Banco de Dados Relacionais e Não Relacionais

Prof. Henrique Batista da Silva

Introdução

Fonte: http://brainsciences.org/launch-of-the-big-data-analytics-journal/

Introdução

A quantidade de dados produzidos nos últimos anos teve um gigantesco aumento.

Dados são gerados por diversas fontes (redes sociais, jornais, revistas e em diversas mídias).

A quantidade massiva de dados a serem manipulados é uma realidade e tende a aumentar cada vez.

O que é Big data

"*Big data pode ser descrito em termos de desafios de gerenciamento de dados que – devido aos crescentes volume, velocidade e variedade dos dados – não podem ser resolvidos com bancos de dados tradicionais.* "
(Amazon Web Services - AWS)

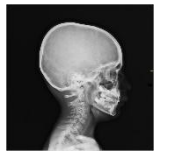
Segundo a definição da AWS, Big Data é caracterizado por:

- Alto volume de dados
- Variedade de fontes (origens) e formatos
- Velocidade em coleta, processamento e análise dos dados

Fonte: https://aws.amazon.com/pt/big-data/what-is-big-data/

Desafio de Big data

Redes sociais



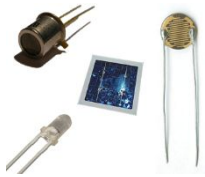
Medicina

e-commerce

TVs

Satélites

Sensores



Coleta de informação

Correlação

Armazenamento

Predição / Análise

Manipulação de grande quantidade de informação

Previsão do tempo

Hábitos de consumo

Business Inteligence

Insight

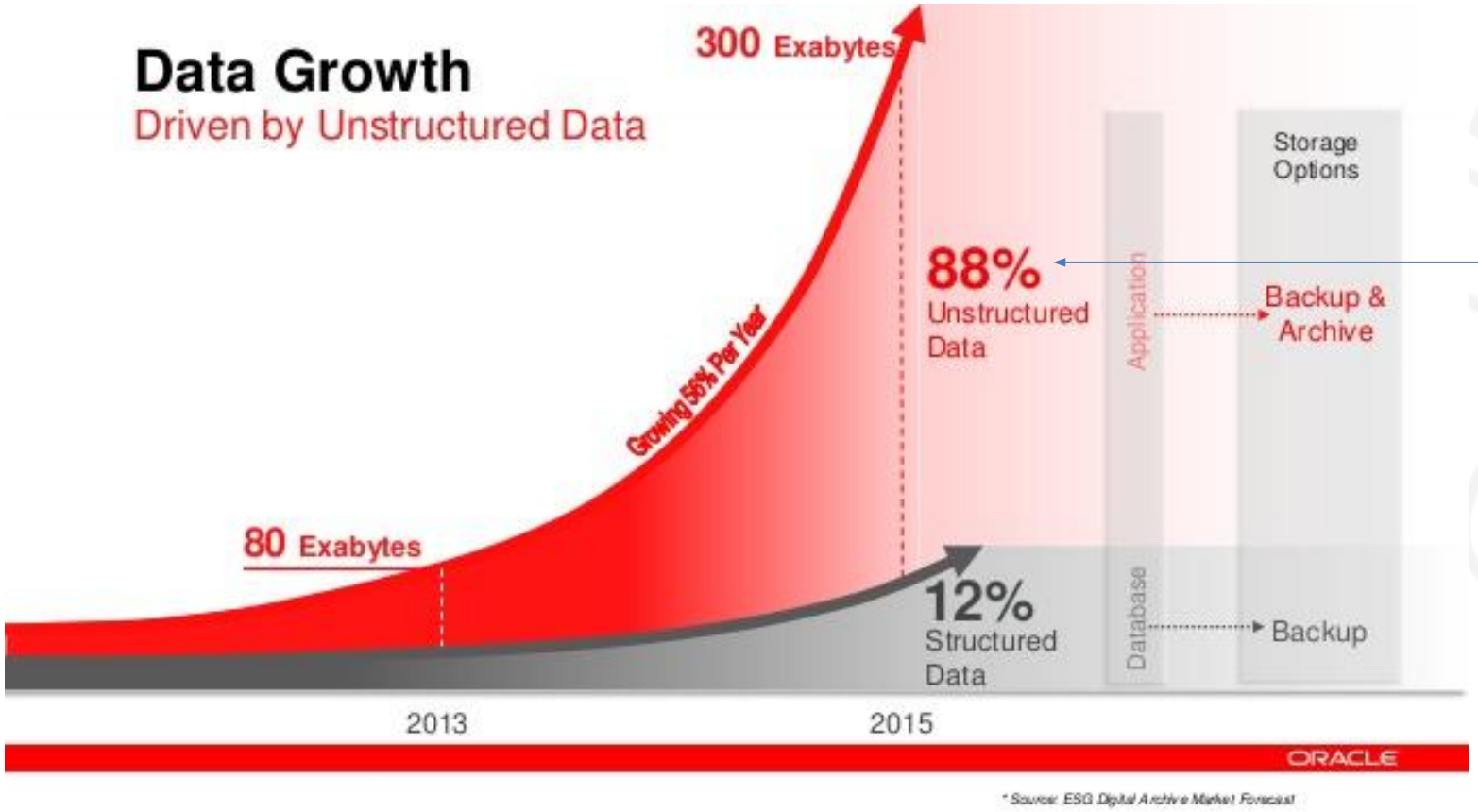
Fonte: imagem adaptada de: Ron Kasabian. Intel Big Data & Cloud Summit, 2013

Insight

- Alguns exemplos:
  - E-commerce (Amazon, ebay, etc.): criar ofertas online para garantir a fidelidade do cliente e obter novos clientes.
    - Sistemas de recomendação: se você gostou X, então talvez pode gostar de Y (análise predicativa)
  - Detecção de fraude: prever a probabilidade que uma determinada transação do cliente seja fraude.
  - Web e redes sociais: Hoje os próprios clientes geram muito conteúdos sobre eles mesmo. Toda esta informação pode ser aproveitada para identificar gostos dos clientes (e oferecer produtos).

Fonte: Arquitetura e Padrões de Big Data (IBM). Disponível em: http://www.ibm.com/developerworks/br/library/bd-archpatterns1/

Qual o tamanho do Big data?



88% dos dados em 2015 são dados não estruturados

Fonte: Fran Navarro. Hardware e Software Engineered to Work Together. Oracle Open World

Introdução à NoSQL

Armazenamento dos dados

A **quantidade massiva de dados**, principalmente dados não estruturados, a serem manipulados é uma realidade e **tende a aumentar cada vez**.

Soluções de armazenamento de dados massivos tornam-se necessárias.

Armazenamento dos dados

Por muito tempo banco de dados Relacionais tem sido o principal meio de armazenamento de dados.

O modelo de dados relacional foi introduzido por Edgar Codd em 1970 (IBM).

Armazenamento dos dados

A ideia de modelo relacional era representar **entidade** e **relacionamento** de maneira uniforme.

Os SGBDs Relacionais mais conhecidos hoje são MySQL (Oracle), Oracle (Oracle) e SQL Server (Microsoft).

Exemplo de um modelo Relacional

Modelo de Relacional (organizado em tuplas, normalizado, e possui integridade referencial)

Tabela: Cliente	
Id	Nome
1	Marcos

Tabela: Pedido		
Id	IdCliente	IdEndEntrega
1	1	1

Tabela: Produto	
Id	Nome
10	Laptop

Tabela: ItemPedido			
Id	IdPedido	IdProduto	Preço
1	2	10	350,00

Tabela: Endereço				
Id	Logradouro	Cidade	Estado	CEP
1	Av. Sen. Salgado Filho	Natal	RN	59.056-000

Alguns desafios para persistências de dados

Atomicidade (a transação é executada totalmente ou **não** é executada), **C**onsistência (sistema sempre consistente após uma operação), **I**solamento (transação não sofre interferência de outra transação concorrente), e **D**urabilidade (o que foi salvo não é mais perdido)

- Força a consistência ao final de cada transação

Alguns desafios para persistências de dados

Difícil implementar escalabilidade para banco de dados relacionais.

Esquema rígido (mudança são mais complexas)

Armazenamento NoSQL

Hoje há novos desafios e banco de dados **NoSQL** foram criados com o objetivo de **manipular volumes maiores de dados**.

NoSQL: “Not Only SQL”: Na prática seria algo como “*NoRelational*” (mas podem ter relacionamentos, porém não são orientados à tabelas)

Vantagens do NoSQL

Modelo de dados mais flexível

NoSQL fornece um **modelo de dados que se adapta melhor às necessidades** da aplicação (modelagem mais rápida e menos código).

Armazenamento NoSQL

Grande quantidade de dados: Necessita de processamento em cluster (armazenamento dividido entre vários servidores).

Favorece replicação de dados e escalabilidade.

Surgimento e características do NoSQL

O termo NoSQL: Meetup em 2009

A ideia principal em torno do nome é **não utilizar SQL** (mas linguagens semelhantes)

São geralmente **projetos de código aberto**.

Em sua maioria são **orientados para execução em clusters**.

Surgimento e características do NoSQL

Não há um esquema para definição de banco de dados NoSQL. É possível adicionar novos campos aos registros sem haver, previamente, uma definição do esquema.

Os três V’s

Surgimento e características do NoSQL

Banco de dados NoSQL surgiram particularmente em resposta aos três desafios (os três V's): **volume** de dados (data volume); **velocidade** de dados (data velocity) e **variedade** de dados (data variety)

Volume de dados

O aumento do volume de dados tem sido um dos principais motivos pela adição de NoSQL.

Grandes bases de dados são complicadas de manipular em um banco relacional.

Tempo de execução de consultas aumenta muito (joins são lentos)

Velocidade de dados

Ou seja, **velocidade em que os dados mudam**.

O banco de dados precisa suportar muitas edições (velocidade de leitura e escrita) e com picos de atividades (banco relacional podem não suportar altos picos de leitura/escrita).

Velocidade de dados

Outro problema com a velocidade de dados é a medida em que **a estrutura dos dados muda**.

Além da rápida mudança dos dados, o banco deve lidar com a rápida mudança no modelo de dados (banco relacional não suporta tal mudança facilmente).

Variedade de dados

Os dados podem ser **densos** (muitos dados sobre determinada entidade) ou **esparsos** (poucos dados), conectados ou desconectados, com regular ou irregular estrutura.

Limitações: bancos relacionais terão muitos campos nulos.

Os 5 V’s

Hoje já são considerados os 5 V’s:

Além de **Volume**, **Velocidade** e **Variedade**:

Veracidade: com grande volume de dados, muitos podem não estar 100% corretos. A acurácia depende da veracidade da fonte de dados.

Valor: Se os dados processados não gerarem valor para o negócio, todo o esforço e investimento tornam-se inúteis.

Replicação master-replica

Replicação master-replica

Antes de conhecer os conceitos relacionados aos bancos de dados NoSQL, vamos entender a ideia de replicação de dados em um ambiente de computação distribuída (cluster).

Replicação master-replica

O réplica terá a cópia dos dados do master.
Escrita ocorre apenas no master e leitura em ambos.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Replicação master-replica

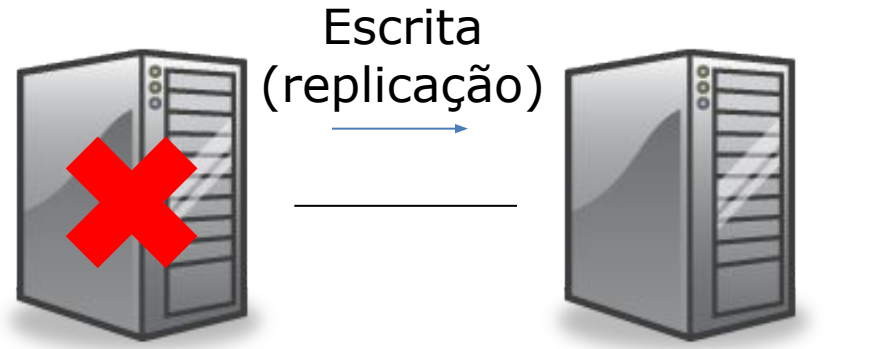
Se houver alguma estratégia de confirmação de escrita antes de propagar, a leitura no *replica* pode ser desatualizada (perdemos a consistência por pouco tempo)

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Replicação master-replica

Uma falha no master antes da propagação, significa perda de dados

Falha no máster antes da propagação.

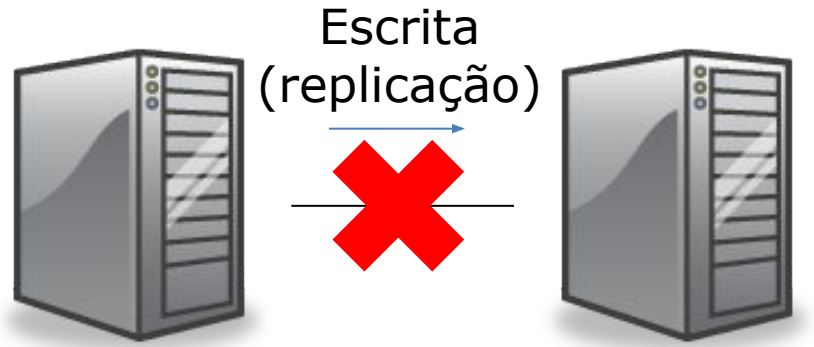


Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Replicação master-replica

Uma falha na comunicação não compromete a leitura e escrita dos dados, mas a leitura no réplica será desatualizada.

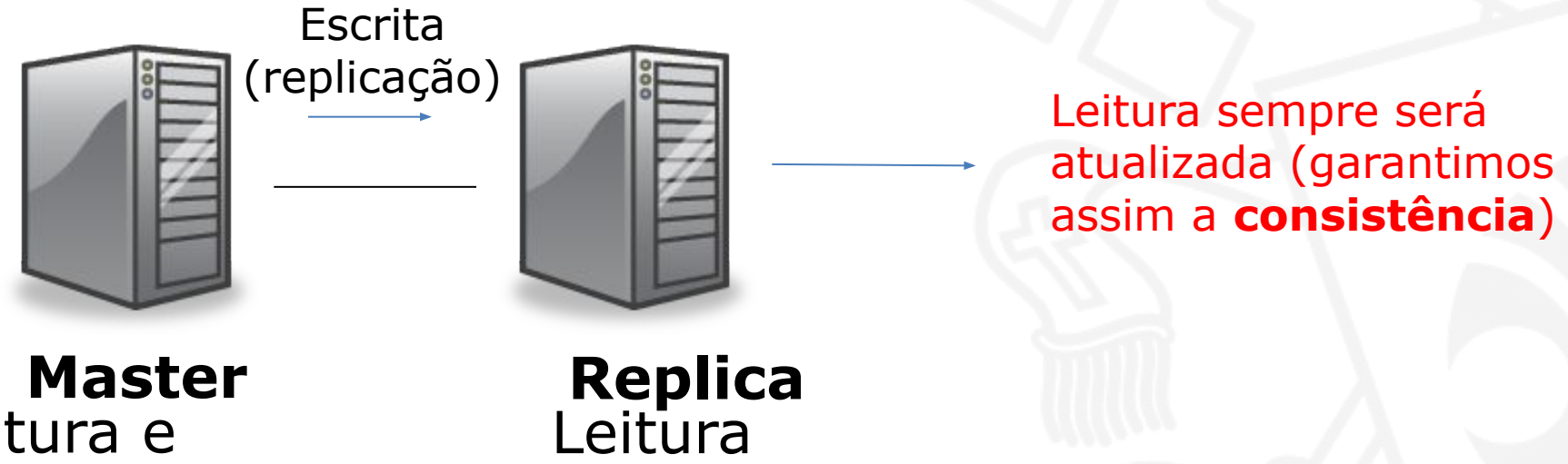
Falha na comunicação entre máster e réplica.



Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Replicação master-replica

Imagine agora que um dado será considerado salvo apenas após a replicação (confirmação de escrita no réplica).



Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Aumentar a disponibilidade

Imagine agora que um dado será considerado salvo apenas após a replicação (confirmação de escrita no réplica).

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Aumentar a disponibilidade

Observe que também é importante avaliar a questão sobre **consistência** e **performance** quando não há falhas na comunicação.

Há estudos da Amazon que mostram quedas nas vendas com o aumento do tempo de resposta

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Consistência e performance

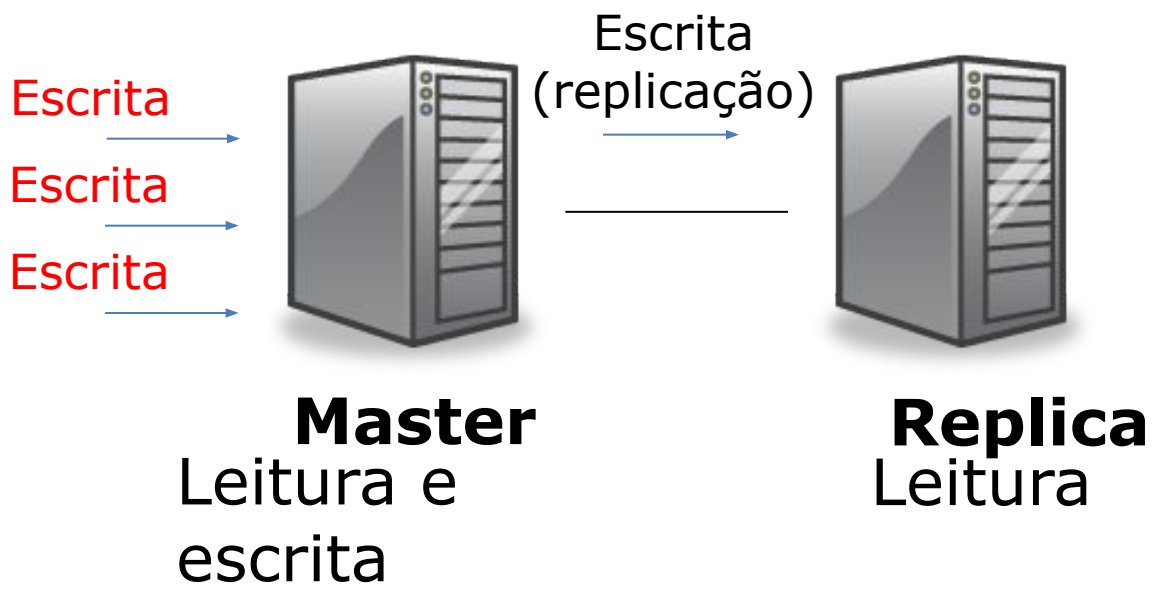
- A questão do que é mais importante depende muito da regra de negócio da aplicação.
- Em algumas situações um leve **atraso no tempo de reposta** pode ser mais prejudicial do que um **atraso na atualização dos dados** (ex.: site de notícia)

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Replicação masterless

Replicação master-replica

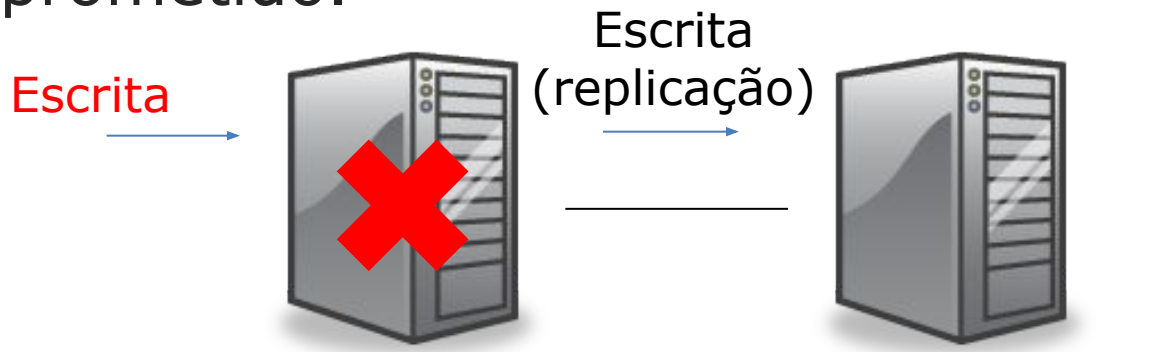
Limitação: todas as escritas serem realizadas no **master**. Há dois problemas claros: (1) queda na performance com muitas requisições;



Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Replicação master-replica

Limitação: todas as escritas serem realizadas no **master**. Há dois problemas claros: (2) uma falha no servidor, o serviço será comprometido.

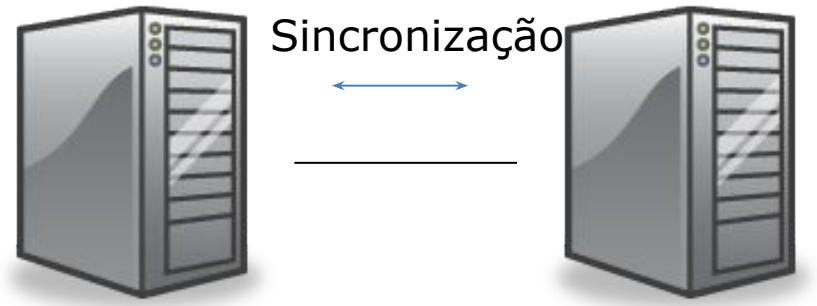


Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Replicação *masterless*

Alternativa: muitos bancos NoSQL são baseados na ideia de que qualquer servidor pode receber requisições de leitura e escrita.

Cluster

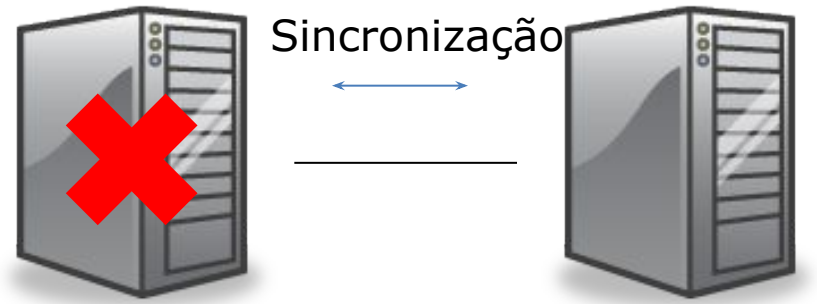


Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Replicação *masterless (p2p)*

Assim, mesmo com falha em um dos servidores do cluster, o serviço se mantem disponível.

Cluster



Nodo
Leitura e escrita

Nodo
Leitura e escrita

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Aumentar a disponibilidade

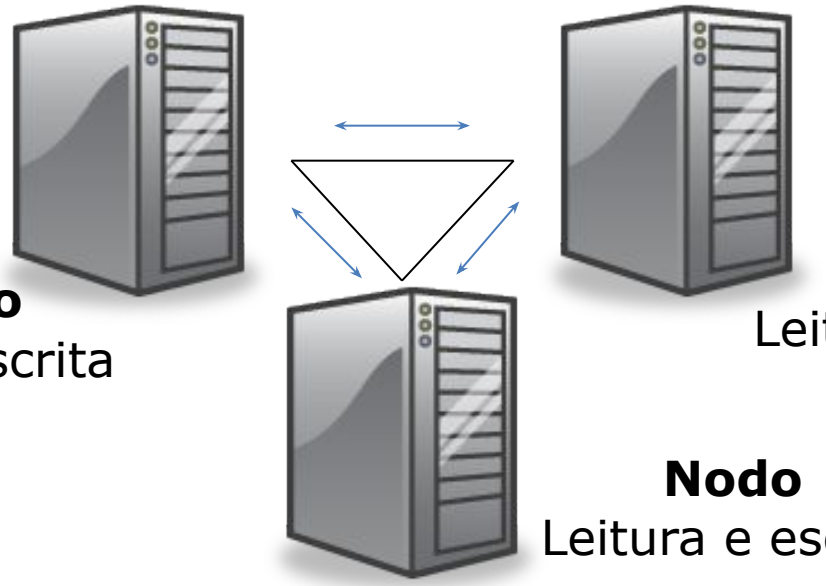
Escrita concorrente: conflito. Na sincronização é necessário verificar qual dado gravado é correto.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Aumentar a disponibilidade

O modelo p2p normalmente é composto por vários servidores em clusters (dezenas, centenas ou até milhares)

Cluster



Nodo
Leitura e escrita

Nodo
Leitura e escrita

Nodo
Leitura e escrita

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Aumentar a disponibilidade

Se há a necessidade por **alta disponibilidade de escrita**, o modelo *masterless* (p2p) pode ser o mais adequado.

A escolha do banco passa pela necessidade de avaliar tais características que cada um pode oferecer.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

CAP Theorem

CAP Theorem

Para muitas soluções em banco de dados é essencial manter **alta disponibilidade** (serviço funcionando por maior tempo possível).

Neste cenário, pelo menos dois servidores são necessários (com computação em nuvem, vários podem ser utilizados)

CAP Theorem

Teorema **CAP** (CAP Theorem):

Consistência (**C** - Consistency); Alta disponibilidade (**A** - Availability); e Tolerância a particionamento dos dados na rede (**P** - Partition-tolerance)

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

CAP Theorem

Dentre as três partições, o teorema CAP diz que é possível apenas obter **duas** delas (impossível ter todas simultaneamente).

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

CAP Theorem

Consistência (Consistency): quando dados são alterados por uma operação, qualquer operação posterior (ex. busca) deverá utilizar o dado atualizado.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

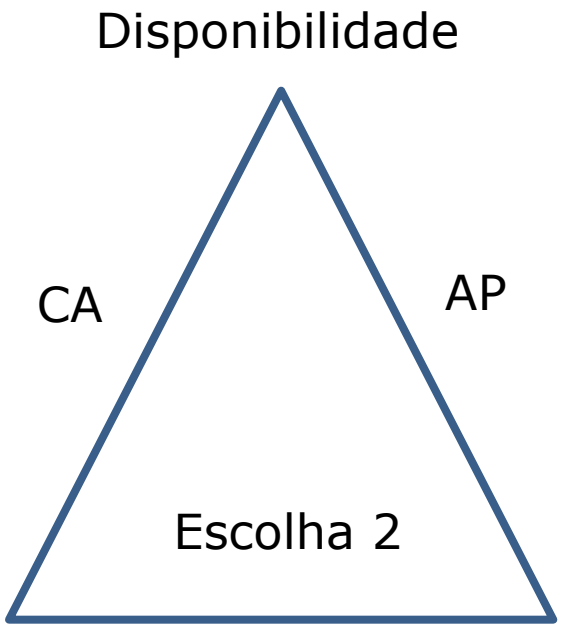
CAP Theorem

Disponibilidade (Availability): o banco deve estar disponível sempre para leitura e escrita.

Tolerância a particionamento (Partition-tolerance): servidores devem suportar falhas na comunicação entre eles por tempo indeterminado.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

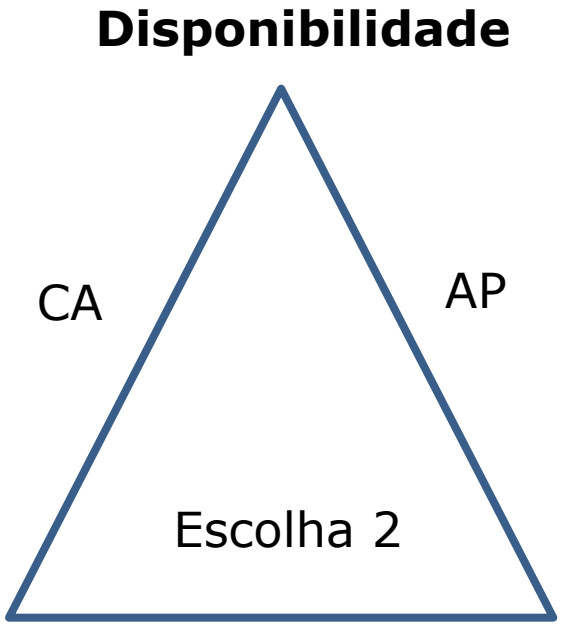
CAP Theorem



quando dados são alterados por uma operação, qualquer operação posterior deverá utilizar o dado atualizado.

Figura adaptada de: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

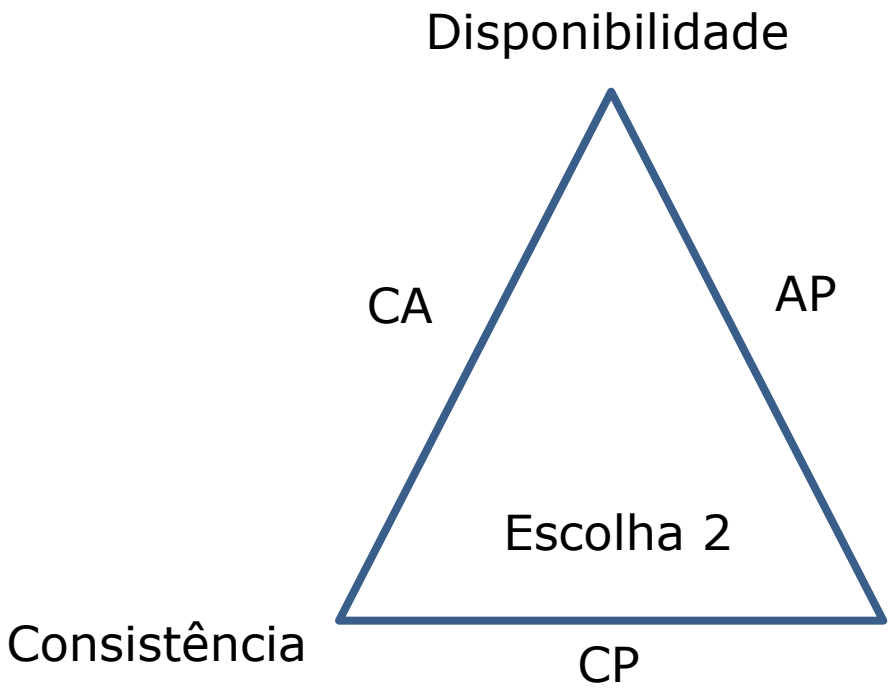
CAP Theorem



O banco deve estar disponível sempre para leitura e escrita

Figura adaptada de: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

CAP Theorem



servidores devem suportar falhas na comunicação entre eles por tempo indeterminado

Figura adaptada de: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

CAP Theorem

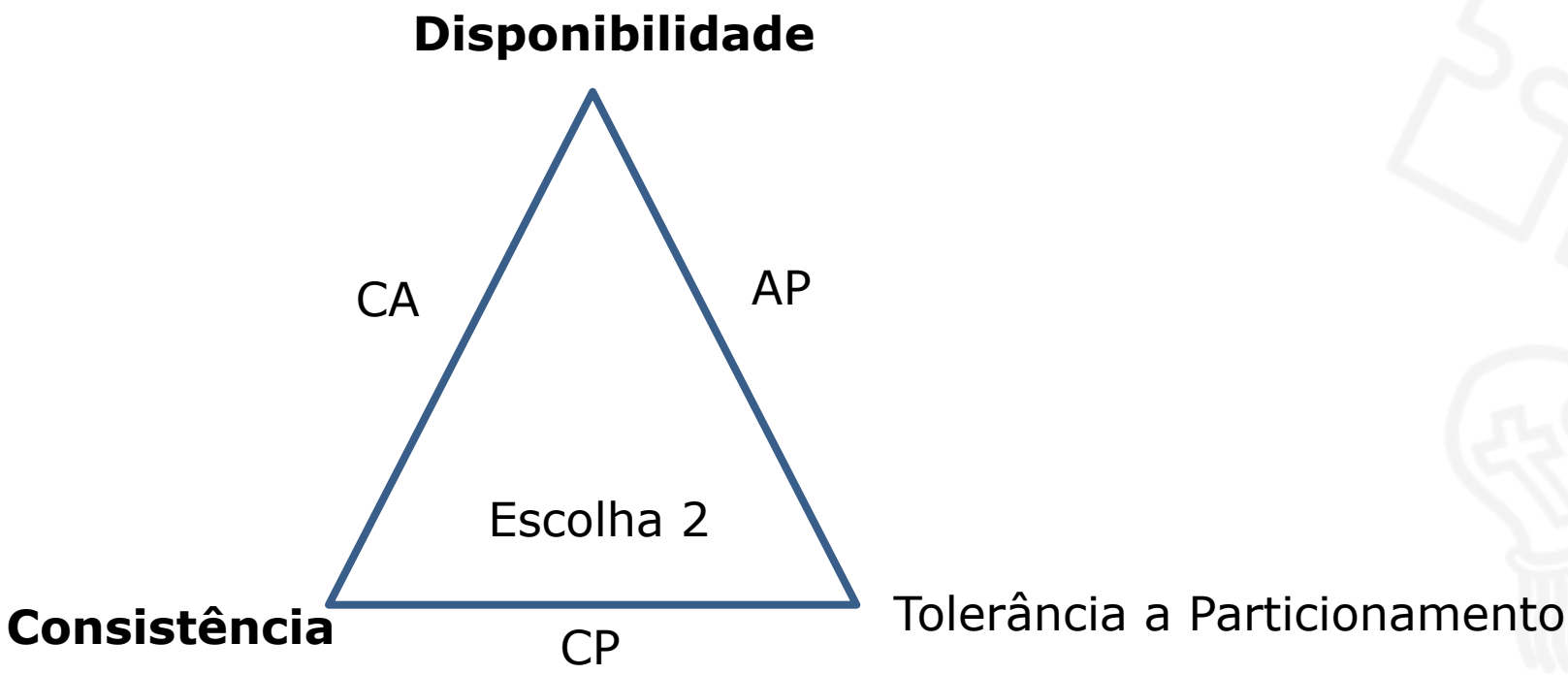
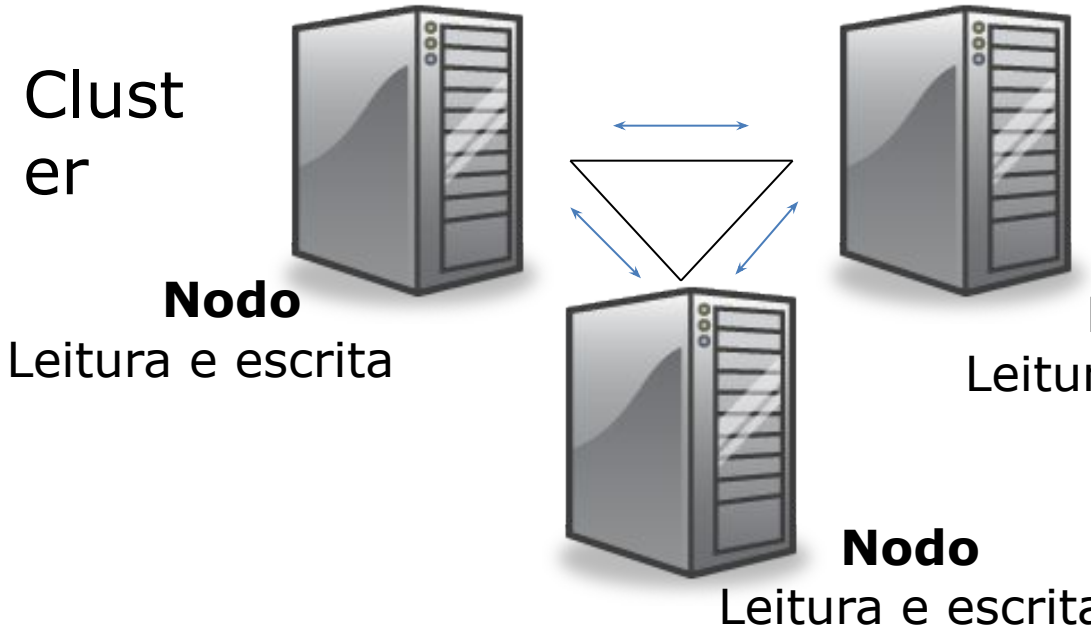


Figura adaptada de: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

CAP Theorem - Consistência

Forte consistência e tolerância a particionamento (CP)

Situação em que necessita de forte consistência (leitura dos dados sempre atualizada).



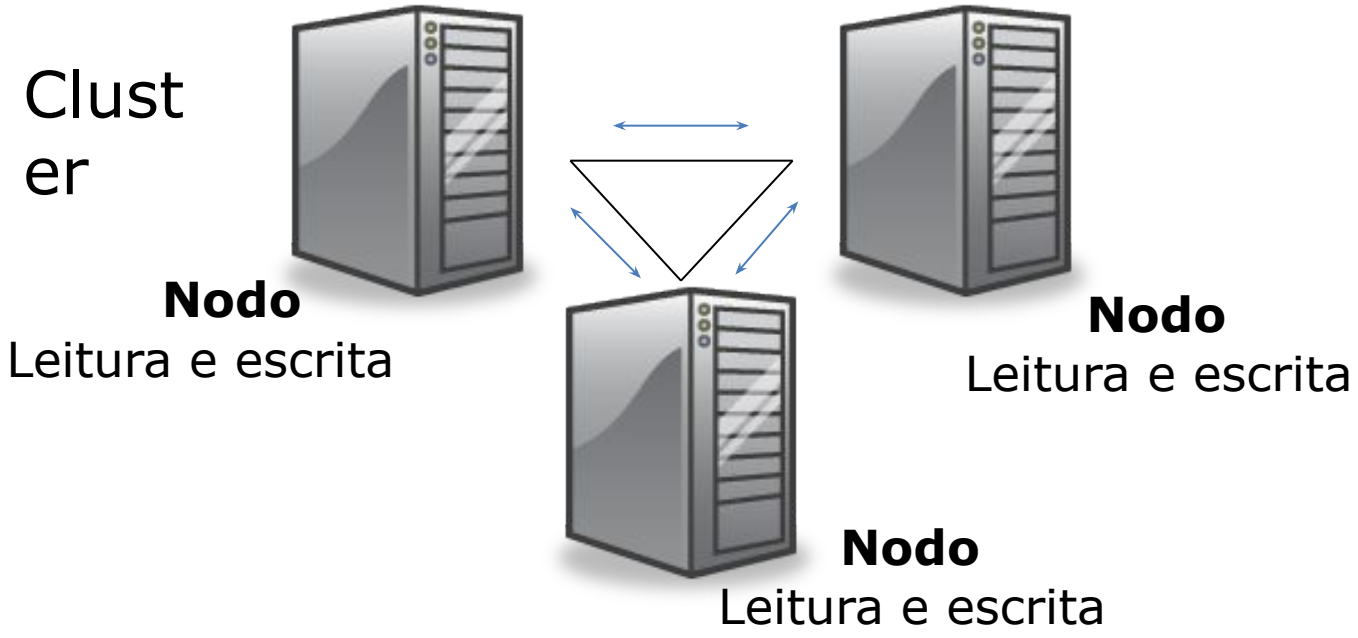
Consenso: suponha um fator de replicação N igual a 3 (figura ao lado). Considere:

- R como número de nós que devem ser contatados para confirma uma leitura;

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Forte consistência e tolerância a particionamento (CP)

Situação em que necessita de forte consistência (leitura dos dados sempre atualizada).



Consenso: suponha um fator de replicação N igual a 3 (figura ao lado). Considere:

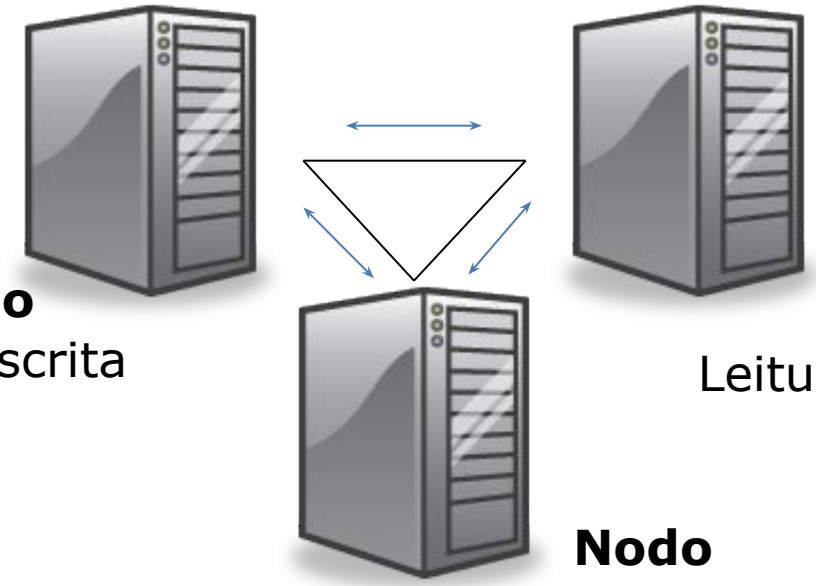
- W números de nós a serem contatados para confirmar a escrita.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Forte consistência e tolerância a particionamento (CP)

Situação em que necessita de forte consistência (leitura dos dados sempre atualizada).

Clust
er



Nodo
Leitura e escrita

Nodo
Leitura e escrita

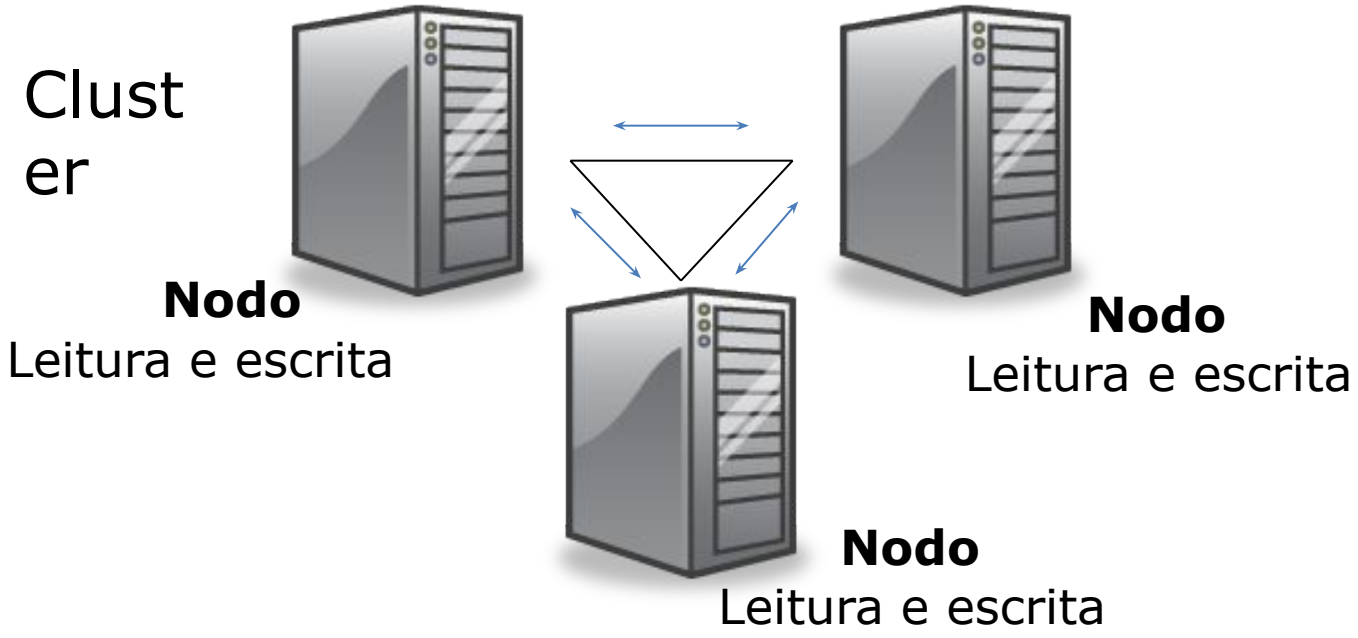
Nodo
Leitura e escrita

Consenso: O relacionamento entre R, W e N pode ser considerado pela desigualdade: $R + W > N$ (leitura altamente consistente).

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Forte consistência e tolerância a particionamento (CP)

Situação em que necessita de forte consistência (leitura dos dados sempre atualizada).

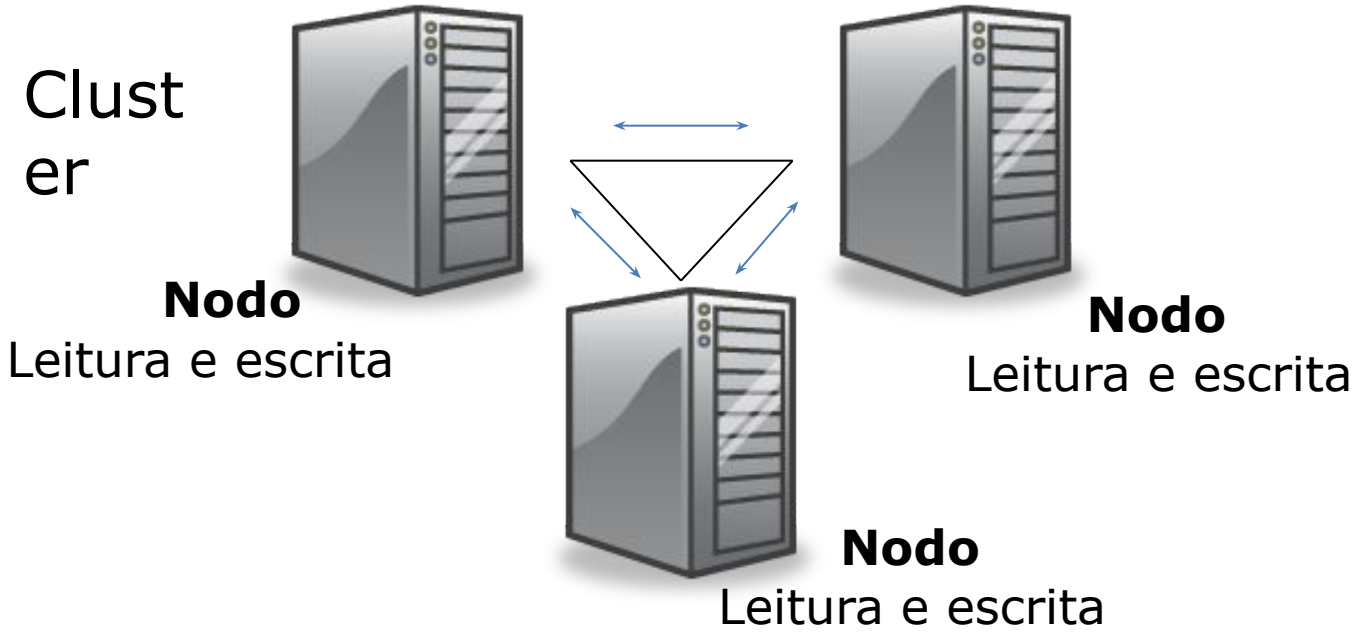


Consenso: Agora considere $R = 1$ e $W = 3$ (N continua sendo 3). Portanto, para escrever no banco é necessária a confirmação de escrita nos três nós (**quórum**).

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Forte consistência e tolerância a particionamento (CP)

Situação em que necessita de forte consistência (leitura dos dados sempre atualizada).

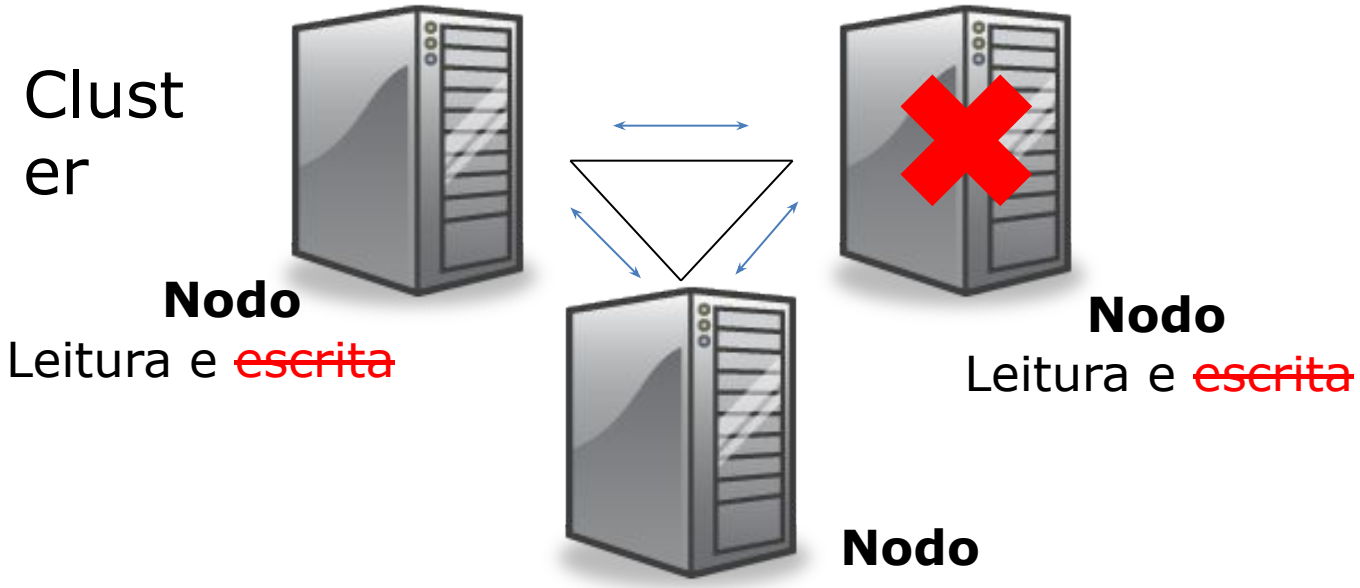


Consenso: Agora considere $R = 1$ e $W = 3$ (N continua sendo 3). Porém, para leitura, basta a confirmação de apenas um nó. Assim: $1(R) + 3(W) > 3(N)$.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Forte consistência e tolerância a particionamento (CP)

Situação em que necessita de forte consistência (leitura dos dados sempre atualizada).



Consenso: Com a falha de um nó, a escrita é comprometida (não há quórum).

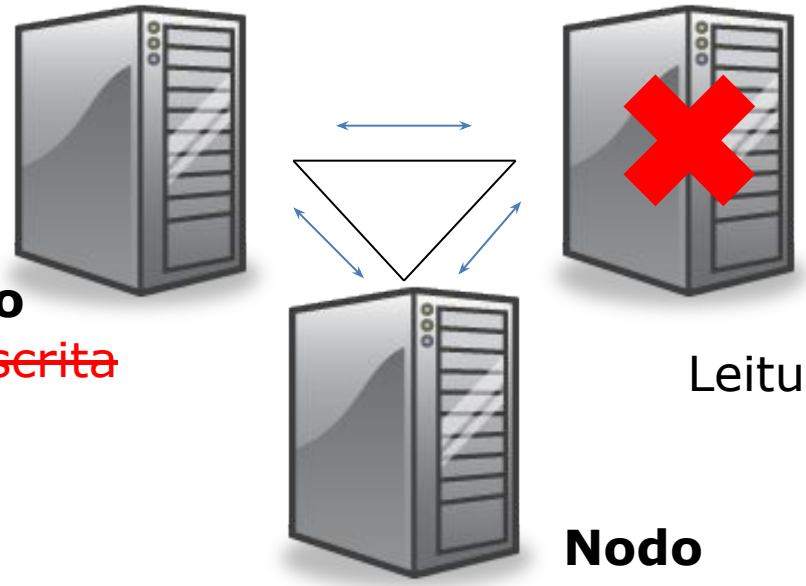
Perde-se a disponibilidade (A) para escrita (lembre-se que $W = 3$).

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Forte consistência e tolerância a particionamento (CP)

Situação em que necessita de forte consistência (leitura dos dados sempre atualizada).

Clust
er



Nodo
Leitura e ~~escrita~~

Nodo
Leitura e ~~escrita~~

Nodo
Leitura e ~~escrita~~

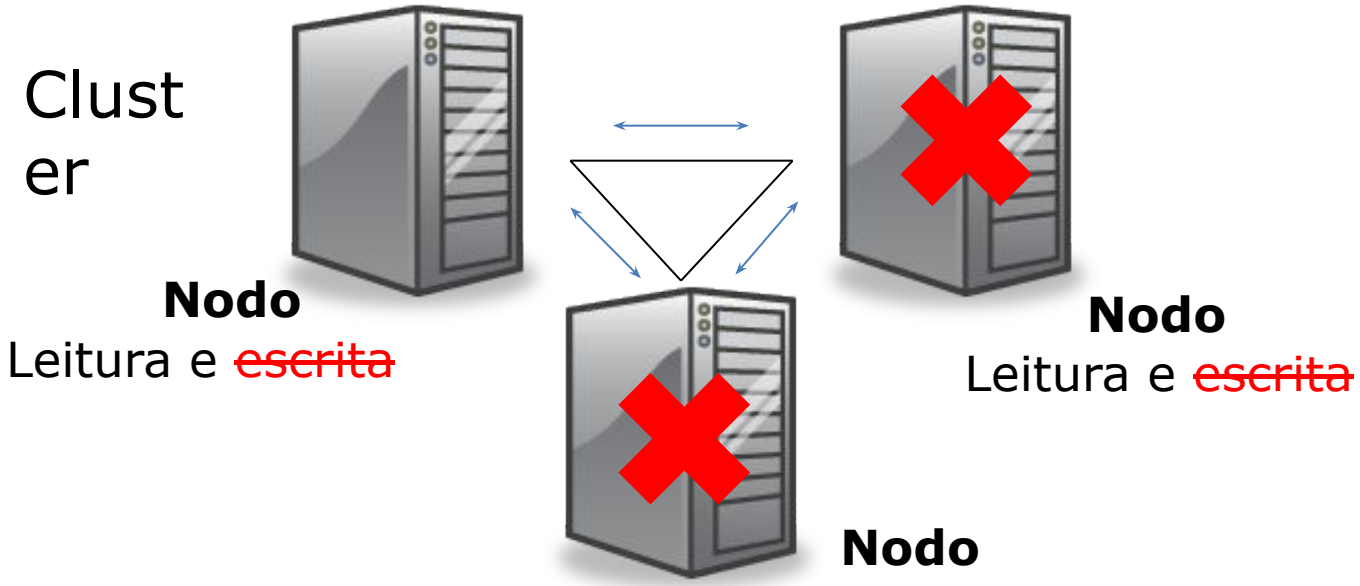
Consenso: Com a falha de um nó, a escrita é comprometida (não há quórum).

Mas a leitura ainda pode ocorrer em qualquer um dos demais nós ($R = 1$).

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Forte consistência e tolerância a particionamento (CP)

Situação em que necessita de forte consistência (leitura dos dados sempre atualizada).



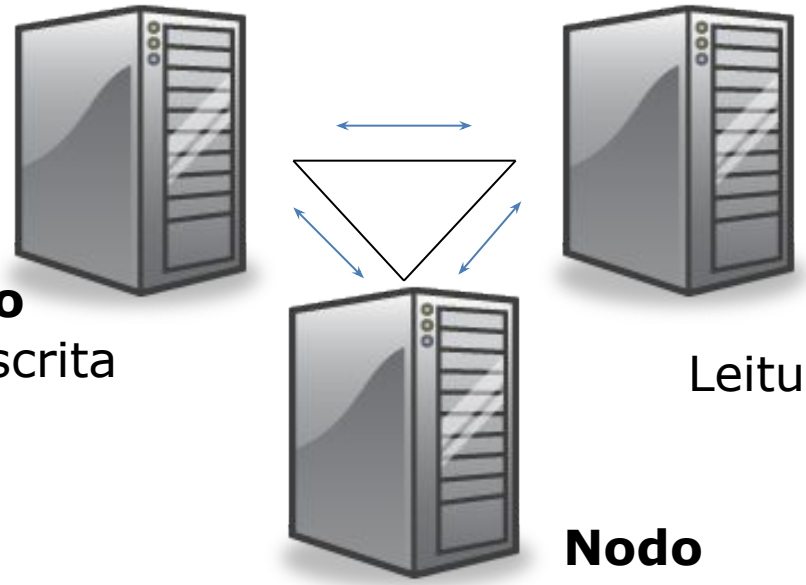
Consenso: Mesmo com falha em dois nós a leitura ainda pode ser realizada e com consistência ($R = 1$).

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Forte consistência e tolerância a particionamento (CP)

Situação em que necessita de forte consistência (leitura dos dados sempre atualizada).

Clust
er



Nodo
Leitura e escrita

Nodo
Leitura e escrita

Nodo
Leitura e escrita

Exemplos de sistemas que implementa o conceito de consenso:

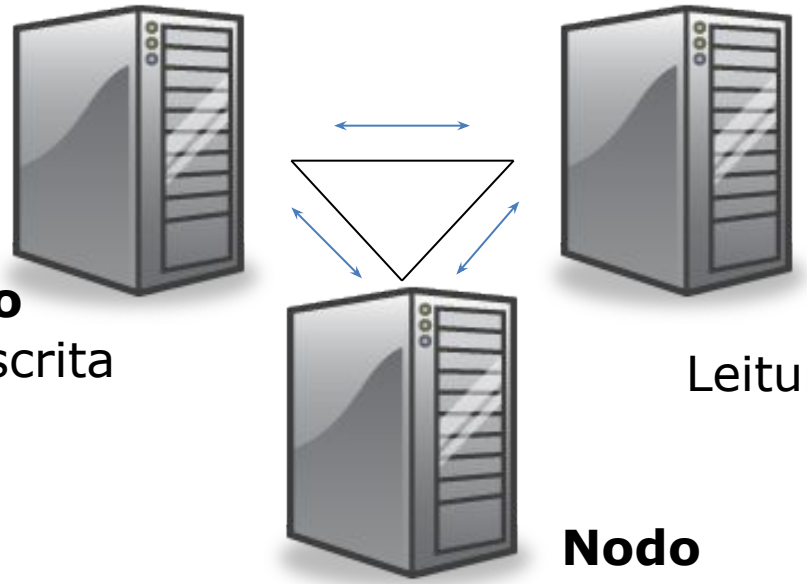
- MongoDB
- Google Big Table

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

CAP Theorem - Alta disponibilidade

Alta disponibilidade e tolerância a particionamento (AP)

Situação em que deve-se garantir alta disponibilidade (de escrita).

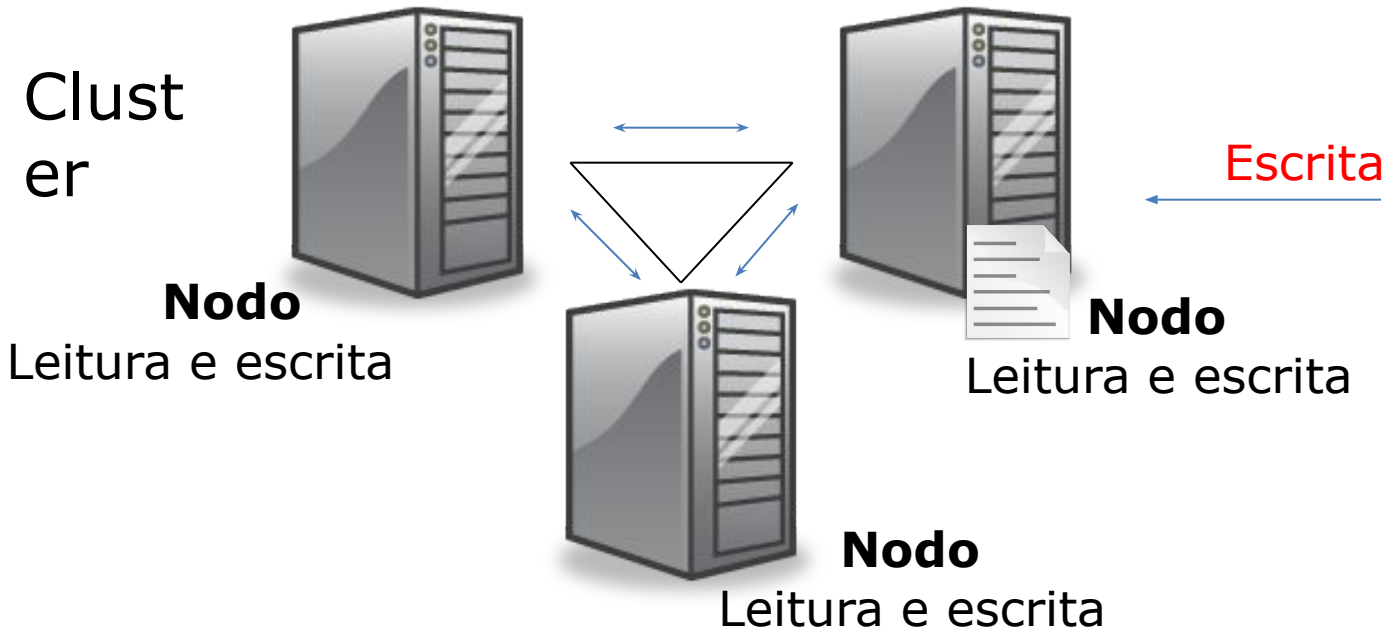


Eventual-consistency : sempre disponível para escrita e depois sincroniza os dados com os demais nós.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Alta disponibilidade e tolerância a particionamento (AP)

Situação em que deve-se garantir alta disponibilidade (de escrita).

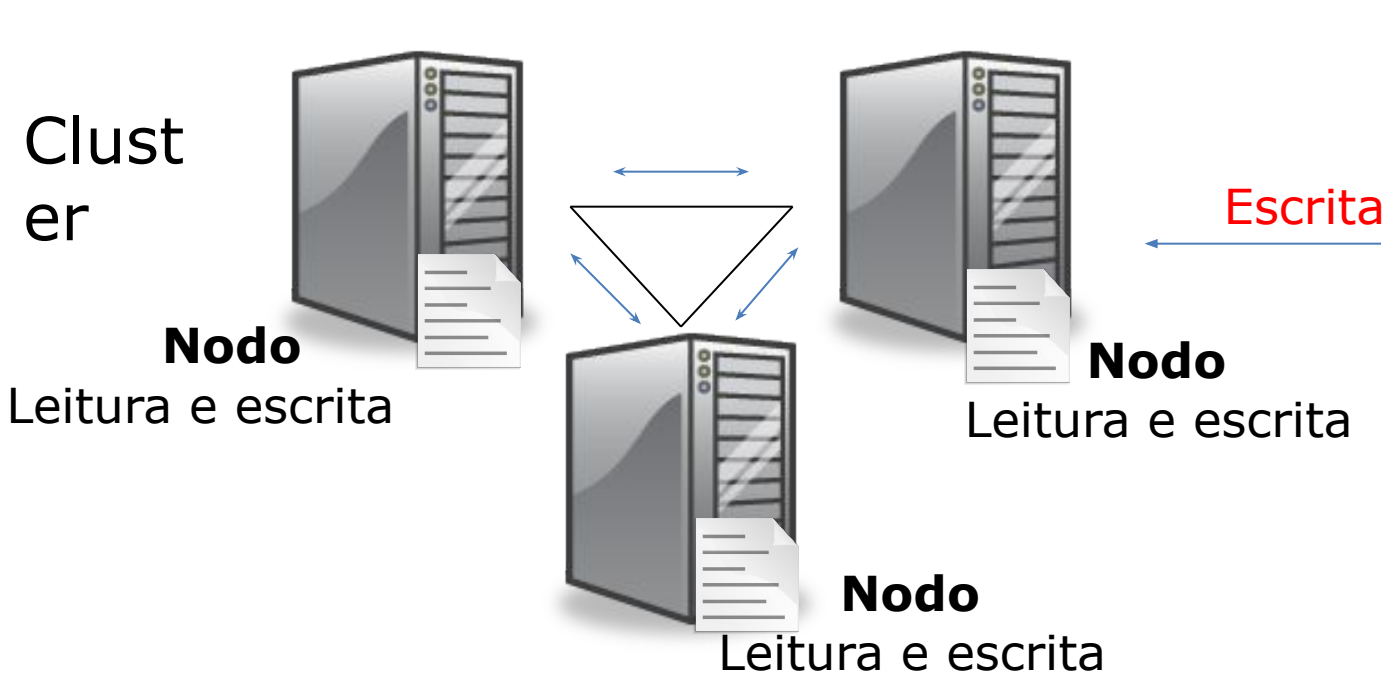


Na escrita os dados são replicados aos demais servidores no cluster

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Alta disponibilidade e tolerância a particionamento (AP)

Situação em que deve-se garantir alta disponibilidade (de escrita).

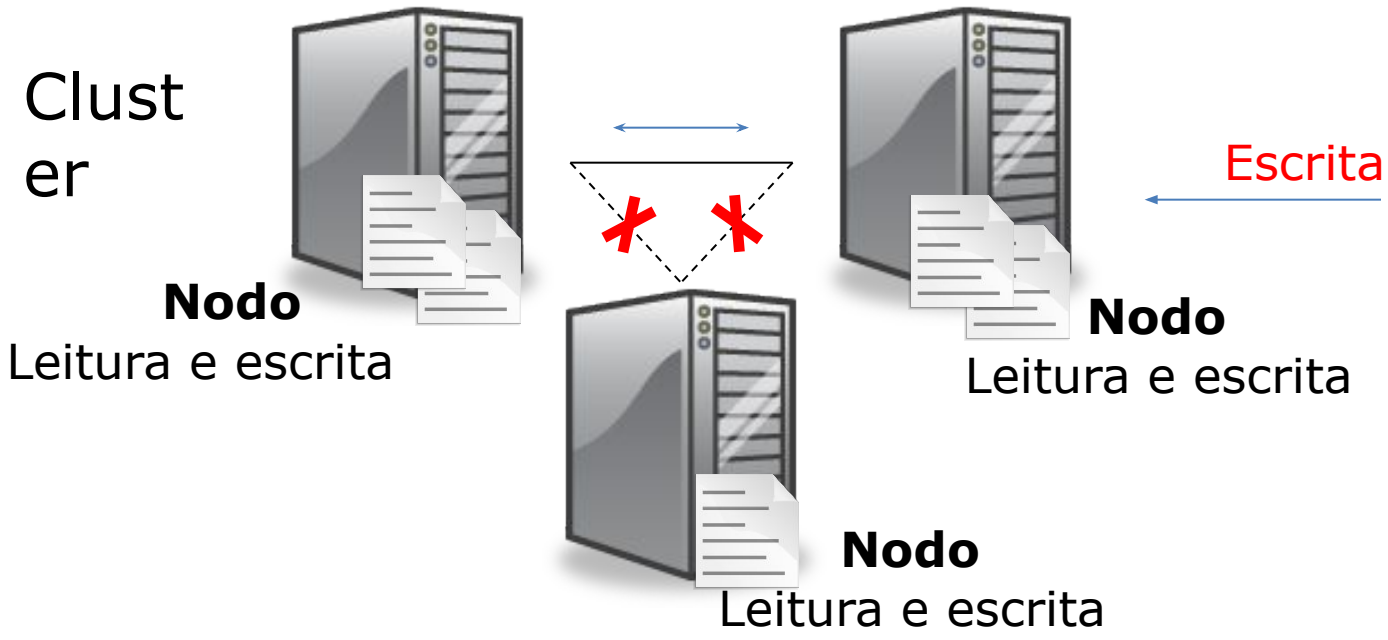


Na escrita os dados são replicados aos demais servidores no cluster

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Alta disponibilidade e tolerância a particionamento (AP)

Situação em que deve-se garantir alta disponibilidade (de escrita).



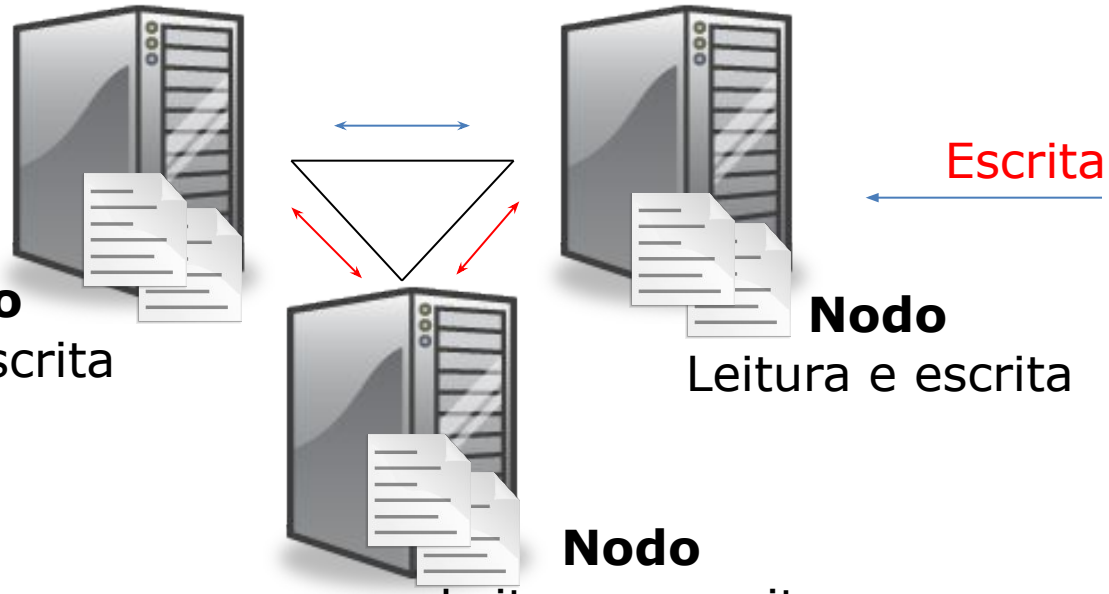
Se em uma escrita a comunicação com algum nó falhar, os dados são replicados apenas aos servidores disponíveis sem que o serviço fique indisponível (escrita ainda é possível).

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Alta disponibilidade e tolerância a particionamento (AP)

Situação em que deve-se garantir alta disponibilidade (de escrita).

Clust
er



Nodo
Leitura e escrita

Escrita

Nodo
Leitura e escrita

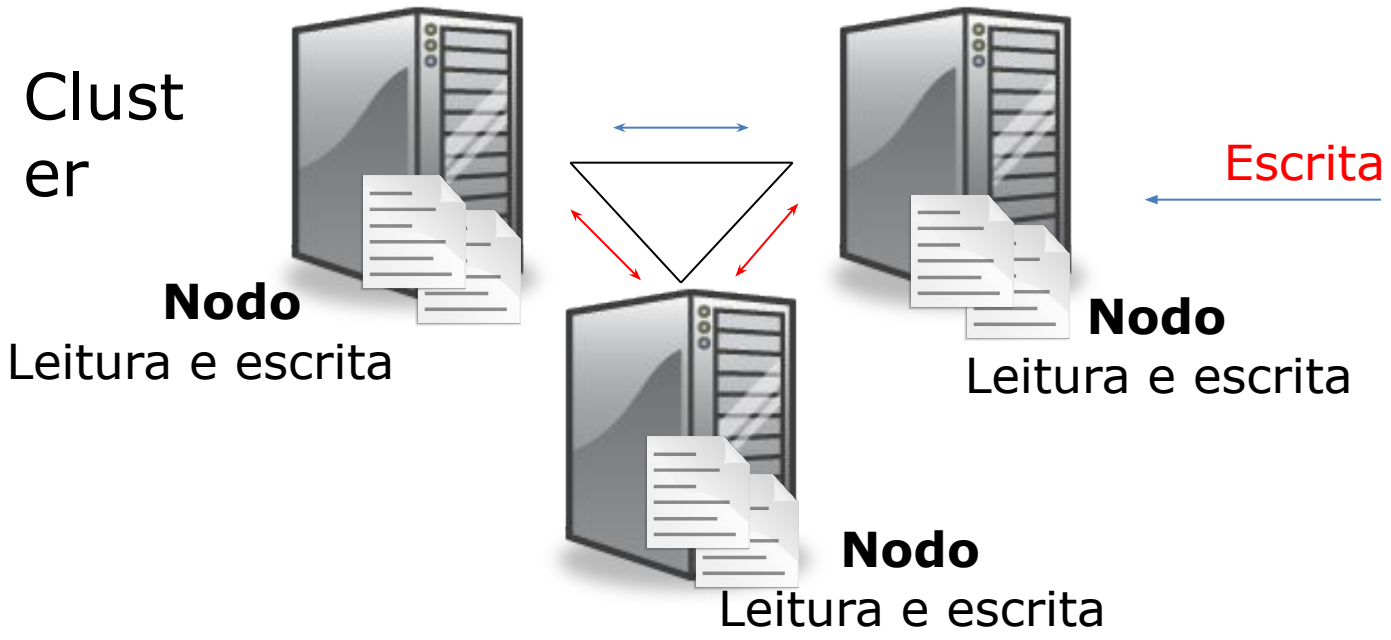
Nodo
Leitura e escrita

Após o reestabelecimento da comunicação, os dados são sincronizados com os demais servidores.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Alta disponibilidade e tolerância a particionamento (AP)

Situação em que deve-se garantir alta disponibilidade (de escrita).



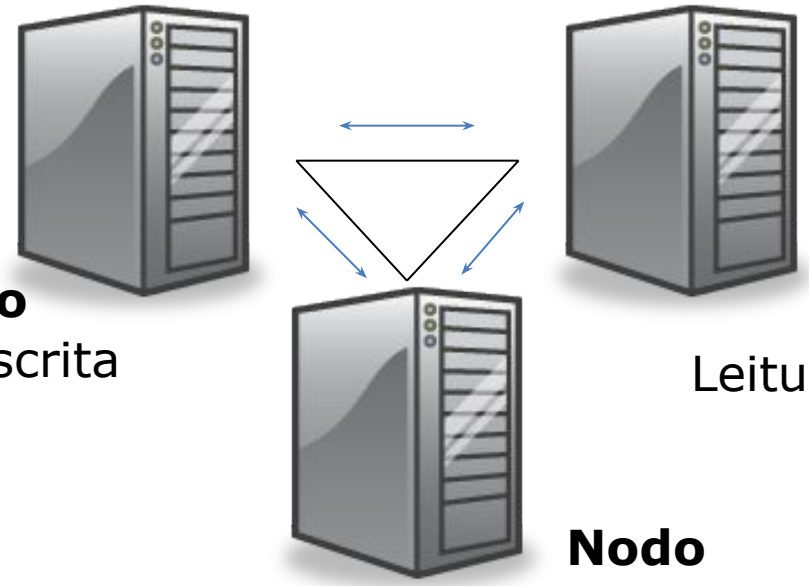
Observe que há uma janela de inconsistência durante o período anterior à sincronização dos dados (leitura desatualizada).

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Alta disponibilidade e tolerância a particionamento (AP)

Situação em que deve-se garantir alta disponibilidade (de escrita).

Clust
er



Nodo
Leitura e escrita

Nodo
Leitura e escrita

Nodo
Leitura e escrita

Exemplos de sistemas que implementa o conceito:

- Cassandra
- Riak
- DynamoDB

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Forte consistência e alta disponibilidade (CA)

- Situação em que o sistema garante apenas alta disponibilidade (em um único nó). Uma máquina não pode ser particionada.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Forte consistência e alta disponibilidade (CA)

- Se houver alguma falha o sistema fica completamente indisponível até que o nó volte a operar.
- Ex.: Banco de dados Relacionais.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

BASE e NoSQL

Alguns desafios para persistências de dados

Atomicidade (a transação é executada totalmente ou **não** é executada), **C**onsistência (sistema sempre consistente após uma operação), **I**solamento (transação não sofre interferência de outra transação concorrente), e **D**urabilidade (o que foi salvo não é mais perdido)

- Força a consistência ao final de cada transação

NoSQL: BASE

Propriedades BASE:

Basically Available – Basicamente Disponível.

Soft-State – Estado Leve

Eventually Consistent – Eventualmente Consistente.

NoSQL: BASE

Uma aplicação funciona basicamente todo o tempo (**Basicamente Disponível**);

O estado do sistema pode mudar (mesmo durante o período sem escrita – devido a consistência eventual) (**Estado Leve**); e

NoSQL: BASE

O sistema torna-se consistente em algum momento após operações de escrita (**Eventualmente Consistente**).

Problema do mundo real

Álbum de músicas

Considere uma aplicação que contém o perfil de **músicos**, **bandas**, **álbuns** e **músicas**.

Iremos resolver um problema do mundo real e veremos quais as limitações ao usar um modelo Relacional.

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Especificação

Ex.: Um disco pode conter o **número de semanas que ficou em primeiro lugar** na Billboard* e o **número de músicas na primeira posição** (nem todo disco irá figurar nas primeiras posições)

* Revista norte americana sobre indústria da música

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Especificação

Cadastro dos álbuns:

além da **banda** e das **músicas**, um álbum também possui por padrão **ano de lançamento**, **ilustrador da capa**, **produtor** ou qualquer outra informação necessária.

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Especificação

É necessário armazenar informações sobre o **estúdio**.

Sabe-se que muitos discos são gravados em um único estúdio, mas podem ser gravados em mais de um ou em nenhum (disco independente).

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Especificação

Observe neste cenário que **muitos campos** não farão sentidos **para muitos discos**.

Seria necessário uma estrutura diferente para cada disco (usar modelo relacional?).

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Modelo Relacional

Album

Cod	nome	artista	dataLanc	estudio	produtor	semanasEmPrimeiro	numMusicasEmPrimeiro
1	The Dark Side Of The Moon	Pink Floyd	4/29/1973		Pink Floyd	1	3
2	Nevermind	Nirvana	1/11/1992	Sound City Studios, Smart Studios (Madison)		1	1
3	Independente	independente	01/01/2017				

Mesma estrutura para todos os registros.

Musica

CodMusica	Nome	CodAlbum

Observe que seria necessário também uma nova tabela chamada Estúdio, uma vez que podem haver mais de um estúdio para cada Álbum

Modelo Relacional

Album

Cod	nome	artista	dataLanc	estudio	produtor	semanasEmPrimeiro	numMusicasEmPrimeiro
1	The Dark Side Of The Moon	Pink Floyd	4/29/1973		Pink Floyd	1	3
2	Nevermind	Nirvana	1/11/1992	Sound City Studios, Smart Studios (Madison)		1	1
3	Independente	independente	01/01/2017				

Musica

CodMusica	Nome	CodAlbum

Uma desvantagem clara para este tipo de problema diz respeito a **quantidade de colunas para representar cada campo possível** (que só fará sentido para alguns registros).

Modelo Relacional (solução alternativa)

Album

Cod	**codBanda**	**nome**
1	1	Master of Puppets
2	1	...And Justice for All

Solução alternativa usando o modelo Relacional que visa evitar a inclusão de registros nulos.

Valores

codAlbum	**codAtributo**	**valor**
1	**1**	03/03/86
2	**1**	25/08/88
1	**2**	Sweet Silence Studio

Atributos

cod	**nome**
1	Data de lançamento
2	Estudio

Modelo Relacional (solução alternativa)

- Observe que a solução torna a manipulação do banco muito mais complexa.
- Veja o exemplo para uma simples consulta que exibe os detalhes de cada álbum.

```
SELECT atr.nome, val.valor
FROM atributos atr JOIN valores val ON val.id-atributo = atr.id
WHERE val.id-album = 1;
```

Limitações

- Problemas mais complexos podem transparecer melhor as limitações do modelo relacional. Ex.: sites de e-commerce, catálogos (Netflix, spotify), etc..

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Limitações

- O objetivo não é resolver problemas não solucionáveis com o banco relacional, e sim obter soluções mais **simples** e **práticas** (e **escaláveis**).

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Dicas!

Links para saber mais:

Para conhecer mais sobre tipos de bancos NoSQL:

https://hostingdata.co.uk/nosql-database/

Banco de dados Chave-Valor

Modelo de votação

Vimos no início a especificação do nosso problema para cadastro de álbuns de músicas.

E vimos que o modelo relacional pode não ser o mais adequado para as características deste tipo de problema.

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Modelo de votação

Agora suponha que gostaríamos de implementar um serviço de votação (exemplo apresentado anteriormente). Teremos uma entidade com o **título** da eleição, uma outra com os **candidatos**, e uma última com os **votos**.

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Modelo de votação

Conforme vimos anteriormente, o problema poderia ser modelado para um banco relacional. No entanto, precisamos alta **disponibilidade** e **performance** (suponha um sistema online durante um programa ao vivo).

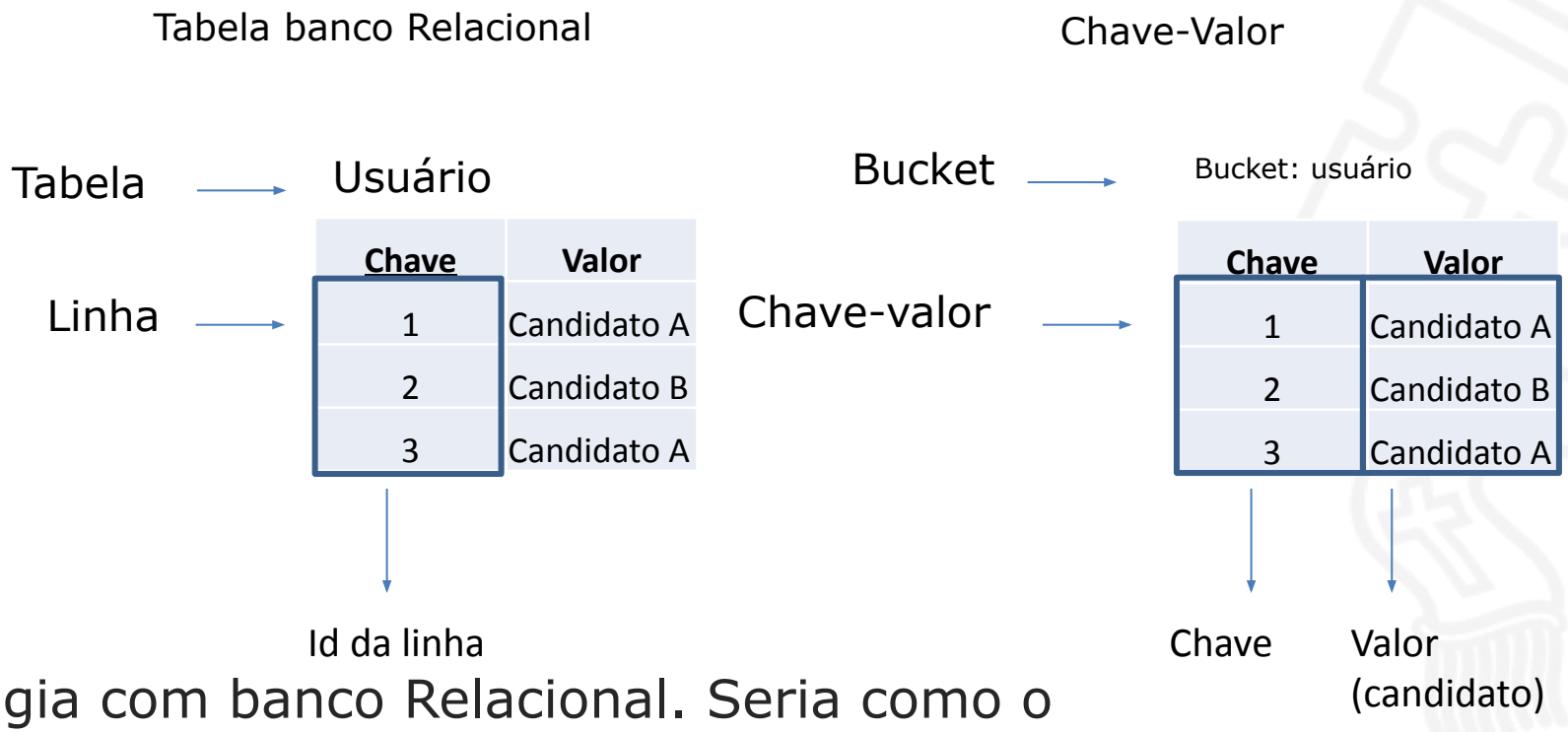
Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Modelo chave-valor

Para o problema de votação, é necessário armazenar dados do **usuário** e em qual **artista** ele votou. Problema relativamente simples, ideal para banco de dados do tipo chave-valor (um banco também muito simples).

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Modelo chave-valor



Analogia com banco Relacional. Seria como o banco inteiro fosse uma única tabela.

Modelo chave-valor

Por serem muito rápidos, este tipo de banco pode usado em diversas situações.

Ex: compartilhamento de dados de seção de usuário entre vários servidores de aplicação.

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Modelo chave-valor

Um exemplo é o banco chave-valor **Redis**.

Outro banco chave-valor é o Riak. São soluções *open source* e estão disponível para Linux e MacOS (Riak).

Modelo chave-valor

O banco pode armazenar qualquer tipo de dados.

Caracteriza-se por ser um banco de dados distribuído de alta disponibilidade.

Modelo chave-valor

Redis significa REmote DIctionary Server e foi criado por Salvatore Sanfilippo. Armazena seus dados em memória (extremamente rápido para escrita e leitura), mas possui opção para persistir em disco.

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Modelo chave-valor

Acesse o site redis.io e veja a documentação e experimento o console de comandos online

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

Desafio 1 - Manipulando dados no Redis

Criando objetos no Redis

Para criar objetos no Redis, precisamos especificar dois valores: **chave**, **valor**.

Chave-Valor

Votacao

Chave	Valor

Criando objetos no Redis

Chave: login do usuário
Valor: o voto do usuário

Chave-Valor

Votacao

Chave	Valor
usuarioA	candidato1

Chave-valor

No Redis utilizamos o comando **set** para especificar a chave e o valor da mesma no banco.

```
> set usuarioA "candidato1"
```

chave valor

O valor é uma string (cadeia de caractere).

Criando objetos no Redis

Chave: login do usuário
Valor: o voto do usuário

Chave-Valor

Votacao

Chave	Valor
usuarioA	candidato1

Chave-valor

No Redis utilizamos o comando **set** para especificar a chave e o valor da mesma no banco.

Esta operação é semelhante a um comando de insert no SQL (SGBDR):

```
INSERT
INTO TABELA (chave, valor)
VALUES ("usuarioA", "candidato1");
```

Criando objetos no Redis

Chave-Valor

Votacao

Chave	Valor
usuarioA	candidato1

Chave-valor

Chave: login do usuário
Valor: o voto do usuário

Para recuperar o registros, utilizamos o método get

```
> get usuarioA
> "candidato1"
```

Resultado da consulta

Esta operação é semelhante a um comando de select no SQL (SGBDR):

```
SELECT valor
FROM TABELA
WHERE chave = "usuarioA"
```

Criando objetos no Redis

Chave-Valor

Votacao

Chave	Valor
usuarioA	candidato1

Chave-valor →

Chave: login do usuário
Valor: o voto do usuário

Para recuperar o registros, utilizamos o método get

```
> get usuarioA
> "candidato1"
```

Observe que no Redis (ou qualquer banco chave-valor) a busca é feita pela **chave** e o banco retornar o seu **valor** correspondente. As chaves nunca possuem valores repetidos no banco.

Criando objetos no Redis

Chave-Valor

Votacao

Chave	Valor
usuarioA	candidato1

Chave-valor →

Chave: login do usuário
Valor: o voto do usuário

É possível exibir também o valor de todas as chaves do banco

```
> keys *
```

Esta operação é semelhante a um comando de select no SQL (SGBDR):

```
SELECT chave
FROM TABELA
```

Manipulando objetos no Redis

Neste tipo de banco, normalmente não se diferencia operações de inserção e de atualização

A atualização é inserir (método **set**) novamente uma valor para uma chave.

O que vale é o último valor.

Removendo registros no Redis

Chave: login do usuário
Valor: o voto do usuário

Chave-Valor

Votacao

Chave	Valor
usuarioA	candidato1

Chave-valor

O Redis também permite deletar registros no banco de dados.

Removendo registros no Redis

Chave-Valor

Votacao

Chave	Valor

Chave: login do usuário
Valor: o voto do usuário

O Redis também permite deletar registros no banco de dados.

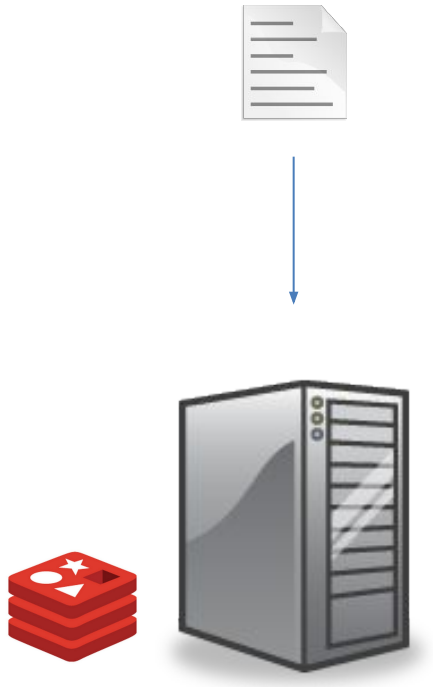
```
> del usuarioA
> (integer) 1
```

Equivalente em SQL

```
DELETE
FROM TABELA
WHERE chave = "usuarioA"
```

Desafio 2 - Lista de Itens visitados

Modelo chave-valor



E-commerce: a cada página acessada pelo usuário é necessário consultar os últimos 25 produtos visualizados por ele e inserir um novo produto nesta lista.

Redis para Lista de Visualizações

Quando usar e não usar

Situações apropriadas para o uso

Armazenamento de informações de seção: Geralmente sessões Web são únicas (possuem *sessionid*). Assim, ao utilizar sistema chave-valor (ao invés de SGBDR), tudo que está relacionado à sessão é armazenado em um único objeto (única operação para armazenar e recuperar).

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Situações apropriadas para o uso

Perfis de usuários (preferências): Como todo usuário tem um *userid* e *username*, além de idioma, fuso horário, produtos que o usuário tem acesso, etc. Tudo pode ser colocado em um único objeto. E assim recuperados com uma única operação.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Situações apropriadas para o uso

Carrinhos de compra: E-commerce possuem carrinhos de compra para cada *userid*. Todas as informações de compra podem ser colocadas no valor da chave *userid* (par chave-valor).

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Situações apropriadas para o uso

Publicação de mensagem: usado para sistemas de mensagens para assinantes. Usa de chaves com TTL (Time to Live – tempo de vida). Estas mensagem são excluídas automaticamente após um tempo, evitando sobrecarga do banco com mensagens desnecessárias.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Situações apropriadas para o uso

Teste o uso do TTl no Redis:

```
> set A "candidato1"
OK
> expire A 5          <-- Chave A irá expirar em 5 segundos
(integer) 1
> get A
"candidato1"
> get A               <-- após 5 segundos
(nil)
```

Referências: https://redis.io/topics/data-types-intro

Referências: https://aws.amazon.com/pt/elasticache/what-is-redis/

Situações para *não* usar chave-valor

Relacionamento entre dados: Não é apropriado quando é necessário saber o relacionamento de diferentes conjuntos de dados utilizando as chaves para correlação.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

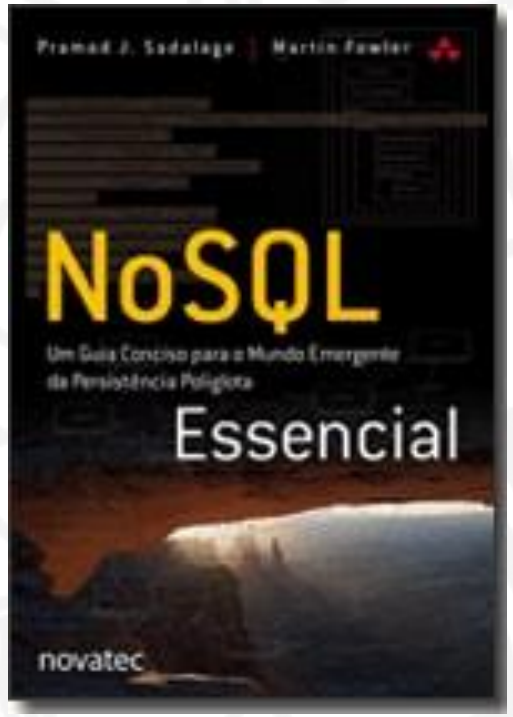
Situações para *não* usar chave-valor

Consulta por dados: Necessidade de pesquisar as chaves baseado nos seus valores (par chave-valor). Não há como inspecionar pelo banco.

Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

Principais Referências

Pramod J.; Sadalage, Martin Fowler. **NoSQL Essencial: Um Guia Conciso para o Mundo Emergente da Persistência Poliglota**. Novatec Editora, 2013.

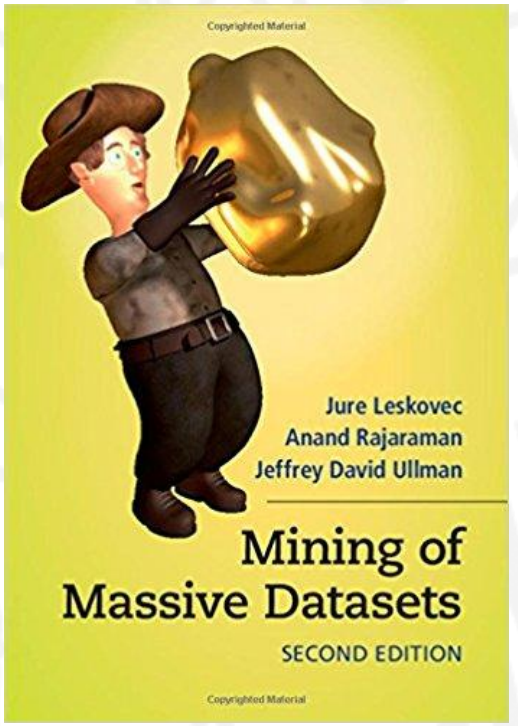


Principais Referências

Paniz, David. NoSQL: **Como armazenar os dados de uma aplicação moderna**. Casa do Código, 2017.

Referências Bibliográficas

Jure Leskovec, Anand Rajaraman, Jeff Ullman. **Mining of Massive Datasets**. 2011. 2 ed.

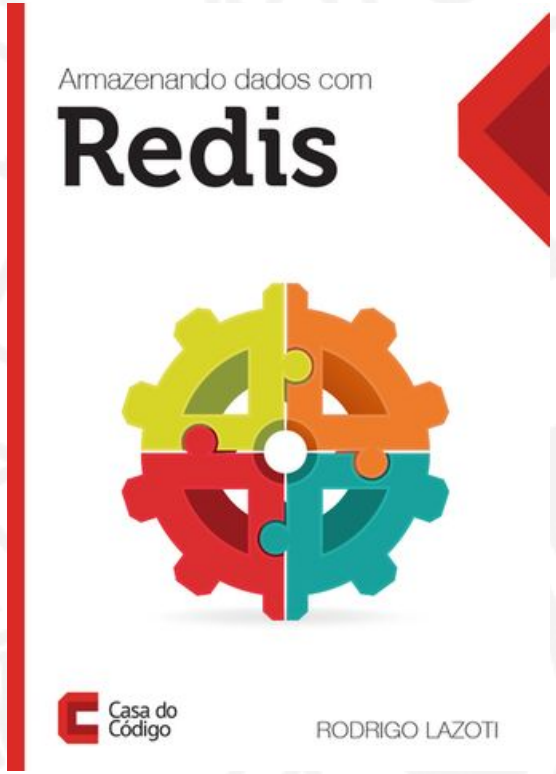


Principais Referências

Boaglio, Fernando. MongoDB: **Construa novas aplicações com novas tecnologias**. Casa do Código, 2017.

Principais Referências

Lazoti, Rodrigo. **Armazenando dados com Redis**. Casa do Código, 2017



Principais Referências

Sasaki, B.; Chao, J.; Howard, R..
Graph Databases for Beginners.
Neo4j.com.

Download gratuito em:
https://neo4j.com/whitepapers/graph-databases-beginners-ebook/?r