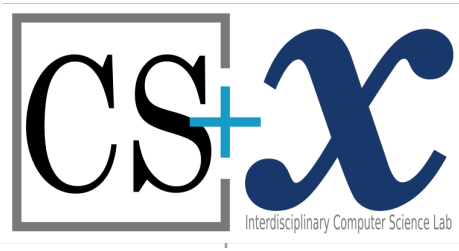
Introdução a Bancos de Dados
NoSQL

Clodoveu Davis
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
clodoveu@dcc.ufmg.br



NoSQL

- Originalmente: “not SQL”, ou BDs não relacionais
  - Curiosamente, há bancos NoSQL mais antigos do que qualquer relacional
- Atualmente “Not Only SQL”, ou não-relacionais, mas podendo usar SQL
  - Uma provocação: “NOt yet SQL”, indicando que ainda ficam atrás dos relacionais em vários aspectos

- Gerenciadores que oferecem mecanismos de armazenamento, organização e recuperação que não utilizam a ideia de tabelas (ou relações, no sentido do modelo relacional)
- Surgem da necessidade de obter melhor desempenho, principalmente em situações que diferem do uso mais comum dos SGBD relacionais

Transação

- SGBDs relacionais implementam o conceito de transação
  - Um bloco de processamento, constituído de diversas operações, incluindo buscas, inserções, exclusões, atualizações
  - Essas operações são interdependentes, ou seja, fazem parte da mesma unidade lógica de processamento no BD
  - Exemplo: transferência bancária

    BEGIN TRANSACTION
    - Entrada: identificação de duas contas correntes, valor a transferir
    - Operação 1: consultar o saldo da conta corrente 1 e verificar se está acima do valor a transferir
    - Operação 2: atualizar a conta 1 e subtrair o valor
    - Operação 3: atualizar a conta 2 e adicionar o valor
    - Operação 4: verificar se a conta 1 tem isenção de tarifa para transferências
    - Operação 5: caso não tenha subtrair da conta 1 o valor da tarifa de transferência

    END TRANSACTION

Transação

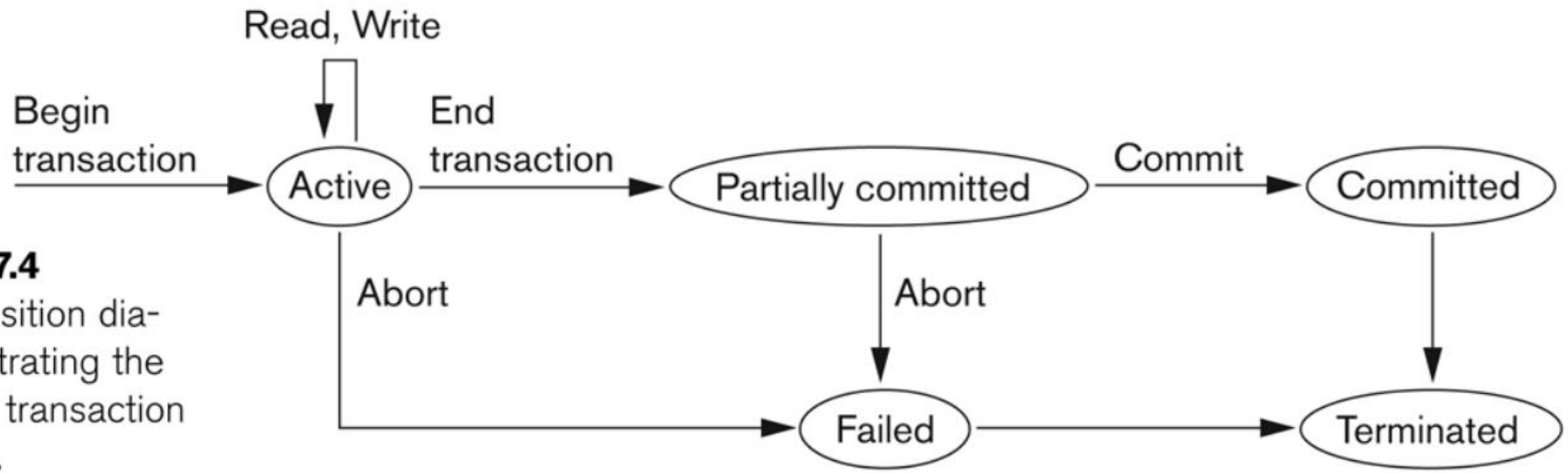


Figure 17.4
State transition diagram illustrating the states for transaction execution.

Bancos de Dados Relacionais: princípios ACID

- Atomicidade

  DEFINIÇÃO DE TRANSAÇÃO

- Consistência

  RESTRIÇÕES DE INTEGRIDADE

- Isolamento

  CONTROLE DE CONCORRÊNCIA

- Durabilidade

  RECUPERAÇÃO DE FALHAS e BACKUP

ACID e desempenho

- Os princípios ACID são um grande argumento em favor do uso de SGBD relacionais
  - Garantia de integridade mesmo sob múltiplos acessos simultâneos
  - Garantia de preservação do conteúdo mesmo em caso de falhas graves
  - Garantia de simplicidade no projeto e implementação das operações no BD
- Porém, garantir ACID representa um custo sobre a eficiência do SGBD
- Embora ACID seja importantíssimo em diversas situações , especialmente quando envolvem um sistema de informação para o qual a segurança e a consistência são críticas, em outras situações é necessário flexibilizar esses princípios

BLOG@CACM

The 'No SQL' Discussion Has Nothing to Do With SQL

By Michael Stonebraker
November 4, 2009
Comments (12)

Recently, there has been a lot of buzz about "No SQL" databases. In fact there are at least two conferences on the topic in 2009, one on each coast. Seemingly this buzz comes from people who are proponents of:

• document-style stores in which a database record consists of a collection of (key, value) pairs plus a payload. Examples of this class of system include CouchDB and MongoDB, and we call such systems **document stores** for simplicity

• key-value stores whose records consist of (key, payload) pairs. Usually, these are implemented by distributed hash tables (DHTs), and we call these **key-value stores** for simplicity. Examples include Memcachedb and Dynamo.

MORE NEWS & OPINIONS

Researchers Find Even 'Fair' Hiring Algorithms Can Be Biased
VentureBeat

The Push for Stricter Rules for Internet Platforms
Pamela Samuelson

NoSQL: A Beginner's Guide

“

Logging: Traditional databases write everything twice; once to the database and once to the log. Moreover, the log must be forced to disk, to guarantee transaction durability. Logging is, therefore, an expensive operation. **Gravações redundantes**

Locking: Before touching a record, a transaction must set a lock on it in the lock table. This is an overhead-intensive operation. **Travamento de registros**

Latching: Updates to shared data structures (B-trees, the lock table, resource tables, etc.) must be done carefully in a multi-threaded environment. Typically, this is done with short-term duration latches, which are another considerable source of overhead. **Conexão e atualização de outras estruturas de dados**

Buffer Management: Data in traditional systems is stored on fixed-size disk pages. A buffer pool manages which set of disk pages is cached in memory at any given time. Moreover, records must be located on pages and the field boundaries identified. Again, these operations are overhead intensive. **Gerenciamento de buffers de memória**

If one eliminates any one of the above overhead components, one speeds up a DBMS by 25%. Eliminate three and your speedup is limited by a factor of two. You must get rid of all four to run a lot faster.

”

“

(...) In effect the “gold standard” of ACID transactions is sacrificed for performance.

However, the net-net is that the single-node performance of a NoSQL, disk-based, non-ACID, multithreaded system is limited to be a modest factor faster than a well-designed stored-procedure SQL OLTP engine. In essence, ACID transactions are jettisoned for a modest performance boost, and this performance boost has nothing to do with SQL.

(...) To go fast, one needs to have a stored procedure interface to a run-time system, which compiles a high-level language (for example, SQL) into low level code. Moreover, one has to get rid of all of the above four sources of overhead.

”

Stonebraker, M. “The NoSQL discussion has nothing to do with SQL”. Communications of the ACM, nov. 2009. https://cacm.acm.org/blogs/blog-cacm/50678-the-no-sql-discussion-has-nothing-to-do-with-sql/fulltext

O Teorema CAP
(Consistency, Availability, Partition Tolerance)

- Em oposição aos princípios ACID
- Consistency
  - Consistência entre réplicas de partes do BD (NÃO é o mesmo que o C do ACID)
  - Assume um sistema com replicação e distribuído
- Availability
  - Disponibilidade do sistema para leituras e gravações
  - Cada requisição de leitura/gravação será processada em sequência, ou retornará com a indicação de que não pode ser realizada
- Partition Tolerance
  - Em relação à organização física dos nós de rede que contêm pedaços (*shards*) do banco de dados
  - O sistema deve poder continuar a operar se a rede tiver uma falha que resulte na criação de novas partições, em que os nós de cada partição possam se comunicar uns com os outros

O Teorema CAP

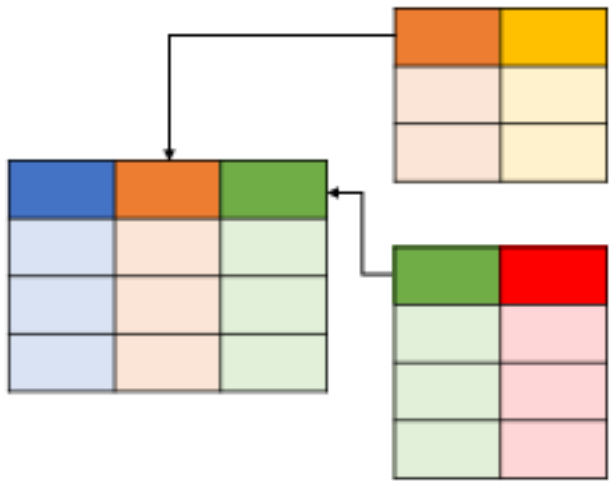
- O Teorema CAP afirma que **não é possível garantir as três propriedades C – A – P simultaneamente** em um sistema distribuído e com replicação de dados
- Nesse caso, o projetista do SGBD tem que escolher duas das três propriedades para preservar
- Em consequência, em geral sistemas NoSQL optam por enfraquecer o C, gerando uma forma de consistência chamada **consistência eventual**

NoSQL

- Principais classes
  - Key-value (chave-valor)
  - Document store (XML databases)
  - Graph database
  - Wide-column store (ou só column-store)

- Outros “não SQL”, mas diferentes
  - Object database
  - Multimodel

SQL DATABASES

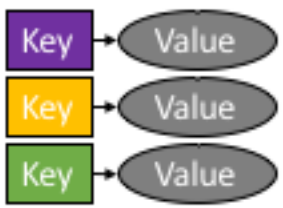


Relational

NoSQL DATABASES

Column

Graph

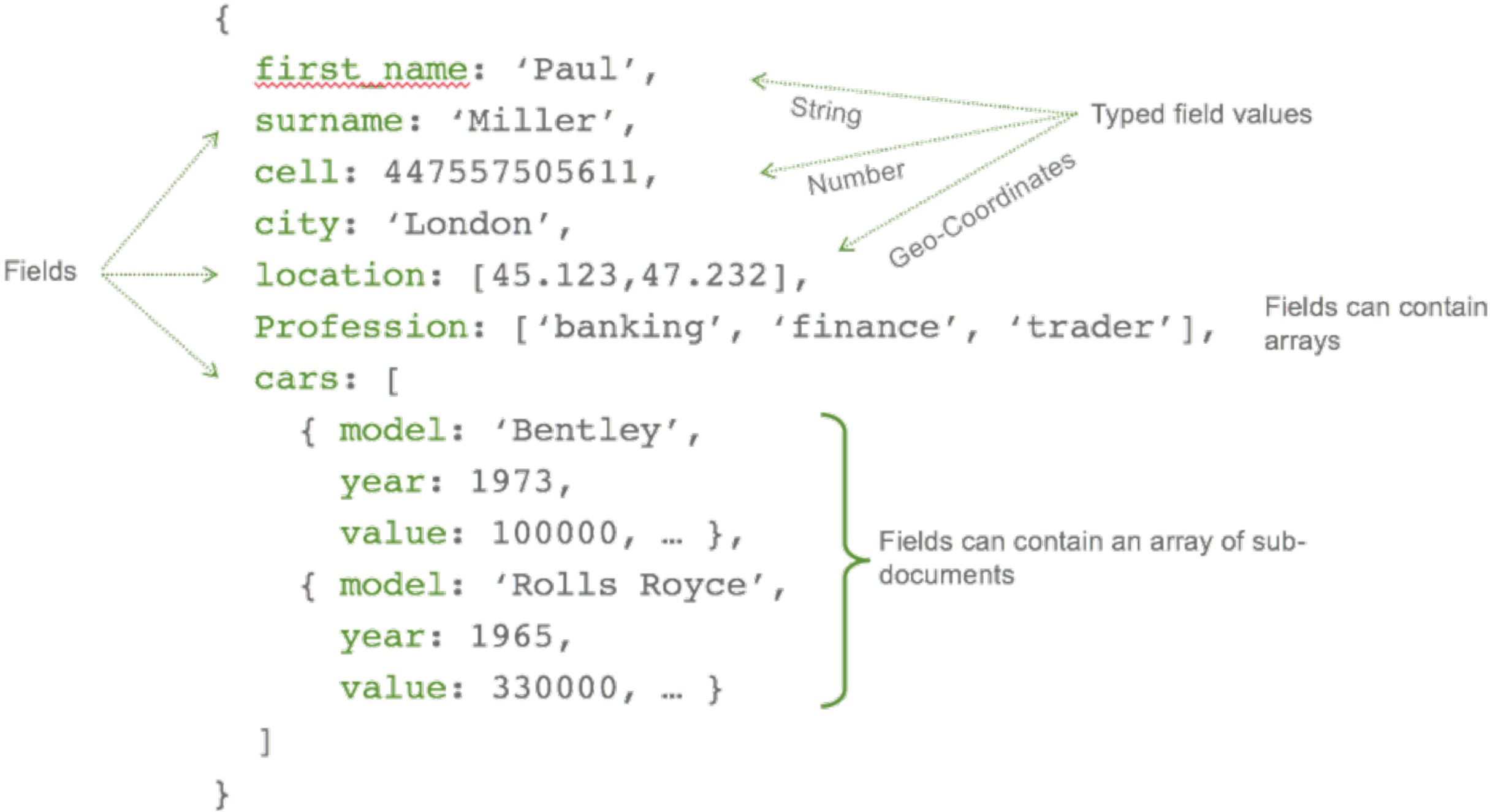


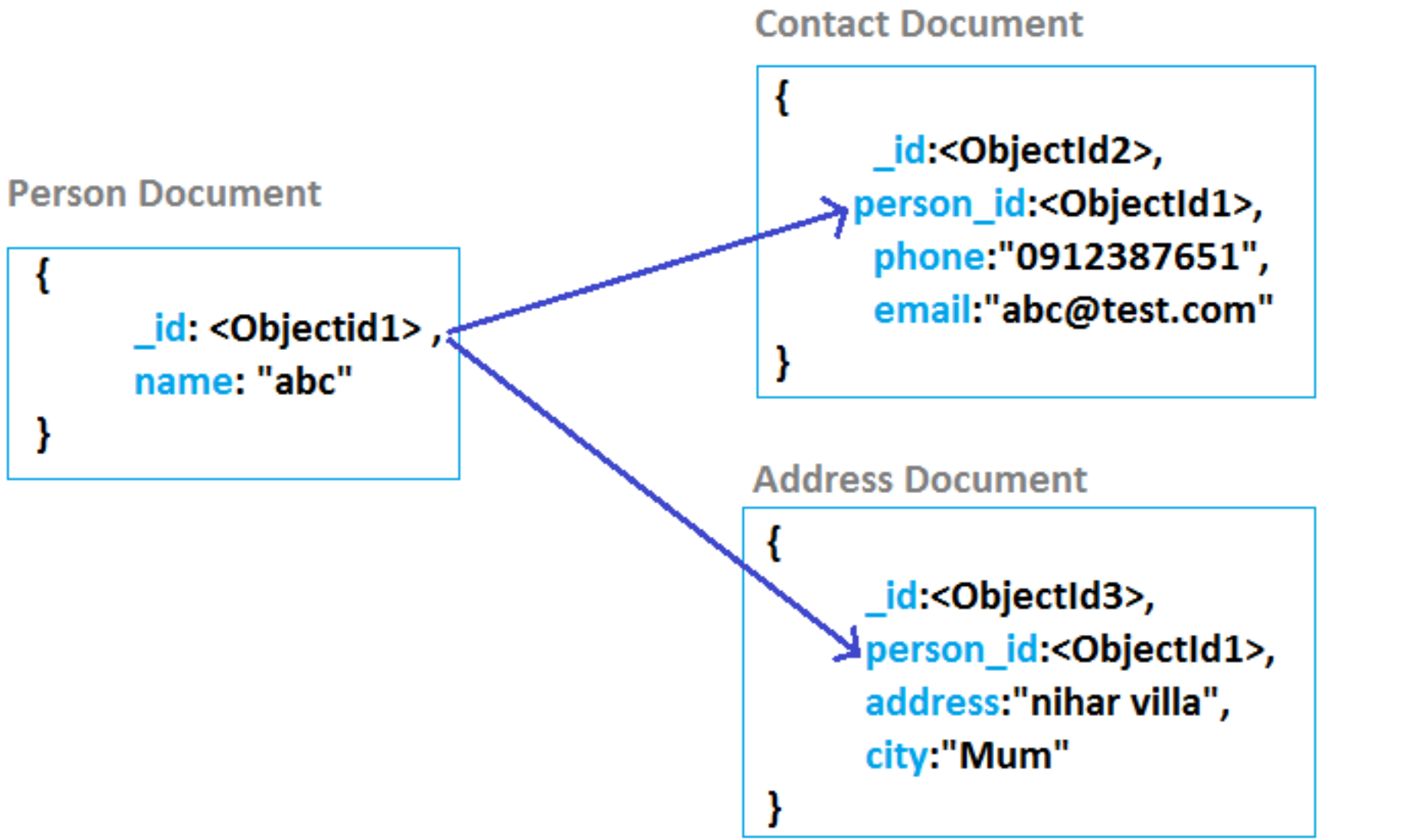
Key-Value

Document

Document store

- Modelo: **MongoDB**
- Em geral, document stores usam XML para definir os documentos
- MongoDB usa BSON (Binary JSON, ou BJSON)
- O BD é denominado **coleção**
- Cada **documento** na coleção tem um ObjectID, automaticamente indexado, fornecido pelo usuário ou gerado pelo sistema
- Componentes dos documentos são como atributos de uma tabela, porém desnormalizados: podem ter múltiplos valores, qualquer tipo, formato livre
  - Relacionamentos M:N podem ser representados com vetores internos
- **Não existe um esquema para a coleção**: valem os atributos encontrados em cada documento


{
first_name: 'Paul',
surname: 'Miller',
cell: 447557505611,
city: 'London',
location: [45.123,47.232],
Profession: ['banking', 'finance', 'trader'],
cars: [
{ model: 'Bentley',
year: 1973,
value: 100000, … },
{ model: 'Rolls Royce',
year: 1965,
value: 330000, … }
]
}
Fields
String
Number
Geo-Coordinates
Typed field values
Fields can contain arrays
Fields can contain an array of sub-documents



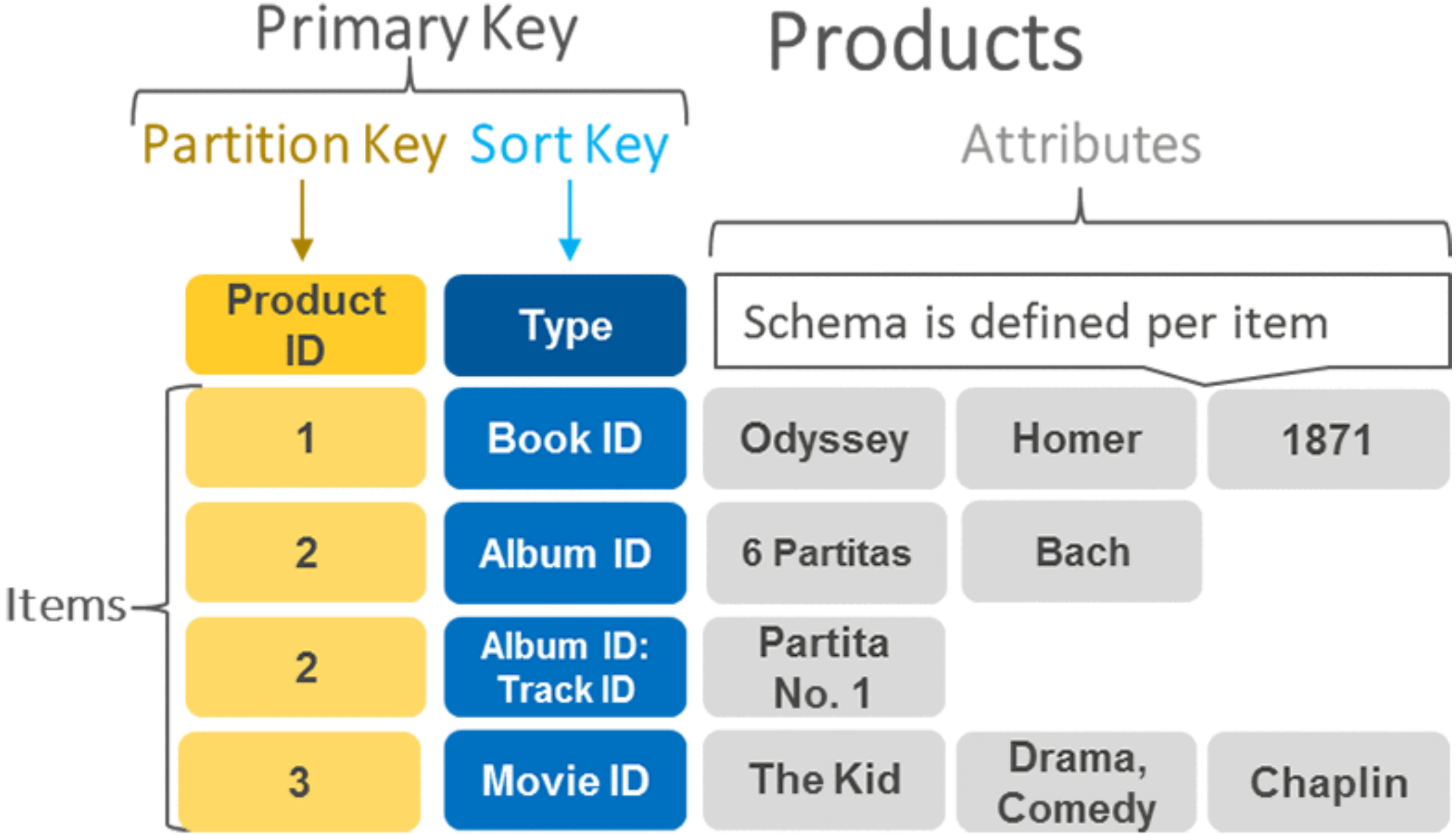
Outros Document Stores

- CouchDB
- RethinkDB

- Máquinas de busca como **Elasticsearch** e **Solr** se organizam de modo a poder usar um document store como banco de dados
- O PostgreSQL tem extensões para armazenamento de documentos usando Hstore

Key-value (chave-valor)

- Modelo: **DynamoDB**
- A cada chave corresponde um valor, que pode ser um dado simples, um dado estruturado (como uma tupla) ou um desestruturado (como um documento JSON/XML), ou ambos (semiestruturado)
- Tabelas no DynamoDB não têm esquema, mas uma coleção de **itens** autodescritivos
- Cada item consiste de diversos pares (atributo, valor), podendo ser multivalorados, associados a uma chave
  - Chave: atributo simples, usado para criar um hash de indexação
  - Chave: par de atributos hash e range type. O primeiro é a chave do hashing, o segundo é usado para ordenar registros com a mesma chave de hashing


Primary Key
Products
Partition Key
Sort Key
Attributes
Product ID
Type
Schema is defined per item
Items
1
Book ID
Odyssey
Homer
1871
2
Album ID
6 Partitas
Bach
2
Album ID: Track ID
Partita No. 1
3
Movie ID
The Kid
Drama, Comedy
Chaplin

Primary key			
Partition key: PK	Sort key: SK		
ORG#MICROSOFT	METADATA#MICROSOFT	OrgName	PlanType
		Microsoft	Enterprise
	USER#BILLGATES	UserName	UserType
		Bill Gates	Member
	USER#SATYANADELLA	UserName	UserType
		Satya Nadella	Admin
ORG#AMAZON	METADATA#AMAZON	OrgName	PlanType
		Amazon	Pro
	USER#JEFFBEZOS	UserName	UserType
		Jeff Bezos	Admin

Organization Item

User items

Item collection

DynamoDB

- Relacional x DynamoDB:
  - SGBDR: flexibilidade total nas consultas, mas as consultas podem ser caras computacionalmente e há limitações de escalabilidade
  - DynamoDB: consultas eficientes, mas limitadas quanto ao que pode ser feito com eficiência; fora do que está previsto no esquema, consultas podem ser caras e lentas
- Diferenças de projeto:
  - SGBDR: busca flexibilidade nos dados, sem se preocupar com detalhes de implementação dos sistemas e com desempenho. A otimização de consultas não condiciona a um tipo de projeto de esquema, mas a normalização ajuda.
  - DynamoDB: esquema projetado para tornar rápidas e baratas as principais consultas. Estruturas de dados internas ajustadas aos requisitos levantados.

Outros key-value

- Oracle NoSQL
- Riak
- Redis
- Berkeley DB
- Project Voldemort (open source do DynamoDB)

Wide column store

- Modelo: **Hbase**
- Usa tabelas, com linhas e colunas, mas armazenando coluna a coluna, em vez de linha a linha. “Wide” porque o conteúdo de uma coluna pode ser complexo, como um documento
- ***Sparse multidimensional distributed persistente stored map***, onde “map” é uma coleção de itens (chave, valor).
- A cada linha, o nome e o formato da coluna podem variar; é como se fosse um key-value bidimensional
- O esquema é flexível, pois novas colunas podem ser criadas e só preenchidas onde é necessário; permite desnormalização e tabelas esparsas; facilita o particionamento automático

Row ID	Customer	Product	Amount
101	John White	Chairs	$400.00
102	Jane Brown	Lamps	$500.00
103	Bill Green	Lamps	$150.00
104	Jack Black	Desk	$700.00
105	Jane Brown	Desk	$650.00
106	Bill Green	Desk	$900.00

- Num SGBDR, a linha é a unidade básica de recuperação dos dados; ao ler linhas inteiras, colunas desnecessárias são recuperadas.
- Operações de agregação, percorrendo várias linhas, são caras

- Num column-store, a leitura é feita a cada coluna, portanto pode-se ler apenas o que for relevante
- Operações de agregação são grandemente facilitadas
- Operações de leitura e gravação de linhas ficam mais lentas
- Acrescentar ou retirar colunas é mais fácil
- Compactar colunas esparsas ou com poucos valores distintos é mais fácil
- Facilita a implementação de OLAP (operações de análise de data warehouses)

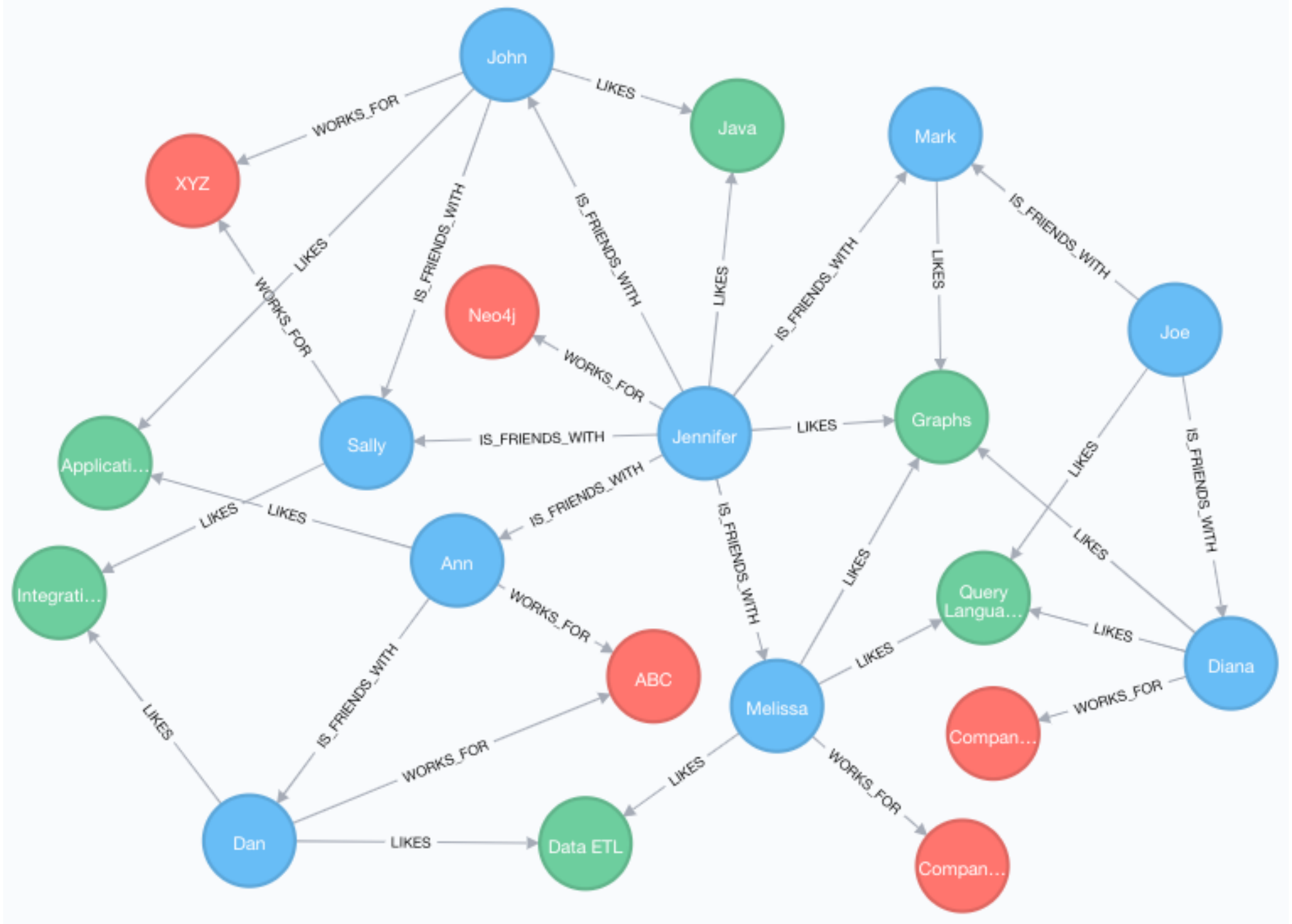
Outros column stores

- SciDB
- Google BigTable (Hbase é uma implementação open source do BigTable)

- Hbase é implementado sobre o HDFS, o sistema operacional distribuído que faz parte do projeto Hadoop
- Hbase não suporta SQL; consultas têm que ser escritas em Java, de forma semelhante a uma aplicação Map-Reduce
- O uso de SQL pode ser provido pelo Apache Phoenix

Graph database

- Modelo: **Neo4j**
- Dados representados comum grafo, uma coleção de vértices e arestas (nós e arcos) (No Neo4j, nós e relacionamentos)
- Tanto nós quanto arcos são rotulados para indicar o que representam, entidades ou relacionamentos
- Um nó pode ter zero, um ou vários labels, que correspondem a entidades
- Relacionamentos são direcionais, e recebem labels também (tipos)
- Paths são gerados para percorrer o grafo (BD), e são uma sequência de nós e relacionamentos que respondem a uma consulta
- Esquema é opcional, e índices podem ser criados para cada label


John
WORKS_FOR
XYZ
LIKES
Java
IS_FRIENDS_WITH
Mark
IS_FRIENDS_WITH
LIKES
LIKES
IS_FRIENDS_WITH
Neo4j
WORKS_FOR
Joe
WORKS_FOR
Graphs
Sally
IS_FRIENDS_WITH
Jennifer
LIKES
Applicati…
LIKES
LIKES
IS_FRIENDS_WITH
LIKES
IS_FRIENDS_WITH
Integrati…
Ann
Query
Langua…
LIKES
LIKES
Diana
WORKS_FOR
ABC
Melissa
LIKES
LIKES
IS_FRIENDS_WITH
Compan…
WORKS_FOR
LIKES
WORKS_FOR
Dan
LIKES
Data ETL
Compan…

Neo4j

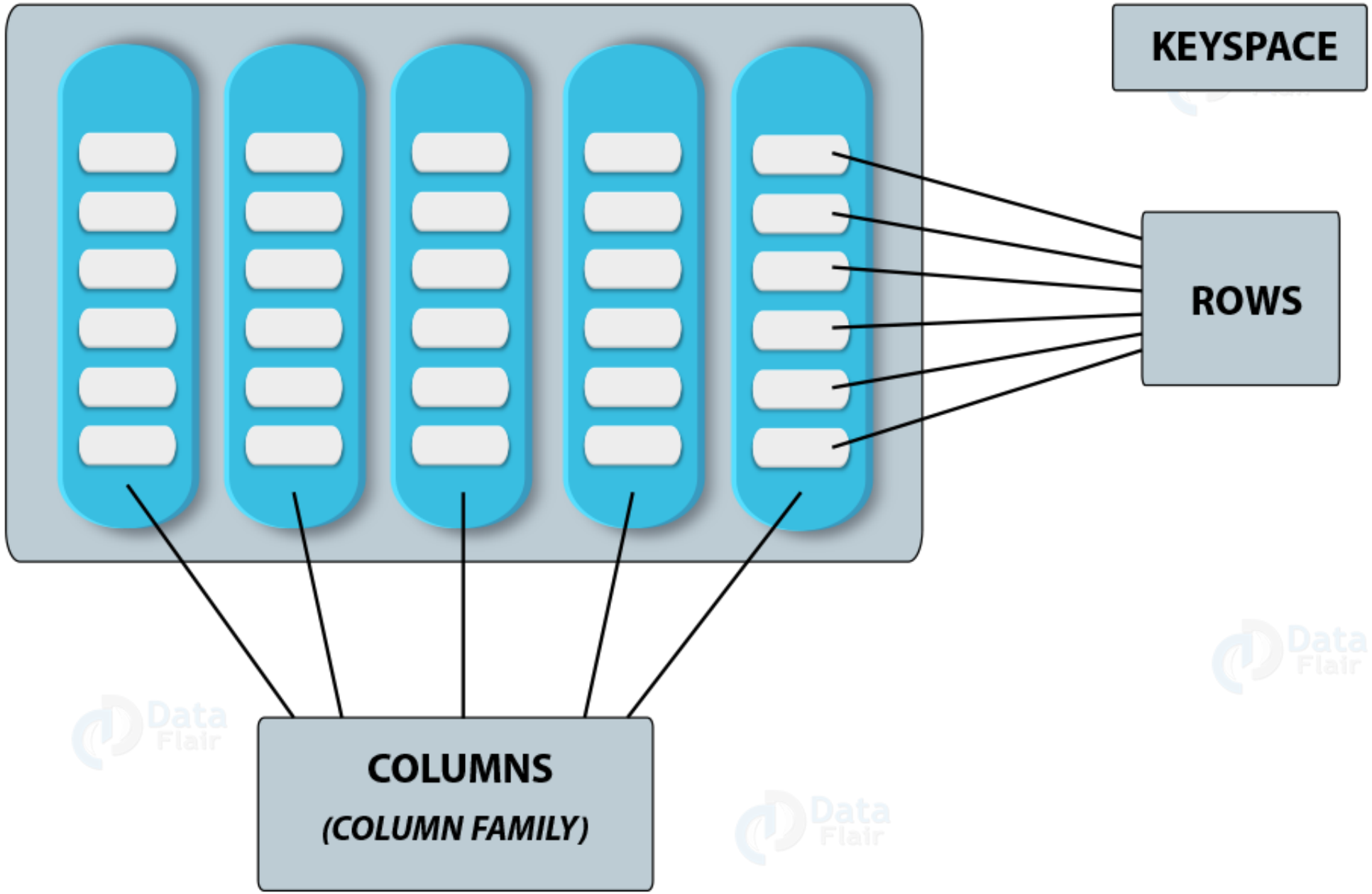
- Linguagens de consulta no grafo: Cypher e Gremlin
- Possui uma interface para visualização do grafo (BD)
- O banco pode ser replicado entre nós servidores master e slave
- Suporta ACID

Outros graph stores

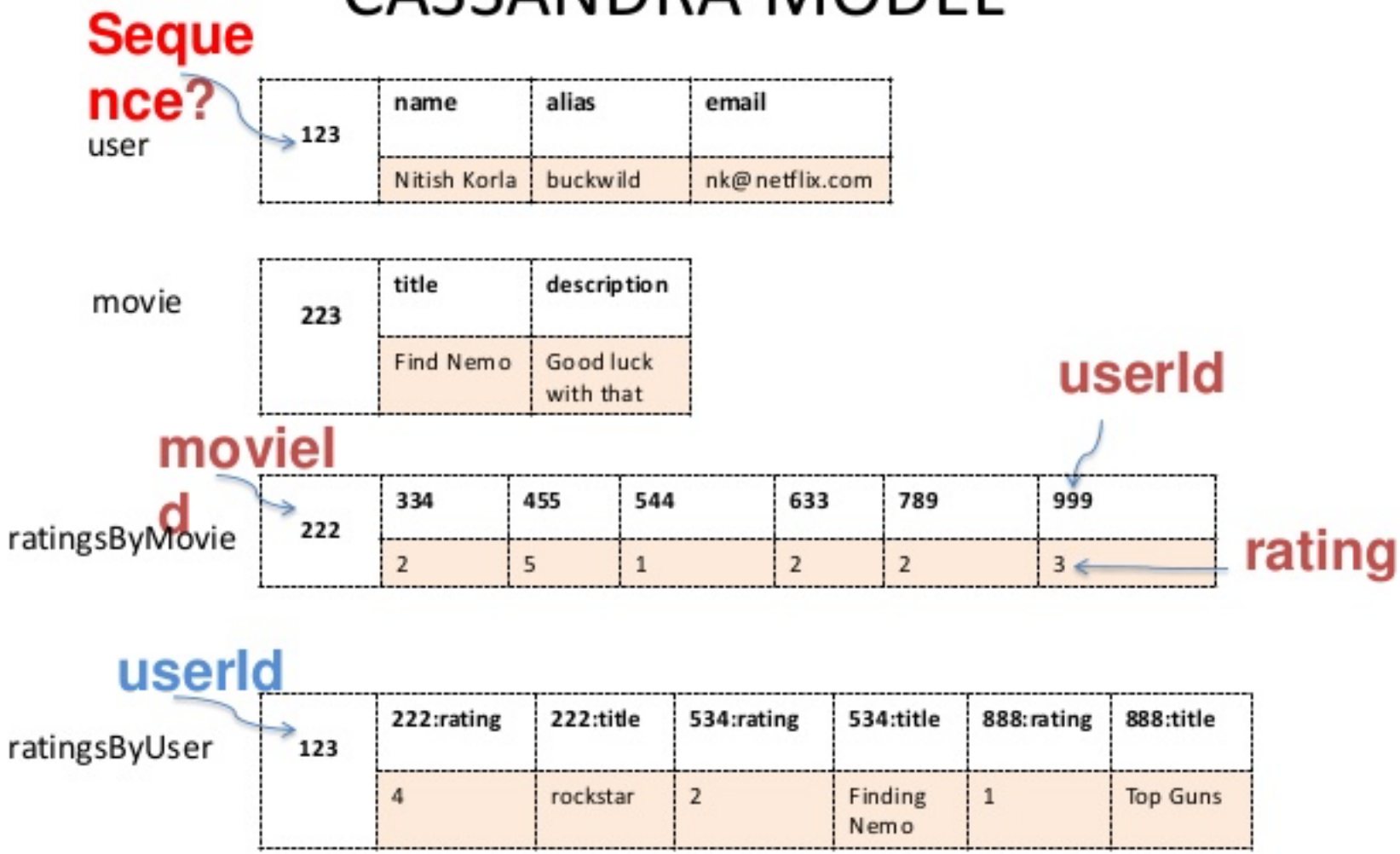
- OrientDB
- Amazon Neptune
- Oracle RDF Graph
- TigerGraph

Apache Cassandra

- O Cassandra não cabe em uma única categoria
- Basicamente um wide-column store, mas pelas características é um híbrido entre key-value e tabular
- Uma família de colunas é chamada de “tabela”
  - Cada linha tem um identificador, row key
  - Cada linha tem múltiplas colunas, cada uma com nome, valor e timestamp
  - Linhas diferentes na mesma família de colunas não compartilham o mesmo conjunto de colunas, e uma coluna pode ser acrescentada a uma ou mais linhas a qualquer tempo
- Não faz joins nem permite subconsultas; em consequência, a desnormalização é incentivada
- É a base do Facebook


KEYSPACE
ROWS
COLUMNS
(COLUMN FAMILY)

CASSANDRA MODEL



Apache Cassandra

- Cassandra suporta clusters de servidores espalhados em múltiplos lugares, com replicação assíncrona dos dados sem a necessidade de ter um “master”
- Como resultado, um BD distribuído de grande porte, com replicação, não teria um ponto único de falha, e teria alta disponibilidade
- Combina as estratégias de replicação do DynamoDB com o modelo de armazenamento do Google BigTable (ou Hbase)

NoSQL, ganhos de desempenho

- Não garantir ACID (ou seja, o SGBD presta menos serviços à aplicação)
- Desnormalização, evitando JOINs (flexibilidade de estrutura, mas maior complexidade para a aplicação)
  - Alguns JOINs podem nem ser possíveis em SGBD NoSQL, ou ser muito caros
- Prioridade para leitura em relação a atualizações (aplicações que fazem muito mais leituras que alterações nos dados)
- Distribuição massiva de dados em múltiplos nós de rede (*sharding*)
- Redundância de dados (replicação de nós, políticas de distribuição de carga)
- Estratégias Big Data: map/reduce, HDFS, etc.

SQL vs NoSQL

- Transações seguras e consistência sempre garantida
- Esquemas bem definidos
- Segurança de acesso

- Aplicações fortemente estruturadas

- ACID

- Desempenho e escalabilidade
- Flexibilidade
- Consistência eventual
- Replicação de dados

- Aplicações escaláveis sem exigência de consistência rígida

- CAP

NewSQL

- Classe de SGBDs que busca trazer estratégias de ganho de desempenho típicas de NoSQL para os relacionais, mantendo ACID e reforçando a capacidade de processamento de transações em ambiente distribuído
- Alguns recursos:
  - Novas arquiteturas internas; destaque para *shared-nothing nodes* em cluster, controle de concorrência distribuído, controle de fluxo, processamento distribuído de consultas
  - Mecanismos de armazenamento otimizados para SQL
  - *Transparent sharding*: partição automatizada de BDs através de múltiplos nós de rede

clodoveu@dcc.ufmg.br

Links

vCard