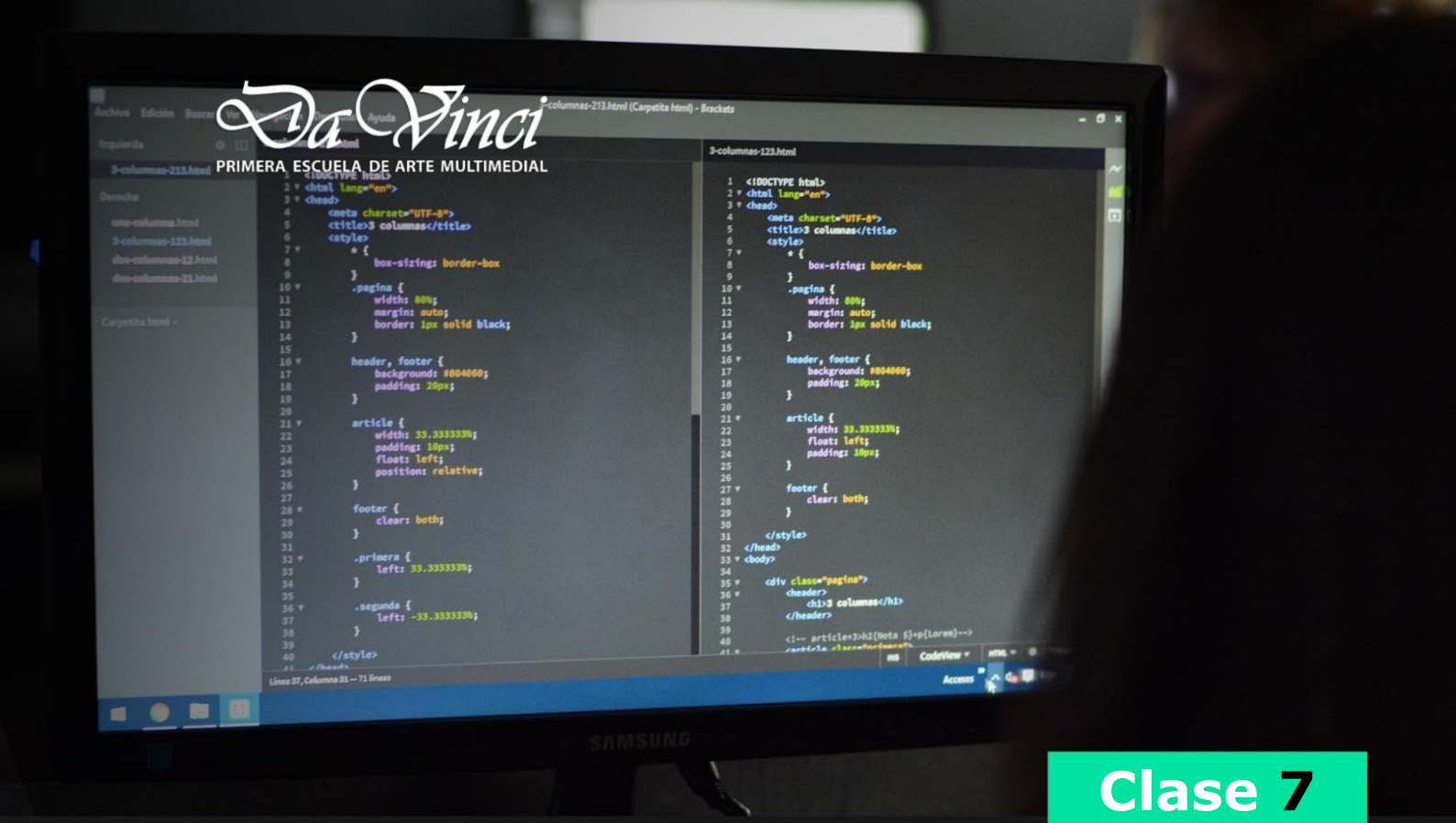

Clase 7

Diseño y Programación Web

Materia:
Aplicaciones Híbridas

Docente contenidista: MARCOS GALBÁN, Camila Belén

Bases de Datos

Para comparar bases de datos SQL (relacionales) y NoSQL (no relacionales), es importante entender sus características, ventajas y desventajas, así como los casos de uso ideales para cada tipo. A continuación, se presenta una comparación detallada:

SQL (Bases de Datos Relacionales)

Características:

- **Modelo Relacional**: Los datos se almacenan en tablas (filas y columnas).
- **Esquema Fijo**: Requiere un esquema predefinido y estructurado.
- **ACID**: Soporte completo para las propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad).
- **Consultas**: Utiliza SQL (Structured Query Language) para realizar consultas complejas.
- **Integridad Referencial**: Soporte para relaciones entre tablas mediante claves primarias y foráneas.

Ventajas:

- **Consistencia y Fiabilidad**: Garantiza transacciones consistentes y fiables.
- **Consultas Complejas**: Adecuado para operaciones complejas y análisis de datos.
- **Integridad de Datos**: Mantenimiento de la integridad referencial y restricciones de datos.
- **Herramientas y Ecosistema**: Amplio soporte de herramientas, documentaciones y comunidad.

Desventajas:

- **Escalabilidad Vertical**: Mayoritariamente escalabilidad vertical, lo cual puede ser costoso y limitado.

- **Esquema Rigido**: Menos flexible al manejar cambios de esquema.
- **Rendimiento**: Menor rendimiento en ciertos casos de uso, especialmente con grandes volúmenes de datos no estructurados.

Casos de Uso:

- **Sistemas Financieros**: Donde la consistencia y transacciones ACID son cruciales.
- **ERP y CRM**: Sistemas que requieren integridad y relaciones complejas.
- **Aplicaciones de Analytics**: Donde se necesitan consultas SQL complejas.

NoSQL (Bases de Datos No Relacionales)

Características:

- **Modelos Diversos**: Documentos, clave-valor, columnas y grafos.
- **Esquema Flexible**: No requiere un esquema fijo; los datos pueden ser semiestructurados o no estructurados.
- **Escalabilidad Horizontal**: Diseñado para escalar horizontalmente mediante particionamiento y replicación.
- **BASE**: Sigue el principio BASE (Basically Available, Soft state, Eventual consistency), aunque algunas bases de datos NoSQL también pueden soportar ACID.

Ventajas:

- **Flexibilidad**: Fácil manejo de datos no estructurados y semiestructurados.
- **Rendimiento y Escalabilidad**: Excelente rendimiento y escalabilidad horizontal.
- **Desarrollo Ágil**: Menos restricciones de esquema, lo que facilita cambios y desarrollo ágil.
- **Casos de Uso Específicos**: Optimizado para ciertos tipos de datos y aplicaciones.

Desventajas:

- **Consistencia Eventual**: En algunos sistemas, puede que no garantice consistencia inmediata.
- **Consultas Limitadas**: Menos capacidades para consultas complejas comparado con SQL.
- **Madurez**: Algunas tecnologías NoSQL son más nuevas y pueden tener menos documentación y herramientas maduras comparadas con SQL.

Casos de Uso:

- **Big Data y Análisis en Tiempo Real**: Manejo de grandes volúmenes de datos que requieren procesamiento rápido.
- **Aplicaciones Web y Móviles**: Que necesitan alta disponibilidad y escalabilidad.
- **Datos No Estructurados**: Como redes sociales, IoT, y contenido multimedia.

La elección entre SQL y NoSQL depende de los requisitos específicos del proyecto, incluyendo la naturaleza de los datos, la necesidad de escalabilidad, y los requisitos de consistencia y rendimiento. Mientras SQL es ideal para aplicaciones que requieren consistencia y relaciones complejas, NoSQL es más adecuado para aplicaciones que manejan grandes volúmenes de datos no estructurados y necesitan escalabilidad horizontal. A continuación, se presentan casos reales y razones por las que se podría optar por una tecnología sobre la otra.

Casos Reales y Razones para Usar SQL

Sistemas Financieros

- **Ejemplo**: Bancos y sistemas de pago en línea como PayPal.
- **Razón**: La consistencia y la integridad de los datos son cruciales. SQL garantiza transacciones ACID, asegurando que todas las operaciones de transferencia de dinero sean completas, coherentes y duraderas. La integridad referencial entre tablas asegura que no haya inconsistencias en los datos de las transacciones financieras.

Aplicaciones ERP y CRM

- **Ejemplo**: SAP, Salesforce.

- **Razón**: Estas aplicaciones manejan grandes cantidades de datos estructurados y requieren integridad referencial y consultas complejas. SQL permite manejar eficientemente las relaciones entre tablas y realizar análisis complejos sobre los datos.

Sistemas de Análisis de Datos

- **Ejemplo**: Plataformas de análisis como Tableau que se conectan a bases de datos SQL como PostgreSQL, MySQL.
- **Razón**: SQL proporciona potentes capacidades de consulta que son esenciales para realizar análisis de datos detallados y generar informes. Las propiedades ACID garantizan que los datos analizados sean consistentes y precisos.

Casos Reales y Razones para Usar NoSQL

Redes Sociales

- **Ejemplo**: Facebook, Twitter.
- **Razón**: Estas plataformas manejan grandes volúmenes de datos no estructurados y semiestructurados (publicaciones, comentarios, interacciones). NoSQL permite escalabilidad horizontal para manejar millones de usuarios y sus interacciones en tiempo real. Las bases de datos de documentos (como MongoDB) y de grafos (como Neo4j) son particularmente útiles para modelar y consultar las relaciones entre usuarios.

Sistemas de Recomendación

- **Ejemplo**: Netflix, Amazon.
- **Razón**: Estos sistemas necesitan procesar y analizar rápidamente grandes volúmenes de datos de usuario para proporcionar recomendaciones personalizadas. Bases de datos de columna (como Apache Cassandra) ofrecen alta disponibilidad y rendimiento para estas operaciones.

Aplicaciones de IoT

- **Ejemplo**: Smart Home systems, sistemas de monitoreo industrial.
- **Razón**: Los dispositivos IoT generan grandes cantidades de datos en tiempo real que deben ser almacenados y procesados rápidamente. Las bases de datos de clave-valor (como Redis) y de tiempo (como InfluxDB) son adecuadas para manejar estos flujos de datos y proporcionar análisis en tiempo real.

Resumen de Comparaciones

Característica	SQL	NoSQL
Modelo de Datos	Relacional (tablas)	Diversos (documentos, clave-valor, grafos, columnas)
Esquema	Fijo y estructurado	Flexible y dinámico
Consistencia	Fuerte (ACID)	Eventual (BASE), algunos soportan ACID
Escalabilidad	Vertical	Horizontal
Consultas	SQL (complejas, potentes)	API específicas (limitadas)
Integridad Referencial	Sí	No (en la mayoría de los casos)
Casos de Uso	Transacciones financieras, ERP, análisis de datos	Redes sociales, IoT, sistemas de recomendación

MongoDB

MongoDB es una base de datos NoSQL que almacena datos en un formato de documento similar a JSON, conocido como BSON (Binary JSON). Fue desarrollada por MongoDB Inc. y se lanzó por primera vez en 2009. A diferencia de las bases de datos relacionales tradicionales, MongoDB no utiliza tablas y filas, sino que almacena datos en documentos flexibles que pueden contener una variedad de tipos de datos.

Es una base de datos NoSQL orientada a documentos que se utiliza para almacenar grandes volúmenes de datos no estructurados y semiestructurados.

Características

Modelo de Datos Flexible

Los documentos en MongoDB no requieren un esquema fijo, lo que permite cambios dinámicos en la estructura de los datos sin necesidad de migraciones complejas.

Escalabilidad Horizontal

MongoDB es diseñado para escalar horizontalmente mediante particionamiento (sharding), lo que permite distribuir datos a través de múltiples servidores.

Consultas Potentes

Ofrece un lenguaje de consulta rico que permite realizar búsquedas avanzadas, agregaciones y manipulaciones de datos.

Replicación

Soporta replicación mediante conjuntos de réplicas (replica sets) para asegurar alta disponibilidad y redundancia de datos.

Índices

Permite la creación de índices en campos de documentos para mejorar el rendimiento de las consultas.

Agregación

Incluye un marco de agregación potente para realizar operaciones de procesamiento de datos como sumas, promedios y uniones.

Ventajas de MongoDB

Flexibilidad en el Esquema

Ideal para aplicaciones con estructuras de datos dinámicas y cambiantes.

Escalabilidad

Capaz de manejar grandes volúmenes de datos mediante la adición de más nodos al clúster.

Desarrollo Ágil

Rápido desarrollo e iteración debido a la flexibilidad del esquema.

Alta Disponibilidad

Replicación automática para asegurar que los datos estén disponibles incluso en caso de fallos de hardware.

Consultas y Agregaciones Potentes

Capacidad de realizar consultas y agregaciones complejas de manera eficiente.

Casos de Uso Comunes

Aplicaciones Web y Móviles

MongoDB es popular en aplicaciones que requieren un almacenamiento flexible y escalable, como plataformas de redes sociales, sistemas de gestión de contenido y aplicaciones de comercio electrónico.

Big Data y Análisis en Tiempo Real

Adecuado para aplicaciones que necesitan manejar grandes volúmenes de datos y realizar análisis en tiempo real.

Internet de las Cosas (IoT)

Utilizado para almacenar datos generados por dispositivos IoT que pueden tener estructuras de datos diversas y cambiantes.

Sistemas de Recomendación

Implementación de sistemas que proporcionan recomendaciones personalizadas basadas en el comportamiento del usuario y otros datos.

Arquitectura de MongoDB

Documentos y Colecciones

- **Documentos**: MongoDB almacena datos en documentos BSON (Binary JSON), que son similares a los objetos JSON. Un documento es una estructura de datos que contiene pares de clave-valor.

- **Colecciones**: Los documentos se agrupan en colecciones. Una colección es equivalente a una tabla en bases de datos relacionales, pero no requiere un esquema fijo.

Instancias y Clústeres

- **Instancia de MongoDB**: Una instancia de MongoDB se refiere a un único proceso de MongoDB.
- **Replica Set**: Un grupo de instancias de MongoDB que mantienen los mismos datos. Proporciona alta disponibilidad mediante la replicación automática de datos.
- **Sharding**: Técnica de escalabilidad horizontal que divide los datos en varios servidores o clústeres para distribuir la carga de trabajo.

Modelo de Datos

Documentos BSON

- Los documentos BSON pueden contener múltiples tipos de datos, como cadenas, números, listas y subdocumentos.
- Ejemplo de un documento BSON:

```
{
    "nombre": "Juan",
    "edad": 30,
    "direccion": {
      "calle": "Calle Falsa 123",
      "ciudad": "Springfield"
    }
  }
```

Colecciones

- Las colecciones no requieren un esquema fijo, lo que permite almacenar documentos con diferentes estructuras dentro de la misma colección.

Lista de comandos útiles

Mostrar todas las bases de datos (db)

- show dbs

Mostrar la db actual

- db

Moverme o crear una db

- use db-name

Eliminar db

- db.dropDatabase()

Crear una colección

- db.createCollection("users")

Eliminar una colección

- db.users.drop()

Crear un índice

- db.users.createIndex({"edad": 1})

Eliminar in índice

- db.users.dropIndex("edad")

Insertar documentos a una colección

- db.users.insertOne({nombre: "Pepe"}) // insertar uno
- db.users.insertMany({nombre: "Juan"}, {nombre: "Camila"}) // Insertar varios

Buscar todos los documentos de una colección

- db.users.find()

Buscar solo uno

- db.users.findOne()

Busqueda especifica

- db.users.find({nombre: "Camila"})

  los que el nombre sea Camila

- db.users.find({ nombre: { $regex: "ca" } });

  los que el nombre contenga "ca" (regex es de expresiones regulares, acá utilizando las mismas se amplían las opciones)