

Bases de Dados NoSQL

Catarina Ferreira da Silva, Pedro Ramos

ISCTE-IUL

Origem

- “NoSQL” – Not Only SQL, ou seja, para além do modelo relacional
- Representa um conjunto de sistemas de gestão de bases de dados que visa responder a um conjunto de necessidades mal resolvidas pelo modelo relacional, nomeadamente
 - Flexibilidade
 - Escalonamento

Flexibilidade

O aumento dos “produtores” e quantidade de dados (por exemplo, Internet of Things) conduz

- ao surgimento de novas soluções de armazenamento
- à necessidade de rapidamente (automaticamente) incorporar nas bases de dados estruturas de informação que não estavam previamente definidas



Flexibilidade

Tem de ser possível adicionar dados de novos sensores sem que a aplicação seja interrompida ou reformulada



Estrutura de dados mais flexível

- As tabelas são substituídas por coleções JSON em MongoDB e as linhas são os documentos

```
{  
    "_id" : ObjectId("5f9fcfc2d76f62eef159c04"),  
    "IDLIVRO" : 4,  
    "Titulo" : "A voz da foz",  
    "Editora" : "Relógio daqui",  
    "Autoria" : {  
        "IDAutor" : "Amarante",  
        "Nome" : "Ana Amarante",  
        "Nacionalidade" : "Portuguesa"  
    }  
}  
  
{  
    "_id" : ObjectId("5f9fcfc2d76f62eef159c0a"),  
    "IDLIVRO" : 15,  
    "Titulo" : "Do tempo de agora",  
    "Editora" : "Bertrand",  
    "Autoria" : {  
        "IDAutor" : "Amarante",  
        "Nome" : "Ana Amarante",  
        "Nacionalidade" : "Portuguesa"  
    }  
}
```

Redundância

Bases dados orientadas a documentos

Podem coexistir diferentes formas de representação

Privilegia consulta por autor

```
{  
    IDAutor: "Amarante",  
    Nome: "Ana Amarante",  
    Nacionalidade: "Portuguesa",  
    Autoria:  
    [  
        {  
            IDlivro: 4,  
            Titulo: "A voz da foz",  
            Editora: "Relógio daqui"  
        },  
        {  
            IDlivro: 15,  
            Titulo: "Do tempo de agora",  
            Editora: "Bertrand"  
        }  
    ]  
}
```

Privilegia consulta por livro

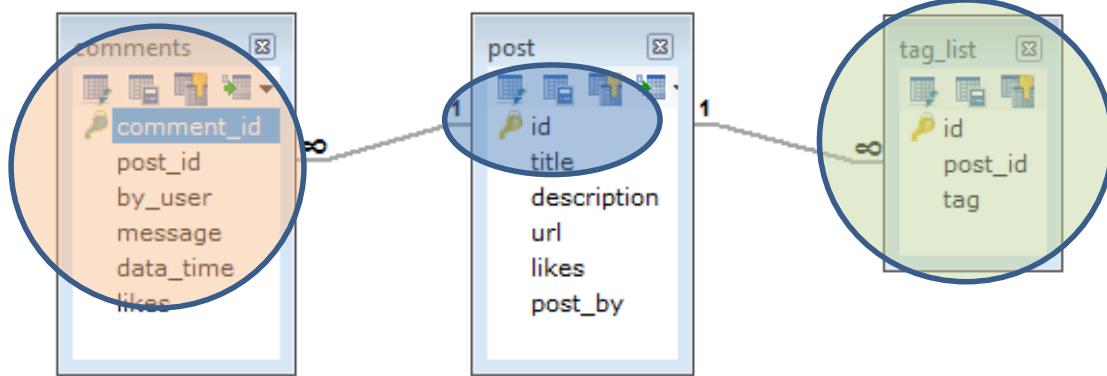
```
{  
    IDlivro: 4,  
    Titulo: "A voz da foz",  
    Editora: "Relógio daqui",  
    Autoria: {  
        IDAutor: "Amarante",  
        Nome: "Ana Amarante",  
        Nacionalidade: "Portuguesa"  
    }  
    {  
        IDlivro: 15,  
        Titulo: "Do tempo de agora",  
        Editora: "Bertrand",  
        Autoria: {  
            IDAutor: "Amarante",  
            Nome: "Ana Amarante",  
            Nacionalidade: "Portuguesa"  
        }  
    }  
}
```

Flexibilidade

Uma colecção versus 3 tabelas.
Vantagens? Desvantagens?

Mais rápido a consultar *posts*

Mais rápido a contar comentários



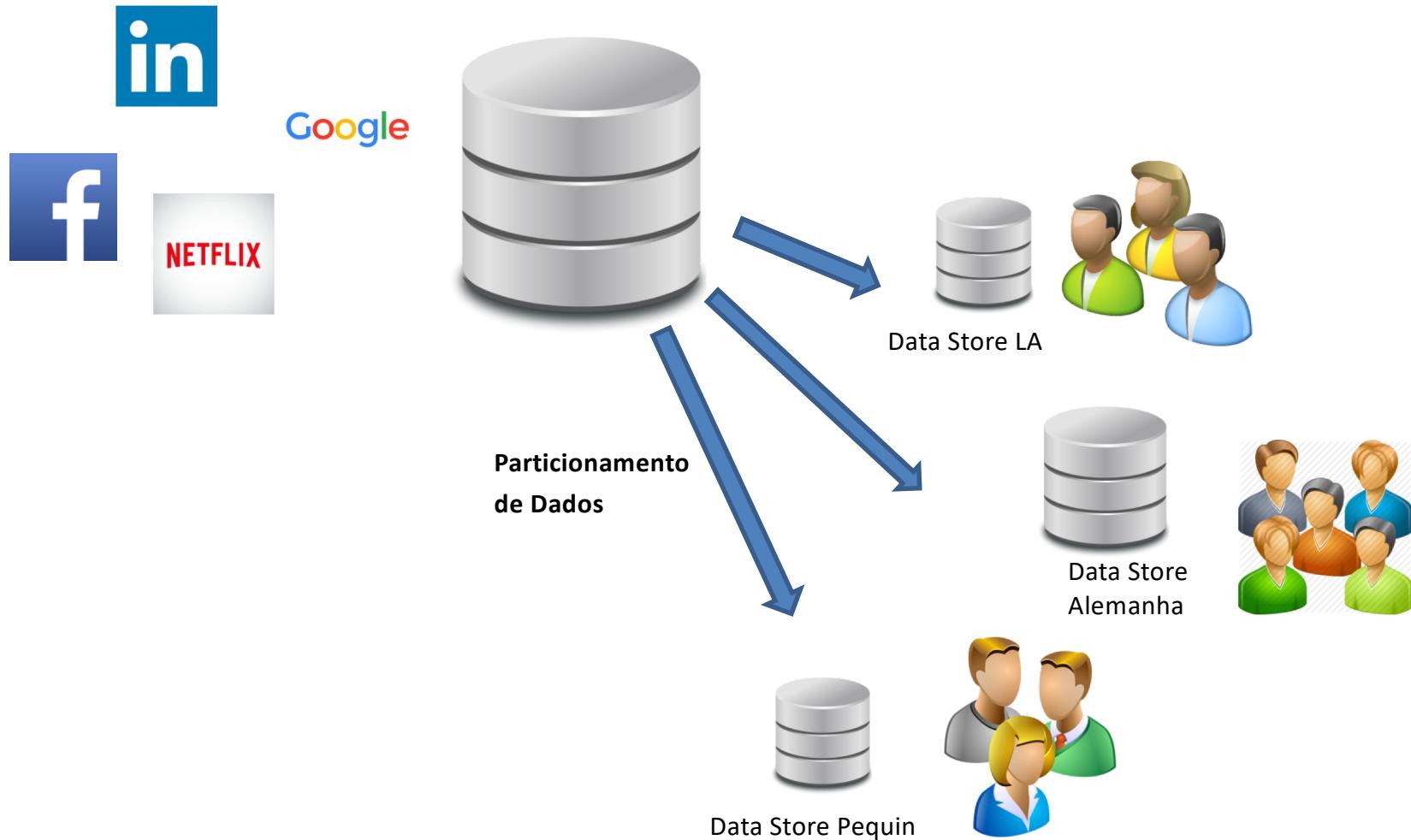
```
{  
  _id: POST_ID  
  title: TITLE_OF_POST,  
  description: POST_DESCRIPTION,  
  by: POST_BY,  
  url: URL_OF_POST,  
  tags: [TAG1, TAG2, TAG3],  
  likes: TOTAL_LIKES,  
  comments: [  
    {  
      user: 'COMMENT_BY',  
      message: TEXT,  
      dateCreated: DATE_TIME,  
      like: LIKES  
    },  
    {  
      user: 'COMMENT_BY',  
      message: TEXT,  
      dateCreated: DATE_TIME,  
      like: LIKES  
    }  
  ]  
}
```

Privilegiar as Consultas e queries pouco complexas

Escalonamento

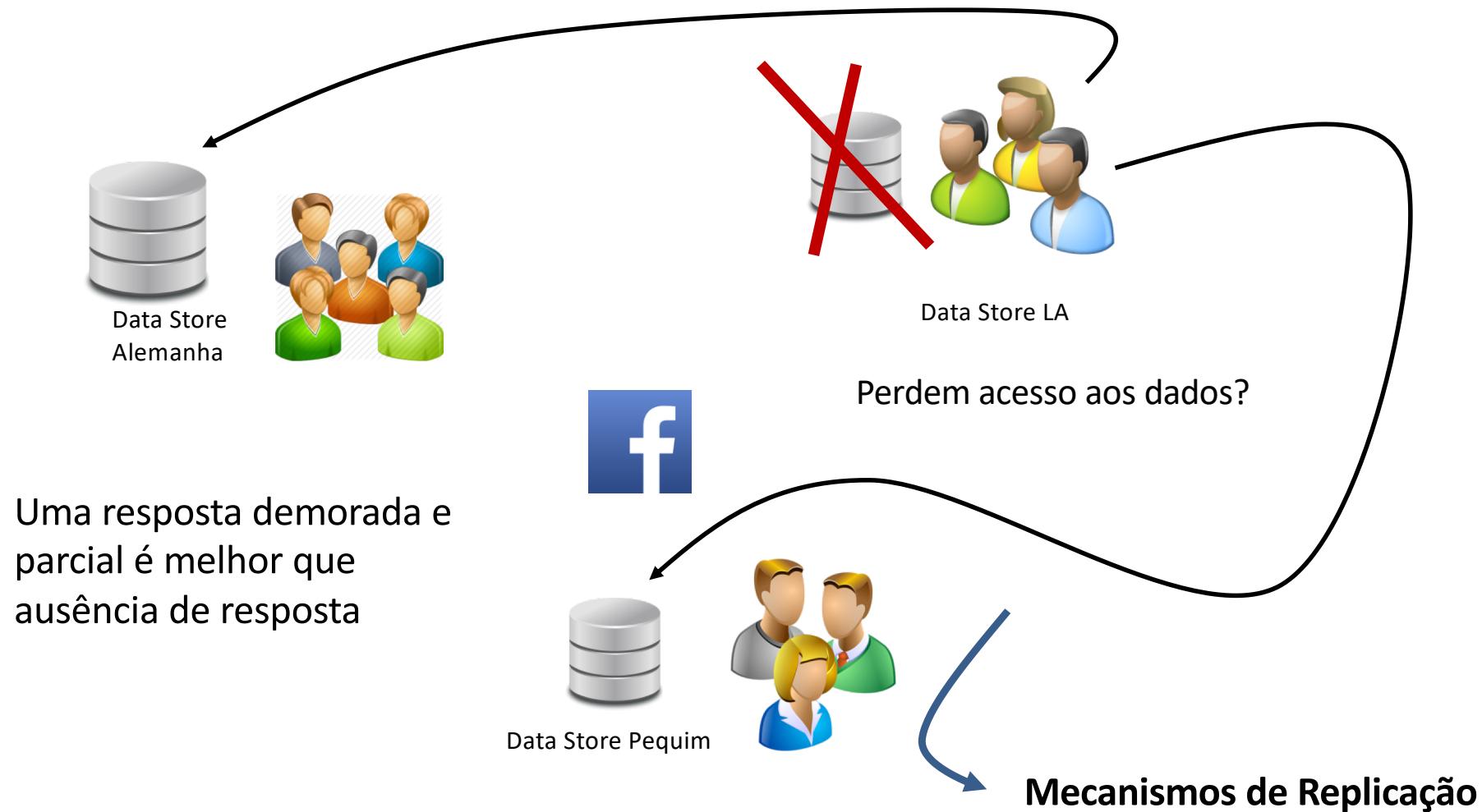


Particionamento



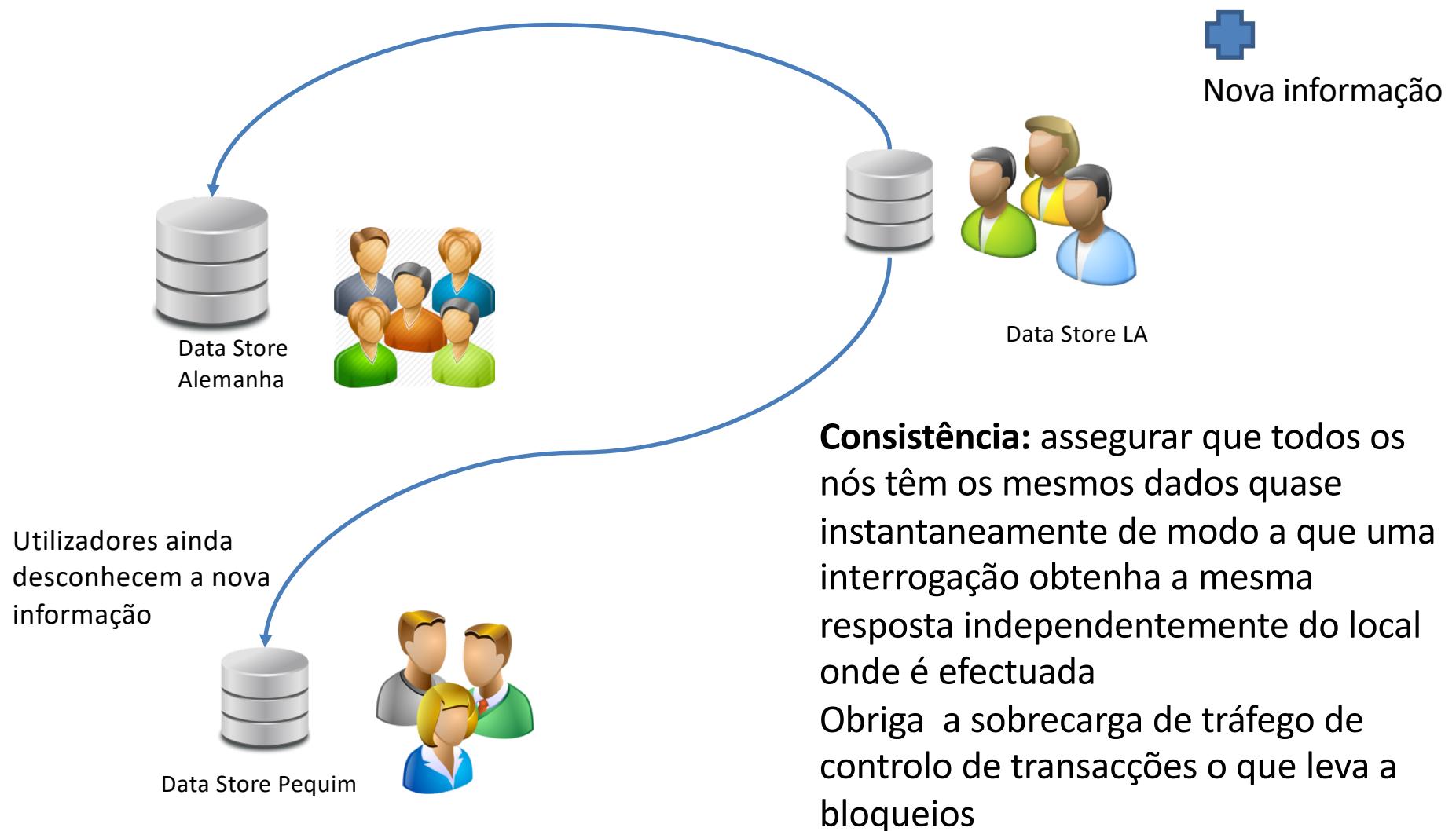
Particionamento

E se ocorrer uma quebra na distribuição de dados, será que os que estão próximos do centro de dados ficam sem resposta?

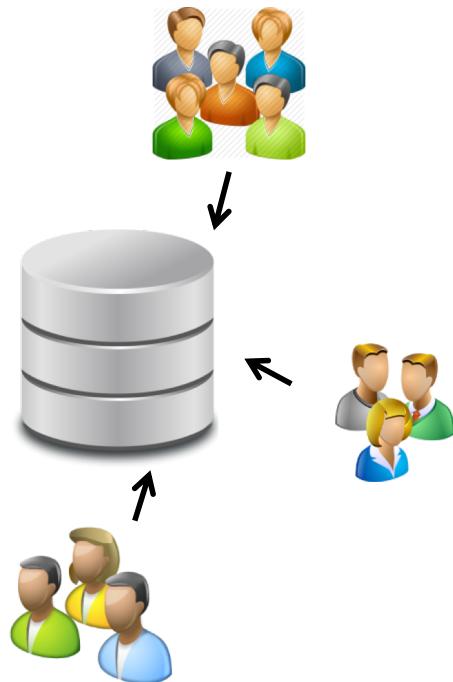


Consistência versus Eficiência

Mecanismos de replicação entre bases de dados é **pouco eficiente** se for **consistente**



Desafios das BDs Relacionais



As bases de dados relacionais têm de assegurar quatro propriedades no que diz respeito às suas transacções (**ACID**):

1. **Atomicidade.** “Tudo ou nada”. Não é feito nenhum commit se a transacção não chegar ao fim
2. **Consistência (sequencial).** Qualquer resposta tem de devolver dados consistentes e o resultado do mais recente commit
3. **Isolamento.** Cada transacção é executada como se fosse a única activa na base de dados
4. **Durabilidade.** Os dados depois de armazenados tornam-se permanentes

No modelo relacional distribuído cada nó da rede só pode ver a informação depois de terminada a escrita em todos os nós

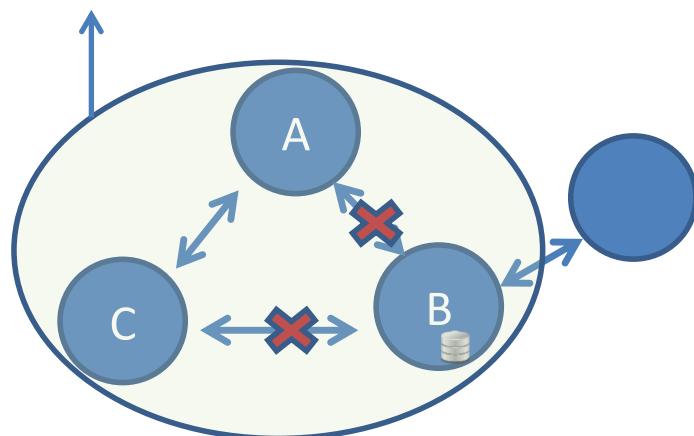
Enquanto que as BD NoSQL assumem um cenário mais flexível em que a consistência pode ser controladamente descurada

Desafios das BDs NoSQL

A consistência da informação entre os nós não é a principal propriedade a manter

- O mais importante é assumir que vão existir falhas (partições) de comunicação de rede (mensagens perdidas), mas apesar dessas falhas o sistema continua a dar “resposta”
- Em caso de falhas o sistema não se “desliga” e o processamento prossegue nas partições disponíveis

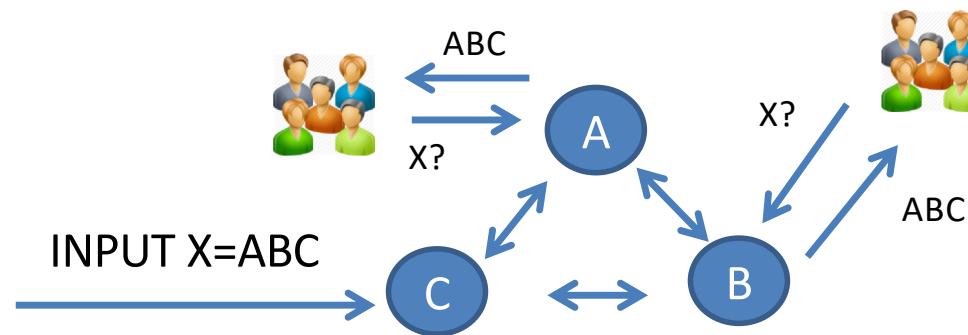
Cluster de 3 nós



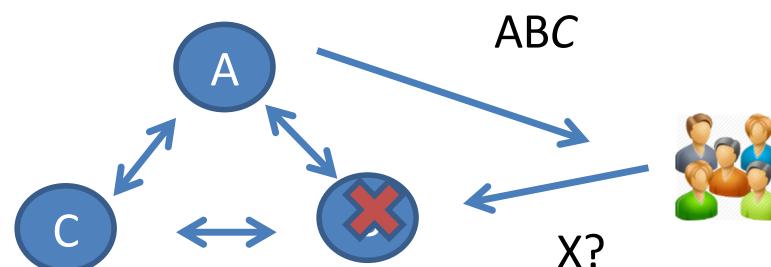
- Apesar do nó B estar desligado da rede ou com comunicações lentas (isto é, estar isolado), internamente, ele mantém as propriedades e comportamento
- Quando o nó B estiver de novo disponível, os nós voltam a sincronizar, e a informação volta a ser coerente após algum tempo

Desafios das BDs NoSQL

Uma propriedade fundamental é a **Consistência (distribuída)**: assegurar que todos os nós têm de ter os mesmos dados

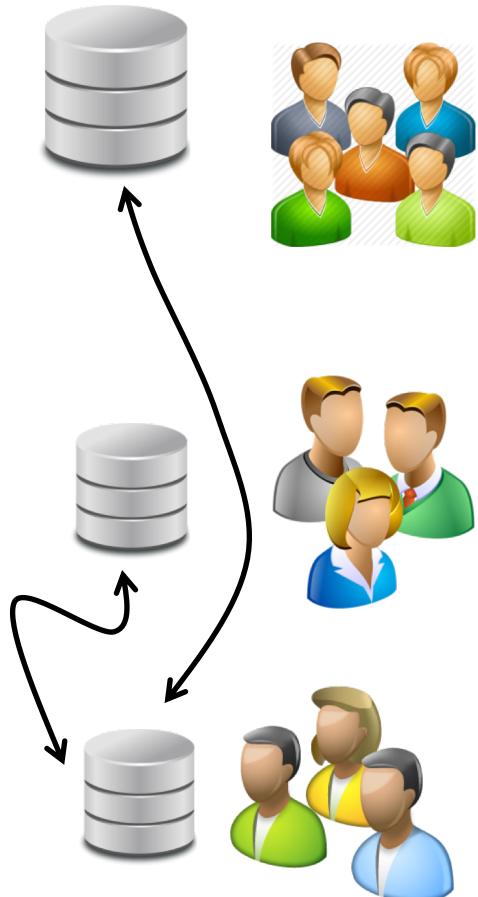


A **Disponibilidade** do sistema (os acessos nunca devolvem erros ou bloqueiam) é um requisito nos sistemas distribuídos. O Google não pode estar off



Teorema CAP

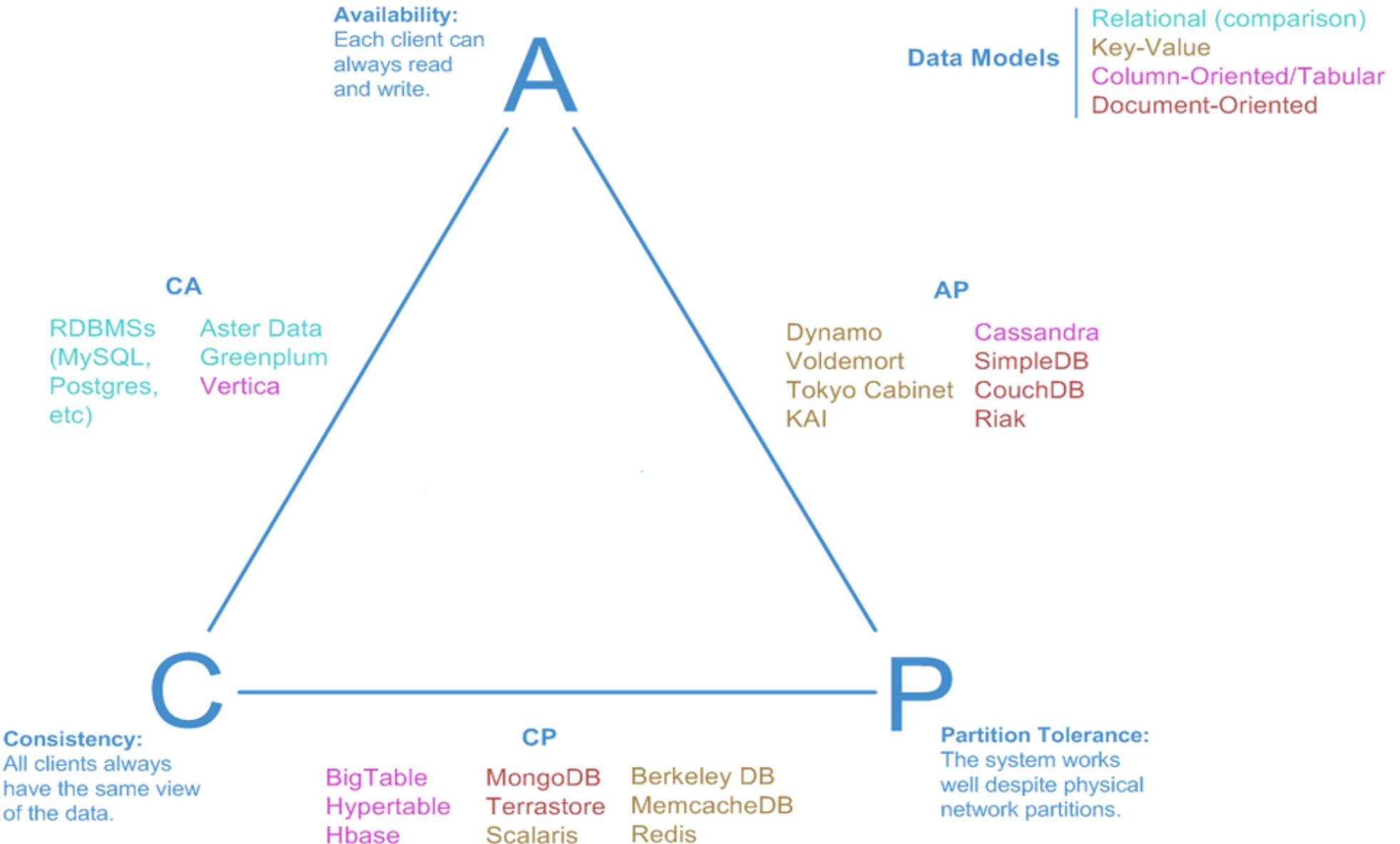
Teorema: Um sistema distribuído tolerante a partições (não se desliga se um nó fica inacessível) não consegue manter simultaneamente a Consistência e a Disponibilidade



Num sistema distribuído, quando ocorre uma quebra de rede (partição), tem de se optar entre

- O sistema continuar a responder possivelmente com informação incoerente (disponibilidade, sem consistência) ou
- O sistema deixar de responder (consistência, mas nem sempre disponível)

Triângulo CAP



Síntese CAP

As Base de Dados Nosql deixam “cair” as pesquisas fáceis de informação - “joins” (exigem informação mais estruturada) em favor de estruturas de dados flexíveis (JSON) mais eficientes

Deixam “cair” o ACID (nomeadamente consistência / sincronismo tempo real) em favor da disponibilidade dos dados. Assegurar uma **Consistência Eventual** (janela temporal de inconsistência), isto é, sincronização offline de cópias locais em que, quando as alterações terminarem, eventualmente as cópias convergirão para o mesmo valor

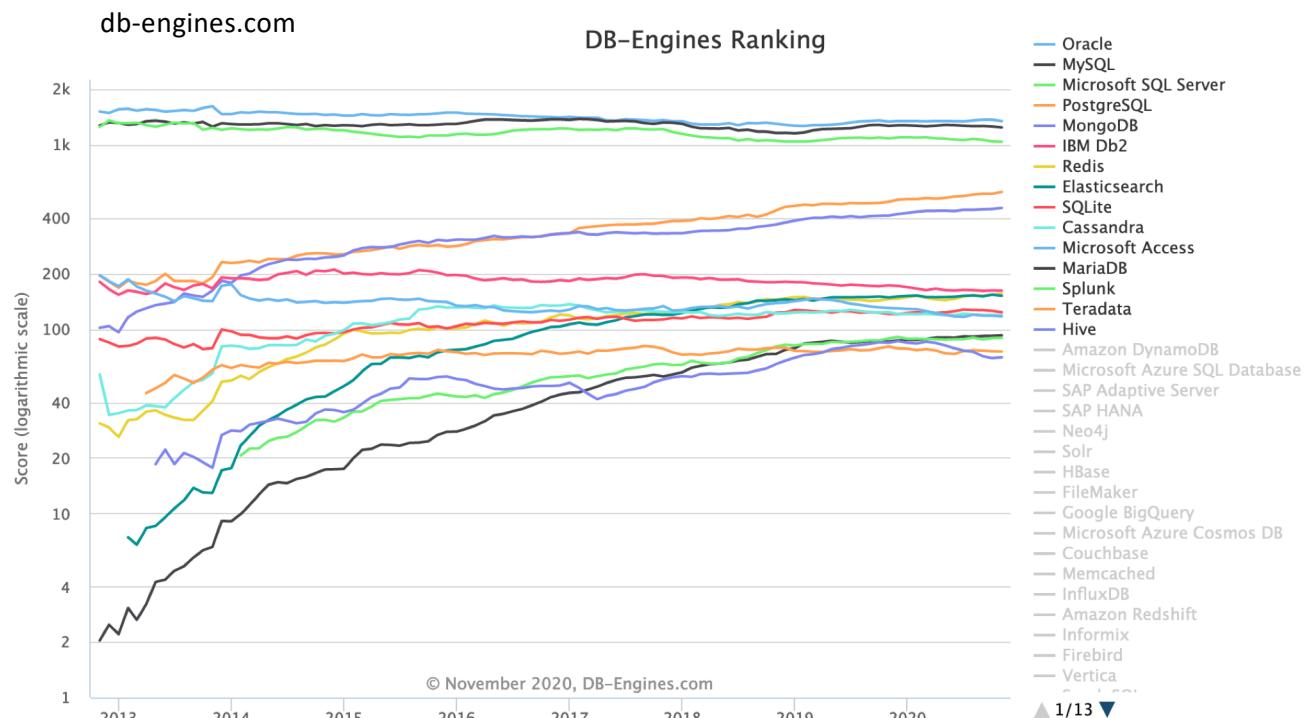
Deixam “cair” o escalonamento vertical (mais capacidade de hardware numa máquina) em favor do horizontal (mais máquinas, replicação de nós/servidores)

Nem todos os fabricantes usam a mesma estratégia: Cassandra opta pela Disponibilidade, mas a HBase dá prioridade à Consistência

Classificação das BD

360 systems in ranking, November 2020

Rank	Nov 2020	Oct 2020	Nov 2019	DBMS	Database Model	Score		
						Nov 2020	Oct 2020	Nov 2019
1.	1.	1.	1.	Oracle +	Relational, Multi-model ↗	1345.00	-23.77	+8.93
2.	2.	2.	2.	MySQL +	Relational, Multi-model ↗	1241.64	-14.74	-24.64
3.	3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model ↗	1037.64	-5.48	-44.27
4.	4.	4.	4.	PostgreSQL +	Relational, Multi-model ↗	555.06	+12.66	+63.99
5.	5.	5.	5.	MongoDB +	Document, Multi-model ↗	453.83	+5.81	+40.64
6.	6.	6.	6.	IBM Db2 +	Relational, Multi-model ↗	161.62	-0.28	-10.98
7.	↑ 8.	↑ 8.	8.	Redis +	Key-value, Multi-model ↗	155.42	+2.14	+10.18
8.	↓ 7.	↓ 7.	7.	Elasticsearch +	Search engine, Multi-model ↗	151.55	-2.29	+3.15
9.	9.	↑ 11.	11.	SQLite +	Relational	123.31	-2.11	+2.29
10.	10.	10.	10.	Cassandra +	Wide column	118.75	-0.35	-4.47



Classificação das BD não relacionais

- De acordo com o modelo de dados
 - BD orientadas a **documentos**
 - MongoDB, CouchDB, Amazon DocumentDB, BigChainDB
 - BD orientadas a **colunas**
 - Cassandra, HBase, Datastax, Microsoft Azure Table Storage, Accumulo
 - BD orientadas a **grafos**
 - Neo4j, JanusGraph, Virtuoso, Dgraph, Giraph, TigerGraph
 - DB orientadas a **chave-valor (key-value)**
 - Redis, Dynamo, Riak, Voldemort, Memcached, Amazon SimpleDB
- De acordo com as propriedades CAP
- De acordo com propriedades funcionais
- ...

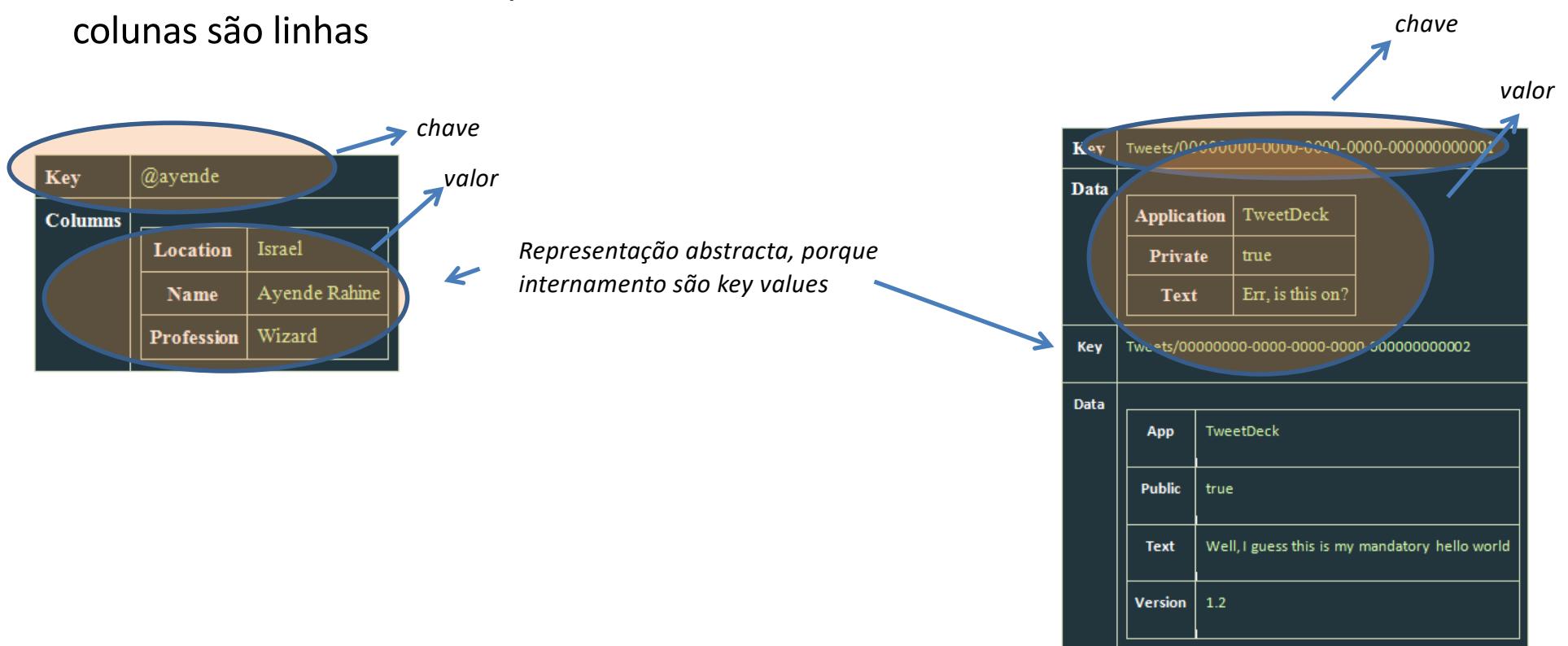
Key Values

Key (unique)	Value
user1923_color	Red
user1923_age	18
user3371_color	Blue
user4344_color	Brackish
user1923_height	6' 0"
user3371_age	34

- São as mais simples
- Adequadas para aceder rapidamente ao nó onde estão os dados
- Ineficiente para procurar por conteúdo (uma hipótese é ter mais índices, mas então torna-se mais lento)

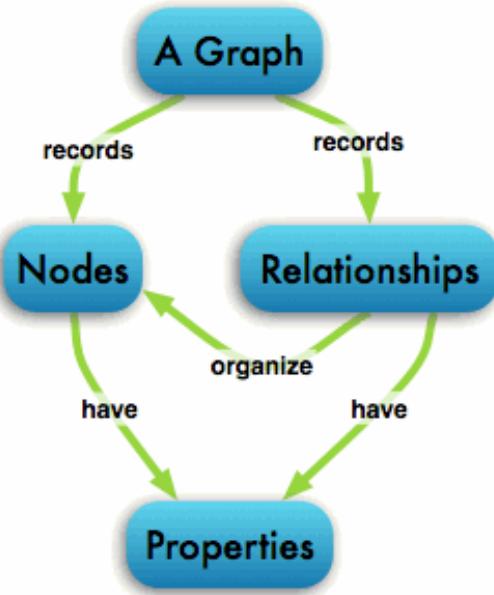
Column Family

- Solução key-value, mas o “value” são uma ou múltiplas colunas
- As colunas são arrumadas por famílias de colunas: é o relacional “invertido”, as colunas são linhas

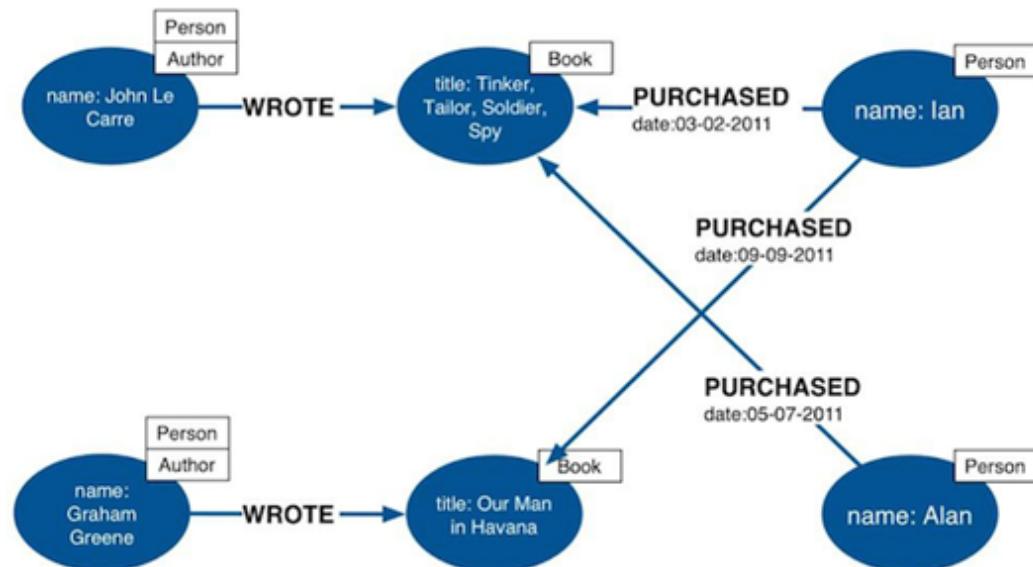


- Esta estrutura de armazenamento evita o desperdício de espaço dos nulos e colunas fantasma

GraphDatabases



- O modelo da informação são grafos que ligam nós (entidades) que contém propriedades (pares key-values)
- Os nós têm etiquetas (*tags*) para explicitar o papel (*role*) que o nó desempenha
- As tags também podem ser indexadas



Document Databases



- Os documentos são estruturas de dados JSON (ou XML) agrupados em coleções
- Versão avançada de Key-Values que permite associar a uma key estruturas mais complexas, tais como valores “encaixados” (*nested*) uns nos outros
- As consultas são mais eficientes e mais ricas que nas outras representações

Mongo DB Modelo de Dados



Coleção Patron

```
{
  "_id: "joe",
  name: "Joe Bookreader"
}
```

Coleção Address

```
{
  patron_id: "joe",
  street: "123 Fake Street",
  city: "Faketon",
  state: "MA",
  zip: "12345"
}
```

```
{
  patron_id: "joe",
  street: "1 Some Other Street",
  city: "Boston",
  state: "MA",
  zip: "12345"
}
```

Coleção Patron

```
{
  "_id: "joe",
  name: "Joe Bookreader",
  addresses: [
    {
      street: "123 Fake Street",
      city: "Faketon",
      state: "MA",
      zip: "12345"
    },
    {
      street: "1 Some Other Street",
      city: "Boston",
      state: "MA",
      zip: "12345"
    }
  ]
}
```

Duas coleções

Versus

Uma coleção

Mongo DB

Modelo de

Dados

coleção book

```
{
  title: "MongoDB: The Definitive Guide",
  author: [ "Kristina Chodorow", "Mike Dirolf" ],
  published_date: ISODate("2010-09-24"),
  pages: 216,
  language: "English",
  publisher: {
    name: "O'Reilly Media",
    founded: 1980,
    location: "CA"
  }
}

coleção book
{
  title: "50 Tips and Tricks for MongoDB Developer",
  author: "Kristina Chodorow",
  published_date: ISODate("2011-05-06"),
  pages: 68,
  language: "English",
  publisher: {
    name: "O'Reilly Media",
    founded: 1980,
    location: "CA"
  }
}
```

coleção publisher

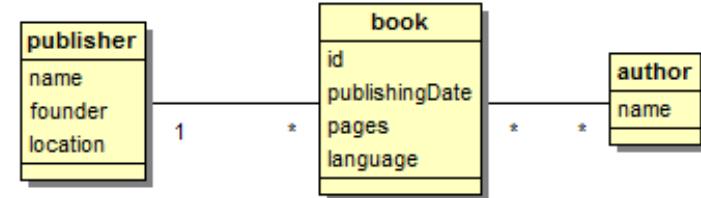
```
{
  name: "O'Reilly Media",
  founded: 1980,
  location: "CA",
  books: [123456789, 234567890, ...]
```

coleção book

```
{
  _id: 123456789
  title: "MongoDB: The Definitive Guide",
  author: [ "Kristina Chodorow", "Mike Dir
  published_date: ISODate("2010-09-24"),
  pages: 216,
  language: "English"
```

coleção book

```
{
  _id: 234567890,
  title: "50 Tips and Tricks for MongoDB De
  author: "Kristina Chodorow",
  published_date: ISODate("2011-05-06"),
  pages: 68,
  language: "English"
```



coleção publisher

```
{
  _id: "oreilly",
  name: "O'Reilly Media",
  founded: 1980,
  location: "CA"
```

coleção book

```
{
  _id: 123456789,
  title: "MongoDB: The Definitive Guide",
  author: [ "Kristina Chodorow", "Mike Dirolf" ],
  published_date: ISODate("2010-09-24"),
  pages: 216,
  language: "English",
  publisher_id: "oreilly"
```

coleção book

```
{
  _id: 234567890,
  title: "50 Tips and Tricks for MongoDB Developer",
  author: "Kristina Chodorow",
  published_date: ISODate("2011-05-06"),
  pages: 68,
  language: "English",
  publisher_id: "oreilly"
```

Estrutura de um documento em MongoDB VS modelo relacional

- Um **documento** equivale a uma **linha** no modelo relacional
- Um documento é um registo numa coleção de documentos
- Uma **coleção** de documentos equivale a uma **tabela** no modelo relacional
- Uma base de dados MongoDB é um conjunto de **colecções de documentos** armazenados
- O esquema não é predefinido antes de se inserirem dados

Sintaxe de um documento MongoDB

- **_id** é um identificador, gerado automaticamente ou não
- **att-1** é um atributo cujo valor é uma *string*
- **att-2** é um atributo cujo valor é um *íntero*
- **att-3** é um atributo cujo valor é uma lista de valores
- **att-n** é um atributo cujo valor é um documento encaixado (*nested*)

```
{  
  _id: <um identificador>,  
  "att-1": "val-1",  
  "att-2": val-2,  
  "att-3": ["val-31", "val-32", ...]  
  ...  
  "att-n": {"val-n1" : "val-n1",  
            ....  
          }  
}
```

As operações sobre os documentos

- As operações CRUD tornam-se IFUR
 - INSERT
 - FIND
 - UPDATE
 - REMOVE
- Uma inserção de dados cria automaticamente uma base de dados e uma coleção se estas não existirem ainda

Referência: <https://docs.mongodb.com/manual/crud/>