Capítulo 6.
Conclusiones

En este capítulo se mencionan las apreciaciones mas importantes de esta tesis de manera general, así como las referentes únicamente al prototipo. También se menciona el trabajo a futuro que puede desarrollarse siguiendo la misma línea.

6.1 Conclusiones Generales

La importancia principal de las multibase de datos y mas concretamente de las bases de datos federadas fuertemente acopladas radica principalmente en su bi-procesamiento. Es decir, en su capacidad de atender consultas globales, al mismo tiempo que permite que las bases de datos componentes sigan atendiendo a sus aplicaciones locales.

La existencia de un esquema global permite que el lenguaje implementado para llevar a cabo las consultas sea fácil de aprender y entender (muy parecido a SQL) debido a que este da a la multibase de datos la apariencia de que se accesa a una base de datos sencilla y por lo tanto las operaciones de distribución son transparentes al usuario.

El problema de la heterogeneidad de las bases de datos componentes puede hacer que algunas tareas lleguen a ser complejas. La tarea de optimización de consultas requiere de información como la velocidad de procesamiento del CPU, la velocidad de entrada/salida entre otros, para cada una de las bases de datos; así como el tamaño de los resultados intermedios de las subconsultas, sin embargo debido a la heterogeneidad de las bases de datos componentes, esta información es difícil de mantener principalmente por las diferentes capacidades de procesamiento de las bases de datos componentes lo cual hace que la optimización de consultas sea una tarea difícil de realizar.

El surgimiento de estándares y nuevas tecnologías permite crear soluciones multibase de datos. Estándares para el desarrollo de manejadores de bases de datos (como el estándar para SQL), así como el de tecnologías para la conexión a múltiples bases de datos (por ejemplo JDBC), y lenguajes capaces de crear aplicaciones que se ejecuten en cualquier plataforma (como java) permiten crear aplicaciones multibase de datos, que aunque consumen un tiempo considerable ofrecen una solución.

6.2 Conclusiones del Prototipo

El generador de analizadores léxico-sintácticos JavaCC es una herramienta fácil de utilizar ya que permite la especificación de los componentes léxicos y la especificación de las reglas sintácticas en el mismo programa. Además ofrece muchos métodos para la creación y manipulación del árbol de análisis sintáctico, el cual es de suma importancia en el desarrollo de lenguajes.

JDBC ofrece una interfaz estándar para accesar múltiples bases de datos. Para esto hace uso de SQL lo cual lo hace fácil de utilizar, sin embargo, debido a que los controladores JDBC no implementan todas las funciones o la mayoría de las que se utilizan en un manejador de bases de datos como es la recuperación de metadatos, hace que disminuya su capacidad, limite su uso y su enfoque multibase de datos. Este es el caso del controlador JDBC para Oracle RDB el cual al no implementar todas las funciones de recuperación de metadatos, no hace posible su operación local desde la interfaz del prototipo.

Como se pudo observar, la integración de las tablas en forma vertical u horizontal y la resolución de conflictos son las actividades que asocian de manera consistente la información de las bases de datos. Las otras operaciones como el producto cartesiano o la selección de tuplas son operaciones que complementan la funcionalidad de este prototipo.

El lenguaje de consultas que se implemento tiene suficiente poder para recuperar casi cualquier consulta, además el poder incluir subconsultas y aceptar cualquier forma de especificación de las condiciones en la cláusula *where* lo hacen un lenguaje aceptable. Aun mas si consideramos que se implementa un módulo para el manejo de errores y una interfaz gráfica que permite la captura y despliegue de resultados de manera ordenada.

Un aspecto importante de este prototipo es que permite la operación global y local desde la misma Interfaz. Las servicios que se ofrecen para una multibase de datos (despliegue de mensajes de error, despliegue de resultados y despliegue del esquema de la base de datos) también se ofrecen para una base de datos local.

El utilizar SQL como lenguaje de consulta para la multibase de datos hace que todo parezca transparente al usuario, que da la apariencia que se accede a una base de datos local.

6.3 Trabajo Futuro

Existen funcionalidades que complementarían la funcionalidad de este prototipo y que lo harían mas poderoso, entre estas tenemos:

Extender el número de bases de datos que conforman la multibase de datos, es decir, que no sean solamente dos las bases de datos que se puedan integrar y consultar. Para esto se tiene que incorporar la información de todas las bases de datos en el catálogo de la multibase de datos y modificar el algoritmo de descomposición de la consulta global, así como el de reconstrucción de la consulta a partir de los resultados de las subconsultas.

Incorporar estrategias para la optimización de las subconsultas. Para hacer esto se requiere que el catálogo de la multibase de datos maneje la información requerida para esta tarea, lo cual se menciono en el capítulo 3.

Anexar funciones al lenguaje como SUM(), COUNT(), AVG() entre otras, así como otras cláusulas como GROUP BY y HAVING.

Incorporar la solución para conflictos que se presentan cuando la información es compleja. Por ejemplo cuando se trata con información espacial que esta representada como tuplas como son los *quadtrees* [López 1998] . Aquí se podría dar solución a conflictos de escala, de nivel de precisión, de visualización entre otros.