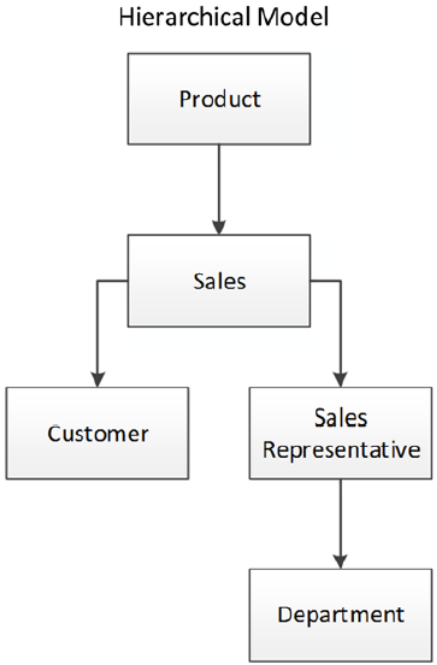
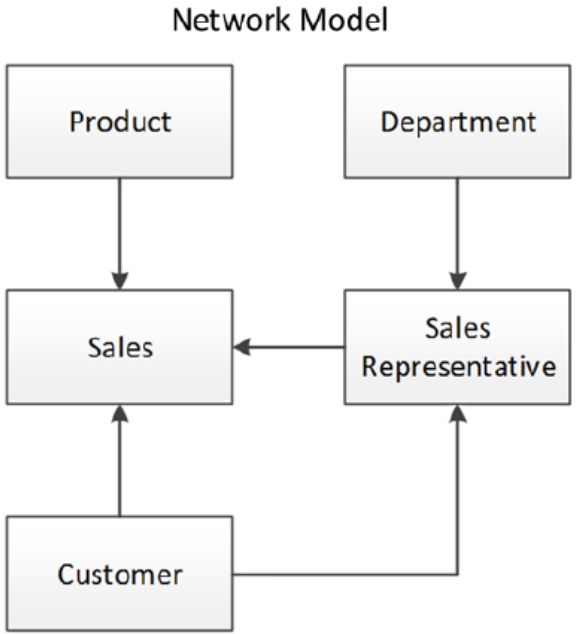
Tema 2. Sistemas de almacenamiento Big Data

viu | Universidad Internacional de Valencia

25.04.23

Un poco de historia…
Primera revolución de las bases de datos





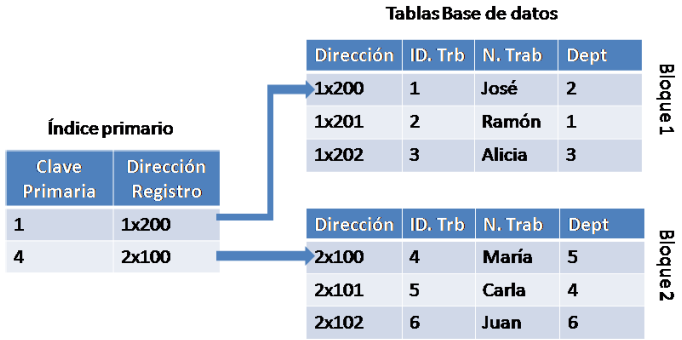
Imágenes de **Next Generation Databases, Guy Harrison**

Primera revolución de las bases de datos

- **Sin embargo los modelos anteriormente mencionados tenían defectos que fueron detectados ya a finales de los 60.**
- **Las bases de datos eran complicadas de usar para personas sin conocimientos de programación.**
- **No tenían fundamento teórico.**
- **Mezclaban la implementación lógica y la física.**
- **Solución: Modelos relacionales**

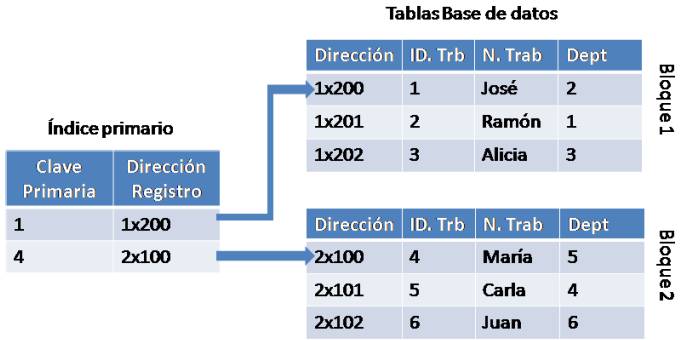
Segunda revolución de las bases de datos

- Los modelos relacionales describen como los datos deben ser presentados al usuario.
- Está compuesto de elementos como las tuplas/entidades, atributos de estos, relacionales, restricciones…
- Cada fila debe ser identificable de forma única.
- Las tablas están relacionadas entre sí, pudiendo realizar JOINS en consultas para poder extraer información de diferentes tablas.
- Se fomenta la normalización



Segunda revolución de las bases de datos

- Este tipo de base de datos han sido utilizadas de forma casi exclusiva hasta los años 2000 y de forma mayoritaria hasta la actualidad.
- Sin embargo a mediados de los 2000 empezaron a surgir problemas:
  - Crecimiento de datos a administrar
  - Internet
  - Problemas de rendimiento
- Necesidad de alternativas



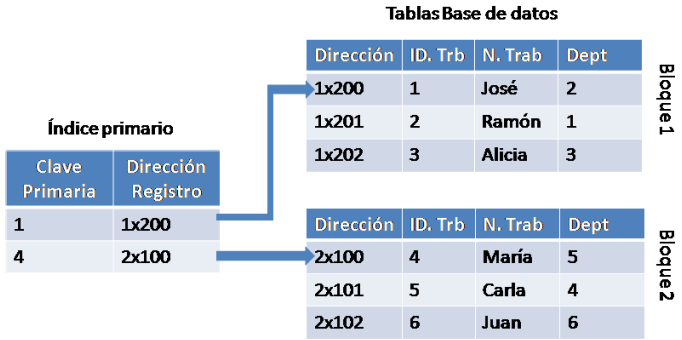
Estado del arte de las bases de datos en los últimos 50 años

- **Utilización de las bases de datos relacionales**
- **Propuestas en 1970 por Edgar Frank Codd (¡hace casi 50 años!)**
- **Modelo de datos basándonos en los datos**
- **Usualmente disponen de un modelo normalizado, pues es el modelo que mejor exprime sus ventajas**
- **Las tablas están relacionadas entre sí, pudiendo realizar JOINS en consultas para poder extraer información de diferentes tablas.**
- **A través de las restricciones de integridad, nos permite asegurar la integridad de los datos siempre que realizamos una consulta**
- **Plenamente establecidas, prácticamente todo el mundo las conoce y sabe usarlas.**

¿Por qué buscamos alternativas?

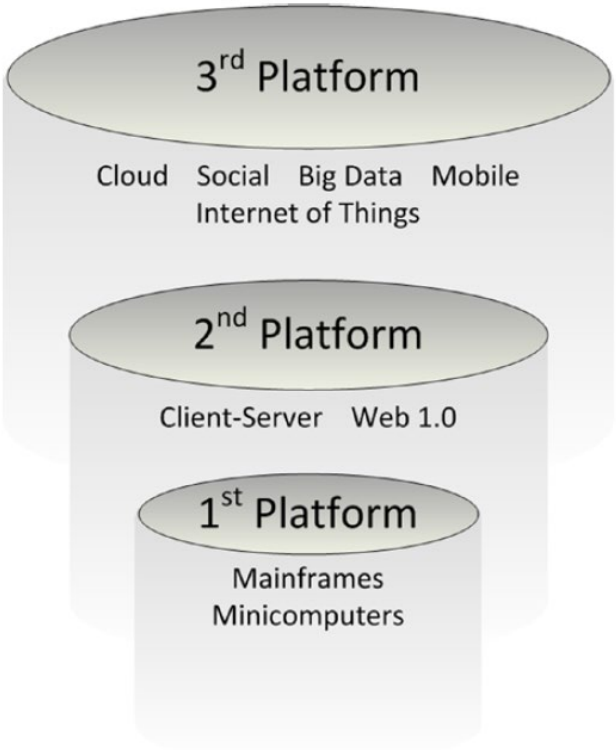
Tercera revolución de las bases de datos

- Este tipo de base de datos han sido utilizadas de forma casi exclusiva hasta los años 2000 y de forma mayoritaria hasta la actualidad.
- Sin embargo a mediados de los 2000 empezaron a surgir problemas:
  - Crecimiento de datos a administrar
  - Internet
  - Problemas de rendimiento
- Necesidad de alternativas



- Aparición de tecnologías para mejorar el procesamiento de datos.
- Google libera el algoritmo MapReduce, dando lugar al proyecto Hadoop.
- Se comienza a utilizar la replicación de bases de datos a través del “Sharding”.
- Aceleración del cloud computing.
- Alternativas a las bases de datos relacionales tradicionales: NoSQL y NewSQL.

Plataformas por revolución



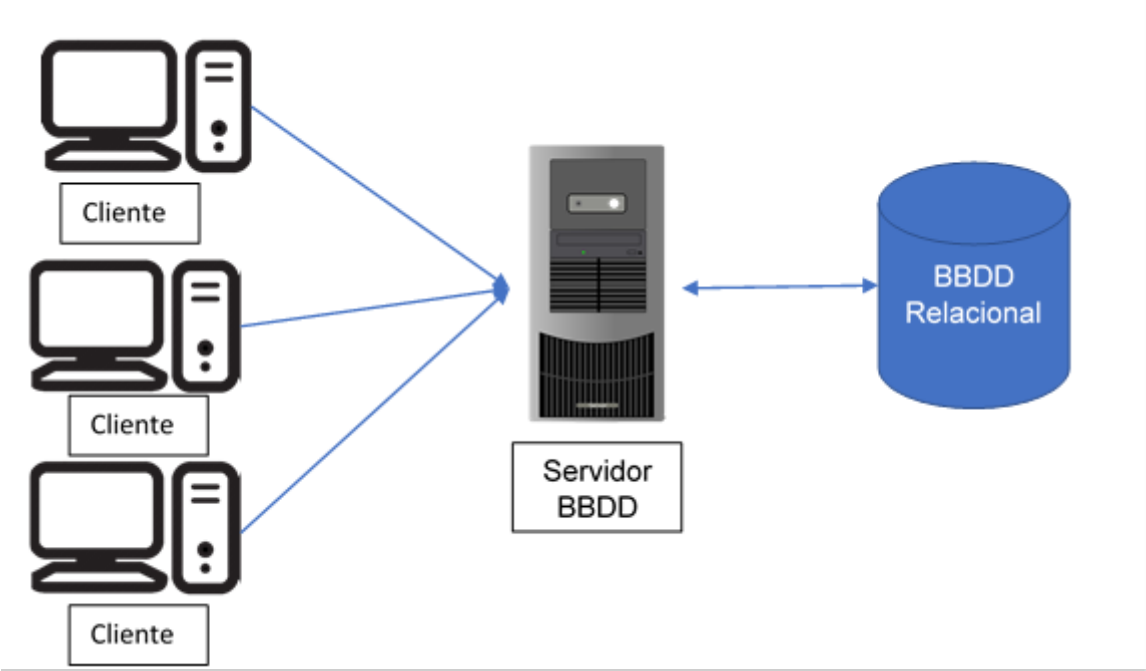
Relational Database
NoSQL
NewSQL
Big Data platforms

Relational Database

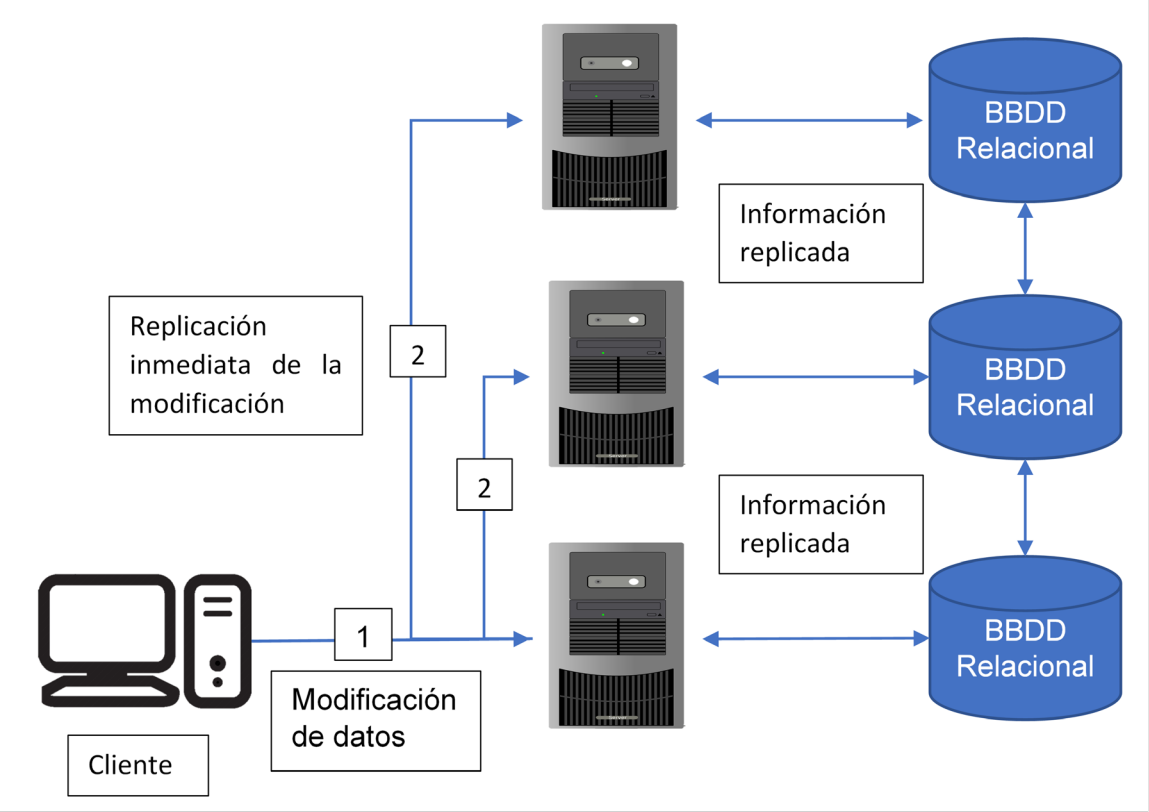
Hierarchical Database
Network Database
ISAM files

Imágenes de **Next Generation Databases, Guy Harrison**

Arquitectura BBDD relacional típica

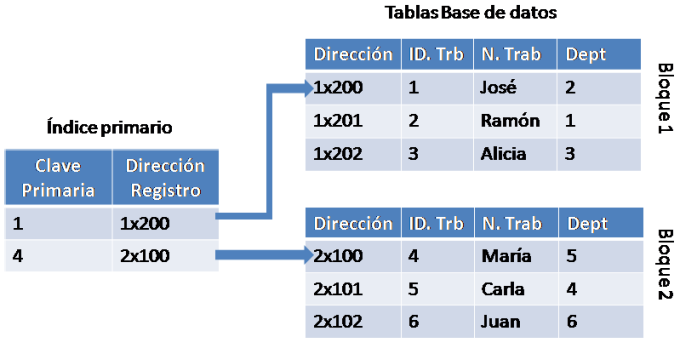


Arquitectura BBDD relacional con replicación



Cuando usar una BD relacional tradicional

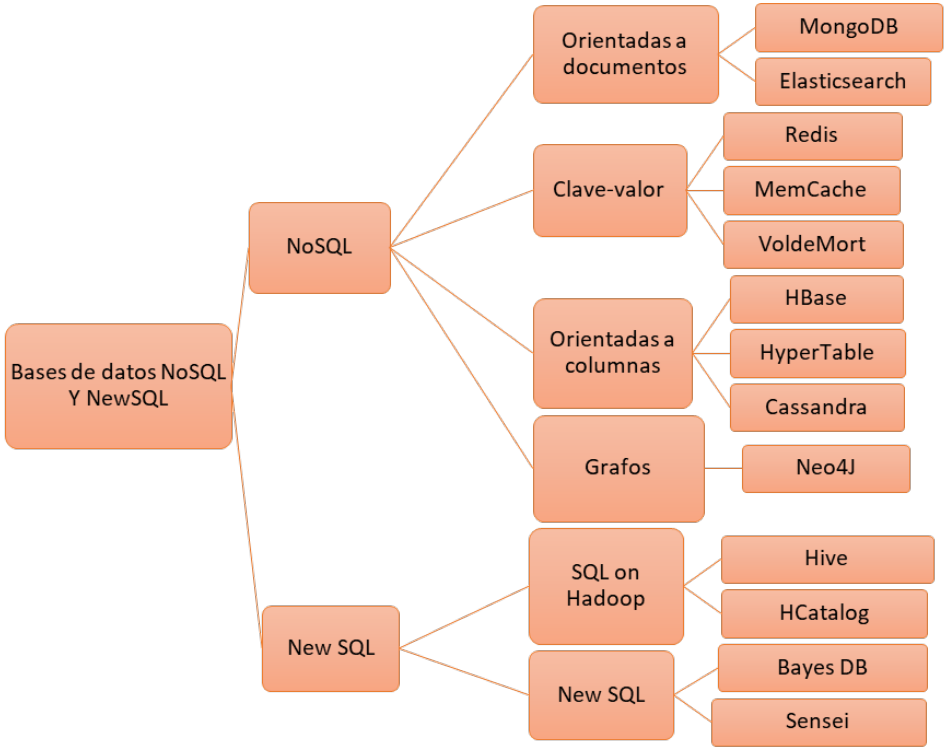
- Sin necesidad de paralelización.
- Sin necesidad de almacenamiento de big data
- Necesidad de tener datos consistentes
- Cuando se tienen que almacenar datos estructurados
- Datos relacionados entre sí.
- Necesidad de almacenar información muy crítica



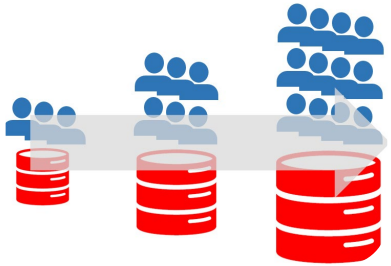
- Sin poder escalar de forma horizontal.
- Imposibilidad de tener Sistema distribuido, solo replicación.
- Costes elevados si se quiere manejar Big Data.
- Problemas de rendimiento
- Datos orientativos de 100.000 operaciones en milisegundos

Operación	Oracle	MySQL	MsSQL	MongoDB	Redis	GraphQL	Cassandra
INSERT	0.076	0.093	0.93	0.005	0.009	0.008	0.011
UPDATE	0.077	0.058	0.073	0.008	0.013	0.007	0.013
DELETE	0.059	0.025	0.093	0.01	0.021	0.017	0.018
SELECT	0.025	0.093	0.062	0.009	0.016	0.010	0.014

Fuente: Comparison of query performance in relational a non-relation databases. (Čerešňák et el, 2019)
Experimentos realizados en MacBook pro del 2015 con 8gb de ram y procesador I5

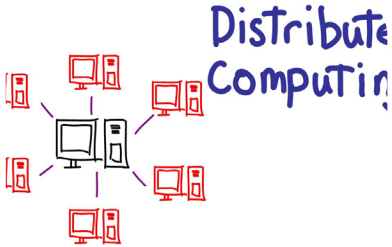

Bases de datos NoSQL Y NewSQL
NoSQL
New SQL
Orientadas a documentos
MongoDB
Elasticsearch
Clave-valor
Redis
MemCache
VoldeMort
Orientadas a columnas
HBase
HyperTable
Cassandra
Grafos
Neo4J
SQL on Hadoop
Hive
HCatalog
New SQL
Bayes DB
Sensei

Algunas ventajas generales (o casi) de sistemas destinados a Big Data



Escalabilidad

Flexibilidad

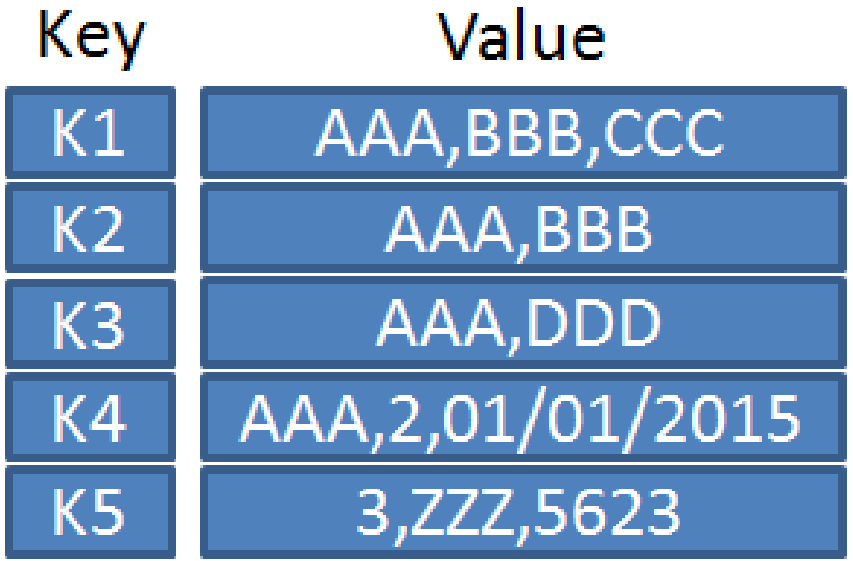


Distribución

Disponibilidad

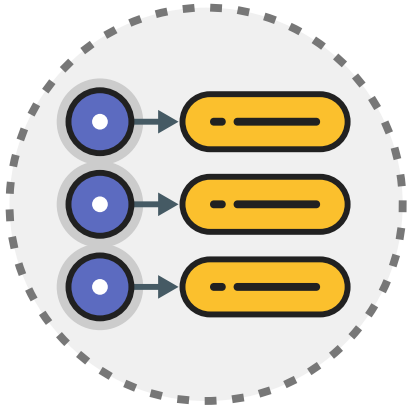
- Satisfacen lo que las relacionales tradicionales no son capaces en Big Data.
- Se clasifican en:
  - Columnas
  - Documentos
  - Grafos
  - Clave-valor
- Cada una es ideal para un tipo de problema en concreto.
- Desde sin esquema a un esquema cercano en solidez a las relacionales

- Sin esquema
- Una clave puede estar asociada a varios valores
- Ejemplos: Redis, Voldemort, Riak



Cuando usar clave valor

- Esquema simple
- Alta velocidad de escritura/lectura y pocas actualizaciones
- Alto rendimiento y escalabilidad
- Poca complejidad necesaria para las consultas

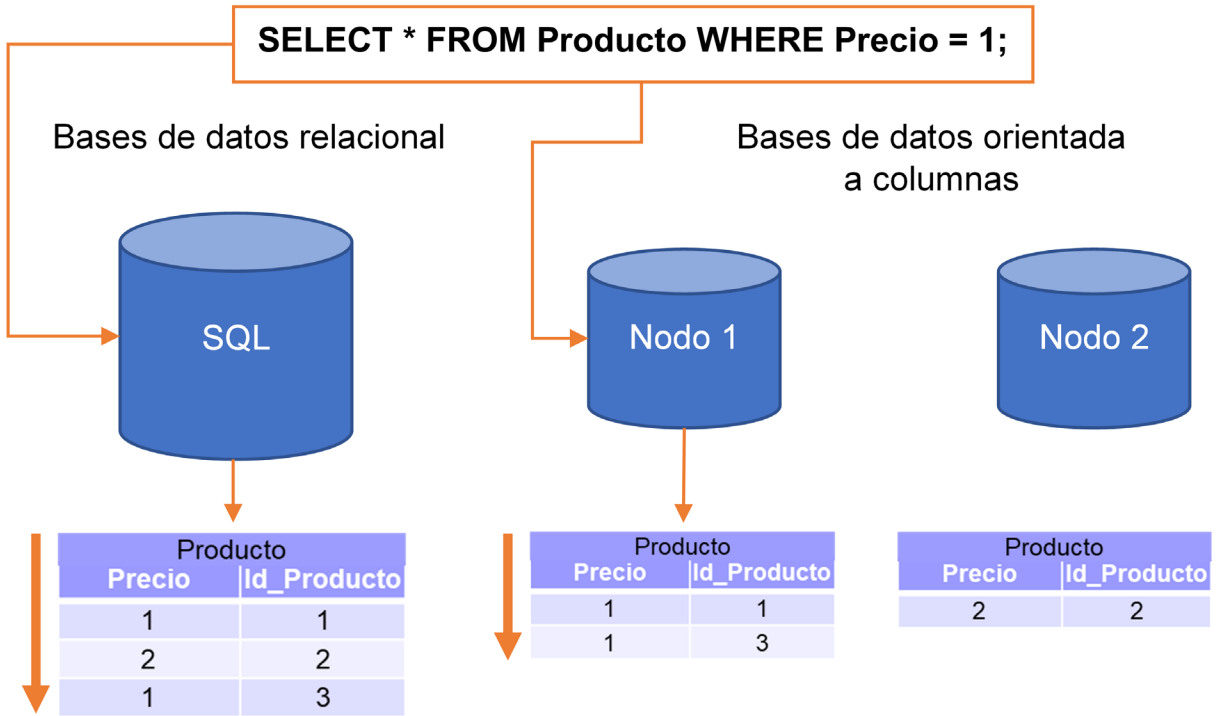


Orientadas a columnas

- Estructura similar en un principio a una base de datos relacional:
  - Tablas
  - Columnas y filas
  - Columnas que pueden ser PK
- Diferencia en el tratamiento de la información
- No hay relaciones entre tablas

id	name	state	birth_date
7b976c48...	Bill Watterson	DC	1953
7c8f33e2...	Howard Tayler	UT	1968
7d2a3630...	Randall Monroe	PA	
7da30d76...	Dave Kellett	CA	

Orientadas a columnas: Procesamiento de datos



Cuando usar una BD orientada a columnas

- Alto volumen de datos
- Altas velocidades de escritura pero no tan exigentes para las consultas
- Datos relativamente estructurados
- Formato de extracción basado en filas y columnas
- Se debe tener una idea ya establecida de las consultas a realizar en la BD

Orientadas a documentos

- Estructura de documentos
- Diseño más natural que una relacional.
- No necesaria la normalización, aunque posible.
- Posibilidad de incluir subdocumentos en los documentos

Casos de uso

- **Administración de contenido**
  - **Blogs, plataformas de vídeo, periódicos…**
  - **Una entidad, un document**
  - **Favorecido por no tener un esquema fuerte**
- **Catálogos**
  - **Tiendas online.**
  - **Cada producto tiene diferentes atributos.**
  - **Aumento de la velocidad**

```
{
  _id: <ObjectId1>,
  username: "123xyz",
  contact: {
             phone: "123-456-7890",
             email: "xyz@example.com"
           },
  access: {
            level: 5,
            group: "dev"
          }
}
```

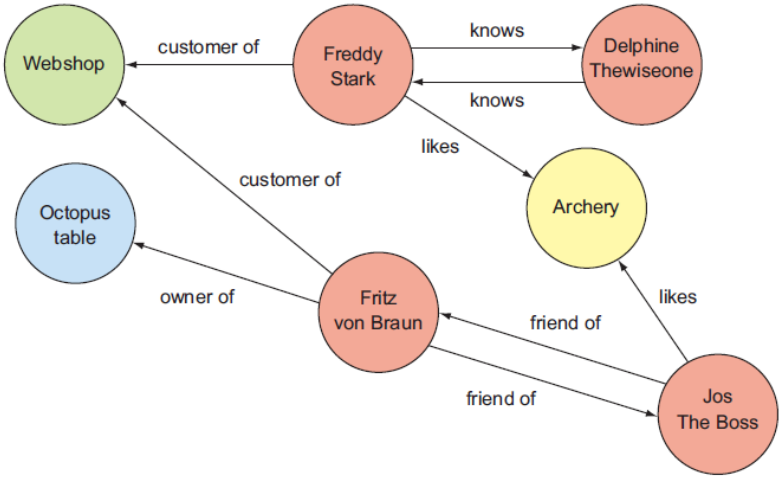
Embedded sub-document

Embedded sub-document

Cuando usar una BD orientada a documento

- Esquema flexible (no lo mismo que sin esquema)
- Si se usan datos estructurados en formatos JSON/BSON o XML
- Cuando se tienen índices complejos (multiclave, geoespacial…)
- Alto rendimiento y un ratio de escritura/lectura balanceado

- Ideal para datos fuertemente relacionados.
- Compuesto de:
  - Nodo:
  - Arista
- Satisfacen ACID
- Normalizada
- Algunas investigaciones sostienen que su rendimiento es muy similar al de las bases de datos relacionales



Imágenes Introducing Data Science**, Davy Cielen et al**

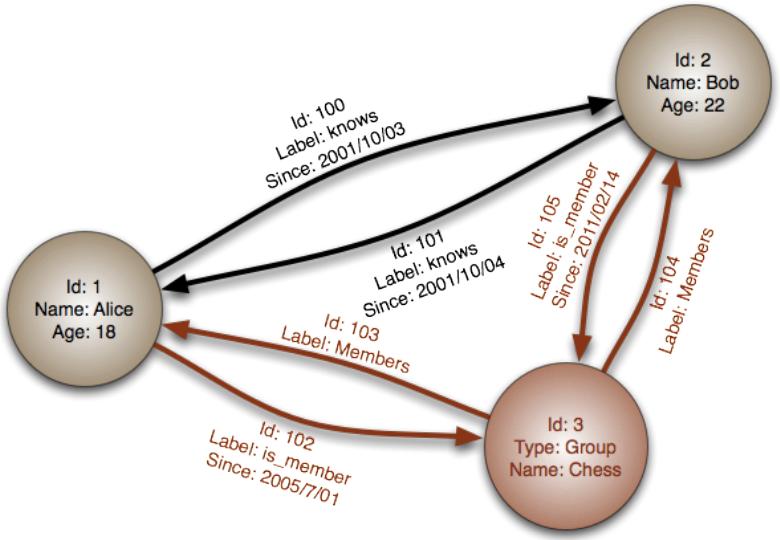
Grafos etiqueta-propiedad (Labeled-property graph)

- Constan de nodos, relaciones, propiedad y etiquetas.
- Los nodos representan entidades
- Nodos y relaciones tienen nombres. Pueden almacenar propiedades
- Relaciones dirigidas.
- Preferibles para almacenamiento.
- Ejemplo: Neo4J

Framework de descripción de recursos (Resource Description Framework)

- Cada nueva información se representa como un nodo.
- Un nodo puede no tener valor asignado.
- Normalmente usado para URLs.
- Lenguaje: SparQL (no confundir con Spark SQL)

- Datos relacionados entre sí
- Necesidad de almacenar propiedades de cada dato y de las relaciones entre ellos
- Necesidad de consultas complejas
- Análisis de patrones entre los datos almacenados



NewSQL

- Alternativa de las bases de datos relacionales a las NoSQL
- Intentan seguir el mismo modelo que las relacionales tradicionales.
- Aun no están plenamente asentadas en el sector, prefiriendo muchas empresas usar una combinación de base de datos relacional y varias NoSQL para sus fines.

Diferentes propuestas. Algunos ejemplos:

- VoltDB: Cada tabla se particiona usando una única columna clave.
- Clustrix: Particiona los datos utilizando un algoritmo hash basándose en una clave primaria definida por el usuario
- NuoDB: Compuesta de varios Storage Managers (SM) y Transaction Managers (TM). Los SMs son los nodos responsables del mantenimiento de los datos, mientras que los TMs son los responsables de procesar las consultas. El sistema de almacenamiento usado por los SMs puede particionar los datos

Cuando usar NewSQL

- Sistemas muy estructurados con relaciones
- Sistemas que tradicionalmente hayan usado bases de datos relacionales y no se pueda estudiar una transición a otro tipo de BBDD.
- Cuando se necesite ACID

Discusión

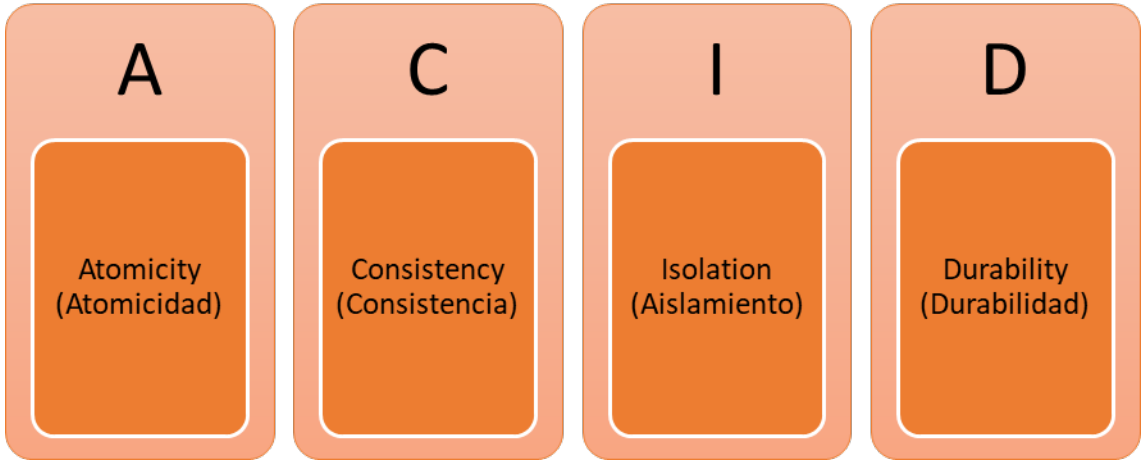
Caso 1

- Comercio intercontinental
- Miles de usuarios cada segundo
- Muchas transacciones
- Cada producto tiene diferentes propiedades
- Presupuesto elevado
- No se puede tolerar caídas

Caso 2

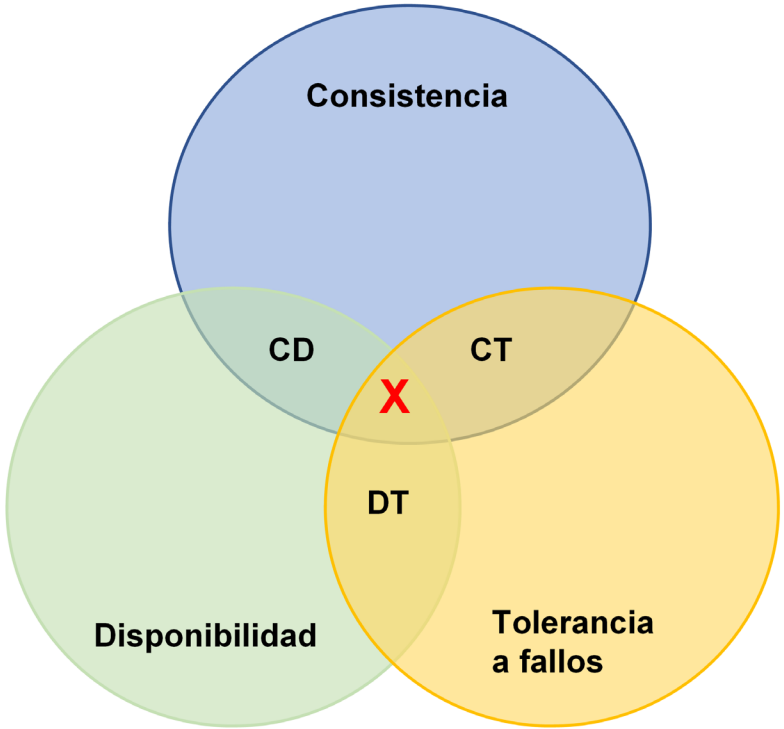
- Agencia de viajes regional
- Nivel medio-bajo de negocio, facturación de $200.000 anuales
- Cientos de usuarios a la hora consultando pero no realizando compras
- Consistencia importante
- Presupuesto limitado

Propiedades ACID



Teorema CAP

Determina que un sistema no puede garantizar consistencia, disponibilidad y ser tolerante a fallos debido al particionado a la vez



Diferencias

ACID

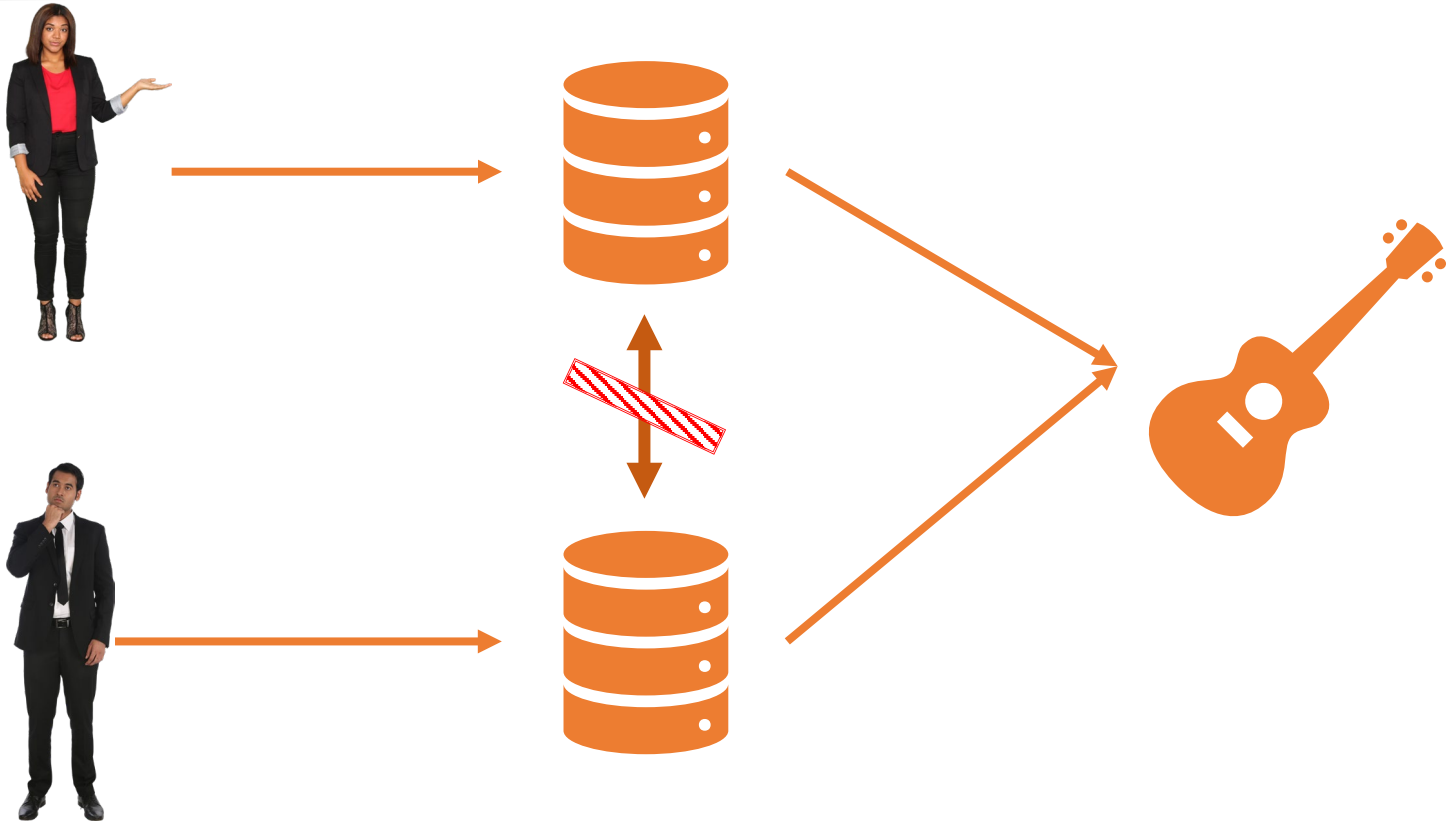
- Se refiere a que todos los datos almacenados en la base de datos cumplan las restricciones del esquema

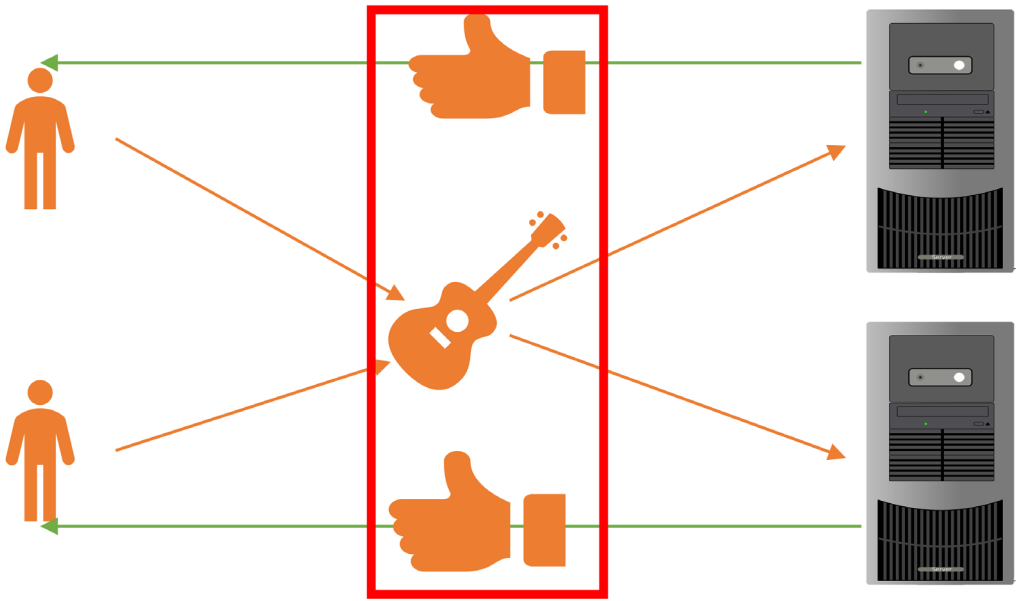
CAP

- Significa que todos los nodos de la base de datos tengan datos consistentes entre sí

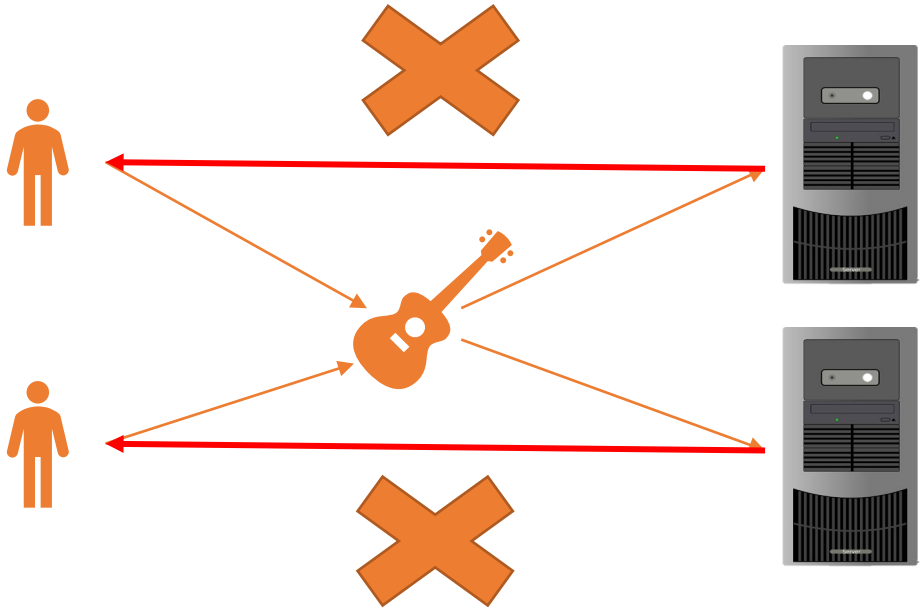
Sin embargo, las implementaciones ACID también aseguran la Consistencia de CAP

Problema ilustrativo CAP





Problemas de stock tras la compra



Compra rechazada para ambos clientes hasta que se solucione el problema de comunicación entre servidores

Ejemplos de elecciones de soluciones comerciales

- **Relacionales: CA**
- **Amazon Dynamo: AP**
- **Terracota: CA**
- **Apache Cassandra: AP**
- **Apache Zookeeper: AP**
- **BigTable: CA**
- **CouchDB: AP**
- **Neo4J: CA (en versiones reciente está intentando conseguir la P siguiendo modelo Cassandra)**
- **MongoDB: CP**
- **Hbase: CP**

Propiedades BASE