Match entre tipo de NoSQL y caso de uso real

Contexto: 🙌

Comprender cuál es el tipo de base de datos NoSQL más adecuado según el problema o aplicación es clave para tomar decisiones de diseño efectivas.

Consigna: ✍️

Relacione diferentes tipos de bases de datos NoSQL con escenarios reales de uso. Luego justifica tu elección y describe qué ventajas aporta ese modelo en particular.

Tiempo🕐: *30 minutos*

🧭 **Paso a paso:**

1. En grupos, analicen 4 casos de uso que entregará el docente (por ejemplo: red social, ecommerce, IoT, análisis financiero).
2. Seleccionen para cada caso el tipo de base de datos NoSQL más adecuado (clave-valor, documental, columnares, grafos o en memoria).
3. Justifiquen su elección considerando estructura, escalabilidad, rendimiento y definición de esquema.
4. Expongan sus decisiones en una breve puesta en común o muro colaborativo.

Caso 1: Red Social (Facebook, Twitter)

Tipo de base de datos NoSQL sugerido: Base de datos de grafos (Ej: Neo4j, ArangoDB)
Justificación:
- Representación natural de relaciones (amigos, seguidores, conexiones).
- Consultas eficientes sobre caminos y relaciones.
- Buena escalabilidad para relaciones complejas.

Caso 2: Ecommerce (Amazon, MercadoLibre)

Tipo de base de datos NoSQL sugerido: Documental (Ej: MongoDB, CouchDB)
Justificación:
- Esquema flexible para representar productos, usuarios y pedidos.
- Permite almacenar documentos JSON con estructura variable.
- Fácil de escalar horizontalmente.
- Alto rendimiento en operaciones de lectura y escritura.

Caso 3: IoT (Internet de las cosas)

Tipo de base de datos NoSQL sugerido: Base de datos por columnas (Ej: Cassandra, HBase)
Justificación:
- Ideal para gestionar grandes volúmenes de datos en tiempo real.
- Optimizada para escritura masiva desde múltiples sensores.
- Alta disponibilidad y tolerancia a fallos.
- Escalable horizontalmente.

Caso 4: Análisis financiero o Big Data

Tipo de base de datos NoSQL sugerido: Columnar (Ej: Apache Cassandra, ScyllaDB)
Justificación:
- Excelente para agregaciones y análisis masivos.
- Optimización por columnas acelera consultas analíticas.
- Alta escalabilidad para grandes volúmenes de datos históricos.

Resumen

Caso de uso	Tipo NoSQL	Ejemplo DB	Justificación principal
Red Social	Grafos	Neo4j, ArangoDB	Relaciones complejas y consultas de redes
Ecommerce	Documental	MongoDB	Flexibilidad y velocidad de desarrollo
IoT	Columnar	Cassandra	Alta escritura y disponibilidad
Análisis financiero	Columnar	Cassandra	Consultas analíticas y volumen de datos

Diseñá tu primera base NoSQL

Contexto: 🙌

Aplicar lo aprendido sobre estructura y tipos de NoSQL a un caso práctico ayuda a fijar conceptos y desarrollar criterio técnico.

Consigna: ✍️

Diseña la estructura de una base de datos NoSQL para una app de mensajería en tiempo real (tipo WhatsApp). Debe incluir el tipo de base de datos elegido, cómo se modelarían los datos y cómo se escalaría.

Tiempo🕐: 30 minutos

🧭 Paso a paso:

1. Determinen qué tipo de NoSQL usarían (por ejemplo: clave-valor para sesiones, documental para mensajes, grafos para contactos).
2. Esbozar un modelo de datos simplificado (colecciones, claves, valores, nodos).
3. Expliquen cómo manejarían la escalabilidad (horizontal vs vertical).
4. Compartan con la clase cómo cada decisión resuelve los desafíos del caso.

1. Tipo de base de datos NoSQL a utilizar

Uso combinado de varios tipos NoSQL:

Función de la app	Tipo de base NoSQL	Ejemplo de motor
Sesiones de usuario	**Clave-valor**	Redis
Mensajes entre usuarios	**Documental**	MongoDB
Lista de contactos / redes	**Grafos (opcional)**	Neo4j (si hay relaciones complejas)

2. Modelo de datos simplificado

Colección: Usuarios

Json
CopiarEditar

```
{
  "id_usuario": "u123",
  "nombre": "Juan",
  "telefono": "+56911111111",
  "estado": "Disponible"
}
```

Colección: Mensajes

json
CopiarEditar

```
{
  "id_mensaje": "m456",
  "remitente_id": "u123",
  "destinatario_id": "u456",
  "mensaje": "Hola, ¿cómo estás?",
  "timestamp": "2025-07-26T10:15:00Z",
  "leído": false
}
```

Clave-Valor para sesiones (en Redis):

bash
CopiarEditar

```
session:u123 -> token_sesion_abc456
```

3. Escalabilidad

- **Escalabilidad horizontal:**
    - Se usarán **shards** en MongoDB para distribuir la colección de mensajes por usuario o región.
    - Redis se escala mediante **particionado y replicación** para manejar múltiples sesiones simultáneas.
- **Tolerancia a fallos:**
    - Réplicas en MongoDB para alta disponibilidad.
    - Redis configurado en modo clúster con nodos maestro-esclavo.
- **Flexibilidad de esquema:**
    - Permite añadir campos personalizados sin romper la estructura existente, útil para añadir funcionalidades como notas de voz o archivos adjuntos.

4. Justificación de decisiones

- Redis asegura **respuesta inmediata** en la autenticación y recuperación de sesión.
- MongoDB permite **almacenamiento eficiente de mensajes estructurados** y consultas rápidas por usuario y timestamp.
- El diseño facilita agregar nuevas funcionalidades sin migraciones complejas.
- El uso de NoSQL permite **alta concurrencia**, ideal para apps con millones de usuarios activos simultáneamente.