Iteración 5 – Sistemas Transaccionales

Análisis de la implementación de transacciones distribuidas

1. **Apoyándose en la documentación disponible en las siguientes referencias, relacionadas con implantación de transacciones distribuidas usando la interfaz XA de X/Open, y con la implantación de la operación distribuida con colas de mensajes, plantee restricciones existentes para los dos sitios que se van a montar en la Aplicación BancAndes. Un ejemplo de restricción podría ser que alguno de los dos sitios, o los dos, no dispone de la interfaz XA.**

   El estándar XA en informática es una especificación de X/Open que permite el procesamiento de transacciones distribuidas, su objetivo es permitir acceder a múltiples recursos dentro de una misma transacción, conservando las propiedades ACID, XA implementa *two phase commit* (2PC) para garantizar consistencia sobre los recursos, pues bien o hacen *commit* o de lo contrario *rollback.* Dentro de las restricciones de esta implementación encontramos que: ambos sistemas deben implementar la interfaz XA, otra restricción es que al XA al implementar 2PC las transacciones y operaciones son bloqueantes sobre los recursos, esto significa que los procesos compiten por los bloqueos de recursos y de no lograrlo deben esperar a que estos sean liberados, esto tiene un impacto en la latencia de la transacción, otra restricción es que si el coordinador no tiene una buena disponibilidad y está constantemente caído no podrá atender con eficiencia los commit o rollback de los participantes y de esta forma los recursos quedarían bloqueados indefinidamente.

   Por otro lado, una cola de mensajes es un componente de software que permite la comunicación inter-procesos, donde el emisor almacena el mensaje en la cola y el receptor lo saca de esta (no necesariamente al tiempo). Una de las principales restricciones de esta implementación es la memoria de la cola, esto se ve reflejado en la cantidad de mensajes que puede almacenar, otra restricción es la seguridad en términos de autorización indicando qué procesos pueden acceder a los mensajes, otra restricción es en términos de las políticas de entrega pues un mensaje puede ser entregado más de una vez, también se puede considerar como restricción que los sistemas y procesos que desean interactuar con la cola deben de conocer el protocolo. Todas estas restricciones y complicaciones se solucionan usando buenas decisiones de diseño a la hora de la implementación.

Restricciones existentes para los dos proyectos

A continuación se listarán ciertas restricciones que podrían ser encontradas en la implementación de transacciones distribuidas usando el estándar XA y Java Message Service (JMS):

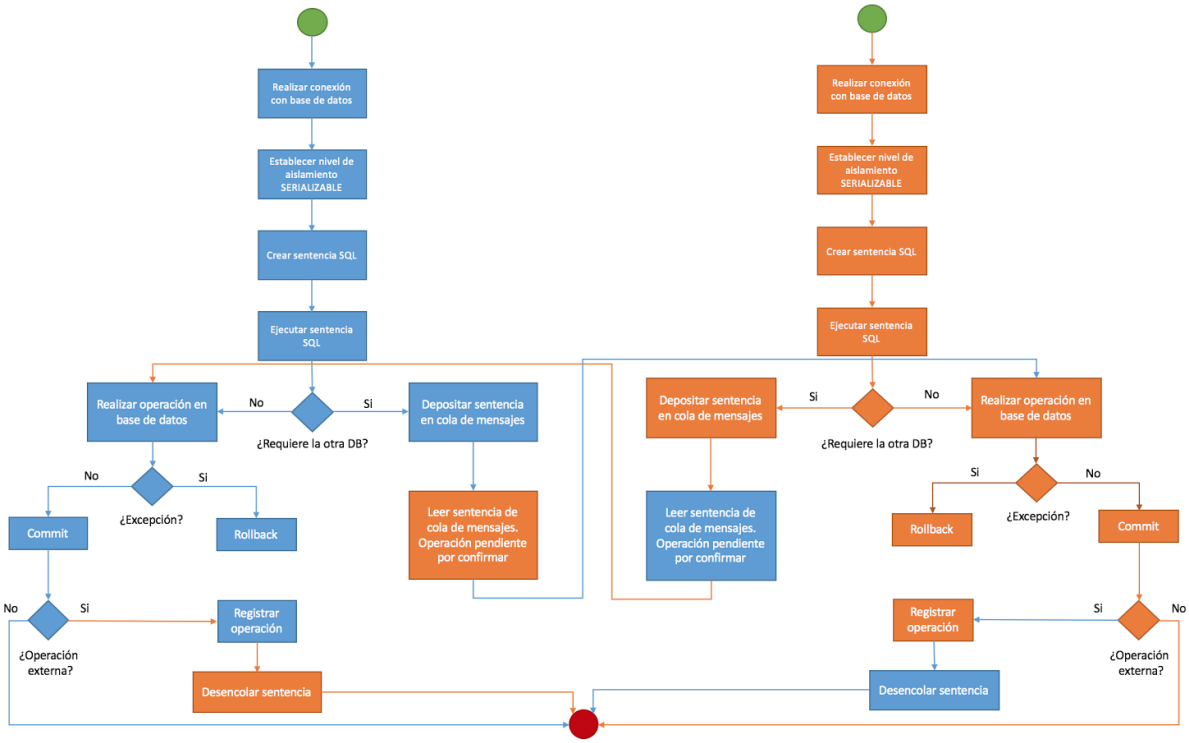
- Uno o ambos sitios no han implementado la interfaz XA de alguna forma.

- Alguna de las colas de mensajes no se encuentra bien definida para una o ambas aplicaciones, y una o ambas aplicaciones no tienen acceso a esta.
- El protocolo de comunicación a través de las colas de mensajes no está definido e implementado en alguna o ambas aplicaciones.
- El API de JMS puede estar mal configurado en alguna o ambas aplicaciones. Por ejemplo el API puede no tener las garantías de entrega correctas o el modo de confirmación incorrecto, por lo que las características ACID que se esperan lograr de la comunicación podrían no lograrse.
- El orden de commits puede estar mal definido en JMS, lo que puede dañar las características de transaccionalidad deseadas.
- Los recursos pueden estar mal definidos en una o ambas aplicaciones, por lo que los cambios y consultas en las bases de datos no podrían ser llevados a cabo.
- Las propiedades de los topics dentro de JMS deben estar igualmente definidos en ambas para que los cliente que estén subscritos reciban información de la misma manera.

2. Ajuste la arquitectura de la aplicación web desarrollada en las iteraciones 1 a 4 para implementar los nuevos requerimientos.

Para esta iteración no fue necesario realizar ajustes a la arquitectura en términos de modelo de entidades, pero el modelo relacional se ajustó adicionando un atributo ID en las tablas comprometidas en la realización de los requerimientos funcionales solicitados, este ID que identifica la aplicación que está interactuando con el sistema y de esta forma saber quien es el autor de las transacciones realizadas para evitar así confusiones. **// TO DO Mostrar Tablas**

3. Especifique la lógica del proceso de RF15. Utilice un diagrama de flujo como formalismo.



4. Analice las estrategias que deben ser desarrolladas para el cumplimiento del requerimiento solicitado, tanto para el caso de uso de colas de mensajes como de two phase commit (aunque no sea implementado). Compare los resultados que se obtendrían al implementar las dos estrategias.

Estrategia Two Phase Commit

Bajo esta estrategia el proceso comienza cuando el nodo coordinador manda la solicitud de pedido a ambas unidades. Al revisar la solicitud, cada unidad del programa debe verificar si posee los recursos necesarios para satisfacer la solicitud, si esto no es así, puede que combinando los recursos varias unidades puedan satisfacer el mismo. Si este es el caso, las subunidades deben hacer commit de toda la transacción, reservando los elementos usados y registrando las fases de transaccion pendientes. Si todos estos commits son exitosos el nodo coordinador puede hacer commit de toda la transaccion para que esta sea registrada y exitosa. Si ambas unidades no pueden satisfacer juntas la transaccion es responsabilidad del nodo coordinador de hacer la transaccion pendiente o imposible de realizar. Si alguna de las unidades durante el proceso antes descrito hace rollback, el nodo coordinador debe asegurarse de darle la orden a todas las unidades de hacer rollback y de registrar la trasaccion como pendiente o insatisfecho.

Estrategia cola de mensajes

La unidad que registró la transaccion debe revisar sus recursos y ver si es capaz de realizar el pedido por si misma, si esto es así debe realizar la transacción por cuenta propia; de lo contrario debe encolar la solicitud de transaccion y, o bien los recursos que hacen falta o los recursos que dispone. La segunda unidad eventualmente lee la solicitud, y procede a verificar sus recursos, desencolando el mensaje de la primera unidad revisa si entre sus recursos y los otros la solicitud puede ser satisfecha. Si este es el caso hace commit y envía un mensaje que confirme el commit. La primera unidad posteriormente leerá dicho mensaje y hará commit por su parte. Si ambas unidades no pueden satisfacer el pedido la segunda unidad deberá informar esto en la cola de mensajes. Al leer el mensaje de fallo la primera unidad deberá hacer rollback de los cambios que realizó y registrar la transaccion como pendiente. Si la segunda unidad hace rollback debe informarle a la primera unidad para que esta mantenga la transaccion como pendiente.

Análisis de impacto de estrategias

La estrategia global elegida es cola de mensajes, dado que se considera que el API JMS ofrece más utilidad y garantías para controlar la transaccionalidad en las operaciones. Se considera además que utilizando JMS la implementación de balanceo de carga es más fácil. En la estrategia global se incluirán dos colas de mensajes, una en la que la aplicación del grupo X será tratada como WebApp 1 y la del grupo Y como WebApp 2 y otra en la que ocurrirá lo contrario.

Los mensajes encolados deberán tener una estructura como la siguiente:

- Si se trata de una solicitud el mensaje en la cola deberá ser: “RF<<n° de requerimiento”
- Los mensajes de commit y rollback deberán ser un mensaje simple de: “commit” y “rollback” respectivamente.

- Las respuestas a las consultas o los mensajes que involucren datos de la tabla deberán seguir un formato tipo <<atributo:valor>> en donde todos los atributos y valores estén separados por comas a la vez.