

NoSQL

Bancos de dados não relacionais

Marino H. Catarino
marinohc@gmail.com

Outubro/2020

O professor

Marino H. Catarino


- **Mestre em Ciências da Computação com ênfase em Big data** - Instituto de Matemática e Estatística – USP (2017)
- **Pós-graduação em Engenharia de Sistemas** - Escola Superior Aberta do Brasil – (2009)
- **Graduação em Ciência da Computação** - Instituto de Matemática e Estatística – USP – (2005)
-
- **Áreas de atuação:** Big data, mineração de dados, banco de dados não relacional.

marinohc@gmail.com



Porque NoSQL?

Qual motivo?



Sistemas Distribuídos + Bancos de Dados

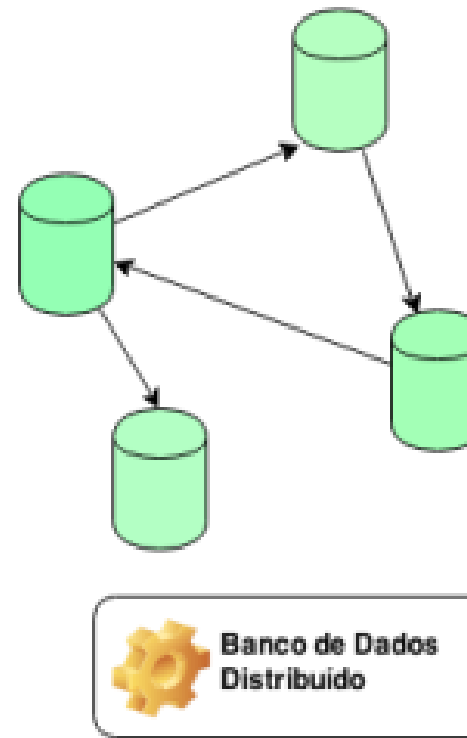
O que é um banco NoSQL?

Um banco de dados distribuído, não relacional, criado para trabalhar com grandes quantidade de dados (big data) utilizando servidores “de prateleira”.

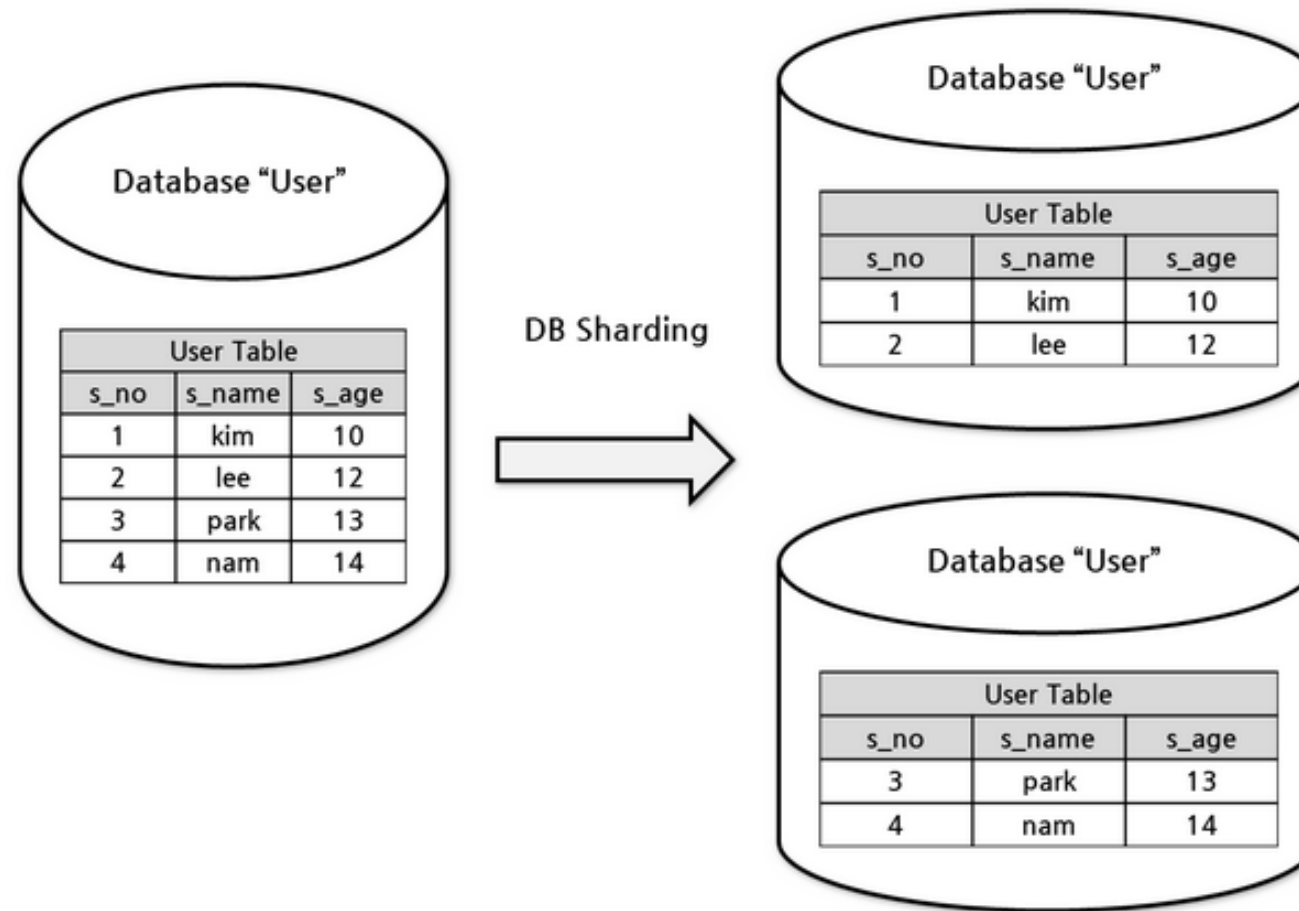
Distribuído?

Como assim distribuído?

- Particionamento de Dados
- Balanceamento de carga
- Tolerância a Falhas



Escalabilidade (particionamento horizontal)



O que é Não Relacional


Não tem transações ACID*

- Não tem SQL**
- Não tem tabelas***

* = alguns NoSQL já suportam transações multi-registro, mas não full ACID

** = alguns NoSQL possuem linguagens de consulta inspiradas em SQL

*** = alguns NoSQL tem tabelas



A C I D

Atomicidade

Isolamento

Consistência

Durabilidade

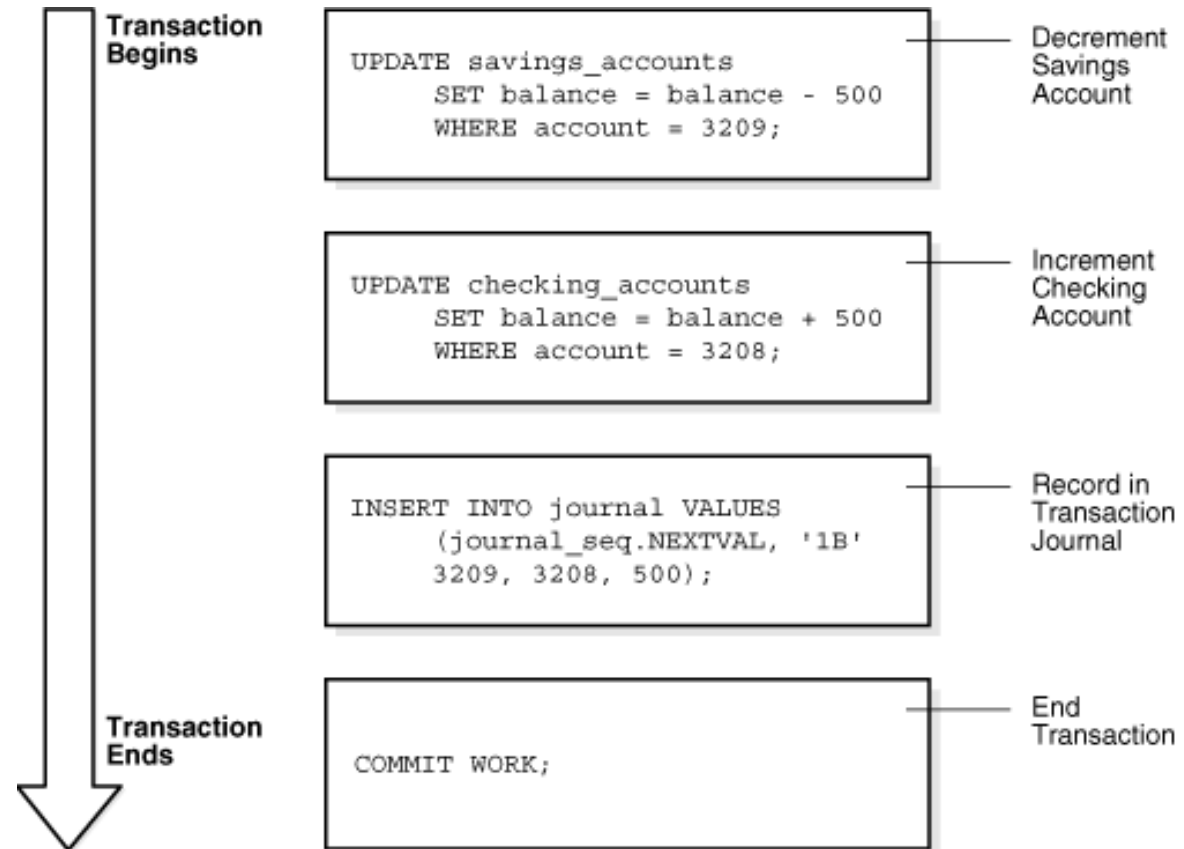


ACID

- Propriedades 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

- Propriedades ACID:





BASE

	Available	Eventually
Basically	Soft state	consistent



BASE

- Propriedades BASE:
 - **B**asically **A**vailable – Basicamente Disponível.
 - **S**oft-State – Estado Leve
 - **E**ventually 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).

Teorema CAP

Teorema CAP

- Definição
- Consistência – **C**onsistency.
- Disponibilidade – **A**vailability.
- Tolerância ao Particionamento - **P**artition tolerance.

Consistência

Significa se um sistema esta consistente, após a execução de uma operação. Por exemplo um sistema é considerado consistente se depois da atualização de um dado, todos os usuários que tem acesso a esse dado, possam acessá-lo em tempo real.

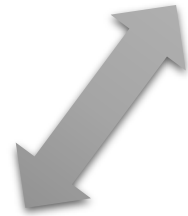
Disponibilidade

- Refere-se á concepção e implementação de um sistema de modo que seja assegurado que esse permanece ativo durante um determinado período de tempo.

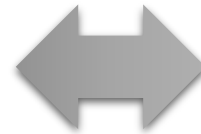
Tolerância ao Particionamento

- Refere-se a capacidade de um sistema continuar operando mesmo depois uma falha na rede.

Consistência



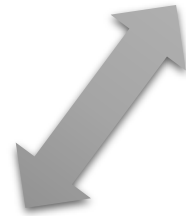
Disponibilidade
e



Tolerância ao
Particionamen
to



Consistência



Disponibilidade
e

- Os sistemas com consistência forte e alta disponibilidade não sabem lidar com a possível falha de uma partição.
- Caso ocorra, sistema inteiro pode ficar indisponível até o membro do cluster voltar.



Consistência

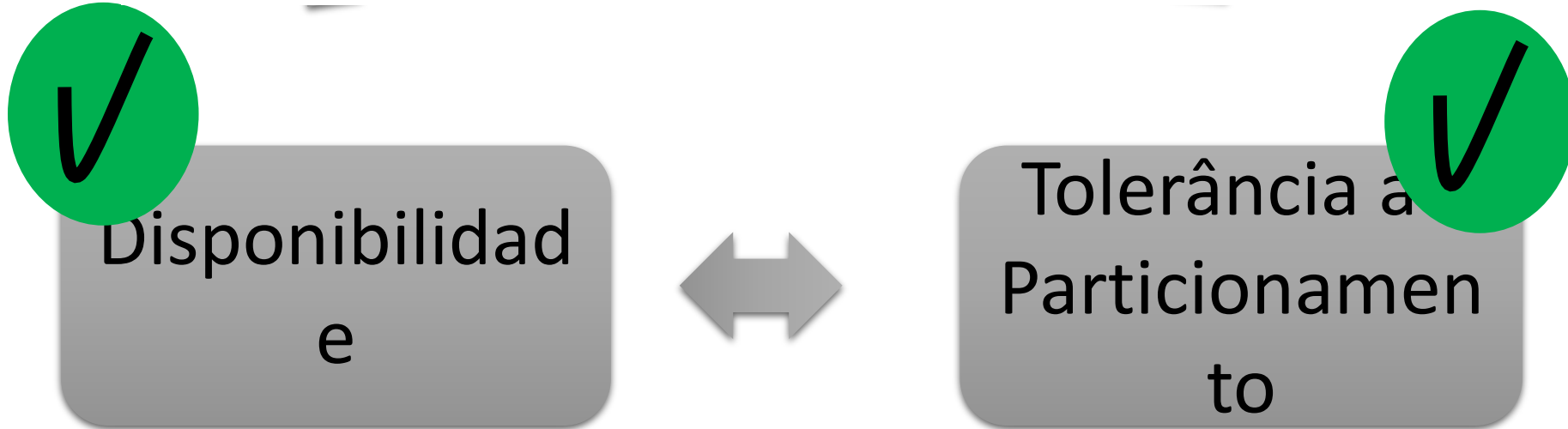
- Para sistemas que precisam da consistência forte e tolerância a particionamento é necessário abrir a mão da disponibilidade (um pouco).
- Exemplos são BigTable, HBase ou MongoDB entre vários outros.



Tolerância a
Particionamen
to

Há sistemas que jamais podem ficar offline, portanto não desejam sacrificar a disponibilidade. Para ter alta disponibilidade mesmo com uma tolerância a particionamento é preciso prejudicar a consistência.

Exemplos de Bancos são: Cassandra, MongoDB, Voldemort.



Principais sistemas no mercado

- MongoDB
- CouchDB
- Cassandra
- Project Valdemort (by LinkedIn)
- Redis (by Google)
- HBase (by Apache)
- Dynamo (by Amazon)
- dentre muitos outros...

Para que é indicado o NoSQL ?

- Indicado para aplicações que irão trabalhar com enormes quantidades de dados, que tem exigências de velocidade em suas consultas e escritas em grande volumes de dados.

Principais utilizadores do NoSQL

- Google - Bigtable.
- Amazon - Dynamo.
- Yahoo - Hadoop.
- Facebook - Cassandra.
- Digg - Cassandra.
- Twitter - Cassandra.
- IBM - Cassandra.
- Netflix - Cassandra.
- LinkedIn - Voldemort.
- Engine Yard - MongoDB.

Características dos NoSQL

- Alta escalabilidade
- Tolerância a falhas
- Alta vazão (throughput)
- Arquitetura Distribuída (cluster ou cloud)
- Open Source*

* = a maioria dos sistemas NoSQL atuais

Origem dos NoSQL

Sistemas internos desenvolvidos pela Google (Bigtable), e Amazon (Dynamo)

A arquitetura destes dois sistemas influencia a grande maioria dos sistemas NoSQL até hoje.

Tipo de armazenamento

Existem diversos tipos de armazenamento, no qual cada um trata os dados de uma forma diferente e que pode ser mais específico para o objetivo desejado. Os tipo de armazenamento são:

- Chave valor
- Família das colunas
- Documentos
- Banco de dados em Grafos

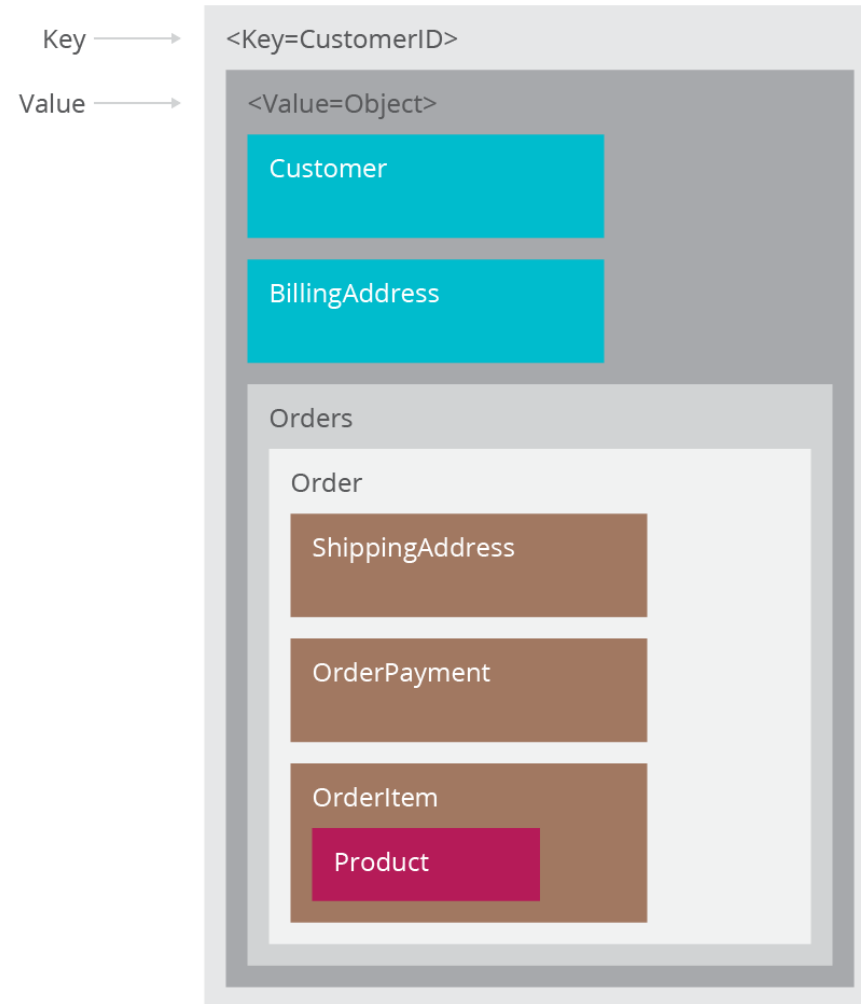
Chave Valor

Esse é o tipo de banco de dados NoSQL mais simples. O conceito dele é uma chave e um valor para essa chave, mas ele é o que aguenta mais carga de dados. Estes tipos de bancos de dados são o que tem a maior escalabilidade:

- Dynamo DB
- Project Voldemort
- Redis
- Riak



Chave/Valor

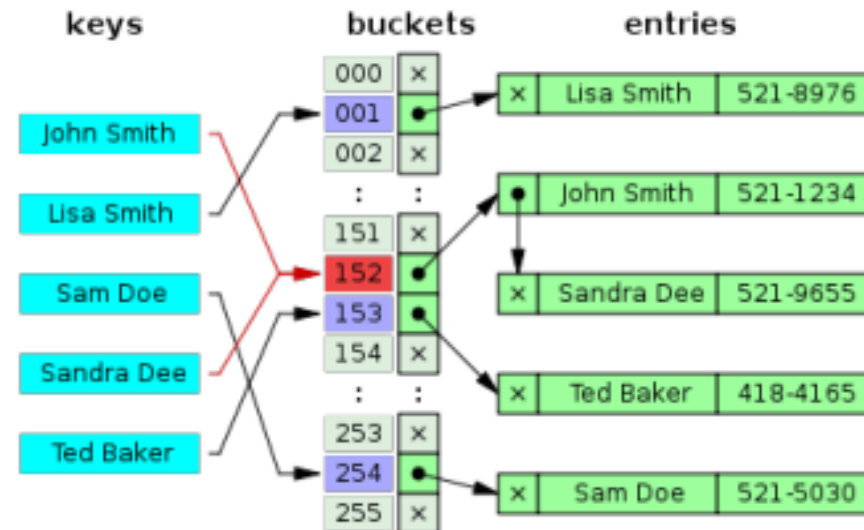
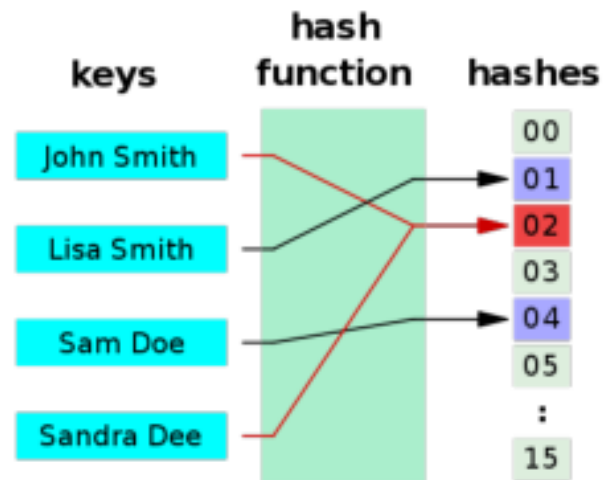


Ex.: Redis, DynamoDb, Couchbase, Riak, Azure Table Storage.

Chave Valor

Hash Table

http://en.wikipedia.org/wiki/Hash_table



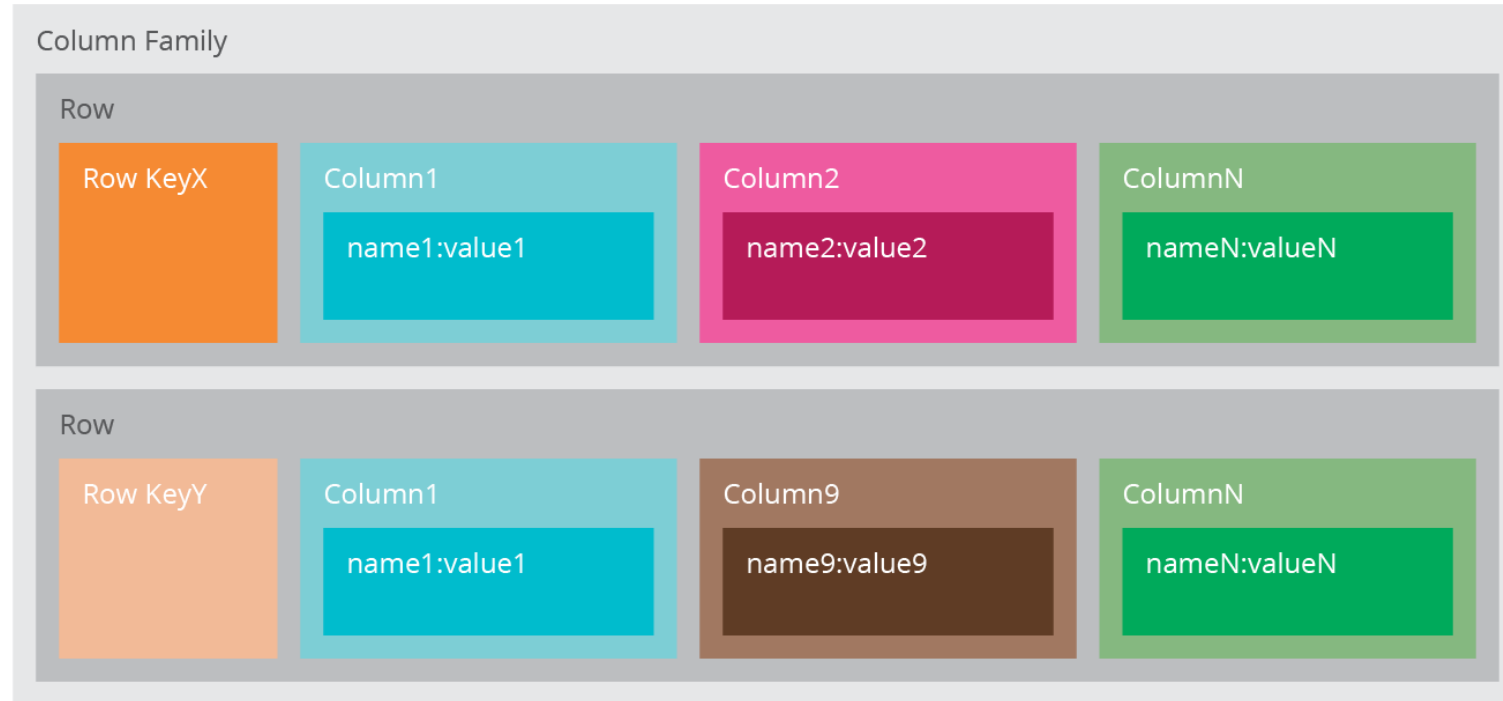
Família das Colunas

Fortemente inspirados pelo BigTable do Google, suportam várias linhas e colunas, e também permitem subcolunas. Além do BigTable do Google, outros que usam essa tecnologia são:

- HBase(Apache)
- HiperTable
- Cassandra(Apache)



Família de colunas



Ex.: HBase e Cassandra

Documentos

Baseados em documentos XML ou JSON, podem ser localizado pelo seu id único ou por qualquer registro que tenham no documento:

- CouchDB(Apache)
- MongoDB
- RavenDB



Orientado a documentos

```
<Key=CustomerID>
{
  "customerid": "fc986e48ca6" ← Key
  "customer":
  {
    "firstname": "Pramod",
    "lastname": "Sadalage",
    "company": "ThoughtWorks",
    "likes": [ "Biking", "Photography" ]
  }
  "billingaddress":
  {
    "state": "AK",
    "city": "DILLINGHAM",
    "type": "R"
  }
}
```

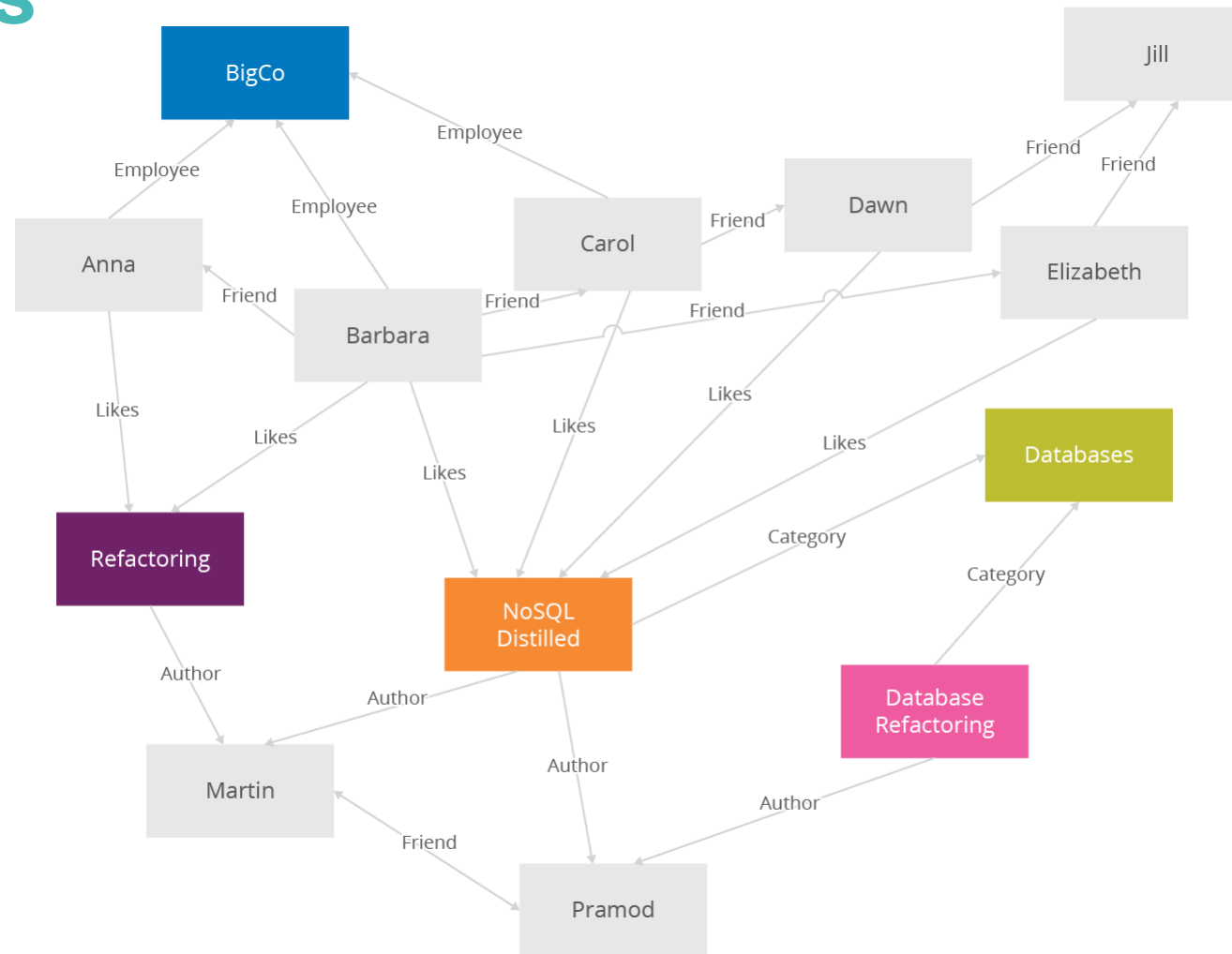
Ex.: MongoDB, CouchDB, RavenDb

Banco de dados em Grafos

Com uma complexibilidade maior, esses bancos de dados guardam objetos, e não registros, como os outros tipos de NoSQL. A busca desses itens é feita pela navegação destes objetos:

- Neo4J
- InfoGrid
- HyperGraphDB

Grafos



Ex.: Neo4J, Infinite Graph, InforGrid, HyperGraphDB

Por que usar NoSQL?

- Novos paradigmas (nem tão novos assim);
- Funcionalidades;
- Escalabilidade;
- Performance;
- Não ficar preso a modelagem;



Obrigado

- Marino H. Catarino
marinohc@gmail.com