Sumário

1. Introdução a Aplicações Não-Convencionais
2. Revisão de Modelagem Conceitual
3. BD Orientado a Objetos (BDOO)
4. BD Objeto-Relacional (BDOR)
5. BD Temporal (BDT)
6. BD Geográfico (BDG)
7. BD XML
8. **BDs nas Nuvens**
9. Gerência de Dados na Web

Tópicos

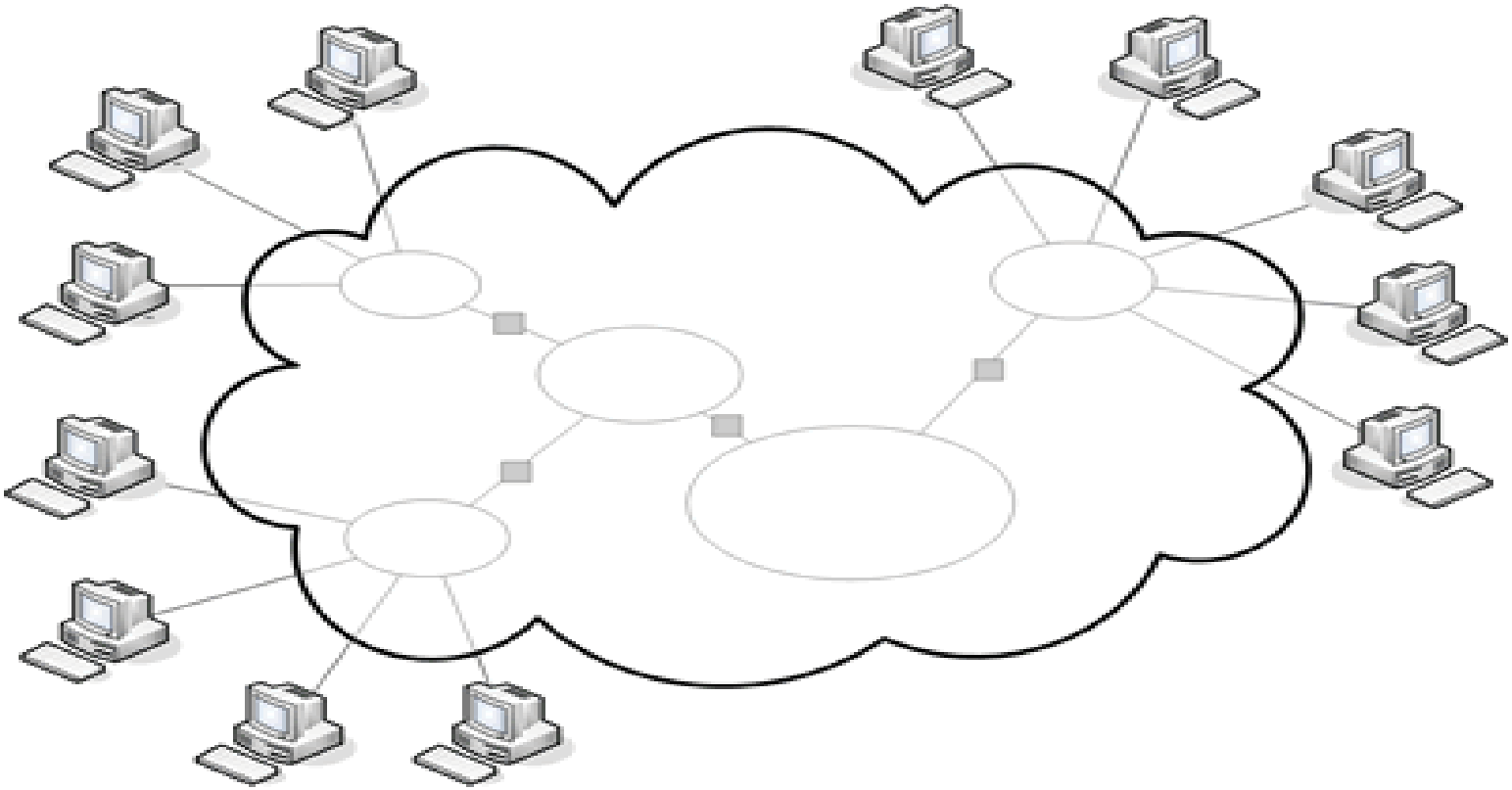
1. Introdução à Computação nas Nuvens
2. Gerência de Dados na Nuvem
3. Categorias de BDs na Nuvem
4. BDs No-SQL
5. Considerações Finais

Tópicos

1. Introdução à Computação nas Nuvens
2. Gerência de Dados na Nuvem
3. Categorias de BDs na Nuvem
4. BDs No-SQL
5. Considerações Finais

Computação na Nuvem (*Cloud Computing*)

- Paradigma de oferta de serviços remotos de computação
  - Via *Internet* ou outra infraestrutura de rede



Computação na Nuvem

- Objetivos
  - Atendimento em larga escala de usuários e organizações sem infraestrutura computacional ou capital
  - Atendimento de organizações com requisitos dinâmicos em termos de demanda computacional
- Características principais dos serviços oferecidos
  - Baixo custo (ou gratuitos)
  - Transparência de acesso
  - Elasticidade (extensão/retração de serviços sob demanda)
    - Analogia com serviços de luz, água, telefone, ...

Níveis de Oferta de Serviços

- Hardware (Infraestrutura) (IaaS)
  - Servidores, disco, rede, …
    - Demanda de processamento, armazenamento, ...
  - Exemplo: *Amazon elastic cloud computing*
- Plataformas (PaaS)
  - SO, ambientes de desenvolvimento, linguagens de programação
    - Exemplo: *Microsoft Azure*

Níveis de Oferta de Serviços

- Software (SaaS)
  - Propósitos específicos e execução em diferentes dispositivos
    - Laptops, celulares, etc
  - Exemplo: *Google drive*
- Gerência de Dados (DaaS)
  - **SGBDs nas nuvens**
    - Exemplos: *Amazon S3, Cassandra, Mongo DB, ...*

Tópicos

1. Introdução à Computação nas Nuvens
2. Gerência de Dados na Nuvem
3. Categorias de BDs na Nuvem
4. BDs No-SQL
5. Considerações Finais

Gerência de Dados na Nuvem

- Vantagens
  - Redução de custos para aquisição de SGBD
  - Delegação de tarefas de administração de dados (baixa intervenção humana)
    - Exemplo: *Tuning, backup, ...* do BD
  - Escalável para processamento de grandes volumes de dados
    - Arquitetura baseada em *Data Centers*

SGBDs na Nuvem - Características

- Escalabilidade
  - *Data centers*, processamento paralelo
- Disponibilidade
  - Replicação de dados, consistência relaxada
- APIs simples para acesso
  - Baixo *overhead* com *parsing*/execução de comandos de linguagens de BD
- Alternância de *workload*
  - Elasticidade na demanda por operações sobre dados e alocação de recursos

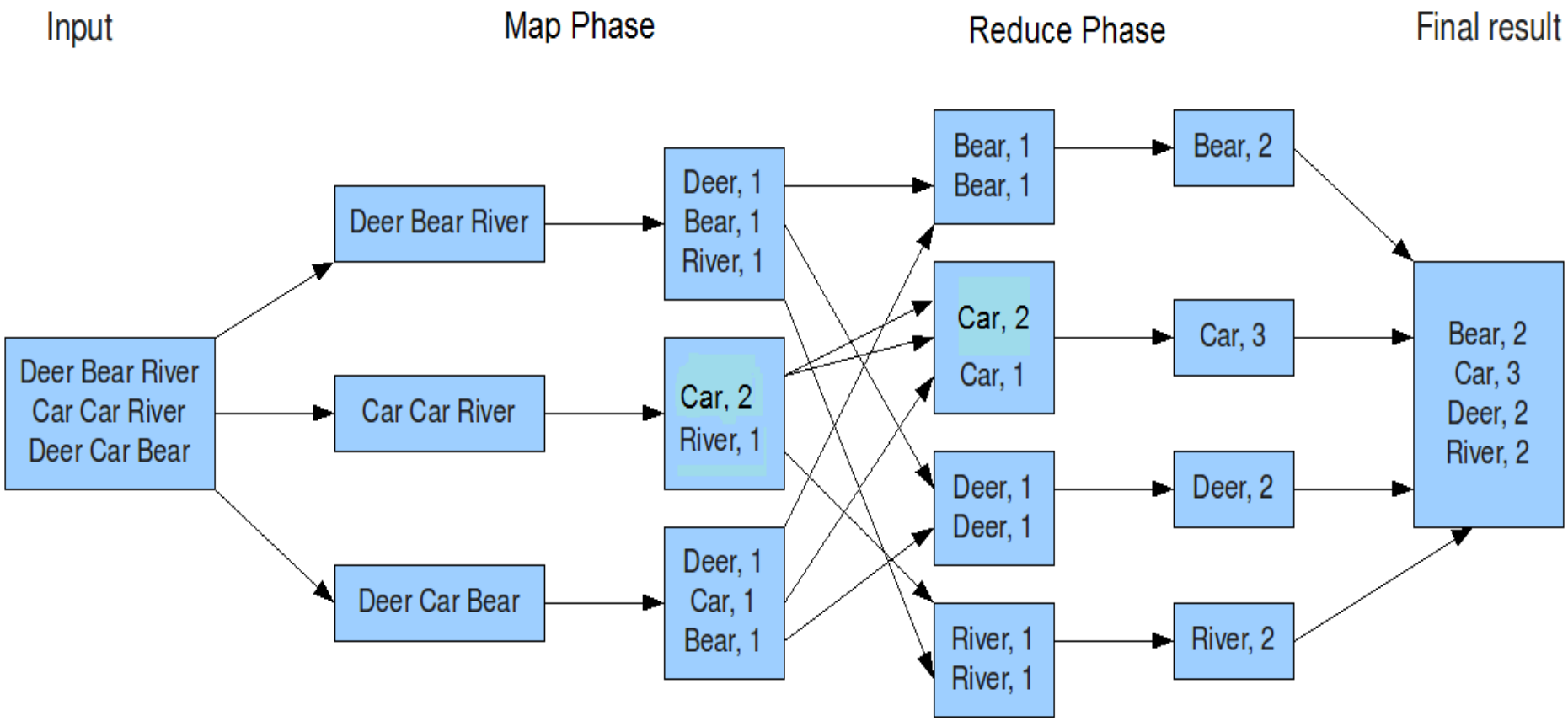
Tecnologias para Armazenamento e Acesso a Dados na Nuvem

1) *Modelos de Programação*

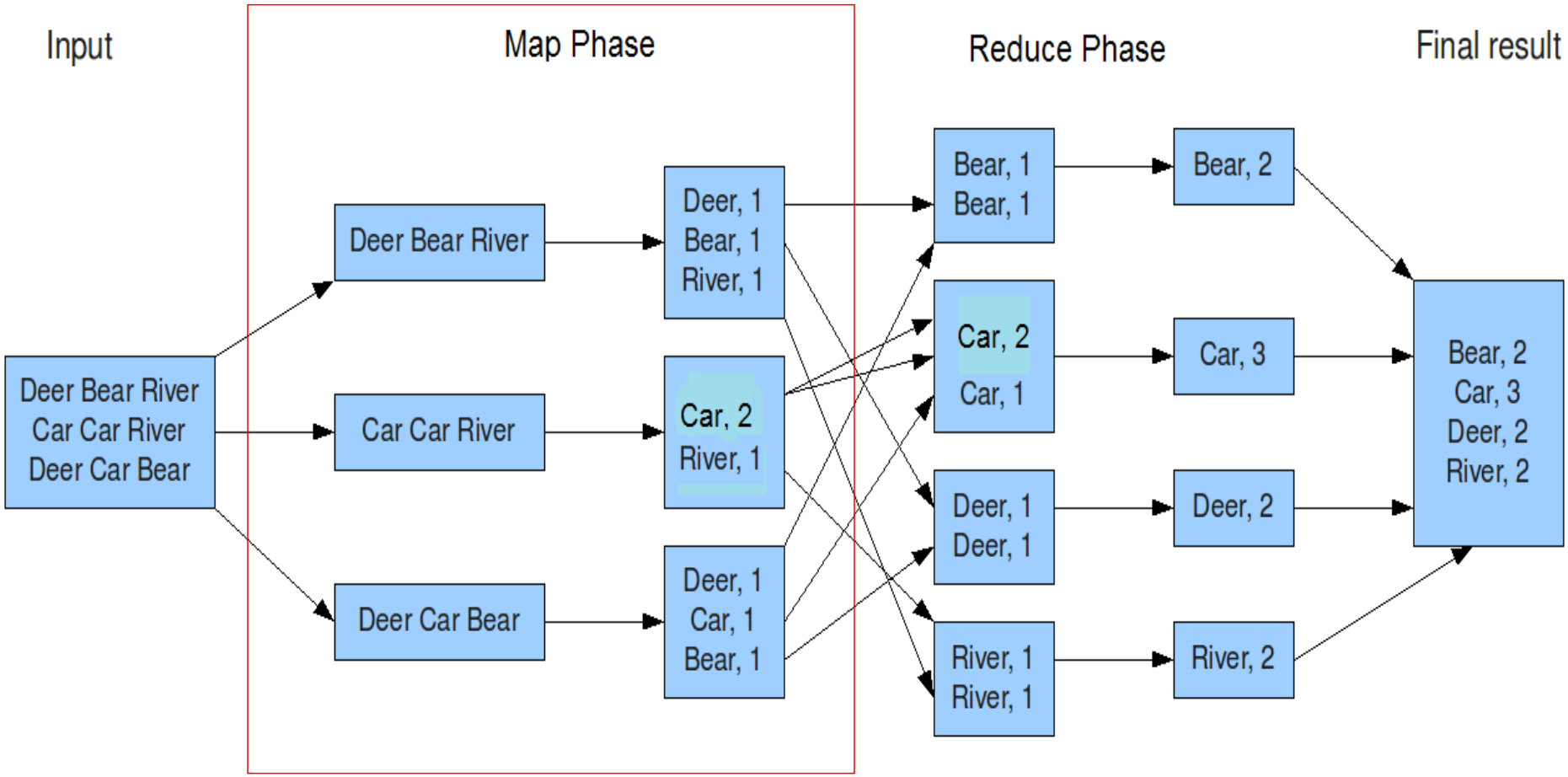
- Exemplo: *MapReduce* (2 operações)
  - Objetivo: paralelizar o processamento de grande volume de dados
    - Paradigma *"dividir para conquistar"*
  - **Map**: recebe uma fonte de dados e retorna um coleção de pares *(chave, valor)*
  - **Reduce**: recebe uma coleção de pares *(chave, valor)* e retorna um resultado sumarizado

```
/* Exemplo: determinar frequência de palavras em docs Web */

MAP(URL key,String value)→ BAG(String key,Int value);

REDUCE(BAG(String key,Int value))→LIST(String key,Int
value);
```

MapReduce - Exemplo

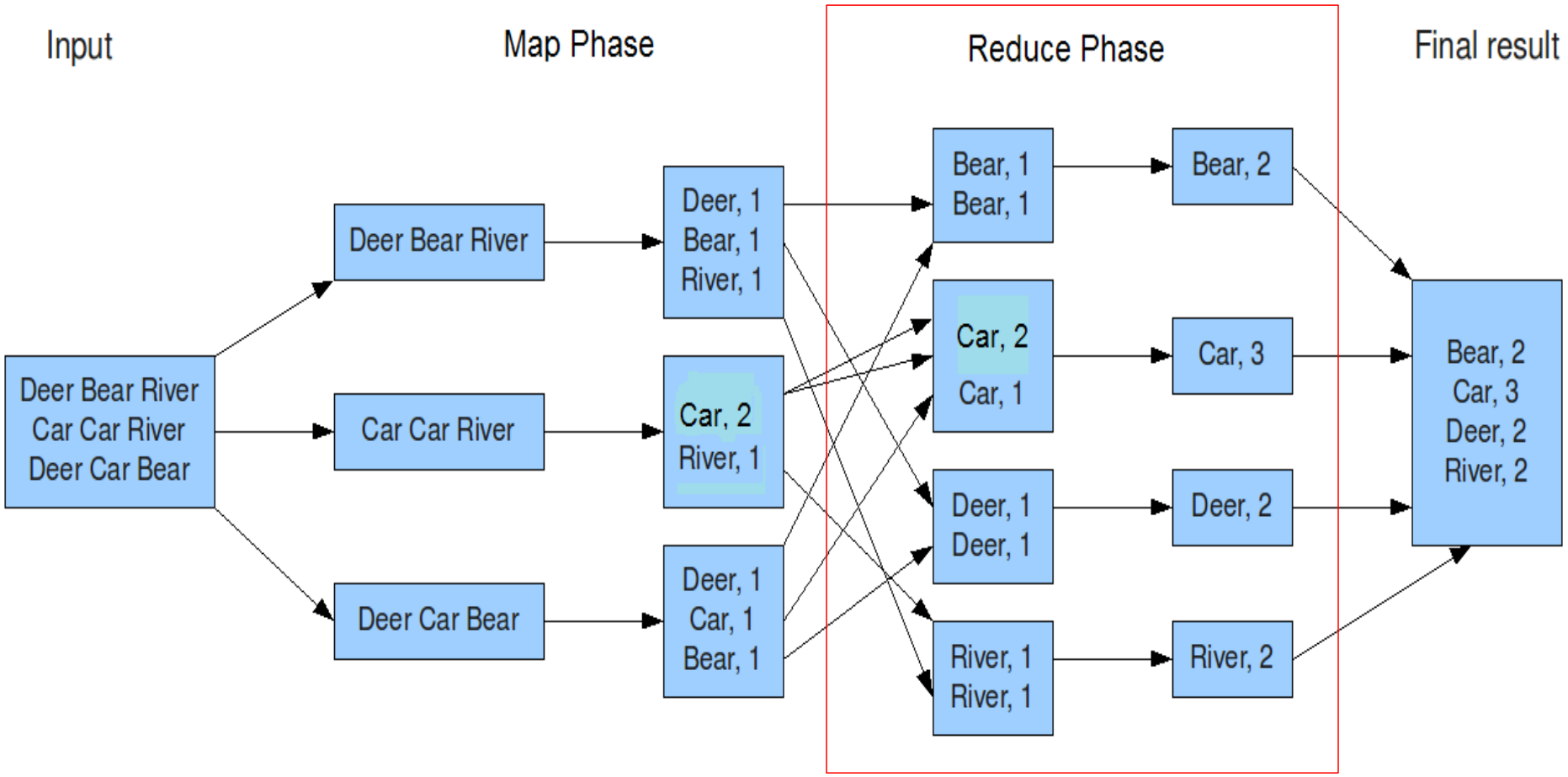


MapReduce - Exemplo



- Divide a massa de dados entre servidores
- Gera um mapeamento dos dados para a característica a ser processada (“palavras e suas frequências”)

MapReduce - Exemplo



- Coleta os pares *(chave,valor)* produzidos em cada servidor
- Gera o resultado final sumarizado (“frequência total de cada palavra”)

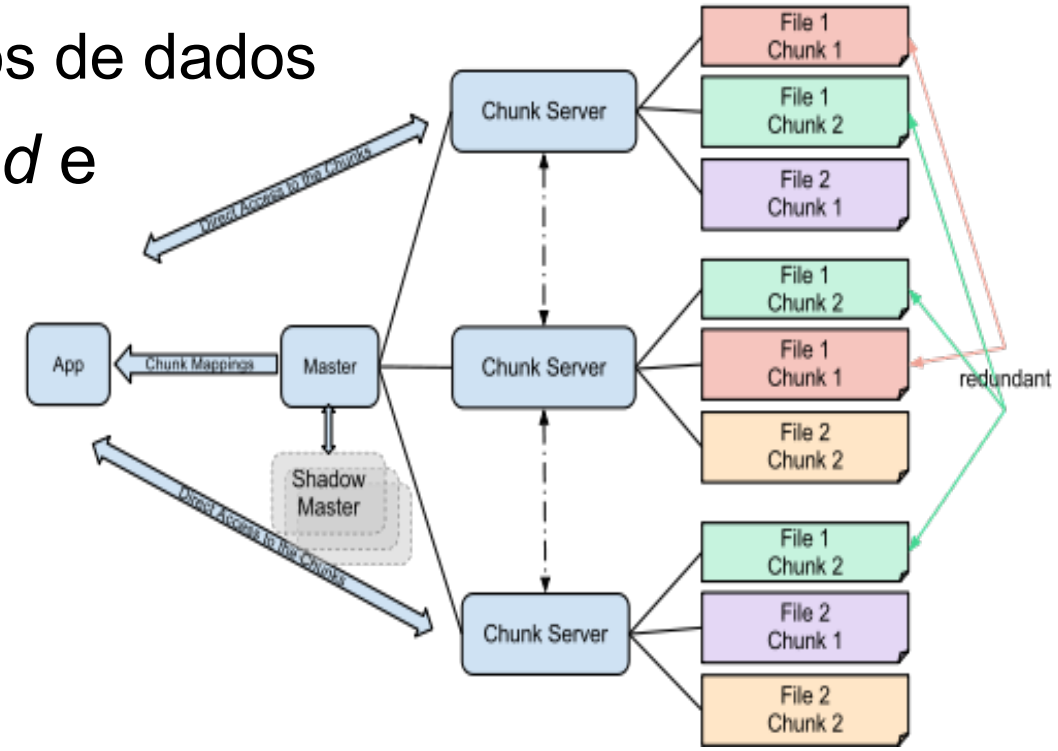
Tecnologias para Armazenamento e Acesso a Dados na Nuvem

2) Sistemas de arquivos

- Exemplos:
  - GFS (*Google File System*)
    - Otimizado para acesso a grandes volumes de dados
    - Sistema de arquivos distribuídos em *data centers*
  - HDFS (*Hadoop File System*)
    - Similar ao GFS
    - Desenvolvido pela Apache
    - *Open source*

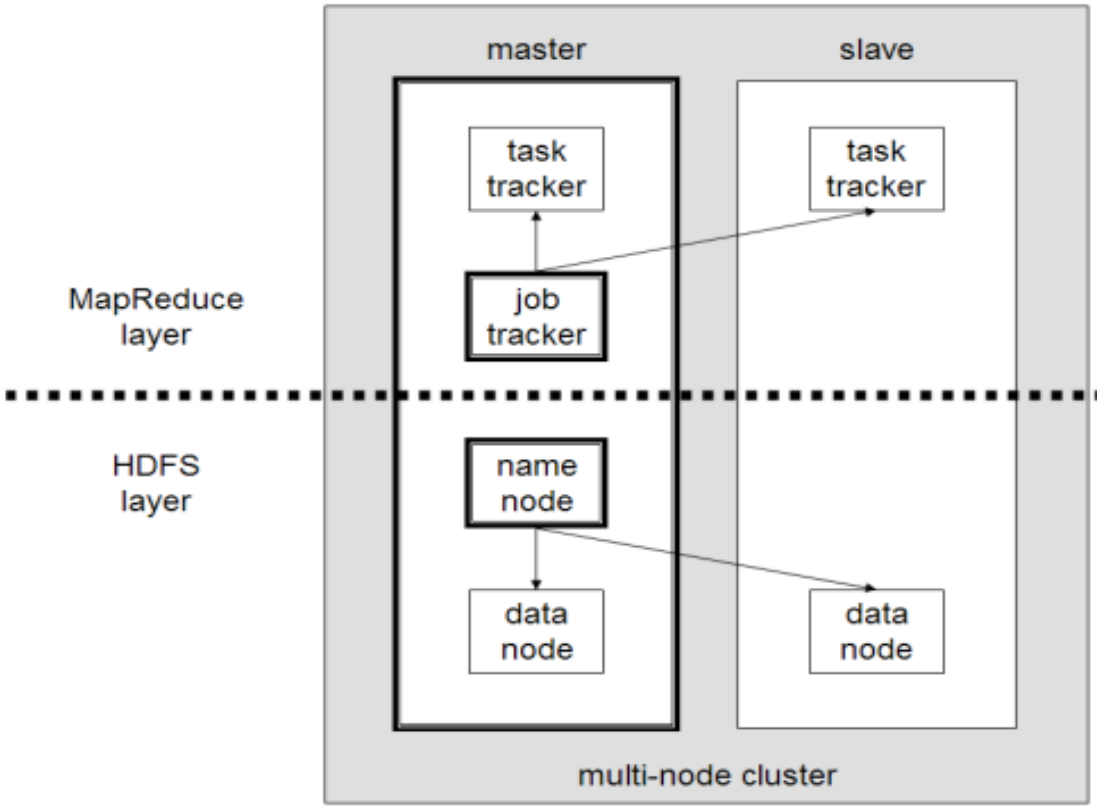
Google File System (GFS)

- Arquivos organizados em partições (*chunks*)
  - *Chunks*: porções de *BigTables*
- Um cluster possui 1 nodo *Master* e nodos *Chunk Server*
  - *Master* : índices e catálogos de metadados
  - *Chunk Server* : arquivos de dados
- Foco em operações *read* e *append*
- Cada *chunk* replicado pelo menos 3x



Hadoop File System (HDFS)

- Arquitetura também é *Master/Slave*
- Centrada em processamento *MapReduce*
  - Nodos *slave* podem manter dados e processar tarefas *Map* e *Reduce*
  - Nodos *master*
    - Mantêm índices e metadados
    - Controlam processos *MapReduce*
    - Podem manter dados, para melhor eficiência de acesso

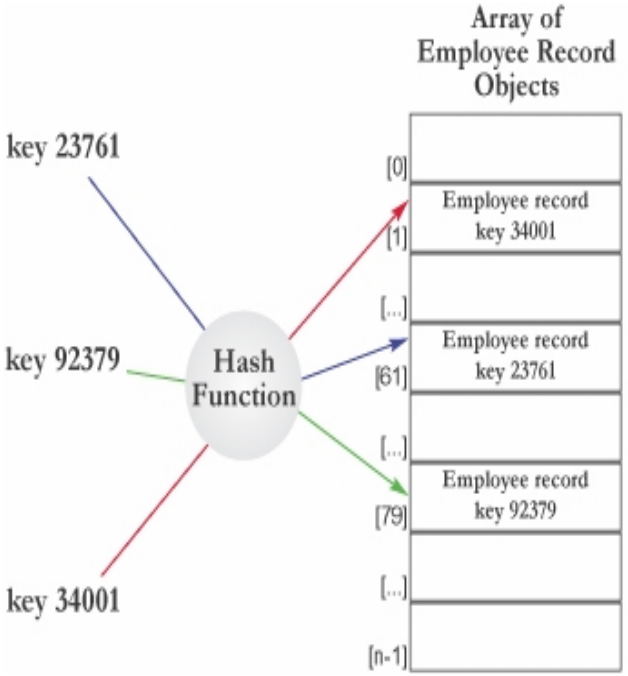


Tecnologias para Armazenamento e Acesso a Dados na Nuvem

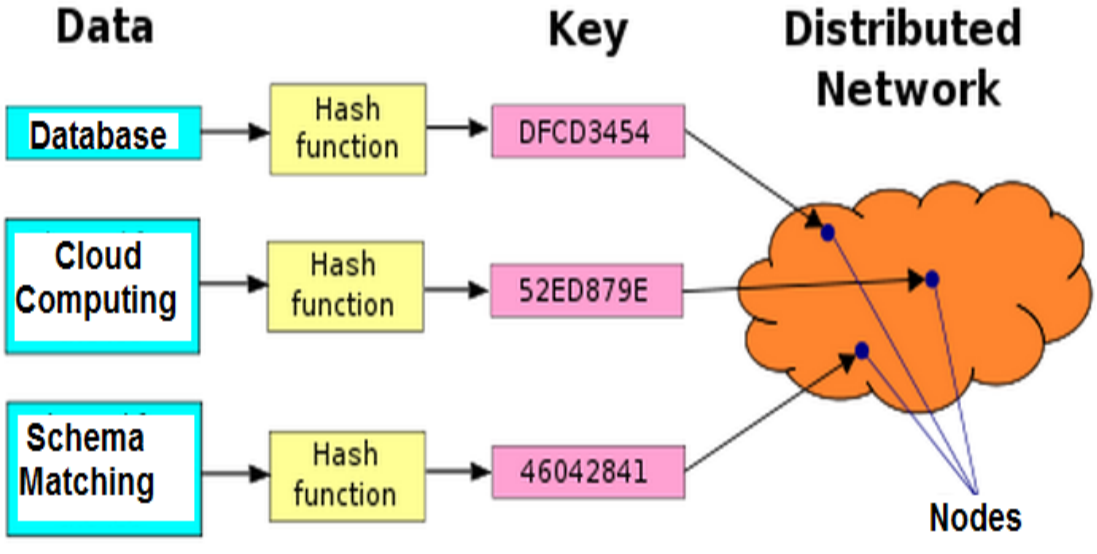
3) Novas estruturas de acesso

- Exemplo: DHT (*Distributed Hash Table*)
  - Armazenamento e acesso baseado em *hash* para grandes volumes de dados
    - Cada valor de chave *K* mapeia para o conjunto de nodos que possuem *K*
    - Cada nodo com seu índice *hash* local que indica a localização de *K*
    - Cada nodo mantém sua cópia da DHT

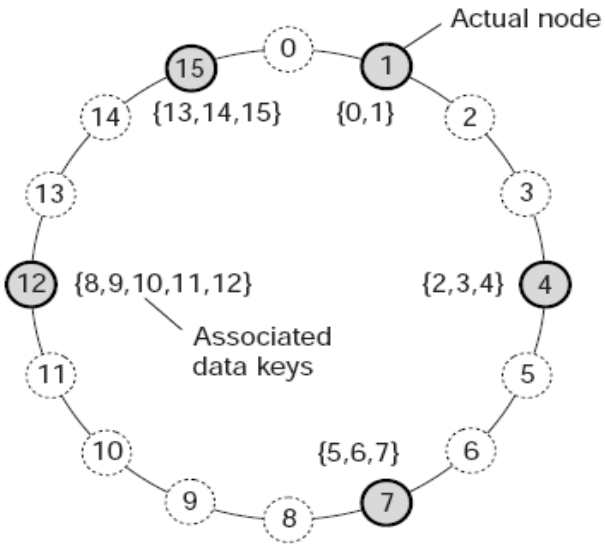
DHT - Exemplo



Hash table

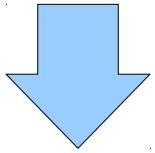


Distributed Hash table



BDs nas Nuvens - Transações

ACID (BDs convencionais)	**BASE** (BDs na nuvem)
Atomicidade Consistência Isolamento Durabilidade	BAsically Available • BD por *default* está disponível Soft State • BD não necessariamente está sempre consistente Eventually Consistent • BD torna-se consistente em um determinado momento



Consistência eventual (fraca)

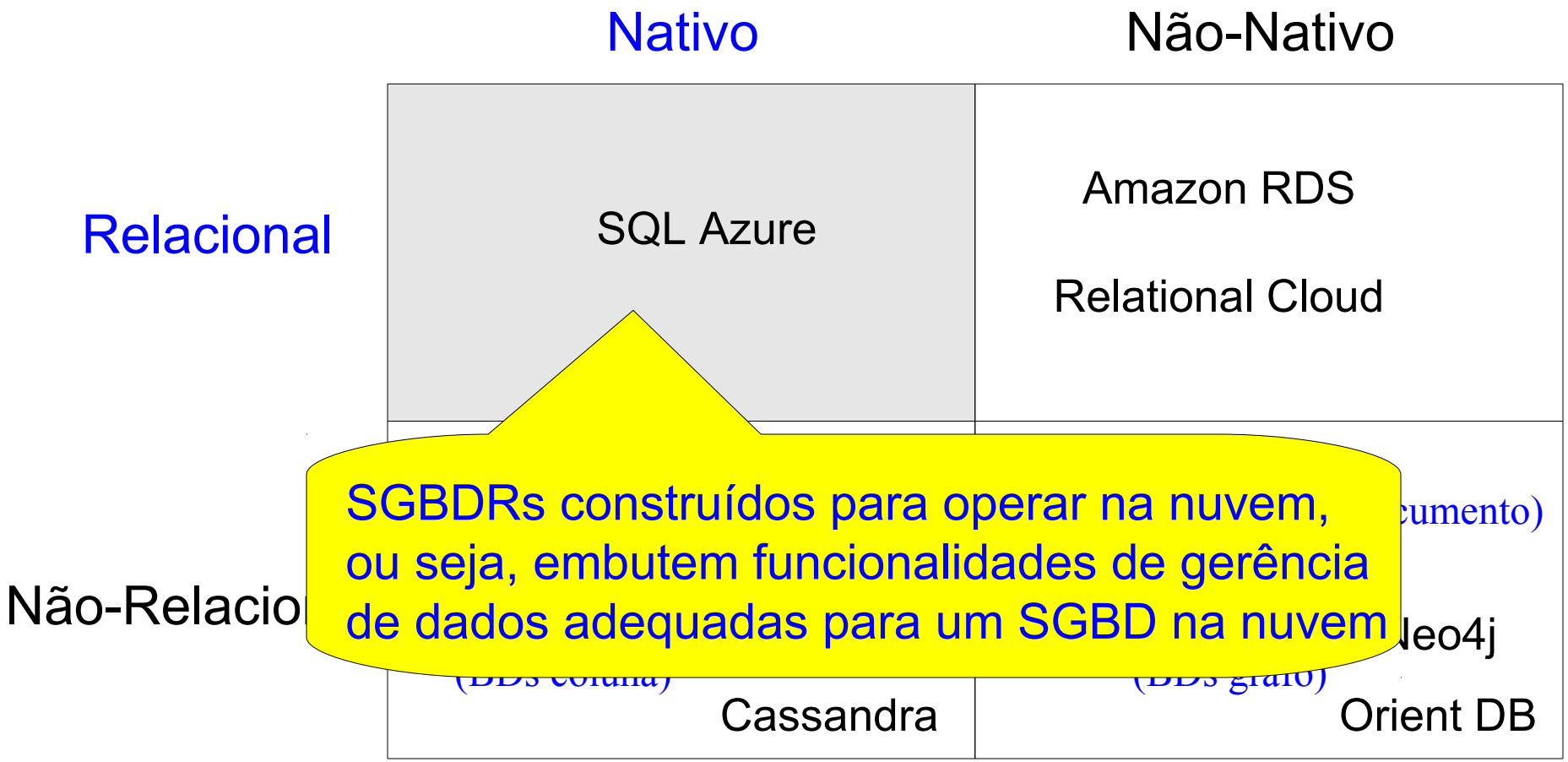
Tópicos

1. Introdução à Computação nas Nuvens
2. Gerência de Dados na Nuvem
3. Categorias de BDs na Nuvem
4. BDs No-SQL
5. Considerações Finais

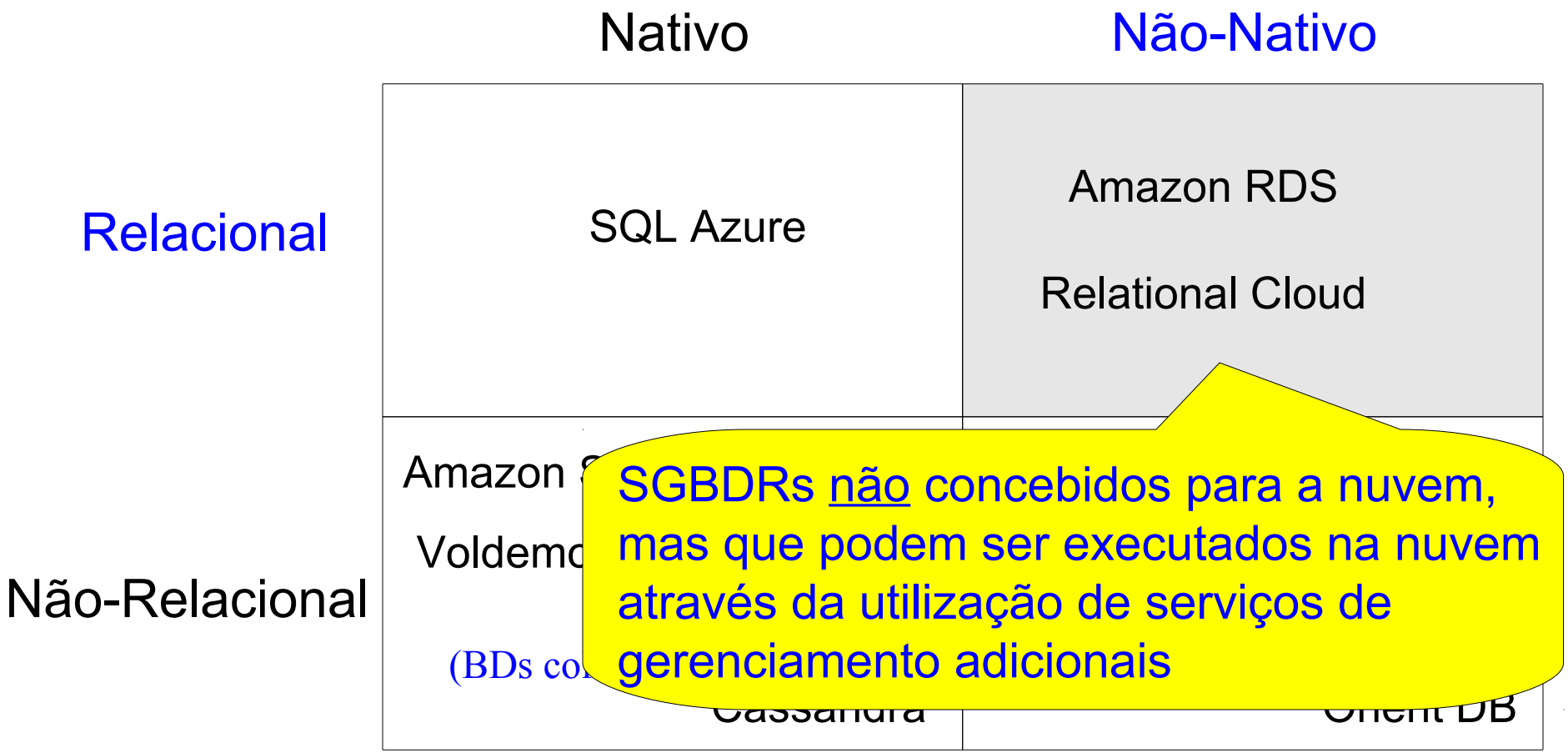
Categorias de BDs na Nuvem

	Nativo	Não-Nativo
Relacional	SQL Azure	Amazon RDS Relational Cloud
Não-Relacional	Amazon S3 Voldemort (BDs chave-valor) HBase Cassandra (BDs coluna)	Couch DB Mongo DB (BDs documento) Neo4j Orient DB (BDs grafo)

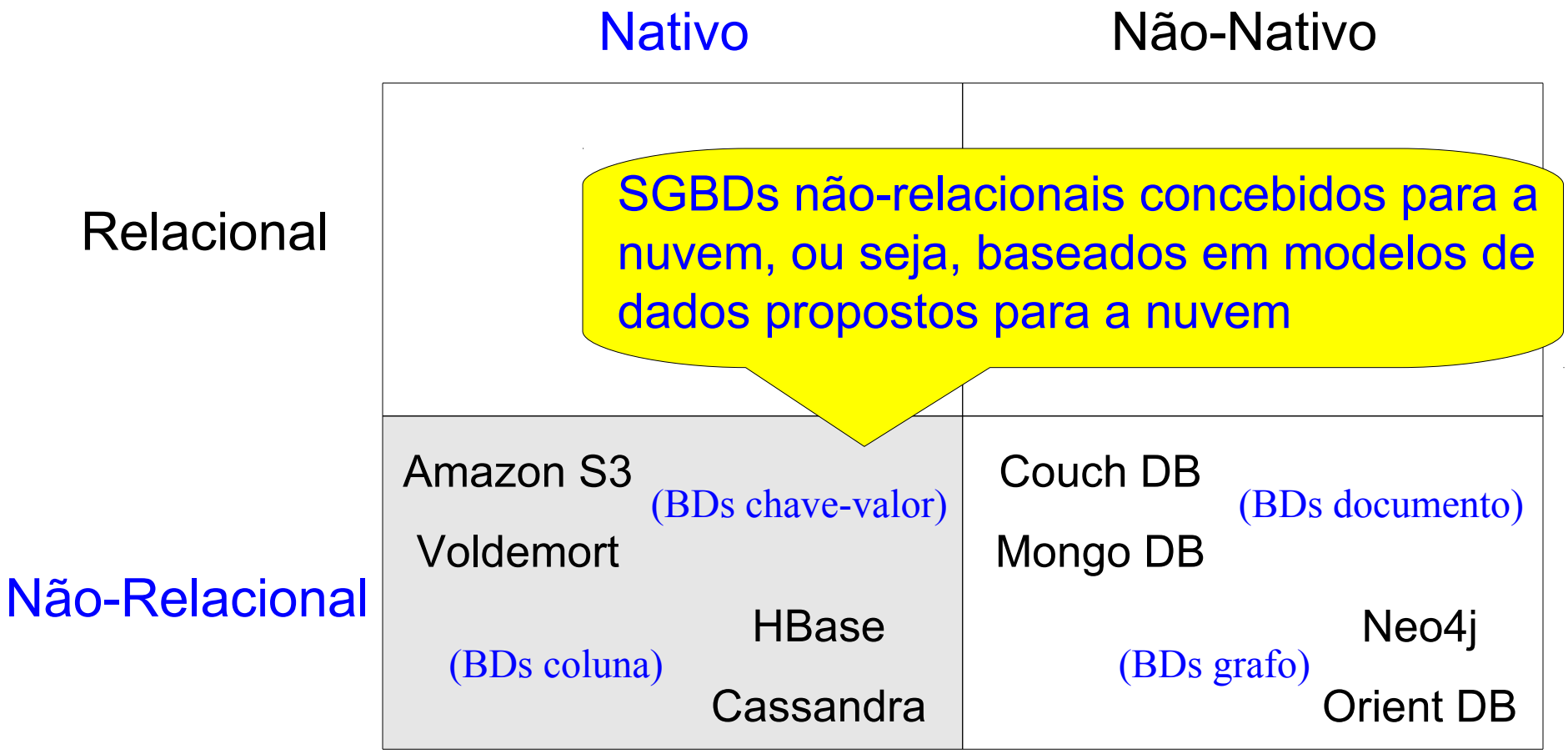
Categorias de BDs na Nuvem



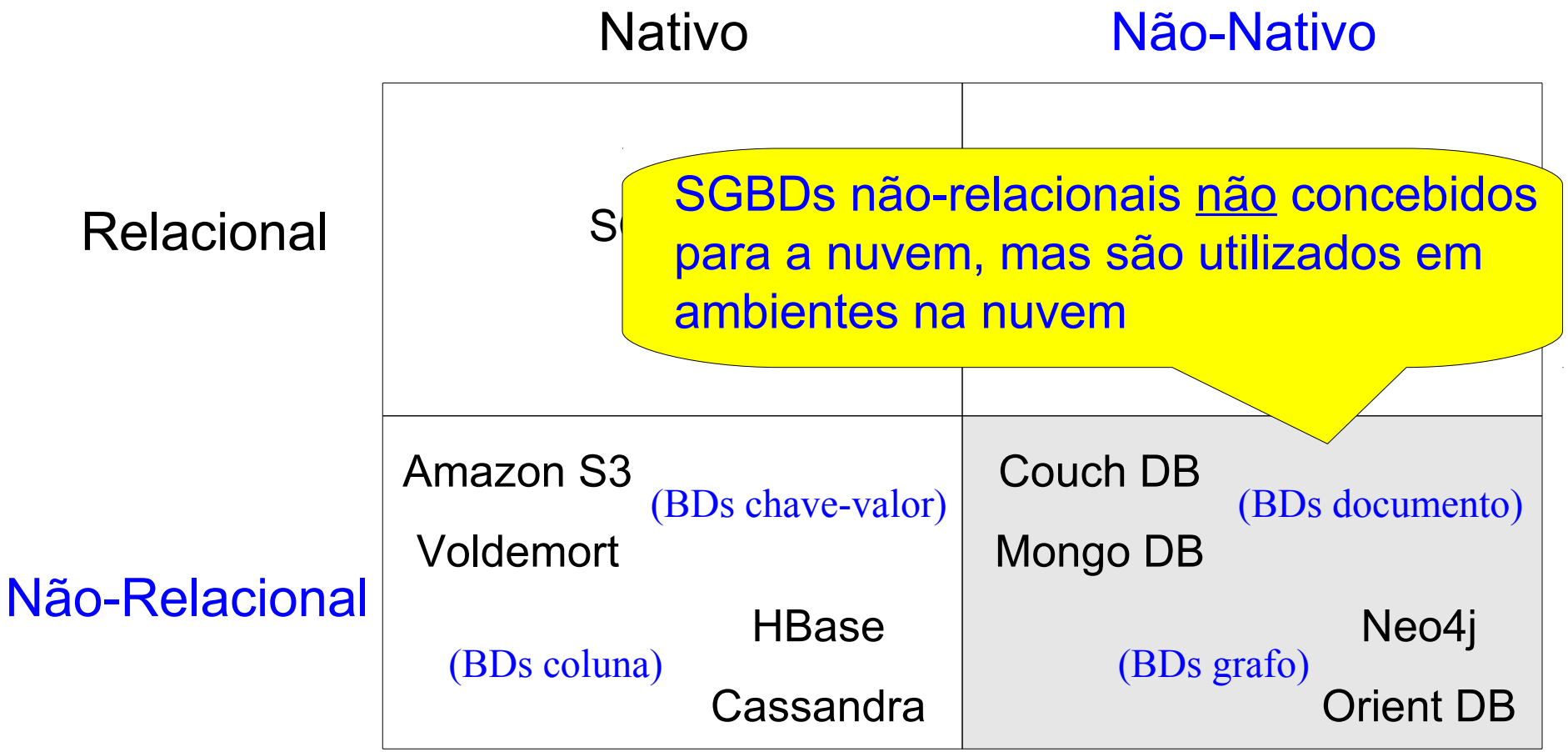
Categorias de BDs na Nuvem



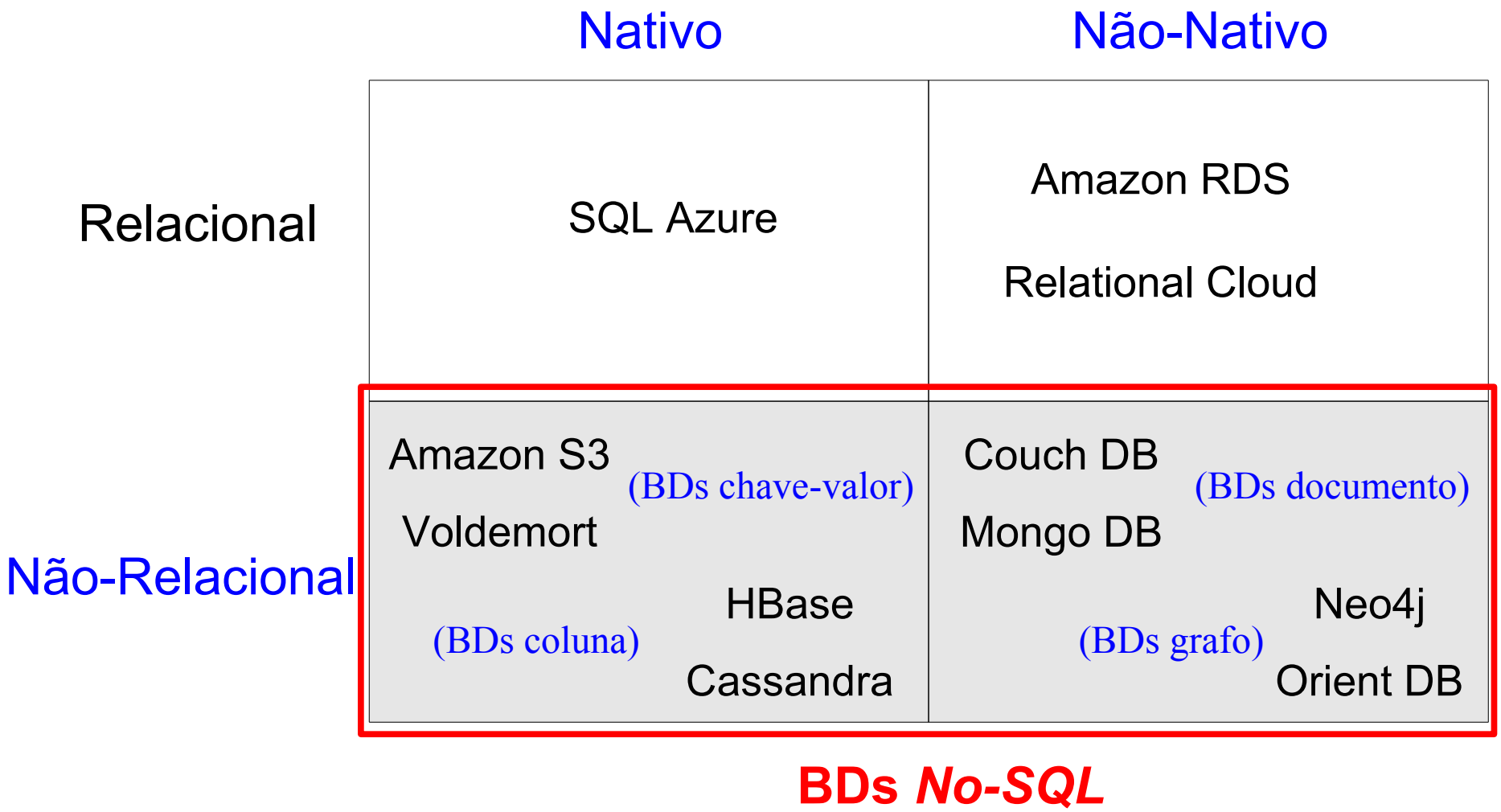
Categorias de BDs na Nuvem



Categorias de BDs na Nuvem



Categorias de BDs na Nuvem



Tópicos

1. Introdução à Computação nas Nuvens
2. Gerência de Dados na Nuvem
3. Categorias de BDs na Nuvem
4. BDs No-SQL
5. Considerações Finais

No-SQL (Not Only SQL)

- Movimento a favor do desenvolvimento de SGBDs não-relacionais para a gerência de dados na nuvem
- Principais Características (foco)
  - Grande volume de dados
  - Consistência fraca
  - Estruturas de armazenamento/acesso simples
    - Interfaces de acesso simples
- Principais aplicações
  - *Social networks*, *e-commerce*, *Web search engines*

BDs No-SQL

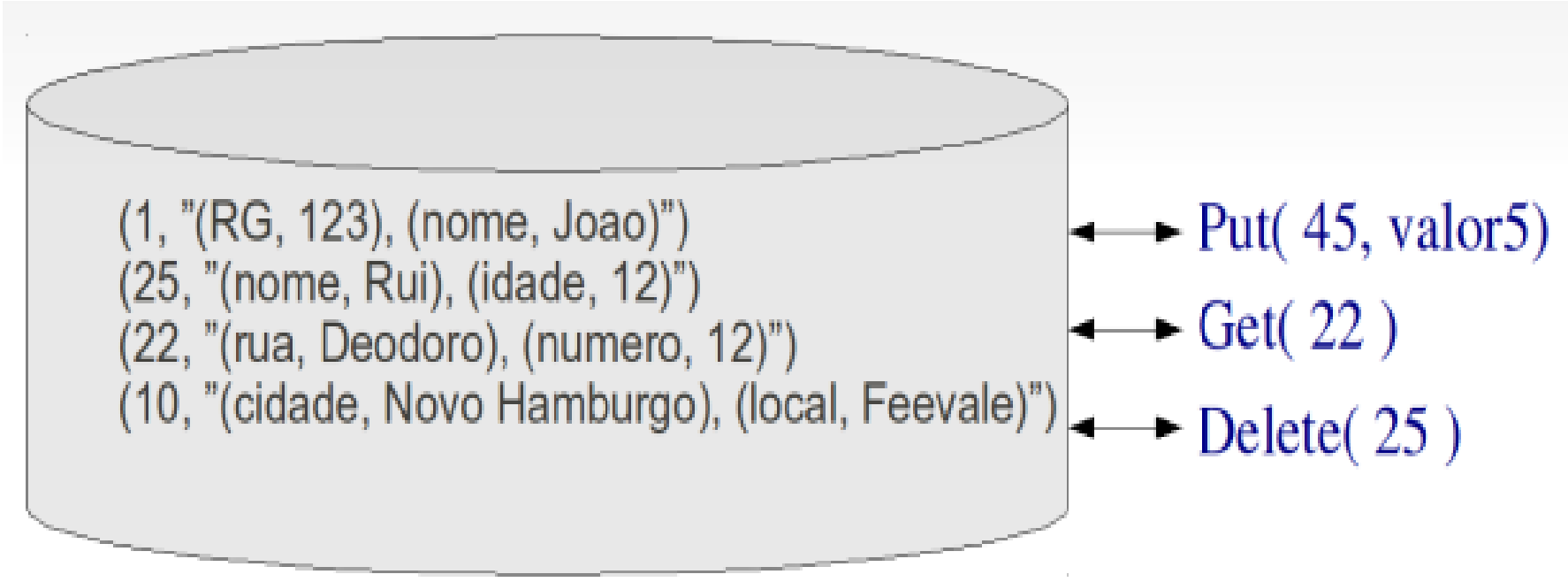
- Principais abordagens
  - BD chave-valor
    - Exemplo: Amazon S3
  - BD de coluna
    - Exemplo: Cassandra
  - BD de documentos
    - Exemplo: CouchDB
  - BD de grafo
    - Exemplo: Neo4j

Alta heterogeneidade em termos de modelos de dados!

Amazon S3 (Simple Storage Service)

- Desenvolvido pela *Amazon.com Inc.*
- Modelo chave-valor
  - Um valor de chave remete a um conjunto de outros valores associados
  - API simples: *get(key), put(key, value)* e *delete(key)*
- Não suporta
  - Definição de esquemas
  - Relacionamentos entre dados
  - Linguagem de consulta

Modelo Chave-Valor



Outro exemplo:

(“Cloud Databases”, “(ISBN, XXX), (edição, 4), ...”)

Cassandra

- Criado pelo Facebook e mantido atualmente pela Apache Foundation
  - Utilizado pelo *eBay* e *Twitter*
- Suportes
  - Definição de esquemas
  - API proprietária
  - Linguagem de consulta
- Não suporta
  - Definição de relacionamentos entre dados

Cassandra – Modelo de Dados

- Modelo de coluna
  - Noção de família de colunas (*column family*)
    - Lista ordenada de colunas
    - Indexada por uma chave
    - *Keyspace*
      - Conjunto de família de colunas
- Modelo mais rígido que o chave-valor
  - Organiza os dados em domínios (*keyspaces*) e classes de dados com os mesmos atributos (*column families*)
    - *Keyspace* ≡ *BD* e *Column family* ≡ *tabela* em um BDR
  - Permite super-colunas
    - colunas compostas por outras colunas

Modelo baseado em Coluna

Keyspace: Livraria

Column Family: Autores

Key	Value	
Emmanuel Silva	Name	Value
	Email	emmanuel@email.com
	Biografia	Biografia do autor aqui...
José Raimundo	Name	Value
	Email	raimundo@gmail.com
	Biografia	Bla bla bla

Column Family: Livros

Key	Value	
Aprendendo Java	Name	Value
	ISQN	181919191
	Ano de publicação	2007
	Autor	Emmanuel Silva

API Thrift

- Modificação de dados
  - Inclusão/exclusão de uma ou mais colunas
  - Exemplos
    - *insert(), batch_insert()* (inclui várias colunas), *remove()*, ...
- Consulta a dados
  - Busca nomes de colunas, seus conteúdos (incluindo sub-colunas) a partir de uma chave ou de uma faixa de chaves
  - Exemplos
    - *get()* (retorna colunas a partir de 1 chave), *multiget()* (várias chaves), *get_slice()* (faixa de chaves), ...

CQL (Cassandra Query Language)

- DDL
  - Criação de *Keyspaces*
  - Criação de famílias de colunas para *keyspaces*
- DML
  - Inserção/atualização de valores de colunas de uma família de colunas, dada uma chave
  - Remoção de colunas ou valores de colunas de uma família de colunas, dada uma chave
  - Consulta a valores de colunas de famílias de colunas, com predicados de seleção simples

CQL - Exemplo

Keyspace: Livraria

Column Family: Autores

Key	Value	
Emmanuel Silva	Name	Value
	Email	emmanuel@email.com
	Biografia	Biografia do autor aqui...
José Raimundo	Name	Value
	Email	raimundo@gmail.com
	Biografia	Bla bla bla

Column Family: Livros

Key	Value	
Aprendendo Java	Name	Value
	ISQN	181919191
	Ano de publicação	2007
	Autor	Emmanuel Silva

```
CREATE KEYSPACE Livraria
WITH strategy_class = SimpleStrategy
AND strategy_options:replication_factor = 3;

USE Livraria;

/* ainda não há suporte para inserção de
super-colunas */

CREATE COLUMNFAMILY Autores
(Key text PRIMARY KEY,
 Email text, Biografia text);

CREATE COLUMNFAMILY Livros
(Key text PRIMARY KEY,
 ISQN bigint,
 Ano int,
 Autor text);
WHERE Ano > 2000;
```

CQL - Exemplo

Keyspace: Livraria

Column Family: Autores

Key	Value
Emmanuel Silva	Name: Email — emmanuel@email.com Name: Biografia — Biografia do autor aqui...
José Raimundo	Name: Email — raimundo@gmail.com Name: Biografia — Bla bla bla

Column Family: Livros

Key	Value
Aprendendo Java	Name: ISQN — 181919191 Name: Ano de publicação — 2007 Name: Autor — Emmanuel Silva

```
INSERT INTO Autores (Key, Email, Biografia)
VALUES (Emmanuel Silva,
'emmanuel@email.com',
'Biografia do autor aqui');

INSERT INTO Livros (Key, ISQN, Ano, Autor)
VALUES (181919191, 2007, 'Emmanuel Silva');

UPDATE Autores
SET Email = 'Esilva@gmail.com'
WHERE KEY = Emmanuel Silva;

DELETE COLUMNS ISQN
 FROM Livros WHERE KEY = Aprendendo Java;

SELECT FIRST 10 REVERSED Ano, Autor
FROM Livros
WHERE Ano > 2000;
```

Couch DB

- Desenvolvido pela IBM
  - Mantido atualmente pela Apache
- Não suporta
  - Definição de esquemas
  - Relacionamentos entre dados
  - Linguagem de consulta
- Modelo de dados baseado em documento
  - Noção de um objeto complexo (BDOO)
    - Atributos com domínios simples ou complexos (listas ou registros)
  - Armazenamento no formato de objetos Java JSON

Modelo Baseado em Documento

```
{ "_id":"discussion_tables",
 "_rev":"D1C946B7",
 "Sunrise":true,
 "Sunset":false,
 "FullHours":[1,2,3,4,5,6,7,8,9,10],
 "Activities": [
   {"Name":"Football", "Duration":2, "DurationUnit":"Hours"},
   {"Name":"Breakfast", "Duration":40, "DurationUnit":"Minutes",
         "Attendees":["Jan", "Damien", "Laura", "Gwendolyn", "Roseanna"]} ]      }
```

Atributos *default*

```
{
 "_id":"some_doc_id",
 "_rev":"D1C946B7",
 "Subject":"I like Plankton",
 "Author":"Rusty",
 "PostedDate":"2006-08-15T17:30:12-04:00",
 "Tags":["plankton", "baseball", "decisions"],
 "Body":"I decided today that I don't like baseball. I like plankton."                }
```

Objetos complexos JSON

Acesso a Dados

- API proprietária
  - Acesso via aplicação ou HTTP
  - Unidade de atualização: documento
  - Recuperação de documentos completos
    - filtros apenas por atributo(s) pré-definidos
      - ID do doc, ID da revisão, ...
- Principais operações
  - *Get* (busca)
  - *Post* (similar a um *insert* com ID gerado pelo SGBD)
  - *Delete*

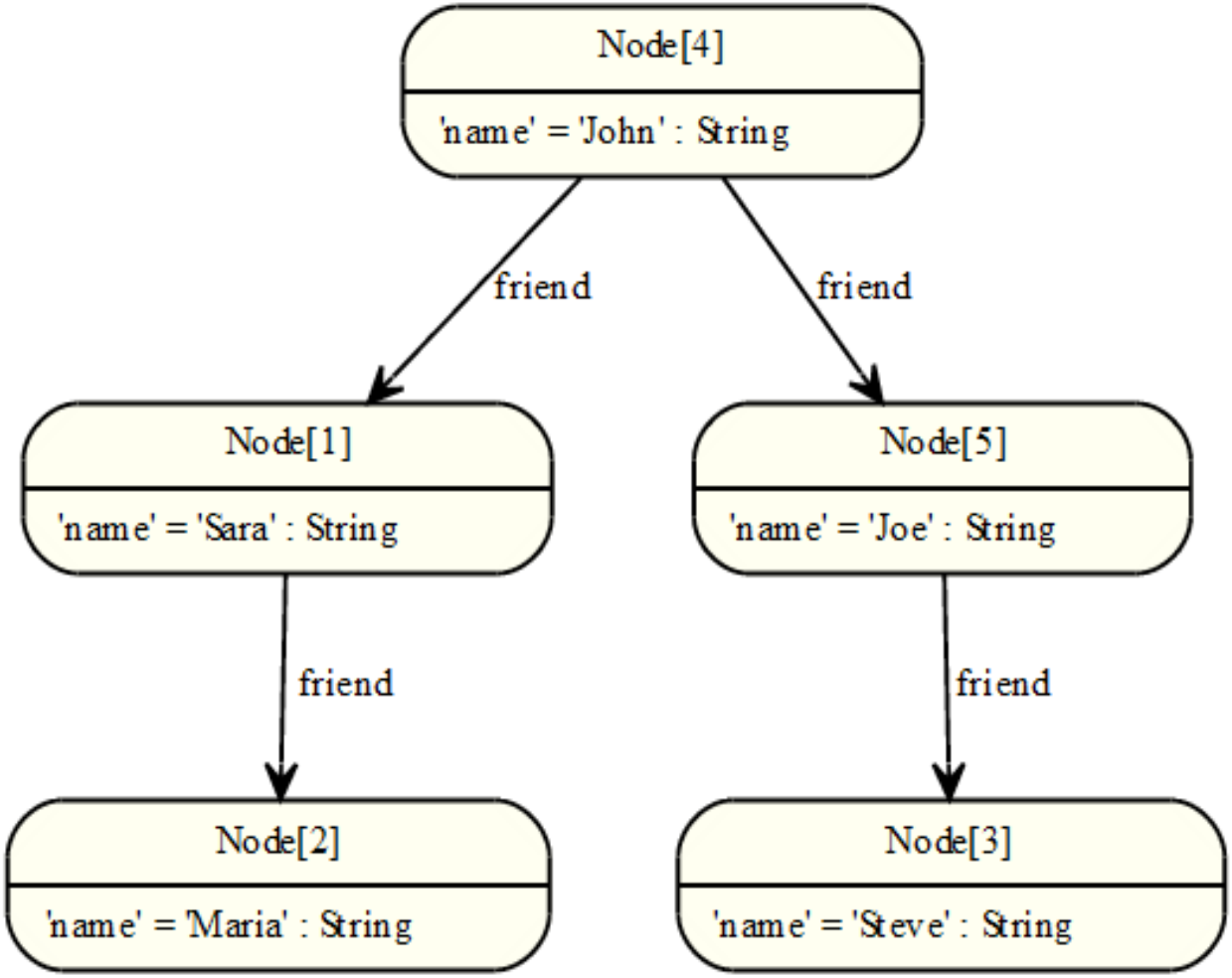
API - Exemplos

```
/* busca por ID do documento */
GET http://mydatabase:5984/discussion_tables

/* busca 10 documentos a partir de IDs que iniciam com 'doc2' */
GET http://mydatabase:5984/_all_docs?startkey="doc2"&limit=10

/* inserção de novo documento */
POST http://mydatabase:5984{
  "Subject":"I like Plankton",
  "Author":"Rusty",
  "PostedDate":"2006-08-15T17:30:12-04:00",
  "Tags":["plankton", "baseball", "decisions"],
  "Body":"I decided today that I don't like baseball. I like
plankton."}

/* remove documentos com um certo ID de revisão */
DELETE http://mydatabase:5984/some_doc?rev=1582603387
```

Neo4j

- Desenvolvido pela Neo Technology
- Não suporta definição de esquemas
- Suporte opcional a transações, *deadlock* e bloqueios
- Modelo de dados de grafo
  - Nodos
    - Itens de dados com ID e atributos
  - Arestas direcionadas e rotuladas
    - Relacionamentos entre dados
    - Podem ter atributos
  - Tipos de atributos
    - Simples ou *arrays* de tipos simples

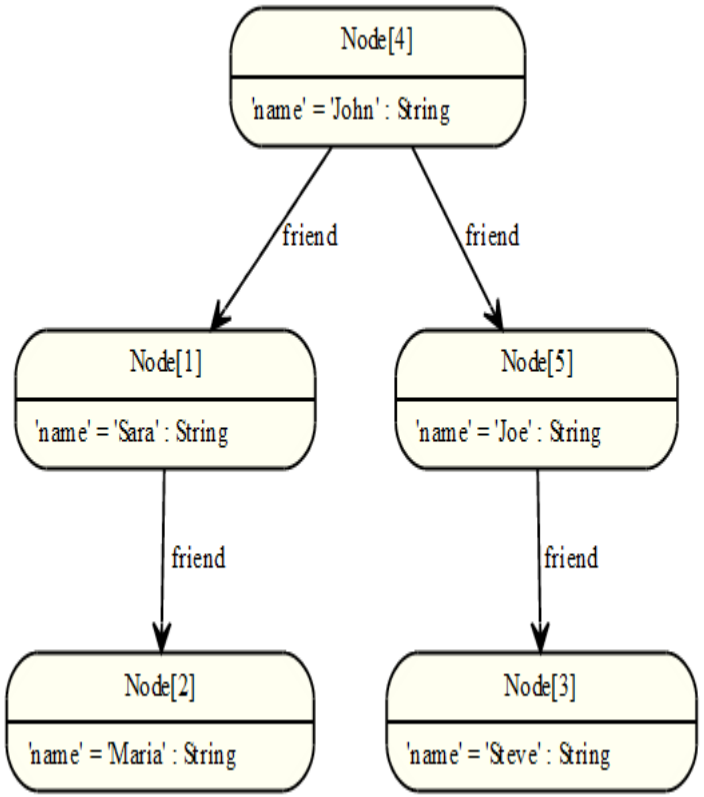
Modelo Baseado em Grafo



Acesso a Dados

- API Rest
  - Suporta acesso via HTTP
  - Principais funcionalidades
    - I/E de nodos e de relacionamentos
    - I/A/E de atributos
    - Navegação em grafos
      - Indicação do nodo de partida, critério de parada, restrições (filtros) para visita a nodos, tipo de busca (largura ou profundidade), ...

API REST - Exemplos

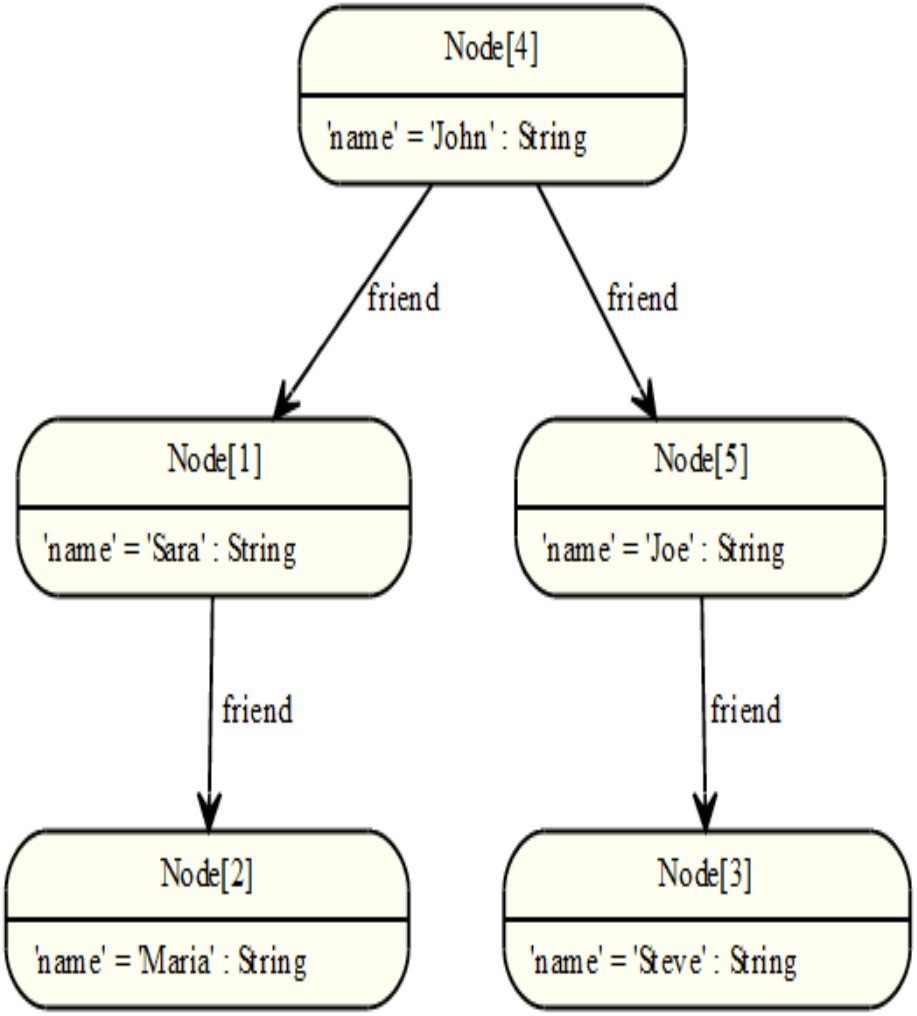



```
/* Cria 2 nodos com atributos e 1 aresta */
URI firstNode = createNode();
addProperty( firstNode, "name", "John");
URI secondNode = createNode();
addProperty( secondNode, "name", "Sara");
URI relation = addRelationship( firstNode,
secondNode, "friend");

/* Varredura em profundidade, pesquisando uma
única vez os atributos dos nodos ligados pelo
relacionamento friend */
...
TraversalDescription t = new
TraversalDescription();
t.setOrder(TraversalDescription.DEPTH_FIRST);
t.setUniqueness(TraversalDescription.NODE);
t.setReturnFilter(TraversalDescription.ALL);
t.setRelationships(new Relationship( "friend",
Relationship.OUT));
...
```

Cypher Query Language

- Linguagem de consulta do Neo4j
- Sintaxe básica
  - Bloco START-MATCH-WHERE-RETURN
    - START
      - Nodos alvo da consulta
    - MATCH
      - Casamentos a serem feitos
    - WHERE
      - Filtros
    - RETURN
      - Resultado da consulta

Cypher - Exemplo




```
START john=node:node_auto_index(name = 'John')
MATCH john-[:friend]->()-[:friend]->fof
RETURN john, fof
```

Resultado:

john	fof
Node[4]{name->"John"}	Node[2]{name->"Maria"}
Node[4]{name->"John"}	Node[3]{name->"Steve"}
2 rows, 2 ms	

Tópicos

1. Introdução à Computação nas Nuvens
2. Gerência de Dados na Nuvem
3. Categorias de BDs na Nuvem
4. BDs No-SQL
5. Considerações Finais

BDs nas Nuvens

- Gerência de grandes volumes de dados
  - Adequado a vários tipos de aplicações
    - Dados na Web, dados de sensores, dados científicos, …
- Desafios associados com a garantia da qualidade desta gerência *on demand*
  - Aumento/melhoria de infraestrutura, migração de dados/serviços para nodos com maior capacidade de processamento, particionamento de dados, ...

Tendências de Pesquisa

- Aprimoramento de gerências
  - Manipulação de dados mais robusta
    - DMLs, suporte a joins, ...
  - Consistência de transações
- Avaliação de desempenho na nuvem
  - Determinar modelos/gerenciadores mais adequados para certos domínios e certos volumes de dados
- Acesso relacional a dados na nuvem
  - Permitir acesso a dados na nuvem através de visões relacionais
    - Facilitar o acesso para aplicações baseadas em BDRs
    - Garantir o menor *overhead* possível no mapeamento relacional-*cloud*
- Integração/Interoperabilidade
  - *Matching* de esquemas/dados heterogêneos, visando consultas a várias fontes
- Metodologias de Projeto de BDs NoSQL