CQRS

Cuando se utiliza el patrón **CQRS (Segregación de Responsabilidad de Comandos y Consultas)** en el desarrollo de software, uno de los desafíos comunes es mantener sincronizadas las **bases de datos de lectura y escritura**. Dado que el patrón CQRS separa los modelos para lectura y escritura, garantizar la consistencia entre estos modelos puede ser complejo. Aquí te presento diferentes estrategias para sincronizar las bases de datos:

1. Event Sourcing (Fuente de Eventos)

- **Cómo funciona**: Con el uso de Event Sourcing, cada cambio en el estado de la aplicación se almacena como un evento, y el sistema puede reproducir estos eventos para reconstruir el estado tanto en la base de datos de lectura como en la de escritura. El lado de escritura en el patrón CQRS almacena los eventos, mientras que el lado de lectura escucha estos eventos y actualiza el modelo de lectura en consecuencia.
- **Ventajas**:
    - Garantiza la consistencia entre las bases de datos de escritura y lectura, ya que ambas se derivan de la misma fuente de verdad (los eventos).
    - Soporta análisis históricos y auditorías, dado que todos los eventos se almacenan.
- **Desafíos**:
    - Requiere un almacén de eventos y una implementación más compleja.
    - Puede ser difícil gestionar la versión de los eventos a lo largo del tiempo.

2. Captura de Datos de Cambio (CDC)

- **Cómo funciona**: CDC captura los cambios de la base de datos de escritura y los propaga a la base de datos de lectura. Esto se puede implementar usando disparadores en la base de datos, registros de transacciones o herramientas externas (como Debezium).
- **Ventajas**:
    - Propagación automática de cambios desde la base de datos de escritura a la de lectura.
    - Relativamente fácil de implementar en sistemas que usan bases de datos relacionales tradicionales.
- **Desafíos**:
    - Puede introducir cierta latencia en la sincronización.
    - Requiere un manejo cuidadoso de los cambios en la base de datos y las transformaciones para el modelo de lectura.

3. Mensajería Asíncrona

- **Cómo funciona**: Los comandos procesados por el modelo de escritura pueden publicar eventos a un intermediario de mensajes (por ejemplo, RabbitMQ, Kafka). El

modelo de lectura se suscribe a estos eventos y actualiza su base de datos en consecuencia.

- **Ventajas**:
  - Acoplamiento débil entre las bases de datos de lectura y escritura.
  - Puede escalar bien en sistemas distribuidos.
- **Desafíos**:
  - La consistencia eventual es común, lo que significa que los modelos de lectura y escritura pueden estar temporalmente desincronizados.
  - Requiere un manejo cuidadoso del orden de los eventos, la desduplicación y los reintentos.

4. Replicación de Base de Datos

- **Cómo funciona**: La replicación es una técnica a nivel de base de datos donde los cambios en la base de datos de escritura se replican automáticamente a la base de datos de lectura. Esto se puede hacer utilizando replicación maestro-esclavo, donde la base de datos de escritura es el maestro y la de lectura es el esclavo.
- **Ventajas**:
  - Simple de implementar cuando se utilizan bases de datos relacionales que admiten replicación.
  - Proporciona sincronización en tiempo casi real.
- **Desafíos**:
  - Flexibilidad limitada para transformar u optimizar datos para consultas de lectura.
  - Pueden surgir problemas de latencia y consistencia durante los retrasos en la replicación.

5. Patrón de Outbox Transaccional

- **Cómo funciona**: Cuando ocurre una actualización en el modelo de escritura, se realiza una entrada en una tabla "outbox" dentro de la misma transacción. Un proceso en segundo plano lee la tabla de outbox y propaga el evento al modelo de lectura. Esto asegura que los eventos se publiquen de manera confiable después de que la transacción se haya confirmado.
- **Ventajas**:
  - Garantiza atomicidad entre las operaciones de la base de datos y la publicación de eventos.
  - Previene la pérdida de datos debido a fallos en la transacción.
- **Desafíos**:
  - Requiere la gestión del procesamiento del outbox y puede introducir algo de latencia en la sincronización.

6. Escrituras Duales (Actualizaciones Sincrónicas)

- **Cómo funciona**: Durante una operación de escritura, ambas bases de datos, de escritura y de lectura, se actualizan de manera sincrónica dentro de la misma transacción o proceso.
- **Ventajas**:

    - Garantiza una fuerte consistencia entre los modelos de escritura y lectura.
- **Desafíos**:
    - Introduce complejidad y posibles cuellos de botella de rendimiento, ya que ambas bases de datos deben actualizarse en tiempo real.
    - Acoplamiento fuerte entre los modelos de lectura y escritura, lo que va en contra de los principios de CQRS.

7. Lógica Directa de Sincronización de Bases de Datos

- **Cómo funciona**: Las operaciones de escritura actualizan el modelo de escritura y luego invocan la lógica de sincronización para actualizar el modelo de lectura mediante código de aplicación personalizado.
- **Ventajas**:
    - Flexible y permite transformaciones personalizadas entre los modelos de escritura y lectura.
- **Desafíos**:
    - Complejo de mantener, especialmente a medida que la aplicación crece.
    - Difícil de garantizar la consistencia y manejar errores entre las actualizaciones de bases de datos.

8. Vistas Materializadas

- **Cómo funciona**: La base de datos de lectura utiliza vistas materializadas que se actualizan periódicamente a partir de la base de datos de escritura. Esto es aplicable principalmente para casos donde hay muchas lecturas y no es crítica la frescura de los datos.
- **Ventajas**:
    - Eficiente para generar resultados de consulta precomputados.
- **Desafíos**:
    - Los datos pueden volverse obsoletos, y no se garantiza una sincronización en tiempo real.

9. Procesamiento por Lotes

- **Cómo funciona**: La sincronización se realiza en lotes, donde los cambios en la base de datos de escritura se procesan periódicamente y se actualizan en la base de datos de lectura.
- **Ventajas**:
    - Adecuado para casos en los que no se requiere sincronización en tiempo real.
    - Puede manejar grandes volúmenes de cambios de manera eficiente.
- **Desafíos**:
    - Los datos pueden volverse obsoletos y puede haber mayor latencia entre las actualizaciones.

1. Event Sourcing (Fuente de Eventos)

- **Ejemplo del mundo real**:
  Un sistema financiero que necesita registrar cada cambio en el saldo de una cuenta bancaria. En lugar de almacenar el saldo actual, se almacenan todos los eventos que afectan al saldo (depósitos, retiros), lo que permite reconstruir el estado de la cuenta en cualquier momento.
- **Tecnologías**:
  - **Axon Framework** para la gestión de eventos y Event Sourcing.
  - **Kafka Streams** o **Eventuate** para procesar y manejar los eventos.
  - **Cassandra** o **MongoDB** para almacenar los eventos.

2. Captura de Datos de Cambio (CDC)

- **Ejemplo del mundo real**:
  En un sistema de gestión de inventarios, los cambios en el stock de productos en la base de datos de escritura se capturan y propagan automáticamente a la base de datos de lectura para asegurar que los usuarios siempre vean los datos actualizados.
- **Tecnologías**:
  - **Debezium** para capturar cambios en bases de datos.
  - **Kafka** para la mensajería entre bases de datos de lectura y escritura.
  - **PostgreSQL** o **MySQL** para implementar CDC mediante registros de transacciones.

3. Mensajería Asíncrona

- **Ejemplo del mundo real**:
  En una aplicación de reservas de viajes, cuando un usuario reserva un vuelo, se publica un evento a un broker de mensajes, que luego activa la actualización de servicios relacionados como hoteles y alquiler de coches. Estos servicios se suscriben a los eventos y actualizan sus bases de datos de manera asíncrona.
- **Tecnologías**:
  - **RabbitMQ** o **Kafka** para mensajería asíncrona.
  - **Spring Cloud Stream** para integración con servicios de mensajería.
  - **Axon Framework** para manejar eventos y acciones compensatorias.

4. Replicación de Base de Datos

- **Ejemplo del mundo real**:
  En una plataforma de medios sociales, las publicaciones de los usuarios pueden replicarse en varias bases de datos de lectura geográficamente distribuidas para permitir una alta disponibilidad y lecturas rápidas en cualquier parte del mundo.
- **Tecnologías**:

    - **MySQL** o **PostgreSQL** para bases de datos con replicación maestro-esclavo.
    - **Cassandra** para replicación en sistemas distribuidos.
    - **Amazon RDS** con soporte de replicación automática.

5. Patrón de Outbox Transaccional

- **Ejemplo del mundo real**:
  En un sistema de ecommerce, cuando se completa una compra, se registra un evento en una tabla "outbox" junto con la actualización de la orden. Un proceso en segundo plano lee la tabla "outbox" y publica el evento a otros servicios, asegurando que no se pierdan datos en caso de fallo.
- **Tecnologías**:
    - **Spring Boot** para la implementación de servicios.
    - **Kafka** para la publicación de eventos.
    - **MySQL** o **PostgreSQL** para el almacenamiento en la tabla "outbox".

6. Escrituras Duales (Actualizaciones Sincrónicas)

- **Ejemplo del mundo real**:
  En un sistema de reservas hoteleras, cuando un usuario reserva una habitación, la base de datos de escritura actualiza la reserva, y simultáneamente, la base de datos de lectura se actualiza para reflejar la disponibilidad actual.
- **Tecnologías**:
    - **PostgreSQL** o **MySQL** para las bases de datos de lectura y escritura.
    - **Spring Boot** para gestionar la lógica de sincronización.

7. Lógica Directa de Sincronización de Bases de Datos

- **Ejemplo del mundo real**:
  En una aplicación de banca online, cuando un cliente realiza una transferencia, la base de datos de escritura se actualiza con los detalles de la transacción, y un código personalizado sincroniza la base de datos de lectura para mostrar el saldo actualizado al cliente.
- **Tecnologías**:
    - **Spring Boot** para implementar la lógica de sincronización personalizada.
    - **MongoDB** o **PostgreSQL** para las bases de datos de lectura y escritura.

8. Vistas Materializadas

- **Ejemplo del mundo real**:
  En una aplicación de informes de ventas, las vistas materializadas precomputadas muestran los datos de ventas agregados y actualizados periódicamente, lo que permite consultas rápidas sin afectar el rendimiento de la base de datos de transacciones.
- **Tecnologías**:
    - **PostgreSQL** con soporte de vistas materializadas.

    - **Oracle** o **SQL Server** para bases de datos relacionales que soportan vistas materializadas.

9. Procesamiento por Lotes

- **Ejemplo del mundo real**:
  En un sistema de facturación, los datos de las transacciones del día se sincronizan en lotes al final del día desde la base de datos de escritura a la base de datos de lectura para generar informes diarios de ingresos.
- **Tecnologías**:
    - **Spring Batch** para el procesamiento por lotes.
    - **Hadoop** o **Spark** para manejar grandes volúmenes de datos en procesamiento por lotes.