

# FIAP GRADUAÇÃO

# Cloud Computing

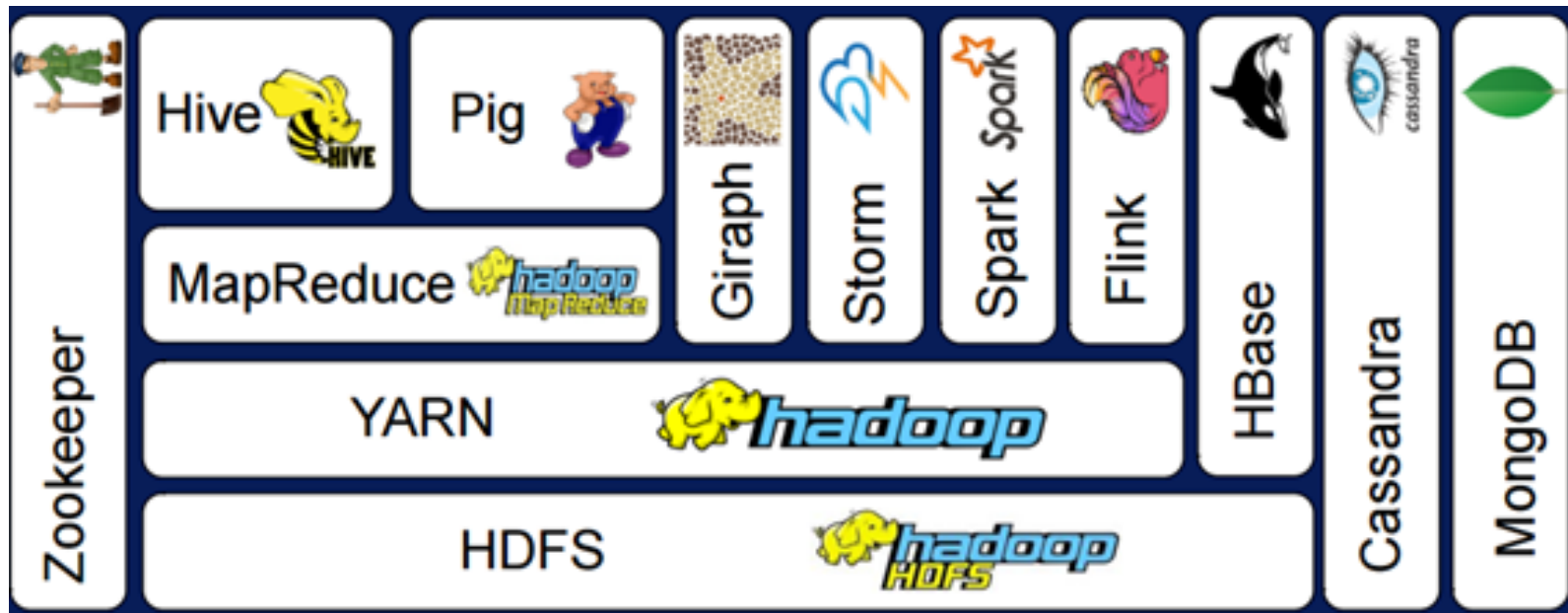
BIG DATA & Analytics

PROF. MILTON

# Fundamentos de Big Data Conceitos

# Ecossistema Big Data

FIAP



# Habilidades e conhecimentos essenciais ao cientista de dados

## Sistemas

- Análise
- Implementação/Teste
- Projeto
- Gestão de dados
- Conhecimento de diferentes tecnologias
- Desenvolvimento de Metodologias
- Programação
- Operações/Manutenção
- Integração
- Documentação



## Técnicas

- Linguagem de programação
- Banco de Dados/Data Warehouse
- Plataformas Open Source
- Domínio de diferentes pacotes
- Visualização de dados
- Arquitetura de Redes
- Sistemas Operacionais
- Computação em nuvem

## Negócios

- Habilidades Interpessoais
- Comunicação
- Automotivação
- Conhecimento específico do setor/negócio
- Habilidade de análise macro
- Negócios online/ecommerce

# Antes de Começarmos

## INTELIGÊNCIA

### DADOS x INFORMAÇÃO x CONHECIMENTO

#### Dados

- Observações sobre o estado do mundo; conjunto de fatos distintos e objetivos relativos a eventos.
- Fragmento bruto, parcial e objetivo da realidade.
- Sequencia de símbolos quantificados ou quantificáveis.

#### Informação

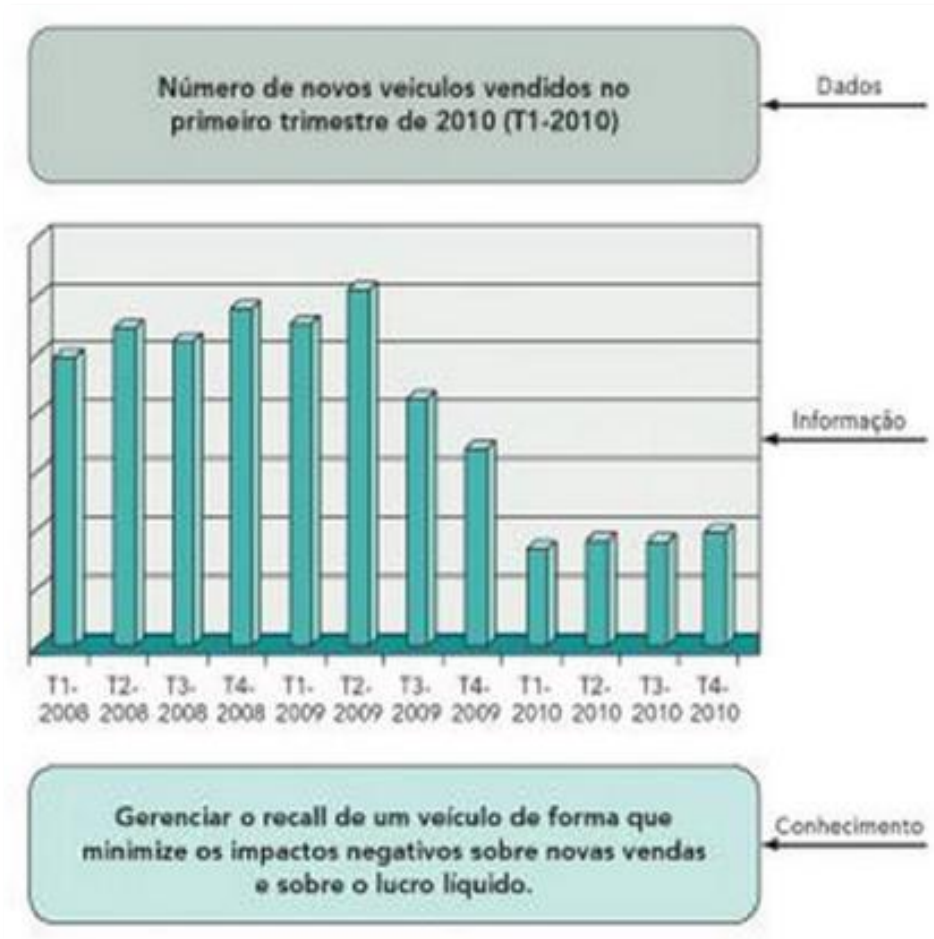
- Dados dotados de relevância e propósito.
- Dados, imagens e textos organizados e comunicados, que fazem sentido ao receptor.
- Abstração que não pode ser formalizada através de teoria lógica ou matemática que representa algo significativo para alguém, por meio de dados contidos em textos, imagens e sons.

#### Conhecimento

- Informação valiosa da mente humana; inclui reflexão, síntese e contexto.
- Crença verdadeira e justificada baseada em experiência, valores e contexto, que proporciona estrutura para avaliação e incorporação de novas experiências e novas informações.
- Conjunto específico e sistematizado de informações, reconhecido aceito e assimilado pelo indivíduo por meio de seu acervo pessoal cognitivo, emocional e experimental.

# Antes de Começarmos

## DADOS x INFORMAÇÃO x CONHECIMENTO





**Em 2020, teremos 44 zettabytes de dados armazenados no mundo!**

FONTE: IDC



“

A cada **dois dias** nós criamos 5 exabytes de dados, isso é o **mesmo** que foi criado do início da civilização até **2003**.

*-Eric Schmidt (CEO do Google)*

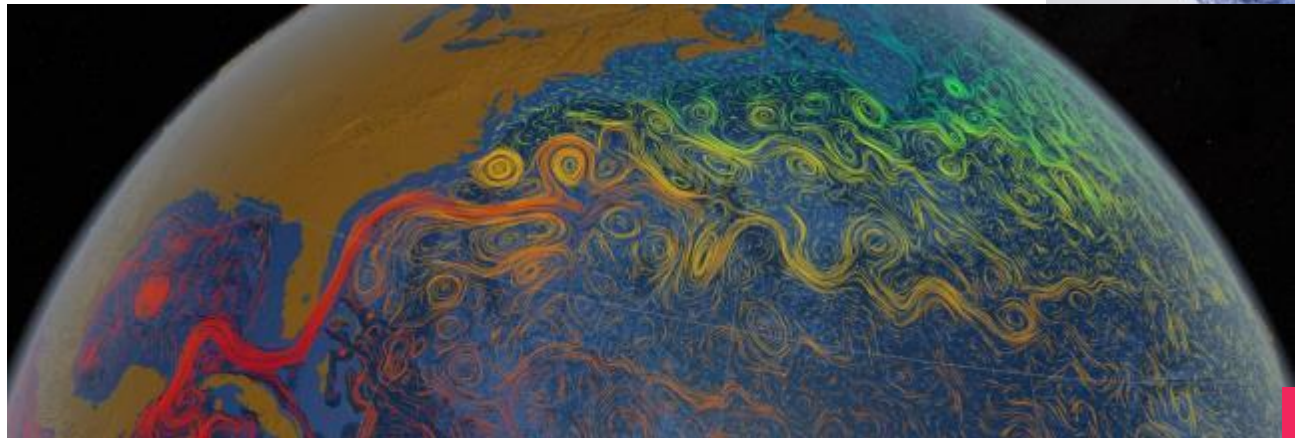
# Visualizando o volumes de dados

Byte	1 grão de arroz
Kilobyte	1 xícara de arroz
Megabyte	8 sacos de arroz
Gigabyte	1 container de arroz
Terabyte	2 navios cargueiros
Petabyte	suficiente para cobrir a cidade de Campinas.
Exabyte	suficiente para cobrir os estados de Minas Gerais, Rio de Janeiro, Espírito Santo e São Paulo.
Zettabyte	preenche o oceano Pacífico.


# *Big Data* não é novidade! FIAP



- Há décadas temos necessidade de processar grandes volumes de dados.
- Astronomia, geologia, oceanografia, meteorologia sempre trabalharam com grande quantidade de dados.



# Big Data não é novidade! FIAP



**IBM IMS**

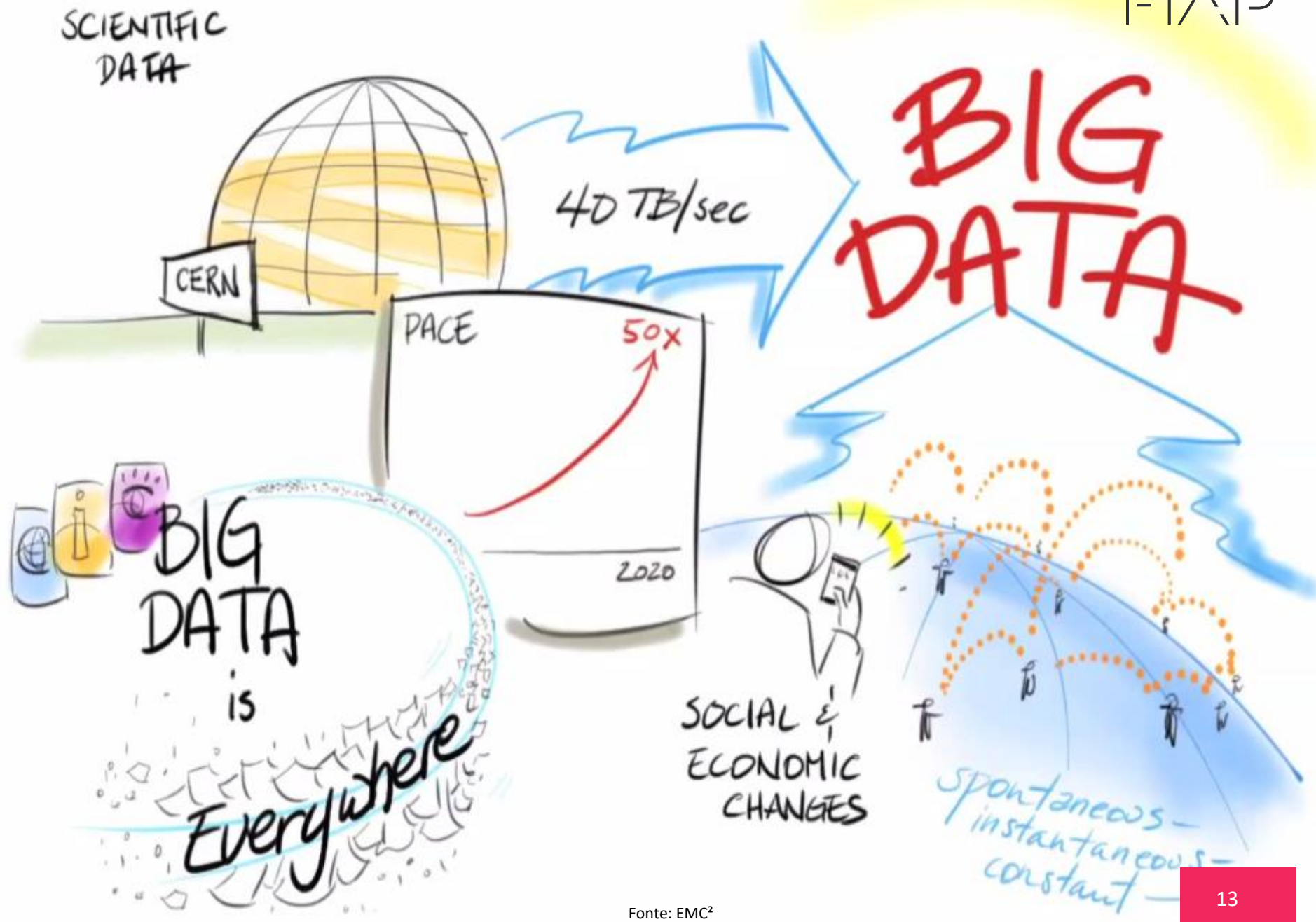
What would you do with 100,000 transactions per second?

→ Learn more

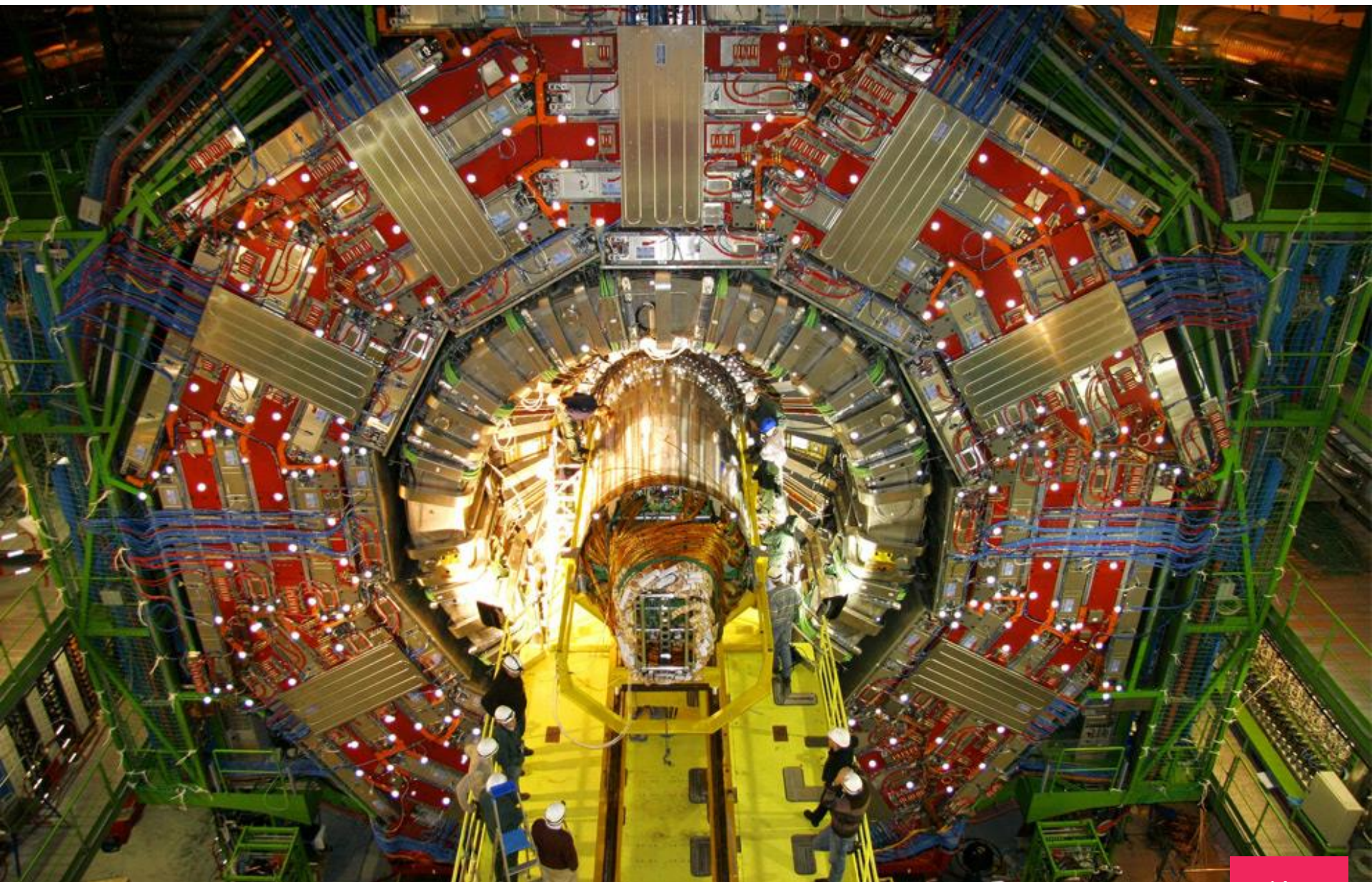
```
EDIT      SYSADM.DEMO.JCLLIB(HIMSESDS) - 01.01      Columns 00001 00072
Command ==>                                         Scroll ==> CSR

***** Top of Data *****
000001 //SYSADMA JOB A123,'BIN-7 QUASAR',CLASS=A,MSGCLASS=Y,NOTIFY=&SYSUID
000002 //*
000003 //* RUN THIS JOB TO CREATE IMS V8 DATASETS
000004 //* AUTHOR : QUASAR CHUNAWALA DATE : 11/06/2010
000005 //*
000006 //STEP001 EXEC PGM=IDCAMS
000007 //SYSPRINT DD SYSOUT=*
000008 //SYSIN DD *
000009 DEFINE CLUSTER(NAME(IMS810.INVENDB)
000010 TRACKS(1,1)
000011 NONINDEXED
000012 RECORDSIZE(2041,2041)
000013 VOLUMES(USER01)
000014 CONTROLINTERVALSIZE(2048))
000015 //
```

- O processamento exige *hardware* específico, *softwares* e desenvolvedores com habilidades analíticas específicas.
- Aumento pela demanda por *hardware* de baixo custo e *softwares* que pudessem ser desenvolvidos por programadores com habilidades de programação convencional.







Large Hydron Collider (LHC) da CERN gera 15 PB por ano



# A internet em um minuto

## 2020 *This Is What Happens In An Internet Minute*



# A internet em um minuto

## 2019 This Is What Happens In An Internet Minute



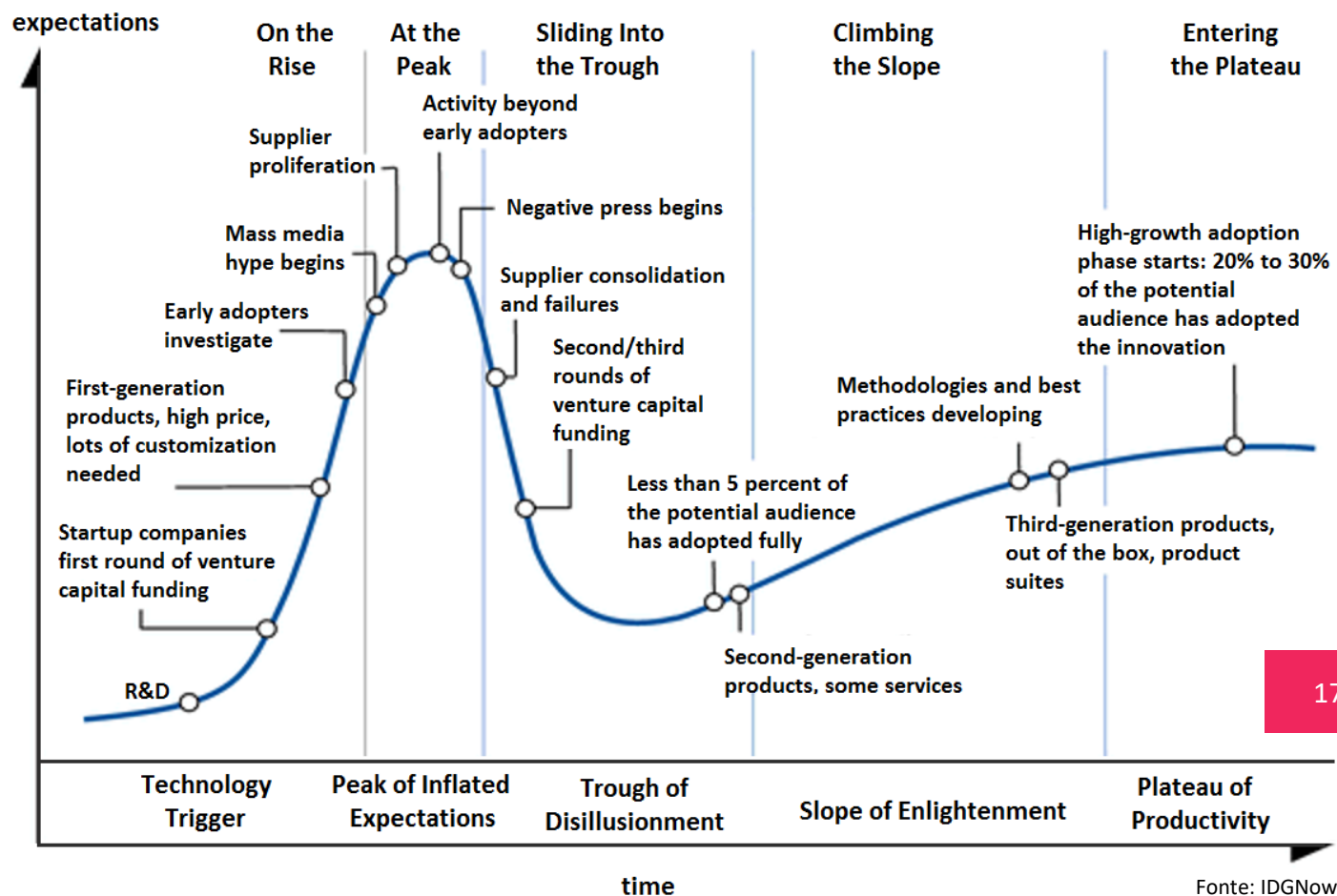
## 2020 This Is What Happens In An Internet Minute



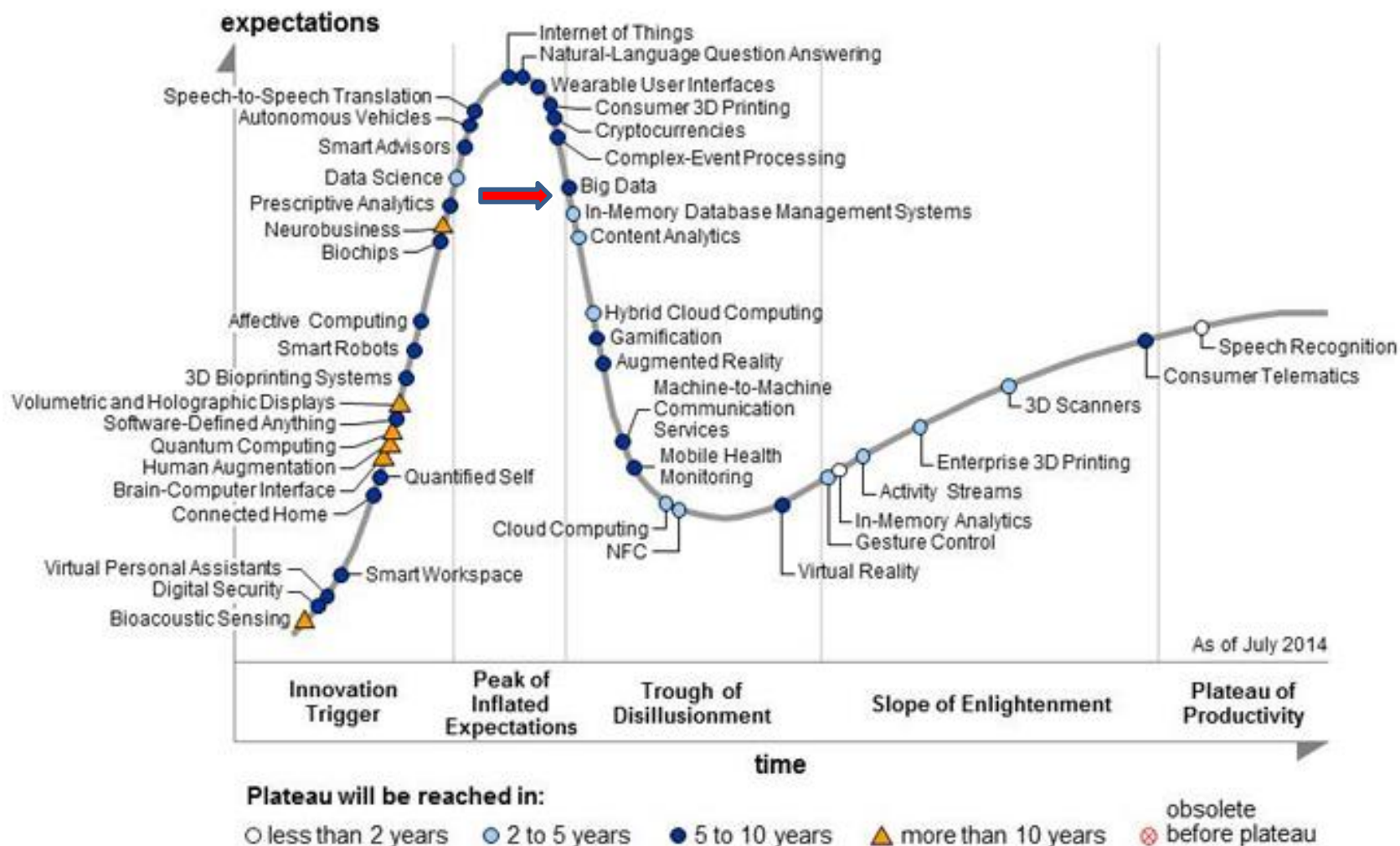




# Hype Cycle para tecnologias emergentes



# Hype Cycle 2014



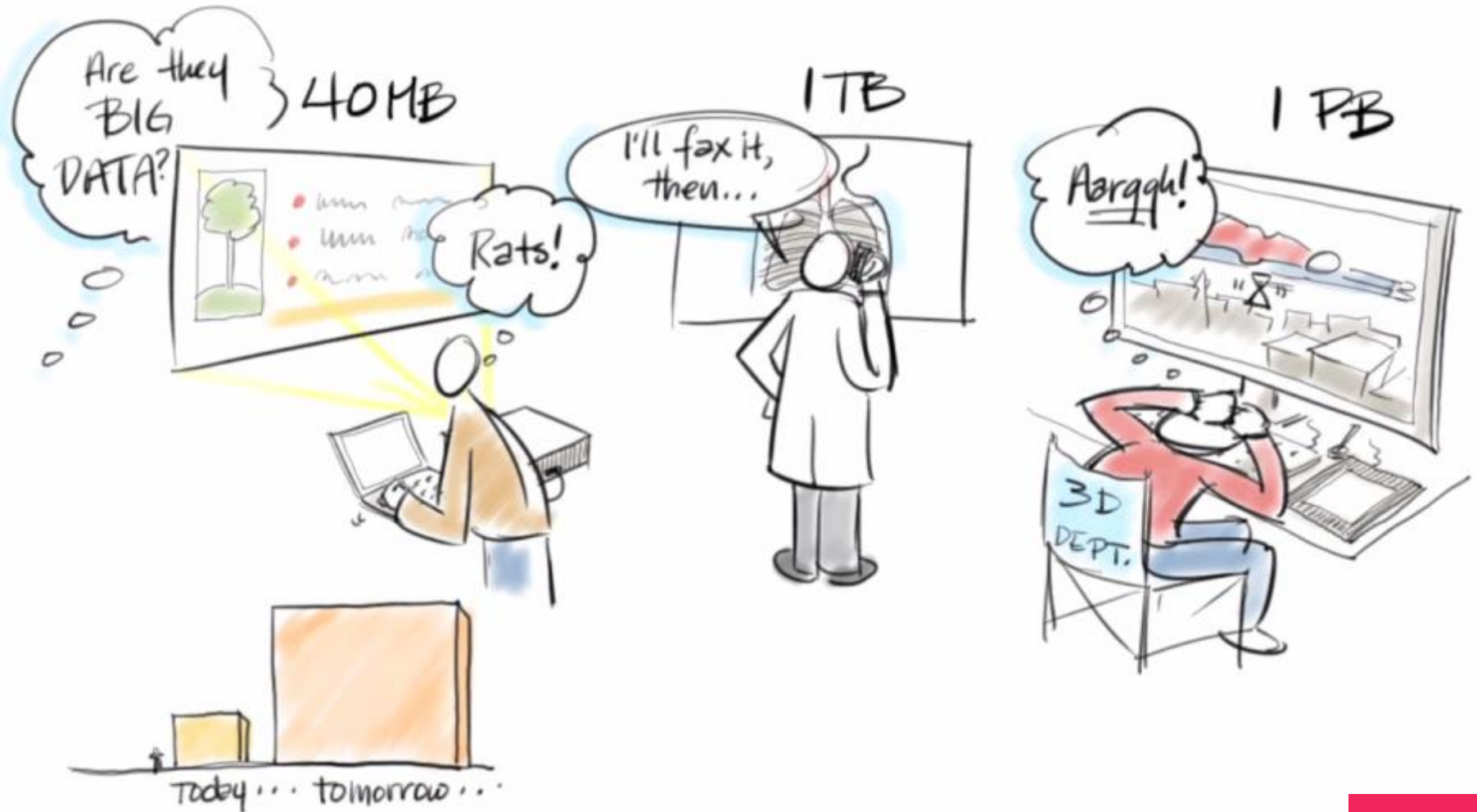
# Hype Cycle 2015

Figure 1. Hype Cycle for Emerging Technologies, 2015

Onde está Big Data?

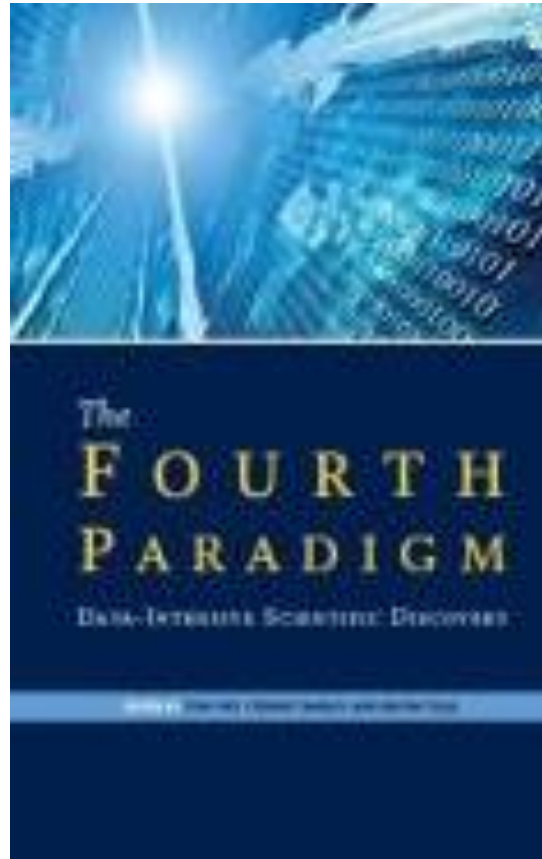


# O que é Big Data



# Dilúvio de Dados

## O Quarto Paradigma



- Há milhares de anos
  - a ciência nasceu de forma empírica, descrevendo fenômenos naturais
- Nos últimos séculos
  - passou a incorporar uma importante componente teórica, utilizando modelos e generalizações.
- Nas últimas décadas
  - surgiu uma forte tendência computacional, com a possibilidade de realização de sofisticadas simulações de fenômenos complexos.
- Nos últimos anos
  - surgiu um quarto paradigma, a exploração de grandes quantidades de dados, que unifica teoria, experimentos e simulação, ao mesmo tempo em que lida com uma quantidade enorme de informação.
- (Jim Gray, 2007)
- [http://research.microsoft.com/en-us/collaboration/fourthparadigm/4th\\_paradigm\\_book\\_part4\\_lynch.pdf](http://research.microsoft.com/en-us/collaboration/fourthparadigm/4th_paradigm_book_part4_lynch.pdf)





# O que o Starbucks sabe sobre os hábitos dos clientes.

FIAP 22

- 43% das pessoas que tomam chá, não usam açúcar.
- 25% das pessoas acrescentam leite ao café, depois que ele esfriou.
- 4.000.000.000 de xícaras de café servidas ao ano.
- 87.000 combinações de consumo possíveis em todo mundo.
- Aplicativo com 6.000.000 de clientes cadastrados e fidelizados.
- <https://www.cnbc.com/2016/04/06/big-data-starbucks-knows-how-you-like-your-coffee.html>



# O que o Walmart sabe sobre os hábitos dos clientes.



FIAP

- “Começar a prever o que vai acontecer, em vez de esperar para que as coisas aconteçam”
- “Não sabíamos, por exemplo, que as vendas de Pop-Tarts sabor morango aumentavam até sete vezes as taxas normais de vendas antes de um furacão”
- “E o item mais vendido antes da passagem de um furacão era cerveja.”
- [http://www.nytimes.com/2004/11/14/business/yourmoney/what-walmart-knows-about-customers-habits.html?\\_r=0](http://www.nytimes.com/2004/11/14/business/yourmoney/what-walmart-knows-about-customers-habits.html?_r=0)
- (Linda M. Dillman, 2004)

# O que o Pão de Açúcar sabe sobre os hábitos dos clientes.

FIAP

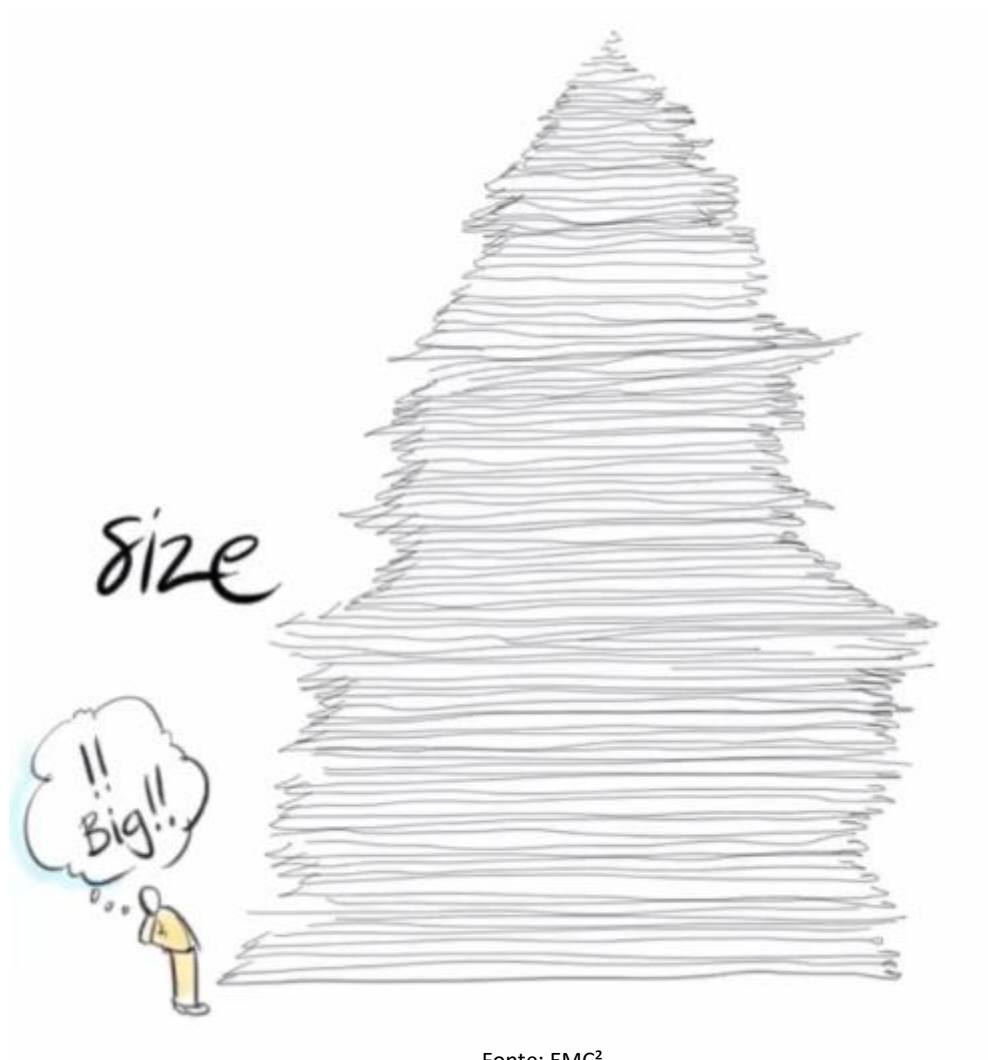
- Se a pessoa começa a comprar fraldas os fornecedores são informados e passam a oferecer papinhas e leites especiais.
- Se o cliente não compra na rede há algum tempo, ofertas especiais exclusivas para cada cliente.
- Produtos enviados para as lojas conforme o perfil da região.



[https://exame.abril.com.br/tecnologia/sistema-do-grupo-pao-de-acucar-usa-big-data-para-fidelizar-clientes/?utm\\_source=blog&utm\\_campaign=rc\\_blogpost](https://exame.abril.com.br/tecnologia/sistema-do-grupo-pao-de-acucar-usa-big-data-para-fidelizar-clientes/?utm_source=blog&utm_campaign=rc_blogpost)

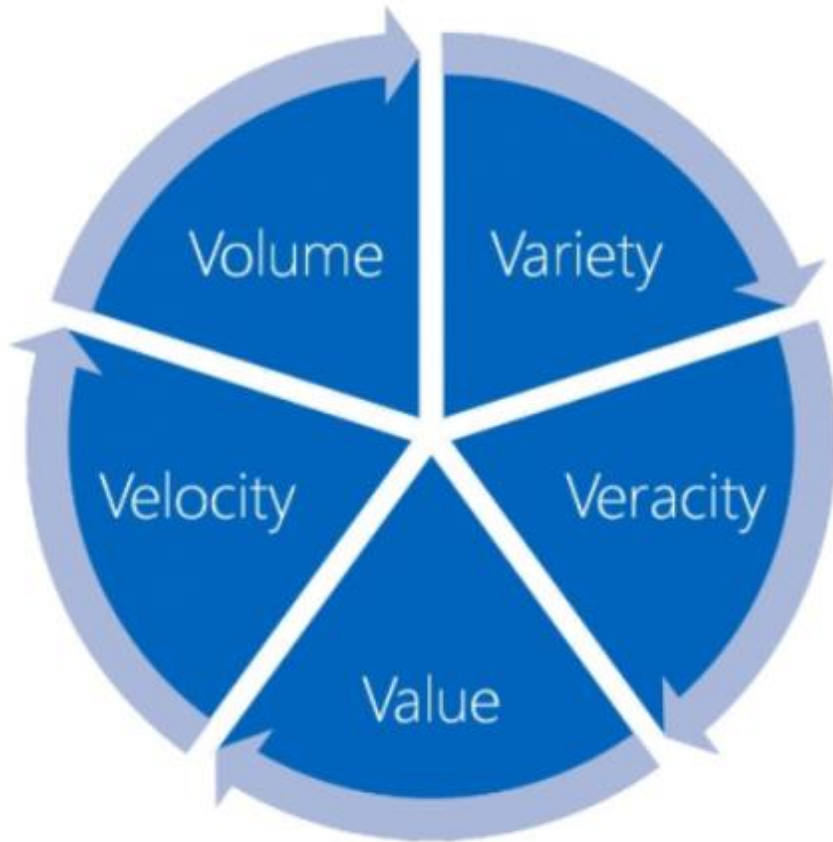


# O que é Big Data



Fonte: EMC<sup>2</sup>

# Os V's



- **Volume:** Grandes de dados, desafios de armazenamento
- **Variedade:** Diferentes formatos de dados estruturados e/ou não
- **Veracidade:** Acurácia e autenticidade
- **Valor:** Retorno desses dados para o negócio/sociedade
- **Velocidade:** Tempo entre o dado gerado e analisado

# O que é Big Data

Research Report

## **Big Data: The Next Frontier for Innovation**

McKinsey & Company

FIAP

- Segundo o McKinsey Global Institute [2011], *Big Data* refere-se:
  - “ao conjunto de dados cujo tamanho está além da capacidade de análise, captura, armazenamento e gerenciamento de uma ferramenta desenvolvida por um software típico”

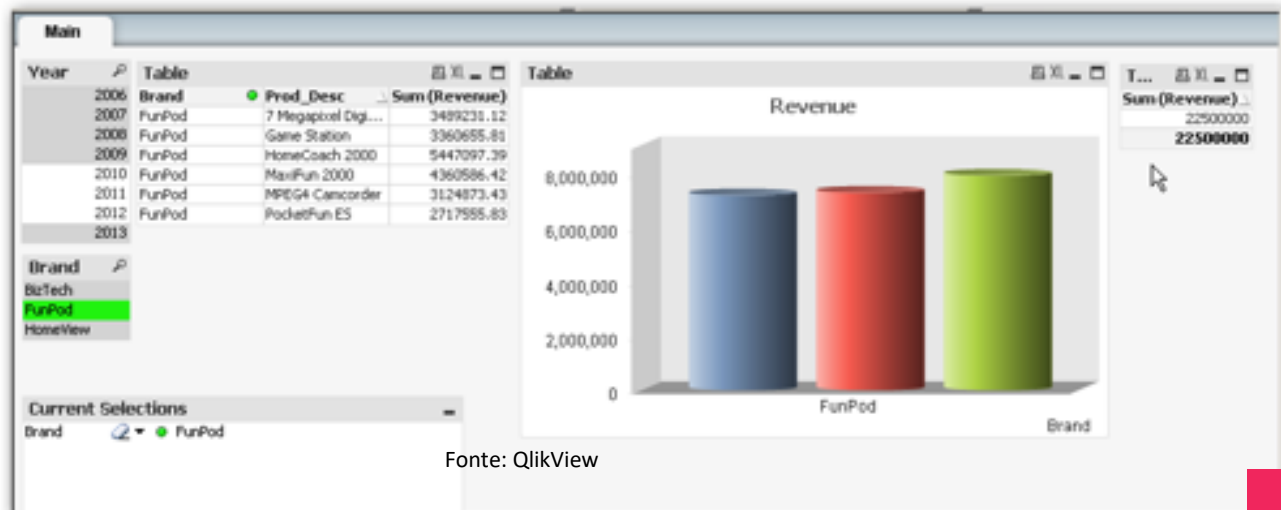
# O que é *Big Data*

- Fonte: EMC<sup>2</sup>



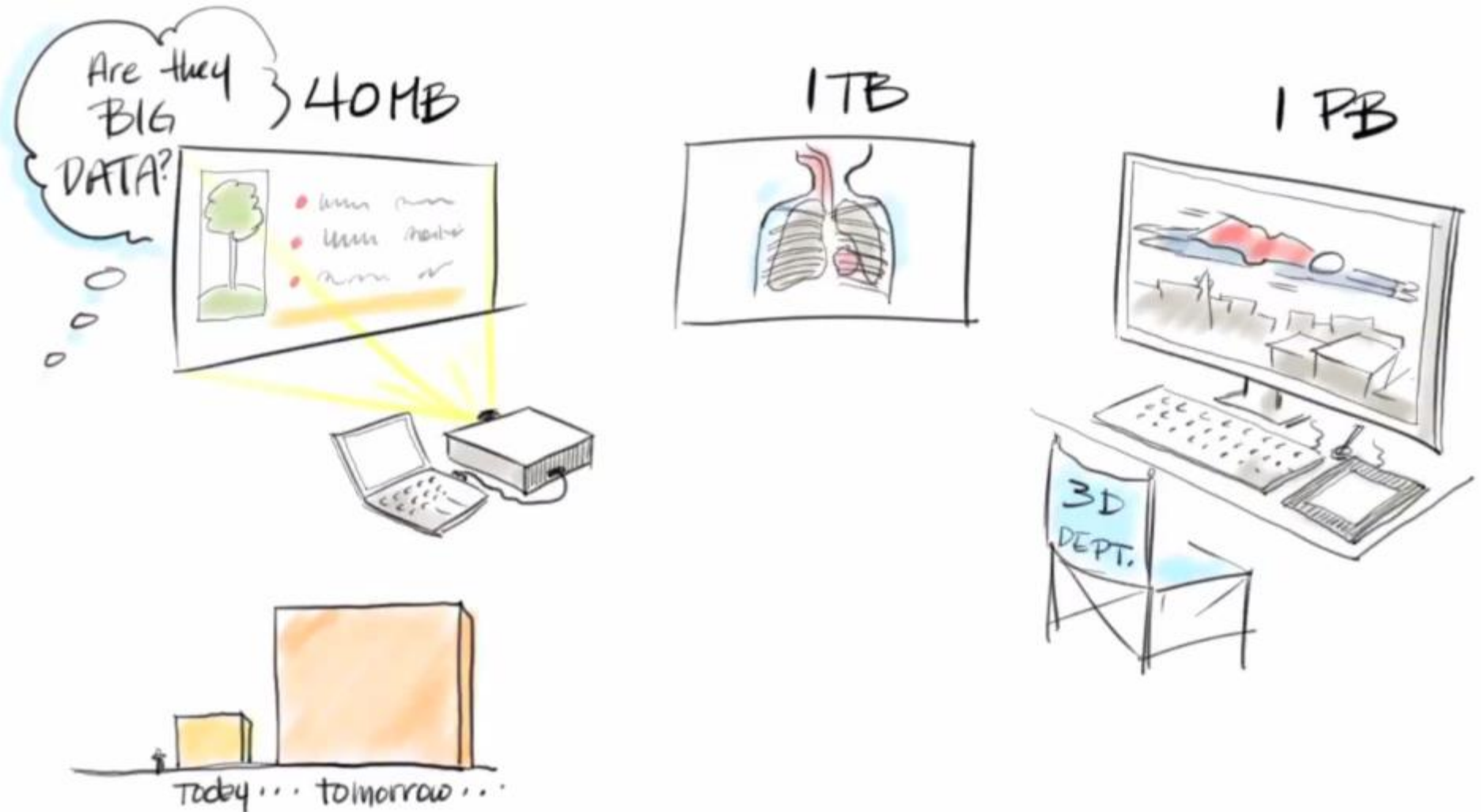
# O que é *Big Data*

- *Big Data* não deve ser definido em número de *terabytes* ou *petabytes*. A quantidade de bytes pode mudar dependendo:
  - do tipo de dados
  - do setor onde a tecnologia está sendo aplicada ou
  - da própria evolução da tecnologia.



# O que é Big Data

FIAP



# O que é *Big Data*

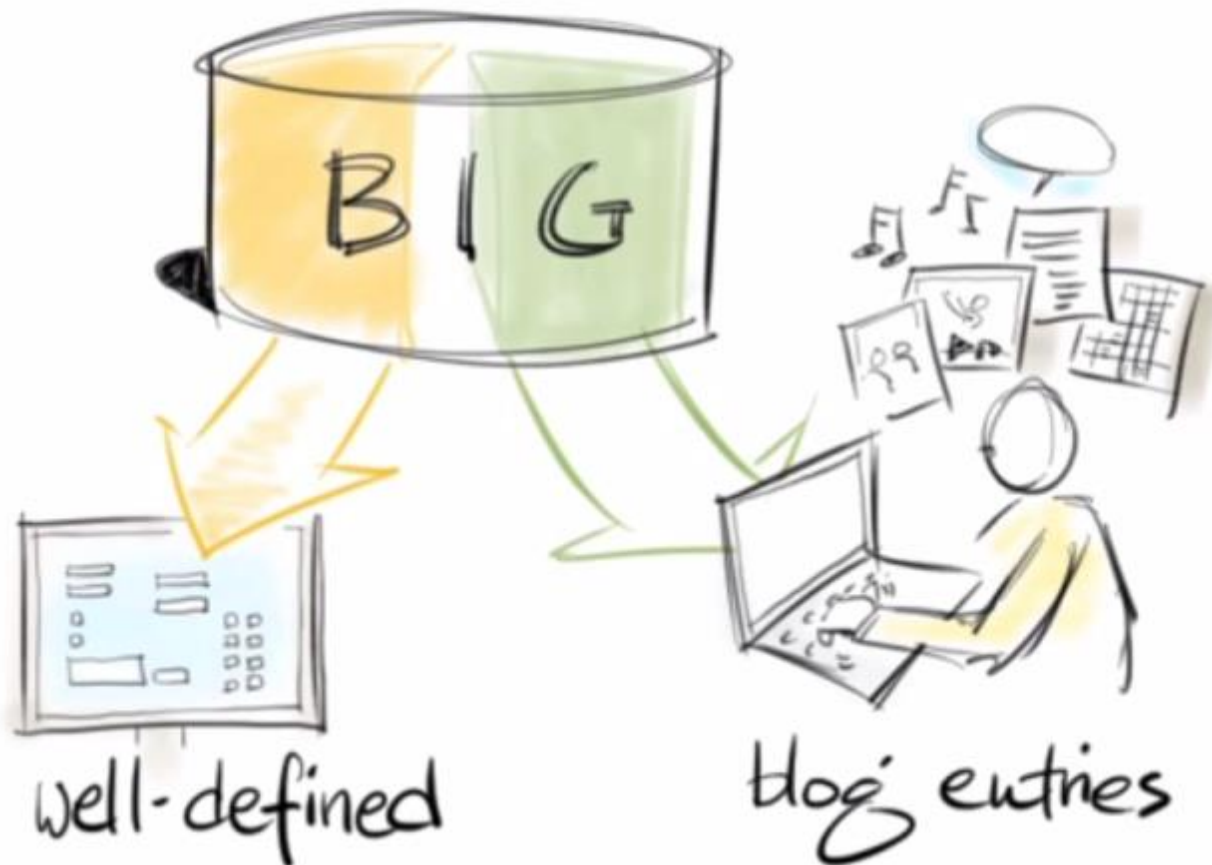
- O termo *Big Data* implica no uso de metodologias e ferramentas para processamento e análise de dados que possam produzir resultados úteis que não possam ser deduzidas/calculadas, de maneira eficiente, através de outros métodos.



# O que é *Big Data*

FIAP

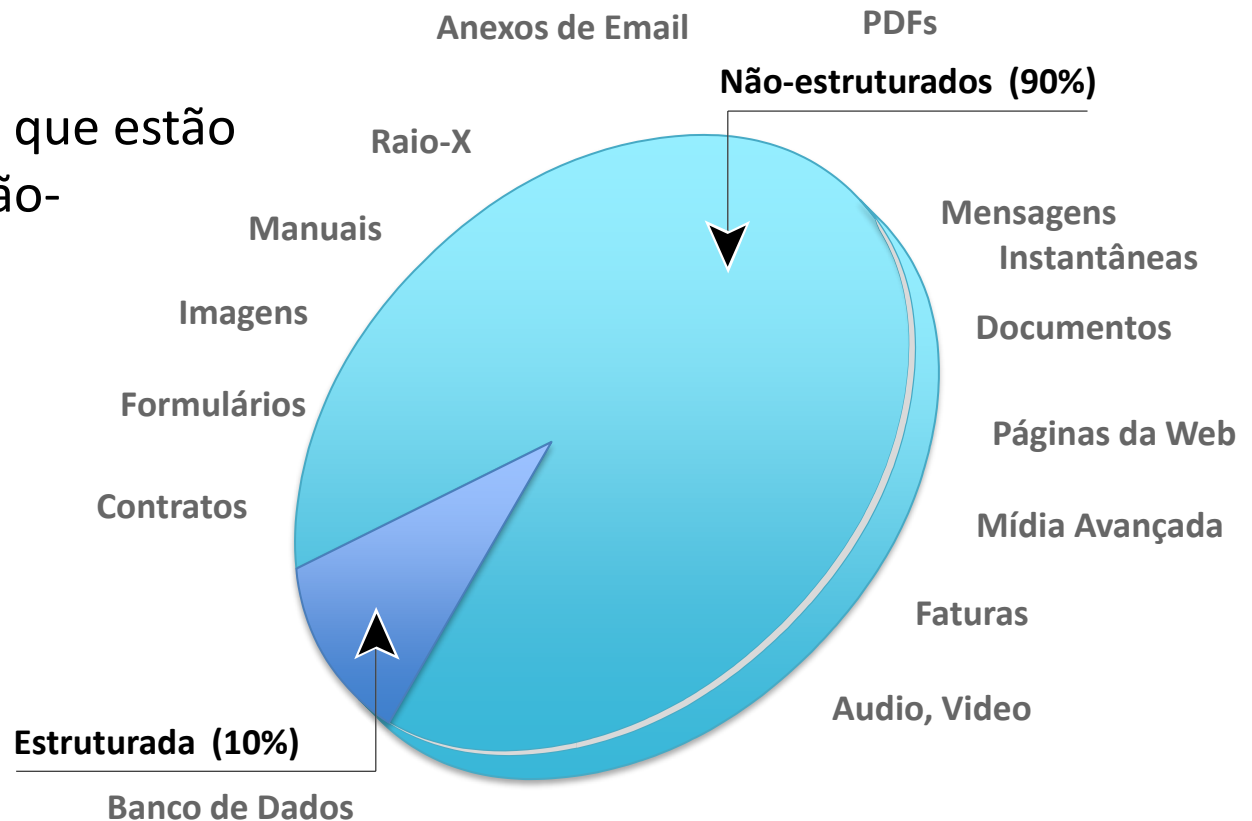
Not all BIG DATA  
are the same  
in **STRUCTURE** ~



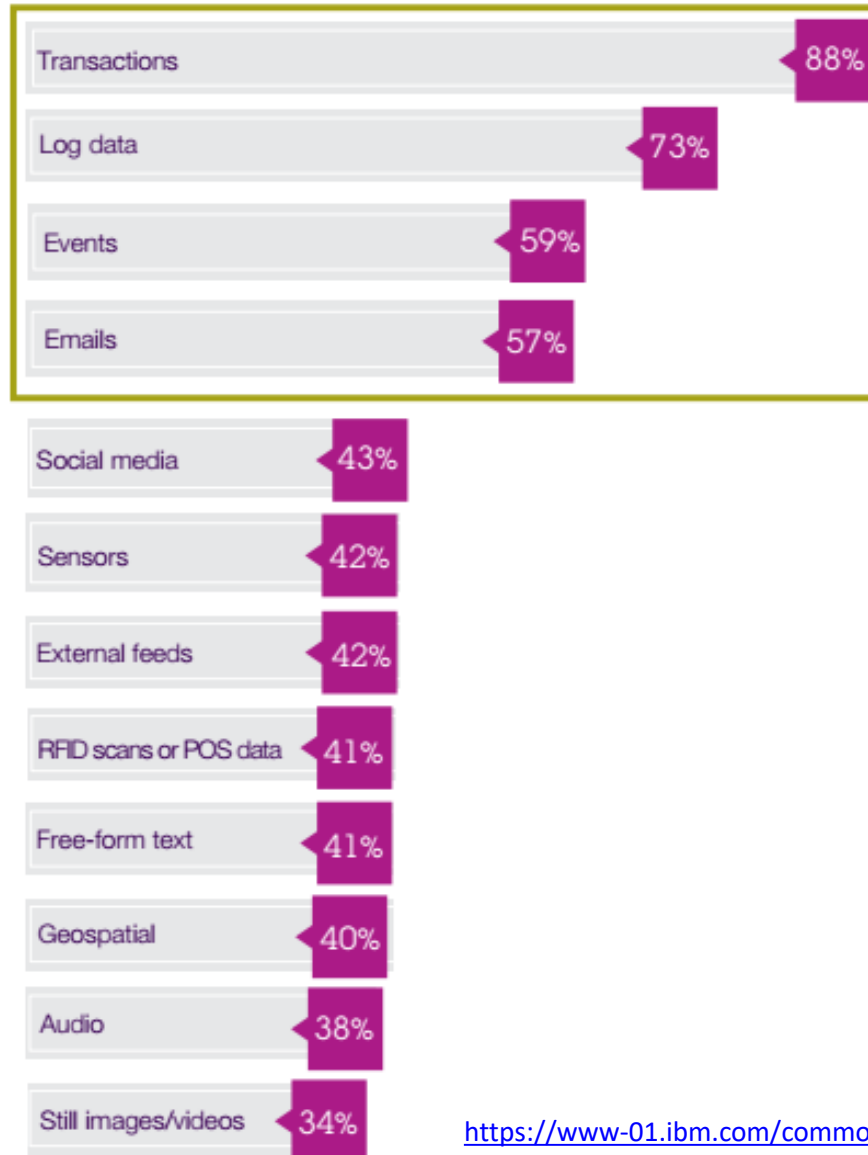


# Tipos de dados

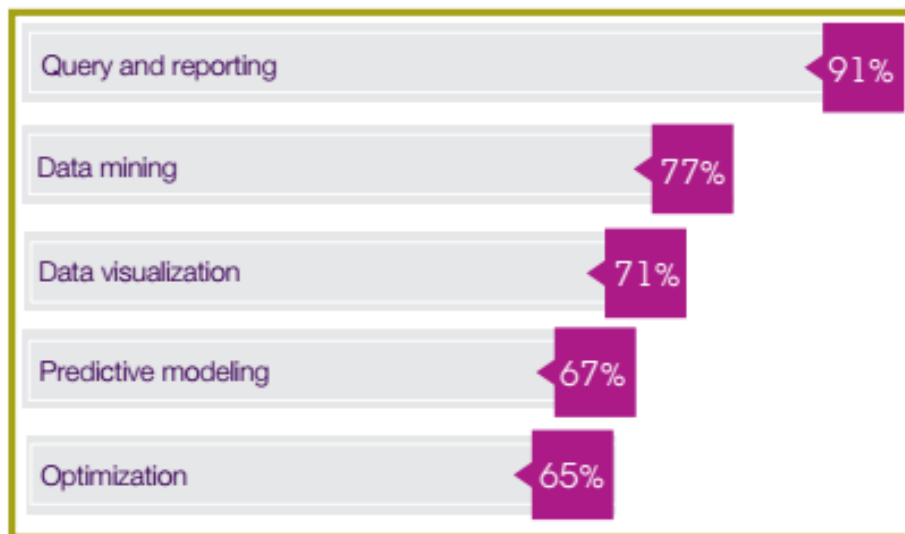
- Os dados podem ser classificados como:
  - Estruturado
  - Não-estruturado
- A maioria dos dados que estão sendo criados são não-estruturados



# De onde vem os dados?



# E para onde eles vão?



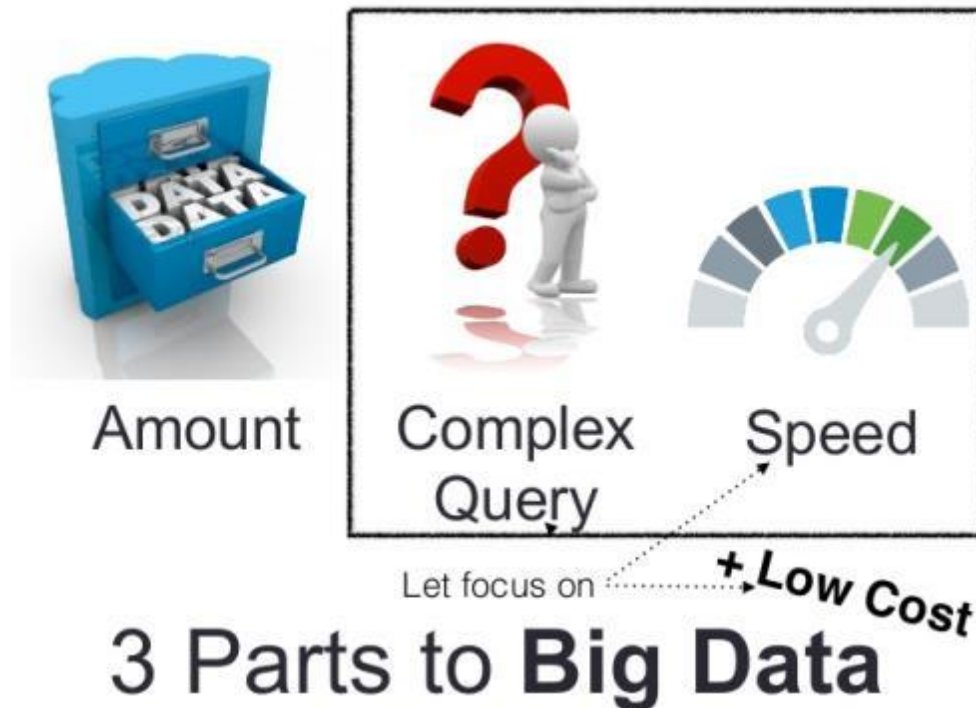
# Google Trends



Fonte: Google Trends

# Declaração de problema

- Como processar *Big Data* usando o estado-da-arte da tecnologia atual sem “estourar” o limite de tempo e o orçamento?



Fonte: EMC<sup>2</sup>

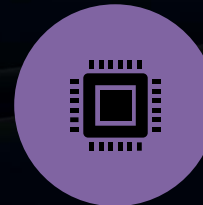
# Onde está o “gargalo”?



VELOCIDADE DAS CPUS  
ESTA CADA VEZ MAIOR  
MAS...



A VELOCIDADE DE ACESSO  
A DISCO, OU VOLUMES DE  
DISCOS AINDA É LENTA.



O AUMENTO DA  
VELOCIDADE DE CPU NÃO  
BENEFICIA MUITO OS  
PROGRAMAS QUE TEM  
NECESSIDADE DE ACESSAR  
GRANDES VOLUMES DE  
DADOS.

# Teorema CAP

- Em qualquer sistema distribuído *stateful* é preciso escolher entre
  - Consistency (consistência forte). Todos os nós veem os mesmos dados ao mesmo tempo.
  - Availability (alta disponibilidade). Toda solicitação recebe uma resposta, seja ela bem-sucedida ou não.
  - Network Partition Tolerance (tolerância a particionamento dos dados na rede). O sistema continua funcionando mesmo que mensagens sejam perdidas ou parte do sistema falhe.
- Entre as três propriedades, somente duas podem ser garantidas ao mesmo tempo.

(Eric Brewer, 2000)

<http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf>



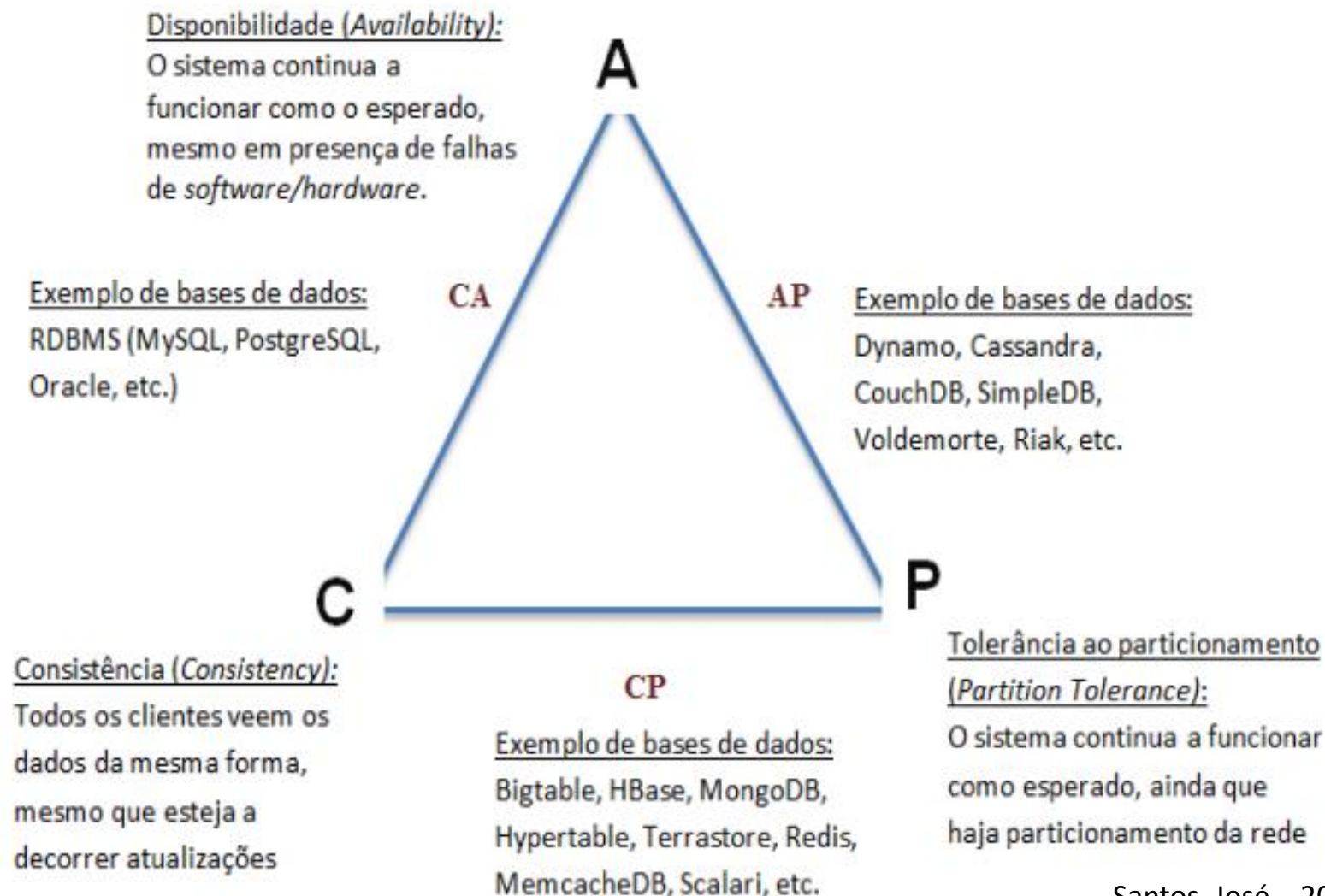
# Teorema CAP

- **CAp** – Bases de dados tradicionais relacionais.
  - Em caso de falha da rede ou de grande latência, elas não conseguem responder a todos os pedidos.
- **caP** – Bases de dados NOSQL.
  - Estes são sistemas altamente tolerantes a falhas, desde que existam um grande número de servidores suportando o sistema.
  - Fornecem disponibilidade mas consistência eventual.
- **CaP** – O sistema não dá garantias de sempre estar disponível.
  - É um sistema onde um nó falha e os outros não podem responder as solicitações.





# CAP



Santos, José – 2014

# ACID versus BASE

- A maioria dos DBMS usam o paradigma ACID
  - Atomicidade: A transação será executada totalmente ou não será executada.
  - Consistência: Garante que o banco de dados passará de uma forma consistente para outra forma consistente.
  - Isolamento: Garante que a transação não será interferida por nenhuma outra transação concorrente.
  - Durabilidade: Garante que o que foi salvo, não será mais perdido.



# ACID versus BASE

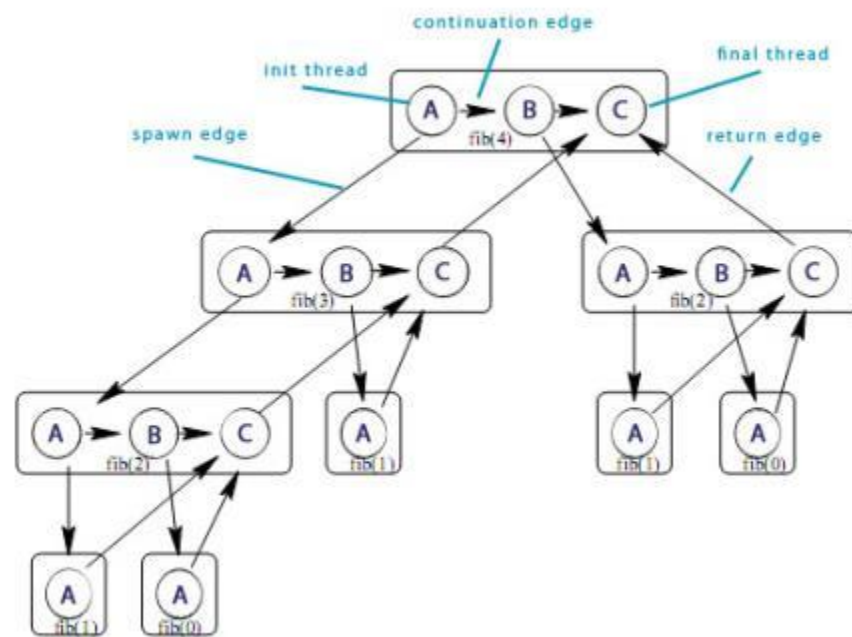
- Propriedades BASE:
  - Basically Available — Basicamente Disponível.
  - Soft-State - Estado Leve
  - Eventually Consistent — Eventualmente Consistente.
- Uma aplicação funciona basicamente todo o tempo (Basicamente Disponível), não tem de ser consistente todo o tempo (Estado Leve) e o sistema torna-se consistente no momento devido (Eventualmente Consistente).

# ACID versus BASE

ACID	BASE
Consistência forte	Fraca consistencia
Isolamento	Foco em Disponibilidade
Transações aninhadas	Respostas aproximadas
Disponibilidade	Mais simples e mais rápido
Conservador (pessimista)	Agressivo (otimista)
Dados complexos, de alta qualidade e alta densidade	Dados simples, de baixa qualidade e baixa densidade
Relacionamentos complexos	Relacionamentos simples
Joins	Evita joins
Schema-centric	Schema-free, desestruturados ou semi-estruturados
Projetado para escalonamento para cima, não para fora	Armazenamento e processamento distribuído
Padrões bem definidos	Os padrões ainda não estão bem definidas
Database-centric	Application-centric e developer-centric

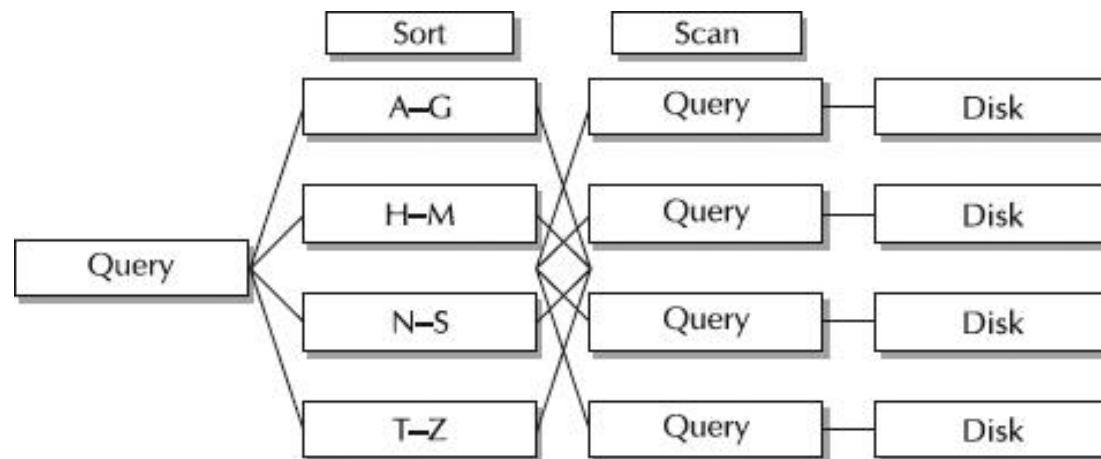
# Procurando soluções

- As linguagens de programação modernas suportam conceitos de multiprogramação, tais como *multi-threading* implementados como bibliotecas (por exemplo, pthreads POSIX) para C++ ou *buit-in* em Java.
- Os modelos de programação paralela podem ser utilizados para reduzir o tempo de processamento ou manipulação de grandes volumes de dados.



# Operações em Paralelo

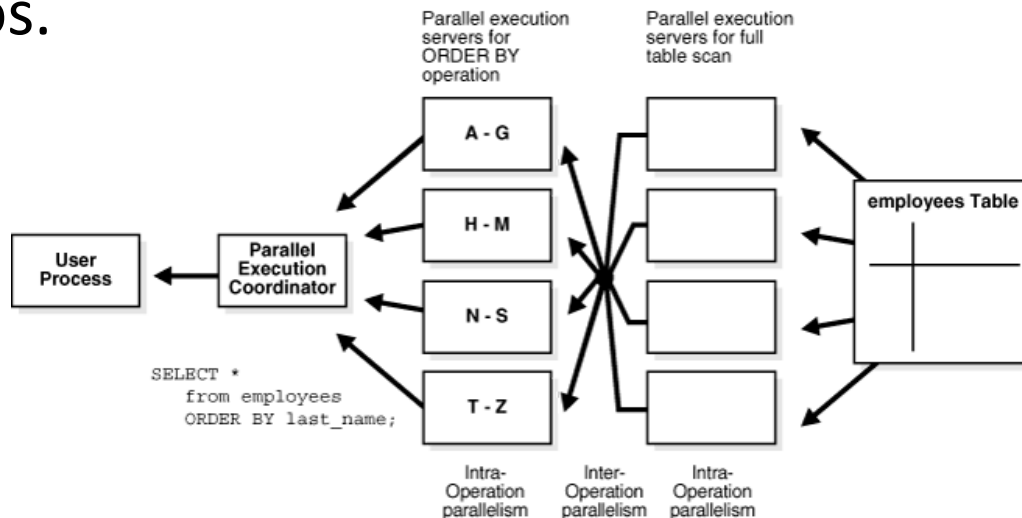
- Operações não sequenciais são quebradas em múltiplos *chunks* que são executadas simultaneamente em várias CPUs da mesma máquina.
- O programa deve ser modelado para tirar proveito do processamento em paralelo.



Fonte: Oracle

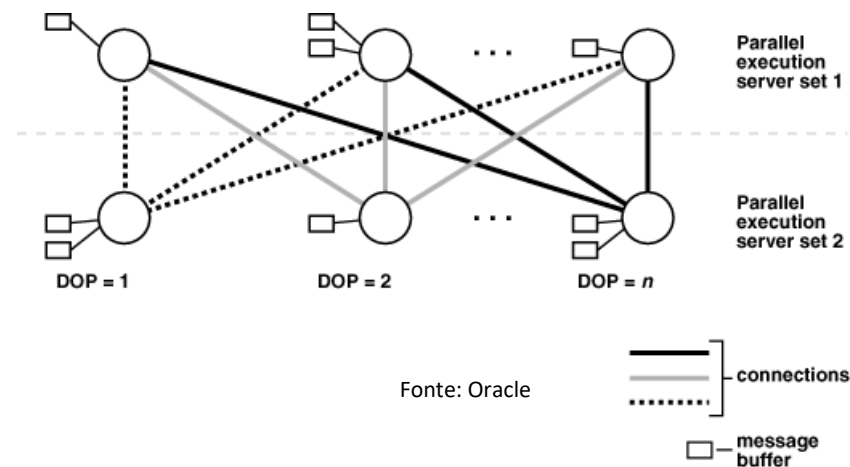
# Operações em Paralelo

- Operações em paralelo aumentam a complexidade do código.
  - Coordenação de tarefas concorrentes: como prevenir que tarefas concorrentes interfiram umas nas outras.
  - Paralelização de algoritmos: nem todo algoritmo pode ser paralelizado.
  - Sincronização e bloqueio de memória compartilhada: protegida por *locks* ou semáforos para evitar a corrupção de dados.



Fonte: Oracle

- SQL padrão é a forma mais comum de manipulação de dados transacionais.
- Há anos os SGBD Oracle, DB2 e SQLServer permitem executar operações de consulta e carga em paralelo.
- Entretanto as consultas paralelas estão restritas ao mesmo nó.

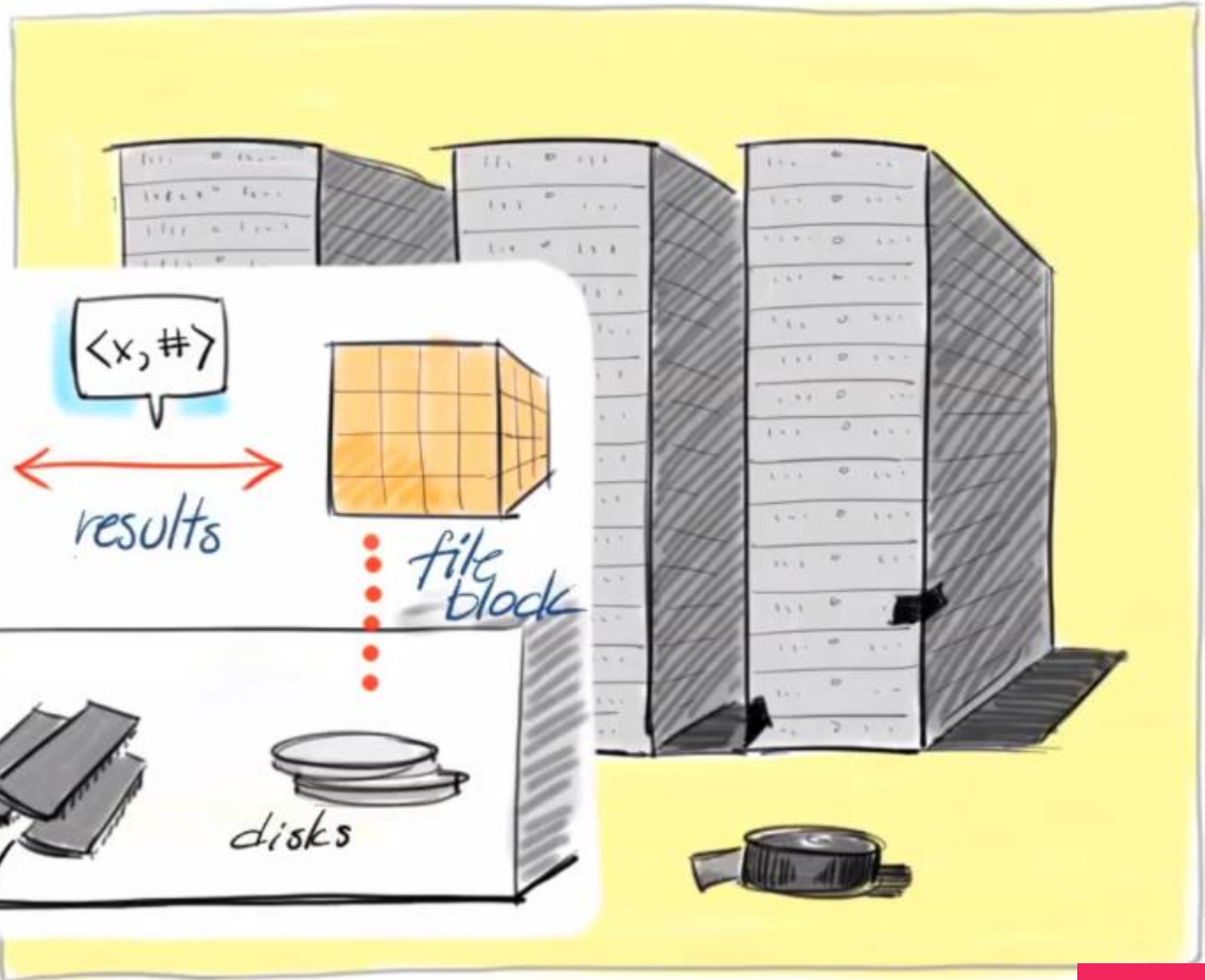




# MapReduce

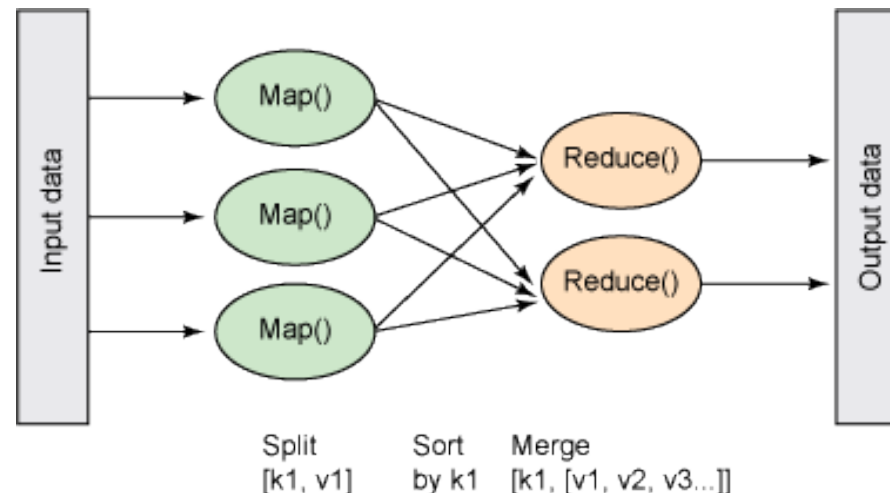
FIAP

Hadoop  
MapReduce



# MapReduce

- Segundo a IBM [2009], MapReduce é um
  - “Modelo de programação que permite o processamento de dados massivos em um algoritmo paralelo e distribuído, geralmente em um cluster de computadores.”
- MapReduce é baseado nas operações de Map e Reduce de linguagens funcionais como o LISP.



Fonte: IBM

# MapReduce

51

- Trata os dados como um conjunto de pares *<Key, Value>*. (chave/valor)
- As operações de entrada leem os dados e geram os pares *<Key, Value>*
- O usuário fornece duas funções Map e Reduce, que são chamadas em tempo de execução.

```
1 map(Tuple tuple):  
2     EmitIntermediate(tuple);  
3  
4 reduce(Tuple groupTuple, Iterator tuples):  
5     int count = 0;  
6     for each tuple in tuples:  
7         count += tuple.get("visits");  
8     Emit(NowTuple(tuple.get("url"), tuple.get("date"), count));
```

# MapReduce

- *Map:*
  - Obtêm uma lista de pares  $\langle \text{Key}, \text{Value} \rangle$ , processa os pares e gera um conjunto de pares  $\langle \text{Key}, \text{Value} \rangle$  intermediário.
  - Repassa o valor intermediário para a função Reduce.
  - Cada par é processado em paralelo.



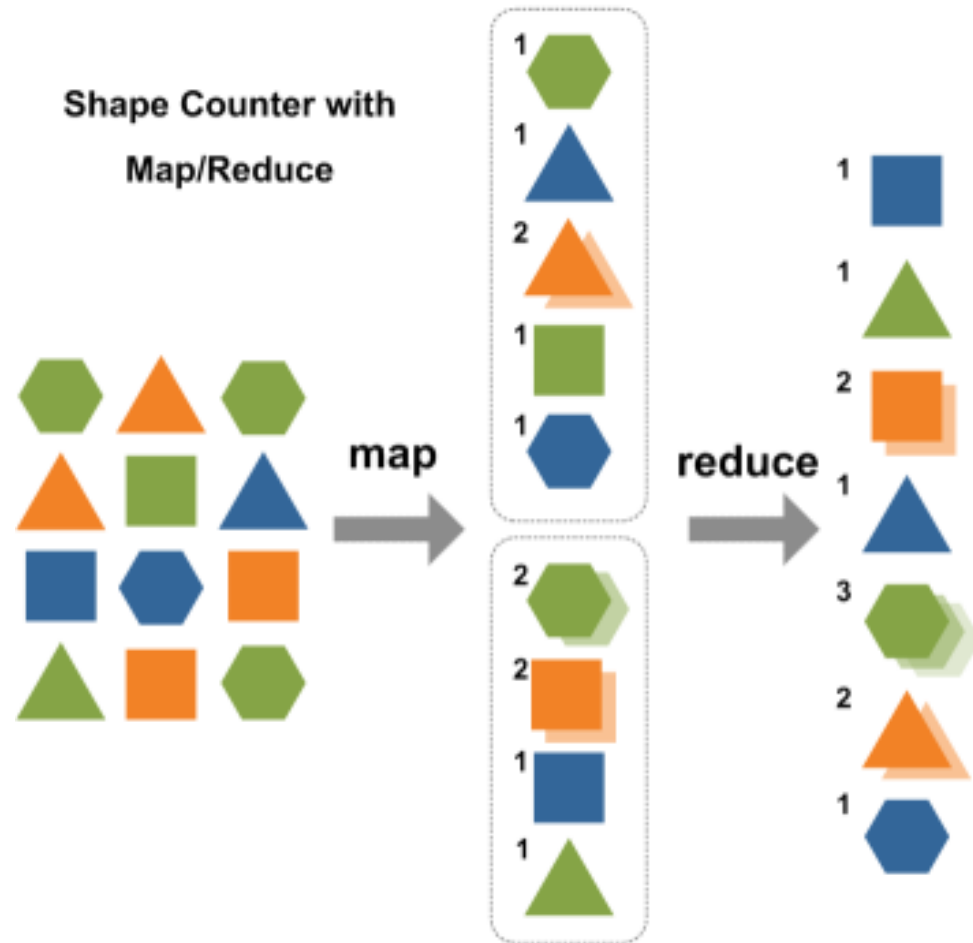
# MapReduce

53

- Reduce:
  - Processa todos os valores associados com a mesma *<Key>*.
  - Mescla os valores para formar um conjunto de valores possivelmente menor.
  - Geralmente, apenas um valor de saída de 0 ou 1 é produzido a cada chamada Reduce.
  - Os valores intermediários são fornecidos à função Reduce do usuário por um iterador permitindo identificar listas de valores que são grandes demais para a memória.



# MapReduce




Fonte: Hadoop

# Exemplo: Contador de Palavras FIAP

- Considere o exemplo do problema proposto pela IBM:
  - Será contado o número de ocorrências de cada palavra em um grande conjunto de documentos.


This is probably the most common error that people make with any sort of data access technology. "Just give me the whole thing, I'll make sense of it on the client side." In the RDBMS world, this would be expressed as "SELECT \* FROM Orders". The problem with such queries is they can potentially...

**Map**



This	1
is	1
probably	1
the	1
most	1
common	1
error	1

**Reduce**



this	2
is	2
probably	1
the	4
most	1
common	1
error	1

# Exemplo: Contador de Palavras

- A função Map emite cada palavra mais uma contagem associada de ocorrências.

```
mapper (filename, file-contents):  
    for each word in file-contents:  
        emit (word, 1)
```



# Exemplo: Contador de Palavras

- A função Reduce soma todas as contagens emitidas para uma palavra específica.

```
reducer (word, values):  
    sum = 0  
    for each value in values:  
        sum = sum + value  
    emit (word, sum)
```

# Exemplo: Contador de Palavras

Incluindo as palavras dog, cat, mouse e hippo em um banco orientado a documentos.

```
db.items.insert({tags: ['dog', 'cat']})  
db.items.insert({tags: ['dog']})  
db.items.insert({tags: ['dog', 'mouse']})  
db.items.insert({tags: ['dog', 'mouse', 'hippo']})  
db.items.insert({tags: ['dog', 'mouse', 'hippo']})  
db.items.insert({tags: ['dog', 'hippo']})
```

# Exemplo: Contador de Palavras

Criando o mapeamento

```
var map = function() {  
  this.tags.forEach(function(t) {  
    emit(t, {count: 1});  
  });  
}
```

# Exemplo: Contador de Palavras

Criando o reduce

```
var reduce = function(key, values) {  
  var count = 0;  
  for(var i=0, len=values.length; i<len; i++) {  
    count += values[i].count;  
  }  
  return {count: count};  
}
```

# Exemplo: Contador de Palavras

Executando o MapReduce

```
> db[result.result].find()
```

```
{ "_id" : "cat",    "value" : { "count" : 1 } }  
{ "_id" : "dog",    "value" : { "count" : 6 } }  
{ "_id" : "hippo",  "value" : { "count" : 3 } }  
{ "_id" : "mouse",  "value" : { "count" : 3 } }
```

# MapReduce – Prós e Contras

## Prós

Modelo conceitual fácil de entender: apenas duas operações

Automatiza a distribuição de dados e agregação de resultados.

Pode ser incorporado por várias linguagens procedurais.

Não é preciso preocupar-se com os detalhes da paralelização e recuperação de falhas.

Elimina locks.

## Contras

Não é simples implementar em linguagens de programação convencionais, como Java, C++ e Python.

O programador deve codificar cada operação de acesso a dados.

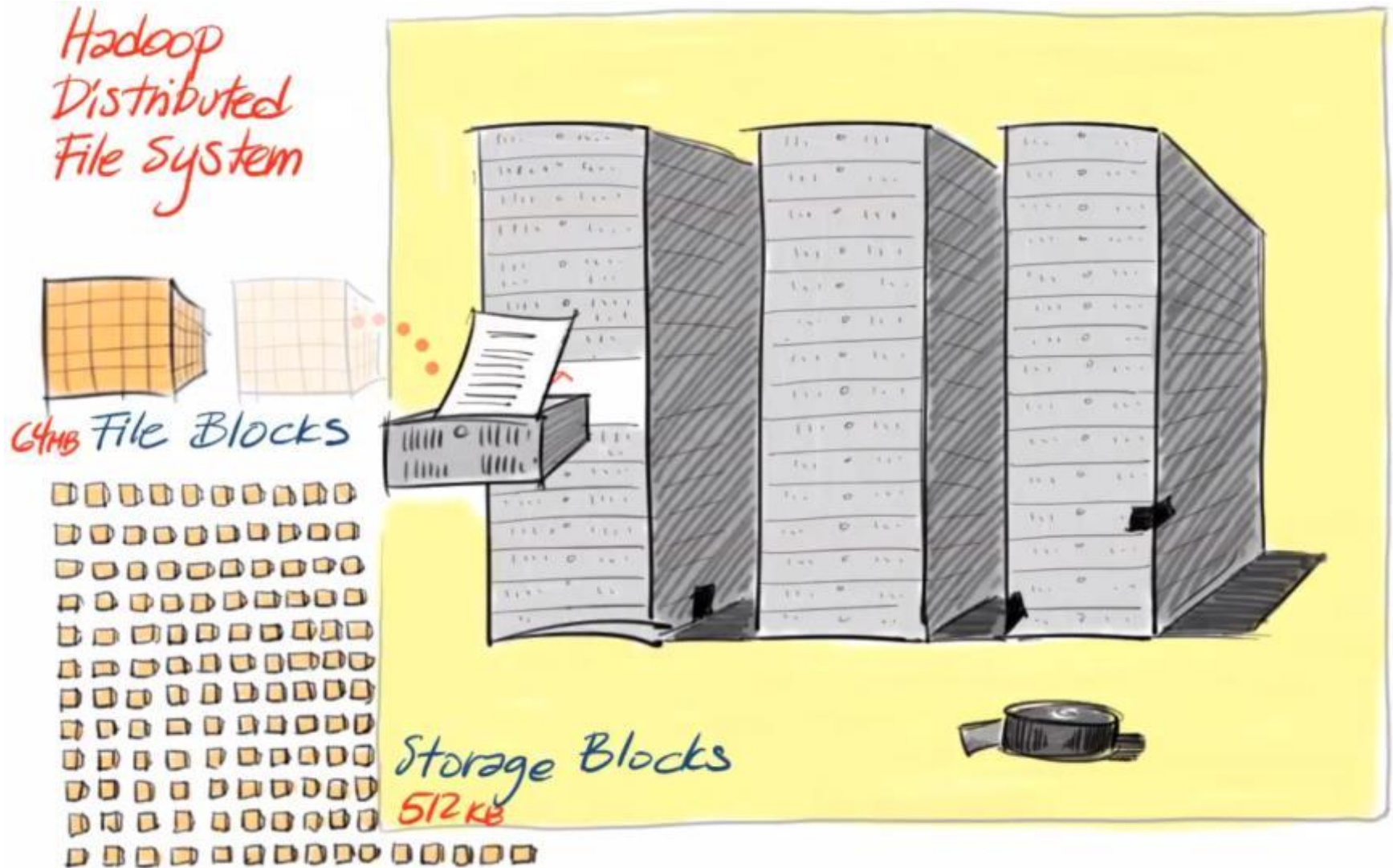
A natureza “opaca” das funções Map e Reduce impedem sua otimização.





# HDFS

FIAP



# HDFS

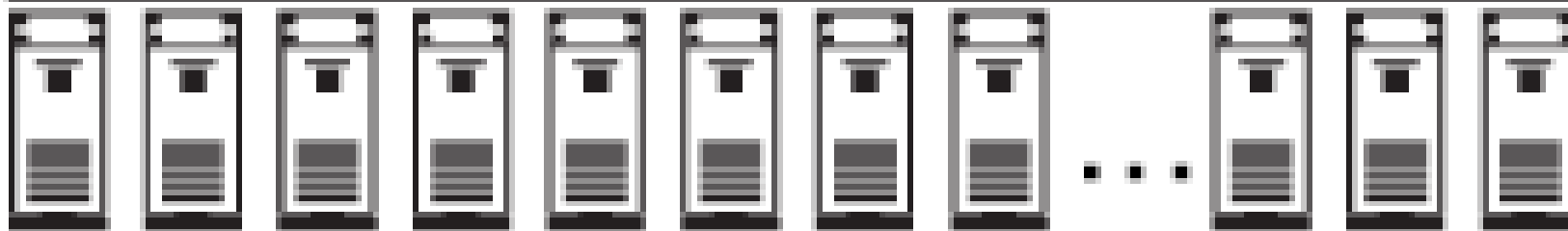
- O *Hadoop Distributed File System* (HDFS) é um sistema de arquivos altamente tolerante a falhas projetado para executar em hardware padrão de baixo custo.
- O HDFS disponibiliza acesso de alto rendimento para os dados do aplicativo e é adequado para aplicativos com grandes conjuntos de dados.

Reduce

Shuffle/sort mapper output

Mapper - read 64+ MB blocks

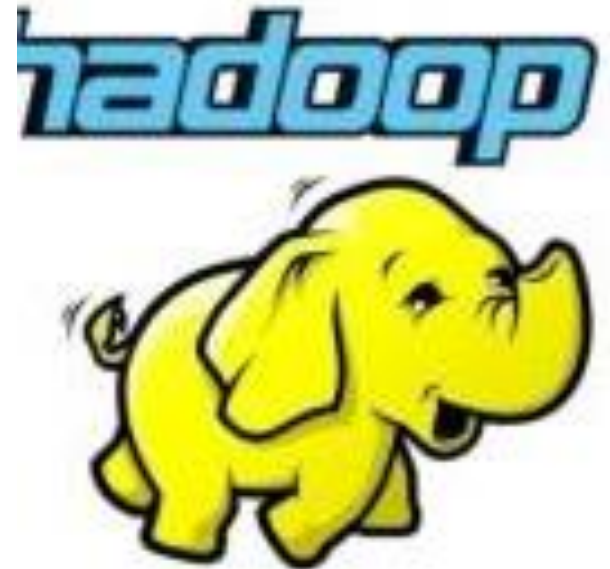
HDFS



# Hadoop

---

- Segundo a Hadoop, “Hadoop é um *storage* confiável e um sistema analítico” [2014]
- Composto por duas partes essenciais:
  - o Hadoop Distributed Filesystem (HDFS), sistema de arquivos distribuído e confiável, responsável pelo armazenamento dos dados
  - Hadoop MapReduce, responsável pela análise e processamento dos dados.
- O nome do projeto veio do elefante de pelúcia que pertencia ao filho do criador, Doug Cutting.



# Pig e Pig Latin

---

- Pig é um mecanismo para executar os fluxos de dados de modo paralelo ao Hadoop.
- Usa uma linguagem chamada Pig Latin para expressar esses fluxos de dados.
- Com a Pig Latin, é possível descrever como os dados de uma ou mais entradas devem ser lidos, processados e, depois, armazenados em uma ou mais saídas de modo paralelo.



# Pig e Pig Latin

---

- Um programa de Pig Latin consiste de uma série de operações ou transformações que é aplicada aos dados de entrada a fim de produzir a saída.





# Pig e Pig Latin

---

- Exemplo de programa para analisar os arquivos de log de um web site e encontrar as 10 páginas mais visitadas de cada categoria.



# Pig e Pig Latin

---

```
visits = load '/data/visits' as
(user, url, time);
gVisits = group visits by url;
visitCounts = foreach gVisits
generate url, count(urlVisits);
urlInfo = load '/data/urlInfo' as
(url, category, pRank);
visitCounts = join visitCounts by
url, urlInfo by url;
gCategories = group visitCounts by
category;
topUrls = foreach gCategories
generate top(visitCounts,10);
Store topUrls into '/data/topUrls';
```





# Hive e HiveQL

- Hive é *framework* para soluções de *Data Warehousing* executado no ambiente Hadoop.
- HiveQL é uma linguagem declarativa, similar ao SQL, usada para criar programas executáveis no Hive.
- O compilador HiveQL traduz os comando em HiveQL em *jobs* do MapReduce e os envia para o Hadoop executar.

# Hive e HiveQL



```
SELECT * FROM sales  
WHERE amount > 10 AND region = "US"
```

```
CREATE TABLE food  
            (id INT, msg STRING)  
PARTITIONED BY (dt STRING);
```

# Hive e HiveQL



```
FROM (  
    FROM docs  
    MAP text USING `python wc_map.py`  
    AS (word, count) CLUSTER BY word  
) words  
REDUCE word, count USING `python  
wc_reduce.py`
```

# O que é um Banco de Dados NoSQL?

- Armazenamento sem Schema
- Armazena dados não-relacionais
- Exemplos de Bancos de Dados NoSQL:
  - Oracle NoSQL
  - IBM Cloudant
  - Cassandra
  - Voldemort
  - MongoDB
  - BigTable
  - DynamoDB
- O termo NoSQL, de 1998, é abreviação para “Not Only SQL”
- <http://nosql-database.org/>



**Cassandra**

**Project Voldemort**  
*A distributed database.*

**ORACLE®**  
**NOSQL DATABASE**





# Principais Características

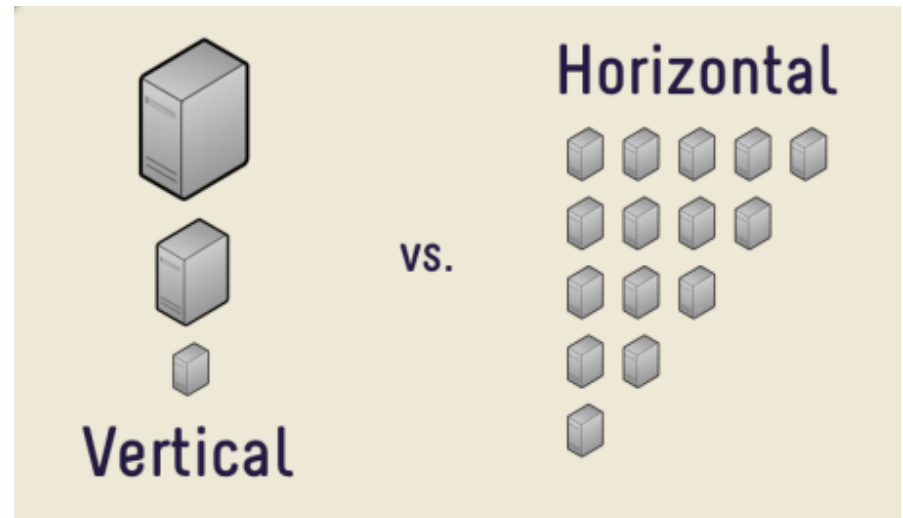
- Escalabilidade Horizontal.
- Ausência de Esquema ou Esquema Flexível.
- Suporte a Replicação.
- API Simples.
- Nem Sempre é Consistente.



# Escalabilidade Horizontal

---

- A escalabilidade Horizontal consiste em aumentar o número de máquinas disponíveis.
- A escalabilidade Horizontal em modelos relacionais seria inviável devido a concorrência.
- Como nos modelos NoSQL não existe bloqueios, esse tipo de escalabilidade é a mais viável.



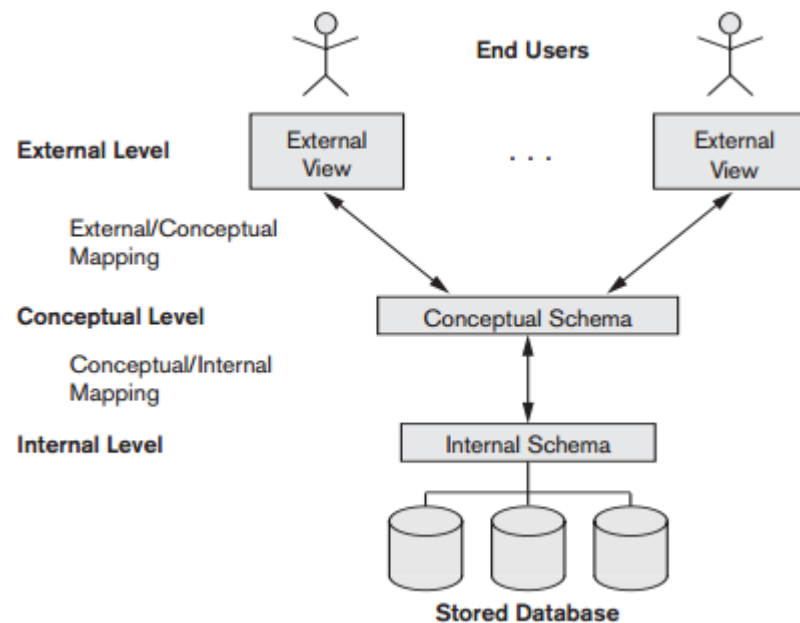
# Conceito



*Em bancos de dados é importante distinguir entre a descrição do banco de dados e o próprio banco de dados.*



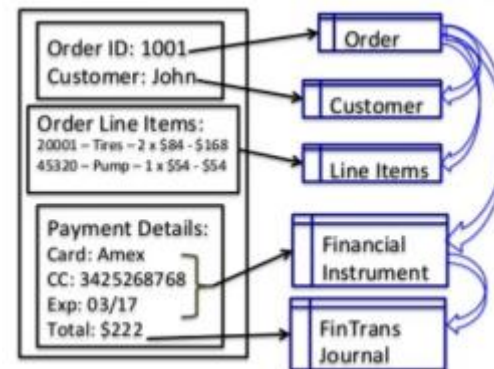
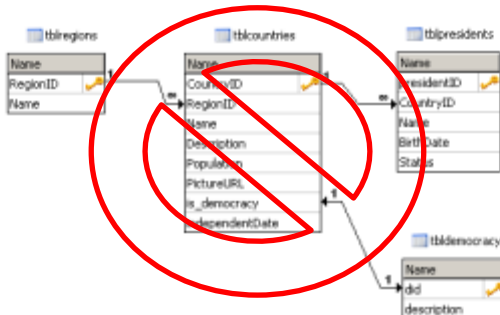
*A descrição é o chamado esquema de banco de dados (schema database), é especificado durante o projeto de banco de dados e não se espera que mude com frequência.*



# Ausência de Esquema ou Esquema Flexível

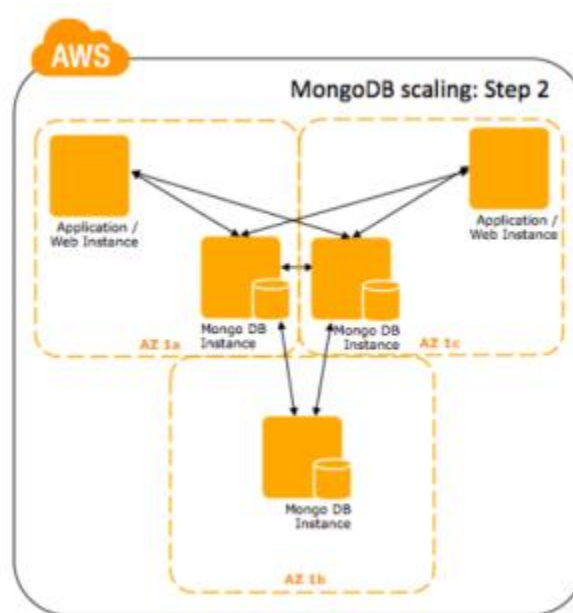
- Apresentam ausência de esquema ou esquema flexível, isso permite uma fácil aplicação da escalabilidade e também um aumento na disponibilidade dos dados.
- Mas também devido a essa ausência, não há garantia da integridade dos dados.

Database schema



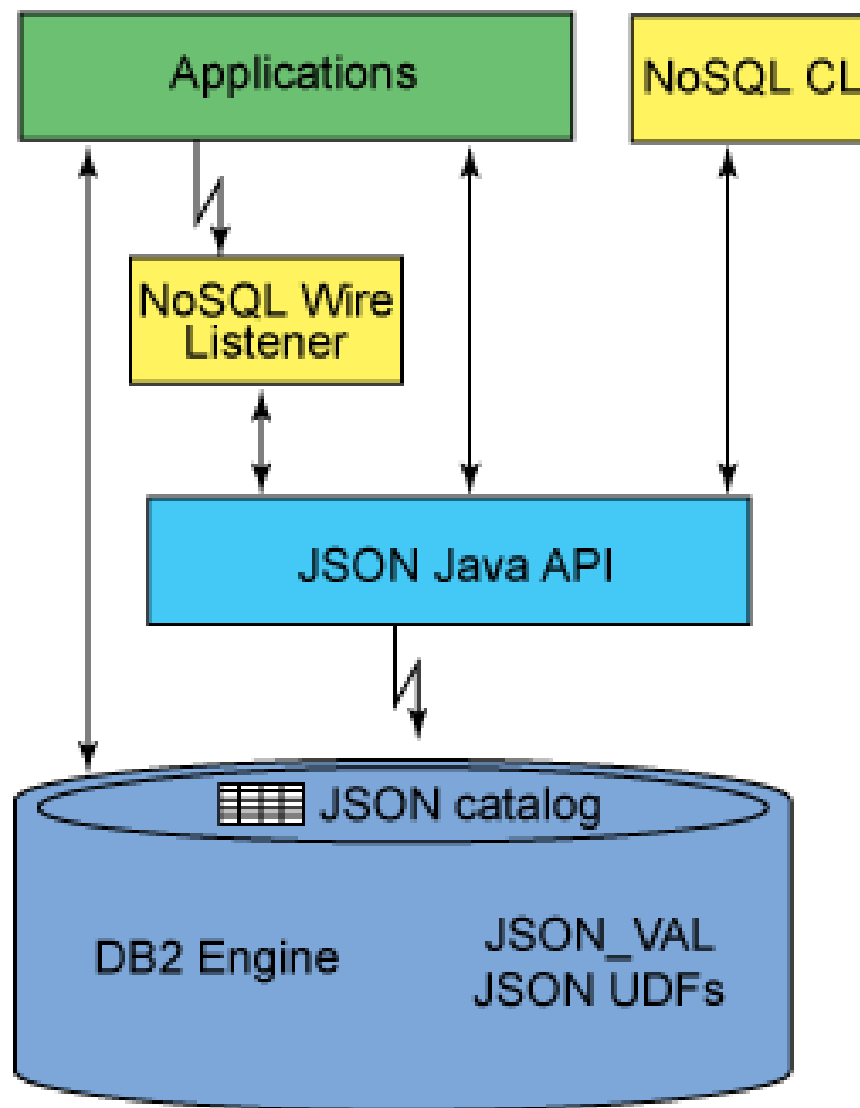
# Suporte a Replicação

- Permitem a replicação de uma forma nativa o que provém uma escalabilidade maior e também uma diminuição do tempo gasto para a recuperação de informações.



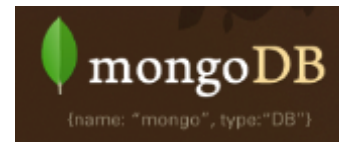
# API Simples

- Para que o acesso as informações seja feito da forma mais rápida possível, APIs são desenvolvidos para que qualquer aplicação possa ter acesso aos dados do banco de dados.



# Nem Sempre é Consistente

- Os bancos de dados NoSQL nem sempre conseguem se manter consistentes.





# Modelo de Dados NoSQL

- Um banco de dados NoSQL pode usar um dos seguintes modelos de dados:
  - Chave/Valor (Key/value)
  - Colunas (Columnar)
  - Documento (Document)
  - Grafos (Graph)

Key/Value  
Volatile

key	value
123	123 Main St.
126	(805) 477-3900

Key/Value  
Persistent

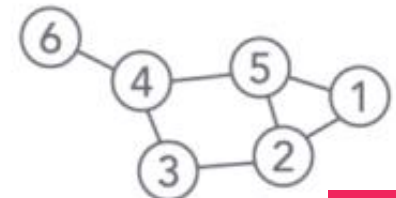
Wide-Column

1	Things	A foo	B bar	C baz	
2	Things	C Apple	E Banana	People	A Pat
3	Languages	A C	B Java	C Python	

Document

```
{ "user" : {
  "id": "124",
  "name": "Emmanuel",
  "addresses": [
    { "city": "Paris", "country": "France" },
    { "city": "Atlanta", "country": "USA" },
  ]
}
```

Graph



# Chave/Valor

- Modelo mais simples.
- Permite a visualização do banco como uma grande tabela.
- Todo o banco é composto por um conjunto de chaves que estão associadas a um único valor.
- Cookies
  - Tem sido o principal mecanismo de armazenamento
- W3C Web Storage
  - Modelo “minimo” de armazenamento
  - Baseado em (chave, valor)

Chave	Valor
Disciplina	Modelagem
Professora	Sandra

# Chave/Valor

Oracle	Riak (Chave/Valor)
Instância de Banco de Dados	Cluster Riak
Tabela	Bucket
Linha	Chave-Valor
ROWID	Chave

- Valor é armazenado, sem preocupação com o que representa
- Aplicação faz o tratamento e se preocupa com o entendimento do valor
- Muito escalar devido o acesso somente pela chave
- Pode armazenar tudo em um Bucket ou criar buckets de domínio
- Limitação: consulta só pela chave, retornando o valor. Valor não pode ser consultado pelo atributo.

# Chave/Valor

## Web Storage API

set Item (chave, valor)	adiciona/atualiza par chave-valor
get Item (chave)	recupera o valor associado à chave
key (n)	recupera a enésima chave
remove Item (chave)	remove o par que possui a chave
length	indica o número de pares chave-valor
clear ()	remove todos os dados do repositório

# Chave/Valor

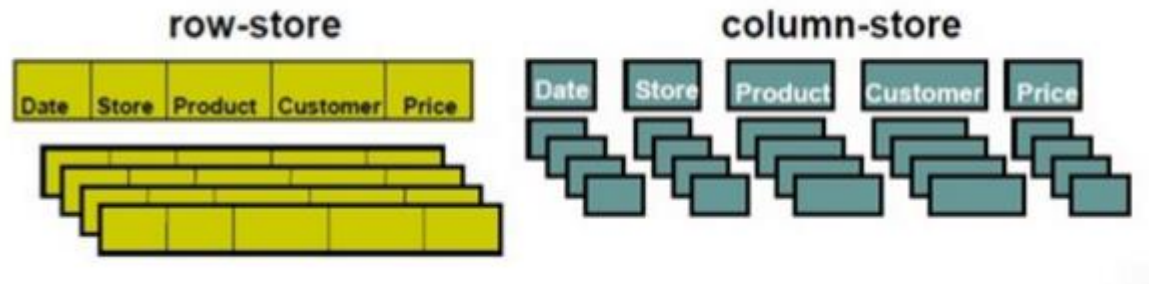
Chave	Valor
Fusca	vermelho
Chevette	verde

- Bom Para
  - Armazenamento de informações de sessão
  - Perfil de usuário e preferências
  - Dados de carrinho de compra
- Ruim para
  - Relacionamento entre dados
  - Transações com múltiplas operações
  - Consultas por dados e atributos
  - Operações por conjuntos

Chave (Campo)	Valor (Instância)
Nome	Edson França
Idade	43
Sexo	Masculino
Fone	999999999

# Orientado a Colunas

- Um pouco mais complexos.
- Os dados são indexados por uma tripla (linha, coluna e timestamp).
- As linhas e as colunas são identificadas por chaves e o timestamp, é o que permite identificar as diferentes versões de um mesmo dado.



<http://db.csail.mit.edu/pubs/abadi-column-stores.pdf>

(Abadi et al., 2013)

# Orientado a Colunas

- Colunas:
  - compostas por um nome que identifica a coluna, além de um valor e um timestamp (que são fornecidos pela aplicação cliente no momento da inserção).
- Família de colunas:
  - é análogo a uma tabela do modelo relacional, ao contrário das colunas as famílias de colunas não são dinâmicas, sendo necessário declará-las anteriormente em um arquivo de configuração.
- Super Colunas:
  - são compostas por outras colunas.
- Super Famílias de Colunas:
  - são compostas somente por Super Colunas

Sales				
	saleid	prodid	date	region
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

(a) Column Store with Virtual Ids

Sales				
	saleid	prodid	date	region
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

(b) Column Store with Explicit Ids

Sales				
	saleid	prodid	date	region
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

(c) Row Store



# Orientado a Colunas

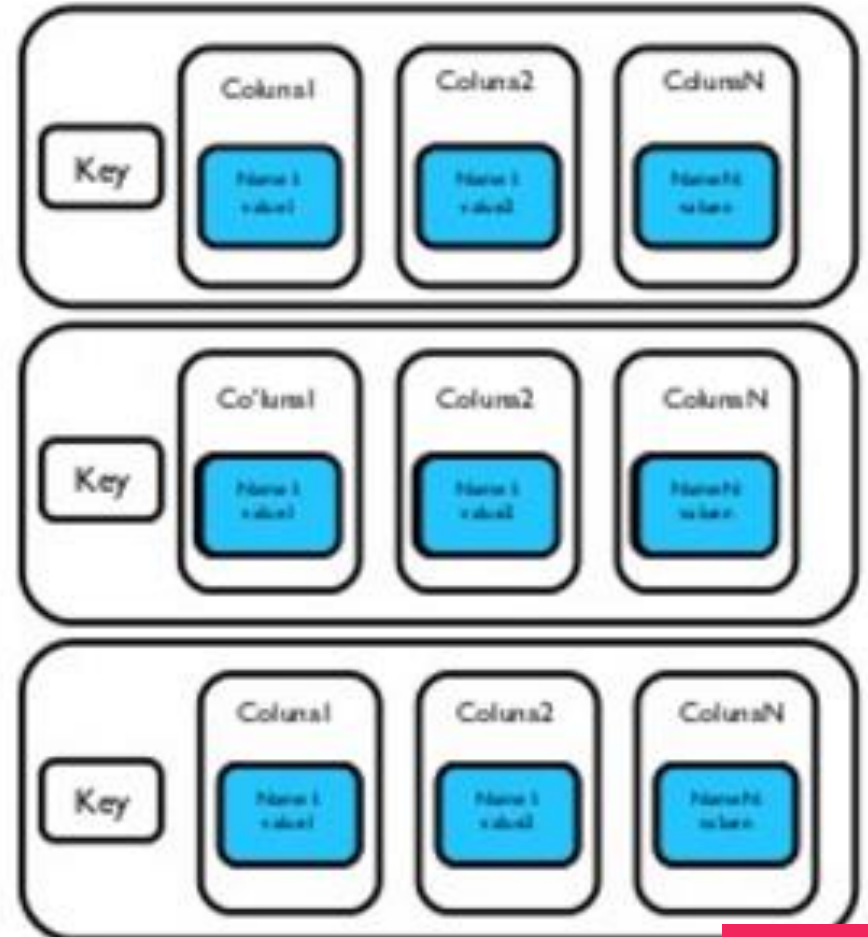
Oracle	Cassandra (colunas)
Instância de Banco de Dados	Cluster
Banco de Dados	Keyspace
Tabela	Família de Colunas
Linha	Linha
Coluna (a mesma para todas as linhas)	Coluna (podem ser diferentes por linha)

- Uso de famílias de colunas padrão e SUPERCOLUNAS (Encadeamento)
- Eficaz para manter dados relacionados agrupados
- Operações básicas usando GET, SET e DEL
- CQL (Padrão semelhante SQL— Cassandra)
- LIMITAÇÃO: Colunas lidas e dessenalizadas de uma vez (Value único).

# Orientado a Colunas

- Bom Para
  - Registro de eventos (log)
  - Sistema de Gerenciamento de Conteúdo (CMS)
  - Contadores
- Ruim para
  - Sistemas que requerem ACID para leituras e gravações

Família de Colunas



## Orientado a Documentos

- Armazena uma coleção de documentos.
- Um documento no geral, é um objeto com um código único e um conjunto de campos, que podem ser strings, listas ou documentos aninhados.
- Sua estrutura se assemelha com de chave-valor.



# Orientado a Documento

Oracle	MongoDB (documento)
Instância de Banco de Dados	Instância MongoDB
Esquema	Banco de Dados
Tabela	Coleção
Linha	Documento
ROWID	_id
Junção	DBRef

- 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.

```
{ "order_id": "1001", "customer": "John",  
  "orderitems": [ { "prodid": "20001", "prodname": "Tires", "Qty": 2, "price": 168 },  
                  { "prodid": "45320", "prodname": "Pump", "Qty": 1, "price": 54 } ],  
  "pcard": "Amex", "pcc": "3425268768", "pexp": "03/17", "ord_tot": 222 }
```

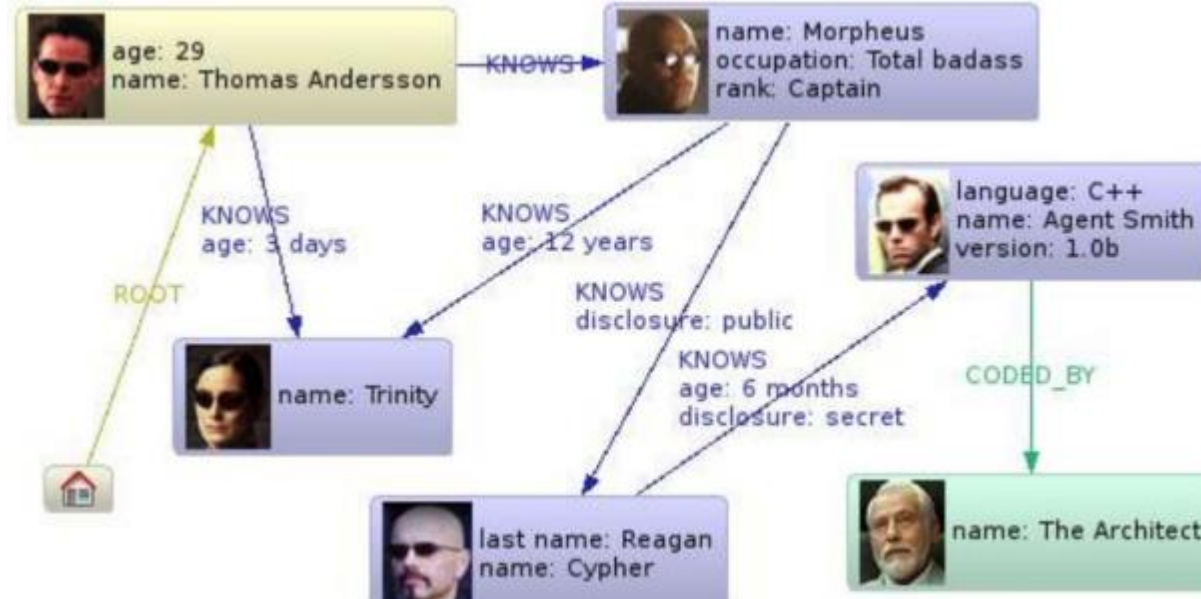
# Orientado a Documentos

- Bom Para
  - Registro de eventos (log)
  - Sistema de Gerenciamento de Conteúdo (CMS)
  - Análise web ou em tempo real (analytics)
  - Aplicativos de comercio eletrônico
- Ruim para
  - Transações complexas com diferentes operações
  - Consultas em estruturas de agregados variáveis

```
{  
  "id": "1234"  
  "firstName": "John",  
  "lastName": "Smith",  
  "isAlive": true,  
  "age": 25,  
  "address": {  
    "streetAddress": "21 2nd Street",  
    "city": "New York",  
    "state": "NY",  
    "postalCode": "10021-3100"  
  }  
}
```

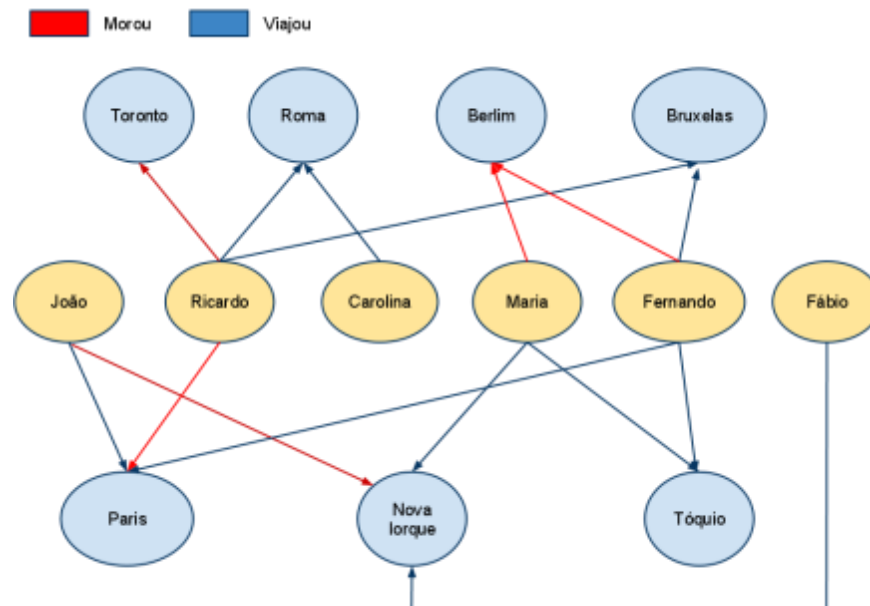
# Orientado a Grafos

- Neste modelo, o banco pode ser comparado com um multigrafo rotulado e direcionado, onde cada nó pode ser conectado por mais de uma aresta.
- 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.



# Orientado a Grafo

- 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 Grafo

- Bom Para
  - Dados conectados
  - Roteamentos, envio e serviços baseados em localização
  - Sistemas de recomendação
- Ruim para
  - Sistemas com atualização em lote (várias entidade atualizadas em uma operação)

```
Relationship persephone = roles.get( "name", "Persephone" ).getSingle();  
Node actor = persephone.getStartNode();  
Node movie = persephone.getEndNode();
```

```
for ( Relationship role : roles.get( "name", "Neo" ) )  
{  
    // this will give us Reeves twice  
    Node reeves = role.getStartNode();  
}
```

```
for ( Node actor : actors.query( "name", "*e*" ) )  
{  
    // This will return Reeves and Bellucci  
}
```

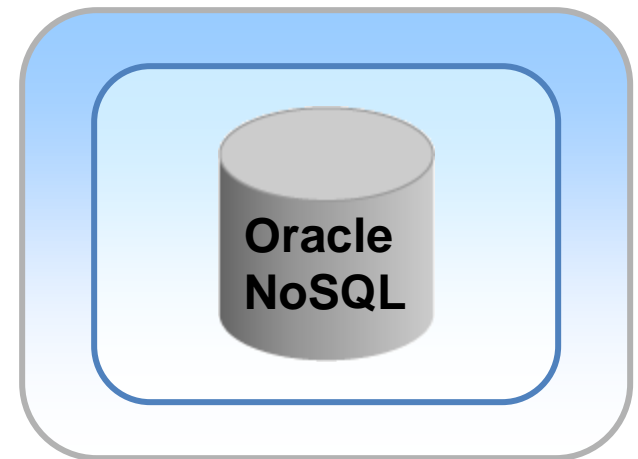
# Pontos a considerar antes de escolher NoSQL

- Ao decidir sobre uma tecnologia de banco de dados de aplicativos analise os seguintes fatores :
  - Analise os dados a serem armazenados.
    - Alto volume , baixo valor ?
    - Se a resposta for "sim", então NoSQL é a melhor escolha.
  - Analise o schema do aplicativo
    - Dinâmico ?
    - Se a resposta for "sim", então NoSQL é a melhor escolha.

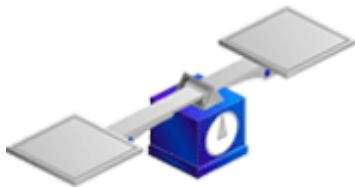


# Banco de Dados Oracle NoSQL

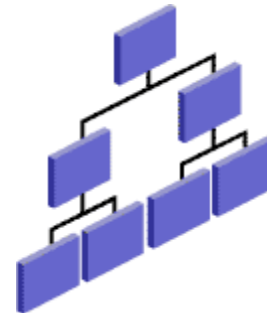
- O Banco de Dados Oracle NoSQL é:
  - Um banco de dados chave-valor (key-value)
  - Escrito em Java
  - Acessado usando Java APIs
  - Desenvolvido com o Oracle Berkeley DB Java Edition
  - A solução Oracle para a acesso a big data



# Principais Características



Load Balancing



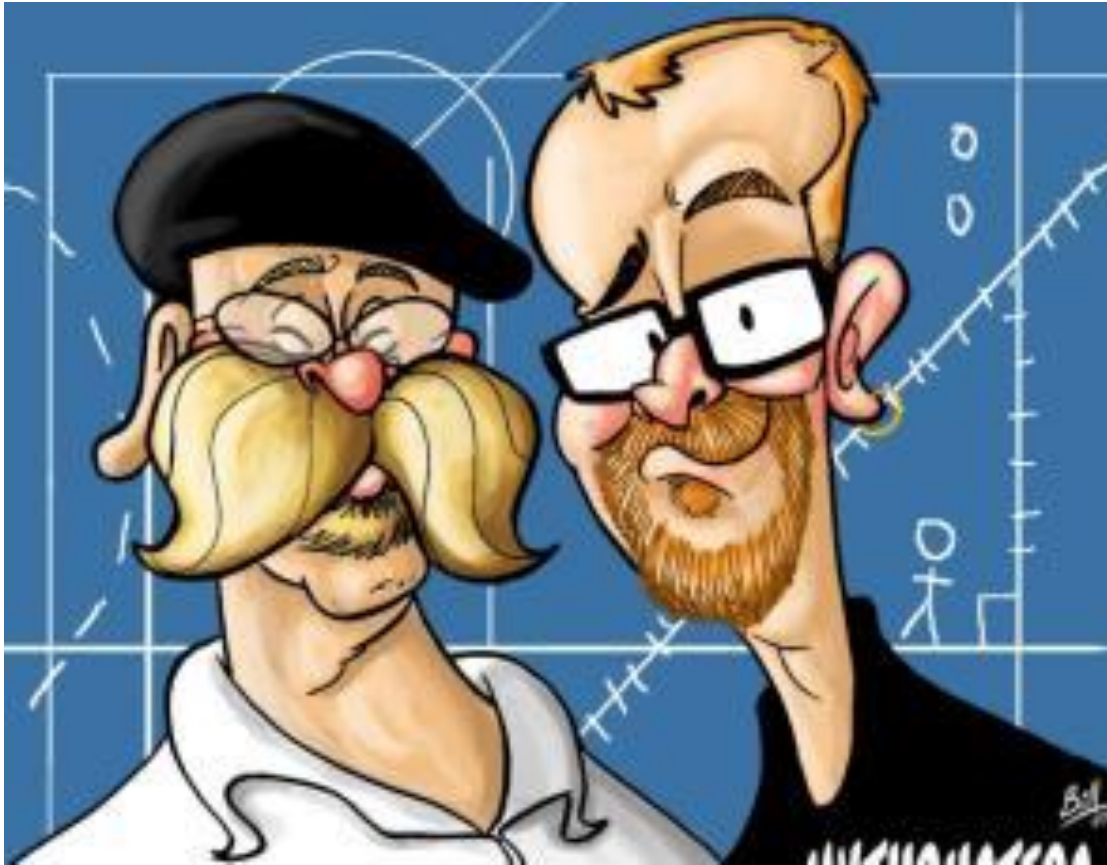
Modelo de Dados Simples

# Banco de Dados Baseado em Documentos

- O banco de dados é uma coleção de documentos indexados.
- Baseado em XML
  - BaseX  
(<http://basex.org>)
- JSON
  - CouchDB  
(<http://couchdb.apache.org>)
  - MongoDB  
(<http://www.mongodb.org>)

```

object
  {}
  { members }
members
  pair
  pair , members
pair
  string : value
array
  []
  [ elements ]
elements
  value
  value , elements
value
  string
  number
  object
  array
  true
  false
  null
  
```



# NOSQL

- NoSQL é escalável.
- Não precisamos de DBAs.
- NoSQL é mais econômico.



NOSQL é escalável

- Grande promessa dos bancos NOSQL: são mais escaláveis que os bancos de dados relacionais.
- O problema com esta mensagem que é vendida por algumas empresas é que ela não é inteiramente verdade.
- Dizer que seu sistema escala sozinho é vender um sonho. Ele pode até ser mais fácil de escalar se comparado a outras soluções, mas ainda sim exigirá algum esforço para escalar, e escalar de forma eficiente





Não precisamos de  
DBAs

- O DBA é responsável por instalar, configurar e manter cada banco de dados com suas peculiaridades.
- O NoSQL não precisa de DBAs no sentido tradicional, mas é preciso que alguém seja responsável por lidar com o banco e com o acesso aos dados.
- Esta função passa a ser parte do trabalho de um desenvolvedor.
- Aplicações reais em produção podem misturar bancos relacionais e não relacionais, o desenvolvedor deve possuir o conhecimento necessário para administrar os dois mundos.



## NoSQL é mais econômico

- Meia verdade.
- O custo em usar um banco de dados relacional pode ser proibitivo devido à escala, ou a licenças envolvidas.
- Existem casos em que uma solução relacional atende perfeitamente todas as necessidades do cliente e pode ser considerada barata.
- Bancos de dados Open Source como MySQL e PostgreSQL são usados sem problemas por um grande número de aplicações, e com sucesso

# Primeiro Cenário

## Sistema de Saúde

- Uma cidade precisa de um sistema de saúde centralizado, com os seguintes requisitos:
  - Deve guardar os registros de saúde de todas as pessoas na cidade em todos os diferentes hospitais.
  - Os médicos devem ser capazes de usar este sistema para entender o histórico de saúde dos pacientes.
  - O sistema deve ser capaz de armazenar diferentes tipos de dados.
  - As políticas de armazenamento de dados podem mudar ao longo do tempo.

***Qual tecnologia que você recomendaria para construir esse aplicativo ?***



# Segundo Cenário

## Sistema de Recursos Humanos

- Uma empresa multinacional precisa de um sistema centralizado de recursos humanos com os seguintes requisitos:
  - Deve armazenar as informações de todos os colaboradores na organização ( tanto os funcionários atuais e quanto os antigos )
  - Para cada funcionário deve armazenar informações como: data de contratação, detalhes da família, histórico de trabalho, histórico saúde, benefícios recebidos da empresa e data de demissão ou aposentadoria.
  - Deve armazenar scans dos documentos do funcionário, foto, impressão digital, amostras de voz e assim por diante.
  - Benefícios da empresa e as políticas de RH podem mudar de tempos em tempos.

*Qual tecnologia que você  
recomendaria  
para construir esse sistema ?*

# Terceiro Cenário

## Sistema de Marketing de Varejo

- Uma nova estratégia de marketing pretende oferecer cupons de desconto para seus clientes quando eles estiverem perto de sua loja.

É preciso armazenar:

- Os perfis dos clientes
- Histórico de compras do cliente
- Sinais GPS de dispositivos móveis
- Detalhes das promoções

*Qual tecnologia que você  
recomendaria  
para construir esse ambiente ?*



Para saber  
mais

- [Software Big Data livre para os Impacientes.](#)
- [Behind Big Data.](#)
- [SDSS-III.](#)
- [MapReduce Tutorial.](#)
- [Big Data in Astronomy.](#)
- [Statistics in Astronomy.](#)
- [Processamento de Dados Distribuídos com Hadoop.](#)
- [Introdução à analítica de dados com base em Hadoop no IBM SmartCloud Enterprise.](#)
- [Mining of Massive Datasets.](#)
- [Escalabilidade horizontal para execução paralela de tarefas.](#)
- [SkyServer DR9](#)

## REFERÊNCIAS



- WATSON, John. Oracle Database 11g Administração I. Editora Bookman, 2010.
- LONEY, Kevin; BRYLA, Bob. Manual do DBA. Editora Osborne – McGraw-Hil, 2011.
- PUGA, Sandra et al. Banco de Dados. Editora Pearson/Prentice Hall, 2014.
- OLIVEIRA, Celso H. P. Guia de Consulta Rápida – Oracle 10G PL/SQL. Editora Novatec, 2005.