Arquitectura de Proyectos de IT

Apunte:

Introducción a MQ y conceptos de mensajería

Autores:
Patricio Echagüe
patricioe@gmail.com

Ing. Gastón Escobar
gescobar@gmail.com

Versión: 0.1

Octubre, 2005

Índice

1. Objetivo

El objetivo de este documento es introducir al lector al concepto de Colas de mensajes a la vez de explicar los conceptos para importantes de esta tecnología. Para cumplir dicho objetivo hemos optado por utilizar un producto en particular, MQ Series de IBM.

2. Introducción

Un sistema de mensajería permite la comunicación entre aplicaciones a través de distintas redes, o dentro de un mismo sistema. MQ brinda una API para que los programas clientes reciban y envíen mensajes desde y hacia las colas (queues) sin poseer una conexión privada y dedicada para ellos. Es decir, un sistema de mensajería permite la comunicación entre aplicaciones de forma indirecta. Esto se logra mediante el envío de mensajes a un administrador de mensajes, al cual están asociadas las aplicaciones, y es este último quien se encarga de administrarlos y distribuirlos según diversas configuraciones.
El programador no especifica una aplicación de destino a la hora de enviar mensajes, sino que especifica el nombre de una cola.
Una aplicación puede tener una o mas colas de entrada y una o diversas colas de salida.
El Arquitecto no debe preocuparse acerca de la tecnología existente en la aplicación destino, o si la aplicación destino no se encuentra disponible (en el caso de los mensajes asincrónicos), sino que el motor de colas se encarga de reenviar el mensaje si la misma se encuentra detenida o bien de iniciarla dependiendo del caso.

3. Mensajes

Un mensaje consiste de dos partes:

- **Data** (información) que es enviada de una aplicación a otra.
- **Header** (cabecera) del mensaje o descriptor. Contiene información de control.

En el Header se especifica el ID del mensaje, tipo, prioridad, tiempo de expiración y el nombre de la cola para la respuesta. El tamaño del mensaje puede tener un máximo de 4 MB o 100 MB dependiendo de la versión de MQ.

4. Propiedades Básicas de los mensajes y colas

4.1.1. Persistencia

El diseño de una aplicación determina si a la hora de enviar mensajes, estos deben llegar obligatoriamente a destino o si pueden ser descartados si no llegan a tiempo, por ejemplo.
En el envío de ***mensajes persistentes***, estos son grabados en un Log a fin de evitar la perdida de información en el caso de una falla en el sistema, falla de alimentación o cualquier otra eventualidad.
Un ***mensaje no persistente***, en caso de una falla, no pueden ser recuperados.

4.1.2. Prioridad

Es la prioridad que tendrá el mensaje al momento de ser entregado a la aplicación que toma el mensaje de la cola. Es decir, mientras más alta sea la prioridad, será puesto más adelante en la cola al momento de ser enviado.
La prioridad puede estar entre 0 (prioridad más baja) y el 9 (prioridad más alta).
Esta propiedad se puede setear en el mensaje, de todas formas también se puede setear una propiedad por defecto en la cola.

4.1.3. Expiración

Cuando un mensaje es enviado a la cola, el cliente puede especificar un valor para el "time-to-live". El motor de colas no va a entregar un mensaje cuyo "time-to-live" ha expirado. Los mensajes que se encuentran en una cola van a ser borrados cuando su tiempo de expiración haya pasado.

Nota: En caso de que nos encontremos dentro de un Application Server utilizando JMS, no es recomendado, en casos generales, setear este valor desde la aplicación, sino hacerlo desde el Application Server.

4.1.4. Backout threshold

La cantidad máxima de veces que un mensaje puede intentar ser procesado. Si este límite es alcanzado, el mensaje es reencolado en la backout queue especificada en la propiedad Backout Requeue name.

MQ guarda un registro de la cantidad de veces que se ha hecho un rollback de un mensaje. Esta situación se produce cuando un mensaje es tomado por una aplicación, la misma no lo puede procesar y lo vuelve a poner en la cola de dónde lo había tomado (es decir, se produce un rollback). De esta forma, una vez alcanzado el límite configurable, el mensaje es reencolado en la cola de Backout. En caso de que, por alguna razón, este encolamiento falle, el mensaje es enviado a la dead-letter queue o bien, eliminado.

4.1.5. Backout Requeue name

El nombre de la cola dónde los mensajes serían reencolados en caso de que el límite marcado por la propiedad backout threshold sea superado

5. Conceptos de Mensajería

5.1. Bajo acoplamiento

Uno de los principales beneficios de los sistemas de mensajería es el bajo acoplamiento. A continuación se discuten dos aspectos.

5.1.1. Acoplamiento de Procesos

El bajo acoplamiento a nivel de procesos se debe a que el proceso emisor no debe preocuparse del estado del receptor, ya que el maganer de colas es quien recibe el mensaje y se encarga de enviárselo al receptor tan pronto como este esté nuevamente en línea o bien sea consultado por este último. A modo de comparación, se podría imaginar al Queue Manager como un servidor de correros que retiene los mismos hasta que sean consultados/extraídos por los cliejntes.

5.1.2. Acoplamiento de Aplicaciones

Las aplicaciones no deben preocuparse de adaptar su forma de comunicación según la interfaz del receptor del mensaje, sino que debe hacerlo adaptándose al manager de colas. Estos últimos suelen tener estándares abiertos para permitir la conectividad. Este enfoque permite que los clientes se focalicen en el formato del mensaje mas que por el formato de la interfaz destino.

5.2. Tipos de mensaje

A continuación se mencionan los cuatro tipos de mensajes soportados por MQ:

- **Datagram**: El mensaje no espera respuesta
- **Request**: El mensaje espera respuesta
- **Reply**: Una respuesta a un *Request*
- **Report**: Mensaje que describe un evento como por Ej. un error en el envío.

5.3. Tipos de mensajería

El tipo de mensajería describe el estilo de interacción entre el emisor y el receptor.

5.3.1. Mensajería Sincrónica

Este enfoque implica procesos acoplados, en el mismo es necesario que ambos, emisor y receptor, se encuentren operando y conectados directamente para que el intercambio de mensajes pueda ocurrir.

5.3.2. Mensajería asincrónica

El cliente que envía el mensaje continúa procesando sin esperar confirmación. Delega en el Queue Manager la responsabilidad de la correcta entrega del mismo. Para este enfoque no es necesario que la aplicación receptora se encuentre operando, ya que es el Manager de MQ (para este caso) quien lo recibe y actúa de intermediario entre ambas aplicaciones. Este enfoque, a diferencia del sincrónico, provee un menor acoplamiento entre las partes.

Nota: Es importante aclarar que, de acuerdo a lo que se definió en los puntos anteriores, cuando utilizamos MQ (o cualquier proveedor de colas), estaríamos utilizando mensajería asincrónica.
Desde el lado de las aplicaciones, también podríamos hablar de un sincronismo o asincronismo, dependiendo de si el proceso se queda esperando o no la respuesta (similar al modelo Request-Reply descrito más adelante). Sin embargo, debemos recordar que en este punto, estamos enfocando el tema desde el punto de vista del tipo de interacción entre las aplicaciones que se comunican.

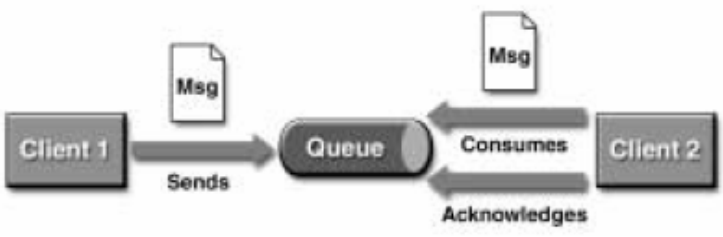
5.4. Modelos de mensajería

Esta clasificación ocurre para mensajes de tipos asincrónicos mencionados en el punto 5.3.2 y describe la cardinalidad entre el par emisor-receptor.

5.4.1. Punto a Punto

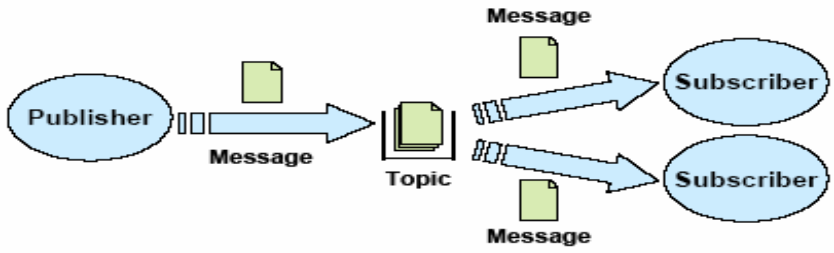
En este modelo, los mensajes son enviados hacia un destino específico, por lo general, el nombre de una cola, donde son extraídos por la aplicación. Es decir, hay un mapeo uno a uno entre el emisor y el receptor. La cola retiene todos los mensajes hasta que estos son extraídos o bien expiran por tiempo. Este modelo posee las siguientes características:

- Cada mensaje tiene un solo consumidor.
- El emisor y receptor no tienen dependencia de tiempo. El receptor puede obtener el mensaje independientemente si este estaba ejecutando cuando el emisor envió el mensaje.
- El receptor notifica cuando el mensaje fue consumido exitosamente (acknowledge).



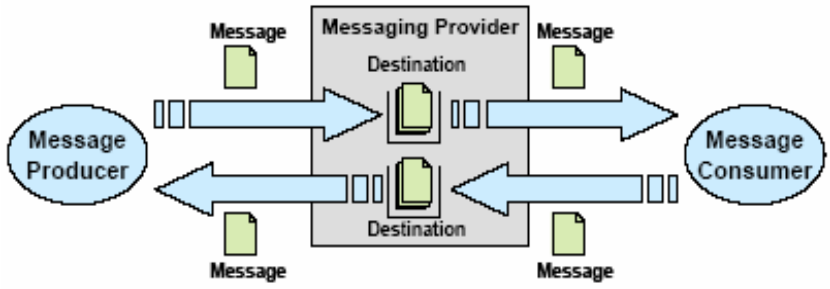
5.4.2. Publish/Subscribe

Cuando se necesita un mayor control sobre los mensajes que deben recibir determinados servicios, se necesita la función de Publicación/suscripción. Los sistemas de Publicación/suscripción suelen proporcionar un entorno seguro y escalable en el que muchos suscriptores se pueden registrar para recibir mensajes de muchos publicadores.
Puntualmente, este modelo describe a un emisor que envía un mensaje a una cola de destino. A esta cola de destino están suscriptas uno o mas clientes, a los cuales les será enviado una copia del mensaje original.
Para obtener los mensajes que cumplen con la suscripción, puede utilizar cualquiera de las funciones estándares de recuperación de mensajes.



5.4.3. Request – Reply

Este modelo se describe como pseudos-sincrónico, ya que utiliza se utilizan mensajes asincrónicos para simular un sincronismo. Aquí, cada aplicación actúa como emisor y receptor a la vez. Además, a pesar de que el Queue Manager permite una mensajería asincrónica actuando como intermediario, el emisor y el receptor se encuentran en un grado mayor de acoplamiento.



5.5. Patrones de Mensajería

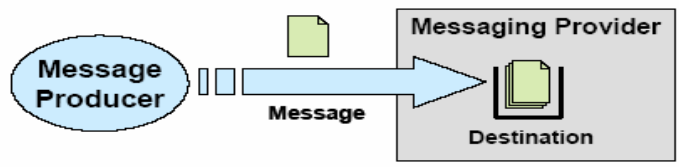
5.5.1. Productores de Mensajes

En este enfoque, la aplicación cliente se conecta al manager de colas, envía el mensaje y se desconecta. De esta manera se despreocupa del estado del mensaje enviado Se lo puede comparar con un *datagrama*.

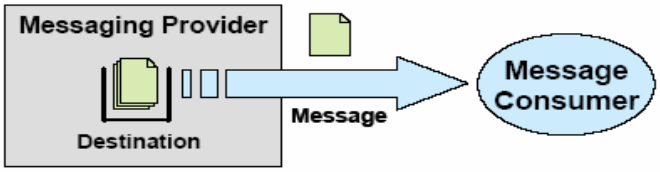
5.5.2. Consumidores de Mensajes

Los consumidores operan de alguno de los siguientes modos:

- ***Pull Mode***: La aplicación receptora se conecta al manager a fin de obtener el mensaje. De antemano desconoce si existe o no mensajes para extraer. Por esa razón es que debe sondear su cola asociada cada determinados períodos de tiempo.

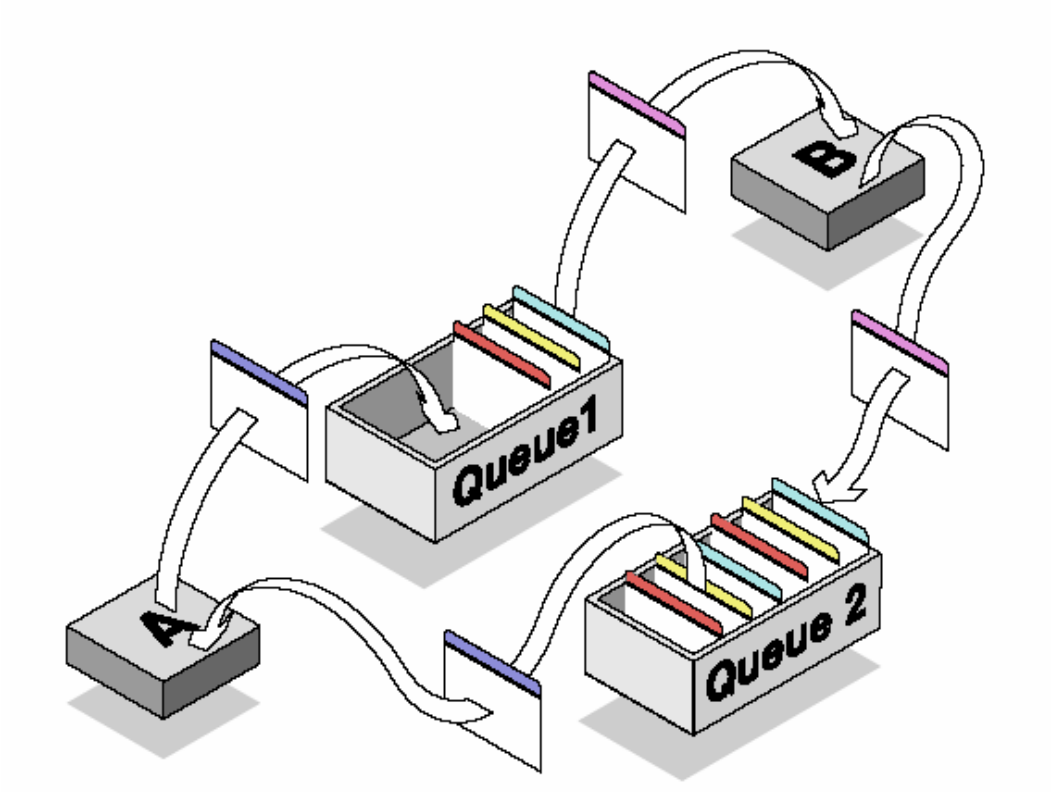


- ***Push Mode***: En este caso es el manager de colas quien inicia la comunicación cuando llega un mensaje para la aplicación cliente.



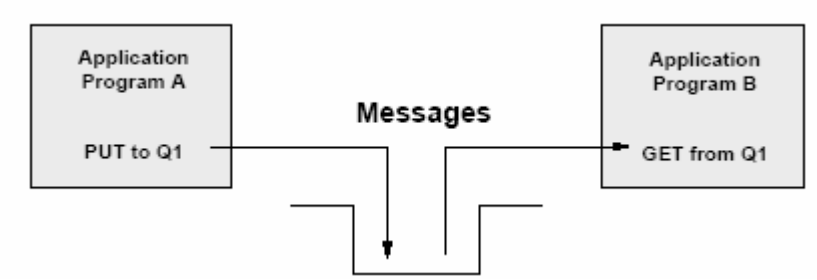
6. Arquitectura

A continuación se presenta una breve descripción de la arquitectura de MQ.
Los recuadros identificados como A y B representan aplicaciones que envían y reciben mensajes.
Las cajas entre las Queue y las aplicaciones A y B representan la ***Message Queuing Interface (API)***, que es uno de los componente de la arquitectura.
La API permite la interacción con el ***Queue Manager*** (representado por las cajas grises centrales) que es el administrador del sistema de mensajería.
La API esta conformada por 13 llamadas.

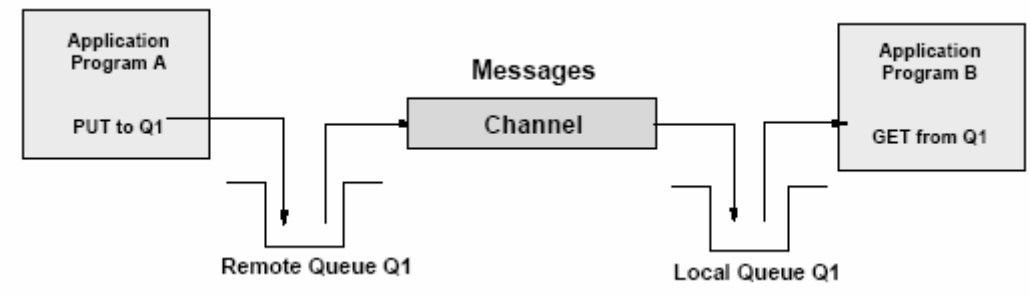

B
Queue1
A
Queue 2

7. Comunicación

La siguiente ilustración muestra dos aplicativos que corren en una misma maquina con un solo Queue Manager.



Otro enfoque es el caso de dos aplicativos donde cada uno esta asociado a un Queue Manager distinto. Este enfoque involucra la existencia de dos Queue Manager y una comunicación especial entre ellos. Los Queue Manager puede estar en una misma o en distintas maquinas.



8. Objetos administrados por el Queue Manager

El Queue Manager es un objeto en mismo y utiliza los siguientes objetos para su operatoria:

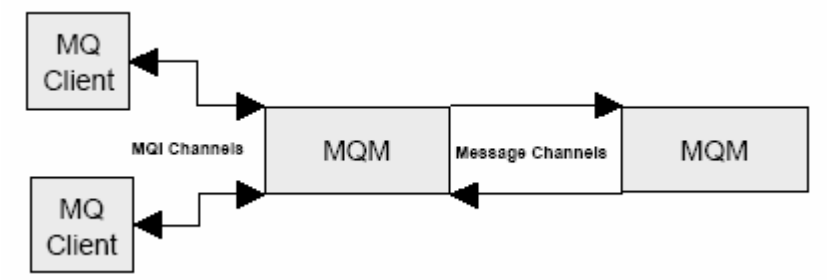
- ***Queues*** (Colas)
- ***Process definition*** (definición de procesos)
- ***Channels*** (canales)

Estos objetos son comunes a diversas plataformas. Existen otros propios a plataformas particulares que no se mencionarán.

Queues: Las colas son utilizadas por las aplicaciones para almacenar los mensajes. Existen colas que son administradas por el Queue Manager Local y otras remotas que son administradas por su correspondiente Queue Manager, pero que igual son conocidas por el primero.
Channels: un canal es una comunicación lógica. Hay dos tipos en MQ:

- *Message Channels*: Conecta dos QueueManager por medio de un canal unidireccional. Para obtener una comunicación bidireccional es necesario establecer dos canales en la misma dirección y sentido contrario.

- *MQI Channels*: Son canales bidireccionales que conectan un cliente MQ con un Queue Manager.



Process Definitions: Define una aplicación a un Queue Manager. Por ejemplo, contiene el nombre de la aplicación y su path a ver invocada cuando llega un mensaje para ella.

9. Cola Local

Una cola es "Local" si es propia del MQ Manager al cual esta conectada nuestra aplicación cliente. Esta es usada para almacenar mensajes de aplicaciones que usan el mismo MQ Manager.

10. Cola Remota

Una cola es "remota" cuando pertenece a un MQ Manager diferente al cual esta conectada nuestra aplicación cliente. Esta no es una cola real, pero contiene las características para parecer que lo es, pero que se relaciona con una cola local en otro MQ Manager. Es tarea del arquitecto decidir donde reside la cola en forma local, para que el resto de los MQ Manager la asocie como cola remota.

NOTA: una aplicación no puede leer mensajes de una cola remota.

Existen otro tipo de colas, pero para los fines de este documento no serán expuestos. Para mas información, por favor referirse a la documentación citada al final.

11. Otros Productos

11.1. Microsoft Message Queuing(MSMQ)

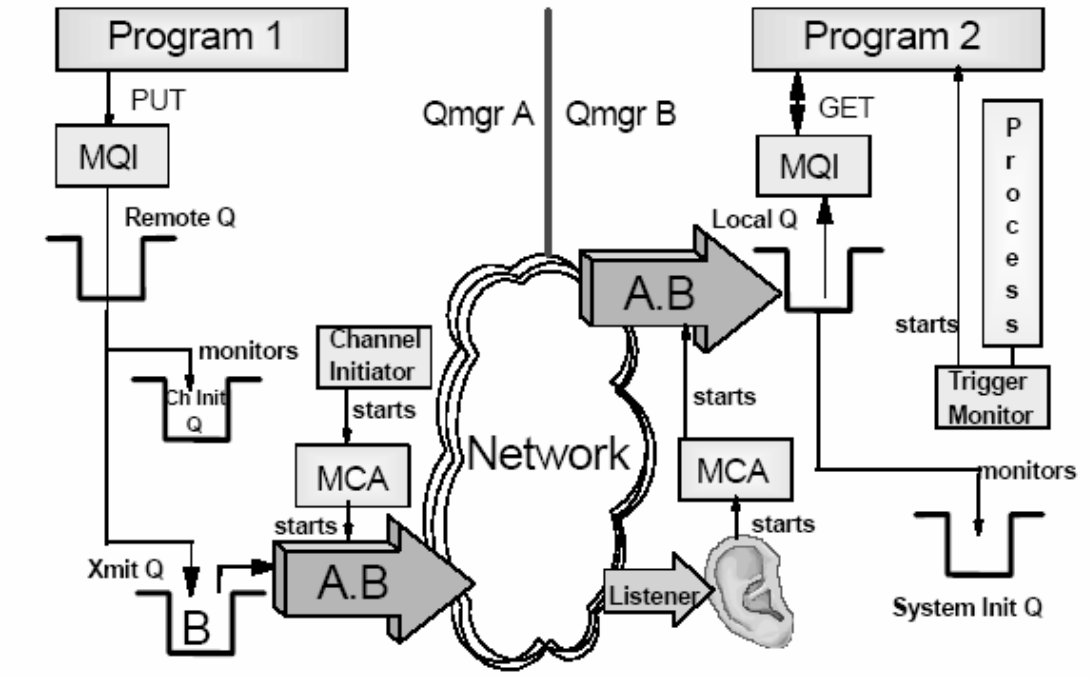
http://www.microsoft.com/windows2000/technologies/communications/msmq/default.mspx

11.2. Oracle Advanced Queuing(AQ)

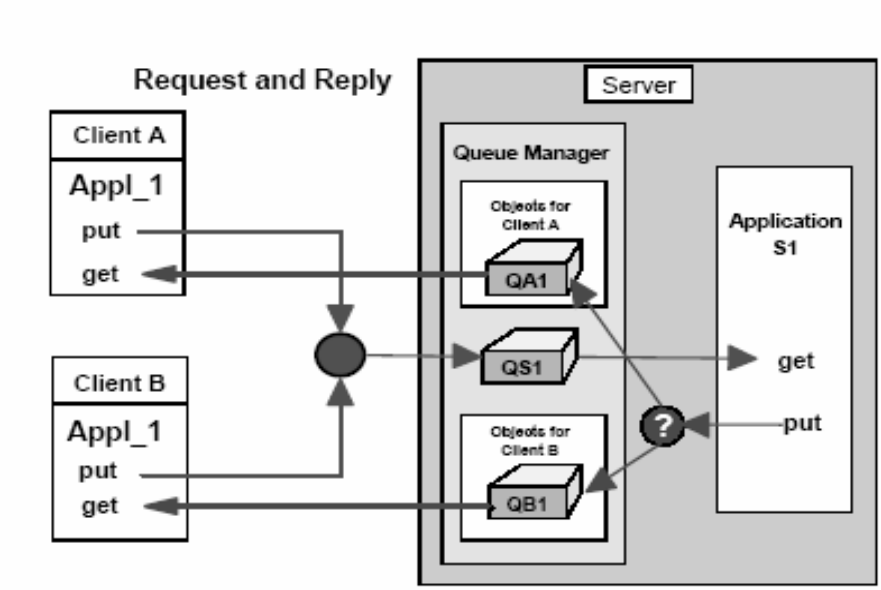
http://www.oracle.com/technology/products/aq/htdocs/9iaq_ds.html

Anexos

12. Cómo Funciona MQ.



13. Pedido y respuesta.



Referencias

http://www.redbooks.ibm.com/abstracts/redp0021.html?Open

http://publib.boulder.ibm.com/infocenter/db2help/index.jsp?topic=/com.ibm.db2.udb.doc/ad/c0010963.htm

http://www.microsoft.com/windows2000/technologies/communications/msmq/default.mspx

http://www.oracle.com/technology/products/aq/htdocs/9iaq_ds.html

http://www.redbooks.ibm.com/abstracts/sg246451.html?Open