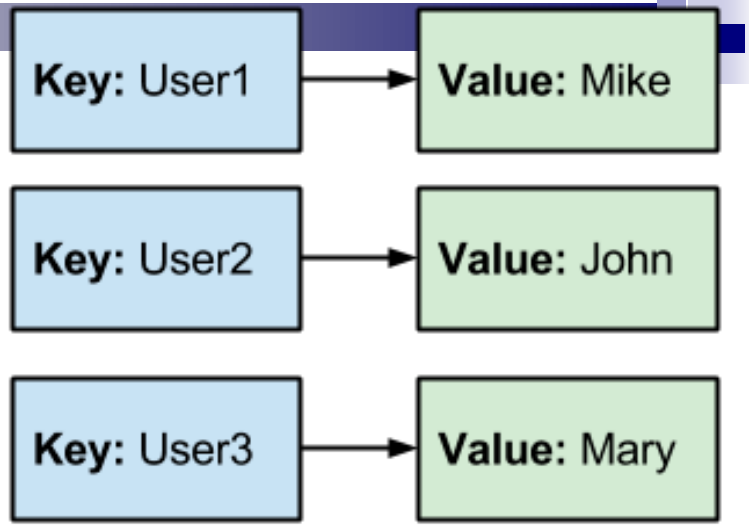
Modelos NoSQL

QUANDO UTILIZAR CADA MODELO

Modelos NoSQL



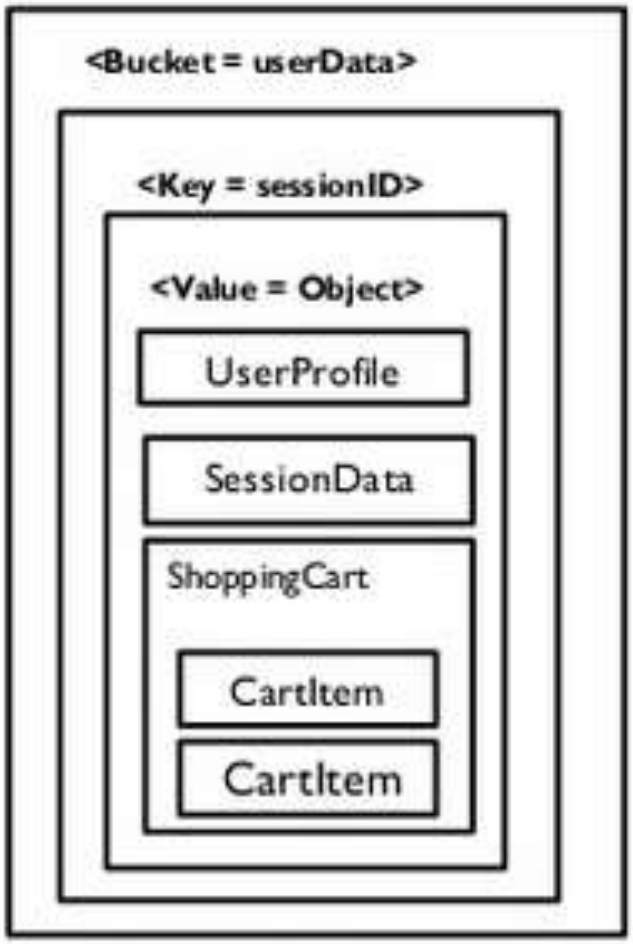
Chave-Valor



- Modelo mais simples
- Pobre semanticamente
- Utiliza Tabela Hash

- Key -> chave primária única
  - Ao gravar no banco, caso a chave já exista, o valor é sobrescrito
- Valor
  - Campo BLOB (JSON, XML, string, etc)
  - Pode armazenar qualquer valor

Chave-Valor



- Valor é armazenado, sem preocupação com o que representa
- Aplicação faz o tratamento e se preocupa com entendimento do valor
- Muito escalar devido o acesso somente pela chave
- Pode armazenar tudo num Bucket ou criar buckets de domínio
- **LIMITAÇÃO: Consulta só pela chave, retornando o valor. Valor não pode ser consultado por atributo.Geração das chaves.**

Chave-Valor

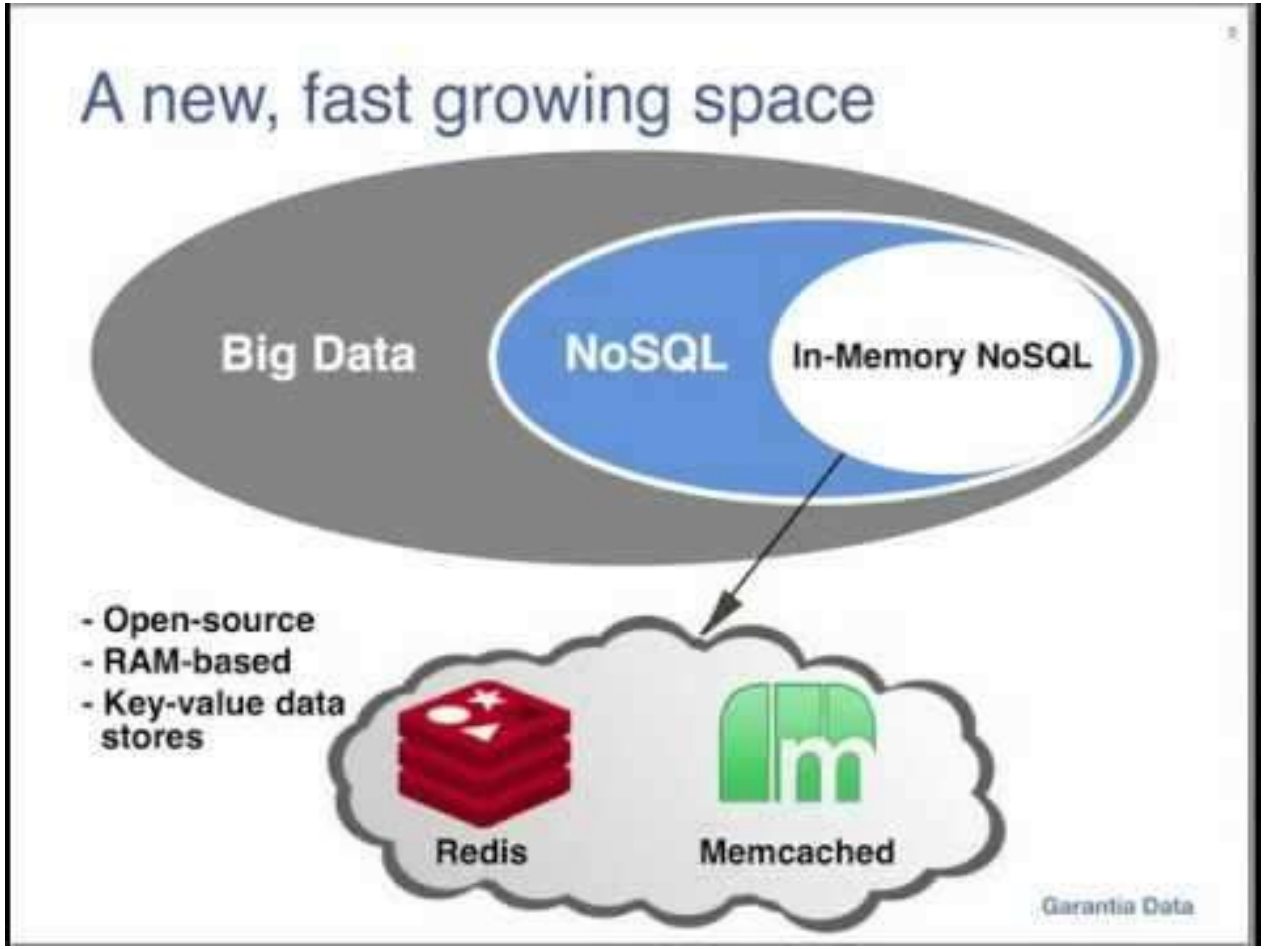
- Quando utilizar
  - Armazenamento de informações de sessão
  - Perfis de usuários, preferências
  - Dados de carrinhos de compras

- Quando **não** utilizar
  - Relacionamentos entre dados
  - Transações com múltiplas operações (rollback)
  - Consultas por dados
  - Operações por conjuntos

Chave-Valor

redis

Chave-Valor


A new, fast growing space
Big Data
NoSQL
In-Memory NoSQL
- Open-source
- RAM-based
- Key-value data stores
Redis
Memcached
Garantia Data

Orientado a Documentos

- Bancos de dados são coleções de documentos
  - Cada documento é identificado por uma chave
  - Os documentos são estruturas de dados composta por uma quantidade variável de campos, com tipos de dados diversos
    - Inclusive um campo pode conter outro documento

```
Joel = {
nome: "Joel da Silva",
disciplina: "Banco de Dados",
curso: "Pós-graduação em Gestão de Tecnologia da Informação"
}
```

Orientado a Documentos

- O documento pode conter objetos e arrays (listas ordenadas)
  - XML
  - JSON
  - BSON

- Acesso fácil aos atributos internos do documento
- Uso de visões materializadas para agregar informações ou estabelecer consultas específicas nos agregados
- Possível fazer consulta dos dados dentro do documento no nível de atributo
- **LIMITAÇÃO: Documentos armazenados devem ser de uma mesma coleção.**

```
{
"país": "Brasil",
"população": 201032714,
"PIB total em trilhões de dólares":  2.422,
"faz fronteira com": [
     "Argentina",
     "Bolívia",
     "Colômbia",
     "Guiana Francesa",
     "Guiana",
     "Paraguai",
     "Peru",
     "Suriname",
     "Uruguai",
     "Venezuela"
],
"cidades": {
   "capital" : "Brasília",
   "mais populosa": "São Paulo"
}
}
```

Orientado a Documentos

- Quando utilizar
  - Registro de eventos (log)
  - Sistema de gerenciamento de conteúdo, plataformas de blog
  - Análises web ou em tempo real
  - Aplicativos de comércio eletrônico

- Quando **não** utilizar
  - Transações complexas que abranjam diferentes operações
  - Consultas em estruturas agregadas variáveis

Orientado a Documentos

Orientado a Documentos

- Confusão com o modelo chave-valor

Orientado a Colunas

- Armazena fisicamente os dados divididos por colunas
  - Cada coluna é um arquivo no disco
  - “chave de linha”

```
Joel = {
nome: "Joel da Silva",
disciplina: "Banco de Dados",
curso: "Pós-graduação em Gestão de Tecnologia da Informação"
}
```

Orientado a Colunas

Row Oriented Database

date	price	size
2011-01-20	10.1	10
2011-01-21	10.3	20
2011-01-22	10.5	40
2011-01-23	10.4	5
2011-01-24	11.2	55
2011-01-25	11.4	66
...	...	...
2013-03-31	17.3	100

Table of Data

date	price	size
2011-01-20	10.1	10
2011-01-21	10.3	20
2011-01-22	10.5	40
2011-01-23	10.4	5
2011-01-24	11.2	55
2011-01-25	11.4	66
...	...	...
2013-03-31	17.3	100

Column Oriented Database

date	price	size
2011-01-20	10.1	10
2011-01-21	10.3	20
2011-01-22	10.5	40
2011-01-23	10.4	5
2011-01-24	11.2	55
2011-01-25	11.4	66
...	...	...
2013-03-31	17.3	100

Orientado a Colunas

Row Oriented
(RDBMS Model)

Multi-valued

id	Name	Age	Interests
1	Ricky		Soccer, Movies, Baseball
2	Ankur	20	
3	Sam	25	Music

null

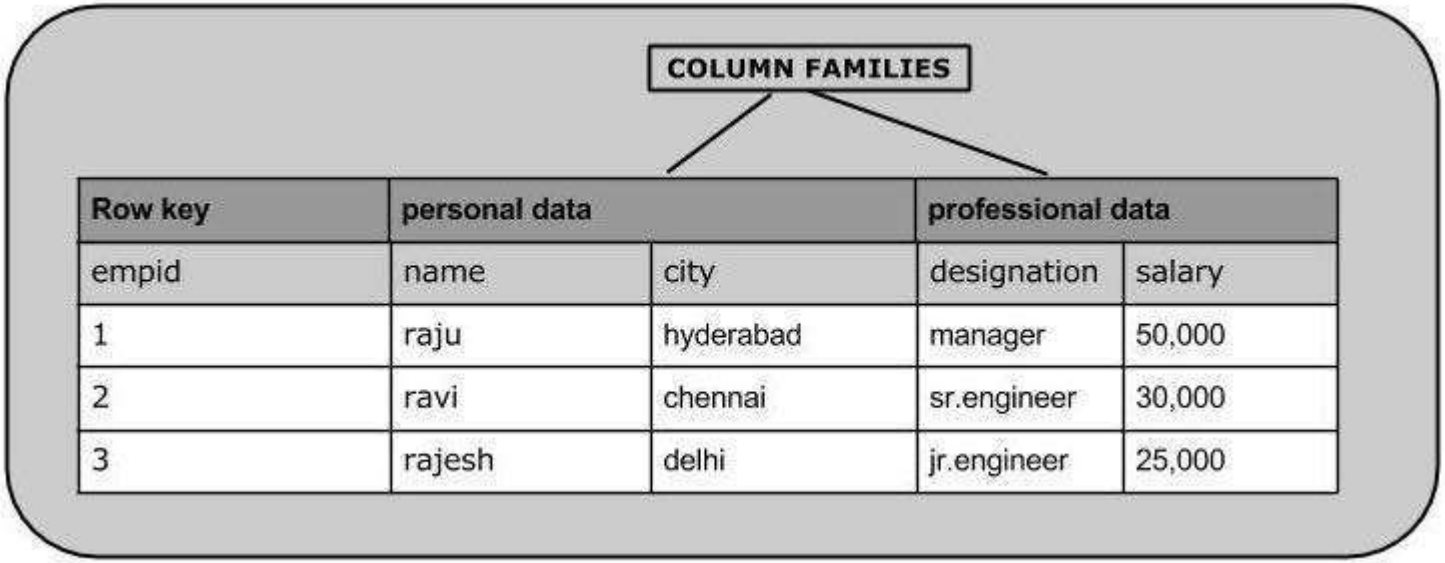
Column Oriented
(Multi-value sorted map)

id	Name
1	Ricky
2	Ankur
3	Sam

id	Age
2	20
3	25

id	Interests
1	Soccer
1	Movies
1	Baseball
3	Music

Orientado a Colunas



Row key	personal data		professional data	
empid	name	city	designation	salary
1	raju	hyderabad	manager	50,000
2	ravi	chennai	sr.engineer	30,000
3	rajesh	delhi	jr.engineer	25,000

Orientado a Colunas

Column-Oriented Database vs Row Oriented Database

Operation	Column-Oriented Database	Row-Oriented Database
Aggregate Calulation of Single Column e.g. sum(price)	✔ fast	slow
Compression	✔ Higher. As stores similar data together	-
Retrieval of a few columns from a table with many columns	✔ Faster	has to skip over unnecessary data
Insertion/Updating of single new record	Slow	✔ Fast
Retrieval of a single record	Slow	✔ Fast

Orientado a Colunas

- Quando usar
  - Registro de eventos (log)
  - Sistemas de gerenciamento de conteúdo
  - Contadores
  - Data Warehouse
- Quando **não** usar
  - Sistemas que requerem ACID para leitura e gravação

Orientado a Colunas

Orientado a Grafos

- Baseado na Teoria dos Grafos
- Dois elementos principais: Nós e Relacionamentos
- Permite armazenar relacionamentos entre entidades
- Possibilita encontrar padrões interessantes entre Nós
- Uma consulta é uma TRAVESSIA (forma de percorrer)
- Custo baixo para inserir relacionamentos novos (oposto de BD Relacional)
- Relacionamentos persistidos e não calculados no momento da consulta
- Modelagem de fácil entendimento

Orientado a Grafos

- Quando usar
  - Dados conectados (redes sociais, jogos)
  - Roteamento, envio e serviços baseados em localização
  - Mecanismos de recomendação
- Quando não usar
  - Sistemas de atualização em lote
  - Sistemas muito grandes

Orientado a Grafos

Modelos NoSQL

Graph	Column	Document	Persistent Key/Value	Volatile Key/Value
neo4j	BigTable (Google)	MongoDB (~BigTable)	Dynamo (Amazon)	memcached
FlockDB (Twitter)	HBase (BigTable)	CouchDB	Voldemort (Dynamo)	Hazelcast
InfiniteGraph	Cassandra (Dynamo + BigTable)	Riak (Dynamo)	Redis	
	Hypertable (BigTable)		Membase (memcached)	
	SimpleDB (AmazonAWS)		Tokyo Cabinet	

Modelos NoSQL

Rank May 2016	Rank Apr 2016	Rank May 2015	DBMS	Database Model
1.	1.	1.	Oracle	Relational DBMS
2.	2.	2.	MySQL	Relational DBMS
3.	3.	3.	Microsoft SQL Server	Relational DBMS
4.	4.	4.	MongoDB	Document store
5.	5.	5.	PostgreSQL	Relational DBMS
6.	6.	6.	DB2	Relational DBMS
7.	↑ 8.	↑ 8.	Cassandra	Wide column store
8.	↓ 7.	↓ 7.	Microsoft Access	Relational DBMS
9.	9.	↑ 10.	Redis	Key-value store
10.	10.	↓ 9.	SQLite	Relational DBMS

Referências

- NoSQL – um guia conciso para o Mundo Emergente da Persistência Poliglota.
  - Sadalage & Fowler
  - Editora Novatec
  - 2013