

# Comparativa RDBMS vs NoSQL

## Contexto: 🗨️

Para elegir una base de datos adecuada, es fundamental entender las diferencias entre los modelos relacional y NoSQL en función de sus características técnicas y sus casos de uso.

## Consigna: 📝

En grupos, analicen un caso de negocio (ej. red social, banco digital o tienda online) y determinen si es más adecuado utilizar una base de datos relacional o NoSQL. Justifiquen su elección en función de consistencia, escalabilidad y disponibilidad.

**Tiempo:** 🕒 30 minutos

## 🔗 Paso a paso:

1. Lean la descripción del caso que les entregará el docente.
2. Identifiquen los requisitos clave (consistencia, volumen de datos, disponibilidad, velocidad, etc.).
3. Seleccionen el tipo de base de datos más apropiado (RDBMS o NoSQL).
4. Argumenten su decisión considerando los principios ACID o BASE y el Teorema de CAP.
5. Expliquen cómo este tipo de base de datos resuelve los desafíos del caso.
6. Compartan sus conclusiones oralmente o mediante una breve presentación.

## Caso de negocio: Red Social (tipo Instagram/TikTok)

### 1. Descripción del caso

Una plataforma de red social que permite a los usuarios publicar fotos y videos, seguir a otros usuarios, dar "me gusta", comentar y recibir notificaciones en tiempo real.

### 2. Requisitos clave

- Alta disponibilidad: los usuarios acceden 24/7 desde cualquier parte del mundo.
- Escalabilidad: millones de usuarios y contenido creciendo constantemente.
- Velocidad de lectura/escritura: debe ser rápido para mostrar contenido y registrar interacciones.
- Consistencia eventual aceptable para ciertos datos (por ejemplo, número de "me gusta").

### 3. Tipo de base de datos seleccionada

**NoSQL** (por ejemplo, Cassandra o MongoDB)

### 4. Justificación y principios (ACID, BASE, CAP)

- **CAP:** En este caso se prioriza **disponibilidad (A)** y **partición tolerante (P)** sobre consistencia estricta (C). Es decir, aceptamos **consistencia eventual** en muchos datos.
- **BASE:** Optamos por una base **BASE** (Basically Available, Soft-state, Eventually consistent), más adecuada que una con propiedades ACID estrictas.
- **Escalabilidad horizontal:** los sistemas NoSQL permiten escalar agregando más servidores fácilmente.
- Las operaciones como dar "me gusta" o seguir a alguien **no necesitan transacciones estrictamente consistentes** en tiempo real.

## 5. ¿Cómo resuelve NoSQL los desafíos?

- Distribuye los datos en múltiples nodos para permitir alta disponibilidad y tolerancia a fallos.
- Maneja eficientemente grandes volúmenes de datos no estructurados (fotos, videos, comentarios).
- Ofrece alta velocidad en escritura y lectura para manejar interacciones masivas de los usuarios.

## 6. Conclusión

Para una red social moderna, donde la prioridad es la escalabilidad, disponibilidad y rendimiento frente a una consistencia estricta, **NoSQL es la opción más adecuada.**

---

## Mapa visual del Teorema de CAP

### Contexto: 🏠

Comprender el Teorema de CAP es clave para diseñar soluciones distribuidas con bases de datos modernas.

### Consigna: 📌

Crea un mapa visual (diagrama, infografía o esquema) que explique el Teorema de CAP y ubique diferentes tecnologías de bases de datos según sus prioridades (CP, CA, AP).

**Tiempo** ⌚: 30 minutos

### 🔗 Paso a paso:

1. Repasa la definición del Teorema de CAP.
2. Define cada propiedad (C, A, P) con un ejemplo simple.
3. Ubica al menos 4 tecnologías reales (MongoDB, Cassandra, PostgreSQL, Redis) en el triángulo de CAP según lo aprendido.
4. Opcional: destaca qué tipo de casos de uso prioriza cada combinación.
5. Comparte tu mapa con la clase (puede ser físico o digital).

---

## Mapa Visual del Teorema de CAP

### ¿Qué es el Teorema de CAP?

En sistemas distribuidos, el Teorema de CAP establece que una base de datos no puede garantizar simultáneamente las tres propiedades siguientes:

- **C - Consistency (Consistencia):** Todos los nodos ven los mismos datos al mismo tiempo.
- **A - Availability (Disponibilidad):** Cada solicitud recibe una respuesta (sin garantía de que contenga los datos más recientes).
- **P - Partition Tolerance (Tolerancia a particiones):** El sistema sigue funcionando, aunque se interrumpa la comunicación entre algunos nodos.

**\*Solo se pueden cumplir dos de las tres propiedades al mismo tiempo.**

---

### Ejemplos simples para cada propiedad:

- **Consistencia:** Cuando publicas un tweet y todos tus seguidores lo ven de inmediato.
- **Disponibilidad:** Puedes acceder a tu red social incluso si momentáneamente los datos no están actualizados.

- **Tolerancia a particiones:** Aunque un servidor falla, otros siguen respondiendo (aunque con retraso o inconsistencia).

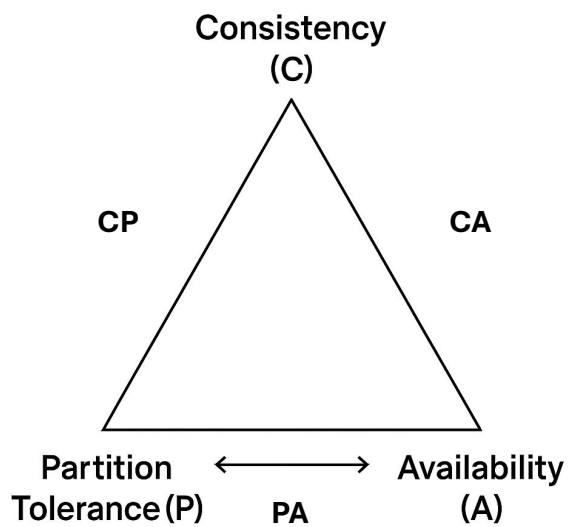
---

#### Ubicación de Tecnologías Reales en el Triángulo CAP:

Tecnología	Clasificación CAP	Caso de uso típico
MongoDB	CA	Apps con alta disponibilidad y consistencia, pero limitadas en tolerancia a particiones (en ciertos entornos).
Cassandra	AP	Redes sociales, analítica masiva — prioriza disponibilidad y partición, acepta consistencia eventual.
PostgreSQL	CP	Aplicaciones financieras o empresariales — consistencia fuerte con tolerancia a fallos, menos enfoque en disponibilidad.
Redis (Cluster)	CP / CA	Datos en tiempo real, como cachés consistentes o contadores.

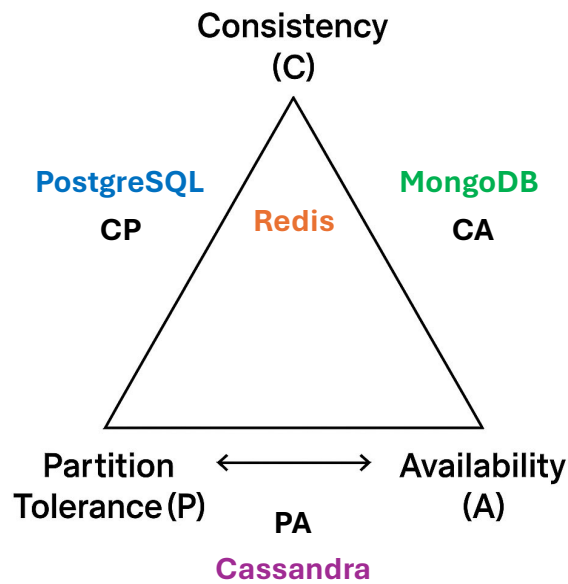
---

#### Visual sugerida (esquema tipo triángulo CAP):



Ubicación sugerida:

- MongoDB: en el borde CA
- Cassandra: en el borde AP
- PostgreSQL: en el borde CP
- Redis: entre CP y CA, según configuración



---

### Conclusión:

Este mapa visual ayuda a entender **qué sacrificios o beneficios implica elegir una tecnología**, según el tipo de aplicación que estés diseñando (por ejemplo, un sistema bancario vs. una red social).