

Iteración 5 - FestivAndes

Análisis de Base de Datos Distribuida

Ana C. Fandiño de la Hoz, David A. Cortes Vesga, Ivan D. Chavarro Garzón,
James H. R. Lake Franco, Paula J. Alvarado Zabala
201326407, 201423363, 201423319, 201531545, 201313033
Sistemas Transaccionales
Universidad de los Andes, Bogotá, Colombia
{ac.fandino10, da.cortes11, id.chavarro10, jhlake, pj.alvarado10}@uniandes.edu.co

Fecha de presentación: Mayo 21 de 2017

Índice

Índice

1. Análisis de la implementación de transacciones distribuidas
 - 1.1. Restricciones de los Sitios
 - 1.2. Ajuste la arquitectura de la aplicación
 - 1.3. Lógica del proceso de RF15
 - 1.4. Análisis de estrategias del requerimiento RF15
2. Especificación e implementación de transacciones distribuidas
 - 2.1. Alistamiento de las bases de datos y de servicios
 - 2.2. Requerimiento RF15 con colas de mensajes
 - 2.3. Requerimiento RF16 con colas de mensajes
 - 2.4. Requerimiento RF16 con two phase commit
 - 2.5. Requerimientos RFC13 y RFC14
 - 2.6. Análisis del impacto de las estrategias de transacciones distribuidas

1. Análisis de la implementación de transacciones distribuidas

1.1. Restricciones de los Sitios

<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 tres sitios que se van a montar en la Aplicación FestivAndes . Un ejemplo de restricción podría ser que alguno de los dos sitios, o los tres, no disponen de la interfaz

XA.>

Para las transacciones distribuidas se tiene el JMS(java message Service) donde se puede definir interfaces comunes que permiten a programas escritos en Java comunicarse con otras implementaciones de mensajería. A la vez esto brinda una comunicación asincrónica y mensajería segura. Para esto se tiene 2 modelos de comunica que puede ser tanto “punto a punto” como “publicación/subscripción”.

En la implementación de los JMS se hace con la participación en un pool de conexiones. Para este caso de las colas de mensajes, se almacenan en una cola tipo FIFO y los consumidores leen de manera sincrónica de la cola.

Como primera restricción es que si en algunas de las solicitudes de transacciones llegan a encontrarse en un deadlock, no se pueda realizar ninguna otra operación durante ese intervalo de tiempo. Esta decisión nos va a garantizar que se cumpla lo elementa el en sistema de transacciones que es el ACID(Atomicidad, Consistencia, Aislamiento y Durabilidad).

1.2. Ajuste la arquitectura de la aplicación

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

Los nuevos requerimientos hacen necesario o implican que las bases de datos son distribuidas y que estas se pueden acceder por más de una aplicación al igual que ser modificada.

El concepto de distribución de base de datos permite que se presenten estas arquitecturas, de todas formas es necesario hacer nuevas implementaciones para asegurar el buen funcionamiento de todo el sistema.

Entre estos se encuentra el hecho de que las diferentes aplicaciones deben trabajar con un mismo tipo de objeto unificado. Es decir que las distintas tablas deberían tener un mismo formato para un objeto (los mismos atributos, constraints, etc.....) y a su vez que las distintas aplicaciones tengan objetos iguales (los atributos y tipos de estos), así la aplicación se comunica como si se comunicara con su base de datos de forma tradicional, como en las anteriores iteraciones.

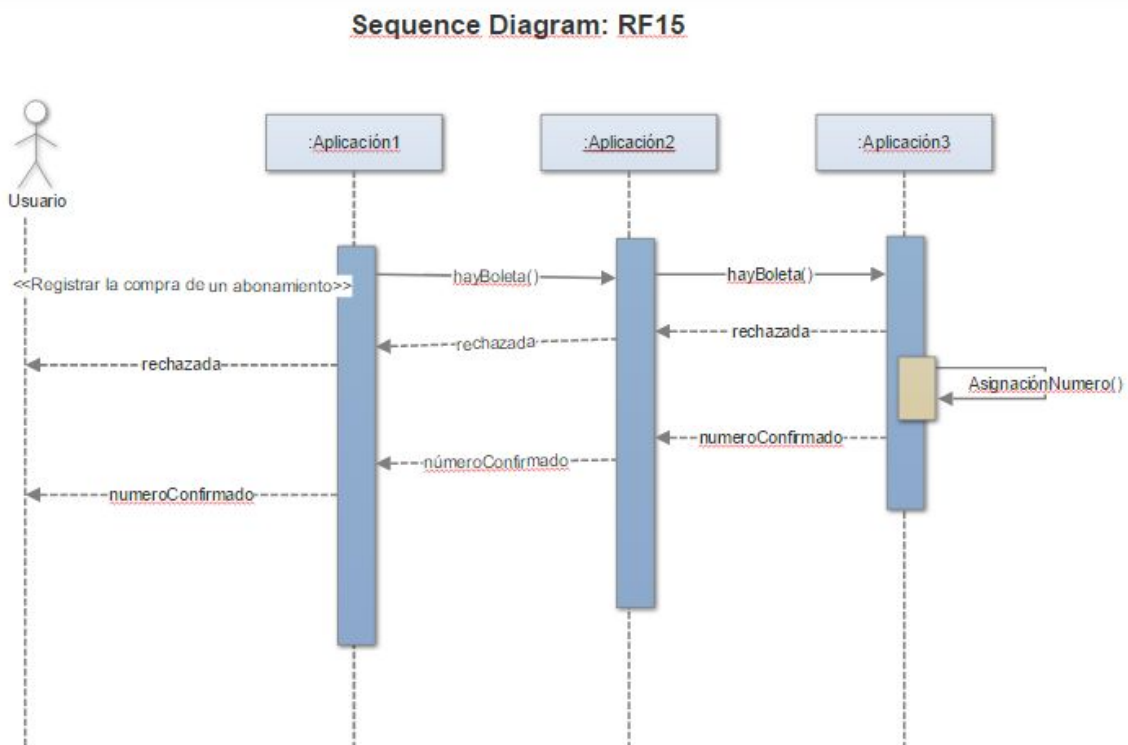
Para lograr esto fue necesario modificar las bases de datos y la lógica de las aplicaciones para poder llegar a tal punto de consistencia de la información.

UML:

1.3. Lógica del proceso de RF15

<(8%) Especifique la lógica del proceso de RF15. Utilice un diagrama de secuencia como formalismo.>

- 1) Se comienza con el estado de rechazada
- 2) Cada unidad indica la boleta que puede asumir
- 3) La entrega de un número de confirmación y el estado de su solicitud al usuario.



1.4. Análisis de estrategias del requerimiento RF15

<(15%) 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. Compare los resultados que se obtendrían al implementar cada una de las dos estrategias.>

2. Especificación e implementación de transacciones distribuidas

2.1. Alistamiento de las bases de datos y de servicios

<(4 %) Asegúrese de tener una copia de la base de datos disponible para cada subgrupo. Para ello, utilicen las cuentas de Oracle de cada uno de los integrantes.>

Asegúrese de que cada subgrupo tiene disponibilidad de hacer uso de los servicios representados por las tres unidades de negocio. >

Para permitir que las distintas aplicaciones se comuniquen de manera adecuada con la base de datos, fue necesario que las bases de datos fueran modificadas para que las aplicaciones se pudieran comunicar con estas de manera adecuada.

2.2. Requerimiento RF15 con colas de mensajes

<Implemente y pruebe el requerimiento RF15 utilizando colas de mensajes como estrategia de comunicación.>



```
WildFly 10.x [JBoss Application Server Startup Configuration] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (23/05/2017, 7:29:55 p.m.)

hread 1-8) WFLYIIOP0009: CORBA ORB Service started
javax.jms.Topic

RMQDestination(destinationName='appl', topic(permanent, amqp)', amqpExchangeName='festival.test', amqpRoutingKey='festival.abonamientos', amqpQueueName='appl')
declare RabbitMQ exchange for topic destination RMQDestination(destinationName='appl', topic(permanent, amqp)', amqpExchangeName='festival.test', amqpRoutingKey='fest
RMQDestination(destinationName='appl', topic(permanent, amqp)', amqpExchangeName='festival.test', amqpRoutingKey='festival.abonamientos', amqpQueueName='appl')

appl

javax.jms.Topic

RMQDestination(destinationName='appl', topic(permanent, amqp)', amqpExchangeName='festival.test', amqpRoutingKey='festival.abonamientos.appl', amqpQueueName='appl')
declare RabbitMQ exchange for topic destination RMQDestination(destinationName='appl', topic(permanent, amqp)', amqpExchangeName='festival.test', amqpRoutingKey='fest
RMQDestination(destinationName='appl', topic(permanent, amqp)', amqpExchangeName='festival.test', amqpRoutingKey='festival.abonamientos.appl', amqpQueueName='appl')

appl

Done!
```

2.3. Requerimiento RF16 con colas de mensajes

<(12%) Implemente y pruebe el requerimiento RF16 utilizando colas de mensajes como estrategia de comunicación. Cada unidad de negocio debe definir y especificar claramente cuáles son sus reglas para la cancelación de las funciones. >

2.4. Requerimiento RF16 con two phase commit

<(12%) Implemente y pruebe el requerimiento RF16 utilizando two phase commit como estrategia de comunicación.>

2.5. Requerimientos RFC13 y RFC14

<(12%) Especifique e implante los requerimientos restantes (RFC13 y RFC14) con la estrategia transaccional indicada. >

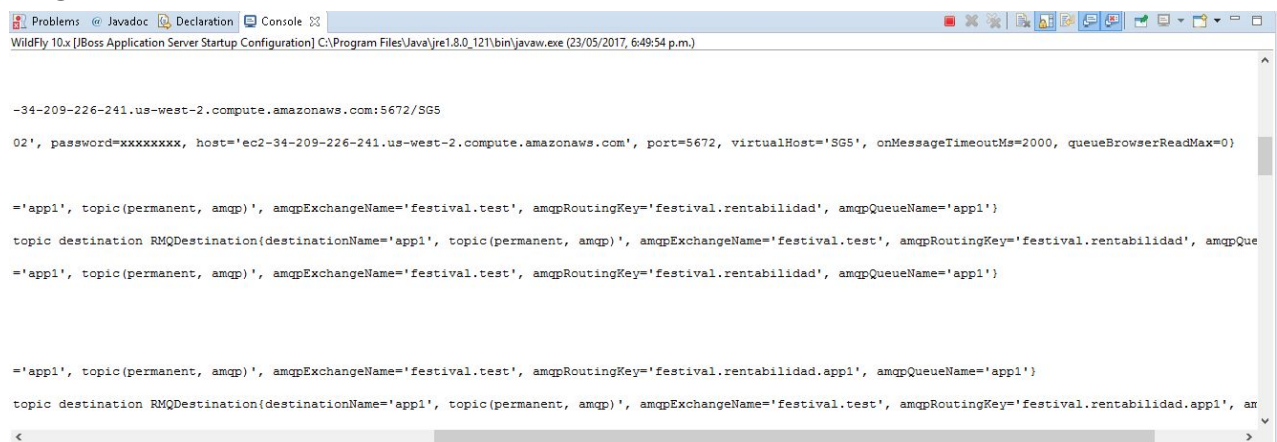
RFC13



```
WildFly 10.x [JBoss Application Server Startup Configuration] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (23/05/2017, 8:03:25 p.m.)

RMQDestination(destinationName='appl', topic(permanent, amqp)', amqpExchangeName='festival.test', amqpRoutingKey='festival.funciones', amqpQueueName='appl')
declare RabbitMQ exchange for topic destination RMQDestination(destinationName='appl', topic(permanent, amqp)', amqpExchangeName='festival.test', amqpRoutingKey='f
RMQDestination(destinationName='appl', topic(permanent, amqp)', amqpExchangeName='festival.test', amqpRoutingKey='festival.funciones', amqpQueueName='appl')
:appl
javax.jms.Topic
RMQDestination(destinationName='appl', topic(permanent, amqp)', amqpExchangeName='festival.test', amqpRoutingKey='festival.funciones.app1', amqpQueueName='appl')
declare RabbitMQ exchange for topic destination RMQDestination(destinationName='appl', topic(permanent, amqp)', amqpExchangeName='festival.test', amqpRoutingKey='f
RMQDestination(destinationName='appl', topic(permanent, amqp)', amqpExchangeName='festival.test', amqpRoutingKey='festival.funciones.app1', amqpQueueName='appl')
:appl
Done!
```

RFC14



```
WildFly 10.x [JBoss Application Server Startup Configuration] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (23/05/2017, 6:49:54 p.m.)

-34-209-226-241.us-west-2.compute.amazonaws.com:5672/SG5
02', password=xxxxxxx, host='ec2-34-209-226-241.us-west-2.compute.amazonaws.com', port=5672, virtualHost='SG5', onMessageTimeoutMs=2000, queueBrowserReadMax=0)

='appl', topic(permanent, amqp)', amqpExchangeName='festival.test', amqpRoutingKey='festival.rentabilidad', amqpQueueName='appl')
topic destination RMQDestination(destinationName='appl', topic(permanent, amqp)', amqpExchangeName='festival.test', amqpRoutingKey='festival.rentabilidad', amqpQue
='appl', topic(permanent, amqp)', amqpExchangeName='festival.test', amqpRoutingKey='festival.rentabilidad', amqpQueueName='appl')

='appl', topic(permanent, amqp)', amqpExchangeName='festival.test', amqpRoutingKey='festival.rentabilidad.app1', amqpQueueName='appl')
topic destination RMQDestination(destinationName='appl', topic(permanent, amqp)', amqpExchangeName='festival.test', amqpRoutingKey='festival.rentabilidad.app1', ar
```

2.6. Análisis del impacto de las estrategias de transacciones distribuidas

<(10%) Analice el impacto de las estrategias de transacciones distribuidas para cada requerimiento adicional solicitado, de forma independiente. Analice cuál es la estrategia global más indicada para FestivAndes de acuerdo con sus reglas de negocio y las condiciones esperadas de operación, en el cumplimiento de todos sus requerimientos como integrador de las unidades de negocio. Justifique plenamente su análisis.>

RFC13:

Este requerimiento hace referencia a todas las funciones existentes, de los tres proveedores de FestivAndes. Para esto fue necesario que hubiera una consistencia global entre las tres aplicaciones y su funcionamiento unificado; esto implica que fue necesario hacer una modificación a la base de datos para que al momento de consultar la misma sentencia sirviera

para las tres aplicaciones y esta retornara los mismo elementos. De la misma forma que fue necesario hacer modificaciones en las tablas hubo que hacer cambios en la lógica de las aplicaciones.

Uno de los principales cambios, al ser distribuida, es que la respuesta debe contener además el proveedor de la función (cuál de las tres aplicaciones la ofrece). El impacto de esto es que el tiempo de consulta aumenta (ya que se consulta sobre las tres tablas), pero no es mucho.

RFC14:

Para este requerimiento es importante comprender que a pesar que las tres aplicaciones tienen la misma estructura relacional, no posee la misma información. De esta forma puede suceder que una compañía no exista en todas las tablas, a pesar de que este caso se presente el requerimiento debe funcionar de manera adecuada.

Para esto se debió hacer un caso de tiempo límite en la búsqueda de la compañía y su rentabilidad.

Esta distribución afecta el rendimiento, puesto que si en un principio la búsqueda solo se tenía que hacer en una grupo de tablas, ahora se debe hacer en más de uno, en este caso en tres ya que cada aplicación tiene su propio grupo. Así aumentando el tiempo tres veces.

RFC15:

Para un abonamiento puede estar compuesto por entradas que estén ofrecidas por distintas aplicaciones. En esto se puede presentar un problema en el caso de que dos o más aplicaciones ofrecen la misma función y por ende las mismas boletas.

Cuando por medio de una de las aplicaciones se hace un abonamiento este debe pasar por cada grupo de tablas asegurando que sea posible adquirir las entradas, estas asignarlas a ese abono y que dejen de estar disponibles.

Al dos aplicaciones tener las mismas entradas, es necesario que al momento de agregar una que exista en otra aplicación esta también sea agregada a ese abono y que a su vez deje de estar disponible.

Para esto se resolvió distribuir las peticiones de una aplicación a las otras, es decir que en el caso de que se quiera crear un abonamiento los procesos sobre las entradas (encontrarlas y modificarlas) son también procesados por las otras dos aplicaciones. Esto es posible gracias a que los grupos de tablas de cada aplicación tienen la misma arquitectura y en el caso de que no exista tal función y por ende esa entrada, el mensaje de error puede ser ignorado, pero a su vez se asegura que en el caso de que exista, esta sea modificada (ocupada).

RFC16:

Este requerimiento tiene como función eliminar una compañía y a su vez todas las funciones que tuviera programadas, esto es hecho por un administrador pero a su vez es necesario que si una compañía existe en dos o más aplicaciones también sea eliminada ahí.

Para esto se implementa una solución muy similar a la del requerimiento 15.

En el caso que una compañía sea eliminada desde cualquier aplicación los mismos procesos se llevan a cabo en el resto de las aplicaciones, tanto la búsqueda como eliminar cualquiera de las funciones que esta tenga asociada. En el caso de que tal compañía no exista, los errores que sean enviados por la base de datos pueden ser ignorados.

Esto, al igual que con el requerimiento 15, es posible gracias a que todas las aplicaciones tienen la misma arquitectura para sus tablas, de esta forma una operación sobre un grupo de tablas de una aplicación tiene la misma funcionalidad en cualquiera de las otras.

RNF9:

Este es la distribución de la base de datos. Que se pueda ejecutar desde cualquier sitio (aplicación) y la información consistente en todos los grupos de tablas.

Esto requirió hacer cambios en la arquitectura de las tablas y el diseño lógico de las aplicaciones.

Por parte de las tablas fue necesario que todos los grupos de tablas tuvieran la misma arquitectura, así una sola sentencia sirviera para todas y que a su vez retornan y se les insertará los mismos elementos.

Por parte de la lógica de las aplicaciones fue necesario que se modifican los objetos con los que funcionaba, pues de la misma forma que las tablas era necesario que estuvieran compuestas por el mismo diseño lógico, así todas las aplicaciones recibieran información con la construyan mismos objetos y viceversa para la modificación de las tablas.

Por otro lado fue necesario hacer modificaciones para que estas se pudieran comunicar entre sí, mas no una con la base de datos de las otras, pues cada grupo de tablas solo es accedido por la aplicación que a la que pertenecen.

RNF10:

La disponibilidad se ve afectada directamente por la distribución, esto se debe a que una aplicación puede recibir peticiones de otra para modificar sus tablas, pero una aplicación no puede modificar las tablas de otra.

Esto implica que en el caso de que una de las aplicaciones no esté en funcionamiento no se pueden llevar a cabo procesos que involucren comunicación entre las tres bases de datos, pues como no se conoce la información de todas las tablas, los resultados y respuestas tendrían una alta probabilidad de estar erróneos.