MultiBases-Web: Procesamiento de consultas distribuidas en sistemas de múltiples bases de datos en la Web

Cristo A. Rodríguez
Dpto. de Ingeniería de Sistemas y Computación
Universidad de los Andes
A.A. 4976 Bogotá-Colombia
e-mail: cr-rodri@uniandes.edu.co

M. Consuelo Franky
Dpto. de Ingeniería de Sistemas y Computación
Universidad de los Andes
A.A. 4976 Bogotá-Colombia
e-mail: cfranky@uniandes.edu.co

Resumen: *Un sistema de múltiples bases de datos (SMultBD) habilita la integración de las diferentes bases de datos que conforman una organización, dando una visión única de la información. La World Wide Web (WWW) y el código móvil brindan una infraestructura adecuada para el desarrollo de sistemas SMultBD por su transparencia a la heterogeneidad y a la localización de recursos. En este artículo se presenta un proyecto de investigación denominado MultiBases-Web que es un SMultBD desarrollado en base a las herramientas del lenguaje Java. La plataforma ofrecida por MultBases-Web facilita la generación de aplicaciones Web que requieran realizar consultas distribuidas sobre múltiples bases de datos heterogéneas ubicadas en distintos sitios Internet. El artículo describe el diseño general y los alcances de esta plataforma.*

Palabras claves*: Bases de Datos Distribuidas, Optimización de consultas distribuidas, Orientación de Objetos, Código móvil, Web, Java.*

1. Introducción

Muchas organizaciones en su proceso de evolución, han generado sistemas de bases de datos autónomos y heterogéneos por departamentos o secciones de la empresa debido a las constantes innovaciones tecnológicas. Pero los requerimientos de los usuarios y las exigencias de competitividad actuales crean la necesidad de compartir la información que se encuentra dispersa en los múltiples sistemas de bases de datos de la empresa. Un sistema de múltiples bases de datos (**SMultBD**) "*multidatabase system*" es una herramienta distribuida que actúa como *front-end* de diferentes sistemas de bases de datos, permitiendo acceder de forma simultánea a aplicaciones heterogéneas y autónomas mediante un único modelo de datos.

La World Wide Web (WWW) y los sistemas distribuidos orientados por objetos (como CORBA [Siegel96], RMI [JavaSoft1] y HORB [Hirano96]) brindan una poderosa y adecuada infraestructura para el desarrollo de SMultBD ya que presentan múltiples ventajas en el manejo de la heterogeneidad, y la transparencia de la localización de recursos, que reducen la complejidad y el código necesario para generar un SMultBD.

El proyecto MultiBases-Web, que actualmente estamos desarrollando, propone un SMultBD basado en las herramientas JDBC [Hamilton96] y RMI (Remote Method Invocation) que ofrece el lenguaje Java [Gosling94]. La plataforma resultante de este proyecto facilita el desarrollo de aplicaciones de consultas globales sobre múltiples bases de datos heterogéneas ubicadas en diferentes sitios Internet, realizando la correspondiente optimización de consultas distribuidas.

El resto del actual artículo está organizado de la siguiente forma: En la Sección 2 se describen los principales aspectos teóricos de los SMultBD. En la Sección 3 se da una breve descripción del lenguaje Java y las facilidades que brinda para el desarrollo de aplicaciones distribuidas sobre múltiples bases de datos. La arquitectura que proponemos para el sistema MultiBases-Web se explica en la Sección 4. La Sección 5 presenta los principales componentes del sistema MultiBases-Web. Finalmente la Sección 6 plantea conclusiones y extensiones futuras de este trabajo.

2. Aspectos de los sistemas Multi-Bases de Datos (SMultBD)

Un SMultBD es un sistema distribuido que actúa como *front-end* de múltiples bases de datos locales, brindando un único esquema global. Las bases de datos locales pueden ser heterogéneas (diferentes modelos de datos y/o diferentes implementaciones) y pueden existir con anterioridad a la creación del SMultBD. Aunque las bases de datos locales deben colaborar con el sistema global, ellas preservan su autonomía en el procesamiento de sus datos locales [Bright94]. A continuación se presentan las principales características de estos sistemas.

2.1 Distribución

En un SMultBD los datos pueden estar distribuidos a través de diferentes bases de datos. Estas bases de datos son almacenadas en un computador simple o en múltiples computadores distribuidos geográficamente e interconectados por una red de comunicación. Tomando como referencia el modelo Relacional, las tablas de datos pueden estar distribuidas mediante fragmentación horizontal y/o vertical, y de cada uno de estos fragmentos pueden existir múltiples copias (replicación). La distribución de datos incrementa la disponibilidad, la seguridad y la eficiencia [Ceri84].

2.2 Heterogeneidad

La heterogeneidad se presenta debido a diferencias tecnológicas, y de diseño de las diferentes bases de datos que conforman un SMultBD [Larson90].

Entre las diferencias tecnológicas tenemos:

- Diferencias en la estructura: Estas se refieren a los diferentes modelos de datos utilizados para representar la información, como: modelo relacional, modelo jerárquico y modelo orientado por objetos.
- Diferencias de capacidades: A pesar de que dos sistemas de bases de datos tengan el mismo modelo de datos pueden existen diferencias en cuanto a las capacidades que soportan cada uno de ellos. Por ejemplo, entre las bases de datos relaciónales existen mecanismos que son opcionales o que no son estándar (ej. Triggers, procedimientos almacenados, etc).
- Diferentes lenguajes de consulta y manipulación: Entre los diferentes lenguajes de consultas utilizados tenemos: SQL, QUEL y OQL.
- Diferencias entre plataformas computacionales: Las bases de datos que conforman el SMultBD pueden estar localizadas en máquinas con sistemas operativos distintos que difieren en sus sistemas de archivos y en la representación de los datos.

Las diferencias debidas a decisiones de diseño determinan la heterogeneidad semántica, que ocurre cuando hay discrepancias acerca del significado, la interpretación o el propósito entre datos relacionados. Los conflictos que se pueden presentar son:

- Conflictos de nombre: Ocurre cuando la misma relación o tabla tienen nombres distintos, o dos relaciones tienen el mismo nombre pero representan objetos diferentes del mundo.
- Conflictos de escala: Cuando los atributos se almacenan en diferentes unidades o escalas.
- Diferentes niveles de abstracción: Algunas bases de datos almacenan con mayor detalle determinada información. Por ejemplo, en una universidad las facultades almacenan el nombre de los profesores, sus especialidades y su disponibilidad de tiempo, mientras que el departamento de personal de la universidad almacena información adicional de los profesores como sueldo, numero de hijos, etc.

2.3 Capacidades de un SMultBD

Para solucionar los problemas generados por la distribución y la heterogeneidad de las diferentes bases de datos que se desean integrar, un SMultBD debe poseer las capacidades de integración de esquemas, administración del esquema global, manejo de transacciones distribuidas y manejo de consultas distribuidas [Gomer90]:

- Integración de esquemas: Cada base de datos local tiene su propio esquema que describe la estructura de los datos almacenados. Con estos diferentes esquemas se debe crear un único esquema global, que resuelva la heterogeneidad semántica y la distribución de datos (fragmentación y replicación).
- Administración del esquema global: Entre las funciones administrativas tenemos la autenticación de usuarios y el mantenimiento del esquema global.
- Manejo de consultas distribuidas: Uno de los objetivos fundamentales de los SMBDs es la posibilidad de combinar datos almacenados en diferentes sitios mediante una única operación de consulta, donde los procesos de optimización adquieren gran importancia debido los costos de transmisión de datos.
- Manejo de transacciones distribuidas: El manejo de transacciones distribuidas provee la habilidad de leer y/o actualizar información de múltiples sitios dentro de una única transacción, preservando las propiedades de atomicidad, consistencia, aislamiento y durabilidad [Ceri84], mediante la implementación de protocolos de control de concurrencia y protocolos de recuperación ante fallas.

3. Java como herramienta de desarrollo de aplicaciones distribuidas sobre múltiples bases de datos.

Java es un lenguaje de programación orientado por objetos (con las características generales de C++) diseñado para crear aplicaciones seguras, portables e interactivas en Internet[Gosling94]. El compilador de Java no produce código binario para una determinada arquitectura como las compiladores tradicionales, sino que genera código móvil, el cual puede ser transferido y ejecutado en diferentes plataformas sin necesidad de recompilarlo.

Mediante Java se pueden crear *Application* y *Applets*. Una *application* es una aplicación que puede ser interpretada en cualquier máquina pero no es código móvil. Por su parte un *applet* es código móvil Java que se ejecuta en lo clientes Web como parte de una hoja HTML.

Java, más que un lenguaje de programación es todo un ambiente de desarrollo. Dicho ambiente, denominado JDK (Java Development Kit)[JavaSoft2] está compuesto por el compilador de Java (*javac*), un interpretador de *applets* denominado *AppletViewer*, un depurador y un amplio conjunto de clases preconstruidas (*built-in*). Dichas clases se encuentran agrupadas en paquetes según su funcionalidad. A continuación se describen brevemente los paquetes más relevantes para el desarrollo de aplicaciones distribuidas sobre múltiples bases de datos.

3.1 AWT

El JDK provee un rico conjunto de funciones gráficas encapsuladas en la biblioteca de clases *Abstract Window Toolkit* (AWT) [Naughton96], que da la posibilidad de crear *applets* para actuar como clientes con adecuadas y robustas interfaces de usuario, en un ambiente cliente servidor.

3.2 NET

Java posee como parte de su núcleo un amplio conjunto de clases que facilitan la generación de aplicaciones que requieren la utilización de los protocolos de Internet como: sockets, datagramas, HTTP y TCP/IP [Naughton96].

3.3 RMI

RMI (Remote Method Invocation) [JavaSoft1] permite el desarrollo de aplicaciones distribuidas, en las cuales los objetos que interactúan para llevar a cabo la funcionalidad deseada, se encuentran en máquinas diferentes. Un programa Java puede invocar métodos de un objeto remoto una vez haya obtenido una referencia a dicho objeto remoto, ya sea por una solicitud de búsqueda del objeto remoto al servidor de nombres de RMI, o por haber obtenido la referencia como un valor de retorno de algún método.

En el procesamiento de una invocación remota existen dos roles; un objeto cliente (quien hace la invocación) y un objeto servidor (quien provee el método). Este último puede ser a su vez cliente de otro objeto (inclusive de su propio cliente).

RMI solo permite la comunicación entre objetos Java, y está basada en el mecanismo de serialización de objetos del JDK 1.1[JavaSoft2], por lo cual los objetos que se utilicen como parámetros de un método remoto deben ser "serializables" (i.e. con capacidad de convertirse en un flujo de bytes, a partir del cual se pueda reconstruir el objeto original).

3.4 JDBC

JDBC (Java Database Connectivity) [Hamilton96] es una interfaz para ejecutar sentencias SQL sobre bases de datos locales o remotas y recuperar sus resultados desde Java. Para utilizar JDBC es necesario que el proveedor del motor de la base de datos que se desea acceder provea el driver JDBC respectivo.

JDBC permite comunicar un *applet* o una *application* con un amplio rango de sistemas de bases de datos relaciónales.

JDBC es un API de bajo nivel que permite desarrollar aplicaciones Java con independencia a los motores de bases de datos que se van a utilizar y con transparencia a los mecanismos de conexión a las bases de datos.

Este API está basado en X/Open SQL CLI (*Call Level Interface*) sobre el cual también está basado ODBC de Microsoft [Signore95].

a) Interfaces que conforman JDBC

El API de JDBC está expresado como una serie de interfaces de Java que le permiten a una aplicación abrir conexiones a una base de datos particular, ejecutar sentencias SQL, y procesar sus resultados.

JDBC está conformado por las siguientes interfaces de Java:

java.sql.DriverManager : Es encargado de activar los drivers y proveer soporte a la creación de nuevas conexiones a las base de datos.

java.sql.Connection : Representa una conexión a una base de datos.

java.sql.Statement : Representa una solicitud a la base de datos. Tiene dos subtipos:

java.sql.PreparedStatement para realizar la precompilación de una sentencia SQL, y

java.sql.CallableStatement para ejecutar llamadas a procedimientos almacenados de la base de datos.

java.sqlResultSet : Controla el acceso a cada una de las tuplas retornadas como resultado a una petición (statement).

b) Restricciones de seguridad de los *Applets* que utilizan JDBC

Las implementaciones más popularizadas de Java son los *applets*, los cuales son cargados a través Internet como parte de un documente HTML. Estos *applets* mediante la funcionalidad de JDBC pueden acceder a una base de datos. Sin embargo, el desarrollado de aplicaciones cliente/servidor está limitado por las restricciones de seguridad impuestas por Java para garantizar que cualquier *applet* que un cliente Web cargue, no hará usos indebidos de los recursos de este cliente. En los clientes Web (*browser*) actuales, un *applet* no puede leer ni escribir sobre los dispositivos de almacenamiento secundario del cliente, ni puede abrir conexiones con servidores o clientes diferentes al servidor que originó dicho *applet*. Sin embargo, con las nuevas características seguridad del JDK 1.1, los clientes Web permitirán ejecutar *applets* confiables que podrán acceder a los recursos locales del cliente.

4. Arquitectura del sistema MultiBases-Web

A pesar que MultiBases-Web pretende ser un SMultBD completo (ver Sección 2), el desarrollo inicial de este proyecto se centra en la integración de esquemas y en el manejo de consultas distribuidas sobre bases de datos relacionales, con soporte a la fragmentación horizontal sin replicación.

Para el procesamiento de consultas distribuidas se utiliza el algoritmo SDD-1 [Bernstein81] por estar enfocado a la reducción de bytes transmitidos (factor crítico en la Web, por los anchos de banda disponibles).

Al construir MultiBases-Web usando RMI [JavaSoft1] se aprovechan las ventajas del código móvil de Java, mediante el cual se pueden desarrollar aplicaciones Cliente-Servidor en las cuales gran parte de la lógica de las aplicaciones se traslada a los clientes, aumentado la escalabilidad de los sistemas. Adicionalmente la heterogeneidad entre plataformas y sistemas operativos es cubierta por las características de interoperabilidad del bytecode de Java [Gosling94].

La arquitectura que proponemos para MultiBases-Web presenta los mismos componentes propuestos por otras implantaciones anteriores al advenimiento de la Web [Gomer90]. En relación con esas implantaciones, el aporte que hacemos en la plataforma MultiBases-Web es aplicar los principios del paradigma de orientación por objetos y las características del código móvil.

4.1 Estrategia global

El proceso de consulta se desarrolla a través de tres fases (ver Ilustración 1). La primera fase extrae de la sentencia de consulta SQL la calificación. los nombres de las tablas y los atributos resultado de la consulta. La calificación corresponde a las condiciones del WHERE de la sentencia SELECT.

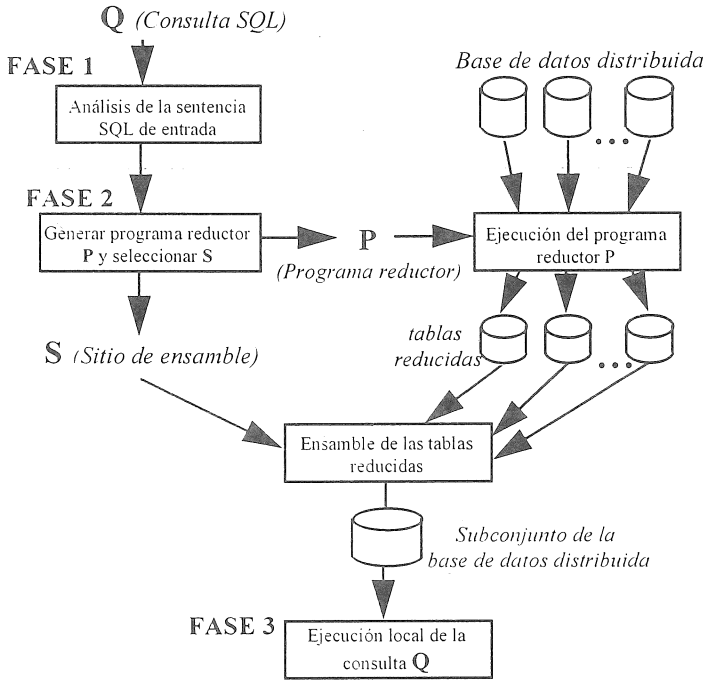

Ilustración 1: Procesamiento de consultas distribuidas

En la segunda fase se construye una base de datos reducida y centralizada en el sitio de ensamble de la respuesta. Dicha base de datos está conformada por tablas temporales que representan cada una de las tablas originales involucradas en la consulta. Las tuplas que conforman las tablas temporales son un subconjunto de las tuplas de las tablas originales.

Tanto las tuplas seleccionadas para conformar las tablas temporales como el sitio de ensamble se determinan mediante un programa reductor. que consiste de una secuencia de operaciones del álgebra relacional (selecciones, proyecciones, y semi-joins). El propósito del algoritmo SDD-1[Bernstein81] es generar dicho programa reductor, aplicando criterios de optimización que buscan reducir la cantidad de datos transmitidos. Dichos criterios están basados en los perfiles de las bases de datos, que son estadísticas acerca del volumen y la distribución de los datos [Ceri84].

La última fase ejecuta la consulta original en el sitio de ensamble a partir de las tablas temporales generadas en la fase 2. Esta operación es local y su procesamiento depende del manejador de bases de datos localizado en el sitio de ensamble.

De esta forma, los componentes básicos de MultiBases-Web están representados por los siguientes subsistemas: **agente de la base de datos Global** (ABG), y **agente de la base de datos Local** (ABL). Complementariamente se cuenta con los subsistemas **Manejador del catálogo global, Reductor de semi-join** y **Ensamblador** (Ver Ilustración 2 : Arquitectura del MultiBases-Web).

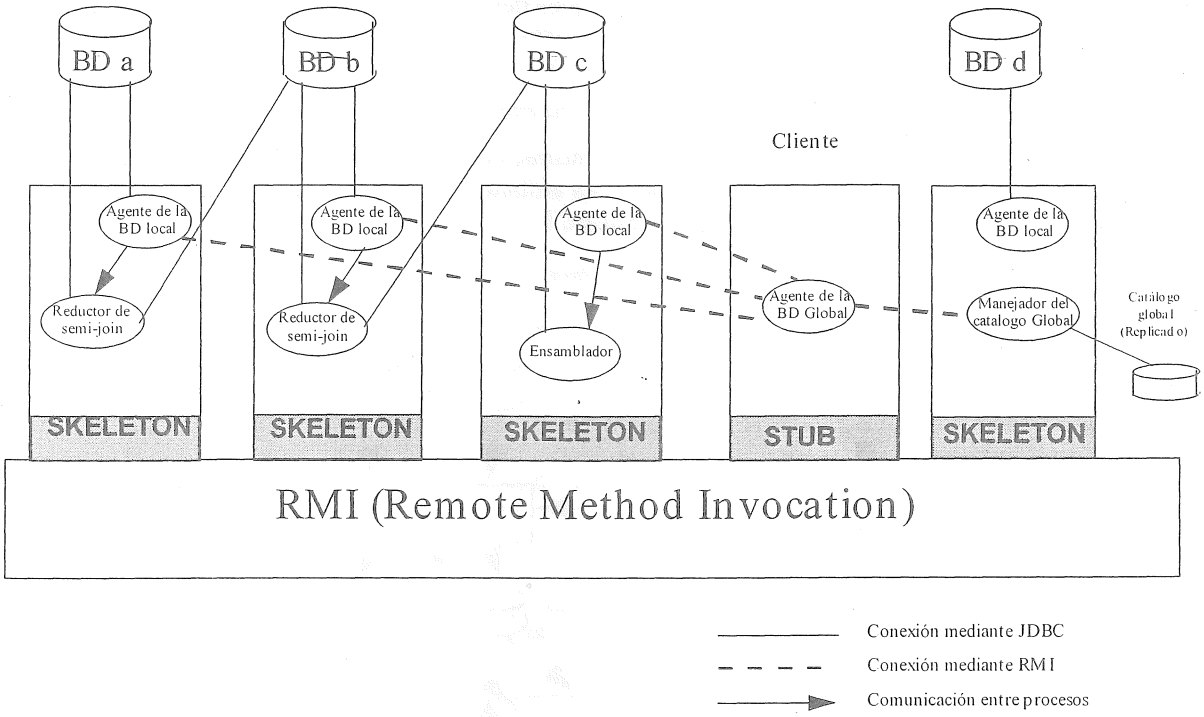

Ilustración 2 : Arquitectura del MultiBases-Web

4.2 Subsistema agente de la base de datos local (ABL)

Una instancia del subsistema ABL es responsable de:

- Proveer comunicación y conexión a la base de datos local. Dicha interfaz está basada en JDBC™ [Hamilton96].
- Mantenimiento del esquema local. Este mantenimiento consiste en actualizar y proveer la información de la base de datos local que es relevante para el esquema global, como el diccionario de datos y la estimación de perfiles [Ceri84].
- Traducir las sentencias del esquema global a la respectiva sentencia local. Para esto se deben almacenar las diferentes transformaciones semánticas de la representación global al esquema local mediante conversión de datos.
- Para la aplicación del algoritmo SDD-1 es necesario crear tablas temporales (denominadas tablas reducidas) conformadas por un subconjunto de las tuplas de las tablas originales y que siguen el esquema global del SMultBD. Estas tablas reducidas deben ser generadas por el ABL según una solicitud dada por el ABG.
- Crear un agente reductor de semi-join que creará o reducirá una tabla local mediante un semi-join con una tabla localizada en un sitio remoto.
- Crear un agente ensamblador encargado de ejecutar la consulta final sobre la base de datos de tablas reducidas que se ha centralizado en este sitio.

4.3 Clase agente de la base de datos global (ABG)

Por cada cliente (*applet* o *application* Java) que desee realizar una consulta al sistema MultiBase-Web se debe obtener una referencia a un objeto de la clase ABG, la cual tiene acceso a la información del catálogo global a través del Manejador del catálogo global.

La principal función de la clase ABG es determinar el programa reductor óptimo **P** de una consulta global dada, y coordinar la ejecución de dicho programa [Bernstein81]. Para la generación de **P** el ABG debe consultar al Manejador del catálogo global los perfiles y el esquema de la base de datos global. Una vez se ha generado **P** se procede a su ejecución que consiste en:

- Enviar a cada uno de los ABL involucrados la solicitud de generar las tablas reducidas iniciales, mediante selecciones y proyecciones locales.
- Para cada uno de los semi-joins $(R\ SJ_{A=B}\ S)$ [Ceri84] de **P**, enviar al ABL que contiene la tabla a reducir (sitio de *R*) la solicitud de ejecutar el semi-join, indicando el sitio y el nombre de la tabla que hará la reducción (sitio y nombre de *S*).

El ABG debe sincronizar la ejecución de los semi-joins, ya que estos deben seguir el orden impuesto en **P**.

- Una vez se han ejecutado todos los semi-joins de **P** se debe enviar al ABL del sitio seleccionado como ensamblador la orden de ensamblar.
- Por último, se crea el cursor que representa el conjunto de tuplas respuesta de la consulta. Este cursor se implementa como un objeto que recupera desde el sitio de ensamble las tuplas respuesta por tandas[1], según las solicitudes del programa cliente.

En caso que el cliente sea un *application* Java, el objeto ABG consiste en una instancia de la clase ABG. Pero debido a las restricciones de los *applets*, si el cliente es un *applet*, el objeto ABG se obtiene como una referencia a un objeto servidor RMI de la clase ABG. Una vez se ha obtenido la referencia al objeto ABG, la programación de un cliente, ya sea una *application* o un *applet* es la misma.

4.4 Clase Manejador del catálogo global

Esta clase es responsable de mantener actualizado el catálogo global, como también proveer información a los objetos que la soliciten. Este catálogo puede estar replicado en múltiples sitios. La información que se almacena en el catálogo consiste en:

- Esquema de la base de datos global.
- Estimación de perfiles de cada una de las tablas (información necesaria para el proceso de optimización) [Ceri84].
- Composición de cada una de las tablas del esquema global. Por cada fragmento de una tabla global se debe almacenar su localización y su calificación.

El esquema de la base de datos global y su estructura se crean de forma manual mediante una aplicación que le permite al usuario observar cada uno de los esquemas locales y de acuerdo a esta información resolver los conflictos semánticos que se puedan presentar mediante la definición de sinónimos y de funciones de conversión de datos.

La estimación de perfiles se hace por procedimientos automáticos que consultan cada uno de los perfiles locales y actualiza cada una de las replicas del catálogo.

El catálogo global constituye otra base de datos del sistema y su acceso desde el Manejador del catálogo global se realiza mediante JDBC.

---

[1] El numero de tuplas de una tanda es un valor constante de la aplicación

4.5 Reductor de semi-joins

Un proceso Reductor de semi-joins es un agente lanzado por un ABL que debe realizar la reducción de la tabla local R según el semi-join $(R\ SJ_{A=B}\ S)$ donde S es una tabla localizada en un sitio remoto. Tanto R como S están expresadas en términos del esquema global.

Este proceso reductor se ejecuta a través de los siguientes pasos:

- Solicitar vía JDBC las tuplas resultado de proyección de los atributos $\mathbf{B}$ sobre S ($\mathbf{PJ}_{\mathbf{B}}\ S$), y crear la tabla temporal T. La cardinalidad de T puede llegar a ser menor que la cardinalidad de S, ya que en T no puede existir más de una tupla con el mismo atributo $\mathbf{B}$.
- Generar una nueva tabla reducida R con el resultado de ejecutar el join entre R y T ($R\ \mathbf{JN}_{A=B}\ T$)

La generación y manipulación de las tablas reducidas locales también se hace vía JDBC.

4.6 Ensamblador

El proceso Ensamblador es creado por un ABL. El objetivo de este proceso es centralizar todas las tablas reducidas y ejecutar el join entre ellas de forma local. Este procedimiento se ejecuta vía JDBC.

Por cada consulta se deben generar las respectivas tablas reducidas, las cuales son independientes de otras tablas generadas por otras consultas. Por está razón es necesario generar estampillas para identificar de forma única cada una de las consultas en curso.

4.7 Interacción entre los agentes para realizar una consulta distribuida

El ABG recibe la consulta representada por una cadena de caracteres, y extrae de ella la calificación, los nombres de las tablas y los atributos resultado de la consulta. Con estos datos y la información del catálogo global se genera el programa reductor $\mathbf{P}$ y se determina el sitio de ensamble $\mathbf{S_e}$, según el algoritmo SDD-1[Bernstein81]. Posteriormente el ABG le solicita a cada uno de los ABLs involucrados en la consulta, crear la primera versión de las tablas temporales mediante operaciones de selección y proyección. Luego, el ABG por cada semi-join $(R\ SJ_{A=B}\ S)$ de $\mathbf{P}$ solicita al ABL del sitio donde está ubicado R, ejecutar el semi-join. Como respuesta a esta solicitud el ABL crea un subproceso Reductor de Semi-join encargado de realizar el semi-join, que consiste en:

1. solicitar al ABL del sitio de S la proyección de $\mathbf{B}$ sobre la tabla S.
2. almacenar el resultado del paso anterior en una tabla temporal T
3. reducir la tabla temporal de R, mediante el join con la tabla T. Con esto, el subproceso Reductor de Semi-join termina.

Una vez se han procesado los semi-joins de $\mathbf{P}$, el ABG solicita al ABL del sitio de ensamble $\mathbf{S_e}$ generar la tabla resultado. El ABL crea el subproceso Ensamblador quien realiza los siguientes pasos:

1. solicitar a cada uno de los ABLs las tablas temporales (previamente reducidas), para crear copias locales de dichas tablas.
2. generar la tabla resultado, mediante la ejecución de la consulta original sobre la base de datos reducida (conformada por las tablas temporales que han sido centralizadas en el sitio de ensamble).

Por último, el ABG puede solicitar al ABL del sitio $\mathbf{S_e}$ las tuplas de la tabla respuesta, según las solicitudes realizadas por el usuario programador a través del API de MultiBases-Web.

5. Herramientas ofrecidas a los usuarios de MultiBases-Web

MultiBases-Web ofrece un editor de esquemas globales y un API de acceso a bases de datos globales.

5.1 Editor del catálogo global

El editor del catálogo global es una aplicación interactiva que permite a los usuarios (i.e. administradores de la base de datos global) diseñar el esquema global a partir de los esquemas locales. El usuario puede consultar cada uno de los esquemas locales y definir su correspondencia dentro del esquema global mediante funciones de conversión y la definición de equivalencias de las relaciones y atributos del esquema local al esquema global. De esta forma el usuario resuelve los problemas de heterogeneidad semántica.

Este editor permitirá la definición de tablas globales y la definición de los fragmentos que la conforman. Para cada atributo de una tabla global, el usuario debe definir una función de conversión que involucra atributos de tablas de una base de datos local.

5.2 API de acceso a bases de datos globales

Este API le permite al usuario programador de aplicaciones las siguientes facilidades:

- Abrir una base de datos global: Mediante la especificación del nombre y localización de una base de datos MultiBases-Web se crea el acceso a la base de datos global respectiva. Con respecto a la arquitectura del sistema, esta operación equivale a instanciar un objeto de la clase ABG (desde una *Application*) u obtener una referencia a un objeto ABG (desde un *Applet*) (Ver numeral 4.3). Previamente el administrador de la base de datos debe haber definido el esquema global mediante el Editor del esquema global.
- Definir una consulta global: Esta definición se realiza mediante una cadena de caracteres que determinan una sentencia global SQL. Mediante este método se válida la sintaxis de la sentencia de acuerdo a la base de datos previamente abierta y se retorna un identificador de la consulta.
- Ejecutar una consulta global: Ejecuta un consulta dada (que en el caso general involucra múltiples bases de datos locales) para retornar el resultado encapsulado en una lista que contiene cada uno de las tuplas de respuesta. La estructura de los elementos de dicha lista debe corresponder con los campos de la respuesta (esto es responsabilidad del programador).
- Cerrar una base de datos global: Con esta instrucción el programador indica que no se realizarán más operaciones sobre la base de datos global, por lo cual se puede destruir el objeto ABG correspondiente.

Para dar mayor facilidad de uso, el API de acceso se diseña extendiendo el estándar JDBC

6. Conclusiones y Extensiones

Si bien, los algoritmos de optimización de consultas distribuidas han sido ampliamente estudiados, la novedad del proyecto MultiBases-Web radica en la aplicación de dichos algoritmos en Internet que ofrece un contexto más amplio y aplicable por el gran numero de host interconectados y por la necesidad de acceder diferentes ambientes de datos de forma integrada. Los aportes complementarios del actual proyecto están representados principalmente en la adaptación del algoritmo SDD-1 para el manejo de tablas fragmentadas y en la experiencia de utilizar una plataforma de objetos distribuidos en una aplicación compleja que nos permite evaluar dicha tecnología y plantear extensiones y/o modificaciones a las metodologías de análisis y diseño orientadas por objetos tradicionales para el soporte de la distribución de procesos y datos.

Ya que Java maneja la heterogeneidad y la interconexión entre plataformas, el diseño de MultiBases-Web se enfocó en la integración de esquemas y en la optimización de consultas. En este proceso de diseño, la arquitectura de objetos distribuidos de RMI influyó positivamente en el modelaje de los objetos, al simplificar los mecanismos de interacción entre procesos que se deben ejecutar en máquinas remotas.

Para la implantación de MultiBases-Web se cuenta con máquinas con sistemas operativos Linux y Unix y con drivers JDBC para motores de bases de datos Oracle, Sybase y mSQL.

Como trabajos futuros se plantea extender MultiBases-Web hacia un SMultDB completo con manejo de transacciones, soporte a la fragmentación vertical y mixta, soporte a la replicación, y manejo del catálogo global dinámico.

Bibliografía

[Bernstein81] P. Bernstein, N. Goodman, "Query Processing in a Systems for Distributed Databases (SDD-1)". ACM Transactions on Database Systems, Vol 6, No 4 Dec 1981

[Bright94] M. Bright, A. Hurson, S. Pakzad, "Automated Resolution of Semantic Heterogeneity in Multidatabases". ACM Transactions on Database Systems, Vol 19, No 2, Junio 1994.

[Ceri84]] S. Ceri, G. Pelagatti, "Distributed Databases: Principles and Systems", McGraw Hill, New York 84.

[Cucalón92] R. Cucalón, "PERSEO: Ambiente de Programación de Aplicaciones Transaccionales Distribuidas sobre Bases de Datos Relacionale: Módulo optimizador", Universidad de los Andes 1992.

[Franky92] C. Franky, J. Abasolo, R. Cucalón, G. Acosta, S. Maya, "PERSEO: Ambiente de programación de aplicaciones transaccionales distribuidas sobre bases de datos relacionales", XVIII Conferencia Latinoamericana de Informática (PANEL CLEI92), 1992.

[Gomer90] T. Gomer, T. Glenn, C. Chin-Wan, "Heterogeneous Distributed Database Systems for Production Use". ACM Computing Surveys, Vol 22, No 3 Sep 1990

[Gosling94] J. Gosling, H. Mc Gilton, "The Java(tm) Language Environment: A White Paper, http://www.javasoft.com/whitePaper/java-whitepaper-1.html

[Hamilton96] G. Hamilton, R. Cattell, "JDBC™: A Java SQL API Version 1.00", http://www.javasoft.com

[Hirano96] HORB, http://www.sundalla.com

[Jaramillo92] C. Jaramillo, "Sistema de consulta a bases de datos distribuidas heterogéneas", Memos de Investigación, No 78, Universidad de los Andes 1992.

[JavaSoft1] Java™ Distributed Systems, http://chatsubo.javasoft.com/current/index.html

[JavaSoft2] Downloading the Java™ Develoment Kit, http://java.sun.com:80/products/jdk/1.1/

[Larson90] J. Larson, A. Sherh, "Federated Database Systems for Managing Distributes, Heterogeneous, and Autonomous Databases". ACM Computing Surveys, Vol 22, No 3 Sep 1990

[Naughton96] P. Naughton, "The Java Handbook", McGraw-Hill, New York 1996 .

[ODMG] ODMG Home Page, http://www.odmg.org

[Siegel96] J, Siegel, "CORBA Fundamentals and programming", John Wiley, 1996

[Signore95] R. Signore, J. Creamer, M. Stegman, "The ODBC Solution", McGraw-Hill, Estados Unidos de América 1995.

[Yu84] C. Yu, C. Chang, "Distributed Query Processing". ACM Computing Surveys, Vol 16, No 4 Dec 1984