

Clase N°2 - Bases de Datos II

Tema de Hoy: **Bases de Datos NoSQL**

HOY :

Hoy nos enfocamos en bases de datos NoSQL.

Repasamos conceptos de SQL para comparar.

Exploramos el universo de NoSQL y su utilidad en apps modernas.

Trabajaremos con MongoDB y su interfaz gráfica MongoCompass.

Bases de Datos Relacionales

SQL: Estructura y organización clásica

Características

- Organiza la información en tablas, como una hoja de Excel.
- Cada tabla tiene filas (registros) y columnas (atributos).
- Utiliza SQL para manejar los datos: SELECT, INSERT, UPDATE, DELETE.
- Se usa cuando la información es bien estructurada y tiene relaciones claras entre entidades.

¿Qué es NoSQL?

Flexibilidad y velocidad para datos modernos

Características

- No requiere estructuras fijas ni relaciones entre tablas.
- Guarda los datos como documentos (ej: JSON), clave-valor, columnas o grafos.
- Ideal para apps web, móviles y redes sociales.
- Escalable horizontalmente, ideal para manejar millones de usuarios.



¿Qué significa escalar?

Escalar verticalmente = mejorar la **misma máquina**

- Ejemplo: agregar más RAM, CPU, SSD.
- Límite físico. Más caro.

Escalar horizontalmente = usar **más máquinas en paralelo**

- Ejemplo: repartir datos en 10 servidores distintos.
- Se usa en apps grandes (WhatsApp, MercadoLibre)
- NoSQL facilita este tipo de escalado.

¿En qué se diferencian SQL y NoSQL?

Característica	SQL (Relacional)	NoSQL (No Relacional)
Estructura	Tablas	Documentos (JSON), otros tipos
Esquema	Fijo	Flexible
Relaciones	Sí (JOINS)	No siempre
Escalabilidad	Vertical	Horizontal
Lenguaje	SQL	API propia / sintaxis distinta
Ejemplos	PostgreSQL, MySQL, SQLite	MongoDB, Firebase, Redis

¿En qué se diferencian SQL y NoSQL?

Profundicemos en las diferencias clave que distinguen a estas dos potentes bases de datos.

Estructura

SQL organiza los datos en tablas, como una planilla Excel, con filas y columnas definidas.

NoSQL utiliza documentos JSON, pares clave-valor, grafos o columnas, ofreciendo una mayor diversidad de modelos.

Esquema

SQL requiere un esquema fijo: debes definir la estructura de las columnas antes de insertar datos.

NoSQL es flexible, permitiendo que los documentos tengan estructuras variadas sin necesidad de un esquema predefinido.

Relaciones

En SQL, las relaciones entre tablas (como JOINs) son fundamentales para vincular datos.

En NoSQL, los datos suelen estar "embebidos" para mejorar la velocidad, aunque esto puede implicar duplicación de información.

Escalabilidad

SQL escala **verticalmente**, lo que significa mejorar el hardware del servidor existente (más RAM, CPU).

NoSQL escala **horizontalmente**, agregando más servidores para trabajar en paralelo.

Lenguaje

SQL usa un lenguaje estandarizado (SELECT, JOIN, INSERT) universalmente reconocido.

NoSQL depende del sistema; por ejemplo, MongoDB usa una sintaxis similar a JavaScript, mientras que Firebase usa SDKs o REST.

Ejemplos

- SQL: PostgreSQL, MySQL, SQLite
- NoSQL: MongoDB, Firebase, Redis

Historia de las Bases de Datos

Breve evolución histórica

Época	Evolución y contexto
Años 60:	Archivos planos (.txt, .csv)
Años 70:	Edgar Codd (IBM) propone modelo relacional
Años 80-90:	SQL se vuelve estándar (Oracle, MySQL, SQL Server)
Años 2000+:	Nace NoSQL por escalabilidad y apps modernas
Hoy:	Bases distribuidas, en la nube, orientadas a performance

Historia de las Bases de Datos

Breve evolución histórica

1960s – Archivos planos:

- Los datos se guardaban en archivos .txt, .csv.
- No había relaciones ni integridad. Acceso secuencial.
- Lenguaje COBOL. Ej: registros de empleados en tarjetas perforadas.

1970s – Bases relationales:

- Edgar Codd (IBM) crea el modelo relacional.
- Se introduce el lenguaje SQL (Structured Query Language).
- IBM System R fue uno de los primeros prototipos.

1980s – Bases comerciales:

- Oracle lanza su SGBD. Aparecen Ingres, DB2, Informix.
- SQL se convierte en estándar ANSI (1986).
- Aplicaciones empresariales comienzan a usarlas masivamente.

1990s – Bases para desarrolladores:

- MySQL y Microsoft Access se vuelven populares.
- Se integran con lenguajes como PHP, VB, Java.
- Uso masivo en sitios web.

2000s – Nacimiento de NoSQL:

- Google, Amazon, Facebook enfrentan retos de escalabilidad.
- Nacen BigTable, DynamoDB, y luego MongoDB (2009).
- Nuevas formas de guardar datos: documentos, claves, grafos.

2010s-2020s – Cloud y tiempo real:

- Bases como servicio (Firebase, Supabase, Mongo Atlas).
- Soporte para apps móviles y escalado automático.
- Ej: WhatsApp, Uber, Spotify usan soluciones cloud escalables.

¿Qué es un SGBD moderno?

SGBD = Sistema de Gestión de Bases de Datos

Es un software que permite **crear, modificar, gestionar, consultar y proteger** una base de datos de forma estructurada.

Más que guardar datos: funciones clave de un SGBD

Características

- Control de acceso y seguridad (roles, permisos)
- Backups automáticos
- Escalabilidad: crecer en usuarios sin romper nada
- Alta disponibilidad (si un nodo cae, otro responde)
- Monitoreo en tiempo real y optimización de consultas
- Ejemplos: PostgreSQL, Supabase, MongoDB Atlas

Principales SGBD de la actualidad



Tipos de bases de datos NoSQL

Tipologías dentro del mundo NoSQL

Tipo	Ejemplo	¿Para qué sirve?	Usado por
Documental	MongoDB	Apps web y móviles con estructura tipo JSON	Uber, eBay, MetLife
Documental	Firebase	Apps móviles en tiempo real (chat, delivery)	Duolingo, Alibaba, The New York Times
Clave-Valor	Redis	Caches, sesiones, likes	Twitter, GitHub, StackOverflow
Columnar:	Cassandra	Big Data, analítica masiva	Netflix, Spotify, Instagram
De Grafos:	Neo4j	Redes sociales, GPS, relaciones	LinkedIn, Airbnb, NASA

Casos donde NoSQL es mejor

¿Cuándo conviene usar NoSQL?

Cuando:

Cuando los datos cambian de forma todo el tiempo

Cuando hay millones de usuarios simultáneos

Cuando se prioriza velocidad sobre estructura

Cuando querés escalar horizontalmente

Ejemplos

Firebase: apps móviles en tiempo real (chat, delivery)

MongoDB: e-commerce con miles de productos dinámicos

Neo4j: redes sociales o sistemas de recomendación

Empezamos con MongoDB y MongoCompass

Primeros pasos en MongoDB

Instrucciones:

1 Abrimos MongoCompass

2 Creamos una base de datos

3 Creamos una colección

4 Insertamos documentos JSON como:

```
{  
  "nombre": "Sofía",  
  "edad": 22  
}
```

- Luego hacemos búsquedas simples y filtros { edad: { \$gt: 20 } }

Tarea práctica: tu primer proyecto con MongoDB

Vamos a aplicar todo lo aprendido hoy



Crea tu Base de Datos

Elige un tema libre (ej: películas, juegos, mascotas, recetas) y da vida a tu primer conjunto de datos en MongoDB.

Inserta tus Datos

Carga al menos 3 documentos, asegurándote de que sigan una estructura lógica dentro de una colección coherente.

Explora con Filtros

Realiza una consulta con filtro para mostrar los elementos que cumplan una condición específica (ej: edad mayor a X, tipo Y).



Sube a GitHub

Sube tu proyecto a GitHub. Puede ser un archivo .json con tus datos o el código que usaste con MongoCompass.

iComparte tu Logro!

Comparte el link de tu repositorio de GitHub en Classroom.

No se preocupen si no sale perfecto, lo importante es intentarlo!

Tu cuenta de GitHub

Guía Paso a Paso

1. Crear tu cuenta en GitHub

- Ingresá a <https://github.com>
- Hacé clic en "Sign up"
- Completá tu nombre de usuario, email y contraseña
- Confirmá tu cuenta con el código que llega por correo
- ¡Listo! Ya tenés tu cuenta de GitHub

2. Instalar Git en tu compu

- Windows: <https://git-scm.com/download/win>
- Durante la instalación, elegí las opciones por defecto.

3. Configurar Git con tu cuenta

Abrí tu terminal en VSCode y ejecutá estos comandos:

```
git config --global user.name "TuNombre"  
git config --global user.email "tucorreo@gmail.com"
```

Subí tu proyecto a GitHub desde VS Code



Comandos básicos de Git (para tu primer proyecto)

📌 Una vez que tengas tu proyecto abierto en VS Code:

1. Inicializá un repositorio Git:

```
git init
```

🔁 Esto convierte tu carpeta local en un repositorio de Git, preparado para el control de versiones.

2. Agregá todos los archivos al *staging area*:

```
git add .
```

✓ El `.` (punto) indica que se agregan todos los archivos y carpetas del directorio actual al área de preparación para el próximo commit.

3. Hacé tu primer *commit*:

```
git commit -m "Primer commit"
```

📝 El mensaje entre comillas ("Primer commit" en este caso) es personalizable y debe describir los cambios realizados. Este comando guarda una instantánea de tus archivos en el repositorio local.

4. Conectá tu proyecto local con GitHub:

En GitHub:

- Creá un nuevo repositorio vacío (sin README, .gitignore ni licencia).
- Copiá la URL HTTPS que te proporciona GitHub.

En VS Code (terminal):

```
git remote add origin https://github.com/tuusuuario/nombre-repo.git
```

Este comando enlaza tu repositorio local con el remoto en GitHub, nombrando a la conexión como "origin".

5. Subí tu proyecto a GitHub:

```
git branch -M main  
git push -u origin main
```

💡 La primera línea renombra tu rama principal a 'main' (estándar moderno). La segunda línea sube todos los cambios (commits) de tu rama local 'main' al repositorio remoto en GitHub ('origin'), y establece un seguimiento para futuras actualizaciones.

ⓘ 🧠 **Consejo:** Siempre que realices cambios significativos en tu código, usá los comandos `git add .`, `git commit -m "Tu mensaje"` y `git push` para mantener tu repositorio de GitHub actualizado y con un historial de versiones claro.

Personalizá tu perfil con un README



¿Querés que tu perfil de GitHub tenga una presentación visual como esta? 

<https://github.com/pietro923>

📌 Seguí estos pasos:



Visitá el Generador

Andá a <https://profile-readme-generator.com/es> para empezar.



Completá tus Datos

Indicá tu nombre, biografía, lenguajes, tecnologías y enlaces a tus redes o portfolio.



Elegí tu Estilo

Seleccioná el diseño y los componentes visuales que más te gusten para tu README.



Copiá el Código

Obtené el código generado en formato Markdown.



Creá un Repo en GitHub

En GitHub, creá un nuevo repositorio con **tu mismo nombre de usuario exacto** (ejemplo: pietro923).



Pegá el Contenido

Dentro de este nuevo repo, pegá el código en un archivo llamado: README.md.



Subí y Confirmá

Subí el archivo al repositorio o usá el botón "Upload" de GitHub.



¡Listo! Ahora tu perfil va a mostrar ese README cuando alguien lo visite.



Pro tip: podés actualizarlo cuando quieras para mostrar tus proyectos o experiencias nuevas.