# MAC5855 - NoSQL

Mauricio De Diana
mdediana@ime.usp.br
@mdediana

# Web 2.0

*Data is the Next Intel Inside*

Inteligência Coletiva

Grande volume de dados

Escala global (*Internet scale services*)

# Web 2.0

Alto grau de paralelização

Hardware comum e barato (*commodity*)

Software livre, código aberto

Dados semiestruturados e crús

# Soluções para dados

Bigtable (Google)

Dynamo (Amazon)

PNUTS (Yahoo!)

Além de:
GFS, Hadoop, HDFS, MapReduce, Chubby, etc

# Soluções para dados

Bigtable (Google)

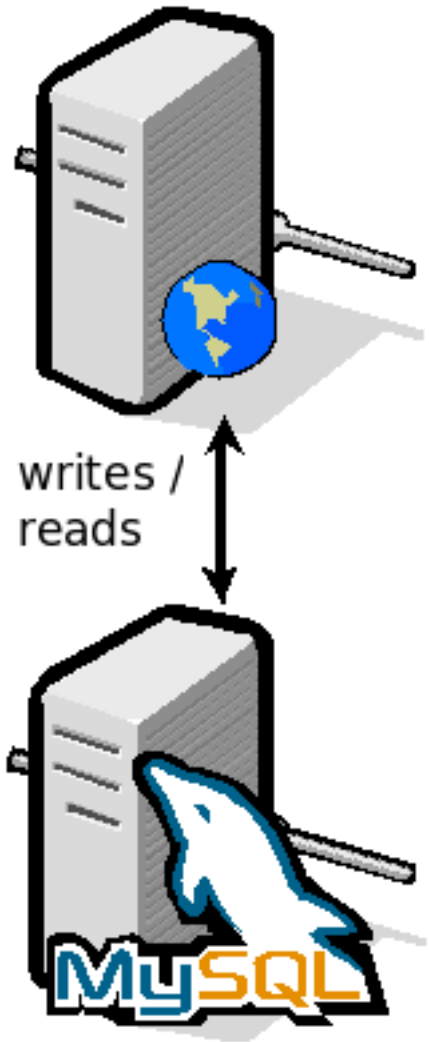
Dynamo (Amazon)

PNUTS (Yahoo!)

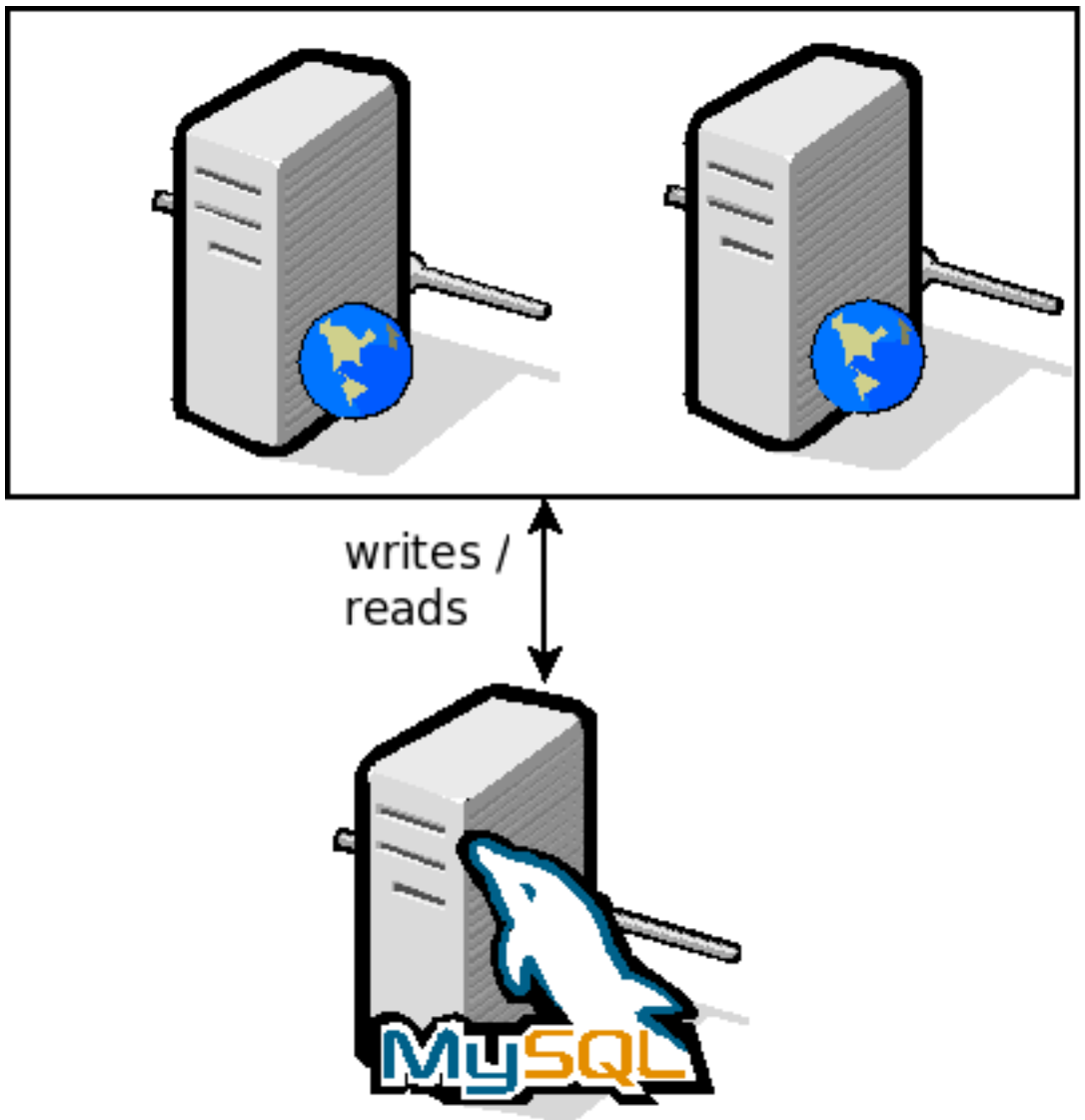
Além de:
GFS, Hadoop, HDFS, MapReduce, Chubby, etc

Enquanto isso, nas startups...

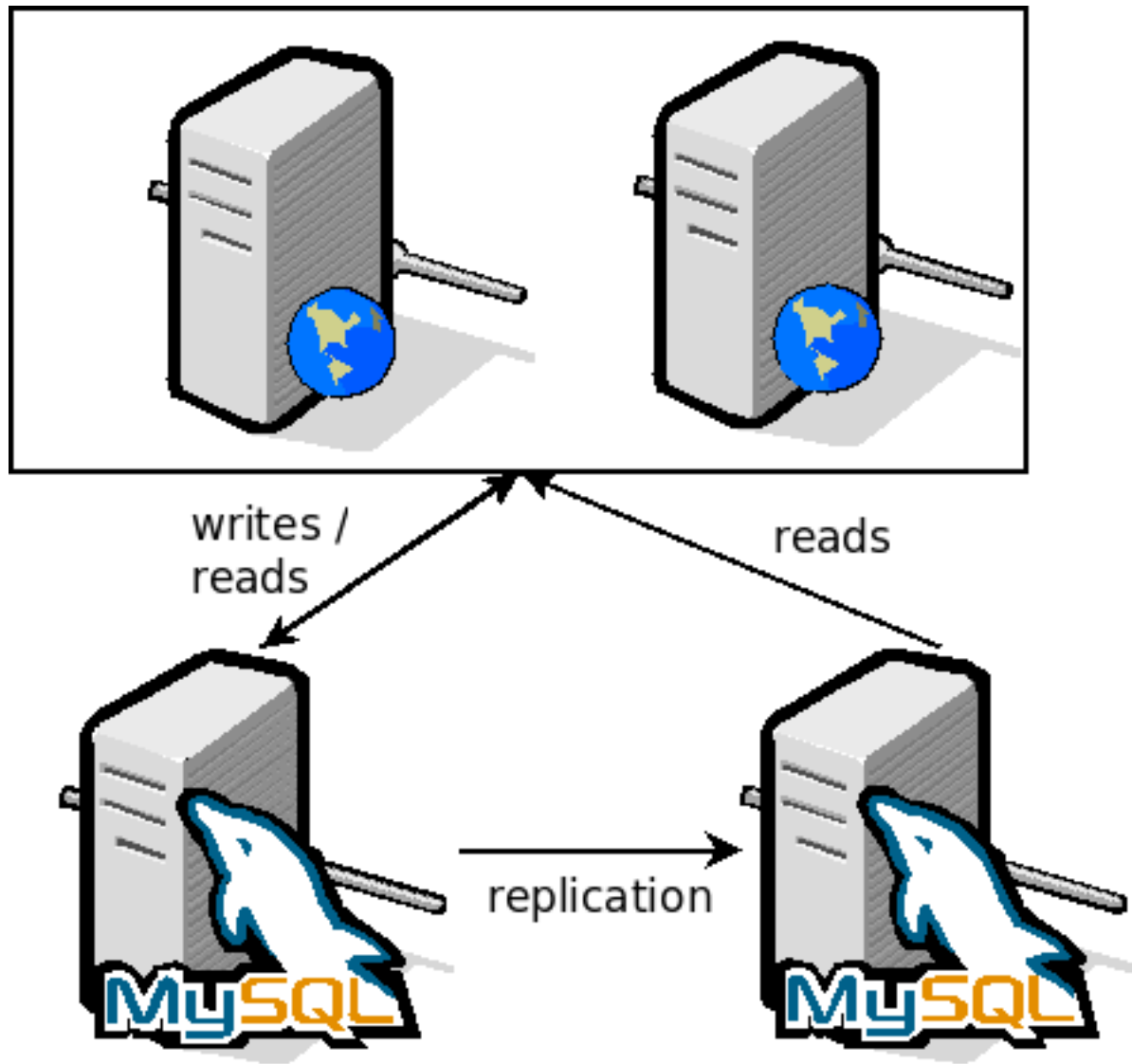
# Escalando relacional



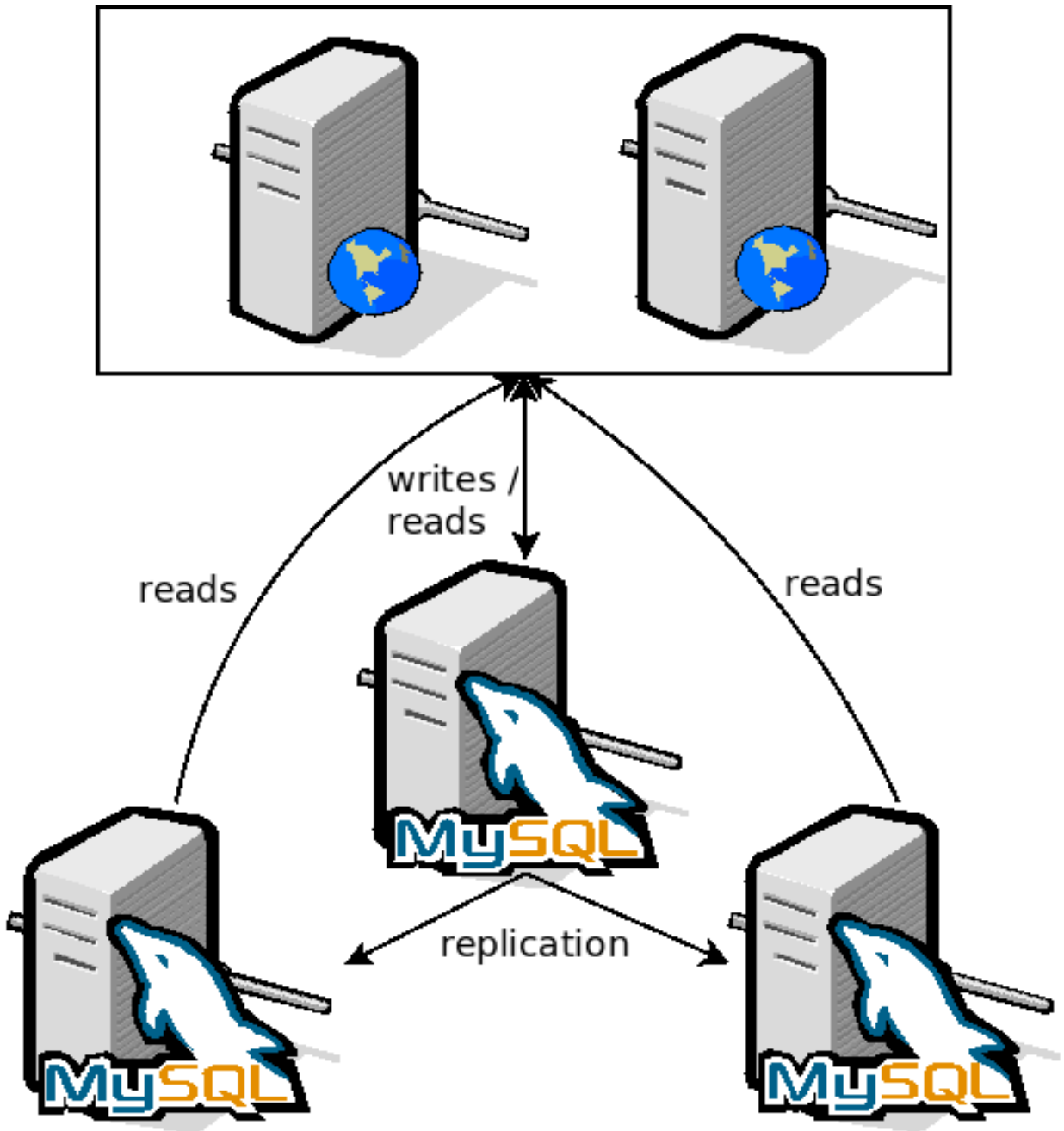
# N servidores web
# 1 db



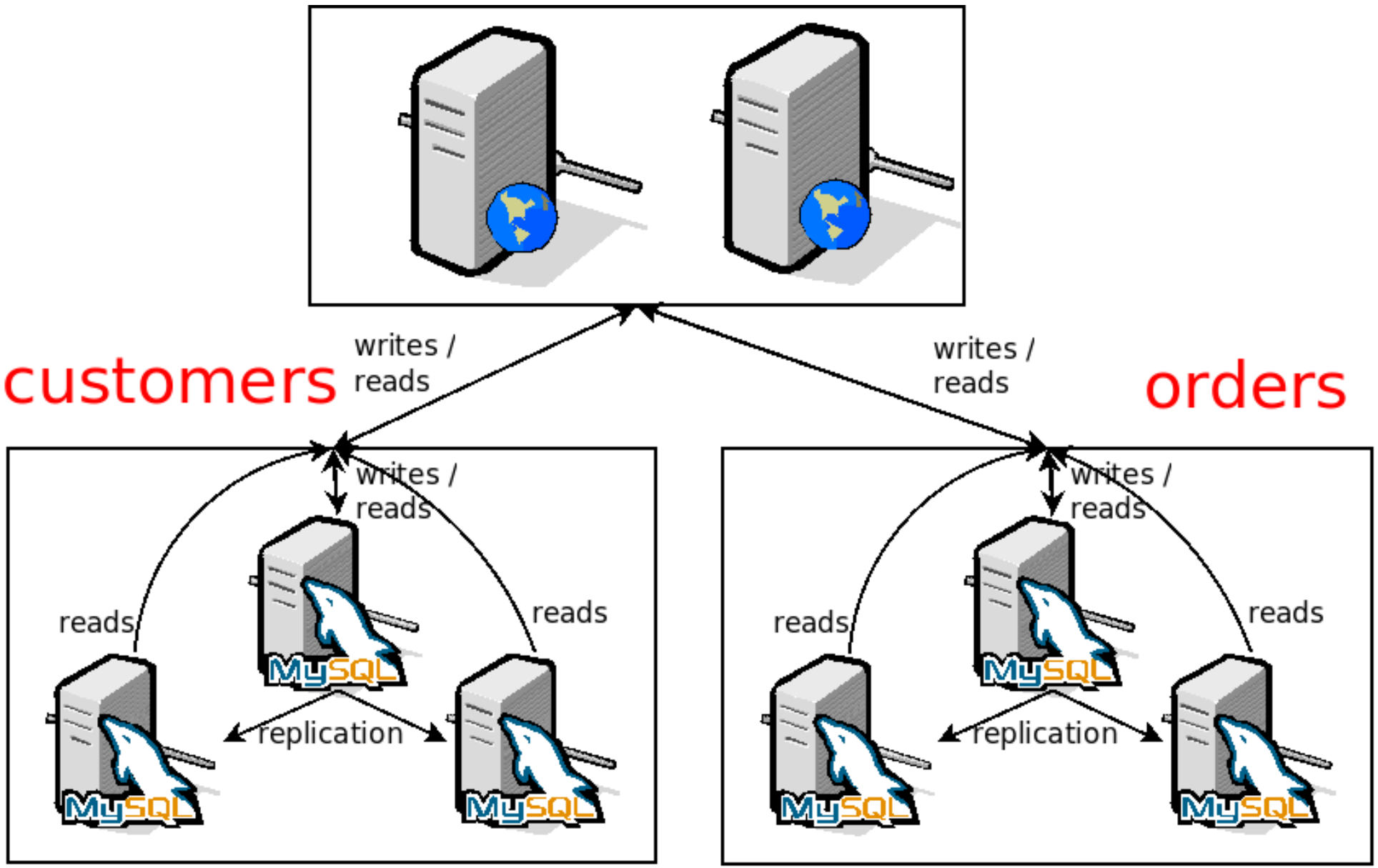
# Mestre / Escravo



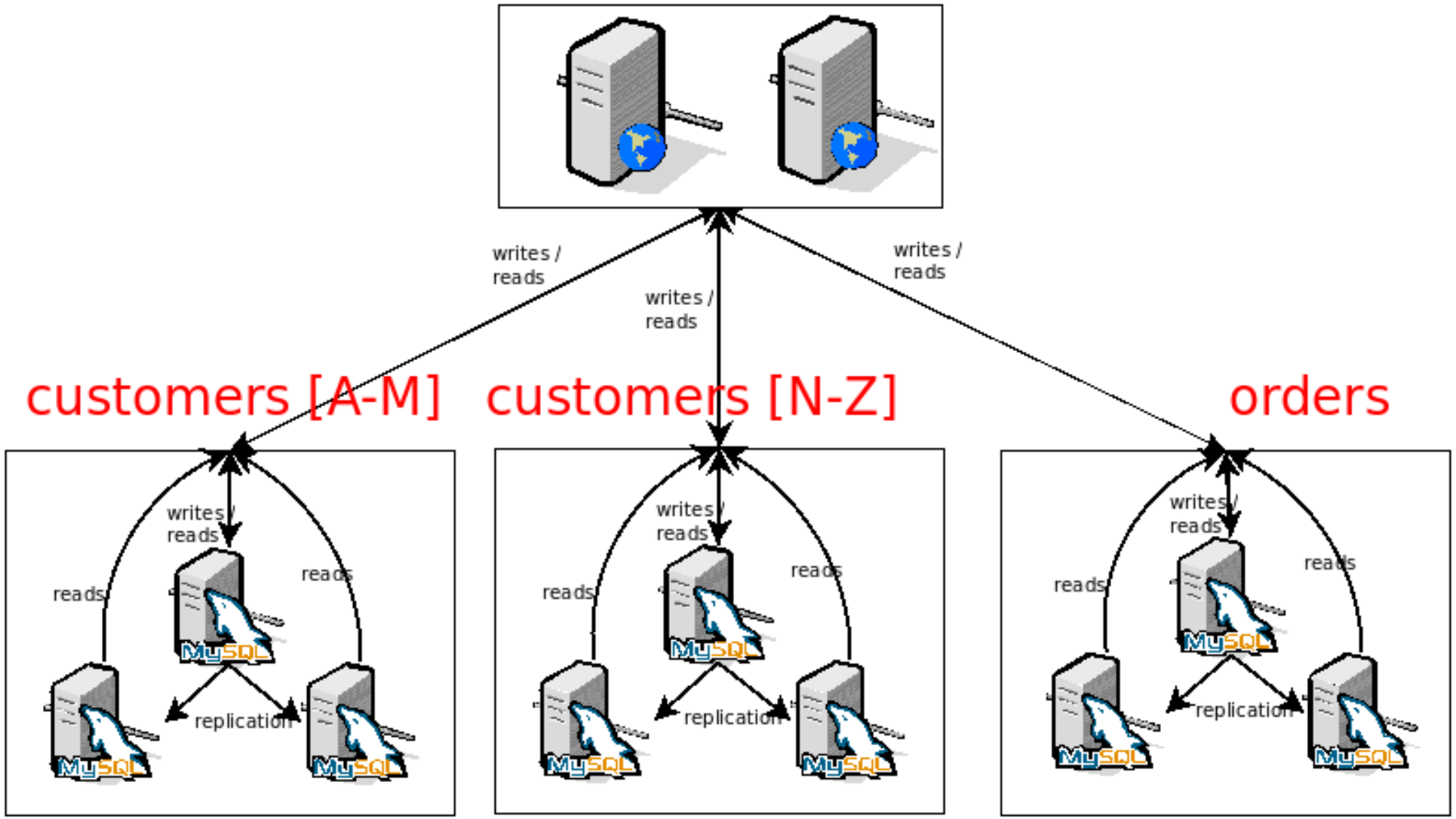
# Mestre / Escravo



# Particionamento funcional

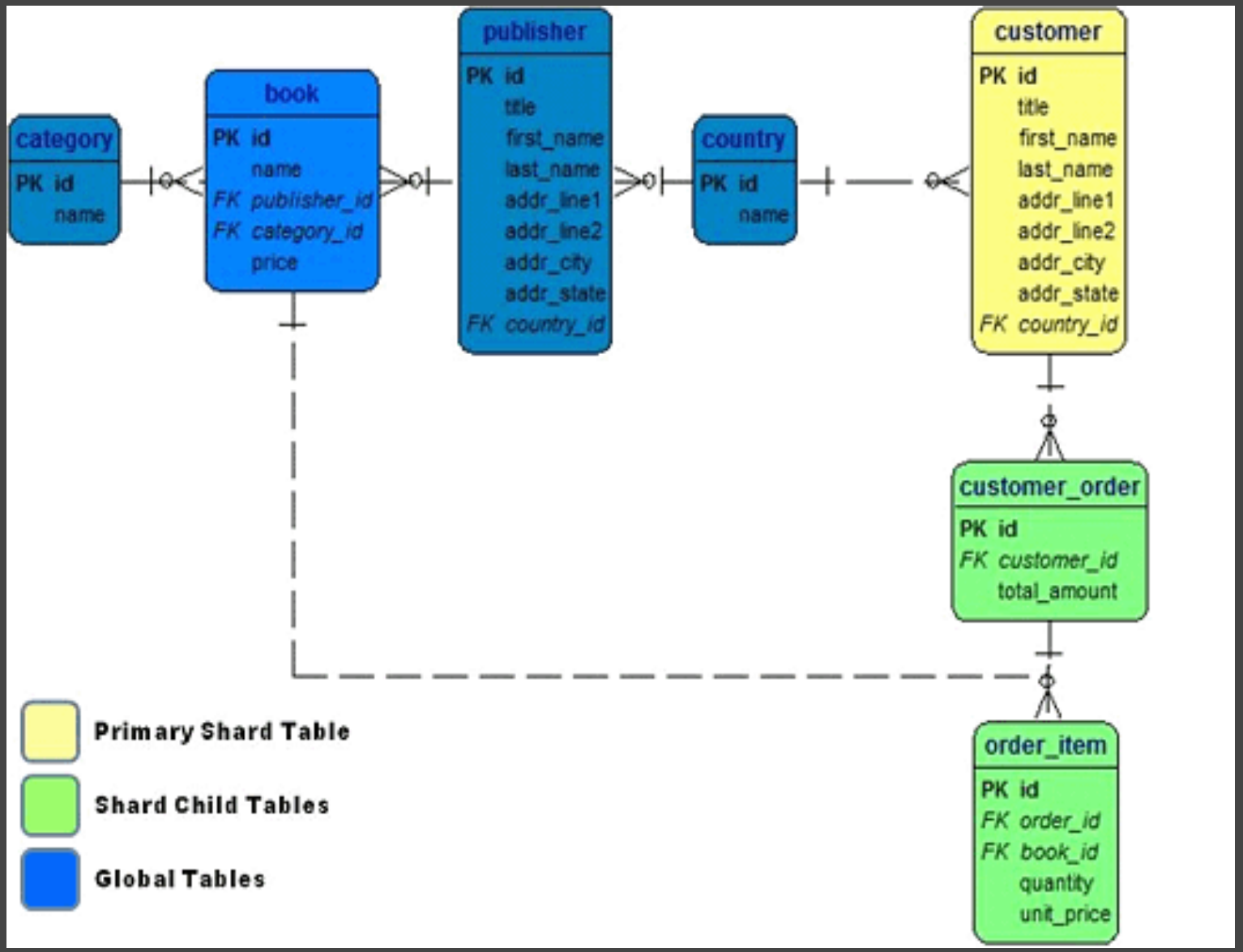


# Particionamento horizontal (*sharding*)




Pritchett, D. (2008). BASE: An Acid Alternative.

# Particionamento horizontal (*sharding*)




Fonte: http://www.codefutures.com/database-sharding/

# Problemas - *Sharding*

Queries distribuídas

*Cross-shard joins*

Integridade de entidade e referencial

Transparência de localização
(Opção: MySQL Proxy)

# Dados em larga escala na web - FLOSS

Hadoop

Cassandra

MongoDB

Neo4J

Redis

Riak

...

# NoSQL

Definição fraca:

SGBDs não-relacionais distribuídos muito usados na Web

# NoSQL

Definição fraca:
SGBDs não-relacionais distribuídos muito usados na Web

Definição ainda fraca:
SGBDs não-relacionais, não-ACID e distribuídos

# NoSQL

Definição fraca:
SGBDs não-relacionais distribuídos muito usados na Web

Definição ainda fraca:
SGBDs não-relacionais, não-ACID e distribuídos

"No SQL" x "Not only SQL"

# NoSQL - características

Escalabilidade horizontal

Sem esquema / Esquema flexível

Replicação simples

API simples

Software livre / Código aberto

Consistência em momento indeterminado

# ACID e escalabilidade

Atomicidade: protocolo distribuído (2PC)

Consistência: problemas com réplicas

Isolamento: *locks* distribuídos

# Exemplos de trocas com ACID em NoSQL

Atomicidade e isolamento: em único banco de dados (Bigtable)

Consistência: em momento indeterminado (Dynamo)

Durabilidade: memória + *snapshotting* (Redis)

# ACID x BASE

**ACID**

Atômico

Consistente

Isolado

Durável

**BASE**

Basicamente disponível *(Basically available)*

*Soft-state*

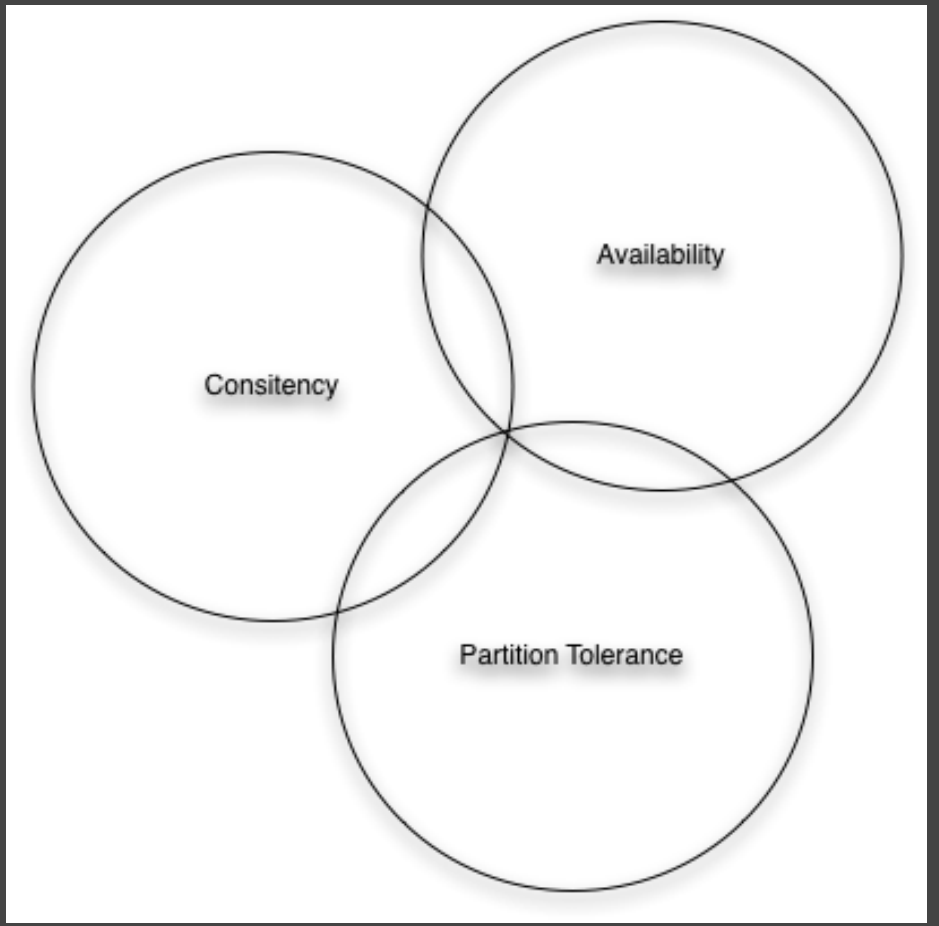
Consistente em momento indeterminado (*Eventually consistent*)

# Teorema CAP

Consistência

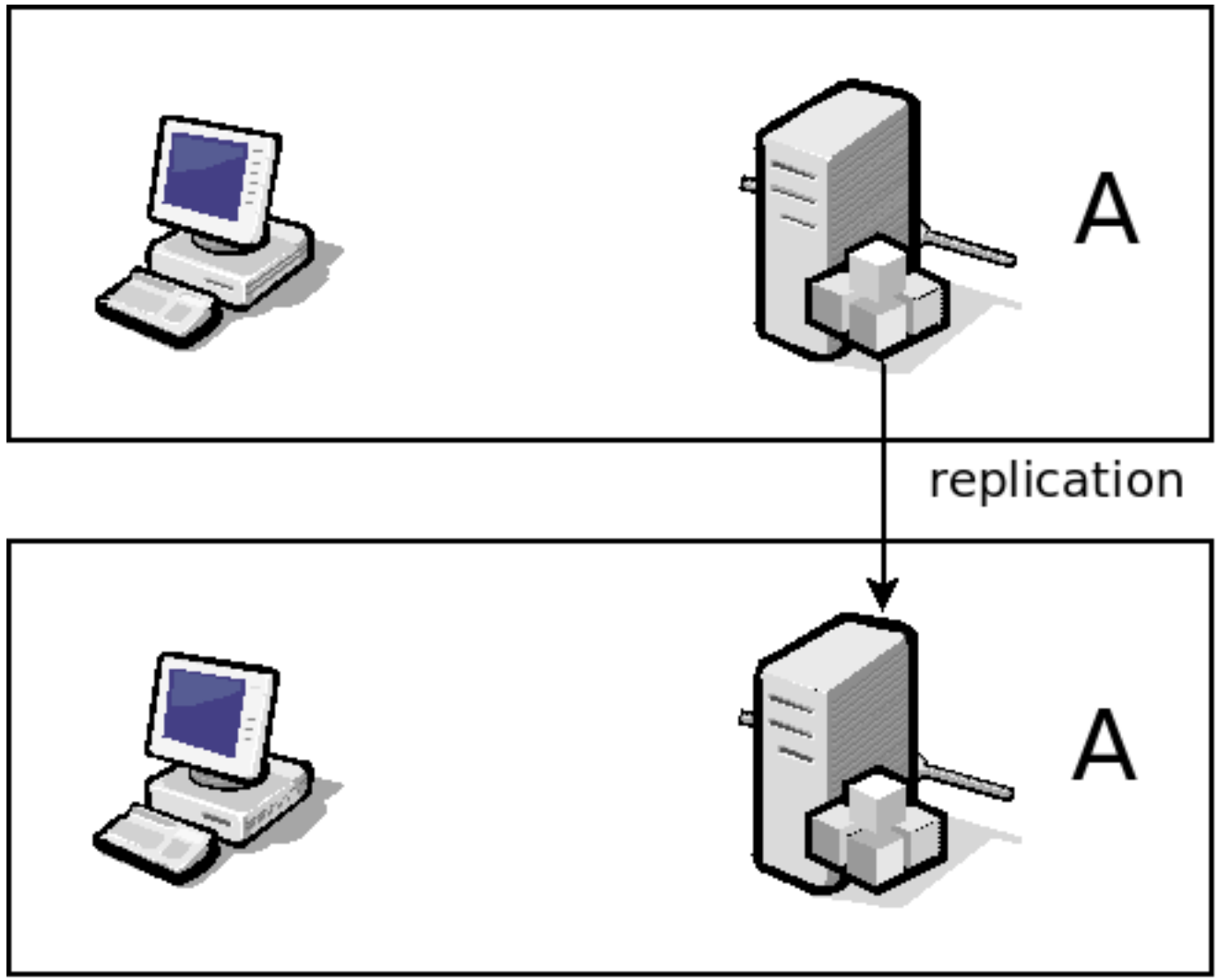
Disponibilidade

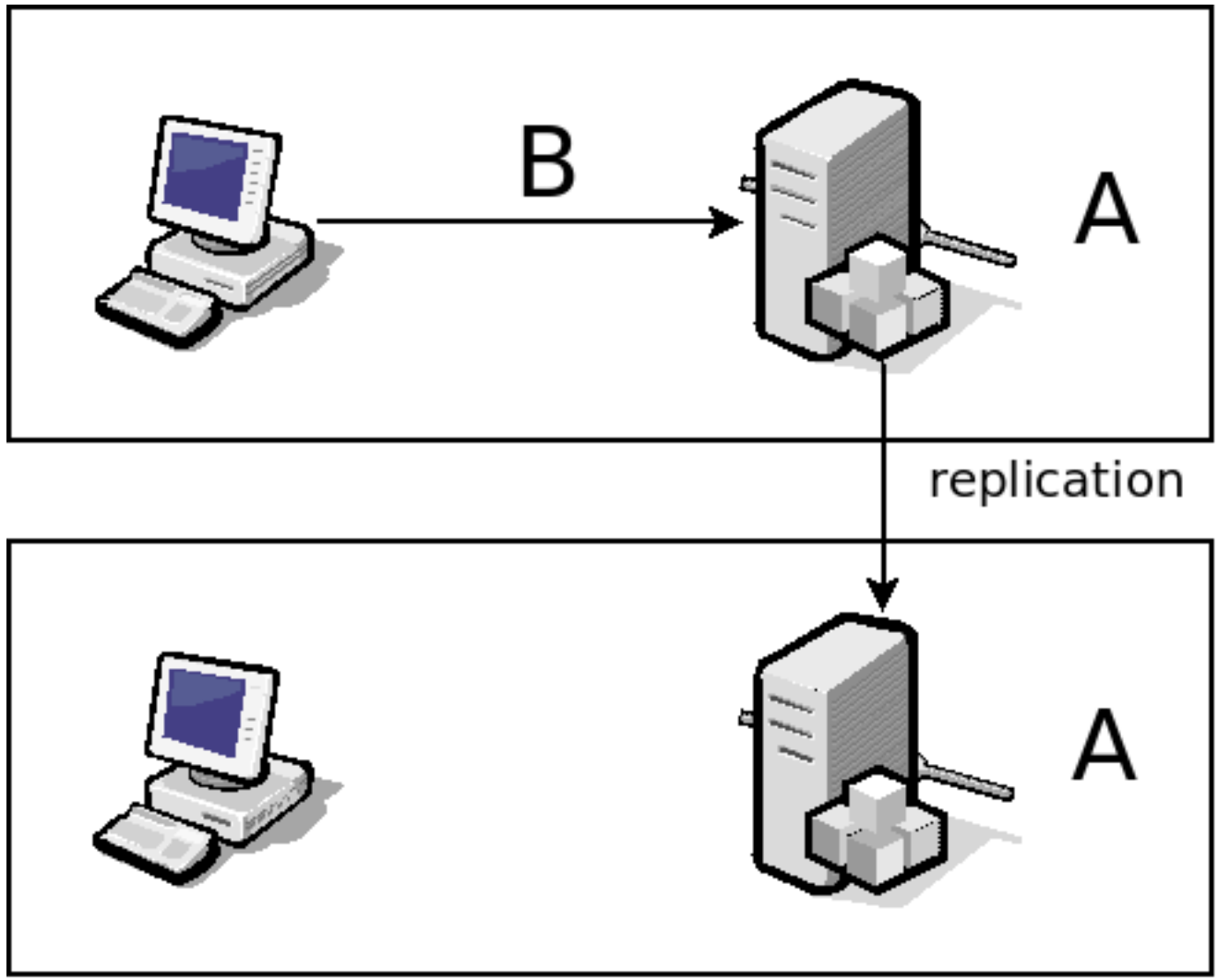
Tolerância à partição

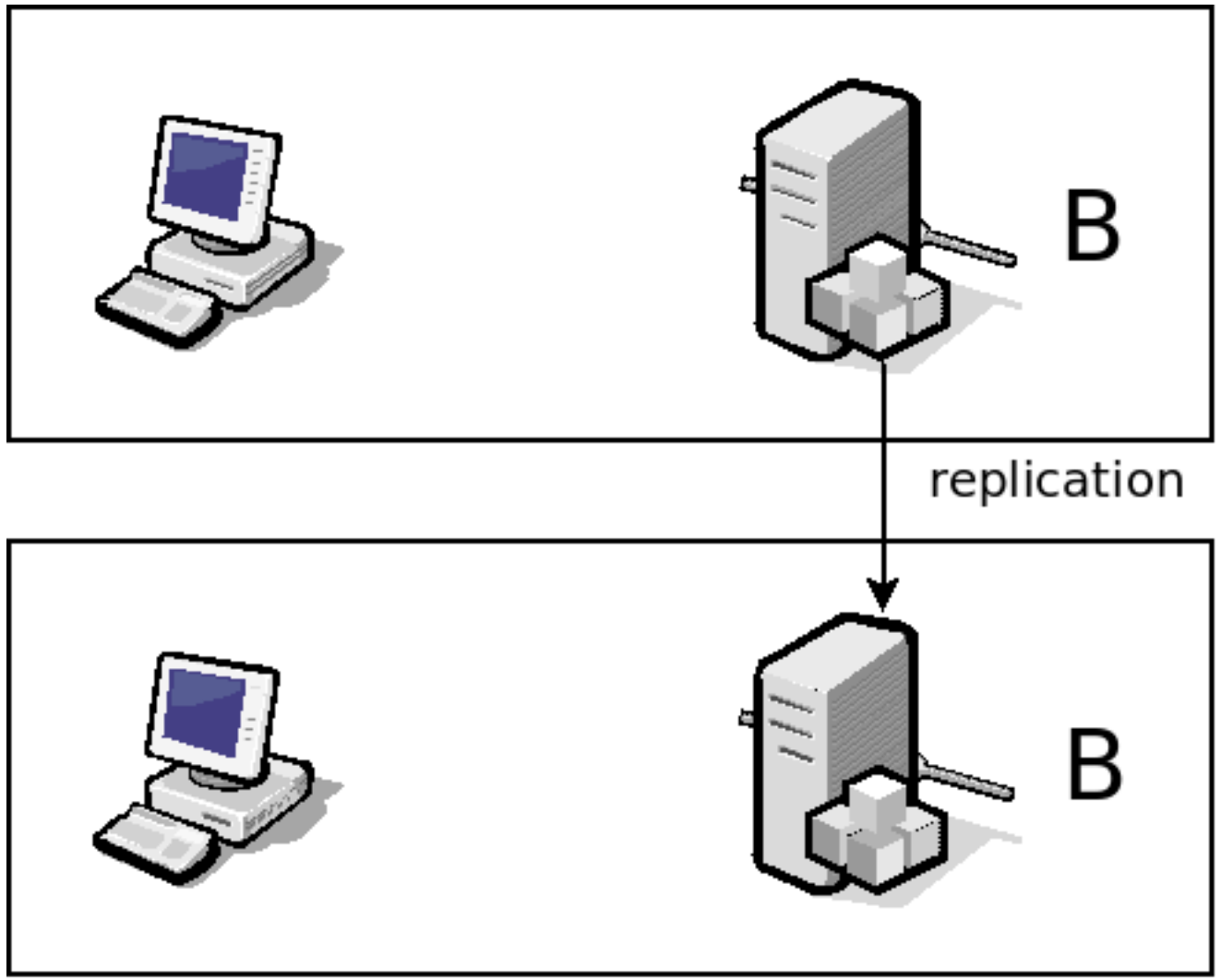


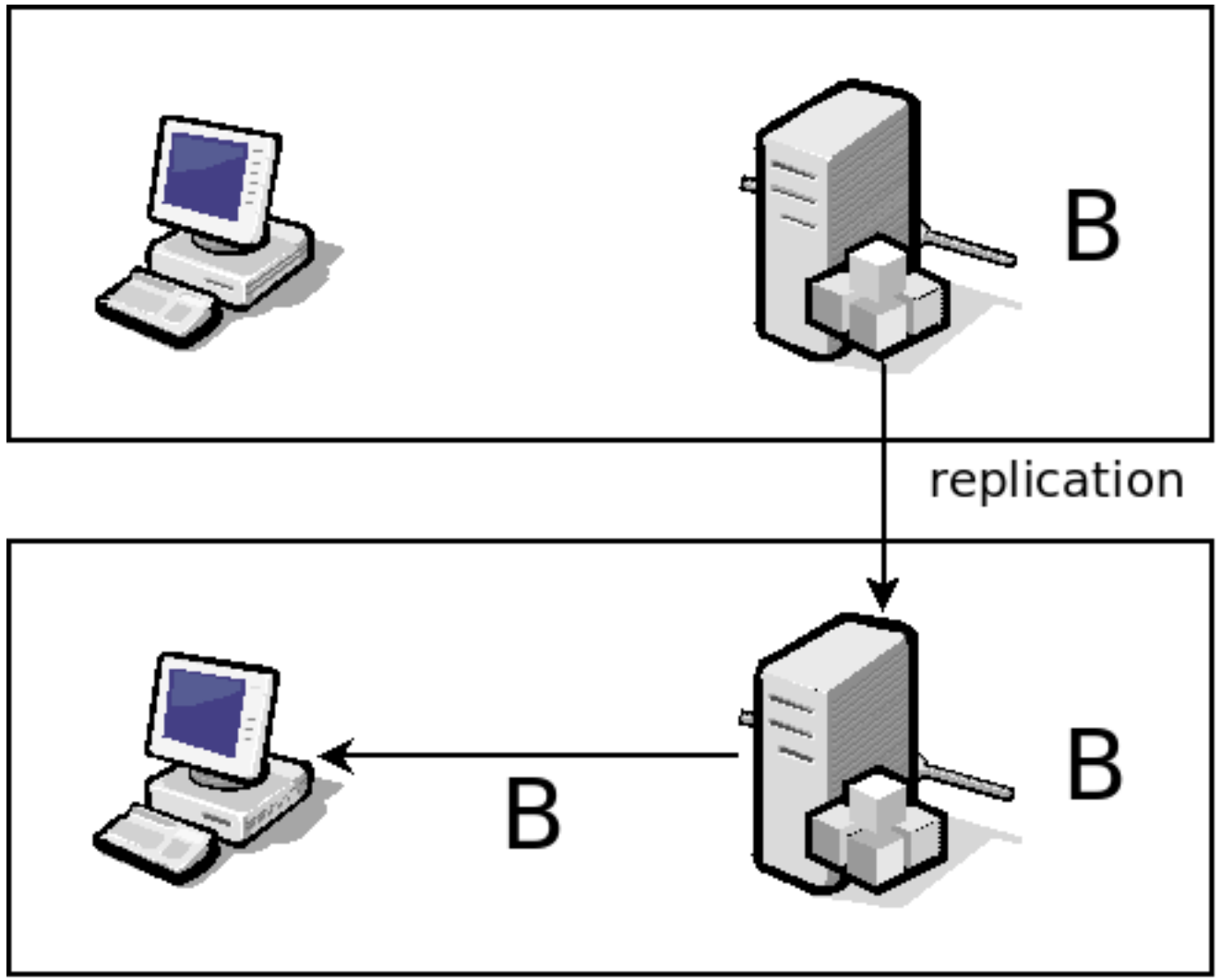

Fonte: http://blog.mattwoodward.com/

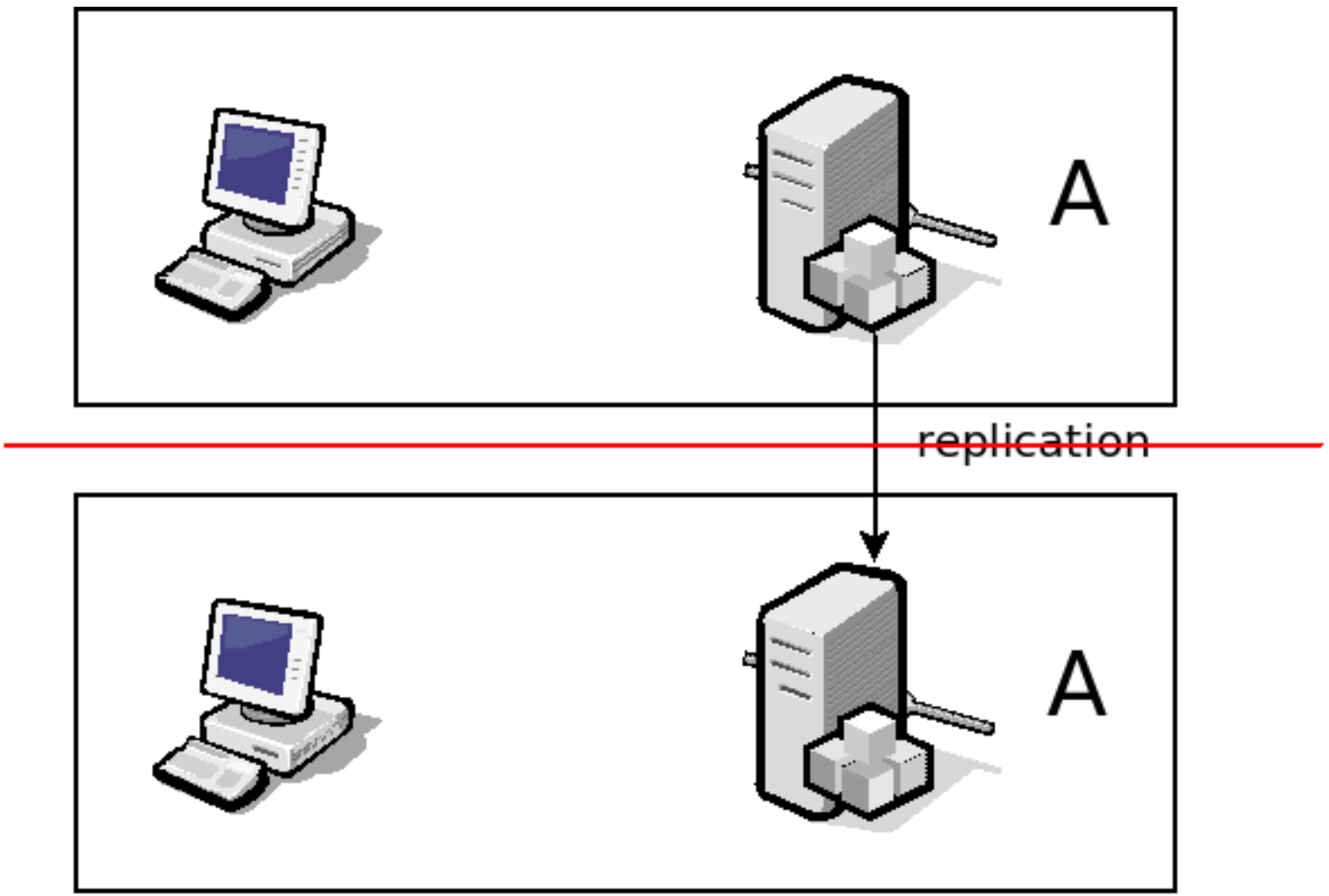
# Consistência em momento indeterminado

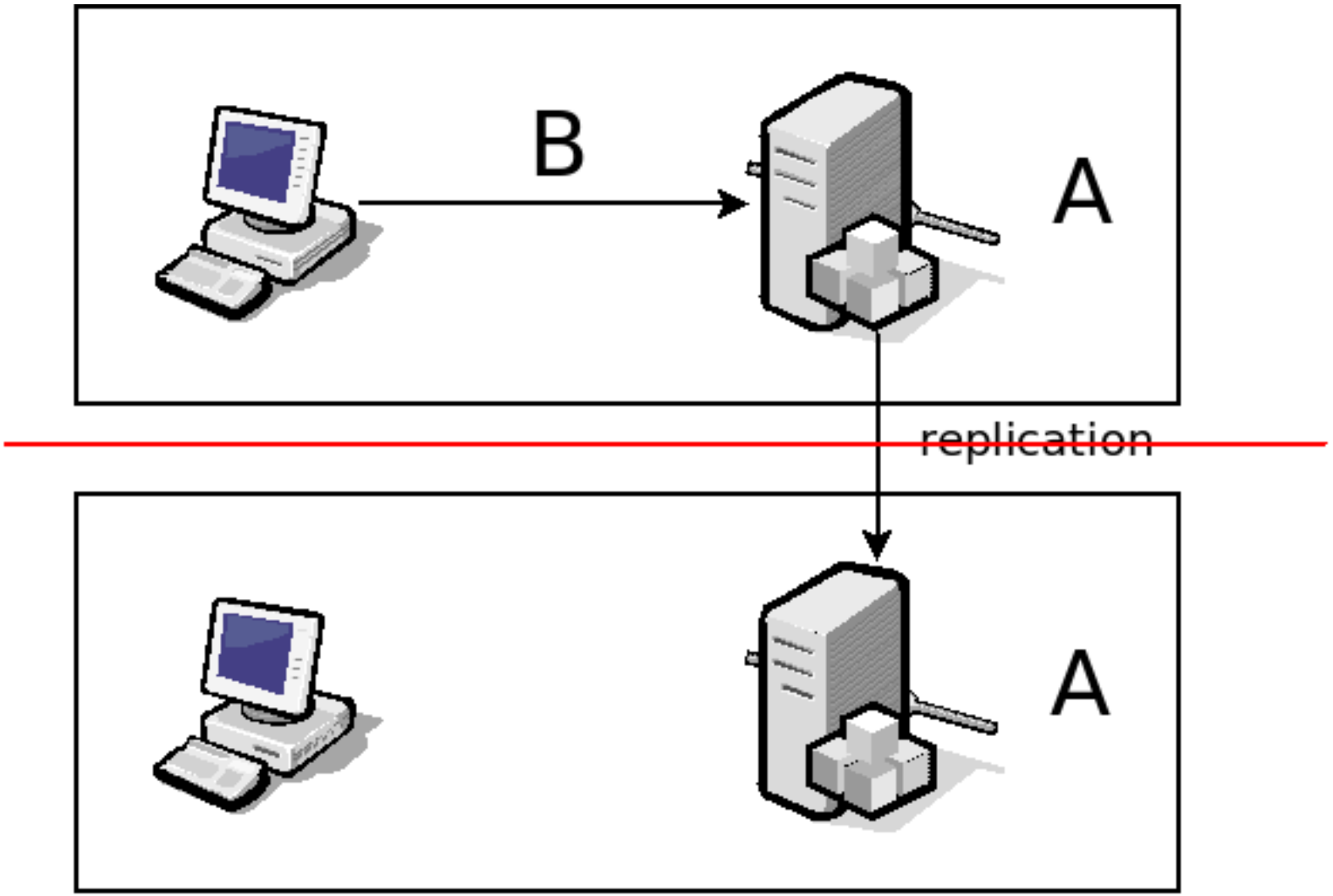


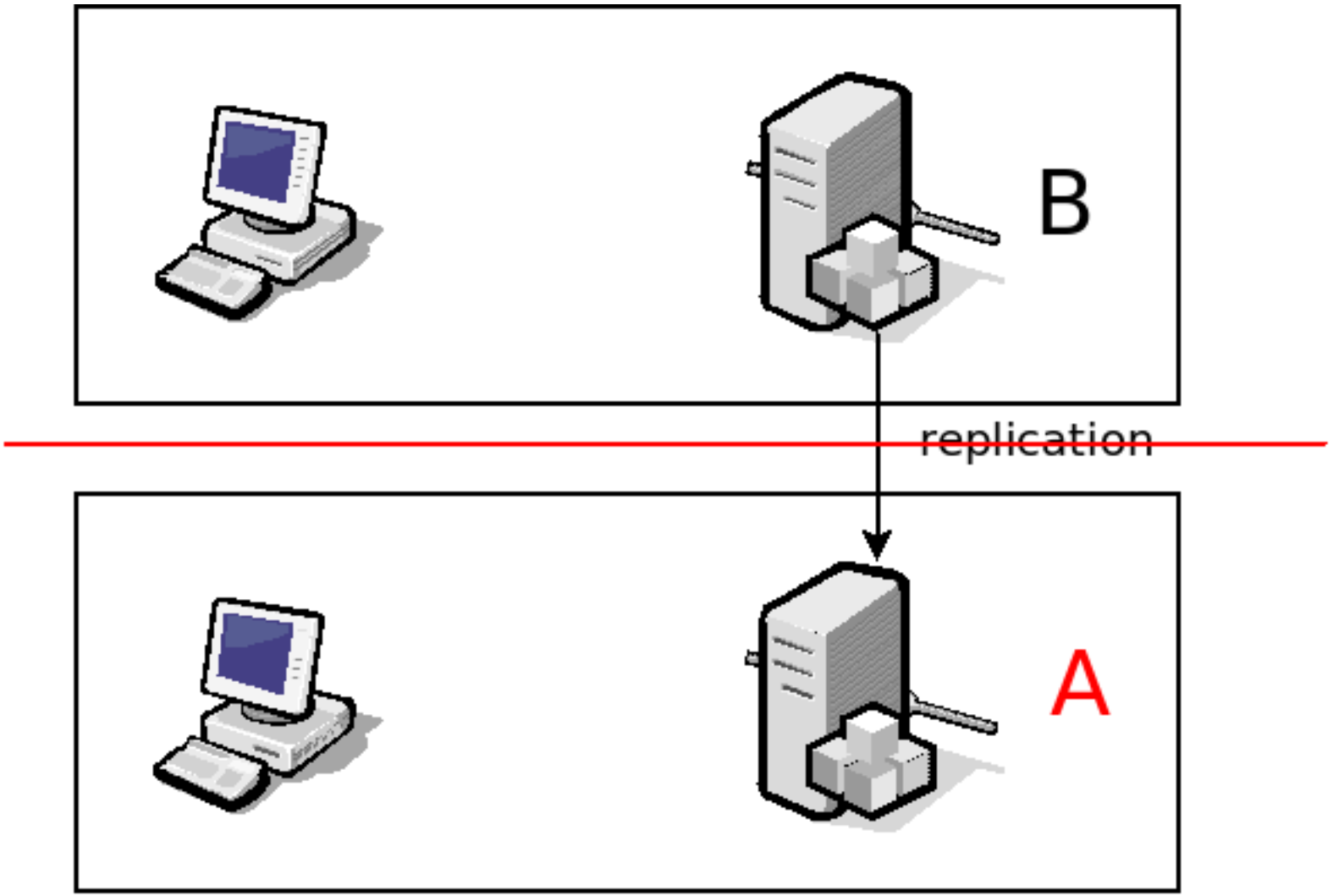

B
A
replication
A

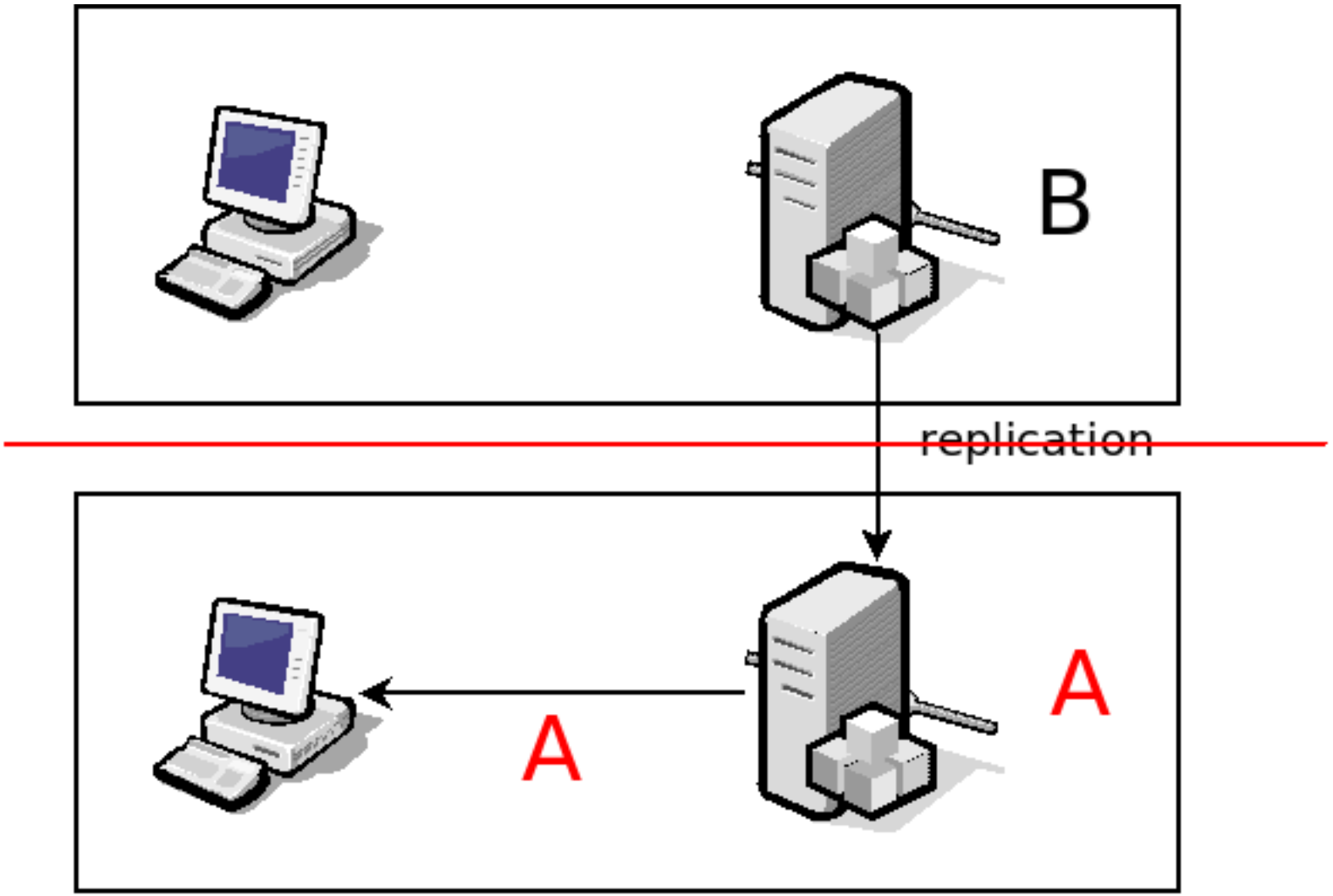

B
replication
B

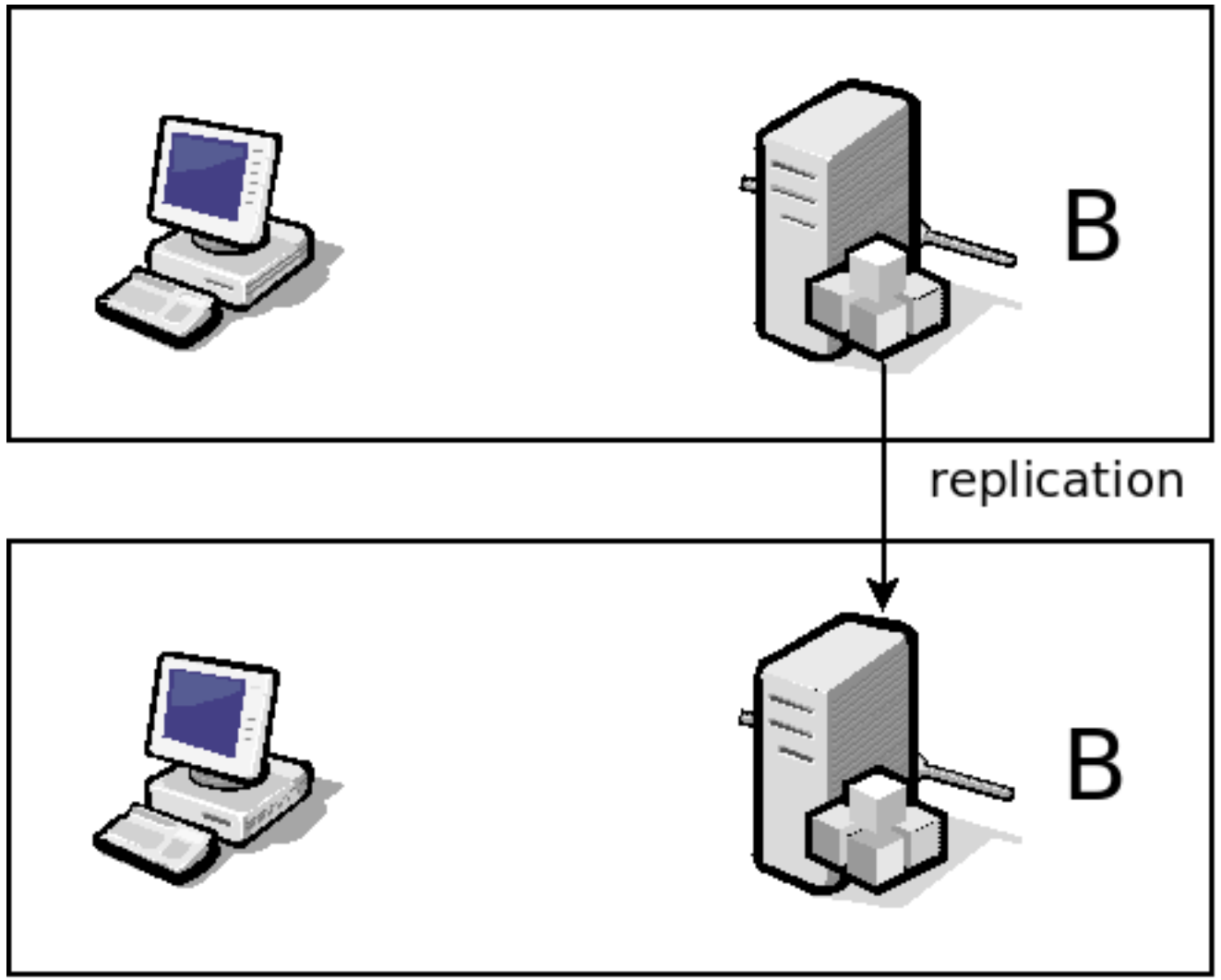

B
replication
B
B


A
replication
A


B
A
replication
A


B
replication
A


B
replication
A
A


B
replication
B

# Consertando inconsistências

Fazer nada

Ação de compensação

# NoSQL - Principais categorias

SGBDs orientados a documentos
*(Document-oriented databases)*

Armazéns chave-valor (DHT)
*(Key-value stores)*

SGBDs orientados a colunas
*(Column-oriented databases)*

SGBDs de grafos
*(Graph databases)*

# SGBDs orientados a documentos

Armazenam coleções de pares de chave-valor

Dados semiestruturados

Esquema flexível

Formatos: JSON, XML, propriedades, etc

*Sharding* simples: documentos sem referências explícitas

Exemplos: MongoDB, CouchDB

# Documento (JSON)

```
{
  autor: 'joao',
  criado : new Date('03-28-2009'),
  titulo : 'Primeiro post',
  texto : 'Aqui está o texto...',
  tags : [ 'exemplo', 'joao' ],
  comentarios : [ { autor: 'edu', comentario: 'Ruim' },
                  { autor: 'maria', comentario: 'Legal' }]
}
```

# Documento (JSON)

```
{
  autor: 'joao',
  criado : new Date('03-28-2009'),
  titulo : 'Primeiro post',
  texto : 'Aqui está o texto...',
  modificado: new Date('05-06-2009'),
  modificado por: 'jose',
  tags : [ 'exemplo', 'joao', 'jose' ],
  comentarios : [ { autor: 'edu', comentario: 'Ruim' },
                  { autor: 'maria', comentario: 'Legal' }]
}
```

# Acesso

## MongoDB (exemplo)

```
> j = { name : "mongo" };
{"name" : "mongo"}

> t = { x : 3 };
{ "x" : 3 }

> db.things.save(j);
> db.things.save(t);
> db.things.find();
{"name" : "mongo" ,
"_id" : ObjectId("497cf60751712cf7758fbdbb")}
{"x" : 3 , "_id" : ObjectId("497cf61651712cf7758fbdbc")}
```

# Acesso

## MongoDB (exemplo)

```
> db.things.find({name:"mongo"}).forEach(
                        function(x) { print(tojson(x));});
{"name" : "mongo" ,
"_id" : ObjectId("497cf60751712cf7758fbdbb")}

> db.things.find({x:3}).forEach(
                        function(x) { print(tojson(x));});
{"x" : 3 , "_id" : ObjectId("497cf61651712cf7758fbdbc")}
```

# Armazéns chave-valor

Tabelas de hash distribuídas (DHT)

Exemplos: Dynamo, Redis, Voldemort

Acesso: get / put

# SGBDs orientados a colunas

Armazenam dados por colunas, não linhas

Mais eficiência manipulando todos os registros (OLAP)

Linhas não têm a mesma quantidades de colunas

Esquema flexível

Exemplos: Bigtable, Cassandra, HBase

# Formato

EmpId, Sobrenome, Nome, Salário

Orientado a linha:

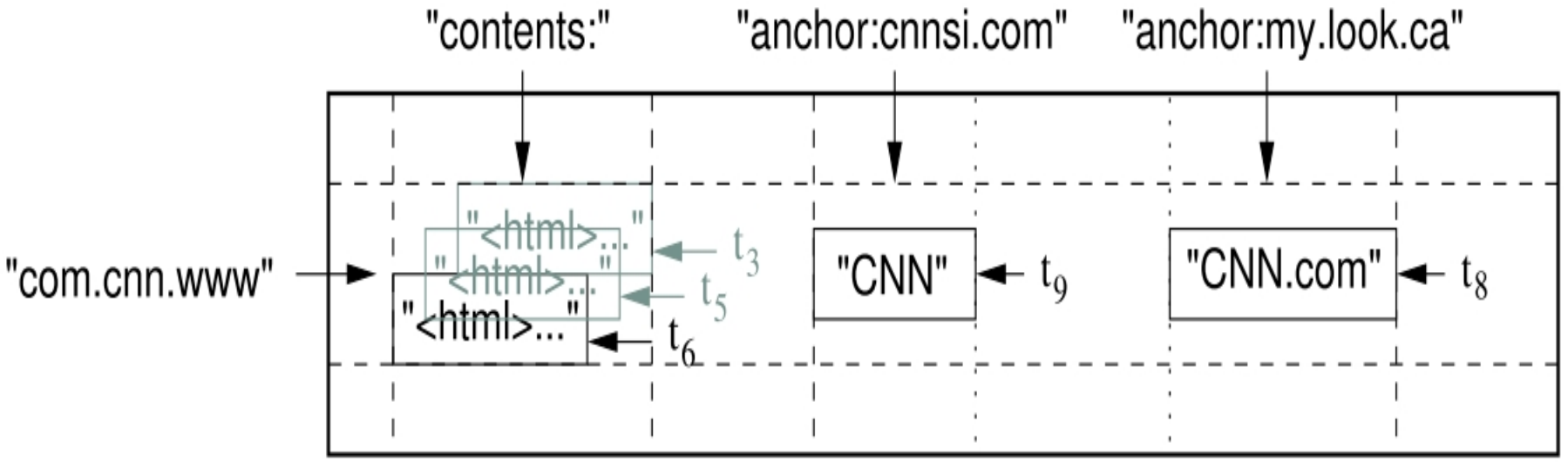
```
1,Silva,João,600;2,Dias,Maria,500;3,Matos,José,440;
```

Orientado a coluna:

```
1,2,3;Silva,Dias,Matos;João,Maria,José;600,500,440;
```

# Acesso

Bigtable (exemplo)
Linha, coluna (família de coluna, qualificador), *timestamp*




Fonte: Bigtable: A Distributed Storage System for Structured Data

# SGBDs de grafos

Armazenam vértices e arestas (e suas propriedades)

Representam interconectividade entre os dados

Operações sobre grafos: percorrer caminho

Exemplos: Infogrid, Neo4J

# Acesso

Neo4J (exemplo)

```
Traverser friendsTraverser = root.traverse(
        Traverser.Order.
BREADTH_FIRST,                    StopEvaluator.
END_OF_GRAPH,
        ReturnableEvaluator.
ALL_BUT_START_NODE,             RelTypes.KNOWS,
        Direction.OUTGOING);
```

# Modelos não-relacionais

Orientado a documentos: dados semiestruturados

Chave-valor: modelo simples

Orientados a coluna: OLAP

Grafos: interconectividade dos dados é tão ou mais importante quanto os dados em si

# NoSQL é a solução?

Cargas de trabalho diferentes precisam de tratamento diferente (Stonebraker)

| OLTP | OLAP |
| --- | --- |
| Operação | Informação (BI) |
| CRUD pequeno e rápido | Batches demorados |
| Consultas simples | Consultas complexas |

# Experimentos

OLTP

H-Store (VoltDB)

82x mais rápido que um SGBD comercial

OLAP

C-Store (Vertica)

124x mais rápido que um SGBD comercial orientado a linha

21x mais rápido que um SGBD comercial orientado a coluna

# Tempos em um SGBDR

*Logging*, *locking*, *latching* e gerenciamento de buffer

1/60 das instruções em uma transação são trabalho útil

20x mais rápido sem esses sub-subsistemas

# Especializações

Problema não está no modelo relacional, nem na SQL

Ordens de grandeza de diferença

Persistência poliglota

# Tendências

Release 2.0 (fev/09)

Claremont Report (jun/09)

ThoughtWorks Radar (jun/10)

# Conclusões

Dois focos:
- Desempenho / Escalabilidade
- Simplicidade

Contexto importa

Necessário balancear vantagens e desvantagens

Futuro incerto (mas promissor)

# Referências

http://nosql-database.org/

http://nosql.mypopescu.com/

Varley, Ian Thomas. No Relation: The Mixed Blessings of Non-Relational Databases

Introdução aos Bancos de Dados Não-Relacionais e NoSQL: Vantagens, Desvantagens e Compromissos (http://www.slideshare.net/mdediana)

# Projetos

MongoDB

Redis

Neo4J

# MongoDB

"MongoDB bridges the gap between key-value stores (which are fast and highly scalable) and traditional RDBMS systems (which provide rich queries and deep functionality)."

"O novo M em LAMP"

# MongoDB

Home: http://www.mongodb.org/

Download: apt-get install mongodb-stable

Shell: http://www.mongodb.org/display/DOCS/mongo+-+The+Interactive+Shell

WebAdmin: http://localhost:28017/

Comandos: db.listCommands() no shell ou http://localhost:28017/_commands

# MongoDB

Driver Java: http://github.com/downloads/mongodb/mongo-java-driver/mongo-2.1.jar

Tutorial Mongo Java: http://www.mongodb.org/display/DOCS/Java+Tutorial

API: http://api.mongodb.org/java/2.1/index.html

No lab: mongod / mongo

# Redis

"Redis is an advanced key-value store. It is similar to memcached but the dataset is not volatile, and values can be strings, exactly like in memcached, but also lists, sets, and ordered sets. All this data types can be manipulated with atomic operations to push/pop elements, add/remove elements, perform server side union, intersection, difference between sets, and so forth. Redis supports different kind of sorting abilities."

# Redis

Home: http://code.google.com/p/redis/

Download: http://code.google.com/p/redis/downloads/list

Docs: http://code.google.com/p/redis/wiki/index?tm=6

Comandos: http://code.google.com/p/redis/wiki/CommandReference

Jedis:
http://github.com/downloads/xetorthio/jedis/jedis-1.1.1.jar

No lab: /opt/redis-2.0.1/redis-server e /opt/redis-2.0.1/redis-cli

# Neo4j

"Neo4j is a graph database. It is an embedded, disk-based, fully transactional Java persistence engine that stores data structured in graphs rather than in tables. A graph (mathematical lingo for a network) is a flexible data structure that allows a more agile and rapid style of development."

# Neo4j

Home: http://neo4j.org/

Download: http://dist.neo4j.org/neo4j-apoc-1.1.tar.gz

Tutoriais:
http://dist.neo4j.org/basic-neo4j-code-examples-2008-05-08.pdf http://www.infoq.com/articles/graph-nosql-neo4j

# Código de exemplo

Paca