

Data Mining



Bases de Datos NoSQL



Contexto

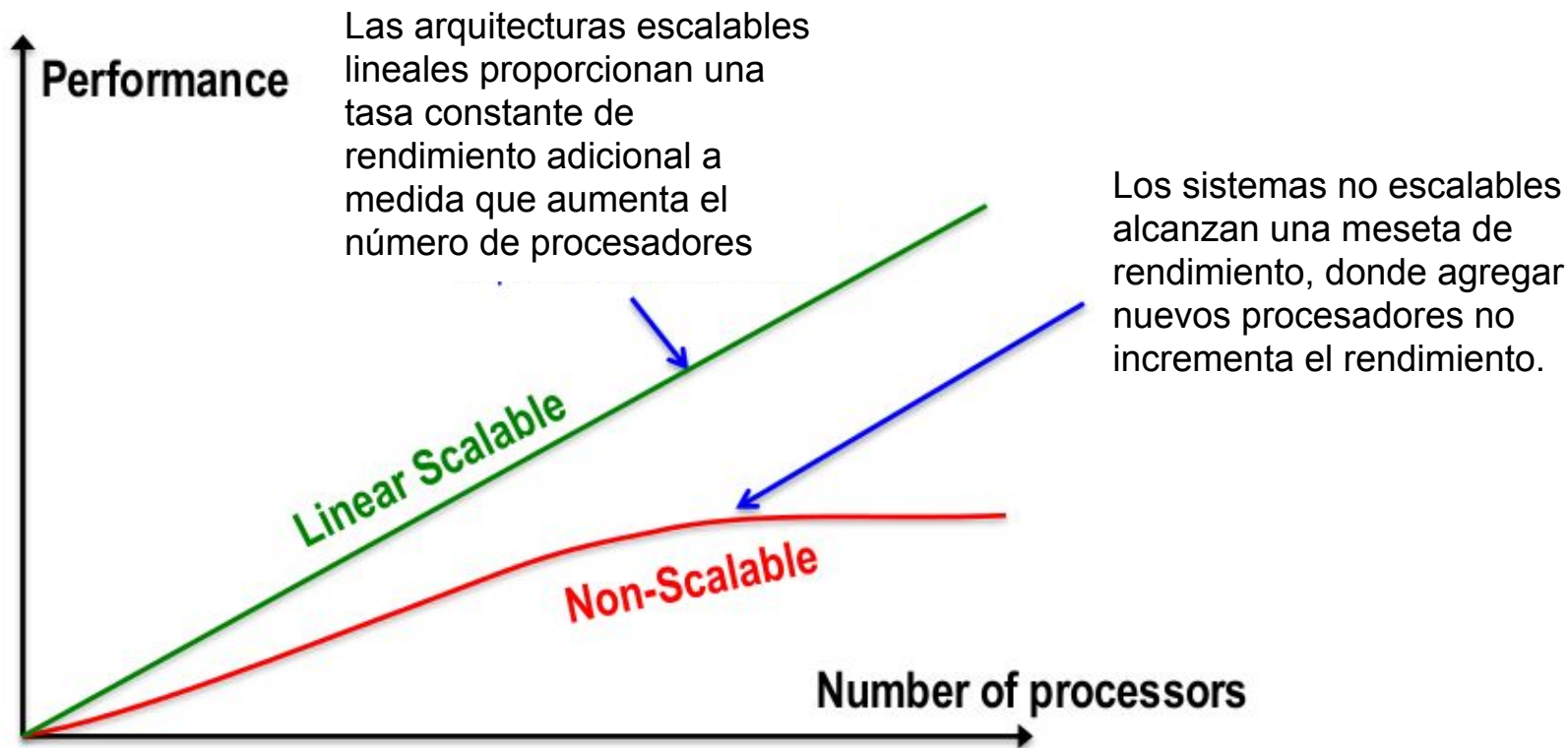


© Can Stock Photo - csp11851757



- ❑ La heterogeneidad de los datos que circulan a través de la web han llevado al límite a los RDBMS.
- ❑ Las soluciones generalmente vienen a partir de escalar las aplicaciones.

Escalabilidad Horizontal



¿Qué es NoSQL?

NoSQL es un conjunto de conceptos que permiten el procesamiento rápido y eficiente de conjuntos de datos poniendo el foco en la *performance*, confiabilidad y agilidad.

¿Qué es y qué no es NoSQL?

It's more than rows in tables.
It's free of joins
It's schema-free
It works on many processors
It uses shared-nothing commodity computer
It supports linear scalability
It's innovative

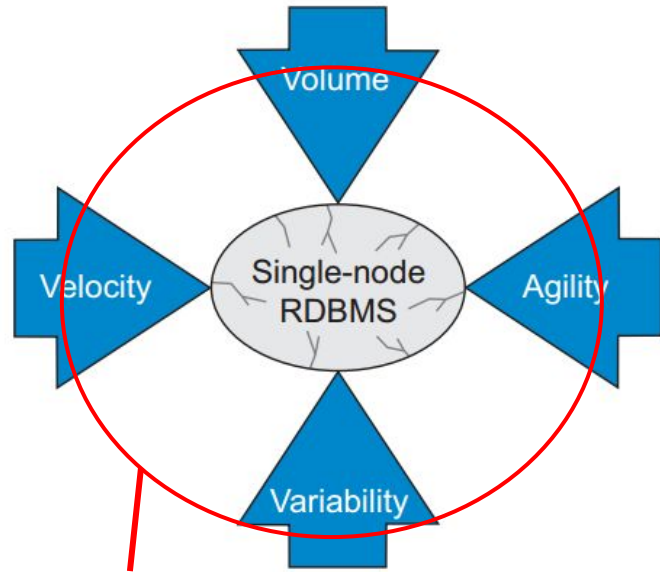
It's not about the SQL language
It's not only open source
It's not only big data
It's not about cloud computing
It's not about a clever use of RAM and SSD
It's not an elite group of products

MMM.. pero no
excluye los sistemas
SQL o RDBMS..

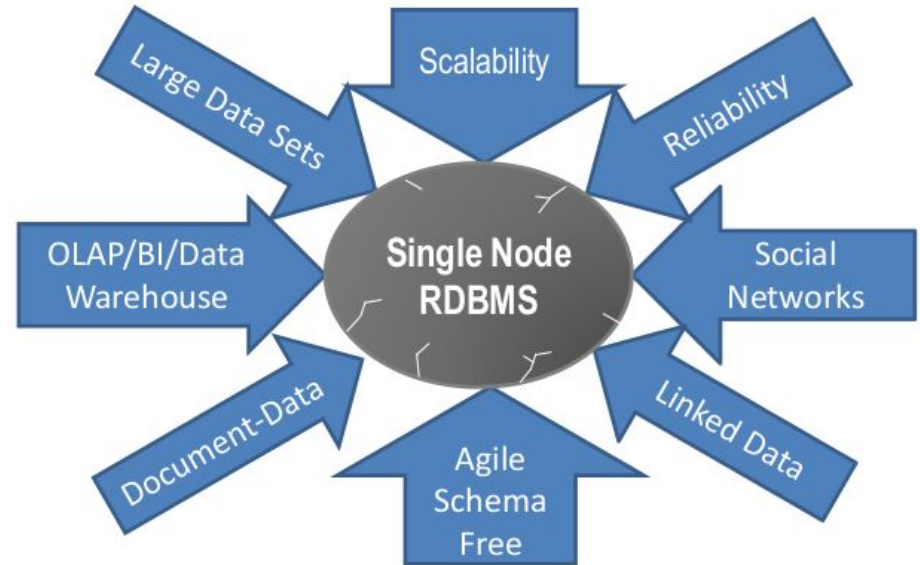


Presiones en una arquitectura RDBMS

Las empresas han encontrado valor en capturar y analizar rápidamente grandes cantidades de datos variables, y realizar cambios inmediatos en sus negocios en función de la información que reciben.



Business Drivers



Control de Transacciones

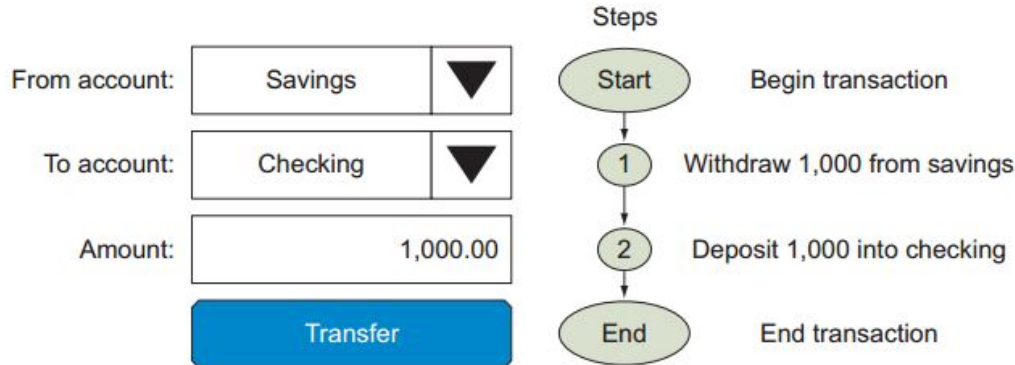
Control de transacciones es importante en entornos informáticos distribuidos con respecto al **rendimiento y la consistencia**

Modelos de control de transacciones:

- ❑ **ACID:** Utilizado en RDBMS
- ❑ **BASE:** Utilizado en muchos NoSQL

Principales diferencias radican en:

- ❑ Esfuerzo de desarrollo
- ❑ La ubicación de los controles de transacción



Para que la transacción se considere confiable, ambos pasos deben funcionar o ambos deben deshacerse.

RDBMS

BEGIN TRANSACTION ... END TRANSACTION

La transacción es responsabilidad del motor de bd.

Control de Transacciones: ACID

RDBMS controla las transacciones a través del uso de la atomicidad, consistencia, independencia y la persistencia.

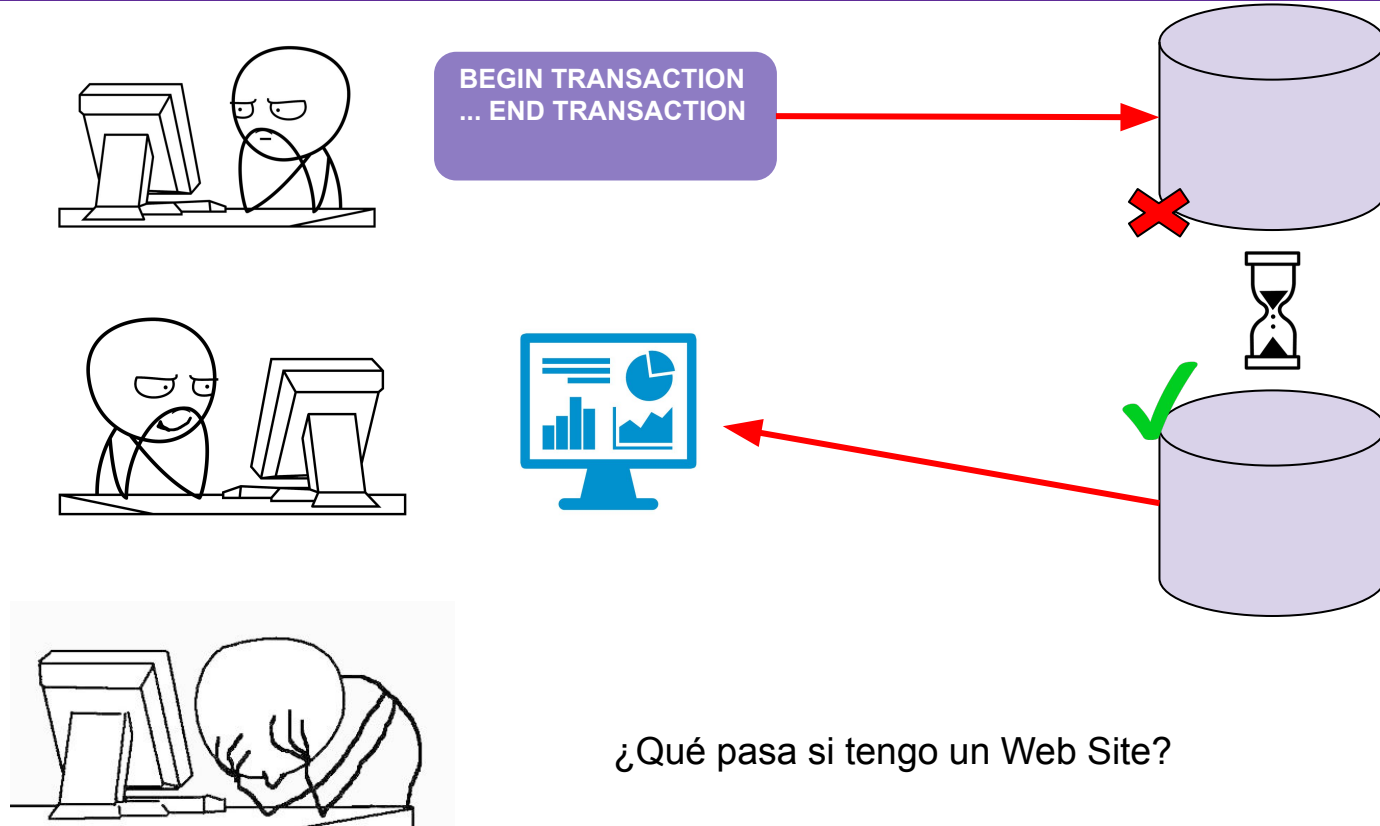
Esto sintetiza las propiedades de ACID.

ACID hace foco en la **consistencia** y en la **integridad** por sobre otros desafíos.

Utiliza estrategias de bloqueo de recursos

Atomicity Todo o Nada La transacción no se puede dividir	Consistency La transacción pasa la base de datos de un estado consistente a otro también consistente
Isolation Las transacciones se ejecutan de manera independiente.	Durability Los cambios en la BD son permanentes. Se persiste un estado consistente.

Situación: ACID



Sin Control de Transacciones: BASE

- ❑ Los websites de e-commerce con **carritos de compras** y **pago+envío** consideran de manera diferente el problema de las transacciones.
 - ❑ Mantenerse inconsistentes por unos minutos es menos importante que no poder tomar un pedido. **Bloquear un pedido** es perder un cliente.
- ❑ La alternativa a ACID es **BASE**
 - ❑ **Basic availability:** permite que los sistemas sean temporalmente inconsistentes para que las transacciones sean manejables.
 - ❑ **Soft-state:** permite algunas inconsistencias temporalmente y los datos pueden cambiar mientras se usan para reducir la cantidad de recursos consumidos
 - ❑ **Eventual consistency:** significa que cuando toda la lógica de servicio es ejecutada, el sistema alcanza un estado consistente.

BASE

- ❑ A diferencia de un RDBMS que se enfoca en la consistencia, **BASE se enfoca en la disponibilidad.**
- ❑ El Objetivo es permitir que se almacenen los nuevos datos, incluso a riesgo de no estar sincronizados durante un corto período de tiempo.

ACID vs BASE

Vs.

Acid

- Get transaction details right
- Block any reports while you are working
- Be pessimistic: anything might go wrong!
- Detailed testing and failure mode analysis
- Lots of locks and unlocks



Base

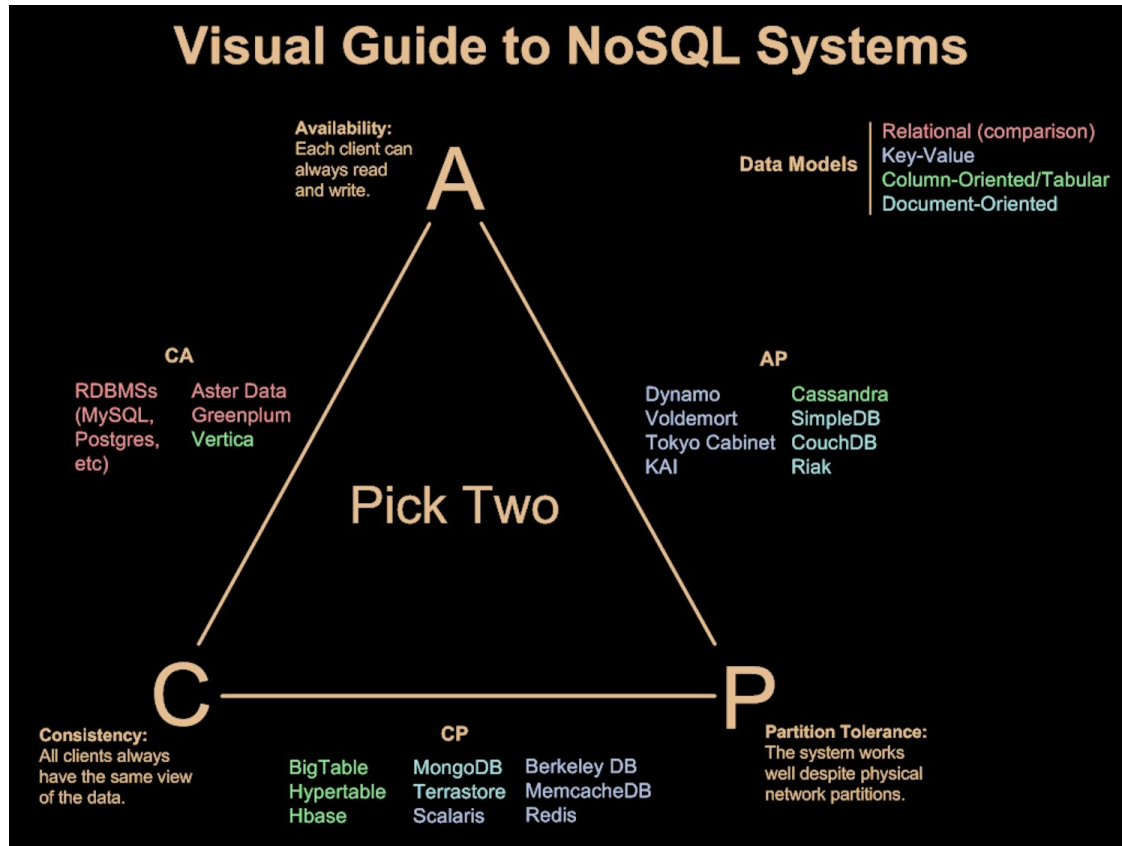
- Never block a write
- Focus on throughput, not consistency
- Be optimistic: if one service fails it will eventually get caught up
- Some reports may be inconsistent for a while, but don't worry
- Keep things simple and avoid locks



Teorema CAP

- ❑ Eric Brewer introdujo por primera vez el Teorema de CAP en 2000.
- ❑ El teorema de CAP establece que **cualquier sistema de base de datos distribuida** puede tener como máximo **dos de las siguientes tres propiedades** deseables:
 - ❑ **Consistency**: Tener una versión única, actualizada y legible de sus datos disponible para todos los clientes
 - ❑ **High availability**: La base de datos distribuida siempre permitirá a los clientes actualizar los elementos sin demora.
 - ❑ **Partition tolerance**: La capacidad del sistema para seguir respondiendo a las solicitudes de los clientes, incluso si hay una falla de comunicación entre las particiones de la base de datos.

Teorema CAP



Tipos de almacenamiento

NoSQL

Type

Typical usage

Examples

Key-value store—A simple data storage system that uses a key to access a value

- Image stores
- Key-based filesystems
- Object cache
- Systems designed to scale

- Berkeley DB
- Memcache
- Redis
- Riak
- DynamoDB

Column family store—A sparse matrix system that uses a row and a column as keys

- Web crawler results
- Big data problems that can relax consistency rules

- Apache HBase
- Apache Cassandra
- Hypertable
- Apache Accumulo

Graph store—For relationship-intensive problems

- Social networks
- Fraud detection
- Relationship-heavy data

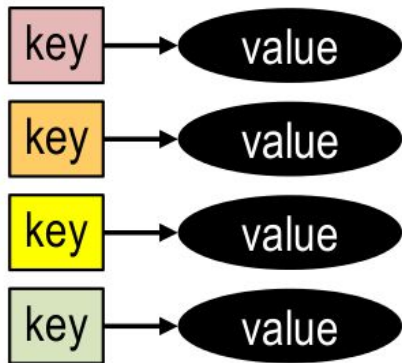
- Neo4j
- AllegroGraph
- Bigdata (RDF data store)
- InfiniteGraph (Objectivity)

Document store—Storing hierarchical data structures directly in the database

- High-variability data
- Document search
- Integration hubs
- Web content management
- Publishing

- MongoDB (10Gen)
- CouchDB
- Couchbase
- MarkLogic
- eXist-db
- Berkeley DB XML

Almacenamiento Key-Value



- Las claves son utilizadas para recuperar los valores
- Los valores pueden ser de cualquier tipo de dato. Por ejemplo: Imagen, video, etc.

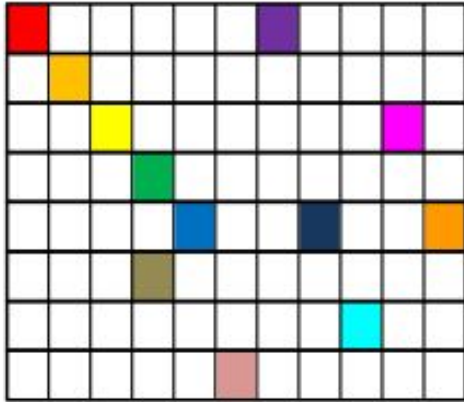
Ejemplos:

Berkley DB,
Memcache,
DynamoDB, S3,
Redis, Riak

PROS: escalable, simple API (put, get, delete)

CONT: no hay forma de consultar el contenido de los valores.

Column-Family



- Las claves incluyen una fila, familia de columnas y nombres de columnas.
 - Puede almacenar blobs versionados en una gran tabla.
 - Las consultas pueden ser realizadas en filas y familias de columnas y nombres de columnas
-
- **Pros:** Alta escalabilidad y disponibilidad
 - **Cons:** No es posible consultar un blob de datos, el diseño del esquema de filas y columnas es crítico.

Column store key-value



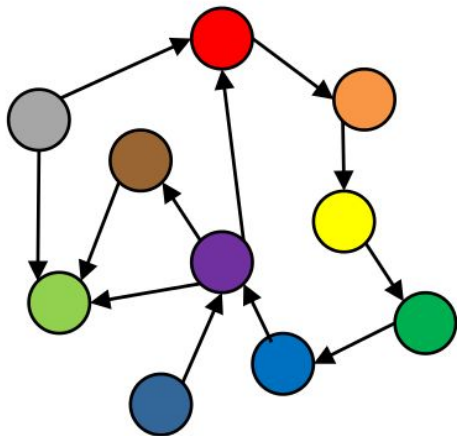
La clave está compuesta de:

- row id (string)
- Column family (grouping of columns)
- Column name (string)
- Timestamp (64-bit value)

Value

- un blob (byte stream)

Graph Store

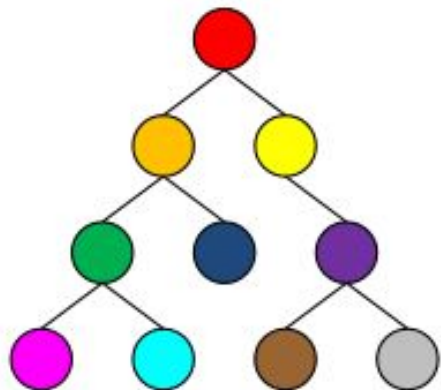


Ejemplos:

Neo4j, AllegroGraph,
Bigdata triple store,
InfiniteGraph,
StarDog

- Los datos son almacenados en series de nodos relaciones y propiedades
- Las consultas son totalmente transversales a la red
- Son ideales cuando la relación entre los datos es una clave:
 - Por ejemplo: Social networks.
- **Pros:** Permite realizar búsquedas rápidas en la red.
- **Cont:** Pobre escalabilidad cuando el grafo sobrepasa el tamaño de la red.

Almacenamiento de documentos



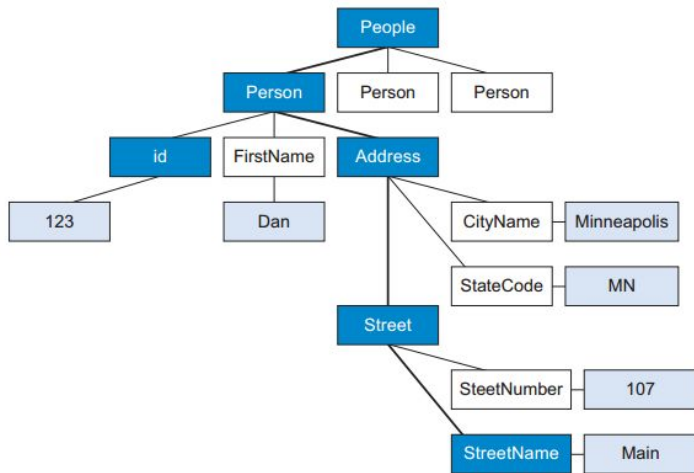
Ejemplos:

MarkLogic, MongoDB,
Couchbase, CouchDB,
RavenDB, eXist-db

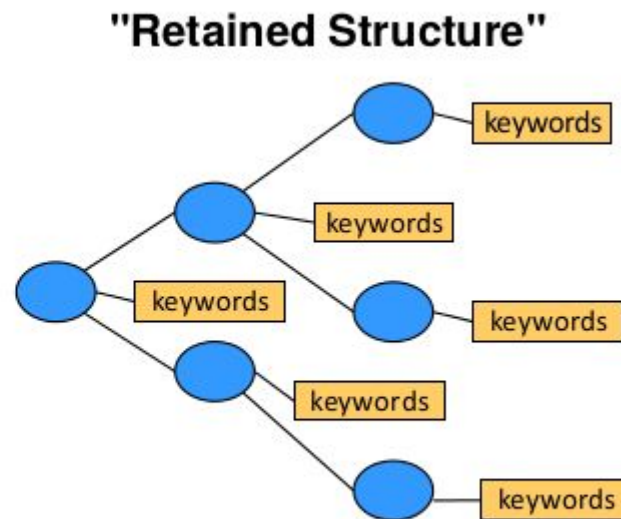
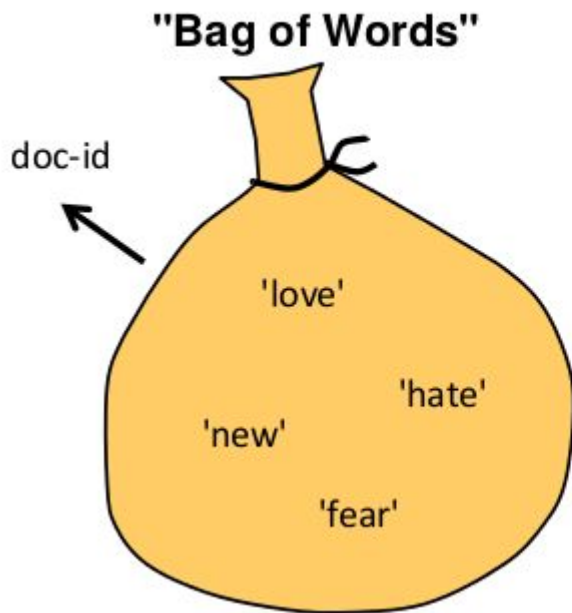
- ❑ Los datos son almacenados en jerarquías anidadas.
- ❑ La lógica de los datos queda almacenada como una unidad.
- ❑ Cualquier ítem en el documento puede ser consultado.

Pros: No existe una capa de mapeo de objetos

Cons: Incompatible con SQL, complejo de implementar



Modelos de NoSQL orientado a Documentos

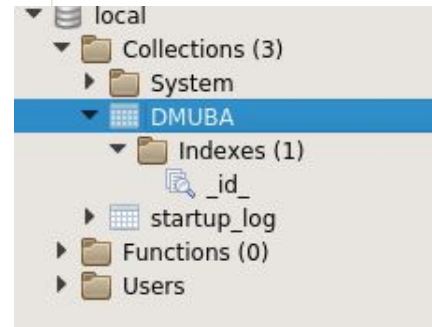


Ejemplo: Base de documentos de bibliografía

Documentos
JSON

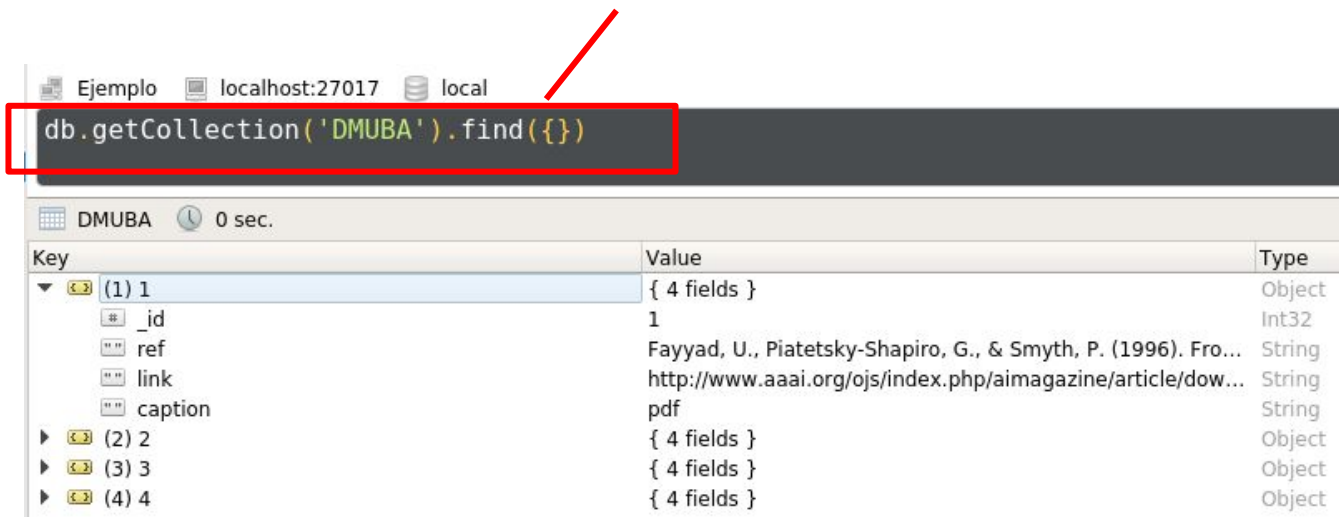
```
{
  "_id": 1,
  "ref": "Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. AI magazine",
  "link": "http://www.aaai.org/ojs/index.php/aimagazine/article/download/1230/1131/",
  "caption": "pdf"
},
{
  "_id": 2,
  "ref": "Jiawei Han, Micheline Kamber, Jian Pei. 2012. Tercera edición. Data Mining: Concepts and Techniques.",
  "link": "",
  "caption": "pdf"
},
{
  "_id": 3,
  "ref": "Daniel T. Larose. 2014. Segunda edición. Discovering Knowledge in Data: An Introduction to Data Mining.",
  "link": "",
  "caption": "pdf"
},
{
  "_id": 4,
  "ref": "Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). Mining of massive datasets. Cambridge university press.",
  "link": "http://infolab.stanford.edu/~ullman/mmds/book.pdf",
  "caption": "pdf"
},

```



Consultar los documentos

No hay un lenguaje de consultas estándar como SQL



The screenshot shows the MongoDB interface. At the top, the query `db.getCollection('DMUBA').find({})` is entered in the command bar and highlighted with a red rectangle. Below the command bar, the database 'DMUBA' is selected, and the execution time is '0 sec.'. The results are displayed in a table with three columns: 'Key', 'Value', and 'Type'.

Key	Value	Type
(1) 1	{ 4 fields }	Object
# _id	1	Int32
" ref	Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). Fro...	String
" link	http://www.aaai.org/ojs/index.php/aimagazine/article/dow...	String
" caption	pdf	String
▶ (2) 2	{ 4 fields }	Object
▶ (3) 3	{ 4 fields }	Object
▶ (4) 4	{ 4 fields }	Object

Recupero TODOS los documentos

Consultar los documentos

Ejemplo localhost:27017 local

```
db.getCollection('DMUBA').find({"ref" : {$regex : ".*2012.*"}});
```

DMUBA 0.001 sec.

Key	Value	Type
▼ (1) 2	{ 4 fields }	Object
# _id	2	Int32
ref	Jiawei Han, Micheline Kamber, Jian Pei. 2012. Tercera edic...	String
link		String
caption	pdf	String

Recupero solo documentos que contengan la secuencia **2012**

Casos de Estudio

LiveJournal's Memcache

- ❑ Need to increase performance of database queries
- ❑ By using hashing and caching, data in RAM can be shared. This cuts down the number of read requests sent to the database, increasing performance.

Google's MapReduce

- ❑ Need to index billions of web pages for search using low-cost hardware.
- ❑ By using parallel processing, indexing billions of web pages can be done quickly with a large number of commodity processors.

Google's Bigtable

- ❑ Need to flexibly store tabular data in a distributed system.
- ❑ By using a **sparse matrix approach**, users can think of all data as being stored in a single table with billions of rows and millions of columns without the need for up-front data modeling.

Bibliografía

- ❑ McCreary, D., & Kelly, A. (2013). Making Sense of NoSQL: A guide for managers and the rest of us.
- ❑ McCreary, D., & Kelly, A. (2014). Making sense of NoSQL. Shelter Island: Manning, 19-20. [[Slides](#)]