Capítulo 2.
Tecnologías disponibles para la comunicación con fuentes de datos heterogéneas

Uno de los aspectos mas importantes en un sistema multibase de datos es la forma en como llevar a cabo la comunicación con las bases de datos componentes. En este capítulo se describen tecnologías actuales que permiten la comunicación con fuentes de datos heterogéneas, y sus características principales. Se describe mas detalladamente la tecnología JDBC, debido a que es la que se utilizo en el desarrollo del prototipo.

Internet ha sido en los últimos años uno de los principales medios de difusión de información. Debido a su popularidad cada vez mas software se ha desarrollado tomando a Internet como punto de partida. Entre estos desarrollos se encuentra una gran variedad de herramientas enfocadas a la recuperación de información desde diversas fuentes, incluyendo bases de datos relaciónales y de objetos, así como archivos planos, archivos de texto, documentos de procesador de palabras y páginas web.

Las páginas de consultas a bases de datos llegan a ser más interesantes, así como complejas cuando se desea integrar datos desde múltiples fuentes de datos. Por ejemplo, cuando deseamos hacer un *reunión* de dos tablas relaciónales en un solo campo de una tabla en una página web. Es aun más interesante si las tablas están almacenadas en dos bases de datos diferentes, en dos servidores diferentes y manejados por dos SMBDs diferentes. Para poder llevar a cabo esto, existen diversas tecnologías para integrar datos en aplicaciones web-base de datos. Aunque algunas de estas herramientas no son parte de la tecnología especifica para web, si son muy útiles para llevar a cabo esta tarea.

2.1 Compuertas para Bases de Datos

Las compuertas, desde una perspectiva de bases de datos, permiten la interoperabilidad de bases de datos, habilitando aplicaciones cliente para conectarse a múltiples fuentes de datos. Las compuertas para bases de datos son un tipo especial de *middleware* (software de enlace) que da a las aplicaciones cliente una interfaz de programación de la aplicación (API) que hace que diversas fuentes de datos parezcan equivalentes.

Una compuerta de base de datos consiste de:

1. Una biblioteca API cliente: La API del lado del cliente, la cual determina el formato y significado de las peticiones que las aplicaciones cliente pueden emitir.

2. Una biblioteca API Servidor: La API en la base de datos del lado del servidor, la cual determina los servicios de bases datos disponibles para los clientes.

3. Ligas: Mecanismos de traducción y mapeo que transforman la API cliente a la API del servidor y viceversa para los datos retornados a las aplicaciones.

2.2 Monitores de Procesamiento de Transacciones

Los monitores de procesamiento de transacciones (MPT) nacieron para manejar procesos y orquestar programas. Esto lo hacen al romper aplicaciones complejas en piezas de código llamadas transacciones. Los MPTs fueron introducidos para ejecutar clases de aplicaciones que podrían servir a miles de clientes. Esto se hace al proveer de un ambiente que ellos mismos intercalan entre clientes y servidores. Al hacer esta intercalación entre clientes y servidores los MPTs pueden manejar transacciones, rutearlas a través del sistema, balancear la carga de su ejecución y reiniciarlas después de una falla. Un MPT puede manejar recursos transaccionales en un servidor o a través de múltiples servidores, y puede cooperar con otros MPTs en arreglos federados [Orfali et al. 1996].

En el contexto de sistemas distribuidos, un MPT provee un número de funciones útiles, incluyendo multiprocesamiento automático, seguridad, y servicios de nombramiento global y resolución de nombres. Un MPT también provee acceso síncrono o asíncrono a fuentes de datos, soporta un gran número de usuarios, y facilita la integración de numerosos ambientes y componentes heterogéneos.

Tuxedo [BEA 1996] es un ejemplo de los monitores de procesamiento de transacciones.

2.3 ODBC

ODBC (conectividad de base de datos abierta) es una API abierta para accesar a bases de datos. La API especifica un conjunto de funciones para manejar conexiones a bases de datos, ejecutar declaraciones SQL (lenguaje de consultas estructurado) y consultar las capacidades del sistema de base de datos. ODBC esta basado en la especificación de la

interfaz a nivel de llamadas de X/Open SQL. Microsoft desarrollo ODBC como una implementación de la especificación de la interfaz a nivel de llamadas para proveer una API común para la conectividad de bases de datos.

ODBC provee una interfaz estándar que permite a las aplicaciones accesar a diferentes fuentes de datos. El código fuente de las aplicaciones no tiene que ser recompilado para cada fuente de datos. Un controlador de bases de datos conecta a la aplicación a una fuente de datos específica. Un controlador de bases de datos es una librería de cargado dinámico que una aplicación puede invocar en demanda de acceso a un origen de datos particular. Por lo tanto la interfaz ODBC define lo siguiente:

Una librería que llama a funciones ODBC que permite a una aplicación conectarse a una fuente de datos, ejecutar sentencias SQL, y recuperar resultados.

Una sintaxis SQL basada en el X/Open y el Grupo de Acceso SQL (SAG) especificación SQL CASE (1992).

Un conjunto estándar de códigos de error.

Una forma estándar para conectarse y registrarse en una fuente de datos.

Una representación estándar de los tipos de datos.

2.4 JDBC

JDBC (conectividad de bases de datos java) es una API a nivel SQL de aplicaciones Java para sistemas de bases de datos SQL. La API JDBC define clases Java para representar conexiones a bases de datos, declaraciones SQL, conjuntos de resultados, acceso a metadatos y más. La API JDBC puede soportar múltiples controladores que están conectados a diferentes SMBDs. JDBC provee una interfaz a nivel de programación para la comunicación con bases de datos de una manera uniforme similar al concepto de ODBC de Microsoft, el cual ha llegado a ser un estándar para accesar SMBDs. El estándar JDBC esta basado en la interfaz a nivel de llamadas X/Open SQL, la misma base que ODBC.

La API de JDBC , define una interfaz estructurada para bases de datos, la cual es el estándar de la industria para accesarlas. Al soportar SQL, JDBC permite a los programadores interactuar y soportar una gran cantidad de bases de datos. Esto significa que las características especificas de la plataforma de la base de datos se vuelven irrelevantes cuando se accesan desde JDBC, así como la heterogeneidad de las bases de datos al poder accesarlas desde un estándar como SQL.

2.4.1 ¿Que es JDBC ?

JDBC es la herramienta de conectividad de bases de datos para Java. JDBC abarca varias cosas dependiendo del contexto:

JDBC es una especificación para usar fuentes de datos en applets y aplicaciones de Java.

JDBC es una API para usar controladores JDBC.

JDBC es una API para crear los controladores JDBC, los cuales realizan la conectividad y transacciones con las fuentes de datos.

JDBC esta basado en la interface a nivel de llamadas SQL X/Open la cual define cómo las interacciones cliente/servidor deben ser implementadas para sistemas de bases de datos.

JDBC define cada aspecto para manipular bases de datos desde applets y aplicaciones Java. Los controladores JDBC ejecutan la traducción especifica de la base de datos a la interface JDBC. Esta interface es utilizada por el desarrollador y así el no necesita preocuparse por la sintaxis especifica de la base de datos cuando se conecta y consulta diferentes bases de datos. El aspecto excitante de JDBC es que los controladores necesarios para la conexión a sus respectivas bases de datos no requieren alguna preinstalación en los clientes: Un controlador JDBC puede ser descargado en el cliente junto con el applet.

JDBC esta ampliamente basado en el estándar ANSI SQL-92 [Patel y Moss 1996]. Esto no significa que un controlador JDBC tiene que ser escrito para bases de datos SQL-92, un controlador JDBC puede ser escrito para cualquier sistema de base de datos y funcionara perfectamente. Aunque el controlador no implemente cada función SQL-92, este aun seguirá siendo un controlador JDBC. Este es uno de los aspectos mas importantes de JDBC ya que de esta manera permite la recuperación de información desde fuentes de datos heterogéneas.

En el apéndice G muestra la implementación de la clase BaseDatos que hace la conexión a varias bases de datos utilizando JDBC.

2.4.2 JDBC cubre ODBC

Los diseñadores de JDBC emplearon una filosofía de diseño que cubriera los conceptos de ODBC (conectividad de base de datos abierta). Las dos principales razones para modelar JDBC como ODBC son:

ODBC es ampliamente utilizado, lo cual ayudaría a que los usuarios lo aprendieran rápidamente.

Hay implementaciones ODBC eficientes en todas las plataformas para casi todas las bases de datos.

Los productos JDBC disponibles se clasifican en dos categorías: aquellos que comunican a un controlador ODBC existente y los que comunican a una API de una base de datos nativa. Javasoft desarrollo lo que se conoce como el Puente JDBC-ODBC para tomar ventaja de un gran número de fuentes de datos habilitadas como ODBC. La ventaja primaria de usar el puente JDBC-ODBC es que las llamadas JDBC son finalmente convertidas en llamadas ODBC, así las aplicaciones pueden fácilmente accesar bases de datos de múltiples vendedores al seleccionar el controlador ODBC apropiado [Shah 1998]. Sin embargo el puente JDBC-ODBC requiere preinstalación en el cliente donde quiera que el programa Java se ejecute, debido a que el puente debe hacer llamadas a métodos nativos para hacer la traducción de ODBC a JDBC. Solamente los controladores JDBC 100% Java pueden ser descargados desde la red junto con el applet, y estos no requiere preinstalación del controlador.

En JDBC, las tareas de base de datos simples como consultas básicas, creación y actualización, pueden ser hechas utilizando métodos simples y directos. Para tareas mas complejas como múltiples ResultSets (clase del paquete sql de Java para almacenar los resultados de una consulta) y procedimientos almacenados con parámetros IN y OUT, el JDBC tiene declaraciones separadas [Patel y Moss 1996]. Para programas de automatización y herramientas del diseñador, JDBC tiene clases y métodos de metadatos, los cuales proveen información acerca de las diversas características soportadas en las bases de datos, estructura de las tablas y otras características.

Un aspecto importante de JDBC es que el diseño estándar de la interface JDBC permite cambiar entre /los controladores y por tanto las bases de datos sin recodificar los programas.

Java. JDBC, por lo tanto, emerge para dar solución al problema de la neutralidad de plataformas y heterogeneidad de fuentes de datos.

2.5 OLE DB

OLE DB es un conjunto de interfaces desarrolladas por Microsoft cuya meta es habilitar aplicaciones para tener un acceso uniforme a datos almacenados en diversas fuentes de información de SMBDs (sistemas manejadores de bases de datos) y No-SMBDs. Las fuentes SMBDs pueden incluir bases de datos mainframe (e.g., IMS, DB2), servidores de bases de datos (e.g., Oracle, SQL Server), o repositorios de datos de escritorio (e.g., Access, Paradox, Fox). Fuentes No-SMBDs pueden incluir información almacenada en sistemas de archivo (e.g., Windows NT, Unix), archivos

secuenciales indexados, correo electrónico, hojas de calculo, herramientas de manejo de proyectos y muchas otras fuentes.

Las áreas funcionales de OLE-DB incluyen acceso a datos y actualizaciones, procesamiento de consultas, información del catalogo, notificaciones, transacciones, seguridad y acceso a datos remotos. OLE DB cubre la infraestructura OLE-COM (módelo de objetos componente), la cual reduce la duplicación innecesaria de servicios y provee un grado mas alto de interoperabilidad no solamente entre diversas fuentes de información, sino también entre ambientes y herramientas de programación existentes [Blakeley 1996].

2.6 CORBA

CORBA (Common Object Request Broker Architecture) es un estándar para la interoperación de objetos en ambientes cliente-servidor heterogéneos. Los objetos CORBA pueden residir en cualquier parte de una red. Estos están empaquetados como componentes binarios que los clientes remotos pueden accesar vía invocación de métodos. El lenguaje y compilador utilizado para crear los objetos servidores, son totalmente transparentes a los clientes. Los clientes no necesitan conocer donde los objetos distribuidos residen o en que sistema operativo se ejecutan.

Un ORB (Object Request Broker) CORBA viene con mecanismos estándar para generar y manejar metadatos. Los metadatos que describen los componentes y sus interfaces se generan usando IDL (lenguaje de definición de la interface). El precompilador IDL del ORB escribe automáticamente sus metadatos definidos IDL en un repositorio de interfaces. Desde el repositorio se pueden actualizar los metadatos usando las operaciones de escritura y actualización del repositorio de interfaces. Los clientes pueden consultar el repositorio de interfaces, ellos pueden descubrir que interfaces están disponibles y como llamarlas. Se puede utilizar el mismo repositorio de interfaces para almacenar la descripción de componentes. Con CORBA se tiene una especificación sólida para crear repositorios de interfaces federadas que pueden operar a través de ORB´s y sistemas operativos heterogéneos [Orfali et al. 1996].

2.6.1 Servicios de Objetos

Los servicios de objetos CORBA son colecciones de servicios a nivel del sistema empacados como componentes con interfaces especificadas con IDL. Estos servicios aumentan y complementan la funcionalidad del ORB. Son utilizados para crear un componente, nombrarlo, e introducirlo dentro del ambiente. OMG tiene definido estándares para los siguientes servicios de objetos: servicio de ciclo de vida, servicio de persistencia,

servicio de nombres, servicio de eventos, servicio de control de la concurrencia, servicio de transacciones, servicio de consultas, servicio de propiedades entre otros. La descripción y operación de estos servicios la encuentra en [Orfali et al. 1996], aquí solo describiremos el servicio de consultas.

2.6.1.1 Servicio de Consultas

Este servicio permite encontrar objetos cuyos atributos reúnen el criterio de búsqueda especificados en una consulta. Las consultas se pueden formular utilizando alguno de los siguientes lenguajes: OQL (lenguaje de consultas de objetos) de ODMG-93, SQL (con extensiones a objetos), o un subconjunto de estos dos lenguajes.

El servicio de consultas puede directamente ejecutar una consulta o delegar esto a algún otro *evaluador de consultas* . Por ejemplo, el servicio puede usar las facilidades de consultas nativas de una base de datos relacional o de objetos para ejecutar una consulta anidada. El servicio de consultas combina los resultados de la consulta desde todos los evaluadores de consultas participantes y retorna el resultado final. Esto significa que se puede usar el servicio de consultas para coordinar federaciones débiles de manejadores de consultas nativas. Lo relevante es que se puede utilizar su propia maquina de búsqueda optimizada, mientras participa en una búsqueda global [Orfali et al. 1996].

Como podemos observar actualmente existen diversas tecnologías que nos permiten establecer la comunicación con las fuentes de datos, esto hace que las diferencias entre plataformas y fuentes de datos no tengan mayor relevancia y no sean un impedimento para compartir su información.

Después de resolver la conexión a las bases de datos componentes en un sistema multibase de datos, es necesario determinar la forma en que debe ser procesada una consulta en un sistema de este tipo. De esto trata el siguiente capítulo.