

Bancos de Dados BASE (NoSQL)

Ricardo Oliveira
ricardo@servidor.uepb.edu.br

Bancos de Dados Relacionais - Normalização

Normalização (3FN)

Normalização é essencial em um modelo de banco de dados relacional.

1FN = Atributos indivisíveis;

2FN = 1FN + Atributos com dependência total da chave primária (sem dependência parcial);

3fn = 2FN + ausência de dependências transitivas.

Normalização (3FN)

Objetivo:

- Acabar com a duplicação de informações.
- Não replique dados, referencie dados, aponte para dados.

Problemas da duplicação:

1. Ocupação de espaço;
2. **Consistência dos dados.**

Normalização

Perguntas:

- E se eu não normalizar?
 - Tudo bem, se houver uma boa razão pra isso.

Mais perguntas:

- SQL é a resposta pra tudo em termos de dados?
- Posso manter meus dados em arquivos, ao invés de um banco de dados?
- Meus dados precisam ser relacionais?
- Eles formam linhas e colunas ou podem ser representados por pares chave: valor?

A melhor pergunta é:

Devo usar ACID ou BASE?

ACID x BASE

A	Atomicity
C	Consistency
I	Isolation
D	Durability

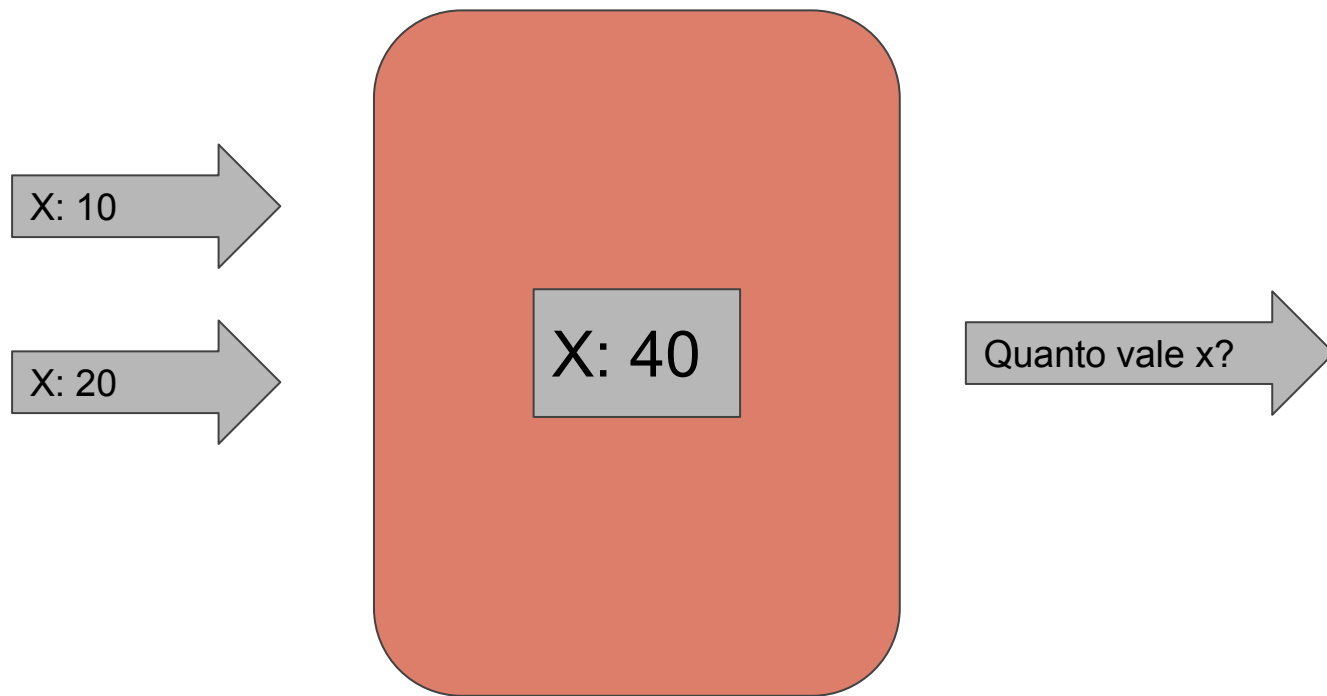
ACID x BASE

B	Basically
A	Available
S	Soft State
E	Eventual consistency

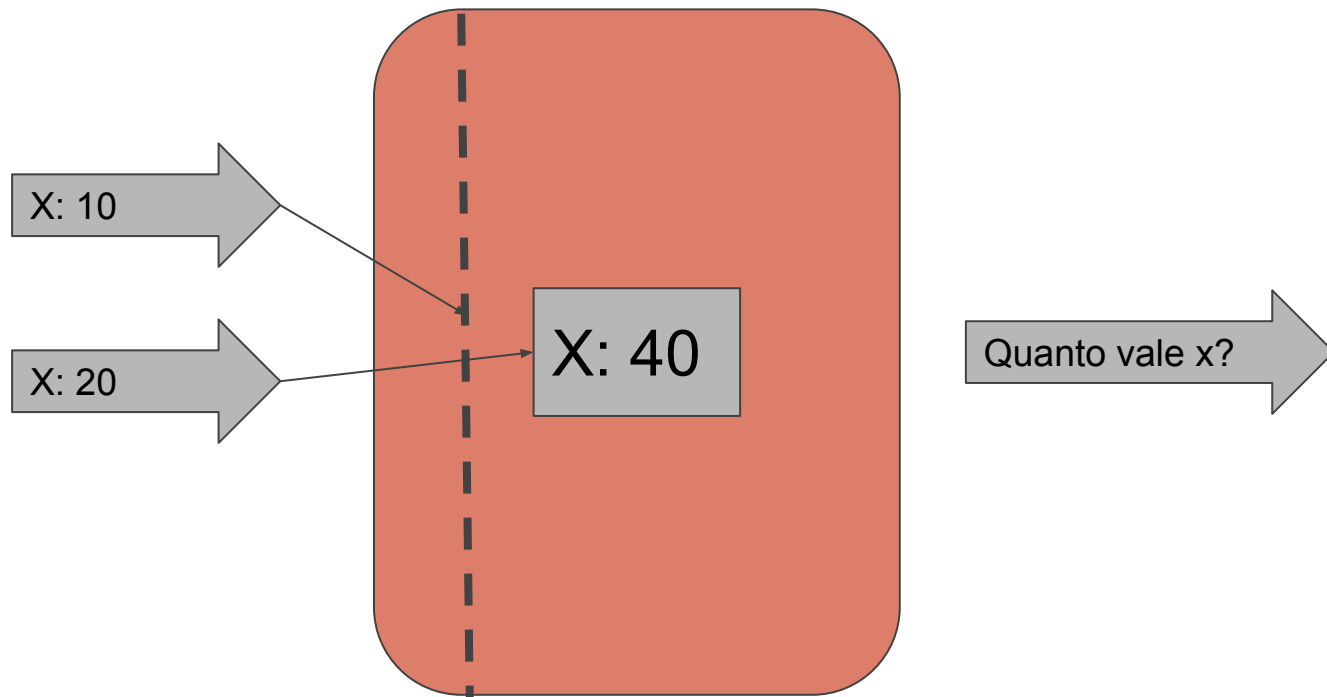
O que é consistência eventual?

- Em um dado instante de tempo, um dado corresponde a 1, para todos os usuários.
- Em algum instante de tempo futuro, alguém modifica este dado de 1 para 2.
- Dependendo de quem olha para este dado, ele pode ser 1 ou 2.
- Eventualmente, se esperarmos o suficiente, o sistema irá se atualizar por inteiro e o dado valerá 2 para todos os usuários.

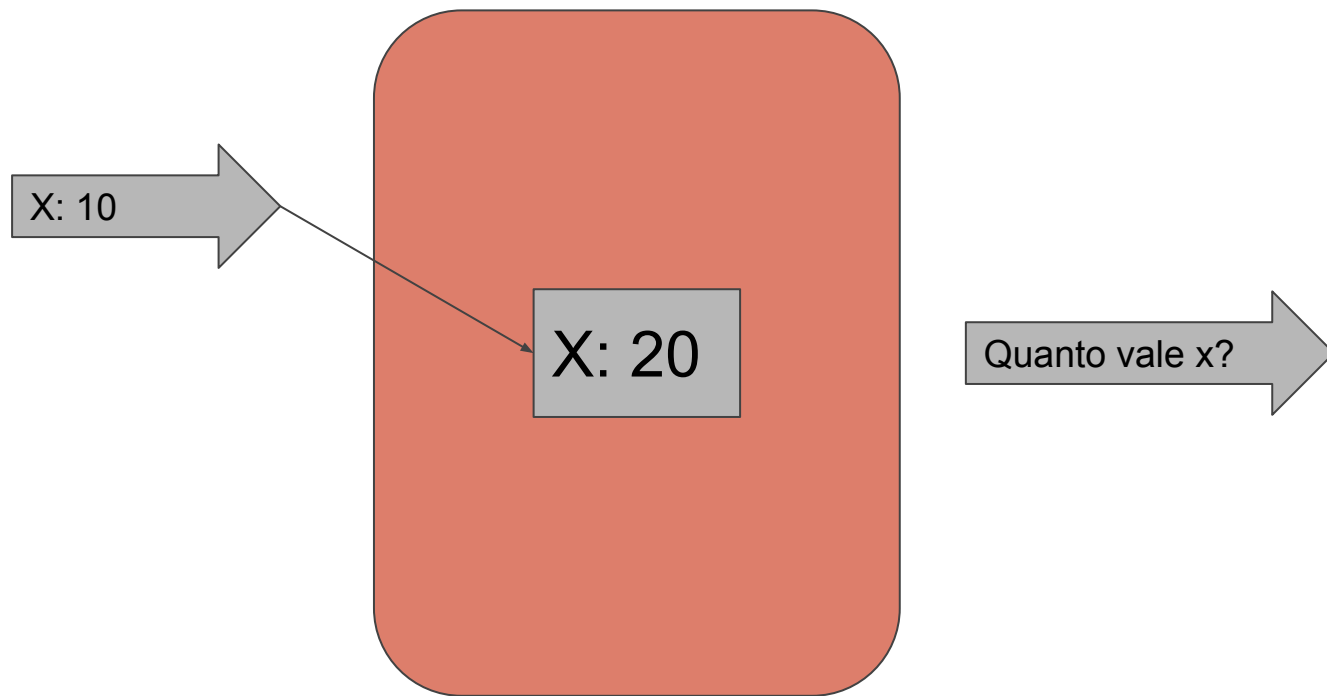
ACID



ACID



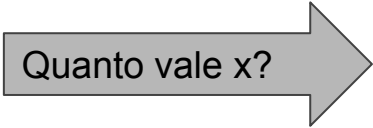
ACID



ACID



X: 10



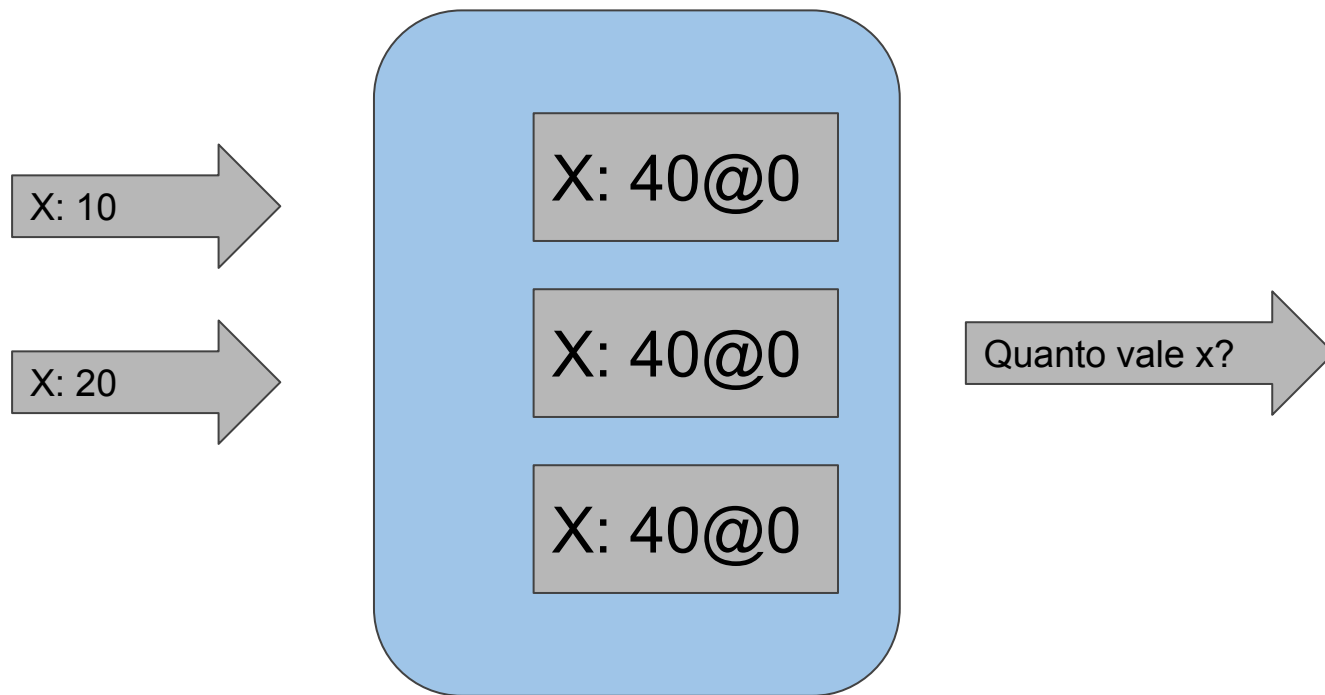
Quanto vale x?

BASE

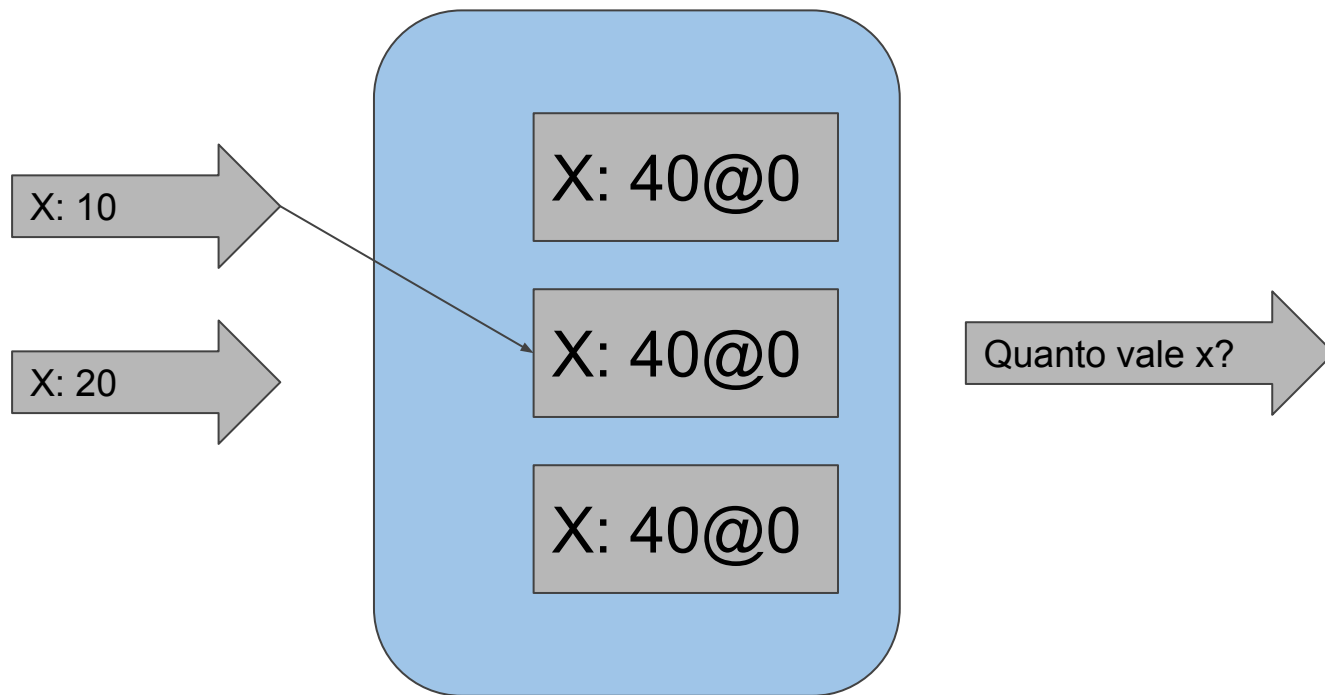
— — —

- Dados distribuídos;
- Replicação dos dados;
 - Múltiplos servidores, múltiplas cópias dos dados.

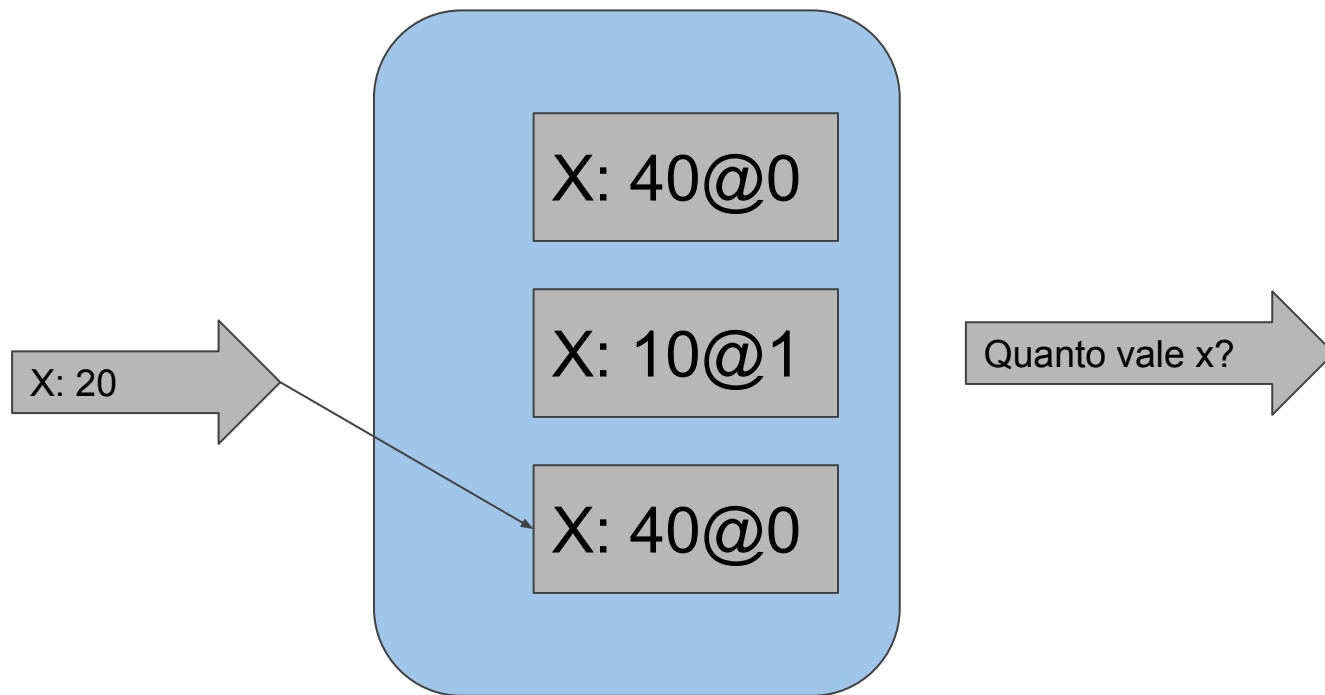
BASE



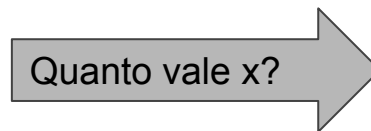
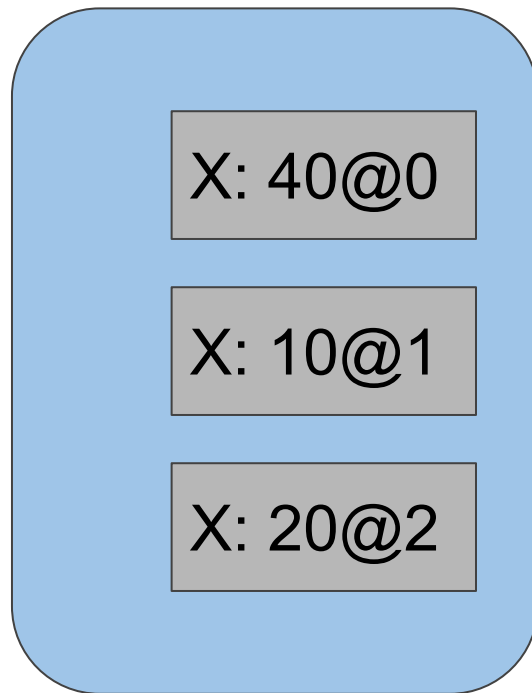
BASE



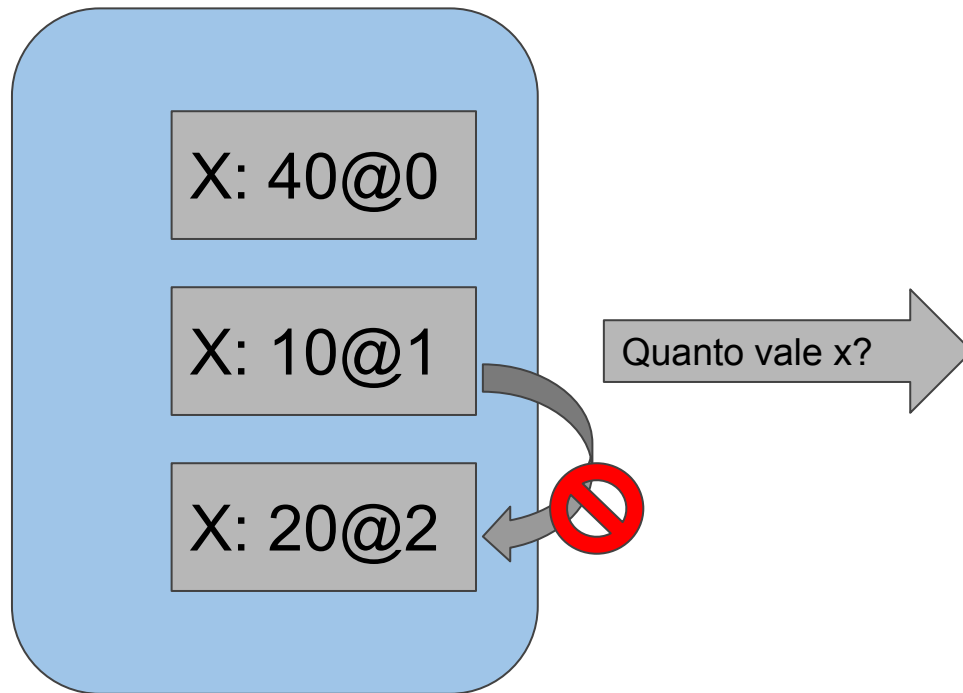
BASE



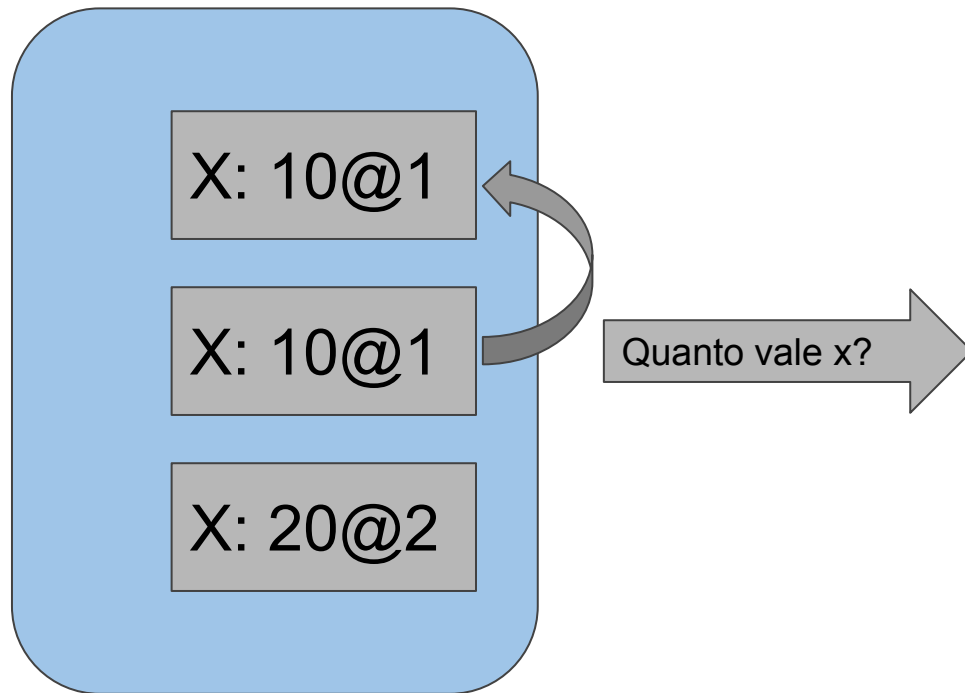
BASE



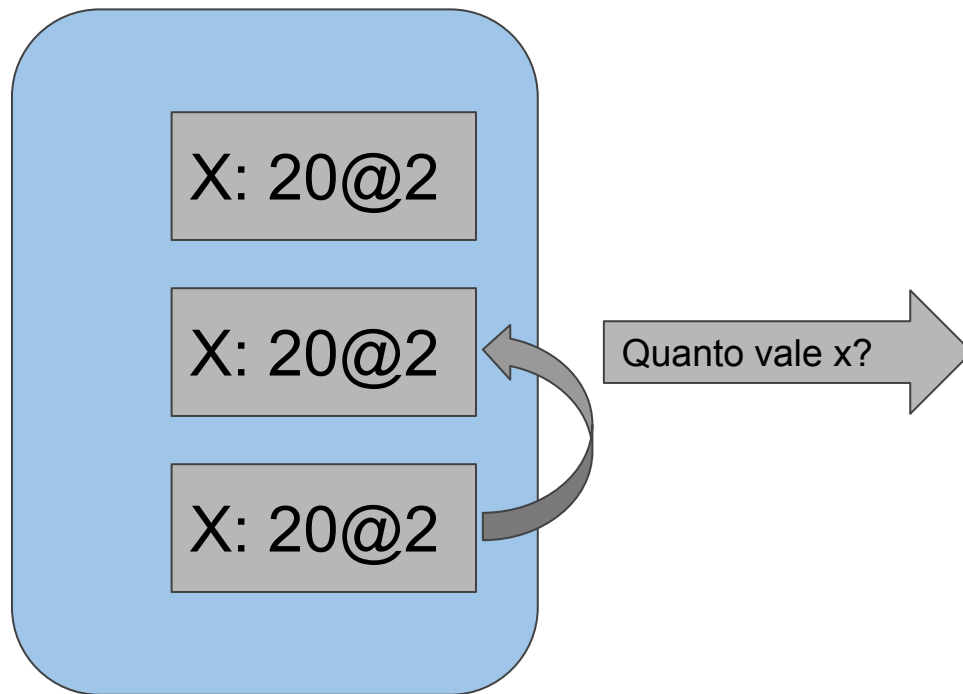
BASE



BASE



BASE



Consistência eventual é ruim?

— — —

Depende.

É, para um banco.

Não é para aplicações que, na maior parte do tempo, lêem dados.

Database Software

— — —

ACID:

- Oracle;
- PostgreSQL;
- MySQL;
- SQLite;
- SQL Server;
- etc.

BASE:

- Mongo;
- Cassandra;
- BigTable;
- ElasticSearch;
- etc.

Por que não usar ACID pra tudo?

— — —

Por que não usar ACID pra tudo?

ESCALA

Escala vertical

— — —

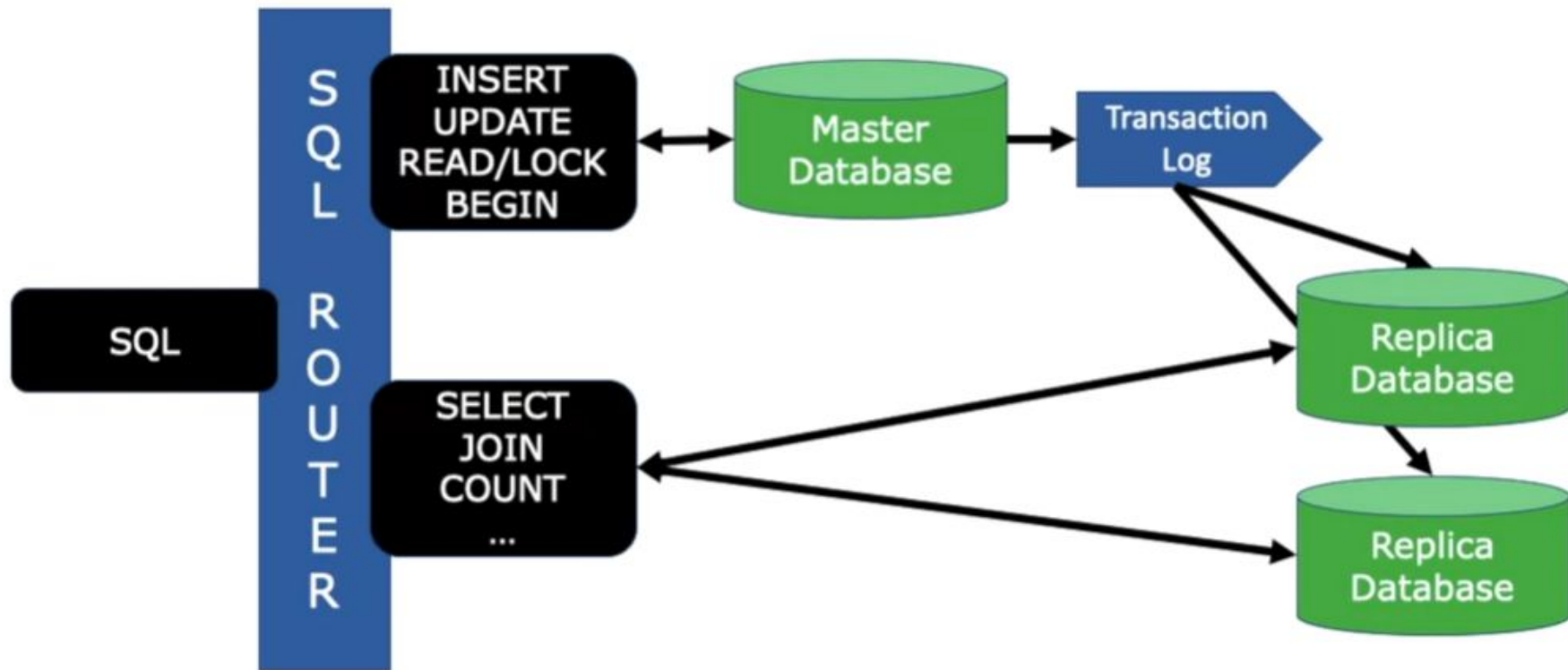
Mais hardware:

- Mais HDs / espelhamento (RAID);
- Mais processadores;
- Mais memória;
- etc.

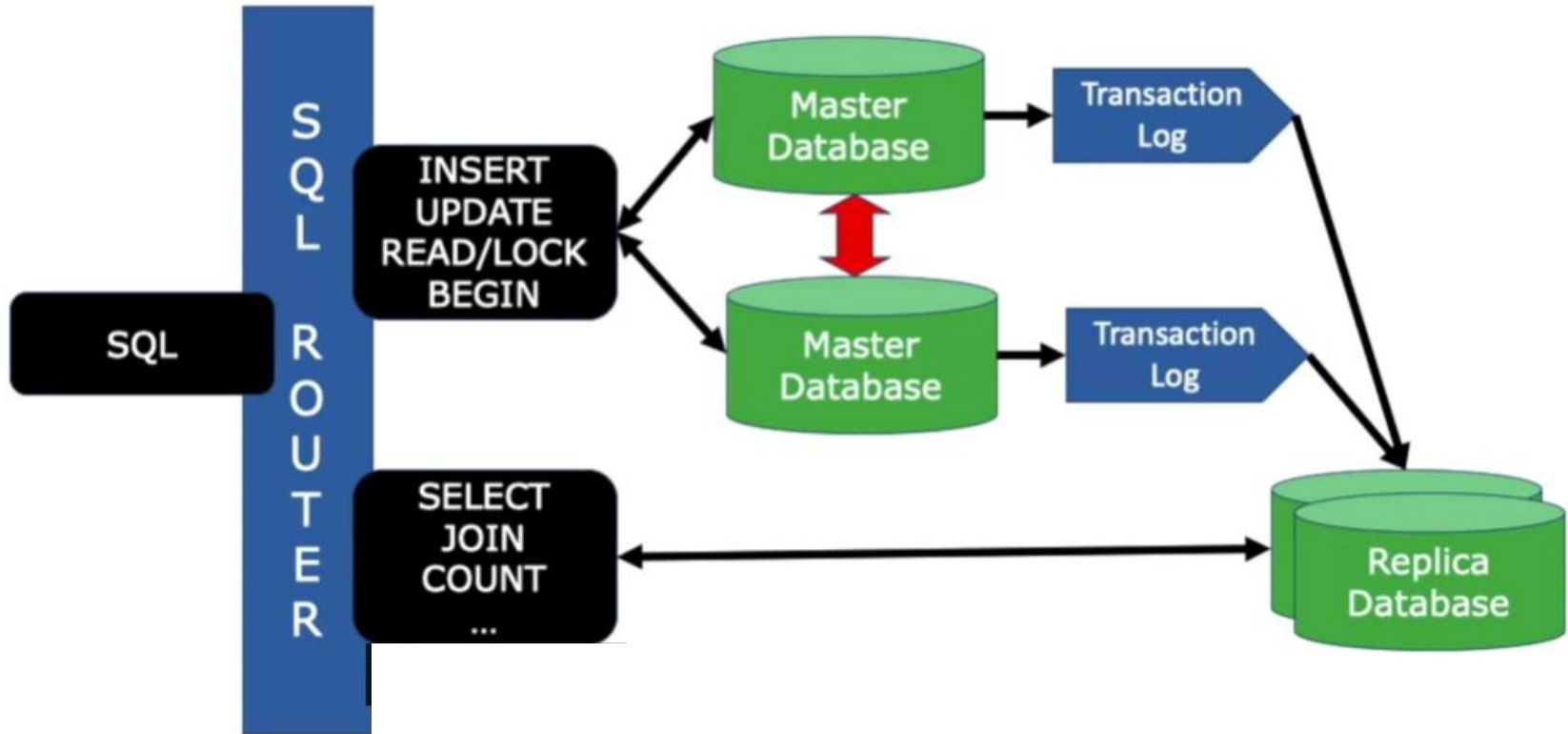
Isso foi feito com sucesso por muitos anos.

Não podia ser suficiente pra sempre.

Réplicas somente para leitura (read only)

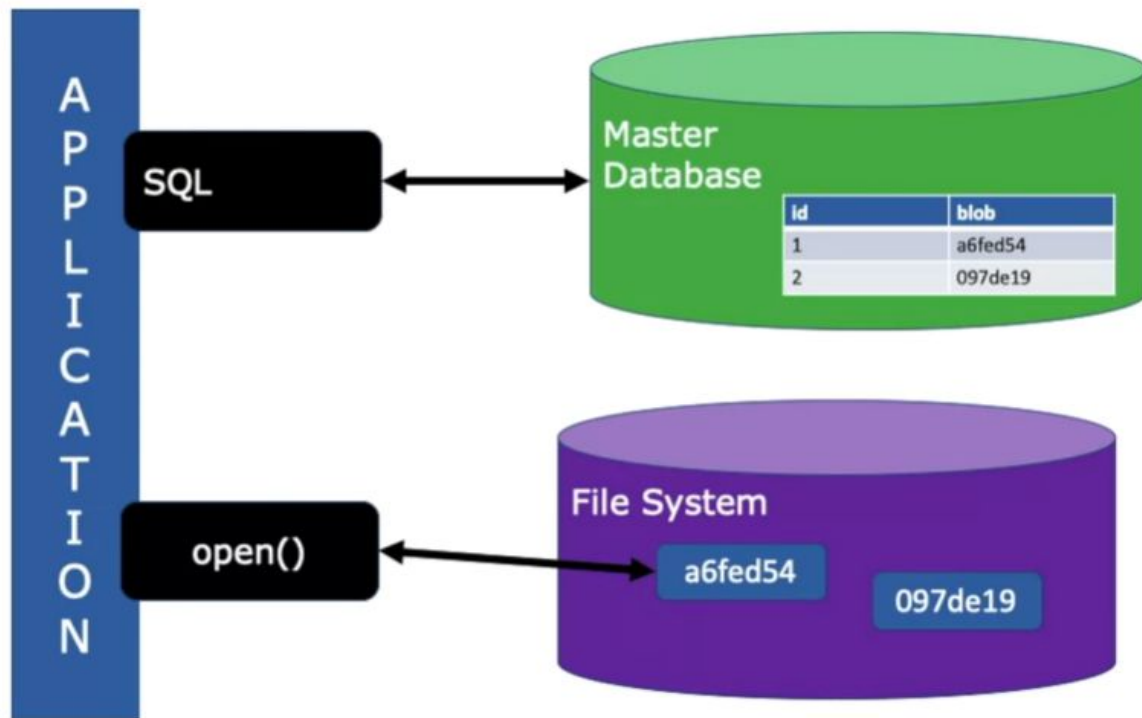


Multi-master



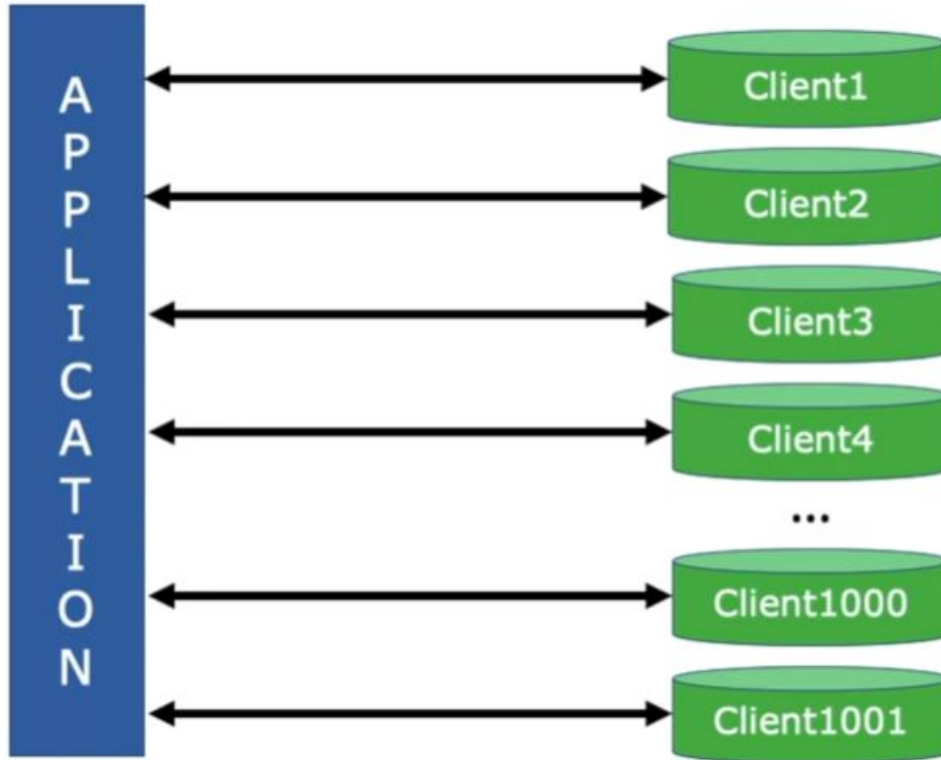
Mais de um tipo de armazenamento

— — —



Multi-tenant (estilo cloud)

— — —





Email
2002

<https://nces.ed.gov/ipeds/CollegeMap/>

Primeira geração de aplicações na nuvem

— — —

2002:

- As corporações possuíam uma máquina cara e de alto desempenho para servir como servidor de suas aplicações.
 - Email;
 - Recursos humanos;
 - Registros de transações;
 - etc.

Hardware e software escalavam verticalmente.

Tudo isso podia rodar em servidores da Amazon. Era nuvem?

Como o cenário mudou?



Search the web using Google!

Special Searches
[Stanford Search](#)
[Linux Search](#)

[Help!](#)
[About Google!](#)
[Company Info](#)
[Google! Logos](#)

Get Google!
updates monthly:

 [Archive](#)

Copyright ©1998 Google Inc.

Gmail

Cada usuário do email acessa unicamente sua porção dos dados.

Nem tudo precisa ser replicado.

- Shards (fragmentos)

Ao invés de um computador de U\$ 1.000.000,00, vários computadores de U\$ 1.000,00.

- Hardware virtualizável.

Search Query
"beautiful code"

Google

Search Results



Load
Balancer

Mixer

Google
Web Server

Ads +
WebSearch

Mixer

Google
Web Server

Load
Balancer



Search Query
"beautiful code"



Load
Balancer

Mixer

Google
Web Server

Ads +
WebSearch

Mixer

Google
Web Server

Load
Balancer



Scatter and gather

Search Results



Solução Google

Problema resolvido em 1/4 de segundo.

- Sem ocupar todo o recurso;
- Várias requisições sendo resolvidas ao mesmo tempo.

A escala é feita aumentando-se o número de servidores;

- De 200 para 300, depois 600 servidores.

Replicação de dados:

- 7 a 10 cópias das mensagens no Gmail;
- Não há necessidade de backup.

Solução Amazon

Cenário:

- CPUs baratas;
- Espaço em disco barato;
- Memória cara;
- Velocidade de disco cara.

Servidores de U\$ 40.000,00 atualizados a cada 2 anos.

Solução Amazon

Amazon Web Services - AWS

- Aluguel de poder computacional:
 - U\$ 10,00 por mês por uma máquina;

Necessidade de rearquitecturar aplicações.

- Usar menos memória;
- Muita disponibilidade de espaço, mas sem velocidade de acesso.

Carpet cluster



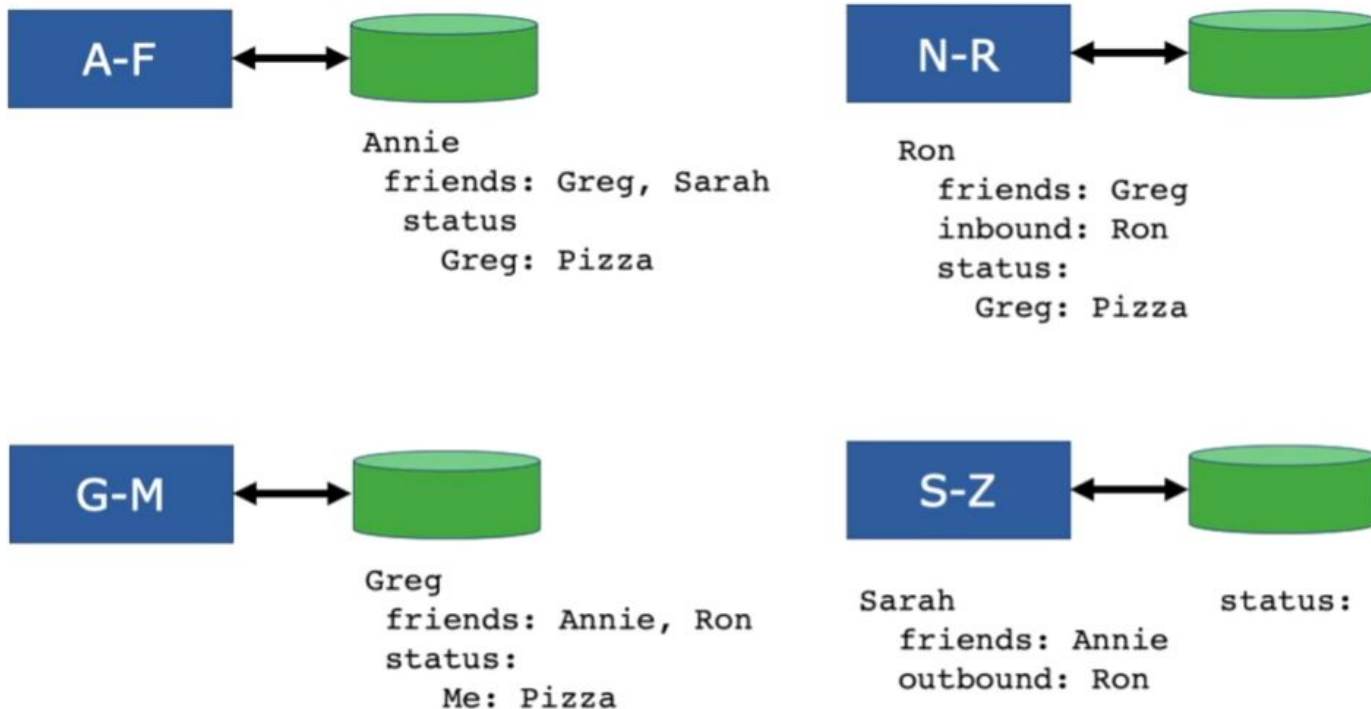
Carpet clusters

É preciso construir aplicações que rodem numa estrutura como essa.

SHARDS (fragmentos)

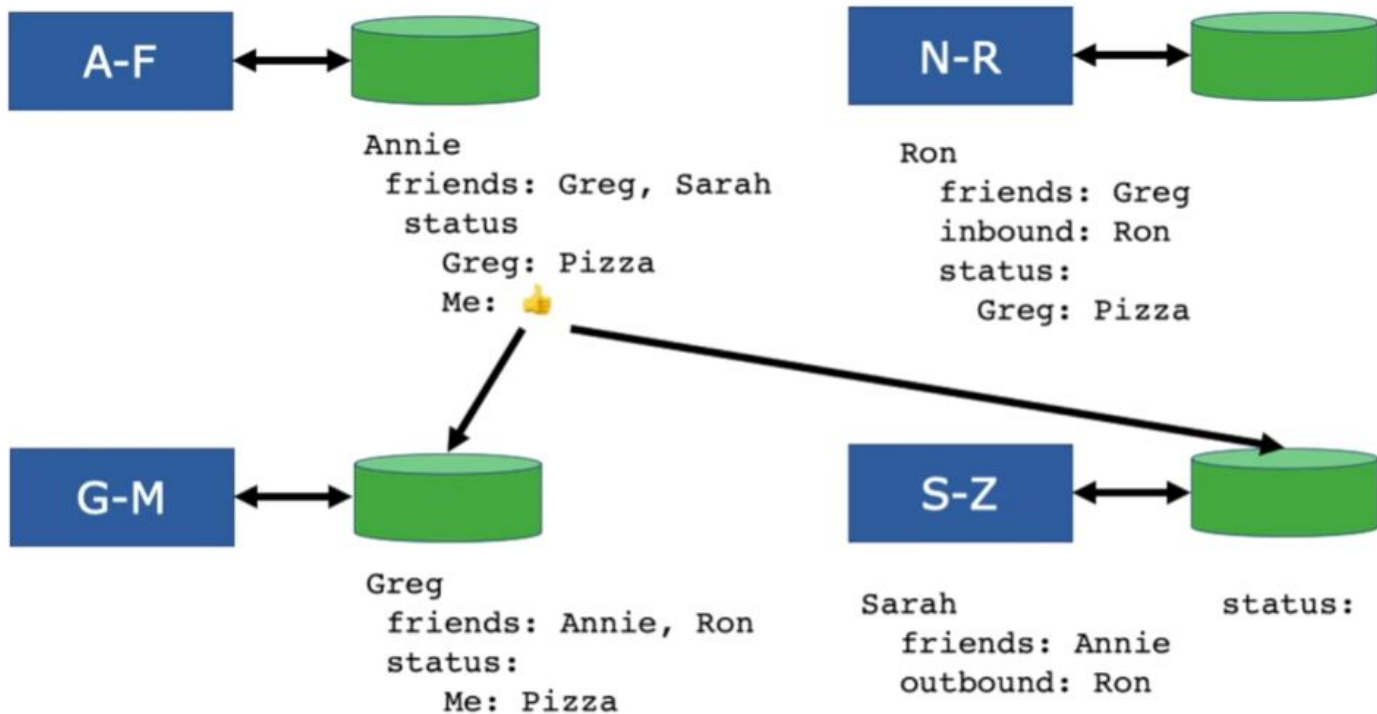
Segunda geração de aplicações na nuvem

Facebook



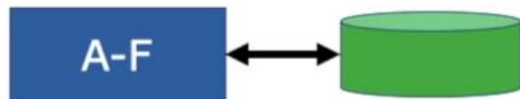
Segunda geração de aplicações na nuvem

Facebook



Segunda geração de aplicações na nuvem

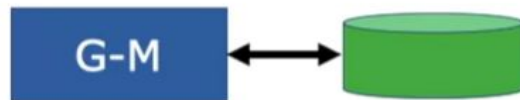
Facebook



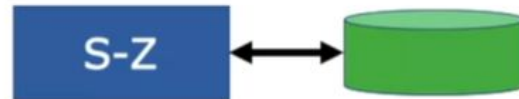
Annie
friends: Greg, Sarah
status
Greg: Pizza
Me: 🍕



Ron
friends: Greg
inbound: Ron
status:
Greg: Pizza
Annie: 🍕 (??)



Greg
friends: Annie, Ron
status:
Me: Pizza
Anne: 🍕



Sarah	status:
friends: Annie	Greg: Pizza (??)
outbound: Ron	Annie: 🍕 (??)

Dados no facebook

Seus amigos estão, por exemplo, em 10 servidores.

- Ler 10 servidores pra construir sua timeline é custoso.
 - Essa informação deve estar disponível antes de você logar.

E se a solução do Facebook for generalizada para aplicações em geral?

- Bancos de dados com consistência eventual (BASE) surgem.

Surgimento do NoSQL

Bancos relacionais:

- Esquema pré-concebido;
- Evoluir o esquema requer cuidado.

NoSQL:

- Dados formam documentos;
- Pares {Chave: Valor};
- Esquema tardio;
- JSON.

Databases NoSQL open source

— — —

- CouchDB(2008)
- MongoDB(2009)
- Cassandra(2008)
 - Facebook
- Elasticsearch(2010)

NoSQL era a solução para tudo e ia enterrar os SGBDs relacionais.

E os SGBDs relacionais?

2013-2014: Os desenvolvedores de SGBDs precisavam reagir.

Também nessa época: Usuários de NoSQL começaram a reclamar.

- Todo mundo precisa fazer um JOIN de vez em quando.

Mudanças tecnológicas 2009 - 2019

— — —

- AWS passou a vender sistemas com 32 CPUs e enormes quantidades de memória.
- SSDs substituindo HDs.
 - Scatter / gather

Escala vertical passou a ser possível.

Reação ao NoSQL

— — —

Oracle, MySQL, PostgreSQL

- Colunas JSON;
- Funcionalidades NoSQL.

Então, SQL não vai morrer?

Dá pra usar SQL e BASE juntos!

- begin transaction, select for update, isso é ACID.
- select, insert, update não são ACID ou BASE.

Tudo se juntou.

SQL com semântica ACID, SQL com semântica BASE.

Como ser BASE num SGBD ACID?

— — —

- Não normalizar, replicar;
- Não usar SERIAL, usar UUID;
- Menos colunas
 - Uma coluna JSON com muitos dados (pares chave: valor);
 - Colunas apenas para indexar;
- Não usar chaves estrangeiras (ou não marcá-las como tal);
 - Sem ON CASCADE;
- etc.

Referência

— — —

Database Architecture, Scale, and NoSQL with Elasticsearch

Universidade de Michigan

Coursera



Ministrado por: **Charles Russell Severance,**

Clinical Professor

School of Information