UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

POSGRADOS

Maestría en

Ingeniería de software

Asignatura:

Patrones de Integración Empresarial

Tarea Nro. 05:

Shared Database

Autor(es):

Cabascango Garcia Amanda Elizabeth
Calo Catota Carlos Edison
Fuentes Espinoza Pablo Gustavo
Guaman Guaman Saul German
Guerra Campuzano Cesar Hugo
Rengel Rivera Mateo Santiago
Vela Moya Christian Eduardo

Patron Shared Database

El patrón Shared Database es un mecanismo de integración en el que múltiples aplicaciones acceden y manipulan una única base de datos centralizada. Este enfoque permite que las aplicaciones compartan datos de manera directa, eliminando la necesidad de intermediarios como servicios web o mensajería. En lugar de cada aplicación tener su propia base de datos aislada, comparten una infraestructura de datos común.

Ventajas.

- **Simplicidad**
  Es una solución sencilla y directa, especialmente para sistemas pequeños o aplicaciones estrechamente relacionadas. Reduce la complejidad de la infraestructura al eliminar la necesidad de múltiples bases de datos y mecanismos de sincronización.
- **Consistencia de datos**
  Garantiza la consistencia de los datos, ya que todas las aplicaciones acceden a la misma fuente de información. Facilita la implementación de transacciones que abarcan múltiples aplicaciones.
- **Rendimiento**
  En ciertos escenarios, puede ofrecer un buen rendimiento al evitar la sobrecarga de la comunicación entre diferentes sistemas.
- **Reducción de Latencia**
  El acceso directo a la base de datos puede reducir la latencia en comparación con otros mecanismos de integración. Menor Costo Inicial: No se requieren herramientas adicionales de integración, lo que puede reducir los costos iniciales.

Desventajas

- **Acoplamiento Fuerte**
  Crea un fuerte acoplamiento entre las aplicaciones, lo que dificulta la evolución independiente y la escalabilidad. Los cambios en el esquema de la base de datos pueden afectar a todas las aplicaciones que la utilizan.
- **Problemas de Concurrencia**
  Aumenta el riesgo de conflictos de concurrencia y bloqueos, especialmente en sistemas con alta carga. Requiere una gestión cuidadosa de las transacciones y el control de acceso.
- **Riesgos de Seguridad**
  Un fallo de seguridad en una aplicación puede comprometer los datos de todas las aplicaciones que comparten la base de datos. Dificulta la implementación de políticas de seguridad granulares.
- **Escalabilidad Limitada**

Dificulta el escalamiento de aplicaciones individuales, ya que todas dependen de la capacidad de la base de datos compartida

Casos de Uso Comunes y Riesgos Asociados

- Casos de Uso Comunes
    - **Sistemas Legacy**: En sistemas antiguos donde la migración a una arquitectura más moderna es costosa o compleja
    - **Aplicaciones Internas:** En entornos donde las aplicaciones son desarrolladas y mantenidas por el mismo equipo.
    - **Prototipos**: Para prototipos rápidos donde la simplicidad y la rapidez de implementación son prioritarias
- Riesgos asociados
    - **Acoplamiento Fuerte**: Cambios en la estructura de la base de datos pueden requerir modificaciones en todas las aplicaciones que la utilizan.
    - **Problemas de Concurrencia**: Conflictos de bloqueo y problemas de rendimiento pueden surgir si no se maneja adecuadamente la concurrencia.
    - **Mantenimiento Complejo**: La evolución del sistema puede ser más compleja debido al acoplamiento y la dependencia directa de la base de datos.

Alternativas

- **Event-Driven Architecture (EDA)**
    - **Descripción**: Las aplicaciones se comunican mediante eventos, lo que reduce el acoplamiento y permite una mayor flexibilidad.
    - **Ventajas**: Mayor desacoplamiento, escalabilidad mejorada, y facilidad para implementar nuevas funcionalidades
    - **Desventajas**: Mayor complejidad en la implementación y gestión de eventos.
- **Microservicios**
    - **Descripción**: Las aplicaciones se dividen en servicios pequeños e independientes que se comunican a través de APIs.
    - **Ventajas**: Alta escalabilidad, desacoplamiento, y facilidad para implementar y desplegar nuevas funcionalidades
    - **Desventajas**: Mayor complejidad en la gestión de servicios y en la comunicación entre ellos.

Conclusión

El patrón **Shared Database** puede ser una solución efectiva para la integración de aplicaciones en ciertos escenarios, especialmente cuando se busca simplicidad y consistencia de datos. Sin embargo, es importante considerar sus desventajas y riesgos, como el acoplamiento fuerte y los problemas de concurrencia. Alternativas modernas como la arquitectura orientada a eventos y los microservicios pueden ofrecer mayor flexibilidad y escalabilidad, aunque a costa de una mayor complejidad en la implementación.

Referencias

Microservices Shared Database: https://microservices.io/patterns/data/shared-database.html/
Pattern Shared Database: https://www.enterpriseintegrationpatterns.com/patterns/messaging/SharedDataBaseIntegration.html/
Event Driver Architecture: https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/event-driven