

N★SQL



- Arthur Azevedo
- Rafael Benedito


Aviso!

- **O que você vai ver/aprender nessa apresentação:**
 - Conceitos de banco de dados NoSQL;
 - Taxonomia de banco de dados NoSQL;
 - Conceitos de Banco de Dados distribuídos;
 - Comparações entre Bancos RDBMS e NoSQL;
 - Exemplos de banco de dados NoSQL;
 - Mitos sobre bancos NoSQL;
- **O que você não vai ver/aprender nessa apresentação:**
 - Implementação de algum banco NoSQL;
 - Dominar o mundo;
 - Ficar rico;

O que essas empresas tem em comum ?

The Facebook logo, featuring the word "facebook" in white lowercase letters on a blue rectangular background.The LinkedIn logo, featuring the word "Linked" in black and "in" in white inside a blue square, with a registered trademark symbol.The Amazon logo, featuring the word "amazon" in black lowercase letters with a curved orange arrow underneath.The eBay logo, featuring the word "eBay" in a stylized font with each letter in a different color: red, blue, yellow, and green.The Twitter logo, featuring the word "twitter" in a light blue, rounded, lowercase font with a white outline.

sones

The Yahoo! logo, featuring the word "YAHOO!" in a red, bold, serif font with a registered trademark symbol.The Google logo, featuring the word "Google" in its characteristic multi-colored font.

- Quando você digita pindamonhangaba no google, e ele traz: "Aproximadamente 20.500.000 resultados (0,15 segundos)", ANTES DE VOCÊ TERMINAR DE DIGITAR, você acha que ele está fazendo um SQL like em um índice???



pindamonhangaba



Pesquisar

Aproximadamente 8 400.000 resultados (0,26 segundos)

Tudo

Imagens

Mapas

Vídeos

Notícias

Shopping

Mais

João Pessoa - PB

Alterar local

A Web

Páginas em português

Páginas de Brasil

Páginas estrangeiras
traduzidas



Pindamonhangaba - SP :: www.pindamonhangaba.sp.gov.br

www.pindamonhangaba.sp.gov.br

Site Oficial da Prefeitura de **Pindamonhangaba** - SP

SP :: www.pindamonhangaba.sp.gov.br - [Vagas de empregos](#) - [Localização](#) - [Hino](#)

Pindamonhangaba - São Paulo maps.google.com.br



Não é feitiçaria,
é tecnologia!



Pindamonhangaba – Wikipédia, a e

pt.wikipedia.org/wiki/Pindamonhangaba

A região da atual **Pindamonhangaba** foi ocupada desde 22 de julho de 1643, registro mais recente.

[Observações](#) - [História](#) - [Moreira César](#) - [Geografia](#)



VIDA DE PROGRAMADOR

.COM.BR

/* SUGESTÃO
ENVIADA POR
@OSLISSA */



#98

CARA, ESTOU FAZENDO
UM CV PARA UMA VAGA...
PODE ME AJUDAR?

O QUE VOCÊ JÁ
COLOCOU?



SEI MEXER NO COMPUTADOR,
DIGITAR, VER E-MAIL,
MEXO NO FRONTPAGE...

MANJA DE SQL?

NÃO...



ENTÃO BOTA AÍ:
"ESPECIALISTA EM
NOSQL"

AH, LEGAL...



NO SQL / Not Only SQL (Não apenas SQL)

- **NO SQL / Not Only SQL**
Não apenas SQL

- Diferentes sistemas de armazenamento de dados para resolver problemas em que os RDBMS não são a melhor solução
- Algo em torno de 10% dos casos
- Hype: alta escalabilidade

Um pouco de história

- Cinco NECESSIDADES do mercado, NÃO SÃO ATENDIDAS a contento pelos produtos de banco de dados e fornecedores disponíveis, são elas:
 - 1. Escalabilidade
 - 2. Performance
 - 3. Consistência Eventual ou Relaxada
 - 4. Agilidade
 - 5. Complexidade

Um pouco de história

- 1. BigTable: A Distributed Storage System for Structured Data
 - 1. Publicado pelo Google;
 - 2. Em Novembro de 2006;
 - 3. No 17º simpósio em design e implementação de sistemas operacionais;
- 2. Dynamo: Amazon's Highly Available Key-Value Store
 - 1. Publicado pela Amazon;
 - 2. Em Outubro de 2007;
 - 3. No 12º simpósio em princípios de sistemas operacionais ;

Principais Características NoSQL

- Escalabilidade Horizontal
- Replicação
- Schema-free
- Clusterização
- MapReduce
- Sharding

Escalabilidade Horizontal

- **Escalabilidade Horizontal (scale out)**
significa adicionar mais nós ao sistema, tais como adicionar um novo servidor e um sistema de software que permita a distribuição do trabalho entre múltiplas máquinas.

Replicação - Escalar por duplicação de informações

Consiste na copia das informações em mais de um banco para aumentar nossa capacidade de recuperar estas informações. Podemos dividir em duas “arquiteturas” principais:

- Master-Slave: Cada escrita em banco resulta em X escritas onde X é o número de slaves. Neste caso temos um banco “Master” que propaga cada write para os bancos “slaves”. Isto aumenta a nossa velocidade de leitura, mas não melhora a capacidade de escrita.
- Multi-Master: Aumentamos o número de Masters em nosso sistema e assim aumentamos nossa capacidade de escrita.

Schema-free

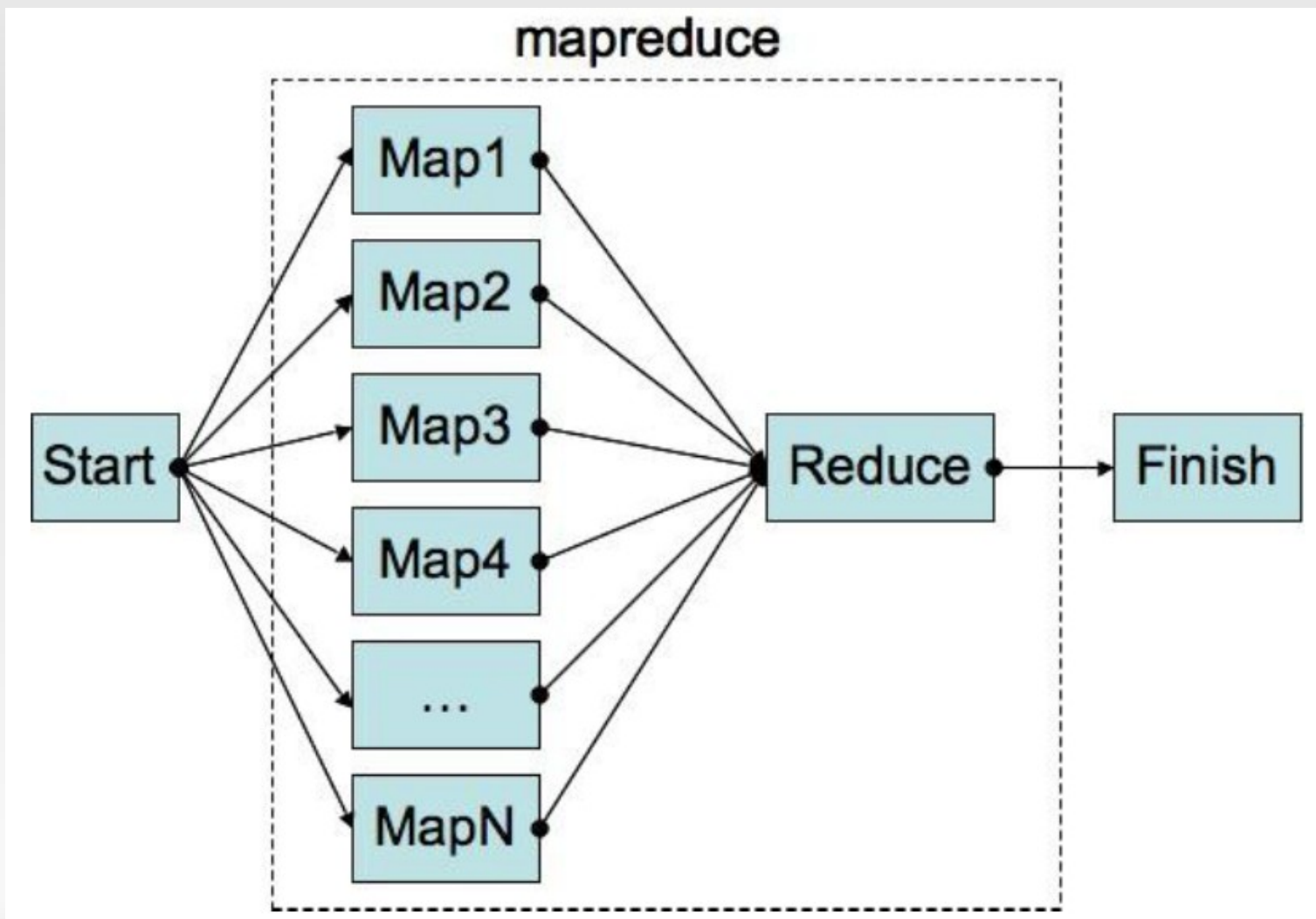
- Schema-free: Um dos fatores que contribuem para um banco de dados
- NoSQL é escalar por causa da ausência de um schema (schema free). Todos os novos bancos tem em comum que eles são key-value stores, ou seja salvam, como o nome sugere, um conjunto de entradas formadas por uma chave associada a um valor e o valor poderia ser de qualquer tipo, um binário ou string que está sendo salvo de forma desnormalizada (schema-free).

Clusterização

- Clusterização: Basicamente compreende um banco de dados armazenado e gerenciado por mais de um servidor, provê uma alta disponibilidade e um alto desempenho do sistema. Assim, a organização se beneficia diminuindo o tempo de inoperabilidade do banco de dados. Esse processo vem como uma solução para reduzir gastos com estrutura de hardware.

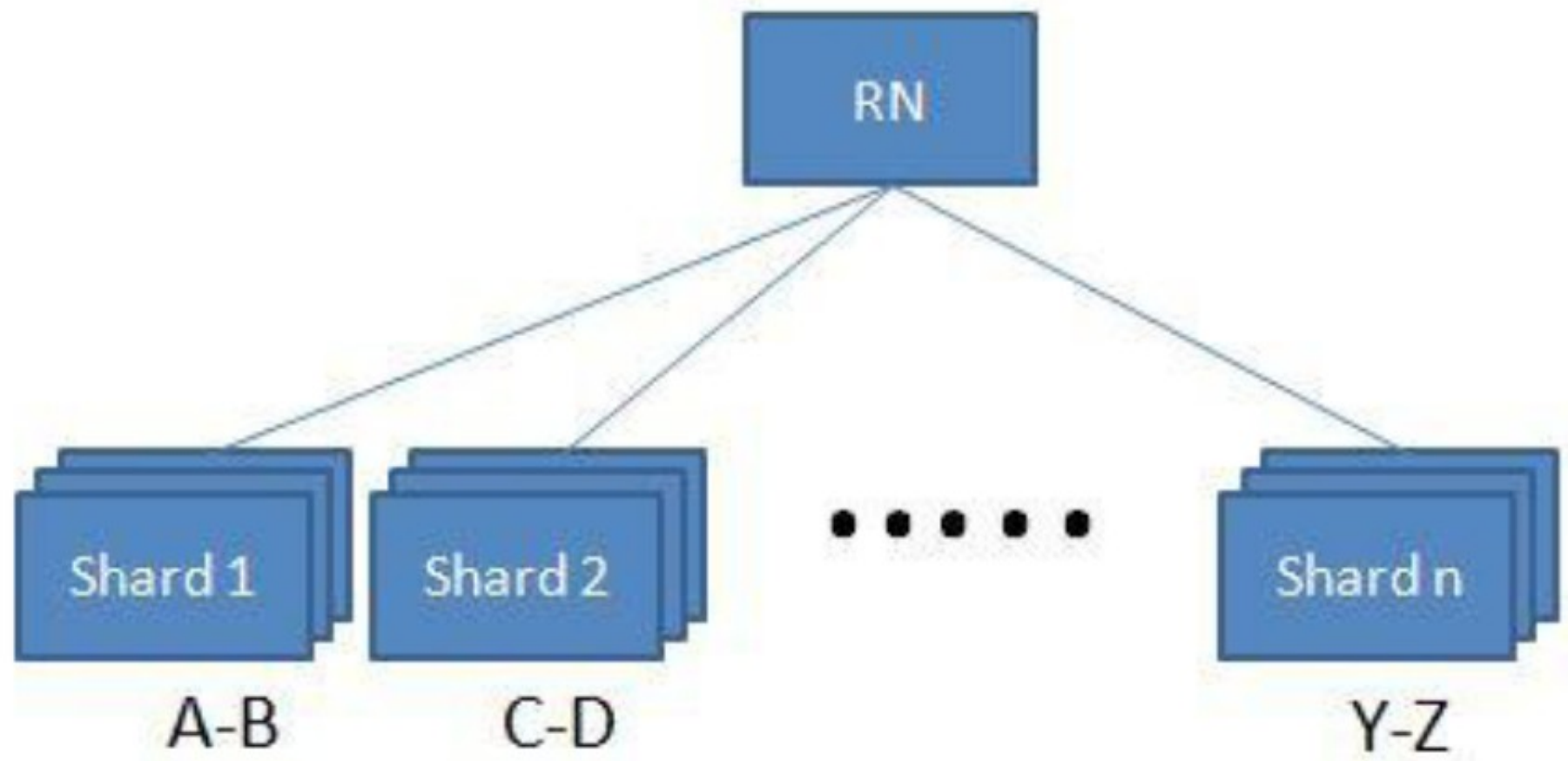
Mapreduce

- Mapreduce: É um algoritmo, patenteado pela Google para gerenciamento em larga escala. Existem duas fases:
 - Map: O nó principal recebe os dados, divide e partes menores e as envia aos outros nós para serem processados. Ao final do processamento estes nós devolvem o resultado ao nó principal.
 - Reduce: O nó principal combina as respostas obtidas pelos outros nós gerando o resultado final do processamento.



Sharding

- Sharding: Consiste em dividir os dados horizontalmente, ou seja, quebrar as tabelas, diminuindo o seu número de linhas e separando-as em ambientes diferentes. Neste ponto todos os dados de uma partição não devem conter referências aos dados de outras partições, sendo que os dados em comum deverão ser replicados entre as bases.



Categorias do NOSQL

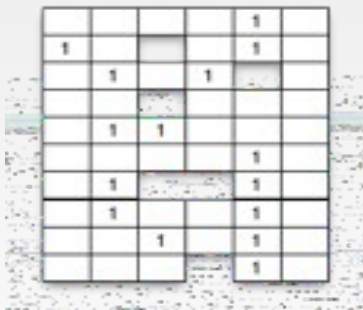
Key-Value



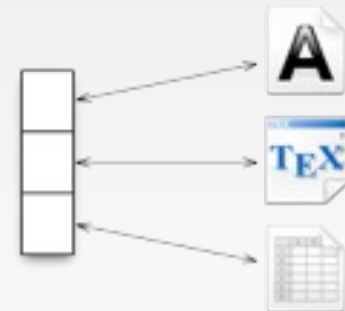
Graph DB



BigTable



Document



Key-value (Chave-Valor)

- Focado em escalabilidade para grandes quantidades de dados;
- Projetado para lidar com grande carga de dados;
- Baseado no Amazon's Dynamo paper;
- Modelo de dados:(Global) coleção de pares chave-valor
- Anel Dynamo de particionamento e replicação;
- Exemplos:
 - Dynamite
 - Voldemort
 - Tokyo

Key-Value



min_index	key100
-----------	--------



next_index	key104
------------	--------



key	value
...	...
key100	page#:6
key101	page#:12
key102	page#:5
key103	page#:4
...	...

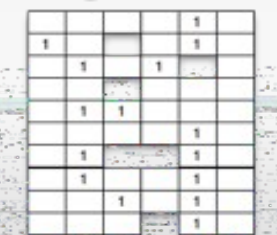


新着順

Big Table clones

- Como bancos de dados relacionais orientado a colunas, mas com twist(toque);
- Tabelas similares aos RDBMS, mas de forma semi-estruturada;
- Baseado no Google's BigTable paper;
- Modelo de dados: ► Colunas → Famílias de colunas → ACL
 - Datas chaveados por: linha, coluna, tempo , índice
 - Row-range → tabela → distribuição
- Exemplos:
 - Hbase
 - Hyperbase
 - Cassandra

BigTable

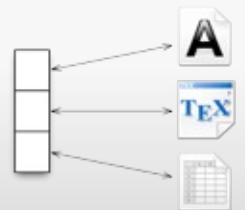


				1	
1				1	
	1			1	
	1	1			
				1	
	1			1	
	1			1	
		1			
				1	

Document databases

- Similar ao chave-valor, mas o BD sabe qual é o valor;
- Inspirado pelo Lotus Notes;
- Modelo de dados: Coleções de coleções chave-valor;
- Documentos são frequentemente versionados;
- Exemplos:
 - CouchDB
 - MongoDB
 - Redis

Document

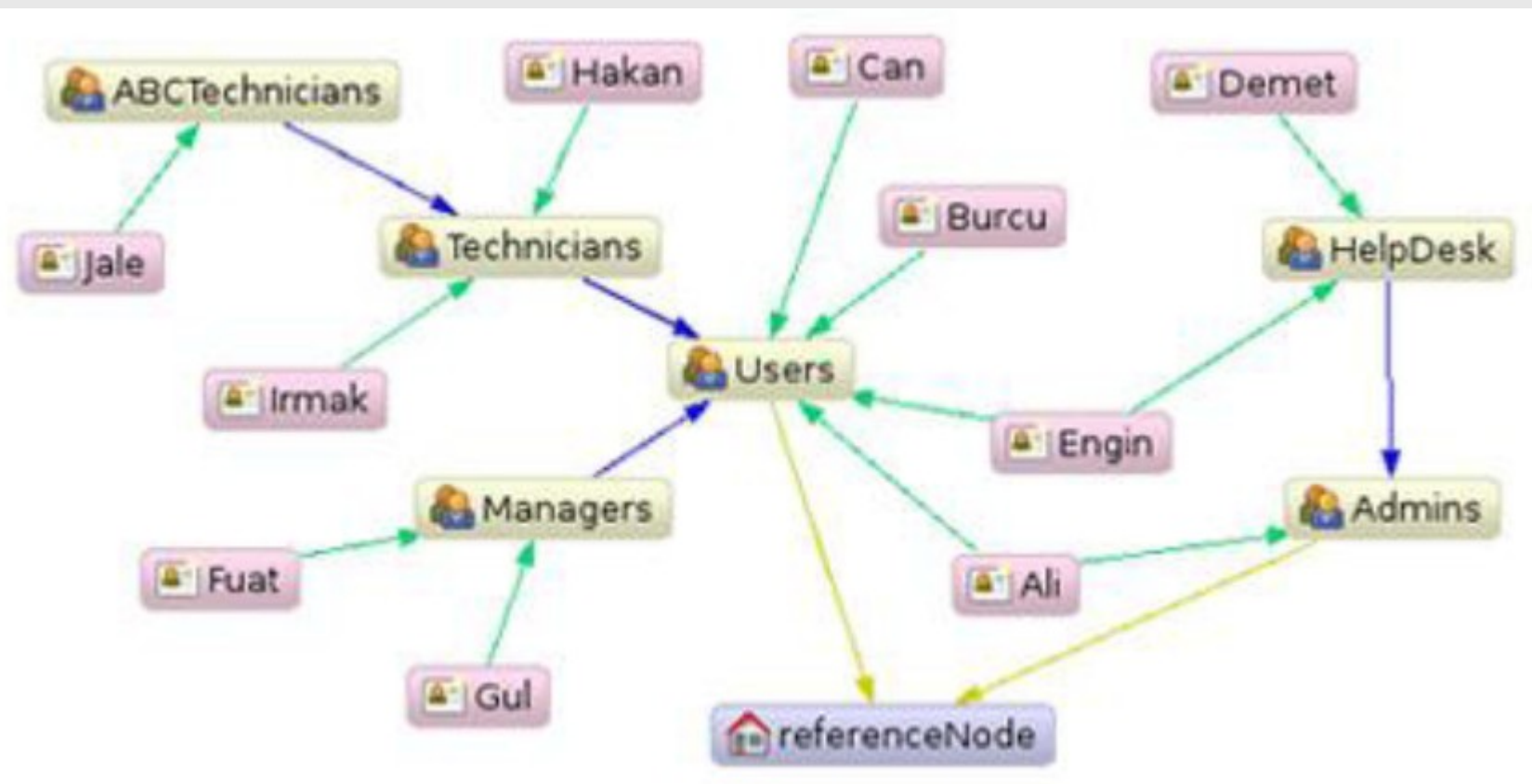


Graph databases

- Focado na modelagem de estruturas de dados – interconectividade;
- Escala à complexidade dos dados;
- Inspirado no teorema matemático dos Grafos($G=(E,V)$);
- Modelo de dados: "Propriedade Grafo" ► Nós
 - Relacionamentos entre nós (primeira classe);
 - Pares de chaves-Valores em ambos;
- Examples
 - Neo4j
 - AllegroGraph

Graph DB



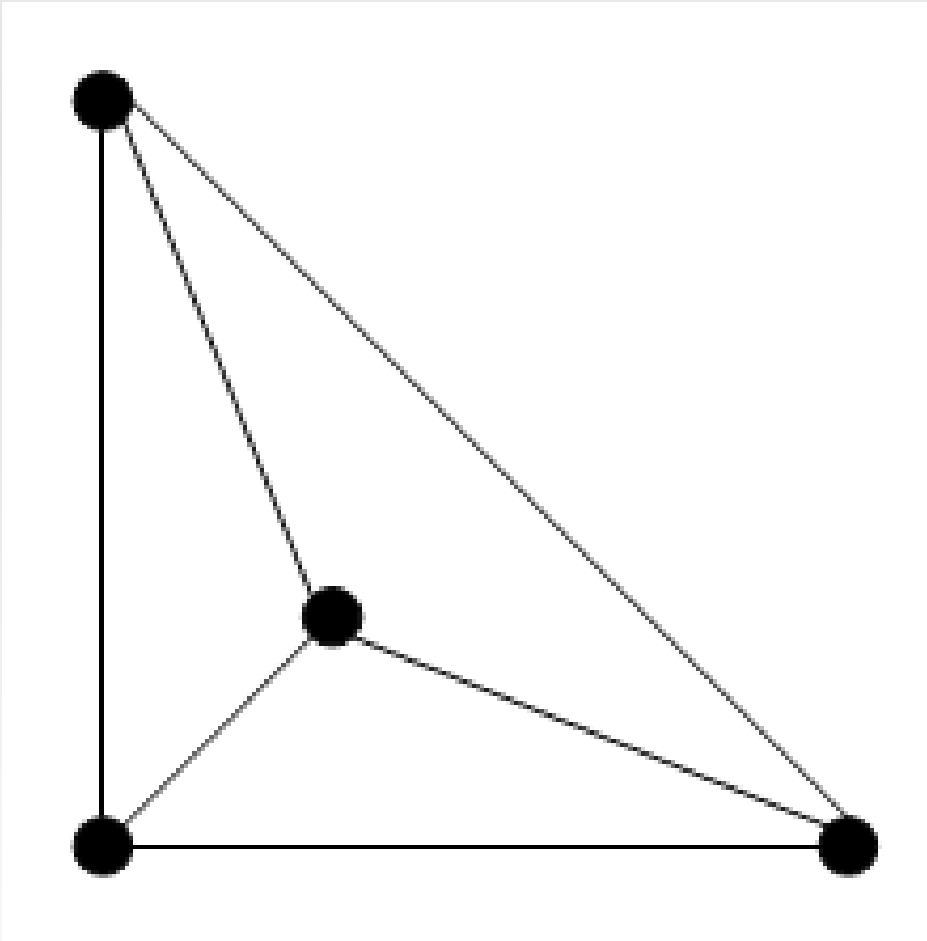


Uma breve pausa.

- Teoria dos grafos

- A teoria dos grafos é um ramo da matemática que estuda as relações entre os objetos de um determinado conjunto.
- Grafo com 4 vértices e 6 arestas. É um grafo completo, conexo e planar.
- Grafo é uma estrutura $G(V,A)$ onde V é um conjunto não vazio de objetos denominados vértices e A é um conjunto de pares não ordenados de V , chamado arestas.
- Dependendo da aplicação, arestas podem ou não ter direção, pode ser permitido ou não arestas ligarem um vértice a ele próprio e vértices e/ou arestas podem ter um peso (numérico) associado. Se as arestas têm uma direção associada (indicada por uma seta na representação gráfica) temos um grafo direcionado, grafo orientado ou digrafo.

Exemplificando...



- Grafo com 4 vértices e 6 arestas.
- É um grafo completo, conexo e planar.

Scaling to size vs. Scaling to complexity



O teorema do Caos, ops CAP :)

- **C**onsistency (Consistência)
 - Todos os clientes podem ver os mesmos dados.
- **A**vailability (Disponibilidade)
 - Todos os clientes podem sempre acessar os dados
- **P**artition tolerance (Tolerância a partições)
 - A capacidade de continuar a trabalhar mesmo quando a topologia de rede é quebrada.
 - A capacidade de recuperar quando a rede volta.
- Dos 3 se preocupe com somente 2 =)

CA: Consistency & Availability

- Tolerância a partição é comprometida;
- Site clusters simples (fácil garantir que todos os nós estão sempre em contato);
- Quando ocorre uma falha, o sistema bloqueia, por exemplo Two Phase Commit (2PC);



CP: Consistency & Partitioning

- Disponibilidade é comprometida;
- Acesso a alguns dados pode ser temporariamente limitado;
- Por exemplo Sharded database



AP: Availability & Partitioning

- Consistência é comprometida;
- Sistema ainda está disponível sob particionamento;
- Alguns dados retornados podem ser temporariamente não atualizados;
- Requer uma estratégia de resolução de conflitos;
- Por exemplo, DNS, cache, replicação master/slave;





ACID vs. BASE



ACID – uma rápida recapitulação

- **A**tomacidade
 - Quando uma parte da transação falhar, toda a transação falha;
 - O estado do BD é inalterado;
- **C**onsistência
 - Uma transação leva o banco de dados de um estado consistente para outro;
- **I**solamento
 - Uma transação não pode ler o estado sujo de outras transações (não pegar lixo)
- **D**urabilidade
 - Commit significa commit. (Comitou, comitou!)

BASE

- O complemento CAP para o ACID
 - Só que tinha que ser chamado BASE =)
- **B**asically **A**vailable (Basicamente disponível)
- **S**oft State (Estado suave)
- **E**ventually Consistent (Consistência eventual)

RDBMS & ACID / NoSQL & BASE

- RDBMSs se esforçam para garantir ACID
 - ACID força a consistência
- Soluções NoSQL muitas vezes escalam através de BASE
 - BASE aceita que os conflitos vão acontecer

Comparações



Comparando estrutura de Dados

- RDBMS
 - Banco de Dados contém tabelas, colunas e linhas;
 - Todas as linhas com a mesma estrutura;
 - Incompatibilidade inerente com ORM(Mapeamento objeto-relacional)
- NoSQL
 - Escolha a sua estrutura de Dados;
 - Os dados são armazenados na estrutura natural (por exemplo, documentos, gráficos, objetos)

Comparando Flexibilidade de Schema

- RDBMS
 - Esquema rigoroso, difícil de evoluir
 - Mantém relações e força a integridade de dados
- NoSQL
 - Estruturas de dados podem ser alteradas dinamicamente
 - Por exemplo Column stores – Cassandra
 - Os dados podem às vezes ser completamente opacos
 - Por exemplo Key/Value – Project Voldemort

Comparando Normalização & Relacionamentos

- RDBMS
 - O modelo de dados é normalizado para remover a duplicação de dados;
 - Normalização estabelece relações entre as tabelas
- NoSQL
 - Desnormalização não é uma palavra "feia"
 - Relações não são explicitamente definidas
 - Dados relacionados são geralmente agrupados e armazenados como uma unidade
 - Por exemplo, document, column

Comparando Acesso à Dados

- RDBMS
 - Operações CRUD usando SQL
 - Acesso aos dados de várias tabelas usando JOIN
 - API genérica, como JDBC
- NoSQL
 - API proprietária e DSLs (por exemplo, Pig/Hive/Gremlin)
 - MapReduce, Grafos transversais
 - REST APIs, formatos portáteis de serialização
 - BSON, JSON, Apache Thrift, Memcached

RDMS vs. NoSQL na prática!

mySQL

SELECT

```
Dim1, Dim2,  
SUM(Measure1) AS MSum,  
COUNT(*) AS RecordCount,  
AVG(Measure2) AS MAvg,  
MIN(Measure1) AS MMin  
MAX(CASE  
  WHEN Measure2 < 100  
  THEN Measure2  
END) AS MMax  
FROM DenormAggTable  
WHERE (Filter1 IN ('A','B'))  
  AND (Filter2 = 'C')  
  AND (Filter3 > 123)  
GROUP BY Dim1, Dim2  
HAVING (MMin > 0)  
ORDER BY RecordCount DESC  
LIMIT 4, 8
```

- ① Grouped dimension columns are pulled out as keys in the map function, reducing the size of the working set.
- ② Measures must be manually aggregated.
- ③ Aggregates depending on record counts must wait until finalization.
- ④ Measures can use procedural logic.
- ⑤ Filters have an ORM/ActiveRecord-looking style.
- ⑥ Aggregate filtering must be applied to the result set, not in the map/reduce.
- ⑦ Ascending: I; Descending: -I

MongoDB

```
db.runCommand({  
  mapreduce: "DenormAggCollection",  
  query: {  
    filter1: { '$in': [ 'A', 'B' ] },  
    filter2: 'C',  
    filter3: { '$gt': 123 }  
  },  
  map: function() { emit(  
    { d1: this.Dim1, d2: this.Dim2 },  
    { msum: this.measure1, recs: 1, mmin: this.measure1,  
      mmax: this.measure2 < 100 ? this.measure2 : 0 }  
  ); },  
  reduce: function(key, vals) {  
    var ret = { msum: 0, recs: 0, mmin: 0, mmax: 0 };  
    for(var i = 0; i < vals.length; i++) {  
      ret.msum += vals[i].msum;  
      ret.recs += vals[i].recs;  
      if(vals[i].mmin < ret.mmin) ret.mmin = vals[i].mmin;  
      if((vals[i].mmax < 100) && (vals[i].mmax > ret.mmax))  
        ret.mmax = vals[i].mmax;  
    }  
    return ret;  
  },  
  finalize: function(key, val) {  
    val.mavg = val.msum / val.recs;  
    return val;  
  },  
  out: 'result1',  
  verbose: true  
});  
db.result1.  
  find({ mmin: { '$gt': 0 } }).  
  sort({ recs: -1 }).  
  skip(4).  
  limit(8);
```

Alguns Bancos NoSQL

- Cassandra
- Hbase
- Voldemort
- CouchDB
- MongoDB
- Neo4j
- E muito mais =)

Cassandra



- Cassandra é um projeto da Apache, que tem como objetivo fornecer uma solução escalável de banco de dados distribuídos, emprega tecnologia do Amazon Dynamo e projetos do Big Table do Google.
- Era open-source pelo Facebook em 2008 e foi projetado por um dos autores originais do Dynamo(Avinash Lakshman), em conjunto com um engenheiro do Facebook Prashant Malik.
- Ele é usado por muitas grandes empresas, como Rackspace, Digg, Facebook, Twitter, e Cisco.
- O servidor de maior produção tem mais de 150 TB de dados.
- Cassandra está disponível sob a licença Apache 2.0.

Características-chave:

- Descentralizado: Todos os nós são iguais; não existe um ponto único de falha.
- Tolerância a falhas: Dados são automaticamente replicados para múltiplos nós para tolerar falhas;
 - Suporte para replicação entre vários data centers;
 - O nó falho pode ser substituído sem deixar o sistema inativo.
- Eventualmente consistente: Dados eventualmente usam um modelo consistente, com otimizações sugeridas como Hinted Handoff e Reparo de leitura para minimizar a inconsistência das janelas.
- Elasticidade: Novas máquinas podem ser adicionadas dinamicamente sem deixar o sistema inativo, operações de leitura/escrita escalam linearmente quando um novo servidor é adicionado.

- Rico modelo de dados: os registros mais complexos são suportados como simples key-values stores;
- Performance comparado ao MySQL.
 - Por exemplo:

MySQL > 50 GB Data

Writes Average: ~300 ms

Reads Average: ~350 ms

Cassandra > 50 GB Data

Writes Average: 0.12 ms

Reads Average: 15 ms

- Cassandra é escrito em Java.
- Utiliza a estrutura de serviços Apache Thrift para fornecer acesso em várias linguagens. Baseado em suas raízes ele enfatiza a plataforma linux, mas pode ser usado no windows também.
- Em meados de março de 2010, Cassandra foi movido de um projeto de incubadora para um projeto de nível superior. Consequentemente, agora tem sua própria página web de nível superior: <http://cassandra.apache.org>.
- Cassandra é muito ativo e vem progredindo rapidamente.

HBase (Apache)



- HBase é um banco de dados colunar “big table”, modelado a partir do BigTable do Google e construído em cima do framework Apache’s Hadoop. Ainda é muito novo, versão atual 0.90x. HBase suporta tabelas extremamente grandes e princípios de banco de dados distribuídos como:
 - Armazenamento elástico (adição dinâmica de novos servidores)
 - Nenhum ponto único de falhas
 - Suporte para processamento MapReduce
 - Serviços implantados com o Apache Thrift
 - REST-ful web service gateway

- Hadoop é usado por pesquisas Yahoo , Facebook, Bing Microsoft , e outros grandes fornecedores. Hadoop e HBase são projetos de código aberto Apache, e ambos são oferecidos pela Amazon Web Services através da sua oferta de Amazon (AWS).
- HBase é um banco de dados colunar projetado para escala em todas as direções: milhares de milhões de linhas, cada uma das quais pode ter milhões de colunas. Cada interseção de linha / célula é uma célula que pode ser versionado. O número de versões armazenadas é configurável.

- Cada tabela Hbase é nomeada. As linhas são classificadas por Ids de linha, que é um valor definido pela matriz de bytes do usuário. As tabelas são organizadas em regiões, que são armazenadas separadamente. Quando uma região fica cheia, ela se divide e algumas linhas são movidas. Cada região tem Ids de linha definidos por [primeira linha, última linha].
- As colunas são organizadas em famílias de colunas , que devem ser declaradas. Por exemplo, uma coluna da família "estação" pode ser declarada, posteriormente um aplicativo pode adicionar valores com os nomes das colunas como "estação:id", "estação:nome", "estação:descrição", etc.

- Hbase usa o framework Apache Zookeeper para acesso distribuído. Quando mapeado para o Hadoop HDFS, Hbase implementa uma classe de arquivo especial que fornece recursos como compressão e filtros. Compressão GZIP é usada por padrão, mas compressão LZO pode ser instalado no Hbase para uma compressão mais rápida e em tempo real.
- Hbase, como grande parte do Hadoop é escrito em Java e enfatiza o uso em plataformas linux, mas funciona também no windows.

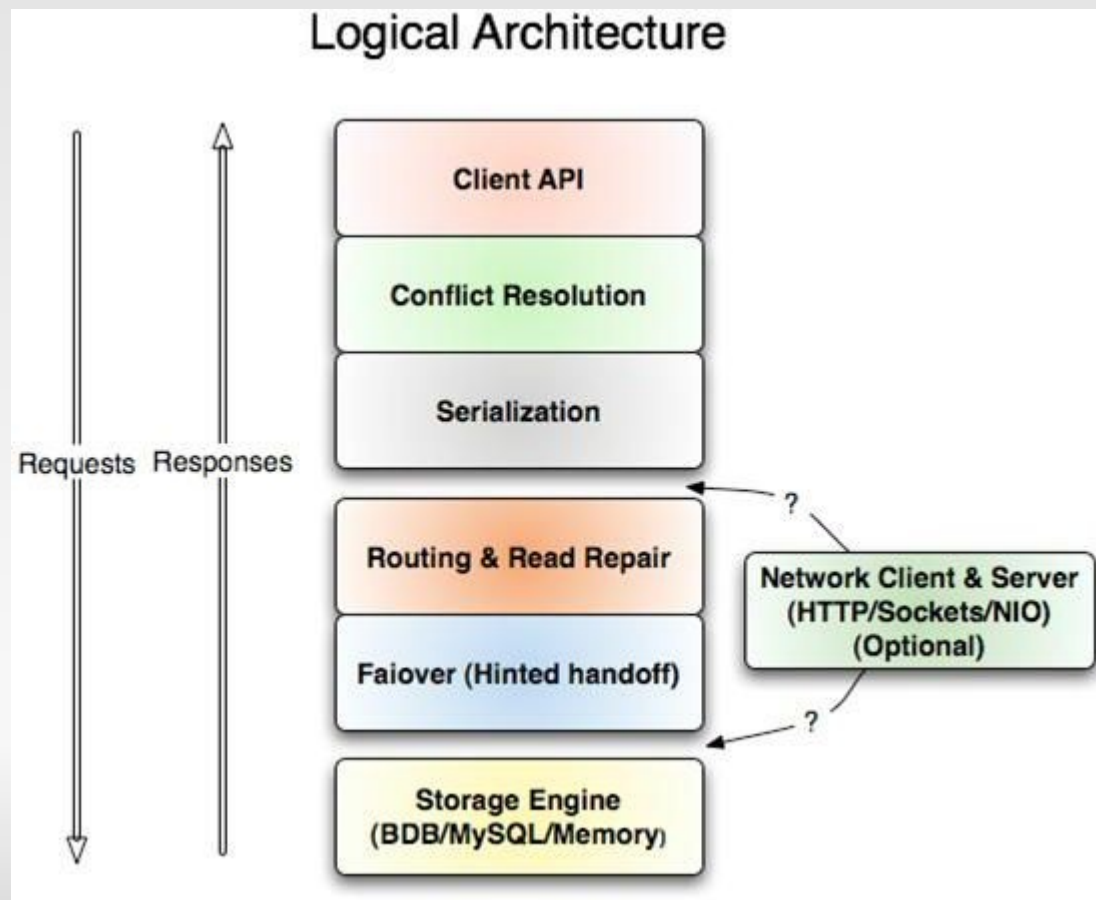
Voldemort (Open Source)

Project Voldemort
A distributed database.

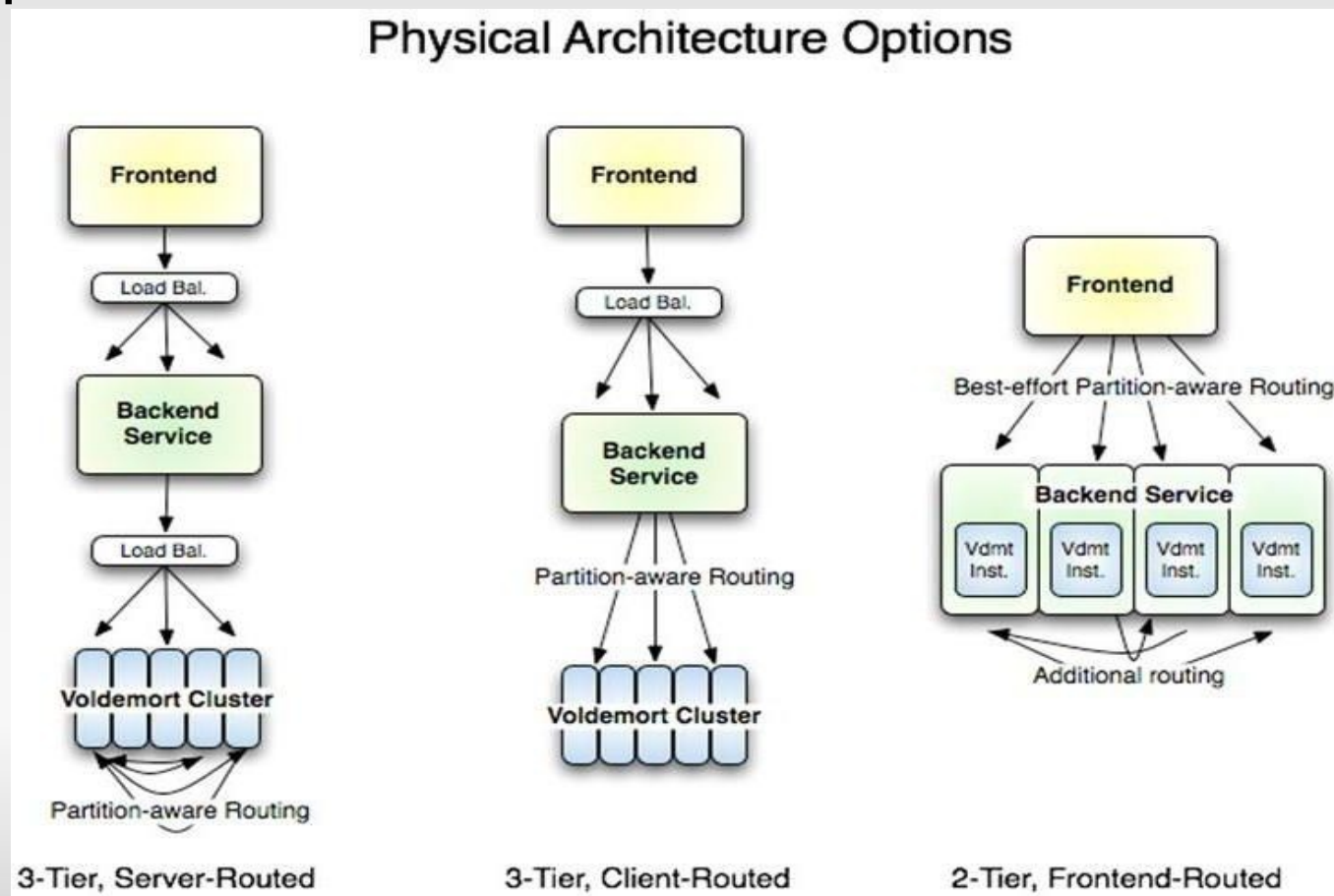
- O banco de dados Voldemort foi construído pelo LinkedIn para as suas necessidades.
- É um key/value store que utiliza princípios Dynamo para processamento distribuído, replicação e tolerância a falhas.
- Pode ser considerado o “Cassandra 1.0” já que o mesmo designer deixou o LinkedIn para criar um banco de dados semelhante para o Facebook.
- Uma vez que o armazenamento é key/value store, Voldemort não suporta diretamente relacionamentos. Além disso, as relações 1-para-muitos devem ser tratadas usando um Mapa incorporado (Ex: `java.util.HashMap`) como um valor.

- Os projetistas afirmam que esse design simplista é uma vantagem já que o desempenho é muito previsível.
- Voldemort teve recentemente seu código aberto e esta disponível sob licença Apache 2.0.
- Stemming from the Dynamo research, Voldemort uses “read repair” for conflict detection and resolution instead of two-phase commit or Paxos-style consensus algorithms. This also means that versioning is used and that applications must participate in conflict resolution.

- A arquitetura de Voldemort emprega uma camada distinta para resolver cada função lógica: a resolução de conflitos de serialização, etc. Isto é ilustrado abaixo:



- Como mostrado, Voldemort requer um mecanismo de armazenamento, como Berkeley DB ou MySQL. Várias opções de arquitetura física estão disponíveis para apoiar vários balanceamentos de carga e metas de desempenho. Isso é mostrado abaixo:



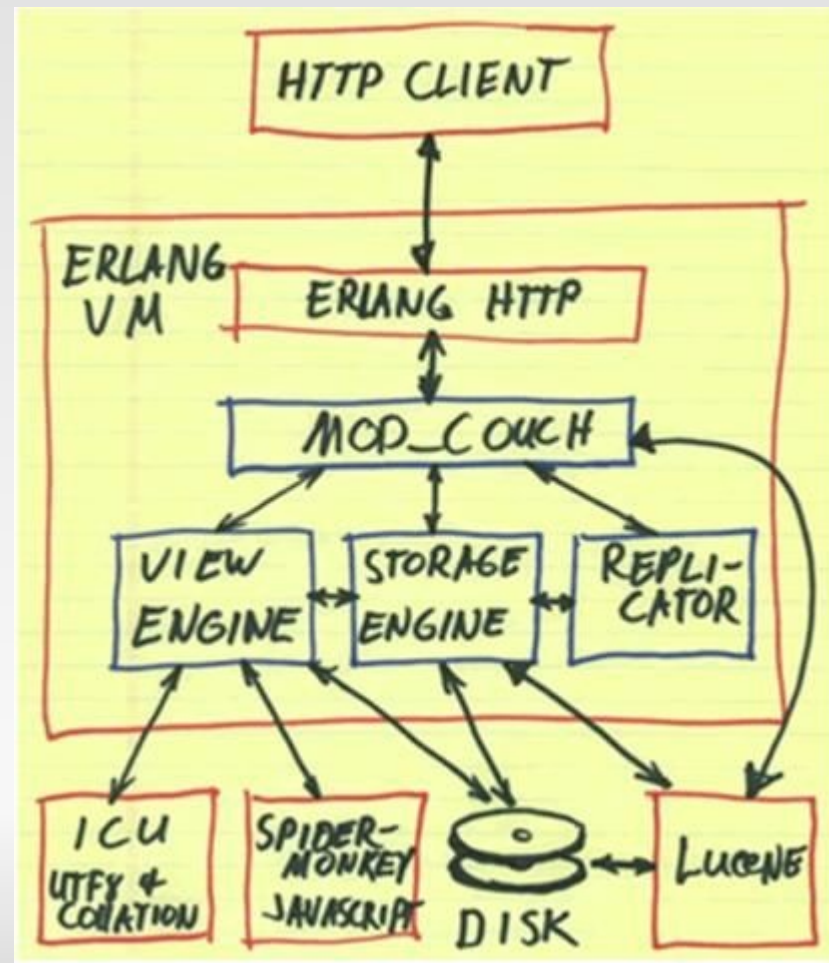
- A API nativa de Voldemort é Java, mas o acesso também é possível via HTTP.
- Porque o código do Voldemort só foi aberto recentemente sua vantagem de adoção em relação a outras opções não é clara. A sua dependência da BDB ou MySQL como uma backend store limita seu uso para aplicações comerciais.

CouchDB (Apache)



- CouchDB é um banco de dados orientado a documentos, muitas vezes foi chamado de banco de dados orientado a documentos “garoto propaganda”. É um ad-hoc, esquema de banco de dados flexível com espaço de armazenamento plano. Escrito em Erlang, CouchDB pode ser consultado e indexado usando expressões MapReduce. Ele suporta JSON e um acesso estilo REST.

- Uma visão geral de sua arquitetura a partir do site da Apache é mostrada abaixo:



- CouchDB usa incrementação, replicação assíncrona com detecção de conflitos e de gerenciamento. A arquitetura multi-master permite que um master pode ficar off-line (escritas ficam na fila até que o master fique online).
- Sob o capô, CouchDB é um armazenamento de chave / valor em que IDs de documentos mapeiam os documentos. Um documento CouchDB é um objeto que consiste em campos nomeados. Valores de campo podem ser strings, números, datas ou até mesmo listas ordenadas e mapas associativos.

- Um exemplo de um documento é um post:

```
"Subject": "I like Plankton"  
"Author": "Rusty"  
"PostedDate": "5/23/2006"  
"Tags": ["plankton", "baseball", "decisions"]  
"Body": "I decided today that I don't like baseball. I like plankton."
```

- Um banco de dados CouchDB é uma coleção plana destes documentos, cada documento identificado por um ID único. Pesquisa de texto completo é fornecido por um índice Lucene. Índices adicionais são visões definidas usando o padrão MapReduce e JavaScript para métodos. (Essa abordagem tem sido descrito como "estranho".)

- Emil Eifrém e Adam Skogman descrevem CouchDB como sendo difícil de usar. Outras pesquisas internas em Quest também categorizam como um banco de dados de propósito específico.
- CouchDB parece ser um dos primeiros exemplos de um DB documento e é melhor usado para aplicações que necessitam especificamente dessa funcionalidade.

MongoDB (Open Source)

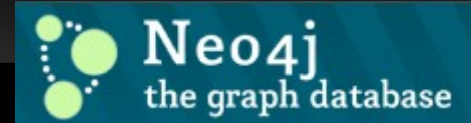


- MongoDB é um sistema de banco de dados orientado a documentos escrito em C++ uma vez foi descrito como “CouchDB sem as loucuras”. No seu site é definido como escalável, de alta performance, open source, livre de esquemas, orientado a documentos. Principais características:
 - Armazenamento orientado a documentos(a simplicidade e o poder de esquemas JSON como dados)
 - Consultas dinâmicas

- Suporte completo para índices, exetendendo-se até inner-objects e arrays embutidos
- Perfil de consulta
- Rápido, nas atualizações locais
- Armazenamento eficiente de objetos de dados binários(Ex: fotos e videos)
- Replicação sem falhas
- Auto-sharding para níveis escaláveis de nuvens

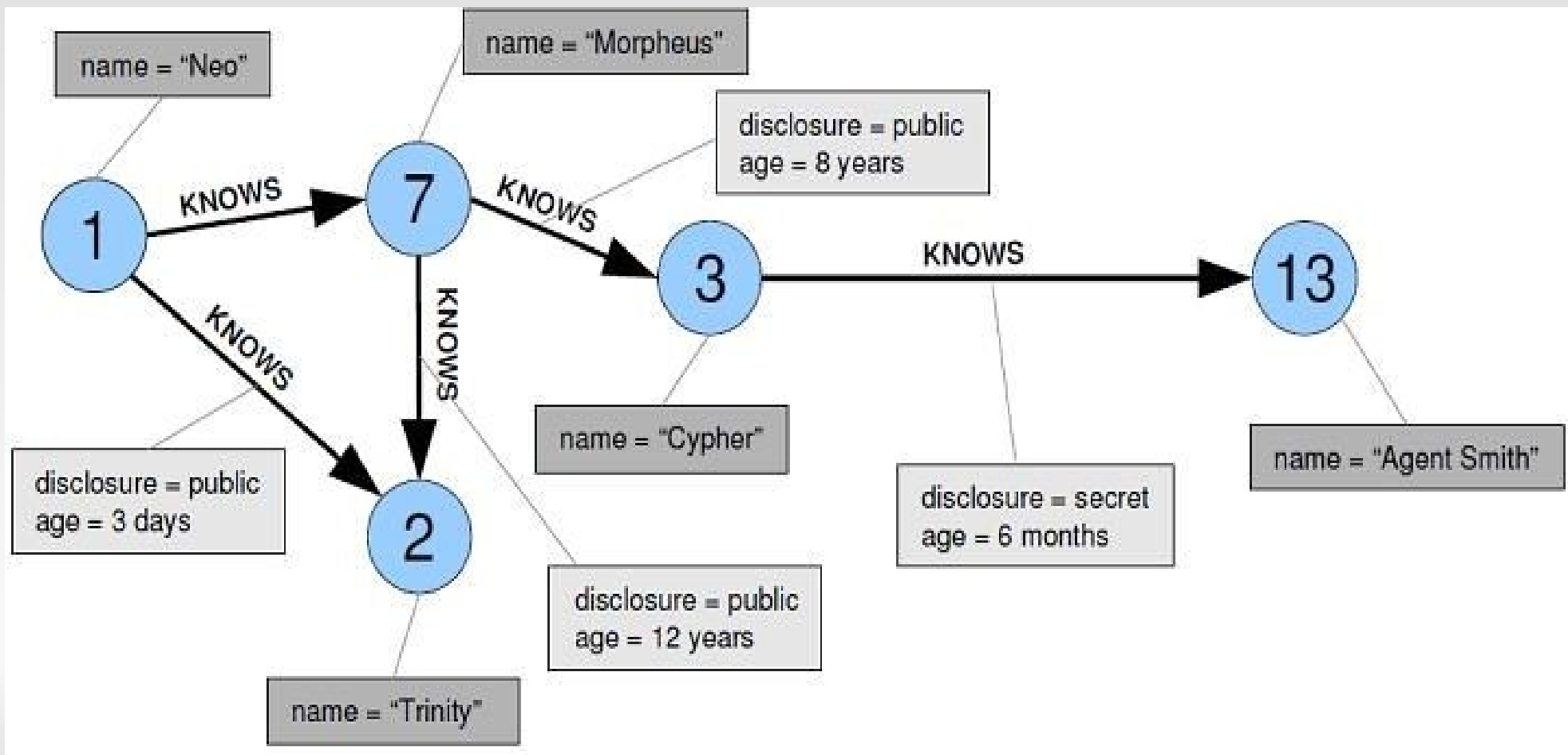
- MapReduce para agregações complexas
- Suporte comercial, treinamento e consultoria
- Acesso programático está disponível em diversas linguagens como C, C++, C#, Erlang, Java, Python, Ruby, e muito mais.

Neo4j (Neo Technologies)



- Neo4j é um banco de dados orientado a grafos para aplicações embarcadas. É escrito em Java, mas tem clientes para Python, Ruby, Clojure, and Scala. Ele suporta transações ACID e fornece replicações master-slave. É fornecido pela Neo Technologies e tem ambas as licenças, AGPL e Comercial. O arquivo JAR do Neo4J tem apenas 500kb, e supostamente suporta bilhões de nós, relacionamentos e propriedades em um único sistema.

- Um exemplo de um grafo típico de informação é mostrado abaixo:



- Uma referência a Neo4j sugeriu que havia suporte muito básico de replicação master/slave, mas é feito aparentemente através de um mecanismo muito suave, como um replicador de disco. Menções de sharding também são feitas, mas um post no blog explica que este deve ser feito essencialmente no cliente. Como resultado, o suporte embutido a escalabilidade parece ser limitado, e pode ser feito em uma única máquina.
- A falta de estratégia de escalabilidade, o foco em um modelo de grafo, e a necessidade de uma licença comercial parecem fazer Neo4j de uso limitado.

Os maiores mitos sobre NoSQL

- NoSQL é escalável
- Não precisamos de DBAs
- NoSQL é mais econômico

NoSQL é escalável

- Uma das grandes promessas dos bancos NOSQL consiste em dizer que eles são mais escaláveis que os bancos de dados relacionais. O problema com esta mensagem que é vendida por algumas empresas é que ela não é inteiramente verdade. Dizer que seu sistema escala sozinho é vender um sonho. Ele pode até ser mais fácil de escalar se comparado a outras soluções mas ainda sim exigira algum esforço para escalar.

Não precisamos de DBAs

- No mundo dos bancos relacionais a figura do DBA sempre está presente. Com sistemas que tem particularidades para cada vendor os DBAs ficam a cargo de instalar, configurar e manter cada banco de dados em suas particularidades. Muita gente diz que quando se trabalha com NoSQL não precisamos de DBAs. Acredito que talvez não no sentido tradicional, mas ainda vamos precisar de alguém responsável por lidar com o banco e com o acesso aos dados. Esta função pode vir a se tornar parte do trabalho de um desenvolvedor ou se tornar a função full time de alguém no seu time que pode ser até um DBA com conhecimentos em NoSQL. Em aplicações reais em produção muito provavelmente será necessário misturar bancos relacionais e não relacionais, possuir alguém que navegue facilmente nos dois mundos em seu time é uma grande vantagem.

NoSQL é mais econômico

- Meia verdade. Muitos fornecedores de NoSQL afirmam que suas soluções vão baratear o custo dos seus clientes. Em parte sim, em algumas situações o custo em usar um banco de dados relacional pode ser proibitivo devido a escala ou a licenças envolvidas. Existem muitos casos entretanto que uma solução relacional atende perfeitamente todas as necessidades do cliente e ainda sim pode ser considerada barata. Bancos de dados open source como MySQL e PostgreSQL são usados sem problemas por um grande número de aplicações com sucesso.

Dúvidas?

Obrigado!

Referências:

- <http://nosql.findthebest.com/d/I/Open-Source>
- <http://pdfcast.org/download/nosql-for-dummies.pdf>
- <http://nosql-database.org/>
- <http://neo4j.org/>
- <http://escalabilidade.com/2010/04/06/introducao-ao-nosql-partes>
- <http://couchdb.apache.org/>
- <http://www.mongodb.org/>
- <http://cassandra.apache.org/>
- http://nosqlpedia.com/wiki/Survey_distributed_databases#Voldemort
- <http://blog.linkedin.com/2009/03/20/project-voldemort-scaling-s>