**Restricciones XA**

1. Ninguno de los dos proyectos posee la interfaz XA, lo cual dificulta en gran manera su implementación para este sistema conjunto
2. El implementar XA es mucho más costoso ya que usa two phase commit, el cual aunque es más efectivo requiere mucho más tiempo para implementar y también en el momento de realizar las actualizaciones en ambas bases es más costoso
3. Al no estar implementada la interfaz XA, si se implementase toda la aplicación debería de cambiar

**Restricciones X/OPEN**

1. La implementación de una cola de mensajes a partir de JMS la cual no garantiza en casos mayores que todas las transacciones puedan realizar commit

**Análisis de requerimientos.**

Primero se partirá al entender las principales características y diferencias entre los dos métodos llamados cola de mensajes y TPC, de esta forma poder comparar y realizar un análisis efectivo y sólido sobre cómo afectarían los diferentes métodos a cada uno de los requerimientos de la iteración. Por otra parte, se podrá decir cuál es el mejor método para implementar según los datos obtenidos por el análisis.

**Cola de mensajes.**

Una cola de mensajes es un contenedor que alberga mensajes mientras están en tránsito. El administrador de colas de mensajes actúa como intermediario en la transmisión de un mensaje desde su origen hasta su destino.

Las características principales de las aplicaciones que utilizan las técnicas de mensajes en colas son:

- No hay conexiones directas entre programas.

- La comunicación entre programas puede realizarse sin dependencia del tiempo.

- El trabajo se puede realizar mediante pequeños programas independientes.

- La comunicación puede ser dirigida por sucesos.

- Las aplicaciones pueden asignar una prioridad a un mensaje.

- La seguridad.

- La integridad de los datos.

- El soporte para la recuperación.

**Commit en dos fases o TPC.**

Es un [algoritmo distribuido](https://es.wikipedia.org/w/index.php?title=Algoritmo_distribuido&action=edit&redlink=1) que permite a todos los nodos de un [sistema distribuido](https://es.wikipedia.org/wiki/Sistema_distribuido) ponerse en acuerdo para hacer [commit](https://es.wikipedia.org/wiki/Commit) a una [transacción](https://es.wikipedia.org/wiki/Transacci%C3%B3n).

### Fase de petición del Commit

1. El coordinador envía un mensaje **consulta para commit** a todos los participantes.
2. Los participantes ejecutan la transacción hasta el punto donde ellos serán preguntados para realizar commit. Ellos pueden escribir una entrada a su log undo y una entrada a su log redo.
3. Cada participante responde con un mensaje **acuerdo** si la transacción tuvo éxito, o un mensaje **abortar** si falló la transacción.
4. El Coordinador espera hasta que tenga un mensaje de cada participante.

### Fase Commit

#### Éxito

Si el coordinador de esta pagina recibió un mensaje **acuerdo** de todos los participantes durante la fase de petición de commit:

1. El coordinador envía un mensaje **commit** a todos los participantes.
2. Cada participante completa la operación, y libera todos los bloqueos y recursos mantenidos durante la transacción.
3. Cada participante envía un **reconocimiento** al coordinador.
4. El coordinador completa la transacción cuando ha recibido todos los reconocimientos.

#### Fracaso

Si algún participante envió un mensaje **abortar** durante la fase de petición de commit:

1. El coordinador envía un mensaje **[rollback](https://es.wikipedia.org/wiki/Rollback" \o "Rollback)** a todos los participantes.
2. Cada participante deshace la transacción usando el log undo, y libera los recursos y bloqueos mantenidos durante la transacción.
3. Cada participante envía un **reconocimiento** al coordinador.
4. El Coordinador completa la transacción cuando han sido recibidos los reconocimientos.

**Desventajas y problemas.**

- Mayor desventaja del protocolo commit en dos fases es el hecho de que es un protocolo bloqueante.

- Un nodo estará bloqueado mientras esté esperando un mensaje.

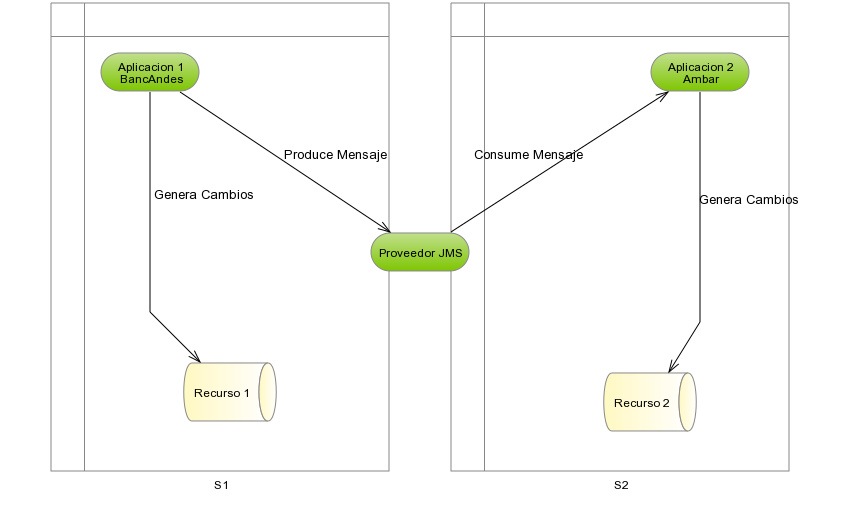
- Si el coordinador falla permanentemente, algunos participantes nunca resolverán sus transacciones.

- Si el coordinador está permanentemente caído, el participante bloqueará indefinidamente, a menos que pueda obtener la decisión global commit/abort desde uno de los participantes.

**Análisis.**

Teniendo claras las principales características y diferencias entre los dos tipos de métodos, es posible partir a analizar cada uno de los requerimientos funcionales desde la perspectiva de cada uno de ellos, garantizando así obtener la forma que se considere óptima para implementar.

**RF15 - REGISTRAR OPERACIÓN SOBRE CUENTA – v22**



En el momento de terminar el proceso sobre la(s) cuenta(s), se debe informar al usuario con un número de confirmación y el estado de la solicitud. El número de confirmación es único en BancAndes2. Si alguna de las unidades no puede confirmar de forma inmediata una operación, dicha operación debe quedar en estado pendiente por confirmar. La operación queda en estado registrado cuando puede ser procesada en su totalidad. Cuando se involucran cuentas de dos entidades bancarias, cada una de las entidades confirma su participación en la operación y debe asegurar que, una vez registrada la operación los saldos y demás información relevante son correctas.

**Cola de mensajes.**

Partiendo de que en este requerimiento existe un estado de pendiente por confirmar, es eficiente utilizar el método de cola de mensajes, ya que este facilita las conexiones entre las bases de datos al no ser conexiones directas entre las bases, además de la comunicación entre las dos aplicaciones se podría realizar entre sucesos. Esto, facilitaría el manejo de las transacciones, tanto exitosas y no exitosas. Por otra parte, el requerimiento pide que se pueda asegurar que los saldos e información bancaria después de las transacciones sean correctos, por lo que esto se puede facilitar en la medida que la cola de mensajes proporciona integridad de los datos y un soporte de recuperación. Por lo cual garantiza un buen rendimiento en la verificación.

**TPC.**

En este caso seria muy difícil solucionar el estado de la solicitud, ya que en este método todas las transacciones deben ocurrir en cadena de otra manera se realizaría rollback. Por otra parte existe la posibilidad de que las transacciones se detengan o se bloqueen durante un momento, ya que existe la posibilidad de que se realicen transacciones que bloqueen a alto nivel los recursos. Esto conduciría a que las otras transacciones tengan que esperar. Además si existe alguna falla en el coordinador, existe la posibilidad que algunos clientes no puedan realizar transacciones o que los recursos se bloqueen en el peor de los casos. La única ventaja que ofrece este método es que el usuario está completamente seguro del éxito de sus transacciones si el TPC lo es.

**Conclusión RF15**

Los resultados al valorar la cola de mensajes fueron superiores a los mostrados por el TPC, ya que la cola de mensajes se ajusta más a lo que el requerimiento necesita. Por otra parte, el TPC no cumple con las necesidades del método y es muy propenso a tener fallas y problemas, además de que es un método bloqueante.

**RF16 - ASOCIAR CUENTA DE PERSONA NATURAL A CUENTA DE PERSONA JURÍDICA2**

**Cola de mensajes.**

En este requerimiento al implementar la cola de mensajes se podría facilitar esta transacción debido a que no se generan bloqueos de las tablas o partes de estas, lo cual nos asegura que podemos asociar una persona natural de una aplicación a una persona jurídica de otra aplicación, sin realizar bloqueos en las dos bases de datos. Si esto llegase a ocurrir podría ocasionar que no se ejecutaran transacciones en las dos bases de datos.

**TPC.**

En este caso podríamos presenciar bloqueos de los recursos en las dos bases de datos, además que si llegara a ocurrir un error en el coordinador en una transacción que involucre las dos bases de datos las dos aplicaciones quedarían sin poder realizar transacciones hasta que se ejecute un roll back.

**Conclusión RF16**

La cola de mensajes es lo más indicado para estas transacciones, ya que respeta la información, además de que una transacción de este tipo puede afectarse según su prioridad para tener un mejor manejo de las transacciones. Por otra parte el TPC puede generar problemas bloqueantes y es solo una transacción, aunque este método pueda garantizar la transacción, no es viable ya que este requerimiento no tiene una gran prioridad para asumir un gran riesgo.

**RF17 - PAGAR NÓMINA DE PERSONA JURÍDICA2**

**Cola de mensajes.**

Al implementar la cola de mensajes tendríamos como resultado un listado de transacciones de consignación las cuales pertenecerían a los pagos de los empleados, este método es excelente para este tipo de transacciones ya que puede realizar una por una en cierto orden, lo que me facilitaría un mejor control en las transacciones y también la posibilidad de recuperar la información para la revisión del resultado final.

**TPC.**

En este caso se utilizaría una gran cantidad de información para realizar todos los pagos a la ves en las dos bases de datos, como lo habían dicho antes este método bloquea la información mientras realiza la confirmación de todas las transacciones, lo que tendría como resultado un gran bloqueo de información de las dos aplicaciones mientras se realizan todos los pagos simultáneamente.

**Conclusión RF17**

En conclusión para este método no es recomendable el TPC debido a la gran cantidad de datos que bloquearía mientras se realiza en pago de todos los empleados, por esta razón es mejor usar la cola de mensajes.

**Conclusión final.**

Teniendo en cuanta que un banco procesa una gran cantidad de transacciones diarias no es posible el bloquear algún tipo de recurso en el momento de realizar operaciones específicas. Por esta razón la cola de mensajes es el método apropiado para una aplicación como esta ya que garantiza un orden en las transacciones, además de brindar seguridad, integridad de los datos, sostenimiento de las tablas relacionales y además tiene un soporte de recuperación para realizar revisiones.

Bibliografía.

<https://es.wikipedia.org/wiki/Commit_de_dos_fases#Desventajas>

<http://www-01.ibm.com/support/knowledgecenter/SSFKSJ_7.5.0/com.ibm.mq.pro.doc/q002630_.htm?lang=es>