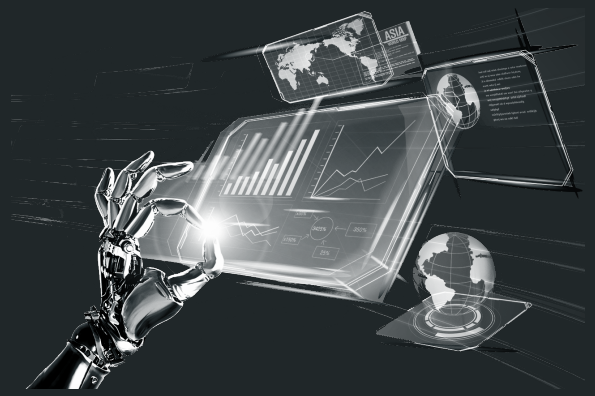


Gestión de Datos para Robótica

T2a - Bases de Datos NoSQL

Álvaro Vázquez Álvarez
Departamento de Electrónica e Computación

alvaro.vazquez@usc.es

Pabellón III - Despacho 4

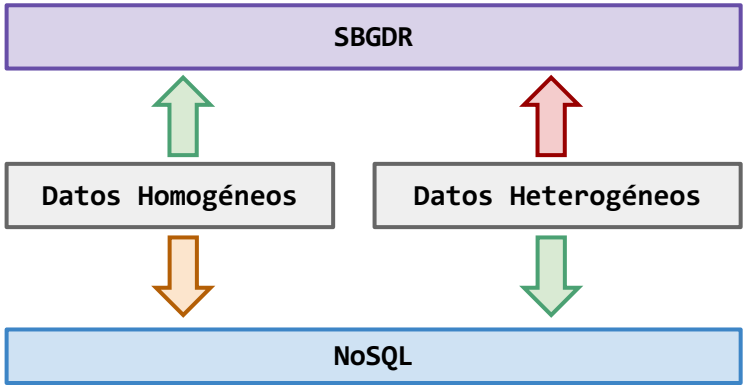
Curso 2023-2024

Tabla de contenidos

Introducción

NoSQL ***(Not Only SQL)***

- Conjunto de tecnologías que permiten el procesamiento rápido y eficiente de conjuntos de datos dando la mayor importancia al **rendimiento**, la **fiabilidad** y la **agilidad**.
- Las BD NoSQL se centran en sistemas complementarios a los SGBD relacionales, que fijan sus prioridades en la **escalabilidad** y la **disponibilidad** en vez de la **atomicidad** y **consistencia** de los datos.



Introducción

Tipos de sistemas NoSQL

- Clave-Valor
  - Cada elemento se almacena con un nombre de atributo (o clave) junto a su valor
- Grafos
  - Almacenan información sobre redes, como pueden ser conexiones sociales
- Columnas
  - Los datos se almacenan como columnas, no como filas
  - Cada fila puede contener un número diferente de columnas
- Documental
  - Cada clave almacena un documento con una estructura similar a JSON

Características

Beneficio de los sistemas NoSQL

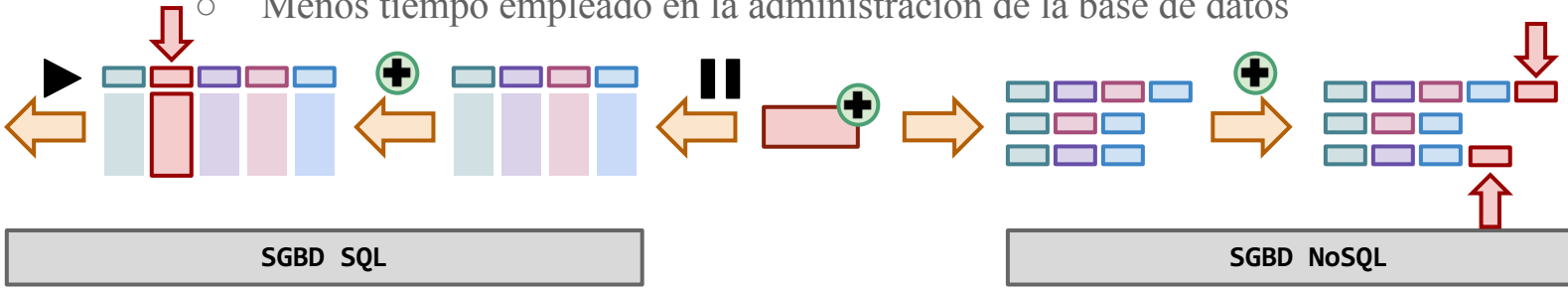
- Soporte para **grandes volúmenes** de **datos estructurados**, **semi-estructurados** y **no estructurados**.
  - Permiten comenzar con un modelo sencillo y con el paso del tiempo, añadir nuevos campos, a datos ya existentes.
- **Sprints ágiles**, iteraciones rápidas y commits/pushes frecuentes de código → emplea una sintaxis sencilla para la realización de consultas y la posibilidad de tener un modelo que vaya creciendo al mismo ritmo que el desarrollo.
- **Arquitectura eficiente** y **escalable** diseñada para trabajar con clusters de ordenadores
  - Más económica
  - Transparente para el desarrollador

Datos Estructurados	Datos Semi-estructurados	Datos No estructurados
Datos con formato o esquema definido que poseen campos fijos	**Datos que no tienen formato fijo, pero contienen etiquetas y otros marcadores**	**Datos sin tipos definidos, se almacenan como documentos u objetos sin estructura uniforme**
Hojas de cálculo, Formularios web, ...	Ficheros XML, HTML, JSON, ...	Audio, Vídeo, Documentos de texto (SMS, WhatsApps, Telegram)

Características

Esquemas dinámicos

- BD relacionales → **definir** el **esquema** (atributos + dominio) **antes** de **añadir** los **datos**.
  - Almacenar nuevos datos implica modificar el esquema → baja tolerancia a cambios
  - Grandes BD relacionales → proceso lento (parar el sistema, actualizar el esquema, actualizar la BD e inicializar el sistema).
- BD NoSQL → se permite la **inserción** de **datos** sin un **esquema** predefinido.
  - Facilita la modificación de las aplicaciones en tiempo real
  - No hay interrupciones de servicio
  - Desarrollo más rápido
  - Integración de código más robusto
  - Menos tiempo empleado en la administración de la base de datos



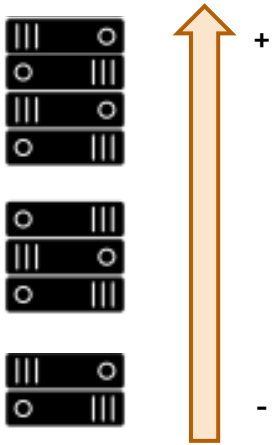
Características

Escalado de las BD

- Escalado vertical: aumentar los recursos del servidor
  - Ventajas:
    - Facilidad de implementación y configuración
    - No requiere un diseño específico en la aplicación y arquitectura para funcionar
    - Resulta económico y fácil de mantener.
  - Inconvenientes:
    - Limitado por la capacidad de un único servidor
    - No permite alta disponibilidad → si el servidor se cae, todo queda inaccesible.
- Escalado horizontal: aumentar el número de servidores (**sharding**)
  - Ventajas:
    - El escalado es prácticamente infinito
    - Permite alta disponibilidad → si un servidor se cae, habrá otro que tenga esos datos
    - Permite balanceo de cargas entre servidores
  - Inconvenientes:
    - Difícil de implementar y sobre todo de mantener
    - Opción poco económica ya que necesita múltiples servidores

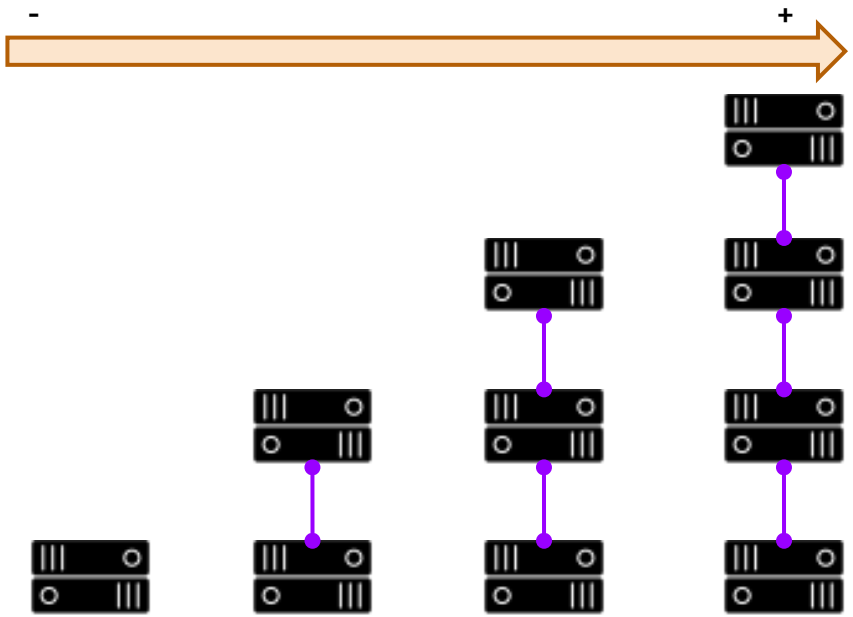
Características

Escalado de las BD



Escalado Vertical

Aumentar tamaño RAM, CPU, etc



Escalado Horizontal

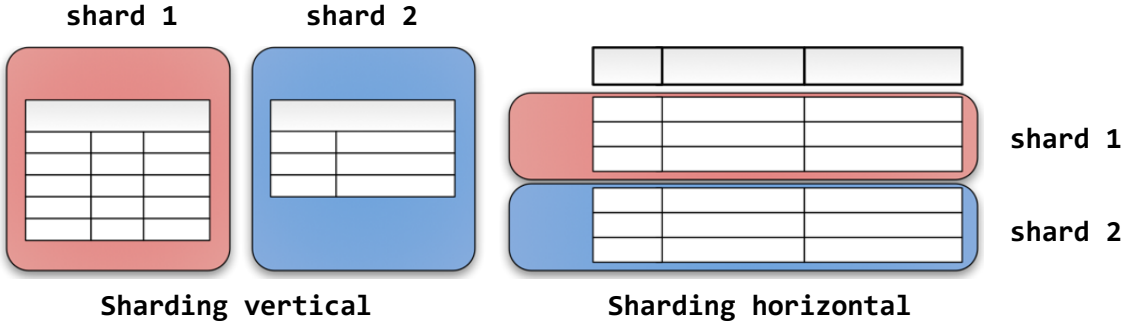
Aumentar el número de instancias (Servidores)

Características

Escalado horizontal

- Tipos de sharding (distribución de la información en las instancias):
  - Horizontal: diferentes filas en diferentes particiones
  - Vertical: diferentes columnas en diferentes particiones

CUIDADO!!
Escalado horizontal NO es lo mismo que sharding.
Escalado horizontal → incrementar el número de servidores
Sharding horizontal → dividir filas en particiones (shards)
Sharding vertical → dividir columnas en particiones (shards).



Características

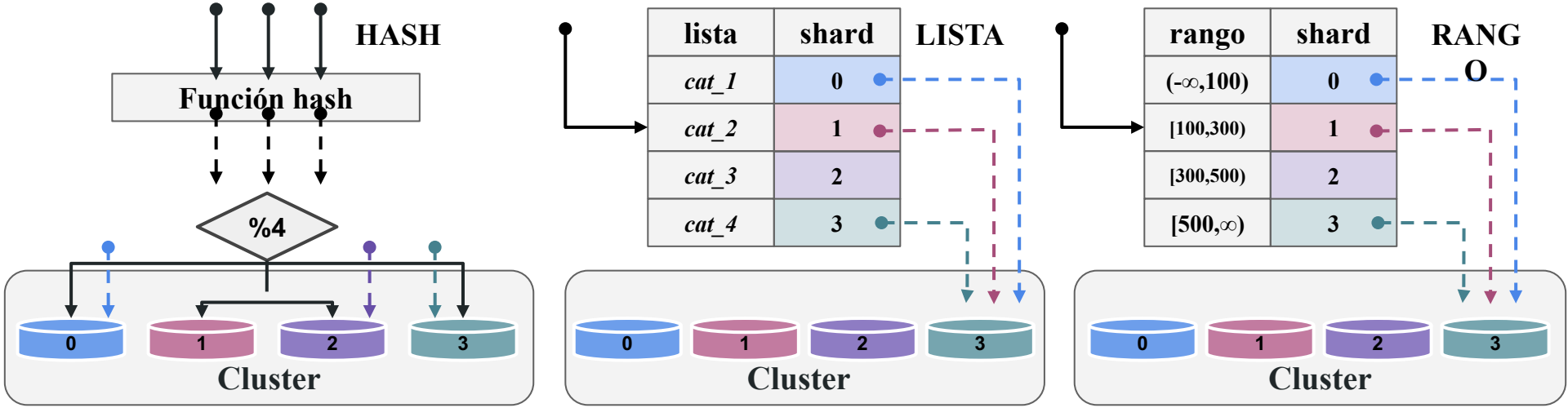
Escalado horizontal (Sharding)

- Sharding en sistemas SQL
  - Compatibles pero NO de forma nativa → requiere usar mecanismos para “simular” que es un único servidor
    - Uso de SAN (sistemas de almacenamiento en red)
    - Implementar código que permite gestionar: posibles fallos sobre los recursos, balanceo y replicación de datos. los joins entre diferentes bases de datos.
- Sharding en sistemas NoSQL
  - Compatibilidad de forma nativa (**auto-sharding**).
    - Dividen los datos entre un número arbitrario de servidores sin que la aplicación sea consciente de la composición del pool de servidores
    - Los datos y las consultas se balancean entre los servidores.
  - La compatibilidad con sharding horizontal y vertical depende del modelo de BD
    - Sharding horizontal: clave-valor, documentales, basadas en columnas.
    - Sharding vertical: basadas en columnas

Características

Escalado horizontal (Sharding)

- Sharding en sistemas NoSQL
  - Arquitecturas de auto-sharding:
    - Hash: devuelve un valor para un elemento que determina a qué partición pertenece
    - Lista: divide en función de las diferentes categorías de un campo.
    - Rangos: divide en intervalos en función de los valores del campo.



Características

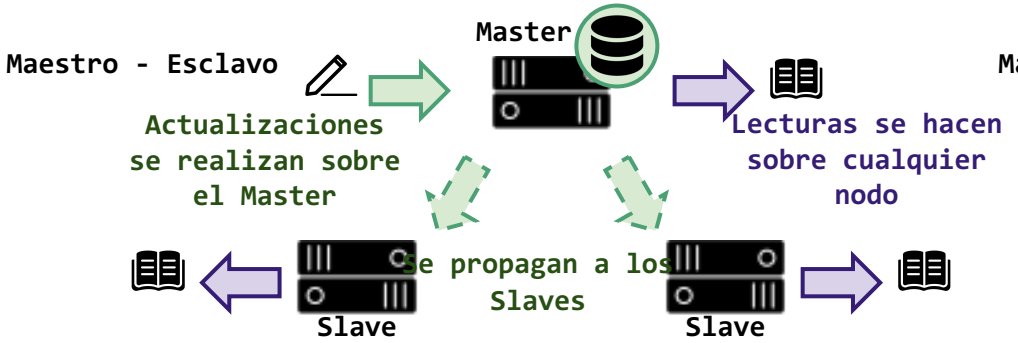
Escalado horizontal (Sharding)

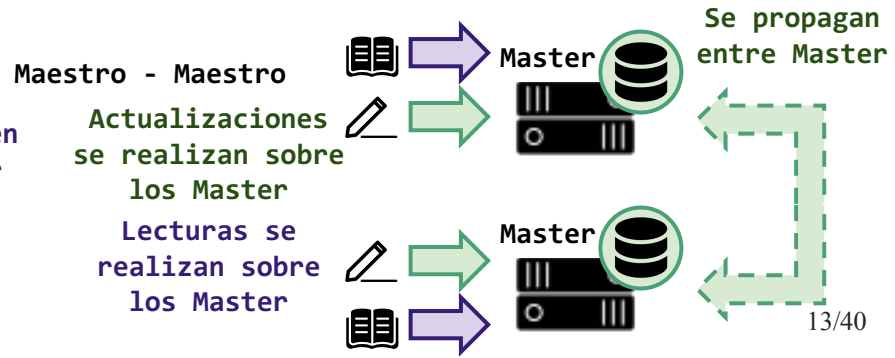
- Cuando hacer Sharding
  - Limitaciones de almacenamiento: los datos no caben en un único servidor (disco/RAM)
  - Rendimiento: al balancear la carga entre servidores, las escrituras son más rápidas.
  - Disponibilidad: si un servidor está ocupado/caído, otro servidor se puede ocupar.
    - Reduce la carga de cada servidor
- Cuando NO hacer Sharding
  - Cantidad de datos pequeña → distribuir los datos conlleva costes altos que puede no compensar con pocos volúmenes de datos.
  - Esperar a que haya una gran cantidad de datos → el proceso de particionado puede provocar sobrecarga del sistema.

Características

Replicación

- Mantener copias **idénticas** de los datos en **múltiples servidores** → equivalente a un RAID.
- Facilita aplicaciones robustas, incluso si alguno de los servidores sufre algún problema.
- Soportada nativa por:
  - Sistemas SQL
  - Sistemas NoSQL
- Tipos de replicación:
  - Maestro - Maestro: ambos servidores replican sobre los otros.
  - Maestro - Esclavo: todas las actualizaciones sobre el servidor maestro se replica sobre el esclavo





Características

Replicación

- Ventajas:
  - Escalabilidad y rendimiento: distribuye las consultas en diferentes nodos.
  - Disponibilidad: ofreciendo tolerancia a fallos o evitando la corrupción de la base de datos.
    - Nodos con réplicas pueden suplir los problemas de otros nodos.
  - Aislamiento: permite controlar cómo y cuándo se propagan los cambios entre nodos (esquema Maestro-Maestro y Maestro-Esclavo).

CUIDADO!!
Sharding NO es lo mismo que Replicacion.
Replicación → copias de datos en varios servidores
Sharding → cada máquina tiene un subconjunto de datos

ENTORNO DE TRABAJO MÁS SEGURO Y CON MEJOR RENDIMIENTO COMBINA:
SHARDING + REPLICACIÓN
CONCRETAMENTE CADA "SHARD" DEBERÍA ESTAR REPLICADO EN 2 O MÁS SITIOS
OJO: NO REPLICAR A LO LOCO!!

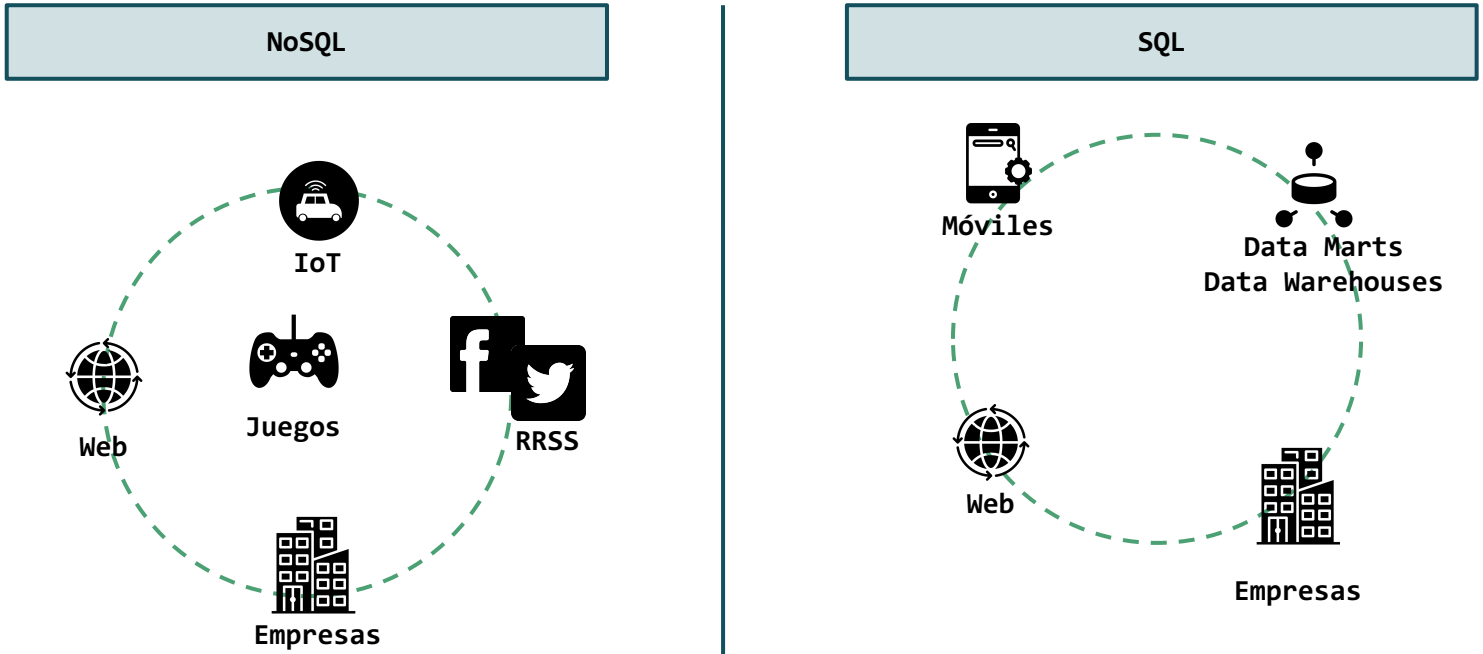
BD SQL vs NoSQL

- **Comparativa**

	SQL		NoSQL	
Rendimiento	Bajo		Alto	
Disponibilidad	Pobre		Buena	
Fiabilidad	Buena		Pobre	
Consistencia	Buena		Pobre	
Almacenamiento	Tamaño Medio-bajo		Gran Tamaño	
Escalabilidad	Alta pero cara		Alta	
Operaciones E/S	Escritura	Lectura	Escritura	Lectura

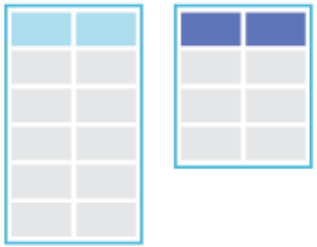
BD SQL vs NoSQL

- Ámbito de aplicación



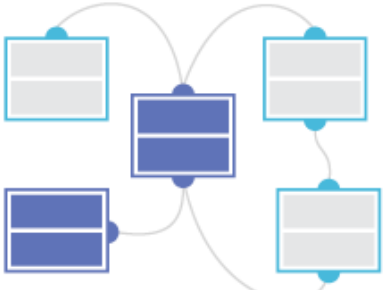
Modelos de Datos

Tipos de modelos de Datos:

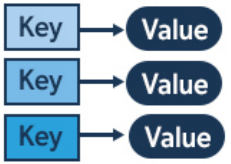


Basada en Columnas

Documentales



Grafos

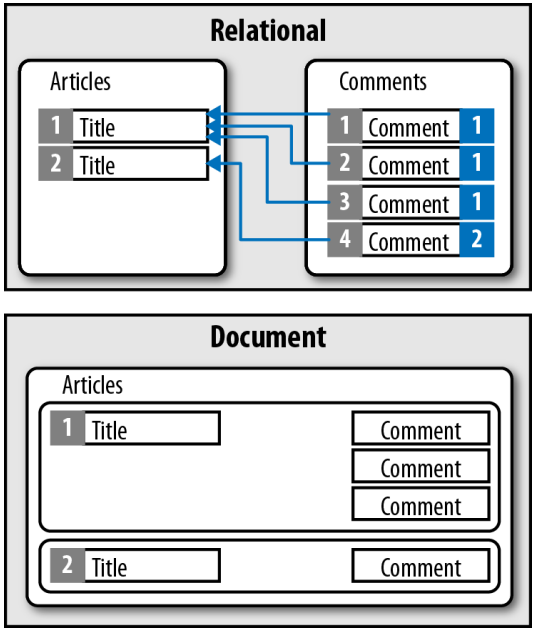


Clave-Valor

Modelos de Datos

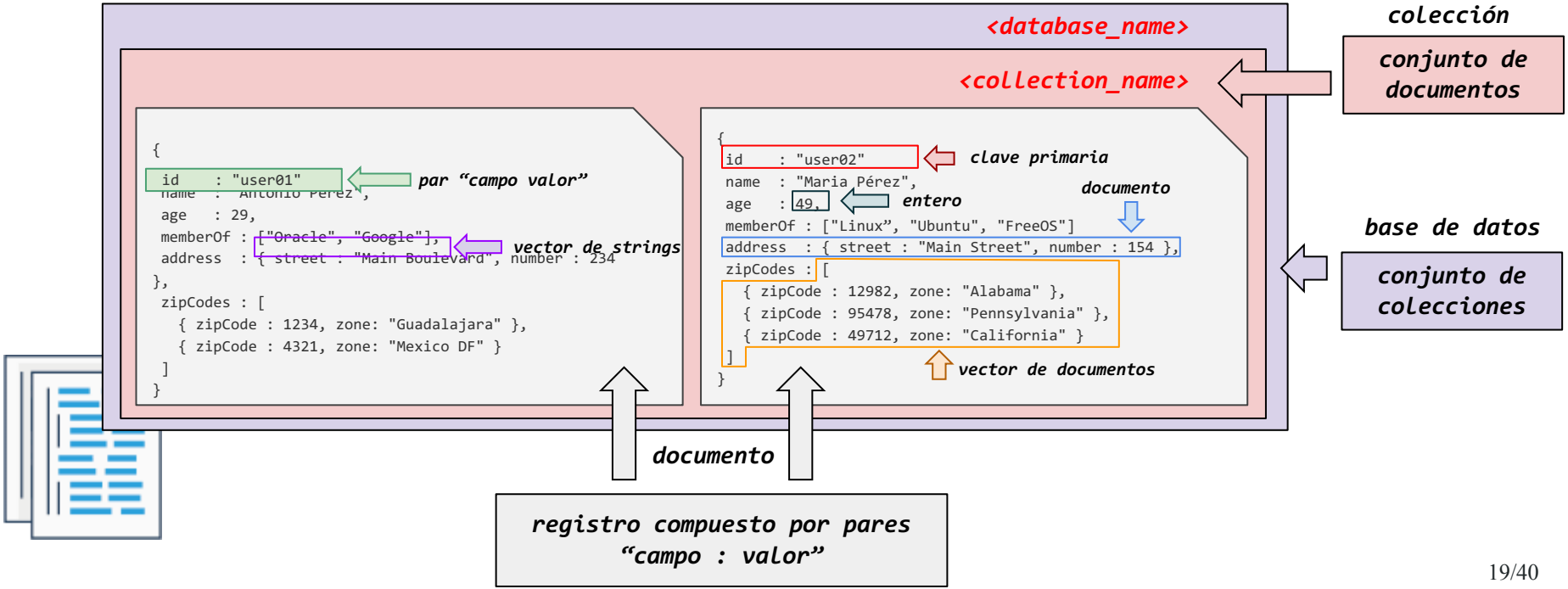
Bases de Datos Documentales

- La información se almacena en documentos
- Estructura similar a JSON o YML
- Se agrupan en colecciones (tablas) o bases de datos
- Contienen uno o más campos → cada campo contiene un valor con un tipo (documento, cadena, vectores, ...)
- Cada registro y sus datos asociados se almacenan de manera unida en un único documento en vez de repartirlos por múltiples columnas y tablas
  - Simplifica acceso a los datos
  - Reduce la necesidad de *joins* y transacciones complejas
- El **esquema** es **dinámico**: cada documento puede contener diferentes campos → permite modelar datos no estructurados.
- Cada documento contiene una clave → identificación unívoca
- Mecanismo completo de consultas → permite acceder a cualquier campo del documento
- Los documentos (y datos anidados) se pueden modificar mediante una única sentencia



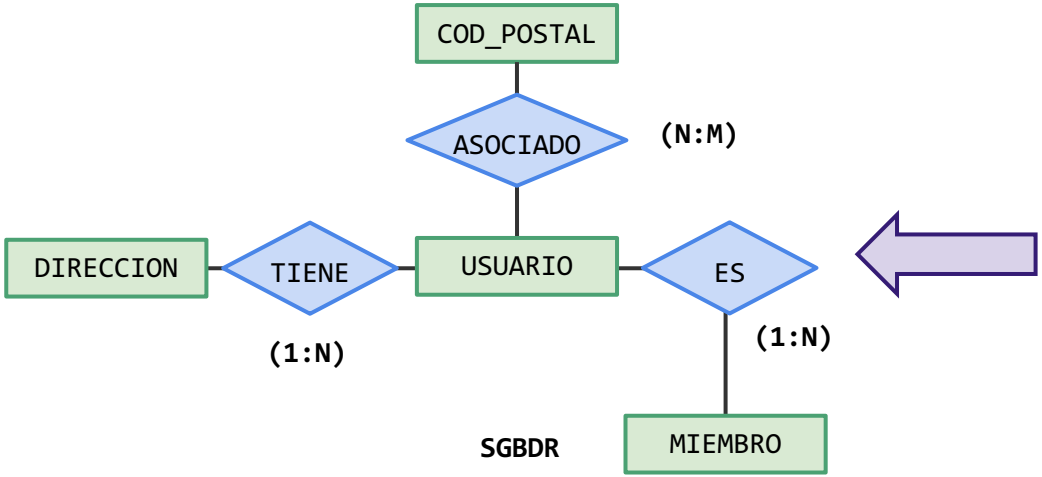
Modelos de Datos

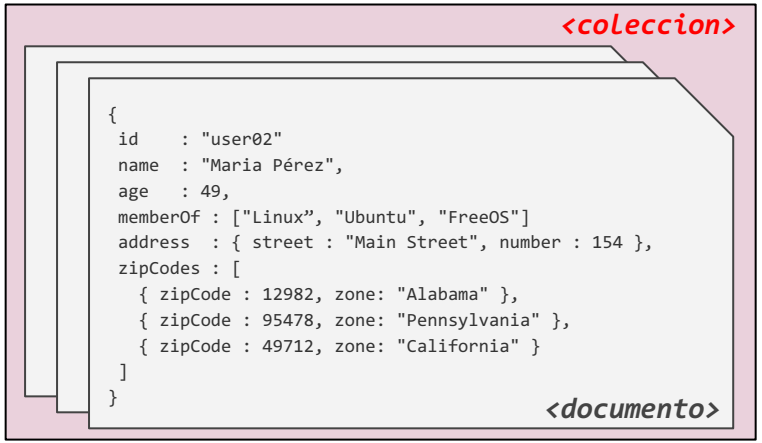
Bases de Datos Documentales: estructura



Modelos de Datos

Bases de Datos Documentales: comparativa con SGBD Relacionales





NoSQL - BD Documental

Relacional	base de datos	tabla	fila	columna	clave primaria
Documental	base de datos	colección	documento	campo	clave primaria

Modelos de Datos

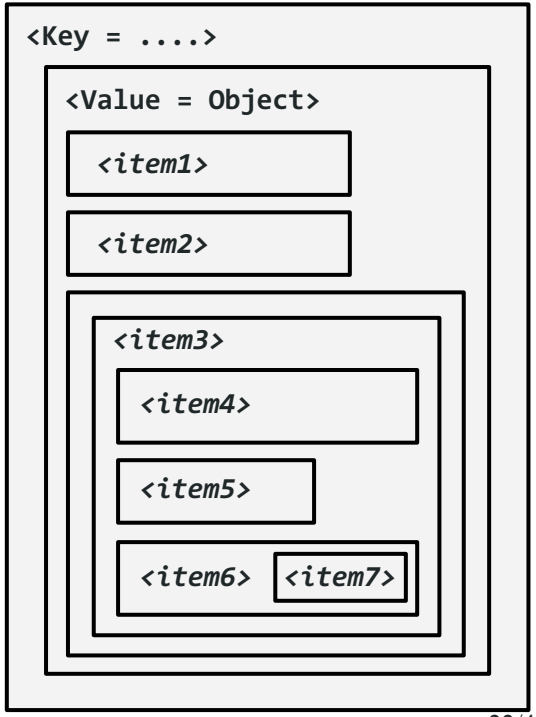
Bases de Datos Documentales: casos de uso

- Propósito general:
  - Amplio abanico de aplicaciones → flexibilidad del modelo de datos
  - Consultas sobre cualquier campo
  - Modelar de manera similar a la POO (Programación Orientada a Objetos).
- Casos de Uso:
  - Sistemas de flujos de eventos
  - Gestores de contenidos, plataformas de Blogging
  - Analíticas Web, datos en tiempo real
  - IoT
  - Aplicaciones e-commerce.
- Software:
  - MongoDB
  - CouchDB

Modelos de Datos

Bases de Datos Clave-Valor

- Mapa *Hash* donde todos los accesos a la base de datos se realizan a través de la clave primaria.
- Sistema NoSQL **más básico**
- Similar a tener una tabla relacional con dos únicas columnas (id, valor).
  - El campo valor puede ser un datos simple o un objeto.
- Operaciones:
  - Obtener el valor por la clave
  - Asignar un valor a una clave
  - Eliminar una clave del almacén
- El valor es independiente al sistema → los datos sólo se pueden consultar por la clave
  - la aplicación (y no el SGBD) es responsable de saber qué hay almacenado en cada valor.



Modelos de Datos

Bases de Datos Clave-Valor: características

- Gran rendimiento y fácilmente escalables
  - Usa sólo accesos por clave primaria.
  - Toda la información se almacena en memoria y se serializa de manera periódica, a costa de una consistencia eventual de los datos
- Algunos almacenes clave-valor permiten:
  - Almacenar datos con cualquier estructura → listas, conjuntos, hashes
  - Realizar operaciones como intersección, unión, diferencia y rango
  - Pueden permitir un segundo nivel de consulta o definir más de una clave.

Modelos de Datos

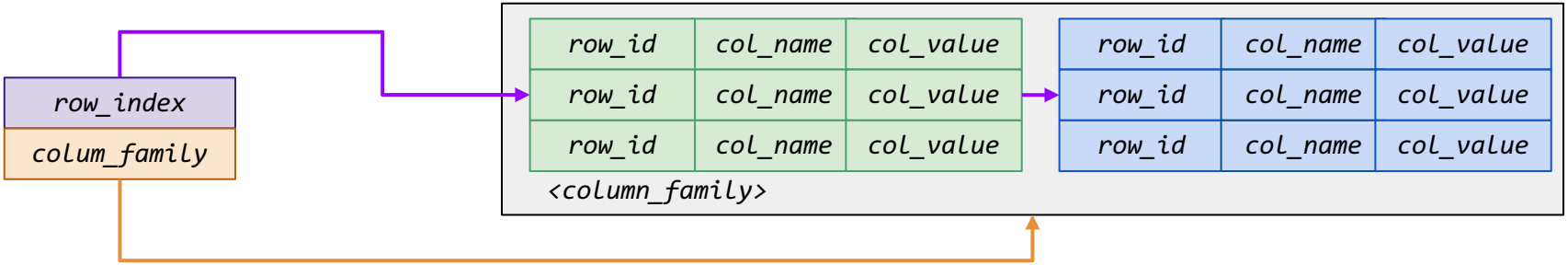
Bases de Datos Clave-Valor: casos de uso

- Propósito General:
  - Representar datos desestructurados → no fuerzan ningún esquema más allá de clave-valor
- Casos de uso:
  - Información sobre sesiones de navegación (sessionid) → *cookies*
  - Perfiles de usuario, preferencias
  - Datos del carrito de la compra
  - Caché de datos
- **No** válido para:
  - Relaciones entre datos
  - Transacciones entre varias operaciones
  - Consultas por los datos del valor
  - Operaciones con conjuntos de claves
- Software:
  - Riak
  - Redis
  - Voldemort → implementación open-source de *Amazon DynamoDB*

Modelos de Datos

Bases de Datos Basados en Columnas

- Sistemas Big Data o Tabulares → *BigTable* de Google
- Estructura formada por dos niveles:
  - Nivel 1: almacén clave-valor
    - Clave representa el identificador de la fila
    - Valor incluye la familia de columnas.
  - Nivel 2: los valores son las columnas.



Modelos de Datos

Bases de Datos Basados en Columnas: comparativa con SQL

id	*Nombre*	*Tipo*	*Contenido*
1	Juan	Oferta	Oferta Moviles
2	Ana	Venta	Venta Samsung
3	Carlos	Pedido	Entrega Pedido
4	Diana	Venta	Venta Huawei

REGISTRO

SGBDR

Relacional vs Columnar

REGISTRO

id	*Columna*	*Valor*
1	Nombre	Juan
1	Tipo	Oferta
1	Contenido	Oferta Moviles
2	Nombre	Ana
2	Tipo	Venta
2	Contenido	Venta Samsung
3	Nombre	Carlos
3	Tipo	Pedido
3	Contenido	Entrega Pedido
...	...	...

BD columnar está formada por 3 columnas (id, nombre atributo, valor)
ID de la BD columnar se denomina **row_key**
El row key es **único** para cada registro de la BD columnar.
+ Un **registro** tiene **varios atributos** y varios **valores**
Cada atributo se identifica por un **nombre de columna** y **tiene un valor**

NoSQL - Base de Datos Columnar

Modelos de Datos

Bases de Datos Basados en Columnas: características

- No desaprovecha espacio cuando se almacenan datos dispersos (no estructurados)

id	*Nombre*	*Tipo*	*Contenido*	*Expiración*
1	Juan	Oferta	Oferta Moviles	12/08/2023
2	Ana	Venta	Venta Samsung	
3	Carlos	Pedido	Entrega Pedido	
4	Diana	Venta	Venta Huawei	25/12/2022

SGBDR

id	*Columna*	*Valor*
1	Nombre	Juan
1	Tipo	Oferta
1	Contenido	Oferta Moviles
1	Expiracion	12/08/2023
..	...	...
4	Nombre	Ana
4	Tipo	Venta
4	Contenido	Venta Samsung
4	Expiración	25/12/2022

NoSQL - Base de Datos Columnar

Modelos de Datos

Bases de Datos Basados en Columnas: características

- Permite modificar los atributos dinámicamente sin cambiar la estructura

id	*Nombre*	*Tipo*	*Contenido*	*Expiración*	*Estado*
1	Juan	Oferta	Oferta Moviles	12/08/2023	Tránsito
2	Ana	Venta	Venta Samsung		Entregado
3	Carlos	Pedido	Entrega Pedido		Tránsito
4	Diana	Venta	Venta Huawei	25/12/2022	Tránsito

SGBDR

id	*Columna*	*Valor*
..	...	...
2	Nombre	Ana
2	Tipo	Venta
2	Contenido	Venta Samsung
2	Estado	Entregado
..	...	...

NoSQL - Base de Datos Columnar

Modelos de Datos

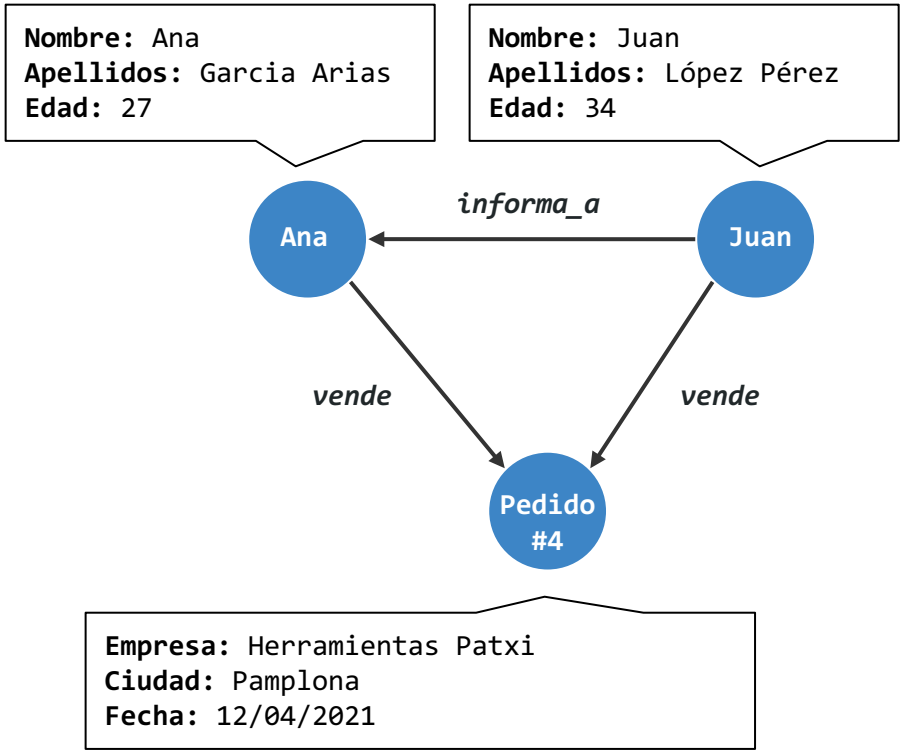
Bases de Datos Basados en Columnas: casos de uso

- Propósito General:
  - Aplicaciones que necesitan consultar los datos por un único valor
    - Grandes volúmenes de datos, alto rendimiento y escalabilidad.
- Casos de uso:
  - Sistemas de flujo de eventos: estados de las aplicaciones, errores.
  - Gestores de contenido, plataformas de Blogging
  - Contadores: almacenar las visitas de cada persona a cada lugar de un site.
- **No** válido para:
  - Sistemas operacionales con transacciones complejas o con consultas agregadas
  - Prototipado inicial o sistemas donde el esquema no esté fijado de antemano
- Software:
  - HBase: en cual se basa en *Hadoop.*
  - Cassandra
  - Amazon SimpleDB

Modelos de Datos

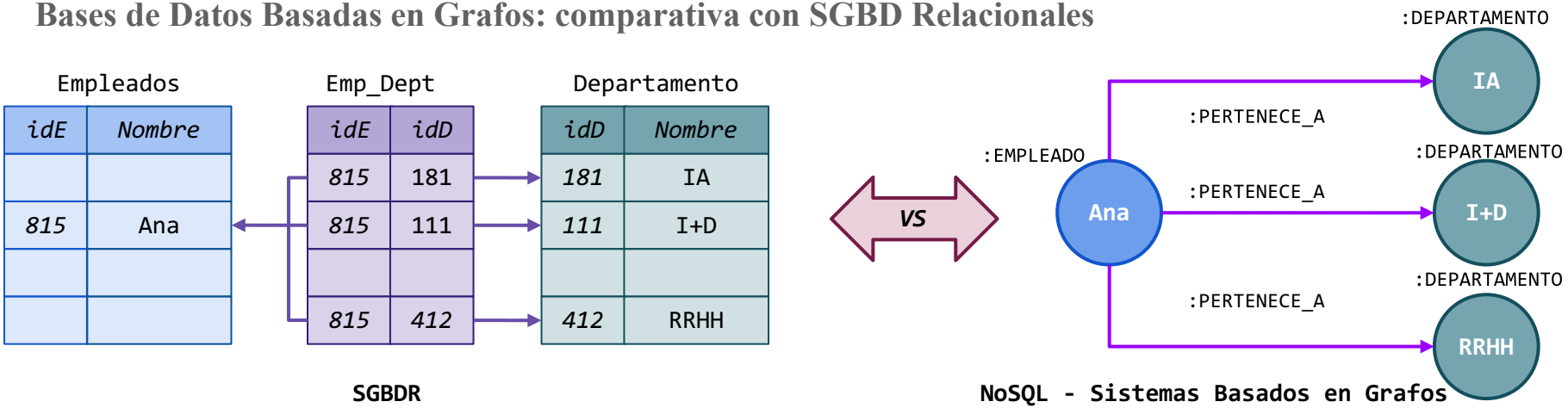
Bases de Datos Basadas en Grafos

- Almacenan entidades y las relaciones entre ellas.
  - Entidades (nodos) tienen propiedades (similar a una instancia de un objeto).
  - Relaciones (vértices): tienen propiedades, y su sentido es importante.
- Se organizan mediante relaciones que facilitan encontrar patrones de información existente entre nodos
  - Permite que los datos se almacenan una vez y se interpretan de diferentes maneras dependiendo de sus relaciones.



Modelos de Datos

Bases de Datos Basadas en Grafos: comparativa con SGBD Relacionales



Relacional	Filas	Columnas	Tablas	Claves foráneas
Grafos	Nodos	Propiedades	Etiquetas en Nodos/Aristas	Aristas entre Nodos

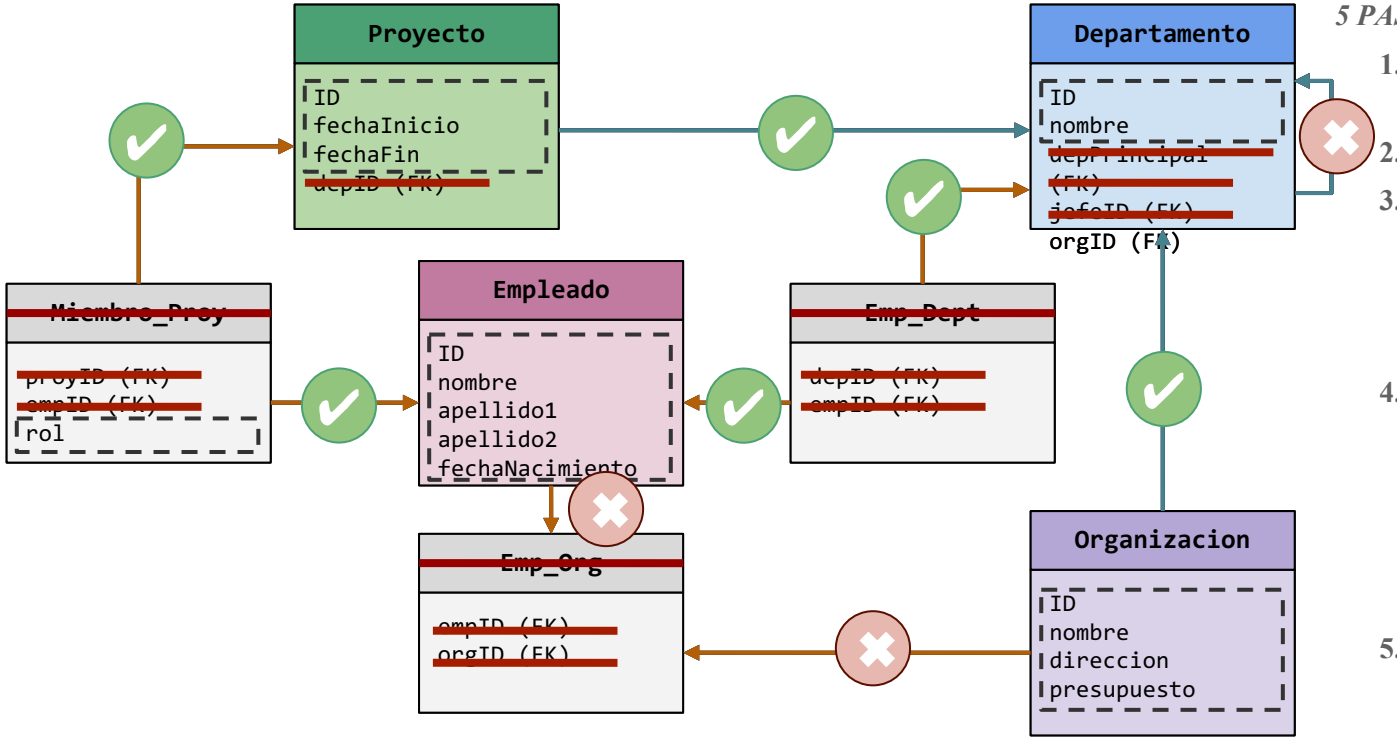
Modelos de Datos

Bases de Datos Basadas en Grafos: características

- Traversing: recorrer el grafo para realizar una consulta.
- Recorrer las relaciones es muy rápido → no hay joins de tablas.
- Permite descubrir potenciales relaciones ocultas entre nodos.
- Poco intuitivo
- Gran curva de aprendizaje
- Permite la conversión de una BD Relacional a una BD basada en Grafos.

Modelos de Datos

Bases de Datos Basadas en Grafos: conversión SGBDR a SGBDG

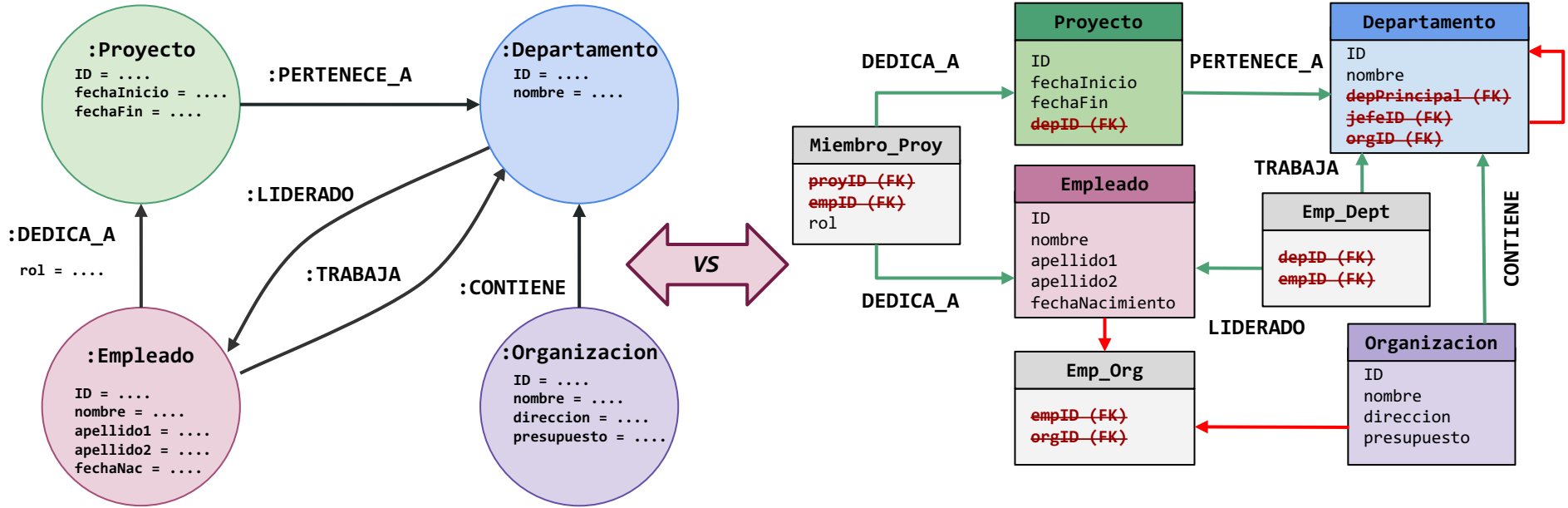


5 PASOS:

1. Identificar las tablas principales
   + nombre → la etiqueta del nodo
2. Eliminar las FK de las tablas
3. Identificar atributos resultantes
   + tablas secundarias → propiedades de la relación
   + Tablas principales → propiedades del nodo
4. Identificar relaciones
   + tablas secundarias: todas las flechas salen hacia tablas principales → relacion
   + tablas principales: flechas salen hacia otras tablas principales → relacion
5. Eliminar tablas secundarias

Modelos de Datos

Bases de Datos Basadas en Grafos: conversión SGBDR a SGBDG



Modelos de Datos

Bases de Datos Basadas en Grafos: casos de uso

- Modelo poco intuitivo y con una importante curva de aprendizaje
- Casos de uso:
  - Datos conectados: redes sociales con diferentes tipos de conexiones entre los usuarios
  - Enrutamiento, entrega o servicios basados en la posición
    - Consultas sobre trayectos cortos, lugares cercanos
  - Motores de recomendaciones: compras, lugares visitados, ...
- **No** válido cuando: hay que realizar múltiples operaciones de inserción/modificación
  - Modificar una propiedad/entidad es muy complejo.
- Software:
  - Neo4J
  - FlockBD
  - HyperGraphDB

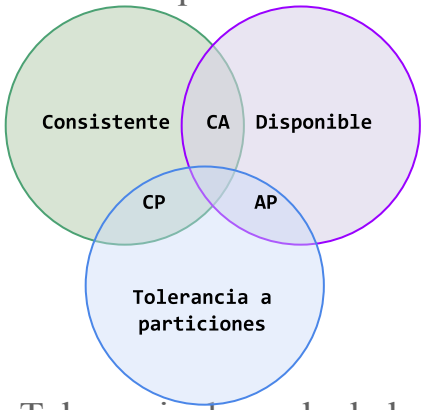
Consistencia

- Sistema **consistente**: las escrituras son visibles inmediatamente.
- Sistema **eventualmente consistente**: las escrituras no son visibles inmediatamente, pero lo serán en un breve periodo de tiempo (**ventana** de **inconsistencia**).
  - Aceptable para:
    - Aplicaciones de sólo lectura
    - Almacenes de datos que no cambian frecuentemente → históricos.
    - Aplicaciones con alta tasa de escritura donde hay pocas lecturas → archivo de log
- Bases de Datos Documentales y Basadas en Grafos
  - Admiten consistencia total y consistencia eventual (mediante configuración).
- Bases de Datos Clave-Valor y Basadas en Columnas
  - Consistencia eventual

EJEMPLO: Sistema de control de STOCK
+ CONSISTENCIA → cada consulta obtendrá el estado real del inventario.
+ CONSISTENCIA EVENTUAL → la consulta puede no mostrar el estado real en un determinado momento (resultados sin actualizar) pero terminará siéndolo.

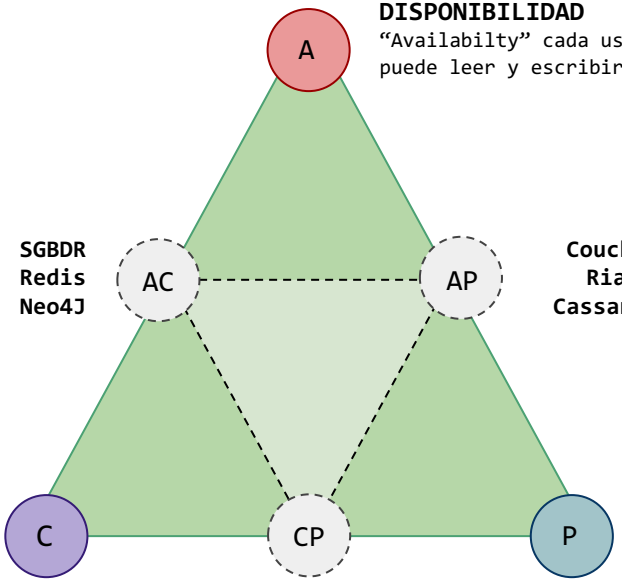
Teorema de CAP

- El teorema de CAP propuesto por Brewer en 2009 indica que se puede crear una base de datos distribuida que cumpla un máximo de dos de las tres características siguientes:
  - Consistente (**C**onsistency): actualizaciones atómicas → se obtienen los valores actualizados.
  - Disponible (**A**vailability): la BD siempre devolverá un valor.
  - Tolerancia a Particiones (**P**artition Tolerance): el sistema funcionará aunque la comunicación con un servidor se interrumpa de manera temporal.
- Posibilidades:
  - CP: Consistente y Tolerante a Particiones
  - AP: Disponible y Tolerante a Particiones
  - CA: Consistente y Disponible
- Usos del Teorema CAP
  - Para seleccionar el SGBD más acorde a las necesidades.
  - Decidir qué característica descartar.
    - La elección se basa en Consistencia o Disponibilidad → Tolerancia depende de la arquitectura.



Teorema de CAP

- Clasificación de los SGBD según CAP



DISPONIBILIDAD
"Availabilty" cada usuario siempre puede leer y escribir

no distribuyen los datos entre diferentes particiones en la red

SGBDR
Redis
Neo4J

CouchDB
Riak
Cassandra

permiten replicar los datos entre sus nodos aunque no garantiza la consistencia en ninguno de los sus servidores

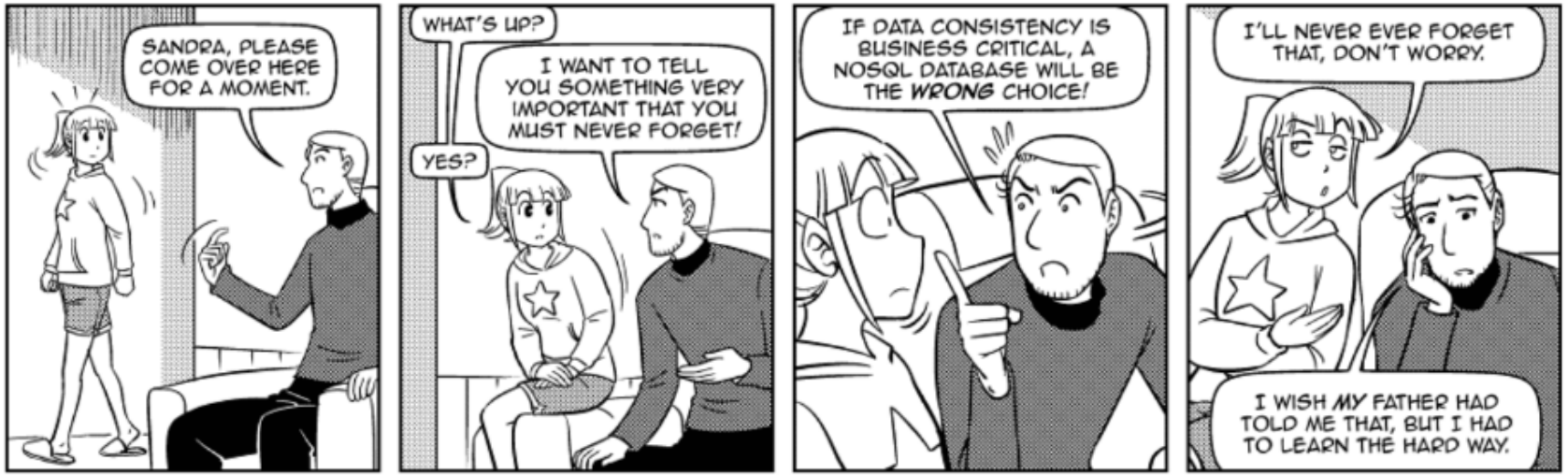
Consistencia
"Consistency" todos los clientes siempre tienen la misma vista de los datos.

Tolerancia a Particiones
"Partition Tolerance" el sistema trabaja bien, no importa las particiones físicas de la red

MongoDB
HBase

permite que las consultas siempre devuelvan la misma vista y que se pueda acceder a todos los nodos (nodo secundario puede convertirse en principal para asumir el rol de un nodo caído)

Recordad ....


SANDRA, PLEASE COME OVER HERE FOR A MOMENT.
WHAT'S UP?
YES?
I WANT TO TELL YOU SOMETHING VERY IMPORTANT THAT YOU MUST NEVER FORGET!
IF DATA CONSISTENCY IS BUSINESS CRITICAL, A NOSQL DATABASE WILL BE THE WRONG CHOICE!
I'LL NEVER EVER FORGET THAT, DON'T WORRY.
I WISH MY FATHER HAD TOLD ME THAT, BUT I HAD TO LEARN THE HARD WAY.

Bibliografía

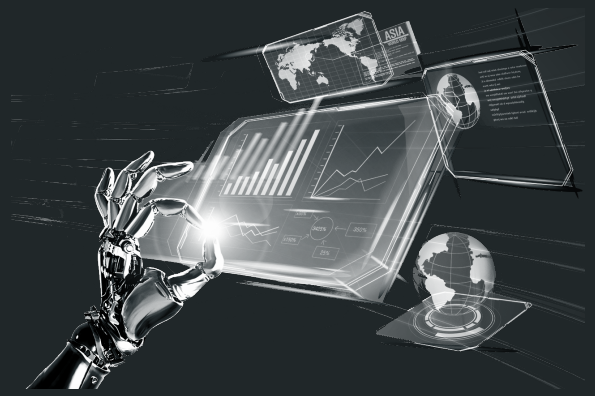
Marcela Sena, 2017. Como pasar de SQL a NoSQL sin sufrir.

Google Inc, 2021. Your Google Cloud database options explained.

Michael Hunter, 2020. From Relational to Graph: A Developer's Guide.

Neo4j, 2016. RDBMS to Graphs.

Neo4j, 2020. Model: Relational to Graph.



Gestión de Datos para Robótica

T2a - Bases de Datos NoSQL

Álvaro Vázquez Álvarez
Departamento de Electrónica e Computación

alvaro.vazquez@usc.es

Pabellón III - Despacho 4

Curso 2023-2024