Capítulo 5.
LMBD: Un lenguaje de consulta para un sistema multibase de datos

Este capítulo describe la forma en que fue implementado el prototipo para el lenguaje multibase de datos. Se explica de manera general que se hizo en cada fase del prototipo y los algoritmos que se utilizaron.

LMBD es un lenguaje de consulta para un sistema multibase de datos. La implementación de este lenguaje comprende los siguientes módulos:

Un analizador léxico y sintáctico

Un analizador semántico

Un descomponedor de consultas

Un despachador de consultas

Un constructor de la consulta global

Una interfaz gráfica para la captura y presentación de la información

Una interfaz de programación de la aplicación para LMBD

De esta forma la arquitectura del procesador de consultas para LMBD es como sigue:

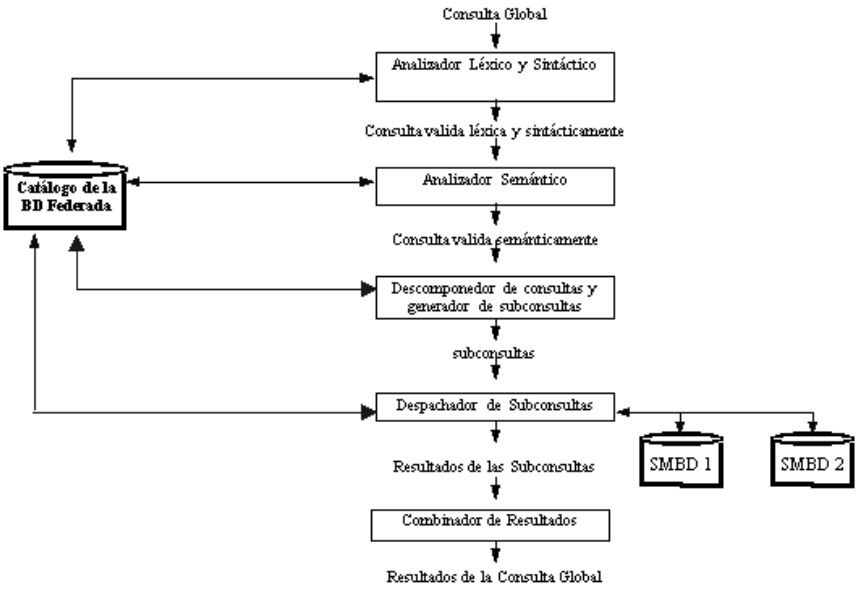

Figura 5.1. Arquitectura del Procesador de Consultas de LMBD

5.1 Desarrollo del Analizador Léxico y Sintáctico

En el apéndice A se presenta la especificación de las reglas sintácticas para el lenguaje utilizando el estilo BNF, así como su implementación en JavaCC.

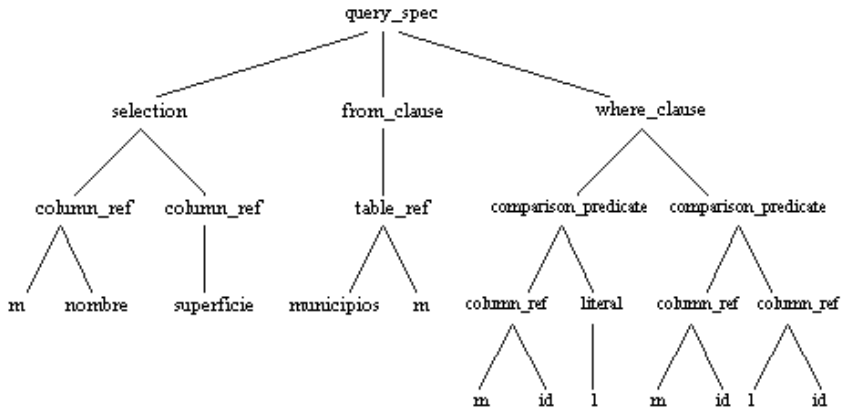

Figura 5.2 Árbol de análisis sintáctico para:
select m.nombre, superficie from municipios m, localidades l where (m.id > 1) and (m.id = l.id);

5.2 Desarrollo del Analizador Semántico

Como podemos observar los prefijos de los atributos fueron sustituidos por el nombre de la tabla a la que pertenecen, así como aquellos atributos que no tenían prefijo les fue agregado también el nombre de la tabla a la que pertenecen, esto para efectos de localización en la fase de descomposición de la consulta.

En el caso en que la selección en el *select* es un * entonces el * es sustituido por los atributos que conforman la tabla (s) en la cláusula *from* .

Al mismo tiempo que se verifican los atributos también se les agrega el tipo de dato que tiene cada uno, para que posteriormente pueda ser verificada la compatibilidad de tipos de datos en las condiciones que conforman la cláusula *where*.

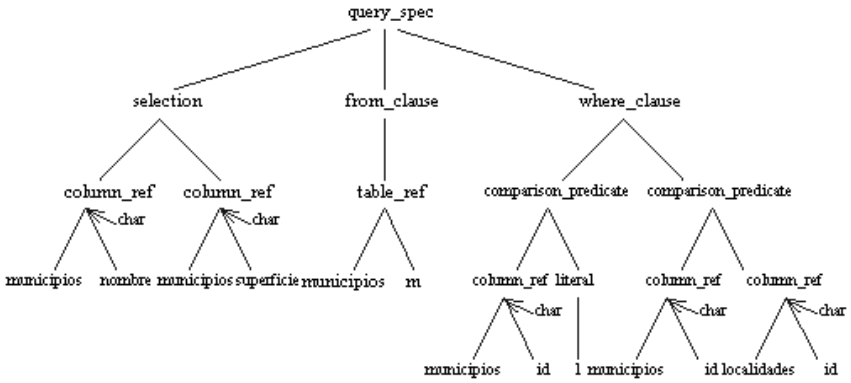

Figura 5.3 Árbol de análisis sintáctico modificado para:
select m.nombre, superficie from municipios m, localidades l where (m.id > 1) and (m.id = l.id);

5.3 Desarrollo del Descomponedor de Consultas

Este proceso se lleva a cabo para poder obtener la información necesaria para construir las subconsultas que deben enviarse a cada sitio.

El algoritmo que se propuso para la descomposición y construcción de subconsultas es como sigue:

1. Por cada atributo que aparece en la cláusula select hacer:

   Ir a la tabla de equivalencias del esquema auxiliar y recuperar la tabla local, atributo local, el sitio donde se ubican y los conflictos que presentan.

   Si el atributo presenta el conflicto de atributo faltante (FA) entonces no agregar el atributo al select.

   Si el atributo presenta el conflicto de composición (CMP) entonces localizar todos los atributos que lo conforman y agregarlos al select.

   Si el atributo presenta el conflicto de horizontalidad (H) entonces agregar el atributo a ambos selects.

   Si el atributo no presenta el conflicto de horizontalidad entonces

   Por cada atributo que se agrega al select también se agrega el atributo designado para ser el atributo con el que se va a hacer la operación *reunión* [Elmasri y Navathe 1997]. de las tablas, es decir el atributo común a ambas tablas.

   NOTA. El atributo se agrega al select que le corresponde de acuerdo al sitio y tabla local.

2. Por cada atributo que aparece como parte de una condición en el where hacer:

   Si el atributo aún no forma parte de la lista de atributos del select agregar el atributo, para posteriormente poder verificar la condición.

   Las condiciones son guardadas en un vector para después utilizarlas en la reconstrucción de la consulta global, estas condiciones sirven para seleccionar las tuplas.

Este algoritmo se lleva a cabo tomando la información del esquema auxiliar de la multibase de datos y la información almacenada en el AAS. Como podemos ver en la figura 5.3 cada atributo debajo del nodo *selection* tiene como nodo hermano el nombre de la tabla a la que pertenece, por lo que en la etapa de descomposición de la consulta ya se tiene la localización del atributo, es decir la tabla a la que pertenece.

En el apéndice C se muestran ejemplos de como se realiza la descomposición de la consulta global y la construcción de las subconsultas.

5.4 Desarrollo del Despachador de Consultas

Este módulo se encarga de enviar las subconsultas al sitio que les corresponde y recuperar la información. Para hacer esto tiene que conocer la ubicación del sitio, la cual se encuentra almacenada en la tabla *localización* (ver apéndice B) del esquema auxiliar de la multibase de datos.

Como se comento en el capítulo dos, la interconexión con las fuentes de datos es a través de JDBC. Así la información de localización del sitio esta en forma de un URL (localizador uniforme de recursos) que es lo que se encuentra almacenado en la tabla *localización* del esquema auxiliar de la multibase de datos.

La clase *BaseDatos* que se implemento, permite la interconexión con las fuentes de datos de manera sencilla. Por ejemplo:

```
BaseDatos.OpenDB("informix", "municipios");
```

realiza la conexión con la base de datos municipios que se encuentra en el servidor informix.

En el presente prototipo las bases de datos que se pretenden integrar

están almacenadas con los manejadores de bases de datos Informix Universal y Oracle RDB.

Obtener el controlador JDBC para Informix fue relativamente fácil, no así con el controlador JDBC para Oracle RDB. Oracle RDB funciona bajo el sistema operativo VMS lo cual implica otras operaciones propias de VMS para cargar el controlador. También cabe mencionar que la versión de JDBC que se utilizo es una versión comercial, por lo que si no se compra, solo se puede descargar para que funcione durante 4 horas, después de este tiempo el controlador se inhabilita, así que si se desea seguir utilizándolo hay que matar el proceso y volver a cargar el controlador.

Al llegar a esta etapa ya se tienen bien construidas todas las subconsultas que conforman la consulta global, por lo que solo hay que mandar a ejecutarlas y recuperar los resultados, para esto se utilizan arreglos de la clase *ResultSet* (clase del paquete sql de java) en los que se almacenan los datos resultantes de las subconsultas.

5.5 Desarrollo del Combinador de Resultados

En esta etapa se combinan los resultados de las distintas subconsultas para darle forma a los resultados de la consulta global. El algoritmo que se propone e implementa para esta tarea es el siguiente:

Por cada select, empezando por el mas interno hacer:

1. Recuperar la información del tipo de integración que se utilizo en cada tabla (esta información se encuentra en la tabla *tipoIntegración* (ver apéndice B) del esquema auxiliar de la multibase de datos).

2. Si la integración fue de tipo *vertical* ejecutar la operación *reunión* con los *ResultSet* (obtenidos en la etapa anterior) de las subconsultas que se refieren a la misma tabla. La operación *reunión* se realiza a traves de los atributos locales (presentes en los *ResultSet*) que fueron designados para ser los atributos comunes en ambas tablas locales y que se encuentran indicados en la tabla *atributosJoin* (ver apéndice B) del esquema auxiliar de la multibase de datos.

3. Si la integración fue de tipo *horizontal* ejecutar la operación *unión* con los resultados de las subconsultas que se refieren a la misma tabla.

4. Mientras se lleva a cabo el paso 2 y 3 se debe también de ir verificando si los atributos involucrados en los resultados presentan algún conflicto y resolverlo.

   Si el atributo en cuestión presenta el conflicto de *atributo faltante* (FA) entonces en el lugar que le corresponde a este atributo agregar un *null*.

   Si el atributo presenta el conflicto de *composición* (CMP) entonces identificar que atributos locales conforman a este atributo y hacer la concatenación de los datos de estos. La información concatenada es agregada en el lugar que le corresponde a este atributo

   Si el atributo presenta el conflicto de *unidades distintas* (UD) entonces recuperar la expresión aritmética que se debe aplicar a este atributo (almacenada en la tabla *auxiliar* del esquema auxiliar de la multibase de datos) y enviarla al evaluador de expresiones incorporado en LMBD para hacer el calculo. El resultado es agregado en el lugar que le corresponde a este atributo.

   NOTA 1. La indicación del tipo de conflicto que presentan los atributos se especifica en el atributo *conflictos* de la tabla *equivalencias* del esquema auxiliar de la base de datos. (ver apéndice B).

   NOTA 2. Al final de este paso toda la información de las tablas ya se encuentra almacenada en una estructura que involucra solo tablas y atributos globales. (ver apéndice E)

5. Realizar el producto cartesiano de las tablas que aparecen en la cláusula from de la consulta global.

   Al mismo tiempo que se lleva a cabo el producto cartesiano, por cada combinación de tuplas completa, seleccionar las tuplas que reúnan las condiciones especificadas en la cláusula *where* y agregarlas a la tabla de salida, de manera que al final del producto cartesiano solo estén almacenadas las tuplas que cumplieron con las condiciones. Las condiciones incluyen también el operador de comparación *in* y la función *exists,* es decir, las que involucran subconsultas (cuando se evalúa *in o exists* ya se tienen los resultados de la subconsulta debido a que se empieza por resolver los selects mas internos).

El algoritmo utilizado para realizar la operación reunión es el de ordenamiento-mezcla [Mishra y Eich 1992] que se muestra en el apéndice D. La clase que modela la estructura de datos utilizada para almacenar los resultados de la operación *reunión,* l *as* proyecciones del select, así como el resultado final se muestra en el apéndice E.

5.6 Desarrollo de la Interfaz de Usuario para la Captura y Presentación de Información

La figura 5.7 tiene en la parte superior un encabezado con los nombres de las columnas de los datos que son recuperados por la consulta, en la parte media esta el área en la que se despliegan los datos, y en la parte de abajo una barra de estado que indica cuantos renglones fueron recobrados y un botón **Close** que se encarga de cerrar la ventana.

El botón **Clean** del área de botones (figura 5.4) limpia el área de captura de la consulta, así como el área de despliegue de mensajes de error.

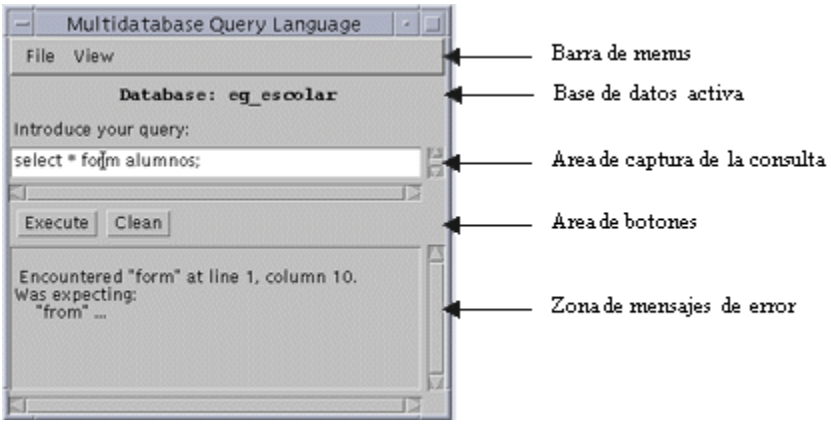

Figura 5.4 Interfaz principal de LMBD

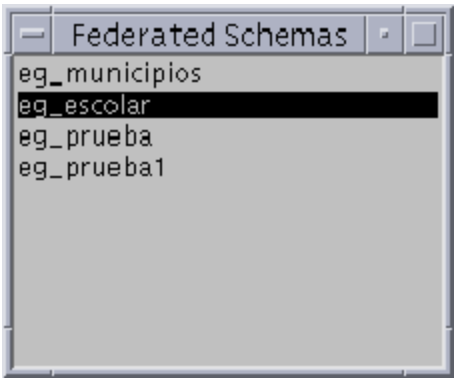

Figura 5.5 Lista de las bases de datos disponibles para consultar

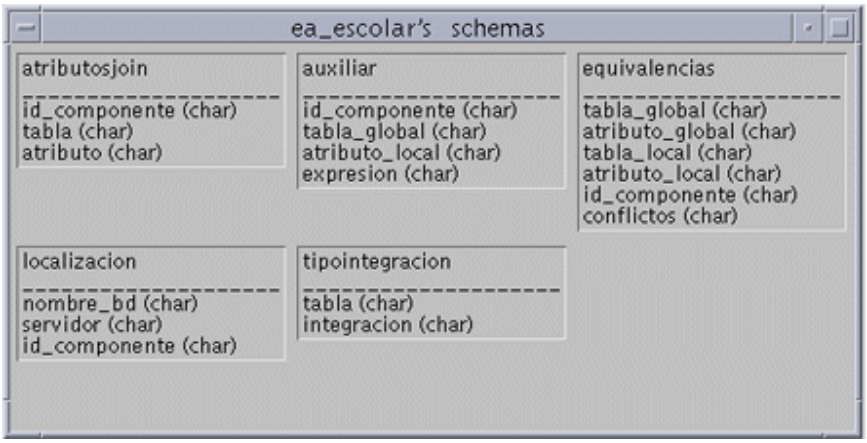

Figura 5.6 Esquema de la base de datos activa

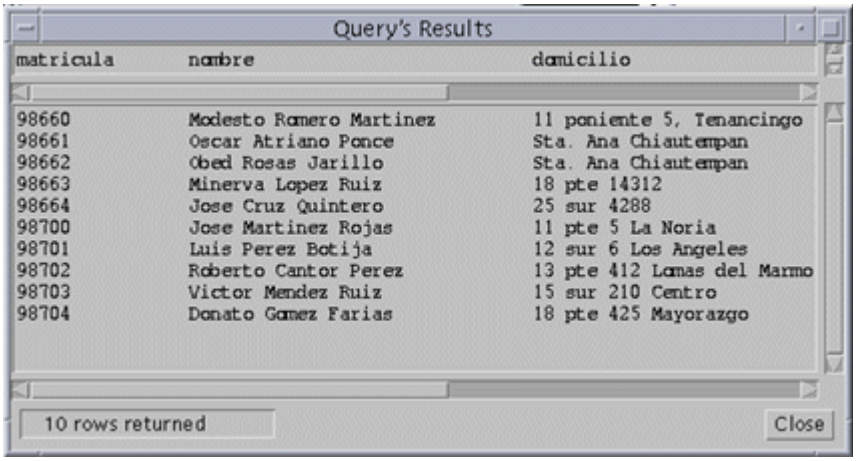

Figura 5.7 Ventana que muestra los resultados de una consulta

5.7 La Interface de Programación de la Aplicación de LMBD

LMBD puede ser manejado desde una aplicación sin necesidad de utilizar la interfaz gráfica, es decir, LMBD puede ser llamado desde un programa Java a través de la clase *MultiBD* . Así, se puede llevar a cabo la manipulación de una o mas multibases de datos dentro de un mismo programa Java. Los resultados de una consulta a una multibase de datos son retornados como un objeto de la clase *Result* (ver apéndice E) y por lo tanto pueden ser manipulados por el programador como desee.

La forma en que se utiliza una multibase de datos dentro de una aplicación es sencilla, solo son necesarios los siguientes pasos:

1. Crear un objeto de la clase MultiDB

```
MultiDB m = new MultiDB();
```

2. Abrir la multibase de datos indicando el nombre del esquema global y el nombre del esquema auxiliar.

```
m.open("eg_municipios", "ea_municipios");
```

3. Someter una consulta

```
Result r = m.execute("select * from municipios;")
```

Para probar la funcionalidad de la clase MultiDB manejada desde un programa Java se desarrollo una aplicación en la que se integra información geográfica e información descriptiva. La información geográfica que se integra esta distribuida en los manejadores de bases de datos Illustra e Informix y se encuentra representada como *quadtrees* (López 1998), y la información descriptiva que se integra esta almacenada con los manejadores de base de datos Informix y Oracle RDB. La integración de la información geográfica se encuentra en una multibase de datos llamada *eg_espacial* y la información descriptiva en una multibase de datos llamada *eg_municipios,* ambas multibases de datos son manipuladas desde un programa Java llamado *Consultas* .

La aplicación *Consultas* dibuja el mapa del estado de Colima recuperando esta información a través de la multibase de datos *eg_espacial* y después permite hacer consultas haciendo click sobre alguno de sus municipios. Después de detectar el municipio seleccionado se genera una consulta que se encarga de recuperar la información descriptiva correspondiente a ese municipio a través de la multibase de datos *eg_municipios* . Finalmente la información es presentada. Esta aplicación es una muestra de como se puede manipular mas de una multibase de datos desde un mismo programa Java.

Como se pudo observar la interfaz gráfica ofrece un ambiente adecuado para la captura y presentación de la información. Además LMBD ofrece servicios como la presentación del esquema de la base de datos activa, ya sea local o una multibase de datos y la capacidad de poder procesar consultas locales y globales a través de la misma interfaz.

La flexibilidad de LMDB de poderse manipular desde un programa Java le da al programador una herramienta poderosa para consultar una multibase de datos cuyos resultados puede manipular a placer, aun mas si consideramos que se puede manipular mas de una multibase de datos desde un mismo programa Java.

En el apéndice C se muestran ejemplos que describen de manera sencilla como se realiza el procesamiento de una consulta, lo cual puede ayudar a comprender mejor la implementación de este lenguaje.

Romero Martínez, M. 1999. **Lenguaje de Consultas para una Multibase de Datos**. Tesis Maestría. Ciencias con Especialidad en Ingeniería en Sistemas Computacionales. Departamento de Ingeniería en Sistemas Computacionales, Escuela de Ingeniería, Universidad de las Américas Puebla. Mayo.