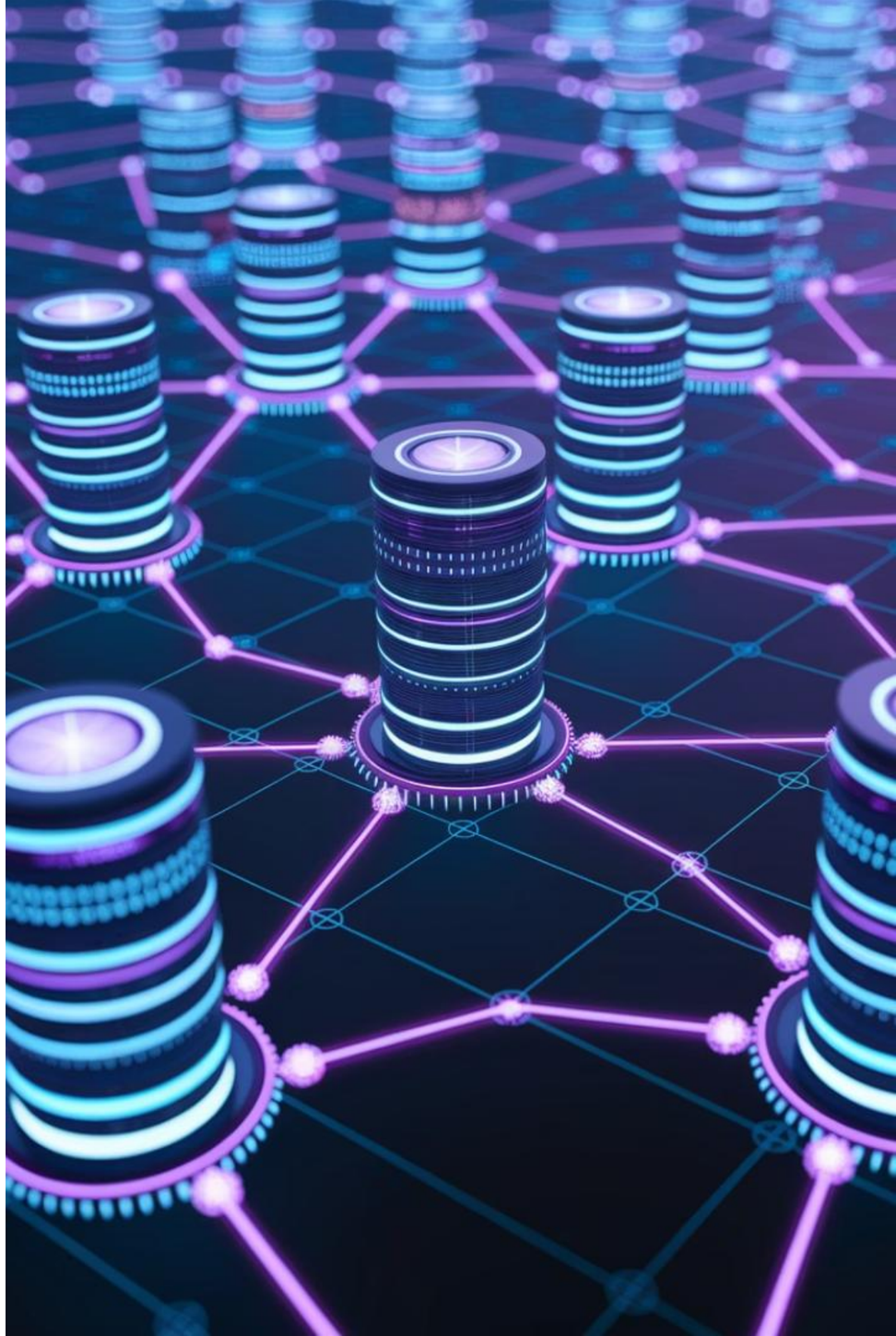
Tecnologías de Bases de Datos: Fundamentos y Tipos

Las bases de datos son el corazón de los sistemas de información modernos. En esta presentación, exploraremos qué son los sistemas de bases de datos, sus diferentes tipos, principios fundamentales y cómo funcionan en entornos reales.

Veremos desde las tradicionales bases de datos relacionales hasta las modernas soluciones NoSQL, analizando sus características, ventajas y casos de uso ideales para cada tecnología.

por Kibernum Capacitación



Reflexión sobre Bases de Datos y Sistemas Críticos

Experiencias con Bases de Datos

- ¿Qué sistemas de bases de datos has utilizado o escuchado antes (como MySQL, MongoDB, Firebase, etc.)?
- ¿Recuerdas en qué contexto?

Diversidad de Tipos de Bases de Datos

- ¿Por qué crees que existen diferentes tipos de bases de datos?
- ¿En qué situaciones podría no bastar con una base de datos tradicional?

Impacto en Sistemas Críticos

- ¿Cómo afecta la disponibilidad, la consistencia o la tolerancia a fallos a un sistema como una tienda online o una red social?

¿Qué es un Sistema de Base de Datos?

Definición Formal

Un sistema computacional que permite la gestión estructurada de información, proporcionando funcionalidades para almacenar, consultar, actualizar y borrar datos de forma controlada.

Componentes Principales

Conjunto organizado de datos y programas que permiten almacenar, modificar y extraer información de forma eficiente y segura.

Ejemplo Práctico

Una agenda digital que guarda nombres, teléfonos y correos, permitiéndote buscar, agregar o borrar contactos. Esto es, en miniatura, un sistema de base de datos.

Tipos de Sistemas de Base de Datos

Bases de Datos Relacionales (RDBMS)

Organizan los datos en tablas (filas y columnas), como una hoja de Excel. Cada fila es un registro y cada columna representa un campo o atributo.

Ejemplos: MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

Caso de uso ideal: Cuando los datos tienen relaciones claras y reglas estrictas, como en un sistema bancario.

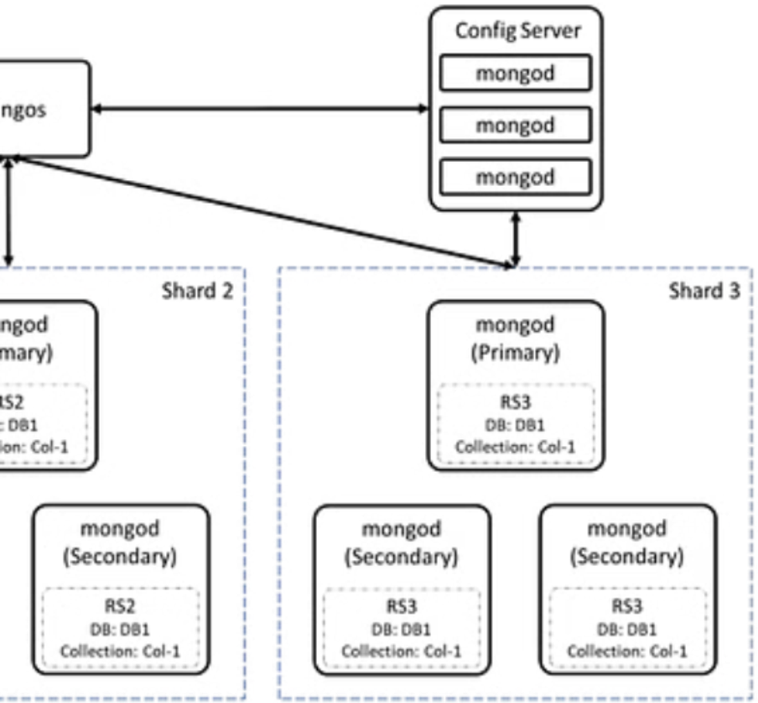
Bases de Datos NoSQL

Diseñadas para almacenar datos no estructurados o semiestructurados, de forma más flexible que las relacionales.

Tipos: Documentales (MongoDB), Clave-valor (Redis), Columnar (Cassandra), Grafos (Neo4j).

Caso de uso ideal: Cuando necesitas escalabilidad o manejar grandes volúmenes de datos que cambian rápido, como redes sociales, catálogos, logs de servidores, etc.

Bases de Datos Documentales - MongoDB



Estructura Flexible

Guarda la información como documentos JSON. En lugar de tablas y filas, MongoDB usa colecciones y documentos.

Casos de Uso

Aplicaciones web modernas, tiendas online (e-commerce), blogs, foros y redes sociales. Ideal cuando los datos son flexibles o cambian mucho.

Ventajas

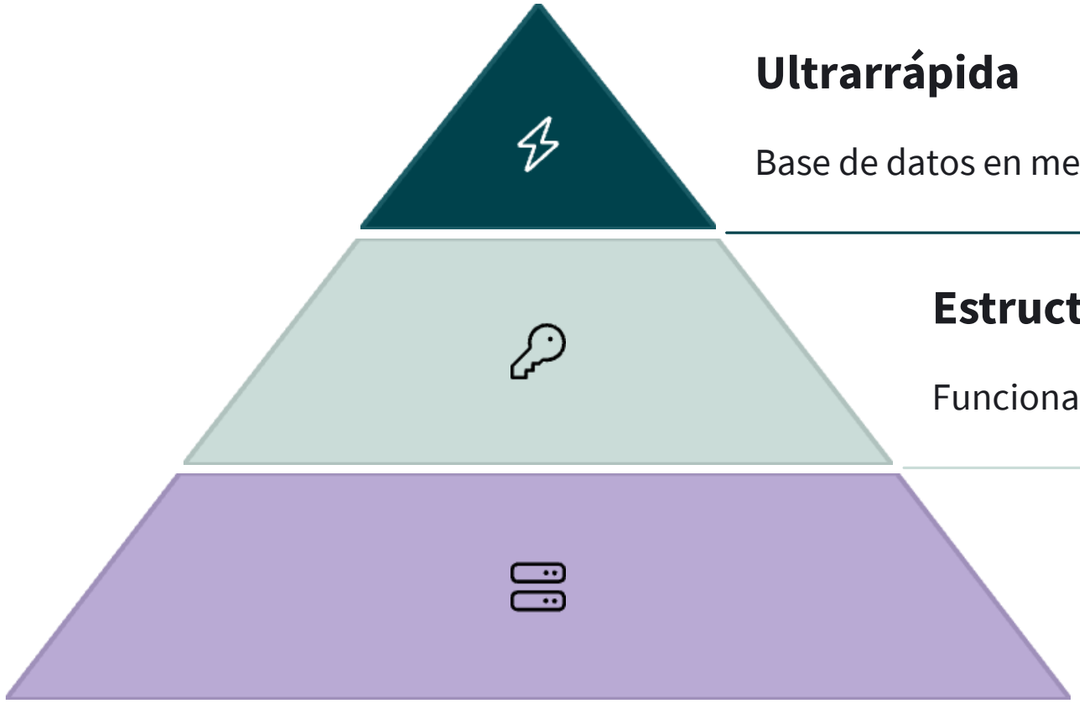
Permite guardar estructuras complejas (objetos anidados), no necesita un esquema fijo y permite búsquedas avanzadas dentro de los documentos.

Ejemplo Bases de Datos Documentales - MongoDB

```
// Insertar un documento en la colección "usuarios"
db.usuarios.insertOne({
  nombre: "Juan Pérez",
  correo: "juan@gmail.com",
  edad: 30,
  direccion: {
    calle: "Calle Falsa 123",
    ciudad: "Santiago"
  }
});
```

Bases de Datos Clave-Valor - Redis

```
# Guardar un valor
SET nombre "Miguel"

# Obtener el valor guardado
GET nombre
```



Ultrarrápida

Base de datos en memoria para acceso instantáneo

Estructura Simple

Funciona como un diccionario: clave → valor

Casos de Uso

Caché, sesiones, contadores, configuraciones

Redis (Remote Dictionary Server) es una base de datos clave-valor que se guarda en memoria, lo que la hace extremadamente rápida. Es como un diccionario gigante, donde cada palabra (clave) tiene un significado (valor).

Su simplicidad y velocidad la hacen ideal para almacenar datos temporales que necesitan ser accedidos frecuentemente, como sesiones de usuario o configuraciones de aplicaciones.

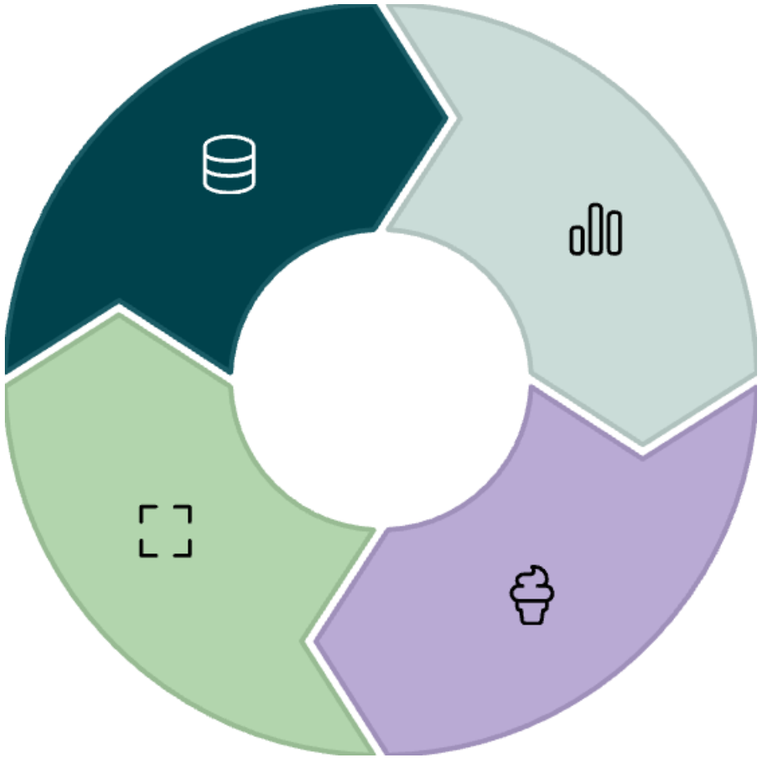
Bases de Datos Columnar - Cassandra

Estructura por Columnas

Guarda la información por columnas, no por filas

Escalabilidad

Añade más nodos fácilmente para crecer



Big Data

Perfecta para grandes volúmenes de datos distribuidos

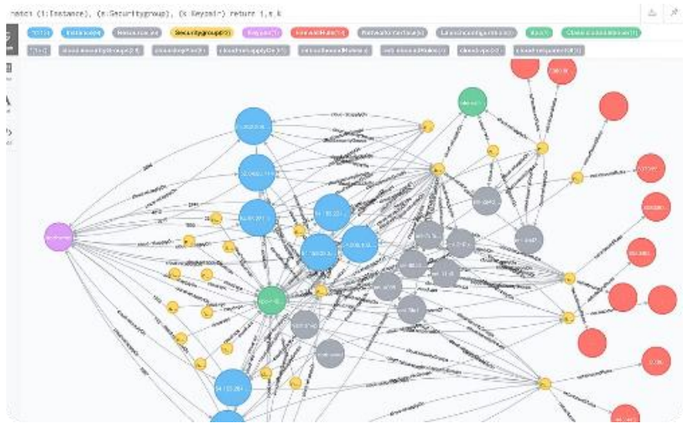
Alta Disponibilidad

Sin punto único de falla, siempre accesible

Ejemplo Bases de Datos Columnar - Cassandra

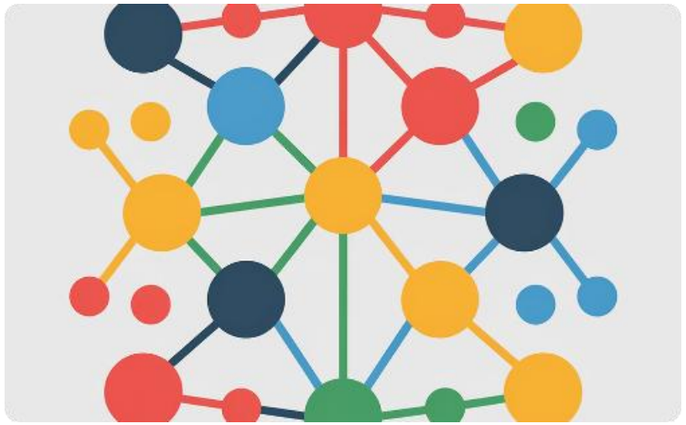
```
-- Crear una tabla
CREATE TABLE empleados (
  id UUID PRIMARY KEY,
  nombre TEXT,
  cargo TEXT,
  salario DOUBLE
);

-- Insertar un empleado
INSERT INTO empleados (id, nombre, cargo, salario)
VALUES (uuid(), 'Laura', 'Analista', 75000.0);

-- Consultar todos los empleados
SELECT * FROM empleados;
```

Bases de Datos de Grafos - Neo4j



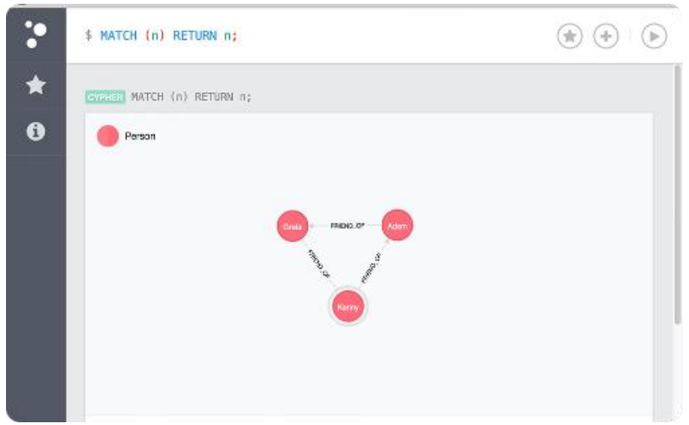
Estructura de Red

En lugar de tablas o documentos, usa nodos (entidades) y relaciones (conexiones) para representar los datos como una red interconectada.



Casos de Uso

Ideal para redes sociales, sistemas de recomendación y análisis de relaciones complejas entre personas, productos o eventos.

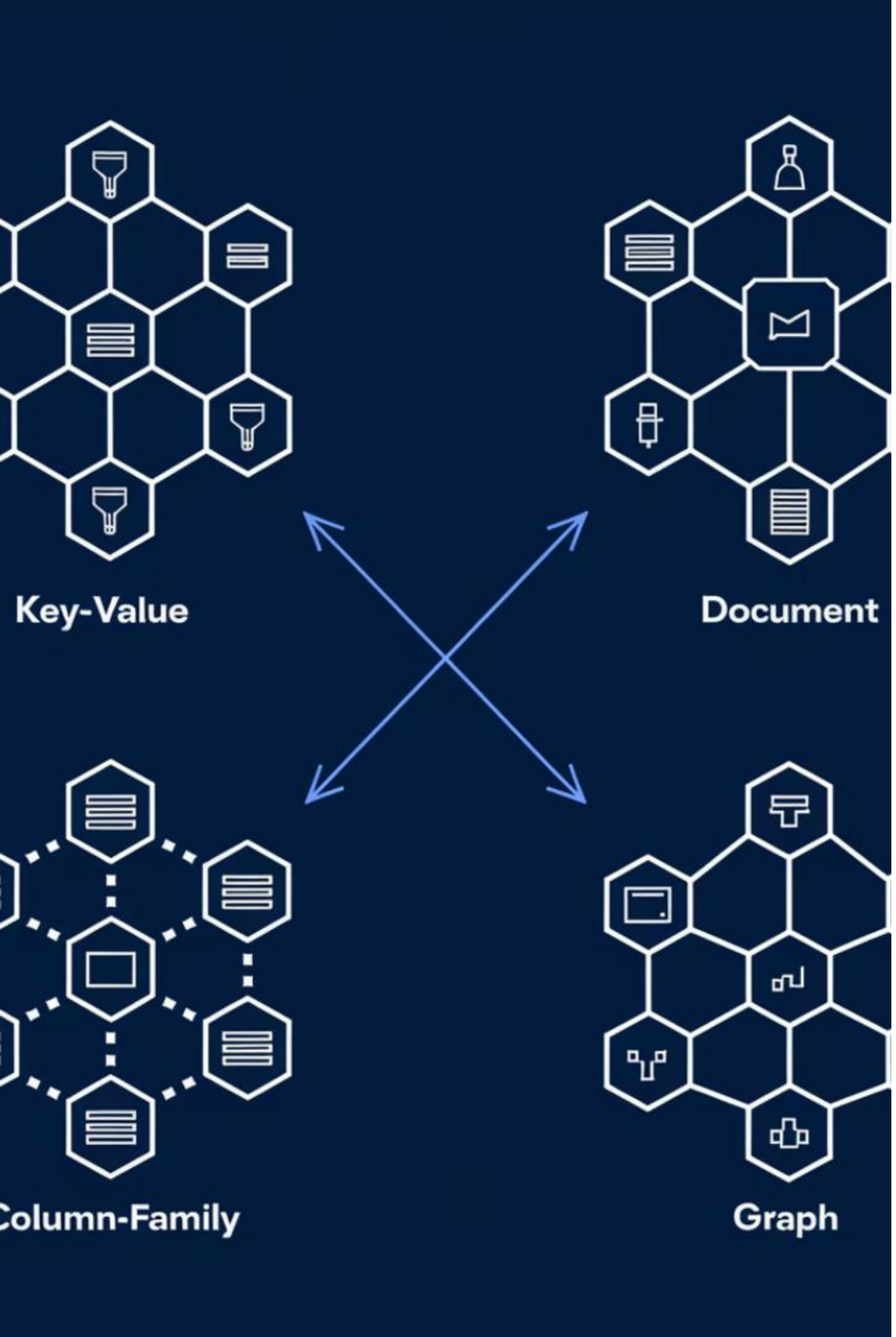


Lenguaje Cypher

Neo4j utiliza Cypher, un lenguaje de consulta que permite expresar patrones de relaciones de forma intuitiva, como si dibujaras una red de conexiones.

Ejemplo Bases de Datos de Grafos - Neo4j

```
// Crear nodos persona
CREATE (a:Persona {nombre: "Carlos"})
CREATE (b:Persona {nombre: "Ana"})

// Crear una relación entre ellos
CREATE (a)-[:AMIGO_DE]->(b)

// Consultar amistades
MATCH (p:Persona)-[:AMIGO_DE]->(amigo)
RETURN p.nombre, amigo.nombre
```



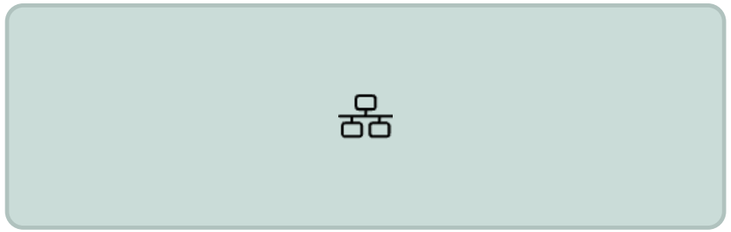
Comparativa de Bases de Datos NoSQL

Tipo	Ejemplo	Formato	Caso de uso común
Documental	MongoDB	JSON	Datos estructurados y flexibles
Clave-valor	Redis	Clave → Valor	Caché, sesiones, tokens
Columnar	Cassandra	Column Families	Big Data, series de tiempo, métricas
Grafos	Neo4j	Nodos y relaciones	Redes, recomendaciones, grafos sociales

Principios Fundamentales: ACID vs BASE

ACID (RDBMS)

Garantiza transacciones seguras y consistentes



BASE (NoSQL)

Prioriza disponibilidad y tolerancia a particiones

Elección según necesidades

Compromiso entre consistencia y disponibilidad

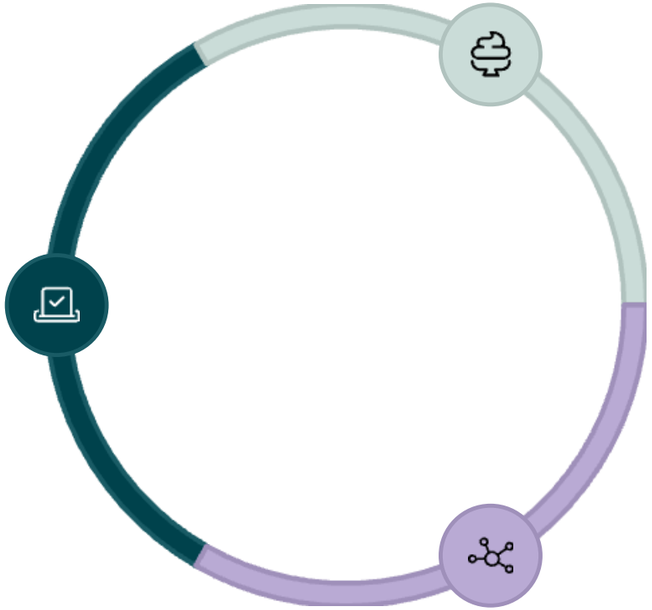
ACID garantiza que las transacciones sean Atómicas (todo o nada), Consistentes (cumplen las reglas), Aisladas (no interfieren entre sí) y Durables (permanecen tras fallos).

BASE es más flexible: Básicamente Disponible (siempre responde), Estado Suave (puede cambiar con el tiempo) y Eventualmente Consistente (los datos se sincronizan eventualmente).

Teorema de CAP

Consistency (C)

Todos ven la misma versión del dato al mismo tiempo.



Availability (A)

El sistema siempre responde, incluso si hay fallos.

Partition Tolerance (P)

El sistema sigue funcionando, aunque haya fallos en la red que dividen al sistema.

El Teorema de CAP, también llamado Teorema de Brewer, establece que una base de datos distribuida solo puede garantizar dos de estas tres propiedades al mismo tiempo. Es como tener tres globos y solo poder sostener dos con tus manos.

Las bases NoSQL suelen elegir: Availability + Partition Tolerance (AP) como Cassandra, o Consistency + Partition Tolerance (CP) como MongoDB en configuración segura.

Transaccionalidad en Bases de Datos

Verificación

Se verifica la disponibilidad del recurso

Bloqueo

Se bloquea el recurso para evitar conflictos

Operación

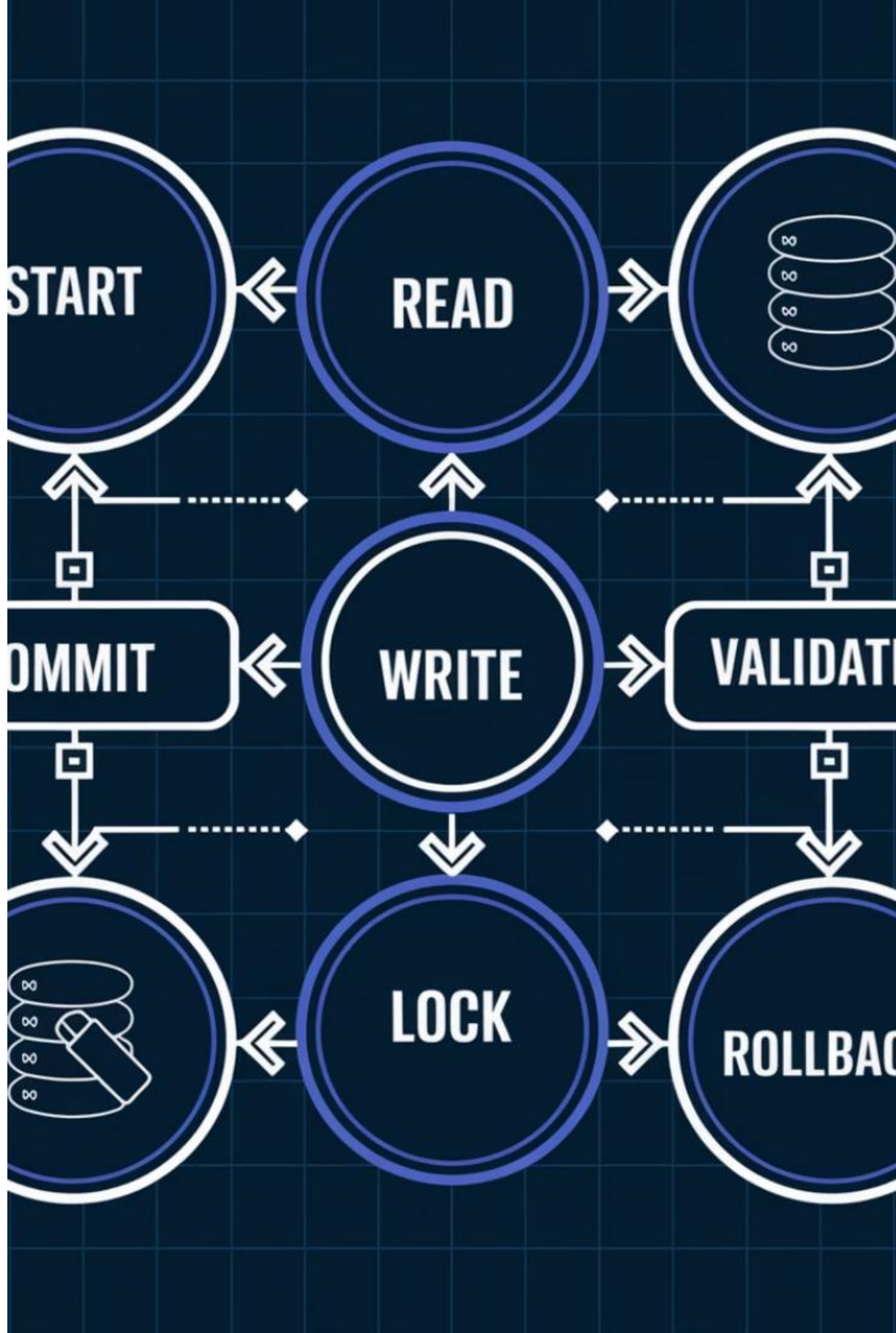
Se realiza la operación principal (ej: pago)

Confirmación

Si todo sale bien: COMMIT. Si algo falla: ROLLBACK

Una transacción es una unidad de trabajo que debe ejecutarse completa y correctamente. En bases de datos relacionales, las transacciones siguen el modelo ACID, garantizando precisión y recuperación ante errores mediante instrucciones como BEGIN, COMMIT y ROLLBACK.

En bases NoSQL, algunos sistemas también permiten transacciones (como MongoDB desde la versión 4.0), pero suelen tener un alcance limitado, priorizando disponibilidad y rendimiento sobre la consistencia estricta.



Enlace a Material Complementario

https://www.youtube.com/watch?v=rO1yB9jOAR0

🎥 Este video explica de forma clara y didáctica los tipos de bases de datos, los principios ACID y BASE, y cómo se aplica el teorema CAP en sistemas modernos, con ejemplos y analogías sencillas.

Preguntas de Reflexión Final

¿Qué diferencias clave identificaste entre las bases de datos relacionales (RDBMS) y NoSQL?

¿Cómo aplicarías el teorema CAP para tomar una decisión entre usar una base de datos que garantice alta disponibilidad o una que garantice consistencia?

¿Cómo te ayuda el principio ACID a comprender el comportamiento esperado en una transacción bancaria?