Manejo de información cartográfica mediante base de datos espaciales distribuida

G. Guzmán, M. Torres, R. Quintero, M. Moreno

Centro de Investigación en Computación - IPN, Av. Juan de Dios Bátiz, esquina con Miguel Othón de Mendizábal, México, D.F., 07738. México
{jguzmanl, mtorres, quintero, marcomoreno}@cic.ipn.mx

Resumen. El uso de Sistemas de Información Geográfica (SIG) ha crecido notoriamente debido a su importancia y relevancia en el manejo de información cartográfica. Actualmente hay SIG que permiten visualizar terrenos hasta sistemas más complejos en los cuales es posible identificar zonas de riesgo, realizar simulaciones de desastres naturales, etc. La mayor parte de dicha información, es almacenada en una base de datos para su empleo posterior. Existen aplicaciones comerciales, con las cuales es posible construir programas empleando algún lenguaje orientado a objetos, como lo es Java. Si bien es cierto, mediante Java se puede establecer una conexión con la mayoría de los manejadores de bases de datos actuales, la obtención de información tiene que hacerse de una base de datos en particular. En el presente trabajo, presentamos una arquitectura, que permite generar vistas geográficas integradas de datos descriptivos en los objetos geográficos, los cuales pueden estar almacenados bajo diferentes DBMS, dando la apariencia de una base de datos única, requiriendo como única condición, tener el mismo esquema físico en cada una de las bases de datos.

Palabras clave: Sistemas de Información Geográfica, Bases de datos distribuidas, información cartográfica, datos vectoriales, vistas geográficas.

1 Introducción

Hoy en día, la mayoría de las empresas privadas así como dependencias gubernamentales, se ven en la necesidad de consultar datos espaciales (datos asociados con la representación de un territorio en la Tierra) para realizar un determinado proyecto o una tarea en particular, por ejemplo, una empresa que desea construir una carretera de una ciudad a otra, requiere previamente información con gran exactitud referente al tipo de suelo, poblados aledaños, estructura topográfica, vías de comunicación, etc. Esto con el fin de que el proyecto a realizar no interfiera con alguna vía de comunicación o asentamiento ya existente.

Para el caso de cuando ocurre un siniestro provocado por un fenómeno meteorológico como puede ser un tornado o una inundación, las dependencias de protección civil, necesitan conocer los poblados que están seriamente afectados o pueden estar en peligro, para proceder a realizar una evacuación y con ello evitar la

© A. Gelbukh, H. Calvo (Eds.)
Avances en la ciencia de la computación en México
Research on Computing Science 13, 2005, pp. 203-210

pérdida de vidas humanas. Así como los dos ejemplos anteriores podríamos citar muchos casos más, en los cuales, los datos espaciales son de gran utilidad.

Desafortunadamente en México, muchas organizaciones realizan la obtención de datos espaciales manualmente, empleando mapas cartográficos en formato analógico, lo cual ocasiona entre otras cosas:

a) Demora en los resultados. La mayoría de los mapas contienen demasiada información (curvas de nivel, vías de transporte, hidrología, poblados, etc.) por lo cual, el encontrar los datos requeridos se vuelve una tarea laboriosa y posiblemente la información se requiera rápidamente.
b) Datos no actualizados. El mapa que se esté consultando puede tener 3, 5, 10 o más años de antigüedad, con lo cual, los datos obtenidos pueden ser erróneos o simplemente no encontrarse.
c) Error en los cálculos. Se puede requerir de medir la distancia que hay entre dos objetos de interés, dado que los mapas generalmente están a escala, la medición manual seguramente dará un resultado demasiado impreciso.

Por lo anterior, diversas instituciones se han dado a la tarea de realizar la conversión de dichos datos en formato analógico a un formato digital que pueda ser almacenado en una computadora. Este proceso conocido comúnmente como digitalización permite consultar la información cartográfica desde una computadora, mediante aplicaciones específicas.

La problemática mayor se presenta cuando se tiene geo-información en formato digital, ya que esta información se encuentra dispersa y la información que se llegan a almacenar en una base de datos está en diferentes tipos de manejadores. Por tal motivo, el área de la Geocomputación actualmente se está enfocando en solucionar problemas de heterogeneidad e interoperabilidad en diferentes bases de datos geográficas.

En la presente propuesta se presenta una técnica que permite dar solución parcial al problema, mediante una arquitectura que permite obtener información almacenada en diferentes bases de datos. El resto del documento se encuentra organizado de la siguiente forma: en la sección 2 se describe la arquitectura propuesta para obtener e integrar información que se encuentra almacenada en diferentes bases de datos, la sección 3 describe el diseño de la arquitectura, la sección 4 muestra los resultados obtenidos con nuestra técnica propuesta y finalmente en la sección 5 se citan las conclusiones y trabajo futuro.

2 Arquitectura propuesta

La información almacenada en una base de datos, puede estar físicamente en un solo dispositivo, o bien, puede estar distribuida de acuerdo a alguno de los siguientes criterios [1]:

a) BD distribuida horizontalmente. Son aquellas compuestas de fragmentos horizontales. Formalmente, un fragmento horizontal de una relación, es un subconjunto de las tuplas de esa relación. Las tuplas que pertenecen al fragmento

horizontal se especifican mediante una condición sobre uno o más de los atributos de la relación. Con frecuencia, solo interviene un atributo.

b) BD distribuida verticalmente. Compuesta por fragmentos verticales de información, donde un fragmento vertical de una relación mantiene sólo ciertos atributos de la relación. Los fragmentos son ubicados en diferentes lugares.

c) BD distribuida mixta. Es una base de datos que está distribuida tanto horizontal como verticalmente.

El lenguaje Java permite realizar conexiones a base de datos por medio del JDBC, pero comúnmente la información debe ubicarse en una sola base de datos. Desafortunadamente, no siempre todos los datos vectoriales que requiere una aplicación SIG se encuentran en una sola base de datos, más aún, las bases de datos pueden ser administradas por diferentes manejadores. A continuación, explicamos una arquitectura, conocida como GIS-DBC, la cual permite distribuir información vectorial de forma horizontal, la cual puede almacenarse en diferentes equipos y estar en distintos manejadores. En la Figura 1 puede observarse el esquema general de la arquitectura propuesta.

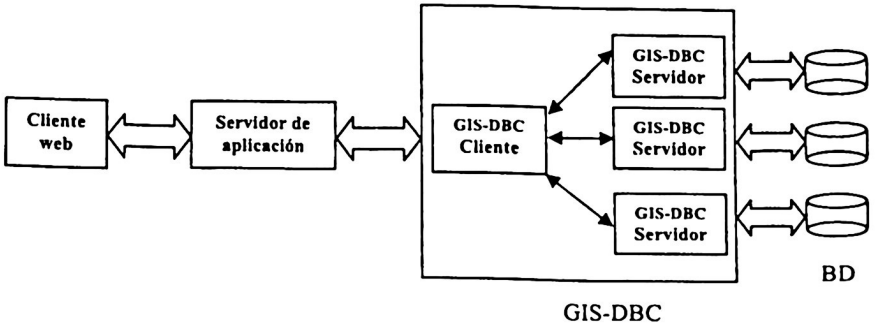

Figura 1. Esquema general de la arquitectura propuesta.

2.1 Cliente web

Utilizando algún navegador de Internet, el usuario accede a la aplicacion para realizar alguna tarea o manipulación de datos vectoriales. Una vez identificada la información que se requiere, el cliente hace una petición al servidor de aplicación.

2.2 Servidor de aplicación

Su función principal es recibir la petición de información por parte del usuario final del sistema. Para proporcionar los datos requeridos, necesita obtener información de la base de datos vectorial, la cual solicita al GIS-DBC.

2.3 GIS-DBC

El GIS-DBC es el encargado de llevar el control de la base de datos distribuida horizontalmente. Al momento en que el servidor de aplicación realiza una consulta SQL, el GIS-DBC hace una búsqueda en cada una de las bases de datos, para poder encontrar la información de interés. Las dos partes más importantes de éste modulo son el GIS-DBC servidor y el GIS-DBC cliente, los cuales, se comunican empleando el mecanismo de RMI.

3 Diseño del sistema

En esta sección se describe el funcionamiento interno del GIS-DBC. Los componentes principales fueron ya previamente mencionados y son los GIS-DBC cliente y servidor.

3.1 El JBDC

Para poder realizar una conexión a una base de datos y ejecutar alguna consulta SQL ya sea para insertar, eliminar, modificar u obtener información almacenada, en un programa en Java, el JDBC define una serie de interfaces. Dichas interfaces, tienen que ser implementadas para cada manejador de base de datos con el que se desee establecer una comunicación. Actualmente, se pueden construir aplicaciones para acceder información a manejadores tales como: Oracle, Informix, SQL Server, MySQL, entre otros [2] [3]. Comúnmente, el proceso involucra la instanciación de objetos de las siguientes clases: *Connection*, *Statement* y *ResultSet*. El proceso de comunicación entre estos objetos, es tal como se muestra en la Figura 2.

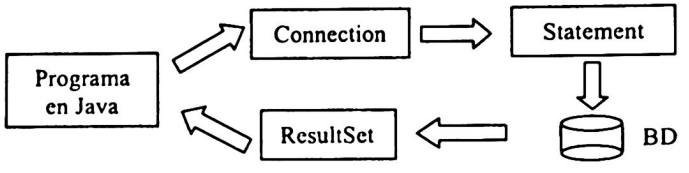

Figura 2. Mecanismo para realizar una consulta SQL mediante Java.

3.2 GIS-DBC cliente

El GIS-DBC cliente es un programa, que se ejecuta como proceso demonio, con la finalidad de revisar periódicamente, si existen peticiones de conexión a una base de datos. Cuando se desea ejecutar una instrucción SQL, por ejemplo de consulta, el GIS-DBC Cliente crea una instancia de la clase *GISDBCConnection*. De manera breve, esta clase implementa la interfaz *Connection* y recibe como argumento una lista que contiene las direcciones IP de las máquinas en las cuales se encuentran las

bases de datos. En el proceso de inicialización del objeto, empleando RMI [4], se trata de establecer una conexión con el programa GIS-DBC Servidor de cada una de las máquinas.

El objeto *GISDBCConnection*, es el que se devuelve al programa que solicita la conexión a la base de datos. Posteriormente, se crean los objetos explicados en el punto anterior (de las clases *Statement* y *ResultSet*) para finalmente ejecutar la instrucción SQL deseada.

Lo importante de señalar es, no existe diferencia significativa, entre un programa que haga uso del GISDBC y de algún otro que realice conexión a una base de datos empleando JDBC, la única modificación radica en solicitar una conexión utilizando la clase *GISDBCDriverManager*, en lugar de la forma tradicional especificada en el lenguaje Java.

Lo anterior es la principal aportación de la arquitectura propuesta, con esto, las aplicaciones SIG previamente escritas, pueden migrarse rápidamente y sin invertir tiempo significativo para poder hacer uso de datos vectoriales que se encuentren en diversas bases de datos, claro está, cumpliendo la restricción de que tengan el mismo esquema físico y de preferencia no tener los mismos datos duplicados en distintas bases de datos.

Con respecto al punto anterior, inicialmente suponemos que los datos no se encuentran duplicados, de hecho, formalmente no sería una base de datos distribuida horizontalmente.

Cuando los datos se encuentran en distintas bases de datos, cada uno de los GIS-DBC Servidores devuelve la información vectorial que encuentra en la base de datos con la que está asociada, para finalmente, el GIS-DBC Cliente construir un objeto *GISDBCResultSet* que contiene todos los datos vectoriales solicitados.

3.3 GIS-DBC servidor

Es el encargado de comunicarse directamente con el manejador de base de datos. Un punto relevante y en el que se debe tener precaución, es que el GIS-DBC Servidor siempre debe estar en ejecución y cargarse antes de que el GIS-DBC Cliente quiera crear una instancia de la clase *GISDBCConnection* por primera vez, de lo contrario, no se podrá conectar a la base de datos en donde no se esté ejecutando en dicho instante, el GISDBC Servidor.

Supongamos que el GIS-DBC Cliente desea crear una instancia de la clase *GISDBCStatement* (la cual implementa la interfaz *Statement*), como no es posible conocer con anticipación de donde se obtendrá la información vectorial, se crea un *GISDBCStatement* que almacena en uno de sus atributos, una lista de las direcciones IP donde se encuentran las bases de datos. Esto con la finalidad de que cuando sea necesario ejecutar una instrucción SQL, dicha instrucción se ejecute en todos los servidores.

4 Pruebas del sistema

Actualmente, se tienen implementados adecuadamente, la mayoría de los métodos de las interfaces *Connection*, *Statement* y *ResultSet*. En la tabla 1 se muestran algunos tiempos de respuesta de dos programas distintos, en el primero de ellos, se empleó conexión directa empleando JDBC a una base de datos única en SQL Server, y en el segundo, usando la arquitectura desarrollada, teniendo dos bases de datos haciendo uso de los manejadores Informix y Oracle.

Tabla 1. Tabla comparativa de tiempos de respuesta.

Instrucción SQL	No. de Registros	Tiempo JDBC	Tiempo GIS-DBC
SELECT	10	2.25 ms.	3.84 ms.
INSERT	5	2.68 ms.	4.11 ms.
DELETE	4	2.30 ms.	3.91 ms.
UPDATE	5	2.54 ms.	5.01 ms.

Como puede apreciarse, el tiempo de ejecución haciendo uso de la arquitectura GIS-DBC es muy cercano a los resultados obtenidos mediante JDBC.

En la Figura 3 se muestra una aplicación escrita en Java, la cual obtiene información de un mapa digital de elevación (DEM por sus siglas en inglés) de diferentes bases de datos.

Figura 3. Visor de modelos digitales de elevación.

Un modelo digital de elevación consiste de un arreglo de muestras de altura medidas a partir del nivel del suelo, las cuales normalmente se encuentran espaciadas en intervalos constantes. Básicamente, un archivo DEM está integrado por tres tipos de registro, usualmente llamados A, B y C. La estructura de estos registros es la siguiente [5] [6]:

Registro A. Contiene información que define las características generales del DEM, incluyendo información de encabezado descriptiva, relacionada con el nombre del archivo DEM, bordes, unidades de medición, valores máximo y mínimo, el número de registros B y parámetros de proyección. Existe un solo registro A en el archivo DEM y aparece siempre como el primer registro.

Registro B. Contiene los datos de elevación y su información de encabezado relacionada. Todos los registros B de los archivos DEM son construidos con datos de bandas unidimensionales, llamados puntos. En consecuencia, el número de puntos totales que abarcan el área descrita por el DEM, es igual al número total de registros B que contiene el archivo.

Registro C. Contiene estadísticas de los datos en el archivo.

Asimismo se está trabajando en el desarrollo de una aplicación para la manipulación de datos espaciales. En esta aplicación se tienen implementados métodos visuales y espaciales como son: acercamiento (*zoom-in*), alejamiento (*zoom-out*), desplazamiento (*panning*), medición de distancia e identificación de atributos. En la Figura 4 se muestra una vista de la aplicación desarrollada.

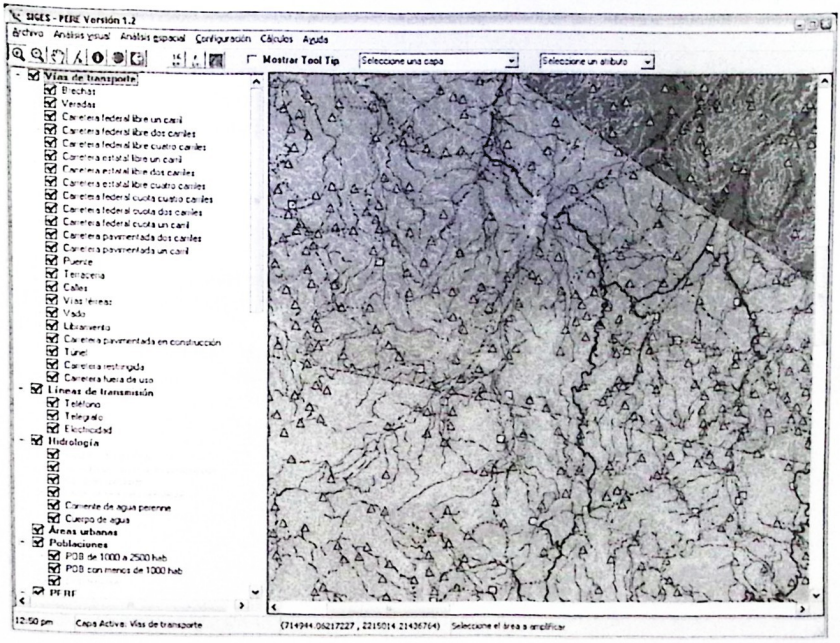

Figura 4. Herramienta SIG para manipulación de datos espaciales.

Esta aplicación fue desarrollada usando el Entorno Integrado de Desarrollo C++ Builder versión 5.0. Aprovechando la arquitectura desarrollada, la aplicación solicita la ejecución de un programa en Java para la obtención de información espacial almacenada en diferentes bases de datos, recopilada dicha información se procede a

guardarla en una base de datos única que es accedida por la aplicación principal para visualizar los datos completos.

5 Trabajo futuro

Existen algunos puntos que todavía requieren de trabajo de investigación para poder encontrar una solución completa. El primer punto, en el que nos encontramos trabajando actualmente, es el problema que surge cuando se necesita combinar información almacenada en diferentes bases de datos que debe combinarse para realizar operaciones más complejas como es la de producto cartesiano.

Otro punto de estudio, es cuando un servidor no se encuentra disponible por situación ajena al GIS-DBC Cliente, como puede ser, problemas en la red, el GIS-DBC Servidor no está en ejecución, entre otros. Como solución inicial, cuando no se puede establecer una conexión entre los GIS-DBC Cliente y Servidor, se lanza una excepción con la cual el usuario recibe un mensaje en el que se le indica que no fue posible conectar con un servidor en particular.

Agradecimientos. Los autores desean expresar su agradecimiento a los revisores del presente documento por sus comentarios pertinentes, así como al CONACYT, CGPI y al CIC – IPN por su apoyo económico para el desarrollo de esta investigación.

Referencias

1. Elmasri R & Navathe S, "Sistemas de bases de datos", Prentice Hall, pp. 704–713, 1999.
2. Deitel & Deitel, "Java How To Program Fifth Edition", Prentice Hall, ISBN: 0131016210, pp. 280-295, 2003.
3. William Grosso, "Java RMI", O'Really, ISBN: 1565924525, pp. 143-178, 2001.
4. Sun Microsystems, "Tutorial de Java", http://www.java.sun.com.
5. U.S. Geological Survey, http://edc.usgs.gov/.
6. http://www.gisdatadepot.com/dem/
7. http://www.mapmart.com/DEM/DEM.htm.