

FIAP

AGENDA:

- Introdução a NoSQL
- ACID X BASE
- TEOREMA CAP
- NoSQL (Características)
- Os SGBDs Que as empresas Usam
- Tipos de SGBDs NoSQL
- CASES



Not Only SQL

- ✓ Os Bancos de dados relacionais começaram a ser apresentados no começo da década de 70 por Ted Codd, porém, só foram implementados com sucesso no final da década de 80.
- ✓ Mas a quantidade de dados que eram armazenados lá no começo dos anos 70, não são as mesmas quantidades de dados que são armazenados atualmente e as ferramentas utilizadas hoje evoluíram muito em relação as ferramentas daquela época.
- ❑ Exemplos de tecnologias que temos hoje:
 - Redes sociais
 - Computação nas Nuvens
 - Web 2.0

Bancos de dados relacionais

- ❖ Esquema rígido
- ❖ Consistência forte
- ❖ Métodos de acesso com suporte a processamento de consultas complexas (otimização e joins)



Qual é o problema com eles então?

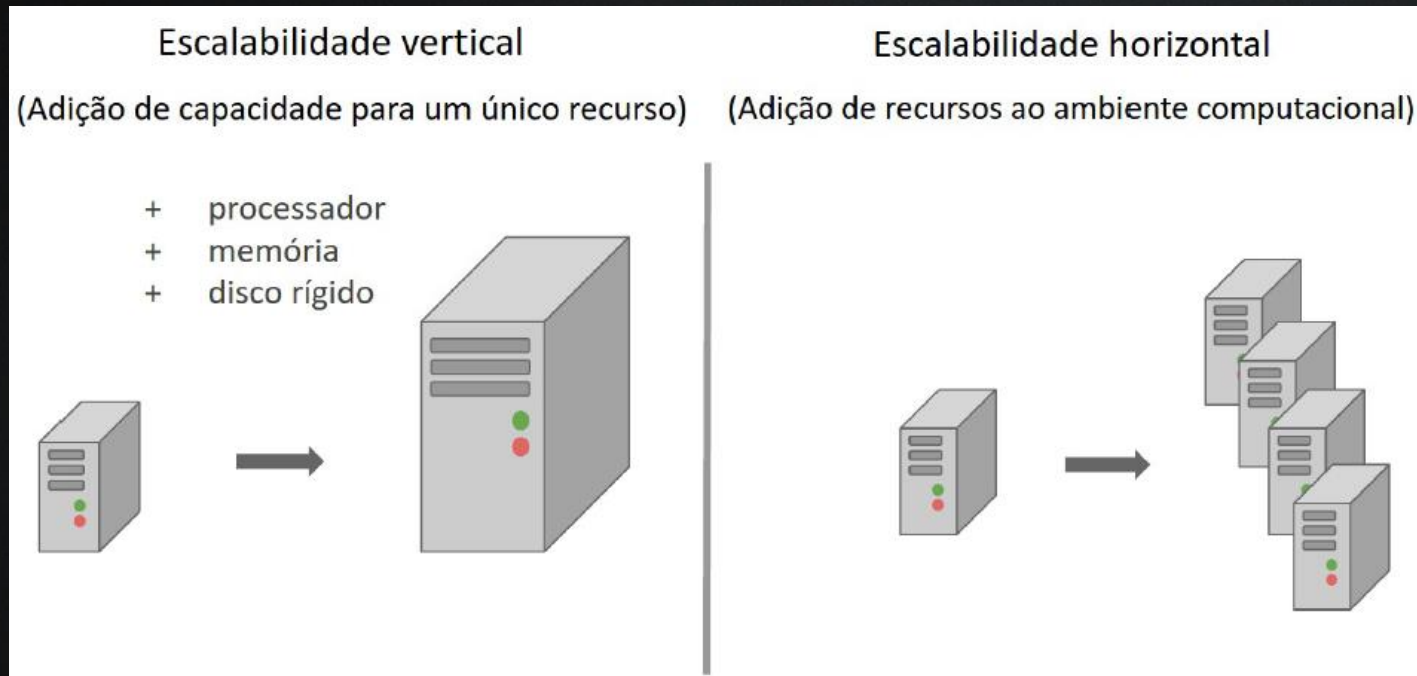
BigData

Enorme quantidade de dados, proveniente de várias fontes diferentes, complexos, dinâmicos, e que demandam tecnologias e técnicas específicas para armazenamento, análise e visualização.

- 1980 –Popularização dos SGBDs relacionais;
- 2009 –Início do movimento noSql.

Qual é o problema com eles então?

■ Escalabilidade



Bancos de dados relacionais escalam, mas quanto maior o tamanho, mais custoso

Surgimento do NoSQL

- ❑ 2006 - Google publica o artigo: BigTable: A Distributed Storage System for Structured Data.
- ❑ 2009 - o termo NoSQL é introduzido por um funcionário da Rackspace Eric Evans em um evento para discutir bancos de dados *open source* distribuídos
- ❑ Not only Structed Query Language (Não apenas SQL)
- ❑ Modelo de persistência de dados
- ❑ Soluções que não se encaixam em banco de dados relacionais

Surgimento do NoSQL

- ❑ O termo “NoSQL” é muito indefinido. Eles geralmente são aplicados a bancos de dados não relacionais que abraçam um esquema livre, executam em clusters e abrem mão da consistência por outras propriedades úteis.
- ❑ Próxima geração de banco de dados principalmente abordando alguns dos pontos: ser não-relacional, distribuídos, *open-source* e escalável horizontalmente. (NoSQL-Database.org)

Características

- ❑ Ausência de esquema ou esquema flexível
- ❑ Nem sempre consistente
- ❑ Falta de padronização
- ❑ A priori não suporta join nem relacionamentos
- ❑ Os bancos de dados NoSQL são projetados para escalar bem em direção horizontal e não depender de hardware.

Características

Sharding

Consiste em quebrar suas tabelas agrupando-as por dados semelhantes, ou seja, realizar o particionamento de dados entre vários computadores de uma forma de preservação da ordem.

Original Table

CUSTOMER ID	FIRST NAME	LAST NAME	CITY
1	Alice	Anderson	Austin
2	Bob	Best	Boston
3	Carrie	Conway	Chicago
4	David	Doe	Denver

Vertical Shards

VS1			VS2	
CUSTOMER ID	FIRST NAME	LAST NAME	CUSTOMER ID	CITY
1	Alice	Anderson	1	Austin
2	Bob	Best	2	Boston
3	Carrie	Conway	3	Chicago
4	David	Doe	4	Denver

Horizontal Shards

HS1			
CUSTOMER ID	FIRST NAME	LAST NAME	CITY
1	Alice	Anderson	Austin
2	Bob	Best	Boston
HS2			
CUSTOMER ID	FIRST NAME	LAST NAME	CITY
3	Carrie	Conway	Chicago
4	David	Doe	Denver

Sharding versus Partitioning

Sharding (*fragmentação*) e o particionamento consistem em dividir um grande conjunto de dados em subconjuntos menores. A diferença é que a fragmentação implica que os dados sejam espalhados por vários computadores, enquanto o particionamento não. O particionamento consiste em agrupar subconjuntos de dados em uma única instância de banco de dados.

ACID x BASE

- **A**tomacidade: garante que as transações sejam atômicas (indivisíveis). A transação será executada totalmente ou não será executada.
- **C**onsistência: garante que o banco de dados passará de uma forma consistente para outra forma consistente.
- **I**solamento: A propriedade de isolamento garante que a transação não será interferida por nenhuma outra transação concorrente.
- **D**urabilidade: A propriedade de durabilidade garante que o que foi salvo, não será mais perdido.

ACID x BASE

O conceito de BASE perde as propriedades de consistência e isolamento em favor de ganhar “disponibilidade e ganho de performance”.

Basically **A**vailable – prioridade na disponibilidade dos dados.

Soft-State – o sistema não precisa ser consistente o tempo todo.

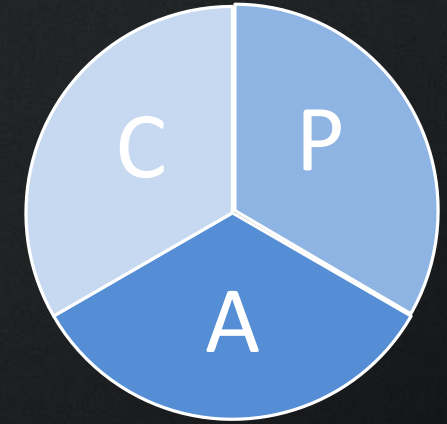
Eventually Consistent - Consistente em momento indeterminado.

ACID x BASE

Um NoSQL se utiliza dessas propriedades. São resumidas da seguinte forma:

“Uma aplicação funciona basicamente todo o tempo e não tem de ser consistente todo o tempo.”

Teorema CAP



Teorema CAP(*Consistency, Availability, Partition tolerance*) possui 3 pontos que são a:

- ❑ Consistência;
- ❑ Disponibilidade;
- ❑ Tolerância a falhas.

Explica que em sistema distribuído é preciso escolher entre duas dessas propriedades não sendo possível usar as três ao mesmo tempo.

Teorema CAP

- ❑ Consistência: significa que uma vez escrito um registro, cada cliente tem sempre a mesma visão dos dados.
- ❑ Disponibilidade: assegurar que este permaneça ativo durante um determinado período de tempo, ou inclusive que ele é tolerante a falhas.
- ❑ Tolerância ao Particionamento: capacidade de um sistema continuar operando mesmo depois uma falha na rede. Ou seja significa garantir que operações serão completadas, mesmo que componentes individuais não estejam disponíveis.

Teorema CAP

Sistemas CA



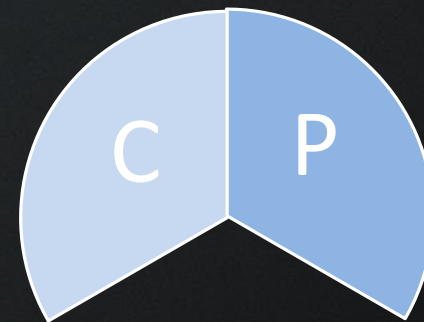
Os sistemas com consistência forte e alta disponibilidade (alta disponibilidade de um nó apenas) não sabem lidar com a possível falha de uma partição.

Caso ocorra o sistema inteiro pode ficar indisponível até o membro do cluster voltar.

- Exemplos disso são algumas configurações clássicas de bancos relacionais.

Teorema CAP

Sistemas CP

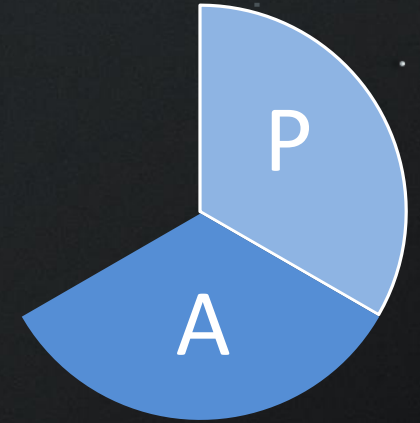


Para sistemas que precisam da consistência forte e tolerância a particionamento é necessário abrir a mão da disponibilidade (um pouco).

- Exemplos desses sistemas CP são BigTable, HBase ou MongoDB entre vários outros.

Teorema CAP

Sistemas PA



Por outro lado existem sistemas que jamais podem ficar offline. Para ter alta disponibilidade mesmo com uma tolerância a particionamento é preciso prejudicar um pouco a consistência.

Esses sistemas aceitam escritas sempre e tentam sincronizar os dados em algum momento depois (read-consistency).

□ Exemplos aqui são Amazon Dynamo, Cassandra ou Riak.

Chave-valor

Modelo simples similar a uma estrutura de indexação, também conhecidos como tabelas de hash distribuídas, armazenam objetos indexados por chaves, e possibilitam a busca por esses objetos a partir de suas chaves.

Chave	Valor
carro_3345_cor	preto
carro_3345_pneu	17
carro_3365_cor	branco
carro_3365_pneu	15
carro_4560_peso	1215
carro_4715_ano	2016

Uma das desvantagens desse modelo é que ele não permite a recuperação de objetos por meio de consultas mais complexas.

Chave-valor

Em relação a manipulação dos dados, as operações são bem simples como:

- `get()`: Usado para retorna o valor
- `set()`: Usado para Capturar o valor.

Alguns bancos que utilizam esse padrão são:

- DynamoDb, Couchbase, Riak, Azure Table Storage, Redis, Tokyo Cabinet, Berkeley DB, etc...

Chave-valor

Quando e Por que utilizar um banco chave-valor?

Bancos de dados NoSQL do tipo chave-valor são ideais para cenários nos quais você precisa armazenar e recuperar dados associados a uma chave única de maneira rápida e eficiente. Eles são especialmente úteis em situações onde a escalabilidade e o desempenho são prioridades. Aqui estão alguns exemplos de casos de uso em que um banco de dados chave-valor é apropriado:

Chave-valor

❑ **Cache:** Um dos usos mais comuns para bancos de dados chave-valor é como cache de dados. Eles podem armazenar resultados de consultas ou informações frequentemente acessadas, reduzindo a carga sobre sistemas de banco de dados mais pesados e melhorando a latência. O Redis é um exemplo popular para essa finalidade.

Sessões de Usuário: Muitos aplicativos web precisam rastrear as sessões dos usuários. Um banco de dados chave-valor pode ser usado para armazenar informações de sessão, com a chave sendo o ID da sessão e os valores contendo dados relevantes, como preferências do usuário ou tokens de autenticação.

❑ **Armazenamento de Configuração:** Chaves e valores podem ser usados para armazenar configurações de aplicativos. Isso permite que você ajuste o comportamento de um aplicativo sem a necessidade de alterar o código-fonte. Alterar um valor em um banco de dados chave-valor é mais rápido do que reimplantar um aplicativo.

Chave-valor

- ❑ **Armazenamento de Metadados:** Em sistemas de gerenciamento de arquivos ou aplicativos que lidam com objetos binários (como imagens, vídeos ou arquivos), você pode usar um banco de dados chave-valor para armazenar metadados associados a esses objetos. Isso pode incluir informações como data de criação, autor, tipo de arquivo, etc.
- ❑ **Gestão de Filas:** Os bancos de dados chave-valor também são usados em sistemas de mensageria para criar filas de mensagens. As chaves representam os itens da fila e os valores contêm as mensagens a serem processadas.
- ❑ **Rastreamento de Pedidos:** Em sistemas de comércio eletrônico, você pode usar um banco de dados chave-valor para rastrear o status dos pedidos. Cada pedido é uma chave e os valores podem conter informações sobre o status do pedido, como "pendente", "enviado" ou "entregue".

Chave-valor

- **Geolocalização:** Em aplicativos que envolvem geolocalização, você pode usar um banco de dados chave-valor para armazenar informações de latitude e longitude associadas a chaves específicas, como pontos de interesse, cidades ou coordenadas geográficas.

Em resumo, os bancos de dados chave-valor são uma escolha sólida quando você precisa de acesso rápido e eficiente aos dados com base em chaves únicas. Eles são particularmente úteis em cenários de cache, armazenamento de configuração, rastreamento de sessões e muitos outros casos em que a escalabilidade e o desempenho são essenciais.

Colunar

Modelo mais complexo que o chave-valor.

Neste tipo de modelo os dados são indexados por uma tripla (linha, coluna e timestamp)

Linhas e as colunas são identificadas por chaves e o timestamp é o que permite identificar as diferentes versões de um mesmo dado.

TABELA ALUNO			
1	678.Y31.X33-40	4.03X.894 Y	Adriano P. S.
2	6X4.616.Y86-83	2.Y77.26X	Roberto M. G.
3	483.704.8Y3-3X	91.X2Y.53Z 1	Augusto S. F.
4	Y87.828.28X-63	8.456.XY2 Z	Mariana F. B.
5	711.2X2.Y78-28	12.37X.Y23 4	Carlos A. E.
6	1XY.064.413-12	6.472.33X Y	Amanda C.
Matricula	CPF	RG	Nome

Colunar

Outro conceito importante sobre esse modelo é o de família de colunas (column family), é usado com o objetivo de agrupar colunas que armazenam o mesmo tipo de dados.

Este modelo surgiu com o BigTable da Google.

Outros bancos de dados que são orientados a coluna:

- Hadoop, Cassandra, Hypertable, Amazon SimpleDB, etc.

Quando e por que utilizar um banco de dados colunar?

Os bancos de dados NoSQL do tipo colunar (também chamados de bancos de dados de família de colunas) são adequados para cenários nos quais você lida com grandes volumes de dados que possuem estruturas complexas e variáveis. Eles são especialmente eficazes para consultas analíticas e agregações de dados. Aqui estão alguns exemplos de casos de uso em que um banco de dados colunar é apropriado:

❑ **Análise de Dados:** Bancos de dados colunares são ideais para análises de dados em que você precisa buscar informações específicas de colunas em um grande conjunto de dados. Por exemplo, em um ambiente de análise de negócios, você pode armazenar dados de vendas em um banco de dados colunar e realizar consultas para calcular métricas como vendas por região, por produto ou por período de tempo.

❑ **Armazenamento de Séries Temporais:** Para coletar e analisar dados de séries temporais, como leituras de sensores, logs de servidores, dados de telemetria e métricas de desempenho, os bancos de dados colunares podem ser uma escolha eficaz. As colunas são tipicamente usadas para armazenar séries temporais de diferentes variáveis, permitindo consultas eficientes por tempo e valor.

❑ **Data Warehousing:** Bancos de dados colunares são frequentemente usados como soluções de data warehousing para armazenar grandes volumes de dados históricos, tornando-os acessíveis para análises de business intelligence (BI). Eles podem lidar com esquemas de dados altamente dimensionais, o que é comum em ambientes de BI.

- **❑ Análise de Dados em Big Data:** Bancos de dados colunares são frequentemente
 - usados em ambientes de análise de big data, onde você precisa consultar e analisar grandes volumes de dados, como logs de servidores, registros de eventos ou dados de sensores. Por exemplo, você pode usar o Apache Cassandra ou o HBase para armazenar e consultar registros de logs de servidores distribuídos.

Em resumo, os bancos de dados colunares são particularmente eficazes quando se trata de consultas analíticas em grandes volumes de dados, onde a eficiência na recuperação de dados de colunas específicas é essencial. Eles são usados em uma variedade de cenários, incluindo análise de big data, data warehousing, sistemas de gerenciamento de conteúdo e muito mais, onde a velocidade e a escalabilidade são importantes. Exemplos de bancos de dados colunares incluem Apache Cassandra, HBase, Amazon Redshift e Google BigQuery.

Grafos

O modelo orientado a grafos possui três componentes básicos:

Nós: são os vértices do grafo;

Relacionamentos: são as arestas;

Propriedades (ou atributos) dos nós e relacionamentos.



Grafos

Vantajoso quando consultas complexas são exigidas pelo usuário, pelo ganho de performance o que permitiria um melhor desempenho das aplicações.

Consultas mais simples e diretas.

Alguns bancos que utilizam esse padrão são:

- Neo4J, Infinite Graph, InforGrid, HyperGraphDB, etc.

Quando e por que utilizar um banco de dados Grafos?

Bancos de dados NoSQL de grafos são projetados para representar e armazenar dados relacionais de maneira eficiente, tornando-os adequados para cenários onde as relações entre os dados são fundamentais. Aqui estão alguns exemplos de casos de uso nos quais um banco de dados de grafos é apropriado:

- ❑ **Redes Sociais:** Redes sociais como o Facebook, LinkedIn e Twitter usam bancos de dados de grafos para armazenar informações de amizade, conexões de rede e interações entre usuários. Esses bancos de dados são ideais para traçar relacionamentos complexos entre pessoas e entidades.
- ❑ **Recomendações e Filtragem Colaborativa:** Plataformas de recomendação, como o Netflix e o Amazon, usam bancos de dados de grafos para analisar o histórico de interações do usuário e identificar padrões de preferência, permitindo assim recomendações precisas de produtos, filmes, músicas, etc.
- ❑ **Sistemas de Recomendação de Conteúdo:** Blogs, sites de notícias e portais de conteúdo podem usar bancos de dados de grafos para entender as preferências dos usuários com base no histórico de leitura e fornecer conteúdo relevante e personalizado.

- ❑ **Redes Sociais:** Redes sociais como o Facebook, LinkedIn e Twitter usam bancos de dados de grafos para armazenar informações de amizade, conexões de rede e interações entre usuários. Esses bancos de dados são ideais para traçar relacionamentos complexos entre pessoas e entidades.
- ❑ **Recomendações e Filtragem Colaborativa:** Plataformas de recomendação, como o Netflix e o Amazon, usam bancos de dados de grafos para analisar o histórico de interações do usuário e identificar padrões de preferência, permitindo assim recomendações precisas de produtos, filmes, músicas, etc.
- ❑ **Sistemas de Recomendação de Conteúdo:** Blogs, sites de notícias e portais de conteúdo podem usar bancos de dados de grafos para entender as preferências dos usuários com base no histórico de leitura e fornecer conteúdo relevante e personalizado.

- ❑ **Análise de Fraude:** Bancos de dados de grafos são úteis para identificar atividades fraudulentas, como fraudes de cartão de crédito, analisando padrões de transações e conexões suspeitas entre contas.
- ❑ **Bioinformática:** Em bioinformática, os bancos de dados de grafos podem ser usados para representar redes complexas de interações genéticas, proteínas e outras moléculas biológicas.


Em resumo, os bancos de dados de grafos são adequados para cenários onde a estrutura de relacionamento dos dados é complexa e fundamental para as operações ou análises. Eles permitem a representação eficiente de conexões e relações, o que é útil em uma ampla gama de aplicações, desde redes sociais até análise de fraudes e sistemas de recomendação. Exemplos populares de bancos de dados de grafos incluem Neo4j, Amazon Neptune e OrientDB.

Documentos

Este modelo armazena coleções de documentos.

Um documento no geral são coleções de atributos e valores.

Este atributo, por sua vez, pode ser um conjunto de campos, que possui strings, listas ou documentos aninhados.

MongoDB	Bancos de dados relacionais	Exemplo
Document	Linha	{ "nome" : "Guilherme Lima", "email" : "gui@alura.com" }
Field	Coluna	{ "nome" : "Guilherme Lima", "email" : "gui@alura.com" }
Collection	Tabela	

Documentos

Tem esquema livre e possui facilidade por utilizar formatos como XML e JSON.

Também apresenta facilidade e flexibilidade em atualizar a estrutura dos documentos.

Dentre os bancos de dados que utilizam esse modelo:

- CouchDB e o MongoDB.

Quando e por que utilizar um banco de dados orientado a Documentos?

Bancos de dados NoSQL orientados a documentos são adequados para cenários onde a flexibilidade na modelagem de dados e a capacidade de armazenar documentos semiestruturados são essenciais.

Segue alguns exemplos de utilização:

❑ **Gerenciamento de Conteúdo Web:** Plataformas de gerenciamento de conteúdo, como blogs e sites de notícias, podem usar bancos de dados orientados a documentos para armazenar artigos, postagens de blog e outros tipos de conteúdo. Cada documento pode representar uma postagem, com campos variáveis como título, corpo e autor.

❑ **Sistemas de Gerenciamento de Conteúdo Empresarial (ECM):** Organizações que precisam gerenciar documentos internos, como contratos, relatórios e documentos de recursos humanos, podem usar um banco de dados orientado a documentos para armazenar e pesquisar esses documentos de forma eficiente.

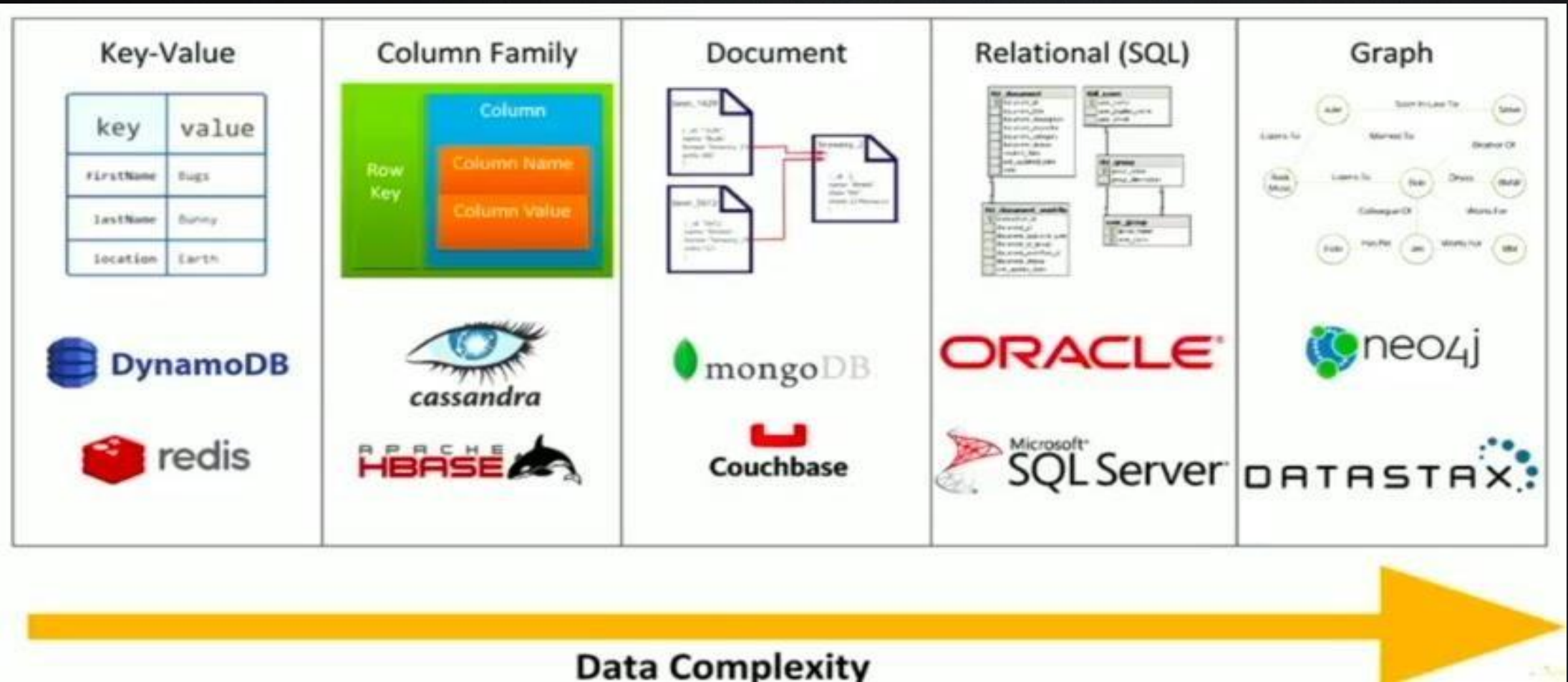
❑ **Sistemas de Comentários e Feedback:** Aplicações que incluem recursos de comentários e feedback de usuários podem usar bancos de dados orientados a documentos para armazenar e consultar esses dados, onde cada comentário ou feedback é representado como um documento.

❑ **Gestão de Documentos Fiscais:** Empresas que precisam armazenar documentos fiscais, como faturas e recibos, podem usar um banco de dados orientado a documentos para organizar esses documentos de forma eficiente.

❑ **Registro de Sensores IoT:** Em aplicações de Internet das Coisas (IoT), onde sensores geram dados semiestruturados, um banco de dados orientado a documentos pode ser usado para armazenar e consultar esses dados.

Bancos de dados orientados a documentos são ideais quando você precisa de flexibilidade na estrutura de dados e pode lidar com documentos semiestruturados. Eles são amplamente utilizados em uma variedade de aplicações, incluindo gerenciamento de conteúdo web, sistemas de e-commerce, sistemas de gerenciamento de documentos, e muitos outros, onde a adaptabilidade na modelagem de dados é crucial. Exemplos populares de bancos de dados orientados a documentos incluem MongoDB, Couchbase e Elasticsearch.

Comparação de complexidade



Mitos sobre NoSQL

NOSQL É ESCALÁVEL! UMA DAS GRANDES PROMESSAS DOS BANCOS NOSQL.

Dizer que seu sistema escala sozinho é vender um sonho.

Pode até ser mais fácil de escalar se comparado a outras soluções mas ainda sim exigirá algum esforço para escalar.

NOSQL É MAIS ECONÔMICO! MEIA VERDADE.

Existem muitos casos entretanto que uma solução relacional atende perfeitamente todas as necessidades do cliente, como MySQL e PostgreSQL.

Quando e qual utilizar?



Fonte: Martin Fowler

Fonte: <http://www.martinfowler.com/bliki/PolyglotPersistence.html>

Dá pra fazer query?

Amazon DynamoDB– Key-Value

SQL Query

```
SELECT _id, name, address  
FROM users  
WHERE age > 18  
LIMIT 5
```

← projection
← table
← select criteria
← cursor modifier

AWS Query

Amazon DynamoDB Explore Table: Reply

List Tables Browse Items

☐ Scan ☒ Query Go New Item Edit Item Copy to New Delete Item

Index Name: [Table] Reply: Id, ReplyDateTime

Hash Key (*): Id equal to Amazon DynamoDB#DynamoDB Thre

Range Key: ReplyDateTime greater than 2014-01-12

Sort: ☒ Ascending ☐ Descending

Dá pra fazer query?

MongoDB - Document

SQL Query

```
SELECT _id, name, address ← projection
FROM   users              ← table
WHERE  age > 18            ← select criteria
LIMIT  5                  ← cursor modifier
```

Operation Find

```
db.users.find(              ← collection
  { age: { $gt: 18 } },     ← query criteria
  { name: 1, address: 1 }   ← projection
).limit(5)                  ← cursor modifier
```

Dá pra fazer query?

Neo4j- Graph

SQL Query

```
SELECT _id, name, address ← projection
FROM users ← table
WHERE age > 18 ← select criteria
LIMIT 5 ← cursor modifier
```

Cyber query

```
MATCH a
WHERE a.age>18
RETURN a.id, a.name, a.address
LIMIT 5
```


Dá pra fazer query?

Cassandra - Column

SQL Query

```
SELECT _id, name, address  
FROM users  
WHERE age > 18  
LIMIT 5
```

← projection
← table
← select criteria
← cursor modifier

CQL – Cassandra Query Language

```
SELECT _id, na  
FROM users  
WHERE age > 18  
LIMIT 5
```

← select criteria
← cursor modifier

Comandos CRUD

(Create, Read, Update, Delete)

são iguais

Cases

The Netflix logo is centered within a red rectangular box. It consists of the word "NETFLIX" in a bold, white, sans-serif font. Each letter has a 3D effect, with a dark grey shadow cast to the right and slightly downwards, giving it a sense of depth.

SGBD:

sistema de processamento de faturas mensais

NOSQL:

Sistema focado em recomendações de melhores filmes.

Cases



SGBD:

Sistemas de processamento de ordem de venda

NOSQL:

Sistema de pesquisa, recomendações e adaptações de preços em tempo real

Conclusão

- ❑ Não há consenso ou padronização;
- ❑ Cada projeto resolve problemas particulares;
- ❑ Gama de possibilidades de armazenamento da informação;
- ❑ Certos cenários bancos relacionais são mais indicados (ACID);
- ❑ Não veio substituir o modelo relacional;

